Class Project 2: Data Mining

Statsmodel and sklearn's Linear Regression model along with Random Forest Classifier on Red Wine Quality Data Set

Course: IS655 1J1 Data Analytics for Information System

Professor: Dr. Lin Lin

Project Submitted by Team 2:

Brian Rooney

|  |  |
| --- | --- |
| |  | | --- | | Parvathy Neelakanta Sarma  Shivakalyan Soundarathiagarajan | |

Srishti Karakoti

Table of Contents

[Classification using Statsmodel 16](#_Toc58944877)

[Classification using Sklearn's LogisticRegression 18](#_Toc58944878)

[Classification using Sklearn's RandomForestClassifier 22](#_Toc58944879)

[Conclusion 26](#_Toc58944880)

**Project Description: Supervised Data Mining (Classification)**

We chose Supervised Data Mining for Classification. The details of my project are described as follows:

**Two Classification Algorithms:**

1. Statsmodel
2. sklearn's Linear Regression model along with Random Forest Classifier.

**Dataset: Red Wine Quality Data Set**

We used machine learning to determine which physiochemical properties make a wine 'good'!

**Name:** Red Wine Quality Data Set

**Source:** UCI Machine Learning Repository

**Input variables:**

* fixed acidity
* volatile acidity
* citric acid
* residual sugar
* chlorides
* free sulfur dioxide
* total sulfur dioxide
* density
* pH
* sulphates
* alcohol

**Output variable:** quality (score between 0 and 10)

**Data Set Characteristics:** Multivariate

**Number of Observations:** 1599

**Number of Attributes/Variables:** 12

**Missing Values:** N/A

**Kaggle Link for the Notebook and Data:** <https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009>

**Import Libraries and Modules**

**Setting up the development environment by importing required libraries and modules:**

* Numpy: It will provide the support for efficient numerical computation.
* Pandas: It is convenient library that supports dataframes. Working with pandas will bring ease in many crucial data operations.
* Matplotlib: It provides a MATLAB-like plotting framework.
* Seaborn: It is a visualization library based on matplotlib which provides a high-level interface for drawing attractive statistical graphics.
* Bokeh: It is a interactive visualization library that targets modern web browsers for presentation.
* Statsmodel: It provides functions and classes for statistical tests and models.
* Sklearn: It is python library for data mining, data analysis and machine learning.

**Source Code of Classification Algorithms**

We accessed the python Machine Learning library sci-kit-learn in a Jupyter Notebook for this analysis. The source code of my classification algorithms can be found at the follow web address:

<https://scikit-learn.org/stable/index.html>

We accessed sci-kit learn and other data analysis packages through an Anaconda distribution. Below we have included a screen shot of our notebook setup.



**Step up Notebook and read in Data**

Code

# Notebook Setup

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from bokeh.plotting import figure, output\_file, show

from bokeh.layouts import row

from bokeh.io import output\_notebook

import statsmodels.api as sm

import statsmodels.formula.api as smf

from patsy import dmatrices

import sklearn

import sklearn.metrics

from sklearn import ensemble

from sklearn import linear\_model

import warnings

warnings.filterwarnings('ignore')

output\_notebook()

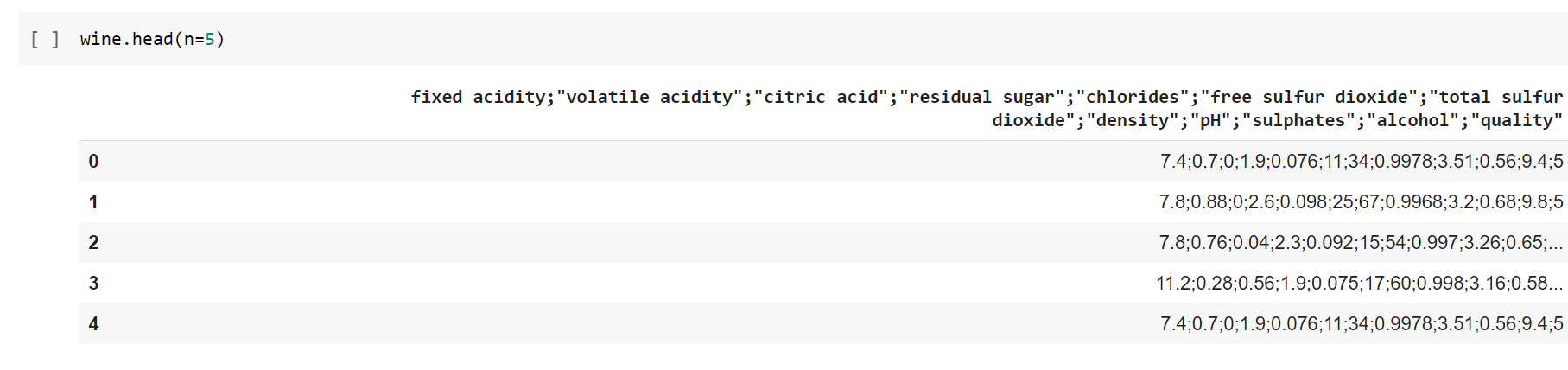
%matplotlib inline

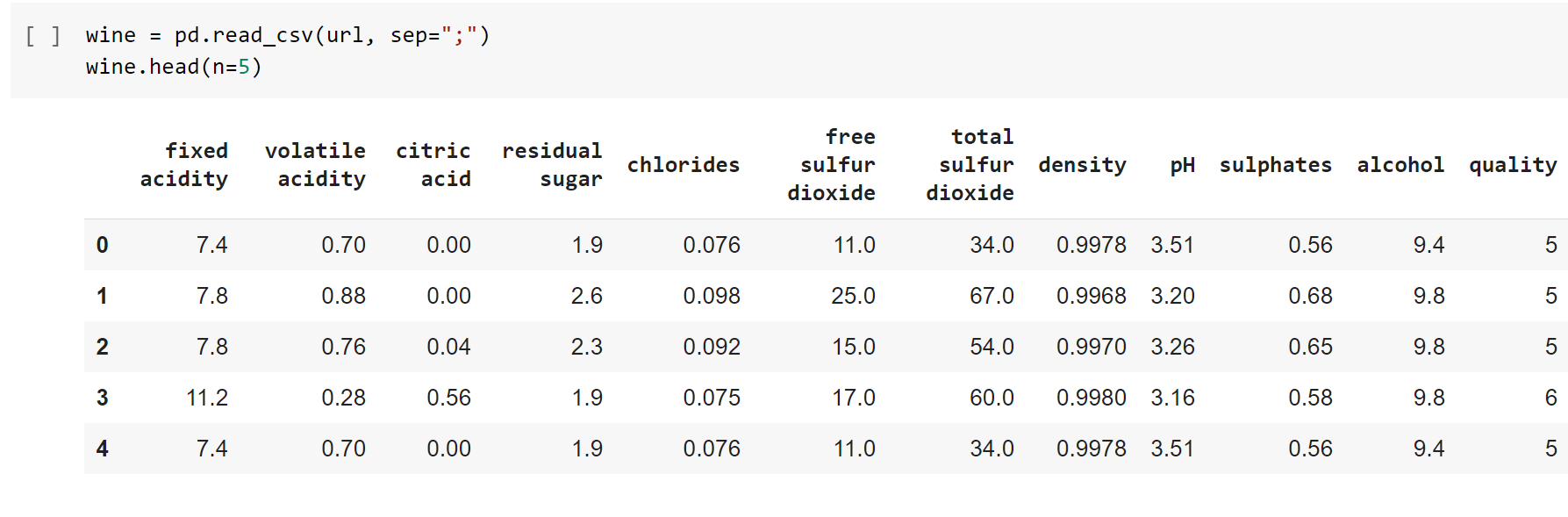
# Import data

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"

wine = pd.read\_csv(url)

Screenshots





**Exploratory Data Analysis and Data Cleaning**

This section allows us to see the size and shape of the data, the Data types of each attribute, and the rows and columns with missing or null values.

Data Cleaning allows us to clean any data by imputing or removing null values. Renaming some of the headers/column names by removing the *'blank spaces'* from it.

Code

print("Shape of Red Wine dataset: {s}").format(s = wine.shape)

print("Column headers/names: {s}").format(s = list(wine))

# Let us check the information about different variables/column from the

dataset:

wine.info()

# Let us look at the summary of the dataset:

wine.describe()

wine.isnull().sum()

wine.rename(columns={'fixed acidity': 'fixed\_acidity','citric acid':'citric\_acid','volatile acidity':'volatile\_acidity','residual sugar':'residual\_sugar','free sulfur dioxide':'free\_sulfur\_dioxide','total sulfur dioxide':'total\_sulfur\_dioxide'}, inplace=True)

wine.head(n=5)

Learning more about the target/response variable/feature:

* Let's check how many unique values does the target feature *'quality'* has?

wine['quality'].unique()

wine.quality.value\_counts().sort\_index()

sns.countplot(x='quality', data=wine)

Let's create a new discreet, categorical response variable/feature ('rating') from existing 'quality' variable.  
i.e. bad: 1-4  
      average: 5-6  
      good: 7-10

conditions = [

    (wine['quality'] >= 7),

    (wine['quality'] <= 4)

]

rating = ['good', 'bad']

wine['rating'] = np.select(conditions, rating, default='average')

wine.rating.value\_counts()

wine.groupby('rating').mean()

Corelation between features/variables:

* Let's check the corelation between the target variable and predictor variables,

correlation = wine.corr()

plt.figure(figsize=(12, 5))

sns.heatmap(correlation, annot=True, linewidths=0, vmin=-1, cmap="RdBu\_r")

correlation['quality'].sort\_values(ascending=False)

* We can observe that, the 'alcohol, sulphates, citric\_acid & fixed\_acidity' have maximum corelation with response variable 'quality'.
* This means that, they need to be further analysed for detailed pattern and corelation exploration. Hence, we will use only these 4 variables in our future analysis

Analysis of alcohol percentage with wine quality:

bx = sns.boxplot(x="quality", y='alcohol', data = wine)

bx.set(xlabel='Wine Quality', ylabel='Alcohol Percent', title='Alcohol percent in different wine quality types')

Analysis of sulphates & wine ratings:

bx = sns.boxplot(x="rating", y='sulphates', data = wine)

bx.set(xlabel='Wine Ratings', ylabel='Sulphates', title='Sulphates in different types of Wine ratings')

Analysis of Citric Acid & wine ratings:

bx = sns.violinplot(x="rating", y='citric\_acid', data = wine)

bx.set(xlabel='Wine Ratings', ylabel='Citric Acid', title='Xitric\_acid in different types of Wine ratings')

Analysis of fixed acidity & wine ratings:

bx = sns.boxplot(x="rating", y='fixed\_acidity', data = wine)

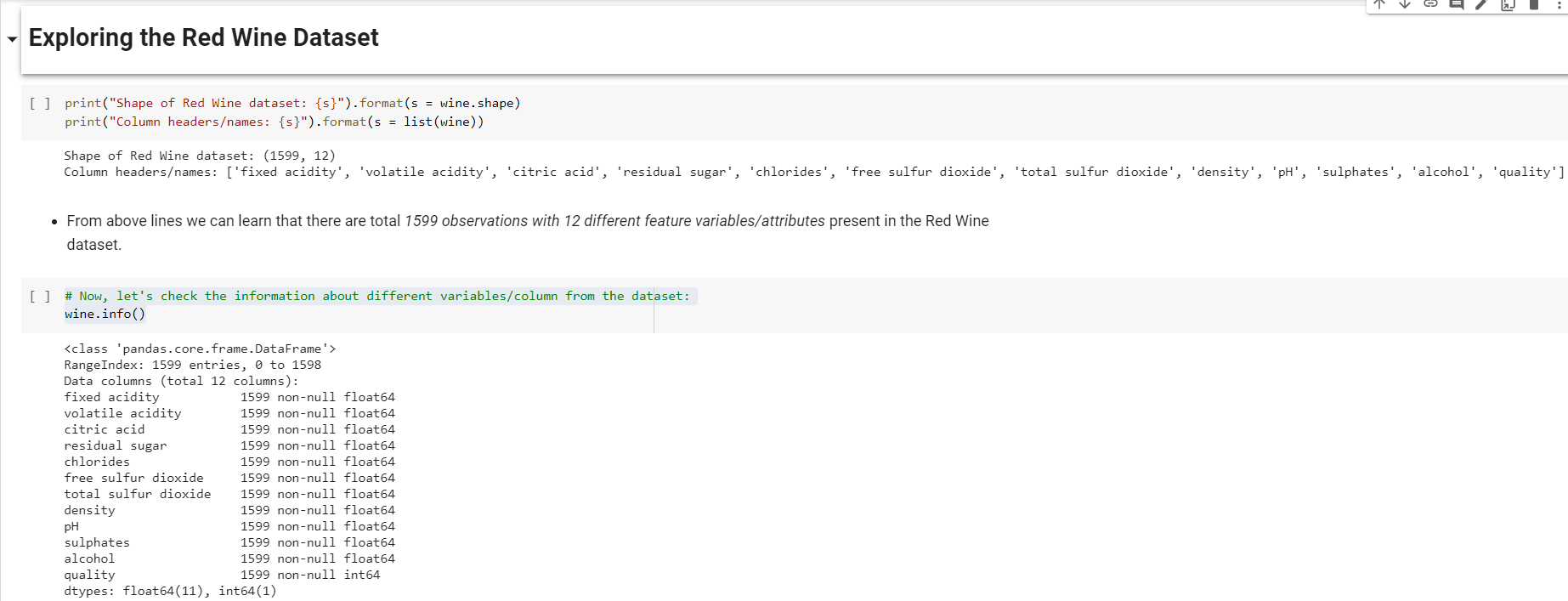
bx.set(xlabel='Wine Ratings', ylabel='Fixed Acidity', title='Fixed Acidity in different types of Wine ratings')

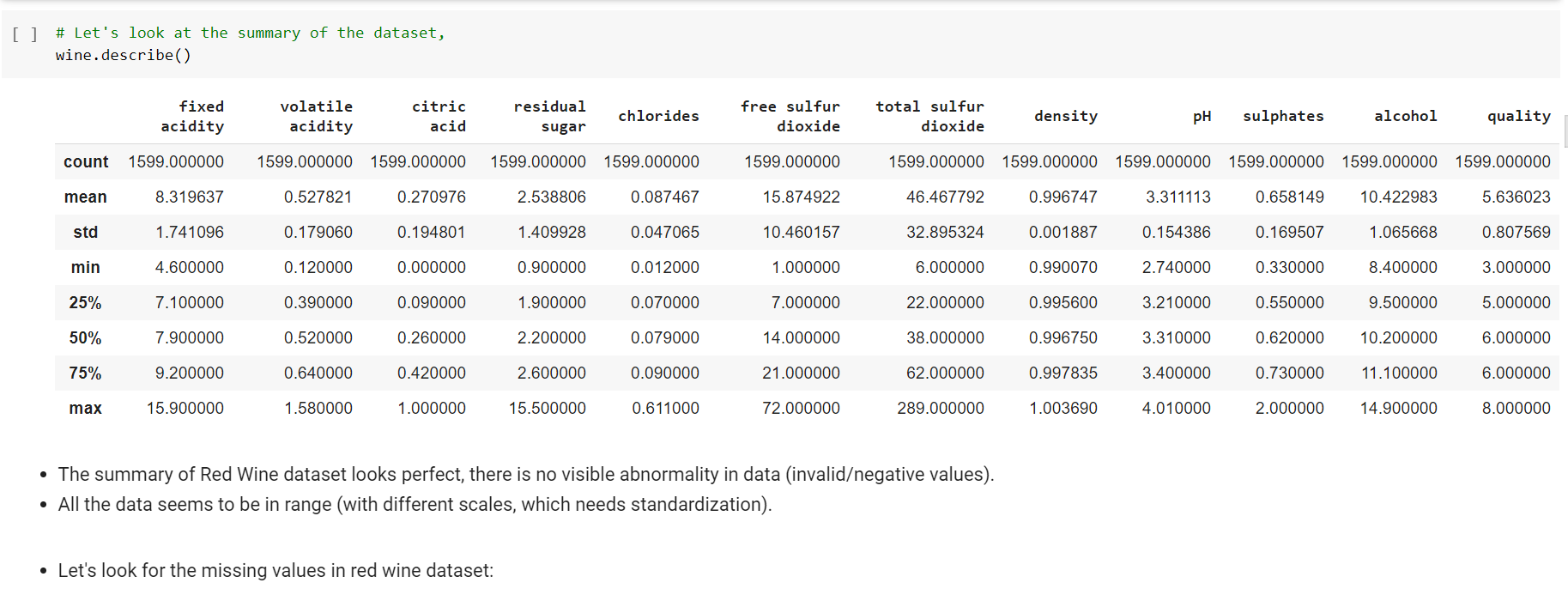
Analysis of pH & wine ratings:

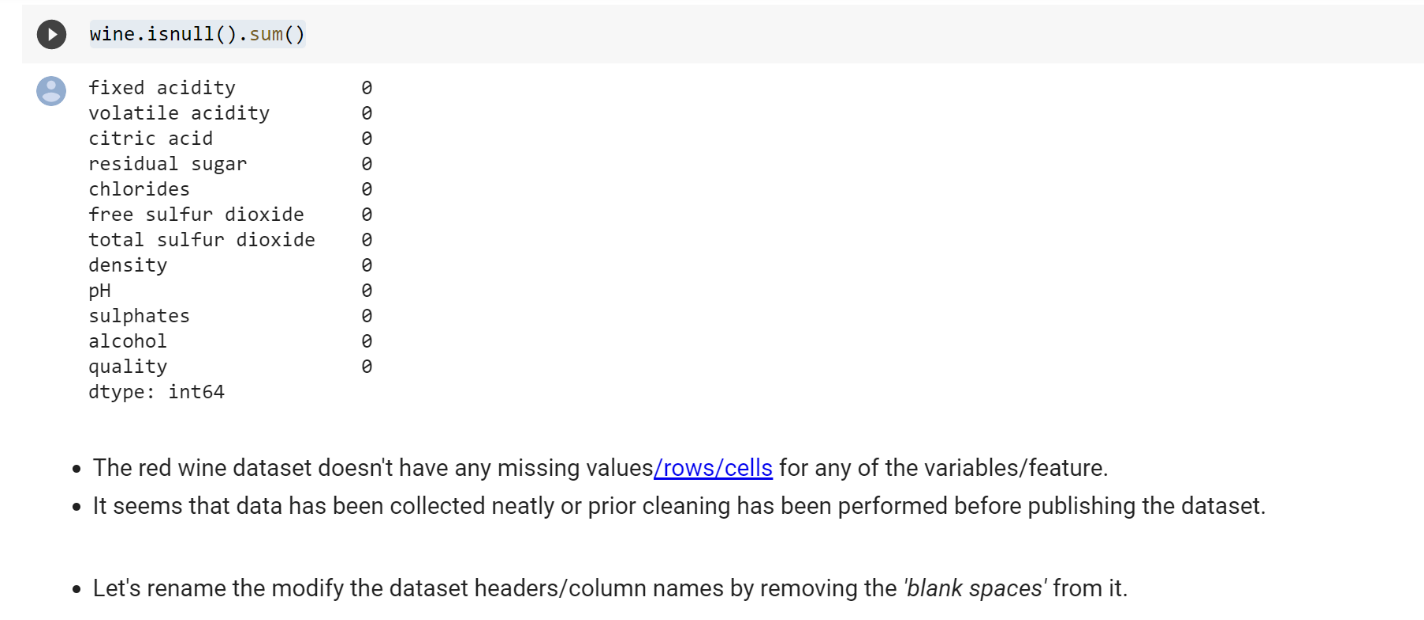
bx = sns.swarmplot(x="rating", y="pH", data = wine);

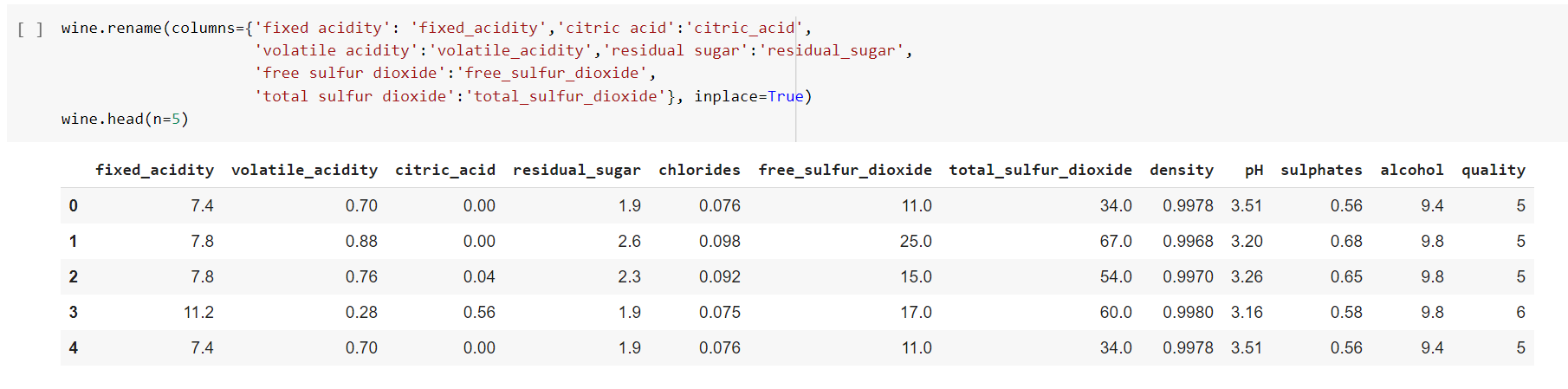
bx.set(xlabel='Wine Ratings', ylabel='pH', title='pH in different types of Wine ratings')

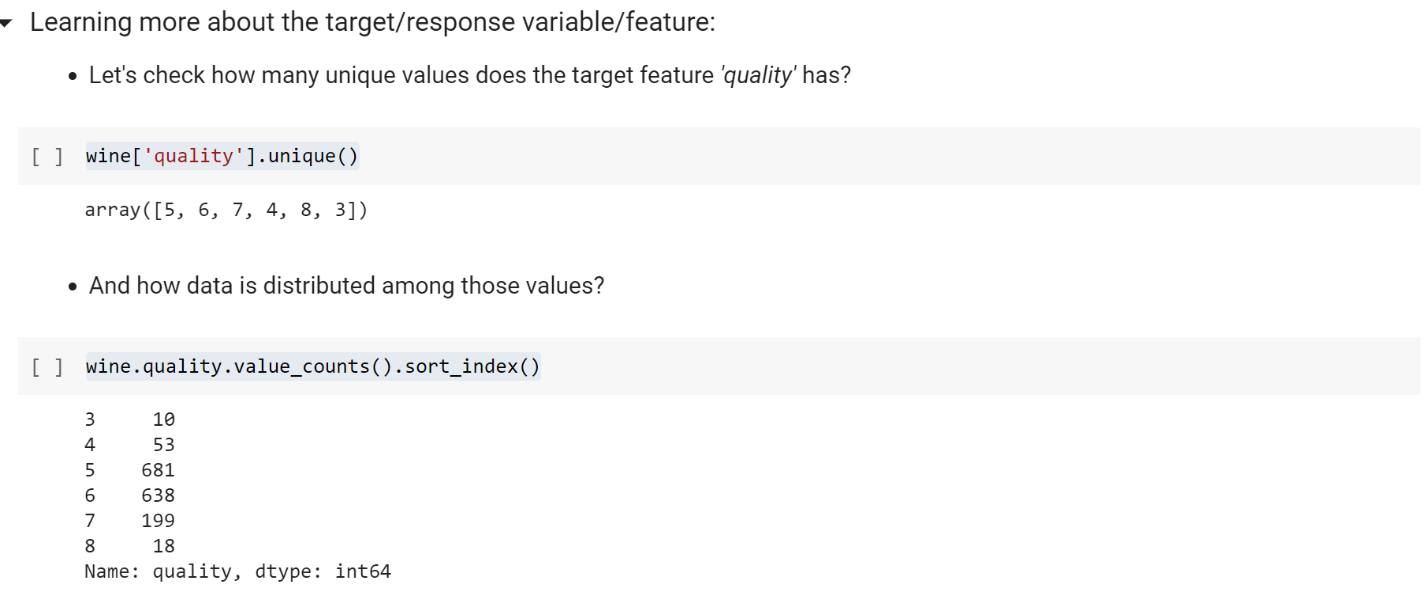
Screenshots





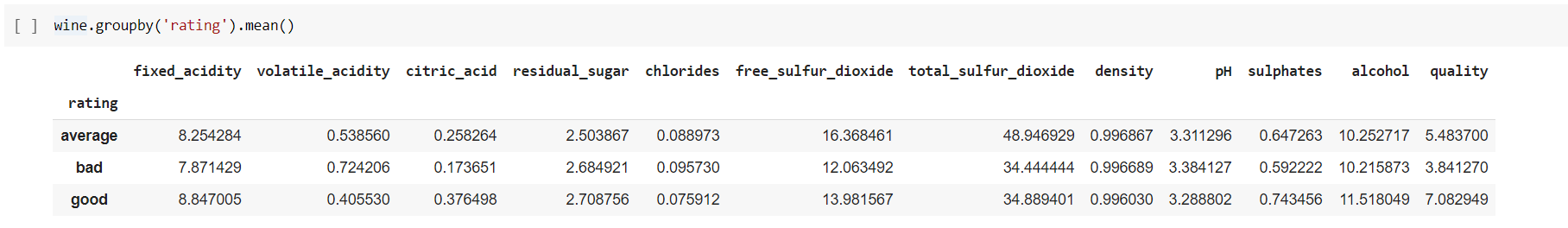


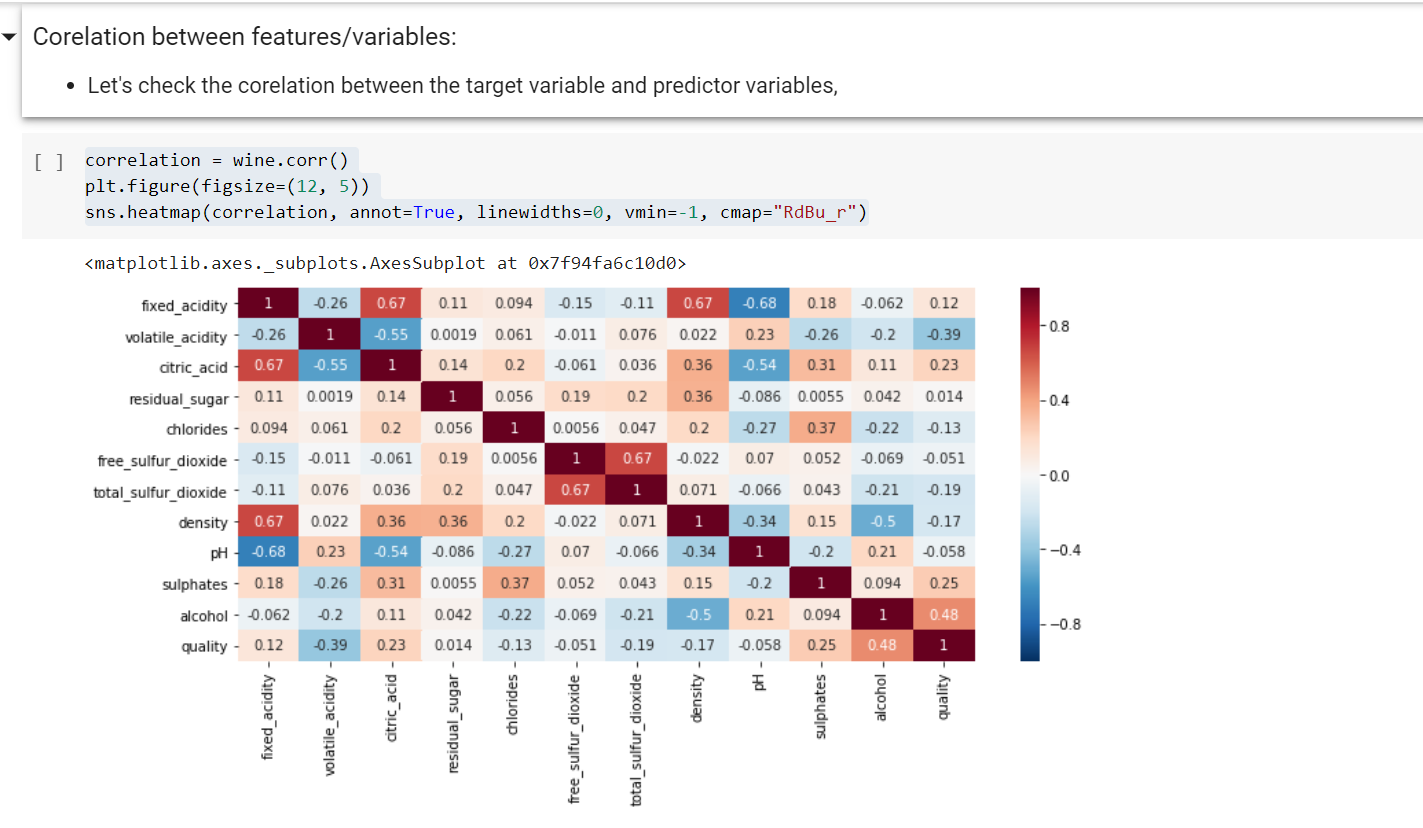


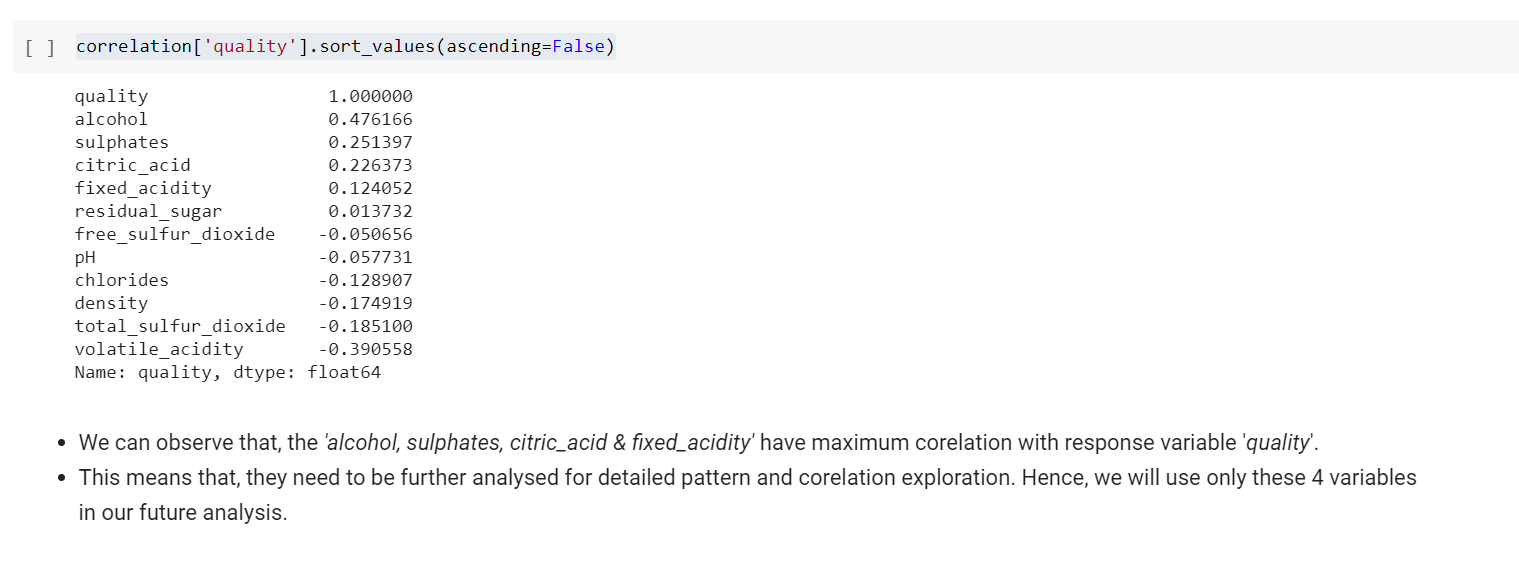


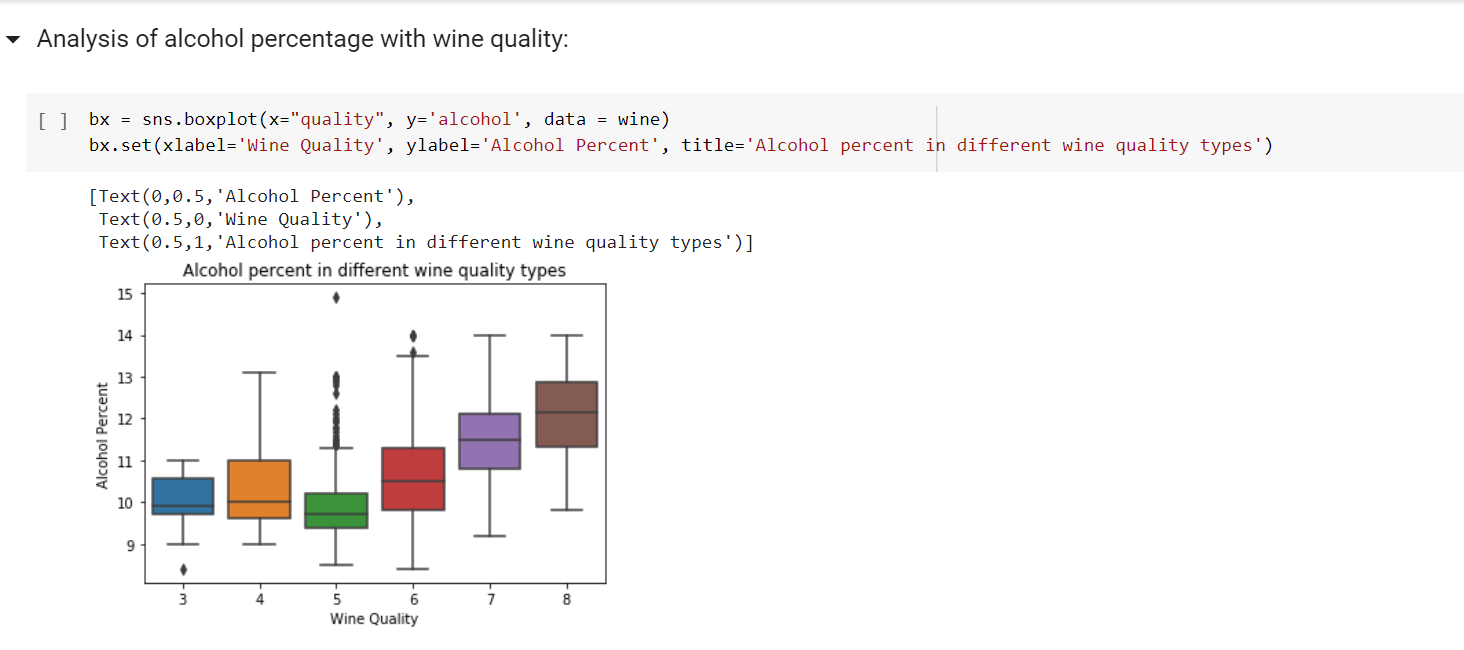


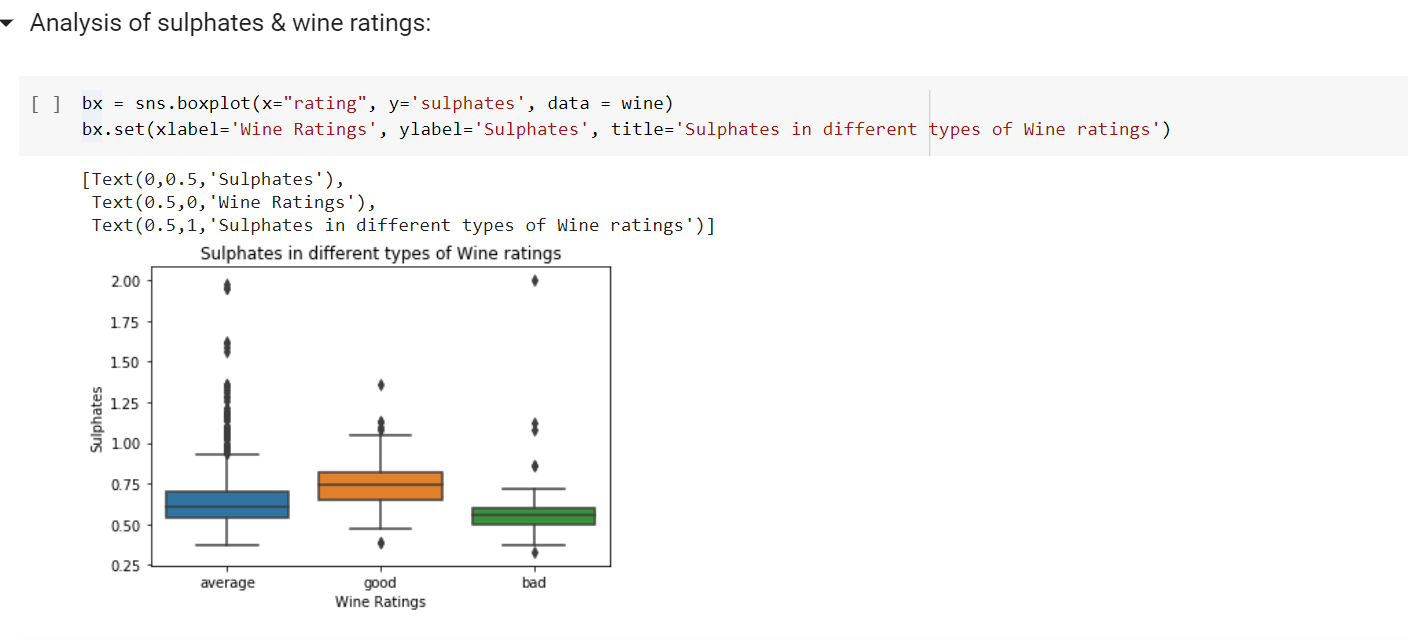


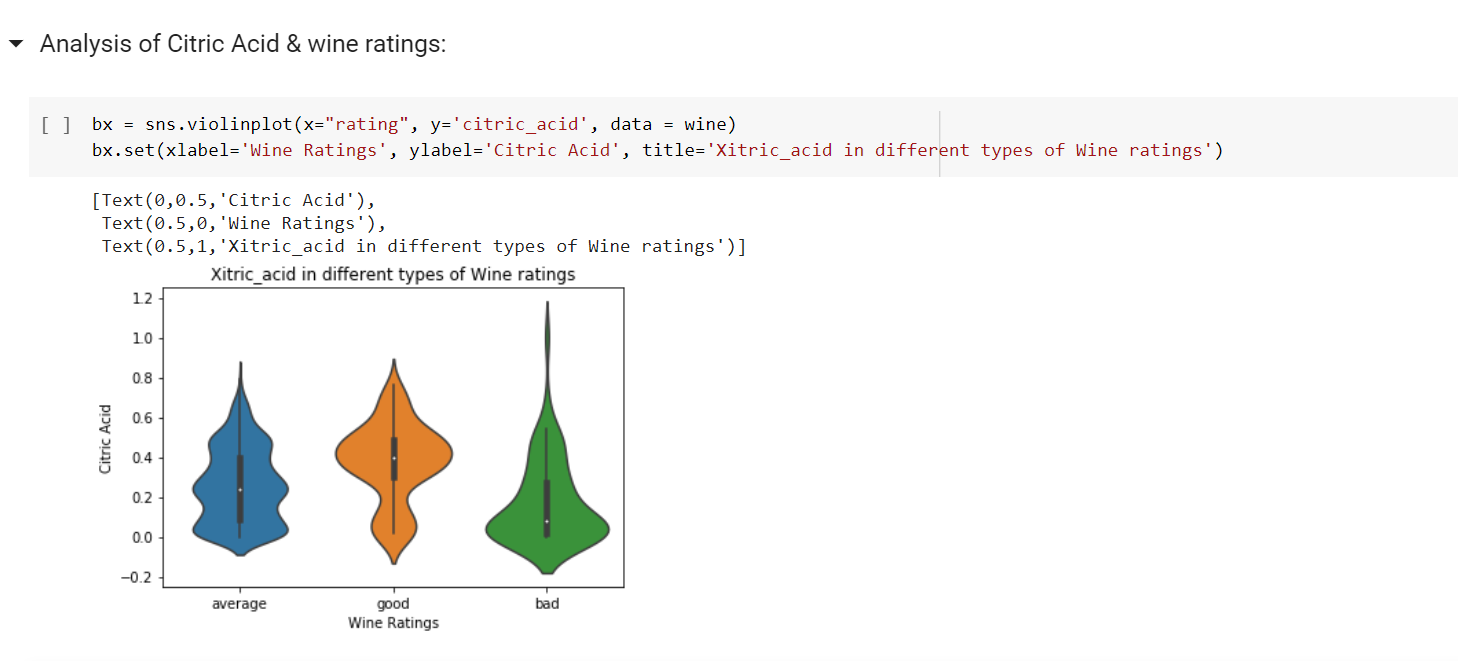


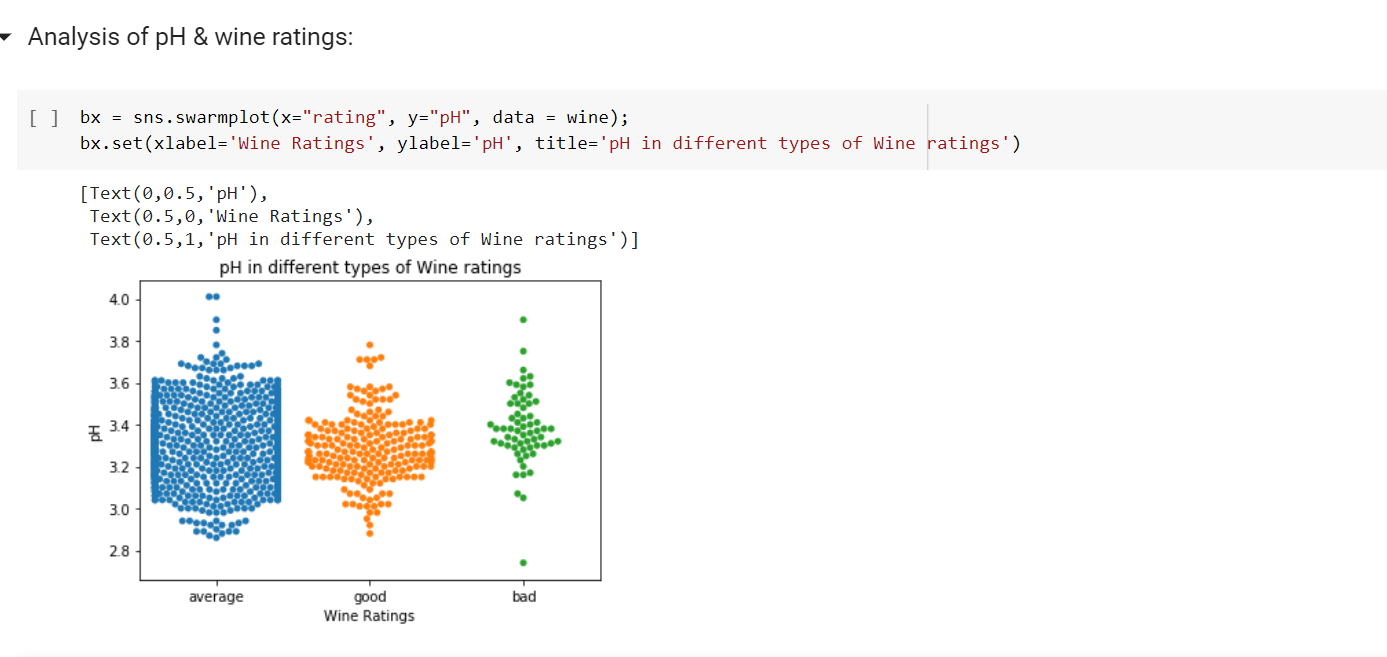
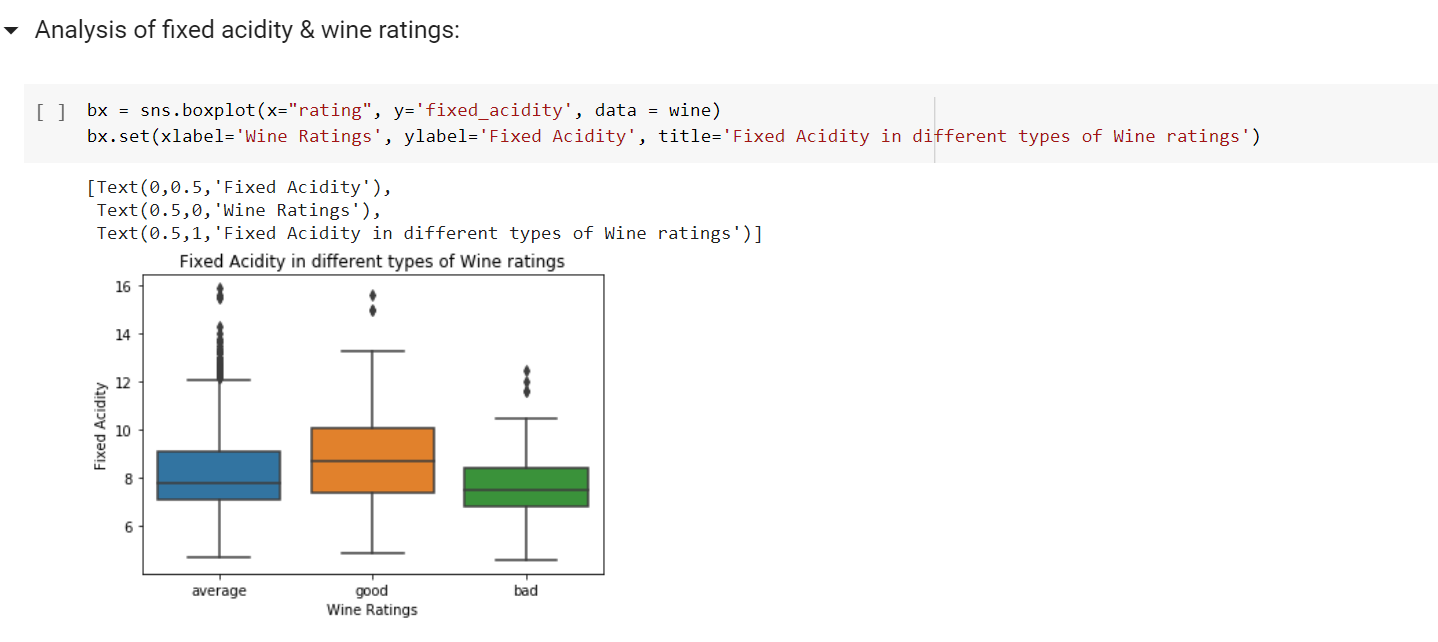










**Linear Regression**

Plot graphs for different quality ratings shows a linear regression between residual\_sugar & alcohol in red wine.

Code

sns.lmplot(x = "alcohol", y = "residual\_sugar", col = "rating", data = wine)

* The linear regression plots above for different wine quality ratings (bad, average & good) shows the regression between alcohol and residual sugar content of the red wine.
* We can observe from the trendline that, for good and average wine types the residual sugar content remains almost constant irrespective of alcohol content value. Whereas for bad quality wine, the residual sugar content increases gradually with the increase in alcohol content.
* This analysis can help in manufacturing the good quality wine with continuous monitoring and controlling the alcohol and residual sugar content of the red wine.

y,X = dmatrices('quality ~ alcohol', data=wine, return\_type='dataframe')

print("X:", type(X))

print(X.columns)

# model=smf.OLS(y, X)

model=sm.OLS(y, X)

result=model.fit()

result.summary()

# model = smf.OLS.from\_formula('quality ~ alcohol', data = wine)

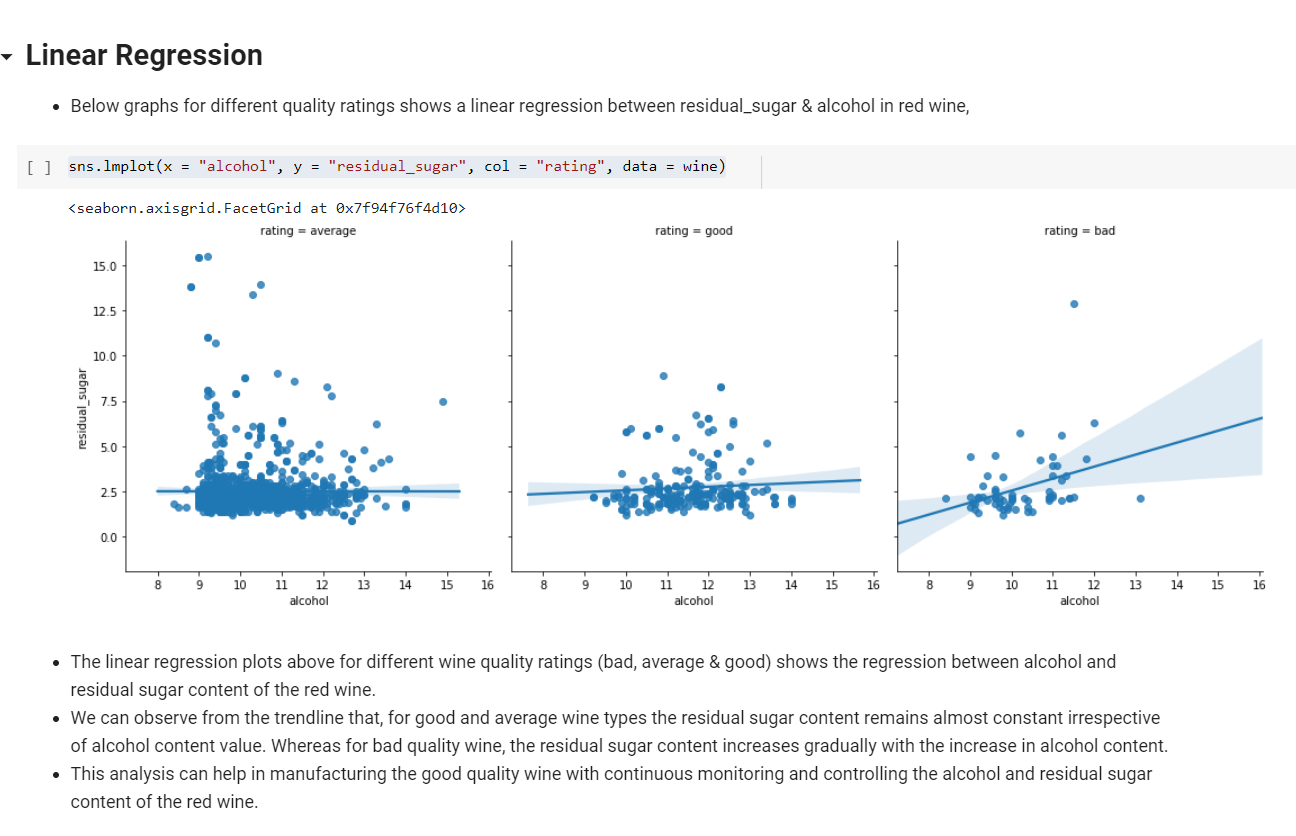
model = sm.OLS.from\_formula('quality ~ alcohol', data = wine)

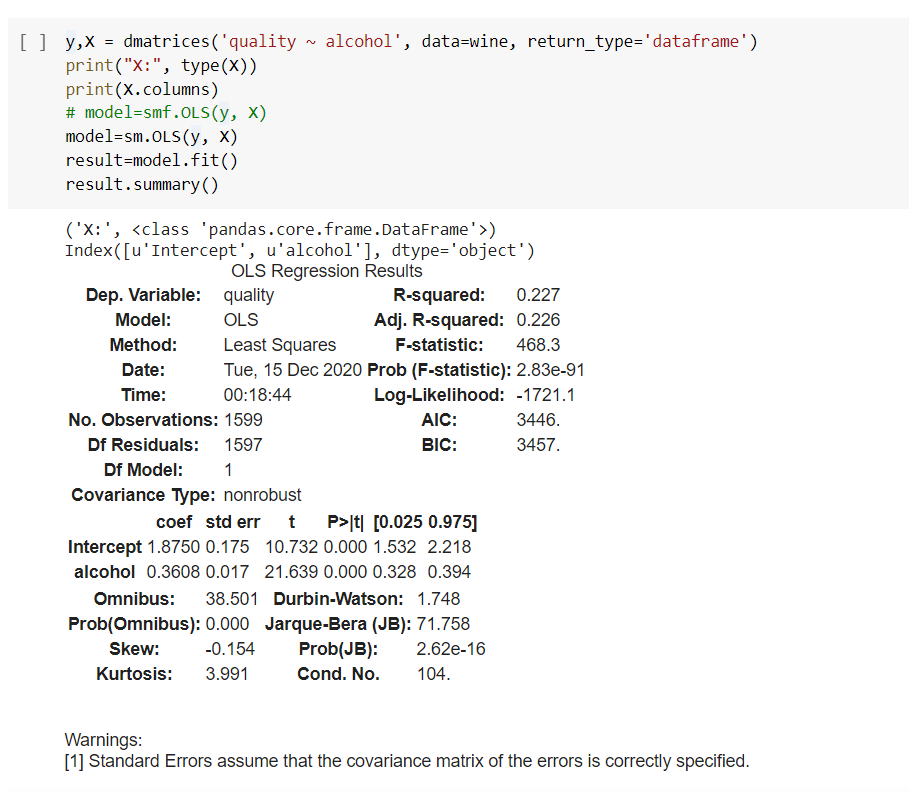
results = model.fit()

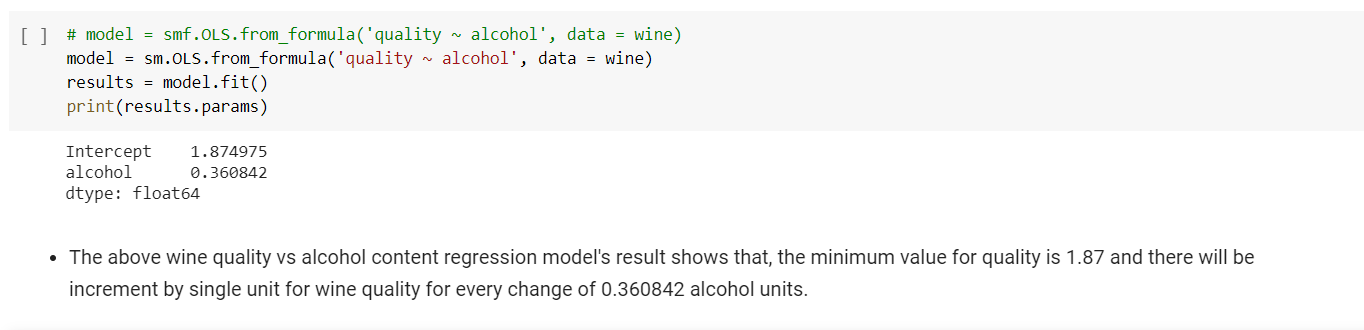
print(results.params)

* The above wine quality vs alcohol content regression model's result shows that, the minimum value for quality is 1.87 and there will be increment by single unit for wine quality for every change of 0.360842 alcohol units.

Screenshots







# Classification using Statsmodel

* We will use statsmodel for this logistic regression analysis of predicting good wine quality (>4).
* Let's create a new categorical variable/column (rate\_code) with two possible values (good = 1 & bad = 0).

Code

wine['rate\_code'] = (wine['quality'] > 4).astype(np.float32)

y, X = dmatrices('rate\_code ~ alcohol', data = wine)

sns.distplot(X[y[:,0] > 0, 1])

sns.distplot(X[y[:,0] == 0, 1])

model = sm.Logit(y, X)

result = model.fit()

result.summary2()

yhat = result.predict(X)

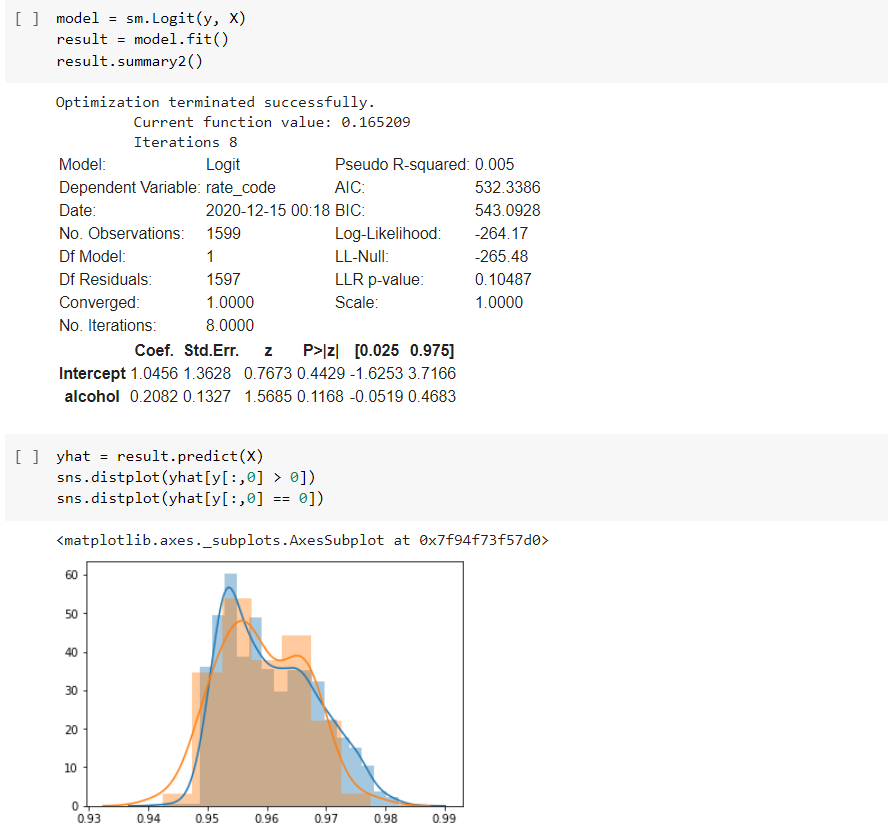
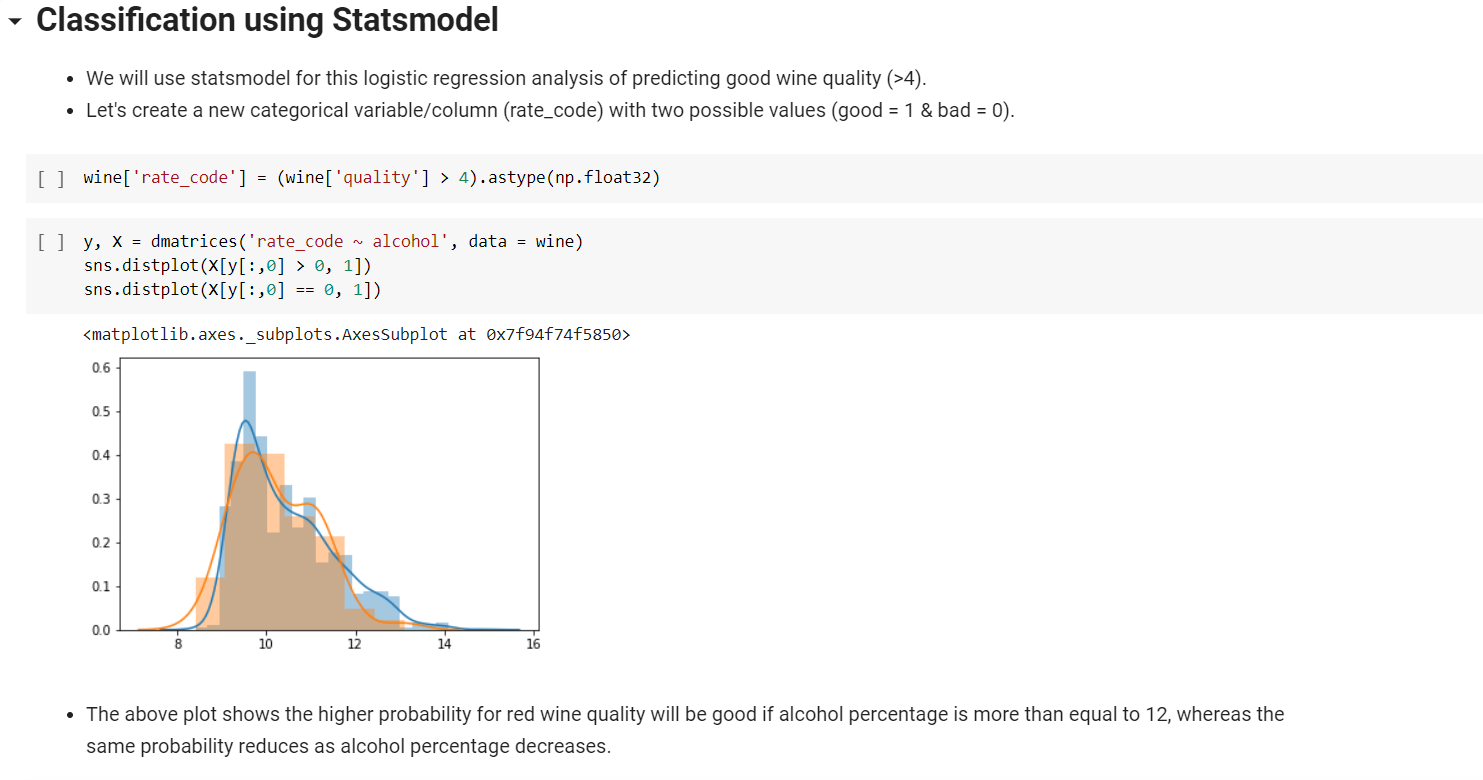
sns.distplot(yhat[y[:,0] > 0])

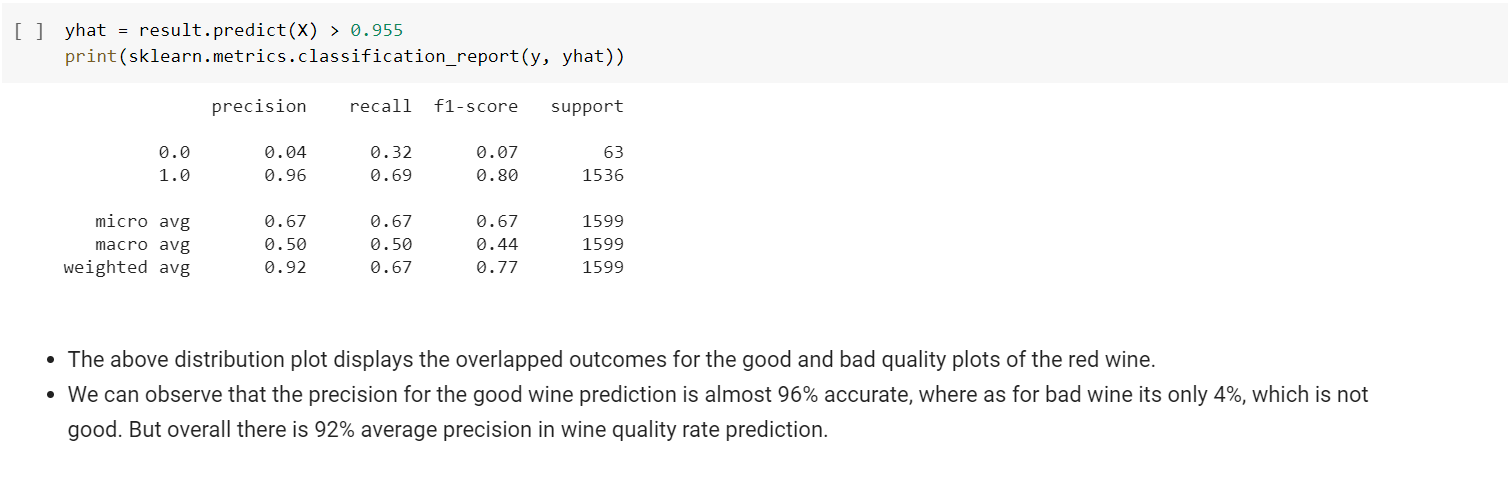
sns.distplot(yhat[y[:,0] == 0])

yhat = result.predict(X) > 0.955

print(sklearn.metrics.classification\_report(y, yhat))

* The above distribution plot displays the overlapped outcomes for the good and bad quality plots of the red wine.
* We can observe that the precision for the good wine prediction is almost 96% accurate, where as for bad wine its only 4%, which is not good. But overall there is 92% average precision in wine quality rate prediction.

Screenshots



# Classification using Sklearn's LogisticRegression

Code

**Splitting the dataset - training and validation**

from sklearn.model\_selection import train\_test\_split

y,X= dmatrices('rate\_code ~ alcohol + sulphates + citric\_acid + fixed\_acidity', data = wine)

# split into train test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

print("Shape of X\_train,X\_test,Y\_train,Y\_test : ")

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

**Train the model**

# Show hyper parameters and try to do tunning

model = sklearn.linear\_model.LogisticRegression()

# train the model with X\_train and y\_train

model.fit(X\_train, y\_train)

**Prediction and classification report**

# Use Trained model to perform prediction with X\_test and y\_test

yhat = model.predict(X\_test)

print(sklearn.metrics.classification\_report(y\_test, yhat))

* The accuracy matrix for sklearn's linear regression model for red wine quality prediction shows the overall 96% precision which is greater than previous statsmodel's average precision.
* Also the precision for good wine (1) prediction is almost 97%.
* But the precision is almost 0% for the bad type of wine (0) with sklearn's linear regression model. Which is not a good sign for the analysis.

**Confusion Matrix and Stats:**

!pip install pandas\_ml

from pandas\_ml import ConfusionMatrix

data = {'y\_Actual':    np.reshape(np.array(y\_test),-1),

        'y\_Predicted': np.reshape(np.array(yhat),-1)

        }

df = pd.DataFrame(data, columns=['y\_Actual','y\_Predicted'])

Confusion\_Matrix = ConfusionMatrix(df['y\_Actual'], df['y\_Predicted'])

Confusion\_Matrix.print\_stats()

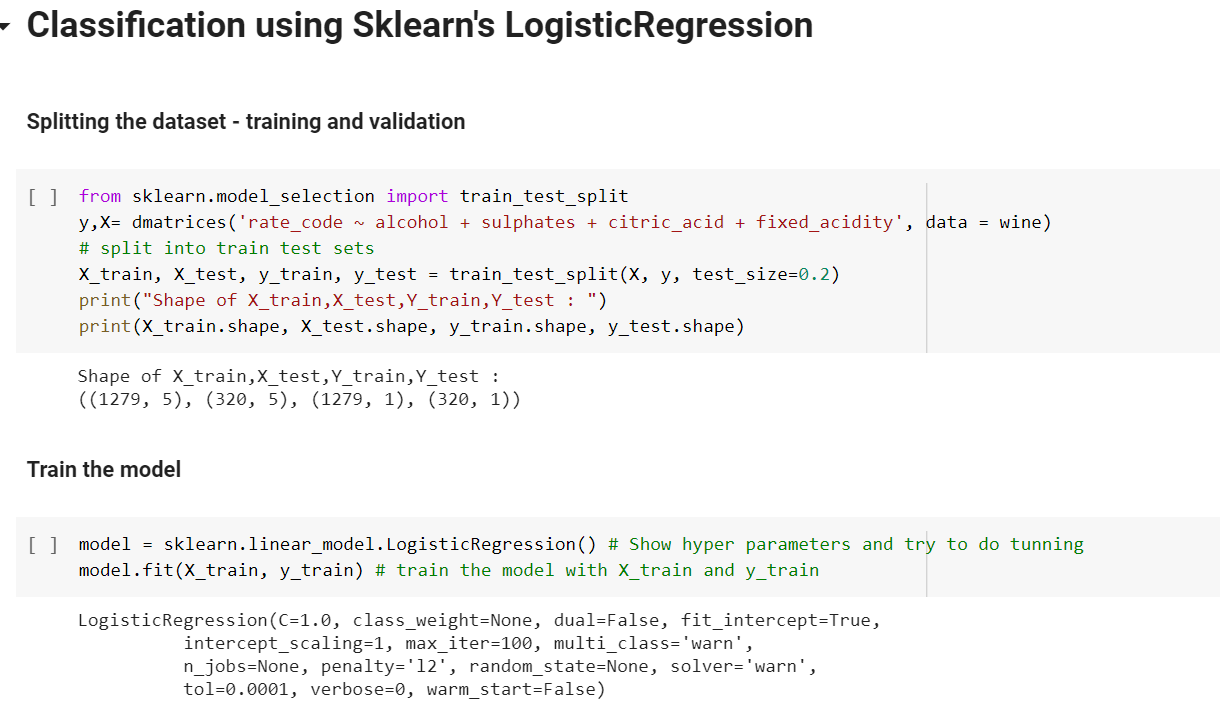
data = {'y\_Actual': np.reshape(np.array(y\_test),-1), 'y\_Predicted': np.reshape(np.array(yhat),-1)}

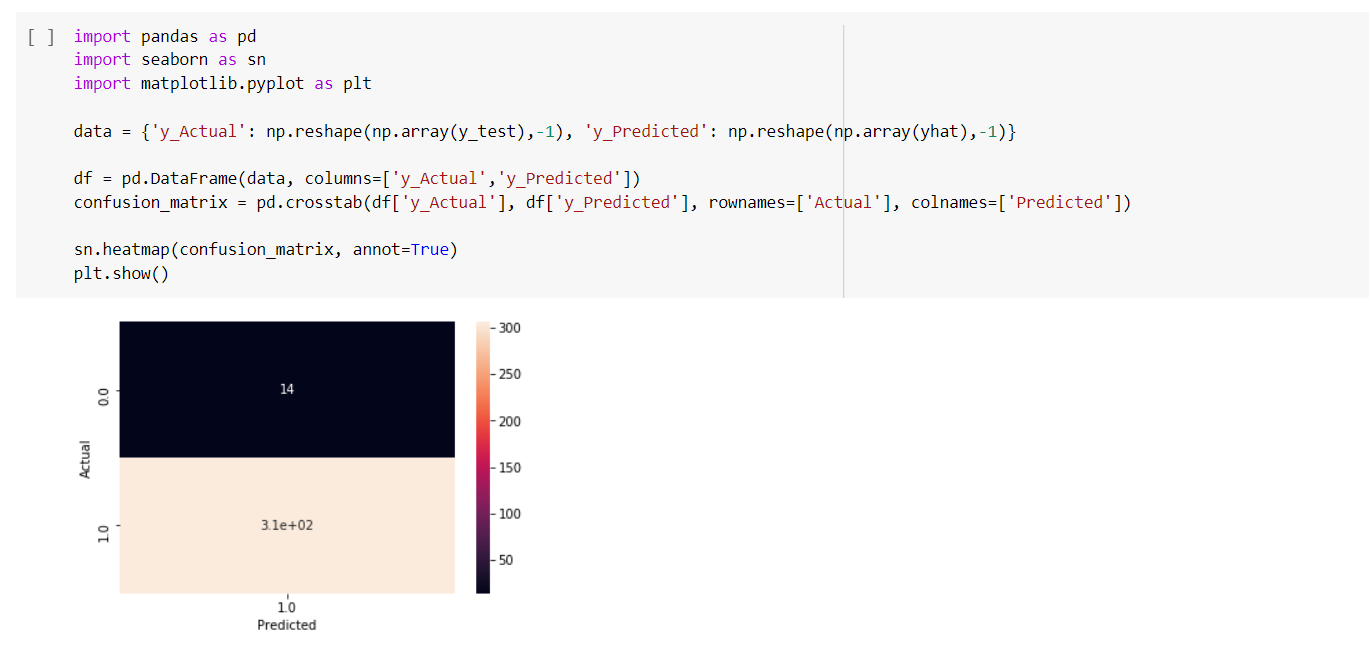
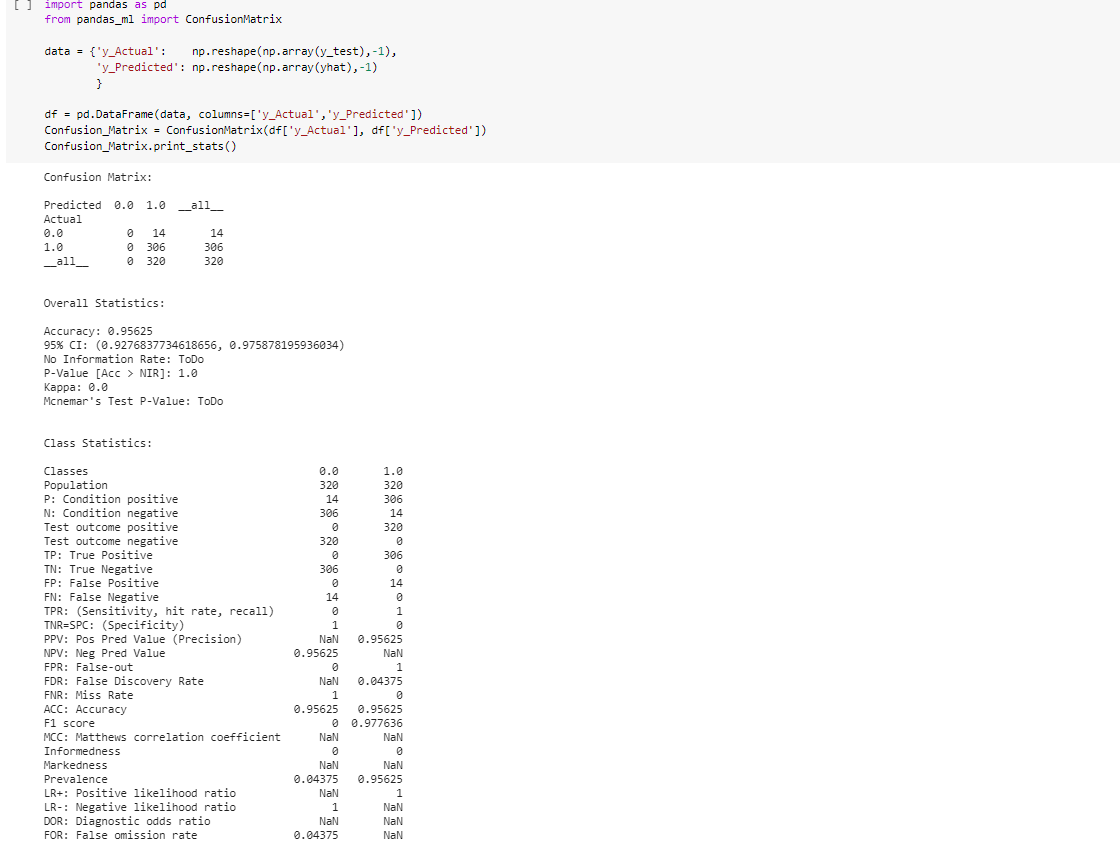
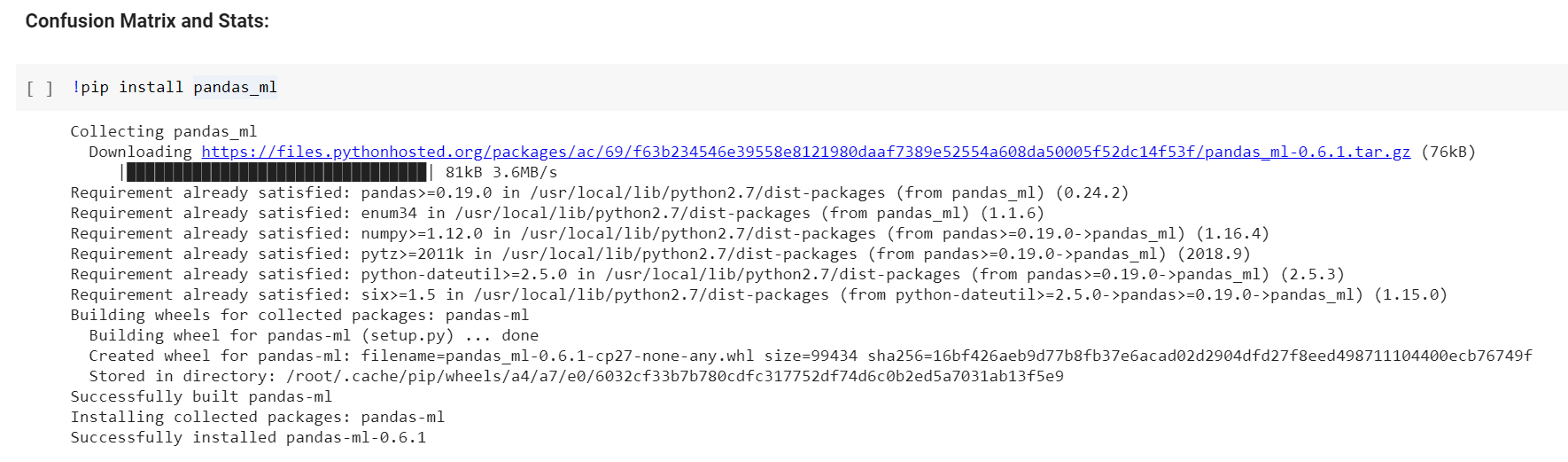
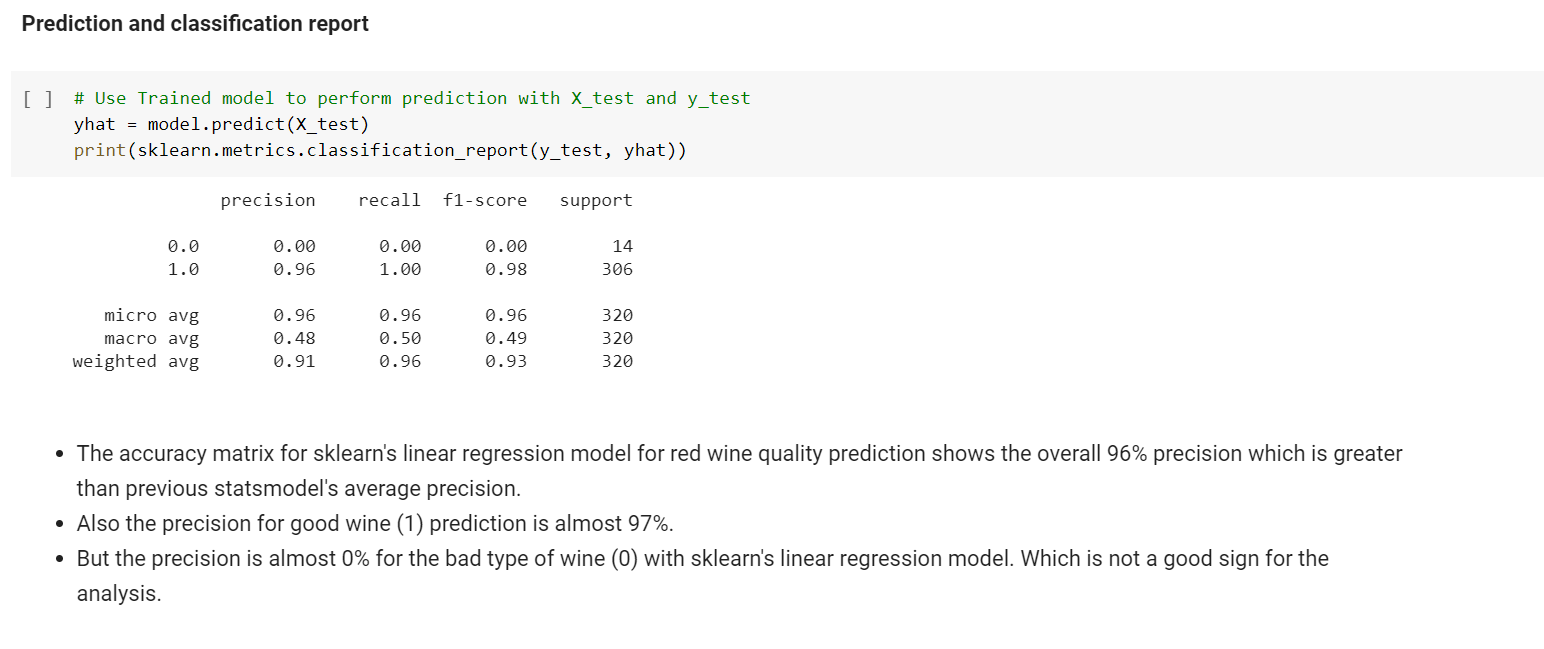
df = pd.DataFrame(data, columns=['y\_Actual','y\_Predicted'])

confusion\_matrix = pd.crosstab(df['y\_Actual'], df['y\_Predicted'], rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion\_matrix, annot=True)

plt.show()

Screenshots****



# Classification using Sklearn's RandomForestClassifier

The models include a simple train test split, training the model, prediction and classification report and a graph of the confusion matrix and stats.

Code

**Splitting the dataset - training and validation**

from sklearn.model\_selection import train\_test\_split

y, X = dmatrices('rate\_code ~ alcohol', data = wine)

# split into train test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

print("Shape of X\_train,X\_test,Y\_train,Y\_test : ")

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

**Train the model**

# Add hyper params and peform tunning

model = sklearn.ensemble.RandomForestClassifier()

# train the model

model.fit(X\_train, y\_train)

**Prediction and classification report**

# Use Trained model to perform prediction

yhat = model.predict(X\_test)

print(sklearn.metrics.classification\_report(y\_test, yhat))

* Here, with the accuracy matrix for sklearn's random forest classifier model for the prediction of red wine quality, we can observe that the values have been improved significantly.
* The precision for the prediction of bad quality wine (0) is almost 100% where as the precision for prediction of good quality wine (1) is approximately 96%.
* This sklearn's random forest classifier model also has the overall precision around 96%, which is far better than the previous two models (i.e. statsmodel and sklearn's linear regression model)

**Confusion Matrix and stats**

from pandas\_ml import ConfusionMatrix

data = {'y\_Actual':    np.reshape(np.array(y\_test),-1),

        'y\_Predicted': np.reshape(np.array(yhat),-1)

        }

df = pd.DataFrame(data, columns=['y\_Actual','y\_Predicted'])

Confusion\_Matrix = ConfusionMatrix(df['y\_Actual'], df['y\_Predicted'])

Confusion\_Matrix.print\_stats()

data = {'y\_Actual': np.reshape(np.array(y\_test),-1), 'y\_Predicted': np.reshape(np.array(yhat),-1)}

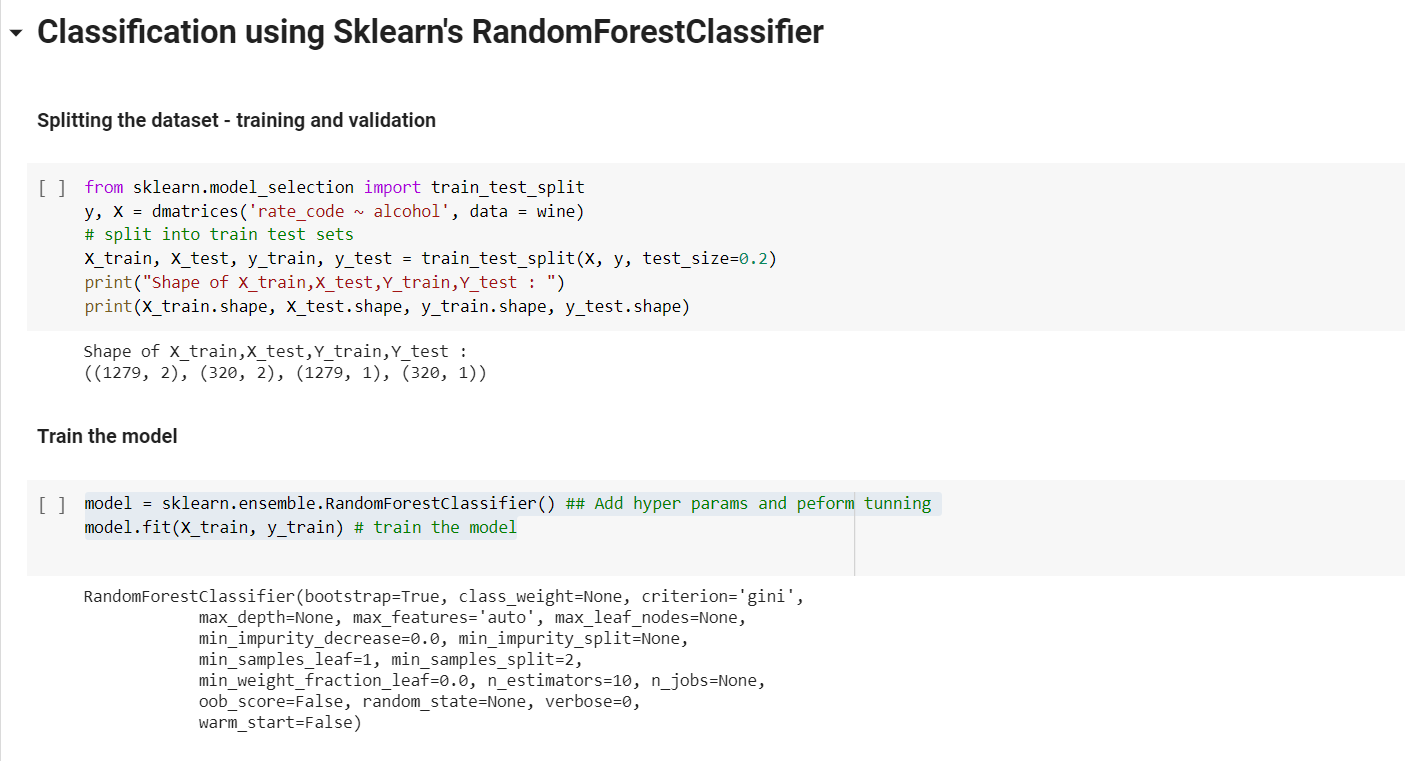
df = pd.DataFrame(data, columns=['y\_Actual','y\_Predicted'])

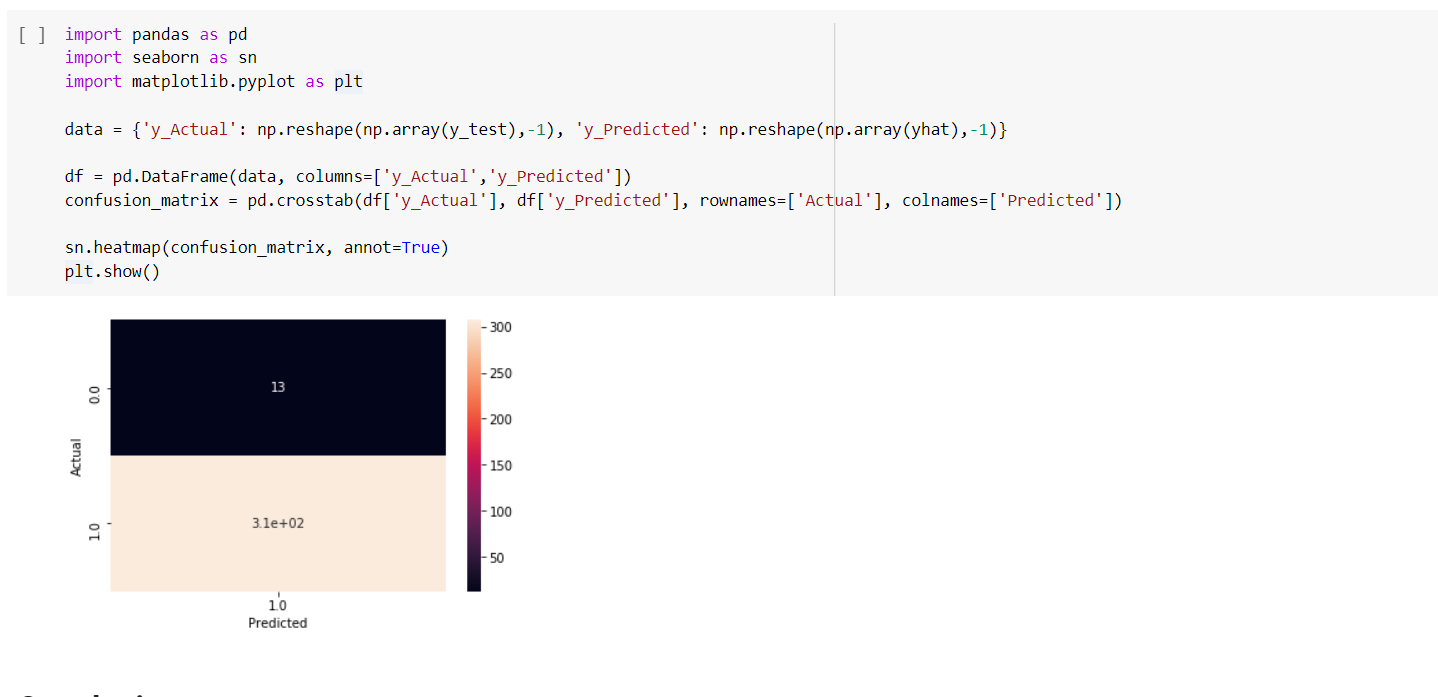
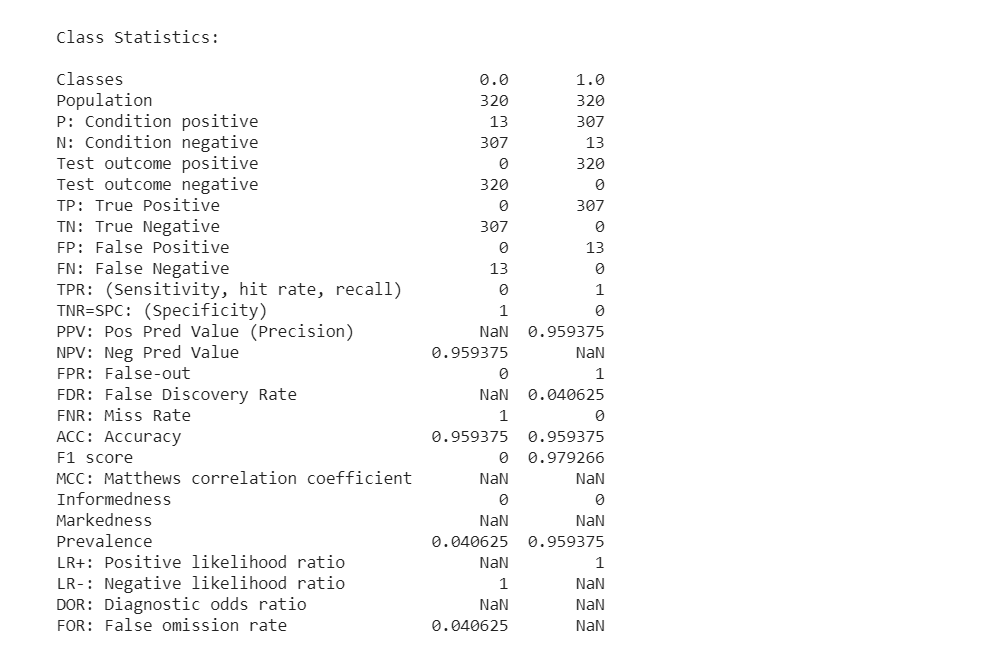
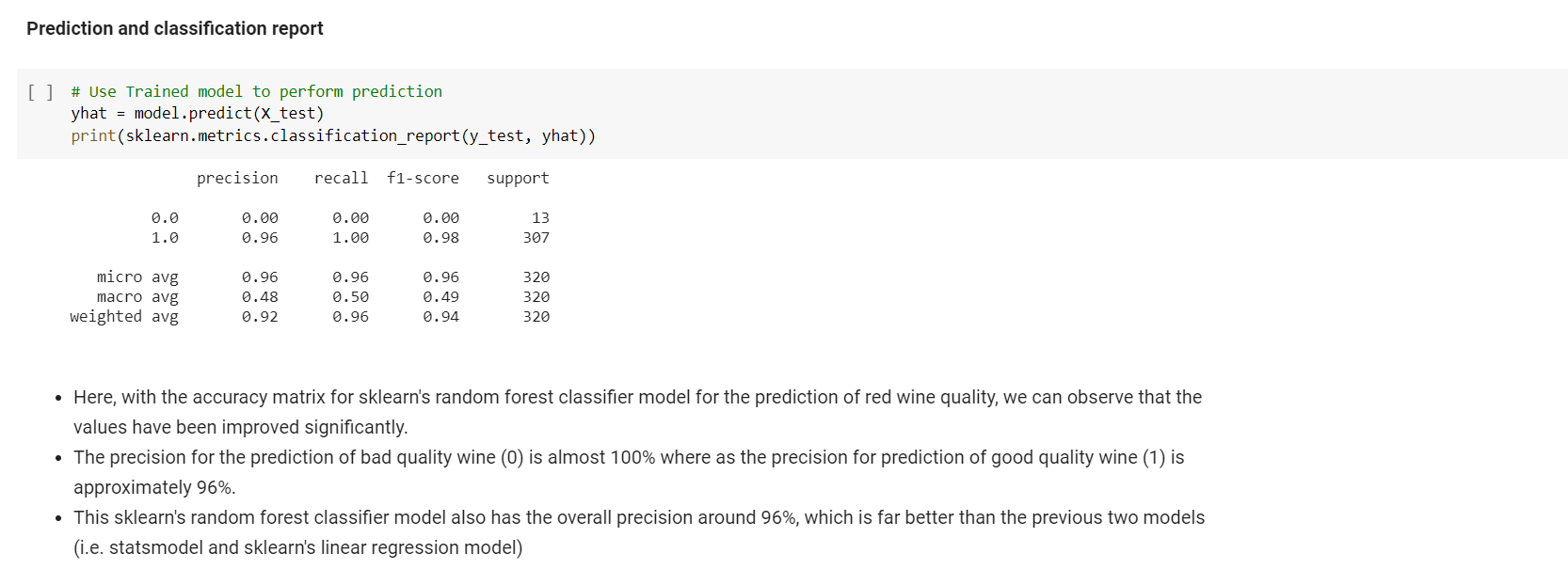
confusion\_matrix = pd.crosstab(df['y\_Actual'], df['y\_Predicted'], rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion\_matrix, annot=True)

plt.show()

Screenshots





# ****Conclusion****

* We observed the key factors that determine and affects the quality of the red wine. Wine quality is ultimately a subjective measure. The ordered factor 'quality' was not very helpful and to overcome this, so we created another variable called 'rating'.
* To make predictions of wine quality and any other if required, we trained two models. As seen, the statsmodel and sklearn's Linear Regression model along with Random Forest Classifier. The Random Forest Classifier performed marginally better and we decided to stick with it if we had to make any more predictions.
* The usage of this analysis will help to understand whether by modifying the variables, it is possible to increase the quality of the wine on the market. If you can control your variables, then you can predict the quality of your wine and obtain more profits.