2/09/2025

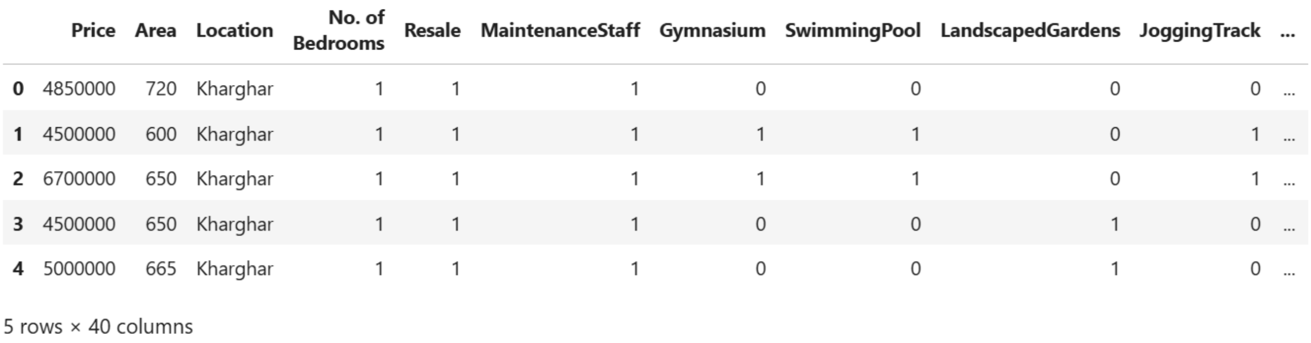
Experiment 6:

Write a Python program to implement Multiple Linear Regression and Decision Tree.

# Implementation of Multiple Linear Regression

```
import pandas as pd
import numpy as np
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 import metrics
Mhousing=pd.read_csv("C:/Users/mitha/Desktop/ML/Mumbai.csv")
Mhousing.head()
```

Output:



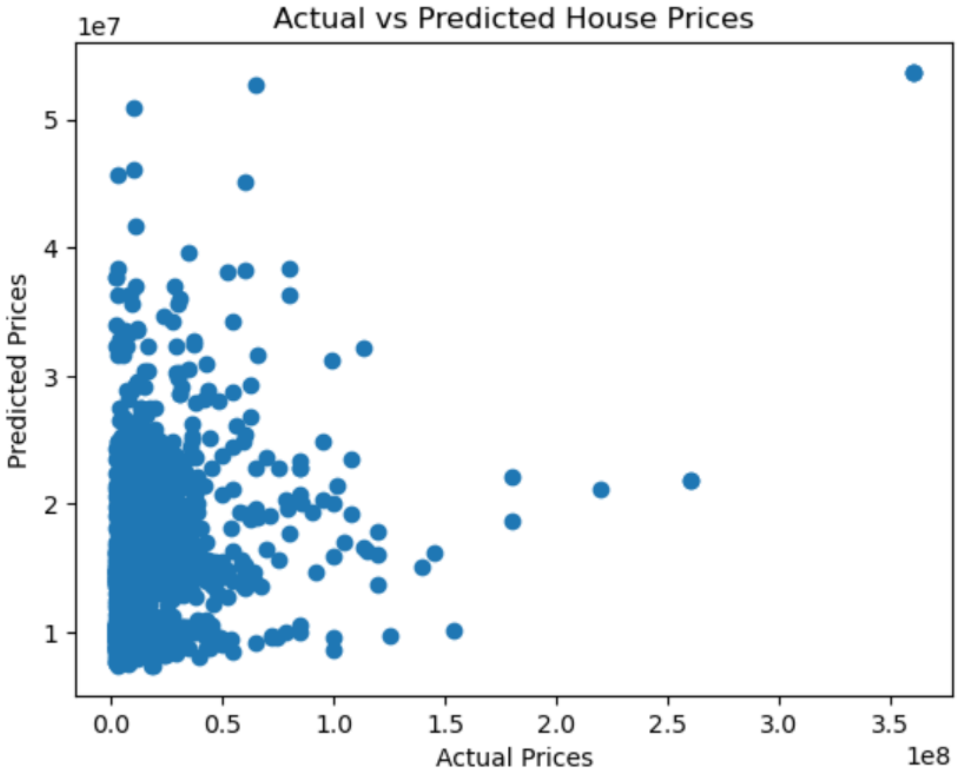
	Price	Area	Location	No. of Bedrooms	Resale	MaintenanceStaff	Gymnasium	SwimmingPool	LandscapedGardens	JoggingTrack	...
0	4850000	720	Kharghar	1	1	1	0	0	0	0	...
1	4500000	600	Kharghar	1	1	1	1	1	0	1	...
2	6700000	650	Kharghar	1	1	1	1	1	0	1	...
3	4500000	650	Kharghar	1	1	1	0	0	1	0	...
4	5000000	665	Kharghar	1	1	1	0	0	1	0	...

5 rows × 40 columns

```
X=Mhousing[['Area','No. of Bedrooms','Gymnasium']]
y=Mhousing['Price']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/3,random_state=0)
lm=LinearRegression()
```

```
lm.fit(X_train,y_train)
predictions=lm.predict(X_test)
plt.scatter(y_test,predictions)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted House Prices')
```

Output:

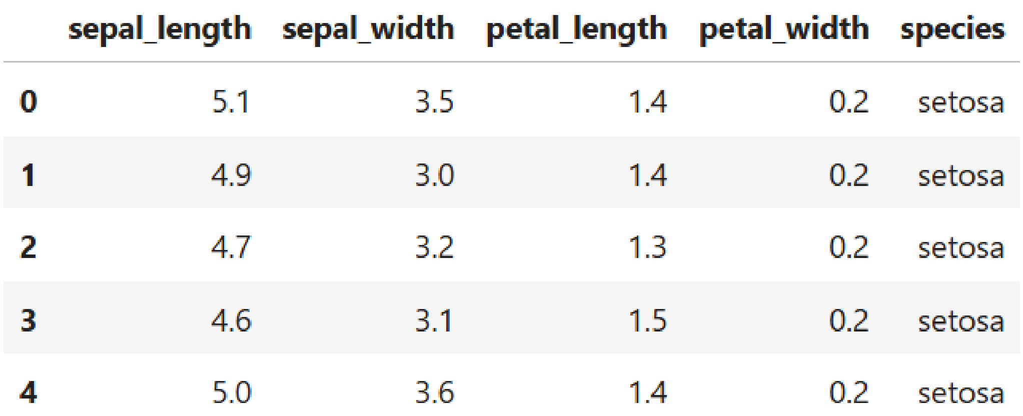


2/09/2025

//Implementation of Decision Tree

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import tree
from sklearn.metrics import
accuracy_score,classification_report
from sklearn.datasets import load_iris
from sklearn.tree import  DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
iris=load_iris()
iris=sns.load_dataset('iris')
iris.head()
```

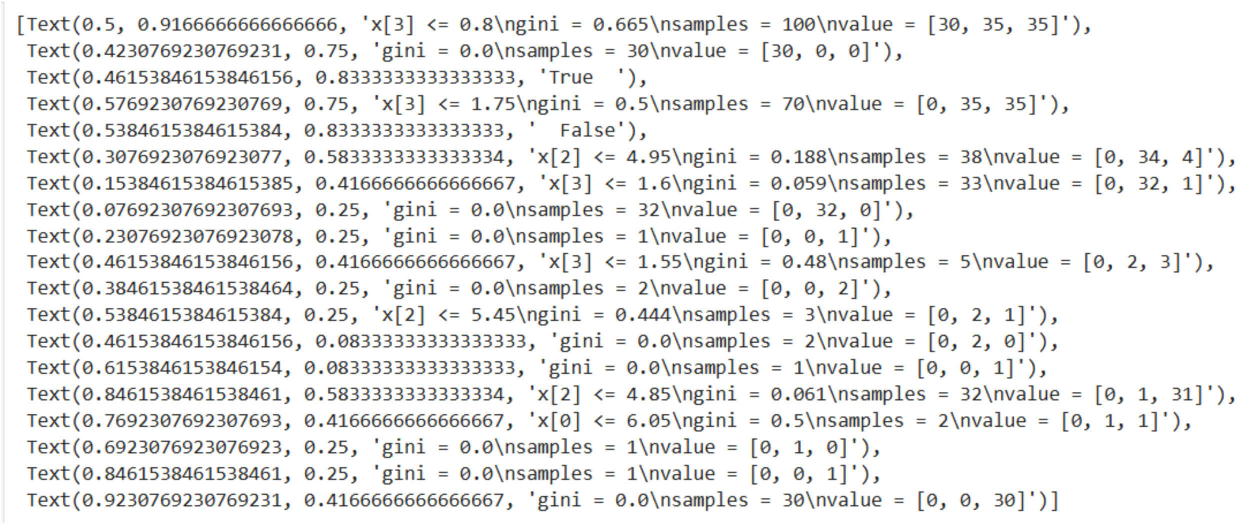
Output:



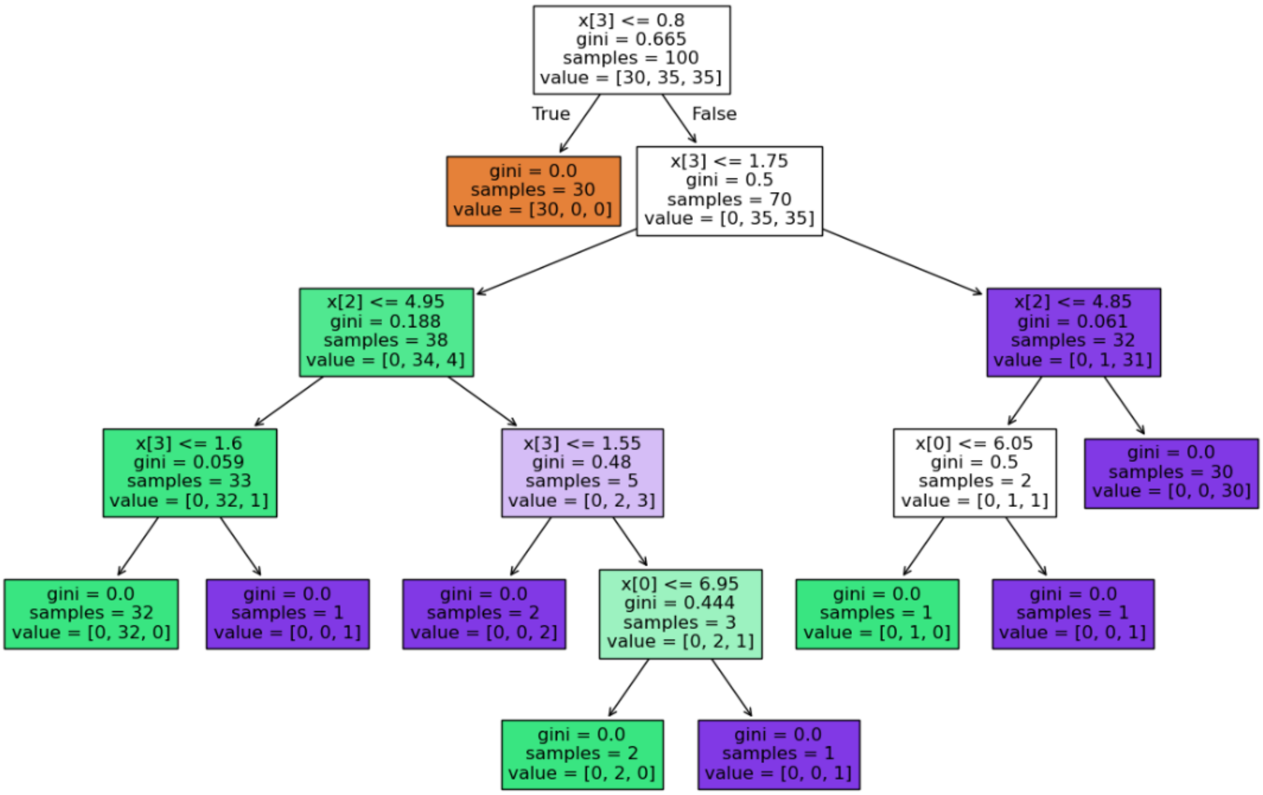
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
X=iris.iloc[:,:-1]
y=iris.species
X_train,X_test,y_train,y_test=train_test_split(X,y,test
_size=0.33,random_state=32)
treemodel=DecisionTreeClassifier()
treemodel.fit(X_train,y_train)
plt.figure(figsize=(15,10))
tree.plot_tree(treemodel,filled=True)
```

Output:

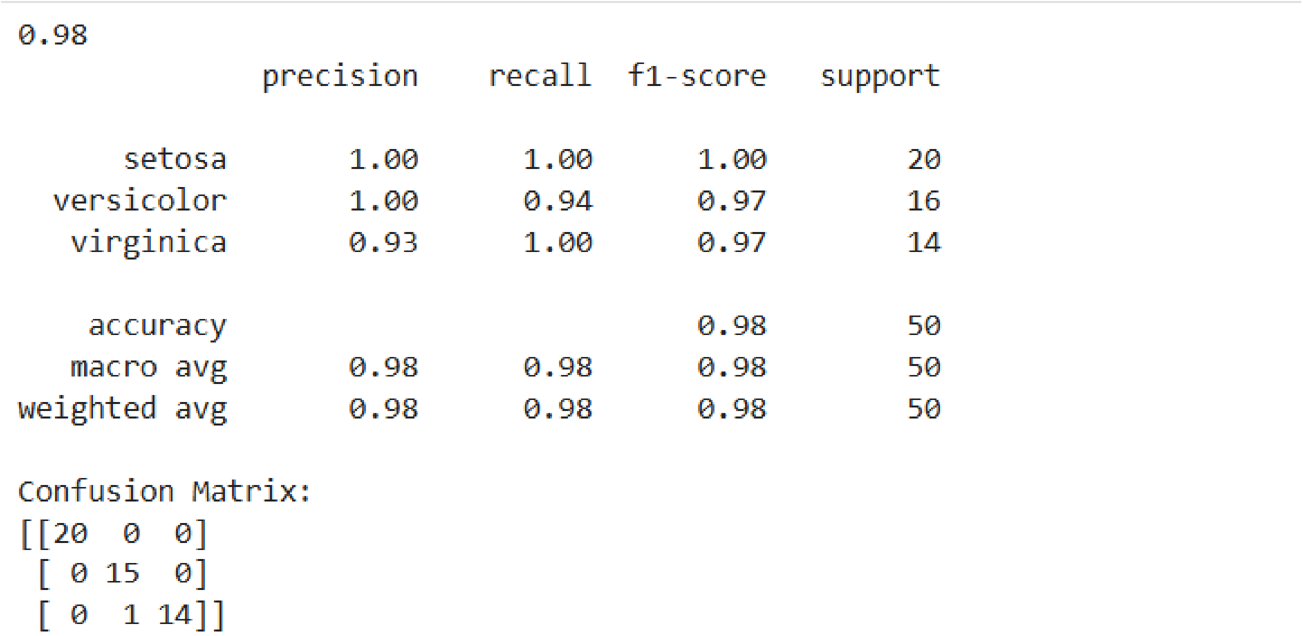


```
[Text(0.5, 0.9166666666666666, 'x[3] <= 0.8\ngini = 0.665\nsamples = 100\nvalue = [30, 35, 35]'),
 Text(0.4230769230769231, 0.75, 'gini = 0.0\nsamples = 30\nvalue = [30, 0, 0]'),
 Text(0.46153846153846156, 0.8333333333333333, 'True  '),
 Text(0.5769230769230769, 0.75, 'x[3] <= 1.75\ngini = 0.5\nsamples = 70\nvalue = [0, 35, 35]'),
 Text(0.5384615384615384, 0.8333333333333333, '  False'),
 Text(0.3076923076923077, 0.5833333333333334, 'x[2] <= 4.95\ngini = 0.188\nsamples = 38\nvalue = [0, 34, 4]'),
 Text(0.15384615384615385, 0.4166666666666667, 'x[3] <= 1.6\ngini = 0.059\nsamples = 33\nvalue = [0, 32, 1]'),
 Text(0.07692307692307693, 0.25, 'gini = 0.0\nsamples = 32\nvalue = [0, 32, 0]'),
 Text(0.23076923076923078, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.46153846153846156, 0.4166666666666667, 'x[3] <= 1.55\ngini = 0.48\nsamples = 5\nvalue = [0, 2, 3]'),
 Text(0.38461538461538464, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.5384615384615384, 0.25, 'x[2] <= 5.45\ngini = 0.444\nsamples = 3\nvalue = [0, 2, 1]'),
 Text(0.46153846153846156, 0.08333333333333333, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(0.6153846153846154, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.8461538461538461, 0.5833333333333334, 'x[2] <= 4.85\ngini = 0.061\nsamples = 32\nvalue = [0, 1, 31]'),
 Text(0.7692307692307693, 0.4166666666666667, 'x[0] <= 6.05\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(0.6923076923076923, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.8461538461538461, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.9230769230769231, 0.4166666666666667, 'gini = 0.0\nsamples = 30\nvalue = [0, 0, 30]')]
```



```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
ypred=treemodel.predict(X_test)
score=accuracy_score(ypred,y_test)
print(score)
print(classification_report(ypred,y_test))
print("Confusion Matrix:")
print(confusion_matrix(y_test,ypred))
```

Output:



```
0.98
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        20
  versicolor       1.00      0.94      0.97        16
   virginica       0.93      1.00      0.97        14

    accuracy                           0.98        50
   macro avg       0.98      0.98      0.98        50
weighted avg       0.98      0.98      0.98        50

Confusion Matrix:
[[20  0  0]
 [ 0 15  0]
 [ 0  1 14]]
```