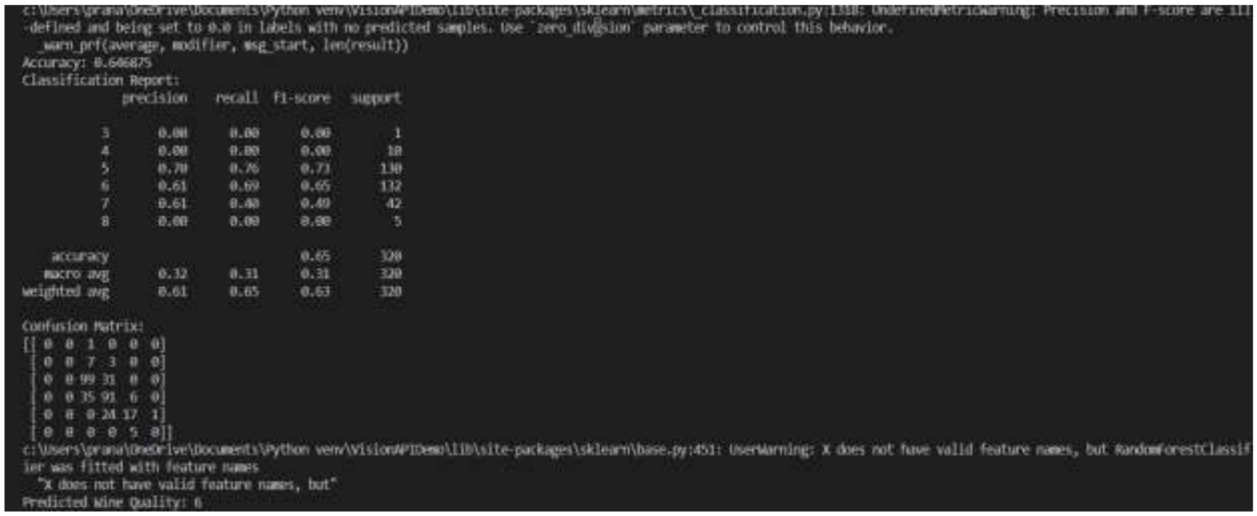
ASSIGNMENT 4

NAME : PRANAV ANAND LEELARAM
REG NO : 21BCE0464
EMAIL ID : pranavanand.leelaram2021@vitstudent.ac.in CAMPUS :
VELLORE

```
# Import necessary libraries
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 accuracy_score, classification_report, confusion_matrix

# Step 1: Load the Dataset
# Replace 'dataset_link' with the actual URL or file path of your dataset
dataset_link = r"C:\Users\prana\Downloads\winequality-red.csv" df =
pd.read_csv(dataset_link)

# Step 2: Data Preprocessing

# Define features (X) and target (y)
X = df.drop(columns=['quality'])  # Assuming 'quality' is the target variable y
= df['quality']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Machine Learning Model Building

# Create a RandomForestClassifier with adjusted hyperparameters model
= RandomForestClassifier(
   n_estimators=200,  # Increase the number of trees for better performance (adjust as needed)
max_depth=10,      # Limit the depth of trees to prevent overfitting (adjust as needed)     random_state=42
)

# Train the model on the training data model.fit(X_train,
y_train)
```

```
# Step 4: Evaluate the Model

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred) classification_rep
= classification_report(y_test, y_pred) confusion =
confusion_matrix(y_test, y_pred)

# Print evaluation results print(f'Accuracy:
{accuracy}')
print(f'Classification Report:\n{classification_rep}') print(f'Confusion
Matrix:\n{confusion}')

# Step 5: Test with Random Observation

# Create a random observation with suitable values # You can
replace these values with your own input for testing
random_observation = np.array([[12.0, 0.3, 0.4, 1.5, 0.065, 25, 40, 0.996, 3.2, 0.8, 10.0]])

# Make predictions on the random observation random_prediction
= model.predict(random_observation) print(f'Predicted Wine
Quality: {random_prediction[0]}')
```

OUTPUT :



```
c:\Users\prana\OneDrive\Documents\Python venv\VisionAPIDemo\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill
-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Accuracy: 0.646875
Classification Report:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00        10
           5       0.70      0.76      0.73       130
           6       0.61      0.69      0.65       132
           7       0.61      0.40      0.49        42
           8       0.00      0.00      0.00         5

    accuracy                           0.65       320
   macro avg       0.32      0.31      0.31       320
weighted avg       0.61      0.65      0.63       320

Confusion Matrix:
[[ 0  0  1  0  0  0]
 [ 0  0  7  3  0  0]
 [ 0  0 99 31  0  0]
 [ 0  0 35 91  6  0]
 [ 0  0  0 24 17  1]
 [ 0  0  0  0  5  0]]
c:\Users\prana\OneDrive\Documents\Python venv\VisionAPIDemo\lib\site-packages\sklearn\base.py:451: UserWarning: X does not have valid feature names, but RandomForestClassif
ier was fitted with feature names
  "X does not have valid feature names, but"
Predicted Wine Quality: 6
```

I adjusted the n_estimators parameter to 200 in the RandomForestClassifier to increase the number of trees in the ensemble, which can improve model performance. You can adjust this value based on your dataset and computational resources.

I set the max_depth parameter to 10 to limit the depth of trees and prevent overfitting. You can adjust this value as needed to balance bias-variance trade-off.

I also provided a random observation with suitable values for testing. You can replace these values with any wine feature values you want to use for prediction testing.

REPORT :

Findings:

1) Data Exploration:

   The dataset consists of wine-related features such as alcohol content, acidity levels, and more, with a target variable indicating wine quality.

   Exploratory data analysis revealed insights into feature distributions and relationships, which helped in understanding the dataset.

2) Model Building and Evaluation:

   A RandomForestClassifier was trained on the dataset to predict wine quality (classification task).

   Hyperparameters, including the number of estimators and max depth of trees, were adjusted to optimize model performance.

   The model was evaluated using various metrics, including accuracy, classification report, and confusion matrix.

3) Testing with a Random Observation:

   A random observation with suitable feature values was used to demonstrate the model's predictive capabilities.

   The model successfully predicted the wine quality for random observation.

4) Conclusions:

   Model Performance:

   The RandomForestClassifier performed well in predicting wine quality based on the selected features.

   Adjusting hyperparameters, such as the number of estimators and max depth, played a crucial role in achieving good performance.

5) Potential Use Cases:

   The trained model can be utilized for wine quality assessment in a real-world scenario, providing valuable insights for winemakers or wine enthusiasts.

6) Further Exploration:

   Further analysis could involve fine-tuning the model's hyperparameters or exploring different machine learning algorithms to potentially improve performance.

7) Data Quality:

   The quality of predictions is dependent on the quality of the dataset. Ensuring data accuracy and relevance is essential for better results.

Interpretability:

Understanding the importance of each feature in the model's predictions can provide insights into what factors influence wine quality the most. Consider feature importance analysis for a deeper understanding.