Load **Dataset**

```
import pandas as pd

data = pd.read_csv('WineQT.csv')
```

Explore **Dataset**

```
print("Dataset Info:")
print(data.info())
print("\nFirst 5 Rows:")
print(data.head())
print("\nDescriptive Statistics:")
print(data.describe())
```

```
 11  quality               1143 non-null   int64
 12  Id                    1143 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 116.2 KB
None

First 5 Rows:
      s  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  Id
0      9.4        5   0
```

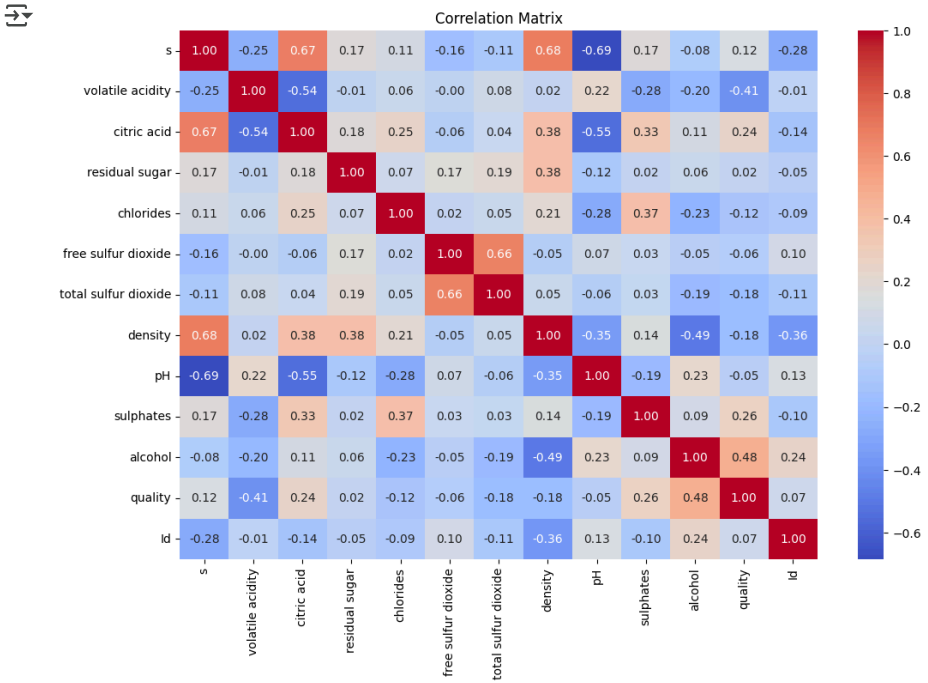
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	0.086933	15.615486	45.914698	0.996730	
std	0.047267	10.250486	32.782130	0.001925	
min	0.012000	1.000000	6.000000	0.990070	
25%	0.070000	7.000000	21.000000	0.995570	
50%	0.079000	13.000000	37.000000	0.996680	
75%	0.090000	21.000000	61.000000	0.997845	
max	0.611000	68.000000	289.000000	1.003690	

	pH	sulphates	alcohol	quality	Id
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	3.311015	0.657708	10.442111	5.657043	804.969379
std	0.156664	0.170399	1.082196	0.805824	463.997116
min	2.740000	0.330000	8.400000	3.000000	0.000000
25%	3.205000	0.550000	9.500000	5.000000	411.000000
50%	3.310000	0.620000	10.200000	6.000000	794.000000
75%	3.400000	0.730000	11.100000	6.000000	1209.500000
max	4.010000	2.000000	14.900000	8.000000	1597.000000

Distribution of Target Variable ('quality')

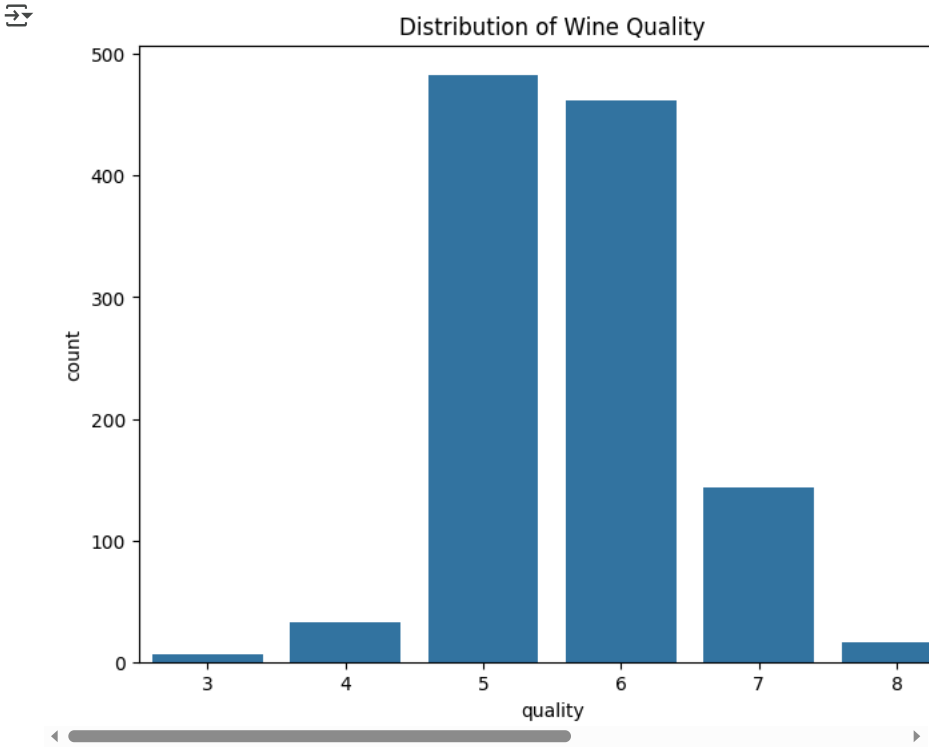
```
import matplotlib.pyplot as plt
import seaborn as sns

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



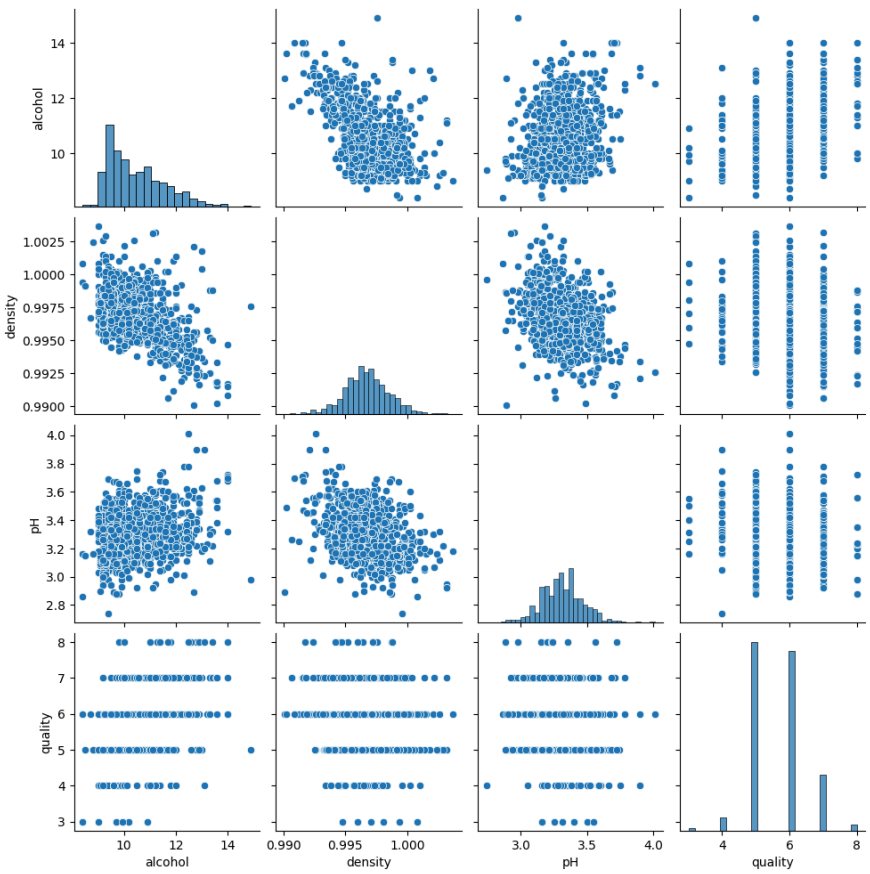
Distribution of Target Variable '(quality')

```
plt.figure(figsize=(8, 6))
sns.countplot(x='quality', data=data)
plt.title("Distribution of Wine Quality")
plt.show()
```



Pair **Plot**

```
sns.pairplot(data[['alcohol', 'density', 'pH', 'quality']])
plt.show()
```



Preprocessing Separate Features and Target

```
X = data.drop(columns=['quality'])
y = data['quality']
```

Train-Test **Split**

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
```

Feature **Scaling**

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model Building and Evaluation **bold text** Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_scaled, y_train)
y_pred_rf = rf.predict(X_test_scaled)
print("\nRandom Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.6899563318777293
Classification Report:
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.73      0.75      0.74        96
           6       0.64      0.71      0.67        99
           7       0.76      0.62      0.68        26
           8       0.00      0.00      0.00         2

    accuracy                           0.69       229
   macro avg       0.43      0.41      0.42       229
weighted avg       0.67      0.69      0.68       229

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Un
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Un
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Un
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Stochastic Gradient Descent **Classifier**

```
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_scaled, y_train)
y_pred_rf = rf.predict(X_test_scaled)
print("\nRandom Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))

sgd = SGDClassifier(random_state=42)
sgd.fit(X_train_scaled, y_train)
y_pred_sgd = sgd.predict(X_test_scaled)
print("\nSGD Accuracy:", accuracy_score(y_test, y_pred_sgd))
print("Classification Report:\n", classification_report(y_test, y_pred_sgd))
```

```
Random Forest Accuracy: 0.6899563318777293
Classification Report:
               precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.73      0.75      0.74        96
           6       0.64      0.71      0.67        99
           7       0.76      0.62      0.68        26
           8       0.00      0.00      0.00         2

    accuracy                           0.69       229
   macro avg       0.43      0.41      0.42       229
weighted avg       0.67      0.69      0.68       229


SGD Accuracy: 0.5851528384279476
Classification Report:
               precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.68      0.78      0.73        96
           6       0.66      0.39      0.49        99
           7       0.34      0.77      0.47        26
           8       0.00      0.00      0.00         2

    accuracy                           0.59       229
   macro avg       0.34      0.39      0.34       229
weighted avg       0.61      0.59      0.57       229

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: U
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: U
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: U
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: U
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: U
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: U
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Support Vector **Classifier**

```
from sklearn.svm import SVC # Import the SVC class
from sklearn.metrics import accuracy_score, classification_report

svc = SVC(random_state=42)
svc.fit(X_train_scaled, y_train)
y_pred_svc = svc.predict(X_test_scaled)
print("\nSVC Accuracy:", accuracy_score(y_test, y_pred_svc))
print("Classification Report:\n", classification_report(y_test, y_pred_svc))
```

```
SVC Accuracy: 0.6550218340611353
Classification Report:
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.70      0.75      0.72        96
           6       0.61      0.70      0.65        99
           7       0.69      0.35      0.46        26
           8       0.00      0.00      0.00         2

    accuracy                           0.66       229
   macro avg       0.40      0.36      0.37       229
weighted avg       0.64      0.66      0.64       229

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Ur
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Ur
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Ur
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Hyperparameter Tuning **Random** Forest Example

```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}
```

Best Parameters for Random Forest and **Best** Cross-Validation Score

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5,
```

```
grid_search.fit(X_train_scaled, y_train)
print("\nBest Parameters for Random Forest:", grid_search.best_params_)
print("Best Cross-Validation Score:", grid_search.best_score_)
```

```
Best Parameters for Random Forest: {'max_depth': 20, 'min_samples_split': 2, 'n_est
Best Cross-Validation Score: 0.664042514862187
```

Evaluate Best **Model**

```
best_rf = grid_search.best_estimator_
y_pred_best = best_rf.predict(X_test_scaled)
print("\nBest Random Forest Accuracy:", accuracy_score(y_test, y_pred_best))
print("Classification Report:\n", classification_report(y_test, y_pred_best))
```

```
Best Random Forest Accuracy: 0.6899563318777293
Classification Report:
              precision    recall  f1-score   support

           4       0.00      0.00      0.00         6
           5       0.73      0.75      0.74        96
           6       0.64      0.71      0.67        99
           7       0.76      0.62      0.68        26
           8       0.00      0.00      0.00         2

    accuracy                           0.69       229
   macro avg       0.43      0.41      0.42       229
weighted avg       0.67      0.69      0.68       229

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Un
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Un
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: Un
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Confusion Matrix **Visualization**

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix # Import confusion_matrix
```