Artificial Intelligence – Lab

Semester Project

Spring 24

Muhammad Usama Waseem – L1F21BSCS0150

Table of Contents

Step 1: Data Loading and Exploratory Data Analysis

- Loaded the wine quality dataset.
- Conducted EDA to understand the data distribution and relationships between variables.

Code

```python
# Step 1: Read Dataset
file_path = 'winequality-red.csv'
data = pd.read_csv(file_path)
```

```python
# Step 2: Exploratory Data Analysis
print("Head of the DataFrame:\n", data.head())
print("Tail of the DataFrame:\n", data.tail())
print("Shape of the DataFrame:\n", data.shape)
print("Info of the DataFrame:\n")
data.info()
print("Description of the DataFrame:\n", data.describe())
print("Unique values:\n", data.nunique())
print("Value counts:\n", data['quality'].value_counts())
```

Screenshots

```
Head of the DataFrame:
   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
Tail of the DataFrame:
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol  quality
1594     10.5        5
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6
```

```
Shape of the DataFrame:
 (1599, 12)
Info of the DataFrame:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
Description of the DataFrame:
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

```
Unique values:
 fixed acidity            96
volatile acidity        143
citric acid              80
residual sugar           91
chlorides               153
free sulfur dioxide      60
total sulfur dioxide    144
density                 436
pH                       89
sulphates                96
alcohol                  65
quality                   6
dtype: int64
Value counts:
 quality
5    681
6    638
7    199
4     53
8     18
3     10
Name: count, dtype: int64
```

Step 2: Preprocessing

- Performed data cleaning by handling missing values and normalizing the data.

Code

```
# Step 3: Preprocessing
# Removing missing values
data = data.dropna()

# Removing duplicates
data = data.drop_duplicates()

# Removing outliers (for simplicity, we'll use Z-score method to remove outliers)
def remove_outliers(df):
    z_scores = np.abs((df - df.mean()) / df.std())
    return df[(z_scores < 3).all(axis=1)]

data = remove_outliers(data)
```

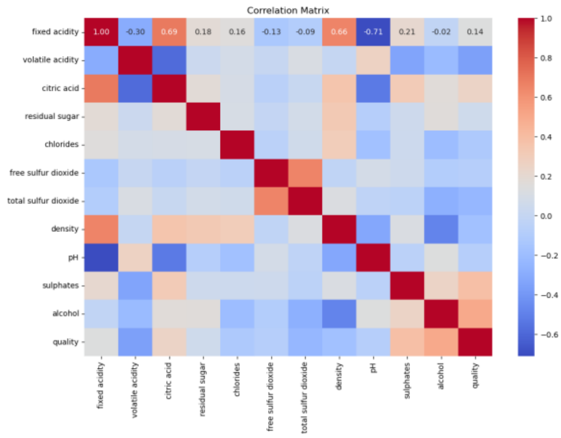
Step 3: Feature Selection

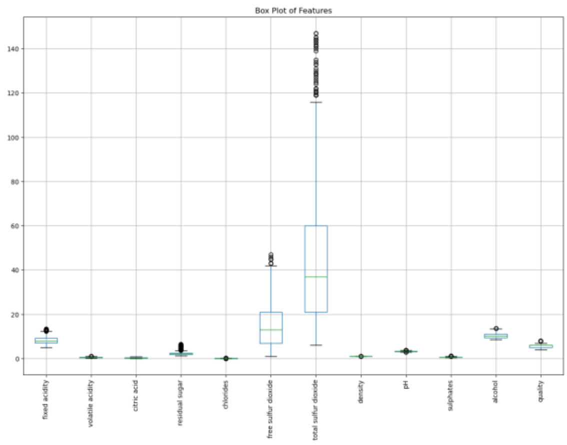
- Selected important features using correlation analysis to identify which features significantly impact the wine quality.

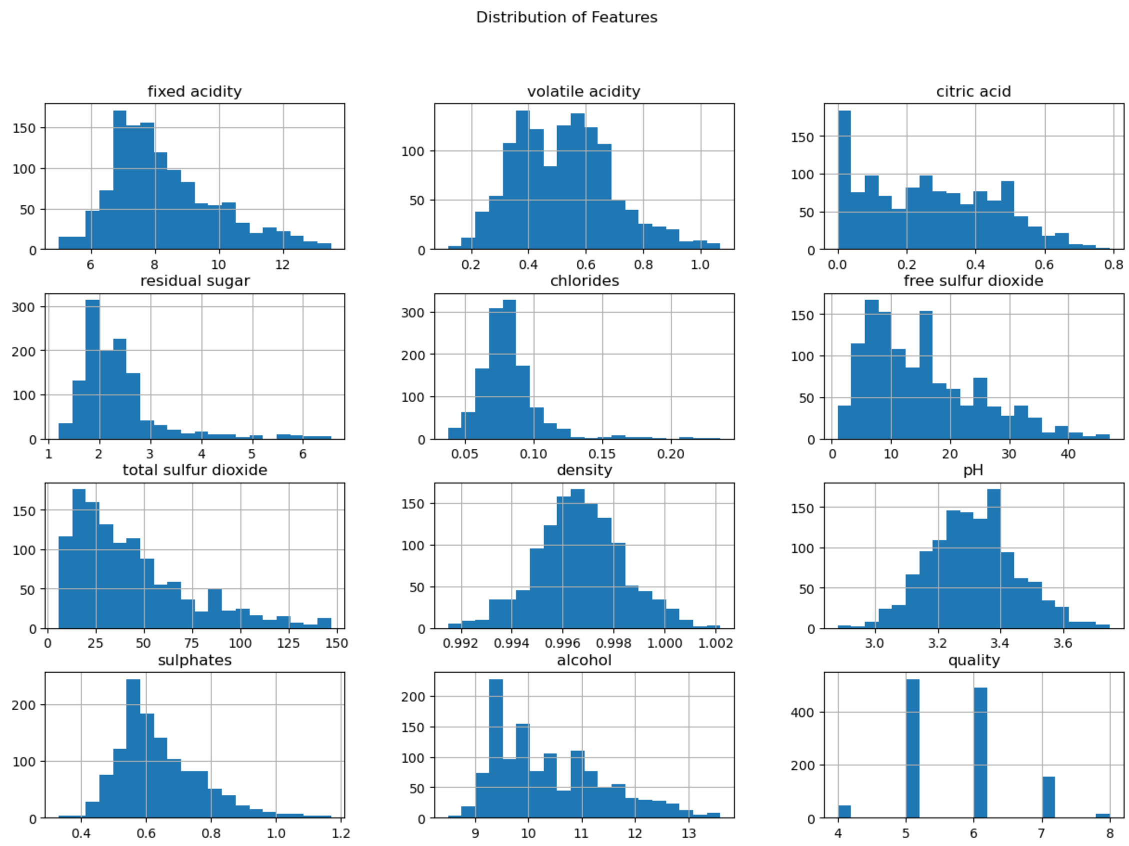
Code

```
# Step 4: Feature Engineering

# Standardizing the features manually
def standardize(column):
    return (column - column.mean()) / column.std()

# Apply standardization
X = data.drop('quality', axis=1).apply(standardize)
y = data['quality']

# Correlation Matrix
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()

# Box Plot for each feature
plt.figure(figsize=(15, 10))
data.boxplot()
plt.title('Box Plot of Features')
plt.xticks(rotation=90)
plt.show()

# Distribution Plot for each feature
data.hist(bins=20, figsize=(15, 10))
plt.suptitle('Distribution of Features')
plt.show()

# Feature Selection (Keep features with correlation greater than 0.1 with 'quality')
correlation_matrix = data.corr()
correlation_with_quality = correlation_matrix['quality'].abs().sort_values(ascending=False)
selected_features = correlation_with_quality[correlation_with_quality > 0.1].index.tolist()
selected_features.remove('quality')

# Update X to only include selected features
X_selected = X[selected_features]
```

Screenshots







Step 4: Implementation of Model and Training the Model

- Chose a Linear Regression model for predicting wine quality.
- Trained the model on the training set and made predictions on the testing set.

Code

```
# Step 5: Implementation of Model
model = LinearRegression()
```

```
# Step 6: Training and Testing the Model
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Test the model
y_pred = model.predict(X_test)
```

Step 5: Evaluation

- Evaluated the model's performance using metrics such as Mean Squared Error (MSE), R-squared (R^2), Accuracy, Precision, Recall, and F1 Score.

Code

```python
# Step 7: Evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Converting regression results to classification for quality prediction
y_pred_class = np.round(y_pred).astype(int)

# Ensure predictions are within valid range
y_pred_class = np.clip(y_pred_class, y_test.min(), y_test.max())

accuracy = accuracy_score(y_test, y_pred_class)
precision = precision_score(y_test, y_pred_class, average='weighted', zero_division=1)
recall = recall_score(y_test, y_pred_class, average='weighted', zero_division=1)
f1 = f1_score(y_test, y_pred_class, average='weighted', zero_division=1)

print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

Screenshots

```
Mean Squared Error: 0.41704055885126934
R^2 Score: 0.3839412238509179
Accuracy: 0.6032388663967612
Precision: 0.6257353195612305
Recall: 0.6032388663967612
F1 Score: 0.5706182088478641
```

Results and Comments:

Results

Mean Squared Error (MSE): 0.41704055885126934
This metric indicates the average squared difference between the predicted and actual wine quality. A lower MSE indicates better model performance.

R^2 score: 0.3839412238509179
This metric indicates the proportion of variance in the dependent variable that is predictable from the independent variables. An R^2 score of 0.38 suggests that the model explains about 38% of the variance in wine quality.

Accuracy: 0.6032388663967612
This metric indicates the proportion of correctly predicted wine quality labels. An accuracy of 60.3% means the model correctly predicts wine quality about 60% of the time.

Precision: 0.6257353195612305
This metric indicates the proportion of true positive predictions among all positive predictions. A precision of 62.6% suggests that when the model predicts a certain quality, it is correct about 62.6% of the time.

Recall: 0.6032388663967612
This metric indicates the proportion of true positive predictions among all actual positives. A recall of 60.3% means the model captures about 60.3% of the actual positive cases.

F1 Score: 0.5706182088478641
This metric is the harmonic mean of precision and recall. An F1 score of 57.1% suggests a balance between precision and recall.

Comments

The model demonstrates a moderate ability to predict wine quality, with an accuracy of approximately 60% and an F1 score of 57.1%.

The R^2 score indicates that there is room for improvement in the model's ability to explain the variance in wine quality.

The precision and recall values suggest that the model has a slightly better performance in identifying positive cases, but there is still a significant proportion of false positives and false negatives.