Ilan Barr, Lindsey Gorman, & Sourav Mahanty
Dr. Jinsong Zhang
Introduction to Machine Learning
18 December 2023

Introduction

Background

Machine learning is a branch of artificial intelligence that allows algorithms to learn patterns and insight from data, improving their performance on tasks over time without explicit programming. There are a myriad of models that enable machines to learn from experience or data and make predictions or decisions over time. The area developed quickly through the late 1990s to the 2010s and is now valuable in numerous domains, given its contributions to advancements. Despite the progress machine learning has made in the past decades, a few challenges still persist. Models require high quality and quantity of data to operate successfully. Further, often, there is a challenging tradeoff in balancing model complexity and generalization. Moving forward, the evolution of machine learning will continue to impact and transform industries, offering promising solutions to complex problems.

Summary of Project

This project explores a wine quality dataset with the goal of building a model to accurately predict the quality of given wines based on select physiochemical properties. These properties include acidity levels, residual sugar, pH, alcohol content, and more. With over 6,000 instances, the dataset is robust and offers numerous ways to approach the prediction challenge. Building accurate models will allow the prediction of quality ratings of new or unseen wines based on their chemical composition, holding value for wine producers and consumers.

The following report will detail the dataset's exploration, analysis, and preprocessing and subsequent development of machine learning models. Three algorithms will be trained and assessed to determine their efficacy in predicting wine quality based on the provided features: K Nearest Neighbors, Artificial Neural Network, and the Random Forest algorithm.

Exploratory Analysis

The wine quality data set features 6,497 unique quality assessments for red and white wines. Each entry contains information on the following attributes: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, and alcohol. There are no missing or null values for any of the data, though based on the spread of values for each attribute, standardization is recommended (Figure A1).

We created a correlation matrix to gain a preliminary understanding of the relationship between attributes (Figure A2). Key insights from the correlation matrix are the correlation between quality with alcohol (0.44) and quality with density (-0.31). However, density and total sulfur dioxide had non-statistically significant relationships with quality. We chose to exclude these attributes from the model to improve accuracy.

Before training and testing the models, we examined the spread of data points within the set. Figure B1 and similar boxplots show that some data were outliers from the 25% and 75% quartiles. Given their variability from the mean data, they likely influence the accuracy of models in predicting quality. We used a limit of 1.5 times the interquartile range to account for these significant outliers. All data outside of this were excluded from the models to improve the accuracy of prediction and application in further

.

settings. Appendix A showcases the distribution of values across features and thus the points outside of 1.5 times the interquartile range.

Methods

Artificial Neural Network

Artificial Neural Networks (ANNs) are a subset of machine learning models inspired by the human brain's neural structure. They consist of interconnected nodes organized in layers: input, hidden, and output. Each node receives input, processes it using a weighted sum, and applies an activation function to produce an output. The input layer receives raw data or features, and the output layer produces the network's final output. Hidden layers are intermediate between the input and output, processing information through weighted connections.

The learning process for ANNs occurs in a few steps. First, with forward propagation, the input data is passed forward through the network layer by layer, producing an initial output. Then, the output is compared to the actual target value, and a loss function measures the prediction error. The error is then propagated backward through the network during backpropagation, adjusting weights and biases using optimization techniques to minimize error. Various optimization algorithms update weights to improve the network's performance. This process is repeated through a series of training and testing.

During training, the network learns patterns from labeled data by adjusting weights and biases iteratively to minimize the error. Then, separate from training, validation data helps assess the model's performance during training and fine-tune hyperparameters. Finally, testing occurs via unseen data to evaluate the model's generalization and performance. Implementing an ANN occurs by following these steps and determining the ideal number of hidden layer nodes that produce the highest rate of accuracy.

K-Nearest Neighbor

The k-Nearest Neighbors (KNN) algorithm is a simple yet effective supervised machine learning algorithm for classification and regression tasks. It operates on the principle of similarity, assuming that similar data points tend to belong to the same class or have similar outputs. The algorithm begins with initialization, prediction given a new data point, classification tasks, and regression tasks.

Initialization for KNN requires storing the training data points and their corresponding labels or output values. The model is then given a new, unseen data point, and the algorithm identifies the k-nearest neighbors based on a chosen distance metric. These neighbors are the k data points from the training set closest to the new data point in the feature space. For classification tasks, the algorithm predicts the class of the new data point by majority voting among its k-nearest neighbors. The class that occurs most frequently among the k neighbors is assigned to the new data point. Finally, for regression, the algorithm predicts the output value for the new data point by taking the average or weighted average of the output values of its k nearest neighbors.

Implementation requires a choice of distance calculation (Euclidean, Manhattan, etc.) to measure the distance between data points in the future space. The model can be evaluated using various evaluation metrics: accuracy, precision, recall, RMSE, etc. There are a few additional key considerations when implementing the KNN model. The choice of k significantly impacts the model's performance. A smaller k may lead to overfitting, while a larger k has the potential for underfitting. Feature scaling or normalization may also be necessary to ensure features equally contribute to distance calculations.

.

Random Forest

The Random Forest is an ensemble learning method that combines the predictions of multiple individual decision trees to improve accuracy and reduce overfitting. It's widely used for classification and regression tasks due to its robustness and ability to handle complex datasets. The Random Forest consists of a collection of decision trees, each training on a random subset of the training data and features. Each tree randomly selects a subset of the training data with replacement. This results in different trees training on different subsets of the data. Additionally, each tree randomly selects a subset of features during the node-splitting step.

Each decision tree is grown by recursively splitting nodes based on feature thresholds that optimize certain criteria. The splits are determined based on the feature that best separates the data at each node, considering only a subset of features at each split. For classification, each tree 'votes' for the class, and the class with the most votes becomes the predicted class for the Random Forest. Evaluation of the efficacy of the model can be performed via numerous metrics, such as accuracy, RMSE, etc.

Key considerations when implementing Random Forest include hyperparameters, feature importance, and how to handle overfitting. Parameters like the number of trees, maximum depth of trees, and minimum samples per leaf are each choices that can significantly impact model performance. The model also provides insight into which features contribute to reducing impurity or error within the trees. Random Forest trees tend to be less prone to overfitting than individual decision trees due to ensemble averaging.

Results & Analysis

After creating the ANN model and going through hyperparameter tuning, the model was only performing at a 63% accuracy.

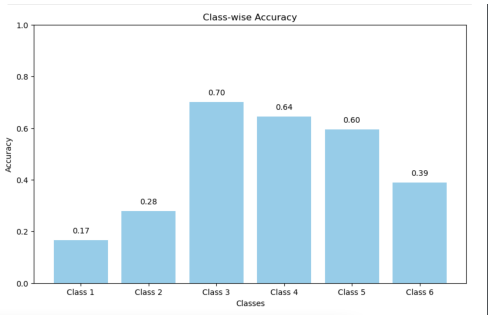


Figure B2.

We concluded that classifying the wine into 6 different quality categories was not necessarily the best way to utilize prediction algorithms, so we created sub-classes of *bad, medium,* and *good* quality wines. These subclasses performed similarly in other features with little significant differentiation in attribute mapping. Bad contained data with quality ratings of 0 to 4, medium with quality ratings of 5 to 6, and good with quality ratings of 7 to 9. This simplified the prediction process greatly, and allowed us to achieve a prediction accuracy higher than 83%.

.

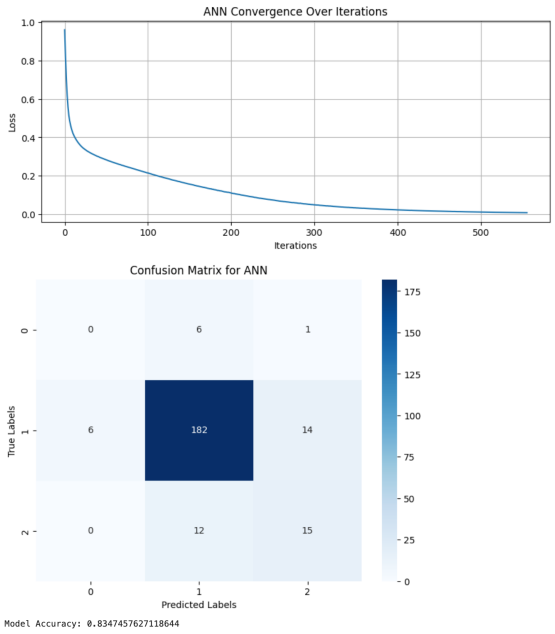


Figure B1.

Similarly, we followed the same structure for both the KNN and Random Forest models which we subsequently created. After pre-processing the data and hyperparameter tuning of the models, we achieved 88% and 86% on the Random Forest and KNN models respectively.

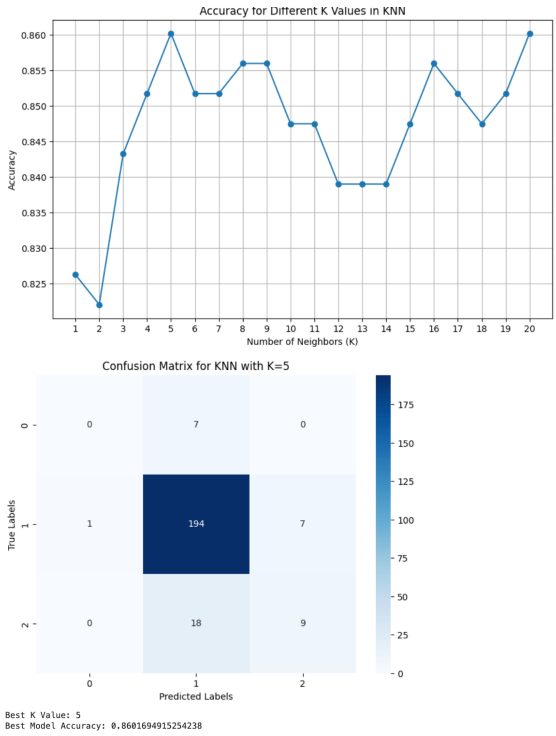


Figure C1

.

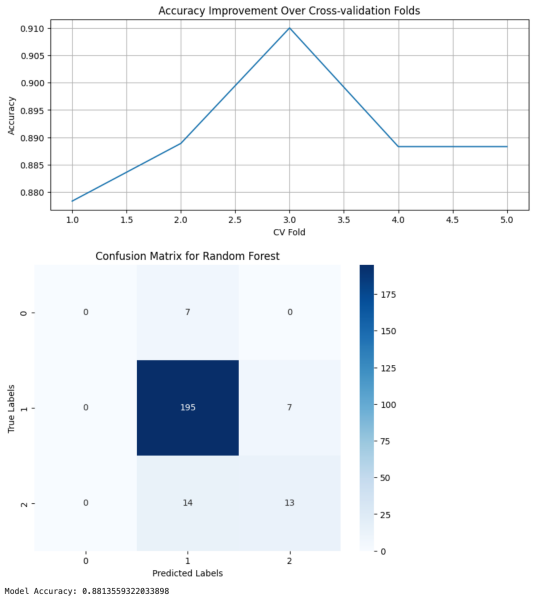


Figure D1.

The enhanced approach of categorizing wine into three broader quality classifications rather than the original six, both simplified the prediction process, while improving the performance of our models across the board. While all models performed particularly well, it should be noted that the Random Forest model, in particular, demonstrated remarkable robustness and adaptability with its 88% prediction accuracy.

Conclusions

In this project, we had the opportunity to take a deep dive into wine quality predictions, and examine possible actionable, data-driven, decisions that can be applied in the food and beverage industry. Through the exploration of a comprehensive dataset containing information on wine quality and various physicochemical properties, we identified key factors that most influence wine quality. We developed three machine learning models - Artificial Neural Networks, K-Nearest Neighbors, and Random Forest, which enabled us to properly evaluate the efficacy of various methods in predicting the quality of wine.

After preprocessing the data and going through the process of feature selection, our original approach of using six classification labels to predict the wine quality proved not to be not as successful as we had hoped. The solution we came up with was to combine the quality classes into three classes total. The simplified prediction process ultimately led the models to achieving significantly higher correct prediction rates, all of them over 80%.

Throughout the project, we navigated the challenges of balancing model complexity with generalization, dealing with data outliers, and optimizing model parameters. These experiences emphasize the evolving and iterative nature of the machine learning process.

Going forward, the methodologies used, and insights gained from this project can be applied to many fields. Predictive modeling is increasingly growing in relevance in the modern world as our machines are more and more capable of performing complex computations in reasonable times. The intersection of machine learning and real world applications presents vast opportunities for industry

.

transformation and growth. The methodology applied in this project can be effectively utilized in real world scenarios, driving innovation and enhancing performance and efficiency.

All three members of the group worked in developing the Python code and implementing various models. Lindsey Gorman completed the exploratory analysis along with Sourav Mahanty. Both Sourav and Ilan Barr worked on iterations of the ANN and KNN. The group discussed improvements to the models and worked collaboratively to choose features and outliers to remove as well as which classes to combine. The final code for the three models used was developed by Sourav. Lindsey and Ilan completed the written portion of the project as well as the appendix.

.

Appendix A

Exploratory Analysis

Below are the associated tables and plots regarding the exploratory analysis performed. It is important to note the significant relationships between attributes in the correlation matrix, specifically those between quality with alcohol (0.44) and quality with density (-0.31). These are the highest absolute values associated with quality.

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000

	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	115.744574	0.994697	3.218501	0.531268	10.491801	5.818378
std	56.521855	0.002999	0.160787	0.148806	1.192712	0.873255
min	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000
50%	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000
75%	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000
max	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

Figure A1. Descriptive Statistics for Attributes

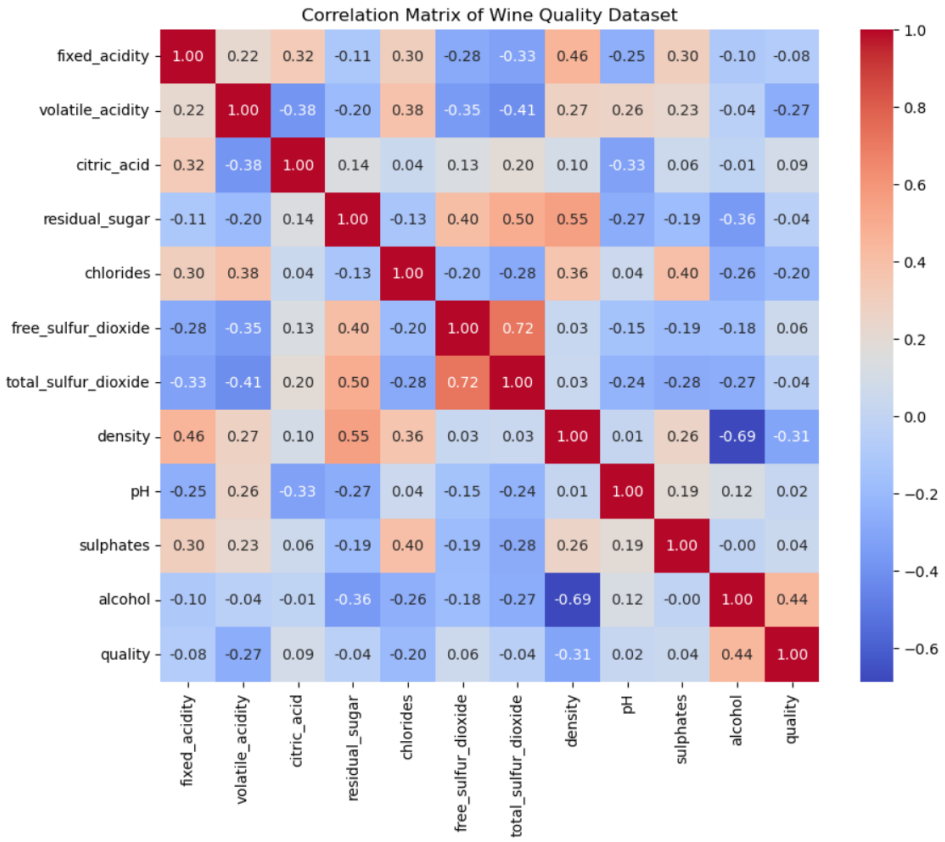


Figure A2.

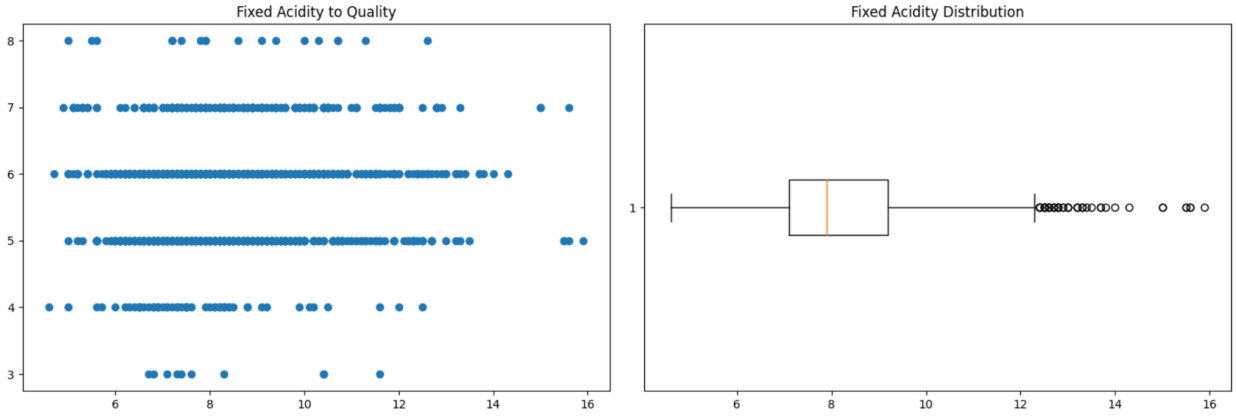


Figure A3.

.

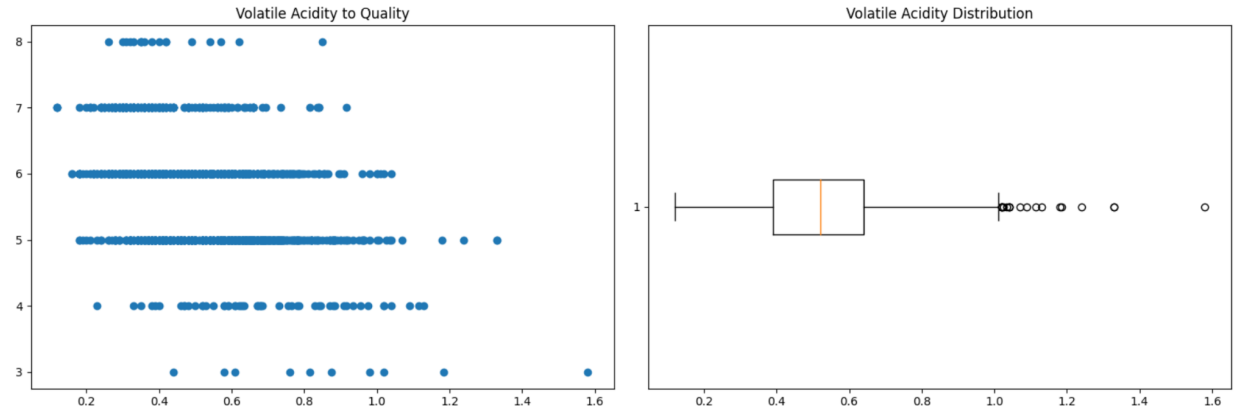


Figure A4.

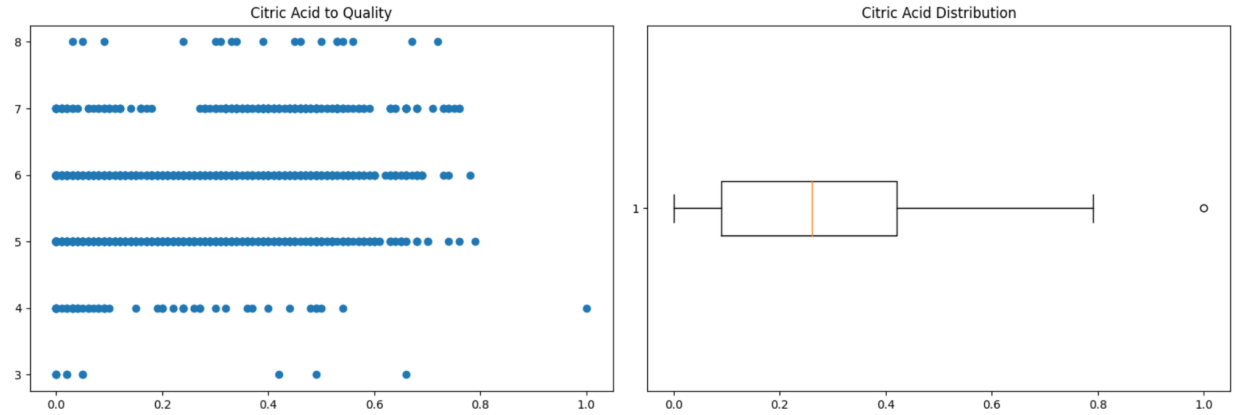


Figure A5.

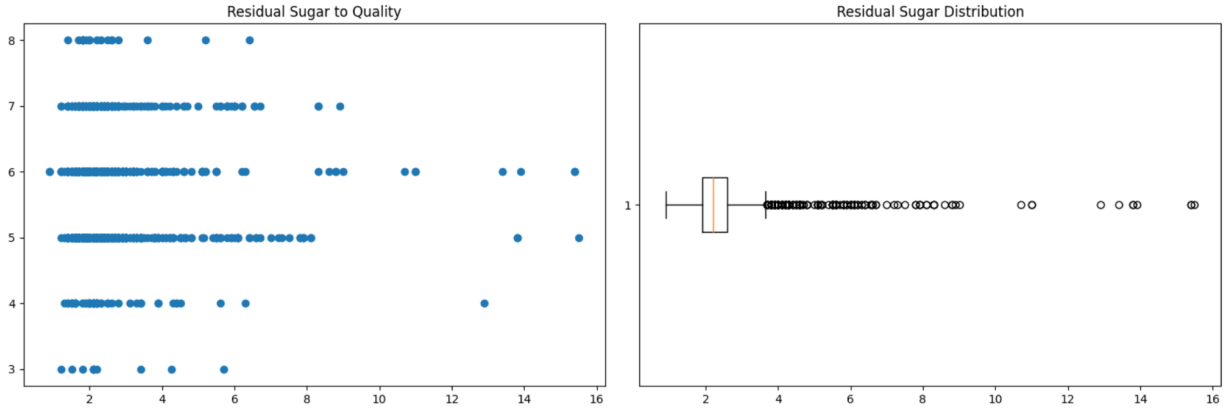


Figure A6.

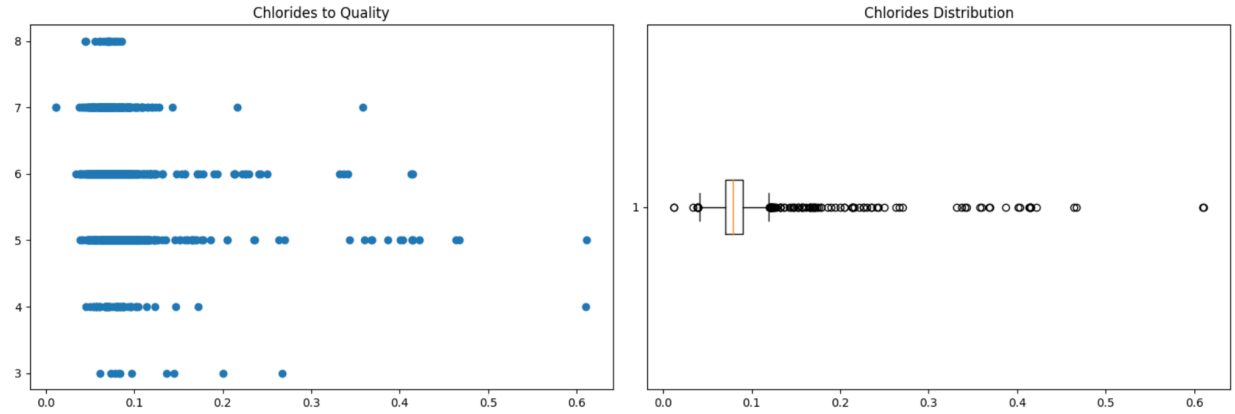


Figure A7.

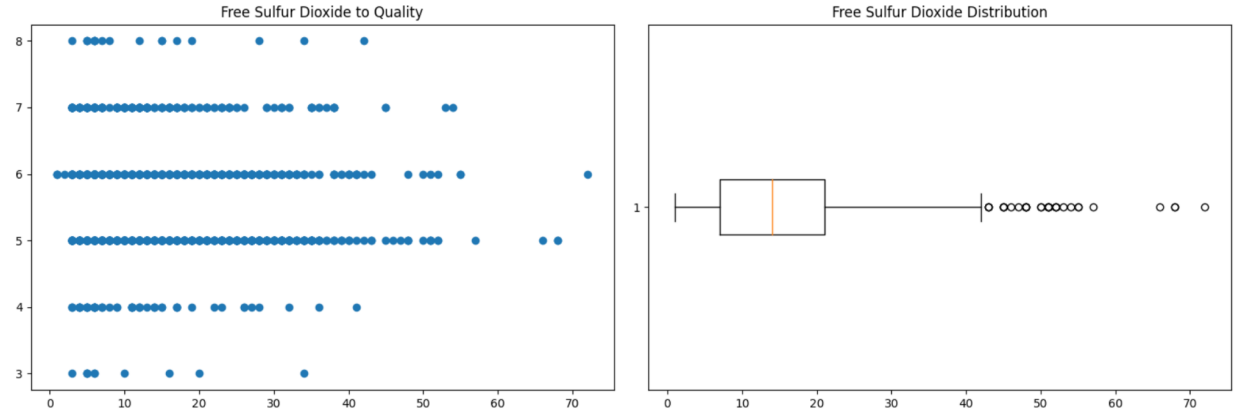


Figure A8.

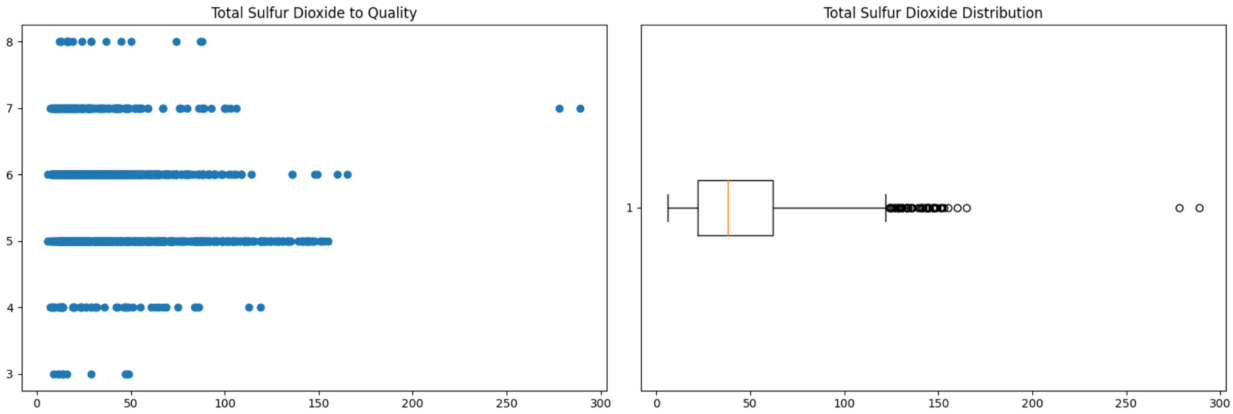


Figure A9.

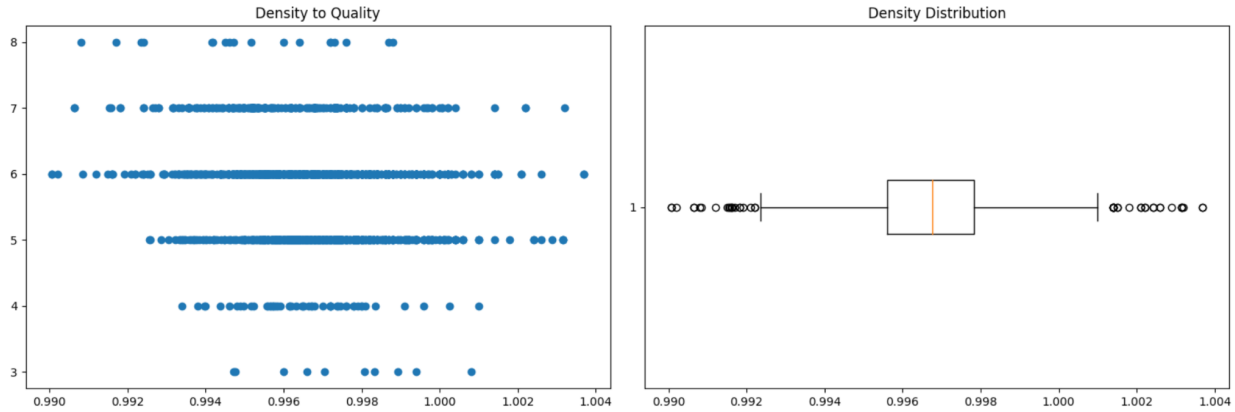


Figure A10.

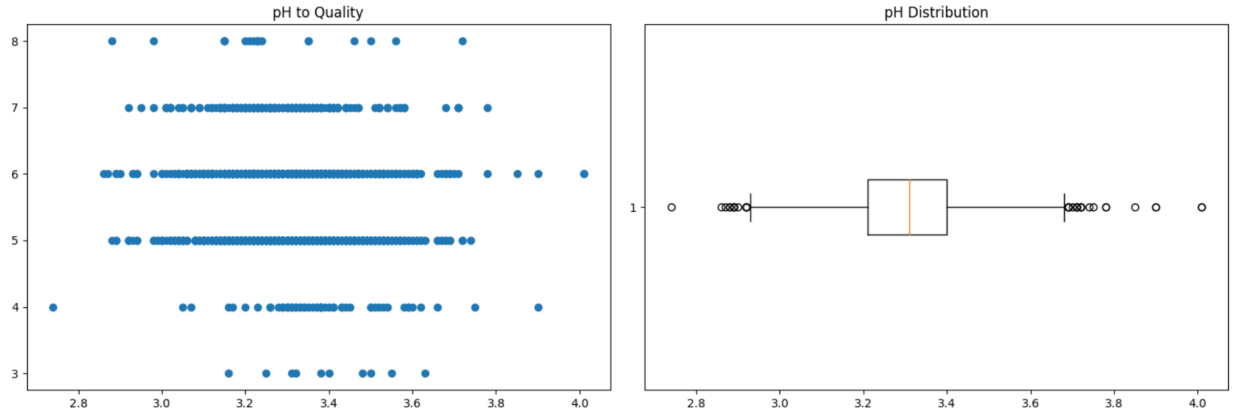


Figure A11.

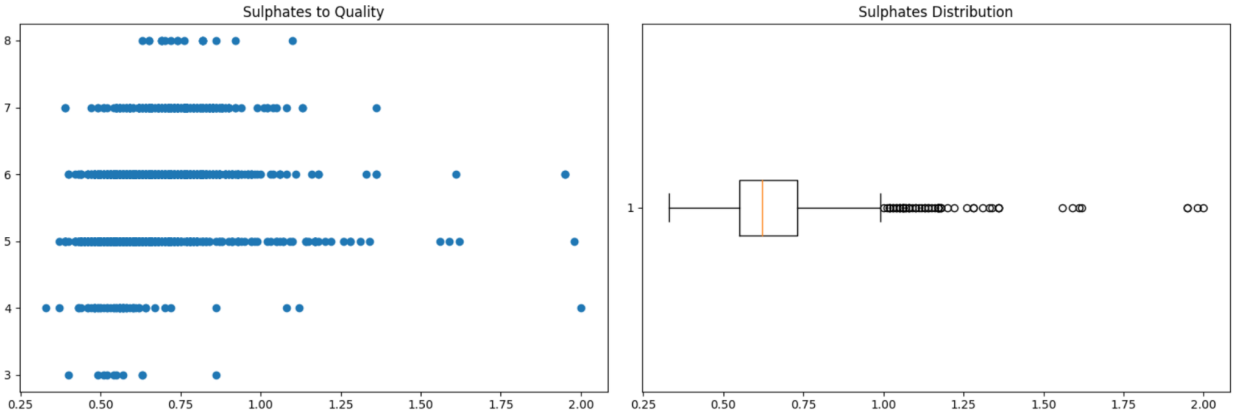


Figure A12.

.

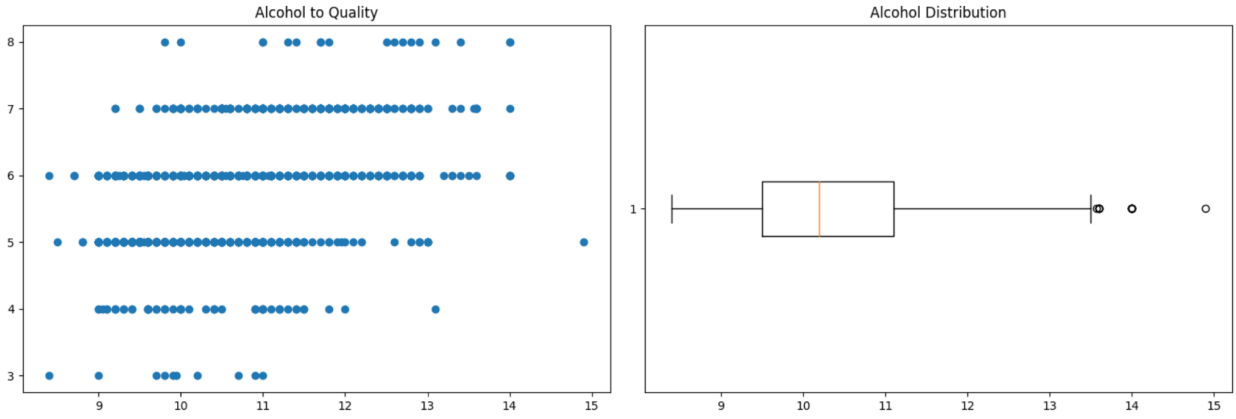
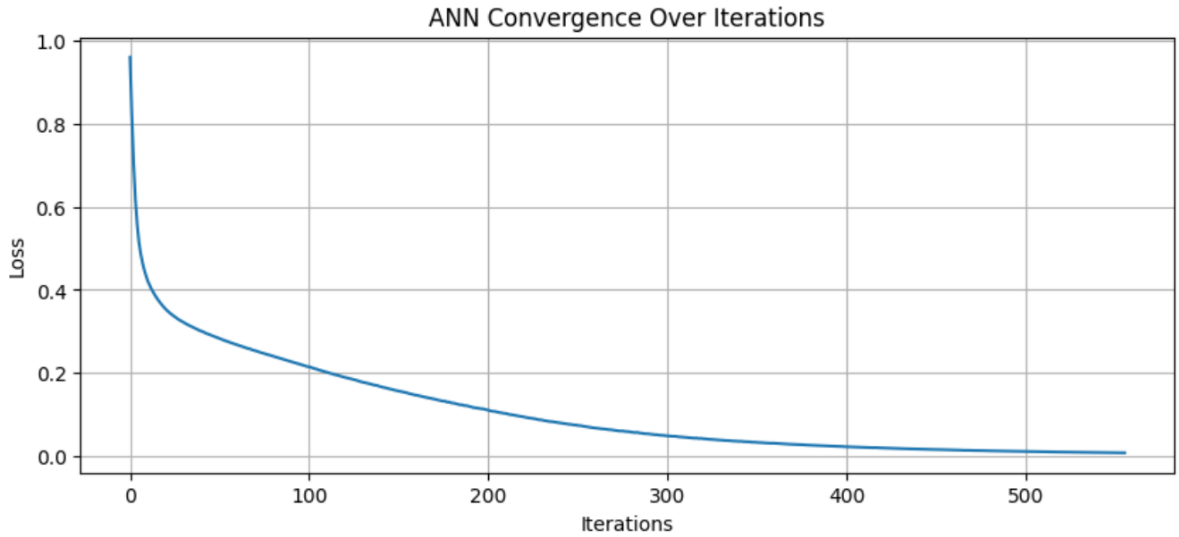

Alcohol to Quality
8
7
6
5
4
3
9
10
11
12
13
14
15
Alcohol Distribution
1
9
10
11
12
13
14
15

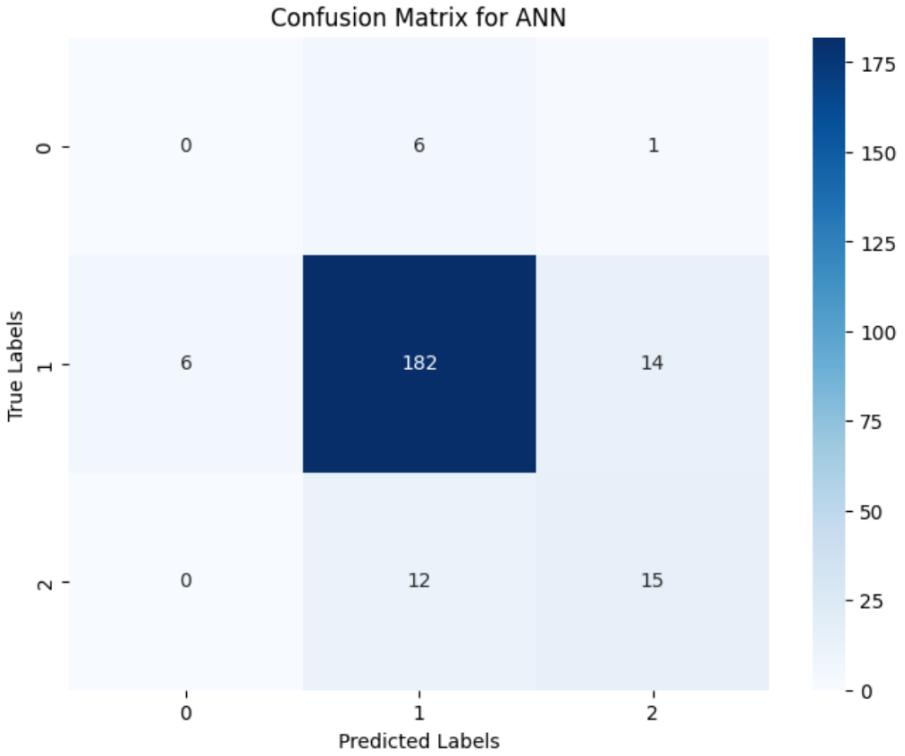
Figure A13.

.

Appendix B

Artificial Neural Network Outputs






```
Model Accuracy: 0.8347457627118644
```

Figure B1.

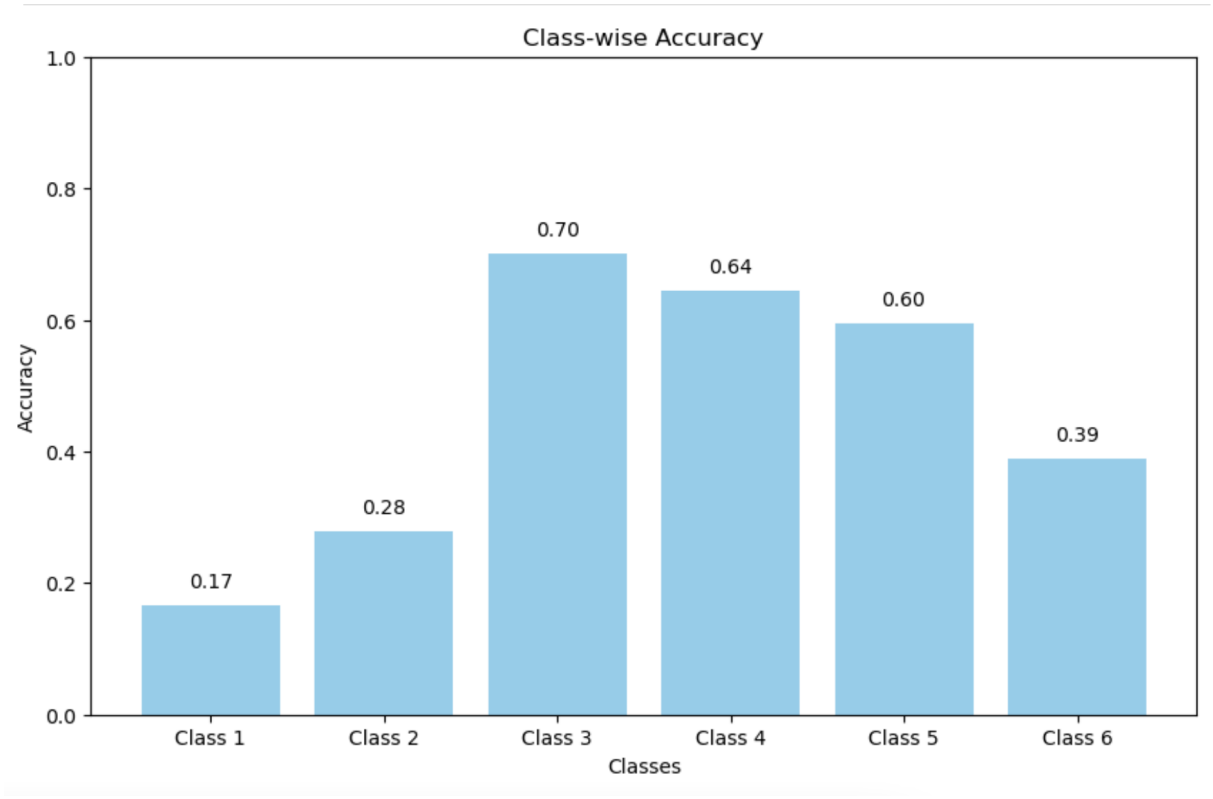
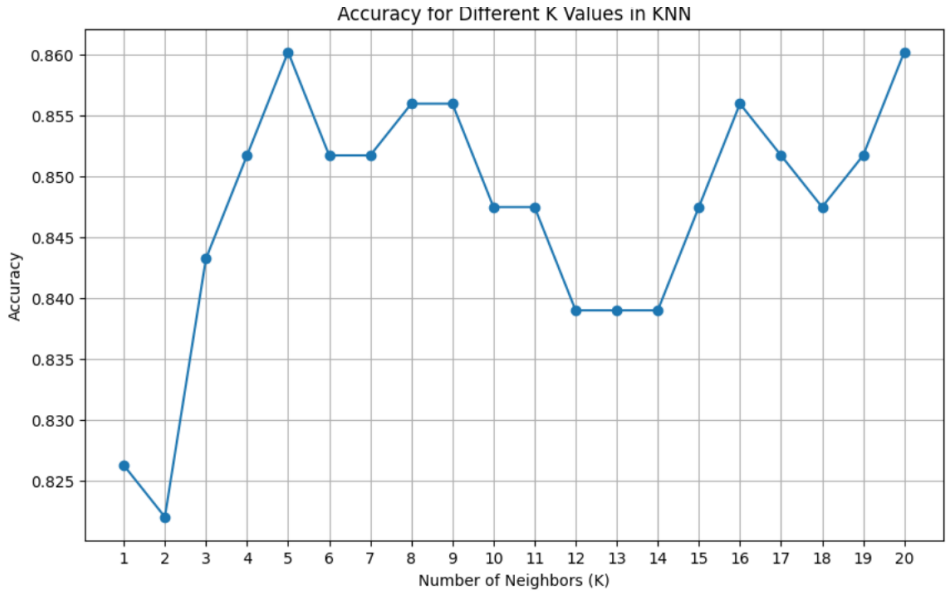
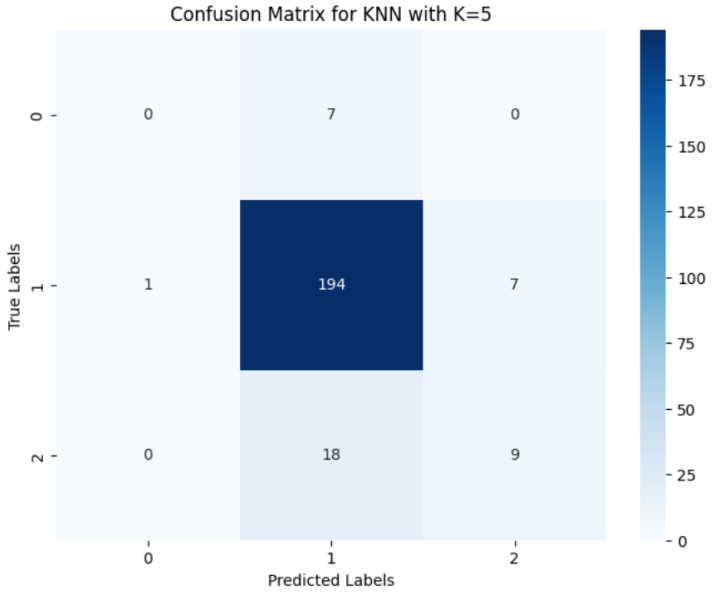

Class-wise Accuracy
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0.17
0.28
0.70
0.64
0.60
0.39
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Classes

Figure B2.

Appendix C

K-Nearest Neighbors Outputs



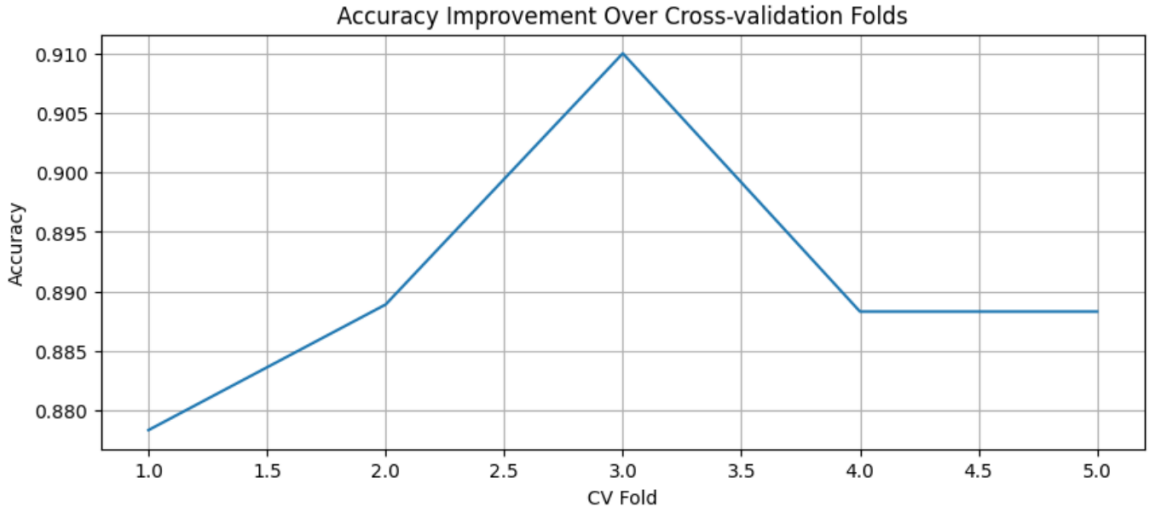


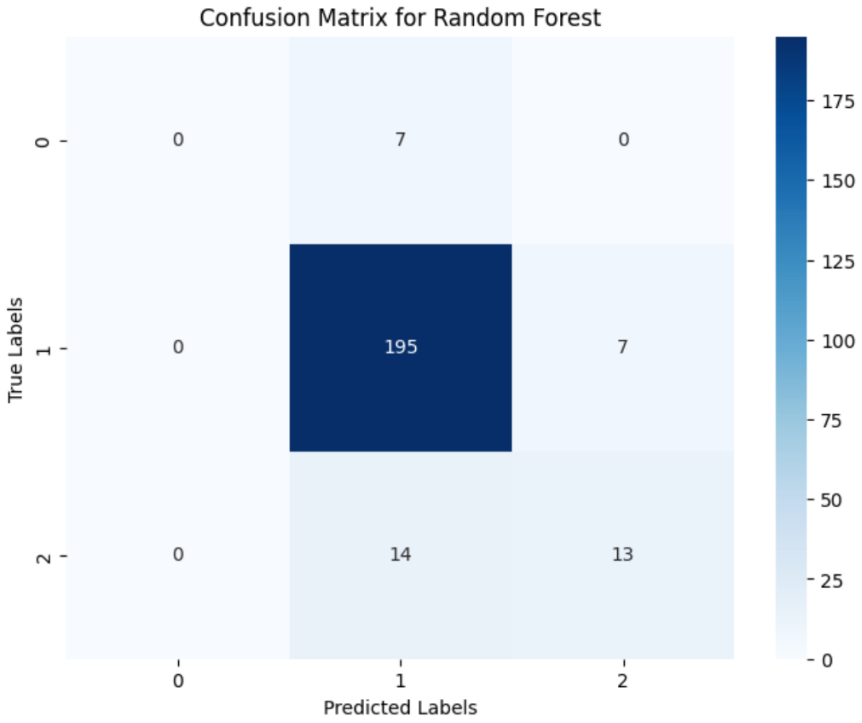

```
Best K Value: 5
Best Model Accuracy: 0.8601694915254238
```

Figure C1.

Appendix D

Random Forest Model Outputs






```
Model Accuracy: 0.8813559322033898
```

Figure D1.

.

Appendix E

Python Code.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
from sklearn.datasets import make_blobs
import plotly.express as px
import plotly.graph_objects as go
import ipywidgets as widgets
from IPython.display import display

wines = pd.read_excel('final_project_data.xlsx')

print("Column Names: ")
print(wines.columns)

print("\n# of Rows: ")
print(wines.shape[0])
```

```
Column Names:
Index(['Fixed Acidity', 'Volatile Acidity', 'Citric Acid', 'Residual Sugar',
       'Chlorides', 'Free Sulfur Dioxide', 'Total Sulfur Dioxide', 'Density',
       'pH', 'Sulphates', 'Alcohol', 'Quality'],
      dtype='object')

# of Rows:
1599
```

3-D Visualization of Features

```
# List of column names (excluding 'Quality')
features = ['Fixed Acidity', 'Volatile Acidity', 'Citric Acid', 'Residual Sugar',
            'Chlorides', 'Free Sulfur Dioxide', 'Total Sulfur Dioxide', 'Density',
            'pH', 'Sulphates', 'Alcohol']

# Dropdown widgets for the features
dropdown_x = widgets.Dropdown(options=features, value='Sulphates', description='X-axis:')
dropdown_y = widgets.Dropdown(options=features, value='Volatile Acidity', description='Y-axis:')
dropdown_z = widgets.Dropdown(options=features, value='Alcohol', description='Z-axis:')

# Initial plot
fig = go.FigureWidget(px.scatter_3d(wines, x=dropdown_x.value, y=dropdown_y.value, z=dropdown_z.value, color='Quality'))
fig.update_layout(width=800, height=600)  # Update the size here

# Update function for the plot
def update_plot(change):
    with fig.batch_update():
        fig.data[0].x = wines[dropdown_x.value]
        fig.data[0].y = wines[dropdown_y.value]
        fig.data[0].z = wines[dropdown_z.value]

        # Update axis titles
        fig.layout.scene.xaxis.title = dropdown_x.value
        fig.layout.scene.yaxis.title = dropdown_y.value
        fig.layout.scene.zaxis.title = dropdown_z.value

# Observe changes in dropdowns
dropdown_x.observe(update_plot, names='value')
dropdown_y.observe(update_plot, names='value')
dropdown_z.observe(update_plot, names='value')

# Display the widgets and interactive plot
widgets.VBox([dropdown_x, dropdown_y, dropdown_z, fig])
```

```
VBox(children=(Dropdown(description='X-axis:', index=9, options=('Fixed Acidity', 'Volatile Acidity', 'Citric …
```

.

```python
# Final Wine Quality Analysis using K-Nearest Neighbors (KNN) with Stratified Sampling and Outlier Removal

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Load the dataset
file_path_excel = '/mnt/data/final_project_data.xlsx'
wine_data = wines

# Outlier Removal
Q1 = wine_data.quantile(0.25)
Q3 = wine_data.quantile(0.75)
IQR = Q3 - Q1
outliers = ((wine_data < (Q1 - 1.5 * IQR)) | (wine_data > (Q3 + 1.5 * IQR))).any(axis=1)
wine_data_no_outliers = wine_data[~outliers]

# Feature Dropping (Remove 'Density' and 'Total Sulfur Dioxide')
wine_data_reduced = wine_data_no_outliers.drop(['Density', 'Total Sulfur Dioxide'], axis=1)

# Merge classes 5 and 6
wine_data_reduced['Quality'] = wine_data_reduced['Quality'].replace(5, 6)

# Split the data with stratified sampling
X = wine_data_reduced.drop('Quality', axis=1)
y = wine_data_reduced['Quality']
X_scaled = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, stratify=y, random_state=42
)

# Find the best K value
k_values = range(1, 21)
accuracy_scores = []
for K in k_values:
    knn = KNeighborsClassifier(n_neighbors=K)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

# Plotting accuracy scores for different K values
plt.figure(figsize=(10, 6))
plt.plot(k_values, accuracy_scores, marker='o')
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Accuracy')
plt.title('Accuracy for Different K Values in KNN')
plt.xticks(k_values)
plt.grid(True)
plt.show()

# Retrain the model with the best K value and generate a confusion matrix
best_K = k_values[np.argmax(accuracy_scores)]
knn_best = KNeighborsClassifier(n_neighbors=best_K)
knn_best.fit(X_train, y_train)
y_pred_best = knn_best.predict(X_test)
best_accuracy = accuracy_score(y_test, y_pred_best)
conf_matrix = confusion_matrix(y_test, y_pred_best)

# Visualizing the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap='Blues')
plt.title('Confusion Matrix for KNN with K=' + str(best_K))
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()

print("Best K Value:", best_K)
print("Best Model Accuracy:", best_accuracy)
```

```python
# Artificial Neural Network (ANN) for Wine Quality Classification

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Load the dataset
wine_data = wines

# Outlier Removal
Q1 = wine_data.quantile(0.25)
Q3 = wine_data.quantile(0.75)
IQR = Q3 - Q1
outliers = ((wine_data < (Q1 - 1.5 * IQR)) | (wine_data > (Q3 + 1.5 * IQR))).any(axis=1)
wine_data_no_outliers = wine_data[~outliers]

# Feature Dropping (Remove 'Density' and 'Total Sulfur Dioxide')
wine_data_reduced = wine_data_no_outliers.drop(['Density', 'Total Sulfur Dioxide'], axis=1)

# Merge classes 5 and 6
wine_data_reduced['Quality'] = wine_data_reduced['Quality'].replace(5, 6)

# Standardize the features
X = wine_data_reduced.drop('Quality', axis=1)
y = wine_data_reduced['Quality']
X_scaled = StandardScaler().fit_transform(X)

# Split the dataset with stratified sampling
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, stratify=y, random_state=42
)

# Constructing the ANN using MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(64, 32), activation='relu', solver='adam', max_iter=900, random_state=42)

# Training the ANN
mlp.fit(X_train, y_train)

# Predicting on the test set
y_pred = mlp.predict(X_test)

# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Plotting the convergence over iterations
plt.figure(figsize=(10, 4))
plt.plot(mlp.loss_curve_)
plt.title('ANN Convergence Over Iterations')
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.grid(True)
plt.show()

# Visualizing the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap='Blues')
plt.title('Confusion Matrix for ANN')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()

print("Model Accuracy:", accuracy)
```

.

```python
# Random Forest for Wine Quality Classification with Outlier Removal and Feature Dropping

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import numpy as np

# Load the dataset
wine_data = wines

# Outlier Removal
Q1 = wine_data.quantile(0.25)
Q3 = wine_data.quantile(0.75)
IQR = Q3 - Q1
outliers = ((wine_data < (Q1 - 1.5 * IQR)) | (wine_data > (Q3 + 1.5 * IQR))).any(axis=1)
wine_data_no_outliers = wine_data[~outliers]

# Feature Dropping (Remove 'Density' and 'Total Sulfur Dioxide')
wine_data_reduced = wine_data_no_outliers.drop(['Density', 'Total Sulfur Dioxide'], axis=1)

# Merge classes 5 and 6
wine_data_reduced['Quality'] = wine_data_reduced['Quality'].replace(5, 6)

# Standardize the features
X = wine_data_reduced.drop('Quality', axis=1)
y = wine_data_reduced['Quality']
X_scaled = StandardScaler().fit_transform(X)

# Split the dataset with stratified sampling
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, stratify=y, random_state=42
)

# Constructing the Random Forest Classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Cross-validation to track accuracy improvement
cv_scores = cross_val_score(rf, X_train, y_train, cv=5)

# Training the Random Forest
rf.fit(X_train, y_train)

# Predicting on the test set
y_pred = rf.predict(X_test)

# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Plotting the accuracy improvement over cross-validation folds
plt.figure(figsize=(10, 4))
plt.plot(range(1, len(cv_scores) + 1), cv_scores)
plt.title('Accuracy Improvement Over Cross-validation Folds')
plt.xlabel('CV Fold')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()

# Visualizing the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap='Blues')
plt.title('Confusion Matrix for Random Forest')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()

print("Model Accuracy:", accuracy)
```

Figure E1.

.