LAB # 7
OPEN-ENDED LAB

OBJECTIVE

The objectives for the tasks involve string analysis, dataframe manipulation, and predictive modeling techniques like KNN, Naïve Bayes, Decision Trees, Linear regression and K-Means Clustering,

Lab Task:

Task:1 Using a dictionary that holds information on students' scores, generate a pandas DataFrame from the provided data. Implement forward fill to interpolate any missing values in the DataFrame.

- **CODE:**

```
import pandas as pd
data = {
    "Student": ["Alice", "Bob", "Charlie", "David", "Eva"],
    "Math": [90, 85, None, 88, None],
    "Science": [None, 78, 92, None, 80],
    "English": [85, None, 87, 90, None]
}
df = pd.DataFrame(data)
df_filled = df.ffill()
print("Original DataFrame:")
print(df)
print("\nDataFrame after Forward Fill:")
print(df_filled)
```

- **OUTPUT:**

```
Original DataFrame:
   Student  Math  Science  English
0    Alice  90.0      NaN     85.0
1      Bob  85.0     78.0      NaN
2  Charlie   NaN     92.0     87.0
3    David  88.0      NaN     90.0
4      Eva   NaN     80.0      NaN

DataFrame after Forward Fill:
   Student  Math  Science  English
0    Alice  90.0      NaN     85.0
1      Bob  85.0     78.0     85.0
2  Charlie  85.0     92.0     87.0
3    David  88.0     92.0     90.0
4      Eva  88.0     80.0     90.0
```

Task:2 You are a data analyst working for a university. The administration wants to predict student performance in a particular course based on their past academic records. Your task is to design and implement a machine learning model.

- **CODE:**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
data = {
    "GPA": [3.1, 3.8, 2.5, 3.4, 3.0, 2.8, 3.6, 2.9, 3.7, 2.4],
    "Attendance": [90, 95, 70, 85, 80, 75, 92, 78, 88, 65],
    "Assignments": [85, 95, 65, 80, 70, 75, 90, 68, 88, 60],
    "Grade": [1, 1, 0, 1, 0, 0, 1, 0, 1, 0]  # 1: Pass, 0: Fail
}
X = pd.DataFrame(data).drop("Grade", axis=1)
y = pd.DataFrame(data)["Grade"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

- **OUTPUT:**

```
Accuracy: 1.0
```

Task:3 The bank wants to develop a system to classify loan applicants as either "high risk" or "low risk" based on their credit history and other factors. Your task is to design and implement a machine learning model.

- **CODE:**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

data = {
    "Credit_Score": [700, 650, 600, 550, 720, 580, 610, 640, 670, 500],
    "Income": [5000, 4500, 4000, 3000, 5200, 3200, 4100, 4600, 4800, 2500],
    "Loan_Amount": [200, 180, 150, 120, 220, 130, 160, 190, 210, 100],
    "Risk": [0, 0, 1, 1, 0, 1, 1, 0, 0, 1]  # 0: Low Risk, 1: High Risk
}
X = pd.DataFrame(data).drop("Risk", axis=1)
y = pd.DataFrame(data)["Risk"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

- **OUTPUT:**

```
Accuracy: 1.0
```

Task:4 A company wants to develop a system to classify emails as either "spam" or "not spam" based on the email's content.

- **CODE:**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

data = {
    "Email_Content": [
        "Congratulations, you have won a lottery!",
        "Meeting scheduled for tomorrow",
        "Claim your free prize now",
        "Can we reschedule the call?",
        "Exclusive offer just for you",
        "Project update attached",
        "Win a brand new car",
        "Please find the invoice attached",
        "Urgent: Your account has been compromised",
        "Let's catch up soon!"
    ],
    "Label": [1, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # 1: Spam, 0: Not Spam
}

emails = pd.DataFrame(data)
X = emails["Email_Content"]
y = emails["Label"]

vectorizer = CountVectorizer()
X_vectorized = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2, random_state=42)

model = MultinomialNB()
model.fit(X_train, y_train)

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

- **OUTPUT:**

```
Accuracy: 0.5
```

Task:5 Let us consider a hospital wants to develop a system to diagnose patients with either "disease A" or "disease B" based on their symptoms.

- **CODE:**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

data = {
    "Fever": [1, 0, 1, 1, 0, 0, 1, 0, 1, 0],
    "Cough": [1, 1, 1, 0, 0, 1, 1, 0, 0, 0],
    "Fatigue": [1, 1, 0, 1, 0, 0, 1, 0, 1, 0],
    "Headache": [0, 1, 1, 0, 1, 0, 1, 0, 1, 0],
    "Diagnosis": [0, 1, 0, 0, 1, 1, 0, 1, 0, 1]
}

X = pd.DataFrame(data).drop("Diagnosis", axis=1)
y = pd.DataFrame(data)["Diagnosis"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

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

- **OUTPUT:**

```
Accuracy: 1.0
```

Task:6 A real estate company wants to predict the prices of houses based on their features.

- **CODE:**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

data = {
    'Feature1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Feature2': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
    'Target': [3, 6, 8, 11, 14, 17, 20, 23, 26, 29]
}

df = pd.DataFrame(data)
X = df[['Feature1', 'Feature2']]
y = df['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

- **OUTPUT:**

```
Mean Squared Error: 0.1928136147443519
```