**Predicting House Prices Using Machine Learning**

**phase 4: Model training and Evaluation**

**Introduction:**

In this phase4 we train and test our given dataset US-Housing (<https://www.kaggle.com/datasets/vedavyasv/usa-housing>)using one machine learning algorithm Random Forest

**Random Forest:**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, **"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."**  Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**

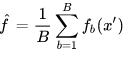
**Bagging:**

The training algorithm for random forests applies the general technique of [bootstrap aggregating](https://en.wikipedia.org/wiki/Bootstrap_aggregating), or bagging, to tree learners. Given a training set *X* = *x1*, ..., *xn* with responses *Y* = *y1*, ..., *yn*, bagging repeatedly (*B* times) selects [random sample with replacement](https://en.wikipedia.org/wiki/Sampling_(statistics)#Replacement_of_selected_units) of the training set and fits trees to these samples:

For *b* = 1, ..., *B*:

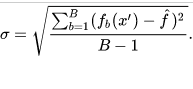
1. Sample, with replacement, *n* training examples from *X*, *Y*; call these *Xb*, *Yb*.
2. Train a classification or regression tree *fb* on *Xb*, *Yb*.

After training, predictions for unseen samples *x'* can be made by averaging the predictions from all the individual regression trees on *x'*:



This bootstrapping procedure leads to better model performance because it decreases the  [variance](https://en.wikipedia.org/wiki/Bias%E2%80%93variance_dilemma)  of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on *x'*:

**Model Training:** 

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the "../input/" directory.*

*# For example, running this (by clicking run or pressing Shift+Enter) will list files in the input directory*

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import explained\_variance\_score

from sklearn.metrics import confusion\_matrix

import os

print(os.listdir("../input"))

import warnings

warnings.filterwarnings('ignore')

*# Any results you write to the current directory are saved as output.*

['kc\_house\_data.csv']

*# X(Independent variables) and y(target variables)*

X = dataset.iloc[:,1:].values

y = dataset.iloc[:,0].values

In [30]:

*#Splitting the data into train,test data*

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=0)

***Multiple Linear Regression:***

mlr = LinearRegression()

mlr.fit(X\_train,y\_train)

mlr\_score = mlr.score(X\_test,y\_test)

pred\_mlr = mlr.predict(X\_test)

expl\_mlr = explained\_variance\_score(pred\_mlr,y\_test)

***Decision Tree***

tr\_regressor = DecisionTreeRegressor(random\_state=0)

tr\_regressor.fit(X\_train,y\_train)

tr\_regressor.score(X\_test,y\_test)

pred\_tr = tr\_regressor.predict(X\_test)

decision\_score=tr\_regressor.score(X\_test,y\_test)

expl\_tr = explained\_variance\_score(pred\_tr,y\_test)

***Random Forest Regression Model***

rf\_regressor = RandomForestRegressor(n\_estimators=28,random\_state=0)

rf\_regressor.fit(X\_train,y\_train)

rf\_regressor.score(X\_test,y\_test)

rf\_pred =rf\_regressor.predict(X\_test)

rf\_score=rf\_regressor.score(X\_test,y\_test)

expl\_rf = explained\_variance\_score(rf\_pred,y\_test)

***Calculate Model Score***

Let's calculate the model score to understand how our model performed along with the explained variance score.

print("Multiple Linear Regression Model Score is ",round(mlr.score(X\_test,y\_test)\*100))

print("Decision tree Regression Model Score is ",round(tr\_regressor.score(X\_test,y\_test)\*100))

print("Random Forest Regression Model Score is ",round(rf\_regressor.score(X\_test,y\_test)\*100))

*#Let's have a tabular pandas data frame, for a clear comparison*

models\_score =pd.DataFrame({'Model':['Multiple Linear Regression','Decision Tree','Random forest Regression'],'Score':[mlr\_score,decision\_score,rf\_score], 'Explained Variance Score':[expl\_mlr,expl\_tr,expl\_rf]})

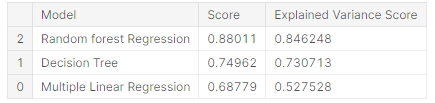
models\_score.sort\_values(by='Score',ascending=False)

**Output:**

Multiple Linear Regression Model Score is 69.0

Decision tree Regression Model Score is 75.0

Random Forest Regression Model Score is 88.0



**Evaluation:**

* !pip install -q hvplot

import pandas as pd

import numpy as np

import seaborn as sns

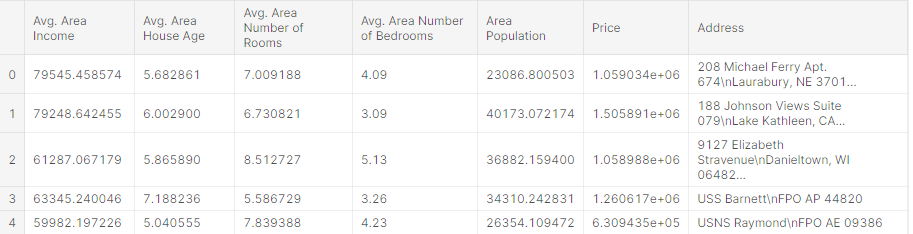
import matplotlib.pyplot as plt

import hvplot.pandas

* ds=pd.read\_csv("/kaggle/input/usa-housing/USA\_Housing.csv)

ds.head()

**Output:**

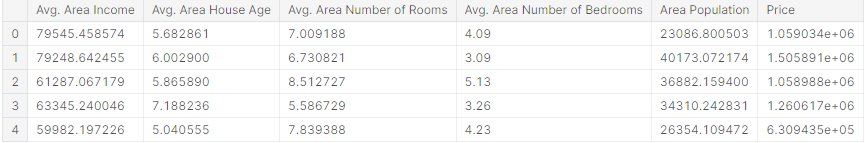


Dropping of the particular column value**:**

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

df.head()

**Output:**



* data.info()

**Output:**

**<**class 'pandas.core.frame.DataFrame'>

RangeIndex: 5000 entries, 0 to 4999

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Avg. Area Income 5000 non-null float64

1 Avg. Area House Age 5000 non-null float64

2 Avg. Area Number of Room 5000 non-null float64

3 Avg. Area Number of Bedrooms 5000 non-null float64

4 Area Population 5000 non-null float64

5 Price 5000 non-null float64

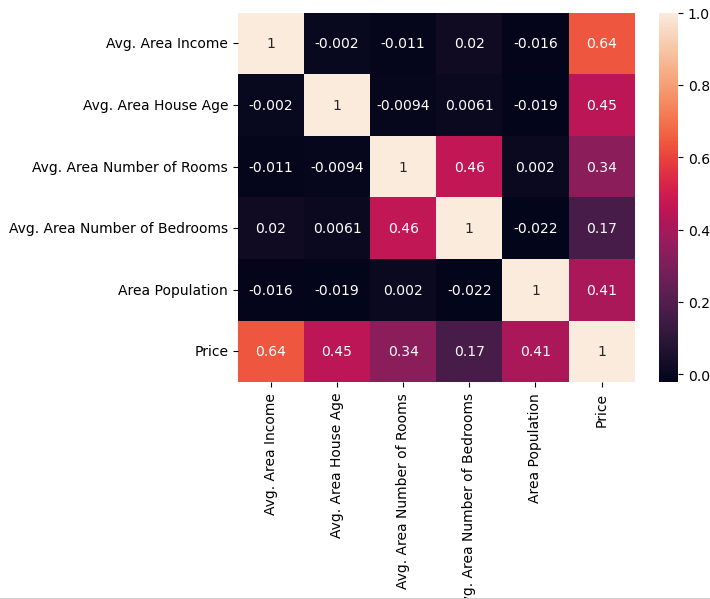
6 Address 5000 non-null object

dtypes: float64(6), object(1)

memory usage: 273.6+ KB

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

**Output:**



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

y=df['Price']

x.column

**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: 94027.38848972939

MSE: 14117343785.892136

RMSE: 118816.42893931856

R2 Square 0.8803719599235804

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Train set evaluation:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MAE: 35362.54528067692

MSE: 1989422953.623089

RMSE: 44602.94781315568

R2 Square 0.9843930758497742

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_