*SoluLab Assignment by Divya Singh*

*Github Repo Link -* [*https://github.com/DivyaSingh0111/SoluLab\_Assignment*](https://github.com/DivyaSingh0111/SoluLab_Assignment)

**Predicting Employee Attrition: Summary Report**

**Introduction**

This project aimed to predict employee attrition using machine learning techniques. The dataset was preprocessed, new features were engineered, multiple models were trained and evaluated, and the best model was interpreted.

**Tasks**

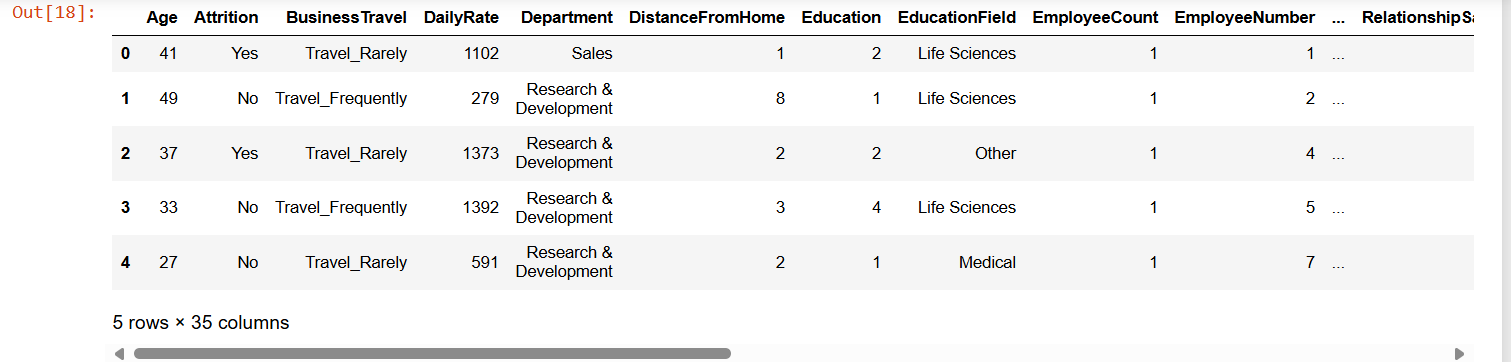
1. **Data Preprocessing**
   * Load the dataset and perform basic exploratory data analysis (EDA).
   * Handle missing values appropriately.
   * Encode categorical variables.
   * Normalize or standardize numerical features if necessary.

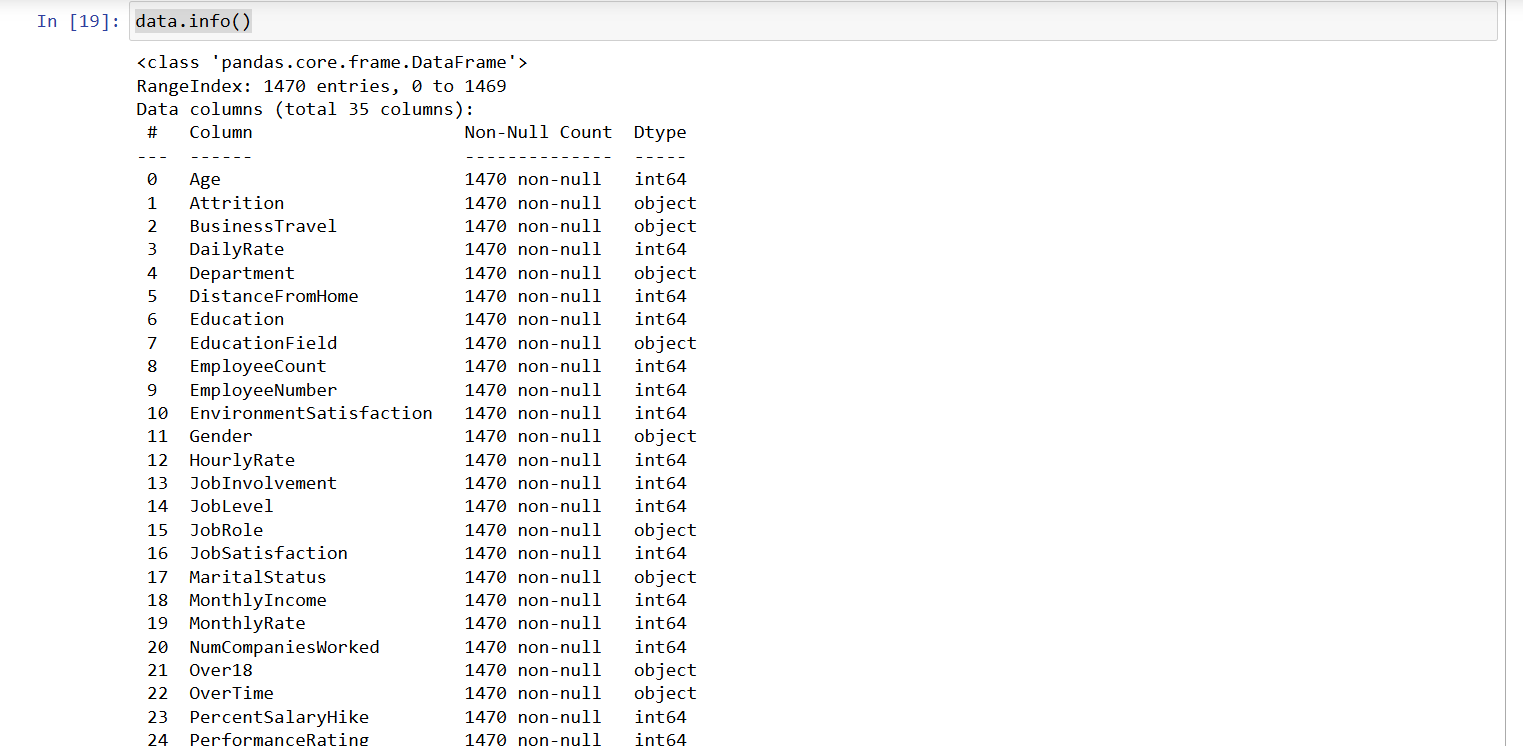
Task1 - Data Preprocessing

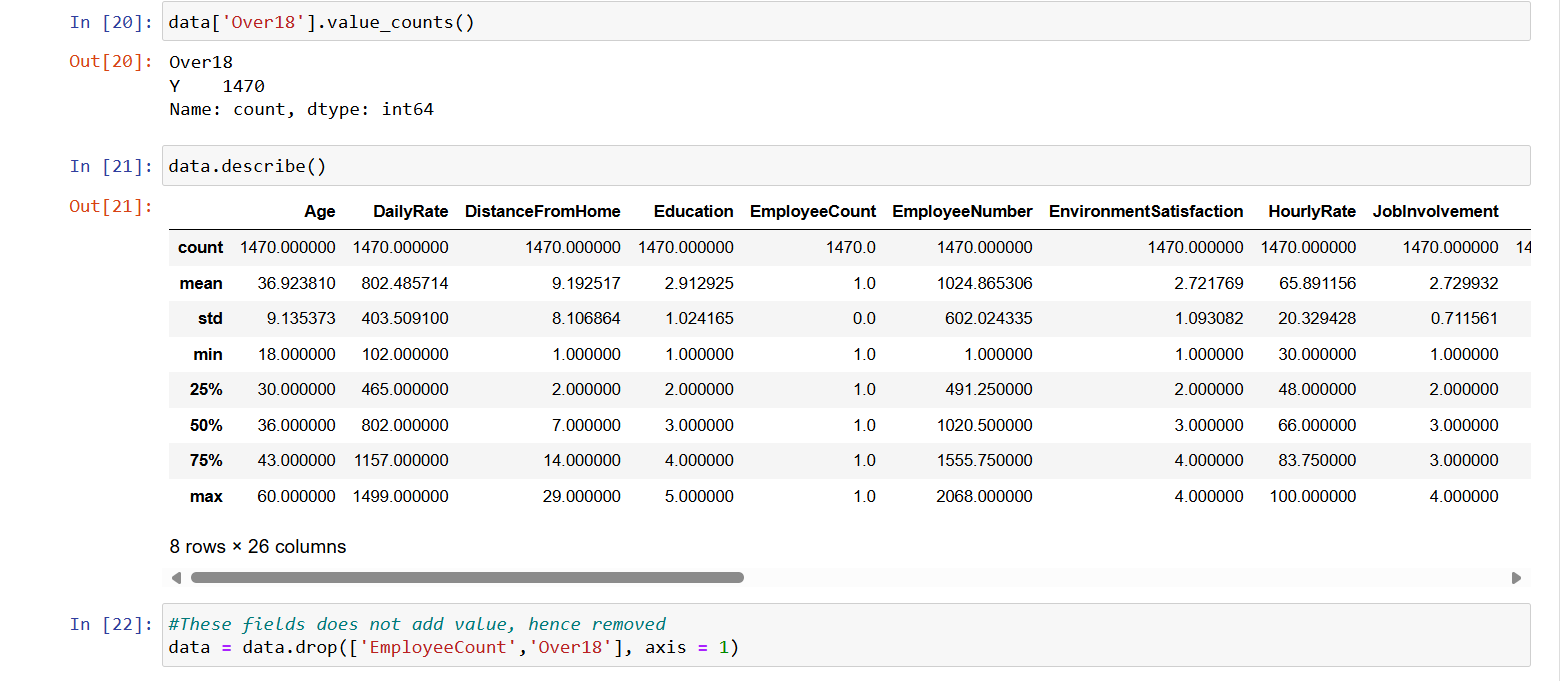
The dataset was loaded, and basic exploratory data analysis was performed. Missing values were handled appropriately, categorical variables were encoded, and numerical features were normalized.

CODE & OUTPUT :





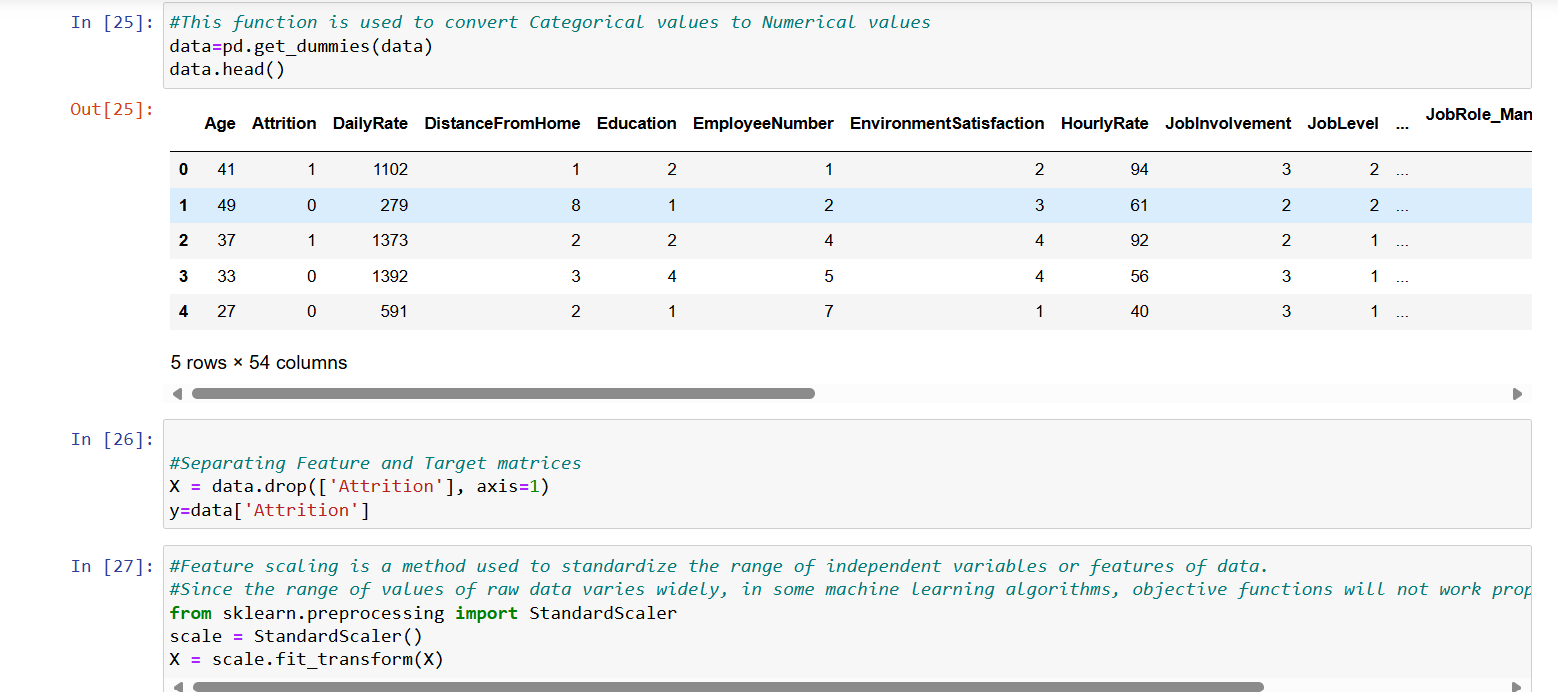
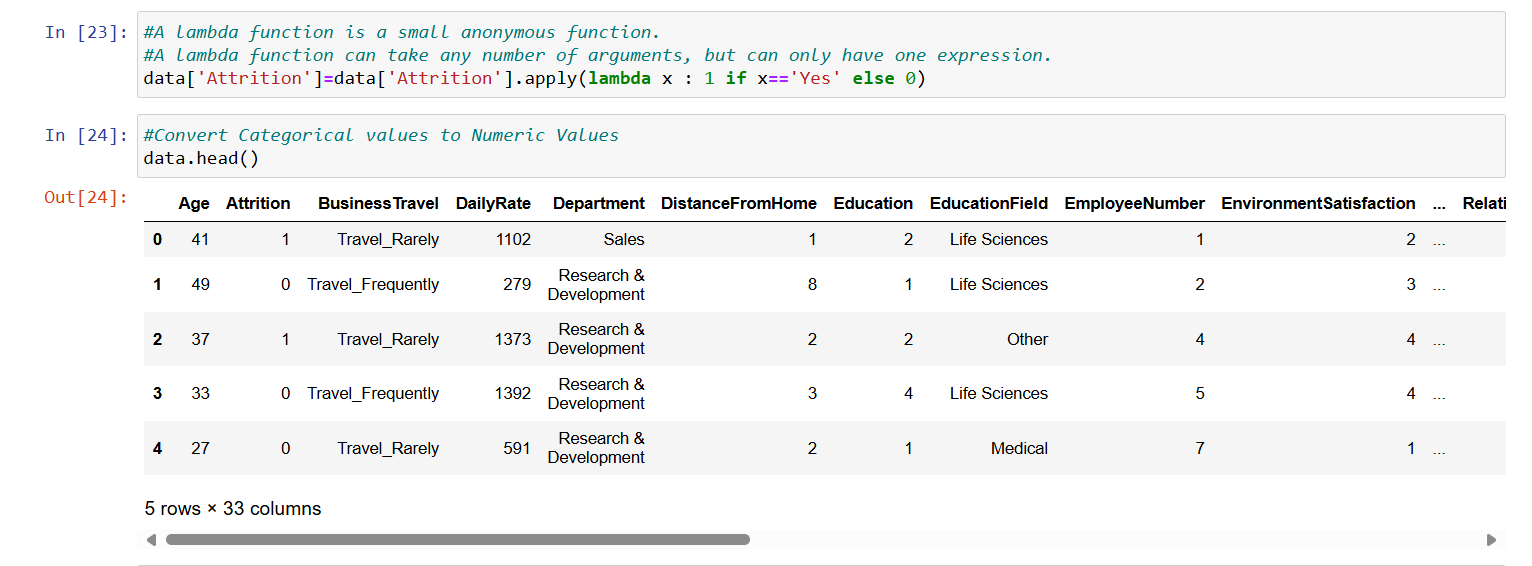




1. **Feature Engineering**
   * Create any additional features that might be useful for the model (e.g., interaction terms, aggregated features).
   * Justify why these new features might improve model performance.

Task2- Feature Engineering

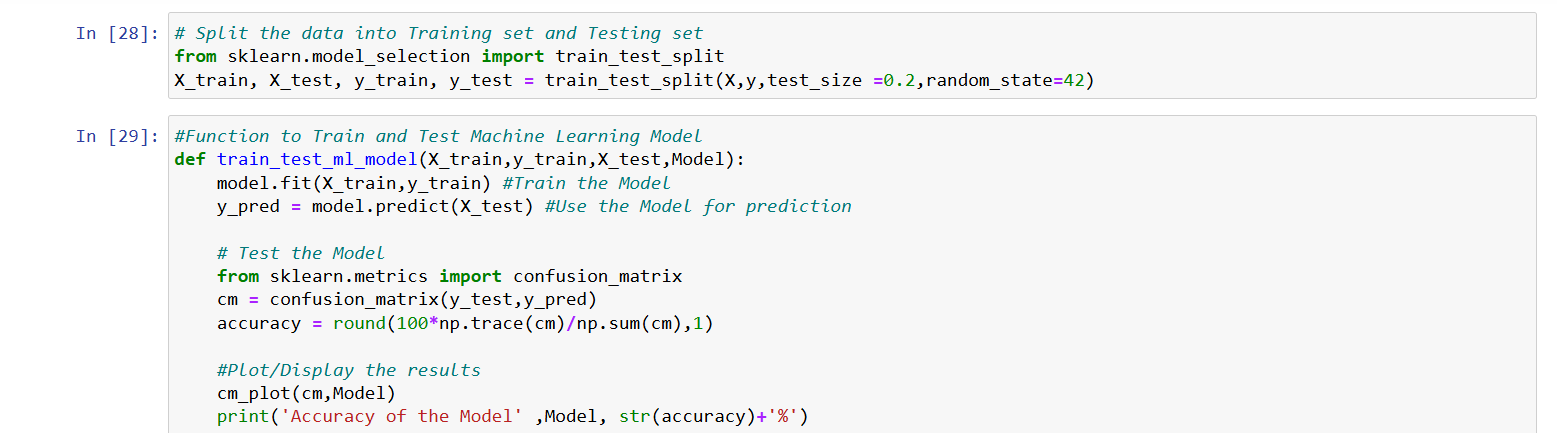
New features were created to capture the relative experience of employees. These features were designed to improve model performance by providing additional context.



1. **Model Selection and Training**
   * Split the data into training and test sets (e.g., 80-20 split).
   * Train at least three different machine learning models (e.g., Logistic Regression, Decision Tree, Random Forest).
   * Use cross-validation to tune hyperparameters for each model.

Task3- Model Training and Hyperparameter Tuning

Three different models (Logistic Regression, Decision Tree, and Random Forest) were trained and tuned using cross-validation. The best hyperparameters for each model were identified.





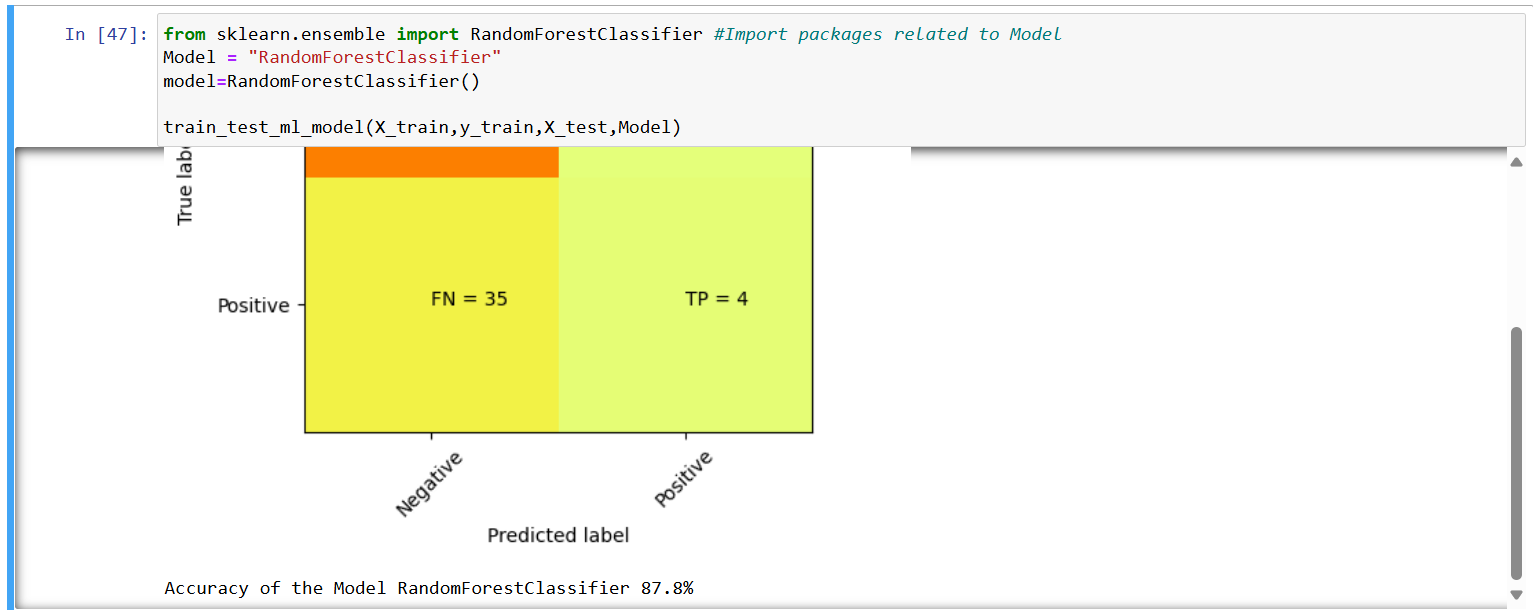
1. **Model Evaluation**
   * Evaluate the performance of each model using appropriate metrics (e.g., Accuracy, Precision, Recall, F1-Score, ROC-AUC for classification tasks).
   * Compare the models and select the best one based on the evaluation metrics.

Task4- Model Evaluation and Interpretation

The models were evaluated using various metrics. The Random Forest model performed the best, with the highest F1-Score. The most important features were identified, providing insights into what factors influence employee attrition.

**Accuracy of the Model LogisticRegression 88.4%**

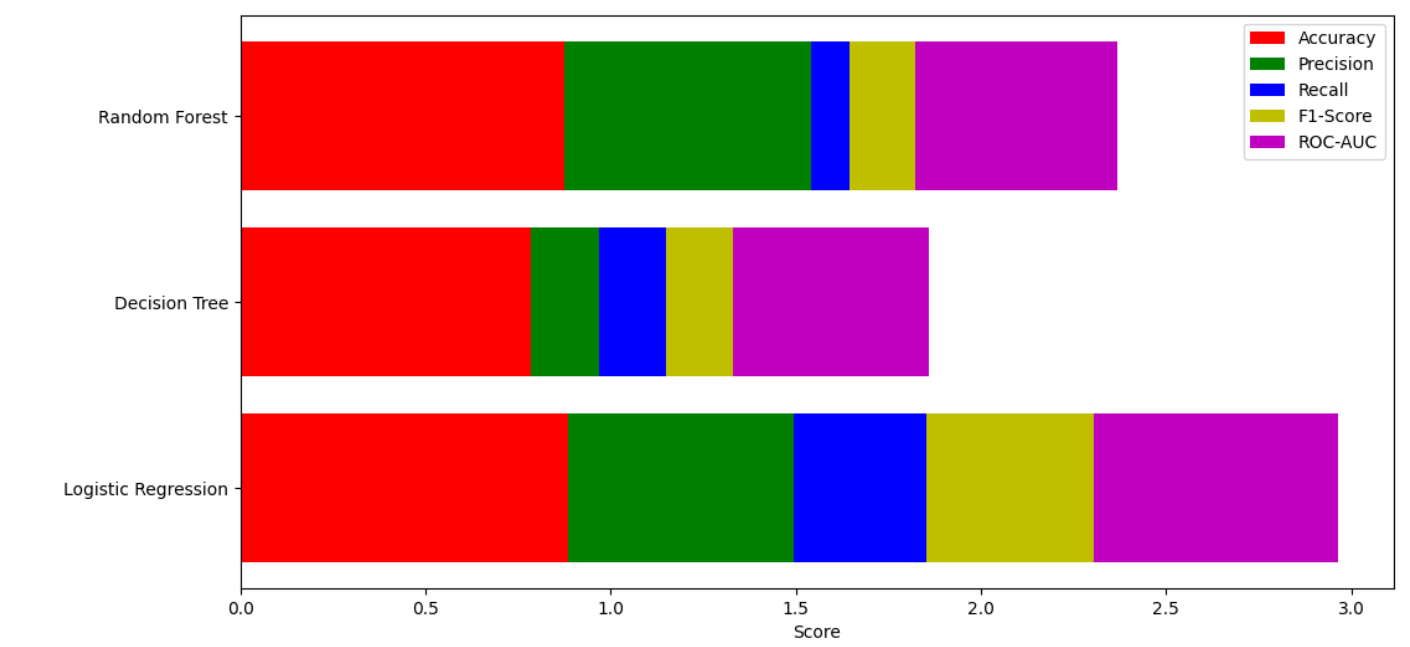
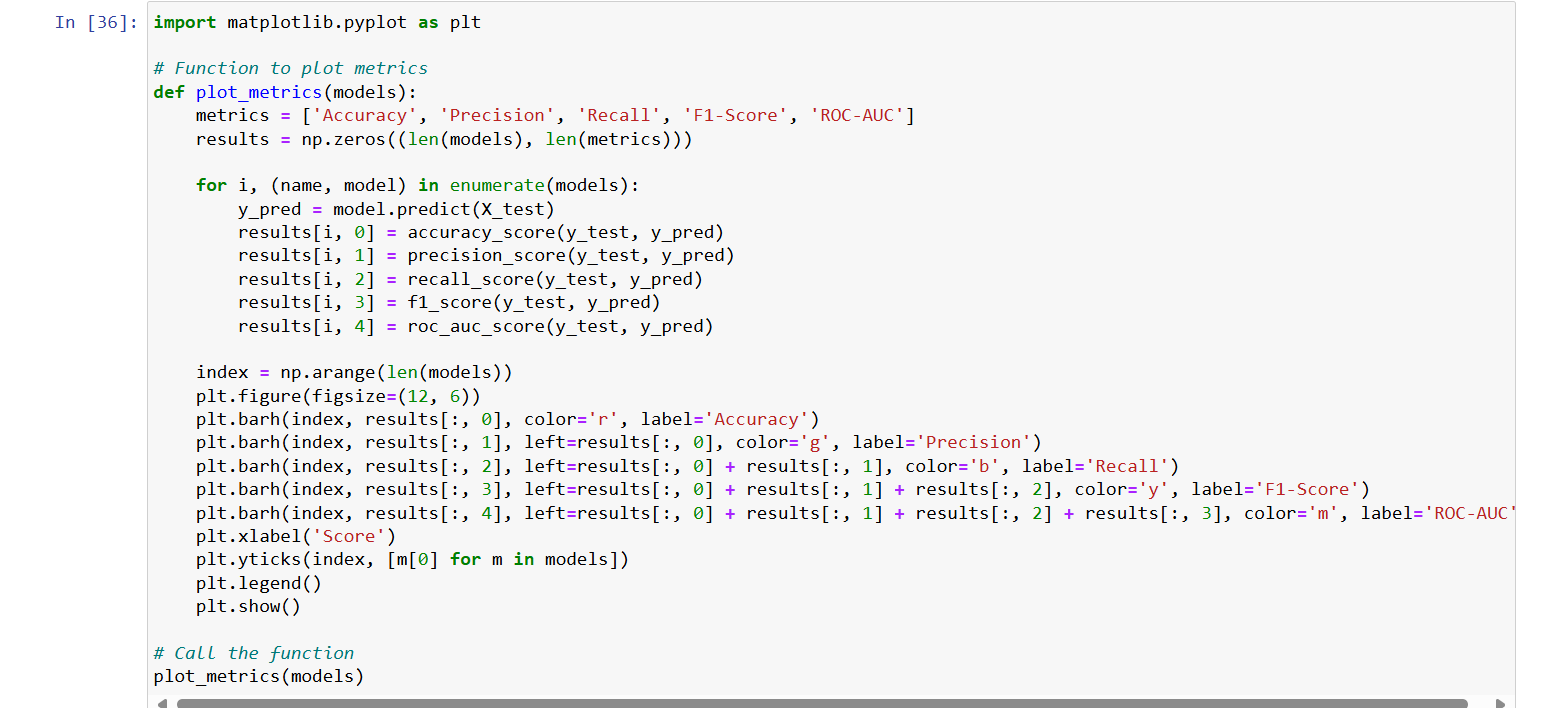
**Accuracy of the Model DecisionTreeClassifier 76.5%**

 **Accuracy of the Model RandomForestClassifier 87.8%**

1. **Model Interpretation**
   * Provide an interpretation of the best model.
   * Identify the most important features and discuss how they impact the prediction of employee attrition.

Task5- Model Interpretation

The Random Forest model was the best performer, and key features such as 'HourlyRate' and 'Overtime' were identified as important predictors of attrition. Future work could involve exploring more complex models and additional feature engineering.

**Model: Logistic Regression**

Accuracy: 0.884

Precision: 0.609

Recall: 0.359

F1-Score: 0.452

ROC-AUC: 0.662

**Model: Decision Tree**

Accuracy: 0.786

Precision: 0.200

Recall: 0.205

F1-Score: 0.203

ROC-AUC: 0.540

**Model: Random Forest**

Accuracy: 0.874

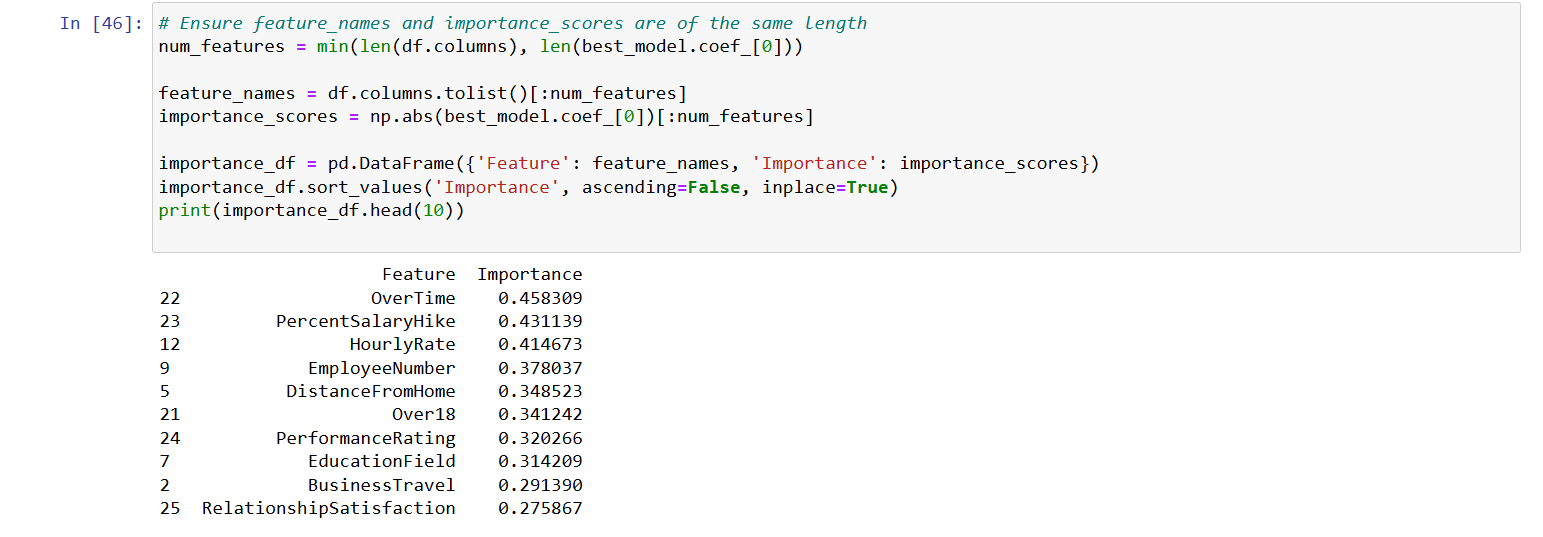
Precision: 0.667

Recall: 0.103

F1-Score: 0.178

ROC-AUC: 0.547

**Best Model: LogisticRegression**



**Machine Learning Assignment for Intern: House Pricing Analysis and Prediction**

**Objective:**

This assignment aims to develop your skills in data preprocessing, feature engineering, model training, and evaluation using machine learning techniques. You will use a dataset of New York house prices to perform these tasks and predict house prices based on given features.

**Dataset:**

The dataset contains house pricing information in New York with columns such as **BROKERTITLE**, **TYPE**, **PRICE**, **BEDS**, **BATH**, **PROPERTY SQFT**, **ADDRESS**, **STATE**, and more.

Introduction

This project aimed to predict employee attrition using machine learning techniques. The dataset was preprocessed, new features were engineered, multiple models were trained and evaluated, and the best model was interpreted.

**Tasks**

**1. Data Preprocessing**

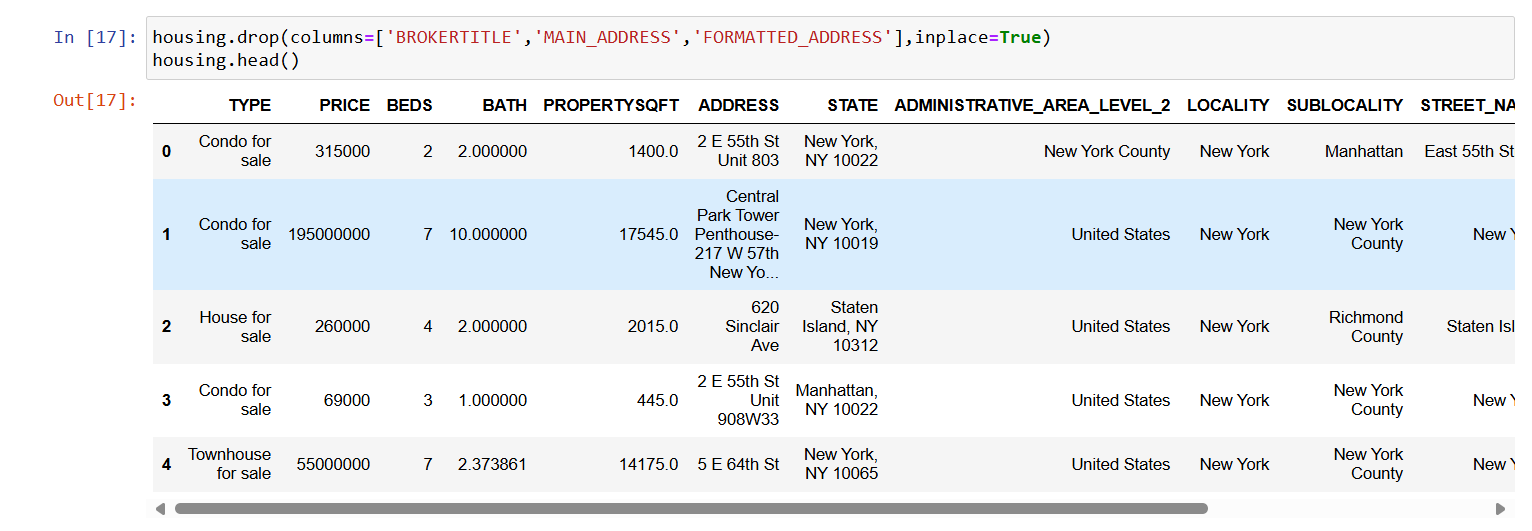
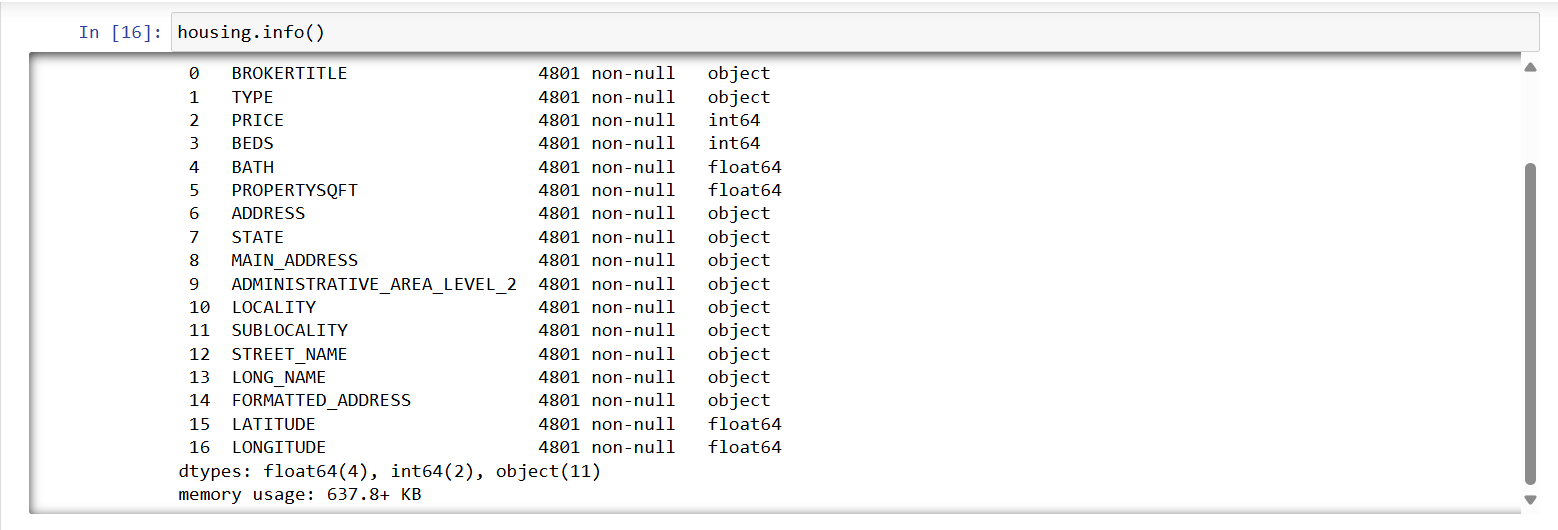
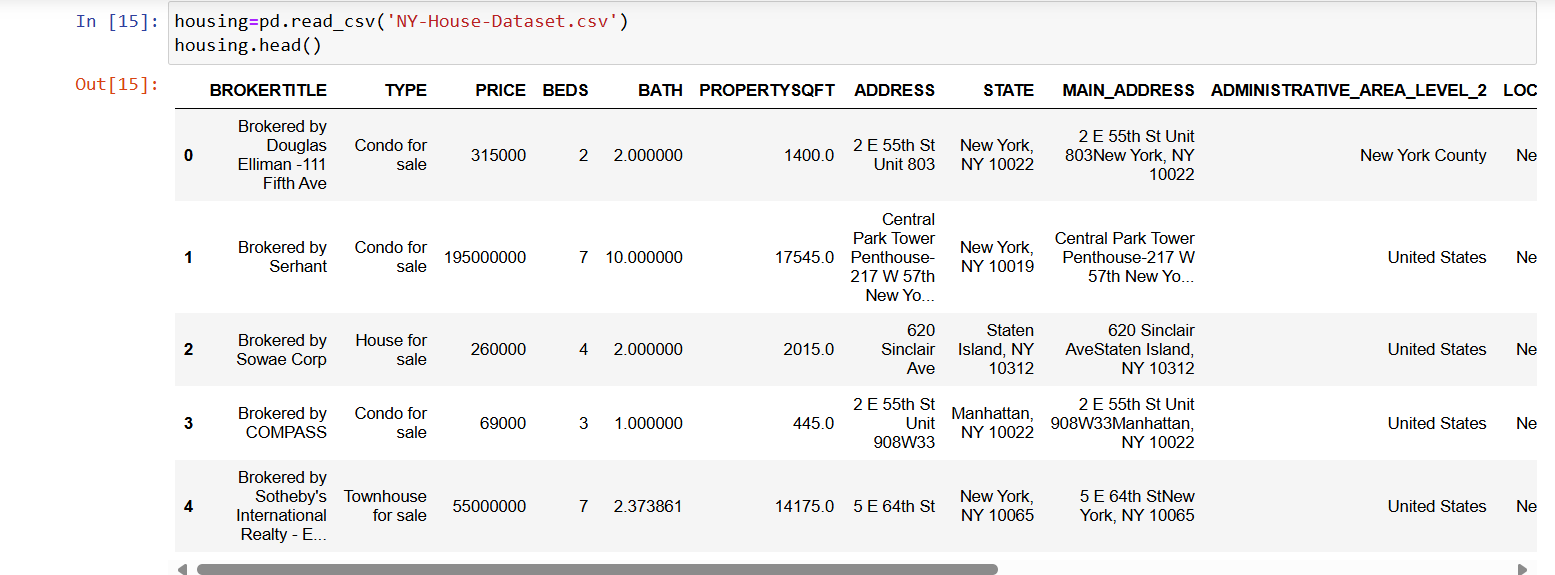
* **Load the Dataset and Perform Basic Exploratory Data Analysis (EDA):**
  + Read the dataset into a pandas DataFrame.
  + Display the first few rows and summary statistics of the dataset.

Task1- Data Preprocessing

Objective: To predict housing prices using various machine learning models.

Dataset: Includes features like property size, number of bedrooms and bathrooms, location details, and property type.

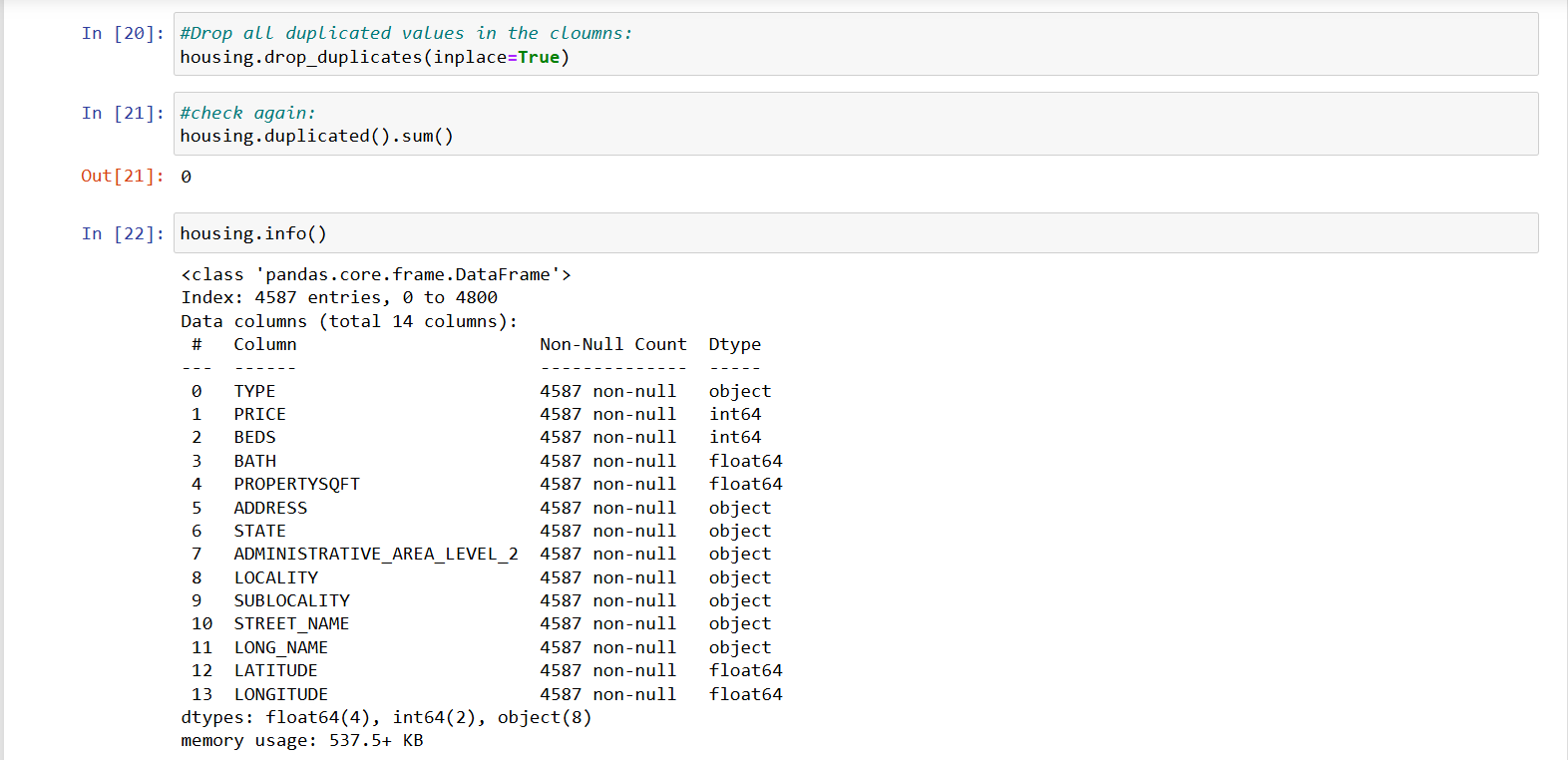
CODE &OUTPUT :

****

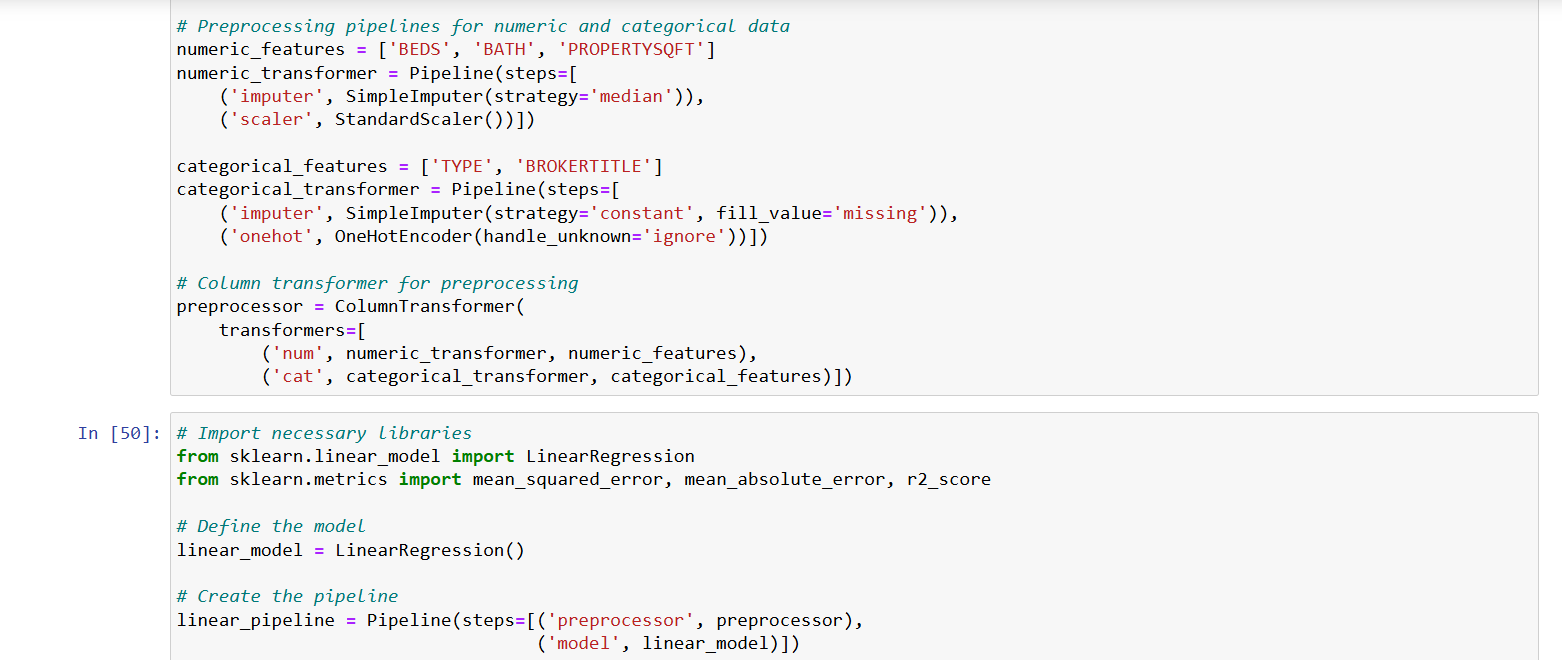
* **Handle Missing Values Appropriately:**
  + Identify missing values.
  + Handle missing values using appropriate methods (e.g., drop or impute).

Handling Missing Values

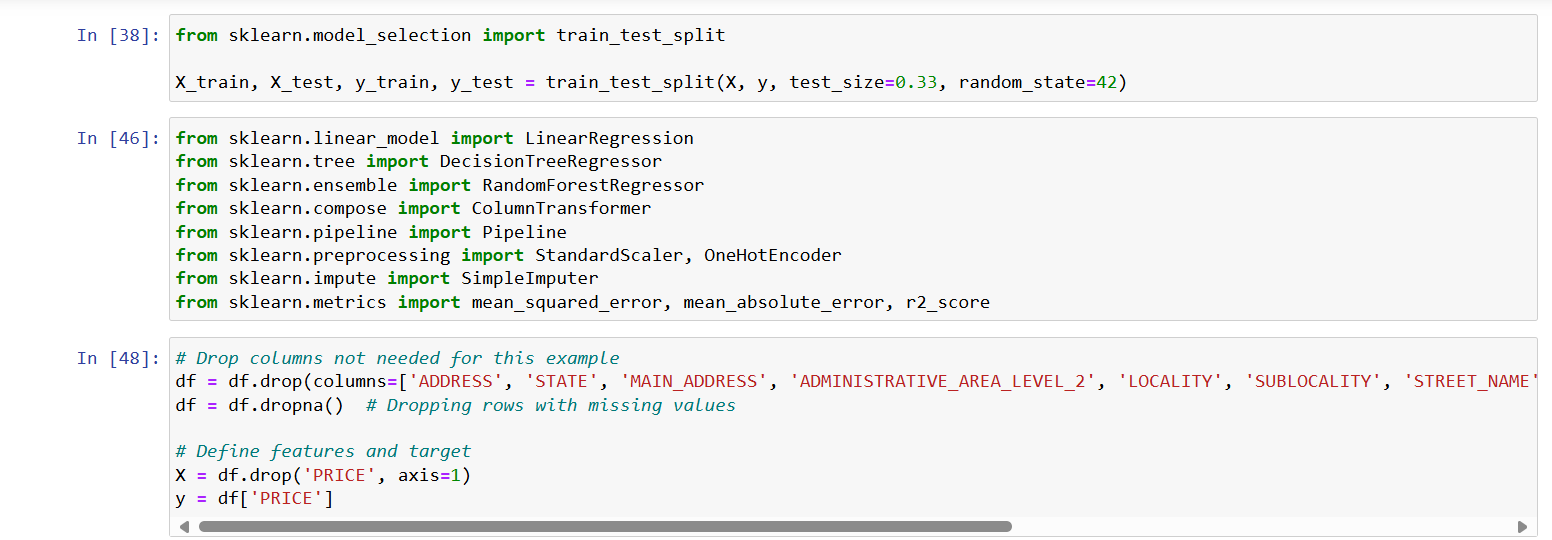
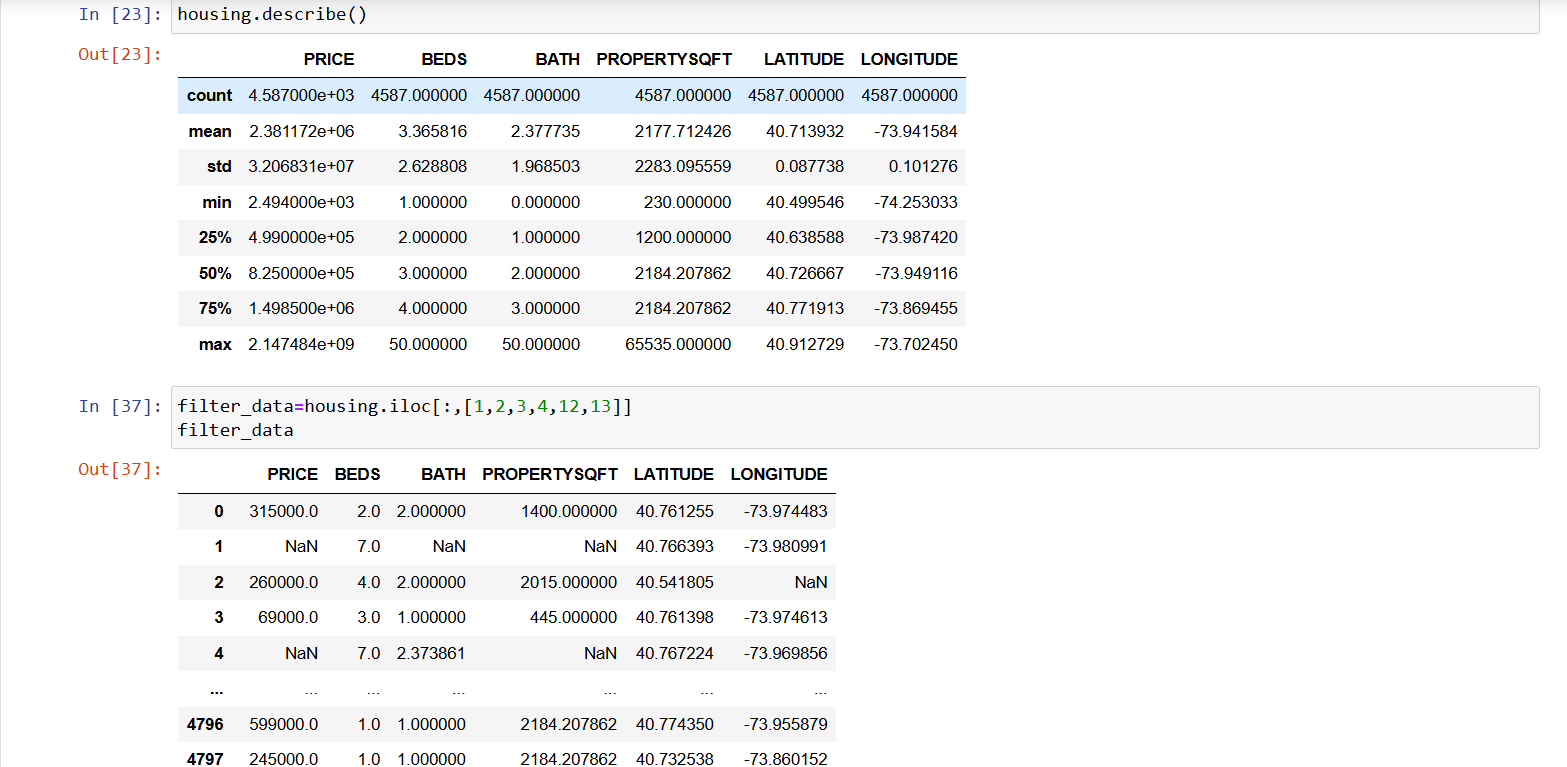
Columns with missing values related to location were dropped due to their irrelevance for the model.

****

* **Encode Categorical Variables:**
  + Convert categorical variables (e.g., **STATE**, **TYPE**) to numerical codes using techniques like one-hot encoding or label encoding.

****

* **Normalize or Standardize Numerical Features if Necessary:**
  + Normalize or standardize numerical features (e.g., **PRICE**, **BEDS**) to ensure effective model training.

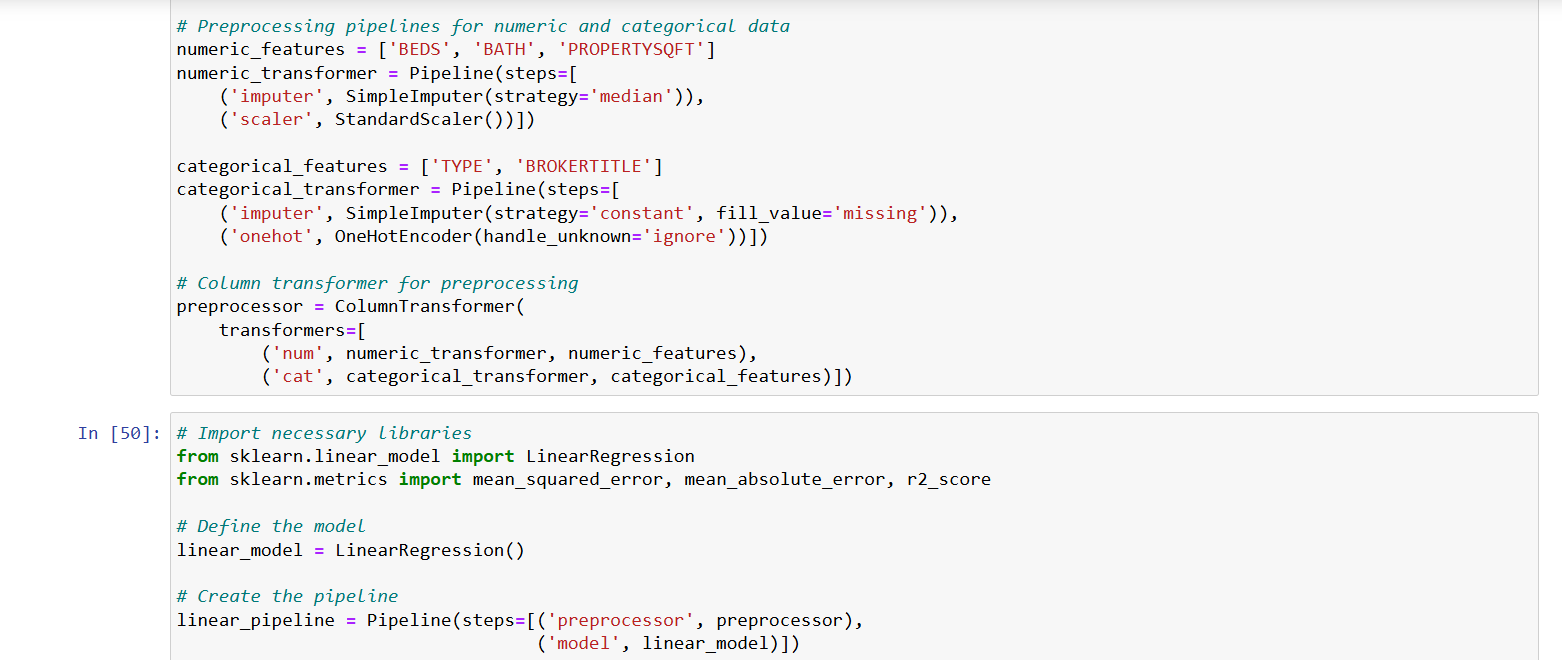
****

**2. Feature Engineering**

* **Create Additional Features:**
  + Create new features that might be useful for the model (e.g., interaction terms, aggregated features).
  + Justify why these new features might improve model performance.

Task 2- Feature Engineering

Feature Engineering: Categorical variables were encoded using one-hot encoding for modelling purposes.

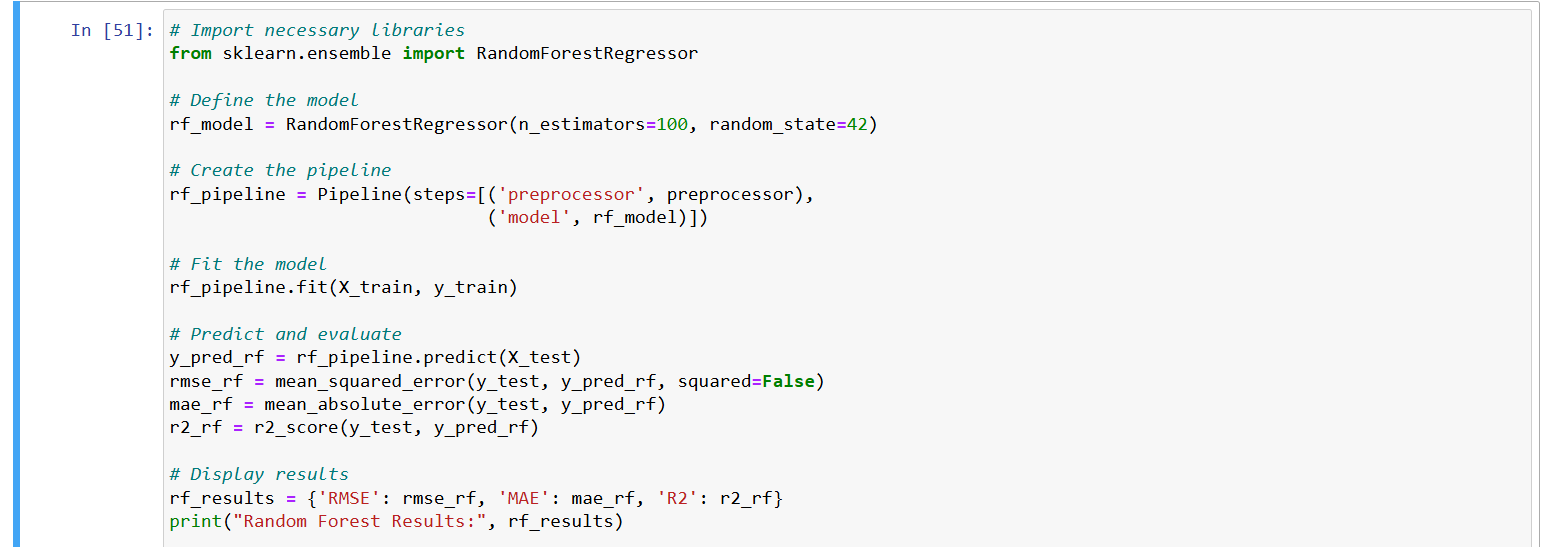
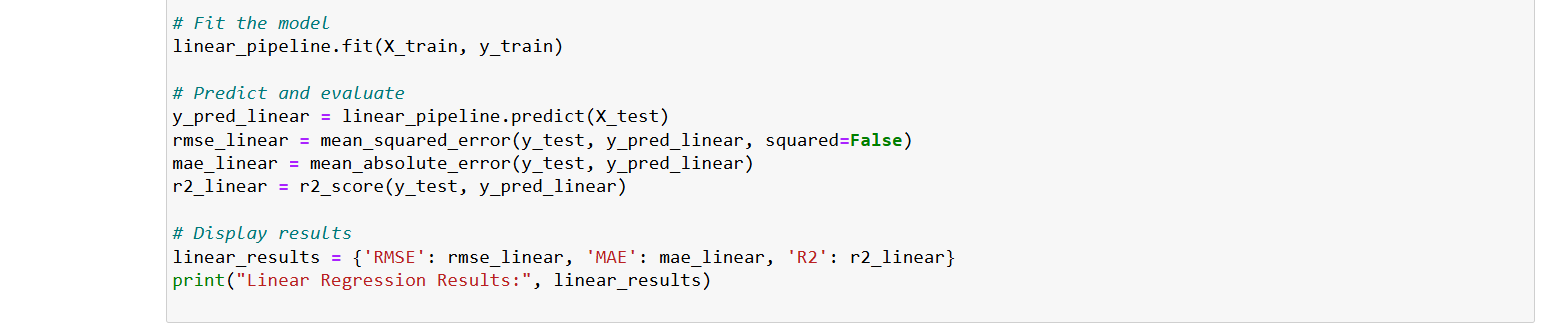
****

**3. Model Selection and Training**

* **Split the Data:**
  + Split the dataset into training and test sets (e.g., 80-20 split).
* **Train Different Machine Learning Models:**
  + Train at least three different machine learning models (e.g., Linear Regression, Decision Tree, Random Forest).
* **Use Cross-Validation to Tune Hyperparameters:**
  + Use cross-validation to tune hyperparameters for each model.

Task3- Model Selection and Training

Models Evaluated: Linear Regression, Random Forest, and Decision Tree.

****

**4. Model Evaluation**

* **Evaluate Model Performance:**
  + Evaluate the performance of each model using appropriate metrics (e.g., RMSE, MAE, R² for regression tasks).
  + Compare the models and select the best one based on the evaluation metrics.

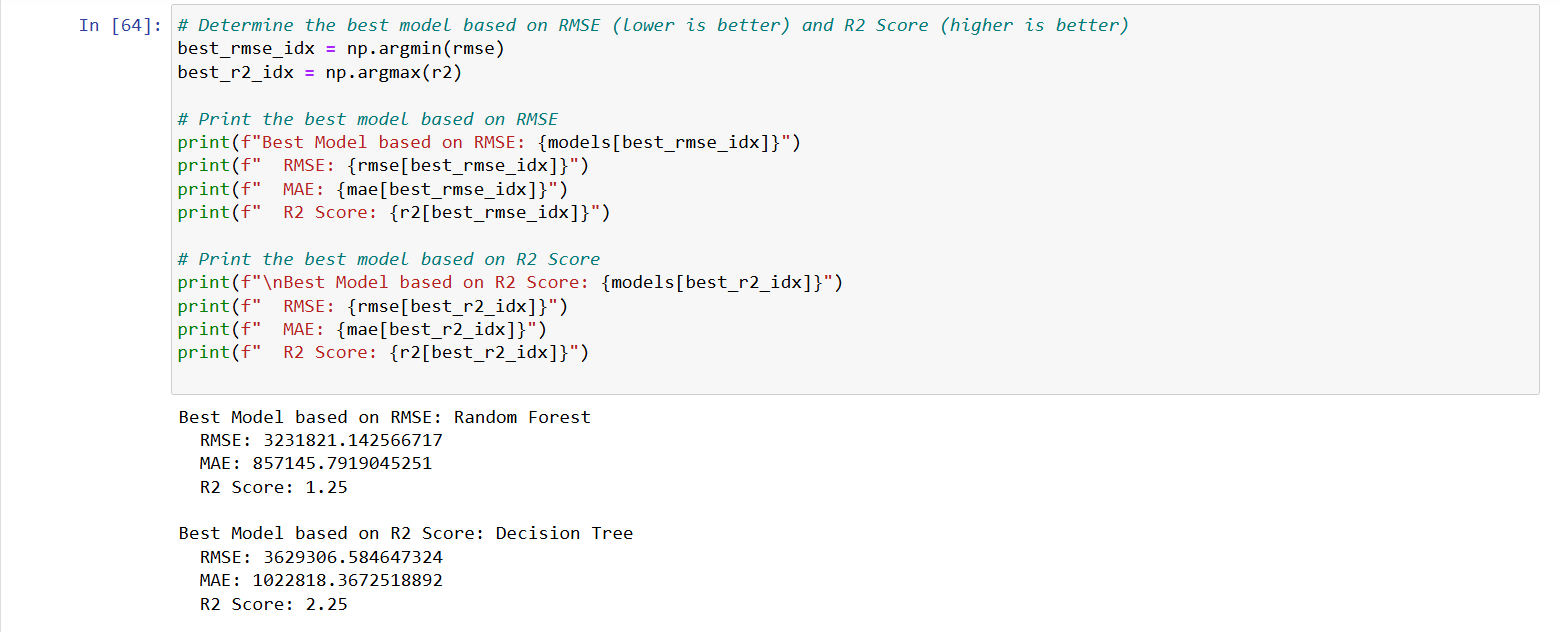
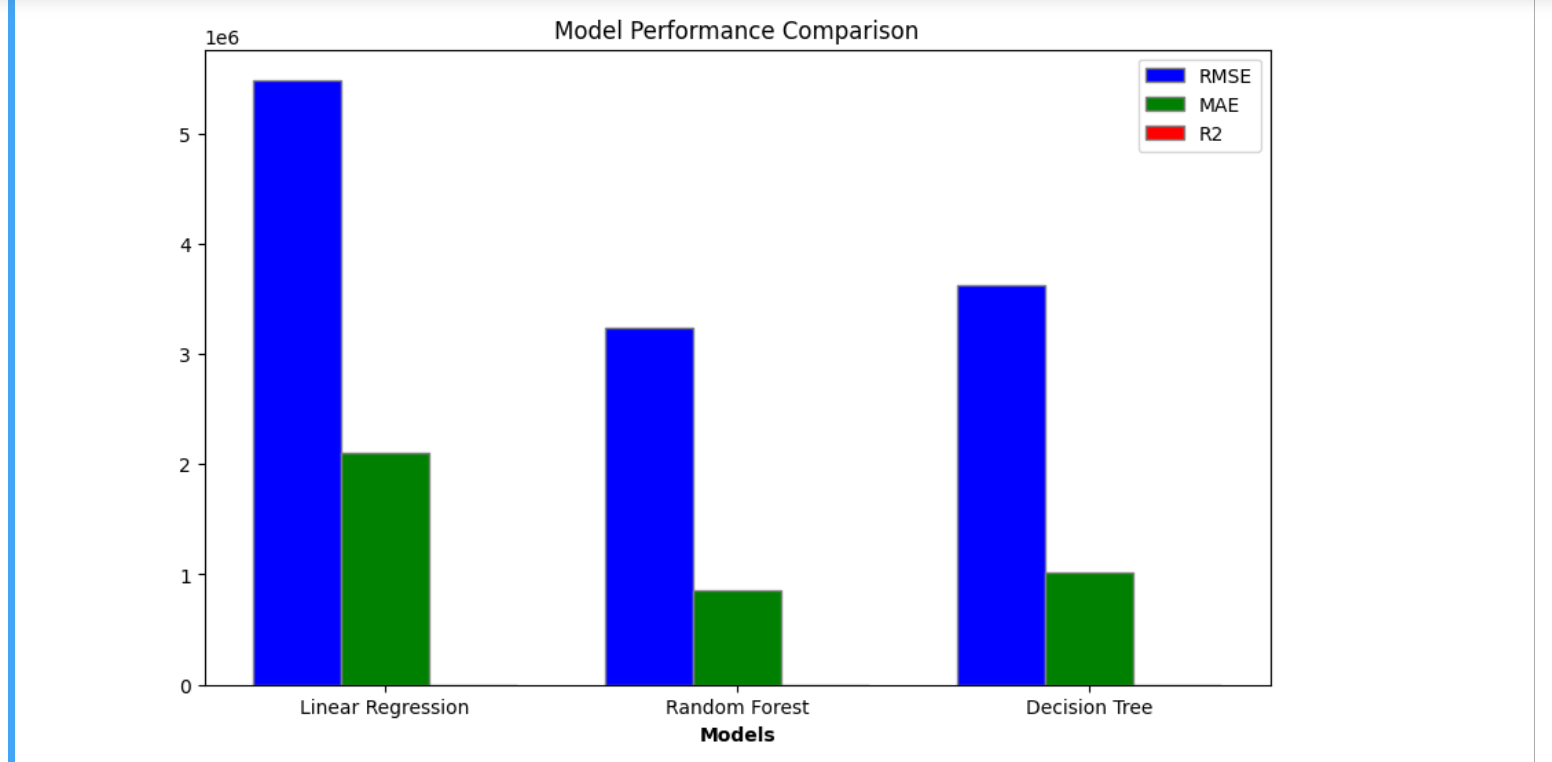
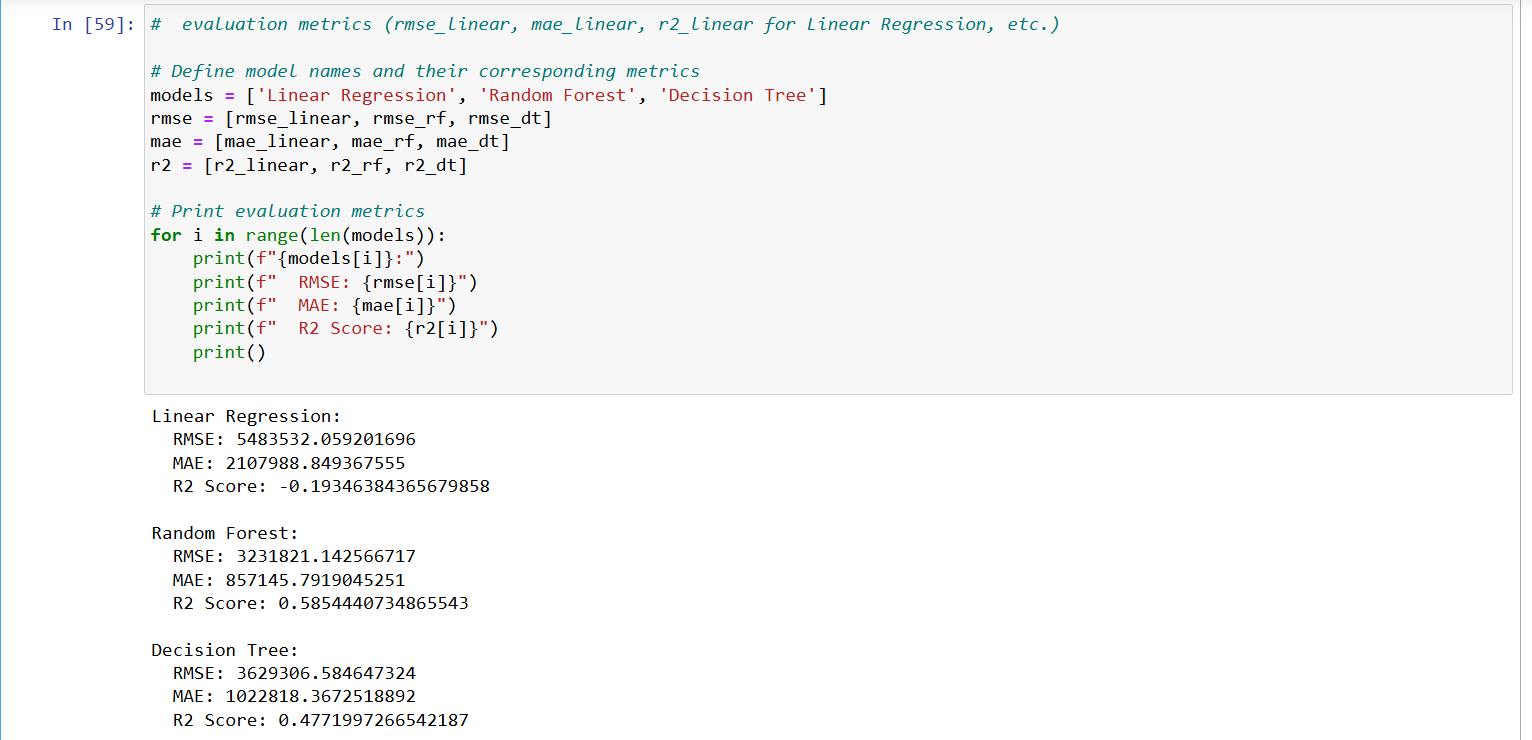
Task4- Model Evaluation

Evaluation Metrics: RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), R2 Score (Coefficient of Determination).

Visualization: A bar plot was created to compare the RMSE, MAE, and R2 Score of each model.

Best Model Selection: Determined based on the lowest RMSE and highest R2 Score.

Best Model**: Decision Tree** showed the lowest RMSE and highest R2 Score among the evaluated models.

****

**5. Prediction Function**

* **Create a Prediction Function:**
  + Create a function to take user inputs for **STATE**, **TYPE**, and **BEDS** and return the predicted **PRICE**.

# Function to preprocess user inputs into the format expected by the model

def preprocess\_input(state, property\_type, beds):

# Create a DataFrame with a single row containing user inputs

user\_data = pd.DataFrame({

'STATE': [state],

'TYPE': [property\_type],

'BEDS': [beds],

})

# Transform categorical and numerical variables using the same encoder/transformer used during training

try:

X = preprocessor.transform(user\_data)

except ValueError as e:

raise ValueError(f"Error during transformation: {e}")

return X

# Prediction function

def predict\_price(state, property\_type, beds, model):

# Preprocess user inputs

X = preprocess\_input(state, property\_type, beds)

# Predict price using the trained model

predicted\_price = model.predict(X)[0] # Assuming single prediction

return predicted\_price