krailabsp

October 16, 2023

[ ]:

```

```

##Q1 A Correlation Matrix 1st One

[ ]:

```python
import numpy as np
import pandas as pd

#1A
df={
    "array1":[15,17,22,16,45,52,62,77,11,98],
    "array2":[11,15,20,31,48,51,71,89,91,100],
    "array3":[104,100,89,81,76,66,69,43,17,11]
}

data = pd.DataFrame(df)
data.corr()
#print(data.corr);
```

[ ]:

```
          array1    array2    array3
array1  1.000000  0.696285 -0.567446
array2  0.696285  1.000000 -0.949262
array3 -0.567446 -0.949262  1.000000
```

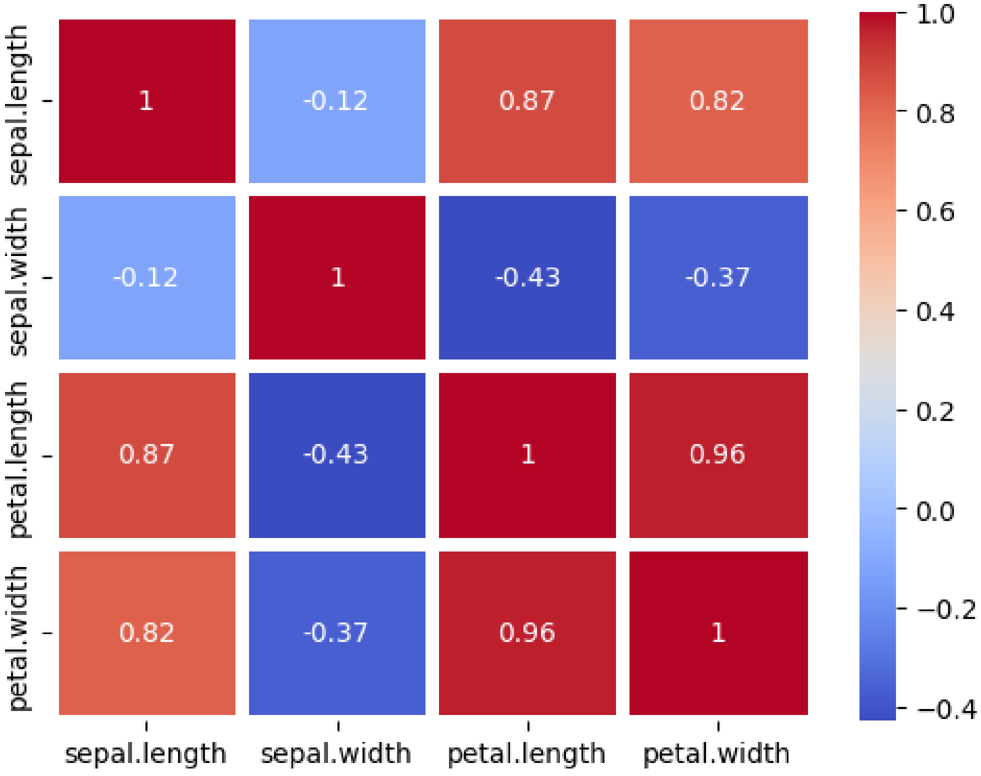
[ ]:

```python
import numpy as np
import pandas as pd

# Q.1 B) Plot the Correlation
data = pd.read_csv("/content/iris.csv")
print(data)
x = data.drop('variety',axis=1)
corr_mat = x.corr()
import seaborn as sns
sns.heatmap(corr_mat,annot=True,cmap='coolwarm',linewidths=5)
```

```
     sepal.length  sepal.width  petal.length  petal.width    variety
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
..             …          …           …          …        …
145            6.7          3.0           5.2          2.3  Virginica
146            6.3          2.5           5.0          1.9  Virginica
147            6.5          3.0           5.2          2.0  Virginica
148            6.2          3.4           5.4          2.3  Virginica
149            5.9          3.0           5.1          1.8  Virginica

[150 rows x 5 columns]
```

[ ]:
```
<Axes: >
```

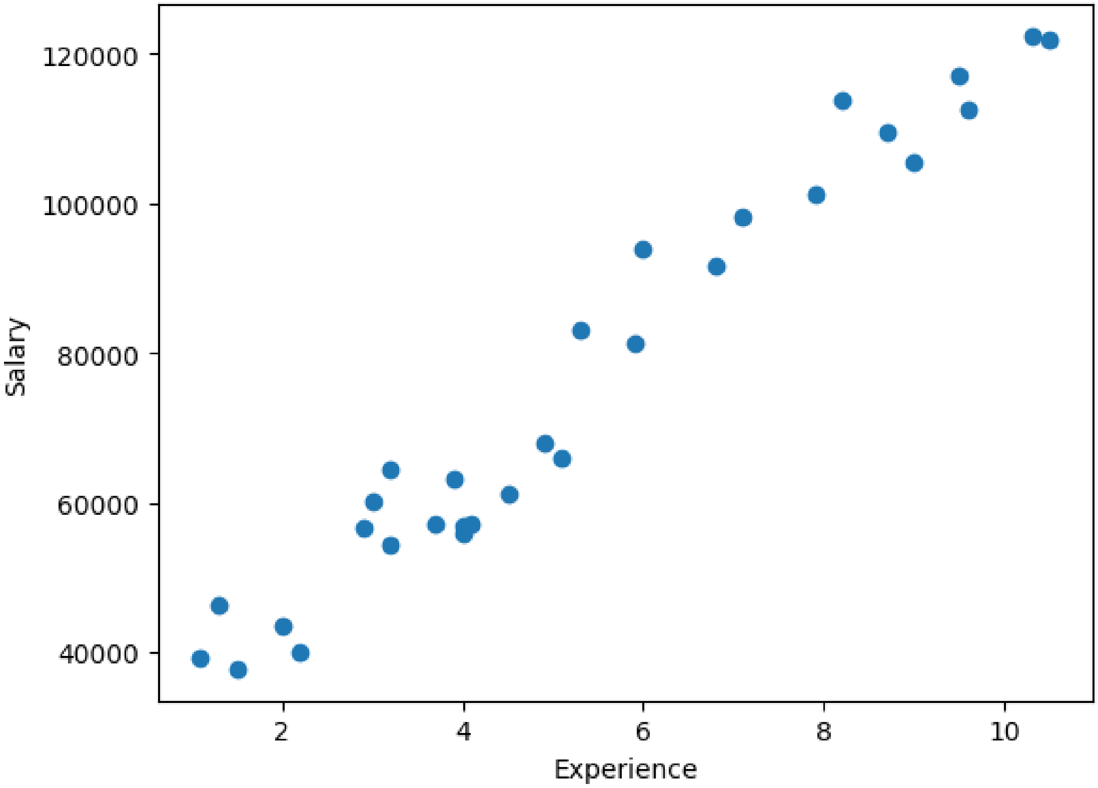


##Q2 Linear Regression

[4]:
```
import pandas as pd
#import os os.getcwd()
df=pd.read_csv('/content/Salary_Data.csv')
df.shape #gives count of no of rows,columns
df.columns #gives names of columns
x=df['YearsExperience'].values
y=df['Salary'].values
```

```
df.corr()

import matplotlib.pyplot as plt
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.scatter(x,y)

from sklearn.linear_model import LinearRegression
obj=LinearRegression() #it gives object of class LR
x=x.reshape(-1,1)
x
obj.fit(x,y) #it provides inp and op columns to model for fitting of data
obj.predict(x) #it predicts
y_pred=obj.predict(x)
print(y_pred)
```

```
[ 36187.15875227  38077.15121656  39967.14368085  44692.12484158
  46582.11730587  53197.09093089  54142.08716303  56032.07962732
  56032.07962732  60757.06078805  62647.05325234  63592.04948449
  63592.04948449  64537.04571663  68317.03064522  72097.0155738
  73987.00803809  75877.00050238  81546.97789525  82491.9741274
  90051.94398456  92886.932681   100446.90253816 103281.8912346
 108006.87239533 110841.86109176 115566.84225249 116511.83848464
 123126.81210966 125016.80457395]
```



##Q3 Logestic Regression

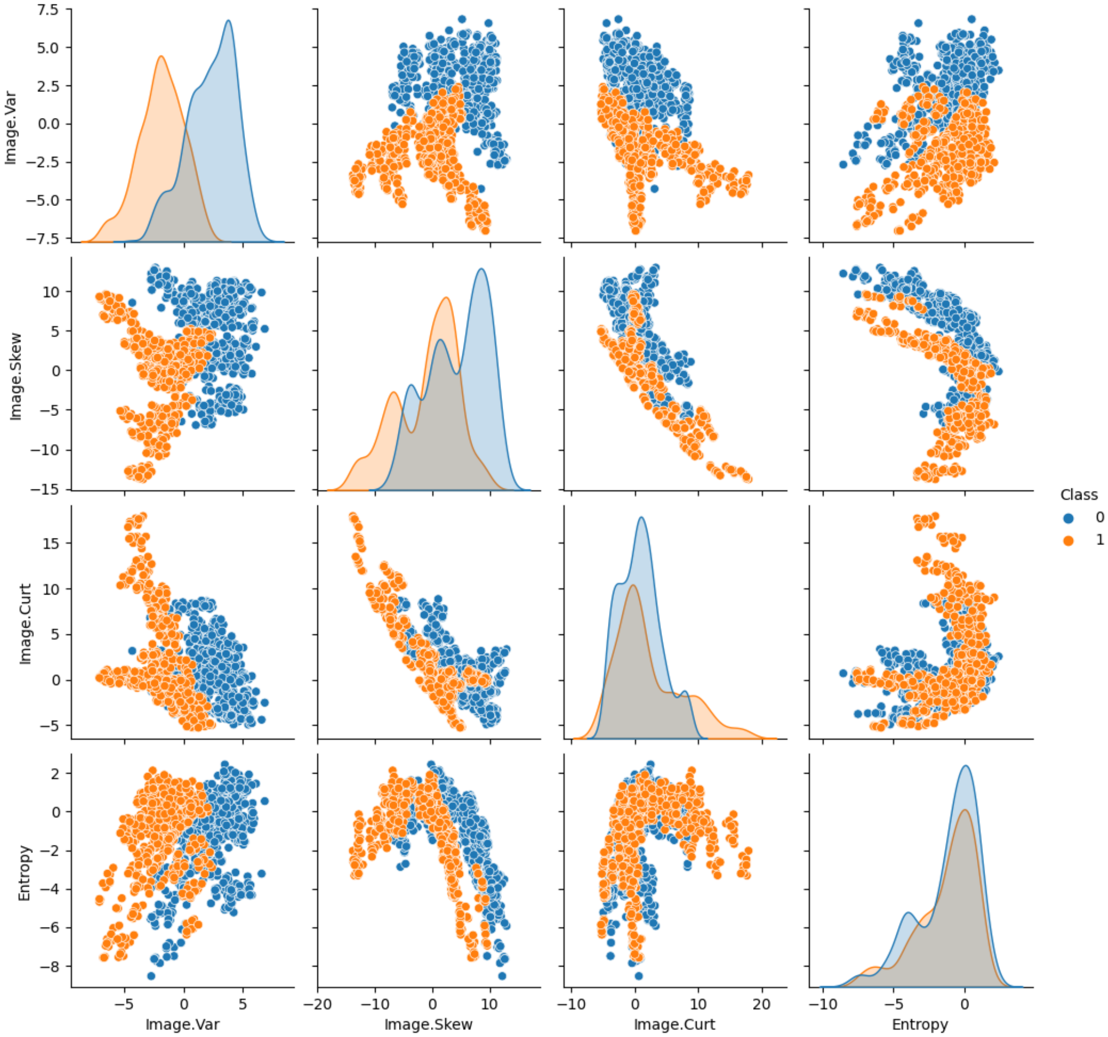
[11]:

```
import pandas as pd
df=pd.read_csv('/content/banknotes.csv')

import seaborn as sns
sns.pairplot(df,hue='Class')

x=df.drop('Class',axis=1)
y=df['Class']
x.shape

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.
 ↪25)
x_train.head()
x_train.shape

from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()

classifier.fit(x_train,y_train)
x_test.shape

y_pred=classifier.predict(x_test)
set(y)
y.value_counts()


result=pd.DataFrame({
'Actual':y_test,
'Predicted':y_pred
})
result
```

[11]:

```
      Actual  Predicted
1023       1          1
642        0          0
1196       1          1
31         0          0
253        0          0
…       …        …
866        1          1
361        0          0
703        0          0
```

```
328          0            0
530          0            0

[343 rows x 2 columns]
```



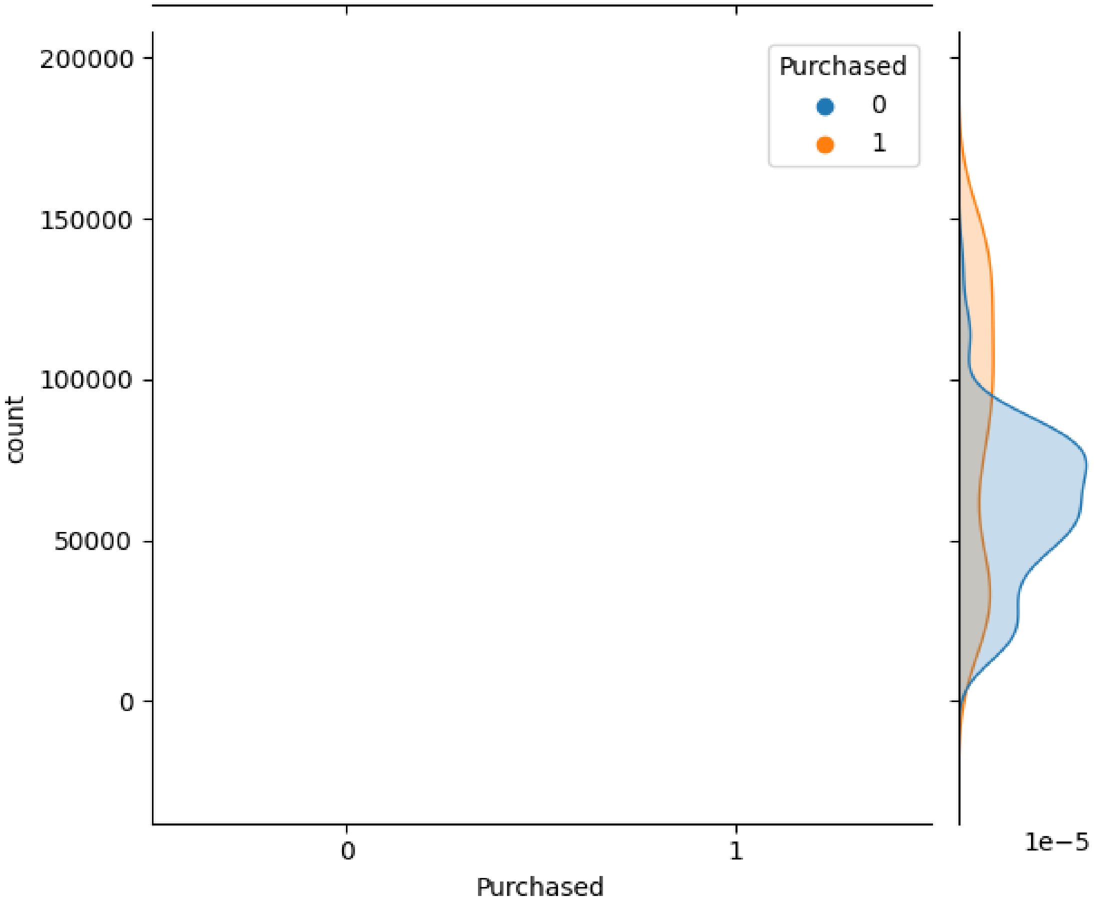
##Q4 Random Forest

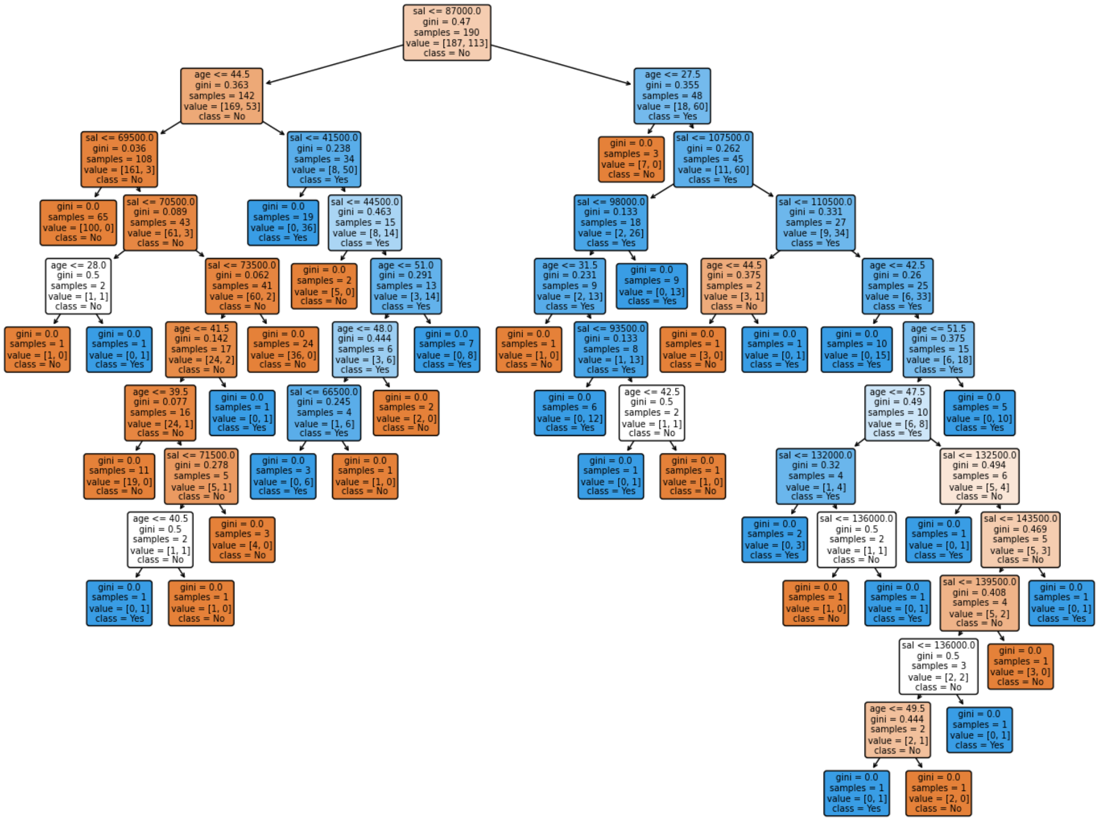
[13]:
```
import pandas as pd
#dataimport
df=pd.read_csv('/content/Social_Network_Ads.csv')
df.shape # count = no of rows and colms
#import
x=df[['Age','EstimatedSalary']]
y=df['Purchased']
```

```
import seaborn as sns
sns.jointplot(x='Age',y='EstimatedSalary',hue='Purchased',data=df)
sns.countplot(x=y)
y.value_counts()

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0,test_size=0.
 ↪25)
x_train.shape
x_test.shape


from sklearn.ensemble import RandomForestClassifier

obj=RandomForestClassifier(random_state=0,n_estimators=10)

#Train the algorithm with data
obj.fit(x_train,y_train)

#Predications
y_pred=obj.predict(x_test)

#Combine the data
result=pd.DataFrame({
    'Actual':y_test,
    'Predicted':y_pred
})

result



new1=[[34,123000]]
new2=[[25,48900]]
obj.predict(new1)
obj.predict(new2)


from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

obj.estimators_[0]
plt.figure(figsize=(16,12))
plot_tree(obj.estimators_[8],fontsize=7,feature_names=['age','sal'],
         class_names=['No','Yes'],filled=True,rounded=True);
obj.feature_importances_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```

```
[13]: array([0.48777687, 0.51222313])
```




sal <= 87000.0 gini = 0.47 samples = 190 value = [187, 113] class = No
age <= 44.5 gini = 0.363 samples = 142 value = [169, 53] class = No
age <= 27.5 gini = 0.355 samples = 48 value = [18, 60] class = Yes
sal <= 69500.0 gini = 0.036 samples = 108 value = [161, 3] class = No
sal <= 41500.0 gini = 0.238 samples = 34 value = [8, 50] class = Yes
gini = 0.0 samples = 3 value = [7, 0] class = No
sal <= 107500.0 gini = 0.262 samples = 45 value = [11, 60] class = Yes
gini = 0.0 samples = 65 value = [100, 0] class = No
sal <= 70500.0 gini = 0.089 samples = 43 value = [61, 3] class = No
gini = 0.0 samples = 19 value = [0, 36] class = Yes
sal <= 44500.0 gini = 0.463 samples = 15 value = [8, 14] class = Yes
sal <= 98000.0 gini = 0.133 samples = 18 value = [2, 26] class = Yes
sal <= 110500.0 gini = 0.331 samples = 27 value = [9, 34] class = Yes
age <= 28.0 gini = 0.5 samples = 2 value = [1, 1] class = No
sal <= 73500.0 gini = 0.062 samples = 41 value = [60, 2] class = No
gini = 0.0 samples = 2 value = [5, 0] class = No
age <= 51.0 gini = 0.291 samples = 13 value = [3, 14] class = Yes
age <= 31.5 gini = 0.231 samples = 9 value = [2, 13] class = Yes
gini = 0.0 samples = 9 value = [0, 13] class = Yes
age <= 44.5 gini = 0.375 samples = 2 value = [3, 1] class = No
age <= 42.5 gini = 0.26 samples = 25 value = [6, 33] class = Yes
gini = 0.0 samples = 1 value = [1, 0] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
age <= 41.5 gini = 0.142 samples = 17 value = [24, 2] class = No
gini = 0.0 samples = 24 value = [36, 0] class = No
age <= 48.0 gini = 0.444 samples = 6 value = [3, 6] class = Yes
gini = 0.0 samples = 7 value = [0, 8] class = Yes
gini = 0.0 samples = 1 value = [1, 0] class = No
sal <= 93500.0 gini = 0.133 samples = 8 value = [1, 13] class = Yes
gini = 0.0 samples = 1 value = [3, 0] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
gini = 0.0 samples = 10 value = [0, 15] class = Yes
age <= 51.5 gini = 0.375 samples = 15 value = [6, 18] class = Yes
age <= 39.5 gini = 0.077 samples = 16 value = [24, 1] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
sal <= 66500.0 gini = 0.245 samples = 4 value = [1, 6] class = Yes
gini = 0.0 samples = 2 value = [2, 0] class = No
gini = 0.0 samples = 6 value = [0, 12] class = Yes
age <= 42.5 gini = 0.5 samples = 2 value = [1, 1] class = No
age <= 47.5 gini = 0.49 samples = 10 value = [6, 8] class = Yes
gini = 0.0 samples = 5 value = [0, 10] class = Yes
gini = 0.0 samples = 11 value = [19, 0] class = No
sal <= 71500.0 gini = 0.278 samples = 5 value = [5, 1] class = No
gini = 0.0 samples = 3 value = [0, 6] class = Yes
gini = 0.0 samples = 1 value = [1, 0] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
gini = 0.0 samples = 1 value = [1, 0] class = No
sal <= 132000.0 gini = 0.32 samples = 4 value = [1, 4] class = Yes
sal <= 132500.0 gini = 0.494 samples = 6 value = [5, 4] class = No
age <= 40.5 gini = 0.5 samples = 2 value = [1, 1] class = No
gini = 0.0 samples = 3 value = [4, 0] class = No
gini = 0.0 samples = 2 value = [0, 3] class = Yes
sal <= 136000.0 gini = 0.5 samples = 2 value = [1, 1] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
sal <= 143500.0 gini = 0.469 samples = 5 value = [5, 3] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
gini = 0.0 samples = 1 value = [1, 0] class = No
gini = 0.0 samples = 1 value = [1, 0] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
sal <= 139500.0 gini = 0.408 samples = 4 value = [5, 2] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
sal <= 136000.0 gini = 0.5 samples = 3 value = [2, 2] class = No
gini = 0.0 samples = 1 value = [3, 0] class = No
age <= 49.5 gini = 0.444 samples = 2 value = [2, 1] class = No
gini = 0.0 samples = 1 value = [0, 1] class = Yes
gini = 0.0 samples = 1 value = [0, 1] class = Yes
gini = 0.0 samples = 1 value = [2, 0] class = No

##Q5 KNN

[14]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score


data = pd.read_csv("/content/banknotes.csv")
X=data.drop('Class',axis=1)
y=data['Class']


# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Initialize the KNN classifier
```

```python
k = 3  # You can change the value of k
knn = KNeighborsClassifier(n_neighbors=k)

# Train the KNN classifier on the training data
knn.fit(X_train, y_train)

# Make predictions on the test data
y_pred = knn.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 100.00%
```