PREDICTING HOUSE PRICES USING MACHINE LEARNING

Objective:

To predict the house prices using Machine Learning.

PROBLEM UNDERSTANDING :

The problem of predicting house prices is complex, as it is influenced by a variety of factors. Machine learning can be used to develop models that can learn from historical data to predict future house prices.

DESIGN CONSIDERATIONS:

- The following are some design considerations for predicting house prices using machine learning:

- **Data Preparation:** The quality of the dataset is crucial to the success of the machine learning model. The data should be cleaned and preprocessed to remove any errors or inconsistencies. Additionally, the data should be transformed into a format that is compatible with the machine learning algorithm.
- **Feature Engineering:** Feature engineering involves creating new features from existing features or transforming existing features in a way that improves the performance of the machine learning model.
- **Model Selection:** There are a variety of machine learning algorithms that can be used to predict house prices. The best algorithm to use will depend on the specific characteristics of the dataset.
- **Model Training:** Once a machine learning algorithm has been selected, the model needs to be trained on the historical data. The training process involves adjusting the parameters of the model to minimize the error on the training data.
- **Model Evaluation:** Once the model has been trained, it needs to be evaluated on a held-out test set to assess its performance on unseen data.

SOLUTION APPROACH:

- The following is a high-level overview of the approach that will be used to predict house prices using machine learning:

- **Data Preparation:** The dataset will be cleaned and preprocessed to remove any errors or inconsistencies. Additionally, the data will be transformed into a format that is compatible with the machine learning algorithm.
- **Feature Engineering:** New features may be created from existing features or existing features may be transformed in a way that improves the performance of the machine learning model.
- **Model Selection:** A variety of machine learning algorithms will be considered for the task of predicting house prices. The best algorithm to use will be selected based on the performance of the algorithms on the training data.

- **Model Training:** The selected machine learning algorithm will be trained on the training data. The training process will involve adjusting the parameters of the model to minimize the error on the training data.
- **Model Evaluation:** The trained model will be evaluated on a held-out test set to assess its performance on unseen data.

Conclusion:

Once the machine learning model has been trained and evaluated, it can be used to predict the prices of new houses. The model can also be used to identify factors that are most influential in determining house prices.

Phases of Development:

- **Data collection.**
- **Pre Processing.**
- **Data Analysis.**
- **Application of Algorithm.**
- **Evaluating the models.**

Collection of Data:

Data processing techniques and processes are numerous.we collected data for having attributes such as location,carpet area,built-up area,age of the property,zipcode, price etc…

Loading and preparing Datasets:

- Here I'm loading the given data set using pandas dataframe.
- Utilizing read excel function for reading the dataset.

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import hvplot.pandas
```

- Load a dataset from a CSV file named 'USA_Housing.csv' located in the Googe Drive input folder
- Display the first few rows of the dataset

```
ds=pd.read_csv("//content/drive/MyDrive/USA_Housing.csv")

ds.head()
```

Preprocessing the Data:

- Create a new DataFrame 'df' by dropping the 'Address' column from the 'ds' DataFrame.
- Display the first few rows of the modified 'df' DataFrame

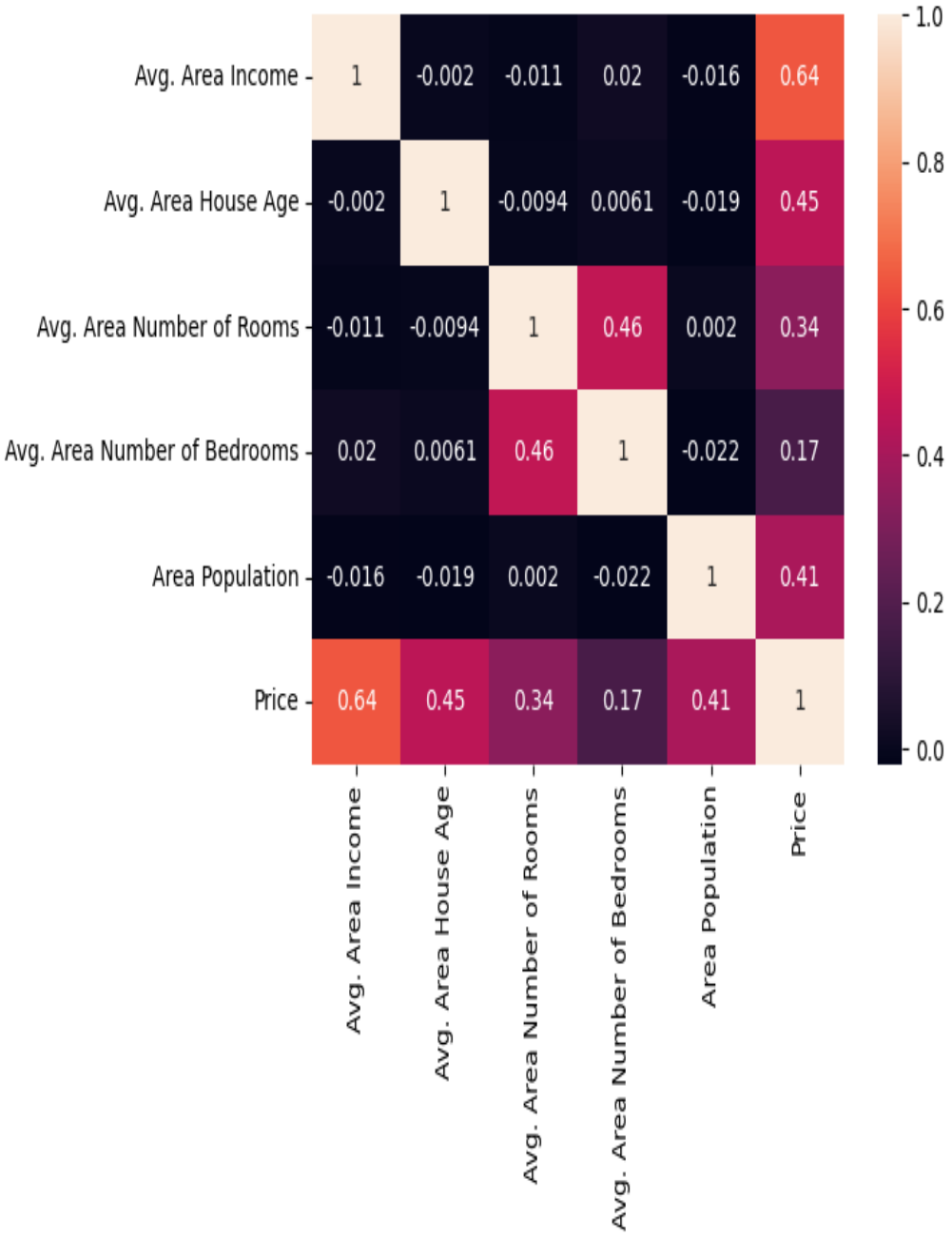
```
df=ds.drop(['Address'],axis=1)

df.head()
```

- Create a heatmap to visualize the correlation between columns in the 'df' DataFrame and annotate the cells with correlation values

```
sns.heatmap(df.corr(),annot=True)
```

OUTPUT:



- Create a new DataFrame 'x' by dropping the 'Price' column from the 'df' DataFrame
- Create a Series 'y' containing the 'Price' column from the 'df' DataFrame

```
x=df.drop(['Price'],axis=1)
y=df['Price']
```

- Print the column names of the 'x' DataFrame

```
x.columns
```

OUTPUT:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population'],
      dtype='object')
```

Data Analysis:

Let us analyse the data from the given data set of data frame.since the data is broken into two modules during the model training.The training set includes the target variable.The decision tree regressor algorithm is applied to the trained data set. The decision tree builds a regression model .

- Create a new DataFrame 'df' by dropping the 'Address' column from the 'ds' DataFrame.
- Display the first few rows of the modified 'df' DataFrame

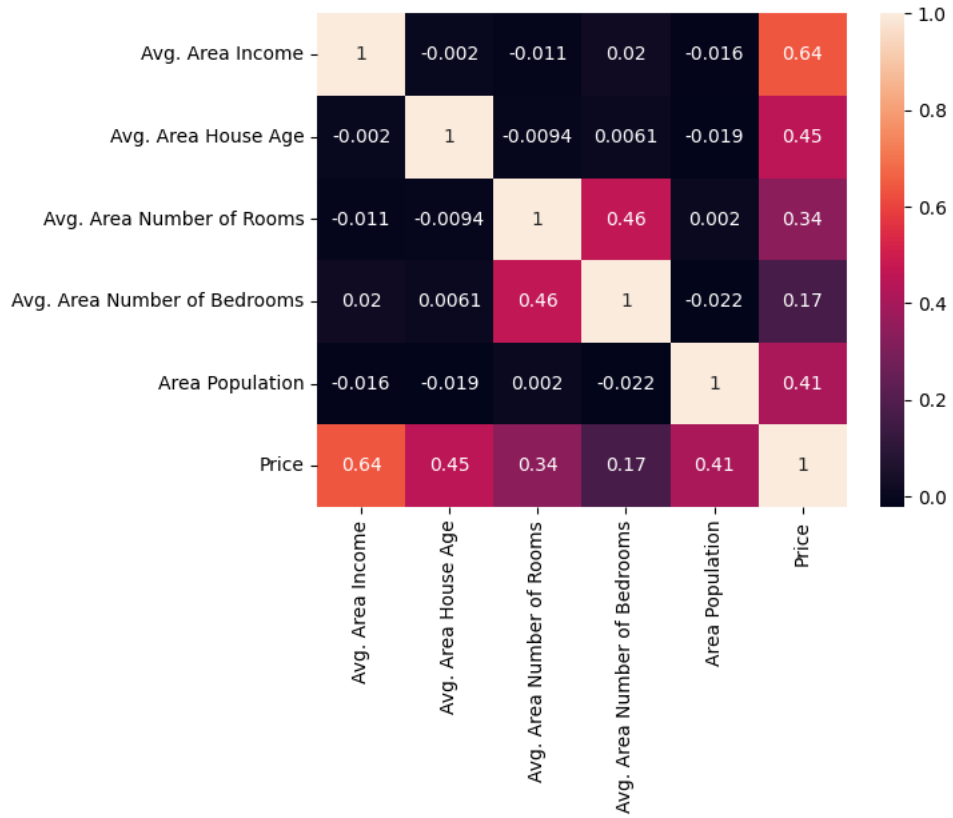
```
df=ds.drop(['Address'],axis=1)
df.head()
```

- Create a heatmap to visualize the correlation between columns in the 'df' DataFrame and annotate the cells with correlation values

```
sns.heatmap(df.corr(),annot=True)
```

<Axes: >

OUTPUT:



- Create a new DataFrame 'x' by dropping the 'Price' column from the 'df' DataFrame
- Create a Series 'y' containing the 'Price' column from the 'df' DataFrame

```
x=df.drop(['Price'],axis=1)
y=df['Price']
```

- Print the column names of the 'x' DataFrame

```
x.columns
```

OUTPUT:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population'],
dtype='object')
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3, random_state=42)
```

```
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

def cross_val(model):
    pred=cross_val_score(model,x,y,cv=10)
    return pred.mean()

def print_evaluate(true, predicted):
    mae=metrics.mean_absolute_error(true,predicted)
    mse=metrics.mean_squared_error(true,predicted)
    rmse=np.sqrt(metrics.mean_squared_error(true,predicted))
    r2_square=metrics.r2_score(true,predicted)
    print('MAE: ',mae)
    print('MSE: ',mse)
    print('RMSE: ',rmse)
    print('R2 Square',r2_square)
    print('_____')

def evaluate(true, predicted):
    mse =metrics.mean_squared_error(true, predicted)
    mae =metrics.mean_absolute_error(true, predicted)
    rmse= np.sqrt(metrics.mean_squared_error(true,predicted))
    r2=square_metrics.r2_score (true, predicted)
    return mae, mse, rmse, r2_square

pipeline=Pipeline([('std_scalar',StandardScaler())])
x_train=pipeline.fit_transform(x_train)
x_test=pipeline.transform(x_test)
```

```
#using LinearRegression
from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
lin_reg.fit(x_train,y_train)

test_pred=lin_reg.predict(x_test)
train_pred=lin_reg.predict(x_train)

print('Test set evaluation:\n_____')
print_evaluate(y_test,test_pred)
print('Train set evaluation:\n_____')
print_evaluate(y_train,train_pred)
```

OUTPUT:

Test set evaluation:

_____

MAE: 81135.56609336878
MSE: 10068422551.40088
RMSE: 100341.52954485436
R2 Square 0.9146818498754016

_____

Train set evaluation:

_____

MAE: 81480.49973174892
MSE: 10287043161.197224
RMSE: 101425.06180031257
R2 Square 0.9192986579075526

```
#using RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
rf_reg=RandomForestRegressor(n_estimators=1000)
rf_reg.fit(x_train,y_train)

test_pred=rf_reg.predict(x_test)
train_pred=rf_reg.predict(x_train)

print('Test set evaluation:\n_____')
print_evaluate(y_test,test_pred)
print('Train set evaluation:\n_____')
print_evaluate(y_train,train_pred)
```

OUTPUT:

Test set evaluation:

_____

MAE: 94148.20645204713
MSE: 14139722963.736708
RMSE: 118910.5670818902
R2 Square 0.8801823224659487

_____

Train set evaluation:

_____

MAE: 35266.36426066166
MSE: 1985024479.8320453
RMSE: 44553.613544044274
R2 Square 0.9844275816579577

ALGORITHM BRIEF OUTLINE:

- Import the python libraries that are required for house prices prediction using linear regression, Ex . numpy is used for convention of data to 2d or 3d array format which is required for linear regression model,matplotlib for plotting graphs,pandas for reading the data from source and manipulation that data ,etc…

- Firse Get the value from source and give it to a data frame and then manipulate this data to require from usind head(), indexing,drop().

- Next we have to train a model its always best to split the data into training data and test data for modelling .

- Its always good to use shapes () to avoid null spaces which will cause error during modelling process.

- Its good to normalize the value since the value are in very large quantity for houses price, for this we may use minmax scaler to reduce the gap between prices so thst its easy and less time consuming comparing and values range usually specified is between 0 to 1 usind fittrans form.

- Then, we have to make few imports for keras like: sequential for initializing the network,1stm to add 1stm layer,dropout to prevent overfitting of layers,dense to add densely connected network layer for output unit.

- In 1stm layer declaration is best to declare the unit, activation, return sequence.

- To compile this model is always best to use adam optimizer and set the loss as required for the specified data.

- Then we can use matplotlib to plot a graph comparing the test predicted value to see the increase and decrease rate of values in each time of the year in a particular place. Based on the people will know when its best time sell or buy a place in a given location.

CONCLUSION:

- Thus the Machine Learning model to predict house prices based on given data set is executed successfully using XG regressor(a upgraded/slighted boosted form of regular liner regression,this gives lesser error).this model further helps people understand whether this place is more suited for them based on heatmap correlation .It also helps people looking to sell a house at best time for greater profit. Any house price in any location can be predicted within minimum error by giving appropriate dataset.

SOFTWARE TOOLS USED

- Keras
- Jupyter
- Visual studio
- R Square
- Adjusted R Square
- MSE
- RMSE
- MAE
- Google colla

LIBRARIES USED:

- Numpy
- Pandas
- Pyplot from matplotlib
- Mpld3