DSBDA LAB 7

Ravitej.R
160119733160

AIM:- To apply various ML classification algorithms on a dataset and compare its efficiency

DESCRIPTION:-

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned.Random decision forests correct for decision trees' habit of overfitting to their training set.

CODE:-

#reading and description the data

```
import pandas as pd
df = pd.read_csv('melb_data.csv')
# print a summary of the data in Melbourne data
df.describe()
#desicion TREE
melbourne_features = ['Rooms','Landsize', 'Lattitude', 'Longtitude','BuildingArea']
#selecting features
X =df[melbourne_features]
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

```
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state = 0)

melbourne_model = DecisionTreeRegressor()
melbourne_model.fit(train_X, train_y)

# get predicted prices on validation data
val_predictions = melbourne_model.predict(val_X)
print(melbourne_model.score(val_predictions,val_y))
print(mean_absolute_error(val_y, val_predictions))
#random forest
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

forest_model = RandomForestRegressor(random_state=1)
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
print(forest_model.score(melb_preds,val_y))
print(mean_absolute_error(val_y, melb_preds))
# naive bayes
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train)
pred=y_pred.predict(val_x)
print(gnb.score(pred,val_y))
print(mean_asolute error(val_y,pred))
```

OUTPUT:-

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	Lattitude	Longtitude	Propertycount
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	13580.000000	13518.000000	13580.000000	7130.000000	8205.000000	13580.000000	13580.000000	13580.000000
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	1.534242	1.610075	558.416127	151.967650	1964.684217	-37.809203	144.995216	7454.417378
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	0.691712	0.962634	3990.669241	541.014538	37.273762	0.079260	0.103916	4378.581772
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1196.000000	-38.182550	144.431810	249.000000
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	1.000000	1.000000	177.000000	93.000000	1940.000000	-37.856822	144.929600	4380.000000
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	1.000000	2.000000	440.000000	126.000000	1970.000000	-37.802355	145.000100	6555.000000
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	2.000000	2.000000	651.000000	174.000000	1999.000000	-37.756400	145.058305	10331.000000
max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	8.000000	10.000000	433014.000000	44515.000000	2018.000000	-37.408530	145.526350	21650.000000

Description of data used

```
250746.88766946417 0.616398
```

MSE AND F SCORE OF DECISION TREE

185165.77777777777 0.73846

MSE AND F SCORE OF RANDOM FOREST

30479.283408642371 0.214546

MSE AND F SCORE OF NAIVE BAYES

AS WE CAN SEE RANDOM FOREST IS THE MOST EFFICIENT ALGORITHM FOR THE GIVEN DATA SET FOLLOWED BY DECISION TREE AND NAIVE BAYES RESPECTIVELY IN BOTH CATEGORIES i.e MSE(MEAN SQUARE ERROR) AND FSCORE