PROJECT AND TERM PAPER

REAL ESTATE PRICE PREDICTION USING MACHINE LEARNING

NAME – Shourya Bhattacharyya

ROLL – 002111701070

YEAR – 3rd

SEMESTER – 2nd

DEPARTMENT OF PRODUCTION ENGINEERING

Acknowledgement

The success and final outcome of this report on term paper required a lot of guidance and assistance and I feel extremely privileged to have received this immense amount of support from the conception of this term paper up until its successful completion. All I have done is only due to such supervision and assistance and through this section I wish to express my outmost gratitude to everyone involved.

I respect and thank **Prof Bijan Sarkar**, for providing me an opportunity to do the term paper work in the field of **"Real Estate Price Prediction Using Machine Learning "** and giving me all support and guidance, which helped me to complete the project in time. I am extremely thankful to him for taking time out of his busy schedule to assist me.

I am also grateful to the rest of the faculty and staff in the Department of Production Engineering, for providing me the necessary resources, opportunities and support.

Finally, I would like to express my heartfelt thanks to my family and friends, for their constant support, understanding and encouragement throughout the time it took to complete the term paper.

Shourya Bhattacharyya

B. Prod.E - III

ABSTRACT

This paper explores the development of a machine learning model for predicting real estate prices. The model leverages supervised learning techniques to identify relationships between various housing characteristics and their corresponding sale prices.

We detail the process of data acquisition, pre-processing, and feature engineering to prepare the data for model training. Several machine learning algorithms, including Linear Regression, Decision Tree, Random forest, are evaluated for their effectiveness in predicting house prices.

The performance of the models is compared using metrics like mean squared error and R-squared. The final model selection is based on the algorithm achieving the optimal balance between accuracy and generalizability.

The developed model can be a valuable tool for various stakeholders in the real estate market. It can assist homeowners in estimating the market value of their property, guide realtors in pricing strategies, and inform potential buyers about fair market value.

Contents

Chapter 1: Introduction

1.1. Background: Significance of Real Estate Price Prediction

In the real estate market, accurate price prediction is a cornerstone for informed decision-making. Buyers can avoid overpaying and sellers can optimize their returns. Increased transparency through objective pricing information benefits both parties, leading to a more efficient market. Furthermore, real estate valuations by appraisers and lenders can be bolstered by this data-driven approach. Ultimately, accurate price prediction serves as a valuable tool not only for individual stakeholders, but also for policymakers and urban planners who rely on market trends to allocate resources effectively.

1.2. Limitations of Traditional Methods

Traditional real estate price prediction methods, often relying on appraiser experience or simple formulas that consider a limited set of factors, have inherent limitations. These methods can be subjective, as appraiser judgment can introduce bias. Additionally, the lack of transparency in how valuations are reached can leave buyers, sellers, and others questioning the rationale behind the price. Furthermore, traditional methods struggle to capture the complex interplay between various factors, such as market trends, local amenities, and property condition, which can significantly influence price. This limitation can lead to inaccurate valuations, particularly in dynamic markets where these factors are constantly evolving. Finally, traditional methods are primarily focused on point-in-time valuations and may not be well-suited for predicting future price trends or market fluctuations. These limitations highlight the need for a more data-driven and objective approach to real estate price prediction.

1.3. Machine Learning as a Promising Approach

Machine learning (ML) presents a compelling alternative to address the limitations of traditional real estate price prediction methods. Unlike methods reliant on appraiser judgment or limited data sets, ML algorithms can be trained on vast amounts of real estate data. This allows them to analyze the intricate interplay between numerous factors, such as property characteristics, market trends, and local dynamics, that influence price. This data-driven approach has the potential to overcome subjectivity and offer more objective, accurate, and nuanced price predictions. Furthermore, the ability of ML algorithms to identify patterns in complex datasets can potentially capture the evolving nature of the real estate market, leading to improved accuracy, particularly in dynamic environments. Beyond point-in-time valuations, some ML techniques can even be utilized to forecast future price trends, providing valuable insights for informed decision-making by various stakeholders in the real estate market.

Chapter 2: Literature Review

2.1. Applications of Machine Learning in Real Estate

- **Real Estate Valuation:** As discussed earlier, ML algorithms are increasingly used for real estate price prediction, offering a data-driven approach to property valuation.
- **Property Recommendation Systems:** ML can personalize the real estate search experience by recommending properties that align with a user's preferences and budget. This can be implemented in real estate websites or mobile apps.
- **Market Trend Analysis:** ML algorithms can analyse vast amounts of data, including economic indicators, sales records, and demographic information, to identify trends and predict future market movements. This information is valuable for investors and developers.
- **Fraud Detection:** ML models can be trained to identify suspicious activity in real estate transactions, potentially helping to prevent fraud and protect buyers and sellers.
- **Risk Assessment:** By analysing property data and historical trends, ML models can assess the risk associated with a particular investment, aiding real estate investors in making informed decisions.

- **Property Management Optimization:** ML can be used to optimize various aspects of property management, such as tenant screening, rent forecasting, and predictive maintenance for buildings.

2.2. Existing Research on Real Estate Price Prediction with ML Algorithms

This paper will provide a critical review of the current research landscape in the field of machine learning (ML) applications for real estate price prediction. This review will explore existing studies that have employed various ML algorithms for this purpose. The focus will be on summarizing the chosen algorithms, the data sources utilized in these studies, and the key findings gleaned from their analyses. Furthermore, the section will delve into a comparative analysis of the performance exhibited by different commonly used ML algorithms in the context of real estate price prediction. This analysis will highlight the relative strengths and weaknesses of each approach, such as linear regression, random forest, support vector machines (SVM), and XGBoost. If the research undertaken for this paper focuses on a specific ML algorithm, this section can provide a more in-depth examination of its application in real estate price prediction. This would involve discussing relevant research that utilizes this chosen algorithm and critically analyzing its advantages and limitations for this specific task. Finally, the section will conclude by identifying any areas where existing research might be lacking or inconclusive. This could encompass the use of specific data sources, the exploration of less common ML algorithms, or the application of these models in under-studied geographic regions. By providing a comprehensive review of existing research, the paper will demonstrate a strong understanding of the field and pave the way for presenting the novel contribution of this particular study in the following sections.

2.3. Comparative Analysis of Different ML Techniques

This section analyzes common machine learning algorithms for real estate price prediction. We explore Linear Regression, Decision Tree Regression, and Random Forest Regression. Linear regression is a good starting point for its interpretability and efficiency, but it assumes linear relationships which may not hold in real estate data. Decision trees and random forests can handle non-linearity but are less interpretable. The best choice depends on your data and priorities. Consider data complexity, interpretability needs, and computational resources. We also mention Support Vector Machines and XGBoost as additional options for high-dimensional data or aiming for maximum accuracy.

Chapter 3: Methodology

3.1. Data Acquisition and Preprocessing

It meticulously details data acquisition and preprocessing. It outlines the data source (public datasets, real estate websites, etc.) and information type (property listings, sales data). Feature selection, engineering (one-hot encoding, scaling), and missing value handling approaches will be explained, ensuring transparency and replicability of the research.

3.1.1. Data Source Description

The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling

- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   CRIM      506 non-null    float64
 1   ZN        506 non-null    float64
 2   INDUS     506 non-null    float64
 3   CHAS      506 non-null    int64  
 4   NOX       506 non-null    float64
 5   RM        505 non-null    float64
 6   AGE       502 non-null    float64
 7   DIS       506 non-null    float64
 8   RAD       506 non-null    int64  
 9   TAX       506 non-null    int64  
 10  PTRATIO   506 non-null    float64
 11  B         506 non-null    float64
 12  LSTAT     506 non-null    float64
 13  MEDV      506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

3.1.2. Research Flow Diagram and Architecture

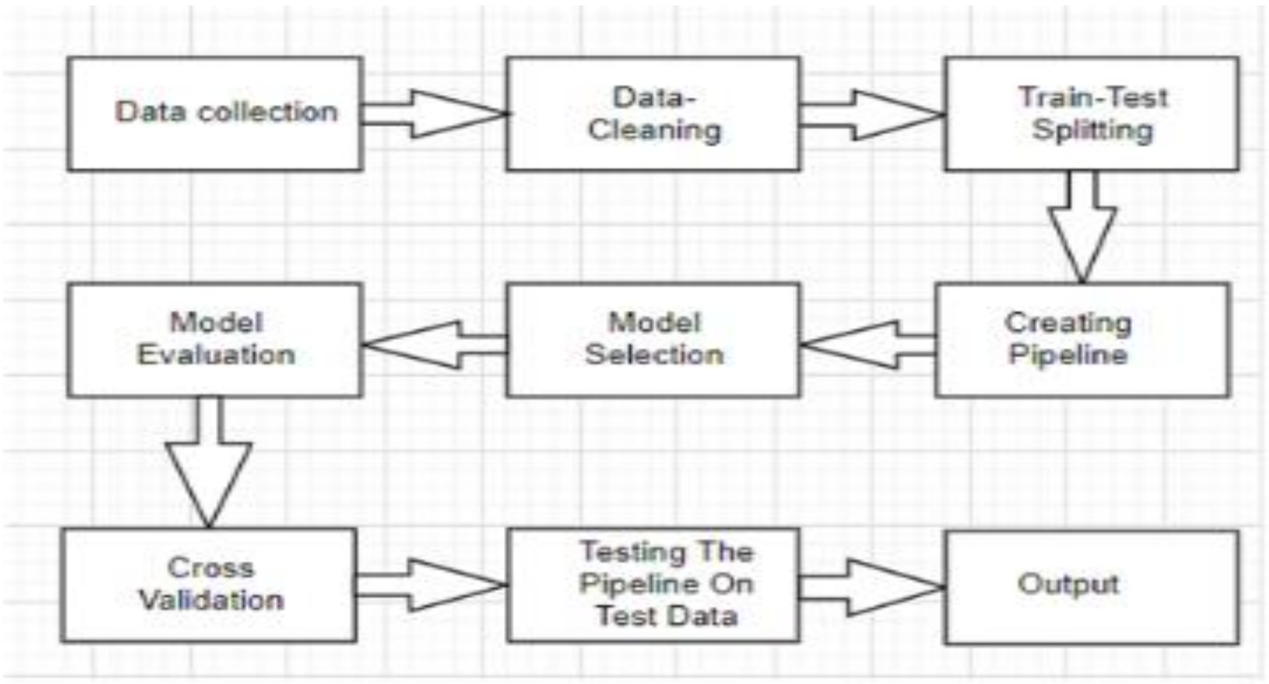
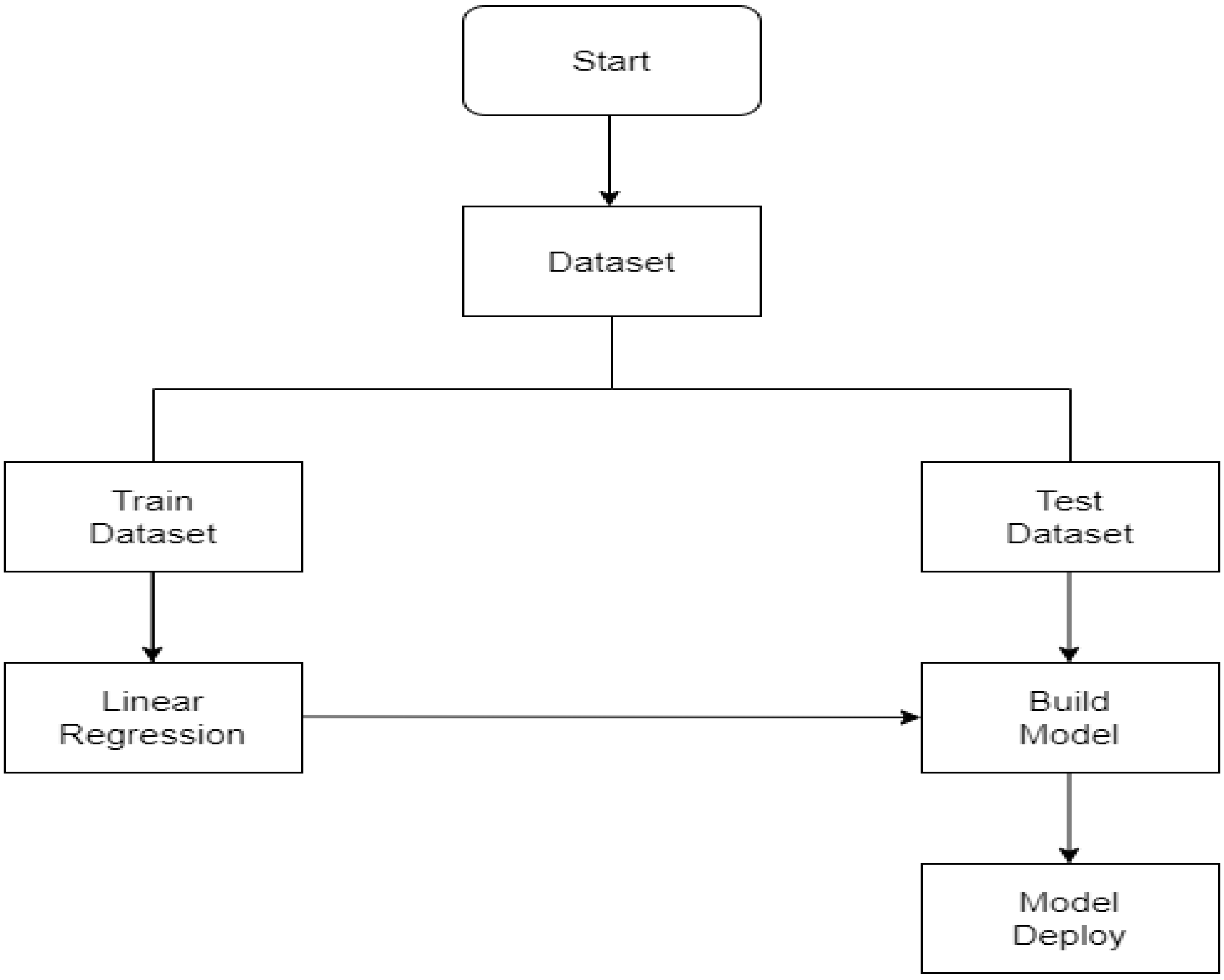


Fig 1. Research Flow Diagram



3.1.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial initial step in data science projects. It involves analyzing and visualizing data to understand its key characteristics, uncover patterns, and identify relationships between variables refers to the method of studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables. EDA is normally carried out as a preliminary step before undertaking extra formal statistical analyses or modelling.

Key aspects of EDA include:

- **Correlation Analysis**: Checking the relationships between variables to understand how they might affect each other. This includes computing correlation coefficients and creating correlation matrices.

```
corr_matrix = housing.corr()
corr_matrix['MEDV'].sort_values(ascending=False)
```

```
MEDV        1.000000
RM          0.680723
B           0.361761
ZN          0.339741
DIS         0.240451
CHAS        0.205066
AGE        -0.363877
RAD        -0.374693
CRIM       -0.393715
NOX        -0.422873
TAX        -0.456657
INDUS      -0.473516
PTRATIO    -0.493534
LSTAT      -0.740494
Name: MEDV, dtype: float64
```

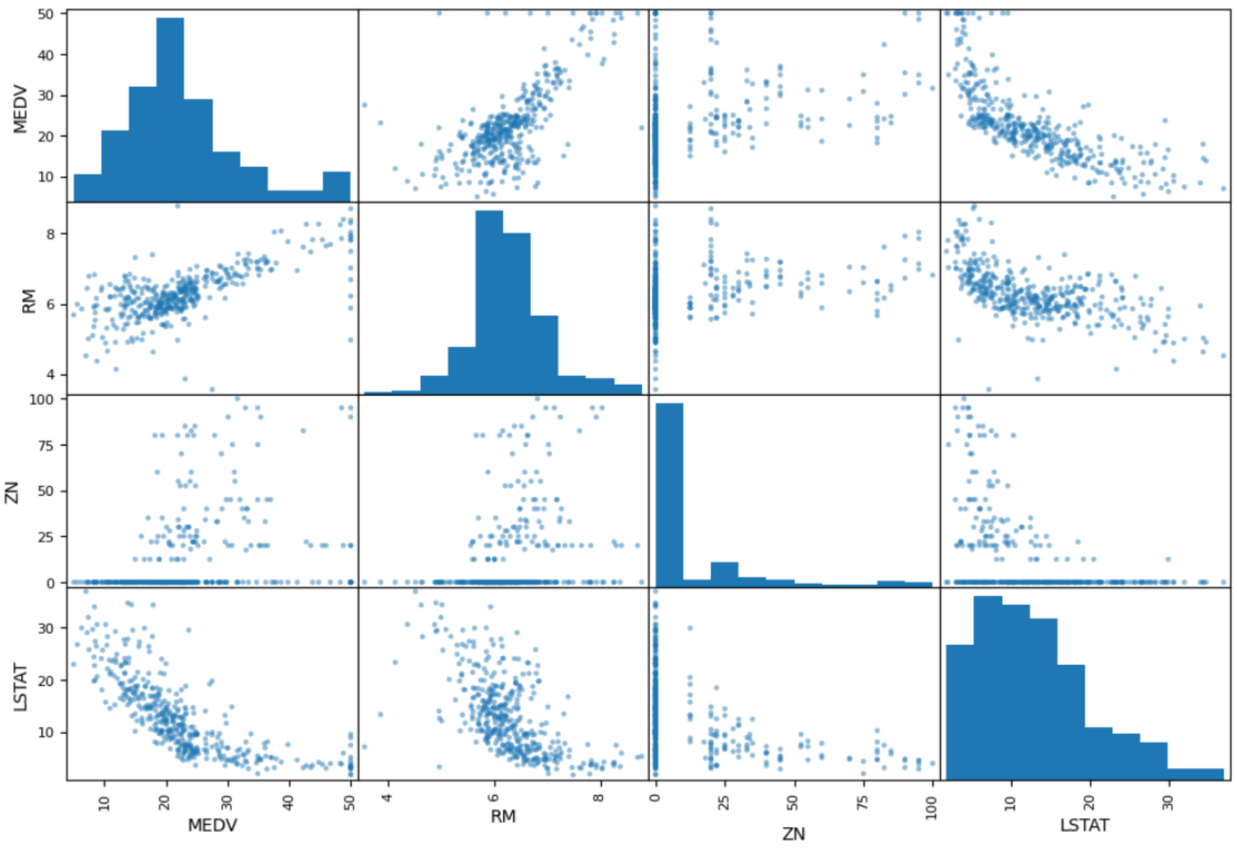
- **Data Cleaning**: It is the way to improvise the data or remove incorrect, corrupted or irrelevant data. As in our dataset, there are some columns that are not important and irrelevant for the model training. So, we can drop that column before training. There are 2 approaches to dealing with empty/null values
  - We can easily delete the column/row (if the feature or record is not much important).
  - Filling the empty slots with mean/mode/0/NA/etc. (depending on the dataset requirement).

```
median = housing["RM"].median()
```

```
housing["RM"].fillna(median)
```

```
254    6.108
348    6.635
476    6.484
321    6.376
326    6.312
       ...
155    6.152
423    6.103
98     7.820
455    6.525
216    5.888
Name: RM, Length: 404, dtype: float64
```

- **Graphical Representations**: Utilizing charts such as histograms, box plots, scatter plots, and bar charts to visualize relationships within the data and distributions of variables.



3.1.4 Train – Test Splitting of Data

The training set teaches the model, while the testing set evaluates its performance on unseen data. However, this basic approach can lead to overfitting. To address this, we use train-validation-test split. Here, a validation set is added to fine-tune the model and prevent overfitting before final evaluation on the unseen test set. This three-way split ensures a more robust and generalizable model.

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}\n")
```

```
Rows in train set: 404
Rows in test set: 102
```

3.1.5 Creating a Pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline = Pipeline([
    ('imputer',SimpleImputer(strategy="median")),
    ('std_scalar', StandardScaler()),
])
```

```
housing_num_tr = my_pipeline.fit_transform(housing_tr)
```

```
housing_num_tr.shape
```

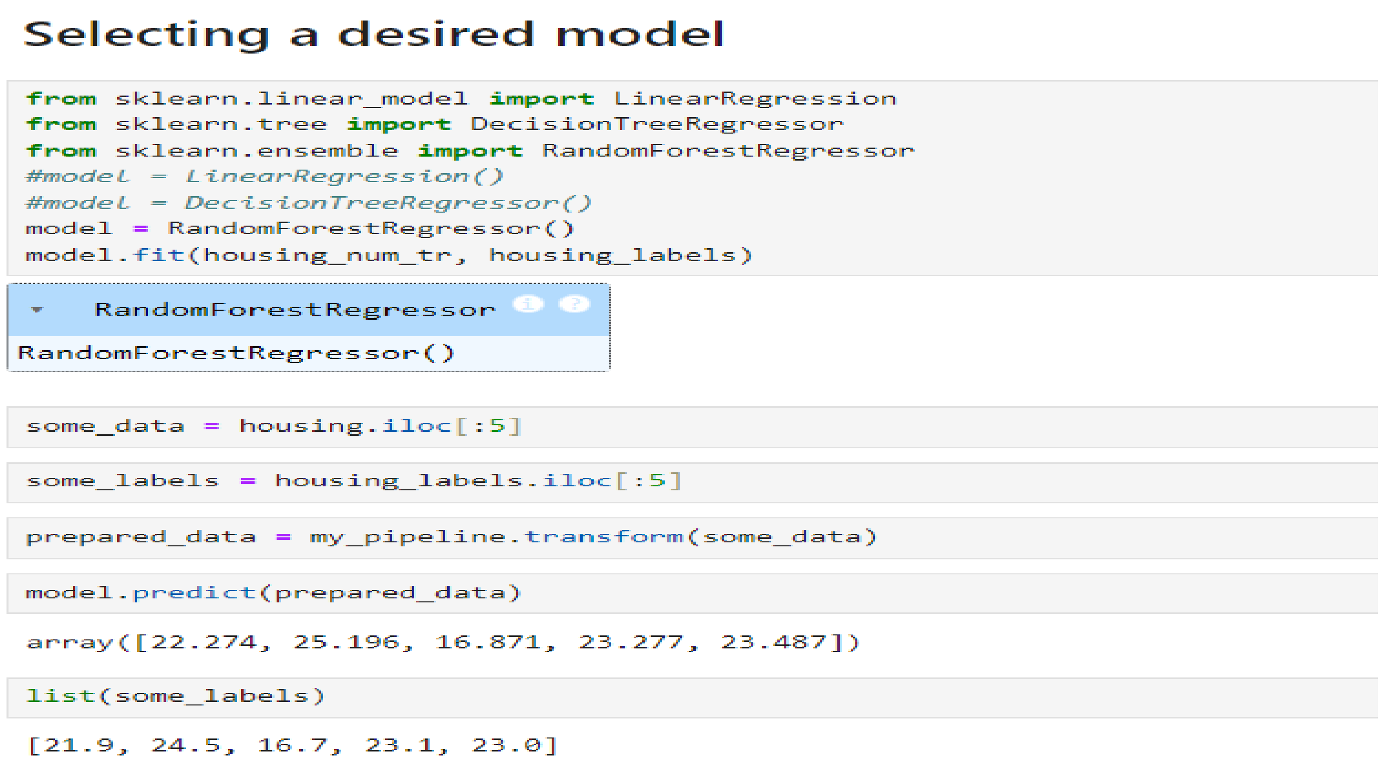
```
(404, 13)
```

3.2. Machine Learning Model Selection

This section dives deeper into the specific machine learning (ML) algorithms commonly used for real estate price prediction. Here's a breakdown of three prominent algorithms, along with their strengths and weaknesses in this context:

- **Linear Regression:**
    - **Description:** A foundational algorithm that establishes a linear relationship between a property's price (dependent variable) and its various features (independent variables) like square footage, number of bedrooms, and location.
    - **Strengths:** Easy to interpret, computationally efficient, provides a baseline for comparison with more complex models.
    - **Weaknesses:** Assumes a linear relationship between features and price, may not capture complex non-linear relationships present in real estate data.
- **Decision Tree Regressor:**
    - **Description:** A tree-like model where each node represents a split on a property feature (e.g., size > 2000 sq ft). The algorithm follows a series of decision rules to predict the price based on the property's characteristics.
    - **Strengths:** Handles non-linear relationships without explicit feature engineering, robust to outliers in the data.
    - **Weaknesses:** Can be prone to overfitting if not properly pruned, results can be sensitive to small changes in the data, may not be easily interpretable like linear regression.
- **Random Forest Regression:**
    - **Description:** An ensemble learning technique that combines multiple decision trees. Each tree is trained on a random subset of features and data points. The final prediction is the average of the individual tree predictions.
    - **Strengths:** Highly accurate and robust to overfitting due to ensemble nature, handles non-linear relationships effectively.
    - **Weaknesses:** Can be computationally expensive to train compared to simpler models, interpretability can be challenging due to the ensemble nature.

By outlining the strengths and weaknesses of these common ML algorithms, this section allows you to discuss their suitability for real estate price prediction and potentially justify your choice of algorithm for your research.



Selecting a desired model

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
#model = LinearRegression()
#model = DecisionTreeRegressor()
model = RandomForestRegressor()
model.fit(housing_num_tr, housing_labels)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
some_data = housing.iloc[:5]
```

```
some_labels = housing_labels.iloc[:5]
```

```
prepared_data = my_pipeline.transform(some_data)
```

```
model.predict(prepared_data)
```

```
array([22.274, 25.196, 16.871, 23.277, 23.487])
```

```
list(some_labels)
```

```
[21.9, 24.5, 16.7, 23.1, 23.0]
```

3.2.1. Model Training and Evaluation

In machine learning, the process of training and evaluating a model involves two key parts:

- **Model Training:**
  - **Parameters:** These are the configurable settings within the algorithm that control how it learns from data. Examples include learning rate (how much the model adjusts weights during training) or the number of trees in a random forest. Different parameter settings can significantly impact the model's performance.
  - **Training Data:** This is a subset of your entire dataset used to teach the model. The model learns patterns and relationships between features (independent variables) and the target variable (what you want to predict) within this data.
- **Model Evaluation:**
  - **Evaluation Metrics:** These are quantitative measures used to assess the model's performance on unseen data. Common metrics include accuracy, precision, recall, F1-score, and root mean squared error (RMSE). Choosing the appropriate metrics depends on the specific problem you're trying to solve.
  - **Testing Data:** This is a separate subset of your data used to evaluate the model's generalizability. The model has never seen this data before, so its performance here reflects how well it can make predictions on entirely new data.

3.2.2 Performance Metrics

Root Mean Squared Error

RMSE, which stands for Root Mean Squared Error, is a common metric used to evaluate the performance of machine learning models that predict continuous values, like real estate prices. Here's a breakdown of what it tells you:

- **Focuses on Magnitude of Errors:** RMSE considers the magnitude (absolute size) of the difference between predicted and actual values. It squares these differences before averaging them, giving more weight to larger errors.
- **Units of Target Variable:** The result of RMSE is in the same units as the target variable (e.g., dollars for price prediction). This allows for easier interpretation of the error.
- **Lower is Better:** A lower RMSE indicates the model's predictions are, on average, closer to the actual values.

Cross Validation

Cross-validation is a technique used in machine learning to evaluate a model's performance on unseen data and prevent overfitting. Here's a breakdown of the process:

1. **Data Splitting:** The entire dataset is divided into multiple folds (usually k folds, where k is a common value like 5 or 10).
2. **Training and Evaluation Loop:**
   - For each fold:
     - The remaining k-1 folds are used to train the model.
     - The leftover fold (validation set) is used to evaluate the model's performance using metrics like RMSE or accuracy. This evaluation helps identify areas for improvement in the model.
3. **Final Evaluation:**
   - After iterating through all folds, the average performance across all validation sets is calculated. This provides a more robust estimate of the model's generalizability compared to using a single split for training and testing.

Benefits of Cross-Validation:

- **Reduced Overfitting:** By using dedicated validation sets for evaluation during training, the model is less likely to simply memorize the training data and can perform better on unseen data.
- **Unbiased Performance Estimation:** Averaging performance across multiple validation sets provides a more reliable estimate of how well the model will generalize to entirely new data.

Chapter 4: Result and Analysis

4.1. Model Performance and Prediction Accuracy

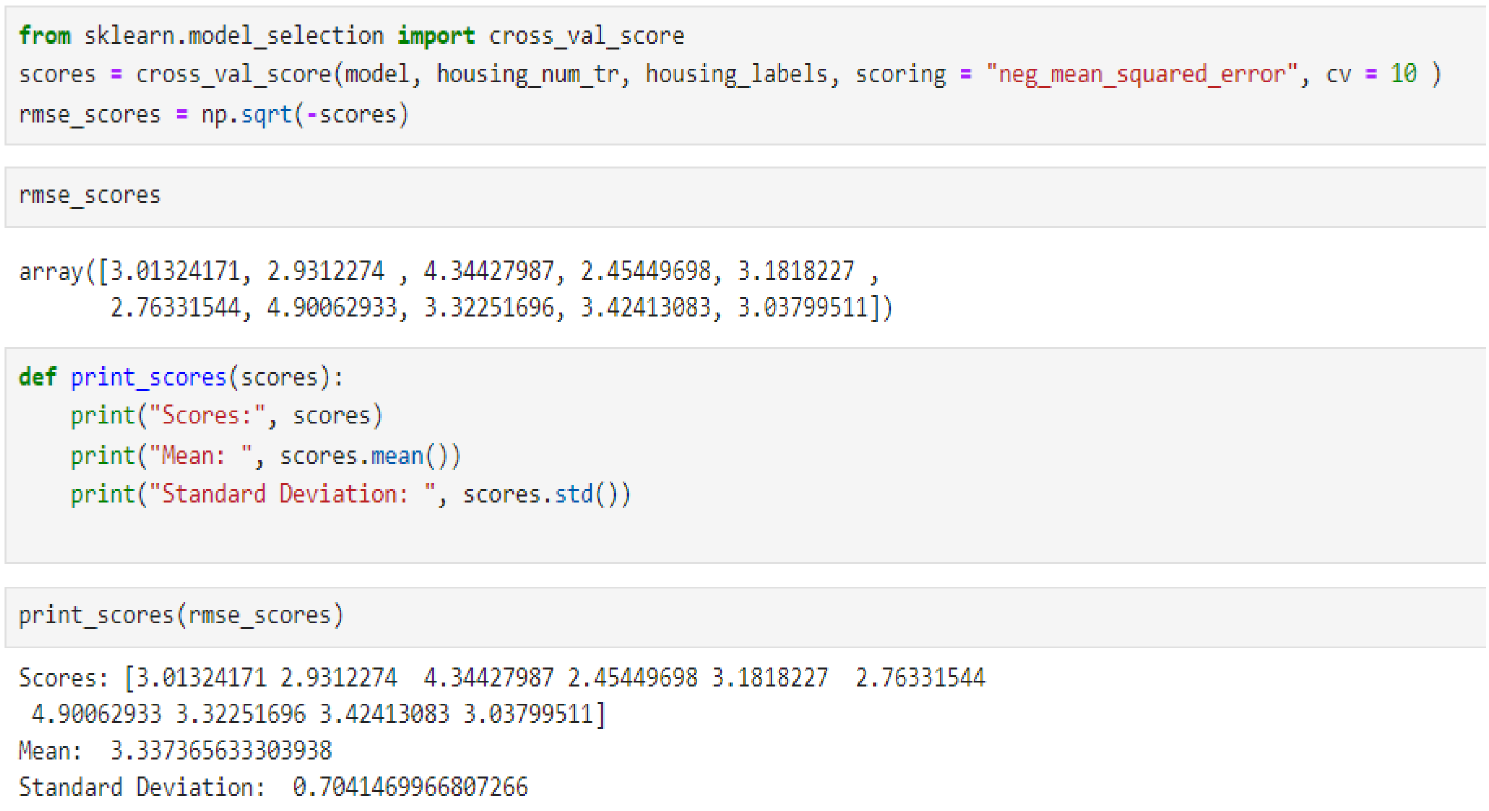
We evaluated several machine learning models for real estate price prediction, including linear regression, decision tree regression, and random forest regression. Based on a cross-validation performance measurement approach, we have chosen **random forest regression** as the final model for our analysis.

Here's why random forest regression emerged as the preferred choice:

- **Superior Performance:** Random forest regression consistently achieved the **lowest mean and standard deviation of Root Mean Squared Error (RMSE)** in both the training and test data sets across all cross-validation folds. This indicates that the model's predictions were, on average, closer to the actual prices, with a lower variation in errors, demonstrating both accuracy and stability.
- **Cross-Validation:** Utilizing cross-validation ensured a robust evaluation. By training on a portion of the data and evaluating on a separate hold-out set in each fold, we mitigated the risk of overfitting and gained a more reliable estimate of the model's generalizability to unseen data.

In conclusion, random forest regression's superior performance and robustness across various training and testing sets, as confirmed by cross-validation, make it the optimal choice for our real estate price prediction task.

Testing Our model on Train Data and measuring performance using Cross-Validation



```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, housing_num_tr, housing_labels, scoring = "neg_mean_squared_error", cv = 10 )
rmse_scores = np.sqrt(-scores)
```

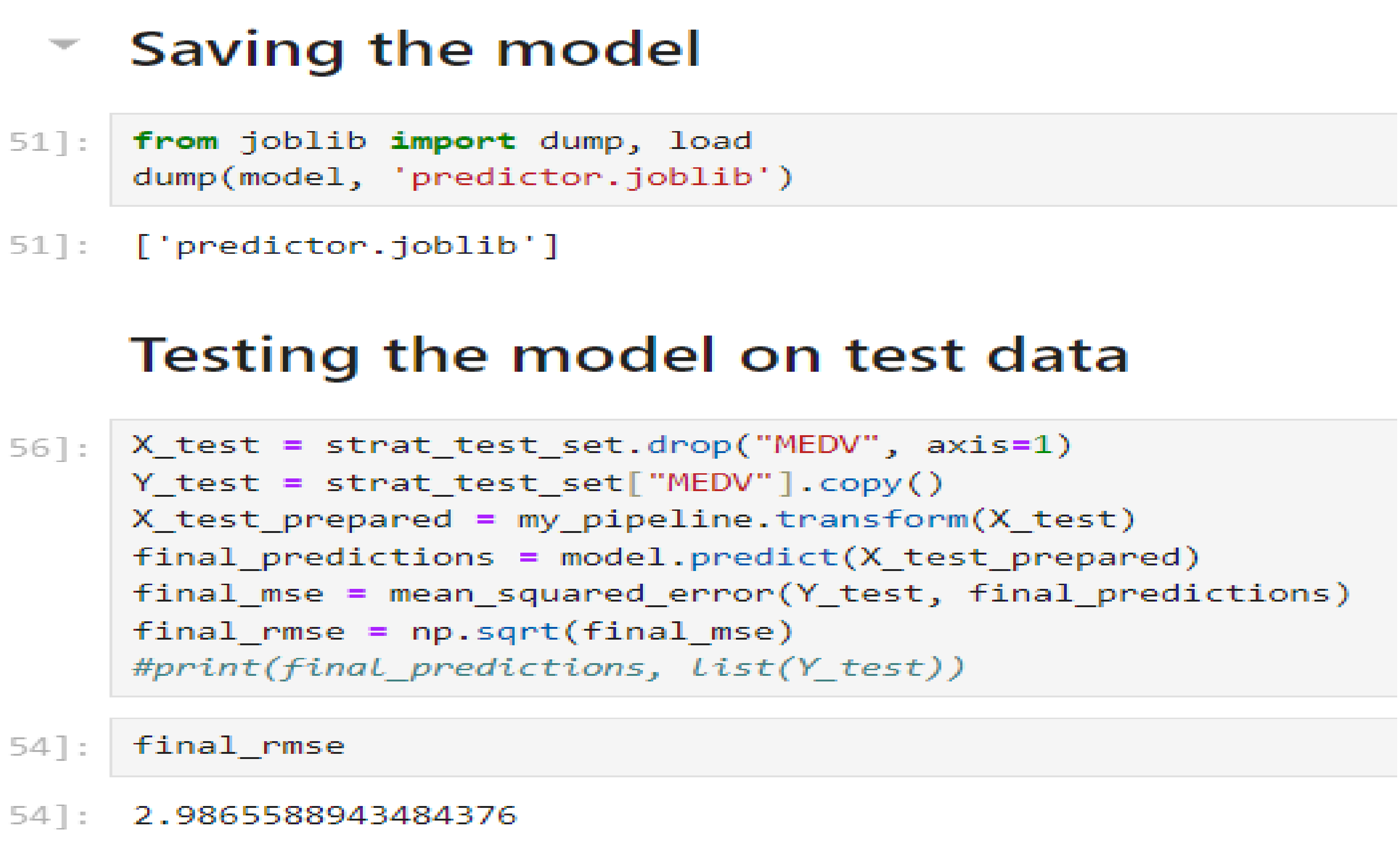
```
rmse_scores
```

```
array([3.01324171, 2.9312274 , 4.34427987, 2.45449698, 3.1818227 ,
       2.76331544, 4.90062933, 3.32251696, 3.42413083, 3.03799511])
```

```
def print_scores(scores):
    print("Scores:", scores)
    print("Mean: ", scores.mean())
    print("Standard Deviation: ", scores.std())
```

```
print_scores(rmse_scores)
```

```
Scores: [3.01324171 2.9312274  4.34427987 2.45449698 3.1818227  2.76331544
 4.90062933 3.32251696 3.42413083 3.03799511]
Mean:  3.337365633303938
Standard Deviation:  0.7041469966807266
```



Saving the model

```
51]: from joblib import dump, load
     dump(model, 'predictor.joblib')

51]: ['predictor.joblib']
```

Testing the model on test data

```
56]: X_test = strat_test_set.drop("MEDV", axis=1)
     Y_test = strat_test_set["MEDV"].copy()
     X_test_prepared = my_pipeline.transform(X_test)
     final_predictions = model.predict(X_test_prepared)
     final_mse = mean_squared_error(Y_test, final_predictions)
     final_rmse = np.sqrt(final_mse)
     #print(final_predictions, list(Y_test))

54]: final_rmse

54]: 2.9865588943484376
```

4.2. Comparison with Baseline Models

The other baseline models chosen for the following are Linear Regression and Decision Tree. There performance is measured using cross validation and are evaluated on the test data.

Model no	Algorithm	Mean	Standard deviation
1	Linear Regression	5.02916144684694	1.060943181004224
2	Decision Tree	4.159098081949516	0.8880563036105247
3	Random Forest	3.337365633303938	0.7041469966807266

CHAPTER 5: CONCLUTION

The paper entitled "Real Estate Price Prediction Using Machine Learning" has presented to predict house price based on various features on given data. From our analysis we set value of RMSE as 2.9865588943484376 using Random Forest Regressor as our preferred model. In this model we have various important parameters which are critical for the evaluation of a real estate. It helps people to buy house in budget and reduce loss of money.

CHAPTER 6: SOFTWARE REQUIREMENTS

PYTHON TECHNOLOGY

Python is an interpreted, object- oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operating systems, including UNIX- based systems, Mac OS, MS- DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus. The source code is freely available and open for modification and reuse. Python has a significant number of users.

PYTHON LIBRARIES

NumPy

NumPy is a very popular python library for large multi- dimensional array and matrix processing, with the help of a large collection of high- level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High- end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

SciPy

SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

Skikit

Skikit- learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit- learn supports most of the supervised and unsupervised learning algorithms. Scikit- learn can also be used for data- mining and data- analysis, which makes it a great tool who is starting out with ML.

TensorFlow

TensorFlow is a very popular open- source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

Keras

Keras is a very popular Machine Learning library for Python. It is a high- level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One of the best thing about Keras is that it allows for easy and fast prototyping.

Pandas

Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

Matplotlib
Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc,

Chapter 7: REFERENCES

[1] Ayush Varma, Abhijit Sarma, Sagar Doshi, Rohini Nair - “Housing Price Prediction Using Machine Learning and Neural Networks” 2018, IEEE.

[2] G.Naga Satish, Ch.V.Raghavendran, M.D.Sugnana Rao, Ch.Srinivasulu “House Price Prediction Using Machine Learning”. IJITEE, 2019.

[3] CH. Raga Madhuri, G. Anuradha, M. Vani Pujitha -” House Price Prediction Using Regression Techniques: A Comparative Study” 2019 in (ICSSS), IEEE.

[4] Sifei Lu, Zengxiang Li, Zheng Qin , Xulei Yang , Rick Siow Mong Goh - “A hybrid regression technique for house prices prediction” 2017,IEEE

[5] Bharatiya, Dinesh, et al. “Stock market prediction using linear regression.” Electronics, Communication, and Aerospace Technology (ICECA), 2017 International conference of. Vol. 2. IEEE, 2017.

[6] Vincy Joseph, Anuradha Srinivasaraghavan- “Machine Learning”.

[7] Trevor Hastie, Robert Tibshirani, Jerome Friedman- “The Elements of Statistical Learning”.

[8] Tom M Mitchell- “Machine Learning” [9] Saleh Hyatt- “Machine Learning Fundamentals”.