PROGRAM 1 : (FIND S)

[10]:
```
import csv
with open('MLlab.csv','r')as cfe:
    reader=csv.reader(cfe,delimiter=',')
    train=[]
    for row in reader:
        train.append(row)
print(train)
```

```
[['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Sa
'High', 'Strong', 'Warm', 'Change', 'No'], ['Sunny
```

[16]:
```
hypo=['0']*6
row=4
col=6
for i in range(row):
    if(train[i][6]=="Yes"):
        for j in range (col):
            if(hypo[j]=='0'):
                hypo[j]=train[i][j]
            else:
                if(hypo[j]!=train[i][j]):
                    hypo[j]='?'
```

[17]:
```
print(hypo)
```

```
['Sunny', 'Warm', '?', 'Strong', '?', '?']
```

[ ]:

PROGRAM 2 : (LINEAR)

```
[1]:  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: df=pd.read_csv('kc_house_data.csv')
     df
```

```
[2]:
                id             date     price  l

    0   7129300520  20141013T000000  221900.0
    1   6414100192  20141209T000000  538000.0
    2   5631500400  20150225T000000  180000.0
    3   2487200875  20141209T000000  604000.0
    4   1954400510  20150218T000000  510000.0
  ...          ...              ...       ...
21608    263000018  20140521T000000  360000.0
21609   6600060120  20150223T000000  400000.0
21610   1523300141  20140623T000000  402101.0
```

```
[3]: df=df.drop(['id','date'],axis=1)
     print(df)
```

```
          price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  \
0      221900.0         3       1.00         1180      5650     1.0
1      538000.0         3       2.25         2570      7242     2.0
2      180000.0         2       1.00          770     10000     1.0
3      604000.0         4       3.00         1960      5000     1.0
4      510000.0         3       2.00         1680      8080     1.0
...         ...       ...        ...          ...       ...     ...
21608  360000.0         3       2.50         1530      1131     3.0
21609  400000.0         4       2.50         2310      5813     2.0
21610  402101.0         2       0.75         1020      1350     2.0
21611  400000.0         3       2.50         1600      2388     2.0
21612  325000.0         2       0.75         1020      1076     2.0

       waterfront  view  condition  grade  sqft_above  sqft_basement  \
0               0     0          3      7        1180              0
1               0     0          3      7        2170            400
2               0     0          3      6         770              0
3               0     0          5      7        1050            910
4               0     0          3      8        1680              0
...           ...   ...        ...    ...         ...            ...
21608           0     0          3      8        1530              0
21609           0     0          3      8        2310              0
21610           0     0          3      7        1020              0
21611           0     0          3      8        1600              0
21612           0     0          3      7        1020              0
```

```
[4]: y=df['price']
     x=df.drop('price',axis=1)
     x
```

[4]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	3	1.00	1180	5650	1
1	3	2.25	2570	7242	2
2	2	1.00	770	10000	1
3	4	3.00	1960	5000	1
4	3	2.00	1680	8080	1
...	...	...	...	...	
21608	3	2.50	1530	1131	3
21609	4	2.50	2310	5813	2
21610	2	0.75	1020	1350	2
21611	3	2.50	1600	2388	2
21612	2	0.75	1020	1076	2

21613 rows × 18 columns

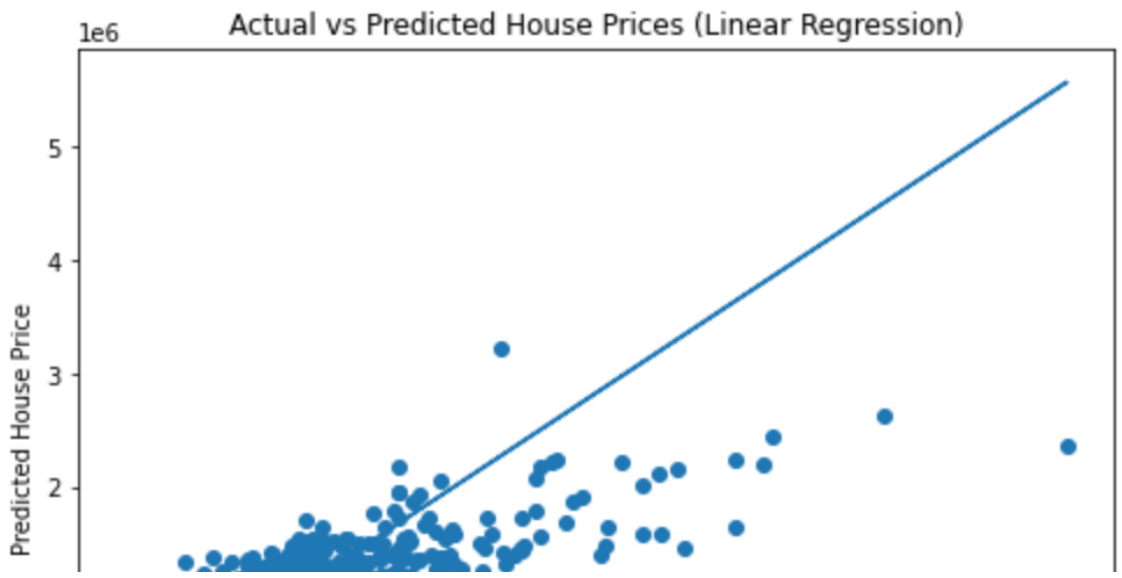
```
[5]:  from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

```
[6]: from sklearn.linear_model import LinearRegression
     reg=LinearRegression().fit(x_train,y_train)
     reg.score(x_test,y_test)
```

```
[6]: 0.7156152938565365
```

```
[7]:  #Predict values for test data
     y_pred = reg.predict(x_test)

     # Plot Actual vs Predicted
     plt.figure(figsize=(8,6))
     plt.scatter(y_test, y_pred)     # actual vs predicted points
     plt.plot(y_test, y_test)        # 45-degree line (perfect predictions)
     plt.xlabel("Actual House Price")
     plt.ylabel("Predicted House Price")
     plt.title("Actual vs Predicted House Prices (Linear Regression)")
     plt.show()
```



PROGRAM 3 : (LOGISTIC)

```
[1]: import pandas as pd
     from sklearn.datasets import load_diabetes
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from sklearn.preprocessing import StandardScaler
     import matplotlib.pyplot as plt
     from sklearn.metrics import accuracy_score,confusion_matrix
     import numpy as np
     import seaborn as sns
```

```
[2]: diabetes=load_diabetes()
     x,y=diabetes.data,diabetes.target
     y_binary=(y>np.median(y)).astype(int)
     y_binary
```

```
[2]: array([1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
            0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
            1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
            1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
            0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
            0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1,
            0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0,
            1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
            1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
            1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1,
            0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1,
            0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
            1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
            0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
            0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
            1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
            0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
            0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
            0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0,
            0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
[3]: x_train,x_test,y_train,y_test=train_test_split(x,y_binary,test_size=0.2,random_state=42)
     #print(len(x_test))
     #print(len(x_train))
```

```
[4]: scaler=StandardScaler()
     x_train=scaler.fit_transform(x_train)
     x_test=scaler.fit_transform(x_test)
```

```
[5]: model=LogisticRegression()
     model.fit(x_train,y_train)
     x_train
```

```
[5]: array([[ 1.49836523,  1.06136988,  0.21990201, ...,  0.71103773,
              0.54748197, -0.06144896],
            [-0.22885822,  1.06136988, -0.41936607, ...,  1.4842858 ,
             -0.01975653,  0.36723647],
            [ 0.08518241, -0.94217861,  1.01898711, ..., -0.06221033,
              0.3312366 , -0.31866022],
            ...,
            [ 0.63475351, -0.94217861, -0.46502808, ..., -0.83545839,
             -0.25375196, -0.06144896],
            [-0.30736838, -0.94217861, -0.53352109, ..., -0.06221033,
             -0.83072436, -0.83308273],
            [-2.03459183, -0.94217861,  0.56236706, ..., -0.83545839,
             -0.13312789, -0.06144896]], shape=(353, 10))
```

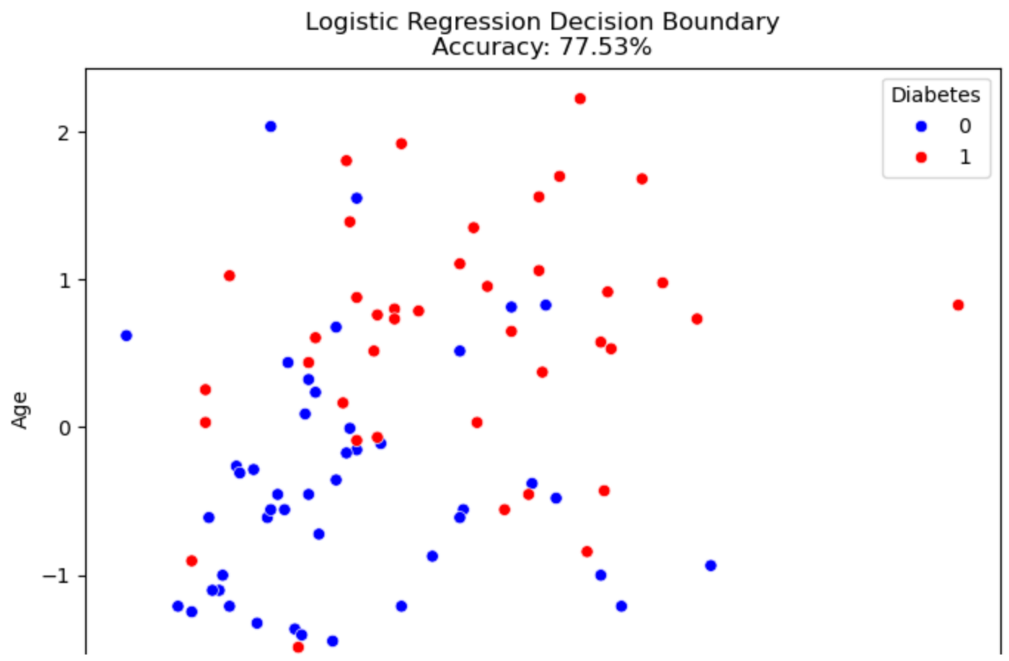
```
[6]: y_pred=model.predict(x_test)
     accuracy=accuracy_score(y_test,y_pred)
     accuracy
```

```
[6]: 0.7752808988764045
```

```
[7]: print(confusion_matrix(y_test,y_pred))
```

```
[[35 14]
 [ 6 34]]
```

```
[8]: # Visualize the decision boundary with accuracy information
     plt.figure(figsize=(8, 6))
     sns.scatterplot(x=x_test[:, 2], y=x_test[:, 8], hue=y_test, palette={
                     0: 'blue', 1: 'red'}, marker='o')
     plt.xlabel("BMI")
     plt.ylabel("Age")
     plt.title("Logistic Regression Decision Boundary\nAccuracy: {:.2f}%".format(
         accuracy * 100))
     plt.legend(title="Diabetes", loc="upper right")
     plt.show()
```



PROGRAM 4 : (DECISION TREE)

```
[6]: import numpy as np
     import pandas as pd
     from sklearn.preprocessing import LabelEncoder
     from sklearn.model_selection import  train_test_split
     from sklearn.tree import DecisionTreeClassifier,plot_tree
     from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
     import matplotlib.pyplot as plt
```

```
[7]: df=pd.read_csv("adult.csv")
```

```
[8]: df.replace('?',np.nan,inplace=True)
     df.dropna(inplace=True)
     df.drop(['fnlwgt','educational-num','marital-status','relationship','race'],axis=1,inplace=True)
```

```
[9]: X=df.drop('income',axis=1)
     y=LabelEncoder().fit_transform(df['income'])
```

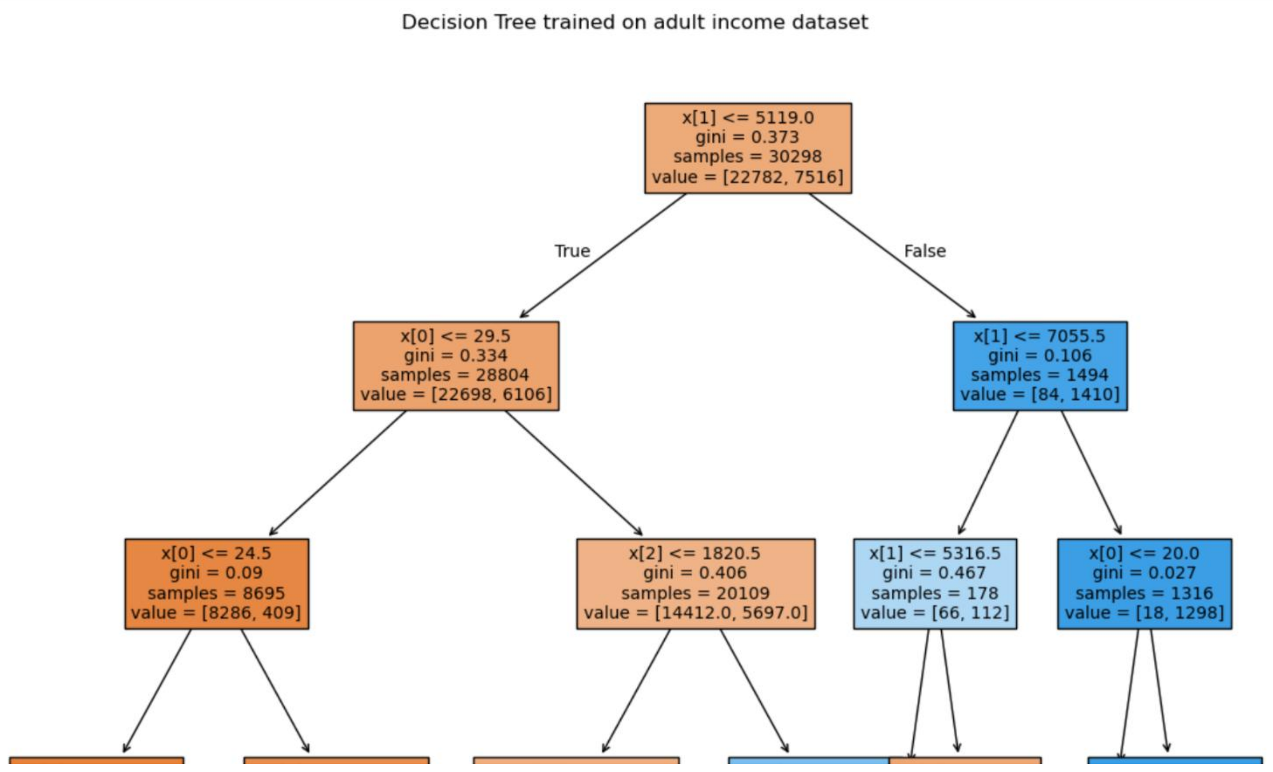
```
[10]: X=pd.get_dummies(X)
      X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=42)
      clf=DecisionTreeClassifier(max_depth=5)
      clf.fit(X_train,y_train)
      y_pred=clf.predict(X_test)
      print("Model Accuracy:",accuracy_score(y_test,y_pred))
```

```
Model Accuracy: 0.8234387563655856
```

```
[11]: cm=confusion_matrix(y_test,y_pred)
      print("\n Confusion Matrix:\n",cm)
      print("\n Classification Report:\n",classification_report(y_test,y_pred))
```

```
 Confusion Matrix:
 [[10736   496]
 [ 2139  1553]]
```

```
[12]: plt.figure(figsize=(14,14))
      plot_tree(clf,fontsize=10,filled=True)
      plt.title("Decision Tree trained on adult income dataset")
      plt.show()
```



PROGRAM 5 : (SVM)

```
[4]: import numpy as np
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.svm import SVC
     from sklearn.metrics import accuracy_score
     from sklearn.preprocessing import LabelEncoder
```

```
[5]: df=pd.read_csv("smaller_adult.csv")
```

```
[6]: df.replace("?",np.nan,inplace=True)
     df.dropna(inplace=True)
     df.drop_duplicates(inplace=True)
     X=df[['age','workclass','educational-num','occupation','gender','hours-per-week']]
     y=LabelEncoder().fit_transform(df['income'])
```

```
[7]: X=pd.get_dummies(X)
```

```
[8]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=42)
```

```
[9]: clf=SVC(kernel='linear',gamma='auto')
     clf.fit(X_train,y_train)
```

```
[9]: SVC

SVC(gamma='auto', kernel='linear')
```

```
[10]: y_pred=clf.predict(X_test)
      print("Accuracy:",accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.7672162948593598
```