

**Course Code : CBS3007**

**Course: Data Mining and Analytics**

Alan Thomas

21BBS0115

Github link for the datasets and code-

<https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA3>

**SECTION 1**

**Aim:**

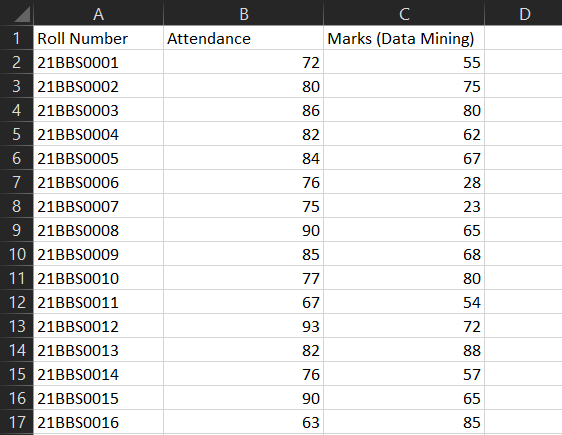
Implementation of KNN classification on dataset

Libraries Used : Numpy, Pandas, sklearn, matplotlib, seaborn

Dataset :

<https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA3/Q1>

Sample Input



**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("21BBS0115")

df=pd.read\_csv(r"C:\Users\LENOVO\Documents\Important\_documents\VIT\Semesters\sem7\DATA MINING\DA\DA3\Q1\student.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

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

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

palette={'Pass': 'green', 'Fail': 'red', 'Drop': 'blue'},

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()

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

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

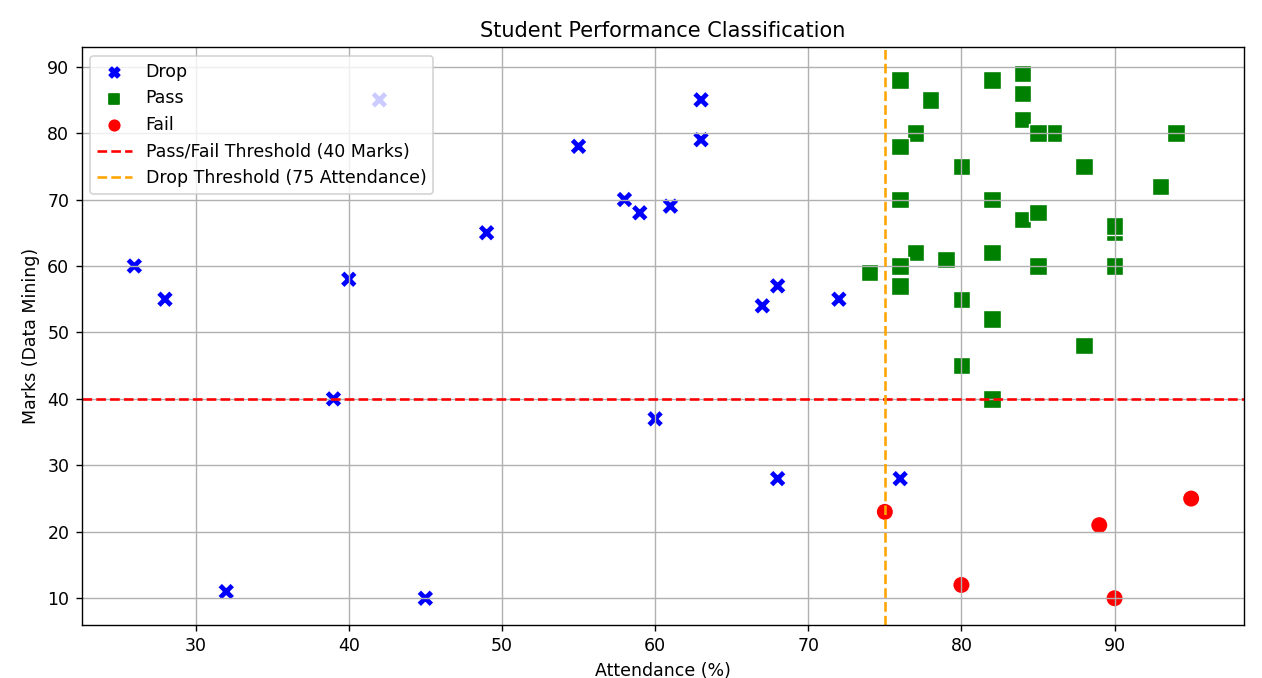
pass\_count = (df['Prediction'] == 'Pass').sum()

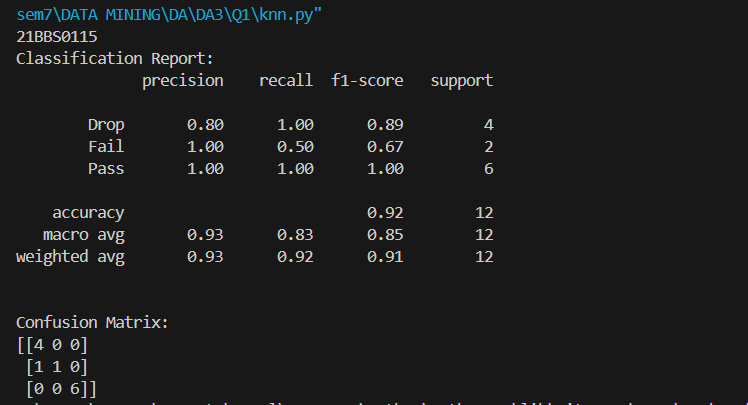
print(f"\nTotal students at risk of dropping out: {drop\_count}")

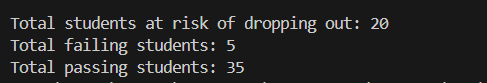
print(f"Total failing students: {fail\_count}")

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

Output :







RESULT:

The KNN was created thereby classifying the students into one of three categories, pass, fail or drop.

# SECTION 2

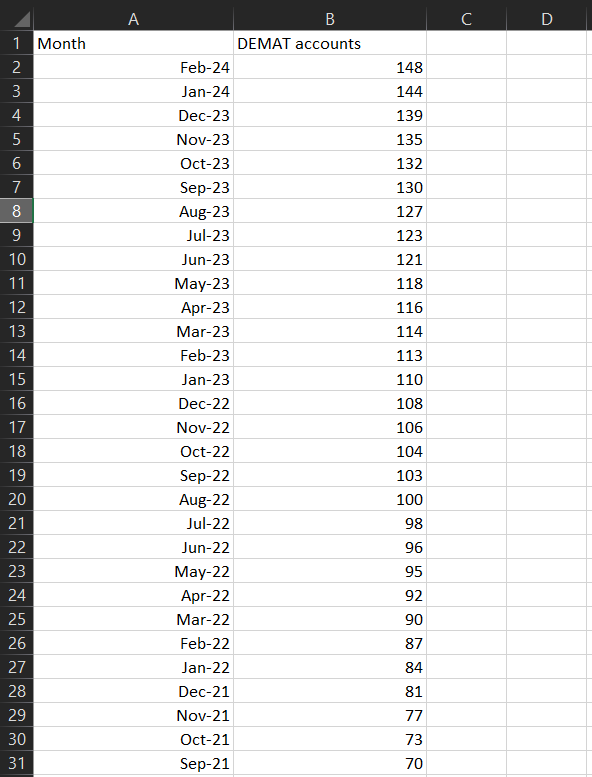
# Aim

Implement the Linear regression Technique for prediction of total number of DEMAT accounts for the month of January 2-25. The dataset consists of 60 past months.

LIBRARIES USED: Pandas, Numpy, Matplotlib, Sklearn

Dataset:   
<https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA3/Q2>

Sample Input



Code

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

from sklearn.metrics import r2\_score

data\_df = pd.read\_excel('Demat\_account\_dataset.xlsx')

data\_df.columns = ["Month" , "DEMAT\_accounts"]

data\_df['Date'] = pd.to\_datetime(data\_df['Month'], format='%B-%Y')

start\_date = pd.to\_datetime('January-2000', format='%B-%Y')

data\_df['Elapsed Time (Months)'] = ((data\_df['Date'].dt.year - start\_date.year) \* 12

+ data\_df['Date'].dt.month - start\_date.month)

data\_df['MontH\_Number'] = data\_df['Date'].dt.month

data\_df['Year'] = data\_df['Date'].dt.year

data\_df['Previous\_Month\_Account'] = data\_df['DEMAT\_accounts'].shift(-1)

data\_2 = data\_df.drop(["Month" , "Date" , "Previous\_Month\_Account"] , axis = 1)

data\_2.head()

X = data\_2[['Elapsed Time (Months)', 'MontH\_Number', 'Year']]

y = data\_df['DEMAT\_accounts']

X = X.dropna()

y = y[X.index]

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)

print("Predictions on test set:", y\_pred)

print("Model coefficients:", model.coef\_)

print("Model intercept:", model.intercept\_)

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

r2 = r2\_score(y\_test, y\_pred)

print("R-squared (model accuracy):", r2)

input\_data = {

'Elapsed Time (Months)': [300,299,298,297,296,295,294,293],

'MontH\_Number': [1,12,11,10,9,8,7,6],

'Year': [2025,2024,2024,2024,2024,2024,2024,2024],

}

input\_df = pd.DataFrame(input\_data)

predicted\_demat\_accounts = model.predict(input\_df)

for i in range(7,-1,-1):

print("Predicted DEMAT accounts for month " , input\_data['MontH\_Number'][i] ,"2025(in millions):", predicted\_demat\_accounts[i])

print("\n\nPredicted DEMAT accounts for January 2025(in millions):", predicted\_demat\_accounts[0])

print("21BBS0115")

X = data\_df[['Elapsed Time (Months)', 'MontH\_Number', 'Year']]

predicted\_accounts = (

model.intercept\_ +

model.coef\_[0] \* X['Elapsed Time (Months)'] +

model.coef\_[1] \* X['MontH\_Number'] +

model.coef\_[2] \* X['Year']

)

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

plt.plot(data\_df['Date'], data\_df['DEMAT\_accounts'], marker='o', label='Actual DEMAT Accounts')

plt.plot(data\_df['Date'], predicted\_accounts, color='red', label='Predicted Linear Regression Line')

plt.title('Demat Accounts Over Time with Linear Regression Line')

plt.xlabel('Month-Year')

plt.ylabel('Number of Demat Accounts')

plt.xticks(rotation=45)

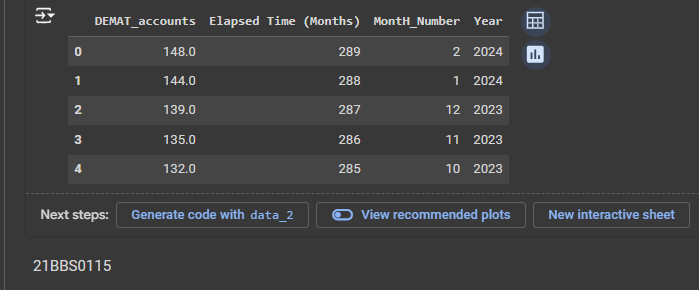
plt.legend()

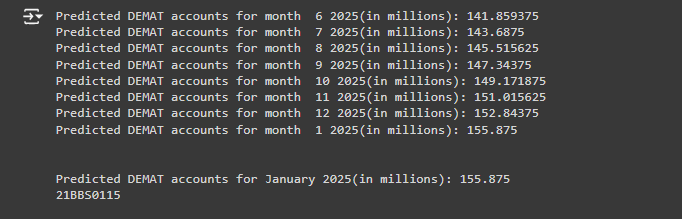
plt.grid()

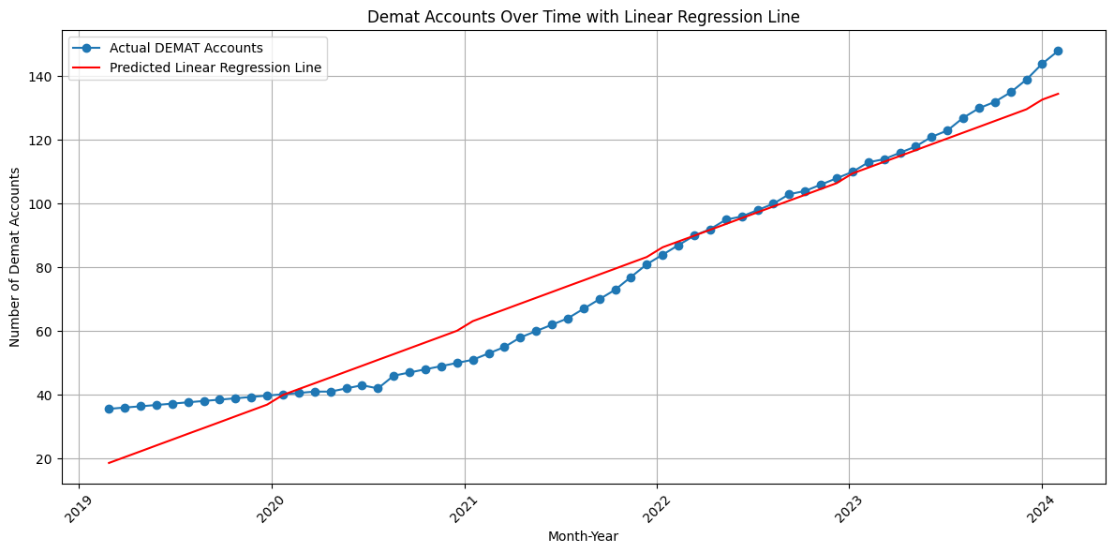
plt.tight\_layout()

plt.show()

Output







**Result:**

Successfully predicted the number of DEMAT accounts in Januray 2025 using 3 features. Tracked the same using a graph to make it clear how the number of DEMAT accounts is progressing through the years.

**SECTION 3**

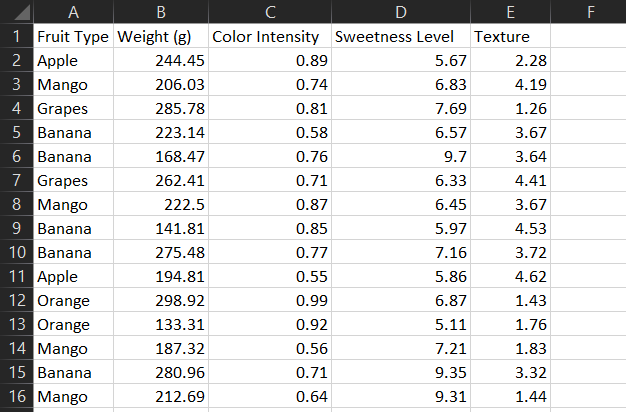
Aim : Implement the Random Forest Supervised Machine Learning Algorithm

Libraries : Numpy, Pandas, sklearn, seaborn

Dataset :

<https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA3/Q3>

Sample Input



# 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("21BBS0115")

df=pd.read\_csv(r"C:\Users\LENOVO\Documents\Important\_documents\VIT\Semesters\sem7\DATA MINING\DA\DA3\Q3\fruits.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='Greens', 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()

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

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

plt.subplot(2, 2, i + 1)

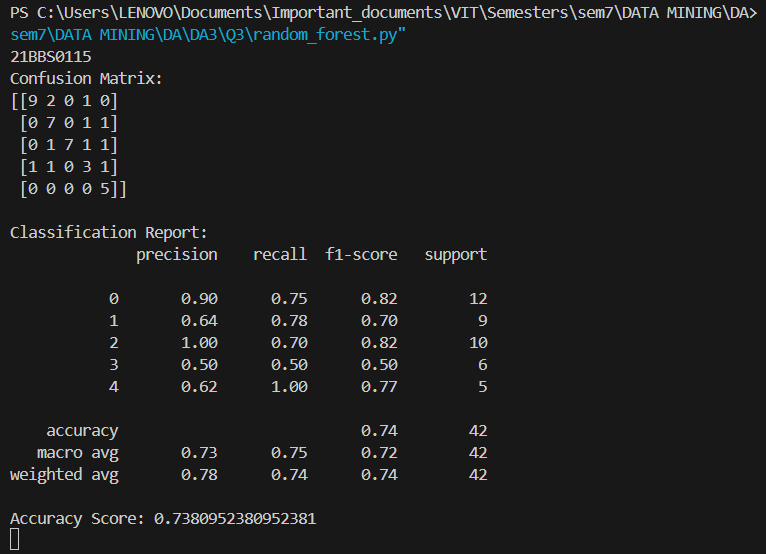
sns.histplot(X[feature], bins=15, kde=True)

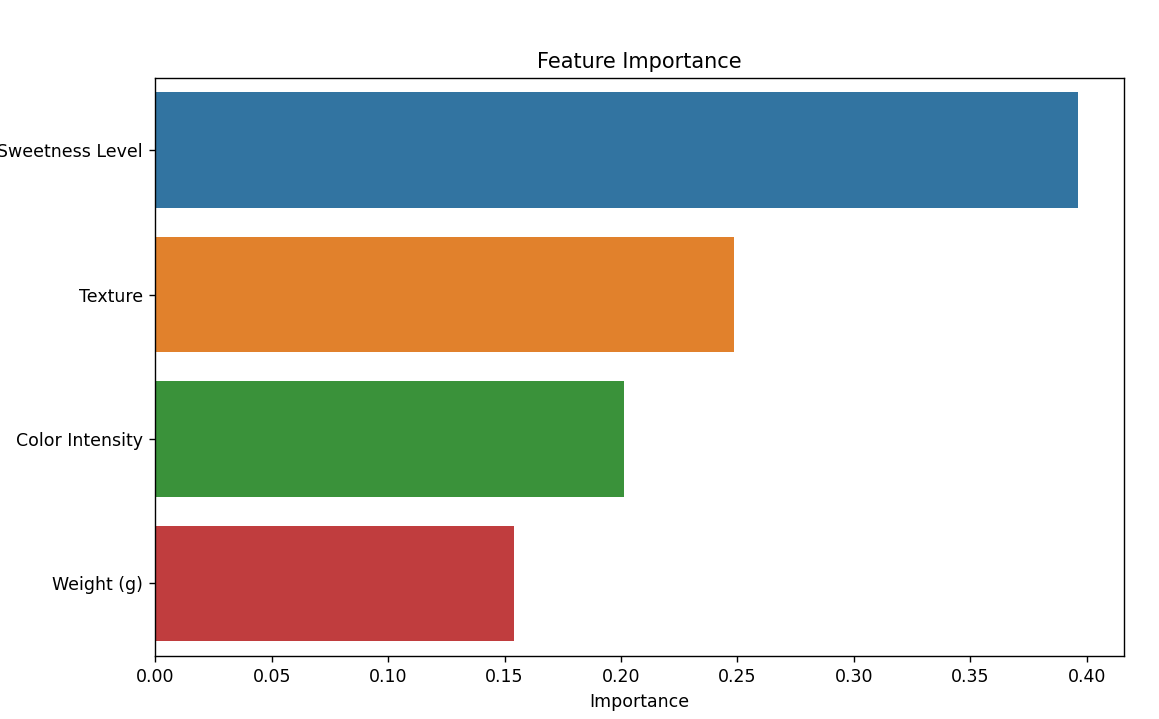
plt.title(f'Distribution of {feature}')

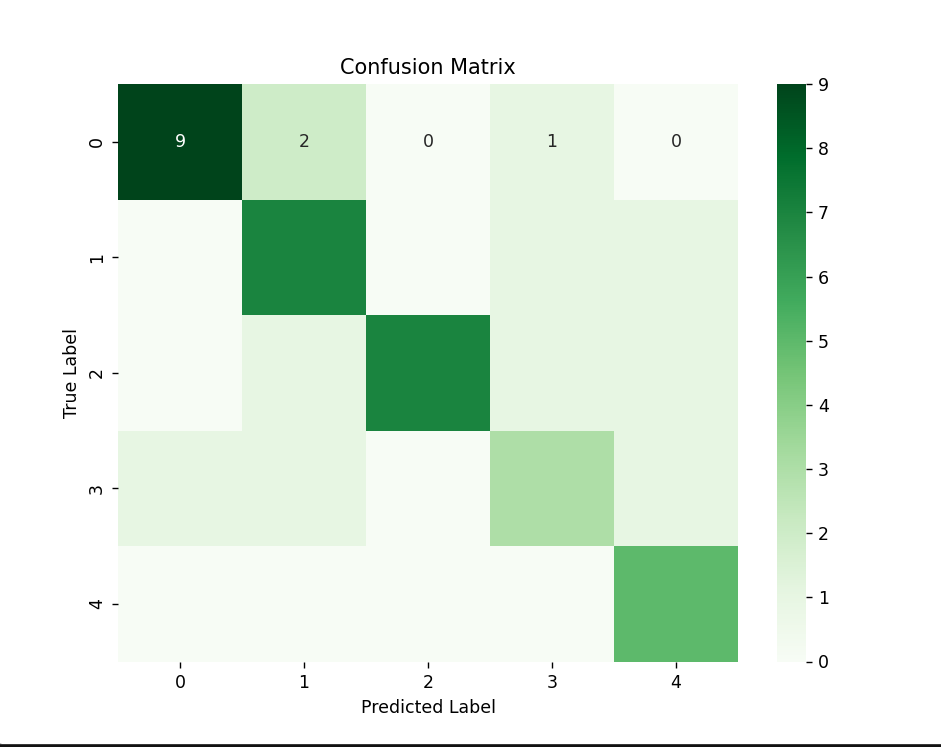
plt.tight\_layout()

plt.show()

Output :







**Result:**

We have created the Random Forest Algorithm, we were able to assign feature importance to each of the 4 features, and we were also able to classify the fruits into one of 5 categories.

XXXXXXXXXXXXXXXXXXXXXXXXX