LAB # 07

Open Ended Lab

Lab Tasks:-

- **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
import numpy as np

data = {
    "Student": ["A", "B", "C", "D"],
    "Math": [85, None, 78, None],
    "Physics": [88, 90, None, 75],
    "Chemistry": [None, 75, 72, None]
}

df = pd.DataFrame(data).set_index("Student")
df = df.ffill()

print(df)
```

Output:-

```
         Math  Physics  Chemistry
Student
A        85.0     88.0        NaN
B        85.0     90.0       75.0
C        78.0     90.0       72.0
D        78.0     75.0       72.0
```

- **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.neighbors import KNeighborsClassifier

df = pd.DataFrame({
    "GPA":[2.5,3.2,3.8,2.9,3.6],
    "Attendance":[70,85,95,60,90],
    "Result":["fail","pass","pass","fail","pass"]
})

X = df[["GPA","Attendance"]]
y = df["Result"]

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,y_train)

print("Prediction:", model.predict(X_test))
```

Output:-

```
Prediction: ['pass']
```

- **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.tree import DecisionTreeClassifier

df = pd.DataFrame({
    "CreditScore":[500,650,720,580,800],
    "Income":[20000,40000,55000,25000,70000],
    "Risk":["high","low","low","high","low"]
})

X = df[["CreditScore","Income"]]
y = df["Risk"]

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)

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

print("Prediction:", model.predict(X_test))
```

Output:-

```
Prediction: ['low']
```

- **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.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

df = pd.DataFrame({
    "text":["win a prize now","meeting at 3pm","free money offer","project update"],
    "label":["spam","ham","spam","ham"]
})

cv = CountVectorizer()
X = cv.fit_transform(df["text"])
y = df["label"]

model = MultinomialNB()
model.fit(X,y)

print(model.predict(cv.transform(["free gift for you"])))
```

Output:-

```
['spam']
```

- **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.neighbors import KNeighborsClassifier

df = pd.DataFrame({
    "Fever":[8,2,9,1,7],
    "Cough":[5,3,6,2,4],
    "Diagnosis":["A","B","A","B","A"]
})

X = df[["Fever","Cough"]]
y = df["Diagnosis"]

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X,y)

print(model.predict([[7,4]]))
```

Output:-

```
['A']
```

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

Code:-

```
import pandas as pd
from sklearn.linear_model import LinearRegression

df = pd.DataFrame({
    "Area":[1000,1500,2000,1200,1800],
    "Bedrooms":[2,3,3,2,4],
    "Price":[120000,180000,250000,140000,220000]
})

X = df[["Area","Bedrooms"]]
y = df["Price"]

model = LinearRegression()
model.fit(X,y)

print("Predicted Price:", model.predict([[1600,3]]))
```

Output:-

```
Predicted Price: [195014.4092219]
```