Classification of Red and White Wine :

Registration ID: 2111177

Introduction:

The wine research utilizes 13 distinct wine metrics, such as alcohol and sulfur content, which were assessed for various wine samples. The objective here is to create a model that can predict the wine class based on the 13 observed characteristics and determine the significant differences between the distinct classes. The classification issue discusses four models and assess their accuracy.

Dataset kaggle reference ="[https://www.kaggle.com/datasets/rajyellow46/wine-quality"](https://www.kaggle.com/datasets/rajyellow46/wine-quality)

Objectives :

1. Data Description and Analysis
2. EDA and Data Visualization
3. Training and Evaluating Model
4. Saving Model for Future Use

File Description :

wine-quality-white-and-red.csv used for training the model and forecasting using the model trained using train split of dataset

Data feautures :

`1. Type:` In general there are red, white and pink wines available and have different features to each. The train data set has red and wine data types

`2. Fixed Acidity:` Fixed acidity is a property of the sample that refers to the group of low volatility organic acids such malic, lactic, tartaric, or citric acids.

`3. Volatile Acidity:` The quantity of acetic acid in wine, which can provide an unpleasant vinegar flavour at high amounts. Formic acid, acetic acid, propionic acid, and butyric acid are the short chain organic acids that may be recovered from the sample using the distillation method and are considered to have a volatile acidity.

`4. Citric Acid:` Citric acid, which is present in wines in small amounts, can give them a "freshness" and flavour. A weak organic acid without colour, citric acid is. Citrus fruits naturally contain it.

`5. Residual Sugar:` It's uncommon to discover wines with less than 1 gramme of sugar per litre of wine once fermentation has stopped. The sugars that remain unfermented in a final wine are referred to as residual sugar. G/L stands for grammes of sugar per litre. The sweetness of a wine is influenced by the residual sugar content, and in the EU, the RS level is associated with particular labelling words.

`6. Chlorides:` How much salt is in the wine and The ions taken from the skins during fermentation are the reason for the increased chloride extraction during the production of red wine. In order to prevent finished wine from exceeding the maximum permitted quantity of 606mg/L chloride, red juice should only contain 356mg/L of chloride ions (356mg/L in red juice multiplied by 1.7 equals 606).

`7. Free sulfur dioxide:` Free sulfites have antimicrobial and antioxidant effects. The sulfites that have interacted with other molecules in the wine medium are known as bound sulfites. The total sulfite concentration is calculated as the sum of the free and bound sulfites.

`8. Total sulfur dioxide:` Total Sulfur Dioxide (TSO2) is the sum of the free SO2 in the wine and the SO2 bonded to other compounds in the wine such as aldehydes, pigments, or sugars.

`9. Density:` Hydrometers are used by winemakers to determine the density of juice, fermenting wine, and produced wine in proportion to pure water. This is referred to as specific gravity (SG).

`10. pH:` For starters, high pH wines are more prone to microbial deterioration. Sulfur dioxide (typically in the form of potassium metabisulfite) has traditionally been used to keep wines stable throughout ageing.

`11. sulphates:` Wine sulfites are naturally present in low concentrations in all wines and are one of hundreds of chemical byproducts produced during the fermentation process.

`12. Alcohol:` The % of alcohol content in wine

`13. Quality::` Score ranges between 3 and 9 based on observed data.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, cal
```

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pyplt


import pickle as pckl
import seaborn as sns
```

Loading wine dataset

```
wine_csv = pd.read_csv('/content/gdrive/MyDrive/Colab Notebooks/wine-quality-white-
```

Double-click (or enter) to edit

Dataset Description :

```
wine_csv.shape
```

```
(6497, 13)
```

```
wine_csv.columns=['TYPE','FIXED_ACIDITY','VOLATILE_ACIDITY','CITRIC_ACID','RESIDUAL
wine_csv.columns
```

```
Index(['TYPE', 'FIXED_ACIDITY', 'VOLATILE_ACIDITY', 'CITRIC_ACID',
       'RESIDUAL_SUGAR', 'CHLORIDES', 'FREE_SULFUR_DIOXIDE',
       'TOTAL_SULFUR_DIOXIDE', 'DENSITY', 'pH', 'SULPHATES', 'ALCOHOL',
       'QUALITY'],
      dtype='object')
```

```
wine_csv.head()
```

	TYPE	FIXED_ACIDITY	VOLATILE_ACIDITY	CITRIC_ACID	RESIDUAL_SUGAR	CHLORID
0	white	7.0	0.27	0.36	20.7	0.0
1	white	6.3	0.30	0.34	1.6	0.0
2	white	8.1	0.28	0.40	6.9	0.0
3	white	7.2	0.23	0.32	8.5	0.0

```
wine_csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   TYPE                  6497 non-null   object
 1   FIXED_ACIDITY         6497 non-null   float64
 2   VOLATILE_ACIDITY      6497 non-null   float64
 3   CITRIC_ACID           6497 non-null   float64
 4   RESIDUAL_SUGAR        6497 non-null   float64
 5   CHLORIDES             6497 non-null   float64
 6   FREE_SULFUR_DIOXIDE   6497 non-null   float64
 7   TOTAL_SULFUR_DIOXIDE  6497 non-null   float64
 8   DENSITY               6497 non-null   float64
 9   pH                    6497 non-null   float64
 10  SULPHATES             6497 non-null   float64
 11  ALCOHOL               6497 non-null   float64
 12  QUALITY               6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

`Information:` Dataset consists of 6497 records and 13 columns and has no NULL data and the data type of Type column is object are all the remaining columns are in numeric

```
wine_csv.describe()
```

FIXED_ACIDITY VOLATILE_ACIDITY CITRIC_ACID RESIDUAL_SUGAR CHLORIDES

statistical Observation:

The mean of fixed acidity is 7.21, the maximum value is 15.9

The mean of volatile acidity is 0.33, the maximum value is 1.58

The mean of citric acid is 0.31, the maximum value is 1.66

The mean of residual sugar is 5.44, the maximum value is 65.8

The mean of chlorides is 0.05, the maximum value is 0.61

The mean of free sulfur dioxide is 30.52, the maximum value is 289

The mean of total sulfur dioxide is 115.74, the maximum value is 440

The mean of density is 0.99, the maximum value is 1.03

The mean of pH is 3.21, the maximum value is 4.01

The mean of sulphates is 0.53, the maximum value is 2

The mean of alcohol is 10.49, the maximum value is 14.90

The mean of quality is 5.81, the maximum value is 9

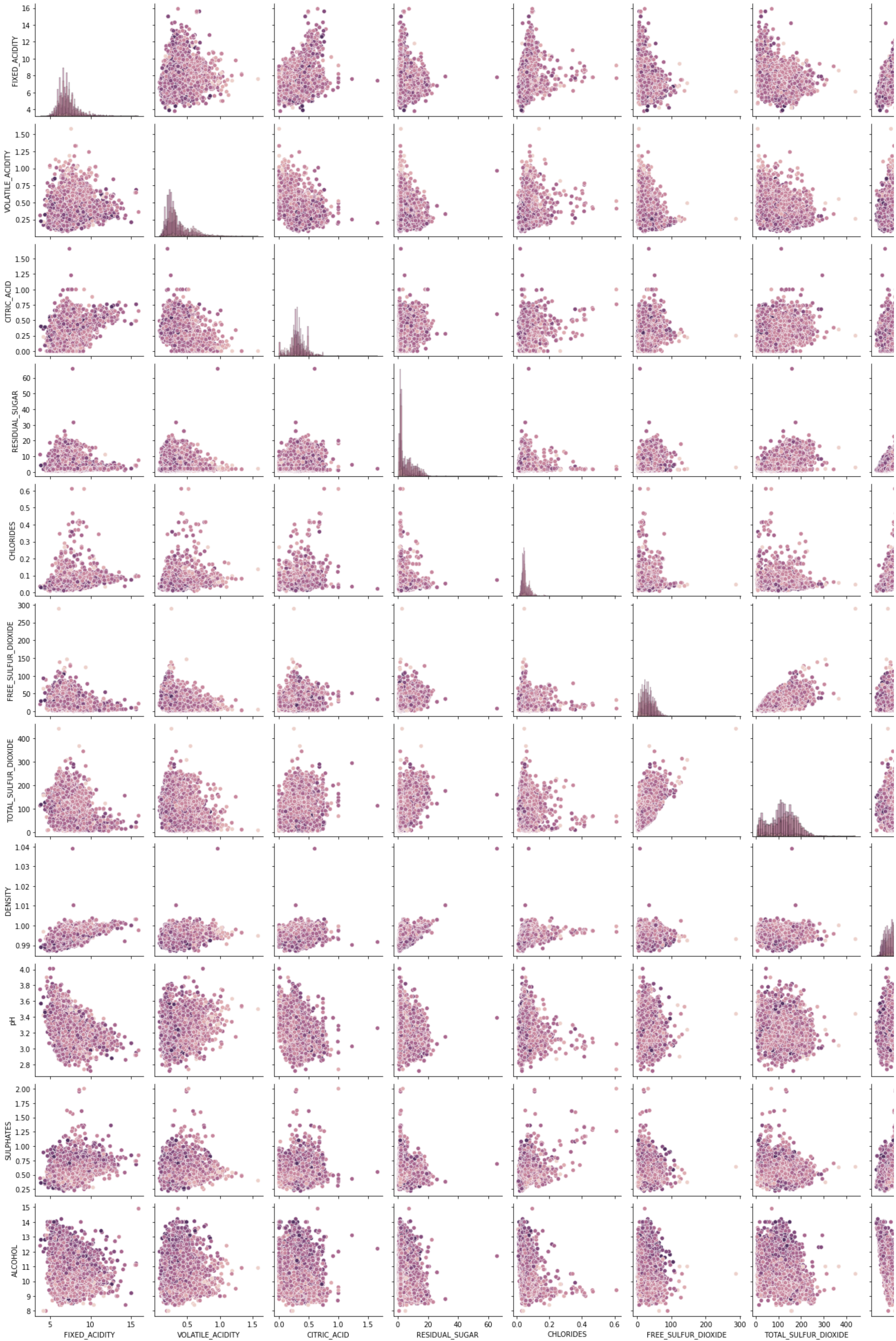
```
wine_csv.groupby(['TYPE']).size().reset_index(name='COUNT')
```

	TYPE	COUNT
0	red	1599
1	white	4898

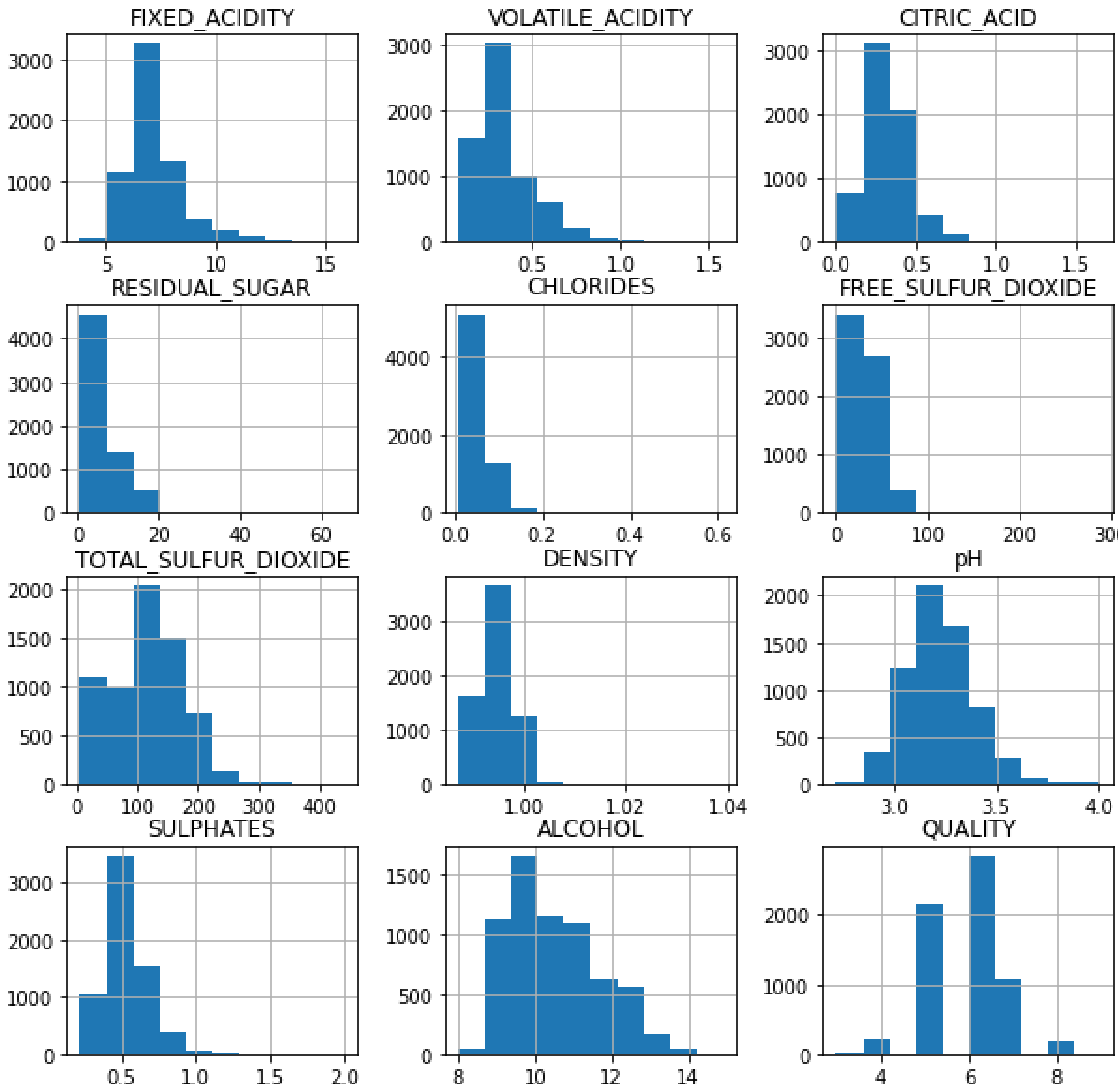
Size: Dataset consists of 1599 records of red wine and 4898 records of white wine

Exploratory Data Analysis

```
sns.pairplot(wine_csv,diag_kind = "hist",hue='QUALITY')
pyplt.savefig('wine_pair_plot.png')
pyplt.show()
```


FIXED_ACIDITY
VOLATILE_ACIDITY
CITRIC_ACID
RESIDUAL_SUGAR
CHLORIDES
FREE_SULFUR_DIOXIDE
TOTAL_SULFUR_DIOXIDE
DENSITY
pH
SULPHATES
ALCOHOL

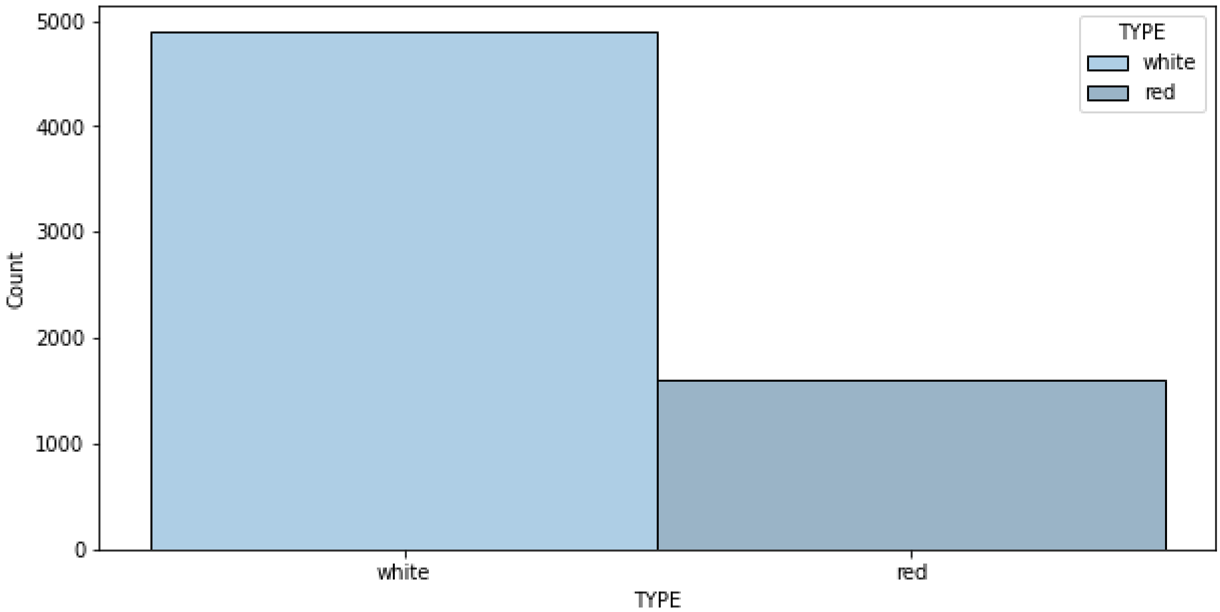
```
wine_csv.hist(bins=10,figsize=(10,10))
pyplt.savefig('Wine_Histogram.png')
pyplt.show()
```

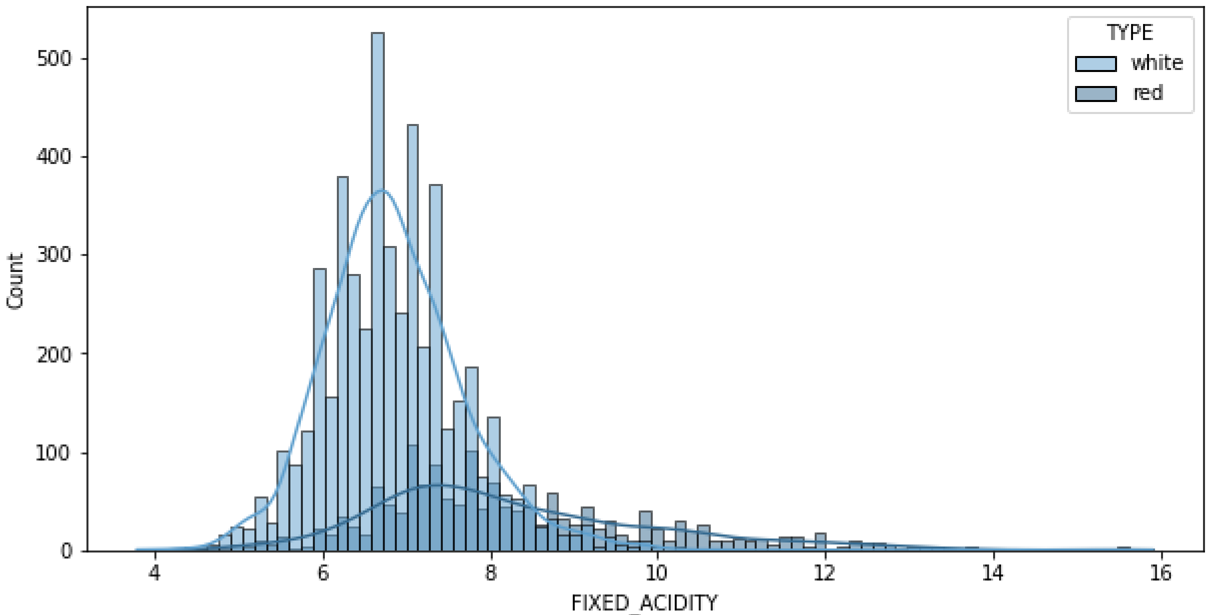


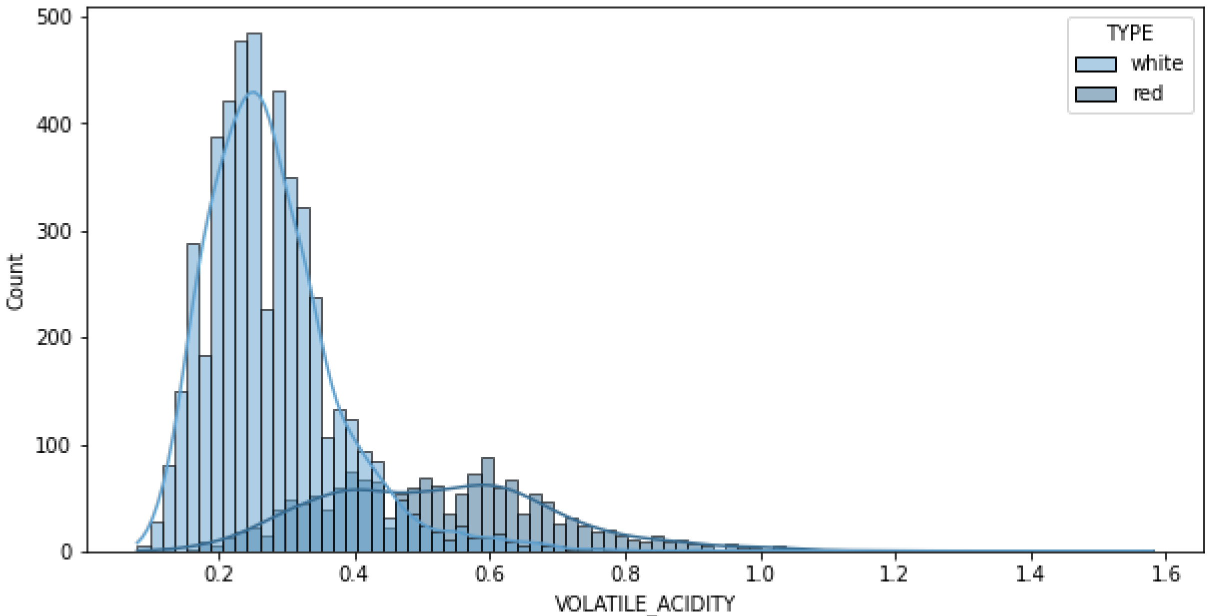
Histogram: Histogram showing frequency distribution of each column.

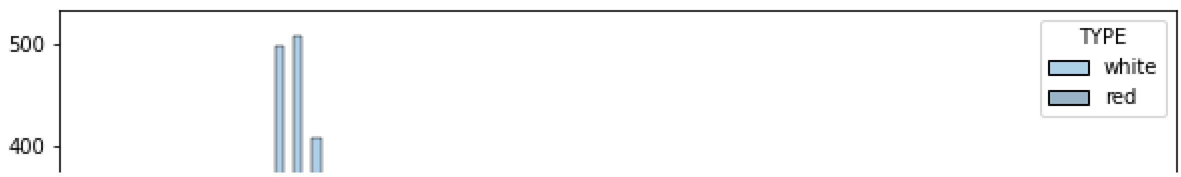
```
for i, val in enumerate(wine_csv.columns):
        if val != "QUALITY":
            plot = pyplt.figure(i, figsize = (10,5))
            sns.histplot(data = wine_csv, x = val,
```

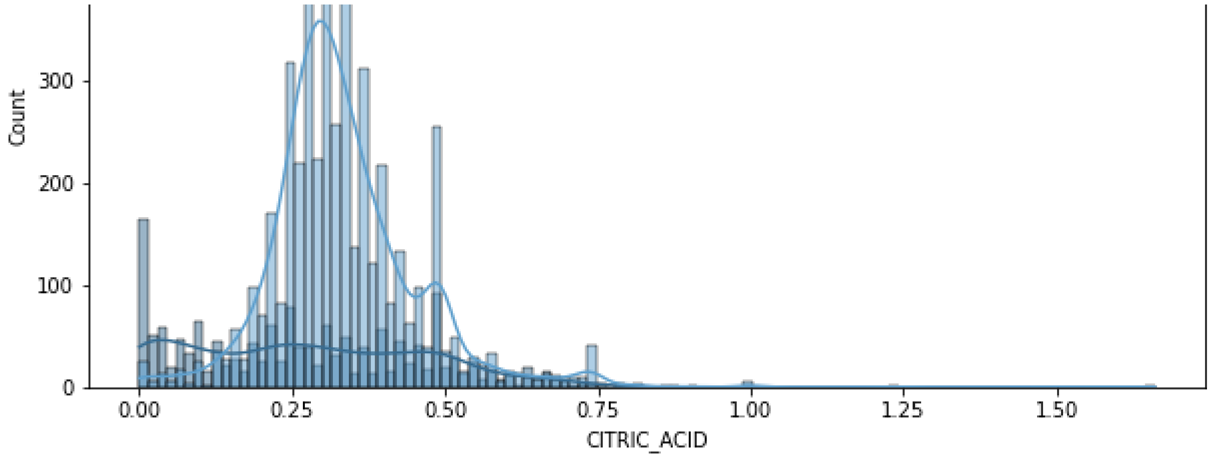
```
                    hue="TYPE", palette="Blues_d",kde=True)
    pyplt.savefig('wine_'+str(val)+'.png')
```

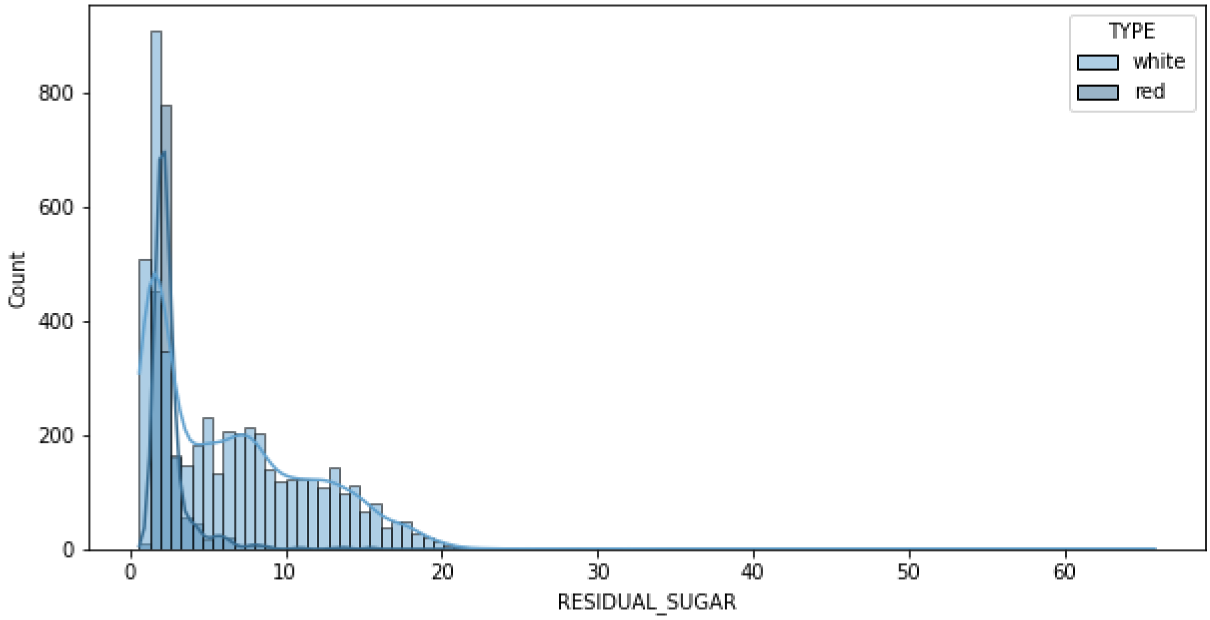

TYPE
white
red
Count
0
1000
2000
3000
4000
5000
white
red
TYPE

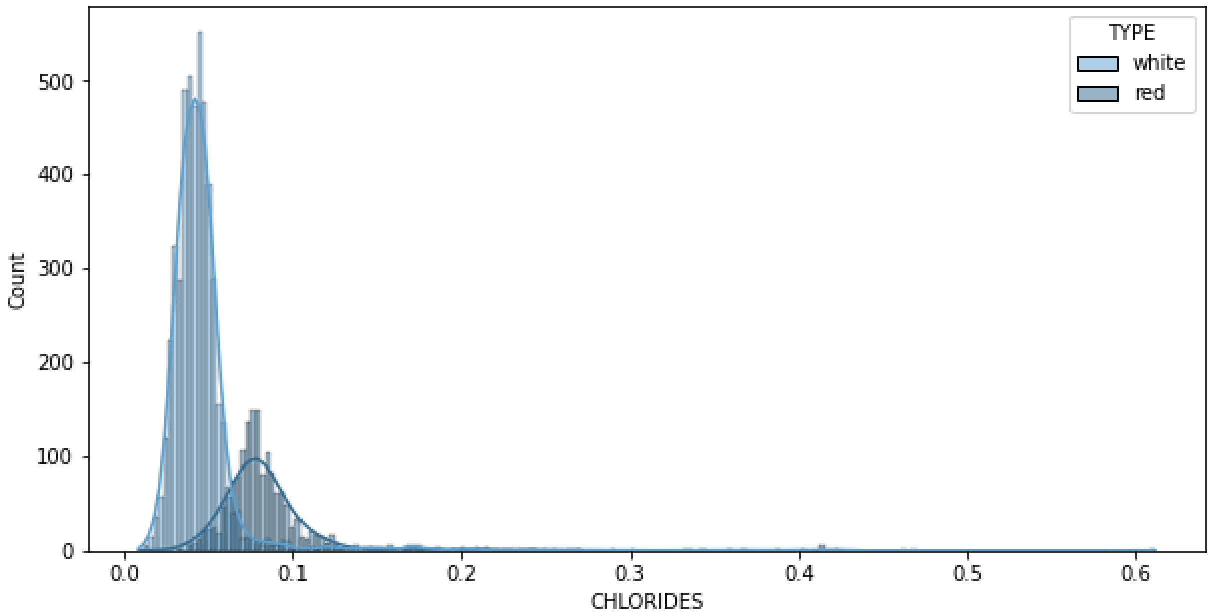

TYPE
white
red
Count
0
100
200
300
400
500
4
6
8
10
12
14
16
FIXED_ACIDITY


TYPE
white
red
Count
0
100
200
300
400
500
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
VOLATILE_ACIDITY

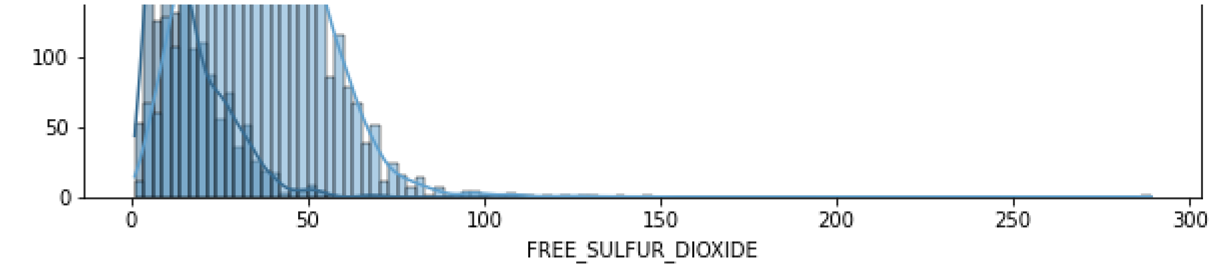

TYPE
white
red
500
400

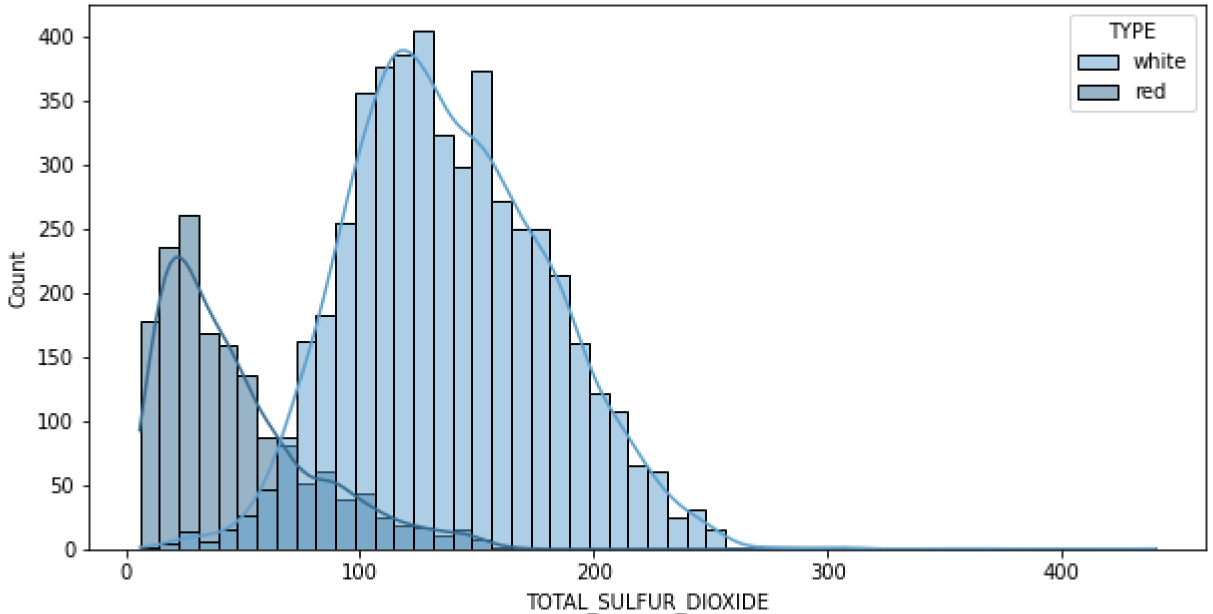

Count
0
100
200
300
0.00
0.25
0.50
0.75
1.00
1.25
1.50
CITRIC_ACID

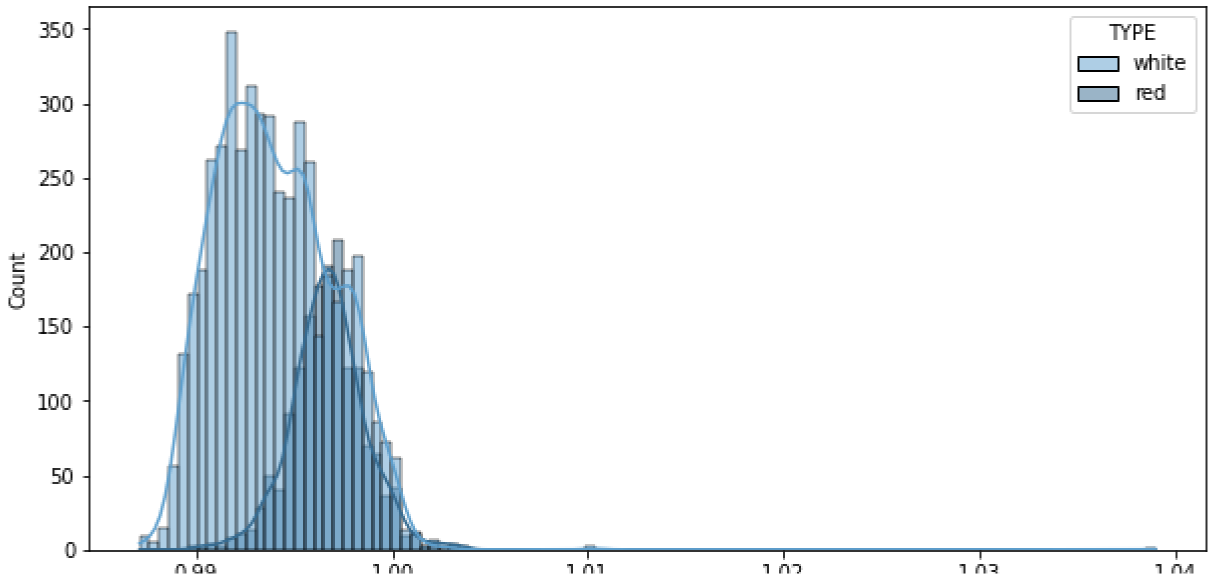

TYPE
white
red
Count
0
200
400
600
800
0
10
20
30
40
50
60
RESIDUAL_SUGAR


TYPE
white
red
Count
0
100
200
300
400
500
0.0
0.1
0.2
0.3
0.4
0.5
0.6
CHLORIDES


TYPE
white
red
Count
150
200
250
300
350



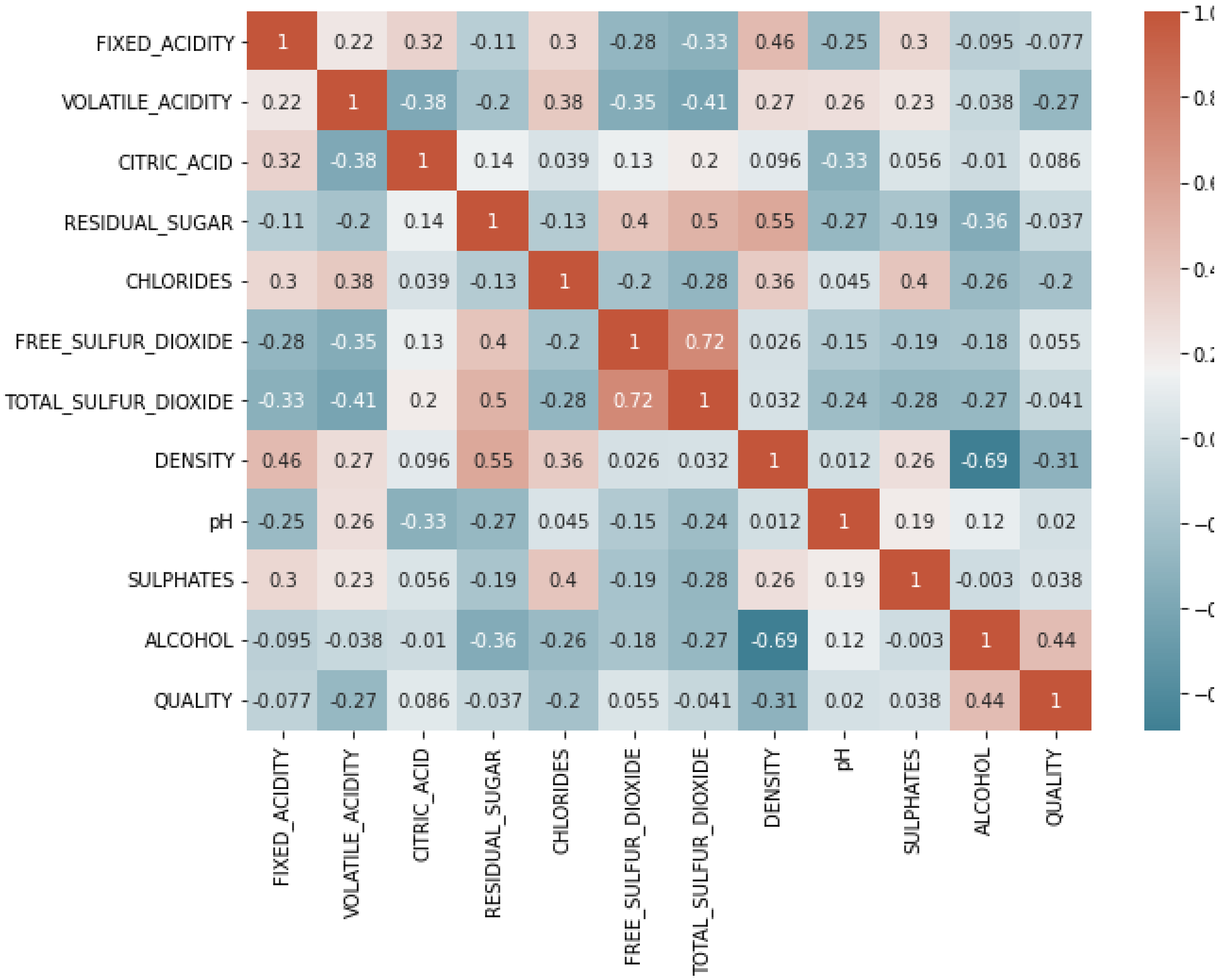




Explanation : Using for loop to show the smooth distribution and show on the histogram contains a group of bars that show the density of the data (i.e., the count of the number of records) for different ranges our variables in our dataset.

```
corr = wine_csv.corr()
fig1, sub = pyplt.subplots(figsize=(10,7))

sub = sns.heatmap(corr,
cmap=sns.diverging_palette(220, 20, as_cmap=True),
ax=sub,
annot=True
)
```

```
pyplt.savefig('wine_correlation.png')
```



Correlation : Using correlation heatmap of seaborn to explain correlation percentage between all the variables

Removing columns with percent greater than 7 of correlation as there is no point of having more columns with same correlation

```
for col in range(len(wine_csv.corr().columns)):
    for val in range(col):
        if abs(wine_csv.corr().iloc[col,val]) > 0.7:
            print(wine_csv.corr().columns[val],wine_csv.corr().columns[col])
```

```
FREE_SULFUR_DIOXIDE TOTAL_SULFUR_DIOXIDE
```

```
wine_csv = wine_csv.drop("FREE_SULFUR_DIOXIDE",axis='columns')
wine_csv.head()
```

	TYPE	FIXED_ACIDITY	VOLATILE_ACIDITY	CITRIC_ACID	RESIDUAL_SUGAR	CHLORID
0	white	7.0	0.27	0.36	20.7	0.0
1	white	6.3	0.30	0.34	1.6	0.0
3	white	7.2	0.23	0.32	8.5	0.0

Display of data : showing data after removal of FREE_SULFUR_DIOXIDE column

```
wine_csv['QUALITY'].unique()
wine_csv.groupby("QUALITY").size()
```

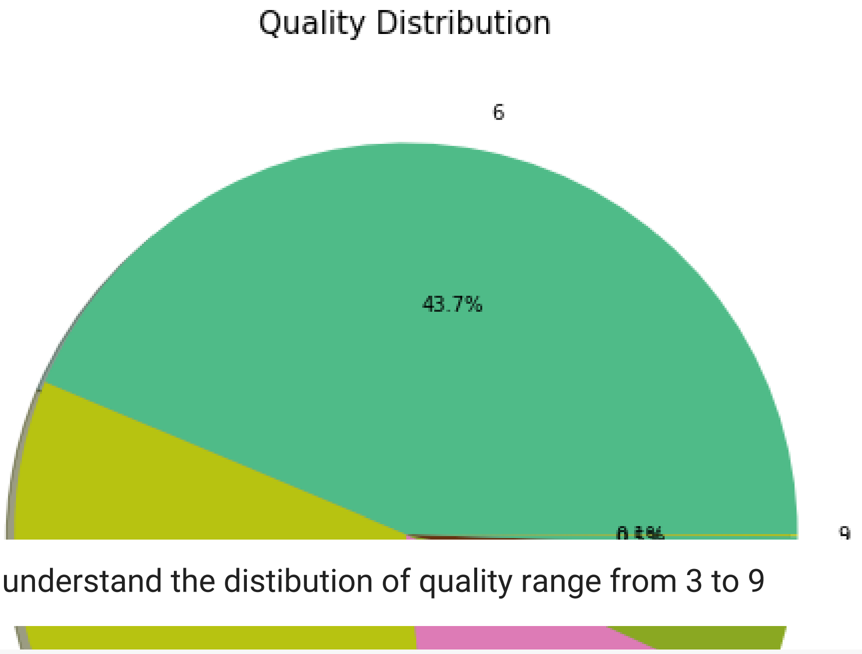
```
QUALITY
3      30
4     216
5    2138
6    2836
7    1079
8     193
9       5
dtype: int64
```

```
wine_csv['QUALITY'].mean()
```

```
5.818377712790519
```

Display of counts and mean : showing data counts of each quality of wine and average of the quality

```
colors = ['#4FBB88', '#B7C311', '#DD7BB6', '#8AA822','#663311']
pyplt.figure(figsize=(9,9))

ind = wine_csv['QUALITY'].value_counts().index
siz = wine_csv['QUALITY'].value_counts().values

pyplt.pie(siz, labels=ind,colors=colors,autopct='%1.1f%%',shadow = True,)
pyplt.title('Quality Distribution',color = 'black', fontsize=15)
pyplt.savefig('Quality Distribution Pie.png')
pyplt.show()
```



Pie chart to understand the distibution of quality range from 3 to 9

```
X = wine_csv.drop("TYPE",axis=1)
X['WINE_QUALITY'] = X['QUALITY'].apply(lambda x: 0 if x>5. else 1)
X.tail()
```

	FIXED_ACIDITY	VOLATILE_ACIDITY	CITRIC_ACID	RESIDUAL_SUGAR	CHLORIDES
6492	6.2	0.600	0.08	2.0	0.090
6493	5.9	0.550	0.10	2.2	0.062
6494	6.3	0.510	0.13	2.3	0.076
6495	5.9	0.645	0.12	2.0	0.075
6496	6.0	0.310	0.47	3.6	0.067

```
y = wine_csv['TYPE']
X.drop("QUALITY",axis=1)
X.head()
```

	FIXED_ACIDITY	VOLATILE_ACIDITY	CITRIC_ACID	RESIDUAL_SUGAR	CHLORIDES	TO
0	7.0	0.27	0.36	20.7	0.045	
1	6.3	0.30	0.34	1.6	0.049	
2	8.1	0.28	0.40	6.9	0.050	
3	7.2	0.23	0.32	8.5	0.058	
4	7.2	0.23	0.32	8.5	0.058	

Double-click (or enter) to edit

```
from sklearn.preprocessing import LabelEncoder
```

```
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

y
```

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

METHODS

Splitting the traning and testing data by using 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=0.25, random_st
```

Features Standardize by mean elimination and scaling to unit variance.

```
from sklearn.preprocessing import StandardScaler
sd = StandardScaler()

X_train_sd = sd.fit_transform(X_train)
X_test_sd = sd.transform(X_test)
```

```
X_train_sd.shape
```

```
(4872, 12)
```

`ML Model Dictionary :` Creating dictionary to refer to all the given models

1. KNeighborsClassifier
2. LogisticRegression
3. DecisionTreeClassifier
4. SupportVectorClassifier
5. RandomForestClassifier

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier


ML_Models = {}
```

```
#LogisticRegressionModel

Log_Reg = LogisticRegression(solver='lbfgs', max_iter=1000)
ML_Models["Logistic Regression"] = Log_Reg

#KNeighborsClassifierModel

K_Neigh = KNeighborsClassifier()
ML_Models["K Neighbors Classifier"] = K_Neigh

#SupportVectorClassifierModel

Sup_Vec = SVC(kernel="linear")
ML_Models["Support Vector Classifier"] = Sup_Vec

#DecisionTreeClassifierModel

Des_Tree = DecisionTreeClassifier()
ML_Models["Decision Tree Classifier"] = Des_Tree

#RandomForestClassifierModel

Rand_Forest = RandomForestClassifier(n_estimators=10, criterion="entropy",random_st
ML_Models["Random Forest"] = Rand_Forest
```

`Displaying Model Dictionary`

```
print(ML_Models)
```

```
{'Logistic Regression': LogisticRegression(max_iter=1000), 'K Neighbors Class
```

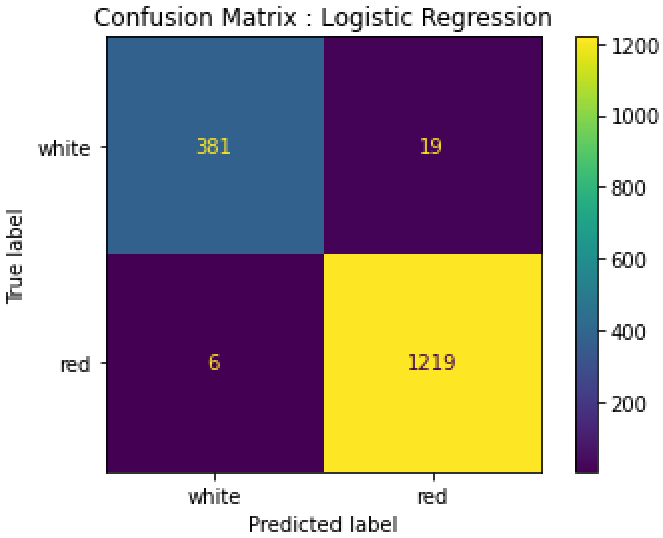
Creating function to display Model metrics

1. Confusion matrix
2. Accuracy score
3. Confusion matrix display
4. Classification report

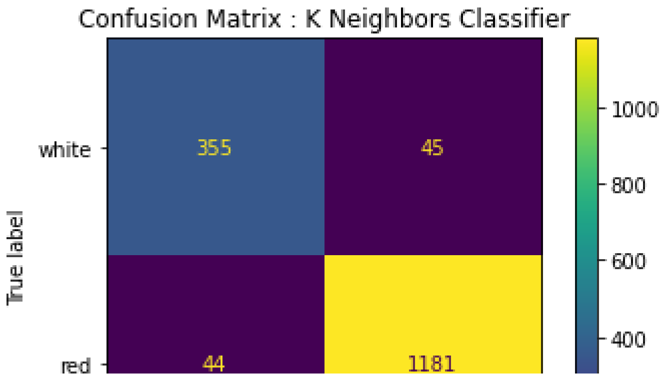
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report



for M in ML_Models:
    ML_Models[M].fit(X_train,y_train)

scores={}
```

```
def ML_Models_prediction(Name,ML_Model,X_test,y_test):
    print("=============================================================")
    print("ML_Model:",Name)
    y_predicted = ML_Model.predict(X_test)
    scores[Name] = accuracy_score(y_test,y_predicted)
    print("Accuracy_Score:",scores[Name])
    Conf_Matrix = confusion_matrix(y_test,y_predicted)
    print("Confusion_Matrix:\n",Conf_Matrix)
    Cond_Matrix_Disp = ConfusionMatrixDisplay(Conf_Matrix,display_labels=["white",'
    plot=Cond_Matrix_Disp.plot()
    print("Classification_Report:\n",classification_report(y_test,y_predicted))
    plot.ax_.set_title('Confusion Matrix : '+ str(Name))
    pyplt.savefig(str(Name)+'_conf_matrix.png')
    pyplt.show()
    print( "=============================================================" )

for M in ML_Models:
    ML_Models_prediction(M,ML_Models[M],X_test,y_test)
```

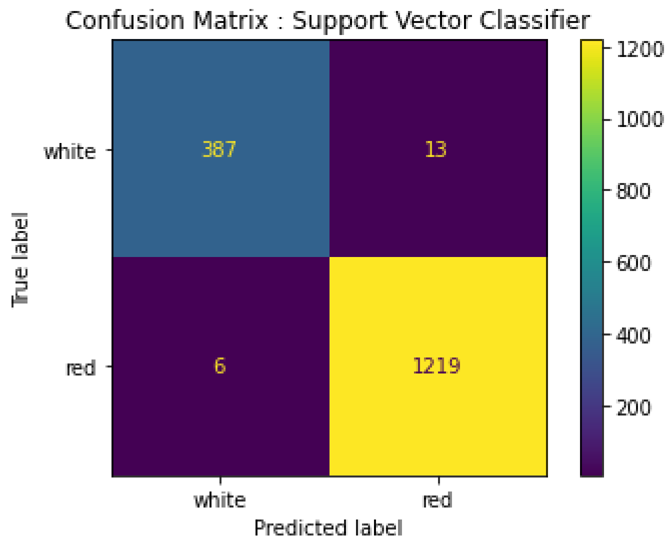
```
==============================================================
ML_Model: Logistic Regression
Accuracy_Score: 0.9846153846153847
Confusion_Matrix:
 [[ 381   19]
 [   6 1219]]
Classification_Report:
               precision    recall  f1-score   support

           0       0.98      0.95      0.97       400
           1       0.98      1.00      0.99      1225

    accuracy                           0.98      1625
   macro avg       0.98      0.97      0.98      1625
weighted avg       0.98      0.98      0.98      1625
```



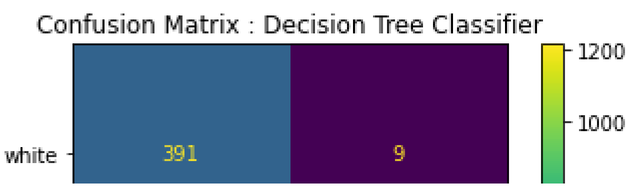

```
==============================================================
==============================================================
ML_Model: K Neighbors Classifier
Accuracy_Score: 0.9452307692307692
Confusion_Matrix:
 [[ 355   45]
 [  44 1181]]
Classification_Report:
               precision    recall  f1-score   support

           0       0.89      0.89      0.89       400
           1       0.96      0.96      0.96      1225

    accuracy                           0.95      1625
   macro avg       0.93      0.93      0.93      1625
weighted avg       0.95      0.95      0.95      1625
```

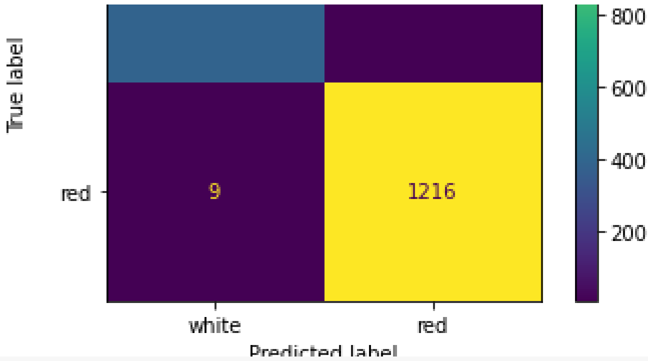



```
================================================================
================================================================
ML_Model: Support Vector Classifier
Accuracy_Score: 0.9883076923076923
Confusion_Matrix:
 [[ 387   13]
 [   6 1219]]
Classification_Report:
               precision    recall  f1-score   support

           0       0.98      0.97      0.98       400
           1       0.99      1.00      0.99      1225

    accuracy                           0.99      1625
   macro avg       0.99      0.98      0.98      1625
weighted avg       0.99      0.99      0.99      1625
```




```
================================================================
================================================================
ML_Model: Decision Tree Classifier
Accuracy_Score: 0.9889230769230769
Confusion_Matrix:
 [[ 391    9]
 [   9 1216]]
Classification_Report:
               precision    recall  f1-score   support

           0       0.98      0.98      0.98       400
           1       0.99      0.99      0.99      1225

    accuracy                           0.99      1625
   macro avg       0.99      0.99      0.99      1625
weighted avg       0.99      0.99      0.99      1625
```






```
ML_Models_List = []
for M1 in ML_Models:
    ML_Models_List.append((M1,ML_Models[M1]))
ML_Models_List
```

```
[('Logistic Regression', LogisticRegression(max_iter=1000)),
 ('K Neighbors Classifier', KNeighborsClassifier()),
 ('Support Vector Classifier', SVC(kernel='linear')),
 ('Decision Tree Classifier', DecisionTreeClassifier()),
 ('Random Forest',
  RandomForestClassifier(criterion='entropy', n_estimators=10,
random_state=0))]
```

```
    accuracy                           1.00      1625
```

```
from sklearn.ensemble import VotingClassifier
Ensemble_Model = VotingClassifier(estimators = ML_Models_List, voting="hard")
Ensemble_Model.fit(X_train,y_train)
ML_Models_prediction("The Base Learner Model Ensemble",Ensemble_Model,X_test,y_test
```

```
==============================================================
ML_Model: The Base Learner Model Ensemble
Accuracy_Score: 0.9901538461538462
Confusion_Matrix:
 [[ 389   11]
```

```
scores
```

```
{'Decision Tree Classifier': 0.9889230769230769,
 'K Neighbors Classifier': 0.9452307692307692,
 'Logistic Regression': 0.9846153846153847,
 'Random Forest': 0.9963076923076923,
 'Support Vector Classifier': 0.9883076923076923,
 'The Base Learner Model Ensemble': 0.9901538461538462}
```

```
weighted avg       0.99      0.99      0.99      1625
```

```
import tensorflow as tf_ml
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import regularizers as reg
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential as seq
from tensorflow.keras.backend import clear_session

# Removing ANN previous model run data
clear_session()

factor=0.0001
rate=0.1

# Model Structure
ANN_Model=seq([tf_ml.keras.layers.Dense(80,input_shape=(12,),activation="relu",kern
               tf_ml.keras.layers.Dropout(rate),
               tf_ml.keras.layers.Dense(60,activation="relu",kernel_regularizer=reg
               tf_ml.keras.layers.Dropout(rate),
               tf_ml.keras.layers.Dense(40,activation='relu',kernel_regularizer=reg
               tf_ml.keras.layers.Dropout(rate),
               tf_ml.keras.layers.Dense(20,activation='relu',kernel_regularizer=reg
               tf_ml.keras.layers.Dropout(rate),
               tf_ml.keras.layers.Dense(units=1, activation='sigmoid')])
```

```
class mycallbackClass(tf_ml.keras.callbacks.Callback):
  def epoch_fun(main,epoch,logs={}):
    if(logs.get("val_accuracy")>0.95):
      print("Achived expected accuracy 90%", logs)
      main.model.stop_training=True
classcall=mycallbackClass()
```

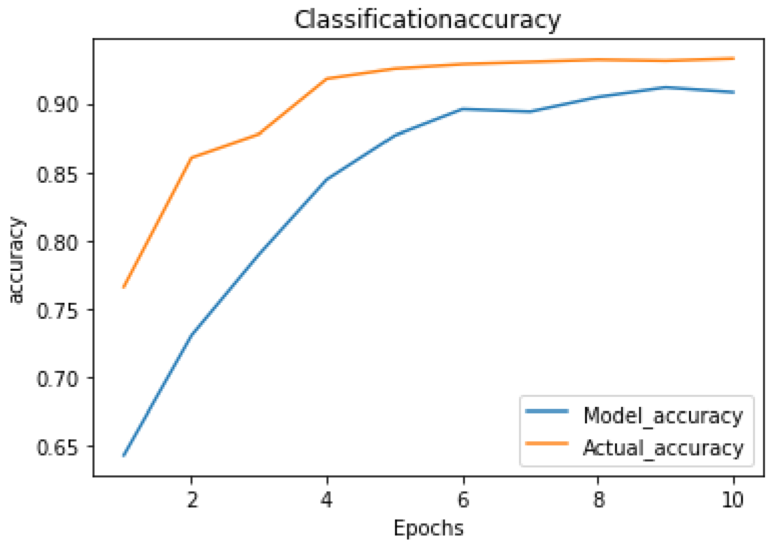
```
def Display_Ann_metric(Hist, cal):
    train_metrics = Hist.history[cal]
    val_metrics = Hist.history['val_'+cal]
    epochs = range(1, len(train_metrics) + 1)
    pyplt.plot(epochs, train_metrics)
```

```
    pyplt.plot(epochs, val_metrics)
    pyplt.title('Classification'+ cal)
    pyplt.xlabel("Epochs")
    pyplt.ylabel(cal)
    pyplt.legend(["Model_"+cal, 'Actual_'+cal])
    pyplt.show()

ANN_Model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['acc
```

```
ANN_Model_history=ANN_Model.fit(X_train, y_train, batch_size = 256,verbose=2,
                  epochs = 10,callbacks=[classcall],validation_split=0.25)
```

```
Epoch 1/10
15/15 - 2s - loss: 1.5662 - accuracy: 0.6431 - val_loss: 0.6370 - val_accurac
Epoch 2/10
15/15 - 0s - loss: 0.6608 - accuracy: 0.7304 - val_loss: 0.3440 - val_accurac
Epoch 3/10
15/15 - 0s - loss: 0.5001 - accuracy: 0.7898 - val_loss: 0.3153 - val_accurac
Epoch 4/10
15/15 - 0s - loss: 0.4161 - accuracy: 0.8446 - val_loss: 0.2712 - val_accurac
Epoch 5/10
15/15 - 0s - loss: 0.3533 - accuracy: 0.8766 - val_loss: 0.2306 - val_accurac
Epoch 6/10
15/15 - 0s - loss: 0.3174 - accuracy: 0.8957 - val_loss: 0.2137 - val_accurac
Epoch 7/10
15/15 - 0s - loss: 0.3072 - accuracy: 0.8938 - val_loss: 0.2121 - val_accurac
Epoch 8/10
15/15 - 0s - loss: 0.2919 - accuracy: 0.9045 - val_loss: 0.2122 - val_accurac
Epoch 9/10
15/15 - 0s - loss: 0.2746 - accuracy: 0.9116 - val_loss: 0.2144 - val_accurac
Epoch 10/10
15/15 - 0s - loss: 0.2860 - accuracy: 0.9080 - val_loss: 0.2061 - val_accurac
```

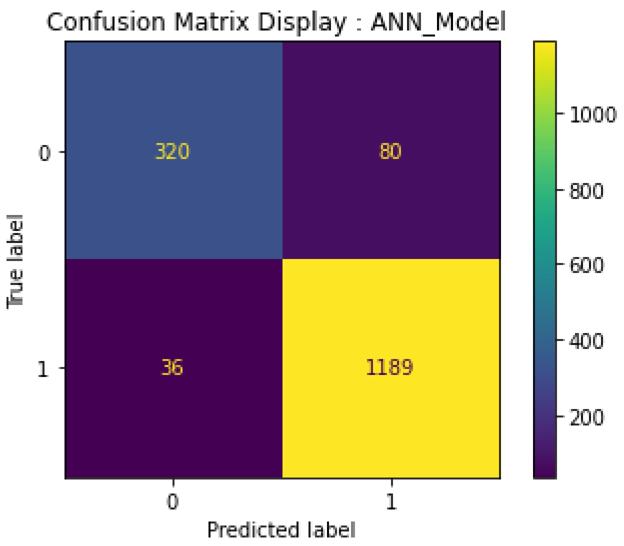
```
Display_Ann_metric(ANN_Model_history, 'accuracy')
```




```
y_predicted=ANN_Model.predict(X_test)>0.5
y_predicted = y_predicted.astype("int")
```

```
print( "==============================================================" )
print("Deep Learning Model: ANN_Model")
print("Accuracy_Score:",accuracy_score(y_test,y_predicted))
Conf_Matrix = confusion_matrix(y_test,y_predicted)
print("Confusion_Matrix:\n",Conf_Matrix)
Cond_Matrix_Disp = ConfusionMatrixDisplay(Conf_Matrix)
plot=Cond_Matrix_Disp.plot()
print("Classification_Report:\n",classification_report(y_test,y_predicted))
plot.ax_.set_title('Confusion Matrix Display : ANN_Model ')
pyplt.show()
print( "==============================================================" )
```

```
==============================================================
Deep Learning Model: ANN_Model
Accuracy_Score: 0.9286153846153846
Confusion_Matrix:
 [[ 320   80]
 [  36 1189]]
Classification_Report:
               precision    recall  f1-score   support

           0       0.90      0.80      0.85       400
           1       0.94      0.97      0.95      1225

    accuracy                           0.93      1625
   macro avg       0.92      0.89      0.90      1625
weighted avg       0.93      0.93      0.93      1625
```




```
==============================================================
```

```
pyplt.bar(range(len(scores)), list(scores.values()), align='center',color=colors)
pyplt.xticks(range(len(scores)), list(scores.keys()),rotation=90)
pyplt.xlabel(" ML Algorithms")
pyplt.ylabel("Model Accuracy Score")
pyplt.show()
```