PROJECT TITLE: WINE QUALITY CLASSIFICATION

SUPERVISED CLASSIFICATION

Project overview:

The Project I have chosen is Wine quality classification. The aim of this project is to use supervised learning on the wine dataset for getting whether the wine is red or white. This is a binary classification algorithm it takes all the information regarding the wine and tells that whether the color off the wine is either red or white. As the wine industry is in the raise the price of the wine depends on the qualities such as pH, acidity, presence of the sugar and some other qualities and alsothe human quality assurance is also needed for the controlled assurance process. The Data is based on the 12 different properties of the wine containing about 4898-white wine and 1599-red wine.
This project dataset is taken from the github
https://raw.githubusercontent.com/devmood/intel-ai-academy/master/Wine_Quality_Data.csv

Problem Statement:

The aim of this project is to predict the wine color by using the properties that are given in the dataset. For this I choose the data set from the github.First thing that has to do is that data preprocessing i.e checking whether the data is containing any null values and deleting them as our data does not contain any null values it is not needed. After that we checked for the output division i.e whether the data is divided evenly or not.Then after all the noise is deleted we trained the dataset ,tested to find the accuracy score and after that using grid search cv we find out the

the parameters that are needed to improve it and then using those parameters we improved the accuracy. This is all done using the classification models

Metrics:

As our data is an imbalanced data and there is an equal importance to the precision and also the recall so we choose to use the f1_score as a metric.F1_score is metric which is the harmonic mean of the precision and recall i.e

F1_score = precision*recall/precision+recall

Precision is ratio of the correctively predicted positive observations to the total no of the positive predicted observations

Recall is the ratio of the correctly predicted to the total no of observations

F1_score is more useful than the accuracy since the data is imbalanced the f1_score works better than any other especially when we use the uneven class distribution.

And i provide the clippings of the both the formulaes and each one individually.

The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Data Exploration:

The dataset is taken from the github
https://raw.githubusercontent.com/devmood/intel-ai-academy/master/Wine_Quality_Data.csv
The data contain mainly the characterstics of the wine which tells that weather the wine is red or white. The dataset contains mainly 12 attributes they are

- **Fixed acidity**
- **Volatile acidity**
- **Citric acid**
- **Residual sugar**
- **Chlorides**
- **Free sulfur dioxide**
- **Total sulfur dioxide**
- **Density**
- **pH**
- **Sulphates**
- **Alcohol**
- **Quality**

The dataset contains 6497 rows in those they are about 4898 - white wine containing and remaining 1599 - red colored the histogram displays that red and white wines

```
# checking the red counts and white value counts
target.value_counts()
```

```
1    4898
0    1599
Name: color, dtype: int64
```

We generated the correlation to check whether the exists or not form the graph we can clearly say the there is no correlation exists.

```
In [75]: data.corr()
```

Out[75]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates
fixed_acidity	1.000000	0.219008	0.324436	-0.111981	0.298195	-0.282735	-0.329054	0.458910	-0.252700	0.299568
volatile_acidity	0.219008	1.000000	-0.377981	-0.196011	0.377124	-0.352557	-0.414476	0.271296	0.261454	0.225984
citric_acid	0.324436	-0.377981	1.000000	0.142451	0.038998	0.133126	0.195242	0.096154	-0.329808	0.056197
residual_sugar	-0.111981	-0.196011	0.142451	1.000000	-0.128940	0.402871	0.495482	0.552517	-0.267320	-0.185927
chlorides	0.298195	0.377124	0.038998	-0.128940	1.000000	-0.195045	-0.279630	0.362615	0.044708	0.395593
free_sulfur_dioxide	-0.282735	-0.352557	0.133126	0.402871	-0.195045	1.000000	0.720934	0.025717	-0.145854	-0.188457
total_sulfur_dioxide	-0.329054	-0.414476	0.195242	0.495482	-0.279630	0.720934	1.000000	0.032395	-0.238413	-0.275727
density	0.458910	0.271296	0.096154	0.552517	0.362615	0.025717	0.032395	1.000000	0.011686	0.259478
pH	-0.252700	0.261454	-0.329808	-0.267320	0.044708	-0.145854	-0.238413	0.011686	1.000000	0.192123
sulphates	0.299568	0.225984	0.056197	-0.185927	0.395593	-0.188457	-0.275727	0.259478	0.192123	1.000000
alcohol	-0.095452	-0.037640	-0.010493	-0.359415	-0.256916	-0.179838	-0.265740	-0.686745	0.121248	-0.003029
quality	-0.076743	-0.265699	0.085532	-0.036980	-0.200666	0.055463	-0.041385	-0.305858	0.019506	0.038485

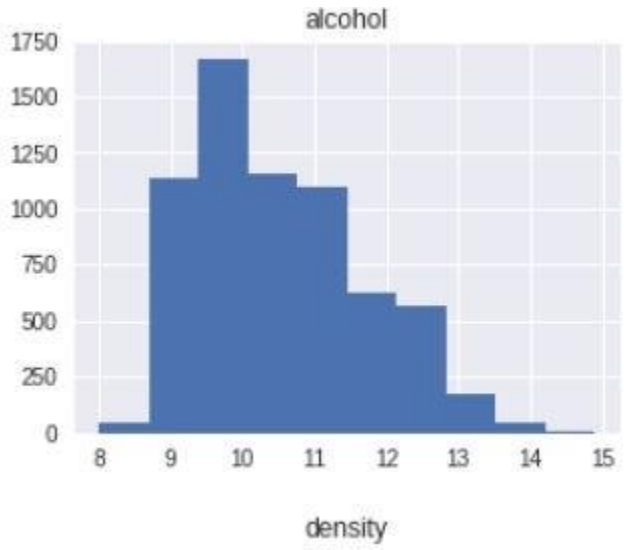
This data contains all the attributes and correlation to other attributes as there is no t much correlation exists we can go further.

Visualization:

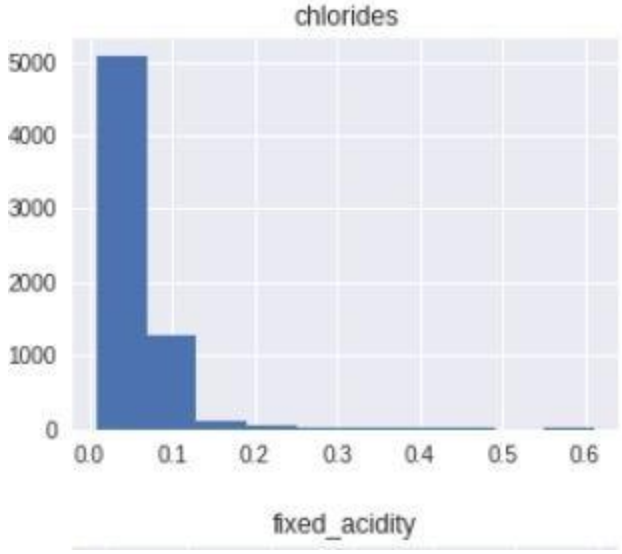
The following are the histogram visualizations for the attributes in the dataset this shows that the each attribute in the dataset is distributed into.

Histograms:

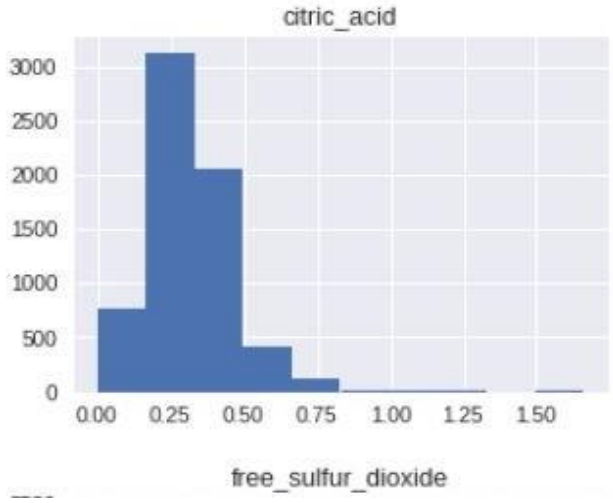
The following histogram shows that distribution of alchol.



The following distribution shows that the distribution of chlorides



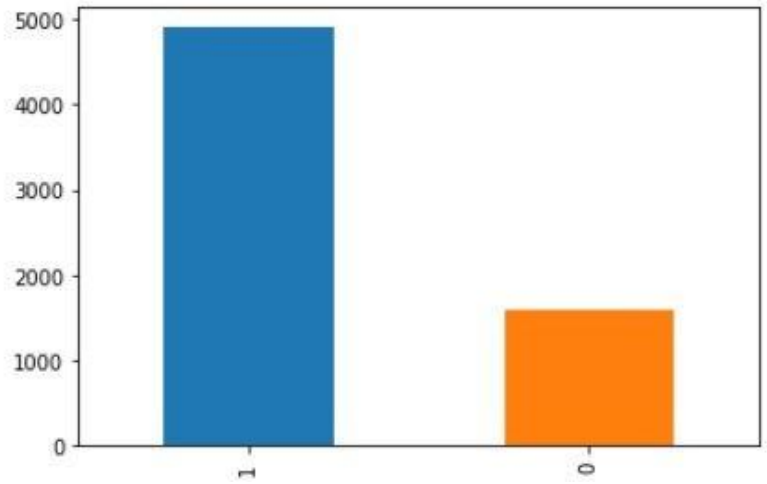
The following distribution shows that the distribution of citiric_acit



The visualisation of red-wine and white-wine clippings can be seen below

```
# plotting the no.of values of red wine and white wine by bar graphs
target.value_counts().plot.bar()
```

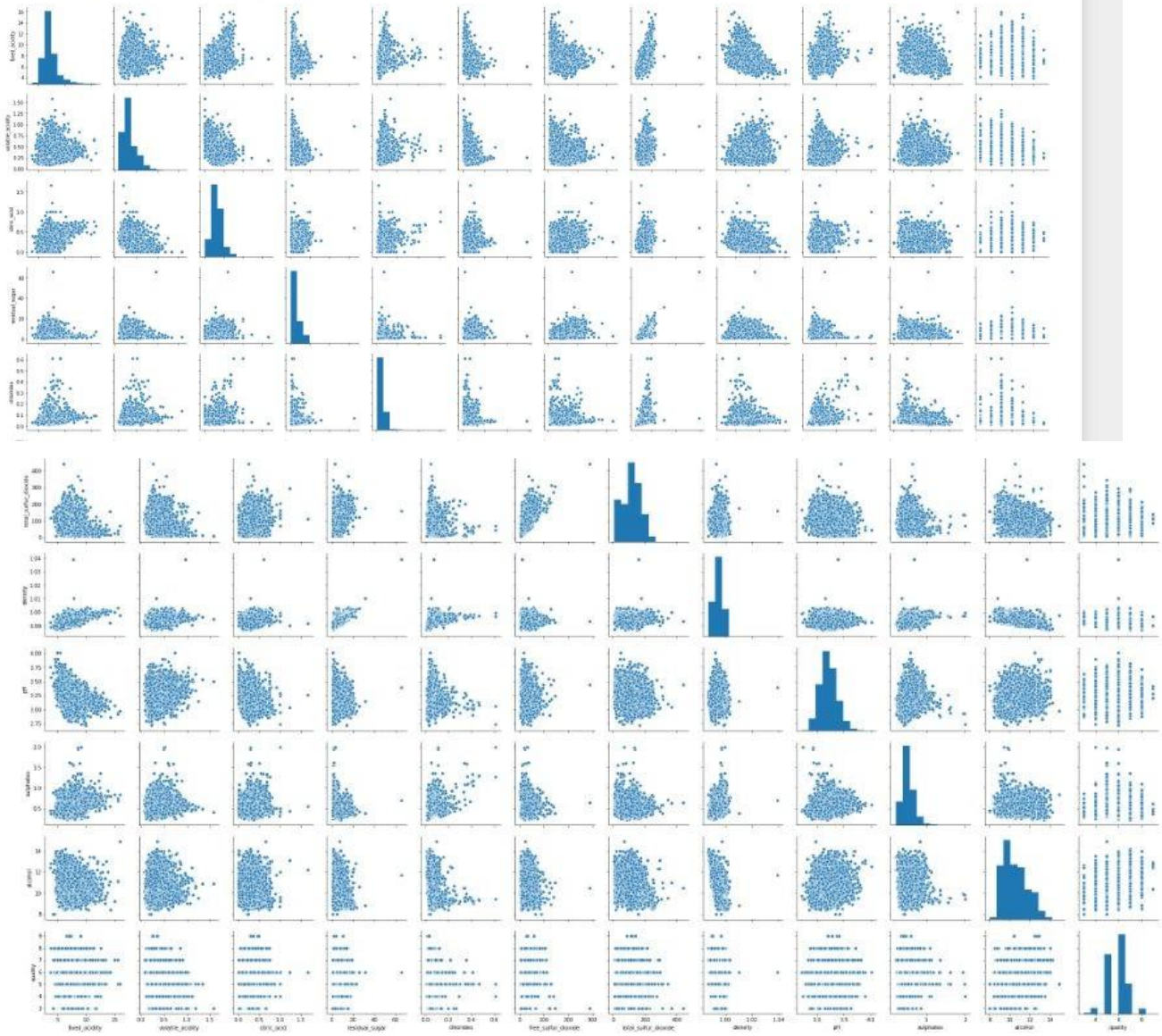
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1124dd7b8>
```



The pair plot of the data can be picturized below. It includes sepal length, sepal width, petal length and petal width of the given data. Here we can check the outliers of the data.

```
# shows pairplot for the dataset
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x1a109ec2f60>
```



Algorithms and Techniques:

The algorithms that are used to make predictions are

- K Nearest Neighbour
- Gaussian NB
- LogisticRegression

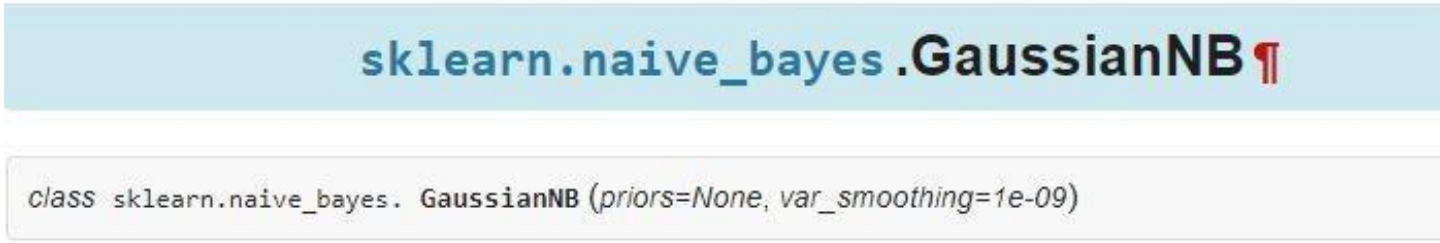
- Decision Tree

K Nearest Neighbour:

K Nearest Neighbour is used in case of the classification and regression .It mainly depends upon the K value if the k value is 3 then it looks only the three nearest neighbours.
It works fine even when the outliers are present it is a robust algorithm.it woks fine even when the training data is large.
Without the parameter k the algorithm does not work fine for all data.It is tough to find out which distance gives the best results

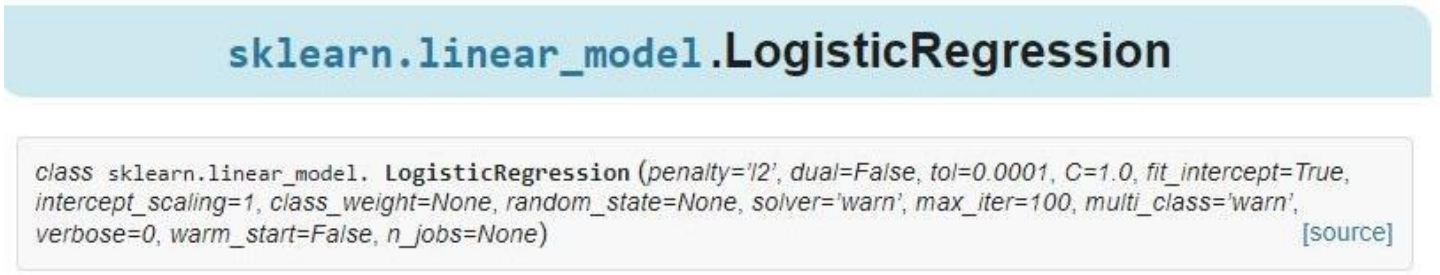
Gaussian NB:

Naive Bayes is a simple but surprisingly powerful algorithm for predictive modeling
It mainly based on the Naive Bayes algorithm it gives great results in case of the textual data
It is easy to train on the small data and it is fast and highly scalable.It is the best choice in case of the textual data .Its predications make very strong analysis.
It is best only for small datasets in case of large dataset it does not work well.

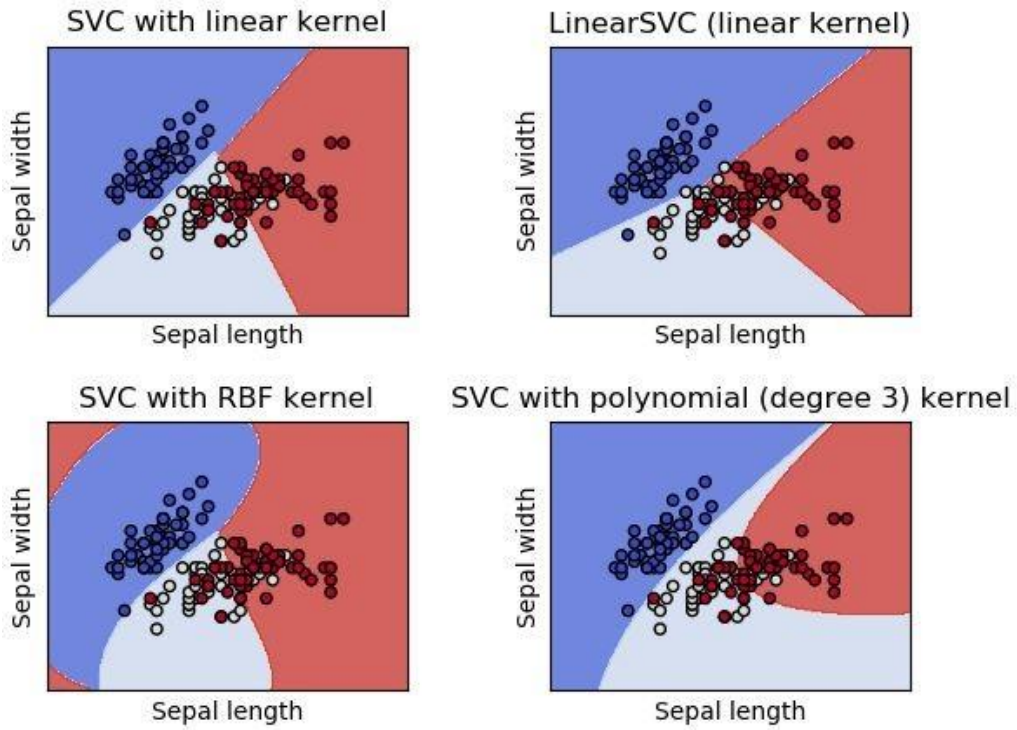


LogisticRegression:

Logistic regression is a predictive analysis.It is appropriate to calculate when dependent variable is binary it mainly basis on log values.
The model is robust when even the variance of data in each group is not equal.It can handle nonlinear effects.
It is best only in case of the numerical data but not in categorical data and Logistic regression attempts to predict outcomes based on a set of independent variables



Linear SVC:



sklearn.svm.LinearSVC

class sklearn.svm. **LinearSVC** (*penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000*) ¶ [source]

Linear Support Vector Classification.

Benchmark:

Here i used bench mark model for my dataset is KNearestNeighbour classifier. By using this model i got an f1_score of 0.96294064477252195. I will compare this benchmark model with other algorithms and I will compare the remaining algorithms. If any algorithm's f1_score is greater the benchmark model, or got an accuracy more than that of the bench mark model, then i will take that particular algorithm to consideration.

Data Preprocessing:

The data preprocessing that done for this model are first i have checked for the null values using isnull().sum() then i found that there are no null values in the data hence no null values are removed .

```
#Loading dataset to data
data = pd.read_csv('wine.csv')
```

```
1 data.isnull().sum()
```

```
fixed_acidity           0
volatile_acidity        0
citric_acid             0
residual_sugar          0
chlorides               0
free_sulfur_dioxide     0
total_sulfur_dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
color                   0
dtype: int64
```

Then i performed the data.describe() to see mean,std,min,max and they are any vlaues that are out range values so that ican perform scaling they are no outranging values hence no scaling has to be performed.

```
1 data.describe()
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates
count	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
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000

Then after that we checked for the correlations if any they are no correlations hence the data is clean then after we copied the column that has to get predicated in to another and dropped the column .Later on that column we have white-4898 and red-1599,we changed these to the white-1 and red-0 and changed all the data in to numerical data using the data.get_dummies()

As our data is numerical there is no need of using get()_dummies.

```
# checking whether the values assigned to target or not
target.head(4898)
```

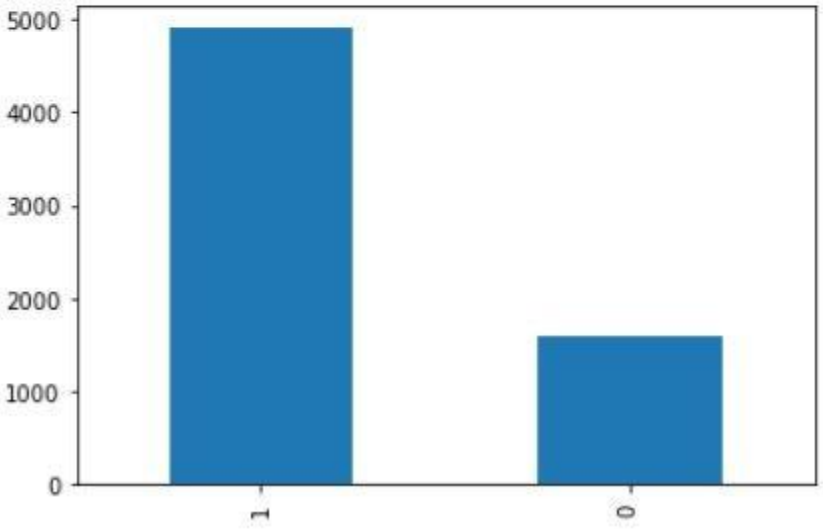
```
0       0
1       0
2       0
3       0
4       0
5       0
6       0
7       0
8       0

27      0
28      0
29      0
       ..
4868    1
4869    1
4870    1
4871    1
4872    1
4873    1
4874    1
4875    1
4876    1
4877    1
```

```
1  target.value_counts()
```

```
1    4898
0    1599
Name: color, dtype: int64
```

```
1  target.value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f12a9149828>
```



Implementation:

As coming to the Implementation , I splitted the dataset to training and testing. I will provide clippings. And I calculated the K Nearest Neighbour algorithm which gave an f1_score of 0.96294064477252195 by taking this as an benchmark model the remaining algorithms that are performed are Logistic Regression,Gaussian NB and Linear SVC.which gave an accuracy of 0.989435556897,0.982916412447 and 0.984440454817 ,I took Logistic regression to improve the F1_score as it already got about 98% accuracy .

Now we splitted the data in to training and testing with 33% for testing and 67% for the training and stored them in the variables X_train,X_test,y_train,y_test.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data,target, test_size=0.33, random_state=12)
```

```
from sklearn.linear_model import LogisticRegression
clf1 = LogisticRegression()
clf1.fit(X_train,y_train)
pred1 = clf1.predict(X_test)
print(f1_score(y_test,pred1))
```

```
0.9894355568970722
```

Then I improved the results by getting the best parameters using Grid Search CV . For this i imported the Grid Search CV and then make_scorer for the f1_score and then i passed some parameters in as list and got the best parameters using the Grid Search CV technique as it checks all the parameters and got an f1_score of 0.9903147699757869

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer
clf_5 = LogisticRegression()
parameters =  {'penalty':['l1','l2'],'C':[1.0,0.9,0.5,0.2]}
scorer = make_scorer(f1_score)
grid = GridSearchCV(clf_5,param_grid=parameters,scoring=scorer)
grid = grid.fit(X_train,y_train)
pred_5 = grid.predict(X_test)
print(f1_score(y_test,pred_5))
```

```
0.9903147699757869
```

The best estimators for the grid search cv are grid.best_estimator_
The best estimators are

```
1 grid.best_estimator_
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l1', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Then i splitted the data into 10 part and checked and calculated score for every part to know whether the model is giving accurate results are not and i got almost all the same from this i can say that the model is well trained.

```
1 from sklearn.model_selection import cross_val_score
2 cross_val_score(grid,data,target,cv=10,scoring=scorer)
```

```
array([0.98383838, 0.98987854, 0.99385246, 0.99182004, 0.99591002,
       0.9969419 , 0.99084435, 0.98370672, 0.99591837, 0.98677518])
```

Refinement:

The models that i used are K Nearest Neighbour ,Logistic Regression, Gaussian NB and Linear

In all the above I got Logistic Regression as the highest values when compared to the K nearest neighbour the benchmark model and also all the other algorithms. Initially we got the score of 0.9894355568970722 .Now to improve the score we used grid search cv technique with this technique we improved the parameters to the

```
1 from sklearn.model_selection import GridSearchCV
2 from sklearn.metrics import make_scorer
3 clf_5 = LogisticRegression()
4 parameters =  {'penalty':['l1','l2'],'C':[1.0,0.9,0.5,0.2]}
5 scorer = make_scorer(f1_score)
6 grid = GridSearchCV(clf_5,param_grid=parameters,scoring=scorer)
7 grid = grid.fit(X_train,y_train)
8 pred_5 = grid.predict(X_test)
9 print(f1_score(y_test,pred_5))
```

```
0.9903147699757869
```

We got a score of 0.9903147699757869 for this we took the best parameters as

C=1.0, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=100,
multi_class='ovr',n_jobs=1,penalty='l1',random_state=None,solver='liblinear',tol=0.0001,verbose=0, warm_start=False

These are best parameters we got for increasing the accuracy

```
1 grid.best_estimator_
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l1', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Results:

Model Evaluation and Validation:

The models i have chosen and their score values are
K Nearest Neighbour = 0.96294064477252195
Logistic Regression = 0.9894355568970722
Gaussian NB = 0.9829164124466138
Linear SVC = 0.9809410363311495
In the above model K nearest neighbour is taken as the Benchmark model and Logistic regression is taken for the further evaluation or to increase the model score. Finally the model score is increased to 0.9903147699757869.This is gained by the grid search cv model as it gave the parameters that are actually suited for the model to increase the score.The parameters that taken are C=1.0, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=100,multi_class='ovr',n_jobs=1,penalty='l1',random_state=None,solver='liblinear',tol=0.0001,verbose=0, warm_start=False

```
1 grid.best_estimator_
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l1', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

The to evaluate the robustness of the model we splitted the data into 10 equal parts and then tested and calculated the score for every set and we got

```
1 from sklearn.model_selection import cross_val_score
2 cross_val_score(grid,data,target,cv=10,scoring=scorer)
```

```
array([0.98383838, 0.98987854, 0.99385246, 0.99182004, 0.99591002,
       0.9969419 , 0.99084435, 0.98370672, 0.99591837, 0.98677518])
```

The every test case we got the score equal and is nearer to the 99% this shows the robustness of the model in case of the solution.

Justification:

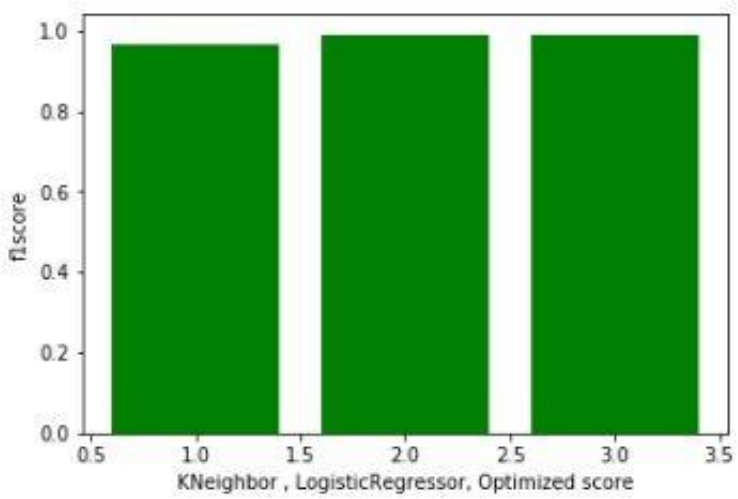
The Benchmark model(KNeighbour classifier) has obtained a f1_score of 0.96% , and the best algorithm we applied later LogisticRegression gices a f1_score of 0.98% which is greater than benchmark's score. So we came to know that our score increased from 0.96 to 0.98%, and we can take the results of LogisticRegression classifier as a best algorithm for this dataset.
We can observe the such a difference between KNeighbor classifier and LogisticRegression and then after we got an optimistic value of 0.99% by applying GridsearchCV to the dataset.

Conclusion:

Free form visualisation:

I considered my Benchmark model KNeighbor's classifier and the LogisticRegression and Optimised value 's and made a scores for all the f1_scores and plotted a bargraph for it.

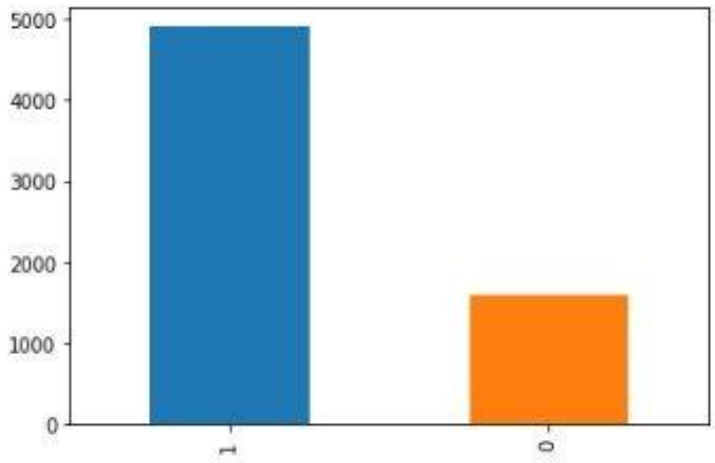
```
lab=[1,2,3]
f1score=[0.96294,0.9894,0.99031]
plt.bar(lab,f1score,color="green")
plt.xlabel("KNeighbor , LogisticRegressor, Optimized score")
plt.ylabel("f1score")
plt.show()
```



And i will visualize the bar graph of red wine and white wine below. As we can see the blue graph represents white wine, and orange color bar represents red wine.

```
target.value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1124dd7b8>
```



Reflection:

** The main aim of our project is to detect color of wine whether it is red wine or white wine by using classifier's.
** I managed the dataset to use the data preprocessing task as removing null values, identifying dummy variables and checked the noise, checked if categorical data exist. And checked the target feature color.
** I stored the result of target column and dropped the output column .
** We have splitted the dataset into training data and other part for testing data.
** I applied benchmark algorithm as K Neighbor's classification and other algorithms and checked the result obtained(f1_score).
** After the step, we performed metrics and taken the best score obtained from the algorithms.
** As this is my first capstone project, so i felt every step interesting.
Difficulties:
** The data is divided into 10 parts and tested, and checked whether the logistic regression is working on new data (Whether its accuracy is decreasing or not).
** I applied GridSearchCV and used parameter tuning and obtained best score of 99% for the Logistic Regression algorithm.

Improvement:

** As we are considering all features, we can also improve our dataset by reducing some of the features and can able to gain good result in accuracy.
** And further we can take new features such as quality of wine and can refer the best product based upon people ages.