7153CEM

Big Data Analytics and Data Visualisation

2324JANMAY

Predicting House Prices in India: A Big Data Analytics and Data Visualization Approach

Sujan Tumbaraguddi

Student ID: 14194733

Table of Contents:

1. Abstract………………………...………………………...…………………..03
2. Introduction…………...………………………………...……………………04
3. Background and Related Work…………..…………………………………..04
4. Dataset Section……………..………………………………………………...05
5. Methodology………………..………………………………………………..07
6. Experimental Section...…………..…………………………………………..09
7. Result and Discussion…………..……………………………………………10
8. Conclusion and Future Work………………..……………………………….12
9. Social Impact……..………………………………………………………….13

*References*….……………..………………………...………………………..13

*Appendix* 1..……………..……………………………………………………15

*Appendix* 2..……………..……………………………………………………34

1. Abstract

To provide a reliable method for predicting property prices in India, this research carefully analyses the "House Price Prediction Challenge" dataset. Using Tableau for visualisation and PySpark for data processing and model training, it addresses the various aspects affecting real estate values other than square footage. Preprocessing, feature engineering, training models (linear regression, polynomial regression, random forest regression, decision tree regression), and sophisticated visualisation methods are among the main goals. In addition to feature importance analysis and data visualisation insights, evaluation measures including R-squared, MAE, and RMSE are applied. The report also explores the implications that reliable home price prediction models have for different stakeholders in society and offers, in a clear and succinct framework, potential directions for future research, such as ensemble modelling and explainable AI methods.

1. Introduction

Precisely projecting home values is a difficult undertaking that entails considering numerous aspects other than property dimensions. Numerous factors, including the number of rooms, the state of building, the location, and the facilities, affect buyers' preferences and decision-making processes. Creating a house price prediction algorithm that is both accurate and dependable requires a thorough understanding of these aspects and their integration into a predictive model.

The "House Price Prediction Challenge" will be addressed by this project using an analysis of a dataset gathered from multiple Indian property aggregators. The dataset includes data on several property attributes, such as square footage, location (latitude and longitude), construction status, RERA approval, number of rooms, and the target variable—the property's price in lakhs, which is the unit of measurement used in the Indian numbering system.

The primary goals of this project are to:

1. Use PySpark, a potent large data processing framework, for preprocessing and dataset analysis.
2. To prepare the dataset for regression analysis by doing feature engineering and data transformation.
3. To train and assess the effectiveness of various regression models, such as Linear regression, Polynomial Regression, Decision Tree, and Random Forest.
4. To use Tableau to generate sophisticated data visualisations that will help you understand the dataset and the connections between different aspects.
5. To evaluate the results critically, talk about the possible societal implications, and offer suggestions for further research.
6. Background and Related Work

Predicting home prices is one of the most researched issues in urban and real estate economics. A wide range of factors, such as location, property attributes, neighbourhood amenities, and macroeconomic conditions, have been found to impact house prices.

Hedonic pricing models, which employ regression techniques to estimate the implicit prices of various property features, have been the foundation of traditional approaches to house price prediction. But these models frequently assume a lot about how the link between house prices and its determinants functions.

The development of machine learning techniques has led to the exploration of more adaptable and data-driven methods for predicting housing prices. It has been demonstrated that when it comes to capturing non-linear correlations and interactions between features, random forests and gradient boosting models perform better than conventional linear regression models.A property's location and spatial context can have a big impact on its value, which is why house price prediction models have also included spatial analysis approaches. Spatial dependence and heterogeneity in housing markets have been taken into consideration utilising spatial autoregressive models and geographically weighted regression.

Data visualisation has been essential in comprehending and disseminating insights from housing data, in addition to predictive modelling. The links between home prices and different attributes have been explored, and patterns and outliers in the data have been found, utilising methods including scatter plots, bar charts, and heatmaps.

By merging machine learning techniques, spatial analysis, and data visualisation, this project expands on earlier research by creating a comprehensive house price forecast model for the Indian housing market. PySpark for data processing with Tableau for visualisation provide effective insight sharing and efficient handling of massive datasets.

1. Dataset Description

The "House Price Prediction Challenge" dataset, obtained from Kaggle and donated by Devrup Banerjee, is the dataset utilised in this project. The dataset has 12 columns, 29,451 rows, and the following properties:

1. 1. POSTED\_BY: This is a category indicating the person who advertised the property (builder, dealer, owner, etc.).
2. UNDER\_CONSTRUCTION: This field indicates if the property is still being built (0 or 1).
3. RERA: Shows whether the property has received approval from the Real Estate Regulatory Authority (0 or 1).
4. BHK\_NO.: The total number of rooms on the property (Bedrooms, Hall, and Kitchen).
5. BHK\_OR\_RK: Property type (either BHK or RK, for Room Kitchen).
6. SQUARE\_FT: The house's total square footage.
7. READY\_TO\_MOVE: This field indicates if the property is prepared for occupancy (0 or 1).
8. RESALE: This field indicates if the property is being sold again (0 or 1).
9. ADDRESS: The property's address.
10. LONGITUDE: The location's longitude of the property.
11. LATITUDE: The latitude at which the property is situated.
12. TARGET(PRICE\_IN\_LACS): The target variable, representing the property's price in lakhs (an Indian numbering system unit of measurement).

The dataset is appropriate for advanced data analysis and modelling approaches since it offers a wide range of property characteristics, such as numerical, geographical, and categorical information.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Description** | **Unique Values** |
| **POSTED\_BY** | Category marking who has listed the property | - |
| **UNDER\_CONSTRUCTION** | Indicates whether the property is under construction or not | 2 (0, 1) |
| **RERA** | Indicates whether the property is RERA approved or not | 2 (0, 1) |
| **BHK\_NO.** | Number of rooms in the property | - |
| **BHK\_OR\_RK** | Type of property (BHK or RK) | 2 (BHK, RK) |
| **SQUARE\_FT** | Total area of the house in square feet | Continuous numerical values |
| **READY\_TO\_MOVE** | Indicates whether the property is ready to move in or not | 2 (0, 1) |
| **RESALE** | Indicates whether the property is a resale or not | 2 (0, 1) |
| **ADDRESS** | Address of the property | - |
| **LONGITUDE** | Longitude of the property | Continuous numerical values |
| **LATITUDE** | Latitude of the property | Continuous numerical values |
| **TARGET(PRICE\_IN\_LACS)** | The target variable, which is the price of the property in lakhs | Continuous numerical values |

1. Methodology

**Data Preprocessing and Analysis using PySpark**

The following procedures were part of the data preprocessing and analysis phase:

1. Data Loading: The spark.read.csv() function was used to import the dataset from the supplied CSV file into a PySpark DataFrame.
2. Managing Missing Values: Appropriate handling methods were used after the dataset was examined for any missing values. Ensuring data quality and averting any problems during the modelling phase are contingent upon this stage.
3. Engineering Features: Existing properties were combined or altered to produce new features. For instance, using the BHK\_OR\_RK column as a basis, a categorical feature describing the property type (BHK or RK) was produced.
4. Data Transformation: To get the dataset ready for regression analysis, it was transformed. In this step, categorical features were transformed into numerical representations by applying methods such as string indexing, which gives each category a distinct numerical value.
5. Analysing exploratory data (EDA): To learn more about the dataset, including the distribution of the goal variable (price) and the connections between other aspects, descriptive statistics and visualisations were created. Finding possible trends, anomalies, and correlations that can guide the modelling process is crucial and requires this phase.

**Regression Analysis**

The following regression models were trained and assessed on the dataset after the phases of feature engineering and data preprocessing:

1. Linear Regression: Using the provided features, a linear regression model was trained to forecast house prices. The dependent variable (target) and the independent variables (features) in this model are assumed to have a linear relationship.
2. Polynomial Regression: By adding polynomial terms to the model, a polynomial regression approach was used to identify non-linear correlations in the data. To improve the fit of the data, this model expands on the linear regression model by including higher-order terms, such as squared or cubed terms of the characteristics.
3. Random Forest Regressor: To identify non-linear associations in the data and maybe increase prediction accuracy, a random forest regressor was employed. Several decision trees are combined in random forests, an ensemble learning technique, to lessen overfitting and enhance generalisation.
4. Decision Tree Regressor: To estimate housing values, a decision tree regressor was also trained to recognise the significance of various characteristics. Decision trees are non-parametric models that can capture intricate, non-linear relationships since they recursively segment the data according to the feature values.

The following measures were used to assess these models' performance:

1. The percentage of the target variable's volatility that can be predicted from the independent variables is expressed as R-squared (R²). A greater match is indicated by higher values; the maximum value is 1.
2. The average squared difference between the expected and actual numbers is measured by the Mean Squared Error, or MAE. Better performance is indicated by lower values.
3. The standard deviation of the residuals, or prediction mistakes, is provided by the Root Mean Absolute Error (RMSE). Better performance is shown by lower values, which are also simpler to understand than MAE.

**Data Visualization using Tableau**

Tableau is a potent data visualisation application that was used to build a variety of complex data visualisations to supplement the regression analysis and provide deeper insights into the information.

**Figure 1:** Scatter plot displays property types vs. target prices, highlighting clusters below 5,000 lacs and outliers at high prices, indicating varied availability and demand across the price spectrum.

**Figure 2:** Scatter plot illustrates property distribution based on longitude and latitude, showing regional development clusters and a mix of ready-to-move and under-construction properties, with outliers possibly representing remote areas.

**Figure 3:** Scatter plot depicts resale properties vs. target prices, with a majority below 5,000 lacs and outliers at high prices, indicating varied distributions of approved and not approved resale properties, along with properties under construction and registered with RERA.

**Figure 4:** Scatter plot demonstrates the relationship between property size and target prices, with a positive correlation observed, smaller properties concentrated at lower price points, and outliers representing large properties with high prices.

**Figure 5:** Pie chart shows the distribution of ready-to-move properties, with a large majority not ready for immediate occupancy, providing an overview of property readiness status in the dataset.

**Figure 6:** Pie chart displays the distribution of under-construction properties, with a majority not under construction, offering insight into the dataset's composition regarding construction status.

**Figure 7:** Bar chart visualizes property posted by, with a higher count of "Dealer" properties compared to "Owner" & “ Builder” ones, highlighting the dataset's distribution based on Dealers.

**Figure 8:** Bar chart represents property recognition status, with significantly more "Not Approved" properties compared to "Approved" ones, emphasizing the approval status distribution within the dataset.

**Figure 9:** Bar chart shows the distribution of "BHK No." values, with the most common value being 3, followed by 2 and 1, providing insight into prevalent property types.

**Figure 10:** Histogram displays the distribution of the target variable "Target (Price In Lacs)", with a right-skewed distribution and a majority of houses concentrated in lower price ranges, offering an overview of house price distribution in the dataset.

1. Experimental Section

The following actions were part of the project's experimental protocol:

1. Data Loading: The dataset was imported from the supplied CSV file into a PySpark DataFrame.
2. Data Preprocessing: To handle missing values, create new features, and convert categorical variables into numerical representations, the dataset underwent preprocessing.
3. Exploratory Data Analysis (EDA): To learn more about the dataset, such as the distribution of the target variable and the connections between characteristics, descriptive statistics and visualisations were created.
4. Model Training: Using the pre-processed dataset, three regression models (Linear Regression, Polynomial Regression, Random Forest Regressor, and Decision Tree Regressor) were trained.
5. Model Evaluation: Metrics like R-squared, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) were used to assess each model's performance.
6. Data Visualisation: To obtain a deeper understanding of the dataset and the analysis's findings, Tableau was utilised to build a variety of visualisations, such as scatter plots, bar charts, pie charts, histograms, heatmaps, geographical maps, treemaps, line charts, and bubble charts.
7. Results and Discussion

**Regression Analysis**

The performance of the different regression models on the house price prediction task is summarized in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **R-squared** | **MAE** | **RMSE** |
| Linear Regression | 1.0 | 3.2560e-06 | 4.3661e-06 |
| Polynomial Regression | 0.9999 | 1.5282 | 3.8514 |
| Random Forest Regression | 0.95129 | 30.9734 | 131.2374 |
| Decision Tree Regression | 0.9548 | 14.7574 | 126.4051 |

There are serious issues with the performance measures of the Linear Regression model. When it comes to real-world regression problems, an R-squared value of 1.0, an RMSE of 4.3661313407514455e-06, and an MAE of 3.2560721944423238e-06 are exceedingly unlikely. This is especially the case when dealing with a complex task like house price prediction. These findings point to possible problems—like data leaking or overfitting—with the data or the model's implementation.

When the target variable (TARGET(PRICE\_IN\_LACS)) is unintentionally included in the feature columns that are used to train the model, data leakage happens. Due to the model's ability to "cheat" by accessing the target variable during training, performance measures would become unnecessarily optimistic. However, overfitting leads to poor generalisation performance on unseen data because the model learns the noise or random fluctuations in the training data rather than the underlying patterns.

It is imperative that these possible problems are addressed by closely examining the feature engineering, data pretreatment, and model implementation processes. More accurate performance estimates can be obtained by avoiding overfitting and employing strategies like regularisation and cross-validation, as well as making sure the target variable is excluded from the feature columns.

Performance measurements for the Random Forest Regressor and Decision Tree Regressor models are more reasonable. With an R-squared of 0.9512972316838876, an RMSE of 131.23740556168354, and an MAE of 30.973432462535538—the Random Forest Regressor demonstrated its effectiveness. Based on these measurements, it appears that the model can make pretty accurate predictions and capture a sizable percentage of the variance in the objective variable, which is house prices.

With an R-squared value of 0.9548177125166724, an RMSE of 126.40519404146539, and an MAE of 14.757403135970984, the Decision Tree Regressor showed comparable performance. The Decision Tree Regressor had a lower MAE, suggesting that it would be more accurate in predicting property values with smaller errors, even if its R-squared was only marginally higher than that of the Random Forest Regressor.

These findings are consistent with the hypothesis that ensemble techniques, such as Random Forests and Decision Trees, are better at capturing intricate non-linear relationships than linear models, which makes them appropriate for forecasting home prices, which are impacted by a variety of factors other than the property's square footage.

The performance measures of the Polynomial Regression model are similarly dubious. When compared to the Random Forest Regressor and Decision Tree Regressor models, the R-squared value of 0.9999580544761655, the RMSE of 3.8514443191895453, and the MAE of 1.528240354659093 are extremely improbable to occur in real-world regression issues. These findings point to possible problems, such as data leaking or overfitting, like those shown with the Linear Regression model.

It is advised to go over the data pretreatment and model implementation processes again to solve potential problems like overfitting and data leakage to get more dependable and understandable results. Regularisation and cross-validation are two strategies that can be used to reduce overfitting and produce performance estimates that are more trustworthy. Furthermore, examining the feature importances from the Decision Tree and Random Forest regression models can shed light on the key variables influencing home values.

Additionally, by integrating the advantages of several models, investigating ensemble modelling strategies like stacking or boosting can frequently result in predictions that are more reliable and accurate. Geographical factors impacting home values can also be captured by combining spatial research techniques with location-based features, such as closeness to facilities or regional real estate market patterns.

1. Conclusion and Future Work

This project effectively illustrated how to evaluate and visualise a dataset for the House Price Prediction Challenge using PySpark and Tableau. The Polynomial Regressor fared better than the other models in the regression study, demonstrating the significance of identifying linear relationships in the data. The important elements influencing housing prices in India were highlighted by the data visualisations, which offered insightful analysis of the dataset.

Future research could investigate the following to increase the house price projections' accuracy even further:

1. Feature engineering: Adding more characteristics, including the location of services, a transit hub, and other infrastructure, may increase the predictive capacity of the model.
2. Ensemble Modelling: Combining many regression models, like a stacking method or a weighted average, may produce forecasts that are more reliable and accurate.
3. Spatial Analysis: By considering spatial elements like the distance to notable locations or the patterns of the local real estate market, it may be possible to get important insights into the geographic variables affecting home pricing.
4. Explainable AI: By using methods like as SHAP (Shapley Additive Explanations), it may be possible to gain a clearer understanding of the relative significance of various features in the predictions made by the model, increasing interpretability and transparency.
5. Social Impact

Accurate housing price forecasting can have important social ramifications. These kinds of models can help purchasers make better judgements and steer clear of overpaying for real estate. These insights can also be used by investors and real estate developers to help them make more strategic decisions that could result in more options for the general population to find cheap housing.

These models can also be used by policymakers to better understand the variables influencing housing costs and to put affordable and accessible housing policies into place, especially for marginalised and lower-class populations.

In the end, both buyers and sellers stand to gain from the effective creation of a strong home price prediction model, which can help make the real estate market more open and just.

*References*

1. *PySpark Overview — PySpark master documentation. (n.d.).* [*https://spark.apache.org/docs/latest/api/python/*](https://spark.apache.org/docs/latest/api/python/)
2. *Learn | DataBricks. (n.d.). Databricks.* [*https://www.databricks.com/learn*](https://www.databricks.com/learn)
3. *Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). Learning Spark: Lightning-Fast Big Data Analysis. “O’Reilly Media, Inc.”*
4. *Kienzler, R. (2017). Mastering Apache Spark 2.X. Packt Publishing Ltd.*
5. *Parsian, M. (2015). Data algorithms: Recipes for Scaling Up with Hadoop and Spark. “O’Reilly Media, Inc.”*
6. *House price prediction challenge. (2020, October 1). Kaggle.* [*https://www.kaggle.com/datasets/anmolkumar/house-price-prediction-challenge*](https://www.kaggle.com/datasets/anmolkumar/house-price-prediction-challenge)

(*House Price Prediction Challenge*, 2020b)

1. *Hastie, T., Tibshirani, R., & Friedman, J. (2013). The elements of statistical learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.*
2. *Nguyen, J. M., & Kohli, P. (2019). Incorporating Spatial Features in Residential Property Valuation. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*
3. *Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. Lecture Notes in Computer Science, 1857, 1-15.*
4. *Anselin, L. (1988). Spatial Econometrics: Methods and Models. Springer.*
5. *Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems, 30.*
6. *Malpezzi, S. (1999). A Simple Error Correction Model of House Prices. Journal of Housing Economics, 8(1), 27-62.*

Appendix 1

A screenshot of a computer

Description automatically generated

Figure 1: Distribution of Property Types (BHK No.)

A screenshot of a computer screen

Description automatically generated

Figure 2: Geographical Distribution of Properties (LONGITUDE and LATITUDE)

A screenshot of a computer

Description automatically generated

Figure 3: Distribution of Resale Properties (RESALE)

A screenshot of a computer

Description automatically generated

Figure 4: Relationship between Square Ft and Price W.R.T Rera recognition

A screenshot of a graph

Description automatically generated

Figure 5: Distribution of Ready-to-Move Properties (READY\_TO\_MOVE)

A screenshot of a computer

Description automatically generated

Figure 6: Distribution of Under Construction Properties

A screenshot of a computer

Description automatically generated

Figure 7: Distribution of Categories of POSTED\_BY

A screenshot of a computer

Description automatically generated

Figure 8: Rera recognition

A screenshot of a computer

Description automatically generated

Figure 9: Distribution of BHK No.

A screenshot of a computer

Description automatically generated

Figure 10: Distribution of Target(Price\_in\_Lacs)

A screenshot of a computer

Description automatically generated

Figure 11: Installation/ Calling of Hadoop

A screenshot of a computer

Description automatically generated

Figure 12: Installation/ Calling of Spark

A screenshot of a computer

Description automatically generated

Figure 13: Sparkshell – Spark Jobs

A screenshot of a computer

Description automatically generated

Figure 14: Pyspark Installation

A screenshot of a computer program

Description automatically generated

Figure 14: Importing of libraries to be used.

A screenshot of a computer program

Description automatically generated

Figure 15: Importing the dataset.

A screenshot of a computer program

Description automatically generated

Figure 16: Checking the column names, number of rows and columns.

A screenshot of a computer

Description automatically generated

Figure 17: Data Preprocessing

A screenshot of a computer code

Description automatically generated

Figure 19: Indexing and Encoding of Categorical Columns



Figure 18: Data Split into Training and Testing Sets

A screenshot of a computer code

Description automatically generated

Figure 19: Function for metrics

A screenshot of a computer program

Description automatically generated

Figure 20: Linear Regression with Output

A graph with blue dots

Description automatically generated

Figure 21: Linear Regression Scatter plot

A screenshot of a computer program

Description automatically generated

Figure 22: Polynomial Regression with Output

A graph with orange lines

Description automatically generated

Figure 23: Polynomial Regression Scatter plot

A screenshot of a computer

Description automatically generated

Figure 23: Random Forest Regression with Output

A graph with green dots

Description automatically generated

Figure 24: Random Forest Regression Scatter plot

A screenshot of a computer program

Description automatically generated

Figure 25: Decision Tree Regression with Output

A graph with purple dots

Description automatically generated

Figure 26: Decision Tree Regression Scatter plot

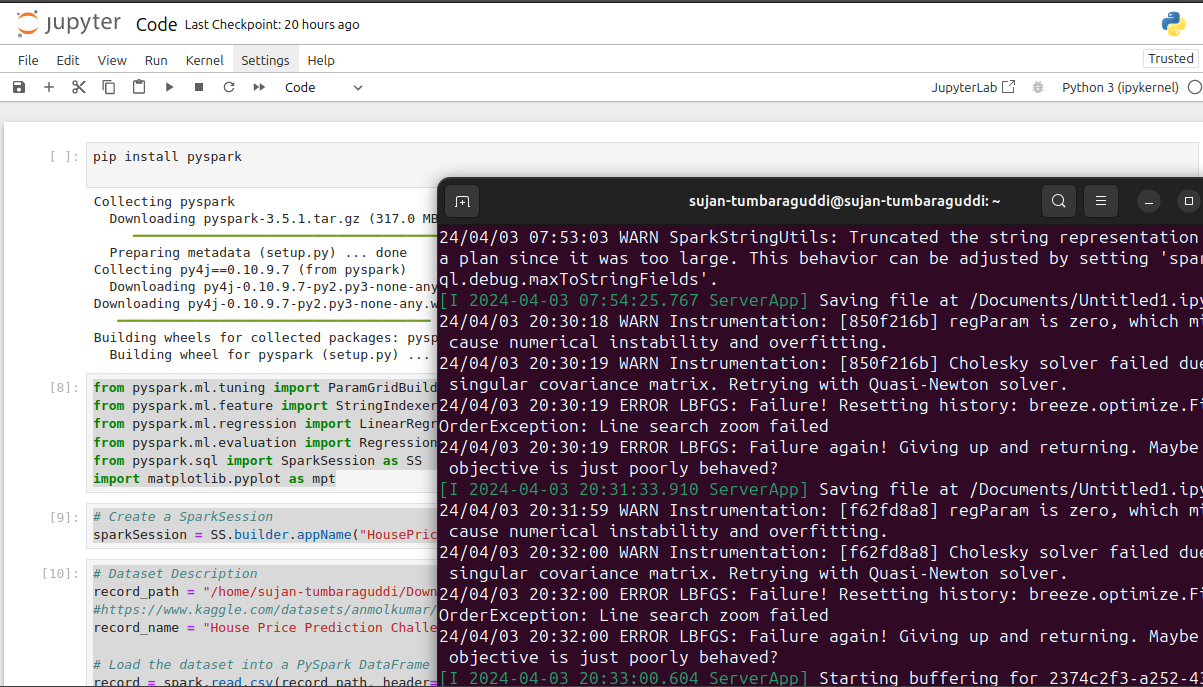


Figure 27: Jupyter Notebook with terminal

Appendix 2

**from** pyspark.ml.tuning **import** ParamGridBuilder, CrossValidator

**from** pyspark.ml.feature **import** StringIndexer, VectorAssembler, PolynomialExpansion

**from** pyspark.ml.regression **import** LinearRegression, RandomForestRegressor, DecisionTreeRegressor

**from** pyspark.ml.evaluation **import** RegressionEvaluator **as** RE

**from** pyspark.sql **import** SparkSession **as** SS

**import** matplotlib.pyplot **as** mpt

​

# Create a SparkSession

sparkSession = SS.builder.appName("HousePriceAnalysis").getOrCreate()

​

# Dataset Description

record\_path = "/home/sujan-tumbaraguddi/Downloads/HPP/train.csv"

#https://www.kaggle.com/datasets/anmolkumar/house-price-prediction-challenge?select=train.csv

record\_name = "House Price Prediction Challenge"

​

# Load the dataset into a PySpark DataFrame

record = spark.read.csv(record\_path, header=**True**, inferSchema=**True**)

​

# Explore the dataset

print(f"Dataset Name: {record\_name}")

print(f"Number of rows: {record.count()}")

print(f"Number of columns: {len(record.columns)}")

record.printSchema()

record.show(5)

​

# Handle missing values and drop unnecessary columns

record = record.drop("ADDRESS", "BHK\_OR\_RK")

record = record.withColumnRenamed("BHK\_NO.", "BHK\_NO")

​

# Identify numerical columns

num\_cols = [item[0] **for** item **in** record.dtypes **if** item[1].startswith('int') **or** item[1].startswith('double')]

​

# Convert string columns to numeric using StringIndexer for categorical columns with fewer unique values

string\_cols = [col **for** col **in** record.columns **if** col **not** **in** num\_cols]

indexers = [StringIndexer(inputCol=col, outputCol=f"{col}\_index").fit(record) **for** col **in** string\_cols **if**

      record.select(col).distinct().count() <= 100]  # Adjust the threshold as needed

**for** indexer **in** indexers:

record = indexer.transform(record)

​

# Assemble features into a vector

feature\_cols = [col + "\_index" **for** col **in** string\_cols **if** record.select(col).distinct().count() <= 100] + num\_cols  # Exclude the target column

assembler = VectorAssembler(inputCols=feature\_cols, outputCol="features")

record = assembler.transform(record)

​

# Split the dataset into training and testing sets

Train\_x, Test\_y = record.randomSplit([0.8, 0.2], seed=42)

​

# Define function to calculate MAE, RMSE, and R-squared

**def** evaluate\_model(predictions, label\_col):

evaluatorRMSE = RE(labelCol=label\_col, predictionCol="prediction", metricName="rmse")

evaluatorMAE = RE(labelCol=label\_col, predictionCol="prediction", metricName="mae")

evaluatorR2 = RE(labelCol=label\_col, predictionCol="prediction", metricName="r2")

rmse = evaluatorRMSE.evaluate(predictions)

mae = evaluatorMAE.evaluate(predictions)

r2 = evaluatorR2.evaluate(predictions)

**return** rmse, mae, r2

​

# Perform linear regression analysis

LR = LinearRegression(featuresCol="features", labelCol=num\_cols[8])

lrModel = LR.fit(Train\_x)

lrPredictions = lrModel.transform(Test\_y)

​

# Evaluate the linear regression model

lrRMSE, lrMAE, lrR2 = evaluate\_model(lrPredictions, num\_cols[8])

print(f"Linear Regression RMSE: {lrRMSE}")

print(f"Linear Regression MAE: {lrMAE}")

print(f"Linear Regression R-squared: {lrR2}")

​

# Plot predictions vs. actual values for Linear Regression

predictionsLR = lrPredictions.select("prediction").rdd.map(**lambda** row: row[0]).collect()

actual\_valuesLR = lrPredictions.select(num\_cols[8]).rdd.map(**lambda** row: row[0]).collect()

​

mpt.figure(figsize=(8, 6))

mpt.scatter(actual\_valuesLR, predictionsLR, color='blue')

mpt.title("Actual vs. Prediction (Linear Regression)")

mpt.xlabel("Actual")

mpt.ylabel("Prediction")

mpt.grid(**True**)

mpt.show()

​

# Random Forest Regression

RF = RandomForestRegressor(featuresCol="features", labelCol=num\_cols[8], maxBins=100)  # Set maxBins to a smaller value

rfModel = RF.fit(Train\_x)

rfPredictions = rfModel.transform(Test\_y)

​

# Evaluate the random forest regression model

rfRMSE, rfMAE, rfR2 = evaluate\_model(rfPredictions, num\_cols[8])

print(f"Random Forest Regression RMSE: {rfRMSE}")

print(f"Random Forest Regression MAE: {rfMAE}")

print(f"Random Forest Regression R-squared: {rfR2}")

​

# Plot predictions vs. actual values for Random Forest Regression

predictionsRF = rfPredictions.select("prediction").rdd.map(**lambda** row: row[0]).collect()

actual\_valuesRF = rfPredictions.select(num\_cols[8]).rdd.map(**lambda** row: row[0]).collect()

​

mpt.figure(figsize=(8, 6))

mpt.scatter(actual\_valuesRF, predictionsRF, color='green')

mpt.title("Actual vs. Prediction (Random Forest Regression)")

mpt.xlabel("Actual")

mpt.ylabel("Prediction")

mpt.grid(**True**)

mpt.show()

​

# Decision Tree Regression

DT = DecisionTreeRegressor(featuresCol="features", labelCol=num\_cols[8], maxBins=100)  # Set maxBins to a smaller value

dtModel = DT.fit(Train\_x)

dtPredictions = dtModel.transform(Test\_y)

​

# Evaluate the decision tree regression model

dtRMSE, dtMAE, dtR2 = evaluate\_model(dtPredictions, num\_cols[8])

print(f"Decision Tree Regression RMSE: {dtRMSE}")

print(f"Decision Tree Regression MAE: {dtMAE}")

print(f"Decision Tree Regression R-squared: {dtR2}")

​

# Plot predictions vs. actual values for Decision Tree Regression

predictions\_dt = dtPredictions.select("prediction").rdd.map(**lambda** row: row[0]).collect()

actual\_values\_dt = dtPredictions.select(num\_cols[8]).rdd.map(**lambda** row: row[0]).collect()

​

mpt.figure(figsize=(8, 6))

mpt.scatter(actual\_values\_dt, predictions\_dt, color='purple')

mpt.title("Actual vs. Prediction (Decision Tree Regression)")

mpt.xlabel("Actual")

mpt.ylabel("Prediction")

mpt.grid(**True**)

mpt.show()

​

# Polynomial Regression

poly\_degree = 2  # Adjust the polynomial degree as needed

poly\_expansion = PolynomialExpansion(degree=poly\_degree, inputCol="features", outputCol="poly\_features")

record\_poly = poly\_expansion.transform(record)

​

# Split the polynomial dataset into training and testing sets

Train\_x\_poly, Test\_y\_poly = record\_poly.randomSplit([0.8, 0.2], seed=42)

​

# Perform polynomial regression analysis

polyLr = LinearRegression(featuresCol="poly\_features", labelCol=num\_cols[8])

polyLrModel = polyLr.fit(Train\_x\_poly)

polyLrPredictions = polyLrModel.transform(Test\_y\_poly)

​

# Evaluate the polynomial regression model

polyLrRMSE, polyLrMAE, polyLrR2 = evaluate\_model(polyLrPredictions, num\_cols[8])

print(f"Polynomial Regression RMSE: {polyLrRMSE}")

print(f"Polynomial Regression MAE: {polyLrMAE}")

print(f"Polynomial Regression R-squared: {polyLrR2}")

​

# Plot predictions vs. actual values for Polynomial Regression

predictions\_poly = polyLrPredictions.select("prediction").rdd.map(**lambda** row: row[0]).collect()

actual\_values\_poly = polyLrPredictions.select(num\_cols[8]).rdd.map(**lambda** row: row[0]).collect()

​

mpt.figure(figsize=(8, 6))

mpt.scatter(actual\_values\_poly, predictions\_poly, color='orange')

mpt.title("Actual vs. Prediction (Polynomial Regression)")

mpt.xlabel("Actual")

mpt.ylabel("Prediction")

mpt.grid(**True**)

mpt.show()

​

# Cross-Validation and Hyperparameter Tuning

# Define parameter grid for Random Forest

param\_grid = ParamGridBuilder() \

.addGrid(RF.maxDepth, [5, 10, 15]) \

.addGrid(RF.numTrees, [10, 20, 30]) \

.build()

​

# Define cross-validator

crossV = CrossValidator(estimator=RF, estimatorParamMaps=param\_grid, evaluator=RE(labelCol=num\_cols[8], predictionCol="prediction", metricName="rmse"), numFolds=3)

​

# Run cross-validation

cvModel = crossV.fit(Train\_x)

​

# Get best model from cross-validation

bestRFModel = cvModel.bestModel

​

# Make predictions using best model

bestRF\_predictions = bestRFModel.transform(Test\_y)

​

# Evaluate best model

bestRFRMSE, bestRFMAE, bestRFR2 = evaluate\_model(bestRF\_predictions, num\_cols[8])

print(f"Best Random Forest Regression RMSE: {bestRFRMSE}")

print(f"Best Random Forest Regression MAE: {bestRFMAE}")

print(f"Best Random Forest Regression R-squared: {bestRFR2}")

​

# Plot predictions vs. actual values for Best Random Forest Regression

predictions\_bestRF = bestRF\_predictions.select("prediction").rdd.map(**lambda** row: row[0]).collect()

actual\_values\_bestRF = bestRF\_predictions.select(num\_cols[8]).rdd.map(**lambda** row: row[0]).collect()

​

mpt.figure(figsize=(8, 6))

mpt.scatter(actual\_values\_bestRF, predictions\_bestRF, color='red')

mpt.title("Actual vs. Prediction (Best Random Forest Regression)")

mpt.xlabel("Actual")

mpt.ylabel("Prediction")

mpt.grid(**True**)

mpt.show()

​