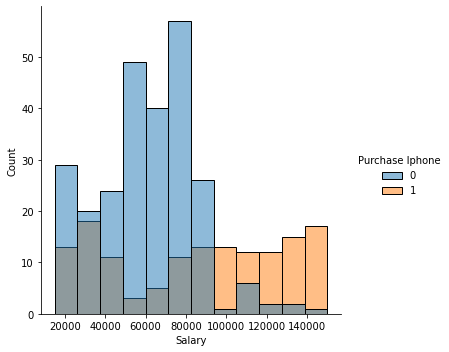
# NAME: HARSHAL.G.PANCHAL

ROLL.NO:43

|  |  |  |
| --- | --- | --- |
| import warnings  warnings.filterwarnings("ignore")  import numpy as np import pandas as pd  import matplotlib.pyplot as plt import seaborn as sns  from sklearn.metrics import accuracy\_score from sklearn.model\_selection import train\_test\_split from scipy.stats import mode  data = pd.read\_csv('iphone.csv') data.head()   |  | | --- | | Gender Age Salary Purchase Iphone   1. Male 19 19000 0 2. Male 35 20000 0 3. Female 26 43000 0 4. Female 27 57000 0 5. Male 19 76000 0 |   data.shape   |  | | --- | | (400, 4) |   sns.scatterplot(x = data['Salary'], y = data['Age'], hue = data['Purchase Iphone'])  <AxesSubplot:xlabel='Salary', ylabel='Age'> |

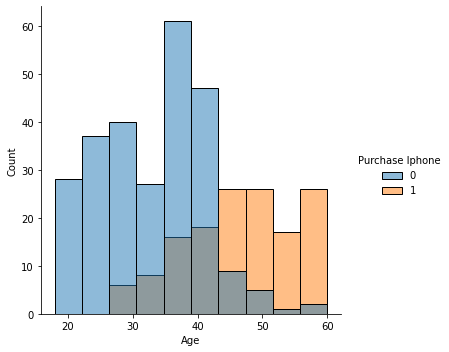


|  |  |
| --- | --- |
| data = data.drop('Gender', axis = 1) data.head()   |  | | --- | | Age Salary Purchase Iphone   1. 19 19000 0 2. 35 20000 0 3. 26 43000 0 4. 27 57000 0 5. 19 76000 0 |   sns.displot(data, x = 'Salary', hue = 'Purchase Iphone')  <seaborn.axisgrid.FacetGrid at 0x20e08950160> |



sns.displot( data, x = 'Age', hue = 'Purchase Iphone')

<seaborn.axisgrid.FacetGrid at 0x20e1dedeca0>

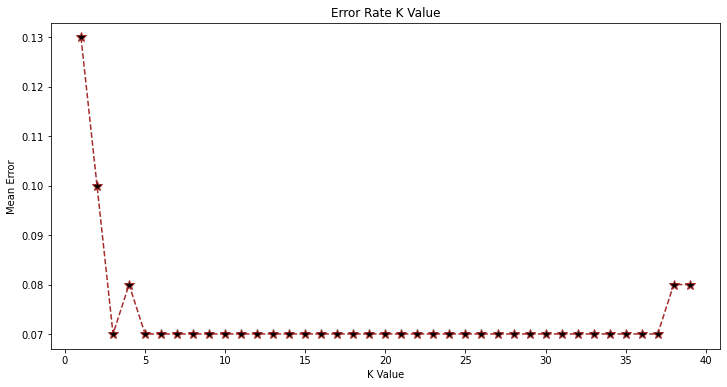


|  |  |  |
| --- | --- | --- |
| import numpy as np  def euclidean\_distance(pt1,pt2): distance = np.sqrt(np.sum((pt1-pt2)\*\*2)) return distance  a = np.array([3,4]) b = np.array([5,9])  print(euclidean\_distance(a,b))   |  | | --- | | 5.385164807134504 |   x = data.drop('Purchase Iphone', axis = 1) y = data['Purchase Iphone'] print(x)   |  | | --- | | Age Salary   1. 19 19000 2. 35 20000 3. 26 43000 4. 27 57000 | |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 4 19 76000 .. ... ...   1. 46 41000 2. 51 23000 3. 50 20000 4. 36 33000 5. 49 36000 | | [400 rows x 2 columns] | |   print(y)   |  |  | | --- | --- | | 1. 0 2. 0 3. 0 4. 0 5. 0 .. 6. 1 7. 1 8. 1 9. 0 10. 1 |  | | Name: Purchase Iphone, Length: 400, dtype: int64 | |   from sklearn.model\_selection import train\_test\_split  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size =  0.25, random\_state = 0) print(x\_train.shape)   |  | | --- | | (300, 2) |   print(y\_train.shape)   |  | | --- | | (300,) |   print(x\_test.shape)   |  | | --- | | (100, 2) |   print(y\_test.shape)   |  | | --- | | (100,) |   from sklearn.neighbors import KNeighborsClassifier unknown\_value = KNeighborsClassifier(n\_neighbors = 7) unknown\_value.fit(x\_train, y\_train) KNeighborsClassifier(n\_neighbors=7) |
| y\_pred = unknown\_value.predict(x\_test) y\_pred   |  |  |  | | --- | --- | --- | | array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, | | | | 1, |  | | | 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, | | | | 0, |  | | | 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, | | | | 0, |  | | | 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, | | | | 1, |  | | | 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1], dtype=int64) | |  |   accuracy = accuracy\_score(y\_test , y\_pred) print("Accuracy:", accuracy)  Accuracy: 0.81 |

# Feature Scaling

|  |
| --- |
| from sklearn.preprocessing import StandardScaler sc = StandardScaler()  x\_train = sc.fit\_transform(x\_train) x\_test = sc.transform(x\_test)  error = []  from sklearn.neighbors import KNeighborsClassifier import matplotlib.pyplot as plt  *#calculating error for K values between 1 and 40* for i in range(1,40): model = KNeighborsClassifier(n\_neighbors = i) model.fit(x\_train , y\_train) pred\_i = model.predict(x\_test)  error.append(np.mean(pred\_i != y\_test))    plt.figure(figsize=(12,6))  plt.plot(range(1,40), error, color='brown', linestyle='dashed', marker='\*',markerfacecolor='black',markersize = 10) plt.title('Error Rate K Value') plt.xlabel('K Value') plt.ylabel('Mean Error')  Text(0, 0.5, 'Mean Error') |



|  |  |
| --- | --- |
| data.head()   |  | | --- | | Age Salary Purchase Iphone   1. 19 19000 0 2. 35 20000 0 3. 26 43000 0 4. 27 57000 0 5. 19 76000 0 |   Age = int(input("Enter New person Age : ")) Salary = int(input("Enter New person Salary :")) newperson = [[Age,Salary]]  result = model.predict(sc.transform(newperson)) print(result)  if result == 1: print("person might purchase Iphone") else: print("person might not purchase Iphone")  from sklearn.metrics import confusion\_matrix, accuracy\_score cm = confusion\_matrix(y\_test, y\_pred)  print("Confusion Matrix:") print(cm)  print("Accuracy of the model : {0}% " . format(accuracy\_score(y\_test, y\_pred)\*100)) |

# By Using Minkowski Distance Methods

|  |
| --- |
| from sklearn.preprocessing import StandardScaler , LabelEncoder from sklearn.metrics import accuracy\_score  df=pd.read\_csv('iphone.csv') df.head()  le = LabelEncoder()  df['Gender'] = le.fit\_transform(df['Gender']) df.head() x y  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size =  0.2, random\_state = 42)  *#feature scaling* scaler =StandardScaler()  x\_train\_scaled = scaler.fit\_transform(x\_train) x\_test\_scaled = scaler.transform(x\_test)  *#Build KNN Model*  knn = KNeighborsClassifier(n\_neighbors=5, p=2) *#p=2 for euclidean distance*  knn.fit(x\_train\_scaled , y\_train) y\_pred = knn.predict(x\_test\_scaled)  accuracy = accuracy\_score(y\_test, y\_pred) print(f'Accuracy: {accuracy:.2f}')  Age = int(input("Enter New person Age : ")) Salary = int(input("Enter New person Salary :")) newperson = [[Age,Salary]]  result = model.predict(sc.transform(newperson)) print(result)  if result == 1: print("person might purchase Iphone") else: print("person might not purchase Iphone") |