Data Mining Practical — Full World Report (All Slips with Proper Answers)

Note: Q3 (Viva) and Q4 (Internal Assessment) are skipped. This report contains clear, complete answers for all practical slips (Q1 & Q2) from the Data Mining practical paper.

Slip 1

Q1. Read dataset `Data.csv` and extract any one (dependent/independent) variable and print.

```
import pandas as pd
df = pd.read_csv('Data.csv')
print('Full dataset head:')
print(df.head())
col = df.iloc[:, 0]
print('\nExtracted column values:')
print(col.head())
```

Q2. Implement Naive Bayes on `Social_Network_Ads.csv` and print confusion matrix.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, classification_report

df = pd.read_csv('Social_Network_Ads.csv')
X = df[['Age','EstimatedSalary']]
y = df['Purchased']
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.25,random_state=42)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('\nConfusion Matrix:')
print(confusion_matrix(y_test, y_pred))
print('\nClassification Report:')
print(classification_report(y_test, y_pred))
```

Slip 2

Q1. Read `Data.csv`, split dataset into training and testing set and print.

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('Data.csv')
X = df.drop(columns=['target'], errors='ignore')
y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)
print('\nX_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
```

Q2. Naive Bayes on `Iris.csv` and print confusion matrix.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix

iris = pd.read_csv('Iris.csv')
X = iris.iloc[:, 1:5]
y = iris['Species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('\nConfusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

Slip 3

Q1. Read `Data.csv` and label encode any one variable.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('Data.csv')
```

```
le = LabelEncoder()
column = df.select_dtypes(include=['object']).columns[0]
df[column + '_encoded'] = le.fit_transform(df[column])
print('\nOriginal and Encoded Values:')
print(df[[column, column + '_encoded']].head())
```

Q2. Perform Simple Linear Regression using sklearn and print estimated coefficients.

```
import numpy as np
from sklearn.linear_model import LinearRegression

X = np.array([1,2,3,4,5,6,7,8]).reshape(-1,1)
Y = np.array([7,14,15,18,19,21,26,23])
model = LinearRegression()
model.fit(X, Y)
print('\nIntercept:', model.intercept_)
print('Coefficient:', model.coef_[0])
```

Slip 4

Q1. Read `Data.csv` and label encode any one variable.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('Data.csv')
le = LabelEncoder()
col = df.select_dtypes(include=['object']).columns[0]
df[col + '_encoded'] = le.fit_transform(df[col])
print('\nEncoded column:')
print(df[[col, col + '_encoded']].head())
```

Q2. Perform Linear Regression using SciPy.

```
import numpy as np
from scipy import stats

X = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
Y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
slope, intercept, r_value, p_value, std_err = stats.linregress(X, Y)
Y_pred = intercept + slope*X
```

```
print('\nSlope:', slope)
print('Intercept:', intercept)
print('\nReal Values:', Y)
print('Predicted Values:', Y_pred)
```

Slip 5

Q1. Perform Feature Scaling on `Data.csv`.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('Data.csv')
X = df.select_dtypes(include=['int64','float64'])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print('\nScaled Data:')
print(pd.DataFrame(X_scaled, columns=X.columns).head())
```

Q2. Perform K-Means clustering on `Mall_Customers_data.csv`.

```
import pandas as pd
from sklearn.cluster import KMeans

mall = pd.read_csv('Mall_Customers_data.csv')
X = mall[['Annual Income (k$)', 'Spending Score (1-100)']]
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X)
mall['cluster'] = kmeans.labels_
print('\nCluster Counts:')
print(mall['cluster'].value_counts())
```

Slip 6

Q1. Convert categorical data into numeric from `PlayTennis.csv`.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

```
play = pd.read_csv('PlayTennis.csv')
le = LabelEncoder()
for col in play.columns:
    if play[col].dtype == 'object':
        play[col + '_enc'] = le.fit_transform(play[col])
print('\nEncoded Data:')
print(play.head())
```

Q2. Perform Hierarchical Clustering on `Mall_Customers_data.csv`.

```
import pandas as pd
from scipy.cluster.hierarchy import linkage, fcluster

mall = pd.read_csv('Mall_Customers_data.csv')
X = mall[['Annual Income (k$)', 'Spending Score (1-100)']]
Z = linkage(X, method='ward')
labels = fcluster(Z, 5, criterion='maxclust')
mall['hcluster'] = labels
print('\nCluster Groups:')
print(mall.groupby('hcluster').size())
```

Slip 7

Q1. Split dataset into training/testing.

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('Data.csv')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)
print('\nTrain and Test sets created successfully.')
```

Q2. Perform Apriori Algorithm on `MilkButter.csv`.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```
transactions = []
with open('MilkButter.csv') as f:
    for line in f:
        items = [i.strip() for i in line.strip().split(',') if i.strip()]
        transactions.append(items)

t = TransactionEncoder()
df = pd.DataFrame(t.fit(transactions).transform(transactions),
columns=t.columns_)
freq = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(freq, metric='confidence', min_threshold=0.5)
print('\nApriori Rules:')
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

Slip 8

Q1. Extract variable from `Data.csv`.

```
import pandas as pd
df = pd.read_csv('Data.csv')
print('\nExtracted variable:')
print(df.iloc[:, 0].head())
```

Q2. Perform Apriori on `store_data.csv`.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

transactions = []
with open('store_data.csv') as f:
    for line in f:
        items = [i.strip() for i in line.strip().split(',') if i.strip()]
        transactions.append(items)

te = TransactionEncoder()
df = pd.DataFrame(te.fit(transactions).transform(transactions),
columns=te.columns_)
freq = apriori(df, min_support=0.004, use_colnames=True)
rules = association_rules(freq, metric='confidence', min_threshold=0.2)
rules = rules[rules['lift'] >= 3]
```

```
print('\nFiltered Rules:')
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

Slip 9

Q1. Perform Feature Scaling on `Data.csv`.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('Data.csv')
X = df.select_dtypes(include=['int64','float64'])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print('\nScaled Features:')
print(pd.DataFrame(X_scaled, columns=X.columns).head())
```

Q2. Perform Decision Tree Classification using Python.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix

iris = pd.read_csv('Iris.csv')
X = iris.iloc[:, 1:5]
y = iris['Species']
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, y)
y_pred = clf.predict(X)
print('\nDecision Tree Model Created.')
print('Confusion Matrix:')
print(confusion_matrix(y, y_pred))
```

Slip 10

Q1. Label encode any one variable from `Data.csv`.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

```
df = pd.read_csv('Data.csv')
le = LabelEncoder()
col = df.select_dtypes(include=['object']).columns[0]
df[col + '_encoded'] = le.fit_transform(df[col])
print('\nEncoded Data:')
print(df[[col, col + '_encoded']].head())
```

Q2. Perform Decision Tree Classification or Hierarchical Clustering (same as Slip 9).

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

iris = pd.read_csv('Iris.csv')
X = iris.iloc[:, 1:5]
y = iris['Species']
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X, y)
print('\nDecision Tree Model Trained Successfully.')
```