

Wine Quality Machine Learning Project

Student Name: Hassan Saud Al Soleiss

Dataset: Wine Quality Dataset (WineQT.csv)

Objective: Apply machine learning models with GridSearchCV and feature selection methods to predict wine quality.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Load the Wine Quality dataset

```
df = pd.read_csv("WineQT.csv")
```

Display the first 5 rows

```
df.head()
```

```
In [12]: # Check dataset information
print("Dataset Shape:", df.shape)

print("\nMissing Values:")
print(df.isnull().sum())

print("\nDataset Information:")
df.info()
```

Dataset Shape: (1143, 13)

Missing Values:

```
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
Id                 0
dtype: int64
```

Dataset Information:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1143 entries, 0 to 1142

Data columns (total 13 columns):

#	Column	Non-Null Count	Dtype
0	fixed acidity	1143 non-null	float64
1	volatile acidity	1143 non-null	float64
2	citric acid	1143 non-null	float64
3	residual sugar	1143 non-null	float64
4	chlorides	1143 non-null	float64
5	free sulfur dioxide	1143 non-null	float64
6	total sulfur dioxide	1143 non-null	float64
7	density	1143 non-null	float64
8	pH	1143 non-null	float64
9	sulphates	1143 non-null	float64
10	alcohol	1143 non-null	float64
11	quality	1143 non-null	int64
12	Id	1143 non-null	int64

dtypes: float64(11), int64(2)

memory usage: 116.2 KB

```
In [13]: # Prepare features and target

# Remove Id because it is only an identifier
X = df.drop(columns=["quality", "Id"])

# Target variable
y = df["quality"]

print("Features shape:", X.shape)
print("Target shape:", y.shape)

print("\nFeatures used:")
print(X.columns.tolist())

print("\nQuality classes:")
print(sorted(y.unique()))
```

Features shape: (1143, 11)

Target shape: (1143,)

Features used:

```
['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides',
'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']
```

Quality classes:

```
[np.int64(3), np.int64(4), np.int64(5), np.int64(6), np.int64(7), np.int64(8)]
```

In [14]: `from sklearn.model_selection import train_test_split`

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.20,
    random_state=42,
    stratify=y
)

print("Training data:", X_train.shape)
print("Testing data:", X_test.shape)
```

Training data: (914, 11)

Testing data: (229, 11)

In [15]: `from sklearn.ensemble import RandomForestClassifier`
`from sklearn.model_selection import GridSearchCV`
`from sklearn.metrics import accuracy_score, classification_report`

```
# Create Random Forest model
rf = RandomForestClassifier(random_state=42)

# Parameters for Grid Search
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}

# Apply GridSearchCV
grid_rf = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

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

# Best model
best_rf = grid_rf.best_estimator_
```

```

# Prediction
y_pred_rf = best_rf.predict(X_test)

# Results
print("Best Parameters:", grid_rf.best_params_)
print("Best Cross-Validation Accuracy:", grid_rf.best_score_)
print("Test Accuracy:", accuracy_score(y_test, y_pred_rf))

print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf, zero_division=0))

```

Best Parameters: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 200}

Best Cross-Validation Accuracy: 0.6531675974298925

Test Accuracy: 0.7161572052401747

Classification Report:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	7
5	0.74	0.85	0.79	97
6	0.69	0.74	0.71	92
7	0.74	0.48	0.58	29
8	0.00	0.00	0.00	3
accuracy			0.72	229
macro avg	0.36	0.34	0.35	229
weighted avg	0.68	0.72	0.69	229

```

In [16]: from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

# Create a pipeline for scaling and SVM
svm_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svm', SVC())
])

# Parameters for Grid Search
param_grid_svm = {
    'svm__C': [0.1, 1, 10],
    'svm__kernel': ['linear', 'rbf'],
    'svm__gamma': ['scale', 'auto']
}

# Apply GridSearchCV
grid_svm = GridSearchCV(
    estimator=svm_pipeline,
    param_grid=param_grid_svm,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

```

```

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

# Prediction
y_pred_svm = grid_svm.predict(X_test)

# Results
print("Best Parameters:", grid_svm.best_params_)
print("Best Cross-Validation Accuracy:", grid_svm.best_score_)
print("Test Accuracy:", accuracy_score(y_test, y_pred_svm))

print("\nClassification Report:")
print(classification_report(y_test, y_pred_svm, zero_division=0))

```

Best Parameters: {'svm__C': 10, 'svm__gamma': 'scale', 'svm__kernel': 'rbf'}
 Best Cross-Validation Accuracy: 0.6171020236594007
 Test Accuracy: 0.6419213973799127

Classification Report:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	7
5	0.70	0.73	0.71	97
6	0.59	0.68	0.64	92
7	0.62	0.45	0.52	29
8	0.00	0.00	0.00	3
accuracy			0.64	229
macro avg	0.32	0.31	0.31	229
weighted avg	0.61	0.64	0.62	229

In [17]: `from sklearn.feature_selection import SelectKBest, f_classif`

```

# Feature Selection Method 1: SelectKBest
selector = SelectKBest(score_func=f_classif, k=6)

# Fit only on training data
selector.fit(X_train, y_train)

# Get selected features
selected_features_kbest = X.columns[selector.get_support()]

print("Selected Features using SelectKBest:")
for feature in selected_features_kbest:
    print(feature)

```

Selected Features using SelectKBest:
 volatile acidity
 citric acid
 total sulfur dioxide
 density
 sulphates
 alcohol

```
In [18]: # Feature Selection Method 2: Random Forest Feature Importance

# Get feature importance from the best Random Forest model
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': best_rf.feature_importances_
})

# Sort features by importance
feature_importance = feature_importance.sort_values(
    by='Importance',
    ascending=False
)

print("Feature Importance using Random Forest:")
print(feature_importance)

# Select the top 6 features
selected_features_rf = feature_importance.head(6)['Feature'].tolist()

print("\nTop 6 Selected Features:")
for feature in selected_features_rf:
    print(feature)
```

Feature Importance using Random Forest:

	Feature	Importance
10	alcohol	0.146657
9	sulphates	0.116011
1	volatile acidity	0.108269
6	total sulfur dioxide	0.097907
7	density	0.087695
8	pH	0.079013
4	chlorides	0.078817
2	citric acid	0.077523
0	fixed acidity	0.077157
3	residual sugar	0.066042
5	free sulfur dioxide	0.064907

Top 6 Selected Features:

alcohol
sulphates
volatile acidity
total sulfur dioxide
density
pH

```
In [19]: # Train Random Forest using selected features

X_train_selected = X_train[selected_features_rf]
X_test_selected = X_test[selected_features_rf]

# Grid Search with selected features
grid_rf_selected = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    cv=5,
```

```

        scoring='accuracy',
        n_jobs=-1
    )

    # Train
    grid_rf_selected.fit(X_train_selected, y_train)

    # Predict
    y_pred_rf_selected = grid_rf_selected.predict(X_test_selected)

    # Results
    rf_selected_accuracy = accuracy_score(y_test, y_pred_rf_selected)

    print("Selected Features:")
    print(selected_features_rf)

    print("\nBest Parameters:", grid_rf_selected.best_params_)
    print("Test Accuracy Before Feature Selection:",
          accuracy_score(y_test, y_pred_rf))

    print("Test Accuracy After Feature Selection:",
          rf_selected_accuracy)

    print("\nClassification Report:")
    print(classification_report(
        y_test,
        y_pred_rf_selected,
        zero_division=0
    ))

```

Selected Features:

['alcohol', 'sulphates', 'volatile acidity', 'total sulfur dioxide', 'density', 'pH']

Best Parameters: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 100}

Test Accuracy Before Feature Selection: 0.7161572052401747

Test Accuracy After Feature Selection: 0.6812227074235808

Classification Report:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	7
5	0.71	0.81	0.76	97
6	0.64	0.71	0.67	92
7	0.75	0.41	0.53	29
8	0.00	0.00	0.00	3
accuracy			0.68	229
macro avg	0.35	0.32	0.33	229
weighted avg	0.65	0.68	0.66	229

In [20]: *# Train Random Forest using SelectKBest features*

```
X_train_kbest = X_train[selected_features_kbest]
```

```
X_test_kbest = X_test[selected_features_kbest]

grid_rf_kbest = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

# Train the model
grid_rf_kbest.fit(X_train_kbest, y_train)

# Prediction
y_pred_rf_kbest = grid_rf_kbest.predict(X_test_kbest)

# Accuracy
rf_kbest_accuracy = accuracy_score(y_test, y_pred_rf_kbest)

print("SelectKBest Features:")
print(list(selected_features_kbest))

print("\nBest Parameters:", grid_rf_kbest.best_params_)

print("Original Random Forest Accuracy:",
      accuracy_score(y_test, y_pred_rf))

print("Random Forest Importance Accuracy:",
      rf_selected_accuracy)

print("SelectKBest Accuracy:",
      rf_kbest_accuracy)

print("\nClassification Report:")
print(classification_report(
    y_test,
    y_pred_rf_kbest,
    zero_division=0
))
```

SelectKBest Features:

['volatile acidity', 'citric acid', 'total sulfur dioxide', 'density', 'sulphates', 'alcohol']

Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}

Original Random Forest Accuracy: 0.7161572052401747

Random Forest Importance Accuracy: 0.6812227074235808

SelectKBest Accuracy: 0.6637554585152838

Classification Report:

	precision	recall	f1-score	support
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	7
5	0.70	0.79	0.74	97
6	0.63	0.66	0.65	92
7	0.64	0.48	0.55	29
8	0.00	0.00	0.00	3
accuracy			0.66	229
macro avg	0.33	0.32	0.32	229
weighted avg	0.63	0.66	0.64	229

In [21]: *# Final comparison of model performance*

```
results = pd.DataFrame({
    'Model': [
        'Random Forest - All Features',
        'SVM - All Features',
        'Random Forest - RF Selected Features',
        'Random Forest - SelectKBest Features'
    ],
    'Accuracy': [
        accuracy_score(y_test, y_pred_rf),
        accuracy_score(y_test, y_pred_svm),
        rf_selected_accuracy,
        rf_kbest_accuracy
    ]
})

results['Accuracy (%)'] = (results['Accuracy'] * 100).round(2)

print("Final Model Comparison:")
display(results[['Model', 'Accuracy (%)']])
```

Final Model Comparison:

	Model	Accuracy (%)
0	Random Forest - All Features	71.62
1	SVM - All Features	64.19
2	Random Forest - RF Selected Features	68.12
3	Random Forest - SelectKBest Features	66.38

Conclusion

In this project, the Wine Quality dataset was prepared and analyzed using machine learning techniques. The dataset contained no missing values.

Random Forest and SVM models were optimized using GridSearchCV. Two feature selection methods were also applied: SelectKBest and Random Forest Feature Importance.

The Random Forest model using all features achieved the highest test accuracy of 71.62%. After feature selection, the accuracy decreased to 68.12% using Random Forest Feature Importance and 66.38% using SelectKBest.

Therefore, the results show that using all available features produced the best performance for this dataset.

In []:

In []: