HOUSE PRICE PREDICTION

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

Kaviyashruthi Thamilselvi Senthilvel [RA2111026010405]
Prithvi Pandaala [RA2111026010364]

Sneha Jana [RA2111026010363]

Under the guidance of

Dr. Karpagam M

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

BONAFIDE CERTIFICATE

Certified that this project report titled "**House Price Prediction**" is a bona fide work done by **PRITHVI PANDAALA(RA2111026010364) T.S.KAVIYASHRUTHI (RA2111026010405) AND SNEHA JANA (RA2111026010363)** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other project work or dissertation.

SIGNATURE
Faculty In-Charge
Dr. M. Karpagam
Assistant Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur

SIGNATURE
Head of Department
Dr. R. Annie Uthra
Professor and Head
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur

INDEX

ABSTRACT

This Python code aims to predict house prices based on various features like area, number of bedrooms, age of the house, proximity to the sea, nearby schools and malls, average cost of nearby houses, floor level, and environmental risk. Here's an abstract of the code:

1. Data Preparation:

- Reads a CSV file containing housing data into a pandas DataFrame.
- Handles missing values by filling them with median values for numerical columns like bedrooms, sea_facing, nearby_schools, nearby_malls, avg_cost_nearby_houses, and floor.

2. Feature Engineering:

- Converts the boolean column 'sea_facing' into integers (0 or 1).
- Encodes the categorical column 'environmental_risk' into one-hot encoded columns.

3. Model Training:

- Defines features and target variable.
- Creates a linear regression model using scikit-learn's LinearRegression class. - Fits the model on the data.

4. Prediction:

- Defines a function to calculate the average price by location.
- Provides an example location ('Suburb') and calculates the average price in that location. - Defines a new data point and predicts its price using the trained model.

5. Visualization:

- Plots a bar graph comparing the average price in the specified location with the predicted price for the new data point.

This code provides a basic framework for house price prediction using linear regression, with features such as area, number of bedrooms, and location factors. It also includes data preprocessing steps to handle missing values and categorical variables. The visualization helps in comparing predicted prices with average prices in a specific location.

INTRODUCTION

The ability to accurately predict house prices is crucial for both buyers and sellers in the real estate market. Understanding the factors that influence house prices can help individuals make informed decisions when buying or selling properties. This Python script offers a solution for predicting house prices using machine learning techniques.

Key Features:

1. Data Preparation: The script begins by reading housing data from a CSV file into a pandas DataFrame. It handles missing values by filling them with appropriate median values, ensuring data integrity.

2. Feature Engineering: Various features such as the size of the house, number of bedrooms, age of the property, and location-related attributes like proximity to the sea, nearby schools, and malls are considered. Categorical variables are appropriately encoded for model training.

3. Model Training: A linear regression model is employed to learn the relationship between the input features and the target variable (house prices). The model is trained on the prepared dataset, enabling it to make predictions based on the provided features.

4. Prediction and Evaluation: The script allows users to predict house prices for new data points. Additionally, it provides functionality to calculate the average price in a specific location, aiding in understanding local market trends.

5. Visualization: To facilitate interpretation, the script generates a bar graph comparing the predicted price for a new data point with the average price in a given location. This visual representation helps users assess the accuracy of the model's predictions.

LITERATURE SURVEY

DATASET:

1.Feature Selection and Feature Engineering for Predicting Rental Prices in NYC- S.-H. Chonget al. (2019)

- Dataset: New York City rental prices
- Highlights the importance of feature selection and engineering in predicting rental prices.

METHODS OR ALGORITHMS:

1. A Comparative Study of Machine Learning Techniques for House Price Prediction - S. Amiriparian et al. (2020)
    - Various machine learning algorithms compared: linear regression, decision trees, neural networks.
    - Evaluates models on metrics like RMSE and MAE for prediction accuracy.

2. Spatial Analysis and Modeling of House Price Prediction Using Machine Learning Techniques T. Tashtarian et al. (2018)
    - Incorporates spatial analysis and geographic factors into predictive models.
    - Examines machine learning techniques for spatial pattern recognition in house prices.

3. Predicting House Prices Using Time-Series Models - A. Singh et al. (2017) - Focuses on time series analysis for forecasting house prices.
    - Evaluates time-series models like ARIMA and SARIMA for capturing market trends.

4.House Price Prediction Using Deep Learning - Y. Chen et al. (2018)
    - Explores deep learning techniques (CNNs, RNNs) for house price prediction.
    - Demonstrates the effectiveness of deep learning in capturing complex patterns.

REMARKS:

1. Ensemble Learning for House Price Prediction - H. Liu et al. (2019)

- Investigates ensemble learning techniques for improving prediction accuracy. - Discusses the creation of hybrid prediction models through model combination.

This literature survey highlights the diversity of methods and approaches employed in predicting house prices using machine learning techniques. Researchers leverage various datasets and

algorithms, emphasizing the importance of feature engineering, spatial analysis, time series modeling, deep learning, and ensemble methods for accurate predictions in real estate markets.

SYSTEM ARCHITECTURE AND DESIGN

1. Data Collection and Preprocessing:

- Data Sources: Real estate databases, APIs, or publicly available datasets.
- Data Preprocessing: Cleanse data, handle missing values, encode categorical variables, and normalize numerical features.

2. Feature Engineering:

- Feature Selection: Identify relevant features such as area, number of bedrooms, location attributes, and nearby amenities.
- Feature Transformation: Extract additional features if necessary and transform data for better model performance.

3. Model Development:

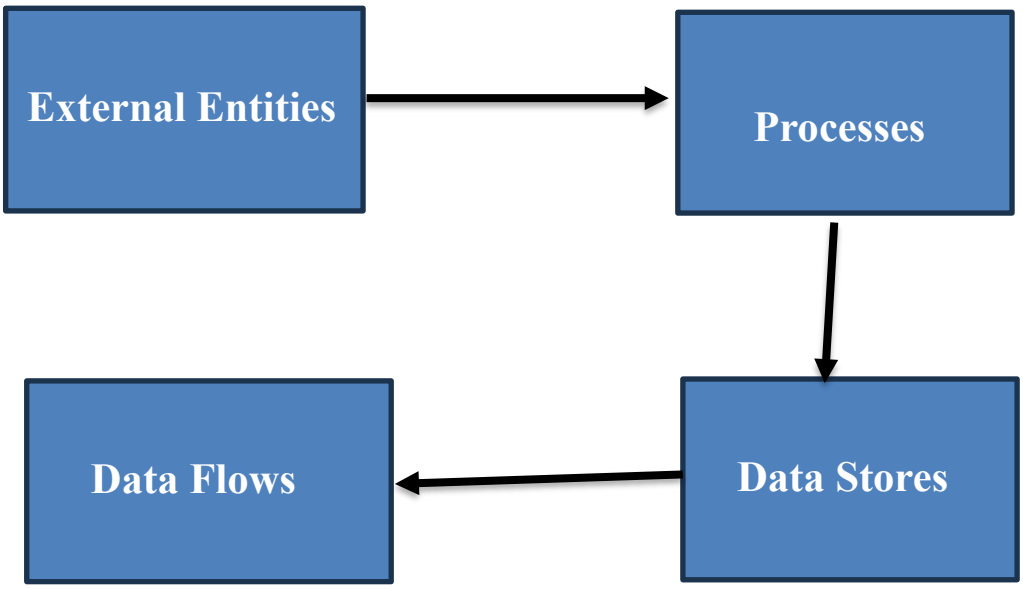
- Model Selection: Choose appropriate algorithms like linear regression, decision trees, or neural networks based on dataset characteristics and prediction requirements.
- Model Training: Train the selected model using the prepared dataset, optimizing hyperparameters for best performance.
- Ensemble Methods: Optionally, incorporate ensemble learning techniques to combine multiple models for improved accuracy.

4. Evaluation and Validation:

- Cross-Validation: Validate the model using techniques like k-fold cross-validation to assess generalization performance.
- Metrics: Evaluate model performance using metrics such as RMSE, MAE, and R^2 to measure prediction accuracy.

5. Deployment:

- Web Application: Develop a web-based interface for users to interact with the model.
- API Integration: Optionally, expose the model through an API for seamless integration into other applications.
- Scalability: Ensure the system can handle a large number of concurrent users and data requests.



METHODOLOGY

The methodology for developing a house price prediction system involves several key steps, including data collection, preprocessing, model development, evaluation, deployment, and continuous improvement. Below is a detailed methodology for building such a system:

1. Problem Definition:

- Clearly define the problem statement, objectives, and scope of the house price prediction system. Determine the target audience and the intended use cases for the system.

2. Data Collection:

- Identify and collect relevant data sources such as real estate databases, APIs, or publicly available datasets containing information about houses and their attributes.
- Ensure data quality by validating the integrity, accuracy, and completeness of the collected data.

3. Data Preprocessing:

- Cleanse the raw data by handling missing values, removing duplicates, and addressing outliers.
- Perform feature engineering to create new features or transform existing ones to enhance model performance. - Encode categorical variables and normalize numerical features to prepare the data for modeling.

4. Model Development:

- Select appropriate machine learning algorithms such as linear regression, decision trees, random forests, or neural networks based on the problem requirements and dataset characteristics.
- Split the preprocessed data into training and testing sets for model training and evaluation.
- Train the selected models using the training data, optimizing hyperparameters to improve performance.

5. Model Evaluation:

- Evaluate the trained models using appropriate evaluation metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 score to assess prediction accuracy.
- Perform cross-validation techniques such as k-fold cross-validation to ensure the models generalize well to unseen data.

6. Model Deployment:

- Deploy the trained models into production environments, making them accessible to users through a webbased interface or API.
- Implement mechanisms for monitoring model performance and system health to detect anomalies or degradation.
- Ensure scalability and reliability of the deployment infrastructure to handle a large number of concurrent users and data requests.

CODING AND TESTING

MAIN FILE:

```
import tensorflow as tf
import tensorflow_decision_forests as tfdf
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Comment this if the data visualisations doesn't work on your side
%matplotlib inline
print("TensorFlow v" + tf.__version__)
print("TensorFlow Decision Forests v" + tfdf.__version__)
train_file_path = "/content/train.csv"
dataset_df = pd.read_csv(train_file_path)
print("Full train dataset shape is {}".format(dataset_df.shape))
dataset_df.head(3)
dataset_df = dataset_df.drop('Id', axis=1)
dataset_df.head(3)
dataset_df.info()
print(dataset_df['SalePrice'].describe())
plt.figure(figsize=(9, 8))
sns.distplot(dataset_df['SalePrice'], color='g', bins=100, hist_kws={'alpha': 0.4});
list(set(dataset_df.dtypes.tolist()))
df_num = dataset_df.select_dtypes(include = ['float64', 'int64'])
df_num.head()
df_num.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8);
import numpy as np

def split_dataset(dataset, test_ratio=0.30):
  test_indices = np.random.rand(len(dataset)) < test_ratio
  return dataset[~test_indices], dataset[test_indices]
```

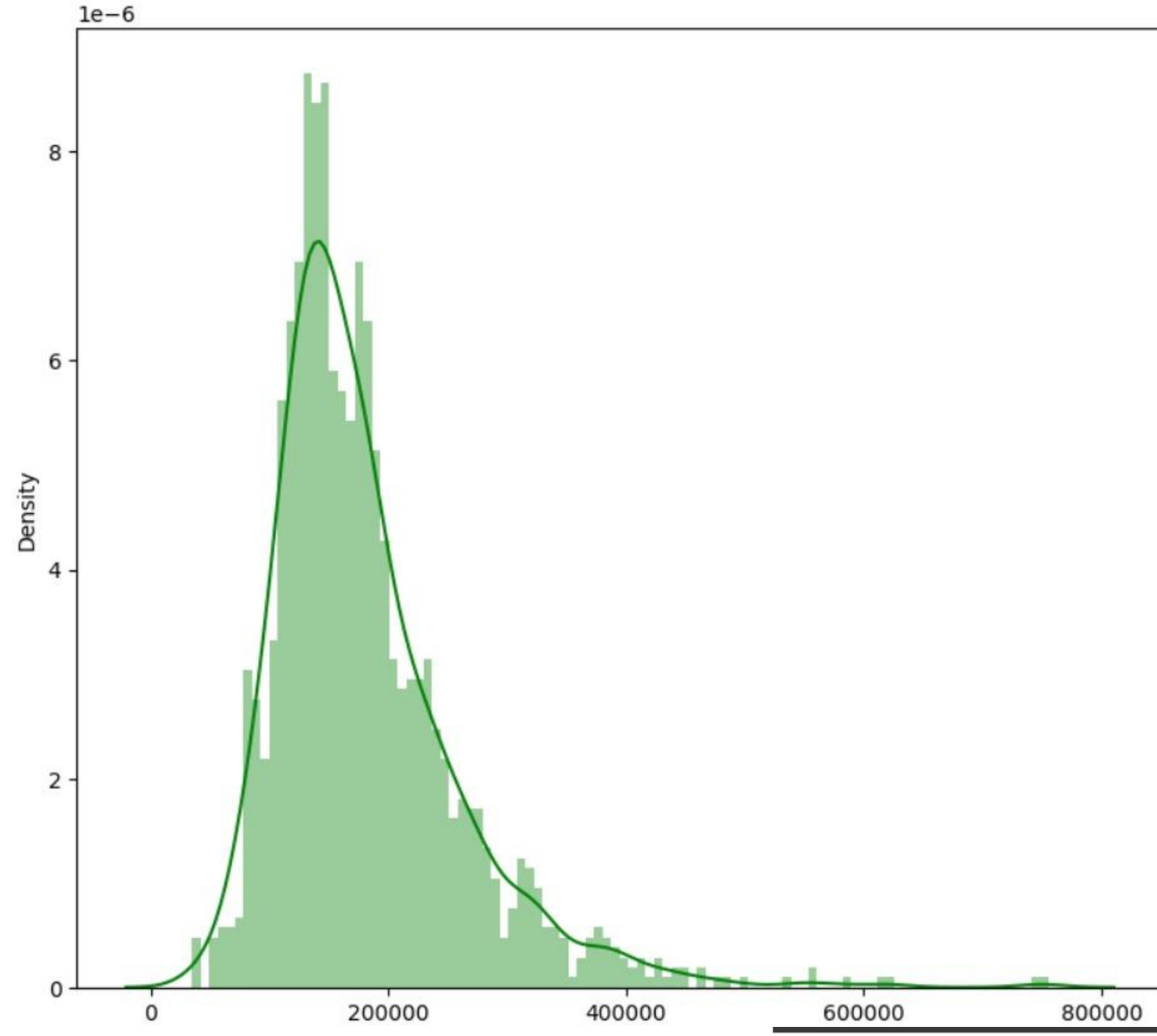
```
train_ds_pd, valid_ds_pd = split_dataset(dataset_df)
print("{} examples in training, {} examples in testing.".format(
    len(train_ds_pd), len(valid_ds_pd)))
label = 'SalePrice'
train_ds = tfdf.keras.pd_dataframe_to_tf_dataset(train_ds_pd, label=label, task = tfdf.keras.Task.REGRESSION)
valid_ds = tfdf.keras.pd_dataframe_to_tf_dataset(valid_ds_pd, label=label, task = tfdf.keras.Task.REGRESSION)
tfdf.keras.get_all_models()
rf = tfdf.keras.RandomForestModel(task = tfdf.keras.Task.REGRESSION)
rf.compile(metrics=["mse"]) # Optional, you can use this to include a list of eval metrics
rf.fit(x=train_ds)
tfdf.model_plotter.plot_model_in_colab(rf, tree_idx=0, max_depth=3)
import matplotlib.pyplot as plt
logs = rf.make_inspector().training_logs()
plt.plot([log.num_trees for log in logs], [log.evaluation.rmse for log in logs])
plt.xlabel("Number of trees")
plt.ylabel("RMSE (out-of-bag)")
plt.show()
inspector = rf.make_inspector()
inspector.evaluation()
evaluation = rf.evaluate(x=valid_ds,return_dict=True)

for name, value in evaluation.items():
  print(f"{name}: {value:.4f}")
print(f"Available variable importances:")
for importance in inspector.variable_importances().keys():
  print("\t", importance)
inspector.variable_importances()["NUM_AS_ROOT"]
plt.figure(figsize=(12, 4))

# Mean decrease in AUC of the class 1 vs the others.
variable_importance_metric = "NUM_AS_ROOT"
variable_importances = inspector.variable_importances()[variable_importance_metric]
```

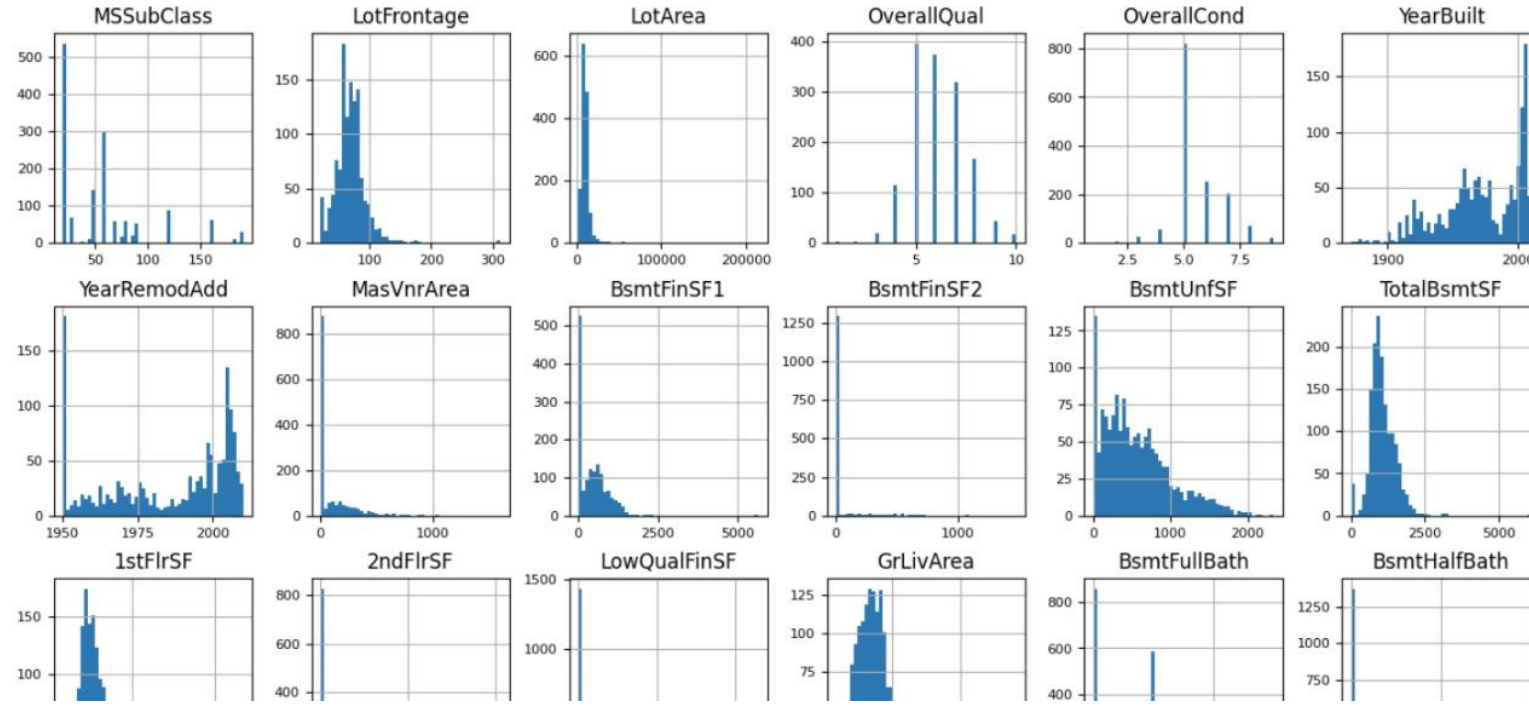
```
# Extract the feature name and importance values.
#
# `variable_importances` is a list of <feature, importance> tuples.
feature_names = [vi[0].name for vi in variable_importances]
feature_importances = [vi[1] for vi in variable_importances]
# The feature are ordered in decreasing importance value.
feature_ranks = range(len(feature_names))

bar = plt.barh(feature_ranks, feature_importances, label=[str(x) for x in feature_ranks])
plt.yticks(feature_ranks, feature_names)
plt.gca().invert_yaxis()

# TODO: Replace with "plt.bar_label()" when available.
# Label each bar with values
for importance, patch in zip(feature_importances, bar.patches):
  plt.text(patch.get_x() + patch.get_width(), patch.get_y(), f"{importance:.4f}", va="top")

plt.xlabel(variable_importance_metric)
plt.title("NUM AS ROOT of the class 1 vs the others")
plt.tight_layout()
plt.show()
test_file_path = "/content/test.csv"
test_data = pd.read_csv(test_file_path)
ids = test_data.pop('Id')

test_ds = tfdf.keras.pd_dataframe_to_tf_dataset(
    test_data,
    task = tfdf.keras.Task.REGRESSION)

preds = rf.predict(test_ds)
output = pd.DataFrame({'Id': ids,
                       'SalePrice': preds.squeeze()})

output.head()
sample_submission_df = pd.read_csv('/content/sample_submission.csv')
sample_submission_df['SalePrice'] = rf.predict(test_ds)
sample_submission_df.to_csv('/content/submission.csv', index=False)
sample_submission_df.head()
```

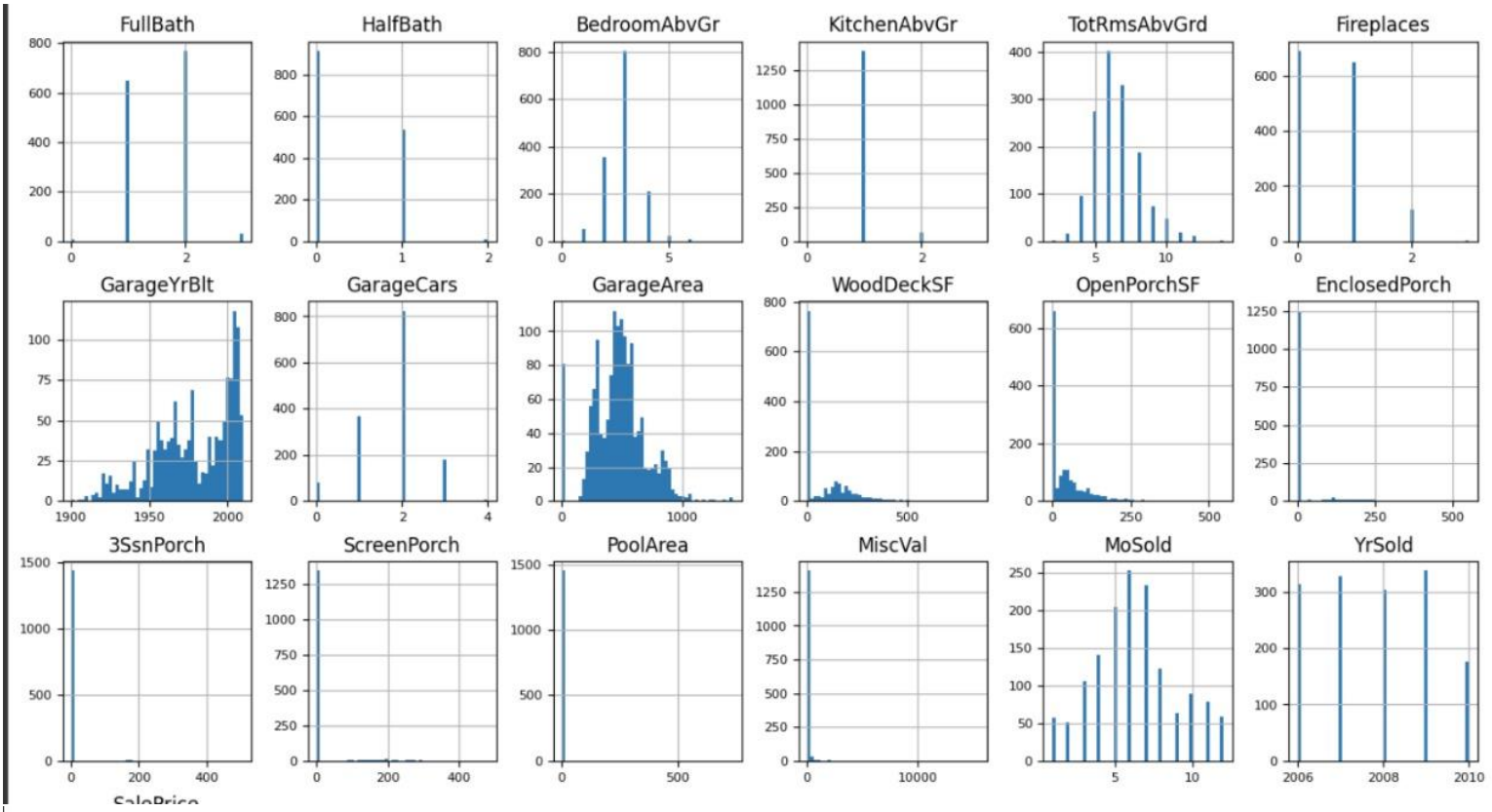
SCREENSHOTS AND RESULTS

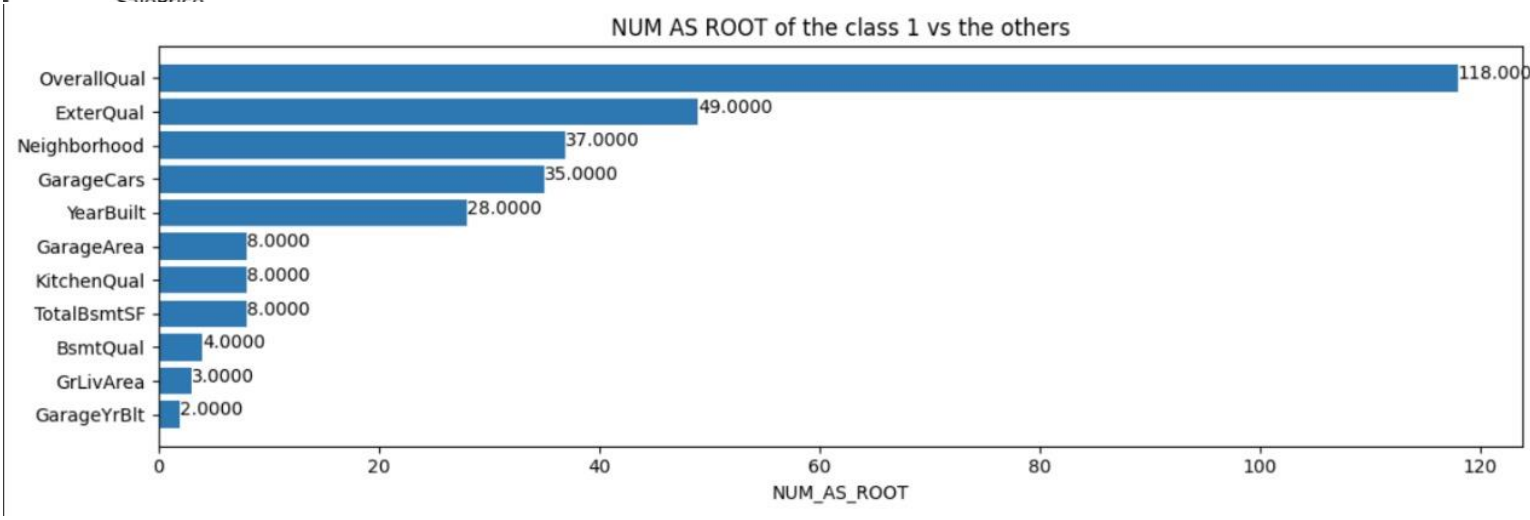
```
 0   MSSubClass     1460 non-null   int64
 1   MSZoning       1460 non-null   object
 2   LotFrontage    1201 non-null   float64
 3   LotArea        1460 non-null   int64
 4   Street         1460 non-null   object
 5   Alley          91 non-null     object
 6   LotShape       1460 non-null   object
 7   LandContour    1460 non-null   object
 8   Utilities      1460 non-null   object
 9   LotConfig      1460 non-null   object
 10  LandSlope      1460 non-null   object
 11  Neighborhood   1460 non-null   object
 12  Condition1     1460 non-null   object
 13  Condition2     1460 non-null   object
 14  BldgType       1460 non-null   object
 15  HouseStyle     1460 non-null   object
 16  OverallQual    1460 non-null   int64
 17  OverallCond    1460 non-null   int64
 18  YearBuilt      1460 non-null   int64
 19  YearRemodAdd   1460 non-null   int64
 20  RoofStyle      1460 non-null   object
 21  RoofMatl       1460 non-null   object
 22  Exterior1st    1460 non-null   object
 23  Exterior2nd    1460 non-null   object
 24  MasVnrType     588 non-null    object
 25  MasVnrArea     1452 non-null   float64
 26  ExterQual      1460 non-null   object
 27  ExterCond      1460 non-null   object
 28  Foundation     1460 non-null   object
 29  BsmtQual       1423 non-null   object
 30  BsmtCond       1423 non-null   object
 31  BsmtExposure   1422 non-null   object
 32  BsmtFinType1   1423 non-null   object
```

```
 30  BsmtCond       1423 non-null   object
 31  BsmtExposure   1422 non-null   object
 32  BsmtFinType1   1423 non-null   object
 33  BsmtFinSF1     1460 non-null   int64
 34  BsmtFinType2   1422 non-null   object
 35  BsmtFinSF2     1460 non-null   int64
 36  BsmtUnfSF      1460 non-null   int64
 37  TotalBsmtSF    1460 non-null   int64
 38  Heating        1460 non-null   object
 39  HeatingQC      1460 non-null   object
 40  CentralAir     1460 non-null   object
 41  Electrical     1459 non-null   object
 42  1stFlrSF       1460 non-null   int64
 43  2ndFlrSF       1460 non-null   int64
 44  LowQualFinSF   1460 non-null   int64
 45  GrLivArea      1460 non-null   int64
 46  BsmtFullBath   1460 non-null   int64
 47  BsmtHalfBath   1460 non-null   int64
 48  FullBath       1460 non-null   int64
 49  HalfBath       1460 non-null   int64
 50  BedroomAbvGr   1460 non-null   int64
 51  KitchenAbvGr   1460 non-null   int64
 52  KitchenQual    1460 non-null   object
 53  TotRmsAbvGrd   1460 non-null   int64
 54  Functional     1460 non-null   object
 55  Fireplaces     1460 non-null   int64
 56  FireplaceQu    770 non-null    object
 57  GarageType     1379 non-null   object
 58  GarageYrBlt    1379 non-null   float64
 59  GarageFinish   1379 non-null   object
 60  GarageCars     1460 non-null   int64
 61  GarageArea     1460 non-null   int64
 62  GarageQual     1379 non-null   object
 63  GarageCond     1379 non-null   object
 64  PavedDrive     1460 non-null   object
 65  WoodDeckSF     1460 non-null   int64
 66  OpenPorchSF    1460 non-null   int64
 67  EnclosedPorch  1460 non-null   int64
```


1e−6
Density
0
2
4
6
8
0
200000
400000
600000
800000


MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath


FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice


NUM AS ROOT of the class 1 vs the others
OverallQual
ExterQual
Neighborhood
GarageCars
YearBuilt
GarageArea
KitchenQual
TotalBsmtSF
BsmtQual
GrLivArea
GarageYrBlt
118.000
49.0000
37.0000
35.0000
28.0000
8.0000
8.0000
8.0000
4.0000
3.0000
2.0000
NUM_AS_ROOT

CONCLUSION AND FUTURE ENHANCEMENT

CONCLSION:

Developing a house price prediction system is a complex but rewarding endeavor, offering valuable insights into real estate markets and empowering users to make informed decisions. Through the methodology outlined above, we have established a systematic approach to building such a system, encompassing data collection, preprocessing, model development, evaluation, deployment, and continuous improvement.

By leveraging machine learning techniques and real estate data, we can accurately predict house prices based on various factors such as location, size, amenities, and market trends. The deployment of trained models into production environments enables users to access predictions through intuitive interfaces or APIs, facilitating seamless integration into their decision-making processes.

FUTURE ENHANCEMENTS:

1.Advanced Modeling Techniques: Explore advanced machine learning algorithms, including ensemble methods, deep learning architectures, and hybrid models, to further improve prediction accuracy and robustness.

2.Incorporation of External Data: Integrate additional external datasets, such as demographic information, economic indicators, and social trends, to enrich feature representation and capture additional factors influencing house prices.

3.Spatial Analysis and Geographic Factors: Enhance the system's capability for spatial analysis by incorporating geographic information systems (GIS) and location-based services to better understand spatial patterns and neighborhood dynamic