SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
Approved by AICTE and Affiliated to the University of Mumbai

Department of Information Technology

Wine Quality Prediction

A mini-project submitted for

Business Intelligence Lab (Semester VI)

by

Name	Sap
Akanksha Mansharmani	60003180002
Janvi Desai	60003180020

Problem statement:
A wine that has diluted and faint flavours is hardly a superb wine. On the other hand, a wine that is concentrated, with defined and strong aromas, is more likely a quality one. Wine sales in the state rose by 1.2% during April 2019.IMFL sales were 1,624 bulk litres and wine sales were 54 bulk litres in April-December 2019. So once you know which quality factors to consider, you can proceed to making a judgment for the wine you have tasted. So this model predicts the quality of red wine.

Link to dataset : https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

Code:

```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
import plotly.express as px
```

Importing Libraries

```
ds=pd.read_csv("/content/dmbi.csv")
```

Loading and Reading the Dataset

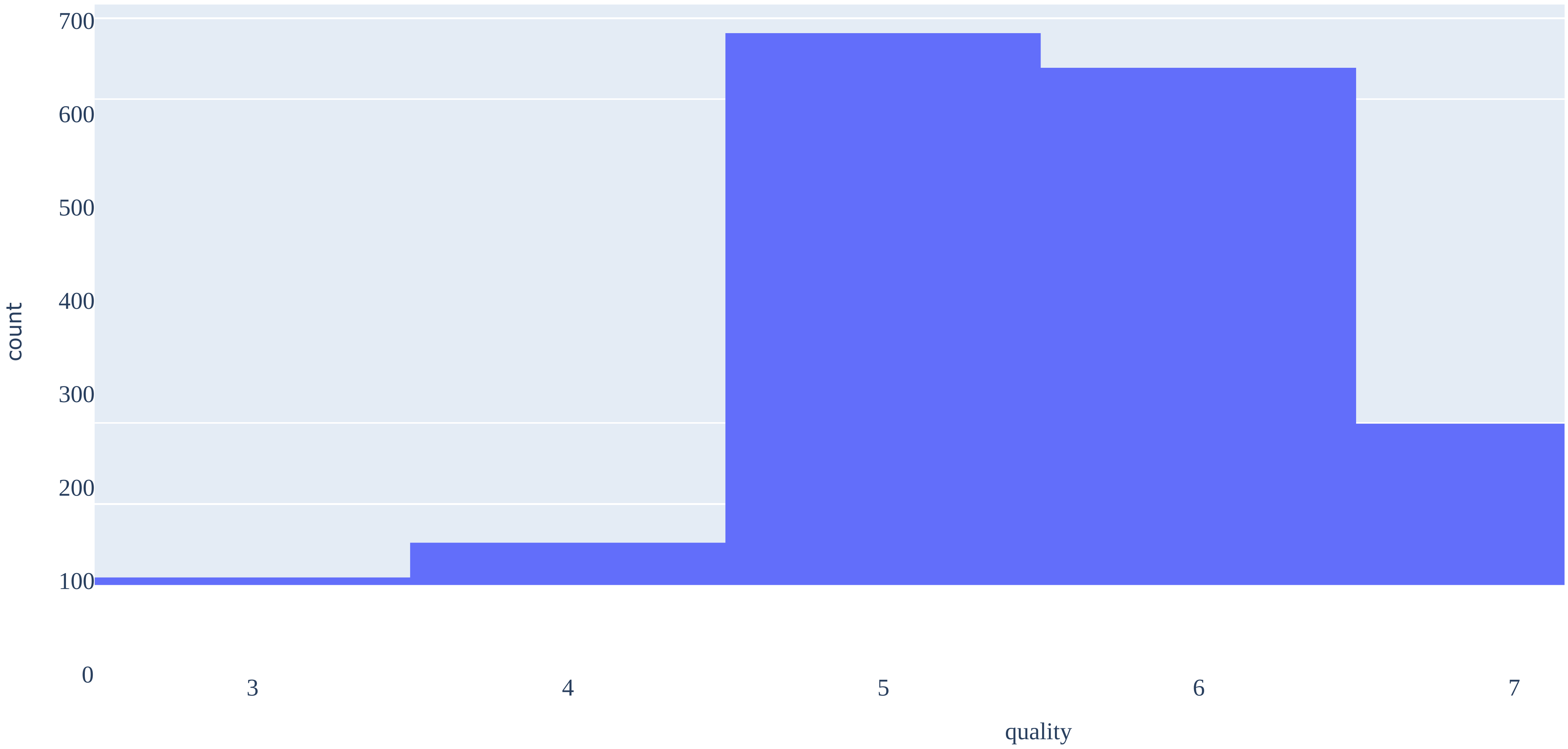
```
ds.head()
```

	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

```
ds.shape
```

```
(1599, 12)
```

```
histogram = px.histogram(ds,x='quality')
histogram.show()
```



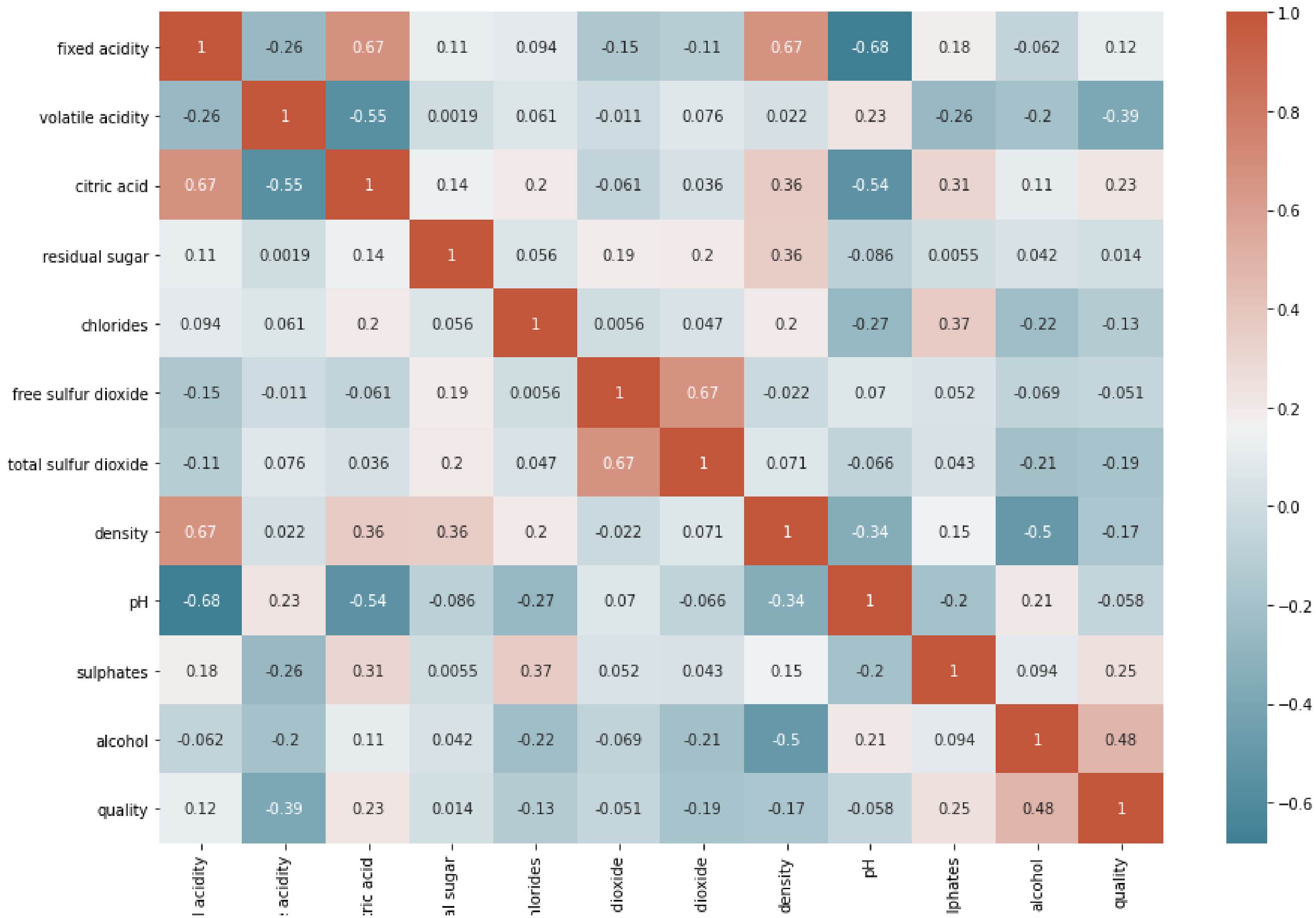
Exploring the Quality variable

corr = ds.corr()

plt.pyplot.subplots(figsize=(15,10))

sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.diverging_palette(220, 20, as_cmap=True))

<matplotlib.axes._subplots.AxesSubplot at 0x7f0efd92f890>



We observe that citric acid and fixed acidity are postively strongly corelated. So are free and total sulphur dioxide. density and fixed acidity influence each other strongly. On the other hand we can also observe that fixed acidity and pH value are negatively corelated.

```
# Create Classification version of target variable
ds['goodquality'] = [1 if x >= 7 else 0 for x in
ds['quality']] # Separate feature variables and target
variable
X = ds.drop(['quality','goodquality'], axis =
1) y = ds['goodquality']
```

Converting the quality varibale into binary. Quality rating above 7 falls into "goodquality" and below 7 falls into "bad quality"

```
X.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.

```
ds['goodquality'].value_counts()
```

```
0    1382
```

```
1        217
Name: goodquality, dtype: int64
```

```
y=ds.iloc[:,-1]
X=ds.iloc[:,0:11]
```

```
X.shape
```

```
(1599, 11)
```

```
X.nunique()
```

```
fixed acidity            96
volatile acidity        143
citric acid              80
residual sugar           91
chlorides               153
free sulfur dioxide      60
total sulfur dioxide    144
density                 436
pH                       89
sulphates                96
alcohol                  65
dtype: int64
```

```
pd.isnull(ds).values.any()
```

```
False
```

checking for null/missing values.

```
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_state=1)
```

Logistic Regression

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
x_train  =  sc_x.fit_transform(x_train)
x_test = sc_x.fit_transform(x_test)
```

Bring down all the variables to the same Scale.

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
```

```
y_pred = logreg.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
      .format(logreg.score(x_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
      .format(logreg.score(x_test, y_test)))
```

```
Accuracy of Logistic regression classifier on training set: 0.88
Accuracy of Logistic regression classifier on test set: 0.88
```

```
cm
```

```
array([[337,  18],
       [ 32,  13]])
```

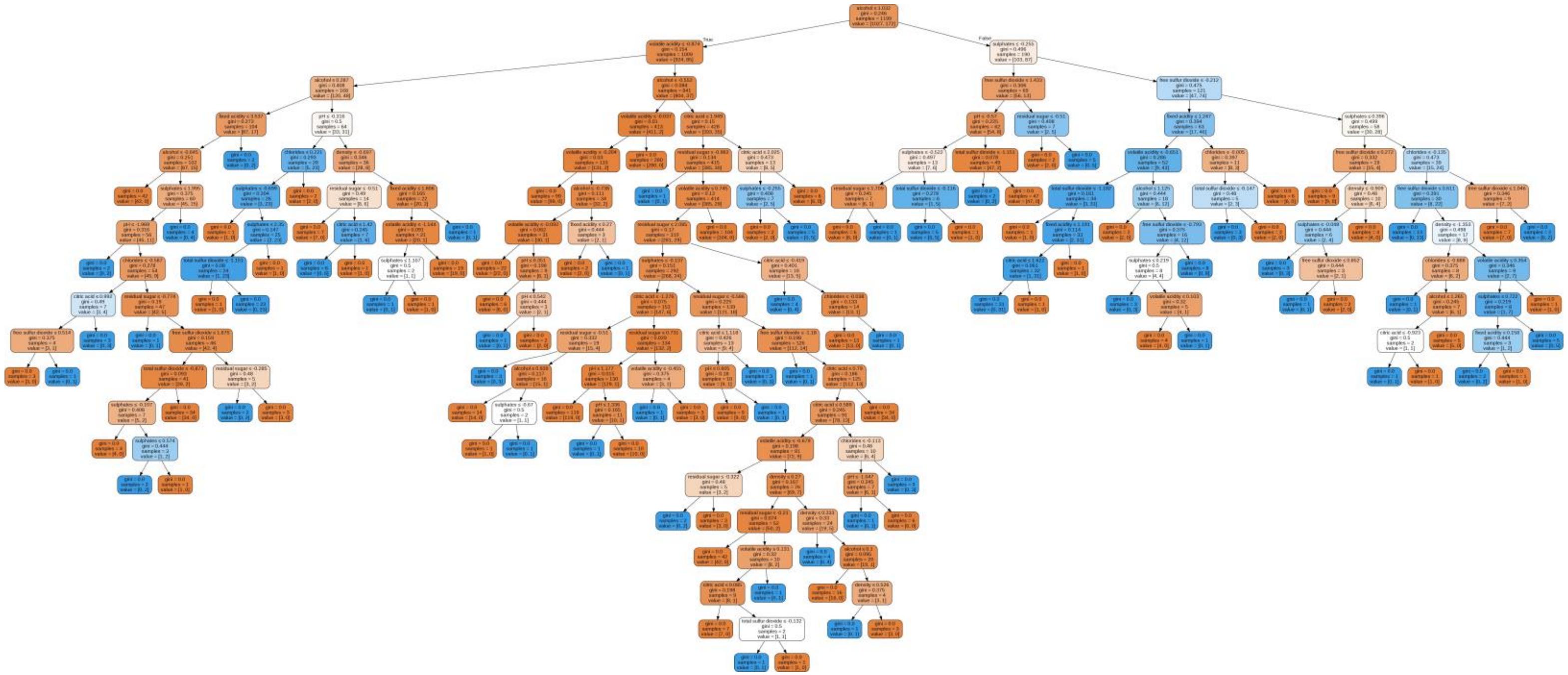
Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
regressor_dtr = DecisionTreeClassifier().fit(x_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(regressor_dtr.score(x_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(regressor_dtr.score(x_test, y_test)))
prediction=regressor_dtr.predict(x_test)
```

```
Accuracy of Decision Tree classifier on training set:
1.00 Accuracy of Decision Tree classifier on test set:
0.86
```

```
from  sklearn.externals.six  import
StringIO from IPython.display import
Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(regressor_dtr, out_file=dot_data, feature_names=X.columns,
                filled=True, rounded=True,
special_characters=True, )
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/externals/six.py:31: FutureWarning:
```



```
The module is deprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for Python 2.7. Please rely on
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
      .format(gnb.score(x_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
      .format(gnb.score(x_test, y_test)))
prediction=gnb.predict(x_test)
```

```
Accuracy of GNB classifier on training set:  0.84
Accuracy of GNB classifier on test set:  0.81
```

```
from sklearn.metrics import
classification_report from sklearn.metrics
import confusion_matrix
pred  =  regressor_dtr.predict(x_test)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test,

    pred)) [[318 37]
     [ 19  26]]
```

	precision	recall	f1-score	support
0	0.94	0.90	0.92	355
1	0.41	0.58	0.48	45
accuracy			0.86	400
macro avg	0.68	0.74	0.70	400
weighted avg	0.88	0.86	0.87	400

RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(random_state=1)
model2.fit(x_train, y_train)
y_pred2 = model2.predict(x_test)
print(classification_report(y_test, y_pred2))
```

	precision	recall	f1-score	support
0	0.94	0.97	0.95	355
1	0.67	0.49	0.56	45
accuracy			0.92	400
macro avg	0.80	0.73	0.76	400
weighted avg	0.91	0.92	0.91	400

```
df_temp = ds[ds['goodquality']==1]
df_temp.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
count	217.000000	217.000000	217.000000	217.000000	217.000000	217.000000	217.000000	217.000000	217.000000	217.000000	217.000000	217.
mean	8.847005	0.405530	0.376498	2.708756	0.075912	13.981567	34.889401	0.996030	3.288802	0.743456	11.518049	7.
std	1.999977	0.144963	0.194438	1.363026	0.028480	10.234615	32.572238	0.002201	0.154478	0.134038	0.998153	0.
min	4.900000	0.120000	0.000000	1.200000	0.012000	3.000000	7.000000	0.990640	2.880000	0.390000	9.200000	7.
25%	7.400000	0.300000	0.300000	2.000000	0.062000	6.000000	17.000000	0.994700	3.200000	0.650000	10.800000	7.
50%	8.700000	0.370000	0.400000	2.300000	0.073000	11.000000	27.000000	0.995720	3.270000	0.740000	11.600000	7.
75%	10.100000	0.490000	0.490000	2.700000	0.085000	18.000000	43.000000	0.997350	3.380000	0.820000	12.200000	7.

```
df_temp = ds[ds['goodquality']==0]
df_temp.describe()
```

By looking into the details, we can see that good quality wines have higher levels of alcohol on average, have a lower volatile acidity on average, higher levels of sulphates on average, and higher levels of residual sugar on average.

count	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.
75%	9.100000	0.650000	0.400000	2.600000	0.091000	22.000000	65.000000	0.997900	3.410000	0.700000	10.

Support Vector Machine

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(x_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(x_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
     .format(svm.score(x_test, y_test)))
prediction=svm.predict(x_test)
```

```
Accuracy of SVM classifier on training set: 0.90
Accuracy of SVM classifier on test set: 0.91
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier
model3 =  AdaBoostClassifier(random_state=1)
model3.fit(x_train, y_train)
y_pred3 = model3.predict(x_test)
print(classification_report(y_test, y_pred3))
```

	precision	recall	f1-score	support
0	0.93	0.94	0.93	355
1	0.45	0.40	0.42	45
accuracy			0.88	400
macro avg	0.69	0.67	0.68	400
weighted avg	0.87	0.88	0.87	400

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
model4 = GradientBoostingClassifier(random_state=1)
model4 fit(x train     y train)
y_pred4 = model4.predict(x_test)
print(classification_report(y_test, y_pred4))
```

	precision	recall	f1-score	support
0	0.93	0.95	0.94	355
1	0.56	0.44	0.49	45
accuracy			0.90	400
macro avg	0.74	0.70	0.72	400
weighted avg	0.89	0.90	0.89	400

XG Boost

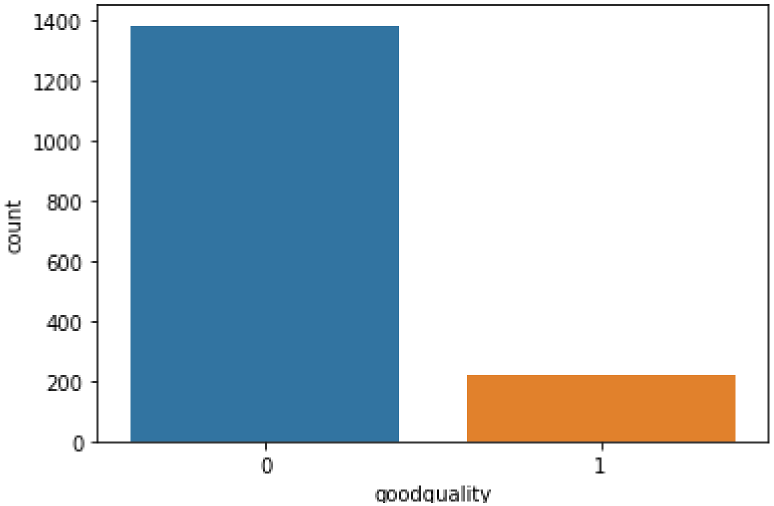
```
import xgboost as xgb
model5 = xgb.XGBClassifier(random_state=1)
model5.fit(x_train, y_train)
y_pred5 = model5.predict(x_test)
print(classification_report(y_test, y_pred5))
```

	precision	recall	f1-score	support
0	0.93	0.95	0.94	355
1	0.54	0.47	0.50	45
accuracy			0.90	400
macro avg	0.74	0.71	0.72	400
weighted avg	0.89	0.90	0.89	400

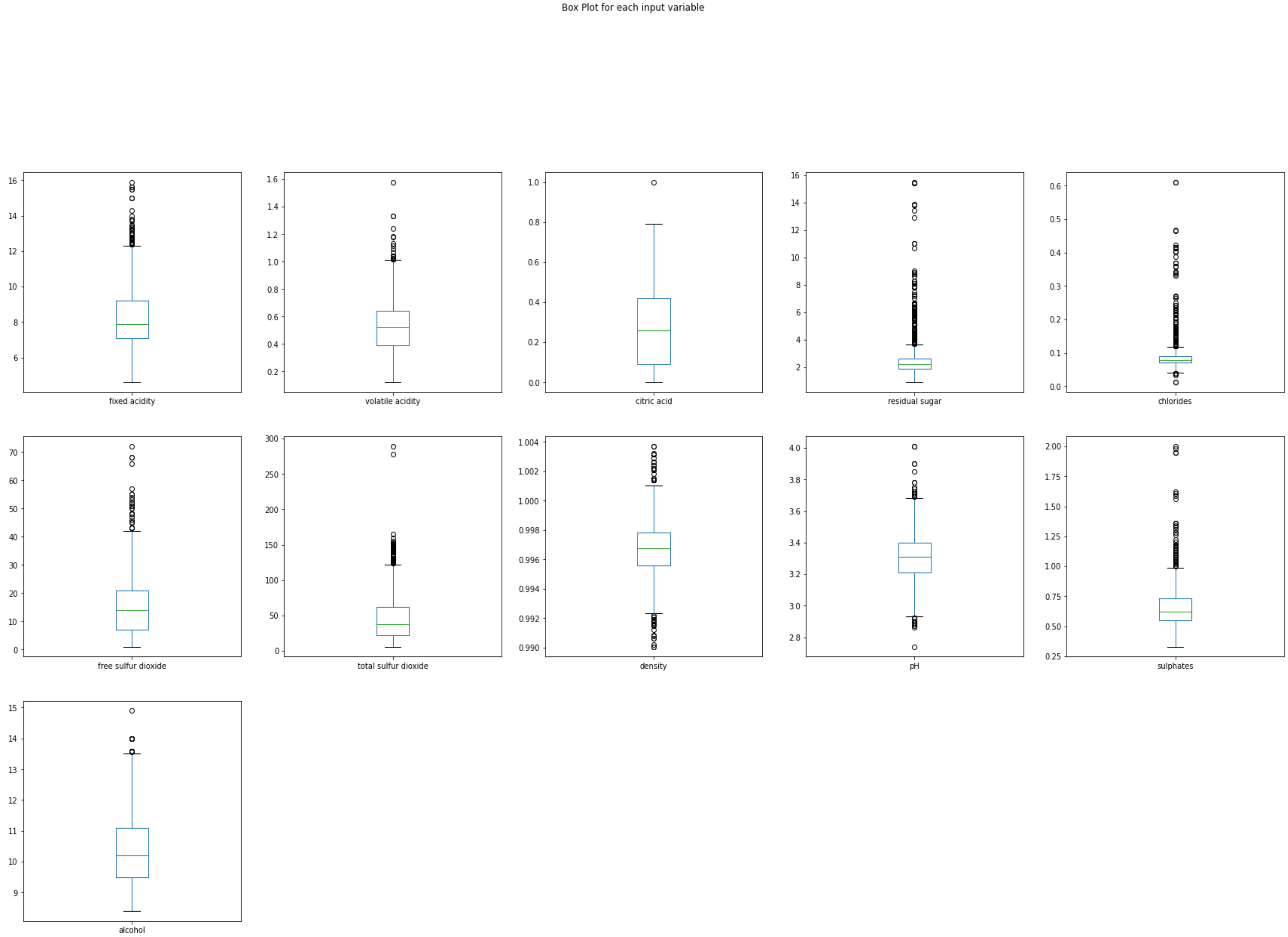
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(y,label="Count")
plt.show()
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other



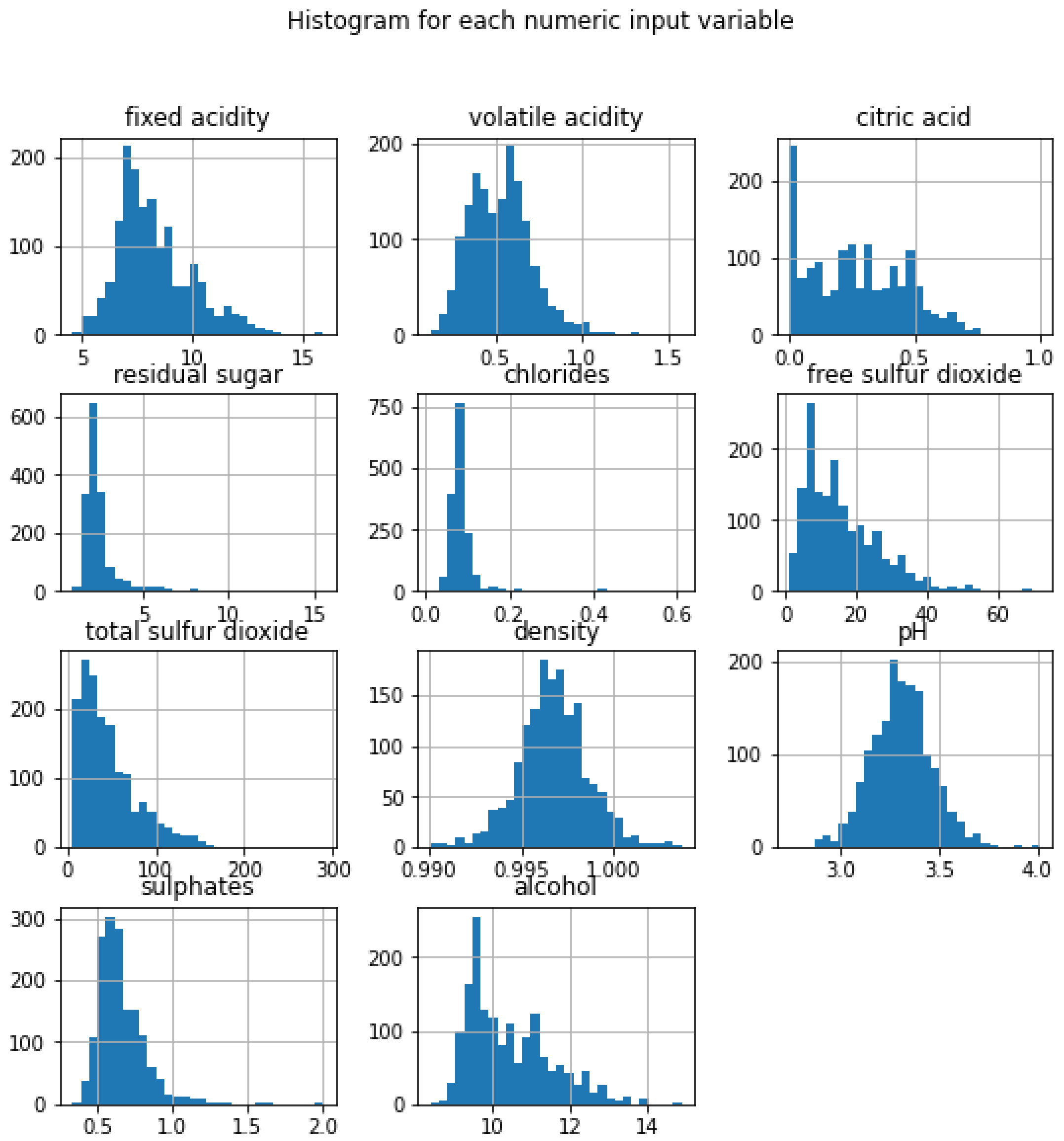

```
X.plot(kind='box', subplots=True, layout=(5,5), sharex=False, sharey=False, figsize=(30,30),
                                        title='Box Plot for each input variable')
plt.savefig('good quality wine')
plt.show()
```



Box Plot for all input variables

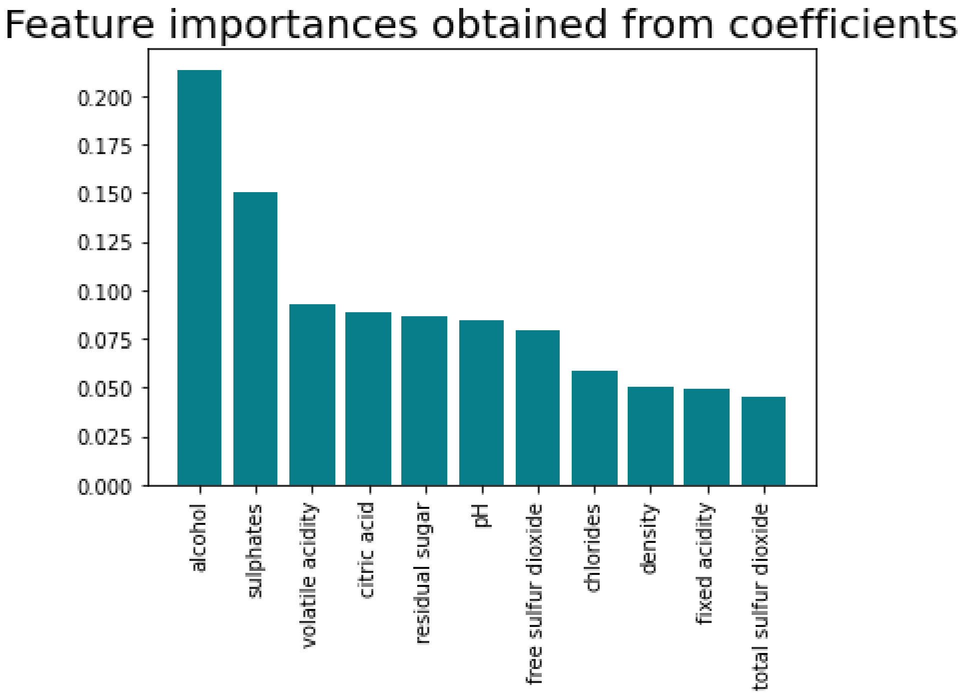
```
import pylab as pl
X.hist(bins=30, figsize=(9,9))
pl.suptitle("Histogram for each numeric input
variable") plt.savefig('hist')
```

```
plt.show()
```




```
importances = pd.DataFrame(data={
    'Attribute':  X.columns,
    'Importance': regressor_dtr.feature_importances_
})
importances = importances.sort_values(by='Importance', ascending=False)
```

```
plt.bar(x=importances['Attribute'], height=importances['Importance'], color='#087E8B')
plt.title('Feature importances obtained from coefficients', size=20)


plt.xticks(rotation='vertical')
plt.show()
```



As we can observe from the given data, alcohol content is the most important factor in determing the quality of wine. This is followed by the sulphate concentration and pH level.
Total sulfur dioxide is the least important feature.

```
X1 = np.array(X)


pip install lime

    Collecting lime
```

```
  Downloading https://files.pythonhosted.org/packages/f5/86/91a13127d83d793ecb50eb75e716f76e6eda809b6803c5a4ff462339789e/lime-0.2.0.1.tar
     |████████████████████████████████| 276kB 6.2MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from lime) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from lime) (1.19.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from lime) (1.4.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from lime) (4.41.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/dist-packages (from lime) (0.22.2.post1)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.7/dist-packages (from lime) (0.16.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (2.8.1)


Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->lime)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->lime) (1.3.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.18->lime) (1.0.1)
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime)
(7.1.2) Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime) (2.5.1)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime) (2.4.1)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image>=0.12->lime)
(1.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->lime)
(1.15.0) Requirement already satisfied: decorator<5,>=4.3 in /usr/local/lib/python3.7/dist-packages (from networkx>=2.0->scikit-image>=0.12->lime) Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-cp37-none-any.whl size=283846
  sha256=b3bbb56617a9583d6edc8dfbbb95f5de7cf3d93efd37c0e0ac5f  Stored in directory:
  /root/.cache/pip/wheels/4c/4f/a5/0bc765457bd41378bf3ce8d17d7495369d6e7ca3b712c60c89
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

```
pip install shap
```

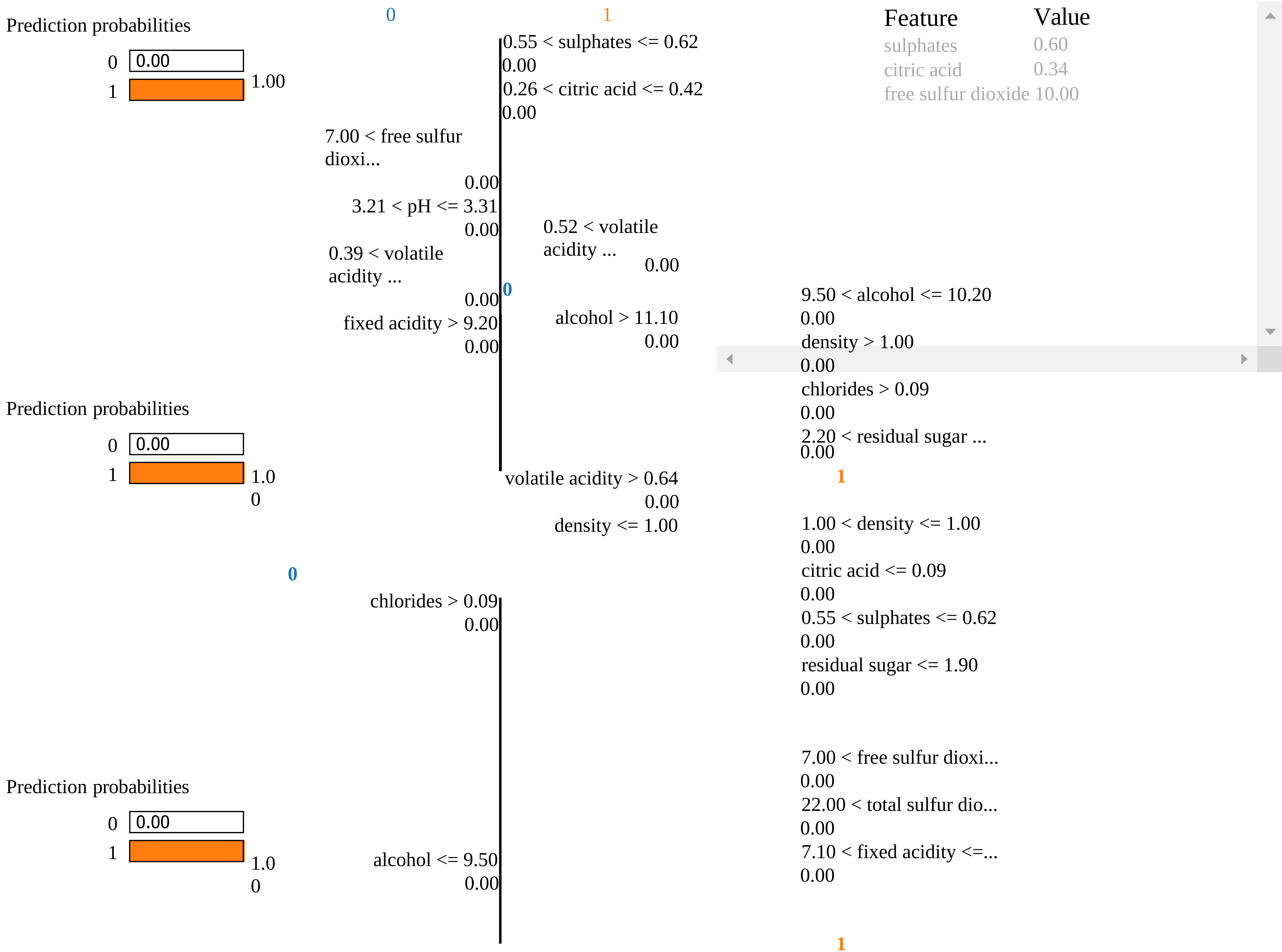
```
Collecting shap
  Downloading https://files.pythonhosted.org/packages/b9/f4/c5b95cddae15be80f8e58b25edceca105aa83c0b8c86a1edad24a6af80d3/shap-0.39.0.tar.
```

```
|████████████████████████████████| 358kB 5.5MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from shap) (1.19.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from shap)
(1.4.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from shap)
(0.22.2.post1) Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from
shap) (1.1.5)
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.7/dist-packages (from shap) (4.41.1)
Collecting slicer==0.0.7
  Downloading https://files.pythonhosted.org/packages/78/c2/b3f55dfdb8af9812fdb9baf70cacf3b9e82e505b2bd4324d588888b81202/slicer-0.0.7-
py3 Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from shap) (0.51.2)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-packages (from shap) (1.3.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->shap) (1.0.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->shap) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->shap) (2018.9)
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/dist-packages (from numba->shap)
(0.34.0) Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba->shap) (54.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->shap)
(1.15.0) Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.39.0-cp37-cp37m-linux_x86_64.whl size=491623
  sha256=8659c27545167c97ce4c980f80b6c4d22bf719d2c6c Stored in directory:
  /root/.cache/pip/wheels/15/27/f5/a8ab9da52fd159aae6477b5ede6eaaec69fd130fa0fa59f283
Successfully built shap

Installing collected packages: slicer, shap
Successfully installed shap-0.39.0 slicer-0.0.7
```

```
import
lime
import
shap
import lime.lime_tabular
number_of_rows = X.shape[0]
random_indices = np.random.choice(number_of_rows, size=4, replace=False)
```

SHAP and LIME are both popular Python libraries for model explainability.
4 datarows are randomly selected to
determine the factors that most affect the quality of the wine independant of the other rows.

```
explainer = lime.lime_tabular.LimeTabularExplainer(X1, feature_names=X.columns, class_names=[0,1], discretize_continuous=True)

for i in random_indices:
   exp1 = explainer.explain_instance(X1[i], regressor_dtr.predict_proba)
   exp1.as_pyplot_figure()
   exp1.show_in_notebook(show_table=True, show_all=False)
```

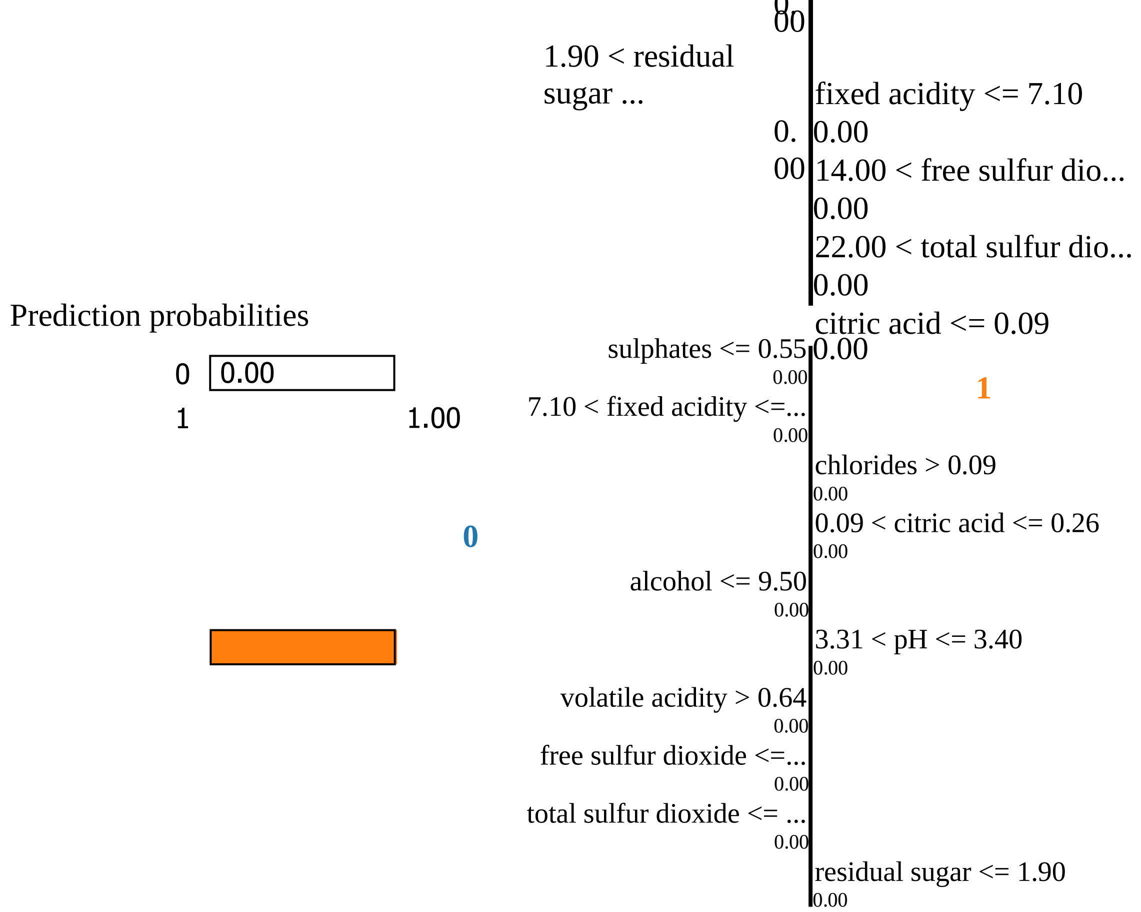


Prediction probabilities
0 0.00
1 1.00
0
1
0.55 < sulphates <= 0.62
0.00
0.26 < citric acid <= 0.42
0.00
7.00 < free sulfur dioxi...
0.00
3.21 < pH <= 3.31
0.00
0.52 < volatile acidity ...
0.00
0.39 < volatile acidity ...
0.00
fixed acidity > 9.20
0.00
alcohol > 11.10
0.00
volatile acidity > 0.64
0.00
density <= 1.00
Feature Value
sulphates 0.60
citric acid 0.34
free sulfur dioxide 10.00
Prediction probabilities
0 0.00
1 1.00
0
chloride > 0.09
0.00
alcohol <= 9.50
0.00
1
9.50 < alcohol <= 10.20
0.00
density > 1.00
0.00
chlorides > 0.09
0.00
2.20 < residual sugar ...
0.00
1.00 < density <= 1.00
0.00
citric acid <= 0.09
0.00
0.55 < sulphates <= 0.62
0.00
residual sugar <= 1.90
0.00
7.00 < free sulfur dioxi...
0.00
22.00 < total sulfur dio...
0.00
7.10 < fixed acidity <=...
0.00
Prediction probabilities
0 0.00
1 1.00
1

pH > 3.40
0.00
0.55 < sulphates <= 0.62
0.00

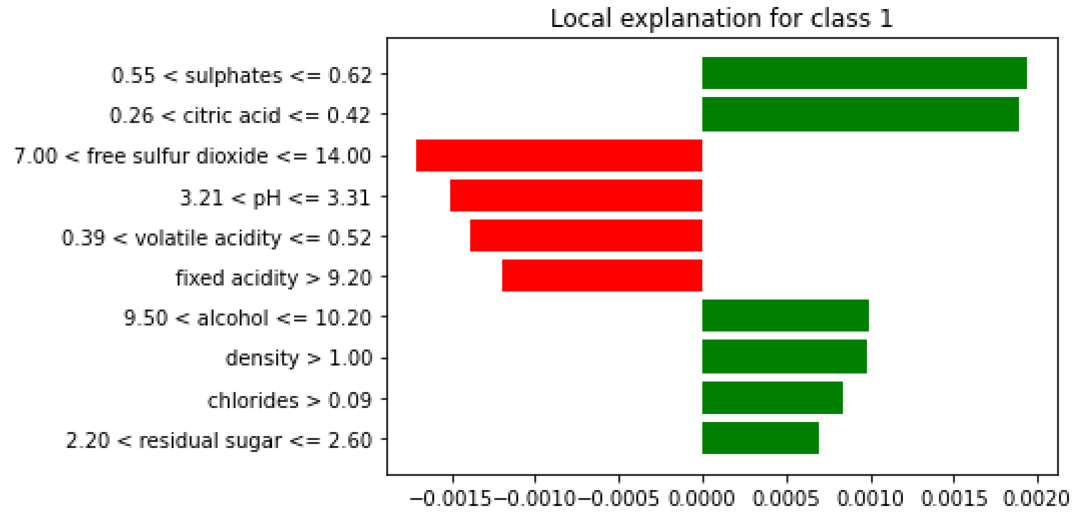
pH	3.24
volatile acidity	0.50
fixed acidity	9.80
alcohol	9.70
density	1.00
chlorides	0.09
residual sugar	2.30

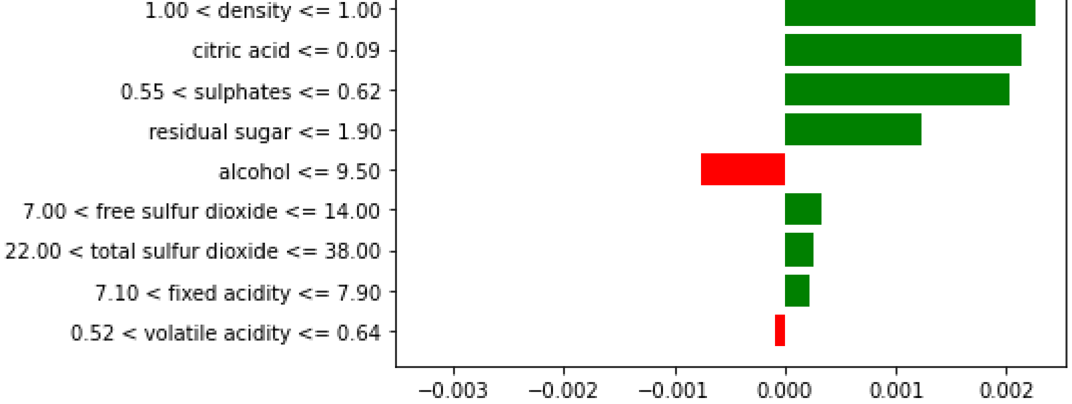
Feature	Value
alcohol	***12.50***
pH	***3.78***
sulphates	***0.61***
volatile acidity	***0.65***
density	***0.99***
residual sugar	***2.15***
fixed acidity	***5 20***

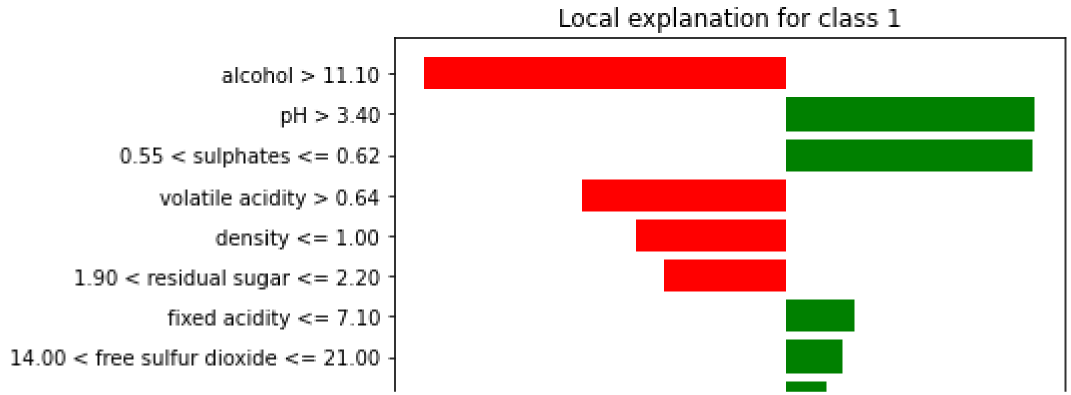


free sulfur dioxide	15.00
total sulfur dioxide	28.00
citric acid	0.00

Feature	Value
sulphates	0.54
fixed acidity	7.40
chlorides	0.19
citric acid	0.12
alcohol	9.50
pH	3.39
volatile acidity	0.67
free sulfur dioxide	5.00
total sulfur dioxide	21.00
residual sugar	1.60







From the above models we can infer that alcohol and sulphates content are most important. Any wine production company should consider this to improve the quality of wine. However it is important to note that the quantities should be accordance with the human body and abiding by the rules set up by the wine governing bodies.