Name :- Shreyash Ashok Pachpol

Reg No :- 21BCE0896

VIT Vellore

ASSIGNMENT 4

Project Title:

Grapes to Greatness: Machine Learning in Wine Quality Prediction

Description:

Predicting wine quality using machine learning is a common and valuable application in the field of data science and analytics. Wine quality prediction involves building a model that can assess and predict the quality of a wine based on various input features, such as chemical composition, sensory characteristics, and environmental factors.

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).

Dataset: link

Task:

- Load the Dataset
- Data preprocessing including visualization
- Machine Learning Model building
- Evaluate the model
- Test with random observation

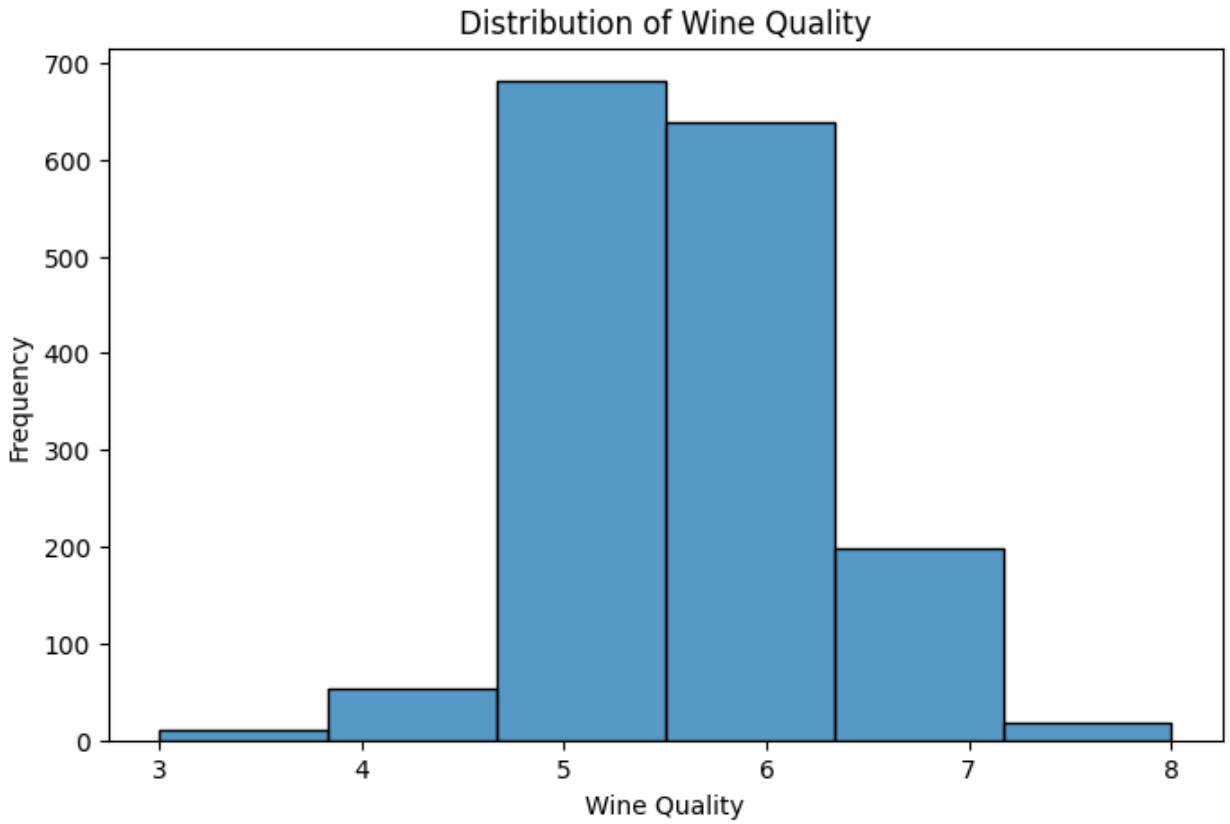
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score,
accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor
from sklearn import ensemble

# Load the dataset
url = '/content/winequality-red.csv'
data = pd.read_csv(url)
```

```python
# Data Preprocessing
# Check for missing values
print(data.isnull().sum())
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```python
# Data exploration and visualization
# Example: Histogram of wine quality
plt.figure(figsize=(8, 5))
sns.histplot(data['quality'], bins=6)
plt.xlabel('Wine Quality')
plt.ylabel('Frequency')
plt.title('Distribution of Wine Quality')
plt.show()
```




```
# Feature selection (if needed)
# Example: Selecting all features except 'quality' for regression
X = data.drop('quality', axis=1)
y = data['quality']

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

# Data scaling (if needed)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Machine Learning Model Building
model = RandomForestRegressor(n_estimators=100, random_state=42)  #
Example: Random Forest Regressor
model.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=42)
```

```
# Model Evaluation (Regression)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.3006603124999999
R-squared: 0.5399271357910311
```

```
# Model Evaluation (Classification, if applicable)
# Example: Convert wine quality to classes (e.g., low, medium, high)
y_train_class = pd.cut(y_train, bins=[0, 4, 7, 10], labels=['low',
'medium', 'high'])
y_test_class = pd.cut(y_test, bins=[0, 4, 7, 10], labels=['low',
'medium', 'high'])

from sklearn.ensemble import RandomForestClassifier  # Import the
RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

model_classification = RandomForestClassifier(n_estimators=100,
random_state=42)
model_classification.fit(X_train, y_train_class)
y_pred_class = model_classification.predict(X_test)

accuracy = accuracy_score(y_test_class, y_pred_class)
conf_matrix = confusion_matrix(y_test_class, y_pred_class)
class_report = classification_report(y_test_class, y_pred_class,
target_names=['low', 'medium', 'high'])

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:\n', conf_matrix)
print('Classification Report:\n', class_report)
```

```
Accuracy: 0.95
Confusion Matrix:
 [[  0   0   5]
 [  0   1  10]
 [  0   1 303]]
Classification Report:
              precision    recall  f1-score   support

         low       0.00      0.00      0.00         5
      medium       0.50      0.09      0.15        11
        high       0.95      1.00      0.97       304

    accuracy                           0.95       320
   macro avg       0.48      0.36      0.38       320
weighted avg       0.92      0.95      0.93       320
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: 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))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: 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))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: 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))
```

```
# Cross-validation (optional)
cv_scores = cross_val_score(model, X, y, cv=5)
print(f'Cross-validation scores: {cv_scores}')
```

```
Cross-validation scores: [0.25905968 0.340902   0.36511431 0.315745
0.26449954]
```

```
# Test with random observation
# Example: Create a new observation and predict its quality
new_observation = np.array([7.0, 0.2, 0.28, 1.8, 0.045, 40, 170, 0.99,
3.0, 0.47, 9.2])
new_observation = scaler.transform(new_observation.reshape(1, -1))
predicted_quality = model.predict(new_observation)
print(f'Predicted Wine Quality: {predicted_quality[0]}')
```

```
Predicted Wine Quality: 4.97
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
```