Assignment-1: Wine Quality Classification using SVM

Vikram Balai 2201CS76

19 September 2025

Contents

1 Introduction

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for **classification and regression** tasks. It works by finding the optimal hyperplane that separates data points of different classes with the maximum margin.

In this project, SVM is applied to classify **wine quality** based on physicochemical properties of red and white wines.

2 Dataset

The Wine Quality Dataset is taken from the **UCI Machine Learning Repository**.

- Red wine: `https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv`
- White wine: `https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv`

2.1 Features

Feature	**Description**
fixed acidity	Amount of non-volatile acids
volatile acidity	Amount of acetic acid
citric acid	Concentration of citric acid
residual sugar	Sugar left after fermentation
chlorides	Salt content
free sulfur dioxide	Free SO_2 concentration
total sulfur dioxide	Total SO_2 concentration
density	Density of the wine
pH	Acidity level
sulphates	Potassium sulphate level
alcohol	Alcohol percentage
quality	Wine quality score (0–10)
type	Red / White

Table 1: Wine Quality Dataset Features

Note: There are no missing values in the dataset.

3 Code

```
# -*- coding: utf-8 -*-
"""Apr_Assignment.ipynb

Automatically generated by Colab.

Original file is located at
```

```
    https://colab.research.google.com/drive/1MJVqy8W4hH67h_DzmMlv0Kq8x5HXbtti

#IMPORTS
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Make plots pretty
plt.style.use("seaborn-v0_8")
sns.set_palette("Set2")

"""#Dataset"""

_DOWNLOAD_URL_WHITE_WINES = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv"
_DOWNLOAD_URL_RED_WINES = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
red_wine = pd.read_csv(_DOWNLOAD_URL_RED_WINES, sep=";")
white_wine = pd.read_csv(_DOWNLOAD_URL_WHITE_WINES, sep=";")
# Add a column to identify type
red_wine["type"] = 1
white_wine["type"] = 0

# Combine datasets
wine = pd.concat([red_wine, white_wine], ignore_index=True)
print("Shape of dataset:", wine.shape)
print("\nColumn names:", wine.columns.tolist())

"""#SVM Object

"""

# -----------------------------
# 1. Imports for modeling
# -----------------------------
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# -----------------------------
# 2. Prepare Data
# -----------------------------
X = wine.drop(columns=["quality"])  # features (remove target + type)
y = wine["quality"]                         # target = quality

# Optionally: collapse into binary classification (Good vs Bad wine)
# Uncomment if you want binary classification instead of multiclass
# y = y.apply(lambda q: 1 if q >= 6 else 0)  # 1=Good (>=6), 0=Bad (<6)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

```

# # Standardize features (important for SVM!)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# ----------------------------
# 3. Create SVM Object & Train
# ----------------------------
svm_clf = SVC(kernel="rbf", C=10, gamma="scale")  # SVM object
svm_clf.fit(X_train, y_train)

"""Prediction and result"""

# ----------------------------
# 4. Predictions & Evaluation
# ----------------------------
y_pred = svm_clf.predict(X_test)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()

# Print classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Listing 1: Code

4 Algorithm

We trained a **Support Vector Classifier (SVC)** with the following setup:

```
from sklearn.svm import SVC

svm_clf = SVC(kernel="rbf", C=10, gamma="scale")
svm_clf.fit(X_train, y_train)
```

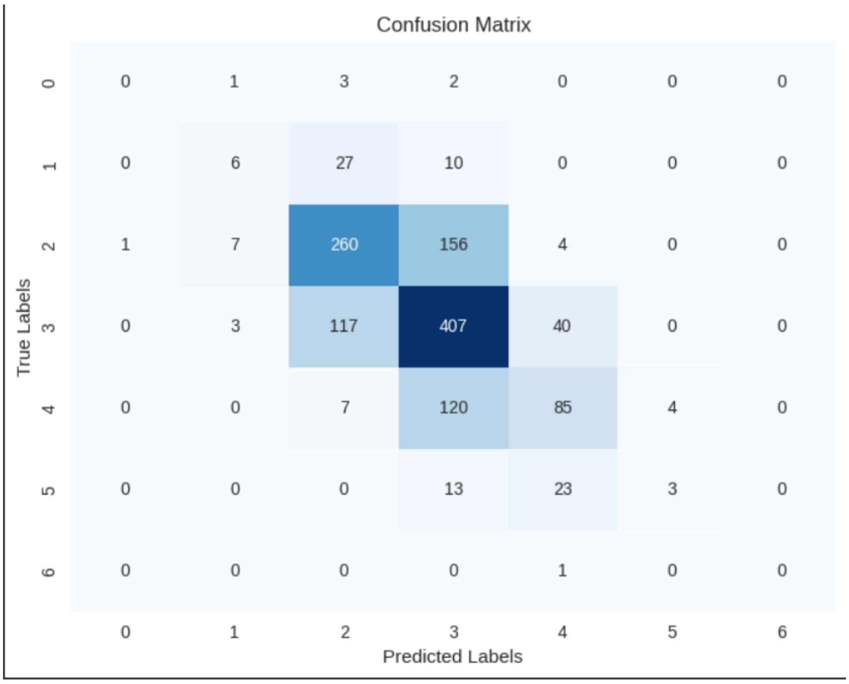
Listing 2: Training SVM Classifier

- Kernel = RBF → allows non-linear classification
- C = 10 → trade-off between margin maximization and classification error
- gamma = ”scale” → influence of a single training example

Dataset was split into **train (80%) and test (20%)** sets with stratification, and features were standardized using `StandardScaler`.

5 Results

The model was evaluated using a confusion matrix and classification report.



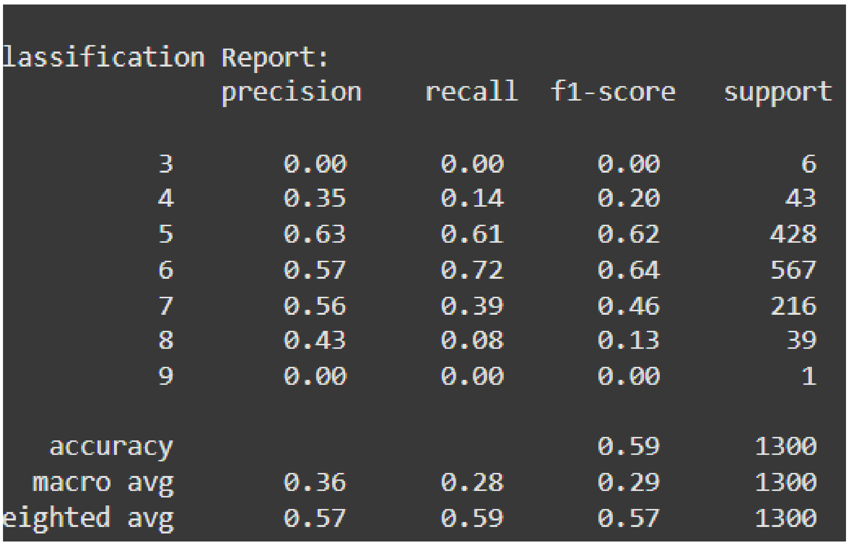
	0	1	2	3	4	5	6
0	0	1	3	2	0	0	0
1	0	6	27	10	0	0	0
2	1	7	260	156	4	0	0
3	0	3	117	407	40	0	0
4	0	0	7	120	85	4	0
5	0	0	0	13	23	3	0
6	0	0	0	0	1	0	0

Figure 1: Confusion Matrix for SVM Classifier

```python
from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from_estimator(svm_clf, X_test, y_test, cmap="Blues")
```

Listing 3: Confusion Matrix Visualization



```
lassification Report:
             precision    recall  f1-score   support

           3       0.00      0.00      0.00         6
           4       0.35      0.14      0.20        43
           5       0.63      0.61      0.62       428
           6       0.57      0.72      0.64       567
           7       0.56      0.39      0.46       216
           8       0.43      0.08      0.13        39
           9       0.00      0.00      0.00         1

    accuracy                           0.59      1300
   macro avg       0.36      0.28      0.29      1300
eighted avg       0.57      0.59      0.57      1300
```

Figure 2: Classification Report Screenshot Placeholder

6 Project Structure

```
Apr_Assignment.ipynb   # Main notebook with code
README.md              # Project description
data/                  # (optional) raw datasets
```

7 How to Run

1. Clone the repository:

   ```
   git clone https://github.com/RenderHaven/SVM_WineQuality.git
   cd your-repo
   ```

2. Install dependencies:

   ```
   pip install -r requirements.txt
   ```

3. Open and run the notebook:

   ```
   jupyter notebook Apr_Assignment.ipynb
   ```

8 Conclusion

- SVM was successfully applied to classify wine quality.
- The model performs reasonably well but shows class imbalance issues.
- Future improvements could include class balancing, hyperparameter tuning, or testing other algorithms like Random Forest or XGBoost.