MSC 641 - Advanced Analytics

Final Project

Wine Quality Prediction

Course Instructor

Dr. Yi Tan

Team Members

Manju Bhargavi Penumarthi

Aditya Gude

Ajay Koteru

James Downard

Project Introduction:

This project aims to develop an analytical classification model to predict the quality of the red variant of the Portuguese "Vinho Verde" style of wine. The data comes from the UCI Machine Learning Repository. The classification model will provide insights into the quality of the wine by predicting whether it is good (1) or bad (0).

Wine Quality Data:

Data Set Characteristics: Multivariate

Number of Attributes: 11 + output attribute

Number of Records: 1599

Number of Missing Values: 0

Data donated: 2009-10-07

Data Attributes:

Fixed Acidity	***Volatile Acidity***
Citric Acid	***Residual Sugar***
Chlorides	***Free sulfur dioxide***
Total Sulfur Dioxide	***Density***
pH	***Sulphates***
Alcohol	***Quality*** (output variable)

Data Summary Statistics:

	Count	Mean	Std	Min	25%	50%	75%	Max
Fixed Acidity	1599	8.319	1.741	4.60	7.1	7.9	9.2	15.9
Volatile Acidity	1599	0.5278	0.1790	0.12	0.39	0.52	0.64	1.58
Citric Acid	1599	0.270	0.194	0	0.09	0.26	0.42	1
Residual Sugar	1599	2.538	1.409	0.9	1.9	2.2	2.6	15.5
Chlorides	1599	0.087	0.047	0.012	0.07	0.079	0.09	0.611
Free sulfur dioxide	1599	15.874	10.46	1	7	14	21	72
Total sulfur dioxide	1599	46.467	32.895	6	22	38	62	289
Density	1599	0.996	0.001	0.99	0.995	0.996	0.997	1
pH	1599	3.311	0.154	2.74	3.21	3.31	3.4	4.01
Sulphates	1599	0.658	0.169	0.33	0.55	0.62	0.73	2
Alcohol	1599	10.422	1.065	8.4	9.5	10.2	11.1	14.9
Quality	1599	5.636	0.807	3	5	6	6	8

The above data represents the descriptive statistics of the red wine dataset. The dataset consists of 1599 samples of red wine with various objective features such as ***fixed acidity, volatile acidity,***

citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulfates, alcohol content**,** and the ***quality*** **of the wine**.

The mean, standard deviation, minimum, maximum, and quartiles (25%, 50%, and 75%) for each of the features are presented in the table. For example, the mean fixed acidity of the wine samples is 8.32, with a standard deviation of 1.74. The minimum fixed acidity is 4.6, and the maximum is 15.9. The 25% quartile of fixed acidity is 7.1, the 50% quartile is 7.9, and the 75% quartile is 9.2.

These descriptive statistics can provide a general overview of the dataset and help identify any potential outliers or unusual patterns. However, they do not provide any insights into the relationships between the features or the quality of the wine. Further analysis and modeling techniques such as regression analysis or decision trees may be necessary to gain more insights and develop accurate predictions of wine quality.

Data Quality Concerns:

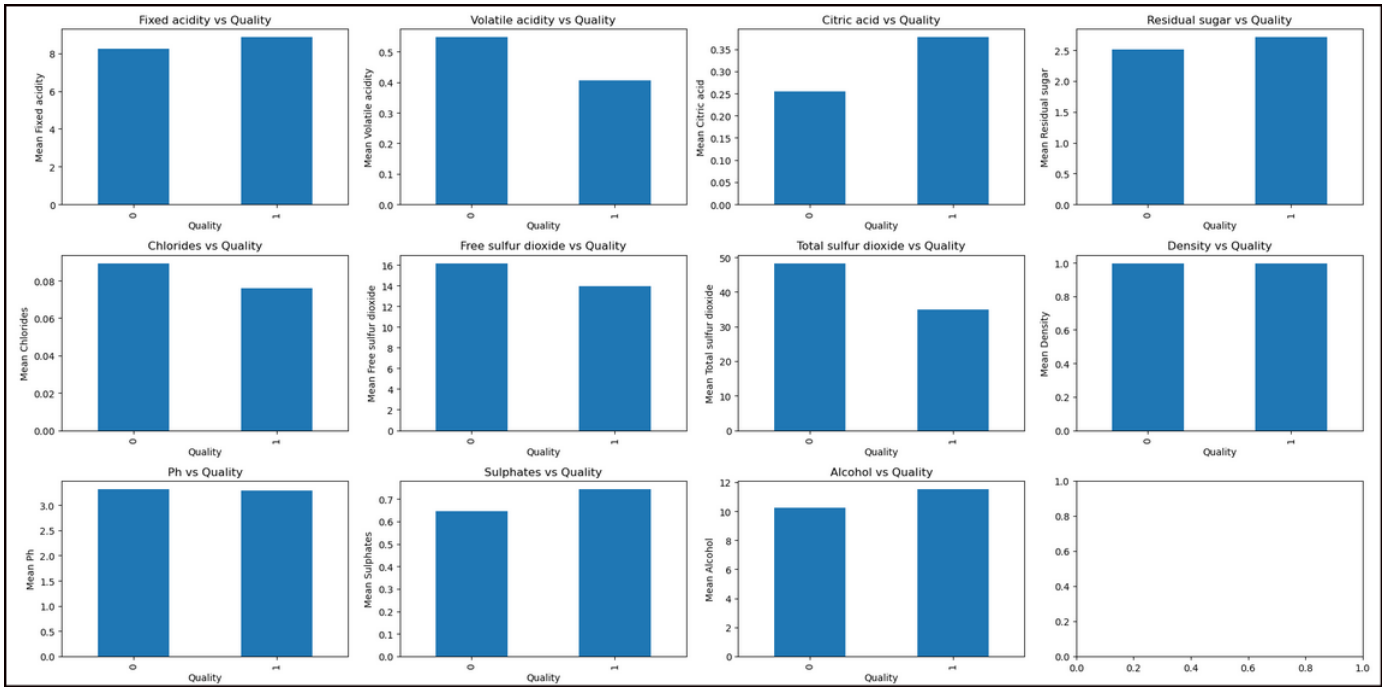
- The different classes of wine are not balanced and are unevenly represented in the data; average-quality wines outweigh both excellent and poor wines. This imbalance will be addressed in the final model using the SMOTE sampling technique.
- None of the features are known to hold more importance than others. Feature selection will need to take place to determine which variables are most important to the model.

Data Preparation:

Converting Target Variable to Binary

1. "Quality" is added as a new column to the data frame "data" with a value of 1 if the value of the original "quality" column is higher than or equal to 7, and 0 otherwise. Here, the "quality" value is transformed into a binary variable that denotes whether the wine is good (quality >=7) or not (quality < 7).
2. Binned quality rankings to either "good" (7, 8) as 1 or "bad" (3, 4, 5, 6) as 0.
3. This new binning is what the models will predict, either 1 or 0.

Visualization (Features vs Target Variable)

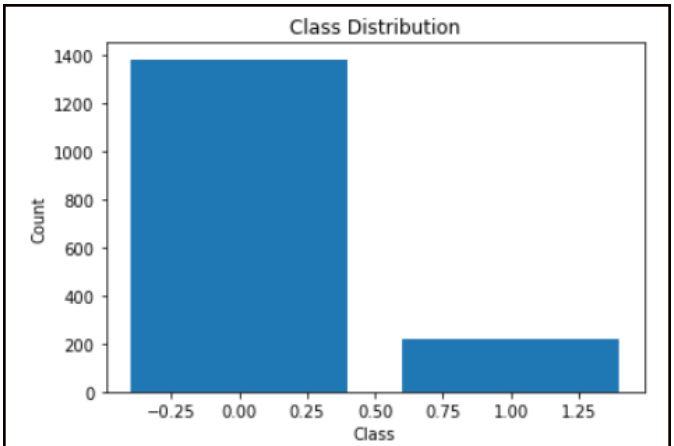


- The bar graph displays the mean value of the target variable ('quality') at each level. These diagrams may be used to comprehend how each independent variable and the target variable are related to one another.

- For instance, by examining the plots, it is clear that "volatile acidity" and "chlorides" have a detrimental impact on wine quality since their mean values are greater for low-quality wines than they are for high-quality wines. Contrarily, as high-quality wines have mean values that are greater than low-quality wines, "citric acid," "sulphates," and "alcohol" have a beneficial impact on wine quality. Based on these graphs, it appears that there is little correlation between the other factors and wine quality.

Checking for Class Imbalance

- We examined the training data for evidence of class imbalance. Class imbalance is when there are not an equal number of samples in each class, which might result in biased model performance.
- The output indicates that the training data is skewed toward the negative class (quality = 0), with 1382 samples having a value of 0 and 217 samples having a value of 1.
- The output verifies that the training data is unbalanced since 86.4% of it belongs to the negative class (quality=0) and only 13.5% to the positive class (quality=1).
- To avoid the model being biased toward the dominant class, it is crucial to resolve the class imbalance in the training data. Class imbalance can be addressed using a variety of methods, including undersampling the dominant class, oversampling the minority class, or combining both.




```
Class Distribution:
0    1382
1     217
Name: quality, dtype: int64

Percentage of Each Class:
0    86.429018
1    13.570982
Name: quality, dtype: float64
```

Data splitting

- Made a new data frame called "x" that has every column from the original data frame called "data," with the exception of the "quality" column.
- Produced a new series "y" that only includes the "quality" column from the original data frame "data."
- Utilized the scikit-learn train_test_split tool to divide the data into training and testing sets. This is done to provide two distinct datasets that may be utilized for our machine learning model's training and evaluation. The data is divided so that 80% is used for training and 20% is utilized for testing. To ensure that the split can be replicated, a random state of 42 is chosen.

The variables that follow are:

- x: A data frame that includes every column from the original data frame's "data" except for the 'quality' column.
- y: A series that just consists of the 'quality' column from the first data frame "data."
- x_train: A data frame used to train the machine learning model that contains 80% of the feature variables from "x."
- 80% of the data used in x_train is represented by a series called y_train that contains the equivalent "quality" values.
- A data frame called x_test, which is used to test the machine learning model, contains 20% of the feature variables from "x."
- y_test: A series that represents the "*quality"* values for the 20% of the data utilized in x_test.

Handling Class Imbalance using SMOTE

- SMOTE (Synthetic Minority Over-sampling Technique) is an algorithm that analyzes training data. SMOTE is a popular method for creating synthetic samples that are used to oversample the minority class in unbalanced datasets.
- The results demonstrate that there are now 2218 samples total, with 1109 samples for the negative (quality = 0) and 1109 samples for the positive (quality = 1).
- By giving the model more samples from the minority class to learn from, handling the class imbalance in the training data using SMOTE can enhance the model's performance.

Data Scaling

- Utilized Scikit-Learn's StandardScaler to scale the data. By doing this, the feature variables are normalized to have a mean of 0 and a standard deviation of 1. This ensures that each feature is of comparable size and prevents any one feature from having an outsized impact on the performance of the model.

Principal Component Analysis (PCA)

- PCA can assist in reducing the computational complexity of the model and avoiding overfitting by reducing the dimensionality of the data. To prevent losing too much data or adding too much noise to the data, it's crucial to choose the number of principal components carefully.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.90)
x_train = pca.fit_transform(x_train)
x_test = pca.transform(x_test)
```

- n_components = 0.90, which indicates that we want to keep enough principal components to account for at least 90% of the variance in the data, creating an instance of the PCA class.

```
sum(pca.explained_variance_ratio_)
```

```
0.9081950548246704
```

```
pca.explained_variance_ratio_
```

```
array([0.28017635, 0.17565914, 0.13957862, 0.11081885, 0.09020956,
       0.05908763, 0.05266491])
```

- Explained Variance ratio determines the overall variance explained by the principal components that were kept after the data were subjected to PCA. The output value of

0.9081 shows that roughly 90% of the variance in the original data is explained by the principal components that were kept.

- Each principal component's explained variance ratio displays the percentage of the overall variance that each component contributes to explaining. The result is an array of seven values, each of which represents a principal component that was kept.

Algorithms

K-Nearest Neighbors (KNN) Classifier:

The K-Nearest Neighbors (KNN) classifier is a type of supervised learning algorithm used for classification tasks. In KNN, the classification of a new data point is based on the class of the K nearest data points in the training dataset.

The working of KNN can be explained as follows:

a. First, the algorithm takes in the training data, which consists of a set of data points with known class labels.

b. When a new data point is presented for classification, the algorithm measures its distance to all the other data points in the training set, using a distance metric such as Euclidean distance or Manhattan distance.

c. The K closest data points to the new point are selected based on the distance metric. The value of K is a hyperparameter that needs to be selected based on the problem at hand.

d. Once the K closest data points are identified, the class of the new data point is determined by taking the most frequent class among the K nearest data points. In

other words, the class with the highest frequency among the K neighbors is assigned to the new data point.

e. Finally, the algorithm outputs the predicted class label for the new data point.

Decision Tree Classifier:

The Decision Tree Classifier is a type of supervised learning algorithm used for classification tasks. The algorithm creates a decision tree model based on the training data, which is then used to classify new data points.

The working of the Decision Tree Classifier can be explained as follows:

a. First, the algorithm takes in the training data, which consists of a set of data points with known class labels.

b. The algorithm then selects the feature that best separates the data based on an impurity metric such as entropy or Gini impurity. The feature with the highest information gain or lowest impurity is selected as the root node of the decision tree.

c. The training data is split based on the selected feature into two or more subsets, with each subset representing a branch of the decision tree.

d. The process is repeated recursively for each subset, selecting the best feature to split the data until a stopping criterion is met. This may be a maximum depth of the tree, a minimum number of samples per leaf node, or a minimum decrease in impurity.

e. Once the decision tree is built, new data points can be classified by traversing the tree from the root node to a leaf node. At each node, the value of the selected feature is compared to a threshold, and the appropriate branch is followed until a leaf node is reached. The class label assigned to the leaf node is then assigned to the new data point.

Random Forest Classifier:

The Random Forest Classifier is a type of supervised learning algorithm used for classification tasks. The algorithm creates an ensemble of decision trees based on the training data, which are then used to classify new data points.

The working of the Random Forest Classifier can be explained as follows:

1. First, the algorithm takes in the training data, which consists of a set of data points with known class labels.
2. The algorithm then creates multiple decision trees using a technique called bagging. Bagging involves randomly sampling the training data with replacement to create multiple subsets of the data, and then building a decision tree on each subset.
3. Each decision tree is built using the same process as the Decision Tree Classifier, selecting the feature that best separates the data based on an impurity metric such as entropy or Gini impurity.
4. Once all the decision trees are built, new data points can be classified by running them through each tree and taking the majority vote on the predicted class labels.
5. Finally, the algorithm outputs the predicted class label for the new data point based on the majority vote.

Support Vector Machine (SVM) Classifier:

The Support Vector Machine (SVM) Classifier is a type of supervised learning algorithm used for classification tasks. The algorithm works by finding the hyperplane that best separates the data into different classes.

The working of the SVM Classifier can be explained as follows:

1. First, the algorithm takes in the training data, which consists of a set of data points with known class labels.
2. The algorithm then tries to find the hyperplane that best separates the data into different classes. A hyperplane is a decision boundary that separates the data points with the maximum margin.
3. If the data cannot be linearly separated, the algorithm maps the data into a higher-dimensional feature space using a kernel function. In this space, the data may be linearly separable, and the hyperplane can be found.
4. The hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. The margin ensures that the classifier is robust to noise and generalizes well to new data.
5. Once the hyperplane is found, new data points can be classified by determining which side of the hyperplane they lie on.
6. Finally, the algorithm outputs the predicted class label for the new data point based on which side of the hyperplane it falls on.

Evaluation Metrics Comparison Table:

	Accuracy	**Precision Score**	**Recall score**	**F1 Score**
KNN	0.8843	0.6666	0.4255	0.5194
Decision Tree	0.8406	0.4630	0.5320	0.4950
Random Forest	0.9062	0.8148	0.4680	0.5945
SVC	0.8687	0.6470	0.2340	0.3437

From the above values, we can observe that the Random Forest Classifier performed well in predicting the wine quality.

Hyperparameter Tuning:

- To discover the ideal collection of hyperparameters for the above obtained best model, the GridSearchCV function is used for hyperparameter tweaking.
- The best hyperparameters identified by GridSearchCV are displayed below:

  max_depth = 15 ***max_features = sqrt***

  min_samples_leaf = 1 ***min_samples_split = 2***

  n_estimators = 200

Best Model:

- A random forest classification model employing the best hyperparameters was found by a grid search on the training data. The model is then applied to forecast the test data, and its performance is assessed using accuracy, precision, recall, and F1 scores.
- The model's acquired results reveal that it has:

  Accuracy = 0.903 ***Precision score = 0.786***

  Recall score = 0.468 ***F1 score = 0.587***

Prediction of Quality on a New Test Dataset:

- The most effective random forest classifier model that was trained on a dataset to predict the quality of red wine is used here. An overall assessment of the classifier model's performance on the test data is provided in this report.

	Precision	Recall	F1 score	Support
0	0.92	0.99	0.95	273
1	0.85	0.49	0.62	47
Accuracy			0.91	320
Macro Avg	0.88	0.74	0.79	320
Weighted Avg	0.91	0.91	0.90	320

- The accuracy, recall, and F1 scores for each class are included in the report, along with the support (number of samples) for each class.
- The accuracy score for the class 0 is 0.92, meaning that 92% of all samples projected to be negative were accurately categorized. According to the recall score of 0.99, 99% of the real bad quality samples were properly identified. The harmonic mean of the accuracy and recall scores is 0.95, and this is the F1 score.
- The accuracy score for the class 1 is 0.85, which means that, of all the samples expected to be positive, 85% were properly identified. Only 49% of the real good quality samples, those with a recall score of 0.49 were correctly identified. The harmonic mean of the accuracy and recall scores is 0.62, and this is the F1 score.
- Precision, recall, and F1-score had macro-averages of 0.88, 0.74, and 0.79 for both classes. These scores have weighted averages of 0.91, 0.91, and 0.90, respectively, which account for the dataset's class imbalance.

- The model performs well overall, with excellent precision and recall scores, in predicting wine quality.

Bibliography ~

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.

Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.