TEAM-9

DIVYA DR

(1SV21CS030)

```
 import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
df = pd.read_csv('/content/drive/MyDrive/archive
(8)/iphone_purchase_records.csv')
```

```
df.head()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 400,\n \"fields\": [\n   {\n   \"column\": \"Gender\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Female\",\n   \"Male\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 10,\n   \"min\": 18,\n   \"max\": 60,\n   \"num_unique_values\": 43,\n   \"samples\": [\n   50,\n39\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Salary\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 34096,\n   \"min\": 15000,\n   \"max\": 150000,\n   \"num_unique_values\": 117,\n   \"samples\": [\n   117000,\n   76000\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Purchase Iphone\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n\"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 130.27423677374767,\n   \"min\": 10.482876597307914,\n   \"max\": 400.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   37.655,\n   37.0,\n   400.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Salary\",\n   \"properties\": {\n   \"dtype\": \"number\",\n

```
\"std\": 47214.004060407126,\n        \"min\": 400.0,\n
\"max\": 150000.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          69742.5,\n          70000.0,\n
400.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Purchase Iphone\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 141.27865845809384,\n        \"min\":
0.0,\n        \"max\": 400.0,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          0.3575,\n          1.0,\n
0.479863963596869\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
df.isnull().sum()
```

```
Gender             0
Age                0
Salary             0
Purchase Iphone    0
dtype: int64
```

```
df = pd.get_dummies(df, columns=['Gender'], drop_first=True)
```

```
X = df[['Age', 'Salary']] # Select 'Age' and 'Salary' as independent
variables
y = df['Purchase Iphone']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=0)
```

```
!pip install scikit-learn
from sklearn.preprocessing import StandardScaler # Import
StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.25.2)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

```
classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski',
p=2)
```

```python
classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

```python
y_pred = classifier.predict(X_test)
```

```python
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.93
```

```python
from sklearn.linear_model import LogisticRegression
```

```python
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
LogisticRegression()
```

```python
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.89
```

```python
from sklearn.linear_model import LinearRegression
```

```python
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```python
y_pred = model.predict(X_test)
```

```python
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.09872045740502204
R-squared: 0.5463214273666266
```

```python
from sklearn.tree import DecisionTreeClassifier
```

```python
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```python
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.91
```

```python
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```python
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.93
```