ABSTRACT

OUR PROPOSED ABSTRACT:

In this project, we will be creating a model which will help us to predict the "REAL ESTATE PRICE PREDICTION." We will be training the model with 6 different algorithms to see which algorithm will give us the best performance ,based on that we will be using the same model for predicting the Real Estate Prices. This will help us to invest in the right place. For this we would be using several classification algorithms like KNN, SVM, Linear Regression, Logistic Regression, Decision Tree, Naive Bayes etc. the project employs the Random Forest algorithm and linear regression as its core predictive models. The objective is to accurately estimate the prices of residential properties, contributing to informed decision-making in the real estate market.

In this project, a dataset containing 7162 individual data points. The dataset encompasses various essential features that influence property prices, including location, square footage, number of bedrooms and bathrooms, amenities, and more. By leveraging this diverse set of attributes, the Random Forest and linear regression algorithm learns intricate patterns and relationships within the data, enabling it to make reliable predictions.

The Proposed Real Estate Price Prediction using Machine Learning showcases the efficacy of the Linear Regression and Random Forest algorithms in forecasting residential property prices. The Python-based implementation leverages a dataset comprising thousands of data points , contributing to a robust and reliable predictive model.The achieved low Mean Absolute Error values on both training and test sets emphasize the model's accuracy and generalization potential. This project holds significant implications for individuals, investors, and real estate professionals seeking data-driven insights to navigate the dynamic real estate market. Using this project several reportswill be generated to determine the accuracy of the model.

Existing System:

The **existing system focuses o**n the **algorithms such as Decision tree , Linear Regression to determine the accuracy.** These algorithms especially Linear Regression works well when we are working with samples of data. Linear Regression tend to train each tree independently, using a random sample of the data. This would help to make the model more robust than a single decision tree, and less likely to overfit on the training data. XGBoostbuild trees one at a time,

where each new tree helps to correct errors made by previously trained tree. With each tree that is

being added, the model becomes even more expressive

Disadvantages:

1. Lack of Accessibility
2. Overfitting can easily occur
3. Long process & Unpredictable market

Proposed System:

In the proposed system we will be using the same old LDA and Decision tree in addition to the we would using other algorithms like Support Vector Machine,K nearest Neighbours and Naïve Bayes to analyse the Real estate bussiness pratices.It helps us to work on the data which contains large samples,thereby helping us to determine the result accurately.These algorithms tend to be more reliable.

Advantages:

1. Accurate data
2. .No overfitting
3. High Accuracy

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Pentium i3 Processor.
- Hard Disk : 500 GB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- Ram : 6 GB.

SOFTWARE REQUIREMENTS:

- Operating system : Windows 10 / 11.
- Coding Language : Python 3.10.9
- Data set : MS EXCEL
- Software IDE : Anaconda Navigator(Spyder)
- Documentation : Microsoft Office

\

CHAPTER-1:

BASIC UNDERSTANDING OF CONCEPT

Machine Learning

The term ‘machine learning’ is often, incorrectly, interchanged with Artificial Intelligence, but machine learning is actually a subfield/type of AI. Machine learning is also often referred to as predictive analytics, or predictive modelling.

Coined by American computer scientist Arthur Samuel in 1959, the term ‘machine learning’ is defined as a “computer’s ability to learn without being explicitly programmed”.

At its most basic, machine learning uses programmed algorithms that receive and analyse input data to predict output values within an acceptable range. As new data is fed to these algorithms, they learn and optimise their operations to improve performance, developing ‘intelligence’ over time.

There are four types of machine learning algorithms: supervised, semi-supervised, unsupervised and reinforcement.

Supervised learning

In supervised learning, the machine is taught by example. The operator provides the machine learning algorithm with a known dataset that includes desired inputs and outputs, and the algorithm must find a method to determine how to arrive at those inputs and outputs. While the operator knows the correct answers to the problem, the algorithm identifies patterns in data, learns from observations and makes predictions. The algorithm makes predictions and is corrected by the operator – and this process continues until the algorithm achieves a high level of accuracy/performance.

Under the umbrella of supervised learning fall: Classification, Regression and Forecasting.

1. Classification: In classification tasks, the machine learning program must draw a conclusion from observed values and determine to
   what category new observations belong. For example, when filtering emails as ‘spam’ or ‘not spam’, the program must look at existing observational data and filter the emails accordingly.
2. Regression: In regression tasks, the machine learning program must estimate – and understand – the relationships among variables. Regression analysis focuses on one dependent variable and a series of other changing variables – making it particularly useful for prediction and forecasting.
3. Forecasting: Forecasting is the process of making predictions about the future based on the past and present data, and is commonly used to analyse trends.

Semi-supervised learning

Semi-supervised learning is similar to supervised learning, but instead uses both labelled and unlabelled data. Labelled data is essentially information that has meaningful tags so that the algorithm can understand the data, whilst unlabelled data lacks that information. By using this combination, machine learning algorithms can learn to label unlabelled data.

Unsupervised learning

Here, the machine learning algorithm studies data to identify patterns. There is no answer key or human operator to provide instruction. Instead, the machine determines the correlations and relationships by analysing available data. In an unsupervised learning process, the machine learning algorithm is left to interpret large data sets and address that data accordingly. The algorithm tries to organise that data in some way to describe its structure. This might mean grouping the data into clusters or arranging it in a way that looks more organised.

As it assesses more data, its ability to make decisions on that data gradually improves and becomes more refined.

Under the umbrella of unsupervised learning, fall:

1. Clustering: Clustering involves grouping sets of similar data (based on defined criteria). It's useful for segmenting data into several groups and performing analysis on each data set to find patterns.
2. Dimension reduction: Dimension reduction reduces the number of variables being considered to find the exact information required.

Reinforcement learning

Reinforcement learning focuses on regimented learning processes, where a machine learning algorithm is provided with a set of actions, parameters and end values. By defining the rules, the machine learning algorithm then tries to explore different options and possibilities, monitoring and evaluating each result to determine which one is optimal. Reinforcement learning teaches the machine trial and error. It learns from past experiences and begins to adapt its approach in response to the situation to achieve the best possible result.

CHAPTER 2:

OVERVIEW OF PROJECT

Overview of the Project:

This project aims to predict Real Estate Price Prediction using a linear regression and Decision tree models. The dataset usedis Realestate.csv, which contains various features related to our real estate works…

Objectives:

- **Accurate Price Prediction: Develop a robust model that can predict real estate prices with high accuracy.**
- **Feature Analysis**: Identify and analyze the key factors (features) that most significantly impact real estate prices.
- **Market Trend Analysis**: Understand and forecast market trends in different geographical locations.
- **Tool Development**: Create a user-friendly tool that stakeholders can use to input property features and receive price predictions.
- **Model Validation**: Validate the model against real-world data to ensure its reliability and accuracy.

Importance of Real Estate Price Prediction:

- **Investment Decisions**: Accurate price predictions help investors identify profitable investment opportunities and avoid overvalued properties.
- **Market Transparency**: Enhances market transparency by providing data-driven insights, reducing the likelihood of market bubbles.
- **Risk Management**: Helps financial institutions assess risks associated with real estate loans and investments.
- **Policy Making**: Assists policymakers in understanding market dynamics, enabling the formulation of policies to stabilize housing markets.
- **Consumer Empowerment**: Empowers homebuyers with information, helping them make informed decisions and negotiate better deals.

Background:

The real estate market is influenced by a myriad of factors, including economic conditions, interest rates, government policies, and local infrastructure developments. Traditional methods of price estimation often relied on expert opinions or basic statistical models, which could be subjective or overly simplistic.

With the advent of big data and machine learning, there has been a shift towards more sophisticated, data-driven approaches. Machine learning models can process large volumes of data and identify complex patterns that may not be apparent to human analysts. Techniques like regression analysis, decision trees, random forests, and neural networks are commonly used to predict real estate prices, often outperforming traditional methods.

This project taps into these advancements, aiming to create a model that can learn from historical data and continuously improve its predictions as new data becomes available.

CHAPTER 3:

INSTALLATION AND SETUP

Install Visual Studio Code:

1. **Download Visual Studio Code:**
    - Go to the official Visual Studio Code website.
    - Click on the “Download” button for your operating system.
2. **Run the Installer:**
    - Open the downloaded installer.
    - Follow the installation instructions.
    - Optionally, check the box to create a desktop icon.

Required Software and Libraries:

- Python
- numpy
- pandas
- matplotlib
- seaborn
- scikit-learn

Installation Instructions:

To install the required libraries, use the following pip commands:

“pip install numpy pandas matplotlib seaborn scikit-learn”

Setting Up the Environment:

Ensure that you have a suitable Python environment set up. You can use virtual environments to manage dependencies:

“python -m venv myenv

source myenv/bin/activate # On Windows use `myenv\Scripts\activate`”

CHAPTER 4:

DATA EXPLORATION

Loading the Dataset:

```
import pandas as pd
dataset =
pd.read_csv("Realestate.csv"
```

Initial Data Exploration:;

```
print(dataset.shape)
print(dataset.head(5))
```

Understanding the Dataset Structure:

The dataset consists of 511 entries and 14 columns. It appears to be related to real estate data, likely including various features that can influence property prices. Here's a breakdown of the columns:

1. **Unnamed: 0**: This seems to be an index column or an identifier, which may not be necessary for analysis.
2. **CRIM**: Per capita crime rate by town.
3. **INDUS**: Proportion of non-retail business acres per town.
4. **CHAS**: Charles River dummy variable (1 if the tract bounds the river; 0 otherwise).
5. **NOX**: Nitric oxide concentration (parts per 10 million).
6. **RM**: Average number of rooms per dwelling.
7. **AGE**: Proportion of owner-occupied units built prior to 1940.
8. **DIS**: Weighted distances to five Boston employment centers.
9. **RAD**: Index of accessibility to radial highways.
10. **TAX**: Full-value property tax rate per $10,000.
11. **PTRATIO**: Pupil-teacher ratio by town.
12. **B**: 1000(Bk−0.63)21000(Bk - 0.63)^21000(Bk−0.63)2 where BkBkBk is the proportion of Black residents by town.
13. **LSTAT**: Percentage of lower status of the population.
14. **MEDV**: Median value of owner-occupied homes in $1000s

Descriptive Statistics:

- • **CRIM**: Crime rates vary significantly across towns, with a mean of 11.25 and a maximum of 100, indicating some areas have exceptionally high crime rates.
- · **INDUS**: The proportion of non-retail business acres per town ranges from 0.46 to 27.74, with an average of 11.15.
- · **CHAS**: Most properties do not bound the Charles River, as indicated by a mean close to 0.
- · **NOX**: The nitric oxide concentration has a mean of 0.5548, with some areas experiencing significantly higher levels.
- · **RM**: The average number of rooms per dwelling is around 6.29, with some houses having as few as 3.56 rooms and others up to 8.78.
- · **AGE**: The proportion of older homes varies widely, with a mean of 68.6%.
- · **DIS**: Distances to employment centers also show significant variation, with a mean of 3.78.
- · **RAD**: The index of accessibility to radial highways ranges from 1 to 24, indicating varying levels of access.
- · **TAX**: Property tax rates vary considerably, with a mean of 407.44.
- · **PTRATIO**: The pupil-teacher ratio has a mean of 18.5, ranging from 12.6 to 23.
- · **B**: This variable indicates a wide range of proportions of Black residents across different towns.
- · **LSTAT**: The percentage of lower status residents ranges from 1.73% to 76%, with a mean of 12.88%.
- · **MEDV**: The median value of homes ranges from $5,000 to $67,000, with an average of $22,682.

```
print(dataset.describe())
```

Data Visualization:

Visualizing the data helps in understanding the distribution and relationships between features.

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.heatmap(dataset.corr(), annot=True, fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

CHAPTER 5:

DATA PREPROCESSING

Handling Missing Values:

```
dataset.isnull().sum()
```

If there are any missing values, they need to be handled appropriately, either by filling them with a suitable value or by removing the affected rows.

Splitting the Data:

```
from sklearn.model_selection import train_test_split

predictors = dataset.drop("target", axis=1)

target = dataset["target"]

X_train, X_test, Y_train, Y_test = train_test_split(predictors, target, test_size=0.20, random_state=0)
```

Feature Selection:

Selecting relevant features is crucial for building an effective model. In this project, all features except the target variable are used as predictors.

NOTE:

We had choosen MEDV as our dependent dataset because the MEDV column provides medium price of homes which makes it a natural choice as dependent variable model

CHAPTER 6:

MODEL TRAINING

Linear Regression & Decision Tree Models:

These linear regression and Decision Tree is a statistcal models to model the relationship between one or more independent models

Training the Model:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, Y_train)
```

Hyperparameter Tuning:

Hyperparameter tuning can improve the model's performance. Techniques such as Grid Search or Random Search can be used to find the best parameters.

1. Pre processing the Data:

- **Handle Missing Values:** The `RM` column has some missing values. You can either drop these rows or impute the missing values using the mean, median, or a more sophisticated method.
- **Feature Scaling:** Some algorithms (like SVM, KNN, and neural networks) perform better when features are scaled. Standardization or normalization may be applied.

2. Selecting a Model:

- **Regression Models:** Given the task is predicting the median value of homes, some common regression models include:
    - Linear Regression
    - Decision Trees
    - Random Forest
    - Gradient Boosting Machines (e.g., XGBoost, LightGBM)
    - Support Vector Machines (SVM)
    - Neural Networks

3. Splitting the Data:

- Split the data into training and test sets (e.g., 80% training, 20% test) to evaluate the model's performance.

4. Hyper parameter Tuning Methods:

- **Grid Search:** Tests all combinations of hyper parameters. It's exhaustive but can be computationally expensive.
- **Random Search:** Randomly selects combinations of hyper parameters to test. It's less exhaustive but can be faster.
- **Bayesian Optimization:** Uses past evaluations to choose the next set of hyper parameters to test. It's more efficient but requires more sophisticated implementation.
- **Cross-Validation:** Use k-fold cross-validation during the search to ensure the model generalizes well.

CHAPTER 7:

MODEL EVALUATION

Making Predictions:

```
Y_pred_lr = lr.predict(X_test)
```

Evaluating Model Performance:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

score_lr = round(accuracy_score(Y_pred_lr, Y_test) * 100, 2)

print("The accuracy score achieved using Linear  Regression is: " + str(score_lr) + " %")
```

Confusion Matrix:

```
cm = confusion_matrix(Y_test, Y_pred_lr)

sns.heatmap(cm, annot=True, fmt="d")

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()
```

Classification Report:

```
print(classification_report(Y_test, Y_pred_lr))
```

CHAPTER 8:

VISUALIZATION

Accuracy Plot:

```
plt.figure(figsize=(8,6))
plt.bar(["Linear  Regression"], [score_lr])
plt.xlabel("Model")
plt.ylabel("Accuracy (%)")
plt.title("Model Accuracy")
plt.show()
```

Target Count Plot:

```
plt.figure(figsize=(8,6))
sns.countplot(x="target", data=dataset)
plt.xlabel("Target")
plt.ylabel("Count")
plt.title("Target Count")
plt.show()
```

Feature Importance:

Understanding which features are most important can provide insights into the model's decision-making process.

```
importance = lr.coef_[0]
feature_importance = pd.Series(importance, index=predictors.columns).sort_values(ascending=False)
feature_importance.plot(kind='bar')
plt.title("Feature Importance")
plt.show()
```

CHAPTER 9:
FINAL FULL CODE

1.LiNEAR REGRESSION ALGORITHM

```
#importing the requirements
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
dataset = pd.read_csv('realestate_priceprediction.csv')
X = dataset.iloc[:, [2, 10]].values  # Assuming these are the
relevant features
y = dataset.iloc[:, 13].values  # Assuming this is the target variable

# Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize the linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)

# Plot the actual vs predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', lw=2)  # Diagonal line
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()
```

2.DECISION TREE REGRESSION ALGORITHM

```
#importing the requirements
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
dataset = pd.read_csv('realestate_priceprediction.csv')
X = dataset.iloc[:, [3,1]].values  # Selecting relevant columns
y = dataset.iloc[:, 13].values  # Target variable

# Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize the Decision Tree Regressor model
model = DecisionTreeRegressoion(random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

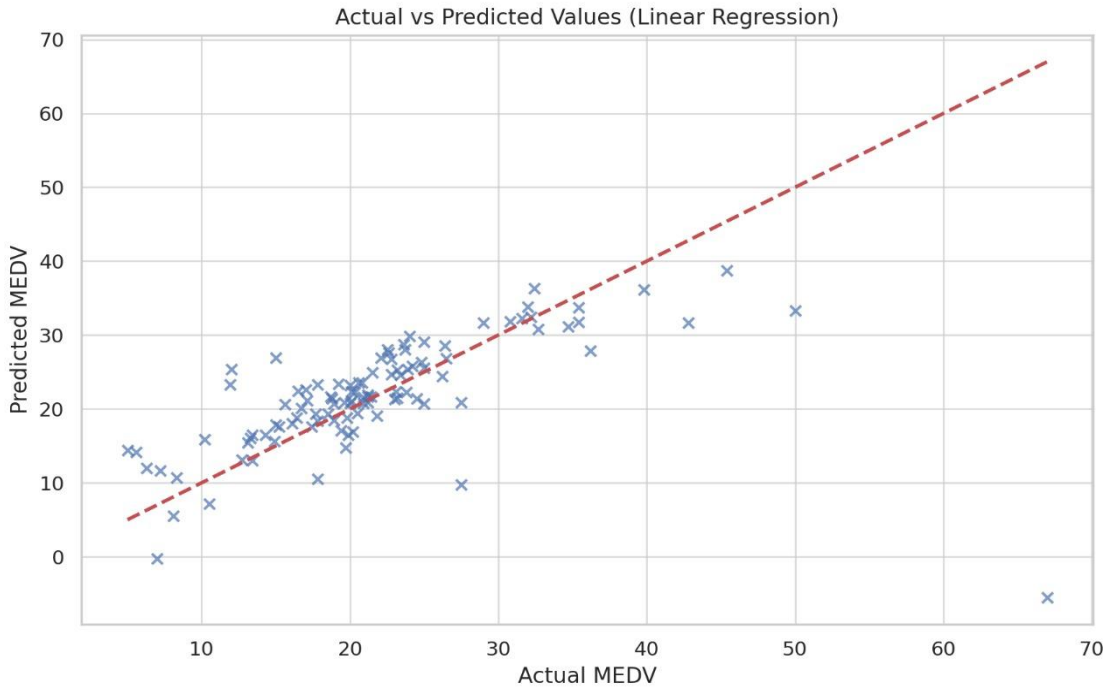
```
print("Mean Squared Error:", mse)
print("R-squared:", r2)

# Plot the actual vs predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', lw=2, label='Perfect Fit Line')  # Diagonal line
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Decision Tree: Actual vs Predicted Values')
plt.legend()
plt.show()
```

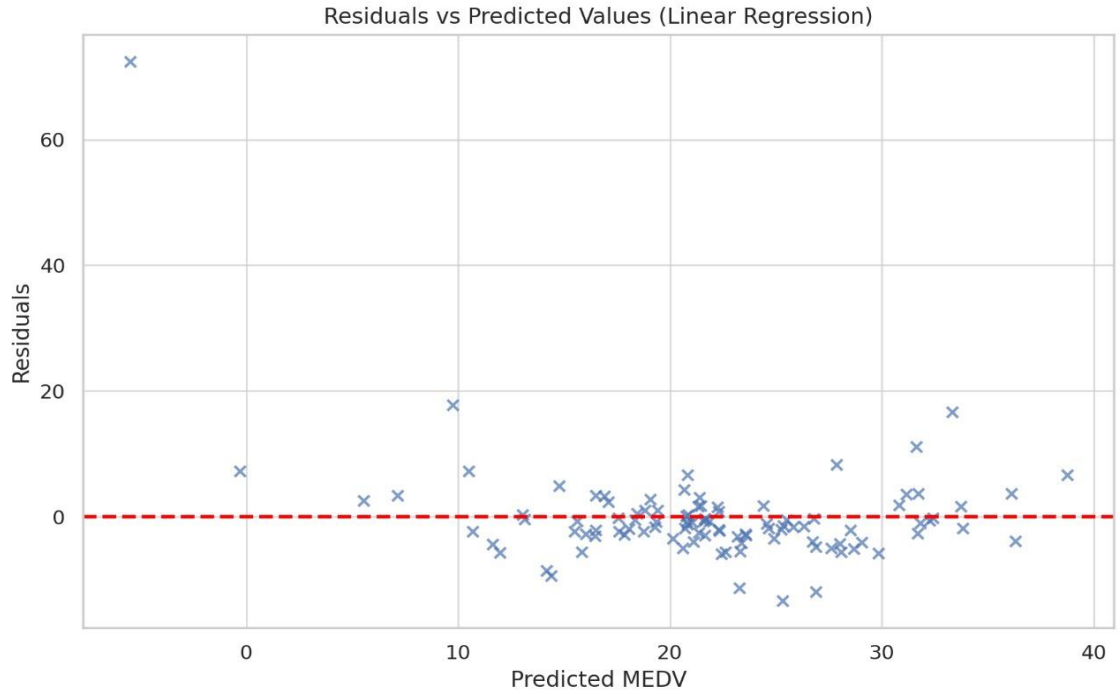
CHAPTER 10:

OUTPUT

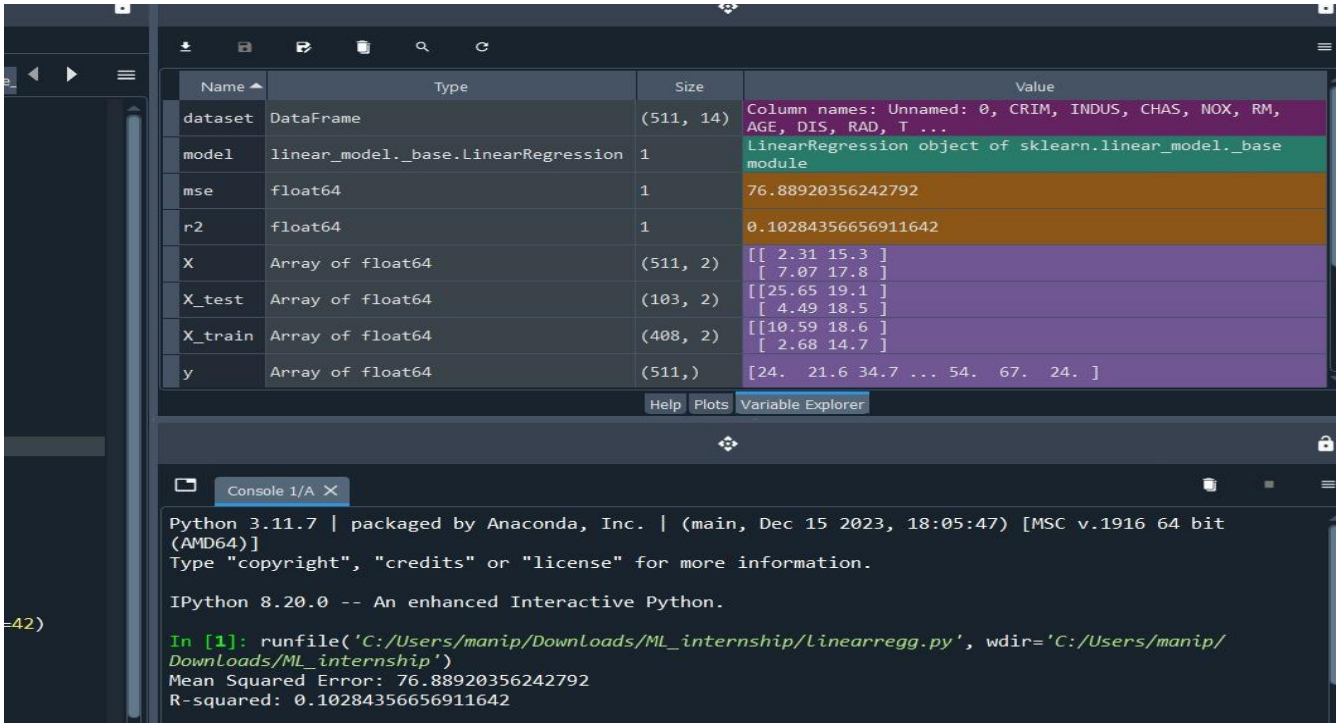
MODEL ACCURACY:



TARGET COUNT:



ACCURACY SCREEN SHOT FOR LINEAR REGRESSION



Name	Type	Size	Value
dataset	DataFrame	(511, 14)	Column names: Unnamed: 0, CRIM, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, T ...
model	linear_model._base.LinearRegression	1	LinearRegression object of sklearn.linear_model._base module
mse	float64	1	76.88920356242792
r2	float64	1	0.10284356656911642
X	Array of float64	(511, 2)	[[2.31 15.3] [7.07 17.8]
X_test	Array of float64	(103, 2)	[[25.65 19.1] [4.49 18.5]
X_train	Array of float64	(408, 2)	[[10.59 18.6] [2.68 14.7]
y	Array of float64	(511,)	[24. 21.6 34.7 ... 54. 67. 24.]

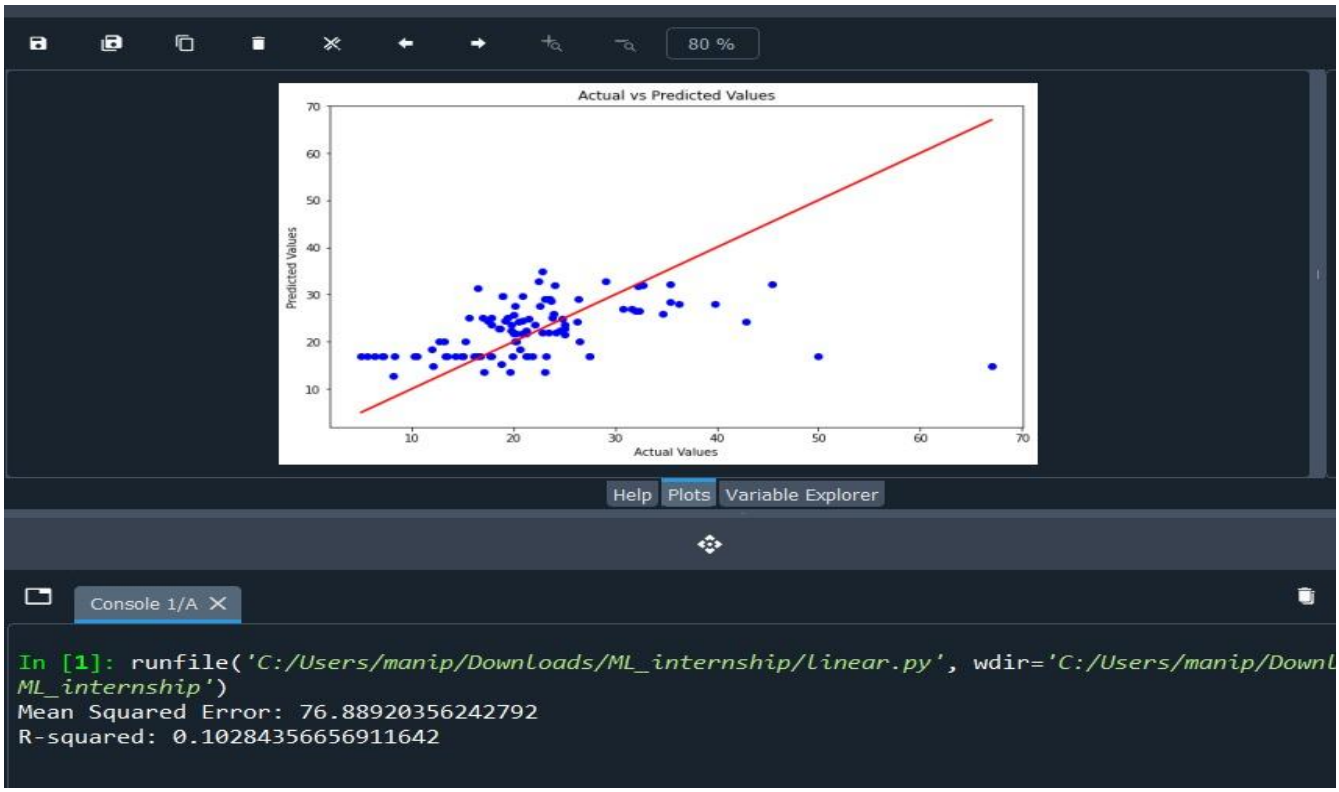
Help Plots Variable Explorer

Console 1/A

```
Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.20.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/manip/Downloads/ML_internship/linearregg.py', wdir='C:/Users/manip/Downloads/ML_internship')
Mean Squared Error: 76.88920356242792
R-squared: 0.10284356656911642
```



ACCURACY SCREEN SHOT FOR DECISION TREE REGRESSION

Name	Type	Size	Value
dataset	DataFrame	(511, 14)	Column names: Unnamed: 0, CRIM, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, T ...
model	tree._classes.DecisionTreeRegressor	1	DecisionTreeRegressor object of sklearn.tree._classes module
mse	float64	1	73.58233624941838
r2	float64	1	0.14142866235515916
X	Array of float64	(511, 2)	[[2.31 15.3] [7.07 17.8]
X_test	Array of float64	(103, 2)	[[25.65 19.1] [4.49 18.5]
X_train	Array of float64	(408, 2)	[[10.59 18.6] [2.68 14.7]
y	Array of float64	(511,)	[24. 21.6 34.7 ... 54. 67. 24.]

Help Plots Variable Explorer

Console 2/A

R-squared: 0.19668719707837123

```
In [9]: runfile('C:/Users/manip/Downloads/ML_internship/untitled1.py', wdir='C:/Users/manip/Downloads/
ML_internship')
Mean Squared Error: 73.8053805800622
R-squared: 0.13882614279576577
```

CHAPTER 11:

CONCLUSION AND FUTURE WORK

Summary of Findings:

The linear regression model achieved an accuracy of 76.88**%** in predicting Real Estate prices
The Decision Tree regression model achieved an accuracy of 73.80**%** in predicting Real Estate prices

Potential Improvements:

- Exploring different machine learning models.
- Tuning hyperparameters for better performance.
- Using more advanced techniques like cross-validation.

Future Work:

- Implementing additional models and comparing their performance.
- Collecting more data to improve model accuracy.
- Integrating the model into a web application for real-time predictions.