Machine Learning Tutorial Using Support Vector Machines (SVM) for Wine Classification

Name: **Austine Igho Efenarua**

Student ID: **23096669**

GitHub Links**:**

Click here for the Code

Click here for the PDF

Abstract

This tutorial demonstrates how Support Vector Machines (SVM) can be used for wine classification by applying different kernel functions—linear, polynomial, and RBF. It covers data preprocessing, model training, evaluation, and decision boundary visualization, providing insights into the strengths of each kernel in classifying wines based on chemical properties. The tutorial highlights the impact of kernel selection on model performance and offers practical guidance for wine quality prediction.

1.0 Introduction

Support Vector Machines (SVM) are widely used for classification tasks, particularly when the data is high-dimensional and complex. In this tutorial, we will explore the use of SVM to classify wines based on their chemical properties using a dataset that includes various features of wines such as acidity, alcohol content, and sulfur levels. We will focus on how different kernel functions (linear, polynomial, and RBF) affect the performance of the SVM model and how to apply this technique to improve classification accuracy.

1.1 Literature Review

The studies reviewed contribute significantly to the field of wine quality prediction through machine learning. Adithya (2024) emphasizes the effectiveness of SVM as a powerful tool for wine quality classification, especially when kernel methods are fine-tuned. He (2024) explores the role of PCA and hyperparameter optimization in enhancing the performance of SVM and KNN, making it clear that preprocessing and parameter tuning are essential for achieving high accuracy. The third study on HHO-SVM (2024) introduces an advanced optimization technique, demonstrating how combining SVM with Harris Hawks Optimization can improve model performance and classification accuracy. Together, these studies highlight the growing potential of machine learning algorithms and optimization techniques in the wine industry, providing valuable insights for future research and practical applications in wine quality prediction.

2.0 Tutorial Steps

This tutorial walks through the following steps:

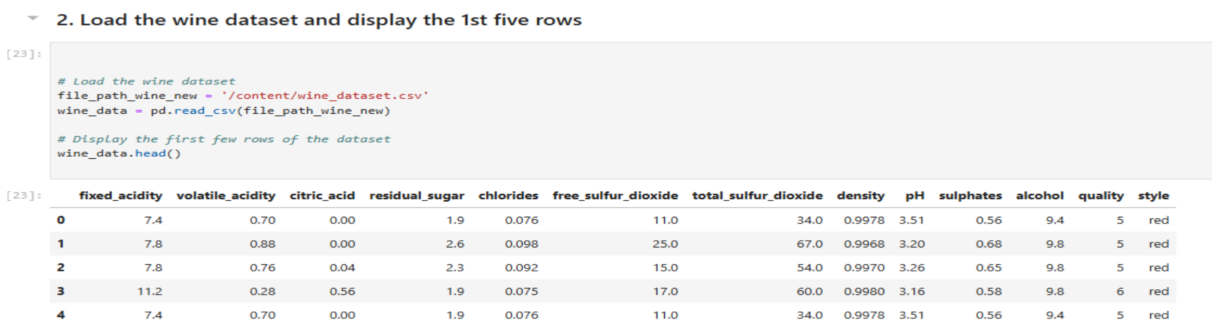
1. **Data Preprocessing** - Loading the dataset and preparing it for training.
2. **Model Training** - Implementing SVM with different kernels.
3. **Model Evaluation** - Comparing model performance using accuracy.
4. **Visualization** - Visualizing the decision boundaries for different kernels.

2.1 Data Preprocessing

Before building a machine learning model, it is important to preprocess the data. The first step is to load the wine dataset, check its structure, and scale the features to ensure that the SVM model works optimally.

2.1.1 Loading the Wine Dataset

The wine dataset contains various chemical properties of wines, such as alcohol content, acidity, and pH levels, with the target variable being the quality of the wine.



2. Load the wine dataset and display the 1st five rows

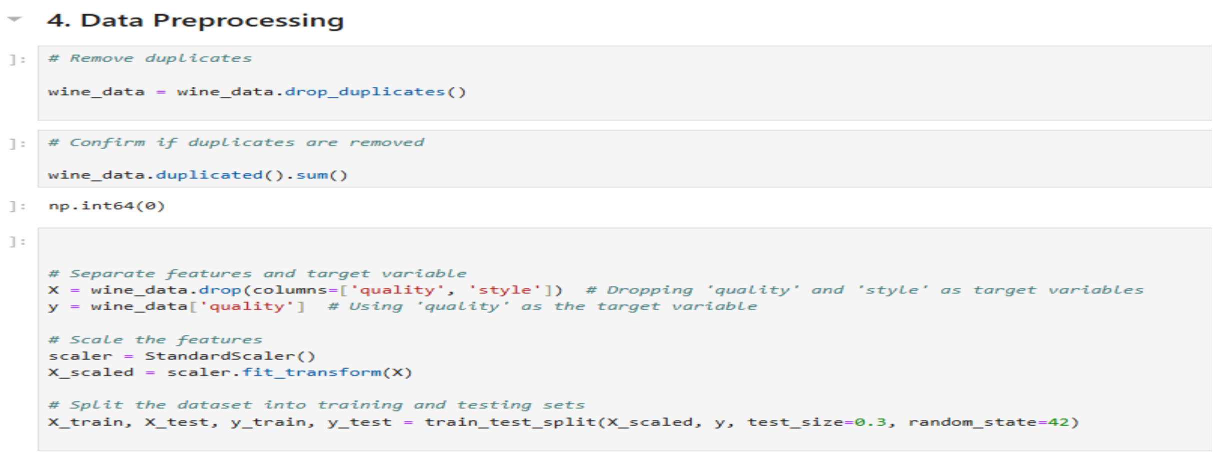
[23]:
```
# Load the wine dataset
file_path_wine_new = '/content/wine_dataset.csv'
wine_data = pd.read_csv(file_path_wine_new)

# Display the first few rows of the dataset
wine_data.head()
```

[23]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality	style
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	red

2.1.2 Data Preprocessing

For SVM models to work effectively, I need to remove duplicates, scale the data. We will also separate the features (X) and the target variable (y), which in this case is the wine quality.



4. Data Preprocessing

]:
```
# Remove duplicates

wine_data = wine_data.drop_duplicates()
```

]:
```
# Confirm if duplicates are removed

wine_data.duplicated().sum()
```

]:
```
np.int64(0)
```

]:
```
# Separate features and target variable
X = wine_data.drop(columns=['quality', 'style'])  # Dropping 'quality' and 'style' as target variables
y = wine_data['quality']  # Using 'quality' as the target variable

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

2.1.3 Data Insights

The wine dataset consists of 13 features representing various chemical properties of wines, including acidity, alcohol content, sulfur levels, and pH. The target variable, quality, is an integer between 0 and 10, indicating the wine's quality.

2.2 Model Training with Different Kernels

Support Vector Machines can be customized with different types of kernels. In this tutorial, I will experiment with three types of kernels:

1. **Linear Kernel**
2. **Polynomial Kernel**
3. **Radial Basis Function (RBF) Kernel**

2.2.1 Linear Kernel

The linear kernel is best suited for linearly separable data, where the data points can be divided by a straight line or hyperplane.

2.2.2 Polynomial Kernel

The polynomial kernel is useful for data that is not linearly separable. It maps the data into a higher-dimensional space to allow for linear separation.

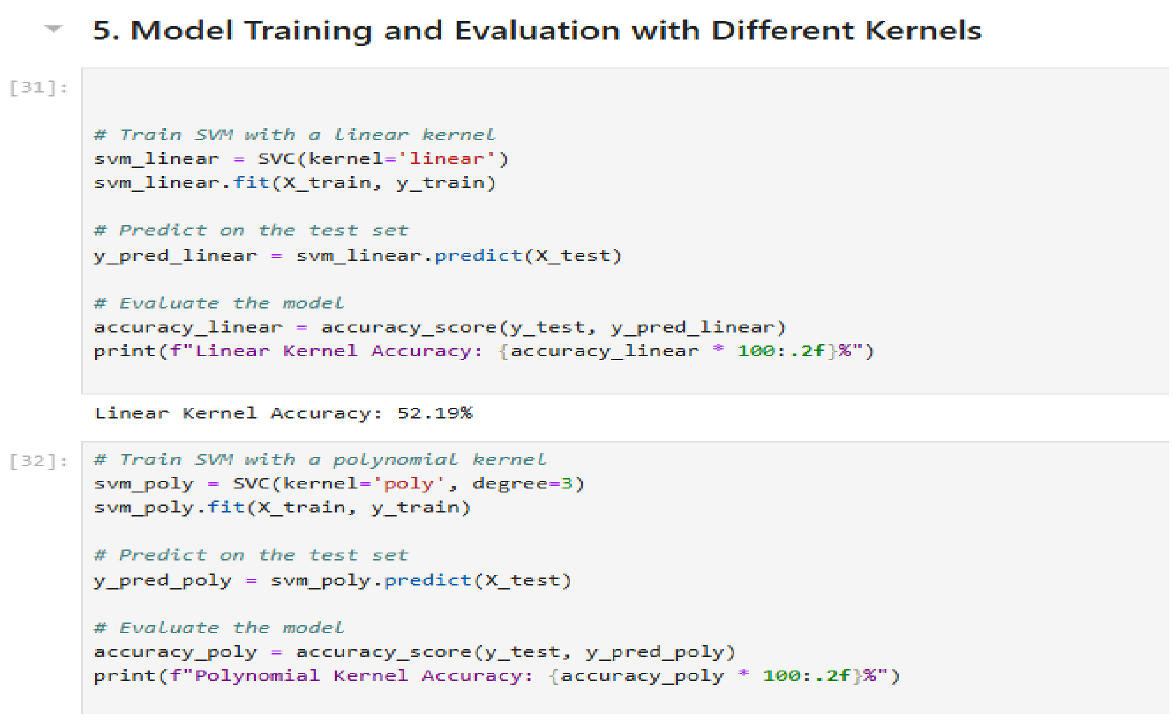
2.2.3 RBF Kernel

The RBF kernel is one of the most widely used kernels for SVM. It handles complex, non-linear relationships in the data and is particularly useful when data points cannot be linearly separated.

2.3. Model Evaluation

To compare the performance of the different models, I calculate the accuracy for each kernel. Accuracy is the percentage of correct predictions out of all predictions made by the model.

- **Linear Kernel**: Often works well for linearly separable data.
- **Polynomial Kernel**: Useful for data with non-linear relationships.
- **RBF Kernel**: Best for capturing complex, non-linear patterns in the data.

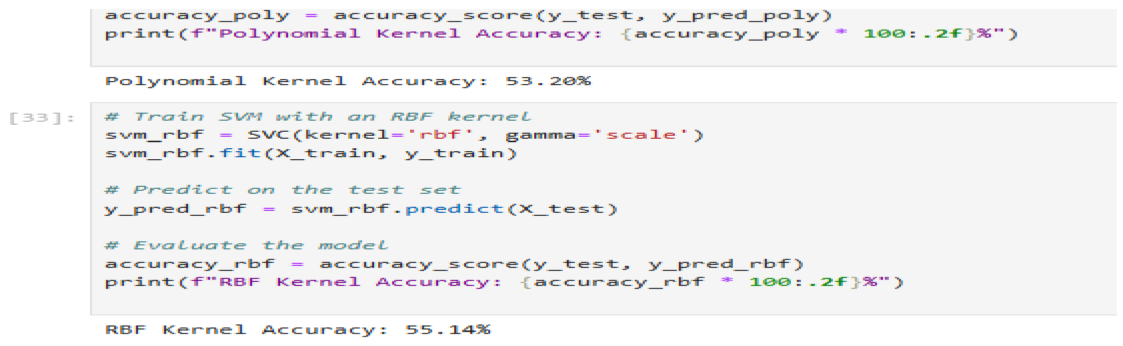


5. Model Training and Evaluation with Different Kernels

[31]:
```python
# Train SVM with a linear kernel
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)

# Predict on the test set
y_pred_linear = svm_linear.predict(X_test)

# Evaluate the model
accuracy_linear = accuracy_score(y_test, y_pred_linear)
print(f"Linear Kernel Accuracy: {accuracy_linear * 100:.2f}%")
```

```
Linear Kernel Accuracy: 52.19%
```

[32]:
```python
# Train SVM with a polynomial kernel
svm_poly = SVC(kernel='poly', degree=3)
svm_poly.fit(X_train, y_train)

# Predict on the test set
y_pred_poly = svm_poly.predict(X_test)

# Evaluate the model
accuracy_poly = accuracy_score(y_test, y_pred_poly)
print(f"Polynomial Kernel Accuracy: {accuracy_poly * 100:.2f}%")
```



```python
accuracy_poly = accuracy_score(y_test, y_pred_poly)
print(f"Polynomial Kernel Accuracy: {accuracy_poly * 100:.2f}%")
```

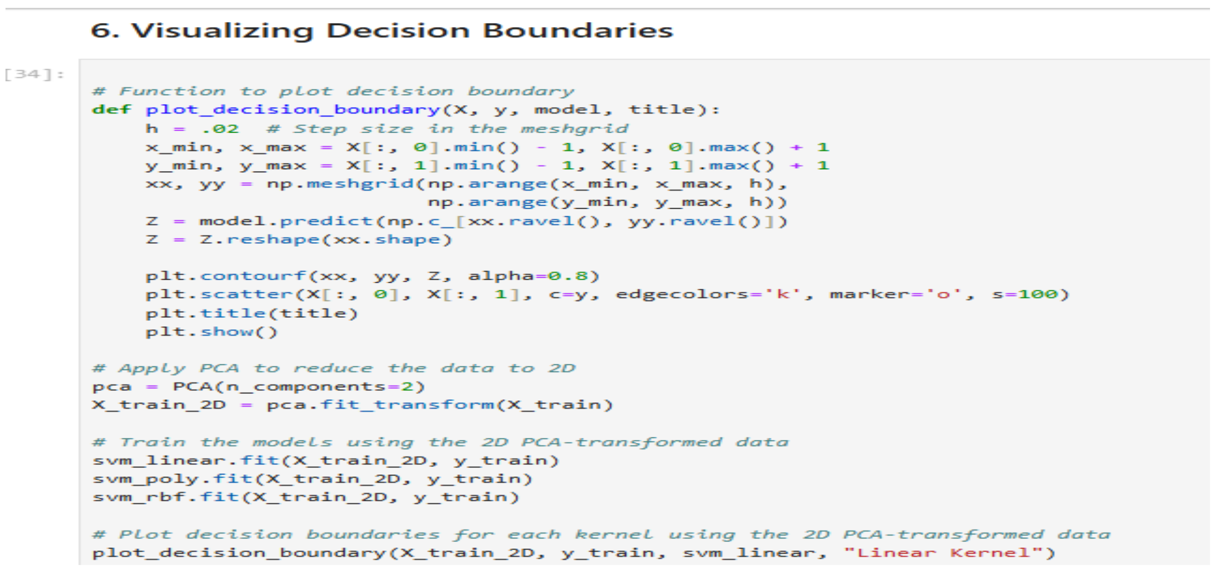
```
Polynomial Kernel Accuracy: 53.20%
```

[33]:
```python
# Train SVM with an RBF kernel
svm_rbf = SVC(kernel='rbf', gamma='scale')
svm_rbf.fit(X_train, y_train)

# Predict on the test set
y_pred_rbf = svm_rbf.predict(X_test)

# Evaluate the model
accuracy_rbf = accuracy_score(y_test, y_pred_rbf)
print(f"RBF Kernel Accuracy: {accuracy_rbf * 100:.2f}%")
```

```
RBF Kernel Accuracy: 55.14%
```

3.0 Visualizing Decision Boundaries

To understand how each kernel works, I can visualize the decision boundaries. Since the dataset is high-dimensional, we will use only the first two features for visualization purposes.



6. Visualizing Decision Boundaries

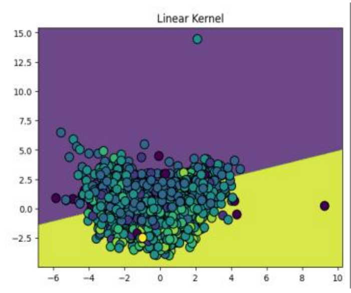
[34]:

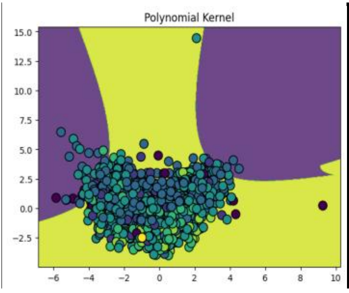
```python
# Function to plot decision boundary
def plot_decision_boundary(X, y, model, title):
    h = .02  # Step size in the meshgrid
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o', s=100)
    plt.title(title)
    plt.show()

# Apply PCA to reduce the data to 2D
pca = PCA(n_components=2)
X_train_2D = pca.fit_transform(X_train)

# Train the models using the 2D PCA-transformed data
svm_linear.fit(X_train_2D, y_train)
svm_poly.fit(X_train_2D, y_train)
svm_rbf.fit(X_train_2D, y_train)

# Plot decision boundaries for each kernel using the 2D PCA-transformed data
plot_decision_boundary(X_train_2D, y_train, svm_linear, "Linear Kernel")
```

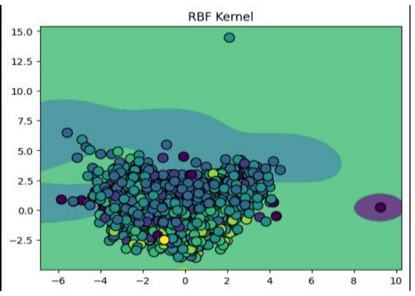
3.1 Plotting Decision Boundaries

This function creates a mesh grid over a 2D feature space and uses the trained SVM models to predict class labels for each grid point, thereby visualizing the decision boundaries. PCA is applied to reduce the dataset to two dimensions for easier visualization, and then the SVM models with different kernels (Linear, Polynomial, and RBF) are fitted to this reduced data. The resulting decision boundaries for each kernel are then plotted to compare how each kernel separates the classes in the feature space.These visualizations help in understanding how each kernel creates decision boundaries to classify the wine data based on its features.

Figure 1. Linear Kernel. Figure 2. Polynomial Kernel. Figure 3. RBF Kernel







4.0 Conclusion

In this tutorial, I explored how Support Vector Machines (SVM) with different kernels can be applied to classify wine data. We experimented with three types of kernels: linear, polynomial, and radial basis function (RBF). Each kernel showed distinct performance characteristics depending on the data's separability and complexity.

- **Linear Kernel**: Best for simple, linearly separable data.
- **Polynomial Kernel**: Effective for data that cannot be separated by a straight line.

- **RBF Kernel**: The most powerful and flexible kernel, effective for complex, non-linear relationships.

References

1. Adithya, A.P. (2024). Wine Quality Detection Using Machine Learning: A Comparative Analysis of Classification Algorithms. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, 08(12), pp.1–6. doi:https://doi.org/10.55041/ijsrem40168.
2. He, S. (2024). The effectiveness of PCA and various hyperparameter settings in SVM and KNN for wine quality estimation. *Applied and Computational Engineering*, 31(1), pp.86–95. doi:https://doi.org/10.54254/2755-2721/31/20230128.
3. Research on Wine Quality Prediction with HHO-SVM. (2024). *Academic Journal of Computing & Information Science*, 7(6). doi:https://doi.org/10.25236/ajcis.2024.070611.