NOVEMBER 2020

# IDS PROJECT REPORT

## USING MACHINE LEARNING ALOGORITHMS TO PREDICT THE RESULT OF A DATASET

Prepared by: Chirag Tejwani(18ucs088)
Piyush Soni(18ucs067)
Anubhav Kumar Goyal(18ucs066)

Submitted To: Dr. Subrat Dash
Dr. Sakthi Balan
Dr. Sudheer Sharma

02

# DATASET DETAILS

Our Dataset is the collection of many instances of WHITE WINE samples that are GRADED FOR THEIR QUALITY on the basis of many factors/attributes.

# INSTANCES AND ATTRIBUTES

Our Dataset had 4898 instances / wine samples and had 12 columns/Attributes that described the quantity of the following attribute in that white wine sample

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol
- Output variable (based on sensory data):
- quality (score between 0 and 10)

03

# OBJECTIVE

The Aim of this report is to predict the quality of the White Wine sample by training our model using three different algorithms.

# ALGORITHMS USED

We Used three Classifiers to Train and Test our Predictions

- Decision Tree
- K- Nearest Neighbour
- Naive Bayes

04

# PREPROCESSING AND IMPORTING LIBRARIES

**We imported the following libraries for the implementation**

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

**Next we fed the dataset with the help of the pandas library read csv command**

```
data = pd.read_csv('winequality-white.csv',
                   names=["fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free s
                          "total sulfur dioxide","density","pH","sulphates","alcohol","quality"])
```

**The Shape of the dataset was checked by the .shape command and the .head() command displayed a few starting rows.**

```
data.shape
```

```
(4898, 12)
```

```
data.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 |

# 05

**Information about the data was gathered by .info() command**

```
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

**Here we first make a copy of the data ,then we sperate the dependent variable (the quality of the wine) and the independent variable.**

```
clean_data = data.copy()
```

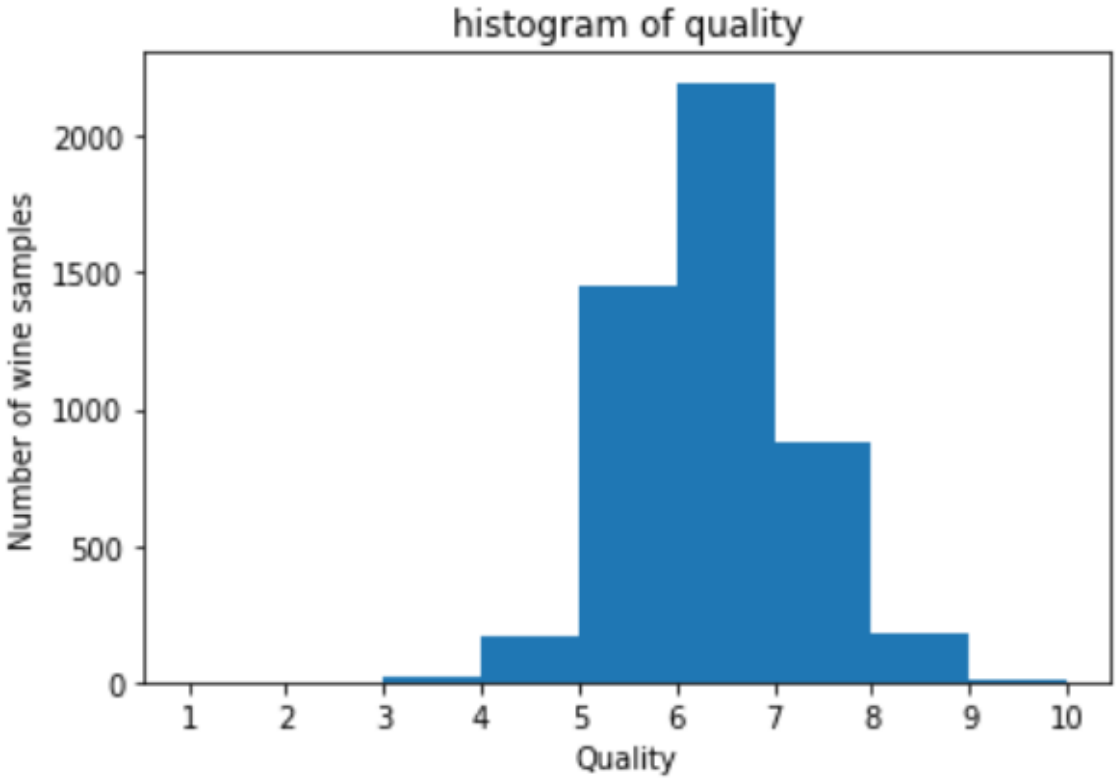
```
# here we copy the dependent variable from the dataset
y = clean_data["quality"].copy()
rest_val = ["fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sul
#now we seperate the rest of the independent variables in a variable x
x = clean_data[rest_val].copy()
```

**To get an idea about the distribution of the data a histogram was plotted depicting no of samples corresponding to each quality index**

The x axis shows the quality index and the y shows the no of wine samples

# 06

```
fig,ax = plt.subplots(1,1)
a = np.array(data.quality)
ax.hist(a, bins = [1,2,3,4,5,6,7,8,9,10])
ax.set_title("histogram of quality")
ax.set_xticks([1,2,3,4,5,6,7,8,9,10])
ax.set_xlabel('Quality')
ax.set_ylabel('Number of wine samples')
plt.show()
```



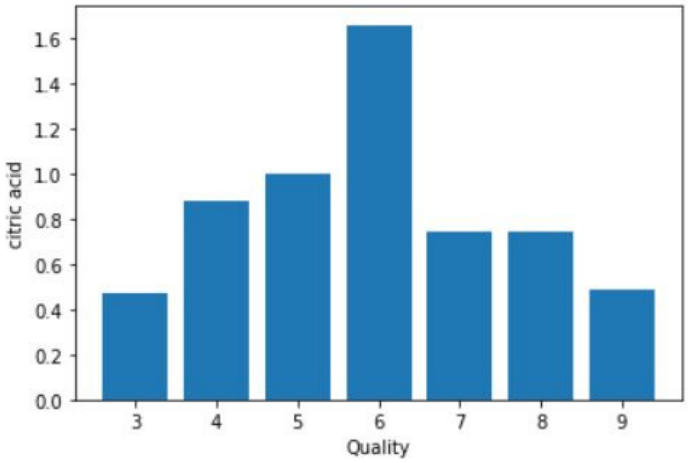

The minimum quality was 3 and maximum was 9

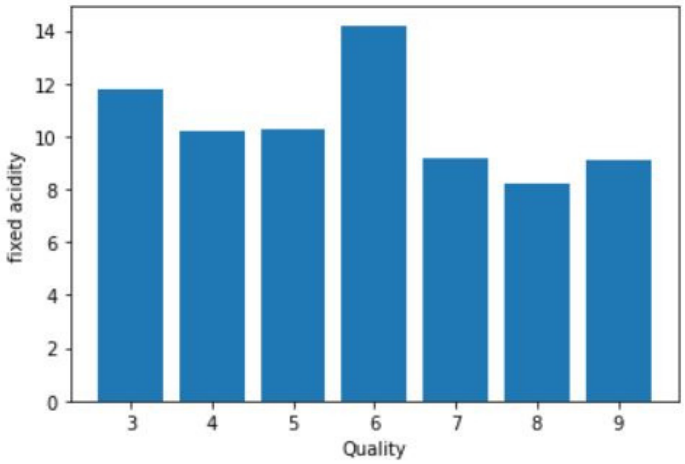
# 07

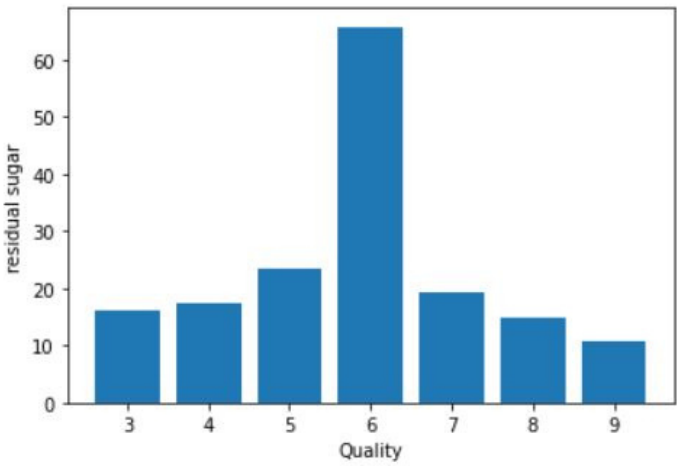
## We plotted the bar graph of each independent variable vs the dependent variable

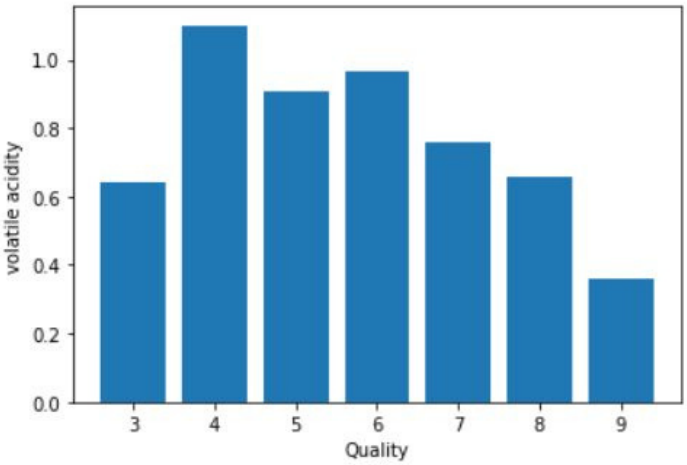
```
temp = ["fixed acidity","volatile acidity","citric acid","residual sugar","chlorides","free sulfur dioxide","total sulfur dioxide","density","pH","sulphates","alcohol"]
```

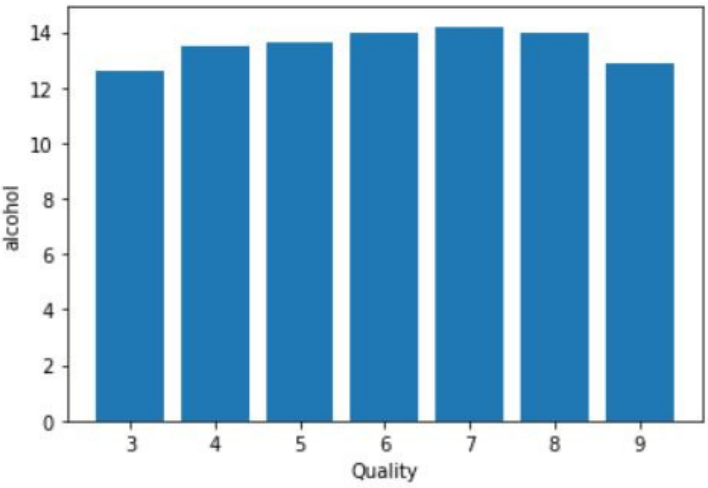
```
for i in temp:
    plt.bar(clean_data['quality'],clean_data[i])
    plt.xlabel('Quality')
    plt.ylabel(i)
    plt.show()
```



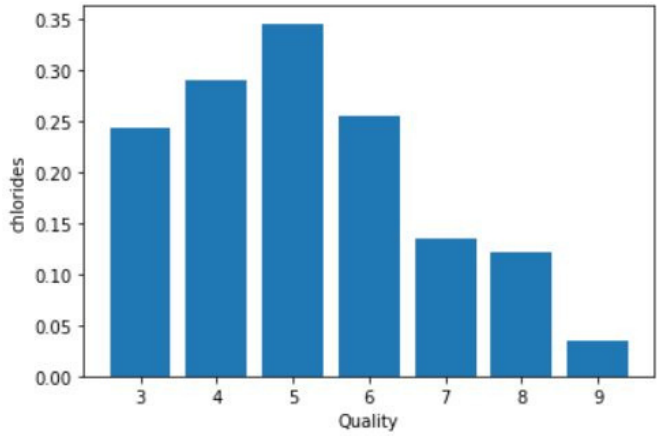



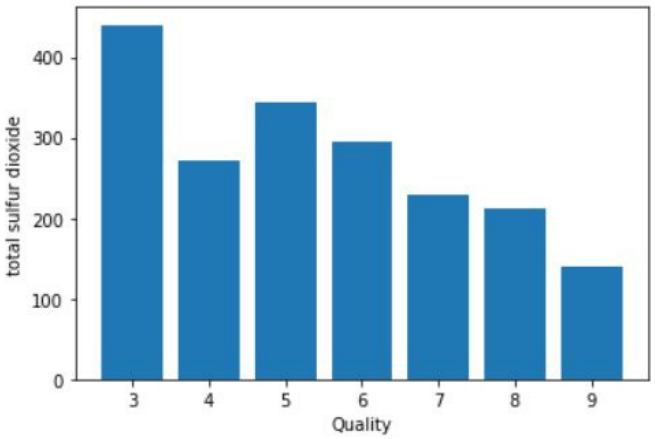



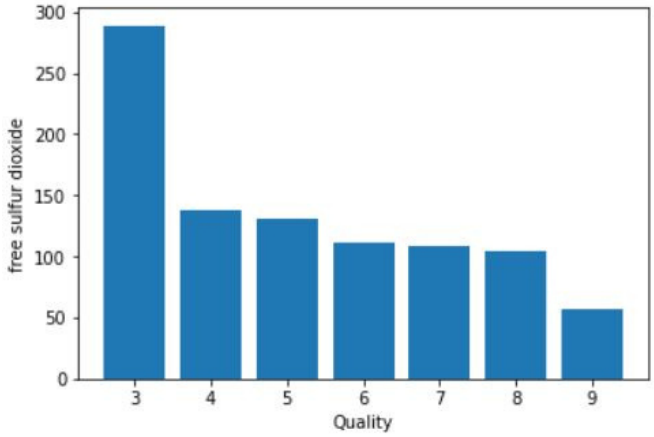



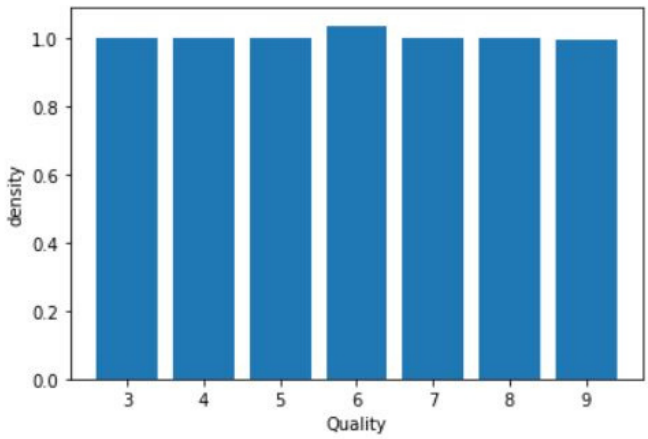



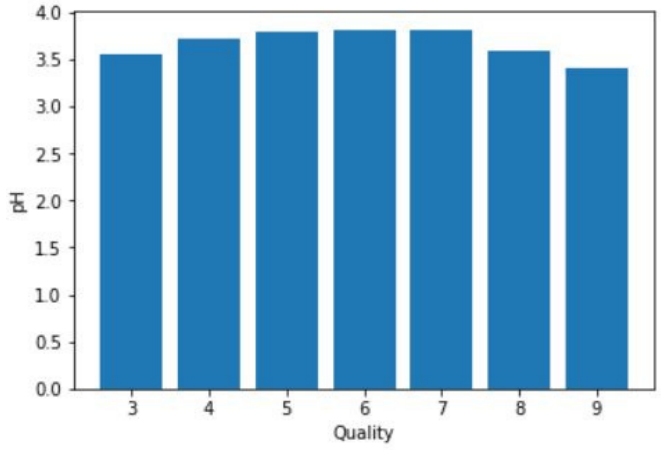
# 08

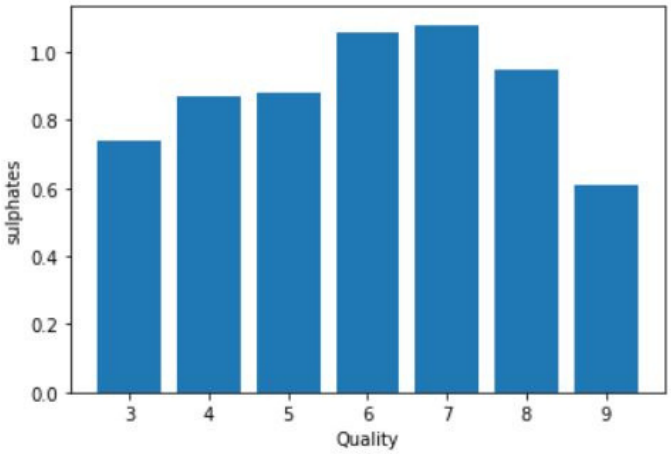

chlorides
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
3 4 5 6 7 8 9
Quality


total sulfur dioxide
400
300
200
100
0
3 4 5 6 7 8 9
Quality


free sulfur dioxide
300
250
200
150
100
50
0
3 4 5 6 7 8 9
Quality


density
1.0
0.8
0.6
0.4
0.2
0.0
3 4 5 6 7 8 9
Quality


pH
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
3 4 5 6 7 8 9
Quality

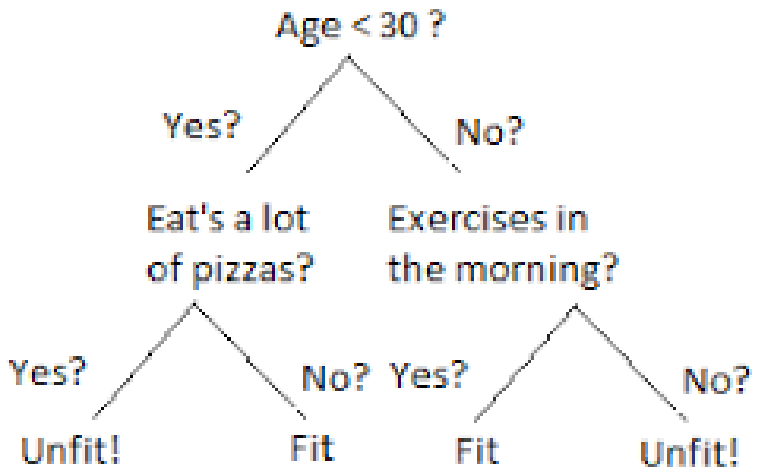

sulphates
1.0
0.8
0.6
0.4
0.2
0.0
3 4 5 6 7 8 9
Quality

09

# DECISION TREE CLASSIFIER

A Decision tree classifier is a very simple and widely used classifier to classify the dataset into various classes. It can simply be thought as a series of question/tests being conducted on the dataset to split the database into various branches that would follow more such test/questions to reach a final class/group.

A Decision Tree Classifier is a Tree Structured Classifier that classifies our data on the basis of the tests conducted at the intermediate nodes of the tree.




The above shown image is an example of such a decision tree.

In general understanding, The classification algorithm generates a tree structure on the basis of the input dataset . The sklearn module which we are using is going to use Gini split and Gini index to divide the groups into further smaller groups. The Decision tree is then used to classify the data. There are two types of nodes the decision tree

- Intermediate/Testing Nodes
- Leaf Nodes /Classes

The Dataset is fed into the root of the tree the intermediate or the testing nodes then segregate the data based on a specific test that is done on the attributes of the data. The process continues till instances reach the leaf nodes which are nothing but classes.

10

# IMPLEMENTING THE DECISION TREE ALOGORITHM

**For the implementation of the decision tree algorithm we first divided our dataset into training dataset (80%) and testing dataset (20%) we then fit the data over the modal generated and train the model.**

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state=0)
Quality_classifier = DecisionTreeClassifier(max_leaf_nodes=14, random_state=0)
Quality_classifier = Quality_classifier.fit(x_train,y_train)
```

**We then predict the results using our trained model over the testing dataset**

```
y_predicted = Quality_classifier.predict(x_test)
```

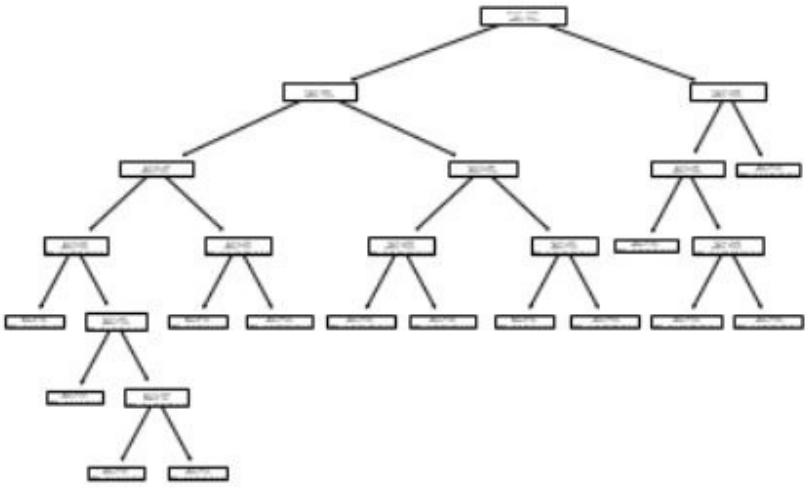
**Finally the accuracy was calculated based on the correct and the predicted results**

```
print("accuracy is",accuracy_score(y_test, y_predicted)*100)
```

```
accuracy is 49.48979591836735
```

**We plotted the decision tree**

```
tree.plot_tree(Quality_classifier)
```



11

# KNN CLASSIFIER

The KNN classifier is a very simple classification algorithm also known as local classifier and is not a exactly a learning classifier but it is rather a computational classifier that classifies an instance based on its nearest neighbors

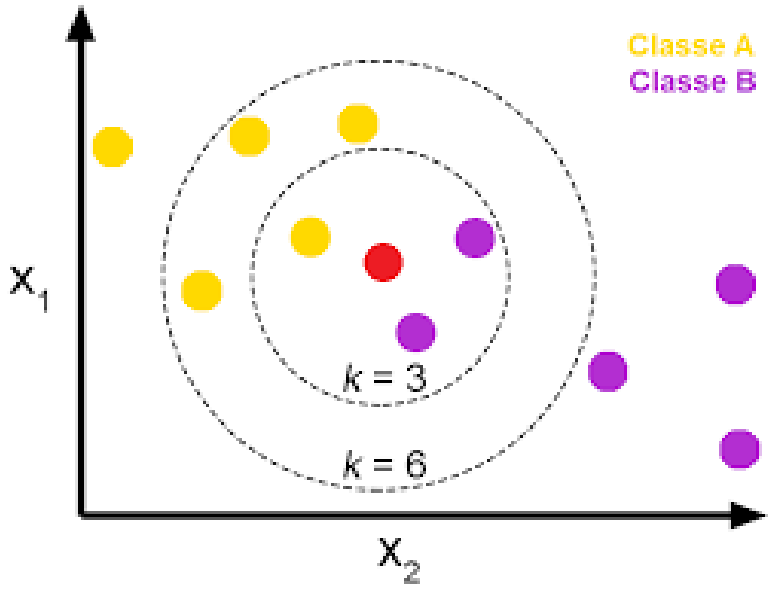
To determine the class of an instance the algorithm simply looks at its k-nearest neighbors the class of the instance is the class that is prominent in its neighborhood

Now one can easily notice the importance of number of neighbors to be analyzed to take the above mentioned decision there may occur two cases:

- If k is very low then our algorithm is very sensitive to noise
- On the other hand if k is very high than this can lead to containing of points from other classes too and hence inconsistent results.

Some important points to consider for the KNN classifier

- We should always select odd values of k for a 2 class problem
- k must not be a multiple of the number of classes
- if k is very high our decision would be more accurate but will increase the time complexity of the algorithm too.



12

# KNN CLASSIFIER

For example in the above shown figure there are two classes A and B which are prepopulated. Now there is a red node/instance that has to be classified now the red node looks at k nearest neighbors (in this case k=3) now there are two neighbors that belong to the Class B and only one neighbor of the Class A.

So the red instance is classified as Class B as Class B is dominant in the neighborhood of the red node/instance.

### We plotted a correlation Matrix using the following code

```
corr_metrics = x.corr()
corr_metrics.style.background_gradient()
```

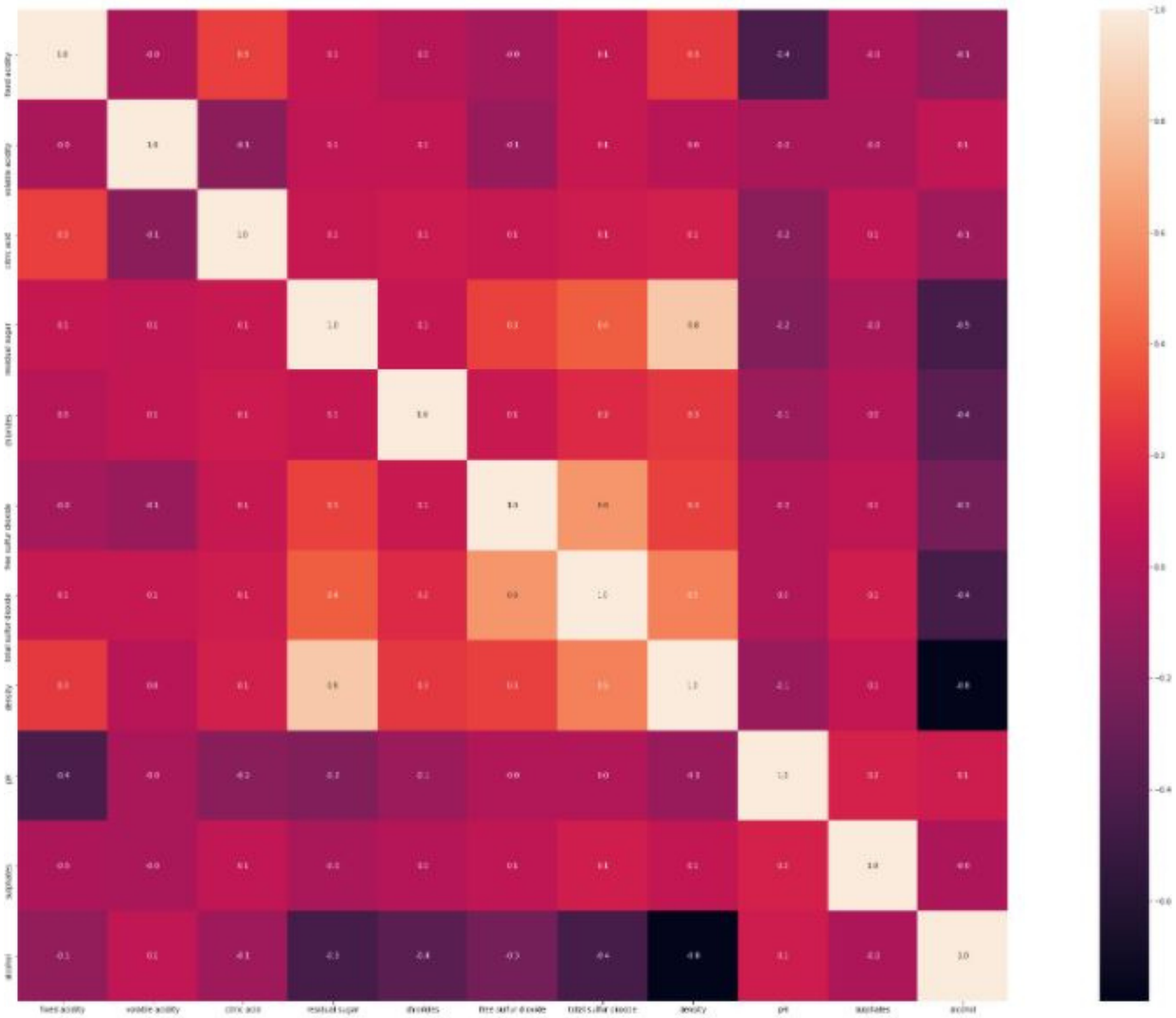
## Correlation Matrix

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcohol |
|---|---|---|---|---|---|---|---|---|---|---|---|
| fixed acidity | 1.000000 | -0.022697 | 0.289181 | 0.089021 | 0.023086 | -0.049396 | 0.091070 | 0.265331 | -0.425858 | -0.017143 | -0.120881 |
| volatile acidity | -0.022697 | 1.000000 | -0.149472 | 0.064286 | 0.070512 | -0.097012 | 0.089261 | 0.027114 | -0.031915 | -0.035728 | 0.067718 |
| citric acid | 0.289181 | -0.149472 | 1.000000 | 0.094212 | 0.114364 | 0.094077 | 0.121131 | 0.149503 | -0.163748 | 0.062331 | -0.075729 |
| residual sugar | 0.089021 | 0.064286 | 0.094212 | 1.000000 | 0.088685 | 0.299098 | 0.401439 | 0.838966 | -0.194133 | -0.026664 | -0.450631 |
| chlorides | 0.023086 | 0.070512 | 0.114364 | 0.088685 | 1.000000 | 0.101392 | 0.198910 | 0.257211 | -0.090439 | 0.016763 | -0.360189 |
| free sulfur dioxide | -0.049396 | -0.097012 | 0.094077 | 0.299098 | 0.101392 | 1.000000 | 0.615501 | 0.294210 | -0.000618 | 0.059217 | -0.250104 |
| total sulfur dioxide | 0.091070 | 0.089261 | 0.121131 | 0.401439 | 0.198910 | 0.615501 | 1.000000 | 0.529881 | 0.002321 | 0.134562 | -0.448892 |
| density | 0.265331 | 0.027114 | 0.149503 | 0.838966 | 0.257211 | 0.294210 | 0.529881 | 1.000000 | -0.093591 | 0.074493 | -0.780138 |
| pH | -0.425858 | -0.031915 | -0.163748 | -0.194133 | -0.090439 | -0.000618 | 0.002321 | -0.093591 | 1.000000 | 0.155951 | 0.121432 |
| sulphates | -0.017143 | -0.035728 | 0.062331 | -0.026664 | 0.016763 | 0.059217 | 0.134562 | 0.074493 | 0.155951 | 1.000000 | -0.017433 |
| alcohol | -0.120881 | 0.067718 | -0.075729 | -0.450631 | -0.360189 | -0.250104 | -0.448892 | -0.780138 | 0.121432 | -0.017433 | 1.000000 |

# 13

## We plotted a heatmap bassed on the correlation matrix

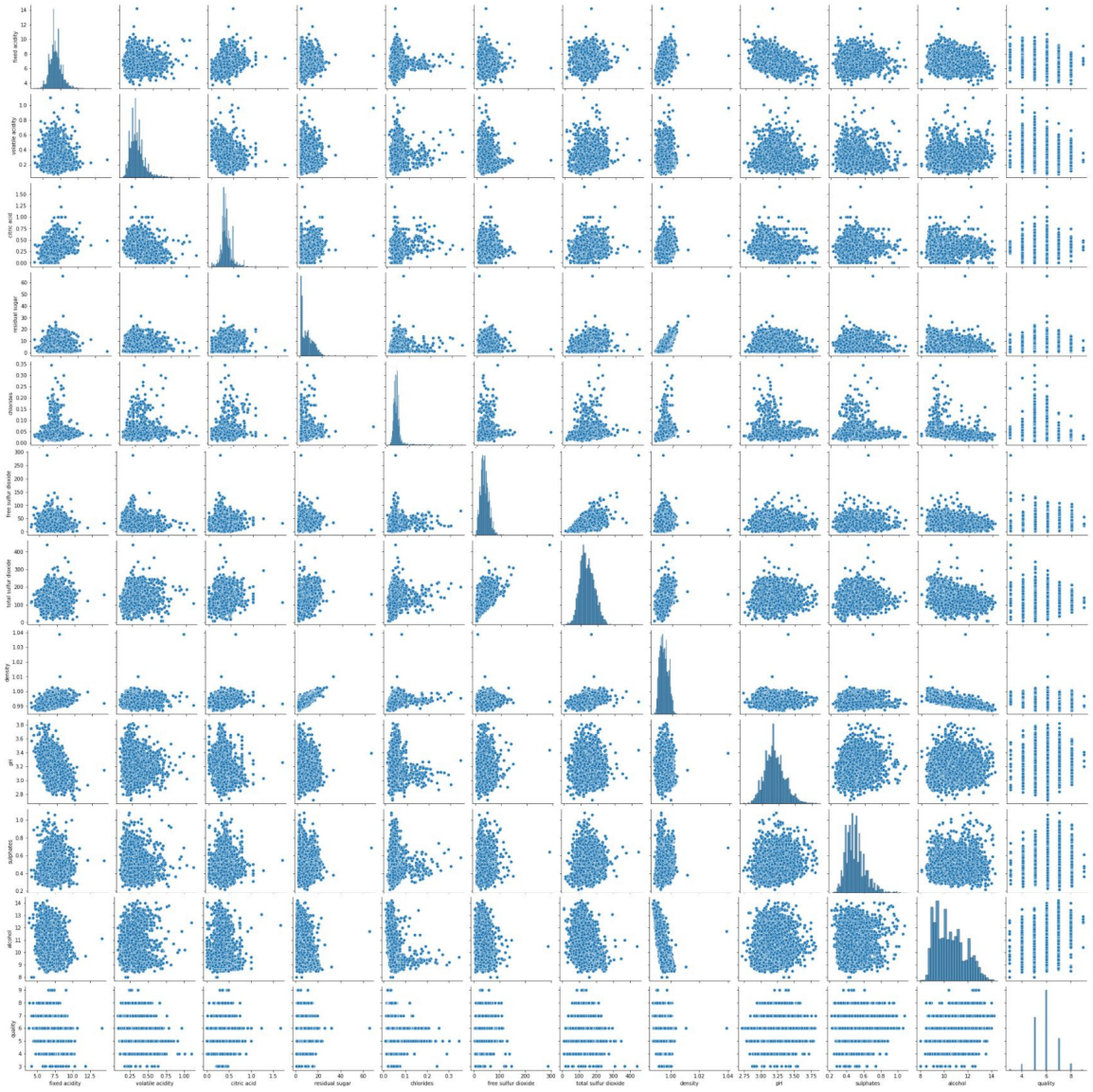
```
plt.subplots(figsize=(45,25))
sns.heatmap( corr_metrics, square=True, annot=True, fmt=".1f" )
```



# 14

## We plotted the scatter plots of the Clean_data dataset

```
sns.pairplot(clean_data)
```



15

# IMPLEMENTING KNN CLASSIFIER

**We varied k (number of neighbors) from 1 to 14 ,in each of the cases, then we trained a model based on KNeighborsClassifier provided in the sklearn module and then tested it for the x_test values and stored the result in predicted. Then to check the accuracy of the model trained on the K neighbours we calculated the accuracy_score for the y_test vs predicted. now we print our values of accuracy for each K neighbours model.**

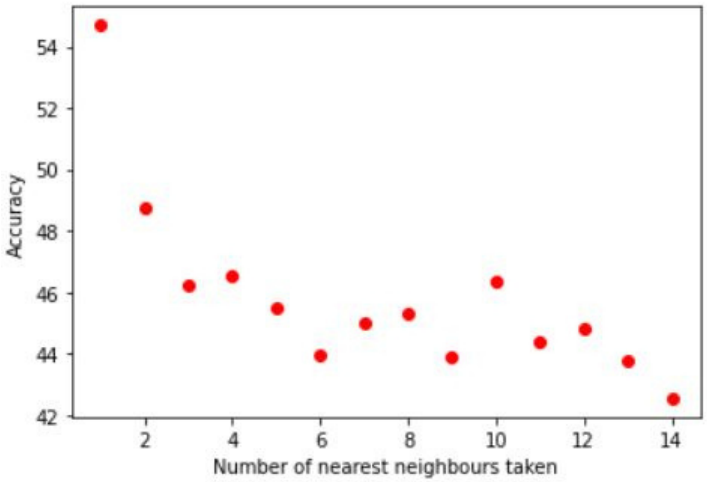
```
l = []
for i in range(1,15):
    knnclf = KNeighborsClassifier(n_neighbors = i)
    knnclf = knnclf.fit(x_train,y_train)
    predicted = knnclf.predict(x_test)
    l.append(accuracy_score(y_test, predicted)*100)
for i in range(len(l)):
    l[i] = int(l[i]*1000)/1000
print(l)
```

**These are the accuracy values for all the respective k values**

```
[54.693, 48.775, 46.224, 46.53, 45.51, 43.979, 45.0, 45.306, 43.877, 46.326, 44.387, 44.795, 43.775, 42.551]
```

**We plotted a graph between k values and the accuracy**

```
plt.plot([1,2,3,4,5,6,7,8,9,10,11,12,13,14], l,'ro')
plt.ylabel("Accuracy")
plt.xlabel("Number of nearest neighbours taken")
plt.show()
```




```
#here we understand that the maximum accuracy will be at 1
```

16

# NAIVE BAYES

**The NAIVE BAYES classifier is a probabilistic classifier and is based upon the below illustrated bayes theorem**

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

$A, B$ = events
$P(A|B)$ = probability of A given B is true
$P(B|A)$ = probability of B given A is true
$P(A), P(B)$ = the independent probabilities of A and B

Now the above formula is only applicable when we have only 2 events A B and gives the probability of A when B is already true. If this formula is generalized for many events then it can be used for our purpose i.e. classification.

An Important assumption that naive bayes algorithm takes is that each feature or attribute is independent of each other. For Example : A person can be tall fair and rich. These all attributes are independent of each other.

In general terms, we calculate the probability of a single instance to belong to a particular class when all the other attributes given are true.

If we have multiple classes then we follow the above procedure for every on a single instance followed by normalizing the results calculated. The class that gives the maximum probability is assigned to the instance.

We will be using the GaussianNB classifier provided by the sklearn library to predict the data on the basis of Naive Bayes classification.

17

# IMPLEMENTING NAIVE BAYES

### We trained a classifier based on GaussianNB algorithm

```
clfNB = GaussianNB()
clfNB.fit(x_train,y_train)
```

### Based on the classifier we predicted the values of x_test and stored it in predicty variable

```
predicty = clfNB.predict(x_test)
```

### We made a confusion_matrix based on the predicty variable

```
cm = confusion_matrix(y_test,predicty)
print(cm)
```

```
[[  3   3   0   2   1   0]
 [  1  11  17  18   4   0]
 [  2  11 149 107  26   0]
 [  6   4 101 154 143   1]
 [  0   0  18  46 116   3]
 [  0   0   1  12  19   1]]
```

### We calculated the accuracy score with the accuracy_score function for the Naive Bayes

```
print("accuracy score is "+str(round(accuracy_score(y_test, predicty)*100)))
```

```
accuracy score is 44
```

### We created classification report as follows:

```
print("classification report \n"+str(classification_report(y_test,predicty)))
```

```
classification report
              precision    recall  f1-score   support

           3       0.25      0.33      0.29         9
           4       0.38      0.22      0.28        51
           5       0.52      0.51      0.51       295
           6       0.45      0.38      0.41       409
           7       0.38      0.63      0.47       183
           8       0.20      0.03      0.05        33

    accuracy                           0.44       980
   macro avg       0.36      0.35      0.33       980
weighted avg       0.45      0.44      0.43       980
```

18

# IMPLEMENTING NAIVE BAYES

**We trained a classifier based on GaussianNB algorithm**

```
clfNB = GaussianNB()
clfNB.fit(x_train,y_train)
```

**Based on the classifier we predicted the values of x_test and stored it in predicty variable**

```
predicty = clfNB.predict(x_test)
```

**We made a confusion_matrix based on the predicty variable**

```
cm = confusion_matrix(y_test,predicty)
print(cm)
```

```
[[  3   3   0   2   1   0]
 [  1  11  17  18   4   0]
 [  2  11 149 107  26   0]
 [  6   4 101 154 143   1]
 [  0   0  18  46 116   3]
 [  0   0   1  12  19   1]]
```

**We calculated the accuracy score with the accuracy_score function for the Naive Bayes**

```
print("accuracy score is "+str(round(accuracy_score(y_test, predicty)*100)))
```

```
accuracy score is 44
```

**We created classification report as follows:**

```
print("classification report \n"+str(classification_report(y_test,predicty)))
```

```
classification report
              precision    recall  f1-score   support

           3       0.25      0.33      0.29         9
           4       0.38      0.22      0.28        51
           5       0.52      0.51      0.51       295
           6       0.45      0.38      0.41       409
           7       0.38      0.63      0.47       183
           8       0.20      0.03      0.05        33

    accuracy                           0.44       980
   macro avg       0.36      0.35      0.33       980
weighted avg       0.45      0.44      0.43       980
```

# 19 REFERENCES

The data set was taken from: UCI Machine Learning Repository: Wine Quality Data Set

HTTPS://SCIKIT-LEARN.ORG/STABLE/MODULES/GENERATED/SKLEARN.NAIVE_BAYES.GAUSSIANNB.HTML

HTTPS://SCIKIT-LEARN.ORG/STABLE/MODULES/GENERATED/SKLEARN.NEIGHBORS.KNEIGHBORSCLASSIFIER.HTML

HTTPS://SCIKIT-LEARN.ORG/STABLE/MODULES/GENERATED/SKLEARN.TREE.DECISIONTREECLASSIFIER.HTML

HTTPS://PANDAS.PYDATA.ORG/

HTTPS://MATPLOTLIB.ORG/

**NumPy**

Why NumPy? Powerful n-dimensional arrays. Numerical computing tools. Interoperable. Performant. Open source.

numpy.org

# CONCLUSION:

> *"Data is the oil of the 21st century."*

We were able to understand and dig in the dataset that we chose and were able to apply few machine learning classifiers to bring out as many inferences as possible from the dataset.

The bar charts give a very good about the dependence of quality on various attributes.

The correlation matrix and the heat map provide a good idea about the relation between the different attributes

According to our implementations the decreasing sequence of accuracy percentage is as follows:

KNN > Decision tree > Naïve Bayes