Rayat Shikshan Sanstha's
KARMAVEER BHAURAO PATIL COLLEGE, VASHI
(Empowered Autonomous)

Reaccredited NAAC with Grade 'A++' (CGPA3.51)|ISO 9001:2015 Certified Institute
'Best College' Award by University of Mumbai

[DEPARTMENT OF INFORMATION TECHNOLOGY]

CERTIFICATE

This is to certify that **Mr.** SHUBHAM VIKAS NAVALE student of Class MScIT – II from **Karmaveer Bhaurao Patil College, Vashi, Navi Mumbai** has satisfactorily completed the practical course in subject **Machine Learning,** as per the syllabus laid by the college during the Academic Year **2025-26.**

ROLL NO.: 25176019

EXAM NO.:25176019

Date:12/11/2025

MANOJ CHOUDHARY
Course Coordinator

MADHURI GABHANE
Head of Department
Information Technology

External Examiner

Index

Sr No	Name of Practicals	Date	Signature
1	Diabetes Prediction Using Pima Indians Dataset		
2	Student Academic Performance Prediction (Pass/Fail Classification)		
3	Customer Churn Prediction for Telecom Industry		
4	Email Spam Detection and Classification		
5	Time-Series Anomaly Detection Techniques		
6	Human Activity Recognition Using Sensor Data		
7	Wine Quality Prediction Using Machine Learning		
8	Loan Eligibility Prediction Model		

Practical No :- 1

Aim:- Pima Indians Diabetes Prediction

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.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report


# --- Load dataset ---

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-
indians-diabetes.data.csv"

columns = [

    'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',

    'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'

]

data = pd.read_csv(url, header=None, names=columns)


# --- Explore dataset ---

print("First 5 rows:\n", data.head())

print("\nInfo:\n")

print(data.info())

print("\nSummary Statistics:\n", data.describe())


# --- Clean data: replace zeros with median ---
```

```
cols_to_fix = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
for col in cols_to_fix:
    data[col] = data[col].replace(0, np.nan)
    data[col] = data[col].fillna(data[col].median())

# --- Correlation heatmap ---
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.tight_layout()
plt.show()

# --- Outcome distribution ---
plt.figure(figsize=(6, 4))
sns.countplot(x='Outcome', data=data)
plt.title("Outcome Distribution")
plt.tight_layout()
plt.show()

# --- Prepare training data ---
X = data.drop('Outcome', axis=1)
y = data['Outcome']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# --- Train model ---
```

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

# --- Evaluate model ---
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy * 100:.2f}%")
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# --- Feature importance plot ---
importances = model.feature_importances_
feat_importances = pd.Series(importances, index=X.columns)
plt.figure(figsize=(10, 6))
feat_importances.sort_values().plot(kind='barh')
plt.title("Feature Importance from Random Forest")
plt.tight_layout()
plt.show()

# --- Predict for new sample patient ---
new_patient_data = pd.DataFrame([{
    'Pregnancies': 2,
    'Glucose': 120,
    'BloodPressure': 70,
    'SkinThickness': 30,
    'Insulin': 80,
    'BMI': 25.6,
```

'DiabetesPedigreeFunction': 0.6,

'Age': 32

}])

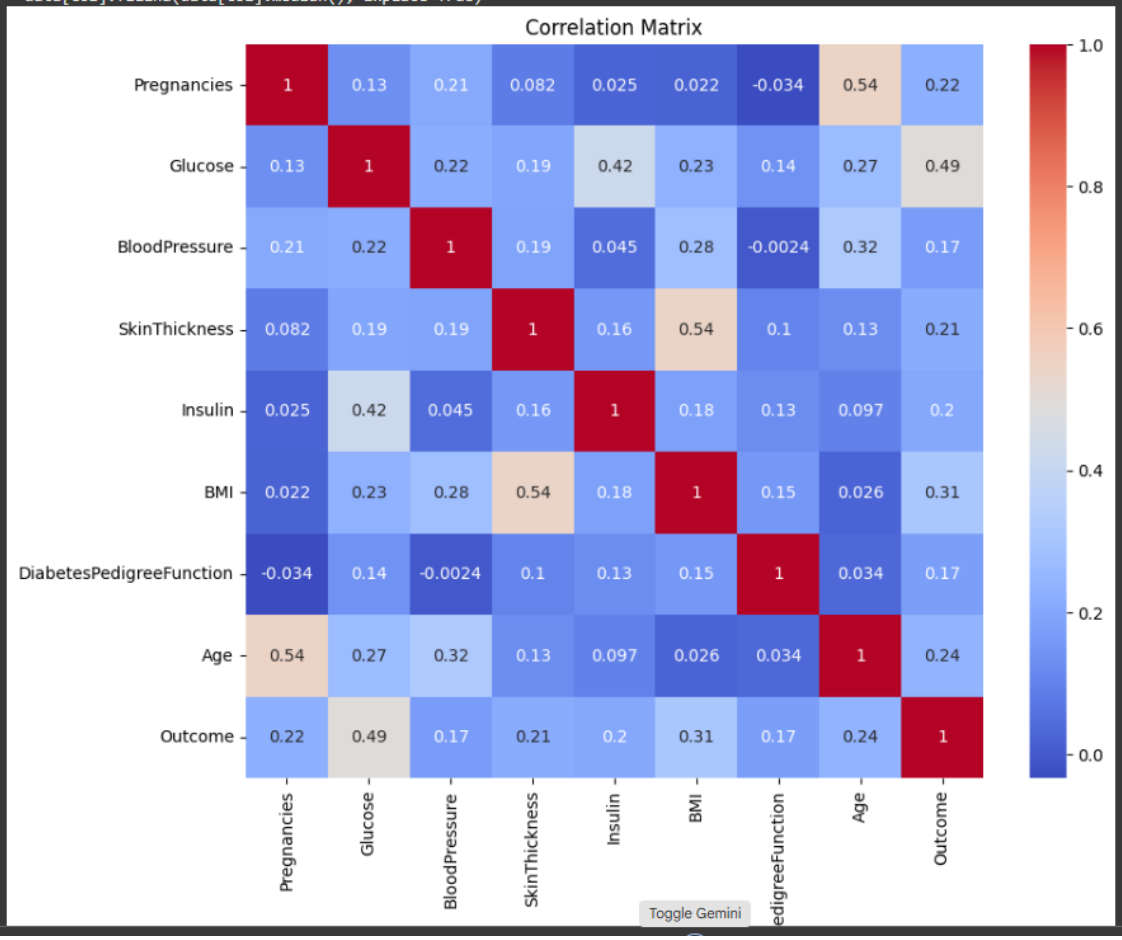
prediction = model.predict(new_patient_data)

print("\nPrediction for new patient:", "Likely diabetic." if prediction[0] == 1 else "Unlikely diabetic.")Output:-

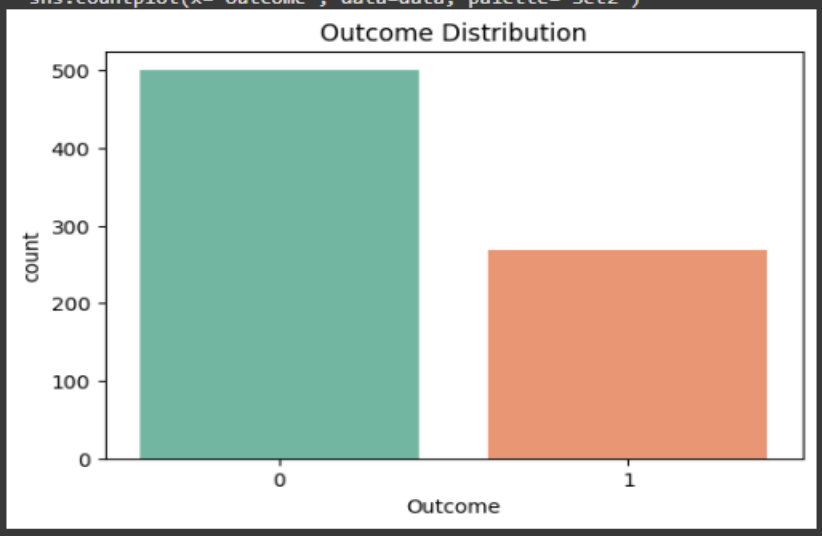
```
First 5 rows:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

```
Summary Statistics:
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
/tmp/ipython-input-966236812.py:29: FutureWarning: A value is trying to be set on a copy
The behavior will change in pandas 3.0. This inplace method will never work because the
```

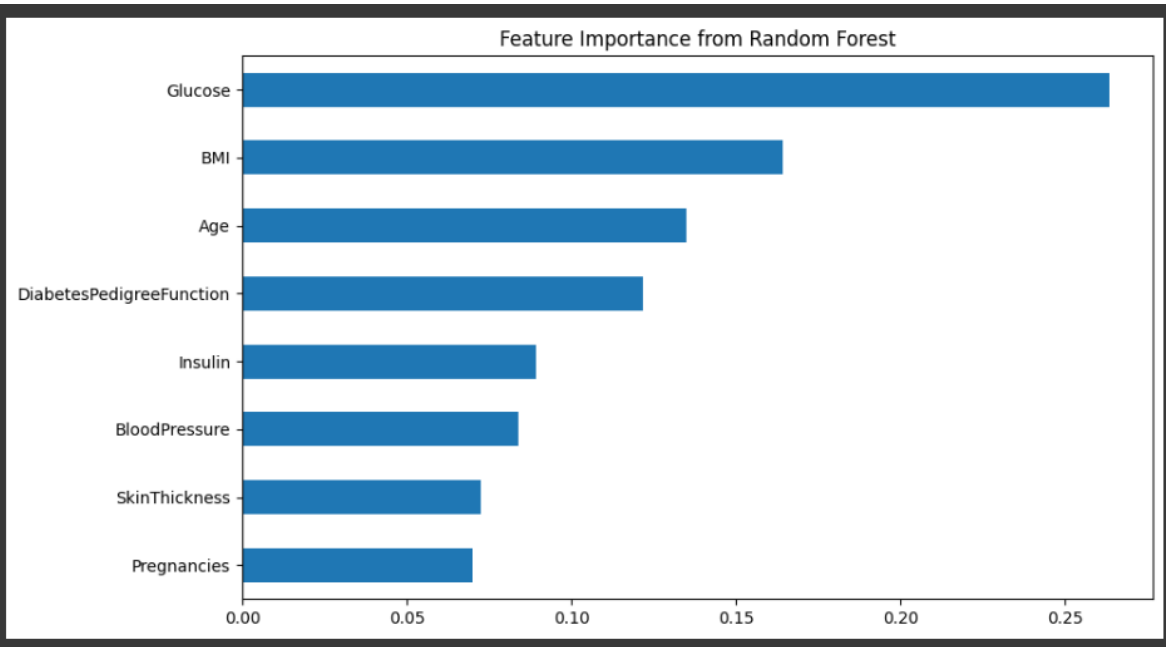
```
data[col].fillna(data[col].median(), inplace=True)
```



```
Passing `palette` without assigning `hue` is deprecated and will be removed
  sns.countplot(x='Outcome', data=data, palette='Set2')
```




```
Model Accuracy: 74.68%

Confusion Matrix:
 [[78 21]
 [18 37]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.79      0.80        99
           1       0.64      0.67      0.65        55

    accuracy                           0.75       154
   macro avg       0.73      0.73      0.73       154
weighted avg       0.75      0.75      0.75       154
```




```
Prediction for new patient: Unlikely diabetic.
```

Practical No :- 2

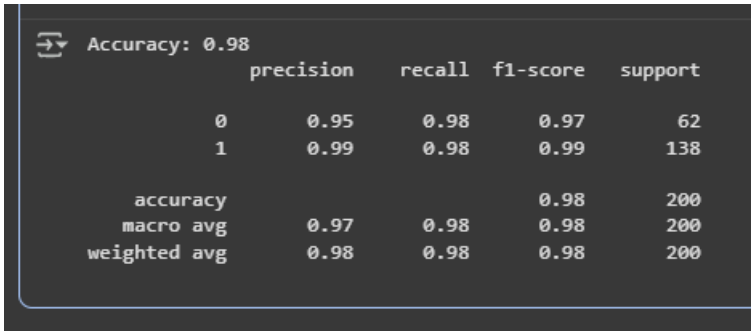
Aim:- Student Academic Performance (Pass/Fail Prediction)

Code:-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load data
df = pd.read_csv('StudentsPerformance.csv')

# Drop 'StudentID' if it exists
df = df.drop(columns=['StudentID'], errors='ignore')

# Create 'Pass_Fail' based on average score
passing_marks = 60
df['Pass_Fail'] = (df[['math score', 'reading score', 'writing score']].mean(axis=1)
>= passing_marks).astype(int)

# Encode categorical columns
df = pd.get_dummies(df, drop_first=True)

# Features and target
X = df.drop('Pass_Fail', axis=1)
y = df['Pass_Fail']
```

```
# Scale features
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)

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

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

Output:-



Practical No:- 3

Aim:- Customer Churn Prediction (Telco)

Code:-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Load
df = pd.read_csv('P3WA_Fn-UseC_-Telco-Customer-Churn.csv')

# Clean
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
# Fix: Avoid chained assignment warning by assigning back the filled column
df['TotalCharges'] = df['TotalCharges'].fillna(df['TotalCharges'].median())
df.drop('customerID', axis=1, inplace=True)

# Encode
df['Churn'] = df['Churn'].map({'Yes':1, 'No':0})
df.replace({'No phone service':'No', 'No internet service':'No'}, inplace=True)
df = pd.get_dummies(df, drop_first=True)

# Split
X = df.drop('Churn', axis=1)
y = df['Churn']
```

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

# Scale
num_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
scaler = StandardScaler()
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])

# Model
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

Output:-

```
/tmp/ipython-input-1682618412.py:12: FutureWarning: A value is trying to be set
The behavior will change in pandas 3.0. This inplace method will never work beca

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.met


  df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)
Accuracy: 0.794180269694819
              precision    recall  f1-score   support

           0       0.83      0.90      0.87      1035
           1       0.65      0.49      0.56       374

    accuracy                           0.79      1409
   macro avg       0.74      0.70      0.71      1409
weighted avg       0.78      0.79      0.78      1409

[[934 101]
 [189 185]]
```

Practical No :- 4

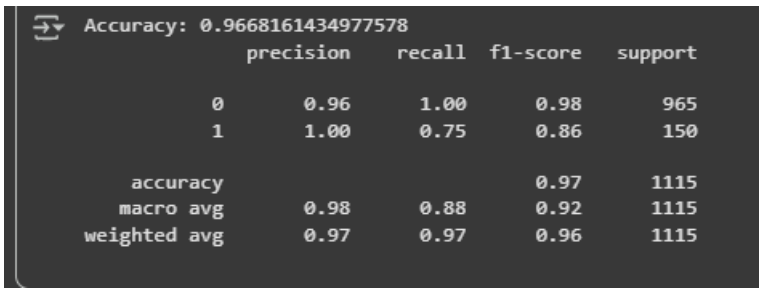
Aim:- Email Spam Classifier

Code:-

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

# Load (example: SMS Spam CSV)
df = pd.read_csv('spam.csv', encoding='latin-1')[['v1','v2']]
df.columns = ['label', 'text']
df['label'] = df['label'].map({'spam':1, 'ham':0})

X_train, X_test, y_train, y_test = train_test_split(
    df['text'], df['label'], test_size=0.2, random_state=42
)

vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf  = vectorizer.transform(X_test)

clf = MultinomialNB()
clf.fit(X_train_tfidf, y_train)
y_pred = clf.predict(X_test_tfidf)

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

```
print(classification_report(y_test, y_pred))
```

Output:-



```
Accuracy: 0.9668161434977578
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       965
           1       1.00      0.75      0.86       150

    accuracy                           0.97      1115
   macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.96      1115
```

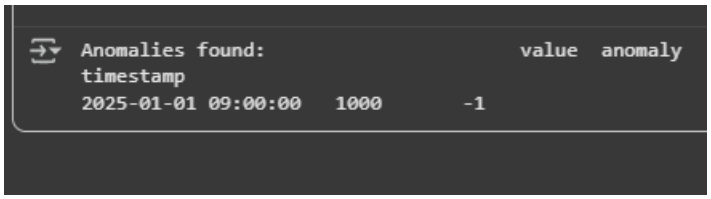
Practical No :- 5

Aim:- Time-Series Anomaly Detection

Code:-

```
import pandas as pd

import numpy as np

from sklearn.ensemble import IsolationForest


df = pd.read_csv('your_timeseries.csv', parse_dates=['timestamp'],
index_col='timestamp')

series = df['value']


model = IsolationForest(contamination=0.01, random_state=42)

df['anomaly'] = model.fit_predict(series.values.reshape(-1,1))

anomalies = df[df['anomaly'] == -1]

print("Anomalies found:", anomalies)
```

Output:-



Practical No :- 6

Aim:- Human Activity Recognition (HAR)

Code:-

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# --- Example dataset creation ---
# 100 samples, 6 sensor features
np.random.seed(42)
X = pd.DataFrame({
    'accel_x': np.random.randn(100),
    'accel_y': np.random.randn(100),
    'accel_z': np.random.randn(100),
    'gyro_x': np.random.randn(100),
    'gyro_y': np.random.randn(100),
    'gyro_z': np.random.randn(100),
})

# Example activity labels
activities = ['Walking', 'Running', 'Sitting']
y = pd.DataFrame({
    'Activity': np.random.choice(activities, size=100)
})
```

```
# Save to CSV (optional)
X.to_csv('X.csv', index=False)
y.to_csv('y.csv', index=False)
print("□ Files 'X.csv' and 'y.csv' created successfully.")

# --- Load CSVs (simulating real dataset) ---
X = pd.read_csv('X.csv')
y = pd.read_csv('y.csv')['Activity']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Train Random Forest classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict and evaluate
y_pred = clf.predict(X_test)
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

Output:-

```
Files 'X.csv' and 'y.csv' created successfully.

Classification Report:

              precision    recall  f1-score   support

     Running       0.50      0.29      0.36         7
     Sitting       0.71      0.71      0.71         7
     Walking       0.33      0.50      0.40         6

    accuracy                           0.50        20
   macro avg       0.52      0.50      0.49        20
weighted avg       0.53      0.50      0.50        20
```

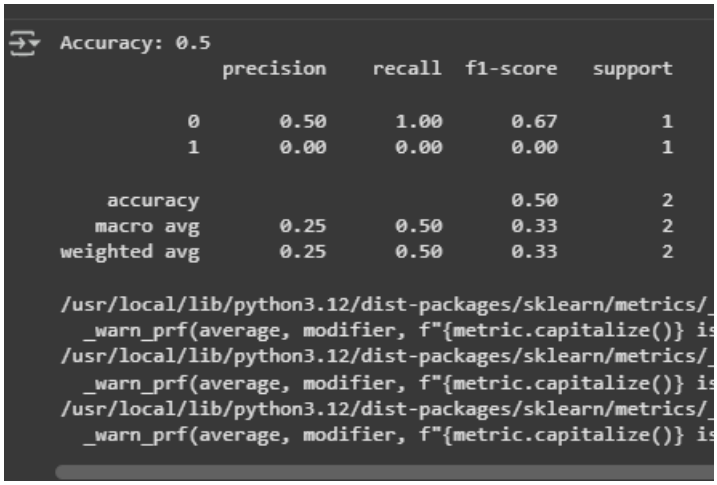
Practical N0:- 7

Aim:- Wine Quality Prediction

Code:-

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

df = pd.read_csv('winequality-red.csv', sep=';')
df['good_quality'] = (df['quality'] >= 6).astype(int)

X = df.drop(['quality', 'good_quality'], axis=1)
y = df['good_quality']

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

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

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

Output:-



```
Accuracy: 0.5
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_
  _warn_prf(average, modifier, f"{metric.capitalize()} is
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_
  _warn_prf(average, modifier, f"{metric.capitalize()} is
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_
  _warn_prf(average, modifier, f"{metric.capitalize()} is
```

Practical No:- 8

Aim:- Loan Eligibility Prediction

Code:-

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

# Load and preprocess data
df = pd.read_csv('loan_data.csv')
df['Loan_Status'] = df['Loan_Status'].map({'Y': 1, 'N': 0})
df.drop(['Loan_ID'], axis=1, inplace=True)
df = pd.get_dummies(df, drop_first=True)

X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']

# Stratified split with larger test size
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# Print class distributions
print("Train class distribution:\n", y_train.value_counts())
print("Test class distribution:\n", y_test.value_counts())

# Train model
```

```
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict on test set
y_pred = clf.predict(X_test)

# Print actual vs predicted labels
print("Actual labels:", y_test.values)
print("Predicted labels:", y_pred)

# Print accuracy and classification report (with zero_division=0 to avoid warnings)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, zero_division=0))
```

Output:-