WINE QUALITY PREDICTION USING REGRESSION OR CLASSIFICATION

A SURVEY REPORT FROM

HALDIA INSTITUTE OF TECHNOLOGY, HALDIA

Submitted by

AMITAVA RANA

Roll No: 22/IT/027 (10300222027)

ELECTRONICS AND COMMUNICATION ENGINEERING

Trainee (Jan To Feb) 2025

DataSpace Academy

EN-34, Seven Hills Building, 3rd floor EN Block, Street No. 8, More, near WEBEL, Sector V, Bidhannagar, Kolkata, West Bengal 70009

BONAFIDE CERTIFICATE

This is to certify that this project report entitled “**Wine Quality Prediction using Regression and Classification”** submitted to DataSpace Academy, is a Bonafide record of work done by **AMITAVA RANA** who carried out the project work at "**DataSpace Academy**", through this Industrial training program under my supervision and guidance in fulfillment of requirements from 6th Jan 2025 to 12th Feb 2025 (6 Weeks).

Mr. Bipul Nath
Data Space Academy

Acknowledgement

This is to declare that this report has been written by me. No part of the report is plagiarized from other sources. All information included from other sources has been duly acknowledged. I affirm that if any part of the report is found to be plagiarized, I shall take full responsibility for it.

Name of Author(s): **AMITAVA RANA**

TABLE OF CONTENTS

INTRODUCTION

The quality of the wine is a very important part for the consumers as well as the manufacturing industries. Industries are increasing their sales using product quality certification. Nowadays, all over the world wine is a regularly used beverage and the industries are using the certification of product quality to increases their value in the market. Previously, testing of product quality will be done at the end of the production, this is time taking process and it requires a lot of resources such as the need for various human experts for the assessment of product quality which makes this process very expensive.

Every human has their own opinion about the test, so identifying the quality of the wine based on humans experts it is a challenging task. There are several features to predict the wine quality but the entire features will not be relevant for better prediction. The research aims to what wine features are important to get the promising result by implementing the machine learning classification algorithms such as Support Vector Machine (SVM), Naïve Bayes (NB), and Artificial Neural Network (ANN), using the wine quality dataset. The wine quality dataset is available on the UCI machine learning repository (Cortez et al., 2009).

The dataset has two files red wine and white wine variants of the Portuguese “Vinho Verde” wine. It contains a large collection of datasets that have been used for the machine learning community. The red wine dataset contains 1599 instances and the white wine dataset contains 4898 instances. Both files contain 11 input features and 1 output feature. Input features are based on the physicochemical tests and output variable based on sensory data is scaled in 11 quality classes from 0 to 10 (0-very bad to 10-very good). 2 Feature selection is the popular data preprocessing step for generally . To build the model it selects the subset of relevant features. According to the weighted of the relevance of the features, and with relatively low weighting features will be removed. This process will simplify the model and reduce the training time, and increase the performance of the model (Panday et al., 2018). We pay attention to feature selection is also the study direction. To evaluate our model, accuracy, precision, recall, and f1 score are good indicators to evaluate the performance of the model.

ABSTRACT

As a subfield of Artificial Intelligence (AI), Machine Learning (ML) aims to understand the structure of the data and fit it into models, which later can be used in unseen data to achieve the desired task. ML has been widely used in various sectors such as in Businesses, Medicine, Astrophysics, and many other scientific problems. Inspired by the success of ML in different sectors, here, we use it to predict the wine quality based on the various parameters. Among various ML models, we compare the performance of Ridge Regression (RR)Support Vector Machine (SVM), Gradient BoostingRegresso (GBR), and multi-layer Artificial Neural Network (ANN) to predict the wine quality. Multiple parameters that determine the wine quality areanalysis shows that GBR surpasses all other models' performance with MSE, R, and MAPE of 0.3741, 0.6057, and 0.0873 respectively. This work demonstrates,how statistical analysis can be used to identify the components that mainly control the wine quality prior to the production. This will help wine manufacturer to control the quality prior to the wine production.

Literature Review

A wide range of machine learning algorithms is available for the learning process. This section describes the classification algorithms used in wine quality prediction.

Classification algorithm

Support Vector Machine

The support vector machine (SVM) is the most popular and most widely used machine learning algorithm. It is a supervised learning model that can perform classification and regression tasks. However, it is primarily used for classification problems in machine learning (Gandhi, 2018). The SVM algorithm aims to create the best line or decision boundary that can separate n-dimensional space into classes. So we can put the new data points easily in the correct groups. This best decision boundary is called a hyperplane The support vector machine selects the extreme data points that helping to create the hyperplane. In Figure 1, two different groups are classified by using the decision boundary or hyperplane: The SVM model is used for both non-linear and linear data. It uses a nonlinear mapping to convert the main preparing information into a higher measurement. The model searches for the linear optimum splitting hyperplane in this new measurement. A hyperplane can split the data into two classes with an appropriate nonlinear mapping to suitably high measurements and for the finding, this hyperplane SVM uses the support vectors and edges (J. Han et al., 2012). The SVM model is a representation of the models as a point in space, the different classes are isolated by the gap to mapped with the aim that instances are wide as would be careful. The model can perform out a nonlinear form of classification (Kumar et al., 2020)

Naive Bayesian

The naive Bayesian is the simple supervised machine learning classification algorithm based on the Bayes theorem. The algorithm assumes that the feature conditions are independent of the given class (Rish, 2001). The naive Bayes algorithm helps to build fast machine learning models that can make a fast prediction. The algorithm finds whether a particular portion has a spot by a particular class it utilizes the probability of likelihood (Kumar et al., 2020)

Artificial Neural Network

The artificial neural network is a collection of neurons that can process information. It has been successfully applied to the classification task in several industries including the Commercial, Industrial, and scientific filed. The algorithm model is a connection between the neurons that are interconnected with the input layer, a hidden layer, and an output layer. The neural network is constant because while an element of the neural network is failing, it can continue its parallel nature without any difficulties.

Approach

The approach to Wine Quality Prediction in the Cloud involves multiple steps, integrating data preprocessing, machine learning model training, and cloud deployment for real-time or batch inference. Here's a structured approach:

Data Collection & Storage:

- Store the dataset in cloud-based storage like AWS S3, Google Cloud Storage, or Azure Blob Storage.
- The dataset contains physicochemical properties of wine, such as acidity, pH, alcohol content, sugar levels etc.

Data Preprocessing:

- Use Google Colab to process and cleaning the data using python and it's libraries.
- Use cloud-based data processing tools like Apache Spark, AWS Glue, or Google Dataflow to clean and normalize data.
- Handling missing values, feature scaling, and encoding categorical variables.

Model Selection & Training:

Different machine learning algorithms can be used:

- **Linear Regression (**for continuous quality scores**)**
- **Random Forest & Decision Trees (**for feature importance and interpretability)
- **Support Vector Machine (SVM)** (for high-dimensional feature space)
- **Deep Learning (Neural Networks**) (for complex patterns and interactions)

Model Deployment:

Deploy the trained model using:

- **Serverless APIs** (AWS Lambda, Google Cloud Functions, Azure Functions)
- **Containerized Microservices** (Docker + Kubernetes)
- **Web Applications** (Flask or FastAPI hosted on cloud platforms)

TOOLS & TECHNIQUES

Predicting wine quality based on its physicochemical properties is a regression/classification problem. Given a dataset with features such as acidity, sugar content, alcohol percentage, and pH levels, the goal is to predict the quality score of the wine. Traditional methods rely on human experts, but automated models improve efficiency and consistency.

1. Tools:-

Python Programming Language: Python served as the primary language for implementing the entire project due to its extensive library support, ease of use, and versatility in data science and machine learning applications.

Jupyter Notebook: This interactive computing environment was used for code development, visualization, and experimentation, facilitating iterative model improvements and clear data analysis.

Google Colab: For model training and experiments, Google Colab provided access to powerful computational resources, such as GPUs and TPUs, which significantly accelerated model training times.

Data Processing Tools: Pandas, NumPy, scikitlearn

Data Visualization: Matplotlib**,** Seaborn,

Cloud Platforms: AWS ,Lambda, S3,Google Cloud (Vertex AI, Cloud Run, ,Microsoft Azure (ML Studio, Blob Storage)

2. Techniques:-

In Wine Quality Prediction in the Cloud, various techniques are used for data processing, model training, and deployment. These techniques can be broadly categorized into Machine Learning Techniques and Cloud Computing Techniques:

1. **Regression Techniques**:
   - Linear Regression: Determines the relationship between wine attributes and quality score.
   - Ridge & Lasso Regression: Used to prevent overfitting by applying regularization.
2. **Classification Techniques**:
   - Decision Trees: Simple yet powerful for classifying wine into quality categories.
   - Random Forest: An ensemble of decision trees that improves accuracy.
   - Support Vector Machines (SVM): Effective in high-dimensional datasets for classification.
   - Neural Networks (Deep Learning): Captures complex patterns for more precise predictions.
3. **Clustering Techniques**:
   - K-Means Clustering: Groups wines based on similar physicochemical properties.
   - Hierarchical Clustering: Builds a hierarchy of clusters to analyze wine attributes.
4. **Feature Engineering & Dimensionality Reduction:**
   - Principal Component Analysis (PCA): Reduces feature space while preserving variance.
   - Feature Scaling (Standardization/Normalization): Improves model performance.

IMPLEMENTATION

The implementation of the Wine Quality Prediction or Classification project involved creating a robust model using the regression or classification model to classify or predict the wine quality. The key steps in the implementation are summarized below:

Data Preprocessing

There was two datasets which is as first concatenated. The concatenated dataset contained 12 columns (6497 values in every column). To prepare the data:

- Feature Engineering: Techniques like handling missing values, duplicate data, and categorical feature were applied to increase data diversity and improve model generalization.
- Exploratory Data Analysis: Plotting the data to get visual idea of the features outcome.
- Data Splitting: The data was divided into training, validation, and testing sets for unbiased evaluation and scaled the data in a common generalizing range.

Model Selection and Transfer Learning

The RandomForestClassifier was chosen for its balance of efficiency and accuracy. The model used the feature input data to get the insight of the wine quality and predict as accurate as possible.

Model Training & Model Evaluation

The model was trained using:

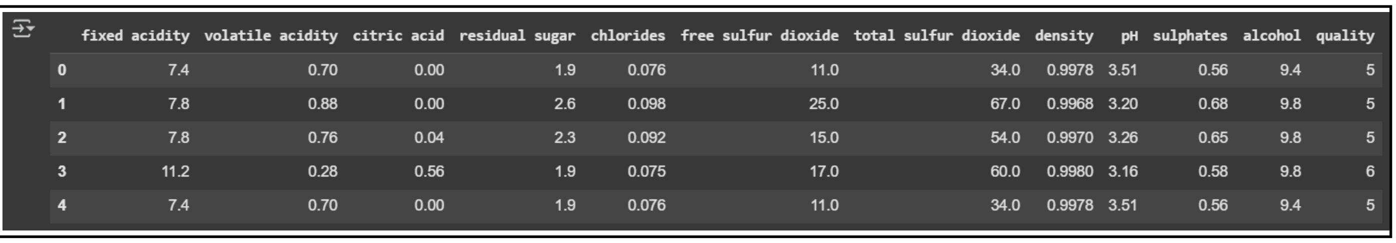
- RandomForestClassifier classification model.
- Hyperparameter Tuning for optimal learning rate, batch size, and dropout rates.
- Accuracy, Precision, Recall, and F1-score metrics.

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

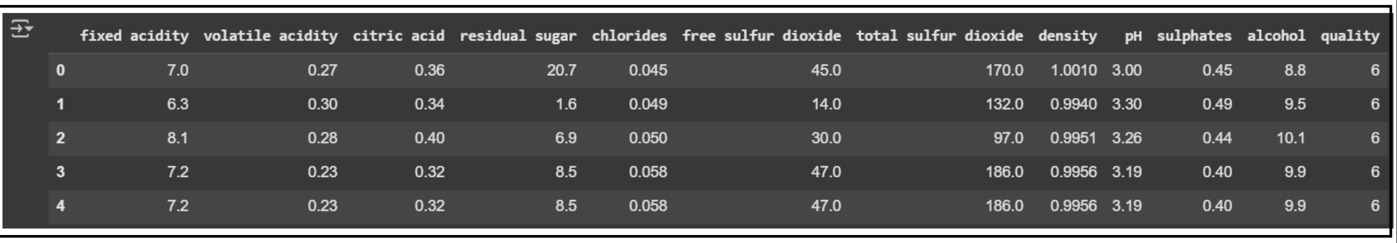
Data Collection

```
df1 = pd.read_csv('/content/winequality-red.csv',sep =';')
df1.head()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

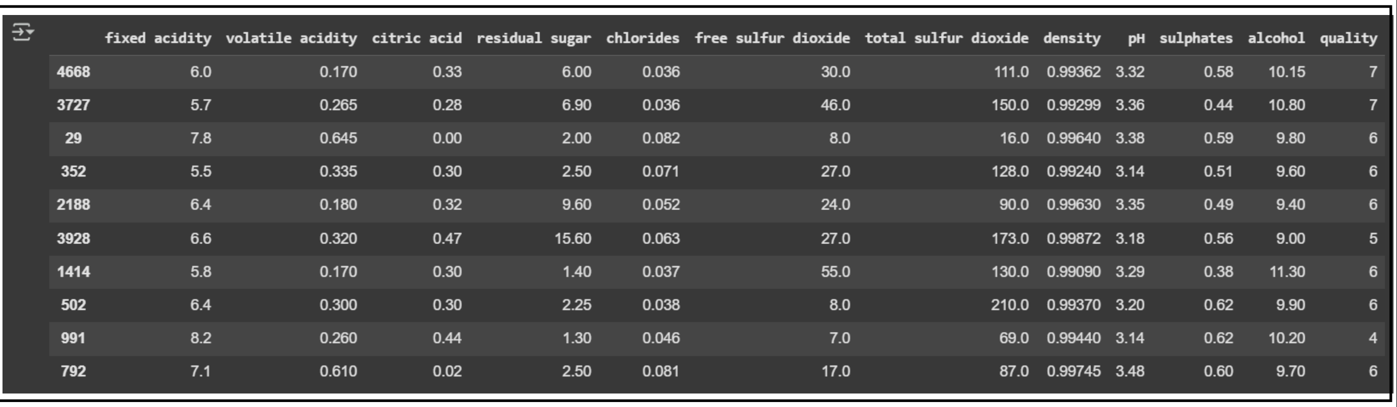
```
df2 = pd.read_csv('/content/winequality-white.csv',sep =';')
df2.head()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

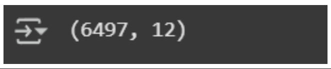
Feature Engineering

```
total = [df1,df2]
df = pd.concat(total)
df.sample(10)
```



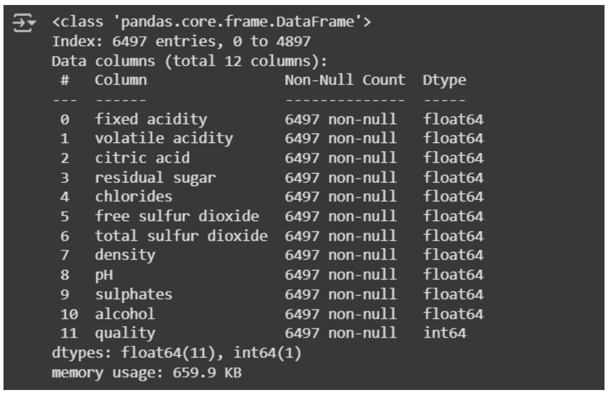
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
4668	6.0	0.170	0.33	6.00	0.036	30.0	111.0	0.99362	3.32	0.58	10.15	7
3727	5.7	0.265	0.28	6.90	0.036	46.0	150.0	0.99299	3.36	0.44	10.80	7
29	7.8	0.645	0.00	2.00	0.082	8.0	16.0	0.99640	3.38	0.59	9.80	6
352	5.5	0.335	0.30	2.50	0.071	27.0	128.0	0.99240	3.14	0.51	9.60	6
2188	6.4	0.180	0.32	9.60	0.052	24.0	90.0	0.99630	3.35	0.49	9.40	6
3928	6.6	0.320	0.47	15.60	0.063	27.0	173.0	0.99872	3.18	0.56	9.00	5
1414	5.8	0.170	0.30	1.40	0.037	55.0	130.0	0.99090	3.29	0.38	11.30	6
502	6.4	0.300	0.30	2.25	0.038	8.0	210.0	0.99370	3.20	0.62	9.90	6
991	8.2	0.260	0.44	1.30	0.046	7.0	69.0	0.99440	3.14	0.62	10.20	4
792	7.1	0.610	0.02	2.50	0.081	17.0	87.0	0.99745	3.48	0.60	9.70	6

```
df.shape
```



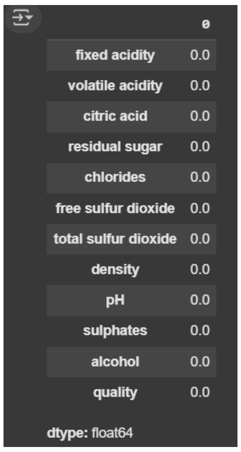
```
(6497, 12)
```

```
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
Index: 6497 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  quality               6497 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 659.9 KB
```

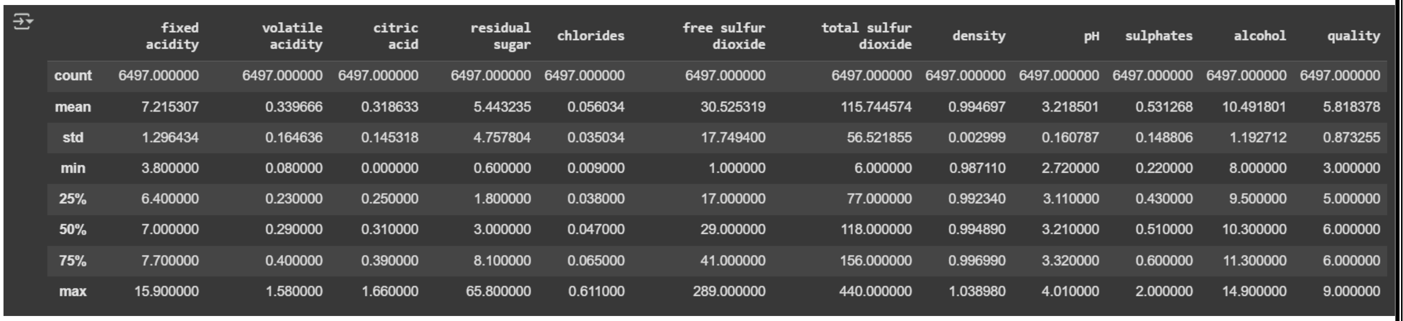
```
df.isnull().sum()/len(df)*100
```



	0
fixed acidity	0.0
volatile acidity	0.0
citric acid	0.0
residual sugar	0.0
chlorides	0.0
free sulfur dioxide	0.0
total sulfur dioxide	0.0
density	0.0
pH	0.0
sulphates	0.0
alcohol	0.0
quality	0.0

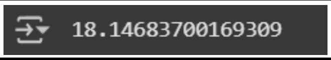
dtype: float64

```
df.describe()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	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	115.744574	0.994697	3.218501	0.531268	10.491801	5.818378
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712	0.873255
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

```
df.duplicated().sum()/len(df)*100
```

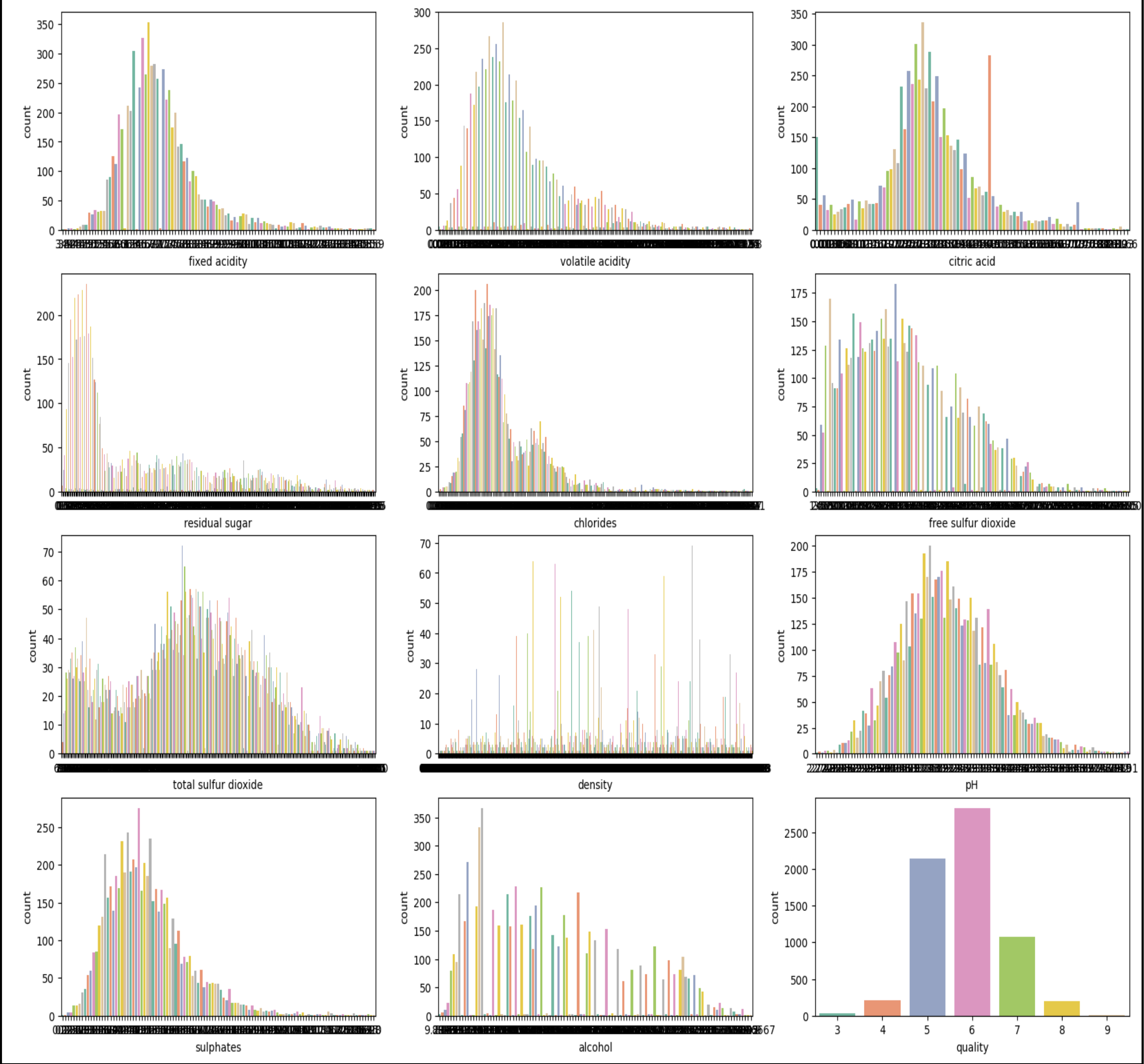


```
18.14683700169309
```

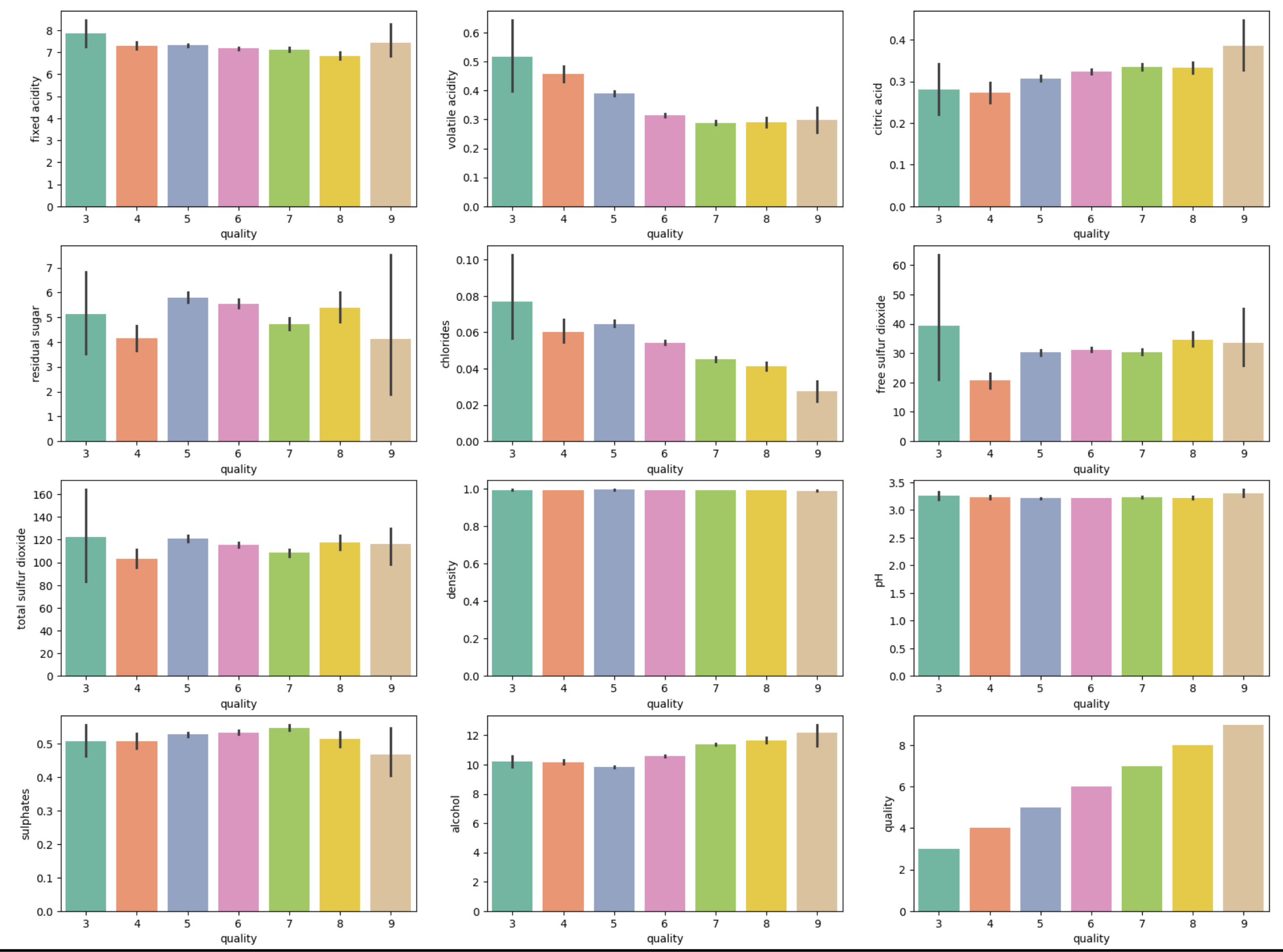
```
#As Duplicated_values> 18%, we are not going to drop it because it will make loss of almost 20% of the total data which affects the accurcy of our regression or classifier model
```

Exploratory Data Analysis

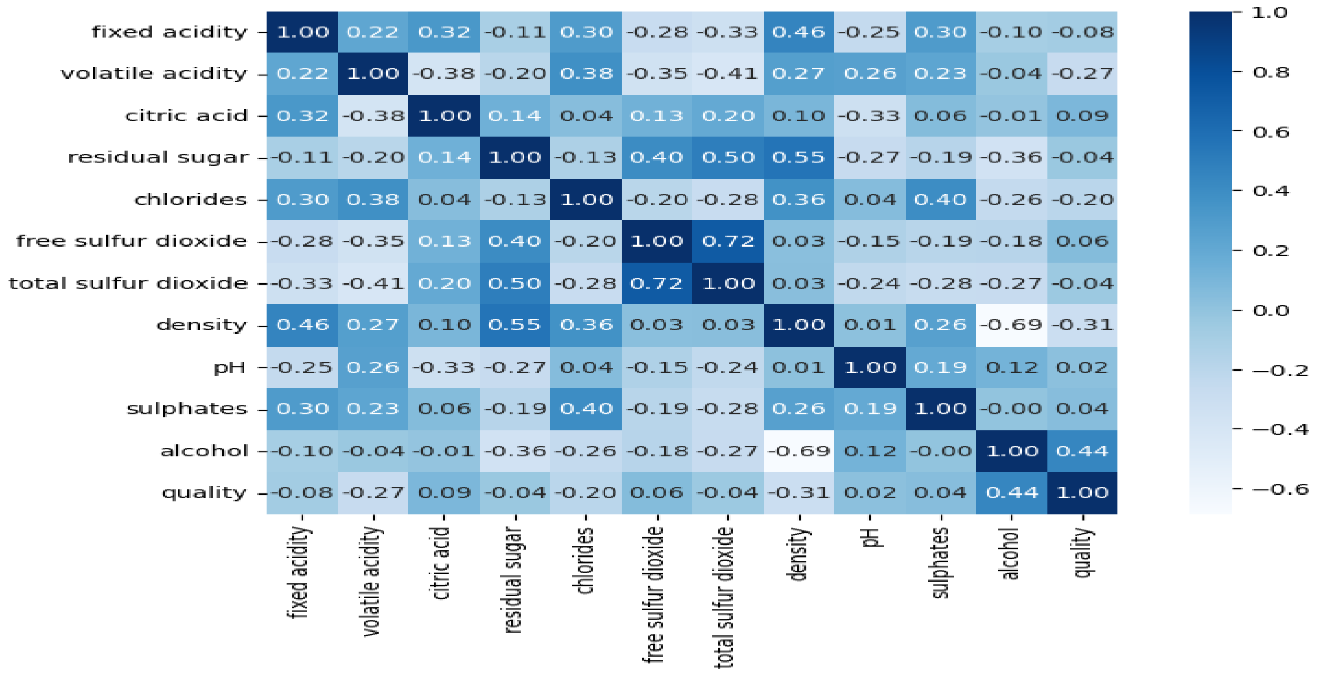
```
plt.figure(figsize=(20,15))
for i,col in enumerate(df):
  plt.subplot(4,3,i+1)
  sns.countplot(x=col,data=df,palette='Set2')
```



```
plt.figure(figsize=(20,15))
for i,col in enumerate(df):
  plt.subplot(4,3,i+1)
  sns.barplot(x='quality',y = col,data=df,palette='Set2')
```



```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),cbar = True, square = True,fmt='.2f',annot=True, cmap='Blues')
```



Feature Selection

```
# Defining Features
x = df.drop('quality',axis=1)
y = df['quality'].apply(lambda y_value: 1 if y_value>=7 else 0)
```

Train-Test Split

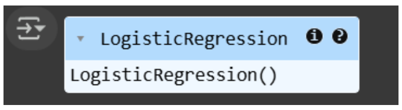
```
# Train-test split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=3)
```

Feature Scaling

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

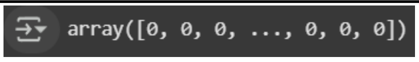
Logistic Regression Model

```
# Model Creation (Logistic Regression)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train,y_train)
```



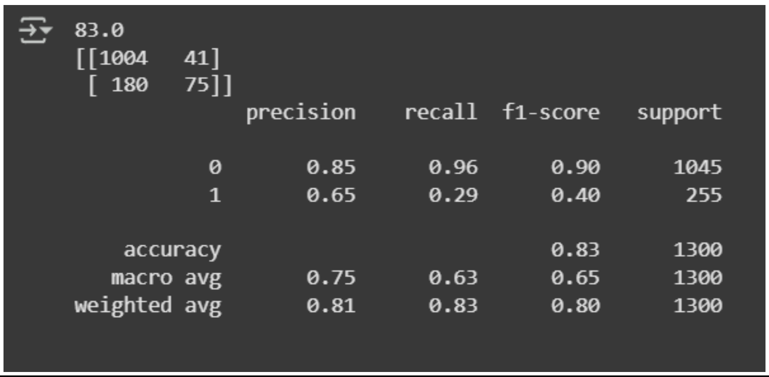
```
LogisticRegression
LogisticRegression()
```

```
# Model Testing (LR)
y_pred = model.predict(x_test)
y_pred
```



```
array([0, 0, 0, ..., 0, 0, 0])
```

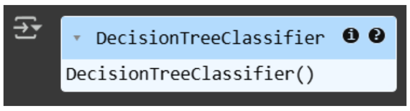
```
# Checking the acuuracy score (LR)
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
acc = accuracy_score(y_test,y_pred)*100
cm = confusion_matrix(y_test,y_pred)
cls_rpt = classification_report(y_test,y_pred)

print(acc)
print(cm)
print(cls_rpt)
```



```
83.0
[[1004   41]
 [ 180   75]]
              precision    recall  f1-score   support

           0       0.85      0.96      0.90      1045
           1       0.65      0.29      0.40       255

    accuracy                           0.83      1300
   macro avg       0.75      0.63      0.65      1300
weighted avg       0.81      0.83      0.80      1300
```

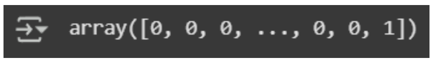
DicisionTreeClassifier Model

```
# Model Creation (DicisionTreeClassifier)
from sklearn.tree import DecisionTreeClassifier
model_dt = DecisionTreeClassifier()
model_dt.fit(x_train,y_train)
```



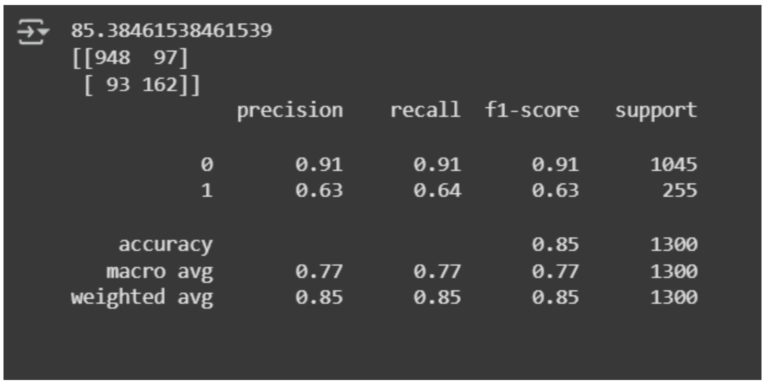
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
# Model Testing (DTC)
y_pred_dt = model_dt.predict(x_test)
y_pred_dt
```



```
array([0, 0, 0, ..., 0, 0, 1])
```

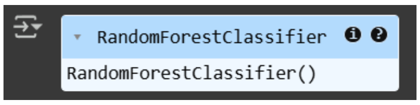
```
# Checking the acuuracy score (DTC)
acc_dt = accuracy_score(y_test,y_pred_dt)*100
cm_dt = confusion_matrix(y_test,y_pred_dt)
cls_rpt_dt = classification_report(y_test,y_pred_dt)

print(acc_dt)
print(cm_dt)
print(cls_rpt_dt)
```



```
85.38461538461539
[[948  97]
 [ 93 162]]
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      1045
           1       0.63      0.64      0.63       255

    accuracy                           0.85      1300
   macro avg       0.77      0.77      0.77      1300
weighted avg       0.85      0.85      0.85      1300
```

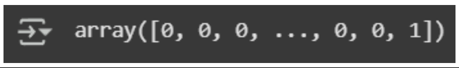
RandomForestClassifier Model

```
# Model Creation (RandomForestClassifier)
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier()
model_rf.fit(x_train,y_train)
```



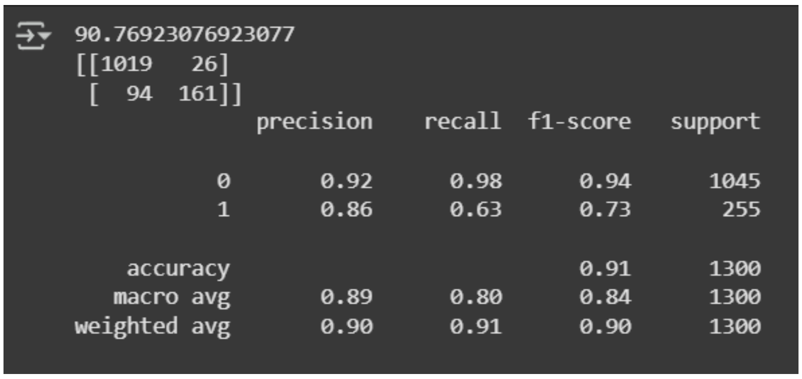
```
RandomForestClassifier
RandomForestClassifier()
```

```
# Model Testing (RFC)
y_pred_rf = model_rf.predict(x_test)
y_pred_rf
```



```
array([0, 0, 0, ..., 0, 0, 1])
```

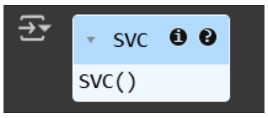
```
# Checking the acuuracy score (RFC)
acc_rf = accuracy_score(y_test,y_pred_rf)*100
cm_rf = confusion_matrix(y_test,y_pred_rf)
cls_rpt_rf = classification_report(y_test,y_pred_rf)

print(acc_rf)
print(cm_rf)
print(cls_rpt_rf)
```



```
90.76923076923077
[[1019   26]
 [  94  161]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.94      1045
           1       0.86      0.63      0.73       255

    accuracy                           0.91      1300
   macro avg       0.89      0.80      0.84      1300
weighted avg       0.90      0.91      0.90      1300
```

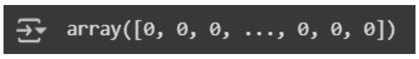
SVC Model

```
# Model Creation (SVC)
from sklearn.svm import SVC
model_svc = SVC()
model_svc.fit(x_train,y_train)
```



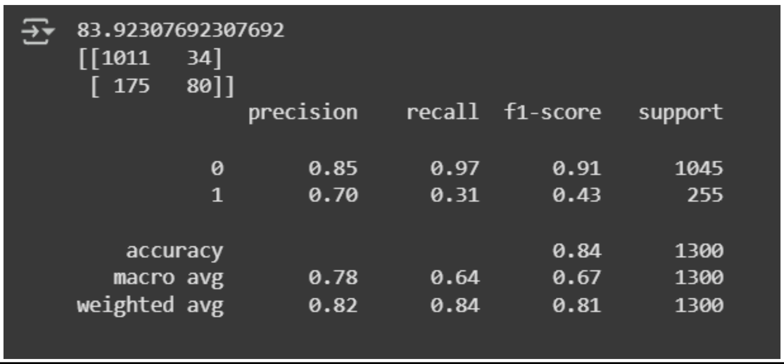
```
SVC
SVC()
```

```
# Model Testing (SVC)
y_pred_svc = model_svc.predict(x_test)
y_pred_svc
```



```
array([0, 0, 0, ..., 0, 0, 0])
```

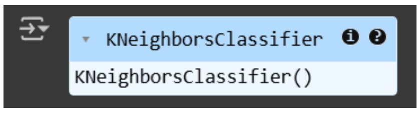
```
# Checking the acuuracy score (SVC)
acc_svc = accuracy_score(y_test,y_pred_svc)*100
cm_svc = confusion_matrix(y_test,y_pred_svc)
cls_rpt_svc = classification_report(y_test,y_pred_svc)

print(acc_svc)
print(cm_svc)
print(cls_rpt_svc)
```



```
83.92307692307692
[[1011   34]
 [ 175   80]]
              precision    recall  f1-score   support

           0       0.85      0.97      0.91      1045
           1       0.70      0.31      0.43       255

    accuracy                           0.84      1300
   macro avg       0.78      0.64      0.67      1300
weighted avg       0.82      0.84      0.81      1300
```

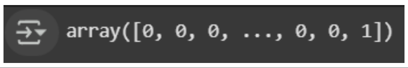
KNeighborsClassifier Model

```
# Model Creation (KNN)
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier()
model_knn.fit(x_train,y_train)
```



```
KNeighborsClassifier
KNeighborsClassifier()
```

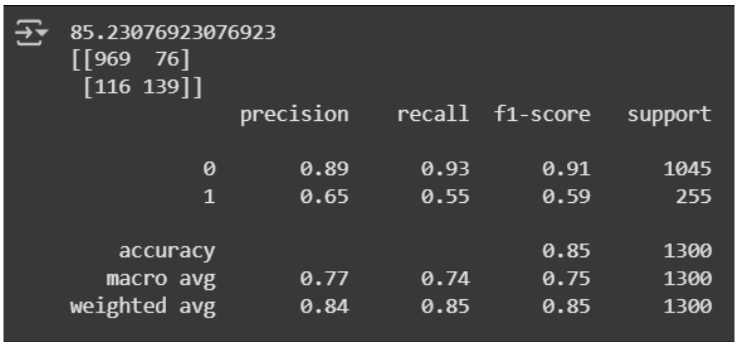
```
# Model Testing (KNN)
y_pred_knn = model_knn.predict(x_test)
y_pred_knn
```



```
array([0, 0, 0, ..., 0, 0, 1])
```

```
# Checking the acuuracy score (KNN)
acc_knn = accuracy_score(y_test,y_pred_knn)*100
cm_knn = confusion_matrix(y_test,y_pred_knn)
cls_rpt_knn = classification_report(y_test,y_pred_knn)
```

```
print(acc_knn)
print(cm_knn)
print(cls_rpt_knn)
```

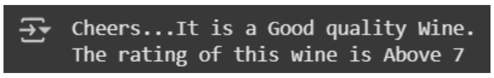


```
85.23076923076923
[[969  76]
 [116 139]]
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      1045
           1       0.65      0.55      0.59       255

    accuracy                           0.85      1300
   macro avg       0.77      0.74      0.75      1300
weighted avg       0.84      0.85      0.85      1300
```

# As of the accuracy score of the RandomForestClassifier is much better than the other regression or classifier models, we are going to use this one for prediction the Wine quality.

Model Deployment

```
# Data Input
input_data = (7.3,0.65,0.0,1.2,0.065,15.0,21.0,0.9946,3.39,0.47,10.0)

#change the input data as numpy array
input_data_as_numpy_array = np.asarray(input_data)

#reshape the numpy array as we are predicting for one datapoint
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

#scaling the input data
input_data_reshaped = sc.transform(input_data_reshaped)

#predicting the given input
prediction = model_rf.predict(input_data_reshaped)

if (prediction[0] == 0):
  print('The Wine quality is not upto the mark!')
  print('The rating of this wine is below 7')
else:
  print('Cheers...It is a Good quality Wine.')
  print('The rating of this wine is Above 7')
```



```
Cheers...It is a Good quality Wine.
The rating of this wine is Above 7
```

RELATED LITERATURE

Based on our research question "what wine features are important to get the promising result?" To answer this question we concatenate two datasets together and get a total dataset then we implement the Pearson coefficient correlation matrices and calculate the relationship among all the features. Then we ranked the features based on high correlation with the quality feature. The analysis of groups of features from left to right is implemented, and from the datasets first 11 features are selected and the last feature is. After identifying the importance of the features we start the implementation of the model. To analyze the performance of the model firstly, we implemented the model on the original data (unbalanced class), and then implemented the model on the balance class, balancing each class.

From these unbalancing and balancing classes, we achieved a better performance result on the balanced class for all the models. Among the three algorithms, the Random Forest Classifier (RFC) algorithm achieved the best performance result from both red and white wine datasets as compare to the support vector machine (SVM) and naïve Bayes (NB) algorithm.

In addition, our model achieved the best 90.76% accuracy from the Random Forest Classifier model by applying the Pearson coefficient correlation matrices for the feature selection.

CONCLUSION

The application of machine learning in wine quality prediction has proven to be an effective approach for evaluating wine characteristics based on chemical properties. By utilizing various algorithms such as Decision Trees, Random Forest, Support Vector Machines (SVM), and Neural Networks, models can accurately classify wine quality and provide insights into key contributing factors.

From analysis and experimentation, it is evident that ensemble methods, such as Random Forest, often yield higher accuracy and robustness in prediction compared to simpler models like Logistic Regression or Decision Trees. Feature selection also plays a crucial role in improving model performance, as certain chemical components (e.g., alcohol content, acidity, and pH) significantly impact wine quality.

However, challenges remain in achieving perfect predictions due to the subjectivity of wine quality assessments and the limitations of available datasets. Future research could focus on incorporating sensory data, advanced deep learning techniques, or hybrid models to enhance accuracy and generalization.

Overall, machine learning provides a powerful tool for wine quality prediction, aiding the wine industry in quality control, process optimization, and product development.

REFERENCES

- Chawla, N.V., 2005. Data Mining for Imbalanced Datasets: An Overview, in: Maimon, O., Rokach, L. (Eds.), Data Mining and Knowledge Discovery Handbook. Springer US, Boston, MA, pp. 853–867. **https://doi.org/10.1007/0-387-25465-X_4**

- Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J., 2009. Modeling wine preferences by data mining from physicochemical properties. Decis. Support Syst. 47, 547–553. **https://doi.org/10.1016/j.dss.2009.05.016**

- Dahal, K., Dahal, J., Banjade, H., Gaire, S., 2021. Prediction of Wine Quality Using Machine Learning Algorithms. Open J. Stat. 11, 278–289. **https://doi.org/10.4236/ojs.2021.112015**

- **Dataset used:-** https://archive.ics.uci.edu/dataset/186/wine+quality