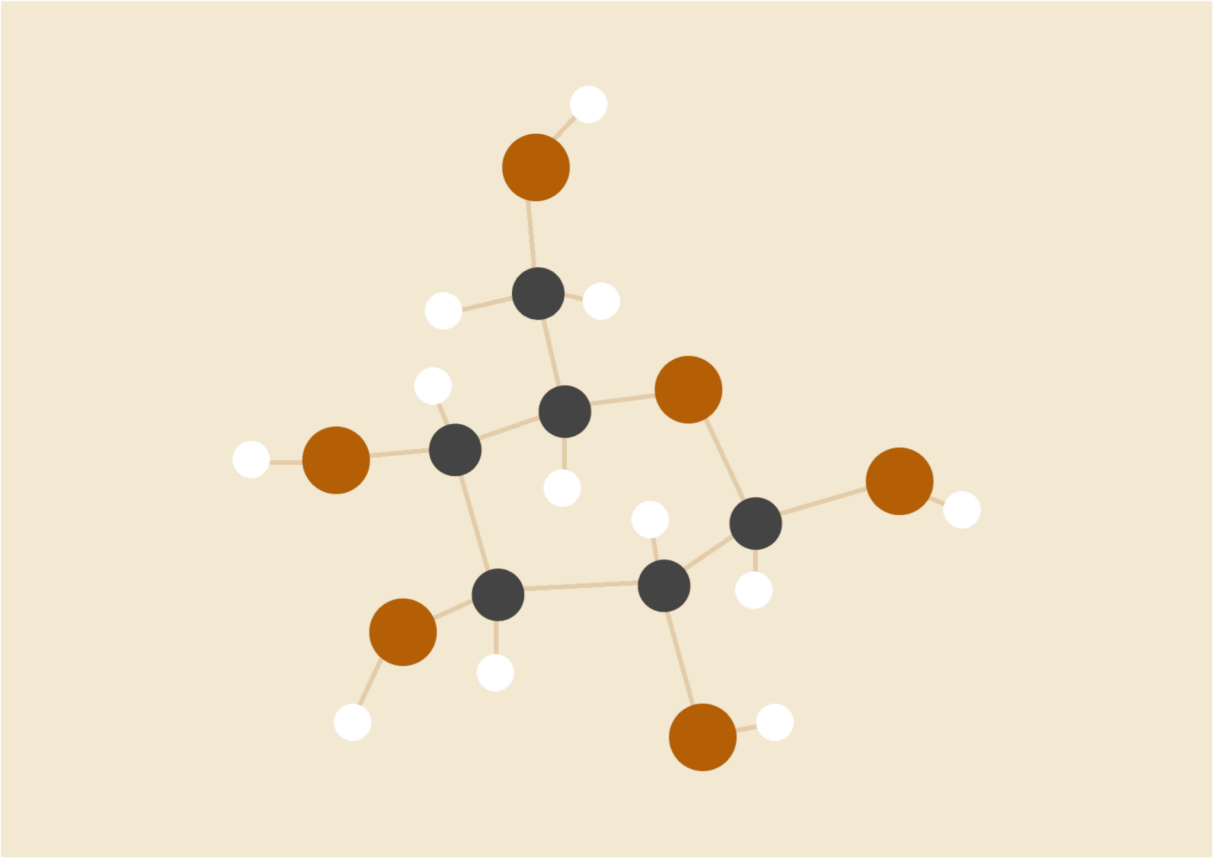
RED-WINE DATA REPORT



Abhinav Kompella

ES21BTECH11002

INTRODUCTION

Wine quality classification:

This is a model written from scratch to determine the quality of wine as either high or low quality given a test sample.

ALGORITHM

- I have used the Gaussian Naive Bayes algorithm
- I have opted for this model because
    - Earlier knowledge of probability and statistics
    - I have observed the data columns and the statistics are nearly in agreement with the normal curve distribution. Images in the jupyter notebook.
    -
- There are 2 more types of naive bayes but gaussian is used for classifying distribution because it assumes a gaussian curve.

- Data is first classified based on whether the quality is 0 or 1. The data is split into 80:20 ratio as train and test data. Then the mean and standard deviation are calculated for every column in the data frames separately for 0 and 1.

In [405]:
```
#calculates the mean and standard deviation of all columns of df0_clean and df1_clean
cols = ['fixed acidity', 'volatile acidity', 'citric acid','residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH',
        'sulphates', 'alcohol', 'k_value', 'l_value',
       'm_value', 'percentage_free_sulphur', 'n_value']

m0 = df0_clean[cols].mean()
m1 = df1_clean[cols].mean()
s0 = df0_clean[cols].std()
s1 = df1_clean[cols].std()
```

- Now that we know the mean and standard deviation, we can get the probability of each and every element in the row given the quality.

In [387]:
```
#function which gives treturns the probability for a given value with the help of standard deviation and mean for tha
def given_class_prob(x, mean, std):
    expo = exp(-((x-mean)**2 / (2 * std**2 )))
    return (1 / (sqrt(2 * pi) * std)) * expo
```

- From there using the bayesian formula for probability we can calculate the probabilities of the wine sample either being quality = 1 or quality = 0.. If the probability is higher for 1, it would be classified as high quality, else low.

- Ex: P(class=0 | X1, X2,X3…) = P(X1 | class = 0)*P(X2 | class = 0)….*P(class=0)
- Here, X1, X2.. are the row elements for a given sample (acidity, alcohol, etc).

Data preprocessing, Exploratory Data Analysis, Feature Engineering

- I used the .info() function on the data frame which gave me the statistics and type of data set used in the columns. It even gave some useful info such as all elements in the dataframe have non-null values.

In [411]:
```
#gives info, shows that there are no null values, shows the datat type and shows the number of columns and entries
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
 12  k_value               1599 non-null   float64
 13  l_value               1599 non-null   float64
 14  m_value               1599 non-null   float64
 15  percentage_free_sulphur  1599 non-null   float64
 16  n_value               1599 non-null   float64
dtypes: float64(16), int64(1)
memory usage: 212.5 KB
```

- I have used the correlation matrix to determine how related and influencing one column is to another. I found out that three columns have a heavy influence on the target variable(quality) and 3 more have near 0 correlation which I planned

on removing them as noise. But removing the columns which have near 0 correlation didn't work out because there was little data to train on and the accuracy dipped. In the snippet below we can see the relation of columns with quality. For Ex: volatile acidity and quality has a value of -0.321, which means it has a heavy influence and negative means lesser the volatile acidity more will be the quality.

```
In [334]: #correlation matrix gives how much each column is related to the other. The farther away from the 0, the more influe
          df.corr()
```

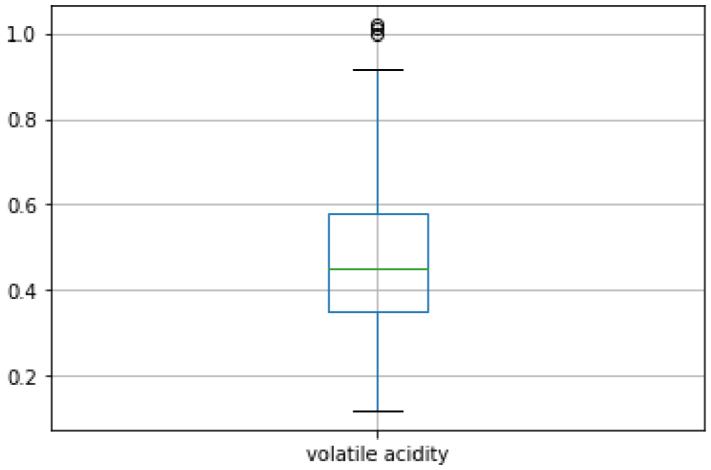
Out[334]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	k_
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.095093	0.9
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.321441	-0.1
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.159129	0.6
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	-0.002160	0.1
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.109494	0.1
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.061757	-0.1
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.231963	-0.1
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.159110	0.6
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.003264	-0.6

-

- Then I used boxplots on the heavily influencing columns and checked for outliers. Since removing outliers from these columns would give a rise in accuracy. Without removing the outliers the accuracy turned out to be 71.2 % and after removing outliers the accuracy went up to 77.2%. Here we can see those circles above the horizontal line which are considered as outliers.

```
In [433]: df1.boxplot(column='volatile acidity')
Out[433]: <AxesSubplot:>
```



- Below is the function which does the outlier removal.

```
#function which returns a dataframe free of outliers using boxplot technique
def find_outliers (df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3 - q1
    null_outliers_df = df[~((df > (q3 + 5*IQR)) | ( df < (q1 - 5*IQR)))] #replaces outlier values with null

    no_outliers = null_outliers_df.dropna().reset_index() #drops rows which contain null

    return no_outliers
```

-

Accuracy

Matches where prediction = actual

Non-match where prediction != actual. I iterated through the rows and found out the probability.

Confusion matrix was also included for better data analysis.

In [410]:
```
#prints model accuracy
matched = 0
for i in range(len(pred)):
    if(pred[i] == actual[i]):
        matched = matched + 1

accuracy = matched*100 / len(pred)
print(accuracy)
```

```
77.1875
```

In [417]:
```
#gives the confusion matrix
confusion_matrix = pd.crosstab(actual, pred, rownames=['Actual'], colnames=['Predicted'], margins=True)
confusion_matrix
```

Out[417]:

Predicted	0	1	All
Actual			
0	139	29	168
1	44	108	152
All	183	137	320