Comparative analysis of machine learning techniques in optimal site selection.

by

Aukik Aurnab
19101485
Shaktiman Choudhury
19101506
Shoubhick Roy Ruhan
19101124
Shikh Muhammad Rifaiya Abrar
19101508
S.M. Riyadh Hossain Rabbi
19101511

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering.

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2023

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing a degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material that has been accepted or submitted for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Students Full Name & Signature:

[illegible]

Aukik Aurnab
19101485

[illegible]

Shaktiman Choudhury
19101506

Shoubhick Roy

Shoubhick Roy Ruhan
19101124

[illegible]

Shikh Muhammad Rifaiya Abrar
19101508

[illegible]

S.M. Riyadh Hossain Rabbi
19101511

Approval

The thesis titled Comparative analysis of machine learning techniques in optimal site selection. submitted by

1. Aukik Aurnab (19101485)
2. Shaktiman Choudhury(19101506)
3. Shouvick Roy Ruhan(19101124)
4. Shiekh Mohammad Rifaiya Abrar(19101508)
5. SM Riyadh Hossain Rabby(19101511)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 19, 2023.

Examining Committee:

Supervisor:
(Member)

[illegible]

Rubayat Ahmed Khan
Senior Lecturer
Department of Computer Science and Engineering
BRAC university

Program Coordinator:
(Member)

Md. Golam Rabiul Alam
Program Coordinator
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi
Head of Department
Department of Computer Science and Engineering
BRAC University

Abstract

Site selection is a crucial aspect of many businesses, as a company's location can significantly impact its success. In recent years, machine learning techniques have been increasingly used to assist with optimal site selection by providing data-driven predictions about the potential success of a given location. Machine learning techniques can be used to assist in the process of selecting the optimal site by analyzing the patterns in data such as demographics, lifestyle services, and geographic features. In this paper, we compare several machine learning techniques for their performance in optimal site selection for features extracted from Open Street Map (OSM) data, WorldPop population data, and Bing satellite imagery. A target dataset corresponding to the features extracted was collected from Yelp data on restaurant check-ins, and this was used as a parameter to determine the human engagement rate of that location with the businesses in that area. Our analysis methods included SVR, Random Forest, XGBoost, Ridge Regression, Lasso Regression, and ElasticNet. The satellite imagery collected from Bing maps were used to train CNN architectures such as; VGG16, VGG19, ResNet, DenseNet, and InceptionV3 and the results were compared. We evaluated the techniques using several metrics, including Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error(MedAE), Max Error(ME), and Median Absolute Deviation(MAD). We used algorithm and strategies that performed the best in related works for this research. One meta model was also implemented in this work by an ensemble learning technique known as stacking. The model that performed the best for the data collected was then determined by looking at the error scores of different models. This work provides an insight into the strengths and limitations of each technique and recommendations for practitioners considering the use of machine learning in site selection. This study demonstrates the potential of machine learning for improving site selection processes and highlights the importance of considering multiple approaches.

Keywords: Optimal Site Selection (OSS), Comparative Analysis, Boosting and Stacking, SVR, Random Forest, XGBoost, Ridge Regression, Lasso Regression, Elasticnet, Convolutional Neural Network (CNN), Ensemble Learning, VGG16, VGG19, ResNet, DenseNet, InceptionV3, Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (MedAE), Max error (ME), Median Absolute Deviation(MAD).

Acknowledgement

We would like to take this opportunity to acknowledge the influence of Hossain Arif, who unfortunately passed away before the completion of this thesis. Hossain Arif sir was a mentor and his guidance and support was invaluable to us during our research. We are also deeply grateful to Rubayat Ahmed Khan, for promptly taking the responsibility of guiding us as our advisor. His invaluable contributions as an advisor and guidance throughout our research has been encouraging

Table of Contents

List of Figures

List of Tables

Chapter 1

Overview

1.1 Introduction

Choosing a location to establish or expand a business is a crucial decision for proprietors, entrepreneurs, and organizations. The location of a business can greatly impact its profitability and stability, making the selection of the optimal location a vital step toward success. However, traditional survey methods for site selection can be costly, time-consuming, and prone to error. Urban and rural city planning play a significant role in the availability of resources, infrastructure, and services necessary for business operations. The factors of population density, transportation, and land use also affect the success of a business. Urban and rural city planning can be used to create environments that foster business growth and success, while a poorly planned site can stifle development and discourage other businesses from locating there. Site selection not only impacts the business itself but also contributes to the overall economic development of a region. A well-planned business site can stimulate growth and attract additional businesses, ultimately benefiting the community.In addition, the environmental impacts of a business must also be considered during site selection. Urban and rural city planning plays a vital role in mitigating negative impacts and preserving the natural environment for sustainability. Given the complexity of the issue, many business owners struggle with allocating funds or conducting research when expanding into rural areas. Our study aims to simplify the process for business owners to find the ideal location for their company, allowing them to expand while contributing to the economic expansion of the region.To make things easy, the use of machine learning algorithms in site selection can lead to a more efficient process by analyzing a large volume of data more quickly and accurately than human analysts. However, it is essential to ensure that the data used to train these algorithms is accurate and representative of the real-world situation. There are numerous studies and publications available that use various algorithms and methods for pre-processing and data analysis to determine an optimal location for a business. However, with so many algorithm alternatives to select from, it can be difficult to determine which machine learning approach is appropriate for a given organization. Our main goal with this study is to provide the best approach which includes proper algorithms and appropriate features to include for these businesses to employ. It would provide an optimal solution to locate a suitable site to establish a business by doing a comparison analysis on multiple machine learning methodologies, including Neural Networks.In summary, site selection and urban and rural planning play a vital role in ensuring the success of businesses and the overall economic development of a region. By considering a wide range of factors, implementing efficient data analysis techniques, and promoting economic development, we can foster an environment that benefits businesses and communities alike. Therefore by implementing a proper machine learning technique on the data available at hand, an efficient and optimal location will benefit the startups in the long run.

1.2 Problem Statement

In view of the current pattern of economic decline, it is crucial for a company just starting out to select areas with the greatest long-term potential for success. To assist such individuals and organizations in preventing their businesses from failing due to a lack of suitable tools to evaluate the optimal location, it is essential to utilize the most effective machine-learning technique and provide them with answers based on data analysis. As the data used in this example may be of varying types and may define various aspects of a location, it can be quite difficult to identify which Machine Learning Algorithm would be most effective in a given situation. As the location score projected by the algorithm will enable the business to operate more efficiently, employment and revenue will increase. Numerous studies have examined and predicted the optimal locations for businesses such as restaurants, hospitals, banks, and others. However, the data used in these instances was extremely limited to statistics pertinent to the companies. Based on our findings, there has been limited research on the comparative analysis of several machine learning techniques for their performance in optimal site selection, using features extracted from Yelp dataset, Open Street Map (OSM) data, WorldPop population data and satellite imagery, and evaluating the techniques using various metrics. In our research, varied types of data, such as satellite imagery, will also be used to train and evaluate Deep Neural Networks that combine transfer learning and customized CNN layers to produce the best possible results for our data. Through a comparison analysis of supervised ML algorithms and neural networks, our ultimate objective is to provide a response to the question of which algorithm and approaches will be more effective for an optimal site selection procedure.We expect that good quality research will generate the most suitable site of the business and also affect the revenue of businesses. For a multitude of reasons, it is of utmost importance in urban planning, the evolution of economies, and individual enterprise. As an example, Bangladesh is currently at the forefront of one of the world's economies with the quickest rate of growth; For smaller enterprises to prosper in our country, it is essential that they select the proper foundations. In addition, the selection of an ideal site for the development of infrastructure can help to ensure that new developments are located in areas of the city that are not only easily accessible, but also well-connected to the rest of the city and compatible with the community that is immediately adjacent to them. An ideal location is one in which the negative effects of the development, such as pollution and traffic congestion, may be reduced to a minimum. This is due to the fact that an optimal site is located in an area where these impacts can be mitigated. The selection of an optimal location for development can aid in attracting enterprises and investment to a region, stimulating economic activity, and leading to the creation of new employment possibilities. This is essential for economic expansion. Additionally, a great location may contribute to an increase in the value of real estate in the surrounding area, which may result in economic growth as a result of rising property values. Choosing a location that is conducive to success is often crucial to the overall success of a personal enterprise. When selecting a location for a personal business, it is important to examine the proximity of customers, the cost of rent or purchase, the availability of parking, and the accessibility of public transportation. A great location is one of the most significant determinants of a company's prosperity and growth in the community in which it is located. Therefore, the purpose of this study is to determine the most accurate and dependable method for forecasting the potential success of a particular location and to provide guidance for practitioners considering the use of machine learning in site selection. This study aims to determine the most accurate and trustworthy approach for estimating the future success of a given place.

1.3 Research Objective

Using the most relevant and geographical data, our primary objective is to compare different machine learning approaches to determine the optimal site for any kind of business, service, or public building. Our research objective is encapsulated by;

- Investigating the effect of different feature sets and feature engineering techniques on the performance of the models.
- Conducting an in-depth analysis of the limitations and challenges of using machine learning for site selection
- Conducting a case study or real-world application of the model, to demonstrate its practical utility and potential impact.
- Exploring the use of CNNs for the satellite imagery on detecting optimal location and comparing it's results with other machine learning models.
- Using the stacking method of ensemble learning, a Meta Model will be constructed to identify the best possible site.
- Considering organization- or business-specific priorities and collecting data regarding that.
- Training and evaluating our model utilizing multiple machine learning algorithms including neural networks.
- Using several evaluation metrics, such as RMSE, MSE, MAE, Max Error, MedAE, and MAD, to determine the performance of the models.
- Conducting a thorough comparison and documentation of the performance of each algorithm.

Chapter 2

Related Work

2.1 Literature review

In order to identify the most effective method for optimal site selection using machine learning for a specific service or business, it is important to review the existing literature in this field. By analyzing the research that has been conducted in the past, we can gain insight into the methods that have been found to be successful, as well as identify any gaps or limitations. In this literature review, we will examine the various techniques and approaches that have been used in previous studies, and evaluate their effectiveness in terms of the specific objectives and criteria of our own research project. By doing so, we aim to gain a comprehensive understanding of the current state of the field, and inform the development of our own methodology for optimal site selection.
Reliable data describing economic activity at a granularity that is informative to governments or organizations require logistically difficult surveys. That is, instead of doing surveys, which take a lot of time, the goal is to predict economic activity using data from satellites. In this work [26] by Neal Jean et al., a method for predicting poverty that is accurate and scalable was proposed to solve this problem. The model worked with three types of data. The economic level is shown by the amount of light at night[19]. The socio-economic level is shown by the look of the landscape during the day. Finally, survey data on household spending and wealth can be used as a reference to check the results found after analyzing the satellite images. The proposed transfer learning model has 3 phases. In phase 1, the CNN was trained using daylight satellite imagery. They used things like roads, cities, and other places that show economic activity or lack thereof as CNN features. Then, in phase 2, they use the results from phase 1 and train the CNN to adapt to estimate the nightlight intensities. In the last step, phase 3, the data from the economic survey is combined with image features that CNN has taken from daytime images. train regression models that can estimate the poverty indicators being looked at. This paper presents a way to keep track of the economy and poverty in developing countries that is accurate and cheap. The research could then map out projections on places where there were gaps in the survey data, helping to find previously unknown areas for aid targeting. Now, the model knows that there is a link between the things in the satellite photos and the level of poverty.
In this study [30], Daniel Arribas-Bel and his team looked at the potential of two

machine learning algorithms, RF and GBR, using the LED index in Liverpool as an example. It figures out the LED index from a very high-resolution (VHR) spatial aerial image's spectral, texture, land cover[13], and structural characteristics. This article compares the results to the results of a spatial lag model and an OLS regression, which are two standard econometric models. Since the author thinks that these two methods are the most popular because they work well and are easy to use, the paper focuses on them instead of other options. Here, Radom Forest(RF) [3]proved to be the optimal strategy, followed by GBR and then the two traditional approaches, the spatial lag model and OLS regression. The paper found that the two machine learning algorithms that were looked at not only did a better job of predicting the future but also offered a promising way to include a larger number of the possible explanatory variables (features) from satellite photos. The research also comes to the conclusion that if these kinds of methods are used more often, deprivation measures will be predicted in a clearer and more accurate way.
In this investigation[40], Dorji and his colleagues are trying to figure out the median income of Thailand's subdistricts by using K-Means clustering, a machine learning technique. They utilized nighttime light data[8], a highly effective approach for distinguishing high-income and low-income zones with accurate decision thresholds. In addition, they utilized road coverage data from OpenStreetMap, population data from WorldPop, and the GADM database[34], which is essential for generating a configuration file containing the territorial borders of each subdistrict. In addition, they implemented BMN data as a source of income data. Then, they determine the characteristics of the data, including the median, standard deviation (SD), and addition of the nighttime light data, the population and road density of the area, the distance between Bangkok and Chaing Mai, and the subdistrict family income. According to their analysis, the subdistrict median annual household income was omitted from the input matrix as X since a gradient boosting regressor (GBR) was fitted along with it as dependent variable Y. The median household income feature for a particular subdistrict was then divided into three levels using the K-means clustering method. However, due to imbalanced classification issues, a gradient-boosting classifier was employed to assemble the feature matrix, which is X, with the objective being the median annual household income. Level 0 had the highest F1 score following their final training, followed by levels 2 and 1. It demonstrates that the algorithm is effective at finding Thai subdistricts that are good candidates for impoverished areas. As a result of being capable of detecting high-income areas with high precision, the government can take the required actions to develop the low-income regions and remove any locations that are ineligible for classification as poor. In the event of flaws, the study failed to consider the breadth of the road, which could have improved the estimate, because Thailand's continuous development of the bulk of its rural areas meant that road density was not as significant as the other qualities. Some subdistricts in the southern region with a level 1 income are classified as level 0 by the model because of their inadequate evening lighting and distance from the Bangkok metropolitan centre. In addition, while southern regions are popular tourist destinations, their research was unable to classify the tourist regions responsible for higher income levels. The author also asserted that they can overcome the aforementioned shortcomings by incorporating several additional characteristics for classifying tourist destinations as having a greater level of revenue. Last but not least, they indicated that they were searching for strategies

to further discriminate between places categorized as low-income regions.
Cloud imagery, both daytime and nighttime, collected from various satellites over the course of a decade was used in Christopher Yeh and his team's additional work[47], which sought to find a local level measurement of who and where the poor people of Africa are based on the publicly available cloud imagery. According to the findings of the research, the desired result can be accomplished by utilizing CNN. In order to create the models, the researchers compiled a wealth index based on asset wealth across 23 nations in Africa. The data for this index came from nationally representative Demographic and Health Surveys (DHS) carried out between 2009 and 2016. In addition to this, photos of the surface reflectance and nighttime lights (also known as nightlights)[26] captured by Landsat and centered on the locations of each cluster were collected by using the Landsat archive that is accessible through Google Earth Engine. After that, the CNN model was trained with these datasets by utilizing the ResNet-18 architecture [25], which was selected because of its balance of compactness and excellent accuracy on the ImageNet image classification task [20]. After that, this effort continued on and attempted to address the three-class classification issue by employing the transfer learning strategy that had been initially presented by Neal Jean[26]. Due to the fact that the primary objective was to generate and evaluate the out-of-country forecasts, the results were not comparable with small-scale area estimation methodologies that use in-country surveys that are capable of making predictions at the local level.
To see if there are modern approaches such as machine learning approaches so that policymakers can collect data and process it without requiring the use of expensive and logistically difficult local surveys, The idea behind the implementation of this research [23] by Swetava Ganguli et al. is to replace expensive logistically difficult surveys by using machine learning. Through the means of CNN we can use the satellite tasking data to train the system into predicting food security outcomes[18]. Through specific data analysis we can track food resources being supplied to an area alongside the economic aspects, we can track the population growth and agricultural resources available to ensure food security standards. Besides the chances of this prediction being accurate increases with more data being fed to the algorithm and with careful supervision and adjustments. The research addresses the current situation of gathering vast amounts of data through manual labor and it itendents to replace that manpower invested by using CNN. The methodical approach of training the neural network through filtered data points ensures a favorable predictability. As it has the possibility to give different outputs depending on the learning process, a highly supervised training process is a prime requirement. The research works with multiple data sources but the most important one is the satellite imaging. The Area of Interest is captured by optical satellites. As optical satellites have the capability to see infrared and ultraviolet, they can measure vegetation health and ocean heat levels too. This particular usage of satellite tasking, however, requires high-resolution imaging to ensure a healthy data input for CNN processing. The research also requires economic activity and population count for further calculation. The CNN model does not incorporate physical non-Markovian weather models to more accurately simulate actual satellite conditions. Again, using GoogLeNet or ResNet may be particularly helpful in ensuring that lower-level features are emphasized in the classification layer.
The objective of Edward J. Oughton and his team's work [52] was to provide a

solution consisting of an approach that presents a machine learning method that predicts telecoms demand metrics as well as spending on mobile services and cell phone adoption by utilizing publicly accessible satellite imagery and applying it to Malawi and Ethiopia. According to the findings of the study, CNN can be used to accomplish the desired result. Using data from the World Bank's LSMS, the research obtained measures of the relevant metrics. The Survey of Malawian Fourth Integrated Household in 20162017 and ESS in 20152016 were downloaded. The paper also got photos via the Planet web API, which queried daylight PlanetScope satellite images with a zoom level of 14 for the years 2014-2016. The CNN model was then trained with the help of these datasets. The paper investigated two methods for training the CNN model: (i) Training a model for a single nation; and (ii) Training a model for many nations. ii) Expanding the scope of the concept to encompass additional nations. CNN has been trained to classify an image's bin, which corresponds to a cluster and its images. Then, the transfer learning methodology was implemented in this work. The Visual Geometry Group which is also known as VGG[36] at the University of Oxford's pre-trained model was used. This is because the architecture is open source as well as highly efficient[36]. Utilizing this method, the VGG model is trained on the ImageNet dataset which contains millions of photos and more than a thousand subjects (e.g. automobiles, trees, and buildings). The research found that the CNN method used in this paper recorded data variance up to 41% for the prediction of cell phone adoption metrics, compared to 720% of the variance discovered by population density [17] and 821% of the variance discovered by nightlight luminosity [12][10], which are common baseline models. Future study advancements are possible if improvement of (1) justification of results, (2) techniques for validation, and (3) the attainability of satellite images, can be addressed. In this study[31], Babenko and his team are aiming to directly measure poverty based on high and medium-frequency satellite photos by training Convolutional Neural Networks (CNNs) algorithms utilizing GoogleNet and a variety of VGG designs. These images were obtained from satellites orbiting the earth at different frequencies. In the course of this inquiry, both urban and rural settings in Mexico will be accessed. The MCS-ENIGH household survey was integrated with other data sources to produce the poverty benchmark data, which was then utilized in their research. The author is of the opinion that the survey is very important because it collects data on income equality for individuals, whatever is a sort of salary matrix that is employed to compute the poverty rate which is official. They also conducted tests using various Intercensus sets to ascertain whether or not the amount of data points for households used to train the CNN'S algorithm had an effect on the reliability of the findings. They counted the number of low-income households in each administrative division to get this information. As a consequence of this, the process of making a point-to-point prognosis starts using satellite photos and concludes with an idea for each administrative region. Then, in order to accomplish one of the goals of the research, which was to determine how Planet and Digital stack up against one another in terms of the trade-offs, they made use of the satellite imagery data that was made available by Digital Globe and Planet images. In addition, it was mentioned that digital imagery from Digital Globe is primarily used in urban areas because there is insufficient coverage in rural regions. After that, they experimented with two different CNN designs, namely GoogleNet [21]and VGG [16]. Not only did they test using a number of different solvers, but they also analyzed with mass

initializations, which were evaluated in comparison to a "dev" set to determine their inherent capabilities. According to what they discovered, the design of GoogleNet is superior to the design of VGG in terms of its ability to be carried out. In addition to that, they experimented with different weighting schemes for the GoogleNet models. After that, they compared fine-tuned models by utilizing weights that were initially set to the values of a model that had been trained on ImageNet. It is challenging to fine-tune a model trained on ImageNet so that it works effectively with a 4-band input because the ImageNet sets of data are mostly built with RGB pictures. Because of this, they only investigated the fine-tuning process utilizing ImageNet to create the image's three band variants, trained from scratch for 4-band input, and ignored the NIR band. Additionally, they trained from scratch for 3-band input. They estimated the CNN predictions for urban regions based on the urban subsample by utilizing imagery from whether Digital Globe satellite [31] [26]or Planet satellite [31] [26]in addition to the ten percent of the certification sets that was withheld. The R2 is estimated to be 0.61 and 0.54, respectively, when using pictures from the Digital Globe and Planet. They then obtain a value of R2 among anticipated and genuine poverty among zero point four seven (0.47) and zero point five four (0.54) for 10 percent of the testing results for CNN speculate the In and Out of Sample.These projections are for CNN Prognostications Out of Sample and In. After combining the land cover categorization that was obtained from Planet images with the CNN predictions, they have come up with an approximate value of R2 among zero point five seven (0.57) and zero point six four (0.64). After comparing the approximation for all MCS-ENIGH placement, the authors' research shows that the coefficient of variance in metropolitan and countryside areas, respectively, decreases to 0.4 and 0.44 when seen in the context of the whole study. In addition to this, they possess several important factors for a variety of reasons. When looking further afield than the 896 localities that were surveyed for the MCS-ENIGH, the power of explanatory declines hurriedly to around zero point three (0.3). Why the accuracy of the MCS-ENIGH regions is so poor is a mystery. It's possible that this is the effect of evaluating tiles according to their geographical location rather than their population. It's also possible that MCS-ENIGH towns are different from non-MCS-ENIGH towns in other respects, such as having more homogeneity in the density of their populations or having a variety of shapes. There are a few other limitations as well, such as the fact that the author and his team were only able to figure out the metropolitan areas because there was insufficient information regarding the rural areas in the case of Digital Globe.
One study analyzed business and check-in data from YELP to find the optimal location for opening a restaurant [38]. Based on the algorithms ranking, regions were ranked using machine learning methods such as the Scikit-learn Linear regression model, Decision Tree, Cross-validation, and Support Vector Machine (SVM) algorithms. To illustrate the conclusions of the study, graphical representations of the accuracy of the decision tree, logistic regression and cross-validation, and support vector machine were compared (SVM). The end-user or businessman is provided with a rating based on the algorithms findings and a ranking that shows the best feasible location for establishing a company based on their rankings. Using the Chi-square approach, superfluous columns were deleted during feature selection, resulting in improved accuracy. The use of SVM prevents data overfitting since it functions well with unstructured or semi-structured data. Although the research

has a very high degree of accuracy, it does not account for any geographical information that may have influenced restaurant evaluations. But by doing so, there were no complex variable combinations that kept the model reasonably clean.
A smart city application was suggested in which the ideal placement for a firm in a smart city was estimated [35]. Consequently, restaurants are used as a use case, and entrepreneurs will obtain the optimal location using a web application that requires them to input certain features and characteristics, such as cost of a meal for a particular consumer, family category, gender specification, ages, and estimates will be made based on these inputs. A cluster of districts will be seen by entrepreneurs as the optimum location. Utilizing machine learning methods and hierarchical clustering algorithms enables this calculation. The algorithm starts by extracting characteristics from the raw data of 35 London areas (20112017). Using the LinearRegression, MLP, and SMOReg algorithms, models are trained. Hierarchical clustering is used to identify commonalities across districts and characteristics. They presented a regression model optimization approach to repeatedly test the model until models with the lowest Relative Absolute Error (RAE) percent for each feature are identified. In this case, the suggested regression model optimization may be utilized to build models with a reduced error rate by using models with multiple methods to estimate different characteristics, resulting in improved precision. The use of RAE (Relative Absolute Error) to assess the error percentage and repeatedly retrain the model contributed to this great level of accuracy. Even so, users of a web application might be given the opportunity to enter values rather than pick established ones. Users might have provided more alternatives as a feature input.
Presently, nightlight is used as an indicator in a variety of research fields. According to [37], the Day/Night Band or shortly known as DNB sensors onboard the S-NPP which is fully known as Suomi-National Polar-orbiting Partnership as well as the Joint Polar Satellite System (JPSS) spacecraft portal provides average daily global readings of night time visibility and near-infrared light (NIR) appropriate for global system scientific and applications study. In[14] and [6], the night light data of the earth was utilized as a proxy variable to assess the regions level of poverty. In a different study[33], night light was utilized to forecast the areas wealth. This type of research focuses specifically on rural regions; thus, by incorporating night light into our study, we can determine a regions economic situation, allowing us to more accurately forecast the conclusion. This research investigates the elements that affect small businesses effectiveness or failure in Ghana's rural areas, a nation where more than 43 percent of the population lives in rural areas.
In research published in 2014 [15], a machine learning model was trained to produce a financial rating (A, B, or C...) based on the geographical characteristics of a specific region. This grade will indicate how ideal a site is for establishing a retail business. It gets geographical data around the place through the Google API, then divides this data into many clusters using the k-medoids method to assign each cluster of shops a unique generic score. The machine learning model is then trained using an Artificial Neural Network(ANN) where geographical data is the feature and clustering-based ranking is the label. According to [50], the K-Medoids clustering technique was employed instead of k-Means to cluster the retailers based on their financial rating since it is more resilient and efficient. Using this technique, it is possible to estimate the financial rating of a nonexistent retail business based purely on the surrounding geographical parameters. The chosen network is suitable

because of its low sum of squared errors throughout all trained datasets but instead of precise prediction on out-of-sample observations. It has very low MAD, MSE, and MEAD values relative to the highest attainable values for the poorest possible predicting performance. The rating is highly precise since the financial data was acquired digitally, leaving no possibility for measurement mistakes, and there was sufficient training data. The models assessment was also compared to several prospective retail outlets, which found commonalities; thus, the ranking was evaluated by specialists. It was fairly simple to create since the Google Places API was used to extract features. Due to the absence of measurement error resulting from digital data collection, the models precision is relatively good. Using the appropriate number of neurons during ANN training resulted in a more accurate model. Some estimates were based on proxy characteristics that are often difficult to extract, such as the schools along with universities numbers,which, due to its difficulty in gathering, is utilized as a surrogate measure for the population's teaching and learning environment. population data inside a circular region. It will be less successful in areas where there are fewer businesses. Instead of a ranking system, a scoring system would have been more appropriate since it would have allowed for more exact comparisons across sites.
This paper[45] represents a new method for improving the site selection process of retail businesses using a combination of spatial competition strategy and a double-channel convolutional neural network (CNN). The spatial competition strategy is based on the idea that the location of a retail store can have an impact on the attractiveness of nearby stores, and the double-channel CNN is a type of machine learning model that is able to analyze large amounts of data and make predictions.The authors first describe the existing methods for site selection in the retail industry, which often rely on subjective judgment and do not take into account the competitive environment. They then propose the use of a spatial competition model, which considers the influence of nearby stores on the attractiveness of a potential site, as well as a double-channel CNN, which can analyze data such as demographic information and consumer behavior to make more accurate predictions about the success of a store at a particular location.In order to test the effectiveness of the model, the authors fundamentally offer a geographical competition index model and a market demand regression model based on the spatial correlation range of features. They then use monthly sales data from fast-moving consumer goods retail establishments in Guiyang. The suggested model outperformed other popular regression techniques, according to the authors, and reduced root mean square error by 22.61 percent when compared to a single-channel CNN and 19.29 percent when compared to XGBoost[45]. 18 sites were suggested based on the model's anticipated findings[45] because they had a strong potential for market demand. They discover that their methodology performs better than current methods, especially in terms of accuracy and efficiency. Finally, this paper presents a novel method for improving the site selection process for retail businesses, which combines spatial competition analysis and machine learning techniques to more accurately predict the success of a store at a particular location.
This paper [39] represents that choosing the right location for a retail shop is a crucial factor in the success of the business. In order to make informed decisions about where to locate a retail shop, it is important to consider the spatial accessibility of different locations. Spatial accessibility refers to how easily a location can be

reached by potential customers, and it is influenced by factors such as the location's proximity to transportation options and the distance to other nearby attractions. To help with the site selection process, the authors propose using a hybrid BP neural network, which is a type of artificial neural network that combines back-propagation with other techniques. The authors explain that this type of neural network is particularly well-suited for modeling and predicting spatial accessibility because it can handle both continuous and categorical data, and it is able to capture complex relationships between variables. To test their approach, the authors conduct a case study in a city in China. They gather data on various locations within the city, including information about the locations' distances to transportation options and nearby attractions, as well as demographic data about the surrounding areas. They then use the hybrid BP neural network to model and predict the spatial accessibility of each location. The results of the case study show that the hybrid BP neural network is effective at identifying suitable locations for retail shops based on spatial accessibility. The model was tested in the city of Guiyang, China and was found to accurately identify 42 locations with high business potential for new retail shops[39]. The authors conclude that their approach could be useful for businesses and other organizations that are looking to choose the best location for a new retail shop.

Chapter 3

Methodology

3.1 Target Identification

Here we aim to identify the target variable that will be used to predict the optimal location for a new restaurant or a new branch. Our target variable is the total number of check-ins for each restaurant in the year 2021 and 2022.We began by using the restaurant data from the Yelp dataset, which initially contained 51,302 records of restaurants. However, in order to ensure that our predictions were based on relevant and active restaurants, we trimmed the dataset to only include restaurants that were currently open, and had at least 50 check-ins in the year 2021 and 2022. This resulted in a final dataset of 1,108 restaurants.The total number of check-ins in the year 2021 and 2022 was chosen as the target variable because it serves as a good indicator of the popularity and success of a restaurant. A higher number of check-ins suggests that the restaurant is well-visited and likely to be successful in terms of revenue and customer satisfaction. By using this variable as the target, we aim to identify locations that are likely to have high footfall, and therefore, suitable for a new restaurant.This target variable will serve as a reliable indicator of the popularity and success of a restaurant, and will be used to predict the optimal location for a new restaurant using advanced machine learning techniques.

3.2 Feature Space Engineering

Our goal was to identify and extract the most informative features from the multiple data sources used in our analysis. To accomplish this, we implemented a variety of techniques, such as scaling, data aggregation and outlier handling. These techniques enabled us to create a comprehensive and informative feature space that effectively represented the characteristics of the area and the potential success of a new restaurant.

3.2.1 Scaling

To ensure that all the features in our dataset were on the same scale, we applied different scaling techniques such as Min-max Scaler, Standard Scaler, Robust Scaler, and MaxAbs Scaler. These techniques offer fair comparison and analysis. We then trained our model using these different scaling techniques and selected the best one based on the performance of the model. The employed methods were:

1. **Min-max Scaler:** This adjusts the data to lie within a defined range, such as between 0 and 1. Subtracting the minimum value from each data point and then dividing by the range accomplishes this (i.e. the difference between the minimum and maximum values).
$$Min - max\ Scaler = \frac{data - min}{max - min}$$
2. **Standard Scaler:** Data can be standardized using the Standard Scaler normalization technique, which involves transforming the data so that it has a mean of 0 and a standard deviation of 1. To accomplish this, we first subtract the mean from each individual data point and then divide that total by the standard deviation. If all the features in a dataset are normalized so that they are all of the same scale and have the same properties, then the performance of the corresponding machine learning algorithms can be enhanced.
$$Standard\ Scaler = \frac{(data - mean)}{standarddeviation}$$
3. **Robust Scaler:** Robust Scaler is comparable to regular scaler normalization, but is less susceptible to data outliers. Instead of utilizing the mean and standard deviation to standardize the data, the median and interquartile range (IQR) are used. The interquartile range is the difference between the 75th and 25th percentiles of the data and is used to compute the scale factor. Then, the median is subtracted from each data point, and the resulting value is divided by the interquartile range (IQR). When the data contains outliers that would normally bias the mean and standard deviation, this method can be advantageous.
$$Robust\ Scaler = \frac{x - Median}{IQR}$$
4. **MaxAbs Scaler:** Max Abs Scaler is a method for scaling data by modifying it so that the maximum absolute value of each feature is equal to one. Each feature is divided by its maximum absolute value. This normalization method is effective when the data contains huge values for specific characteristics, which can cause them to dominate the model. By scaling the data in this manner, all of the model's features are on the same size and have equal value. It is very beneficial when working with sparse data because it scales the data without eliminating the sparsity.
$$MaxAbsScalar = \frac{data}{max(|x|)}$$

After using these scaling techniques each type of data were tested on all the ML models. Out of all of the models Lasso Regression performed better without any scaling of data. While the rest of the models performed well after min-max scaling. That is why min-max scaled data was used for further analysis. For the target or label data min-max scaler was used as well to convert the values of check-ins in each restaurant into a value between 0 to 1. This provided better and more efficient performance from every model.

3.2.2 Data Aggregation

We employed data aggregation as a key technique to extract informative features from various data sources. Data aggregation is the process of collecting, combining, and summarizing data from multiple sources into a single, more meaningful, and useful format. In order to make it easier to analyze and interpret the data, data aggregation aims to condense large and complex datasets into a more manageable and meaningful format. This results in a more complete and informative feature space that better represents the characteristics of the area and the potential success of a new restaurant. There are several techniques used in data aggregation, such as:

1. **Summarization:** Reducing a dataset to its most important or relevant elements by removing or consolidating redundant or less important data. For example, calculating summary statistics such as mean, median, and standard deviation.
2. **Grouping:** Classifying data into groups or categories based on certain criteria, such as location, time, or other characteristics. This allows for the comparison of groups and the analysis of patterns within them.
3. **Filtering:** Removing irrelevant or unnecessary data from the dataset.
4. **Merging:** Combining multiple datasets into a single dataset to create a more comprehensive view of the data.

Data aggregation can be applied at different levels of granularity, from low-level data elements to high-level summaries. Depending on the type of data and the goals of the analysis, different data aggregation techniques may be more appropriate.

In our work, we used data aggregation to combine information from different data sources such as Yelp, Bing Maps, Open-StreetMap (OSM), and Worldpop population count. We then grouped, filtered, and merged the data to extract meaningful features that encapsulated the key characteristics of the area. We also aggregated OSM features we got from different sectors, which are discussed in the OSM data extraction section. We used the aggregated data to train and evaluate our model to predict the optimal location for a new restaurant.

3.2.3 Handling Outliers with Z-score

We also handled outliers in our dataset using the z-score technique. The z-score, also known as the standard score, is a measure of how far a data point is from the mean of a dataset in terms of standard deviation. It is calculated using the following formula:

$$z = \frac{x - \mu}{\sigma}$$

Where x is the data point, μ is the mean of the dataset, and σ is the standard deviation.
By using this method, we were able to identify data points that fall outside of a certain threshold (typically 3 standard deviations), which are considered outliers. These data points were then removed from the dataset to ensure that our analysis was based on a more accurate and representative sample of the data.
This method is useful because it allows us to identify outliers in a standardized way,

regardless of the scale of the data. It is also relatively simple to calculate and easy to implement. However, one disadvantage of this method is that it assumes that the data is normally distributed, which may not always be the case. In cases where the data is not normally distributed, other methods such as interquartile range (IQR) or modified z-score method may be more appropriate. It was an effective technique for identifying and handling outliers in our dataset. By removing these outliers, we were able to ensure that our analysis was based on a more accurate and representative sample of the data. This helped to enhance the performance and reliability of our model in predicting the optimal location for a new restaurant.
Through the use of scaling techniques, data aggregation, and outlier handling, we were able to extract the most relevant and useful information from the data sources used in our analysis. These techniques helped us to generate a comprehensive and informative feature space that better represented the characteristics of the area and the potential success of a new restaurant.

3.3 Machine Learning Approaches and Algorithm

Machine learning(ML) is a study field which allows machines to learn except specific programming[1]. This process analyzes and interprets data by using algorithms and statistical models, allowing the machine to continuously improve its performance and ability to make decisions over time. It is based on the idea that algorithms can learn from data, find patterns, and make decisions with as little human help as possible. It involves putting a lot of data into a computer program, which then uses statistical analysis to find relationships in the data. The algorithm can then utilize these patterns to generate predictions or decisions without being explicitly instructed to do so. Machine learning is an efficient method for resolving complex problems. We'll examine how various machine learning models approach the problem of determining the optimal place to open a business in this paper. We will be working with big data derived from a variety of sources, such as mobile check-ins, OSM maps data, satellite imagery, and other populace data. There are many subfields within machine learning. In this work, we'll be focusing on supervised machine learning techniques.

Supervised machine learning is the process of training a model with labeled data when the correct output for each example in the training set is provided. If we look at an input variable (P) and an output variable (Q), then the prediction function that the algorithm learned through training is the function f(P). The equation for supervised machine learning will be,

$$Q = f(P).$$

But it should be kept in mind that the specific form of the prediction function will depend on the model type and the problem being solved. There are two major categories of supervised machine learning.

1. **Classification:** Classification is a type of supervised machine learning in which the input data is used to predict a discrete class label. In this type of learning, the data is organized into categories. For instance, a classifier may be

trained to determine whether or not a certain shot would result in a goal. The model would make a prognosis of either "goal" or "not goal" when it was given a "shot" as an input. One of the binary type classifiers is represented here. Another illustration of this would be using labeled training data to determine the kind of animal depicted in an image. The training data would consist of photographs of various animals together with the appropriate identification label for each picture, such as "dog," "cat," "owl," and so on. The model would become able to categorize new pictures by looking at the patterns and characteristics that it discovered in the training data.

2. **Regression:** Another form of supervised machine learning, regression forecasts a continuous numerical value by using an input dataset. The model is trained to predict a matched numerical label in a problem that is of the regression kind. In this sort of problem, the input data is a set of features. Therefore, the figures produced by a regression model are more comparable to actual numerical values. Regression has a variety of approaches, including nonlinear regression, logistic regression, and linear regression, to mention a few. At the core of the easy modeling technique known as linear regression is the assumption of a linear connection between the input data and the output value.

In this research an assortment of regression models have been used to draw a more concrete figure of comparison when finding an optimal location. Noteworthy models used in this paper are Support vector Regression (SVR), Random Forest (RF), XGBoost algorithm, Ridge Regression algorithm, Lasso Regression algorithm, ElasticNet algorithm, CNN and Neural networks. We chose these algorithms because they are popular and have been widely used in related works. SVR is a powerful algorithm that can handle not only linear but also non-linear regression problems. Random Forest is an ensemble algorithm that combines multiple decision trees to improve the accuracy and reduce overfitting. XGBoost is an improved version of Random Forest, it uses gradient boosting to optimize the decision trees. Ridge Regression and Lasso Regression are both linear regression algorithms, with Ridge Regression used to address overfitting and Lasso Regression used for feature selection. ElasticNet is a linear regression algorithm that combines the Ridge and Lasso Regression, it is useful when there are multiple correlated features. We used neural networks for a regression problem because they are suitable for regression problems as they are able to learn complex relationships between inputs and outputs and capable of modeling non-linear relationships in the data. CNNs are powerful for image recognition and are suitable for processing satellite imagery. Overall, these algorithms were selected for their performance in related works and their ability to handle different types of data and features

3.3.1 Support Vector Regression(SVR)

Support Vector Machine(svm) is a supervised learning algorithm technique which is used for regression along with classification applications[9].We worked with Support vector regression(svr) which is based on the principles of svm.In order to understand how svr works, let's consider the figure 3.1.

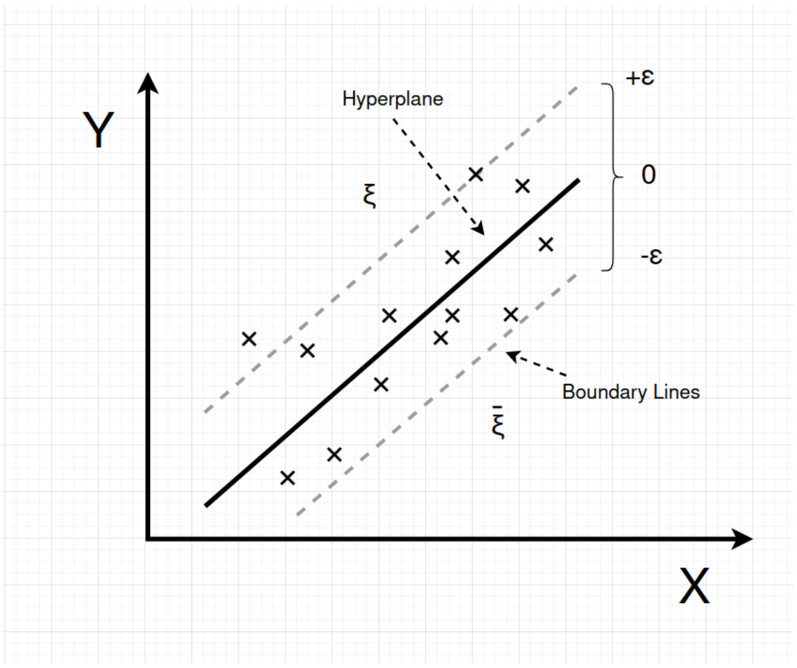

Figure 3.1: Support vector regression(based on svm) workflow

The black line between x and y axis is called Hyperplane which indicates the line of separation between two data classes. In order to create margin between data sets, we used two Boundary lines.The distance between a Hyperplane and each boundary line is considered as epsilon.There are two epsilon, one is for positive boundary line and other for negative own.The "X" symbol indicates the data sets.The closer point from the hyperplane is called support vectors.We can see that most of the data points are situated near Hyperplane and rest of the data points situated inside the positive and negative boundary lines.So, we will try to find the planes in some way that the data point situated apart from positive and negative boundary lines are reduced.These approach will reduce the error as much as possible. The application area where this algorithm is used is given below.

- It is known as a robust algorithm which can handle non-linear data and perform accurately with noisy or outlier data[5].
- Even with a small training data, it can provide correct predictions along with the ability to generalize efficiently to new data sets.
- It is also efficient with large datasets and usable for high-dimensional data.

There are some cases when SVR is not applicable for uses.This is discussed below.

- The kernel function used by the SVR method may not be appropriate for particular dataset[5].
- The parameters of the SVR algorithm might be improperly selected.
- It is possible that the dataset is too tiny for the algorithm to generate accurate results.
- The data set may contain excessive noise or outliers.

3.3.2 Random Forest

The objectives of classification and regression can be fulfilled with the help of Random Forest, an ensemble technique for machine learning. Lets consider figure 3.2 for understanding how RF algorithm works. We take some training data sets. For

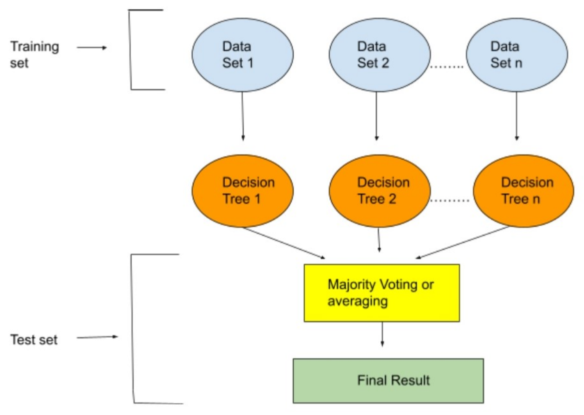

Figure 3.2: Working priciples of Random Forest Algorithm

each set of training data, it will generate corresponding decision trees. Then the algorithm will take the prediction from each decision tree in place of depending on one, and also predict the final result based on the majority vote of prediction or averaging. The last step can be done with test data sets. The working principle of this algorithm is discussed below.

- Random Forest is simple to design, can manage a lot of features, and often makes very precise predictions.
- Random Forest builds a lot of decision trees at the time of training step and combines their predictions to produce a final set of results.
- The decision trees that comprise a Random Forest are trained on different samples of the data and feature subsets and avoid becoming overly particular,while at the same time, improves the model's ability to generalize.

Some of the areas when Random Forest is used are given below,

- Random Forest is particularly useful in scenarios where interactions between features and target variables are complex and non-linear [22].

There are a few potential issues that could prevent the Random Forest algorithm from generating the best possible outcomes.

- Random Forest could not be able to create reliable predictions if data has poor quality, with many missing or incorrect values in the data.
- Random Forest may overfit to training data and perform poorly on test data if there is a substantial degree of correlation in the data [22].

- Incorrectly configured hyperparameters, such as the maximum depth with each tree as well as the amount of decision trees, may cause poor performance of Random Forest and it is important to use cross-validation for fine-tuning these hyperparameters for optimal performance.

3.3.3 XGBoost

XGBoost is a gradient-boosting approach that sequentially generates decision trees.

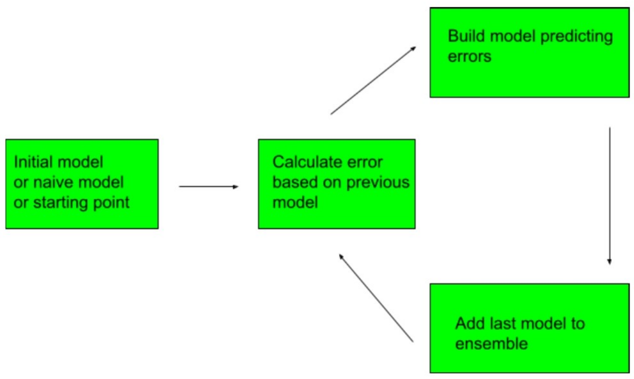
Build model predicting errors
Initial model or naive model or starting point
Calculate error based on previous model
Add last model to ensemble

Figure 3.3: Working priciples of XGBoost Algorithm

In figure 3.3, beginning with the initial model or starting point, the algorithm will calculate error based on the previous model(initial model).Then in the next step, it will build a new model predicting those previous stage errors.Then it will include the last model in the ensemble.Lastly, it will again repeat step 2 which will calculate the errors based on the previous model until the accurate final model.There are a lot of working principles of this algorithm which are discussed below.

- The model utilizes predictions provided by the trees that came before it to enhance its performance.
- It is a well-known solution for many data science issues because of its speed, scalability, and performance.

XGBoost is used in some areas. Below they some of them are discussed,

- In the context of site selection, XGBoost can be used to estimate future success of a place by comparing patterns in data such as demographics, economic indicators, and geographic factors to the region in question and identifying similarities[27].
- XGBoost is able to handle enormous datasets, high-dimensional data, missing values and class imbalance.
- XGBoost has a reputation for excellent precision, and low generalization error rates in addition to handling large and high-dimensional datasets, missing values, and class imbalance [27].

There are a variety of reasons that can be said if XGBoost fails to have the desired impact.They are,

- The poor performance of XGBoost model can be also caused by non-optimal hyperparameters, such as learning rate and maximum tree depth, which are not properly calibrated that results in suboptimal performance[41].
- XGBoost can be susceptible to poor performance when data is not optimal for the method, such as high imbalanced or high number of missing values.

3.3.4 Ridge Regression

In terms of linear regression method, ridge regression is a common algorithm that is used to explain the relationship between one or more than one independent variables and a dependent variable.

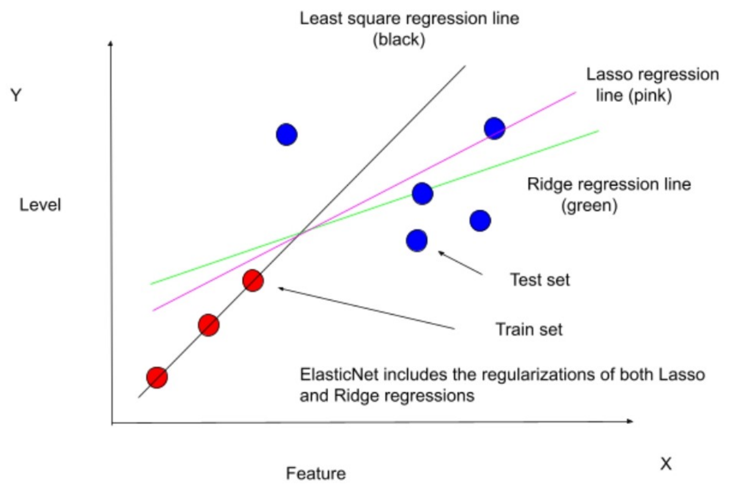
Least square regression line (black)
Lasso regression line (pink)
Y
Level
Ridge regression line (green)
Test set
Train set
ElasticNet includes the regularizations of both Lasso and Ridge regressions
Feature
X

Figure 3.4: Ridge,Lasso and elasticnet Regression Workflow

In figure 3.4, the X and Y axis indicate the feature and level.Red points and blue points indicate train and test sets.Black and green line indicates the least square regression line and ridge regression line.Here, the least square regression line is best fitted with the train set but failed to fit with the test data sets.So we can say the model is overfitting.What ridge regression is going to do is, it reduces the slope a little bit and try to fit with the test data sets.Ridge regression basically uses a penalty term.In some cases, the ridge regression may not be able to fit with the train sets accurately but it will fit with the test sets perfectly and for this reason, it will reduce the error calculation. We can also see that some blue points are fitted with the ridge regression line in these figures.It is applicable to any quantity of independent variables.

- Ridge regression is a linear regression which includes a penalty term using L2 regularization to the reduced objective function.

- The penalty term is calculated by multiplying the total of all coefficient squares(except the intercept) by a regularization value in ridge regression[2].
- The regularization term in Ridge Regression aims to reduce the magnitude of the coefficients by shrinking them close to zero which improves the model's ability to generalize the data by preventing overfitting.

Ridge Regression is used in many different areas. they are,

- Ridge Regression is a good choice for site selection projects because it can handle high-dimensional data and multicollinearity(correlation between independent variables), making it a feasible solution for such tasks.
- Ridge Regression also allows to identify the most significant predictors of a model.

There are numerous potential explanations to explore if Ridge Regression fails to produce the desired result.

- Ridge Regression may not perform well when the data is not suitable for the method, such as when it contains a large number of categorical variables or nonlinear correlations.
- Poor performance of Ridge Regression model can also be caused by non-optimal calibration of hyperparameters such as regularization strength[2].

3.3.5 Lasso Regression

The Lasso Regression is one of the linear regression techniques that is used to describe the connection between one or many independent variables and a dependent variable.If we want to understand the workflow of lasso regression, it is similar to ridge regression but the difference is that lasso regression use different penalty term and the slope of lasso regression line is a little bit different with ridge regression slope if we look at the figure 3.4.The lasso regression line is indicate with pink color there.
The working principle of this algorithm is discussed below.

- Lasso Regression is a form of linear regression that uses regularization to prevent overfitting that encourages lowering the amount of the model's coefficients, setting some of them to zero, effectively removing the corresponding feature and improving the model's generalizability.
- "L1 regularization" which includes a penalty term to the reduced objective function in order to achieve the desired outcome is applied by Lasso Regression.[11].
- To increase the model's ability to generalize and avoid overfitting, the penalty term is computed by multiplying the total of the absolute coefficient values, except the intercept term, by a regularization parameter.
  In many cases Lasso Regression is useful. They are discussed below,
- Lasso Regression is useful because it can effectively handle high-dimensional data and multicollinearity.

- By putting the coefficients of less important variable to zero as well as finding a model's most relevant predictors, Lasso Regression is effective for variable selection.[28].

There are a lot of probable explanations to explore in the event that Lasso Regression does not produce the anticipated outcome.

- One potential explanation for poor performance is due to the data not being optimal for it, such as having a large number of categorical variables or nonlinear connections [11].
- The Lasso Regression model's hyperparameters (such as the regularization strength) were not calibrated appropriately which resulted in less than ideal results.

3.3.6 ElasticNet Regularization

ElasticNet is a method for linear regression which represents the association between one or more independent variables and a dependent variable.The workflow of elasticnet is similar with ridge and lasso algorithm but the slight difference is that it uses both penalty terms from ridge and lasso algorithm.
The working principle of ElasticNet Regularization is discussed below.

- ElasticNet uses regularization to prevent overfitting and improve model generalization, similar to Lasso and Ridge Regression [4].
- ElasticNet includes the L1 and L2 regularizations of both Lasso and Ridge regressions.

The working principle of this algorithm is discussed below.

- ElasticNet can handle high-dimensional data and multicollinearity well, making it a good choice for site selection tasks.
- ElasticNet can identify important predictors in a model by setting the coefficients of less important variables to zero [7].

If ElasticNet fails to produce the optimal result, it could be due to a variety of reasons.

- The algorithm may not perform well if the data used is not well-suited, such as containing large number of categorical variables or nonlinear relationships, which may be one of the possible reasons.
- The ElasticNet model's hyperparameters (such as the regularization strength and the mixing parameter) may not have been tuned appropriately, resulting in suboptimal performance.

3.3.7 Neural Network

A Neural Network, a statistical mechanism for approximating the output based on a large number of inputs, is the back-end of deep learning feature extraction. Neural Network trains a model on a large number of data points without the need for explicit programming of procedural instructions, similar to other statistical methods that construct probability distributions to produce inferences from data. The model is constructed to discover patterns in the distribution of the data, which can then be used to draw conclusions about samples not observed during the training process. It is assumed that the large number of data points in the sampling distribution used for training are representative of the population, and thus the patterns within the data can be used to capture the relationships between variables in the population distribution.

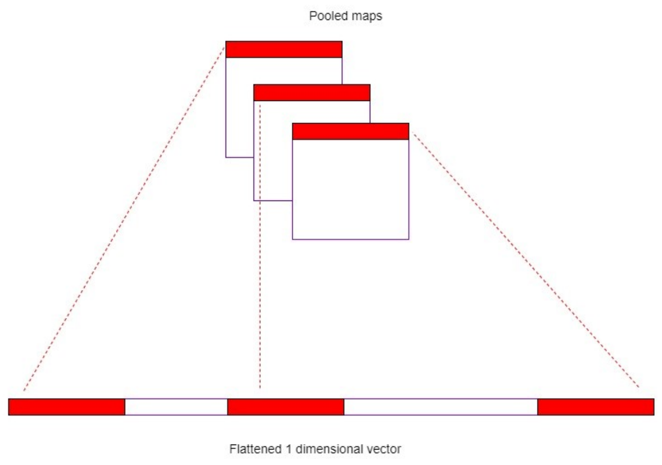

Figure 3.5: Flattened pooled maps for model input

The model is constructed with three main components:

- **Input layer:** The input data must be converted to a numerical format and then fed to the input layer. These numerical representations may be vectors, matrices, or tensors. In order to simulate the interconnected neural circuits of the human brain, the input layer contains a number of nodes (neurons) that connect to other nodes in subsequent layers of the network. Each node receives input data and transmits it to the next layer through a channel;
- **Hidden layers:** Between the root and the terminal nodes of the network are the hidden layers, which receive data from the input layers as parameters and perform nonlinear transformations of these inputs with specified weights and biases before directing them through the activation function for the output layer, which determines whether a node is activated or not;
- **Output layer:** The data finally enters the output layers, and an output result is calculated using the activation function. This output is compared to the ground-truth data, and the loss is calculated to determine the model error and its distance from the target. Using this data, the optimizer adjusts the model's weights and biases and continues to feed them back through the network in an iterative process known as back-propagation until the training is complete or the desired accuracy metrics are achieved.

There are some other factors and parts of Neural Networks that can play a huge role in the performance of the model. The teqniques that were mainly used in this work to get an optimal result are:

- **Dropout Layer:** Dropout is a regularization technique used in neural networks to prevent overfitting by randomly omitting a percentage of neurons during each training iteration. This facilitates generalization by preventing complex co-adaptations to the training data. The dropout layer is typically placed between the layers of a neural network that are fully connected.
- **Batch normalization:** This technique is used to improve the performance and stability of neural networks by normalizing the activations of the previous layer in each batch. This facilitates the reduction of internal covariant shift and the acceleration of the training procedure. Typically, batch normalization is applied before the activation function of a layer.
- **Activation functions:** Activation functions are mathematical functions applied to the output of a neuron to determine the output of the neuron. The two most frequently employed activation functions are the ReLU (Rectified Linear Unit) and the sigmoid function. The ReLU is defined as max(0,x) and is used in hidden layers to solve the problem of vanishing gradients. In output layers, the sigmoid function (S) is used to obtain probability values between 0 and 1.

$$Sigmoid,\ S(x) = \frac{1}{1+e^{-x}}$$

- **Loss Function:** A loss function is a metric that measures the difference between a neural network's predicted output and its actual output. The goal of training a neural network is to minimize the loss function's value. Mean Squared Error (MSE), Mean Absolute Error (MAE) and Huber loss are frequent loss functions for regression problem.
- **Optimizers:** Optimizers are algorithms used to adjust the parameters of a neural network in order to minimize the loss function's value. Stochastic gradient descent (SGD), Adam, and RMSprop are common optimizers.
- **Learning Rate (LR):** The learning rate is a hyperparameter that determines the size of the steps at which the optimizer updates the neural network's parameters. A low learning rate can lead to slow convergence, whereas a high learning rate can cause the optimal solution to be overshot.
- **Early Stopping:** Early stopping is a technique used to prevent overfitting in neural networks by terminating the training process before the model overfits the training data. This is achieved by monitoring the model's performance on a validation set and stopping training when performance begins to degrade.
- **Number of Epochs:** An epoch is a complete traversal of the entire training dataset. During training, the number of epochs is a hyperparameter that determines how many times the model will cycle through the training dataset. The greater the number of epochs, the greater the model's exposure to the data, but the greater the likelihood of overfitting.

3.3.8 Ensemble Learning

Ensemble learning is a method of combining multiple models in order to improve the overall performance of a machine learning system. This is done by training multiple models on the same data and then combining their predictions. The idea behind ensemble learning is that multiple models can provide a more robust and accurate prediction than a single model alone. There are several different ensemble learning methods, including:

- **Bagging:** Random Forest is an ensemble learning method that is based on the concept of bagging. Bagging stands for Bootstrap Aggregating, which is a technique that involves training multiple models independently on different subsets of the data. In the case of Random Forest, each model is a decision tree and the subsets of data are created by randomly selecting a subset of the features for each tree. The final prediction of a Random Forest model is made by averaging the predictions of all the decision trees in the forest.
- **Boosting:** XGBoost is an ensemble learning method that is based on the concept of boosting. Boosting is a technique that involves training multiple models sequentially, where each model focuses on correcting the mistakes made by the previous model. In the case of XGBoost, each model is a decision tree and the training process is designed to give more weight to the examples that were misclassified by previous trees. The final prediction of a XGBoost model is made by combining the predictions of all the decision trees in the sequence.
- **Stacking:** SVM, Ridge Regression, Lasso Regression, and Elasticnet are not ensemble learning algorithms. They are individual models that can be used for prediction, but they don't work by training multiple models and combining their predictions. They can be used as base models in ensemble learning methods like stacking, where they are trained independently and then the predictions of these models are combined by training a meta-model.

Ensemble learning can be effectively used in optimal site selection, by combining predictions from different models trained on different data sets and features, such as satellite imagery, demographic data, and environmental data. By combining these predictions, the ensemble model can provide a more accurate prediction of the suitability of a site for a particular use, such as the location of a new retail store or a new housing development.

3.4 Hyperparameters of Machine Learning Algorithms

Hyperparameters are a kind of parameters which are established before to train a model and are not learnt from data.Tuning some hyperparameters is a crucial step in getting excellent performance and libraries such as grid search cv are used for this purpose.Grid search is a strategy that involves defining a collection of hyperparameters and training along with evaluating the model of these hyperparameters for every possible combination. These guarantees that the best combination of hyperparameters will be found but is computationally expensive. Grid search is considered to be more effective as it evaluates all possible combinations of hyperparameters, however, it depends on the amount and quantity of the data set of hyperparameters to tune.

3.4.1 Support Vector Regression(SVR)

The main hyper-parameters that can be adjusted in order to fine-tune an SVR are:

- **Kernel:** One of the most important parameters to tune in an SVR is the kernel type. The kernel is basically a function which maps the input data into a higher-dimensional linearly separable space. Kernels with linear, polynomial, and radial basis functions are frequently used. Without any transformation, the linear kernel computes a dot product between the input features. It's fast to compute and works well when the data is linearly separable. The polynomial kernel can capture more complex non-linear relationships between the input features. The radial basis function(RBF) kernel is determined by the distance between input characteristics and a specific center point. It's a smooth and versatile kernel function that can capture a wide range of non-linear relationships. In some cases, linear kernel may work best, while in other cases polynomial or RBF kernel may be more appropriate.
- **Gamma:** It is a kernel-specific parameter. Gamma is a parameter used in non-linear kernels, such as the RBF kernel, that controls the width of the kernel.A broad value of gamma results in a smaller width of RBF kernel and a higher degree of complexity.It means that the decision boundary will be more sensitive to individual data points. On the other hand, a smaller gamma value results in a larger width of RBF kernel and a lower degree of complexity. It means that the decision boundary will be less sensitive to individual data points.
- **Epsilon:** The "epsilon" parameter controls the width of the margin in the Support Vector Regression (SVR) algorithm. The purpose of SVR is to reduce the error between the true value and the predicted value while maintaining the error below a specific threshold, indicated by epsilon. A smaller epsilon value results in a smaller margin and a more restrictive model, where the majority of the data points will lie within the margin, while a larger epsilon value results in a larger margin and a less restrictive model, where more data points may lie outside of the margin. In some cases, a small value of epsilon may work best, while in other cases a large value of epsilon may be more appropriate.

3.4.2 Ridge Regression:

Tuning the Ridge Regression algorithm involves adjusting some of the parameters that are discussed below.

- **Intercept term:** The intercept term, also known as the bias term, is the value of the predicted response when all predictor variables are equal to zero.It controls whether or not the model has a non-zero y-intercept.In general, when the data has an inherent offset or baseline, including the intercept term in the model can be beneficial, as it allows the model to make predictions even when all of the feature values are zero.In some cases, the data has been centered or standardized, and there is no inherent offset, the intercept term may not be necessary and excluding it may simplify the model and improve its performance.Some implementations may have an option to include the intercept term in the regularization by setting the "fit intercept" parameter to True. This can be useful when the data is not centered, and want the model to have a non-zero average prediction.The intercept is typically included by default, but it can be set to zero by setting the intercept parameter to false.It's important to note that if the data is normalized or standardized, the intercept term will not be equal to the mean of the response variable.

- **Solver Parameter:** The solver parameter controls the optimization algorithm used to find the optimal coefficients. Different solvers have different strengths and weaknesses. Ridge regression can be solved using different optimization algorithms such as "svd", "cholesky", "lsqr", "sparse cg", "auto", "saga" and "sag".The selection of a solver will be determined by the features of the data and the available computation time. "sag" solver for example, is generally faster for large datasets, but "cholesky" is more efficient for small datasets. "Auto" solver will choose the solver that best suit the problem at hand, it will select "svd" solver when the number of samples is less than a certain threshold, otherwise it will use "cholesky" solver.On the other hand, "saga" solver is a stochastic gradient descent solver that is well-suited for large-scale problems with a large number of samples and features.It can also deal with sparse data as well as robust to outliers.The "saga" solver is generally slower than "auto" solver as it performs an additional pass over the data to compute the gradients. However, it can be more accurate than "auto" solver in some cases.The "auto" solver will select the best solver for the problem at hand, but it's recommended to experiment with different solvers to see which one gives the best performance for specific problem.

3.4.3 Lasso Regression

The process of optimizing the performance of the Lasso Regression algorithm includes adjusting certain parameters that are presented in the following discussion.

- **Alpha:** To fine-tune a Lasso regression machine learning algorithm, we can adjust the hyper-parameter alpha, which controls the amount of regularization applied to the model.A smaller alpha value will result in less regularization and a more complex model, while a larger alpha value will result in more regularization and a simpler model.
- **Intercept:** The fit intercept parameter defines if or not an intercept term ought to be included in the design. If set to True, the model will include an intercept term and will be able to fit the data better, but the number of parameters that must be estimated will rise, making the design more complicated. If set to False, the model will not include an intercept term, resulting in a simpler model.
- **Positive:** The "positive" parameter in Lasso Regression controls whether or not to force the coefficients of the model to be positive.This is useful in certain cases where only positive values of the coefficients make sense, such as when

the outcome variable is positive as well.When the positive parameter is set to True, the Lasso algorithm will only produce non-negative coefficient estimates. When it is set to False, the Lasso algorithm will not impose any constraints on the coefficients.

- **Precompute:** The "precompute" parameter in Lasso Regression controls whether or not to precompute the Gram matrix (i.e. the dot product of the design matrix with itself) and the Gram matrix of the design matrix and the target vector. The precomputing of these matrices can save computation time if multiple Lasso fits are performed with the same input data.When True is used for the precompute parameter, the Gram matrix will be precomputed and sent to the Lasso regression. When False is used, the Gram matrix will be computed on the fly.

3.4.4 ElasticNet

Fine-tuning the ElasticNet algorithm involves adjusting certain parameters which will be outlined below. This process is necessary for achieving optimal performance on a specific task which is given below.

- **Alpha:** The "alpha" parameter in ElasticNet controls the overall strength of the regularization term, it is used to balance the trade-off between fitting the data well and avoiding overfitting.A higher value of alpha will result in a more robust model, but can also lead to underfitting.A lower value of alpha will result in a less robust model, but may be more prone to overfitting. In some cases, a small value of alpha may work best, while in other cases a large value of alpha may be more appropriate.
- **Ratio:** The "ratio" parameter in ElasticNet controls the balance between L1 and L2 regularization.A ratio of 1 corresponds to Lasso regularization, which will result in a model with sparse coefficients, where many of the coefficients are exactly equal to zero.A ratio of 0 corresponds to Ridge regularization, which will result in a model with non-sparse coefficients, where most of the coefficients are non-zero but small.Both L1 and L2 regularization will be combined for a standard between 0 and 1. When tuning an elastic net, it is best to try different values and choose the one that provides the greatest performance on the validation set. Some of the cases, a ratio closer to 1 may work best, while in other cases a ratio closer to 0 may be more appropriate.
- **Selection:** The "selection" parameter in ElasticNet controls the type of optimization method used to regularize the model. ElasticNet uses coordinate descent method to optimize the model's parameters. The selection parameter can take two values: "cyclic" or "random"."Cyclic" selection method uses the cyclical coordinate descent optimization method. This method cycles through the feature space, updating the coefficients of the features at each iteration. This approach is generally considered to be faster than the "random" selection method, but it may not be as accurate."Random" selection method uses the random coordinate descent optimization method. This method updates the coefficients of the features randomly at each iteration. This approach is generally considered to be slower than the "cyclic" selection method, but it may be

more accurate.However, the ideal value of the selection parameter will depend on the data and the issue being addressed. In some cases, "cyclic" may work best, while in other cases "random" may be more appropriate.

3.4.5 Random Forest

In order to tune a Random Forest for a specific task, we will need to adjust several parameters. The main hyper-parameters that can be adjusted in order to fine-tune a Random Forest are:

- **Maximum depth of the tree:** The decision tree's maximum depth, which refers to the number of splits or layers. A larger maximum depth will result in a more complex model and a higher risk of overfitting, while a smaller maximum depth will result in a simpler model and a lower risk of overfitting.
- **Minimum samples per leaf:** The minimum amount of samples that are needed to make a leaf.This parameter is used to control the complexity of the tree and to prevent overfitting, by requiring a minimum number of samples at each leaf node. Increasing this value will result in smaller leaves and a simpler model, while decreasing this value will result in larger leaves and a more complex model.
- **Minimum samples per split:** The least amount of samples needed to split an internal node. It is used to regulate the complexity of the tree and avoid overfitting by requiring a minimum sample size at each internal node prior to splitting. A simpler model and a lower risk of overfitting will come from increasing this value, while a more complicated model and a higher risk of overfitting will follow from reducing this value.
- **Maximum features:** Maximum amount of features analyzed while searching for the appropriate split.These parameter is used to regulate the tree's complexity and minimize overfitting by limiting the amount of features examined while splitting the tree.Increasing this value will result in a more complex model, while decreasing this value will result in a simpler model.
- **Estimators:** The estimator's parameter determines the number of trees utilized in the model.More trees will result in a more complex model that can capture more complex relationships in the data, but it will also be more computationally expensive and may lead to overfitting. Fewer trees in the model will result in a simpler model that may be less prone to overfitting, but it may not capture all the relevant relationships in the data.

3.4.6 XGBoost

The main hyper-parameters that can be adjusted in order to fine-tune XGBoost are:

- **Learning rate:** The learning rate also known as "eta" in XGBoost, controls the step-size at which the algorithm moves in the direction of the negative gradient.A higher learning rate indicates the algorithm will take bigger steps and converge faster, whereas a lower learning rate means it will take smaller steps and will require more iterations to converge.Lower learning rates provide more accurate models but take longer to train.A higher learning rate will provide a less accurate model, but take faster to train.In some cases, a small value of learning rate may work best, while in other cases a large value of learning rate may be more appropriate.

- **Maximum depth of the tree:** The decision tree's maximum depth, which indicates the number of splits or layers.Trees with a large depth will have more levels, which means that they will maintain more complex relationships in the data and will have more splits. Trees with a small depth will have fewer levels and will maintain less complex relationships in the data.A bigger maximum depth creates a more complicated model with a higher risk of overfitting, whereas a smaller maximum depth creates a simpler model with a lower risk of overfitting.Depending on the situation, a small or large value of maximum depth may work well.
- **Estimators:** The "estimators" parameter regulates the amount of trees utilized by the model in XGBoost (eXtreme Gradient Boosting).More trees will result in a more complex model that can capture more complex relationships in the data, but it will also be more computationally expensive and may lead to overfitting. Fewer trees in the model will result in a simpler model that may be less prone to overfitting, but it may not capture all the relevant relationships in the data.Trying a variety of estimators and choosing the one that performs best on the validation set is a familiar approach.A small value of estimators may work best in some cases, while in other cases, a large value of estimators may be more appropriate.

3.5 Deep Neural Network

Deep neural networks (DNNs) are a type of neural network architecture that are characterized by having multiple layers, often more than 10 layers. They are also called deep learning networks. These layers are typically composed of multiple artificial neurons, which are connected and organized in a way that allows the network to learn a hierarchy of features from the input data.Deep neural networks are different from traditional neural networks, also called shallow neural networks, in a number of ways. The most important difference is the number of layers, as deep neural networks have many more layers than traditional neural networks. Additionally, deep neural networks are often trained using unsupervised learning techniques, such as autoencoders and restricted Boltzmann machines, which allows them to learn a hierarchical representation of the data.The benefits of using deep neural networks include their ability to learn complex, non-linear relationships in data, their ability to handle large and high-dimensional data, and their ability to automatically learn useful features from the data without the need for manual feature engineering. They are widely used in many fields such as computer vision, natural language processing, and speech recognition.DNNs have also shown remarkable performance in many applications such as image classification, object detection, speech recognition, natural language processing and many more. They are also the backbone of many state-of-the-art models in various fields.

3.5.1 Convolutional Neural Network

Convolutional neural networks, or CNNs, are a specialized type of neural network architecture that are particularly effective in analyzing and understanding visual data. Rather than relying on the traditional, fully connected layers found in other neural network architectures, CNNs use convolutional layers that scan the input data with a set of filters. These filters are designed to extract features from the input, such as edges or textures, and are trained to be sensitive to specific patterns in the data. The extracted features are then passed through additional layers, like pooling, which reduce dimensionality and make the network more robust to small distortions. Ultimately, these features are used to make predictions through fully connected layers. CNNs have been successfully applied in a variety of tasks, including image and object recognition, and semantic segmentation. They have proven to be particularly powerful in analyzing visual data, making them a popular choice for many computer vision tasks.

The Convolutional Layer

Before the implementation of convolutional layers, it was therefore impossible to train and process real-world image data with neural networks. As previously stated, it is a preprocessing front-end of a neural network that helps reduce the high dimensionality of a pixel-based image to a lower dimensionality, feature-based representation of that image. The latter can then be loaded as the input vector to a Deep Neural Network for additional processing and learning.

Translational Variance

Another issue with image processing based on its pixel representation is a translational variation or image recognition from various perspectives. Pixels are highly dependent on their position within an image in order to display the correct image. However, for computer vision, this restricts the capacity for learning because any model that attempts to train on these pixel representations would overlook the position of the image's objects [43]. It would be more difficult for the machine to recognize a jumping and rolling cat if it could only train on images of that cat lying in a specific location. Training on each and every possible cat position in an image is also an impossible task to accomplish.

Feature Detection

For these difficult-to-process pixel-based images, it is preferable not to implement recognition through pixel positions, but rather by identifying and detecting the image's features, shapes, and objects. The methodology simulates how humans identify objects in the dark: from a distance, we can only recognize the shape of an object (whether it has rounded or sharp edges); as we move closer, we can begin to identify its profile and shape; and finally, our eyes determine what the object is by drawing a logical conclusion from the previously identified features. Similarly, a convolutional layer detects image features by employing a set of filters - n-by-m size matrices with values designed to identify specific image features, such as edges and shapes. These filters perform a sliding window technique across the image's pixels, selecting pixel values that indicate the probable presence of a feature at a given location and storing them in a feature map. This map would therefore contain information on all the features within the image. As opposed to processing the entirety of an image's pixels, we now only need to process the feature map as the input sequence, which is significantly smaller than the former.

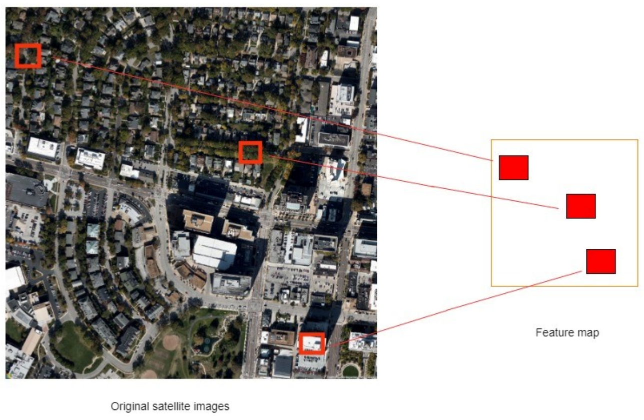

Figure 3.9: Feature Map created by sliding window of filters across the satellite image.

In the convolutional layer, we can specify the filter size, the number of filters and the distance travelled by each filter as it slides (known as strides) across the image. By selecting an appropriate filter size (typically 3x3 or 5x5), we can transform the image into a representation with fewer dimensions. Nevertheless, each filter that traverses an image generates its own feature map that captures a particular characteristic of the objects within that image. This implies that with larger filters and a higher stride, we can have less data, but the more filters we can learn from an image, the more feature maps (and thus data) we would still need to process. The utilization of pooling layers eliminates this additional issue.

Pooling

Multiple feature maps result in increased data volume. This, however, can be managed through pooling, which is analogous to down-sampling. Using this method, we can set a pooling layer and its stride across each individual feature map, similar to how a filter is applied to the initial image. For a 2x2 pool size with a stride of 2, we can reduce approximately 75 percent of the pixel data while preserving the detected features and their spatial inter-dependencies [43]. Two distinct types of operations must be specified in order to condense the feature information captured in the feature maps within the pooling layers:

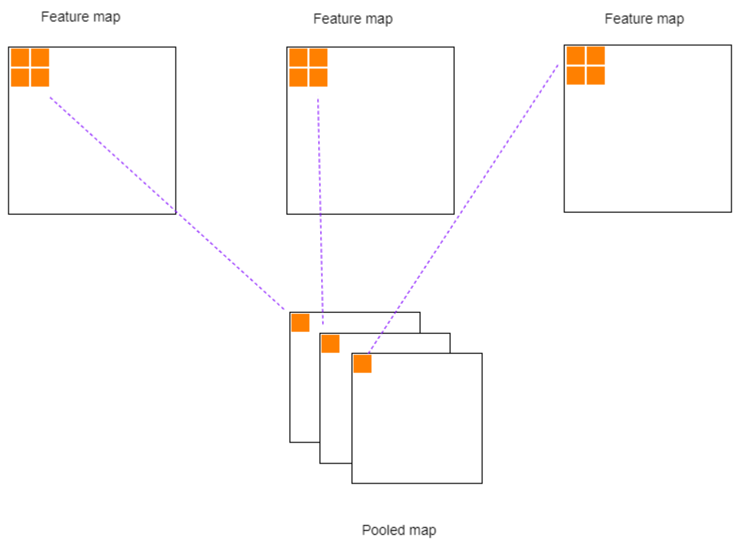

Figure 3.10: Pooled map created from feature maps

1. **Max Pooling:** Retrieves the highest values for each pooling stride patch on the individual feature map;
2. **Average Pooling:** Calculates the mean of all values for every path of the pooling stride on the individual feature map.

The max pooling operation discards the majority of information in order to obtain only the most significant features in the input space, such as edges, whereas the average pooling operation takes information from all features present in the input. The maximum pooled layer will theoretically contain images with high contrast, whereas the average pooled layer will contain smoother and less grainy images.
The pooling layers also reduce the sensitivity of the feature maps to the input's feature location. This is accomplished through the process of down-sampling with pooling: the higher resolution input is reduced to a lower resolution representation, and as a result, the refined features of the upstream process are transformed into a less detailed and more abstract version that retains the structural elements. As a result, the lower representation of the features can rely less on the feature's position in the input because it no longer needs to cast detailed and granular information on the feature. Pooling is a fundamental component of the architecture that summarizes the detected features in the feature maps.

Flattening

The final step of the front-end process of the Convolutional layer is transforming the information obtained from the feature map and the pooling operation (2D matrices of the pooled feature maps) into a 1D vector that can be used as training and learning input for the Deep Neural Network. Each pooled map is appended sequentially to the next in order to maintain the spatial relationship present in the pooled information, thereby forming the long 1D vector for the input space.

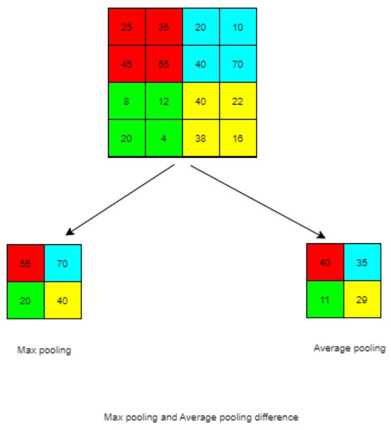

Figure 3.6: Difference between max pooling and average pooling

3.5.2 CNN Architectures

Some of the most well-known and widely used CNN architectures include VGG16 and VGG19, developed by the Visual Geometry Group at the University of Oxford in 2014, ResNet developed by Microsoft Research in 2015, InceptionNet developed by Google in 2014, and DenseNet developed by Gao Huang and his team at the Chinese University of Hong Kong in 2017. These architectures have been widely adopted in various applications such as image classification, object detection, and semantic segmentation. VGG is known for its depth and use of small convolutional filters, ResNet for introducing residual connections to improve training of deeper networks, InceptionNet for its use of Inception modules that effectively handle multi-scale information, and DenseNet for its dense connectivity pattern between layers which improves the flow of gradients. These architectures have proven to be powerful and efficient in analyzing visual data and have become the backbone of many state-of-the-art models in computer vision tasks.

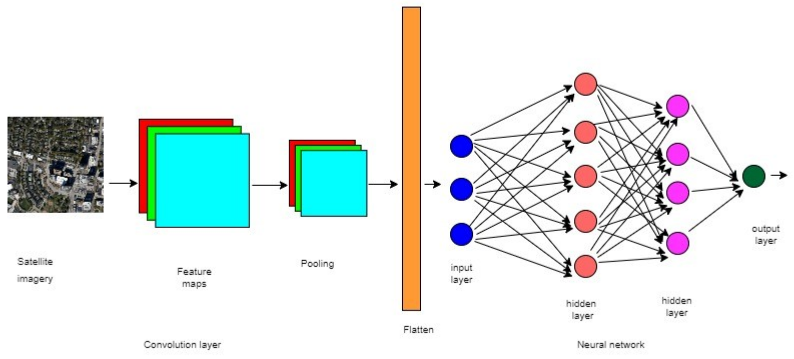

Figure 3.7: Graphical model of Convolutional Neural Network Architecture

VGG16

Oxford's Visual Geometry Group has created the VGG16 Convolutional Neural Network model. K. Simonyan and A. Zisserman introduced Very Deep Convolutional Networks for Large-Scale Image Recognition in their study [16]. The model was created to compete in the ILSVRC-2014 image recognition competition.In ImageNet, a dataset with 14 million images divided into 1000 classes, it attained an accuracy of 92.7 percent[46].In order to reduce the size of the feature maps, the model is composed of convolutional layer stacks along with a max pooling layer.The design directs the learning of the coarse features of the images in the prior layers, while the subsequent layers could learn the fine-grain features by increasing the number of filters with max-pooling inserted in between to prevent the data and parameters transferred between these layers from growing exponentially.The Deep Neural Network component is made up of two layers which are hidden with 4096 nodes each, along with a neural output layer with 1000 class-specific nodes.

VGG19

VGG19 is a convolutional neural network(CNN) model trained on the "ImageNet" dataset for image classification.Karen Simonyan and Andrew Zisserman introduced it in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition." [49].This model form with convolutional and max pooling layers, followed by a few fully connected layers.The convolutional layers extract features from the input images, while the fully connected layers are used for classification.VGG19 is known for its use of very small (3x3) convolutional filters and a very deep network with 19 weight layers.Each convolutional layer contains a huge amount of filters, that allows a huge amount of features from the input image. It makes the model so strong, but also very computationally expensive, requiring a lot of memory and processing power.VGG19 is frequently used like a baseline model for activities like image classification and has won many contests, including the ImageNet Large Scale Visual Recognition Challenge which is also known as ILSVRC in short[48]. It is a good choice for tasks where a high level of accuracy is required, but the trade-off is a longer training time and a larger model size.VGG19 may not be the best choice for tasks that require real-time processing or have limited computational resources, as it is a relatively large and resource-intensive model. It may also not be suitable for tasks where the input images have a large scale variance or are significantly rotated, as VGG19 was trained on relatively centered and normalized images.

Resnet

ResNet (Residual Network) is a convolutional neural network (CNN) architecture which was introduce to improve the performance of deep neural networks on image classification tasks. It is based on the concept of "residual learning", which involves training a network for learning the residual function between the input and the desired output, in place of trying to predict the output instantly from the input.There are two types of ResNets.1)ResNet with identity connections consists of a series of layers, each of which has an "identity connection" that bypasses the layer and directly connects the input to the output. This allows the network to learn the residual function between the input and the output, rather than having to learn the input-output mapping directly.2)ResNet with bottleneck connections is similar to the ResNet with identity connections, but it uses "bottleneck" layers to reduce the parameter numbers in the network [54]. A bottleneck layer has a smaller number of filters than the preceding and following layers, which reduces the number of parameters in the network.ResNets are used for a wide range of tasks, including image classification, object detection, and semantic segmentation [44]. They are particularly useful for tasks that require the network to learn for recognizing patterns in high-dimensional data, like images[44].There are some situations in which ResNets may not be the best choice for a particular task. For example, if the input data is very small (e.g., a few pixels), a ResNet may be too complex and may not be able to learn the necessary patterns effectively. In such cases, a simpler network architecture may be more appropriate.For our work in this project we used Resnet152 and ResNet50 to train and test on the satellite imageries. ResNet50 is a specific variant of the ResNet architecture that was introduced by Kaiming He Et al(2016).[24] The number "50" in the name indicates the number of network levels. ResNet50 employs bottleneck blocks comprised of 1x1 convolutional layers, 3x3 convolutional layers, and 1x1 convolutional layers. These blocks help reduce the amount of network parameters, hence increasing the network's efficiency.The output layer is a global average pooling layer that generates a vector of features that can be employed for classification or other purposes. ResNet50 is a shallower variant of ResNet than ResNet152; it has fewer layers and fewer parameters, was trained on a large dataset (ImageNet), and achieved a good performance on an image classification task. While, ResNet152 is a deeper variant of ResNet, it has more layers and more parameters. Both of these is used in a variety of applications, including object detection, semantic segmentation, and so on.

Inception

Inception is a machine learning (ML) technique that refers to the process of training a neural network on a dataset and subsequently using the trained network to generate additional data that can be used to train the network further[51]. This process is often used to improve the accuracy and effectiveness of the network by allowing it to learn from a larger, more diverse dataset.Inception is typically used in the field of computer vision, where it can be applied to tasks such as image classification, object detection, and segmentation [29]. It can also be used in natural language processing (NLP) tasks such as language translation and text generation [53].One of the main benefits of using Inception is that it allows the network to learn more about the data it is processing, which can lead to better performance on a variety of tasks. However, Inception is not always applicable, and it may not be the best approach for certain types of data or tasks.For example, if the dataset is very small, there may not be enough data to effectively train the network using the Inception method. In this case, it may be more effective to simply train the network on the available data, rather than trying to generate additional data using Inception.Similarly, if the task is relatively simple and does not require a complex model to solve, the added complexity of using Inception may not be justified. In these cases, it may be more efficient to use a simpler approach to training the network.
A specific variant of the Inception architecture that was introduced by Christian Szegedy Et al(2016) [29], the InceptionV3 was used in our work.The Inception architecture is distinguished by its use of Inception modules, which are layers that

compute multi-scale convolutional features. Inception v3 builds upon the original Inception architecture by introducing a number of enhancements and alterations, such as:

1. **Factorization of convolution layer:** Inception v3 employs factorized convolutions to reduce the computational cost and number of network parameters.
2. **Batch Normalization:** Inception v3 utilizes batch normalization, which improves the training's stability and speed.
3. **Auxiliary classification:** Inception v3 employs auxiliary classifiers, which are additional classifiers added to the network's intermediate layers. These classifiers contribute to the enhancement of the network's overall performance.
4. **Asymmetric Multi-Scale:** Inception v3 uses a different scaling method on the input image referred to as "Asymmetric Multi-Scale," which improves the network's accuracy.

The ImageNet dataset is used to train the Inception v3 architecture, which achieves state-of-the-art performance on image classification tasks.

DenseNet

DenseNet is a type of convolutional neural network (CNN) architecture renowned for its ability to improve the flow of information and gradients throughout the network, thereby mitigating the problem of vanishing gradients that can occur in deep networks. DenseNet's architecture consists of the following key components:

- **Dense Blocks:** These are the network's building blocks, and they contain multiple convolutional filter layers. Each layer in a dense block is connected to each other layer in the block, and each layer's output is concatenated with the inputs of all subsequent layers. This allows for a more efficient flow of information and gradients, as each layer can access the features learned by all previous layers in the block.
- **Transition Layers:** These layers are used to reduce the number of network filters and limit the growth of network parameters. They consist of a 1x1 convolutional layer, which decreases the number of filters, and a 2x2 average pooling layer, which decreases the spatial dimensions of the feature maps.
- **BottleNeck Layer:** This layer is a combination of 1x1 and 3x3 convolutional filters with the objective of reducing the number of parameters while preserving the essential characteristics.
- **Output Layer:** The output layer is a global average pooling layer that generates a vector of features that can be used for classification or other purposes.

DenseNet also uses skip connections, which let data move from one layer to the next without having to be joined together.
Out of several versions and models of DenseNet available, DenseNet121 was used to find the optimal target value of any location in our work. DenseNet121 is a specific variant of the DenseNet architecture that was introduced in the paper "Densely Connected Convolutional Networks" by Gao Huang Et al. [32] It is a deeper variant

of DenseNet, it has more layers and more parameters, and it was trained on a large dataset (ImageNet) and achieved a good performance on an image classification task. It is also used in many other applications, such as object detection, semantic segmentation, and so on.

3.5.3 Transfer Learning

A model that has been trained on one task can be used as a starting point to train another model on a different but related task. This process is known as transfer learning. Transfer learning's main benefit is that it enables us to use previously acquired knowledge to enhance performance on new tasks. Depending on the particular problem at hand and the model architecture in use, this can be accomplished in a number of different ways. Some common techniques include fine-tuning a pre-trained model, using a pre-trained model as a feature extractor, and training a model on multiple related tasks.Transfer learning can significantly reduce the amount of data and computational resources needed to train a new model, which is one of its main benefits. A model can be trained on a large dataset of images and then fine-tuned on a smaller dataset of images specific to a new task where there may be a limited amount of data available for a particular task. In terms of optimal site selection, transfer learning can be used to improve the accuracy of a model that predicts the suitability of a given location for a specific use, such as building a new store or factory. For example, if a model has been trained to predict the suitability of locations for building a new retail store, it can be fine-tuned using data on the characteristics of locations that are suitable for building a new factory. This can allow the model to make more accurate predictions about the suitability of new locations for building a factory, as it is able to use the knowledge learned from the retail store task to inform its predictions.

3.6 Research Workflow

We choose restaurant check-in as the variable that will decide the location's optimality. Therefore, our initial goal was to collect restaurant location and check-in information. This information was obtained from the Yelp dataset. After obtaining the Yelp dataset, the data was filtered, and only restaurants with more than 50 check-ins in 2021 and 2022 were assembled.To begin the collection of the feature dataset, a 5 km buffer area was selected. The overpass API was then used to extract OSM data on relevant elements, such as hospitals, entertainment, sustenance, roads, education, etc., that were within a 5 km radius of each restaurant in our target dataset. To acquire the population dataset, a geotiff file containing a population count per kilometer based on 2020 census data modified to UNDP population statistics was retrieved from the "Humanitarian Data Index." Before the tiff file could be converted to csv, it had to be converted to the XYZ file format. Using latitude, longitude, from the csv file of the population, compound indexing was developed in sqlite for faster queries. The Haversine formula was used to compute coordinates that lie within the buffer area of the restaurants. The population count of each of these coordinates was added together to get the total population count of each restaurant's buffer area, which was then combined into a .csv file. The demographic and feature data were integrated into a single CSV file. Then, we preprocessed the data by scaling and removing outliers using z-score. The SelectKBest algorithm was used to apply weights to the features to determine if it enhanced model performance. Our dataset was split 8:2 for training and testing purposes. In order to execute the regression models, the parameters have to be optimized for optimal results. Each model required a unique set of alterations and adjustments to its parameters. For the neural network architecture, the number of hidden layers, number of neurons, optimizer, learning rate, loss function, early stopping, and epochs were to be determined. Our machine learning models and neural networks were then trained and evaluated. Additionally, we constructed a meta-model based on the stacking ensemble technique that comprised various ML models.

Implementation of CNN architectures with satellite imagery was another important aspect of our paper. Using the Bing Maps API, we initially retrieved satellite images of our target restaurants. The acquired png images were preprocessed with the VGG16 image preprocessor, and each pixel was scaled from 0 to 1. The photos were ultimately renamed based on the target value of each restaurant. In order to deploy the CNN architecture, we divided the data into train and test datasets in the proportion of 7:3. In addition, training data were separated into training and validation sets to prevent overfitting by monitoring validation loss and adopting early stopping. We also defined the image's dimension and batch size as additional parameters. Since we had to perform regression, for CNN architectures such as VGG16, VGG19, DenseNet121, Resnet50, Resnet152, and InceptionV3, we added a dense layer with one neuron. After evaluating the architectures and determining that DenseNet121 performed the best, we retained only the convolution layer and removed the top layer containing 1000 neurons. The layer was then flattened, and a batch normalization layer, a dropout layer, and a dense layer were added. Tuning was performed on parameters including the optimizer, learning rate, loss function, early stopping, and number of epochs. Before and after transfer learning, the performance of the model was evaluated and compared to the initial architecture. Our data revealed that, among the three possible configurations, the modified architecture performed better than the original architecture. All error and precision measurement metrics were recorded in order to select the most effective model and examine the feature weights. In the end, we were able to identify the most effective model for optimal business site selection and the most influential factors affecting the outcome. A flowchart to better visualize the work flow can be observed in figure 3.8

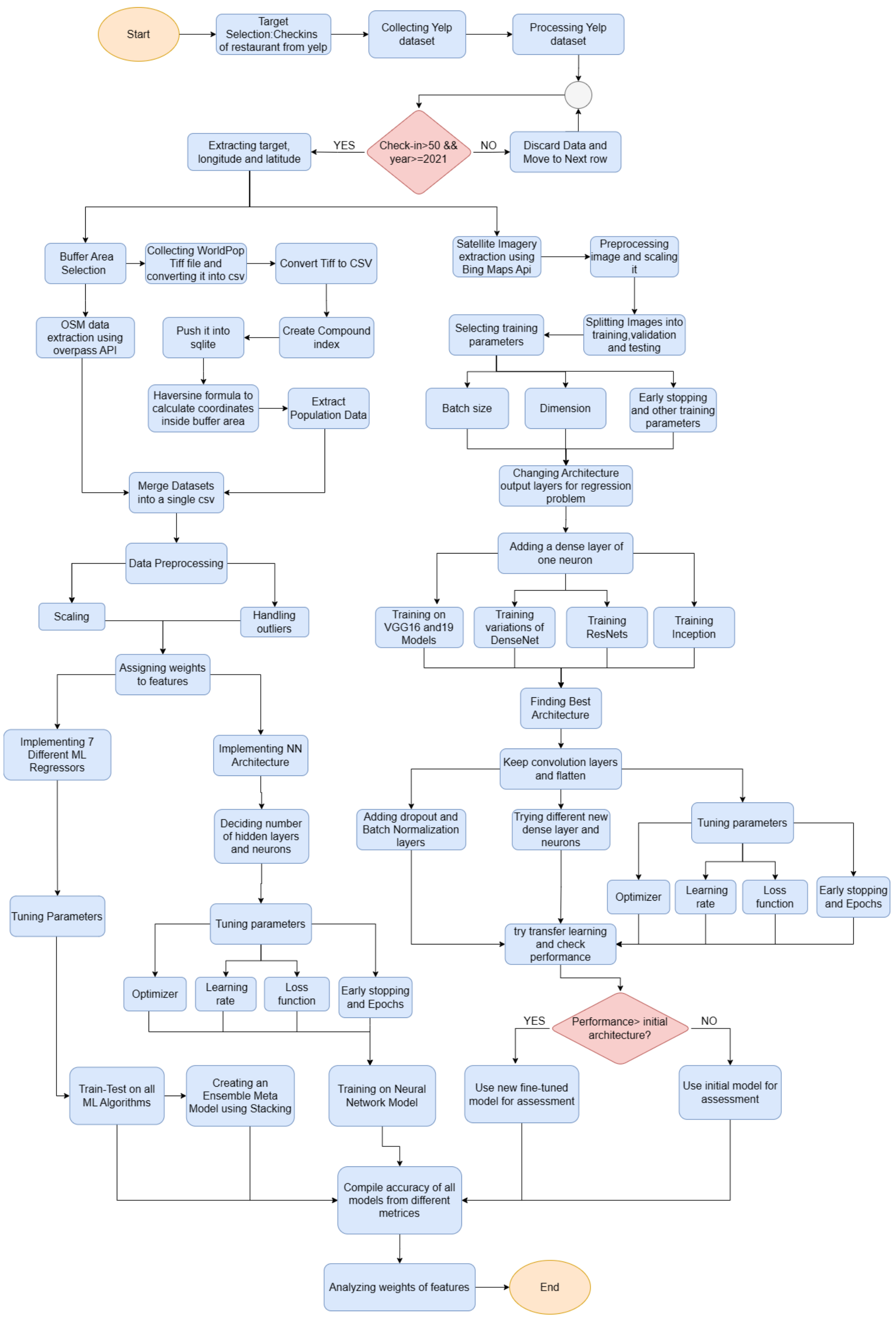
Start
Target Selection:Checkins of restaurant from yelp
Collecting Yelp dataset
Processing Yelp dataset
Check-in>50 && year>=2021
YES
NO
Extracting target, longitude and latitude
Discard Data and Move to Next row
Buffer Area Selection
Collecting WorldPop Tiff file and converting it into csv
Convert Tiff to CSV
Satellite Imagery extraction using Bing Maps Api
Preprocessing image and scaling it
OSM data extraction using overpass API
Push it into sqlite
Create Compound index
Selecting training parameters
Splitting Images into training,validation and testing
Haversine formula to calculate coordinates inside buffer area
Extract Population Data
Batch size
Dimension
Early stopping and other training parameters
Merge Datasets into a single csv
Changing Architecture output layers for regression problem
Data Preprocessing
Adding a dense layer of one neuron
Scaling
Handling outliers
Training on VGG16 and19 Models
Training variations of DenseNet
Training ResNets
Training Inception
Assigning weights to features
Finding Best Architecture
Implementing 7 Different ML Regressors
Implementing NN Architecture
Keep convolution layers and flatten
Deciding number of hidden layers and neurons
Adding dropout and Batch Normalization layers
Trying different new dense layer and neurons
Tuning parameters
Optimizer
Learning rate
Loss function
Early stopping and Epochs
Tuning Parameters
Tuning parameters
try transfer learning and check performance
Optimizer
Learning rate
Loss function
Early stopping and Epochs
Performance> initial architecture?
YES
NO
Train-Test on all ML Algorithms
Creating an Ensemble Meta Model using Stacking
Training on Neural Network Model
Use new fine-tuned model for assessment
Use initial model for assessment
Compile accuracy of all models from different metrices
Analyzing weights of features
End

Figure 3.8: Research Workflow

Chapter 4

Data Collection and Processing

4.1 Data Source

As a source, we used the dataset of yelp, Satellite Imagery from Bing Maps, OpenStreetMap (OSM), and Worldpop Population count. We used the restaurant data from the yelp dataset as our use case. This target is used as the label, and the rest of the data were used as features. We decided to use these sources because most of them are open-source with a good API, so it's easier to retrieve data without any cost.

Source	**Type of data**		**Dataset Size**
	Raw Data	**After Processing**	
Yelp Data	CSV	CSV	8.65GB
OSM Data	CSV	CSV	200KB
Bing Map Data	CSV	CSV	20.4MB
WorldPop Data	GEOTIFF	CSV	3.7GB

Table 4.1: Data Sources, Type and Size.

4.1.1 Yelp Dataset

We drew information for the restaurant data from the Yelp dataset [59]. Yelp is a well-known platform for businesses to post ratings and reviews; therefore, it is a dependable source of information for restaurants. The Yelp dataset can provide a large and diverse collection of data on any existing business type, including restaurants. The six sub-datums of the Yelp dataset are reviews, check-ins, businesses, tips, users, and photos. More than 1.5 million pieces of business-related information, star ratings, and check-ins were collected from the business, reviews, and check-ins sub-datasets. We utilized the latitude, longitude, and check-in information provided by Yelp for various restaurants in our work.This check-in information served as our dependent variable as we sought to predict the optimal location for a new restaurant based on check-in data from existing restaurants. The latitude and longitude helped us locate the restaurant and its surroundings and find necessary information from other datasets that were used as features. This data can be used to guide site selection and identify areas where similar restaurants are likely to be successful. It is essential to note that the authenticity of the Yelp dataset may vary, as some businesses may have multiple fake reviews or ratings posted. However, we do not utilize Yelp's review or rating data in our work. We employed various data-cleaning techniques, such as selecting restaurants that have at least 50 check-ins from 2021 and removing outliers, as well as multiple datasets from other sources to cross-verify the information. The Yelp dataset can also provide data on the competition and

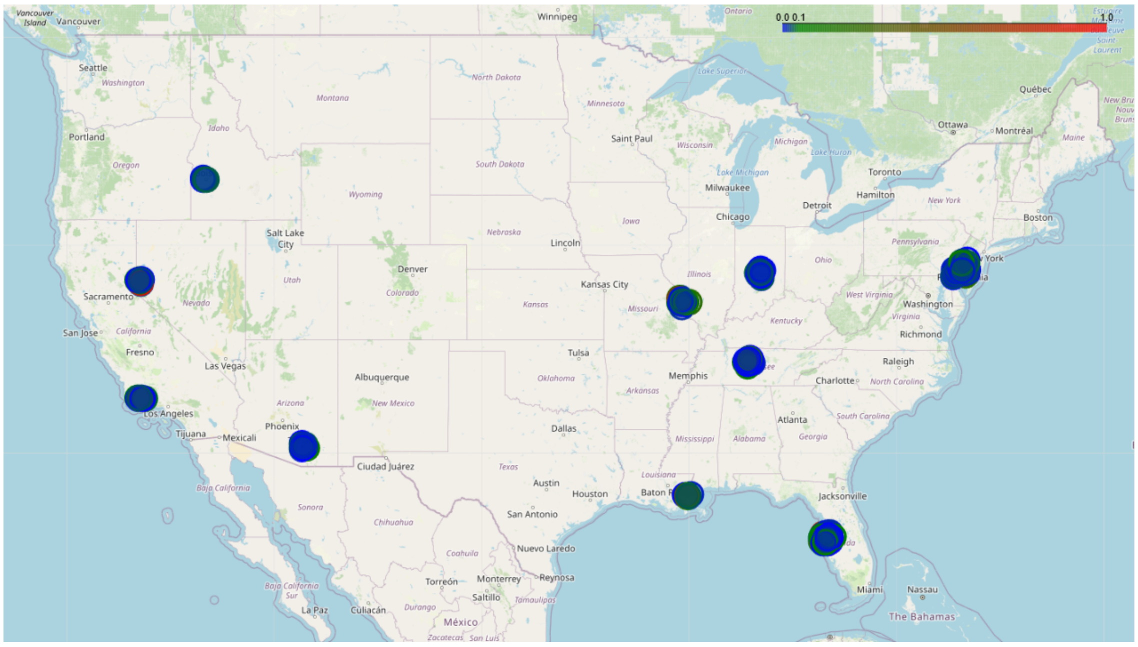

Figure 4.1: Restaurant Check-in data plotted over USA Map

density of restaurants in a certain area. This can be used as a factor to determine where to open up a new restaurant by avoiding over-saturated areas or looking for potential gaps in the market. The dataset can also be used to analyze the geographic distribution of restaurants, which can provide information about the concentration of different types of restaurants in different areas. This can be used to identify areas with a high demand for certain types of restaurants and to target specific demographics or customer groups.

4.1.2 Bing Maps

The satellite imagery data for this research was collected using the Bing Maps API, which is a widely used and reliable service provided by Microsoft [57]. The images were specifically captured with the aim of analyzing the potential site for restaurants. 1108 images were collected, with each image centered on a specific restaurant location using the latitude and longitude provided. The images were captured at a zoom level of 15, providing a pixel resolution of 4.78 meters and an image size of $224X224X3$. This high resolution allows for a detailed analysis of the features present in the images such as the surrounding area, foot traffic, and competition. Additionally, the image size of $224X224X3$ was selected after careful consideration, as it matches the input layer dimensions of some well-known CNN architectures and also provides a balance between computational efficiency and preserving the details of the image. Each image covers a total area of 1146441.3184 m^2, which is sufficient to capture the necessary information for the research. The collected data and images were then used to train a CNN model to predict the target value of the optimal site for the restaurants. This approach will enable the identification of the most suitable location for opening new restaurants by analyzing the attributes of the surrounding area.

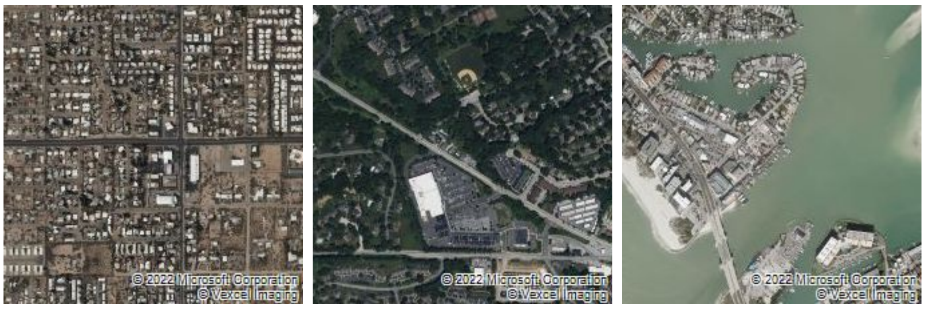
© 2022 Microsoft Corporation
© Vexcel Imaging
© 2022 Microsoft Corporation
© Vexcel Imaging
© 2022 Microsoft Corporation
© Vexcel Imaging

Figure 4.2: Bing Satellite Imagery.

4.1.3 Open Street Map Data(OSM)

Despite its relative accessibility, Google Static Map is still a proprietary enterprise product, and acquiring images incurs a cost [56]. Even though this price is reasonable compared to the cost of a national population-representative survey, it still creates a barrier to information access and a delay in analysis (Google places a limit on Static Map image requests per day, which for a country of substantial land mass would create a delay). Therefore, an additional data source from Open Street Map for the United States' mapping of geographical features has been adopted. As a crowd-sourced community, Open Street Map (OSM) offers a free alternative to Google Static Image that provides geo-mapping of natural and man-made features in the national landscape of the user's choice. This would improve the predictive accuracy of models constructed using these resources. To gain a comprehensive understanding of the areas surrounding popular restaurants, we harness the vast wealth of feature data available on Open Street Map through the Overpass API. Our method involved utilizing the coordinates of restaurants found in the Yelp dataset as a starting point and then using the Overpass API to extract a wealth of information about a 5-kilometer radius surrounding these locations. These details included, allowed us to gain a detailed and nuanced understanding of the areas in question, uncovering valuable insights and information.

4.1.4 Population Count Dataset

The dataset containing the population count per kilometer was collected from the "Humanitarian Data Exchange" service [58]. The dataset collected was prepared by Worldpop. This organization produces several datasets and geomaps these data by the use of different methods and end applications. We used the population count per kilometer that was adjusted to match the corresponding UNDP estimates out of the several sets of population data available. The dataset from these sources has been used in several published and acclaimed works, and its authenticity is highly regarded. For our research, we used the data in XYZ format and then it was converted into CSV format for putting it in SQLite database. In simple terms, XYZ format data are csv files with only three columns of values. The first two columns represent latitude and longitude, respectively, while the third column may represent the data from the GeoTIFF file, which in this case is the population count within a 1km radius of the coordinate. The visualization of our data is as follows;

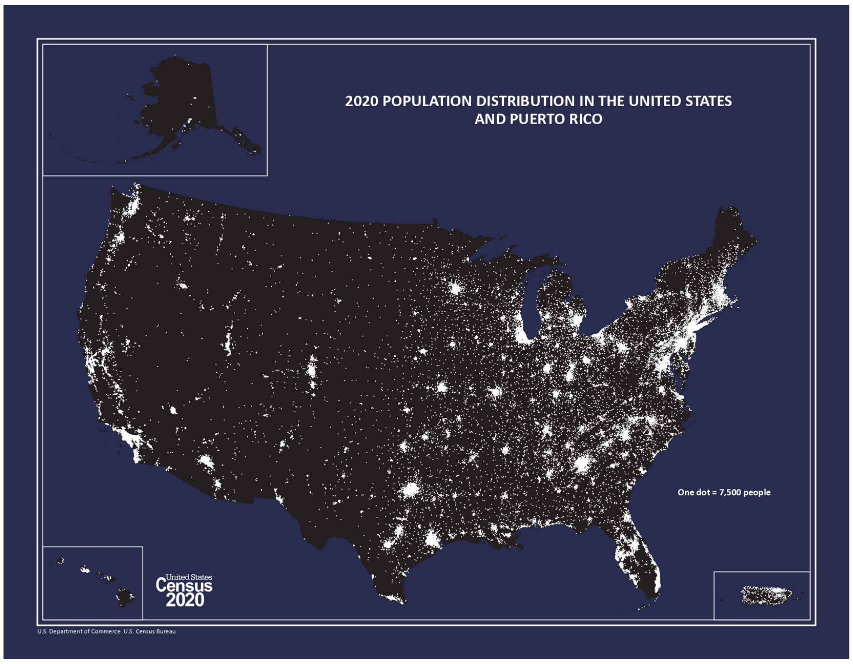
2020 POPULATION DISTRIBUTION IN THE UNITED STATES AND PUERTO RICO
One dot = 7,500 people
United States Census 2020
U.S. Department of Commerce U.S. Census Bureau

Figure 4.3: Visualization of Population count per KM [55]

4.2 Data Extraction

4.2.1 Buffer Area

A buffer area, also known as a buffer zone or a buffer, is a designated area around a specific feature or location that serves as a protective or separation zone. In the context of our thesis, the buffer area is used to identify potential restaurant sites by taking into account the proximity and accessibility of the location to certain amenities such as roads, public transportation, and population density. The buffer area is determined by setting a certain distance from the target location and creating a zone within that distance. This zone is then analyzed to identify the presence of certain amenities and to determine the suitability of the location for a restaurant. The buffer area allows us to take into account the surrounding environment and factors that may influence the success of a restaurant in a particular location. To gather relevant features such as roads, buildings, and other infrastructure, we chose to use a buffer area of 5km around each restaurant. This choice of buffer area was based on several considerations, such as the average distance that customers are willing to travel to reach a restaurant and the potential competition within the area. In our research, we used OpenStreetMap (OSM) data to gather the features within the buffer area of 5km. Using the latitude and longitude of each restaurant, we were able to extract the relevant information from OSM and use it to train a CNN model to predict the target value of the optimal site for the restaurant. This approach allowed us to analyze a comprehensive set of features within a defined distance from each restaurant, providing a more accurate assessment of the potential success of a new restaurant in that location. The choice of a 5km buffer area for gathering features in our research was based on the consideration that it represents an appropriate distance for assessing the potential customer base and competition for a new restaurant.

4.2.2 Yelp Data Extraction

Using the pandas library's read json function, the two files "yelp academic dataset business.json" and "yelp academic dataset checkin.json" are read to begin the data extraction process from the Yelp dataset. The first file, "yelp academic dataset business.json", contains business information such as the business's ID, name, address, city, state, zip code, latitude, longitude, stars, review count, whether the business is open or closed, attributes, categories, and operating hours. There are 150,346 rows in this file, which represents the total number of businesses. The second file, "yelp academic dataset checkin.json", has 131,930 rows and contains information such as the business ID, check-in date, and check-in count. Noting that the number of check-in rows does not equal the number of businesses indicates that some businesses lack check-in data.

Using a custom function named "count dates," the next step is to convert the dates of check-ins to the number of check-ins. This function counts the number of dates separated by commas in the "date" column to determine the number of check-ins. The result is stored in a column titled "number of check-ins."

The two dataframes are then merged together based on the "business id" column. This allows us to link the check-in data with each business's business information. Next, we remove from the merged dataframe columns such as "business id", "address", "city", "state", "postal code", "attributes", and "hours" because they are not required for the analysis.

The merged dataframe is then subjected to data cleansing and filtering. Using the "is open" column, we eliminate all businesses that are closed. Additionally, we eliminate businesses that lack check-ins in 2021 and 2022. This is accomplished by counting the number of "2021" and "2022" occurrences in the "check-ins" column and storing the count in a new column titled "check-ins after 2021."

The dataframe is then filtered to include only rows for which the "categories" column contains the string "Restaurants" and the "checkins after 2021" column has a value greater than 50. This is completed with a boolean mask, str.contains(), and gt().

Finally, the resulting dataframe is reset with an index, and columns such as "check-ins", "number of checkins", and "index" are removed because they are deemed unnecessary. The resultant dataframe contains business information and check-in data for restaurants that are currently open, have check-ins in 2021 and 2022, and have more than 50 check-ins in the specified years. This dataframe can then be used for further analysis and modeling in the optimal restaurant site selection problem.

It is reasonable to include only restaurants with more than 50 check-ins in 2021 and 2022, as this ensures that the businesses included in the analysis are active and popular. Restaurants with a small number of check-ins may not be suitable for the optimal site selection problem because they may not attract enough customers to be considered successful.

Assuring that the businesses included in the analysis are operational and eligible for consideration in the optimal site selection problem, it is also fair to consider only currently operating restaurants. As they are no longer in operation, closed businesses may not be appropriate for the analysis.

Name	*Latitude*	*Longitude*	*Stars*	*Review Counts*	*Categories*	*Check-ins From 2021*
Helena Avenue Bakery	34.41444	-119.691	4	389	Food, Restaurants, ..., Bakeries	129
Copper Vine	29.95065	-90.0744	4.5	350	Nightlife, Pub, ..., Event Spaces	111
Mike's Ice Cream	36.16265	-86.776	4.5	593	Ice cream & Frozen Yogurt, ..., Food	53
Santa Barbara Shellfish Company	34.40871	-119.685	4	2404	Live/Raw Food, Restaurants, ..., Nightlife	445
The Love	39.95066	-75.1709	4	618	Restaurants ... Burgers	100

Table 4.2: Snippet of Extracted Yelp Dataset

Inclusion of only restaurants with check-ins occurring from 2021 ensures that the data used in the analysis is current and accurate. It is also important to note that we utilize OSM data extracted from the Overpass API, which is regularly maintained and updated. We also used the WorldPop population data for 2020, so check-in data from a year before might not show how popular and successful a business is right now. If we only consider check-ins from 2021, we can make more accurate predictions and decisions regarding the optimal site selection problem.

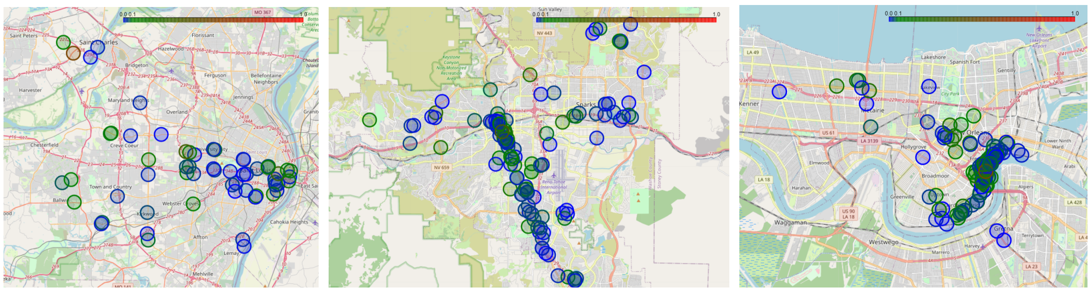

Figure 4.4: Yelp Data Plotted on Map. (Shown: Three clusters of data only)

4.2.3 OSM Data Extraction

For each restaurant coordinate collected from the yelp dataset, we extracted a lot of data within a 5km buffer area surrounding those coordinates with overpass API From Open Street Map Data. We categorized these data into 9 features. The features are hospitals, entertainment, sustenance, roads, education, facilities, building, routes, and tourism. Each feature included lots of sectors. For example, education feature included college, driving school, kindergarten, language school, library, training facility, music school, school, and University. For each coordinate's 5 km buffer area, we counted the nodes from each feature and represented the total count from all sectors as the representation of the Feature. For Hospital the sectors are daycare, dentist, doctors, nursing home, Pharmacy, veterinary, social medical facilities, clinics, and treatment facilities. For entertainment, the sectors are arts centre, brothel, casino, cinema, community centre, conference centre, fountain, events venue, gambling, theatre, studio, nightclub and social centre. For sustenance, the sectors are bar, Biergarten, cafe, fast food, ice cream, pub, and restaurant. For roads, the sectors are bicycle, bus, detour, ferry, foot, hiking, railway, road, running, train, tram, trolley, and light rail. For education, the sectors are college, driving school, kindergarten, language school, library, training, music school, school, and university. For facilities the sectors are, barbeque, bench, dog toilet, dressing room, drinking water, give box, mail-room, parcel locker, shelter, shower, telephone, toilets, and water point.

Features	**Sectors**
Hospital	Daycare, Dentist, Doctors, Nursing home, Pharmacy, Veterinary, Social Medical Facilities, Clinics
Entertainment	Arts centre, Brothel, Casino, Cinema, Community Centre, Conference Centre, Fountain, Events Venue, Gambling, Theatre, Studio, Nightclub and Social Centre
Sustenance	Bar, Biergarten, Cafe, Fast Food, Ice Cream, Pub and Restaurant
Roads	Bicycle, Bus, Detour, Ferry, Foot, Hiking, Railway, Roads, Running, Train, Tram, Trolley and Light rail
Education	College, Driving School, Kindergarten, Language School, Library, Training Facility, Music School, School and University
Facilities	Barbeque, Bench, Dog Toilet, Dressing Room, Drinking Water, Give Box, Mailroom, Parcel Locker, Shelter, Shower, Telephone, Toilets and Water Point
Buildings	Apartments, Barracks, Bungalow, Hotel, Farm, House, Residential, Tree House, Dormitory, Terrace, Semidetached House, Detached, Stilt House, Commercial, Industrial, Office, Retail, Supermarket, Warehouse, Church, Monastery, Mosque, Temple, Shrine, Transportation, Civic, Fire Station, Public, Toilets, Stadium, Hut, Shed, Garage, Parking, Storage Tank, Castle, Military, Tent, Construction and Bunker
Routes	Bus Stop, Train Stations and Metro Stations
Tourism	Picnic Site, Museum, Viewpoint, Zoo, Motel, Guest House, Gallery, Camp Site, Camp Pitch, Attraction, Artwork, and Aquarium

Table 4.3: List Sectors for Each Feature

For building, the sectors are, apartments, barracks, bungalow, hotel, farm, house, residential, tree house, dormitory, terrace, semidetached house, detached, stilt house, commercial, industrial, office, retail, supermarket, warehouse, church, monastery, mosque, temple, shrine, transportation, civic, fire station, public, toilets, stadium, hut, shed, garage, parking, storage tank, castle, military, tent, construction and bunker. For routes the sectors are, bus stop, train stop and metro station. For tourism, the sectors are picnic site, museum, viewpoint, zoo, motel, guest house, gallery, camp site, camp pitch, attraction, artwork and aquarium. We counted all the nodes of these sectors of each feature and represent the total count as the representation of that feature. A sample of Code snippet used to extract the "Education" feature is given below:

```python
def get_education_query(user_input):
        prefix = """[out:json][timeout:50];(""""#this is string of syntex in 'Overpass QL' language
        collegenode="""node["amenity"="college"](around:"""
        driving_schoolnode="""node["amenity"="driving_school"](around:"""
        kindergartennode = """node["amenity"="kindergarten"](around:"""
        language_schoolnode = """node["amenity"="language_school"](around:"""
        librarynode = """node["amenity"="library"](around:"""
        trainingnode = """node["amenity"="training"](around:"""
        music_schoolnode = """node["amenity"="music_school"](around:"""
        schoolnode = """node["amenity"="school"](around:"""
        universitynode = """node["amenity"="university"](around:"""
        suffix = """);out body;>;out skel qt;"""
        q = user_input[2]+','+user_input[0]+','+user_input[1]    #(radius,latitude,longitude) in a string form the user input
        built_query = prefix + collegenode+ q +');'+ driving_schoolnode+ q +');' + kindergartennode+ q+');'+ language_schoolnode+ q +');'  + librarynode+ q+');' + trainingnode+ q+');' + music_schoolnode+ q+');' + schoolnode+ q +');' + universitynode+ q +');' +suffix
        #combine all the above strings in correct order to form a query
        return built_query

def extract_nodes_data_from_OSM(built_query):
        api = overpy.Overpass()                         # creating a overpass API instance
        result = api.query(built_query)                 # get result from API by sending the query to overpass servers
        list_of_node_tags = []                          # initializing empty list , we'll use it to form a dataframe .
        for node in result.nodes:                       # from each node , get the all tags information
                node.tags['latitude'] =  node.lat
                node.tags['longitude'] = node.lon
                node.tags['id'] = node.id
                list_of_node_tags.append(node.tags)
        data_frame = pd.DataFrame(list_of_node_tags)  # forming a pandas dataframe using list of dictionaries
        if(data_frame.empty):
                return 0
        else:
                return data_frame['id'].count()
        data_frame.to_csv('output_data.csv')

query = get_education_query(lat,lng,rad)
data_frame = extract_nodes_data_from_OSM(query)
```

Similar code snippets were used to generate each of the features separately. However, for each feature, the get_featureName_Query function called different queries to calculate the number of nodes.

4.2.4 Bing Maps Image Extraction

Satellite imagery is essential for the optimal site selection problem because it provides a visual representation of the location that can be used to identify key features, such as the presence of major roads, buildings, and other structures. Using the Bing Maps API, satellite imagery is accessed, and a series of steps are performed to extract, store, and prepare the images for use in CNN model training. This section describes how to obtain satellite images, including accessing the API and storing and preparing the images for analysis. The extraction of satellite imagery from Bing Maps begins with the construction of a URL to access the Bing Maps API. The URL is constructed using the latitude and longitude of the location from which the image will be obtained, and the API returns an image as a response. The "get bing image" function is used to extract the images by calling the Bing Maps API with each location's latitude and longitude. The API returns a response, which, if successful, is converted into an image and saved in a particular directory using the target variable as the filename. A loop is then implemented to traverse each row of the dataframe containing the location's latitude, longitude, and target variable. For each row, the "get bing image" function is called with the associated values of latitude, longitude, and the target variable, which is set to the image's name. This code creates satellite images for each location in the dataframe, saving them in the specified directory with the variable target as the file name. These images can then be used to train a CNN model for the problem of optimal site selection.

4.2.5 Population Data Extraction

Humanitarian Data Index is a hub that compiles data from several sources. WorldPop uses census data from countries and then provides Geotiff files that contain the population as a layer of data over the map by coordinates. In order to collect this data the Geotiff files were downloaded and then converted to .XYZ format data using an end application named QGIS, a free and open-source geographic information system. We used QGIS 3.22.14 'Biaowiea' to convert the Geotiff files to our desired format. A simple raster conversion tool was used for this purpose available on QGIS. XYZ file format, on the other hand, is a simple file format that is used to represent 3D point data. It is commonly used in scientific and engineering applications to represent 3D coordinates. In our case, we used it to represent Latitude, Longitude, and Population. XYZ file allowed us to finally obtain a CSV file that is aggregated over 1km which then we can iterate through and find all the coordinates that are between our parameters. We can then calculate the total population in that region for each of our restaurant locations. A hurdle arises when we try to find the distance between each location by going through each of them and calculating the distance. Our population data has over 15 million coordinates and our yelp dataset has over a thousand data points. However, our time complexity is $O(n^2)$. So, even if each calculation takes about 500ms to process, it will require us about 5 hundred years to calculate all the locations in this way. Therefore, we needed a more optimized approach to finding the coordinates between our desired radius. Our first problem is the lookup. Using a database instead of CSV and turning Latitude and Longitude into the compound index can alleviate this problem. We are using MySQL for our purpose. MySQL is a widely used open-source relational database management system (RDBMS). It is based on the Structured Query Language (SQL) and is commonly used in web applications to store and manage data. MySQL is known for its reliability, flexibility, and ease of use. We then turn Latitude and Longitude into a compound index. A compound index is a database index that uses multiple fields, or columns, to index a table. It allows for faster retrieval of data by combining the performance benefits of multiple single-column indexes. This is particularly useful when you often query the database using multiple fields in the WHERE clause. This will allow us to perform fast lookups and not brute force through all the coordinates. However, in order to perform the lookup we need to find the coordinates that are between our 5km parameter away from our restaurant coordinates. That is why we are using latitude and longitude offset to find out the sets of coordinates that is 5km away from our restaurant location. In order to find the offsets, we can keep increasing the offsets by a small number of steps for latitude and longitude individually and finally calculate the coordinates. To find the distance between two coordinates we are using geopy's distance of geodesic function. A geodesic is a generalization of the notion of a "straight line" to "curved spaces". In the presence of a Riemannian metric, geodesics are defined to be the shortest path between two points on a surface. In geopy, the geodesic function uses the haversine function to calculate the distance. The Haversine formula is an equation that gives the distance between two points on the surface of a sphere, such as the Earth. It is often used in navigation and other applications that involve measuring distances on a sphere. The Haversine formula is an implementation of the "great-circle distance" calculation, which is the shortest distance between two points on the surface of a sphere. The Haversine method gives an accurate way of determining the distance between any specified longitude and latitude [42].

$$Central\ Angle\ (\Theta) = \sqrt{sin^2\left(\frac{\Delta lat}{2}\right) + cos\,(lat2)\cdot cos\,(lat1)\cdot sin^2\left(\frac{\Delta lon}{2}\right)}$$

$$Distance\ (D)\ =\ 2\cdot R\cdot \sin^{-1}(\Theta)$$

[Here, R is the radius of the sphere (such as the Earth)]
After calculating the offsets, we can do a simple SQL query to find all the coordinates inside those Latitude and Longitudes and utilize our compound indexing. This approach is a million times faster than our previous method and took not more than five minutes to compute all the population in a 5km area for each of our restaurant coordinates.
We finally obtained the coordinates that are used as a base to plot the population to coordinates and then find the total count within the range of 5km of each restaurant on our list of targets.

```
FUNCTION find_population_in_area(latitude, longitude, distance_km, offset_step):
    longitude = float(longitude)
    latitude = float(latitude)
    location = latitude, longitude
    longitude_offset = 0
    WHILE geodesic_distance(location, (latitude, longitude + longitude_offset)).km <
    ↪  distance_km:
        longitude_offset += offset_step
    latitude_offset = 0
    WHILE geodesic_distance(location, (latitude + latitude_offset, longitude)).km  <
    ↪  distance_km:
        latitude_offset += offset_step
    result = mysqlDB.execute("
        SELECT * FROM population WHERE
        (longitude BETWEEN {longitude - longitude_offset} AND {longitude + longitude_offset})
        ↪  AND
        (latitude BETWEEN {latitude - latitude_offset} AND {latitude + latitude_offset})")
    total_population = 0
    FOR EACH population IN result:
        total_population += population
    RETURN population
distance = 5  // in Kilometer
population_list = new Array()
offset_step = 0.00015
FOR EACH latitude AND longitude IN yelp_dataset:
    population = find_population_in_area(latitude, longitude, distance, offset_step)
    population_list.push(population)
```

4.3 Data Pre-processing

The objective of the data preprocessing phase was to prepare the dataset for machine learning models. The final dataset was obtained by combining multiple datasets and retaining only relevant columns, including a target column representing the scaled number of check-ins from 2021 and 10 feature columns, including building, sustenance, tourism, hospital, roads, facilities, entertainment, education, routes, and population. The variable of interest was scaled with the MinMaxScaler method. This scaling method was selected after evaluating various scalers, including the standard scaler, maxabs scaler, and robust scaler, and observing that MinMaxScaler performed the best in the majority of models.

Target	*Building*	*Sustenance*	*Tourism*	*Hospital*	*Roads*	*Facilities*	*Entertainment*	*Education*	*Routes*	*Population*
129	0	98	29	1	5246	42	14	28	271	89381.4
111	3	609	82	2	9420	110	58	207	259	142661
53	8	268	72	6	18819	87	41	80	134	137326
445	0	95	28	1	4656	40	14	25	255	89381.4
100	16	1064	250	3	24925	608	84	129	433	537635

Table 4.4: Sample of Dataset Before Pre-processing

To further improve the quality of the dataset, the z-score method was used to detect outliers. This method identifies outlier indices by calculating the z-score of each data point and comparing it to a threshold value. In this instance, the threshold was set to 3, and the outlier rows were eliminated from the dataframe. This action decreased the number of rows from 1108 to 1002 rows. We also looked at different ways to scale the data and found that Lasso regression worked better without scaling the data. This could be because Lasso regression doesn't change much when the scale of the features changes. In a ratio of 8:2, the dataset was divided into training and test sets. Initially, we also attempted to assign weights to scaled features based on the coefficient or feature importance from SelectKBest. However, it was observed that this decreased accuracy, so we reverted to not assigning weights. This may be due to the fact that assigning weights to the features based on their importance assumes that the feature's importance is constant across the entire dataset, which may not be the case.

Chapter 5

Model

After data extraction and preprocessing, we trained and evaluated our models using a variety of machine learning algorithms and neural networks. Support Vector Regression (SVR), Random Forest (RF), XGBoost, Ridge Regression, Lasso Regression, ElasticNet, MLP, and CNN were among these. To optimize the performance of these models, grid search cross-validation was utilized to tune their hyperparameters. We trained each model and extracted feature weights or coefficients, as determined by the models, for each feature. To evaluate the performance of our models, we utilized a variety of accuracy metrics, including Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error, Max error, and MAD (MAD), to gain a thorough understanding of the models' performance.

5.1 Implementing ML Models

5.1.1 Tuning Each ML Model

To optimize the performance of our machine learning models, we tuned each model's hyperparameters using grid search and cross validation. By training and evaluating the models with various parameter combinations, we were able to determine the optimal parameter combination for each algorithm. For instance, the optimal combination of parameters for the SVR model was an epsilon value of 0.05, a gamma value of 2, and the kernel as rbf. Similarly, we discovered that the optimal combination for the XGBoost model was a learning rate of 0.05, a maximum depth of 1, and 132 estimators. This hyperparameter tuning allowed us to improve the precision of our models. The optimal combination of parameters for the Random Forest model was determined to be a maximum depth of 1, log2 as max features, 4 as minimum samples leaf, 11 as minimum samples split, and 129 as number of estimators. We used an alpha value of 0.0005, a l1 ratio of 0.3, and cyclic selection for the ElasticNet model. We discovered that setting fit intercept to true and solver to "saga" produced the best results for ridge regression. For Lasso Regression, alpha was set to 0.0002, fit intercept was set to true, positive was set to false, and precompute was set to false. These parameters yielded the best results for us. By fine-tuning each model's hyperparameters, we were able to improve the overall performance of our models and achieve the best possible results. A summary of all the tuned parameters can be gauged from the table below:

Model	Parameters	Tuned Value	Model	Parameters	Tuned Values
SVR	*Epsilon*	0.05	***ElasticNet***	*Alpha*	0.0005
	Gamma	2		*L1 Ratio*	0.3
	Kernel	rbf		*Selection*	Cyclic
XGBoost	*Learning Rate*	0.05	***Ridge Regression***	*Fit Intercept*	True
	Max Depth	1		*Solver*	saga
	N Estimator	132			
Random Forest	*Max Features*	log2	***Lasso Regression***	*Alpha*	0.0002
	Max Depth	1		*Fit Intercept*	True
	N Estimator	129		*Positive*	False
	Min Samples Split	11		*Precompute*	False
	Min Samples Leaf	4			

Table 5.1: Tuned Parameter Values

5.1.2 Optimizing Multilayer Perceptron

During the process of the MLP being fine-tuned, the architecture of the model was modified in a number of different ways in order to improve its overall performance. In order to find the most efficient configuration, various combinations of the number of layers, the number of neurons, and the dropout rate were tested. In addition, several different optimizers, including Adagrad, SGD, RMSprop, and Adadelta, were put through their paces in order to determine which one would be the most effective when used in conjunction with the model. Following an analysis of the obtained results, it was concluded that the RMSprop model with a learning rate of 0.01 offered the most satisfactory levels of performance. During the process of tuning the neural network, the loss function was also an important consideration. Several different loss functions, including Mean Squared Error (MSE), Mean Absolute Error (MAE), and Huber Loss (with delta = 1.0), were evaluated in order to find the one that worked best with the model. In the end, it was determined that MSE produced the most satisfactory results. Early stopping with a patience of 10 was implemented to prevent overfitting and to ensure that the model would generalize well to new data. This method terminates the training process as soon as the model no longer shows any signs of improvement. Additionally, the model's optimal weights are reinstated so that it does not become stuck in a local minimum. For the purpose of this study, a sequential model of a neural network with three hidden layers, each of which contains 20248, 64, and 32 neurons, respectively, was chosen as the optimal neural network architecture. This architecture was selected because of its capability of learning the complex relationships contained within the dataset as well as its capacity of extracting high-level features via multiple layers. The overfitting problem is solved by employing a dropout rate of 0.2. The model is compiled using the optimizer RMSprop, and the learning rate is set to 0.01, and the loss function is set to MSE. Batch normalization is utilized after each hidden layer to quicken the learning process and reduce the internal covariate shift. A sigmoid activation function is present in the output layer, which contains only one neuron. One neuron was present in the output layer because, in a regression problem, the goal is to predict a continuous value rather than a categorical value. A single neuron can output a scalar value, which is suitable for representing a continuous prediction. The interpretation of the model's predictions is also made simpler by using an only one neuron in the output layer because it is obvious that the model is making a single prediction. The application of the Sigmoid activation function in the output layer contributes to the final output being in the range of 0 to 1 as much as possible.

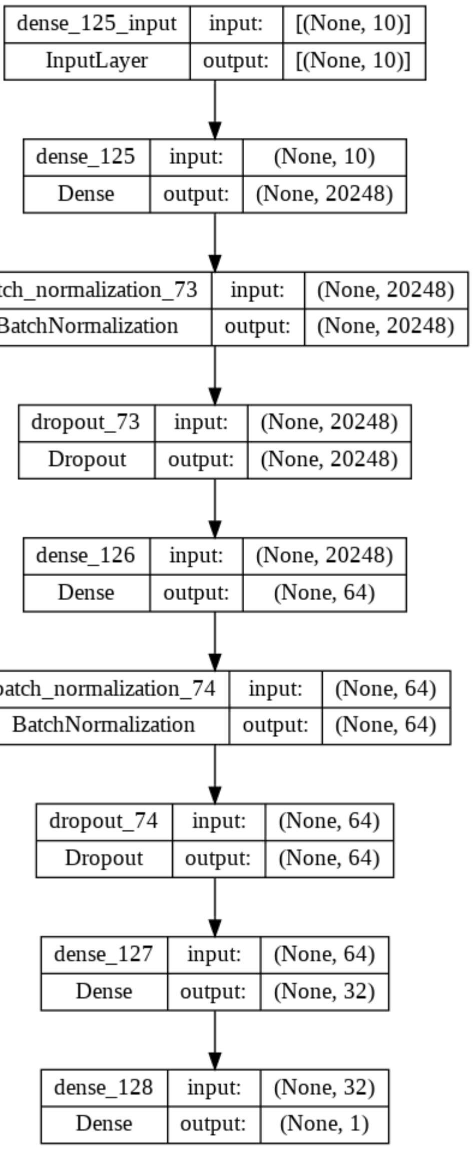
dense_125_input | input: | [(None, 10)]
InputLayer | output: | [(None, 10)]
dense_125 | input: | (None, 10)
Dense | output: | (None, 20248)
batch_normalization_73 | input: | (None, 20248)
BatchNormalization | output: | (None, 20248)
dropout_73 | input: | (None, 20248)
Dropout | output: | (None, 20248)
dense_126 | input: | (None, 20248)
Dense | output: | (None, 64)
batch_normalization_74 | input: | (None, 64)
BatchNormalization | output: | (None, 64)
dropout_74 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_127 | input: | (None, 64)
Dense | output: | (None, 32)
dense_128 | input: | (None, 32)
Dense | output: | (None, 1)

Figure 5.1: Neural Network Architecture

In the end, the combination of this architecture, the fine-tuning of the optimizer and loss function, and the implementation of early stopping produced a model that performed well on the dataset and provided more accurate predictions.

5.1.3 Training and Creating Meta Model by Stacking

We trained several machine learning models, including Support Vector Regression (SVR), Random Forest (RF), Lasso Regression, Ridge Regression, XGBoost, and Neural Network, after adjusting the hyperparameters of each model. On the basis of numerous evaluation metrics, including mean absolute error, mean squared error, root mean squared error, median absolute error, maximum error, and MAD, the performance of these models was evaluated. These metrics gave us insight into the performance of each model and allowed us to compare them. To create a meta model, we utilized the stacking regression technique, which improves overall performance by combining the predictions of multiple models. In our implementation, we employed Ridge Regression, Random Forest, XGBoost, Lasso Regression, Elastic Net, and SVR as the base models, and Ridge Regression as the final estimator. The StackingRegressor from the Sklearn library was utilized. We omitted neural networks or MLP from our meta model for a variety of reasons. First, neural networks can have a large number of parameters, making it difficult to determine the optimal set. In addition, neural networks are computationally costly and training can be slow. In addition, our dataset is relatively small, and the performance of neural networks is data-dependent. Therefore, we decided to omit neural networks from our meta model and rely on an ensemble of other models that performed well individually.

5.2 Implementing CNN Model

In the implementation of the CNN model, the data was trained on various pre-trained models, including VGG16, VGG19, ResNet152, ResNet50, DenseNet121, and InceptionV3, in order to evaluate their performance using various evaluation metrics, including RMSE,MSE, MAE, MedAE, and MAD. Comparing the results, we discovered that both the DenseNet121 and VGG16 models performed admirably. DenseNet 121, which had fewer parameters, was selected as the preferred model. Following this, we fine-tuned the DenseNet 121 architecture further and chose the best-tuned model based on the evaluation metrics.

5.2.1 Assessing Different Architectures

In this implementation, the performance of various CNN architectures was evaluated using the provided regression problem. As foundational models, the VGG16, VGG19, ResNet152, ResNet50, DenseNet121, and InceptionV3 models were utilized. These pre-trained models were trained on a large dataset and have already learned the features and patterns of images, which can be customized for the current problem. For each of these architectures, a dense output layer with a single neuron and sigmoid activation function, which is suitable for regression problems, was added to the pre-trained model. Because our output variable was scaled with a minmax scaler to fall within the range of zero to one, the Sigmoid function was applied to the dense layer. A mean squared error loss function and an Adam optimizer were then utilized to compile the model. The model was trained for 200 epochs, with the patience parameter set to 10 to avoid overfitting. In some models, the use of pre-trained weights from the Imagenet dataset has advantages and disadvantages. The model can leverage the pre-trained model's general knowledge and adapt it to the specific dataset, which can improve the performance of the final model. Due to the fact that the pre-trained model has already learned features and patterns from a large dataset, fine-tuning the model enables it to learn the specific features and patterns of the target dataset. On the other hand, there are disadvantages to using pre-trained weights. As the model has already been trained on a large dataset, it may not require as much training on the specific dataset, which can increase the likelihood of overfitting. Using pre-trained weights can also increase the computational cost and training time of a model. Another strategy is to avoid using pre-trained weights, in which the layers of the pre-trained model are frozen and not updated during the training process. This method reduces the risk of overfitting, as well as the computational cost and training time required for the model. However, it may not be able to utilize the general knowledge of the pre-trained model, and the performance of the final model may be diminished. It is essential to note that the decision to use pre-trained weights depends on the specific problem, dataset, and available resources. In some instances, using pre-trained weights can significantly improve the model's performance, whereas in others it may be unnecessary or even detrimental. Therefore, it is essential to evaluate the model's performance using various techniques and make an informed decision based on the outcomes. The purpose of experimenting with various architectures is to identify the model with the best performance for the given problem. Each architecture has distinctive properties and characteristics that make it appropriate for a variety of problems and data types. For instance, the VGG16 and VGG19 architectures are well-known for their ability to capture spatial information and are frequently employed in image classification tasks. ResNet152, ResNet50, DenseNet121, and InceptionV3 architectures, on the other hand, are renowned for their capacity to handle deeper networks and are frequently employed for tasks requiring more complex feature extraction. By experimenting with various architectures, we can compare their performance on the given problem and dataset. This enables us to determine the optimal architecture for the problem and dataset, thereby enhancing the performance of the final model. In addition, by experimenting with various techniques, such as freezing layers or utilizing pre-trained weights, we can further refine the model and achieve superior results. Then, we determined the optimal architecture in terms of precision and computational speed. The tuning of this model is discussed in a later section.

5.2.2 Fine Tuning Selected Architecture

In the preceding step, we trained and evaluated a number of CNN architectures, including VGG16, VGG19, ResNet152, ResNet50, DenseNet121, and InceptionV3, in order to identify the model with the best performance for our regression problem. We discovered that DenseNet121 and VGG16 had comparable performance, but because DenseNet121 had fewer parameters, we chose it as our base model to further refine. Fine-tuning is the process of modifying a pre-trained model to better fit our particular problem and data set. To achieve this, the DenseNet 121 model was fine-tuned through trial and error. The DenseNet121 model was initially loaded with pre-trained weights but without the fully connected layers on top. Then, we added a few additional layers to the model to enhance its performance for our particular problem. First, we added a flattening layer to transform the output of DenseNet121 into a 1D array so that it could be fed to a fully connected layer. Afterwards, we added a dropout layer with a rate of 0.2 to prevent overfitting by randomly removing 20 percent of the neurons during training. Next, we added a batch normalization layer to improve the training process's stability and speed by normalizing the input to the next layer. Then, two layers with 32 neurons each and a ReLU activation function were added. The ReLU activation function is a popular option for neural networks because it accelerates the training process. To avoid overfitting, we then added a second dropout layer with a rate of 0.2. We added a single neuron to the final output layer because we are attempting to solve a regression problem with a continuous target variable. We used the sigmoid activation function in this layer because the range of our target variables is between 0 and 1.

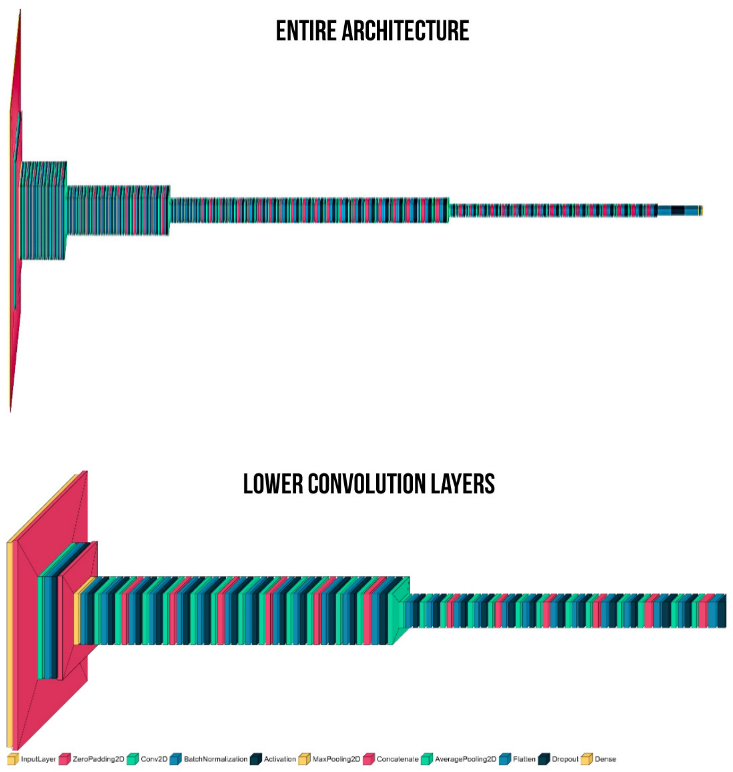

Figure 5.2: Modified DenseNet Architecture

In addition, we loaded the models both with and without their pre-trained weights. In some models,some of the layers of the pre-trained model were frozen, and weights from the Imagenet dataset were loaded into the model. This indicates that the pre-trained model's weights will not be updated during the training process. This is done to capitalize on the knowledge of the pre-trained model and fine-tune the model for the specific task at hand. The goal is to adapt the general knowledge of the pre-trained model to the specific dataset. This technique is based on the assumption that the pre-trained model has already learned features and patterns from a large dataset, and that by fine-tuning the model, it can learn the specific features and patterns of the target dataset. In some models, however, the layers of the pre-trained model were not frozen and the model was loaded without the pre-trained weights. This indicates that the pre-trained model's weights will be updated during training.

Chapter 6

Result and Analysis

The data we used to make predictions were not linear. In machine learning, linearity is an important factor to consider when analyzing data. The relationship between a dependent variable and one or more independent variables that is represented by a straight line is referred to as linearity. The change in the dependent variable is directly proportional to the change in the independent variable in a linear relationship (s). Linearity is a desirable characteristic because it simplifies the modeling process and facilitates straightforward interpretation of the results. However, in many actual situations, the data may differ from what we observed. Non-linear data can be more complex and harder to model, as the relationship between the dependent and independent variables is not represented by a straight line. It can also result in increased errors and less precise predictions. Therefore, the linearity of the data was determined by analyzing scatter plots and using a linear regression model to fit the data.

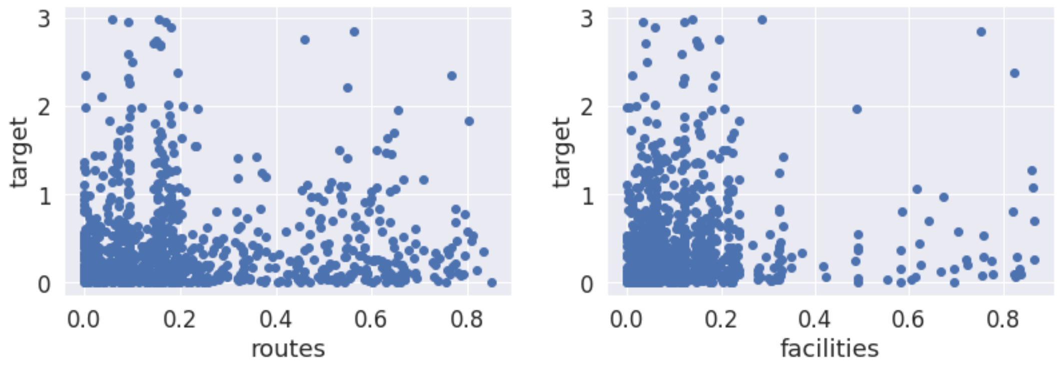

Figure 6.1: Scatter Plot of Routes and Facilities

After feeding the data into a linear regression model, the resulting values indicated a low R-squared value of 0.04, a weak correlation coefficient of 0.20005, and a high Mean Squared Error of 0.2568 and Least Squares Error of 257.33. These are all indications that the linear regression model is not a good fit for the data. This non-linearity of the data has important implications for the analysis. We were required to use non-linear models capable of recognizing the data's complex relationships more accurately. In addition, it was necessary to perform additional preprocessing and feature engineering in order to transform the data for improved linearity. Similarly, in order to evaluate the accuracy of the models, it was essential to select appropriate evaluation metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Squared Error (MSE), Median Absolute Error (MedAE), Max Error (ME), and Median Absolute Deviation (MAD)

6.1 Feature Correlation

We used the Pearson correlation coefficient to visualize the relationship between the various features in our dataset. The Pearson correlation heatmap is an essential instrument for comprehending the relationships between a dataset's features. The heatmap displays the Pearson's r correlation coefficient, which ranges from -1 to 1. A value of -1 represents a perfect negative correlation, 0 represents no correlation, and 1 represents a perfect positive correlation.

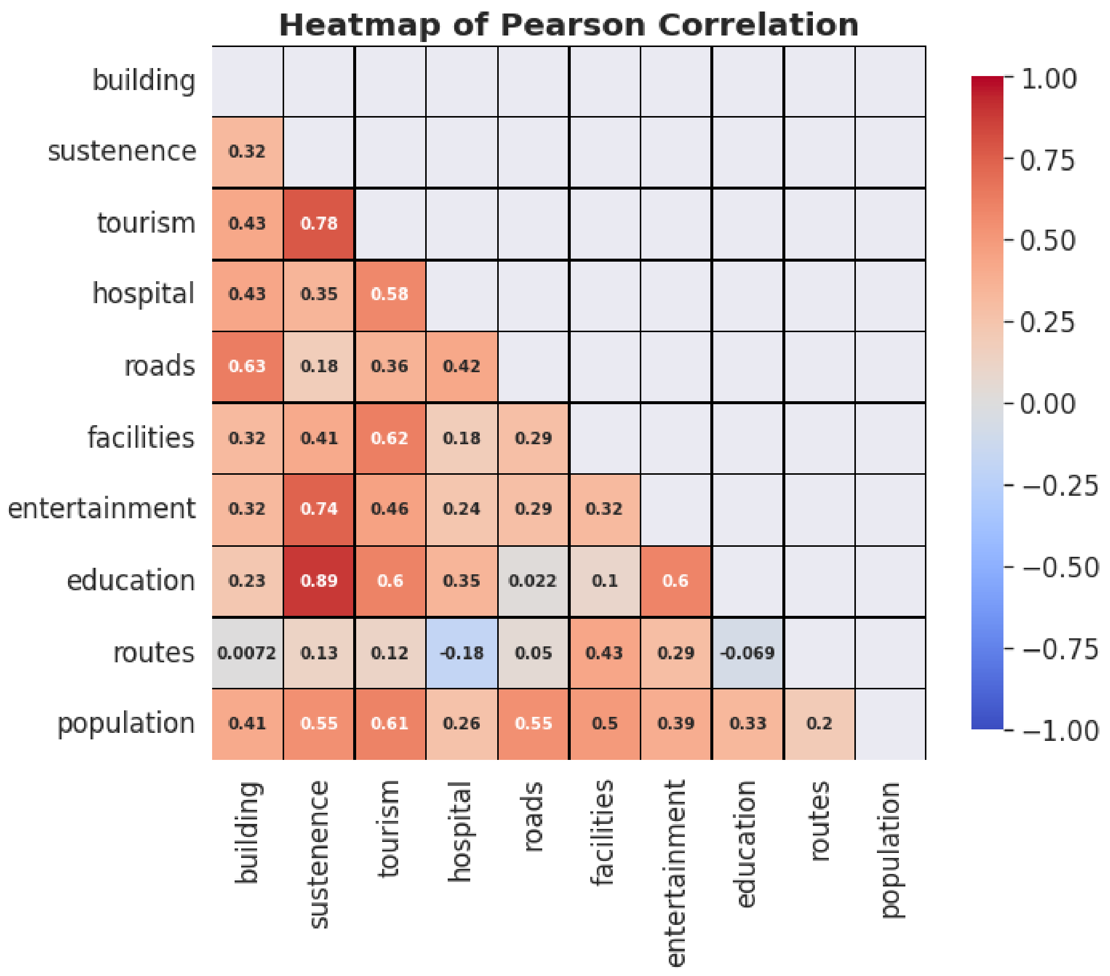

Figure 6.2: Pearson Correlation Coefficient Heatmap

It is evident from the heatmap that the features in this dataset have a low correlation with one another, as the majority of coefficients are close to 0. This is advantageous for machine learning models, as it suggests that the features contain distinct information and can provide complementary data to the model, resulting in improved

performance. Even though there is a small amount of negative correlation in the dataset, it should not pose a problem because it can provide additional information that can help improve the performance of the model.
For example, the correlation coefficient between "roads" and "education" is 0.02198, indicating a very weak correlation between the two. Similarly, the coefficient for "routes" and "education" is -0.068539, indicating a weak negative correlation. This indicates that including both of these features in a machine learning model can provide unique information and improve the performance of the model.
However, there are some features with a moderate correlation coefficient, such as "sustenance" and "tourism" (r = 0.775072) and "education" and "sustenance" (r = 0.887370), which can result in overfitting if both features are used together in the model. This was taken into consideration during model train-testing, and feature space engineering was performed on our dataset. In conclusion, the low correlation between the features in this dataset is advantageous for our machine learning models because it indicates that the features contain distinct information. To prevent overfitting, this additional step of determining the correlation between the features was performed.

6.2 Feature Weights

While studying the feature weights assigned by various models, we discovered that model performance differs when weights are assigned manually as opposed to when the model decides by itself. The selectKbest f-regressor was used to evaluate the significance of each feature. After ML training, we discovered that the selectKbest model allocated relatively high weights to features such as sustenance and tourism, whereas the SVR model assigned significantly lower weights to sustenance. In addition, the XGBoost algorithm assigns weights in the form of feature importance scores, with sustenance ranking highest, followed by roads and population.

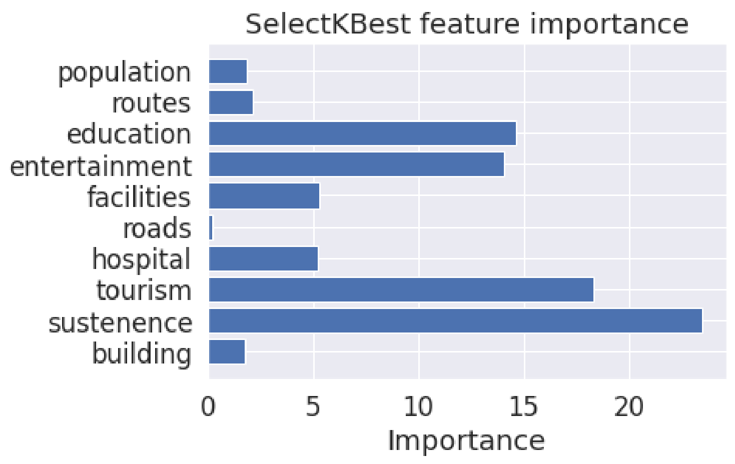

Figure 6.3: SelectKBest Feature Importance

In addition, based on the weights assigned by various models, it may be argued that aspects such as sustenance and tourism played a significant impact in defining the objective variable. The models' high importance scores for these features corroborate this claim. In contrast, several models assigned relatively low relevance scores and weights to features such as buildings, thus we opted to exclude them from our feature list. However, eliminating it did not increase the accuracy of the predictions, indicating that some models were able to use this feature to generalize the data. Additionally, the Lasso, Ridge, and ElasticNet models assigned zero weights to certain of the features, representing a feature elimination; this had the same impact as eliminating those features. This is evidenced by the zero weights provided to buildings, facilities, education, and population by ElasticNet and Lasso Regression. Some of the characteristics, such as facilities, education, and population, were assigned near-zero weight by ridge regression. Therefore, these models do not use these values, which can be considered equivalent to elimination of features for these models. In addition to varying feature weights, these models also allocated negative weights to some features. For instance, the ElasticNet model assigned a negative weight to the feature "roads," suggesting that the feature has a negative correlation with the target variable and, as a result, the model is assigning it a negative weight. Likewise, the Ridge Regression and Lasso Regression models assigned negative weights to "roads" and other characteristics, including "population" and "facilities." This indicates that these features are less linked with other features, and an increase in these features results in a reduction of the target variable. It offers a unique perspective on the association between the features and the target variable.

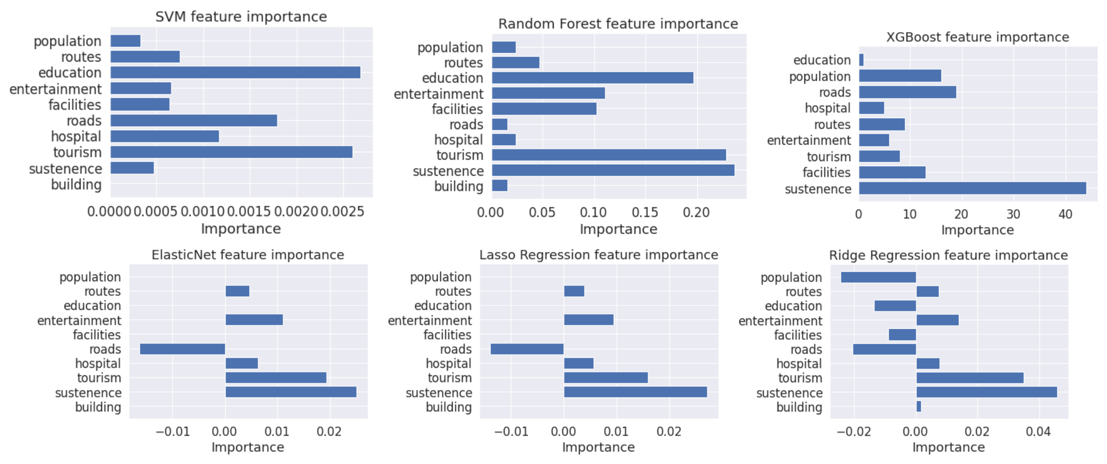

Figure 6.4: Feature Importance Assigned by ML Models

During our comparative investigation of multiple machine learning models, we tested with both manually giving weights and eliminating features, as previously described. However, according to our findings, models performed best when neither manual weight assignment nor feature elimination was used. Some models, such as Lasso and ElasticNet, may have assigned zero weights to certain features, resulting in feature elimination. It is also plausible that the approach and assumptions of each feature played a role in the performance of the models with and without manual weight assignment and feature elimination.

6.3 Accuracy Metrices

As previously mentioned in our paper, it is crucial to select accuracy metrics that are appropriate for nonlinear datasets when evaluating the performance of our machine learning models.The accuracy metrics that we used to evaluate the models are RMSE, MAE, MSE, MedAE, ME and MAD. By taking the square root of the mean squared error, the Root Mean Squared Error (RMSE) measures the average magnitude of the error. This metric is commonly utilized because it penalizes larger errors more severely, making it suitable for our dataset as it can measure the error's magnitude. Mean Absolute Error (MAE) measures the average magnitude of an error without taking into account its direction. This metric is especially useful for us because it is resistant to outliers, which is essential given that our dataset is non-linear. Mean Squared Error (MSE) is applicable to our data set because it penalizes larger errors more severely than smaller ones. The Median Absolute Error (MedAE) is comparable to the Mean Absolute Error (MAE), but it is based on the median of the errors, making it more resistant to outliers. It is a useful metric for our dataset because it provides a better understanding of the error variance. Max Error (ME) measures the largest error in the dataset and is useful for determining the model's performance in the worst-case scenario. This is important for our dataset because it provides insight into the error's upper limits. The Median Absolute Deviation (MAD) subsequently measures the median of the absolute deviation of the errors from the median. It is a robust metric that is unaffected by outliers, making it appropriate for our nonlinear dataset. RMSE, MAE, MSE, MedAE, ME, and MAD have been selected as a suitable combination of accuracy metrics for non-linear datasets. These metrics provide a comprehensive evaluation of the performance of the model, capturing the magnitude and variability of the errors as well as the error's worst-case scenario. By combining these metrics, we can gain a deeper understanding of the performance of the model and pinpoint areas for improvement.

6.4 Model Assessment

As mentioned previously, different error metrics were used to evaluate the performance of both ML models and CNN architectures. The performance of the regression models and meta models was distinct from that of the CNN model. We experimented with various combinations of feature space engineering and tuning model hyperparameters to determine the optimal solution to the site selection problem.

6.4.1 Performance of ML model

The MLP model performed the best among all the models, with the lowest RMSE, MSE, MAE, and medAe. One of the key reasons for the superior performance of the MLP model is its ability to handle non-linear data. The Neural Network model is able to learn and model non-linear relationships in the data, which is often the case in real-world datasets. This is achieved by using non-linear activation functions in the hidden layers like Relu which was used in our case which allows the model to learn complex, non-linear relationships in the data.
The Ridge model was the second best performer, followed by the Lasso and Elastic Net models. Ridge and Lasso are linear models with regularization, which can prevent overfitting and improve generalization. Elastic Net is a combination of Ridge and Lasso, which can balance the strengths of each. The XGBoost and Random Forest models performed similarly, with slightly higher error values than the Neural

Network, Ridge, Lasso, and Elastic Net models. These models are ensemble methods that can improve performance by combining multiple decision trees. However, they may not be as effective as the Neural Network model in capturing complex relationships in the data.
The SVR model performed the worst among all the models. However, the SVR model had the lowest Max Error (ME) among all the models, which indicates that it has fewer extreme errors. The Max Error is the largest error between the predicted and actual values. It is a metric that measures how far the model's predictions deviate from the actual values. This indicates that the SVR model has fewer extreme errors and therefore has a more consistent performance. This may be beneficial in certain situations where it is important to avoid extreme errors. One possible reason SVR performed the worst is that the dataset and task in this study may be complex and have a large number of features; SVR may not be as effective in capturing complex relationships in the data in this case. Additionally, SVR is a powerful model for classification, but it may not perform as well for regression tasks.

Model	RMSE	MSE	MAE	MedAE	Max Error	MAD
SVR	0.051758	0.002678	0.042388	0.0418288	0.198737	0.042388
XGBoost	0.050079	0.002507	0.037569	0.031878	0.216459	0.037569
Random Forest	0.050105	0.002510	0.037740	0.032165	0.214662	0.037740
Ridge Regression	0.049876	0.002487	0.037315	0.031676	0.208979	0.037315
Lasso Regression	0.049927	0.002492	0.037519	0.031214	0.210440	0.037519
ElasticNet	0.049924	0.002492	0.037477	0.030851	0.210131	0.037477
MLP	0.049618	0.002461	0.036860	0.030752	0.209639	0.036860
Meta Model	0.049963	0.002496	0.037595	0.033810	0.213958	0.037595

Table 6.1: Table of Evaluation Matrices

The Meta model, which combines multiple models, performed similarly to the Lasso and Elastic Net models in this study. This suggests that the stacking method used to create the Meta model was not able to improve the performance of the individual models. In this specific case, the Meta model was created by stacking the Lasso, Elastic, Ridge, SVR, XGBoost, and Random Forest models. One possible reason for this could be that the individual models used in the Meta model were not diverse enough. If the models used in the Meta model are similar in terms of their structure and assumptions, then the combination of these models may not lead to a significant improvement in performance. Additionally, the dataset and task in this study may not be complex enough to benefit from the use of a Meta model. Another possible reason is that the stacking method used in the Meta model was not able to effectively combine the predictions of the individual models. It's also possible that the individual models used in the meta model were already performing well and that combining them didn't improve their performance any further.
Overall, the results of this study suggest that the Neural Network or MLP model is the most suitable for this task, followed by the Ridge, Lasso and Elastic Net models, while the XGBoost, Random Forest and SVR models have similar performance. The SVR model has the lowest max error, which can be useful when we want to avoid extreme errors. However, the overall performance is not as good as the other models. We used a Sequential model for the Neural Network and although the model

Model	RMSE	MSE	MAE	MedAE	Max Error
VGG16	0.09141	0.00835	0.04629	0.03554	0.92388
VGG19	0.09141	0.00835	0.04640	0.03580	0.92362
ResNet152	0.09141	0.00835	0.04645	0.03591	0.92350
ResNet50	0.09141	0.00835	0.04635	0.03569	0.92372
DenseNet121	0.09141	0.00835	0.40628	0.03551	0.92391
InceptionV3	0.09141	0.00835	0.04647	0.03595	0.92347
Modified DenseNet	0.09088	0.00824	0.04476	0.02454	0.82192

Table 6.2: Table of Evaluation Matrices of CNN

performed well, we believe that we could have achieved even better results with a larger dataset and more fine-tuning of the model. One of the reason is the larger dataset would have allowed the model to learn more complex relationships in the data and generalize better to unseen data. Another reason is that fine-tuning a Neural Network model can be a time-consuming and difficult task. With a larger dataset, it would have been possible to use techniques such as cross-validation to more effectively fine-tune the model's hyperparameters. With more data, the model would have been able to generalize better and give more accurate predictions.

6.4.2 Performance of CNN architecture:

The CNN algorithm showed a different trend of values in evaluation metrics. In this study, we trained a number of convolutional neural network (CNN) architectures on a dataset of 1108 satellite images collected from Bing Maps. The goal was to find the best CNN architecture for the task of predicting numerical values from the images. We compared several popular architectures, including VGG16, VGG19, Resnet152, Resnet50, DenseNet121 and InceptionV3.
From the results, it is clear that the VGG16, VGG19, Resnet152, Resnet50 and InceptionV3 performed similarly, with similar RMSE, MSE, and MAE. However, DenseNet 121 performed slightly better than the other architectures. One of the reasons for this could be its efficient use of parameters, which allows it to learn more effectively from the data. Additionally, the lower number of hyperparameters also made it easier to train, reducing the chance of resource exhaustion and resulting in faster training time. After selecting DenseNet121 as the best architecture, we fine-tuned it by modifying the top layers of the architecture. This resulted in an even better-performing model with a lower RMSE, MAE, MSE, and Max Error score. This suggests that fine-tuning the CNN architecture by modifying the top layers can lead to improved performance since the original model was built for classification problems. It is important to note that, given that this was a regression problem, an additional hidden layer with a single neuron was incorporated into the output layer of each of our models. Furthermore, to account for the target variable range of 0 to 1, a sigmoid activation function was applied to the output layer. It should be noted that the dataset
The dataset we used was relatively small, and with a larger dataset and more fine-tuning, the performance of the DenseNet121 model could potentially be even better. Nevertheless, our results demonstrate the potential of CNNs for extracting valuable

information from satellite images and highlight the importance of carefully selecting and fine-tuning the architecture for a specific task.

6.4.3 Performance Analysis

After reviewing the assessment metrics of the many machine learning models utilized in this study, it is evident that the neural network model trained using OSM and HumDataIndex outperformed all other models. We found that the neural network model handled the dataset's complexity and non-linearity more effectively than the other models. The CNN model, which was optimized with DenseNet121 and trained on a different satellite image than other ML models, performed worse than all of those models. The error metrics reveal that the model's error levels are more than double those of the other models. The CNN model trained on satellite imagery performed worse than other models in this study for a number of fundamental reasons. According to our hypotheses, one of the potential causes could be that the dataset utilized to train and evaluate the model was not optimal for a CNN model. There may not have been sufficient characteristics or complexity in the data to adequately train the model.
In addition, the DenseNet121 architecture might not have been optimal for the particular regression issue being addressed.Another possibility is that insufficient data was used to train the model, leaving it underfitted for the task. Due to a lack of training data, the model may not have been able to learn the underlying patterns in the data. Subsequently, the model might not have been fine-tuned enough. This could have led to improper hyperparameter settings, which would have contributed to the model's subpar performance. It is also possible that other models did better due to the fact that we employed a variety of numerical data as features to train them instead of satellite data. So the nature of that dataset was suitable for predicting our target value. Another probable explanation for the CNN model's low performance could be the quality of the satellite imagery used. The low resolution of the photos made it challenging for the model to extract useful features from the data.
It is also possible that the model's architecture was not optimal for the intended purpose. For example, if the problem required more fine-grained predictions, a simpler architecture may not have been able to capture the necessary level of detail. The architecture implemented in this work, DenseNet121, was originally intended for object classification on the Imagenet dataset. The model's convolutional layers may have been optimized for that specific task and thus were unsuitable for a regression problem involving satellite imagery. The architecture may not have been able to effectively extract relevant features from the data for the task of predicting target values in a regression problem. So, a CNN architecture that was made specifically for the regression problem regarding satellite imagery might have been better for this goal.

Chapter 7

Future Work and Conclusion

7.1 Future Prospects

While conducting the study on this comparative research we were faced with many challenges. We noticed that real-world problems and data sets have complex relationships and the relationship between the features and the label was non-linear. Because of that, we had to choose algorithms that could work with non-linear data. In the future, we would like to use the similar methodology and linear data to analyze how much it changes our generalized model. Because of the non-linearity of the data, machine learning algorithms performs slightly better than the CNN models. We used CNN architectures such as VGG16, VGG19, and DenseNet121 all of which were used for object classification problems. However, these CNN architectures were trained using the Imagenet dataset because they are not best tuned for satellite imagery. When conducting research in the future, architectures that are best tuned to work with satellite imagery should be used. Along with satellite imagery, other images could be incorporated that contain relevant features and trained in a multichannel CNN architecture to produce better results. On the same note, for DenseNet121, the top layer prediction did not produce expected outcomes and was lacking precision due to our limited parameters, fewer satellite data, and short training time. There is a need to collect larger and higher resolution satellite imagery datasets to generate the best outcome that these architectures can offer. Although the MLP model is performing slightly better than the other machine learning algorithms, if the hyperparameters were optimized more using a bigger dataset, the result generated would have been much better. In that case, it has the potential to significantly outperform the machine learning models working on numerical data.

7.2 Conclusion

Site selection for new enterprises and non-franchise establishments is a topic that is less frequently explored but can have many benefits for both businessmen and clients. Most startups and business owners struggle to determine the best location since they are unsure how to approach this challenging task due to the numerous dynamic variables involved. We have, however, successfully developed a method that uses a combination of multiple models and open-source data, such as satellite

imagery, Open Street Map features, and World Population Data, to identify the best location for a new business, in this case, a restaurant. Our techniques have been evaluated using different metrics, and we can select the most generalized model with the lowest margin of error.
As our work used several different ML models, we observed how each of these models assigned different weights to different features. Out of all the features in our dataset, sustenance and tourism were the most weighted features in most of the models. Tourism and sustenance have a strong correlation, as evidenced by the Pearson correlation heat map. In our dataset, Sustenance represents cafes, fast-food restaurants, pubs, catering companies, and so on. Therefore, it was evident that an area that had more tourism also had more food stores, and vice versa. Our analogy for tourism and sustenance highly affects the optimality of a location such that when a similar type of business occupies an area and also attracts tourism, it will be a more optimal choice to start relevant businesses there. Out of all the models employed to calculate the optimality of a location, MLP proved to be the most efficient. It gave an RMSE value of 0.0496, an MSE value of 0.00246, an MAE value of 0.0368, a MedAe value of 0.0307, and a MAD value of 0.036. All of these error matrices and evaluation matrices were lower than all the other models. DenseNet 121 architecture with optimized top layers performed the best of the models that were train-tested with satellite imagery. However, these CNN models were unable to generalize the dataset well or understand the patterns in the dataset, and their performance was inferior to that of the ML models.
Our research also highlights the importance of considering factors such as urban and rural city planning, population density, transportation, and land use when selecting a business site. We believe that our research will make it easier for entrepreneurs and business owners to find the perfect location for their enterprise and will serve as a viable alternative to complex survey data. Furthermore, our work will contribute to the ongoing development of optimal site selection and aid future researchers and policymakers in their efforts to improve the living standards of the area. Our goal is to bring about a change where not only the business owner but also the community can prosper.

Bibliography

[1] A. L. Samuel, “Machine learning,” *The Technology Review*, vol. 62, no. 1, pp. 42–45, 1959.

[2] A. E. Hoerl, R. W. Kannard, and K. F. Baldwin, “Ridge regression: Some simulations,” *Communications in Statistics-Theory and Methods*, vol. 4, no. 2, pp. 105–123, 1975.

[3] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[5] K. Lau and Q. Wu, “Local prediction of non-linear time series using support vector regression,” *Pattern recognition*, vol. 41, no. 5, pp. 1539–1547, 2008.

[6] A. M. Noor, V. A. Alegana, P. W. Gething, A. J. Tatem, and R. W. Snow, “Using remotely sensed night-time light as a proxy for poverty in africa,” *Population Health Metrics*, vol. 6, no. 1, pp. 1–13, 2008.

[7] C. De Mol, E. De Vito, and L. Rosasco, “Elastic-net regularization in learning theory,” *Journal of Complexity*, vol. 25, no. 2, pp. 201–230, 2009.

[8] C. D. Elvidge, P. C. Sutton, T. Ghosh, B. T. Tuttle, K. E. Baugh, B. Bhaduri, and E. Bright, “A global poverty map derived from satellite data,” *Computers & Geosciences*, vol. 35, no. 8, pp. 1652–1660, 2009.

[9] C. Campbell and Y. Ying, “Learning with support vector machines,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 5, no. 1, pp. 1–95, 2011.

[10] V. Henderson, A. Storeygard, and D. N. Weil, “A bright idea for measuring economic growth,” *American Economic Review*, vol. 101, no. 3, pp. 194–99, 2011.

[11] C. R. Genovese, J. Jin, L. Wasserman, and Z. Yao, “A comparison of the lasso and marginal regression,” *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2107–2143, 2012.

[12] J. V. Henderson, A. Storeygard, and D. N. Weil, “Measuring economic growth from outer space,” *American economic review*, vol. 102, no. 2, pp. 994–1028, 2012.

[13] J. Stoler, D. Daniels, J. R. Weeks, D. A. Stow, L. L. Coulter, and B. K. Finch, “Assessing the utility of satellite imagery with differing spatial resolutions for deriving proxy measures of slum presence in accra, ghana,” *GIScience & Remote Sensing*, vol. 49, no. 1, pp. 31–52, 2012.

[14] X. Lin and Y. Zu, *Multi-criteria gis-based procedure for coffee shop location decision*, 2013.

[15] M. H. Satman and M. Altunbey, “Selecting location of retail stores using artificial neural networks and google places api,” *International journal of Statistics and Probability*, vol. 3, no. 1, p. 67, 2014.

[16] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[17] H. Bagan and Y. Yamagata, “Analysis of urban growth and estimating population density using satellite images of nighttime lights and land-use and population data,” *GIScience & Remote Sensing*, vol. 52, no. 6, pp. 765–780, 2015.

[18] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, “Deepsat: A learning framework for satellite imagery,” in *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, 2015, pp. 1–10.

[19] F.-C. Hsu, K. E. Baugh, T. Ghosh, M. Zhizhin, and C. D. Elvidge, “Dmsp-ols radiance calibrated nighttime lights time series with intercalibration,” *Remote Sensing*, vol. 7, no. 2, pp. 1855–1876, 2015.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[22] M. Belgiu and L. Drgu, “Random forest in remote sensing: A review of applications and future directions,” *ISPRS journal of photogrammetry and remote sensing*, vol. 114, pp. 24–31, 2016.

[23] S. Ganguli, J. Dunnmon, and D. Hau, *Predicting food security outcomes using cnns for satellite tasking*, 2016.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] ——, “Identity mappings in deep residual networks,” in *European conference on computer vision*, Springer, 2016, pp. 630–645.

[26] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, “Combining satellite imagery and machine learning to predict poverty,” *Science*, vol. 353, no. 6301, pp. 790–794, 2016.

[27] S. Ramraj, N. Uzir, R. Sunil, and S. Banerjee, “Experimenting xgboost algorithm for prediction and classification of different datasets,” *International Journal of Control Theory and Applications*, vol. 9, no. 40, 2016.

[28] S. Reid, R. Tibshirani, and J. Friedman, “A study of error variance estimation in lasso regression,” *Statistica Sinica*, pp. 35–67, 2016.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[30] D. Arribas-Bel, J. E. Patino, and J. C. Duque, “Remote sensing-based measurement of living environment deprivation: Improving classical approaches with machine learning,” *PLoS one*, vol. 12, no. 5, e0176684, 2017.

[31] B. Babenko, J. Hersh, D. Newhouse, A. Ramakrishnan, and T. Swartz, "Poverty mapping using convolutional neural networks trained on high and medium resolution satellite images, with an application in mexico," *arXiv preprint arXiv:1711.06323*, 2017.

[32] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[33] N. B. Weidmann and S. Schutte, "Using night light emissions for the prediction of local wealth," *Journal of Peace Research*, vol. 54, no. 2, pp. 125–140, 2017.

[34] G. A. Areas, "Gadm database of global administrative areas, version 2.8. 2015," *URL http://www. gadm. org*, 2018.

[35] T. Bilen, M. Erel-Özçevik, Y. Yaslan, and S. F. Oktug, "A smart city application: Business location estimator using machine learning techniques," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, 2018, pp. 1314–1321.

[36] Z. Kolar, H. Chen, and X. Luo, "Transfer learning and deep convolutional neural networks for safety guardrail detection in 2d images," *Automation in Construction*, vol. 89, pp. 58–70, 2018.

[37] M. O. Román, Z. Wang, Q. Sun, V. Kalb, S. D. Miller, A. Molthan, L. Schultz, J. Bell, E. C. Stokes, B. Pandey, *et al.*, "Nasa's black marble nighttime lights product suite," *Remote Sensing of Environment*, vol. 210, pp. 113–143, 2018.

[38] I. F. Shihab, M. M. Oishi, S. Islam, K. Banik, and H. Arif, "A machine learning approach to suggest ideal geographical location for new restaurant establishment," in *2018 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, IEEE, 2018, pp. 1–5.

[39] L. Wang, H. Fan, and Y. Wang, "Site selection of retail shops based on spatial accessibility and hybrid bp neural network," *ISPRS International Journal of Geo-Information*, vol. 7, no. 6, p. 202, 2018.

[40] U. J. Dorji, A. Plangprasopchok, N. Surasvadi, and C. Siripanpornchana, "A machine learning approach to estimate median income levels of sub-districts in thailand using satellite and geospatial data," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, 2019, pp. 11–14.

[41] R. Zhang, B. Li, and B. Jiao, "Application of xgboost algorithm in bearing fault diagnosis," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 490, 2019, p. 072 062.

[42] R. A. Azdy and F. Darnis, "Use of haversine formula in finding distance between temporary shelter and waste end processing sites," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1500, 2020, p. 012 104.

[43] A. Ferlitsch, *Deep Learning Design Patterns: Best Practices and Techniques.* Manning Publications, 2020, ISBN: 1617298263.

[44] F. He, T. Liu, and D. Tao, "Why resnet works? residuals generalize," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 12, pp. 5349–5362, 2020.

[45] J. Ouyang, H. Fan, L. Wang, M. Yang, and Y. Ma, "Site selection improvement of retailers based on spatial competition strategy and a double-channel con-

volutional neural network," *ISPRS International Journal of Geo-Information*, vol. 9, no. 6, p. 357, 2020.

[46] D. Theckedath and R. Sedamkar, "Detecting affect states using vgg16, resnet50 and se-resnet50 networks," *SN Computer Science*, vol. 1, no. 2, pp. 1–7, 2020.

[47] C. Yeh, A. Perez, A. Driscoll, G. Azzari, Z. Tang, D. Lobell, S. Ermon, and M. Burke, "Using publicly available satellite imagery and deep learning to understand economic well-being in africa," *Nature communications*, vol. 11, no. 1, pp. 1–11, 2020.

[48] M. Bansal, M. Kumar, M. Sachdeva, and A. Mittal, "Transfer learning for image classification using vgg19: Caltech-101 image data set," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2021.

[49] N. Dey, Y.-D. Zhang, V. Rajinikanth, R. Pugalenthi, and N. S. M. Raja, "Customized vgg19 architecture for pneumonia detection in chest x-rays," *Pattern Recognition Letters*, vol. 143, pp. 67–74, 2021.

[50] P. Gyimah and R. N. Lussier, "Rural entrepreneurship success factors: An empirical investigation in an emerging market," *Journal of Small Business Strategy*, vol. 31, no. 4, pp. 5–19, 2021.

[51] R. Kumar, "Image classification using network inception-architecture appications," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 10, no. 1, pp. 339–342, 2021.

[52] E. J. Oughton and J. Mathur, "Predicting cell phone adoption metrics using machine learning and satellite imagery," *Telematics and Informatics*, vol. 62, p. 101 622, 2021.

[53] R. Tamilarasi and S. Gopinathan, "Inception architecture for brain image classification," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1964, 2021, p. 072 022.

[54] R. Wightman, H. Touvron, and H. Jégou, "Resnet strikes back: An improved training procedure in timm," *arXiv preprint arXiv:2110.00476*, 2021.

[55] *2020 population distribution in the United States visualization.* [Online]. Available: https://www.census.gov/library/visualizations/2021/geo/population-distribution-2020.html.

[56] *Openstreetmap wiki api v0.6.* [Online]. Available: https://wiki.openstreetmap.org/wiki/API_v0.6.

[57] Rbrundritt, *Imagery api - bing maps.* [Online]. Available: https://learn.microsoft.com/en-us/bingmaps/rest-services/imagery/.

[58] *United states of america - population density.* [Online]. Available: https://data.humdata.org/dataset/worldpop-population-density-for-united-states-of-america.

[59] *Yelp open dataset.* [Online]. Available: https://www.yelp.com/dataset.