**Data Mining and Analytics**

**CBS3007**

**DA 3**

**HARSHITH KUMAR**

**21BBS0163**

**QUESTION 1**

**AIM**

Implementation of KNN classification on dataset

**PACKAGES**

Pandas, Numpy, Matplotlib, Seaborn, Sklearn

**SAMPLE INPUT**

<https://github.com/harshith363/Lab-Mining/blob/main/DA3/student_data.csv>

**Code**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

print("Harshith Kumar 21BBS0163")

df = pd.read\_csv("student\_data.csv")

X = df[['Attendance', 'Marks (Data Mining)']]

y = np.where(X['Attendance'] < 75, 'Drop',

             np.where(X['Marks (Data Mining)'] < 40, 'Fail', 'Pass'))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_scaled, y\_train)

y\_pred = knn.predict(X\_test\_scaled)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

all\_predictions = knn.predict(scaler.transform(X))

df['Prediction'] = all\_predictions

print("\nStudent Classifications:")

for \_, row in df.iterrows():

    print(

        f"Roll Number: {row['Roll Number']}, Classification: {row['Prediction']}")

# Plotting

plt.figure(figsize=(12, 6))

# Scatter plot for Attendance vs Marks, colored by Prediction

sns.scatterplot(data=df, x='Attendance', y='Marks (Data Mining)', hue='Prediction',

                palette='Set1', style='Prediction', markers={"Drop": "X", "Fail": "o", "Pass": "s"}, s=100)

plt.title("Student Performance Classification")

plt.xlabel("Attendance (%)")

plt.ylabel("Marks (Data Mining)")

plt.axhline(40, color='red', linestyle='--',

            label='Pass/Fail Threshold (40 Marks)')

plt.axvline(75, color='orange', linestyle='--',

            label='Drop Threshold (75 Attendance)')

plt.legend()

plt.grid()

plt.show()

# Count the number of students in each category

drop\_count = (df['Prediction'] == 'Drop').sum()

fail\_count = (df['Prediction'] == 'Fail').sum()

pass\_count = (df['Prediction'] == 'Pass').sum()

print(f"\nTotal students at risk of dropping out: {drop\_count}")

print(f"Total failing students: {fail\_count}")

print(f"Total passing students: {pass\_count}")

**OUTPUT**

A graph with red and blue squares

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**RESULT**

<>

**QUESTION 2**

**AIM**

Implement the Linear regression Technique for prediction

**PACKAGES**

Pandas, Numpy, Matplotlib, Sklearn

**SAMPLE INPUT**

<https://github.com/harshith363/Lab-Mining/blob/main/DA3/demat_account_counts.csv>

**CODE**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

print(“Harshith Kumar 21BBS0163”)

df = pd.read\_csv("demat\_account\_counts.csv")

df['Month'] = pd.to\_datetime(df['Month'])

df = df.sort\_values('Month')

df['Month\_Num'] = (df['Month'] - df['Month'].min()).dt.days // 30  # Approximate months

X = df[['Month\_Num']]

y = df['Count of DEMAT Accounts']

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}")

jan\_2025\_num = (pd.to\_datetime('2025-01-01') - df['Month'].min()).days // 30

predicted\_count = model.predict([[jan\_2025\_num]])

print(f"Predicted count of DEMAT accounts for January 2025: {int(predicted\_count[0])}")

# Visualization

plt.figure(figsize=(12, 6))

plt.scatter(df['Month'], df['Count of DEMAT Accounts'], color='blue', label='Actual Data')

plt.plot(df['Month'], model.predict(X), color='red', label='Regression Line')

plt.scatter(pd.to\_datetime('2025-01-01'), predicted\_count, color='green', label='Prediction for Jan 2025', s=100)

plt.title("DEMAT Accounts Count Prediction")

plt.xlabel("Month")

plt.ylabel("Count of DEMAT Accounts")

plt.legend()

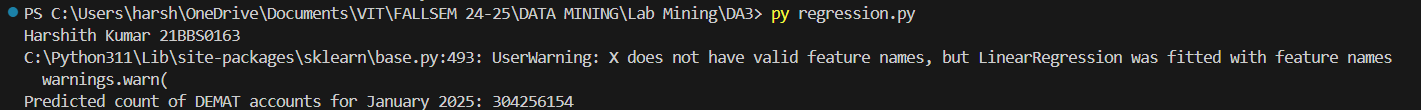
plt.grid()

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**OUTPUT**



**RESULT**

<>

**QUESTION 3**

**AIM**

Implement the Random Forest Supervised Machine Learning Algorithm

**PACKAGES**

Pandas, Seaborn, Matplotlib, Sklearn

**SAMPLE INPUT**

<https://github.com/harshith363/Lab-Mining/blob/main/DA3/fruits_dataset.csv>

**CODE**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print("Harshith Kumar 21BBS0163")

df = pd.read\_csv("fruits\_dataset.csv")

df['Fruit Type'] = df['Fruit Type'].astype('category').cat.codes

X = df.drop('Fruit Type', axis=1)

y = df['Fruit Type']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

print("Accuracy Score:", accuracy\_score(y\_test, y\_pred))

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=df['Fruit Type'].astype('category').cat.categories, yticklabels=df['Fruit Type'].astype('category').cat.categories)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# Plotting Feature Importance

feature\_importances = model.feature\_importances\_

features = X.columns

importance\_df = pd.DataFrame({'Feature': features, 'Importance': feature\_importances})

importance\_df = importance\_df.sort\_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(x='Importance', y='Feature', data=importance\_df)

plt.title('Feature Importance')

plt.show()

# Plotting Distribution of Features

plt.figure(figsize=(12, 10))

for i, feature in enumerate(X.columns):

    plt.subplot(2, 2, i + 1)

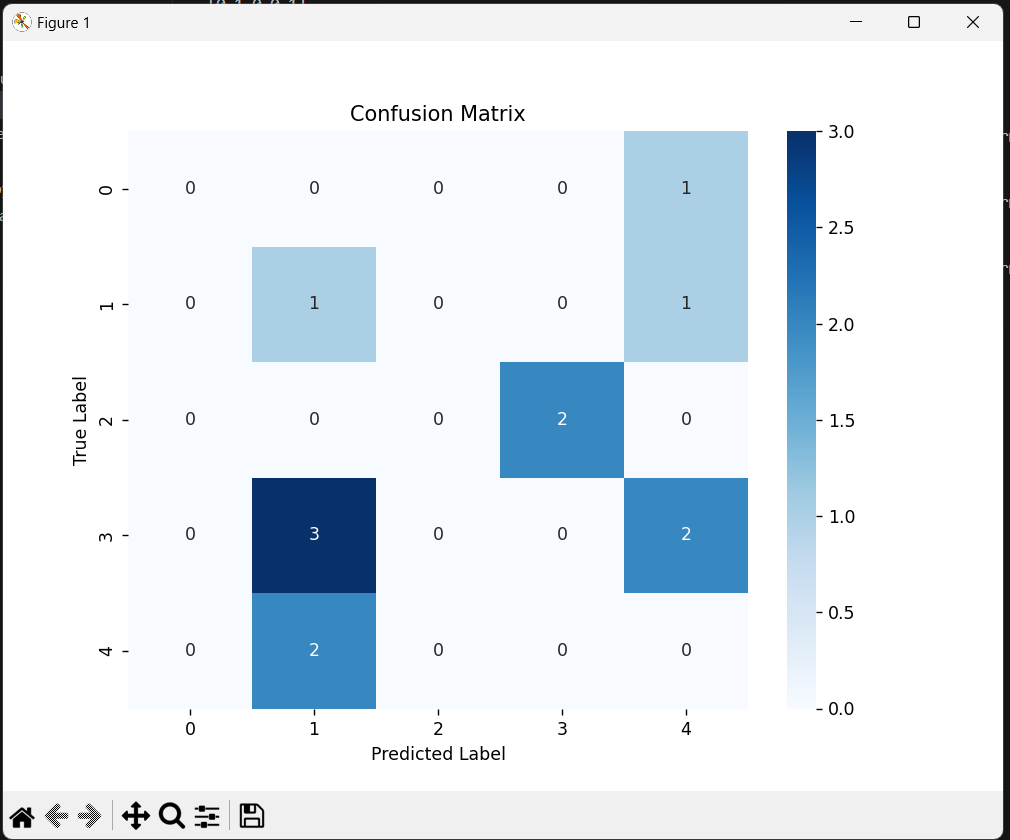
    sns.histplot(X[feature], bins=15, kde=True)

    plt.title(f'Distribution of {feature}')

plt.tight\_layout()

plt.show()

**OUTPUT**



A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated

A computer screen shot of text

Description automatically generated

**RESULT**

<>