Machine Learning
Assignment 3
Aakanksha Darekar
202200733
A1 09

Consider remaining features of the wine data and prepare a prediction model for predicting quality of wine

Dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	fixed acidi	volatile ac	citric acid	residual su	chlorides	free sulfur	total sulfu	density	pH	sulphates	alcohol	quality	
2	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5	
3	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5	
4	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5	
5	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6	
6	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5	
7	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5	
8	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5	
9	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7	
10	7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7	
11	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5	
12	6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5	
13	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5	
14	5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5	
15	7.8	0.61	0.29	1.6	0.114	9	29	0.9974	3.26	1.56	9.1	5	
16	8.9	0.62	0.18	3.8	0.176	52	145	0.9986	3.16	0.88	9.2	5	
17	8.9	0.62	0.19	3.9	0.17	51	148	0.9986	3.17	0.93	9.2	5	
18	8.5	0.28	0.56	1.8	0.092	35	103	0.9969	3.3	0.75	10.5	7	
19	8.1	0.56	0.28	1.7	0.368	16	56	0.9968	3.11	1.28	9.3	5	
20	7.4	0.59	0.08	4.4	0.086	6	29	0.9974	3.38	0.5	9	4	

Code:

```
# Importing 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
from imblearn.over_sampling import SMOTE

# Load the dataset
# Replace 'wine_quality.csv' with your actual file path
file_path = 'C:/Users/admin/OneDrive/Desktop/6SEM/ML/winequality_red.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Dataset Preview:\n", data.head())
```

```
# Separating features and target variable
X = data.drop(columns=['quality'])  # Features
y = data['quality']                 # Target variable

# Splitting 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)

# Display dataset split info
print(f"Training set size: {X_train.shape[0]}")
print(f"Testing set size: {X_test.shape[0]}")

# Apply SMOTE to handle class imbalance
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

# Display class distribution after applying SMOTE
print("Class distribution after SMOTE:\n", pd.Series(y_train_balanced).value_counts())

# Initialize the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train_balanced, y_train_balanced)

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

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Detailed classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, zero_division=0))

# Confusion matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Feature importance (optional)
feature_importances = model.feature_importances_
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
```

```
    'Importance': feature_importances
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importance_df)

def plot_confusion_matrix(y_test, y_pred, labels=None):
    cm = confusion_matrix(y_test, y_pred, labels=labels)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
    disp.plot(cmap='Blues', values_format='d')
    plt.title("Confusion Matrix")
    plt.show()

# Call the functions
plot_confusion_matrix(y_test, y_pred)

# Plot the distribution of actual vs predicted values
def plot_actual_vs_predicted(y_test, y_pred):
    plt.figure(figsize=(10, 6))
    sns.histplot(y_test, color="blue", label="Actual", kde=True, bins=10, stat="density")
    sns.histplot(y_pred, color="orange", label="Predicted", kde=True, bins=10, stat="density")
    plt.title("Actual vs Predicted Distribution")
    plt.xlabel("Quality")
    plt.ylabel("Density")
    plt.legend()
    plt.show()

plot_actual_vs_predicted(y_test, y_pred)

# Plot feature importance
def plot_feature_importance(model, feature_names):
    feature_importances = model.feature_importances_
    importance_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance': feature_importances
    }).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.xlabel("Importance")
    plt.ylabel("Feature")
    plt.show()
```

# Call the functions
plot_feature_importance(model, X.columns)

Output:

```
Dataset Preview:
    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56
```

```
    alcohol  quality
0       9.4        5
1       9.8        5
2       9.8        5
3       9.8        6
4       9.4        5
Training set size: 1279
Testing set size: 320

Class distribution after SMOTE:
 quality
6    551
5    551
4    551
7    551
8    551
3    551
Name: count, dtype: int64
Accuracy: 0.62
```

Classification Report:

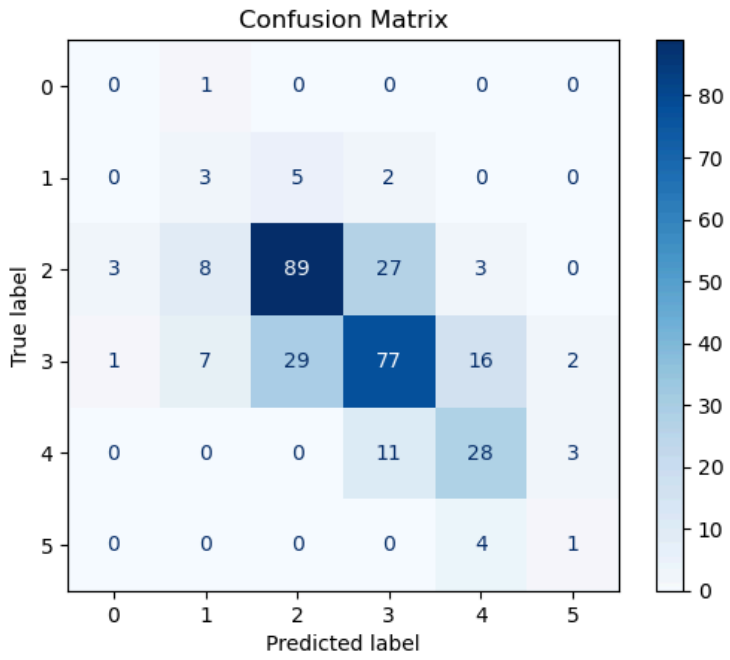
	precision	recall	f1-score	support
3	0.00	0.00	0.00	1
4	0.16	0.30	0.21	10
5	0.72	0.68	0.70	130
6	0.66	0.58	0.62	132
7	0.55	0.67	0.60	42
8	0.17	0.20	0.18	5
accuracy			0.62	320
macro avg	0.38	0.41	0.39	320
weighted avg	0.65	0.62	0.63	320

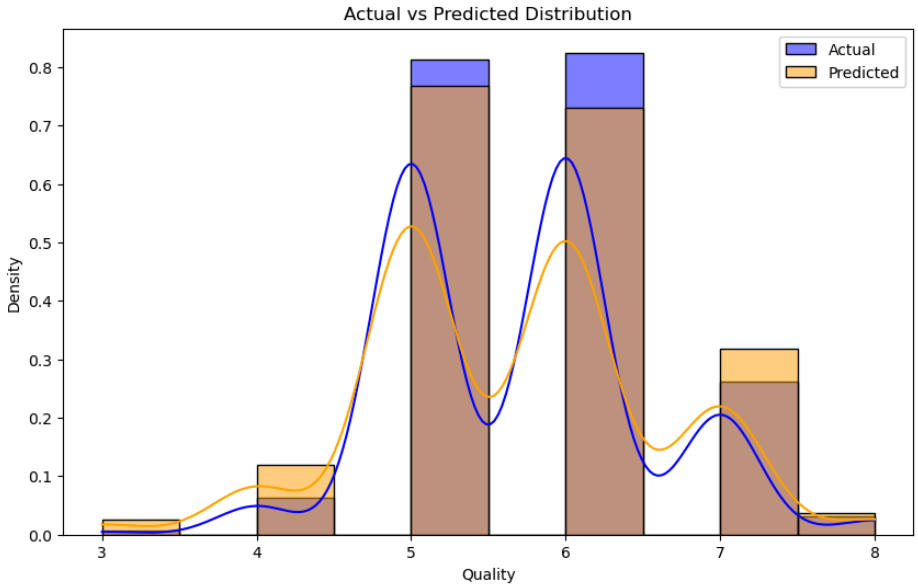
Confusion Matrix:

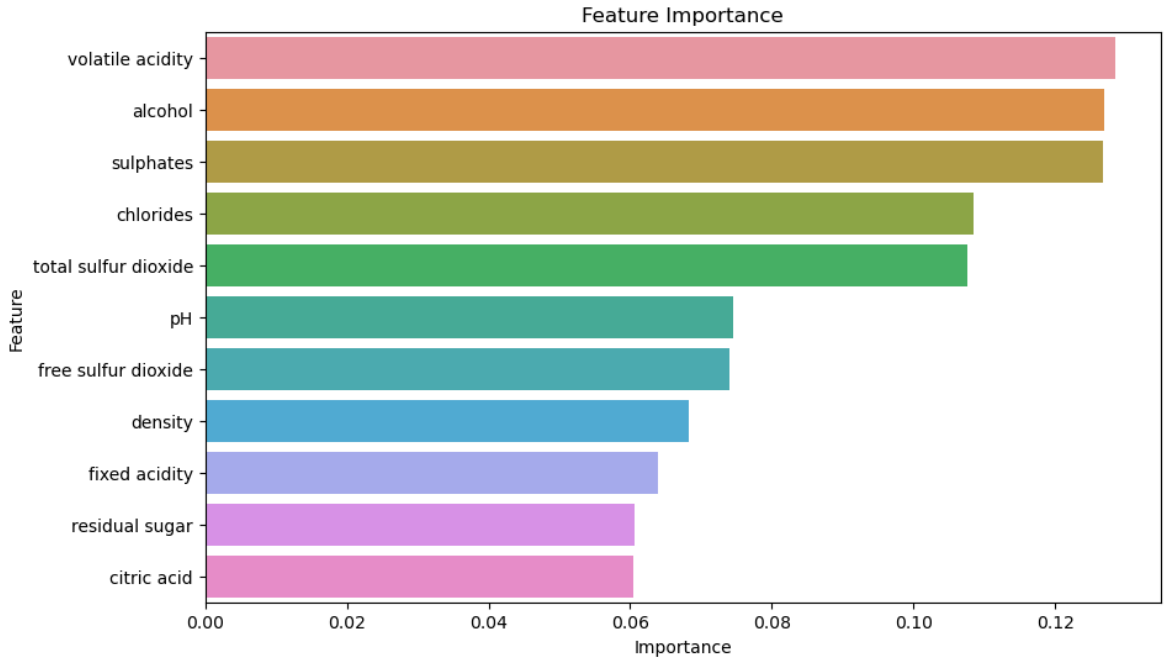
```
[[ 0  1  0  0  0  0]
 [ 0  3  5  2  0  0]
 [ 3  8 89 27  3  0]
 [ 1  7 29 77 16  2]
 [ 0  0  0 11 28  3]
 [ 0  0  0  0  4  1]]
```

Feature Importances:

	Feature	Importance
1	volatile acidity	0.128442
10	alcohol	0.126994
9	sulphates	0.126812
4	chlorides	0.108406
6	total sulfur dioxide	0.107594
8	pH	0.074475
5	free sulfur dioxide	0.073986
7	density	0.068334
0	fixed acidity	0.063913
3	residual sugar	0.060637
2	citric acid	0.060407


Confusion Matrix
True label
Predicted label
0 1 0 0 0 0
0 3 5 2 0 0
3 8 89 27 3 0
1 7 29 77 16 2
0 0 0 11 28 3
0 0 0 0 4 1
0 1 2 3 4 5
80 70 60 50 40 30 20 10 0


Actual vs Predicted Distribution
Actual
Predicted
Density
Quality
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
3 4 5 6 7 8


Feature Importance
volatile acidity
alcohol
sulphates
chlorides
total sulfur dioxide
pH
free sulfur dioxide
density
fixed acidity
residual sugar
citric acid
Feature
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Importance