California State University, Northridge

Data Mining COMP 541

Wine Quality Prediction - Final Report

By - Group 4 :

Sarkis Shil-Gevorkyan

Keshav Kumar

Briana Pimentel

Harshleen Sadhnani

December 10, 2023

Table of Contents

Objective

Based on the Red Wine Quality Dataset, our objective is to build a model that can predict the quality (rating from 1-10) of the wine, based on the contents and ingredients. In terms of data mining goals, we will perform EDA to gain insights into the dataset's characteristics, including data distributions, correlations, and potential outliers. And also want to evaluate and compare different machine learning models (e.g., Decision Tree) for their performance in classifying the wine quality. This includes Experiment with hyperparameter tuning to optimize the selected classification algorithm(s) and achieve the best possible model performance.

Background Information

The Red Wine Quality dataset is a labeled and structured dataset that can be easily imported using the Python Pandas library. It was obtained from Kaggle and saved as a CSV file. The dataset includes a number of quantitative characteristics linked to the composition and quality of wine. These consist of the target variable, quality, which represents customer ratings on a scale of 1 to 10, as well as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, and alcohol. Preprocessing of the data involved handling missing values and changing data formats in order to address data integrity and compatibility concerns. The dataset, which is categorized as tabular data, is arranged in rows and columns to allow for systematic analysis and comparison.Attribute types include both discrete (e.g., free sulfur dioxide, total sulfur dioxide, and quality) and continuous attributes, all falling under the category of numeric attributes. Notably, the dataset exclusively comprises numeric data. Overview of the dataset's origin, characteristics, preprocessing steps, and suitability for analytical tasks.

Design Principle and Scope

Scope

- Binary Classification: The primary goal is to perform binary classification to determine whether a wine is "good" or "not good" based on predefined quality Input variables (e.g., quality > 6.5 = good).
- Multi classification: We also implemented a multi level classification where we were able to categorize the wine quality into 5 groups.
- Feature Importance: Identify the physicochemical properties (input variables) that significantly influence wine quality classification.
- Model Selection: Evaluate and compare different machine learning algorithms (e.g., Decision Trees and Random forest) for their performance in classifying wine quality.
- Hyperparameter Tuning: Experiment with hyperparameter tuning to optimize the selected classification algorithm(s) and achieve the best possible model performance.
- Model Interpretability: Aim to create a model that provides interpretable results, allowing us to understand which physicochemical factors contribute to a wine being classified as “Good” or “Bad”.
- Exploratory Data Analysis (EDA): Perform EDA to gain insights into the dataset's characteristics, including data distributions, correlations, and potential outliers.

Data Sources

We found our dataset from kaggle.
https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

Data mining technology

- Tools: python, pandas , scikit-learn, Matplotlib , Kaggle.
- Techniques: Binary Classification model to correctly predict if the wine is good or not based on the input variable.
- Implementation: Explored various different algorithms such as, Logistic regression, Decision tree, K nearest neighbor, and Random forest.

Data Exploration And Data PreProcessing

The data exploration is to uncover the dataset characteristics to have a better understanding of the nature of given data. The central tendency measure's purpose is to find the middle of the distribution of data. By calculating the midrange, mode, median, and mean. By determining the range, quartiles, and IQR. The purpose of data exploration is to get a sense of the data's dispersion, or how the data is distributed.And measure the skewness to see the deviation of the 12 attributions. Since the goal here is to check for asymmetry, or distortion of a symmetric distribution, The ending result is Features: “residual sugar," "chlorides," “free sulfur dioxide," “total sulfur dioxide,” and “sulfates” have a skewness greater than 1, meaning that their distributions are heavily skewed. The features “fixed acidity," “volatile acidity,” and “alcohol” have skewness that is between 0.5 and 1, and their distributions are moderately skewed. And the features “citric acid," "density," "pH,” and “quality” have a skewness between 0 and 0.5, conclude that distributions are approximately symmetric.The next step in data exploration is checking for class imbalances to see if class values are balanced or not. Based on the class count of the quality of wine, there is a greater number of ratings of 5 or 6 quality than the others; therefore, there is a bias more towards the quality of 5 and 6 than for the rest. Then explore the correlation in the dataset to show that one or more attributes are related to each other. It gives an idea of the degree of the relationship between the two attributes. The heat map diagram is a 2-dimensional data visualization technique that represents the magnitude of individual values within a dataset as a color. The variation in color may be due to hue or intensity. In this dataset, the lighter the color, the greater the positive correlation between two attributes, and the darker the color, the greater the negative correlation.

The data preprocessing is cleaning up the given dataset before executing the next step.The initial dataset contains an imbalance, so one way to overcome this is by using a random oversampler. Using RandomOverSampler() means resampling the minority class with a replacement. After that task, it even distributes every class, and there is also more data, which can help the model perform better. The next step is checking for missing data and using isnull (checks for empty cells) and isna (checks for NaN

values in cell).Values were checked, and fortunately, the dataset did not contain any. Then check for outliers using the boxplot diagram to have a visual of the result. The result shows a ton of outliers that could cause a negative analysis. The next step is to replace all outliers with the median value of that column or feature. The median, first and third quartiles, and the IQR, upper and lower bounds, would be defined. This function would then go through each value in every column, and any values outside of those bounds would be replaced with the median value. That will help with outlier issues. Then the feature selection in data preprocessing to select relevant out of the 12 attributions to be used evaluates the models. We chose the random forest to ascertain the significance of each feature in order to choose which features to include in our model. The built-in Random Forest algorithm "feature_importances_" indicates which features are pertinent. When a variable is chosen to split a node, the impurity importance of all the trees is added up to determine the impurity importance. In general, a feature's importance increases with its decrease in impurity. We would sort the impurity importance scores after obtaining the list and then plot the results on a bar chart. Following the bar chart's plotting, we eliminated three of the least significant features: "pH," "free sulfur dioxide," and "residual sugars." After cleaning the data and selecting the features to use, Now proceed to build and evaluate the models.

Central Tendency Measures

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000
range	11.300000	1.460000	1.000000	14.600000	0.599000	71.000000	283.000000	0.013620	1.270000	1.670000	6.500000	5.000000
Midrange	5.650000	0.730000	0.500000	7.300000	0.299500	35.500000	141.500000	0.006810	0.635000	0.835000	3.250000	2.500000
IQR	2.100000	0.250000	0.330000	0.700000	0.020000	14.000000	40.000000	0.002235	0.190000	0.180000	1.600000	1.000000

```
df.mode()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.2	0.6	0.0	2.0	0.08	6.0	28.0	0.9972	3.3	0.6	9.5	5

Skewness

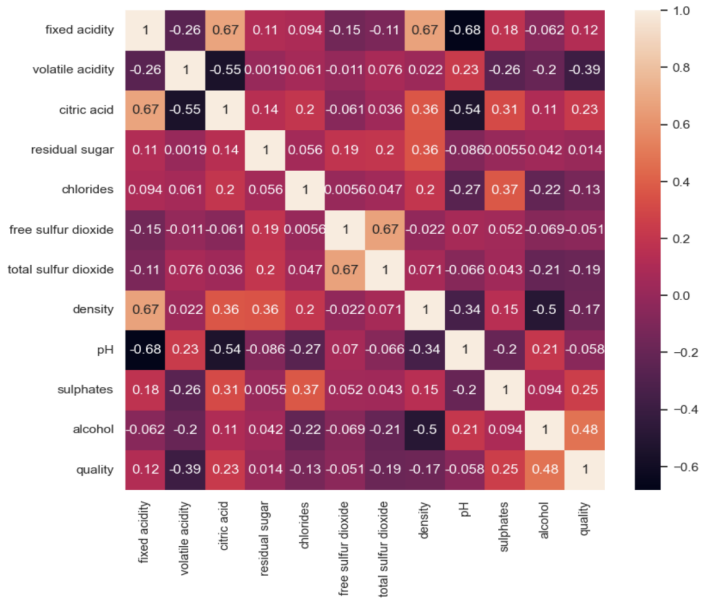
```
In [7]: df.skew()

Out[7]: fixed acidity           0.982751
        volatile acidity        0.671593
        citric acid             0.318337
        residual sugar          4.540655
        chlorides               5.680347
        free sulfur dioxide     1.250567
        total sulfur dioxide    1.515531
        density                 0.071288
        pH                      0.193683
        sulphates               2.428672
        alcohol                 0.860829
        quality                 0.217802
        dtype: float64
```

Class Imbalanced

```
In [8]: class_count = df.groupby('quality').size()
        class_count

Out[8]: quality
        3     10
        4     53
        5    681
        6    638
        7    199
        8     18
        dtype: int64
```

Correlation

```
In [9]: sns.heatmap(df.corr(numeric_only=True), annot=True)
        sns.set(rc = {'figure.figsize':(10, 10)})
```



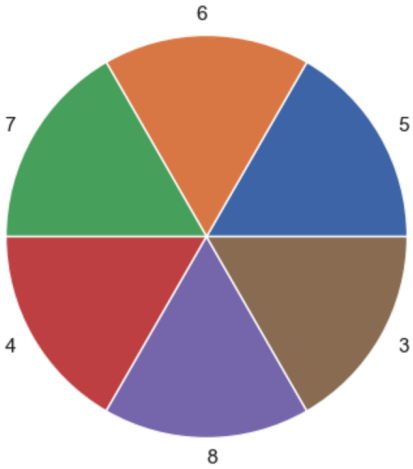
Using Random Oversampler to Fix Class Imbalance

```
: #using random oversampling to fix class imbalance
  from imblearn.over_sampling import RandomOverSampler

  ros = RandomOverSampler(sampling_strategy = "not majority")

  x_res, y_res = ros.fit_resample(x,y)
```

```
: y_res.value_counts()
```

```
: 5    681
  6    681
  7    681
  4    681
  8    681
  3    681
  Name: quality, dtype: int64
```

```
: classes = df['quality'].unique()

  # Count the number of instances in each class
  counts = y_res.value_counts()

  # Create a pie chart
  plt.figure(figsize=(5, 5))
  plt.pie(counts, labels=classes)
  plt.show()
```



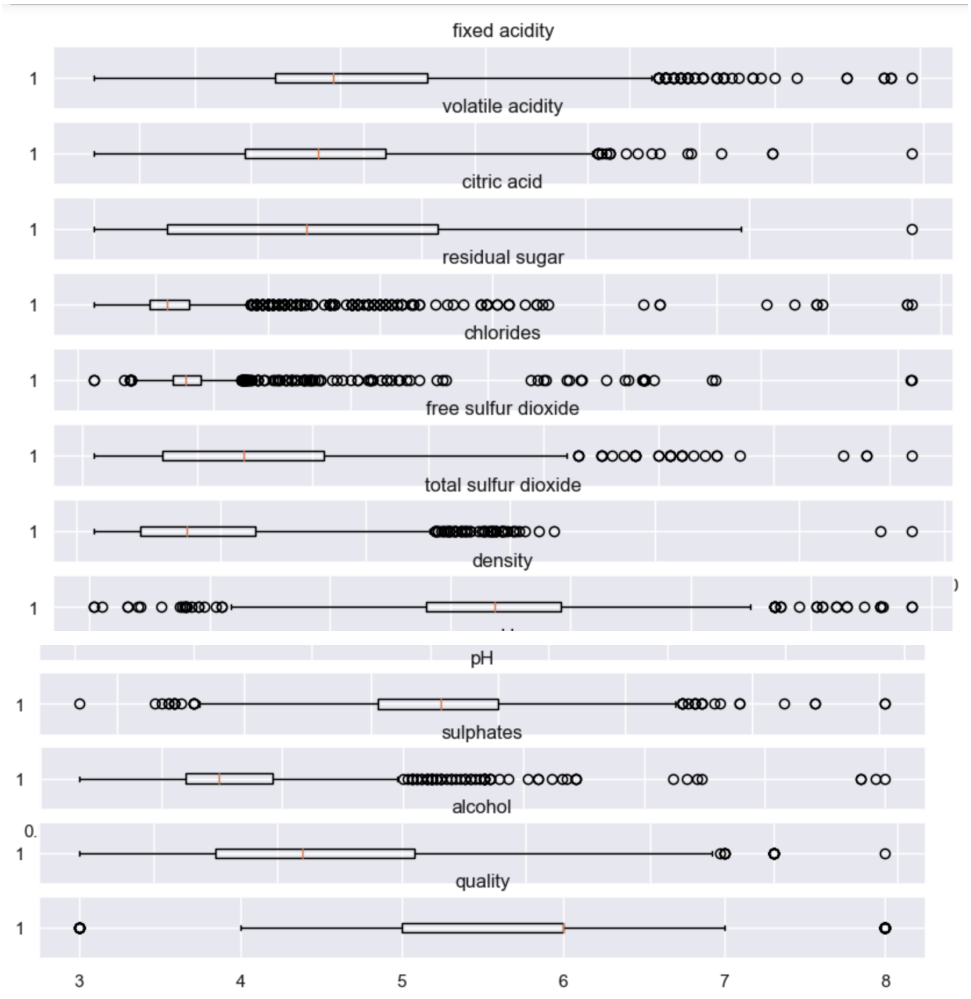
Checking for Missing Data

```
: #Checking for any Missing, Null, or NA values
  print(df.isnull().sum())
  print('\n', df.isna().sum())
```

```
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
dtype: int64

 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
dtype: int64
```

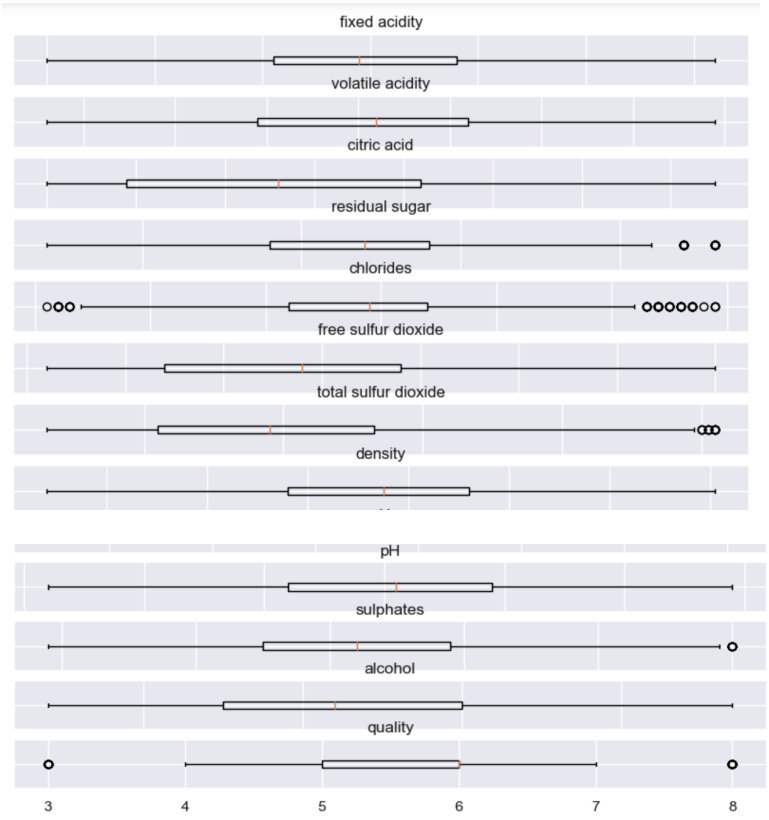
Checking for Outliers using Boxplots

```
#Using boxplots to check for outliars
fig, axes = plt.subplots(nrows=df.shape[1], ncols=1, figsize=(10, 10))

# Loop over the features
for i, feature in enumerate(df.columns):

    # Create a boxplot
    ax = axes[i]
    ax.boxplot(df[feature], vert = False)
    ax.set_title(feature)

# Show the figure
plt.show()
```



```
|: #Function to replace outliers with the median value
   def replace_outliers_with_median(df, threshold = 1):

       # Iterate over each column in the DataFrame
       for column in df.columns[:-1]:
           # Calculate the median and interquartile range (IQR) for the current column
           median = df[column].median()
           q1 = df[column].quantile(0.25)
           q3 = df[column].quantile(0.75)
           iqr = q3 - q1

           # Define the lower and upper bounds for outliers
           lower_bound = q1 - threshold * iqr
           upper_bound = q3 + threshold * iqr

           # Replace outliers with the median for the current column
           df[column] = np.where((df[column] < lower_bound) | (df[column] > upper_bound), median, df[column])

       return df

   df = replace_outliers_with_median(df)
```

Boxplots after Outliers Cleaning



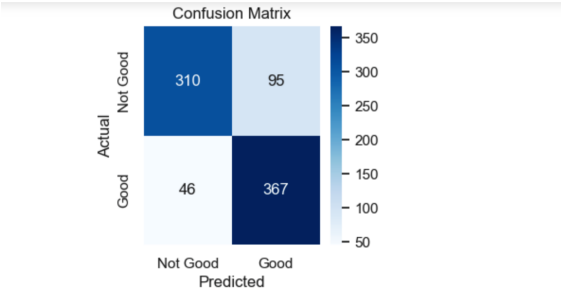
Modeling and Evaluations

When beginning to create the machine learning model for training, we chose Supervised Learning techniques for this project, specifically Classification Algorithms such as Logistic Regression, Decision Tree, K Nearest Neighbor, and Random Forest. We used these, due to their ability to predict categorical outcomes. The rationale behind this is that the wine quality was categorized into classes of ratings from 1-10. We began by splitting the dataset, where 80% of the data was used for training and the other 20% for testing. The modeling tool was run on the prepared dataset, resulting in the generation of multiple models. These models were then evaluated based on their performance metricsThe resulting models are mathematical representations that predict wine quality based on various input features such as acidity, sugar content, alcohol percentage, etc. Each model assigns different weights to these features based on their significance in predicting the outcome. We basically used 3 models which are as follows.

Logistic regression

- Is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. To start, we performed a binary classification, where qualities 3, 4 and 5 were considered “Not Good” and “6, 7, 8” were considered good. Using Scikit learn, we implemented Logistic Regression, with hyper parameters: saga solver and L2 penalty. L2 penalty determines strength of regularization and helps prevent overfitting.

```
: from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver = 'saga', penalty = 'l2')

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```




```
In [176]: print('\nClassification Report:')
          print(classification_report(y_test, y_pred))

Classification Report:
              precision    recall  f1-score   support

        Good       0.87      0.77      0.81       405
    Not Good       0.79      0.89      0.84       413

    accuracy                           0.83       818
   macro avg       0.83      0.83      0.83       818
weighted avg       0.83      0.83      0.83       818
```

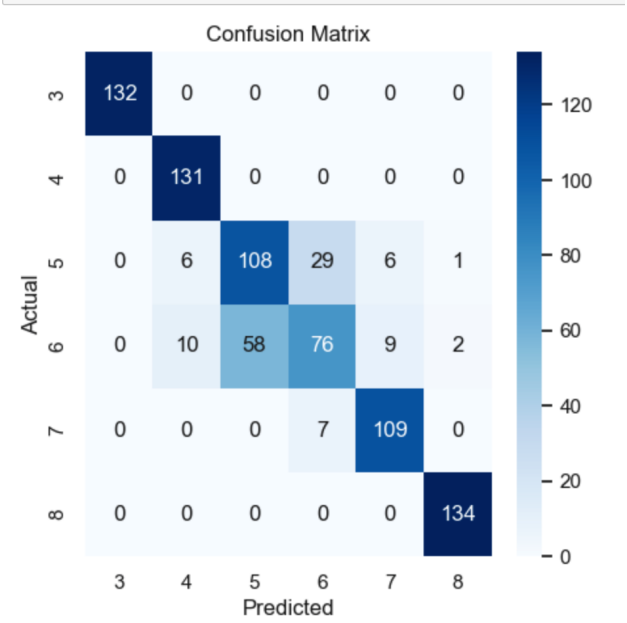
Based on the confusion matrix and the classification report, we can see that the binary classification has performed relatively well, giving an overall accuracy of roughly 83% Based on the chart, we can also see that the "Good" prediction has a higher precision rate whereas the "Not Good" prediction has a higher recall rate.

Decision Tree

- Is a class discriminator that recursively partitions the training set until each partition consists entirely or dominantly of examples from one class. Each non-leaf node of the tree contains a split point that is a test on one or more attributes and determines how the data is partitioned. The Decision Tree Model was created with a max depth of 12 from root to leaf. Uses entropy, which measures the impurity to measure the homogeneity of the dataset, which helps determine the best split. We then feed it the training set and then predict y with the test set.

```
: tree_clf = DecisionTreeClassifier(criterion = "entropy",
                                    min_samples_split = 3,
                                    max_depth = 12)

  tree_clf.fit(X_train, y_train)
  y_pred = tree_clf.predict(X_test)
```

-




```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           3       1.00      1.00      1.00       132
           4       0.89      1.00      0.94       131
           5       0.65      0.72      0.68       150
           6       0.68      0.49      0.57       155
           7       0.88      0.94      0.91       116
           8       0.98      1.00      0.99       134

    accuracy                           0.84       818
   macro avg       0.85      0.86      0.85       818
weighted avg       0.84      0.84      0.84       818
```

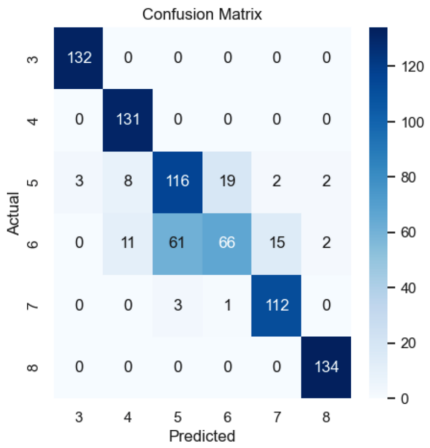
-

- Based on the results the Decision Tree Model does a very good job determining for qualities 3, 4 and 8. Also does well with predicting for quality 7. The model does happen to somewhat struggle a bit distinguishing between quality 5 and 6. Overall, performs relatively well with an accuracy of 84%

K-Nearest Neighbor

- Is a supervised learning classifier that uses proximity to make predictions about the grouping of an individual datapoint. When creating the KNN classifier, we did so where it would find the 2 nearest neighbors from a datapoint and then make a majority vote to classify that datapoint.

```
]: from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors = 2)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```



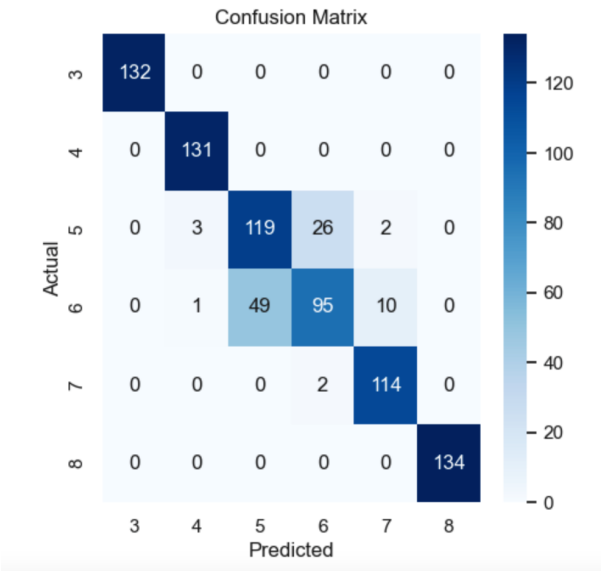

```
KNearestNeighbor Classification Report:
              precision    recall  f1-score   support

           3       0.98      1.00      0.99       132
           4       0.87      1.00      0.93       131
           5       0.64      0.77      0.70       150
           6       0.77      0.43      0.55       155
           7       0.87      0.97      0.91       116
           8       0.97      1.00      0.99       134

    accuracy                           0.84       818
   macro avg       0.85      0.86      0.85       818
weighted avg       0.84      0.84      0.83       818
```

- Based on the results, the K Nearest Neighbor Model does a very good job determining for qualities 3 and 8. Also does well with predicting for quality 4 and 7. However it has difficulty distinguishing the qualities of 5 and 6. Overall, performs relatively well with an accuracy of 84%

Random Forest

- Is an ensemble classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.The Random Forest Model includes 200 trees in the model.We then feed it the training set and then predict y with the test set.

```
model = RandomForestClassifier(n_estimators = 200)
model.fit(X_train,y_train)

y_pred = model.predict(X_test)
```



Random Forest Classification Report:

	precision	recall	f1-score	support
3	1.00	1.00	1.00	132
4	0.97	1.00	0.98	131
5	0.71	0.79	0.75	150
6	0.77	0.61	0.68	155
7	0.90	0.98	0.94	116
8	1.00	1.00	1.00	134
accuracy			0.89	818
macro avg	0.89	0.90	0.89	818
weighted avg	0.89	0.89	0.88	818

- Based on the results, the Random Forest Model does a very good job determining for qualities 3, 4 and 8. Also does very well with predicting quality 7.There is still some struggle distinguishing between quality 5 and 6. Overall, performs very well with an accuracy of 89%.

Summary

The Red Wine Quality Prediction project is a comprehensive exploration into predicting wine quality based on the Red Wine Quality Dataset. The overarching goal is to develop a robust model capable of accurately assessing wine quality on a scale of 1 to 10, utilizing a variety of data mining techniques.

The dataset, obtained from Kaggle and formatted as a CSV file, is rich in quantitative attributes that play pivotal roles in determining wine quality. These attributes encompass factors like fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, and alcohol. The project's initial phase involves preprocessing, where missing values are addressed, and data formats are optimized for enhanced data integrity.

Exploratory Data Analysis (EDA) forms a critical part of the project, shedding light on the dataset's nuances, distributions, and correlations. This phase aims to uncover patterns and potential outliers that could significantly impact the predictive model. With a keen focus on transparency and interpretability, the project emphasizes understanding the dataset's characteristics thoroughly.

Moving forward, the project delves into the modeling and evaluation, selecting and comparing various machine learning models. A number of algorithms were used during the modeling process, which were Logistic Regression, Decision Tree, K Nearest Neighbor and the Random Forest.The project also explores hyperparameter tuning to fine-tune the model's performance. The performance was then evaluated through a confusion matrix which shows how well the models have performed in classifying the target. Some things to consider would be the Precision, Recall and Accuracy.

In the end, after we applied data mining techniques to assess wine quality. The Random Forest Model had the overall best performance in classifying the quality of the red wine, based on the selected features, over the Decision Tree, K Nearest Neighbor, and the Logistic Regression.

- Random Forest - 89%
- Decision Tree - 84% (Better Precision than KNN)
- K Nearest Neighbor - 84% (Better Recall than Decision Tree)
- Logistic Regression - 83%

By deploying the Random Forest Model, we can accurately determine what makes a wine have a quality of 7 or more, being considered as a good tasting wine, and what makes wine have a quality of 4 or less, being considered as a poor tasting wine. With that this allows us to further determine and understand the contents of the wine that allows for such a high rating, which will eventually lead to more successful business outcomes.

References and Appendix

Dataset:
https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

Github:
https://github.com/SarkisM4/Wine_Quality_Prediction/blob/main/Wine%20Quality%20Prediction.ipynb
Chapman, Pete, Clinton Julian, Kerber Randy, Khabaza Thomas and Wirth, Rudiger.
“CRISP-DM 1.0.” CRISP-DM 1.0.pdf

“2. Oversampling.” Imbalance Learn. https://imbalanced-learn.org/stable/over_sampling.html

“1.1.11. Logistic Regression.” Scikit-Learn.
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

“1.6. Nearest Neighbor.” Scikit-Learn. https://scikit-learn.org/stable/modules/neighbors.html#

“1.10. Decision Trees.” Scikit-Learn. https://scikit-learn.org/stable/modules/tree.html

“1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking.” Scikit-Learn.
https://scikit-learn.org/stable/modules/ensemble.html