A Project Report

on

The New Age of House Hunting: Real Estate Price Prediction

Submitted in partial fulfillment of the
requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr. Kirti Shukla
Associate Professor
Department of Computer Science and Engineering

Submitted By

Varnica Bajaj 18SCSE1010716
Aman Saxena 18SCSE1010502

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

MAY, 2022

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"THE NEW AGE OF HOUSE HUNTING: REAL ESTATE PRICE PREDICTION"** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JANUARY-2022 to MAY2022**, under the supervision of **Dr. Kirti Shukla, Associate Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

VARNICA BAJAJ, 18SCSE1010716

AMAN SAXENA, 18SCSE1010502

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Kirti Shukla

Associate Professor

CERTIFICATE

The Final Project Viva-Voce examination of **Varnica Bajaj: 18SCSE101716, Aman Saxena: 18SCSE1010502** has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

ACKNOWLEDGEMENT

I would like to express my very great appreciation to Galgotias University for the valuable, constructive opportunities and source of path for the planning and development of this project. We are thankful to god for willingness to give so generously to create this project.

I would also like to thank our project supervisor **Dr. Kirti Shukla, Associate Professor** for guiding me and sharing their precious time and knowledge with patience. Without their participation and contribution, this project would not have been successful.

I am profoundly thankful to **Dr. SPS Chauhan (Program Chair and Dean of Computer Science and Engineering)**, Galgotias University, Greater Noida India, for allowing this paper to be my own work.

Finally, I want to express my immense gratitude to my friends and my parents for their endless encouragement and support through all these years. Without whom this project would have been a distant reality.

Thank you.

Abstract

In the world of growing technologies, we see every day that something or the other things have been implemented to make human lives easier and comfortable. Similarly, Machine Learning plays a vital role in every field starting from Face Recognition to Medical Diagnosis and all the fields are benefitted from Machine Learning in recent years. The current machine learning algorithm aids us in improving our results. It works on improving and enhancing things. The real estate market is one of the most price-sensitive in the world, and it is always changing as the prices can vary dramatically depending on a variety of criteria. It is one of the most important domains in which machine learning techniques may be applied to improve and predict costs with great accuracy. As a result, we give several key features to consider when projecting house prices with high accuracy in this research. This research paper examines the application of existing machine learning techniques to the existing dataset, with the goal of implementing this prediction for real-world use by consumers.

Table of Contents

List of Figures

Acronyms

AI	Artificial Intelligence
ML	Machine Learning
RMSE	Root Mean Square Error

CHAPTER-1

Introduction

1.1Introduction

Machine Learning (ML) is an important part of today's business and research. It uses algorithms and neural network models to gradually improve the performance of computer systems. Machine Learning algorithms create a mathematical model using sample data, often known as training data, and use it to make judgments without being explicitly programmed to do so. People and real estate agencies buy and sell houses for a variety of reasons. People buy to live in or as an investment, while agencies buy to run a company. In any case, we believe that everyone should receive precisely what they pay for. Overvaluation and undervaluation in property markets have long been a problem, and there aren't enough detecting tools. A primary pass is given by broad measurements such as house/real-estate price-to-rent ratios. However, in order to make a decision on this matter, an in-depth analysis and judgment are required. This is where machine learning comes in; by training an ML model with hundreds of thousands of data points, a solution can be constructed that is powerful enough to properly anticipate pricing while also catering to the needs of everyone.

The main goal of this work is to take these Machine Learning Techniques and turn them into ML models that can be used to service people. A Buyer's first goal is to find their ideal home with all of the facilities they require. Furthermore, people search for these houses/real estate's with a budget in mind, and there is no guarantee that they will find a product at a fair price. Similarly, a seller seeks a specific number to use as a price tag for the property; this cannot be a wild guess; extensive study is required to arrive at a house value. Furthermore, there is a chance that the product will be underpriced. If the price is forecast for these users, it may be possible for them to obtain estates at their fair market value, not more or less.

Problems of the Previous Models

• When consumers first consider purchasing a home or real estate, they frequently go online to research trends and other relevant information. People do this so that they can find a home that has all they require. People keep track of the prices associated with these residences while performing these tasks. The average person, on the other hand, lacks thorough understanding and correct information on what the true cost should be. This can lead to misinformation since people believe the prices on the internet are accurate.

• When looking for a home, the second thing that comes to mind is contacting numerous estate agents. The issue is that these agents should only be paid a fraction of the money only to look for a house and establish a price for you. People, in most circumstances, believe this price tag because they have no other options. There may be instances where agents and sellers have a hidden deal and the customer is sold an overpriced home without his or her awareness.

• When a person considers selling his or her home, they compare it against hundreds of thousands of other homes that have been listed all over the world. Calculating the price by comparing it to multiple estates takes a long time and has the danger of being inaccurate. CHAPTER-1 Introduction • Large real estate companies must sell a variety of items and must assign staff to manage each of these things. The projection of a price tag is also based on a person, therefore there is opportunity for human mistake. These allocated individuals must also be compensated. However, having a machine crunch the figures for you can save you a lot of time and money while also providing precision that a human cannot.

1.2 Formulation of Problem

We'll be considering a given dataset, and will be working on that only. It would be a model of Supervised Learning as it will involve features and labels. It would fall under the category of Regression task as we'll be working on finding a value rather than classifying it. It would fall under Batch Learning as the dataset is already provided.
Based on the data set containing **listings of apartments** offered for sale in various regions of Russia we have developed a pricing models that could be used to evaluate apartments located in the **city of Saint Petersburg**.
Similar approach could be used to create regression models for other regions.
Our task is to create a model which will predict the price for any new house by looking at the features. Real Estates would use this model to predict house prices in a given area and will invest in the area if it is undervalued.

1.2.1 Tool and Technology Used

Jupyter Notebook is used. It is an open source web application that allows us to create and share documents that contains line code, equations, visualizations and narrative text.

Models used:

- **XGBoost**

  XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

  XGBoost algorithm was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their paper at SIGKDD Conference in 2016 and caught the Machine Learning world by fire. Since its introduction, this algorithm has not only been credited with winning numerous Kaggle competitions but also for being the driving force under the hood for several cutting-edge industry applications. As a result, there is a strong community of data scientists contributing to the XGBoost open source projects with ~350 contributors

and ~3,600 commits on GitHub. The algorithm differentiates itself in the following ways:

A wide range of applications:

1. Can be used to solve regression, classification, ranking, and user-defined prediction problems.
2. Portability: Runs smoothly on Windows, Linux, and OS X.
3. Languages: Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.
4. Cloud Integration: Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

- **LGBM**

It is a gradient boosting framework that makes use of tree based learning algorithms that is considered to be a very powerful algorithm when it comes to computation. It is considered to be a fast processing algorithm.

While other algorithms trees grow horizontally, LightGBM algorithm grows vertically meaning it grows leaf-wise and other algorithms grow level-wise. LightGBM chooses the leaf with large loss to grow. It can lower down more loss than a level wise algorithm when growing the same leaf.

Why is LightGBM popular?

It has become difficult for the traditional algorithms to give results fast, as the size of the data is increasing rapidly day by day. LightGBM is called "**Light**" because of its computation power and giving results faster. It takes **less memory to run** and is able to **deal with large amounts of data**. Most widely used algorithm in Hackathons because the motive of the algorithm is to get good accuracy of results and also brace GPU leaning.

When to use LightGBM?

LightGBM is not for a small volume of datasets. It can easily overfit small data due to its sensitivity. It can be used for data having more than 10,000+ rows. There is no fixed threshold that helps in deciding the usage of LightGBM. It can be used for large volumes of data especially when one needs to achieve a high accuracy.

- **CatBoost**

Catboost is a boosted decision tree machine learning algorithm developed by Yandex. It works in the same way as other gradient boosted algorithms such as XGBoost but provides support out of the box for categorical variables, has a higher level of accuracy without tuning parameters and also offers GPU support to speed up training.

Catboost is used for a range of regression and classification tasks and has been shown to be a top performer on various Kaggle competitions that involve tabular data. Below are a couple of examples of where Catboost has been successfully implemented:

- Cloudflare use Catboost to identify bots trying to target it's users websites.
- Ride hailing service Careem, based in Dubai, use Catboost to predict where it's customers will travel to next.

CatBoost advantages

- CatBoost implements oblivious decision trees (binary tree in which same features are used to make left and right split for each level of the tree) thereby restricting the features split per level to one, which help in decreasing prediction time.
- It handles categorical features effectively by ordered target statistics.
- It is easy to use with packages in R and Python.
- It has effective usage with default parameters thereby reducing the time needed for parameter tuning.

Techniques applied:

- **Data cleaning**

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, the success or failure of a project relies on proper data cleaning. Professional data scientists usually invest a very large portion of their time in this step because of the belief that **"Better data beats fancier algorithms"**.

If we have a well-cleaned dataset, there are chances that we can get achieve good results with simple algorithms also, which can prove very beneficial at times especially in terms of computation when the dataset size is large.

Obviously, different types of data will require different types of cleaning. However, this systematic approach can always serve as a good starting point.

Steps involved in Data Cleaning:

1. **Removal of unwanted observations**

   This includes deleting duplicate/ redundant or irrelevant values from your dataset. Duplicate observations most frequently arise during data collection and Irrelevant observations are those that don't actually fit the specific problem that you're trying to solve.

   - Redundant observations alter the efficiency by a great extent as the data repeats and may add towards the correct side or towards the incorrect side, thereby producing unfaithful results.
   - Irrelevant observations are any type of data that is of no use to us and can be removed directly.

2. **Fixing Structural errors**

   The errors that arise during measurement, transfer of data, or other similar situations are called structural errors. Structural errors include typos in the name of features, the same attribute with a different name, mislabeled classes, i.e. separate classes that should really be the same, or inconsistent capitalization.

   a. For example, the model will treat America and America as different classes or values, though they represent the same value or red, yellow, and red-yellow as different classes or attributes, though one class can be included in the other two classes. So, these are some structural errors that make our model inefficient and give poor quality results.

3. **Managing Unwanted outliers**

Outliers can cause problems with certain types of models. For example, linear regression models are less robust to outliers than decision tree models. Generally, we should not remove outliers until we have a legitimate reason to remove them. Sometimes, removing them improves performance, sometimes not. So, one must have a good reason to remove the outlier, such as suspicious measurements that are unlikely to be part of real data.

4. **Handling missing data**

Missing data is a deceptively tricky issue in machine learning. We cannot just ignore or remove the missing observation. They must be handled carefully as they can be an indication of something important. The two most common ways to deal with missing data are:

a. Dropping observations with missing values.
   i. The fact that the value was missing may be informative in itself.

ii. Plus, in the real world, you often need to make predictions on new data even if some of the features are missing!

b. Imputing the missing values from past observations.

i. Again, "missingness" is almost always informative in itself, and you should tell your algorithm if a value was missing.

ii. Even if you build a model to impute your values, you're not adding any real information. You're just reinforcing the patterns already provided by other features.

Missing data is like missing a puzzle piece. If you drop it, that's like pretending the puzzle slot isn't there. If you impute it, that's like trying to squeeze in a piece from somewhere else in the puzzle.
So, missing data is always an informative and an indication of something important. And we must be aware of our algorithm of missing data by flagging it. By using this technique of flagging and filling, you are essentially allowing the algorithm to estimate the optimal constant for missingness, instead of just filling it in with the mean.

- Filtering outliers

Outliers are those data points which differs significantly from other observations present in given dataset. It can occur because of variability in measurement and due to misinterpretation in filling data points.

How the Outlier are introduced in the datasets?

Most common causes of outliers on a data set:

Data Entry Errors: Human errors such as errors caused during data collection, recording, or entry can cause outliers in data.

Measurement Error **(instrument errors)**: It is the most common source of outliers. This is caused when the measurement instrument used turns out to be faulty.

Experimental errors (data extraction or experiment planning/executing errors)

Intentional (dummy outliers made to test detection methods)

Data processing errors (data manipulation or data set unintended mutations)

Sampling errors (extracting or mixing data from wrong or various sources)

Natural Outlier (not an error, novelties in data): When an outlier is not artificial (due to error), it is a natural outlier. Most of real world data belong to this category.

How to detect Outliers?

Different outlier detection technique

Outlier can be of two types: Univariate and Multivariate. Above, we have discussed the example of univariate outlier. These outliers can be found when we look at distribution of a single variable. Multi-variate outliers are outliers in an n-dimensional space.

a) Hypothesis Testing
b) Z-score method
c) Robust Z-score
d) I.Q.R method
e) Winsorization method (Percentile Capping)
f) DBSCAN Clustering
g) Isolation Forest
h) Linear Regression Models (PCA, LMS)
i) Standard Deviation
j) Percentile
k) Visualizing the data

- **Feature engineering**

The feature engineering pipeline is the preprocessing steps that transform raw data into features that can be used in machine learning algorithms, such as predictive models. Predictive models consist of an outcome variable and predictor variables, and it is during the feature engineering process that the most useful predictor variables are created and selected for the predictive model. Automated feature engineering has been available in some machine learning software since 2016. Feature engineering in ML consists of four main steps: Feature Creation, Transformations, Feature Extraction, and Feature Selection.

Feature engineering consists of creation, transformation, extraction, and selection of features, also known as variables, that are most conducive to creating an accurate ML algorithm. These processes entail:

- **Feature Creation:**

Creating features involves identifying the variables that will be most useful in the predictive model. This is a subjective process that requires human intervention and creativity. Existing features are mixed via addition, subtraction, multiplication, and ratio to create new derived features that have greater predictive power.

- **Transformations:**
  Transformation involves manipulating the predictor variables to improve model performance; e.g. ensuring the model is flexible in the variety of data it can ingest; ensuring variables are on the same scale, making the model easier to understand; improving accuracy; and avoiding computational errors by ensuring all features are within an acceptable range for the model.

- **Feature Extraction:**
  Feature extraction is the automatic creation of new variables by extracting them from raw data. The purpose of this step is to automatically reduce the volume of data into a more manageable set for modeling. Some feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis.

- **Feature Selection:**
  Feature selection algorithms essentially analyze, judge, and rank various features to determine which features are irrelevant and should be removed, which features are redundant and should be removed, and which features are most useful for the model and should be prioritized.

  Steps in Feature Engineering

  The art of feature engineering may vary among data scientists, however steps for how to perform feature engineering for most machine learning algorithms include the following:

- **Data Preparation:**
  This preprocessing step involves the manipulation and consolidation of raw data from different sources into a standardized format so that it can be used in a model. Data preparation may entail data augmentation, cleaning, delivery, fusion, ingestion, and/or loading.
- **Exploratory Analysis:**
  This step is used to identify and summarize the main characteristics in a data set through data analysis and investigation. Data science experts use data visualizations to better understand how best to manipulate data sources, to determine which statistical techniques are most appropriate for data analysis, and for choosing the right features for a model.
- **Benchmark:**
  Benchmarking is setting a baseline standard for accuracy to which all variables are compared. This is done to reduce the rate of error and improve a model's predictability. Experimentation, testing and optimizing metrics for benchmarking is performed by data scientists with domain expertise and business users.

CHAPTER-2

Literature Survey

In the twenty-second century, real estate has evolved into much more than a need; it now signifies much more. Not just for those interested in purchasing Real Estate, but also for those who sell it. Real Estate Property, is not only a man's essential need, but it also signifies his wealth and prestige today. Real estate investment appears to be profitable in general since property values do not depreciate rapidly. Changes in real estate prices can have an impact on a variety of people, including homeowners, bankers, policymakers, and others. Real estate investment appears to be an appealing investment option. As a result, estimating the value of real estate is a difficult task.
As a result, anticipating the value of real estate is an essential economic indicator.

Every firm in the real estate industry today is working hard to gain a competitive advantage over its competitors. There is a need to make the process easier for a regular person while yet achieving the greatest results.

Supervised learning is when we teach or train a computer using well-labeled data, which means that part of the data has already been tagged with the correct answer. Following that, the computer is given a fresh collection of examples so that the supervised learning algorithm may analyze the training data and produce a correct result from the labeled data. The analysis of land price trends is deemed necessary in order to support urban planning decisions. The real estate market is a stochastic process that is inherently unstable. Investors make judgments based on market movements in order to maximise their profits. For their decision-making, developers want to know about future trends. A considerable amount of data that determines land price is required for analysis, modelling, and forecasting in order to effectively estimate property prices and future trends.

The elements that influence land prices must be investigated, and their impact on price must be calculated. A review of previous data should be considered. It can be concluded that developing a simple linear mathematical relationship for these time-series data is not feasible for predicting. As a result, it became critical to develop a non-linear model that can accurately fit the data characteristics in order to analyse and forecast future trends. Because the real estate industry is rapidly evolving, the analysis and forecasting of land values using mathematical modelling and other scientific methodologies is a pressing requirement for all parties involved. The rise in population and industrial activity can be attributed to a number of sources, the most significant of which is the recent surge in the knowledge sector, namely Information Technology (IT) and ITenabled services. Land demand began to rise, and housing and real estate activities began to boom. To make space for multistory and highrise buildings, the barren regions and paddy fields were demolished. Investments began to stream into the real estate industry, and the price of land did not follow a consistent trend over time. The need to forecast land price trends was felt by everyone in the business, including the government, regulatory organisations, financial institutions, developers, and investors.

CHAPTER-3

Functionality/Working of Project

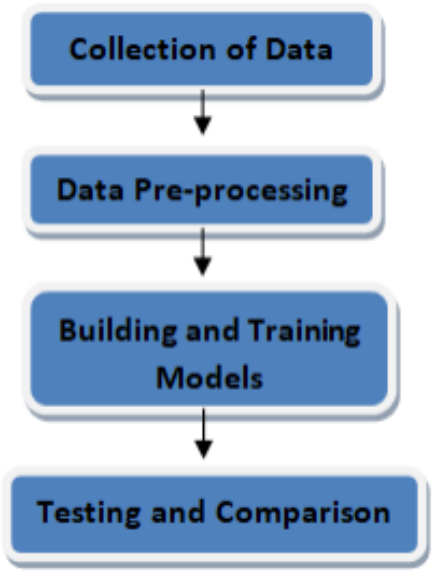

Fig.1 Diagram Flow Research

1. Data collection

Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained. Following are some of the problems that can arise in data collection:

- Inaccurate data. The collected data could be unrelated to the problem statement.
- Missing data. Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.
- Data imbalance. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.
- Data bias. Depending on how the data, subjects and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age or region, for example. Data bias is difficult to detect and remove.
- Several techniques can be applied to address those problems:

- Pre-cleaned, freely available datasets. If the problem statement (for example, image classification, object recognition) aligns with a clean, pre-existing, properly formulated dataset, then take advantage of existing, open-source expertise.
- Web crawling and scraping. Automated tools, bots and headless browsers can crawl and scrape websites for data.

- Private data. ML engineers can create their own data. This is helpful when the amount of data required to train the model is small and the problem statement is too specific to generalize over an open-source dataset.
- Custom data. Agencies can create or crowdsource the data for a fee.

2. Data preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

It involves below steps:

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

3. Building and Training Model

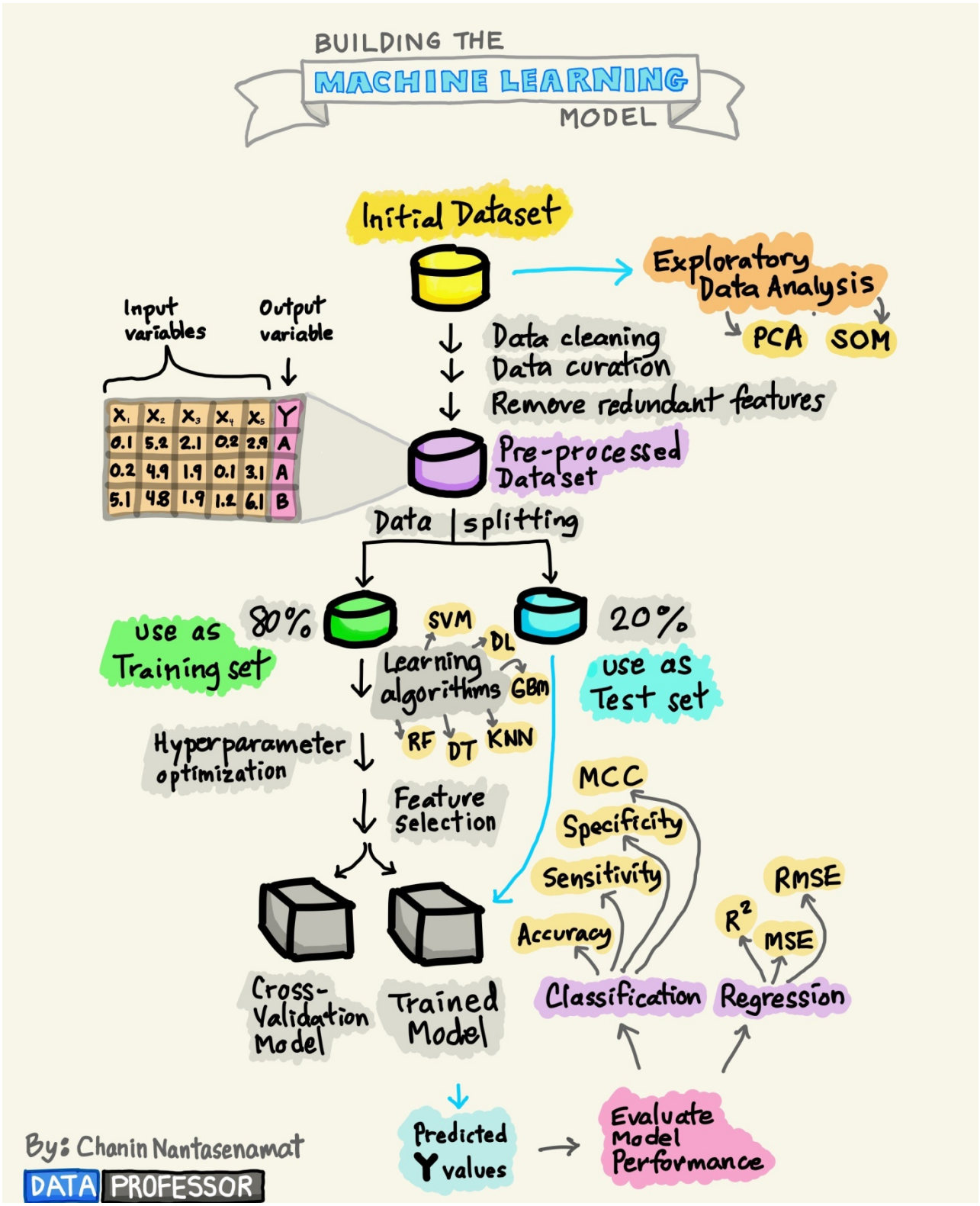

Fig 2. Machine Learning Model

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The

training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model. This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model.

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.

Supervised learning is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done based on the deviation of the processed result from the documented result when the inputs are fed into the model.

Unsupervised learning involves determining patterns in the data. Additional data is then used to fit patterns or clusters. This is also an iterative process that improves the accuracy based on the correlation to the expected patterns or clusters. There is no reference output dataset in this method.

Dataset

The dataset has 13 fields.

- date - date of publication of the announcement;
- time - the time when the ad was published;
- geo_lat - Latitude
- geo_lon - Longitude
- region - Region of Russia. There are 85 subjects in the country in total.
- building_type - Facade type. 0 - Other. 1 - Panel. 2 - Monolithic. 3 - Brick. 4 - Blocky. 5 - Wooden
- object_type - Apartment type. 1 - Secondary real estate market; 2 - New building;
- level - Apartment floor
- levels - Number of storeys
- rooms - the number of living rooms. If the value is "-1", then it means "studio apartment"
- area - the total area of the apartment
- kitchen_area - Kitchen area
- price - Price. in rubles

Features and data types

- **Categorical features:**
    - Region (numerically encoded geographical area, identifies either a large city like Moscow or Saint Petersburg or a federal region / district)

    - Building type (numerically encoded type of the building where the apartment is located)
    - Object type (apartment type, where 1 stands for secondary real estate market, 11 - new building)
- **Numerical features:**
    - Area (total floor area of the apartment in sq. meters)
    - Kitchen area (kitchen area in sq. meters)
    - Rooms (number of rooms in the apartment, -1 stands for studios with open-space layout)
    - Level (floor the apartment is located on, could be treated as a categorical feature as well)
    - Levels (total number of storeys in the building)
- **Geospatial features:**
    - Latitude - geographical coordinate of the property
    - Longitude - geographical coordinate of the property
- **Temporal features:**
    - Date (date the listing was published)
    - Time (exact time the listing was published)

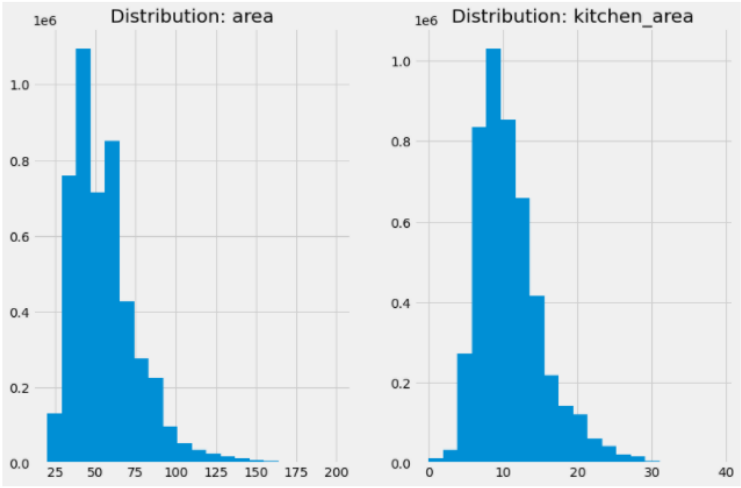

Fig. 3 Distribution Area

```
pos = 0
for pos, feature in enumerate(['level', 'levels']):
    levels = data[feature].value_counts()
    sp = plt.subplot(1, 2, pos + 1)
    plt.bar(levels.index, levels.values)
    plt.title(feature)
plt.show()
```

Fig. 4 Code of Level and Levels

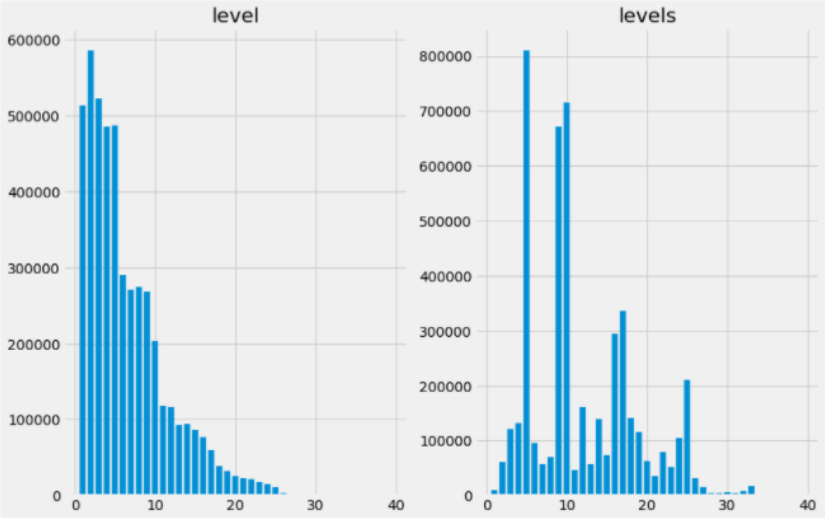

Fig. 5 Graph of Level and Levels

```
# Regions are encoded with numeric IDs.
regions = data['region'].value_counts()
print(regions.head(10))

plt.hist(regions.values, bins=5)
plt.title('Listings by Region')
plt.show()
```

Fig. 6 Code of Listings by Region

```
9654    812372
2843    575693
81      480497
2661    453621
3       411225
2922    213368
6171    205363
3230    196428
3991    132777
5282    103757
Name: region, dtype: int64
```

Fig. 7 Regions data printed

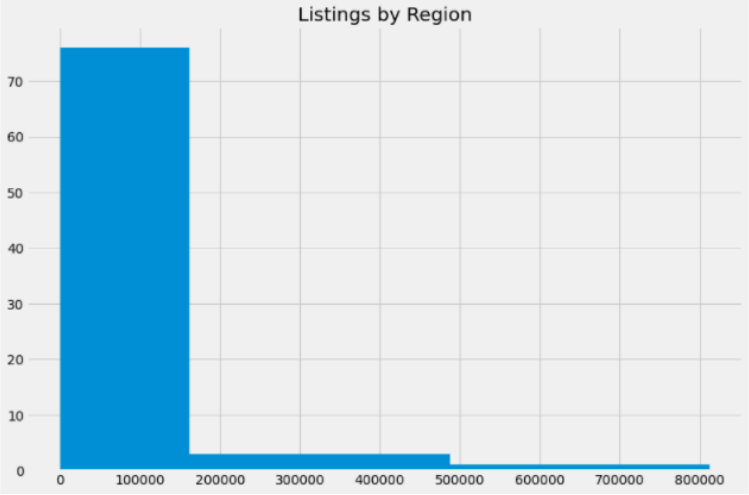

Fig. 8 Listings by Region

```
correlation = data.corr()
ax = sns.heatmap(correlation, center=0, cmap='RdBu_r')
l, r = ax.get_ylim()
ax.set_ylim(l + 0.5, r - 0.5)
plt.yticks(rotation=0)
plt.title('Correlation Matrix')
plt.show()
```

Fig. 9 Code of Correlation Matrix

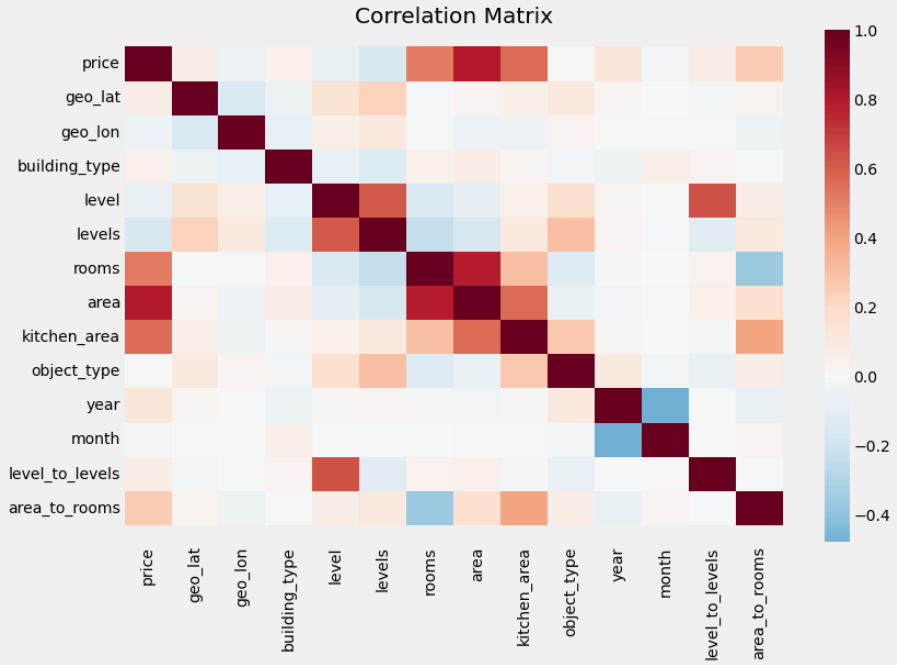

Fig. 10 Correlation Matrix

```
scores = []

for train_index, test_index in kf.split(data, y):

    x_train, x_test = data.iloc[train_index, :], data.iloc[test_index, :]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    model_xgb = XGBRegressor(objective='reg:squarederror')

    model_xgb.fit(x_train, y_train, eval_set=[(x_test, y_test)],
                  eval_metric='rmse', early_stopping_rounds=50,
                  verbose=0)

    scores.append(model_xgb.best_score)
    model_xgb.save_model(f'xgboost{len(scores)}.bin')

print('XGBoost average RMSE:', np.mean(scores))
for i, score in enumerate(scores):
    print(f'Model {i} RMSE: {score}')
```

Fig. 11 Code of XGBoost Average RMSE

```
XGBoost average RMSE: 1796384.875
Model 0 RMSE: 1819126.875
Model 1 RMSE: 1814934.125
Model 2 RMSE: 1779928.875
Model 3 RMSE: 1794775.875
Model 4 RMSE: 1773158.625
```

Fig. 12 XGBoost Average RMSE

```python
# Display feature importance.
importance = pd.DataFrame({
    'features': data.columns,
    'importance': model_xgb.feature_importances_
})
importance.sort_values(by='importance', inplace=True)

plt.figure(figsize=(12, 16))
plt.barh(importance['features'], importance['importance'])
plt.title('XGBoost Feature Importance')
plt.show()
```

Fig. 13 Code of XGBoost Feature Importance

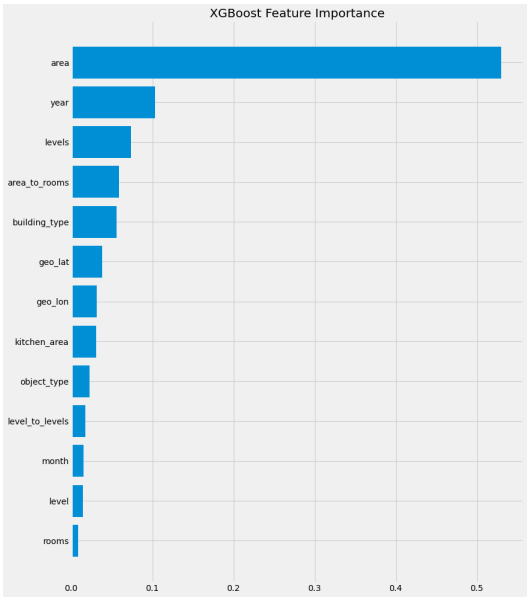

Fig. 14 XGBoost Feature Importance

```
scores = []

for train_index, test_index in kf.split(data, y):

    x_train, x_test = data.iloc[train_index, :], data.iloc[test_index, :]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    model_lgb = LGBMRegressor(objective='regression', metrics='rmse')

    model_lgb.fit(x_train, y_train, eval_set=(x_test, y_test),
                  eval_metric='rmse', early_stopping_rounds=50,
                  categorical_feature=['building_type', 'object_type', 'month'],
                  verbose=0)

    scores.append(model_lgb.best_score_['valid_0']['rmse'])
    model_lgb.booster_.save_model(f'lgbm{len(scores)}.txt',
                                  num_iteration=model_lgb.best_iteration_)

print('LGBM average RMSE:', np.mean(scores))
for i, score in enumerate(scores):
    print(f'Model {i} RMSE: {score}')
```

Fig. 15 Code of LGBM Average RMSE

```
LGBM average RMSE: 1944575.1940743965
Model 0 RMSE: 1967774.5116417012
Model 1 RMSE: 1970392.6348458976
Model 2 RMSE: 1924283.2638095245
Model 3 RMSE: 1940086.2632823393
Model 4 RMSE: 1920339.2967925186
```

Fig. 16 LGBM Average RMSE

```
importance = pd.DataFrame({
    'features': data.columns,
    'importance': model_lgb.feature_importances_
})
importance.sort_values(by='importance', inplace=True)

plt.figure(figsize=(12, 16))
plt.barh(importance['features'], importance['importance'])
plt.title('LGBM Feature Importance')
plt.show()
```

Fig. 17 Code of LGBM Feature Importance

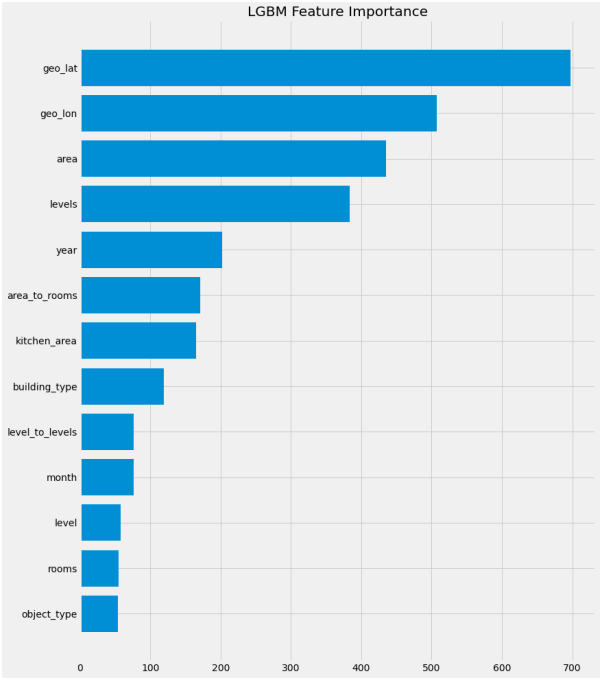

Fig. 18 LGBM Feature Importance

```
scores = []

for train_index, test_index in kf.split(data, y):

    x_train, x_test = data.iloc[train_index, :], data.iloc[test_index, :]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    model_cb = CatBoostRegressor(eval_metric='RMSE',
                                 cat_features=['building_type', 'object_type', 'month'])
    model_cb.fit(x_train, y_train, eval_set=[(x_test, y_test)],
                 early_stopping_rounds=20, use_best_model=True,
                 verbose=0)

    scores.append(model_cb.best_score_['validation']['RMSE'])
    model_cb.save_model(f'catboost{len(scores)}.cbm')

print('Average RMSE:', np.mean(scores))
for i, score in enumerate(scores):
    print(f'Model {i} RMSE: {score}')
```

Fig. 19 Code of CatBoost Average RMSE

```
Average RMSE: 1735834.5166338547
Model 0 RMSE: 1763860.722312027
Model 1 RMSE: 1743654.0839787482
Model 2 RMSE: 1718068.9236082511
Model 3 RMSE: 1735937.805302658
Model 4 RMSE: 1717651.0479675892
```

Fig. 20 CatBoost Average RMSE

```
# Display feature importance.
importance = pd.DataFrame({
    'features': data.columns,
    'importance': model_cb.feature_importances_
})
importance.sort_values(by='importance', inplace=True)

plt.figure(figsize=(12, 16))
plt.barh(importance['features'], importance['importance'])
plt.title('CatBoost Feature Importance')
plt.show()
```

Fig. 21 Code of CatBoost Feature Importance

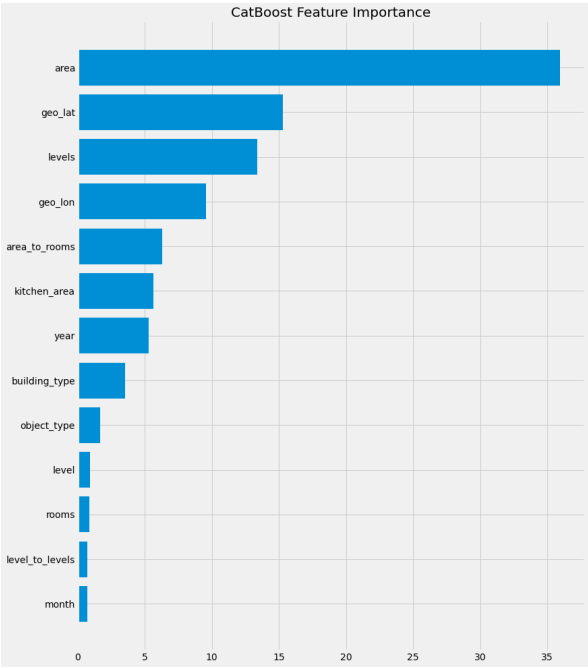

Fig. 22 CatBoost Feature Importance

CHAPTER-4

Results and Discussion

On comparing the RMSE of all the three models, we found that CatBoost has lesser RMSE, so it's the most appropriate model for this dataset. CatBoost provides great results with default parameters.

```
XGBoost average RMSE: 1796384.875
Model 0 RMSE: 1819126.875
Model 1 RMSE: 1814934.125
Model 2 RMSE: 1779928.875
Model 3 RMSE: 1794775.875
Model 4 RMSE: 1773158.625
```

```
LGBM average RMSE: 1944575.1940743965
Model 0 RMSE: 1967774.5116417012
Model 1 RMSE: 1970392.6348458976
Model 2 RMSE: 1924283.2638095245
Model 3 RMSE: 1940086.2632823393
Model 4 RMSE: 1920339.2967925186
```

```
Average RMSE: 1735834.5166338547
Model 0 RMSE: 1763860.722312027
Model 1 RMSE: 1743654.0839787482
Model 2 RMSE: 1718068.9236082511
Model 3 RMSE: 1735937.805302658
Model 4 RMSE: 1717651.0479675892
```

CHAPTER-5

Conclusion

Conclusion and Future Scope

Every person's dream is to possess their own home. We want individuals to acquire houses and real estate at their rightful pricing using this proposed approach, and we don't want them to be duped by shady agents looking for a quick buck. Furthermore, this model will assist large corporations by providing precise projections that will allow them to determine prices and save them a great deal of time and money. The essence of the market is correct real estate values, which we seek to secure with this approach. The system is capable of self-training and price prediction using the raw data presented to it.

After reviewing various research papers as well as numerous blogs and articles, a set of algorithms was chosen that could be used on both of the model's datasets. After extensive testing and training, it was established that the CatBoost Algorithm produced the greatest results of all the algorithms as it has less errors and was more accurate. The system was capable of handling massive amounts of data and was powerful enough to predict the prices of numerous houses with diverse features. The method is simple to use and saves time

REFRENCES

1. *Dabreo, S. et al. (2021) "Real Estate Price Prediction", International Journal of Engineering Research & Technology, 10(4), p. Available at: https://www.ijert.org/real-estate-price-prediction (Accessed: 16 December 2021).*
2. *Truong, Q. et al. (2020) "Housing Price Prediction via Improved Machine Learning Techniques", Procedia Computer Science, 174, pp. 433-442. doi: 10.1016/j.procs.2020.06.111.*
3. *Kuvalekar, A. et al. (2020) "House Price Forecasting Using Machine Learning", SSRN Electronic Journal. doi: 10.2139/ssrn.3565512.*
4. *Real Estate Value Prediction Using Linear Regression (2021). Available at: https://ieeexplore.ieee.org/document/8697639 (Accessed: 16 December 2021).*
5. *(2021) Ijitee.org. Available at: https://www.ijitee.org/wp-content/uploads/papers/v8i9/I7849078919.pdf (Accessed: 16 December 2021).*
6. *(2021) Ijmtst.com. Available at: https://www.ijmtst.com/volume6/issue12/20.IJMTST0612058.pdf (Accessed: 16 December 2021).*