Experiment 4: Principal Component Analysis

Aim:
To PCA on a dataset and compare accuracy of models before and after the dimensionality reduction

Problem Statement:
Choose a classification dataset of your choice from any of the following Repository Links, download it:

1. Kaggle: https://www.kaggle.com/
2. UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php

Perform Linear Regression on the chosen dataset.

Your notebook should contain:
1. Basic EDA
2. Creation of MultiLayered Perceptron Model using sklearn and subsequent training on Training set
3. Test the model on test set by printing on the classification metrics

[***Hint***: Follow the steps in Titanic notebook uploaded on moodle under Expt 3 reference material]

Tool/Language:
Programming language: Python
Libraries: numpy, pandas, sklearn, matplotlib, seaborn

Code with visualisation graphs:
1) **Dataset Chosen:** Red Wine Quality
2) **Dataset Description:** This dataset is related to red variants of the Portuguese "Vinho Verde" wine.
   a) Input variables (based on physicochemical tests):
      - fixed acidity (most acids involved with wine or fixed or non-volatile)
      - volatile acidity (the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste)
      - citric acid (found in small quantities, citric acid can add 'freshness' and flavour to wines)
      - residual sugar (the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/litre)
      - chlorides (the amount of salt in the wine)
      - free Sulphur dioxide (the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulphite ion)
      - total Sulphur dioxide (amount of free and bound forms of S02)
      - density (the density of water is close to that of water depending on the percent alcohol and sugar content)
      - pH (describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic))

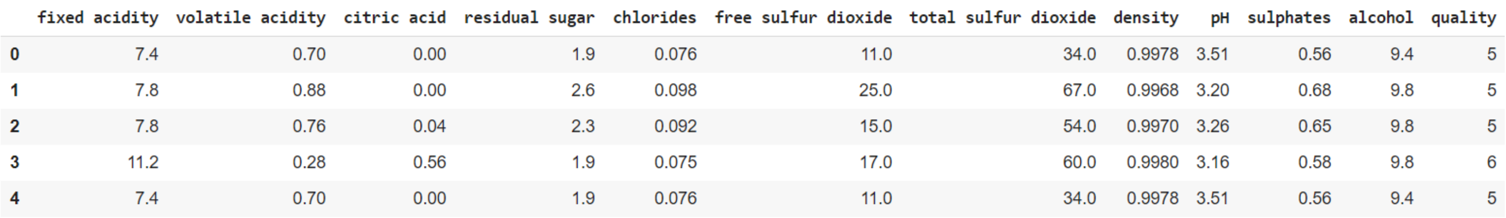
- sulphates (a wine additive which can contribute to Sulphur dioxide gas (S02) levels, which acts as an antimicrobial)
- alcohol

b) Output variable (based on sensory data):

- quality (score between 0 and 10)

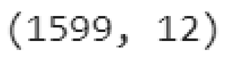
3) Code:

```
from google.colab import files
uploaded = files.upload()

# Basic Pre-processing
import numpy as np
import pandas as pd
import io

# For Model Selection and Training
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sco
re

df = pd.read_csv(io.BytesIO(uploaded['winequality-red.csv']))
df.head()
```



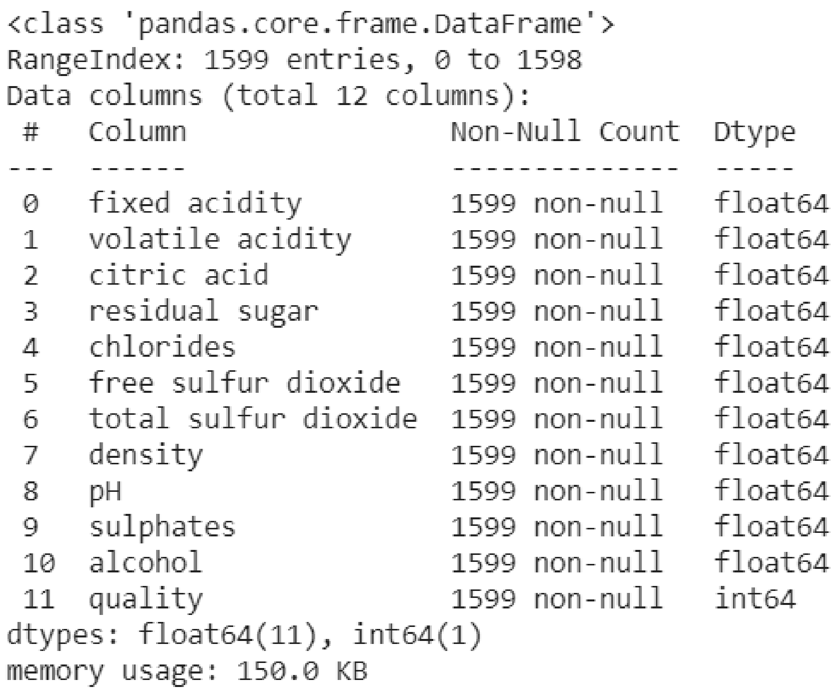
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	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
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```
df.shape
```

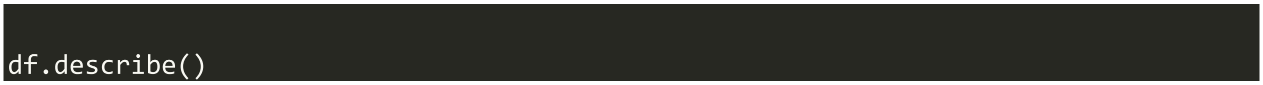


```
(1599, 12)
```

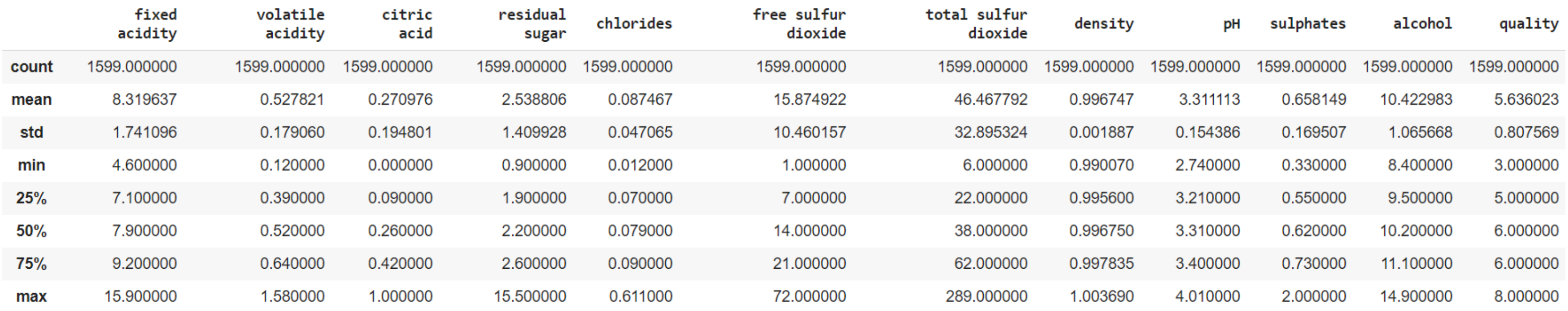
```
df.info()
```



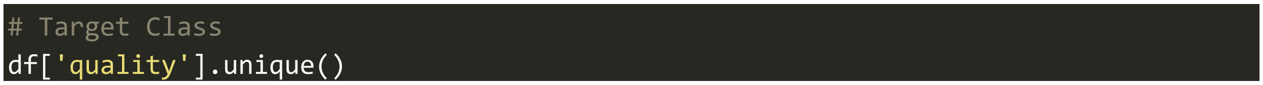
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   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
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```



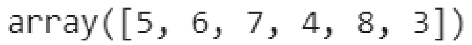
```
df.describe()
```



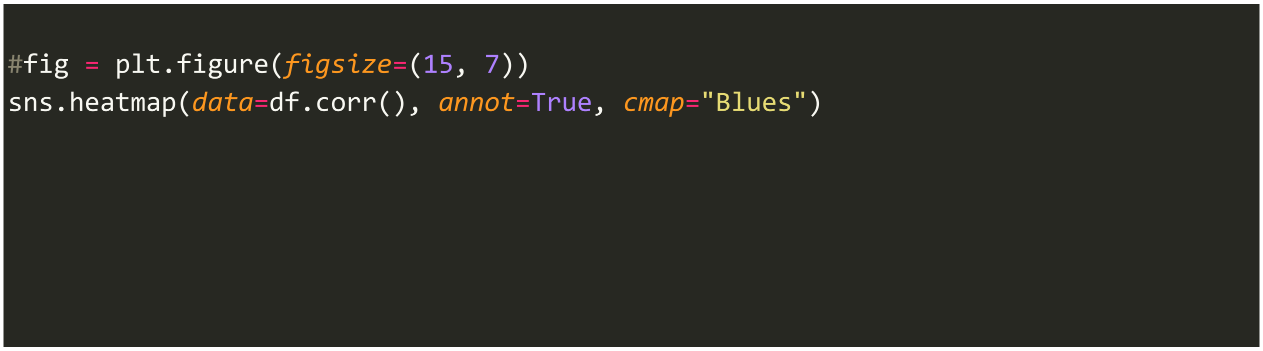
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000



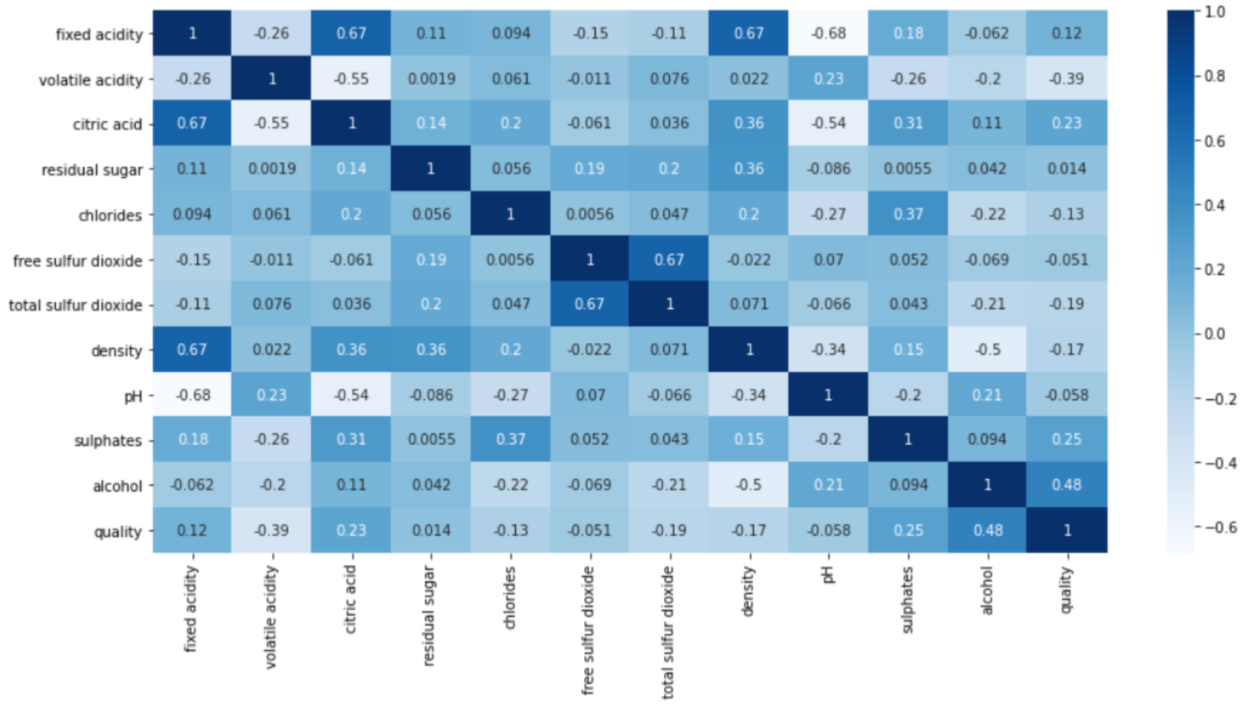
```
# Target Class
df['quality'].unique()
```



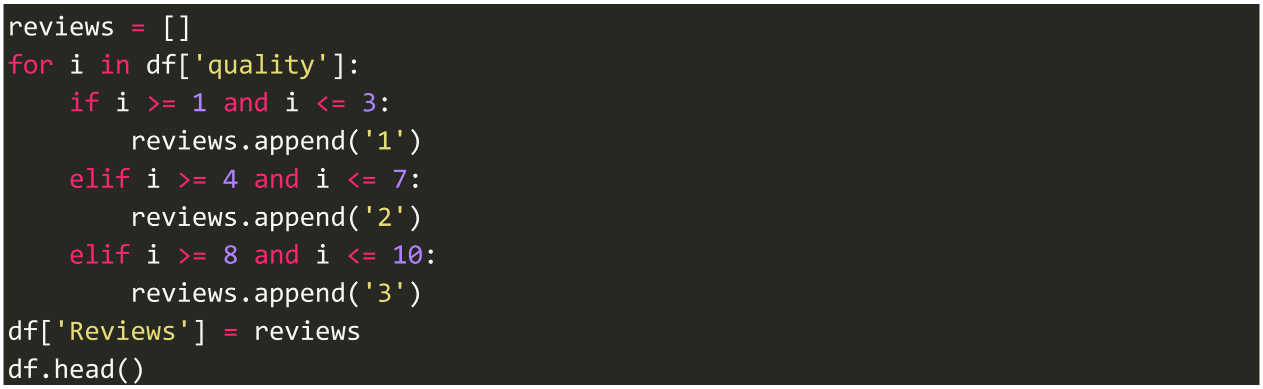
```
array([5, 6, 7, 4, 8, 3])
```



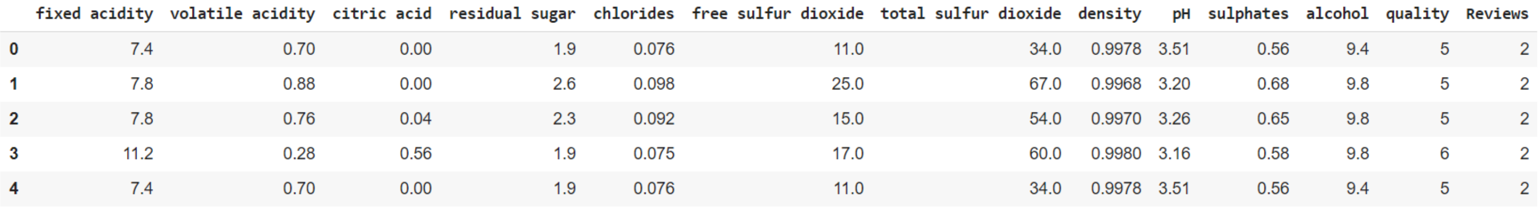
```
#fig = plt.figure(figsize=(15, 7))
sns.heatmap(data=df.corr(), annot=True, cmap="Blues")
```



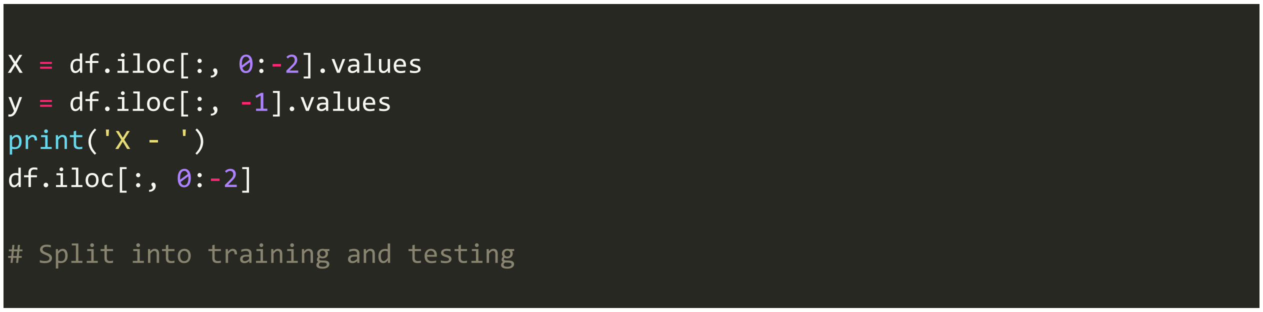
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1	-0.26	0.67	0.11	0.094	-0.15	-0.11	0.67	-0.68	0.18	-0.062	0.12
volatile acidity	-0.26	1	-0.55	0.0019	0.061	-0.011	0.076	0.022	0.23	-0.26	-0.2	-0.39
citric acid	0.67	-0.55	1	0.14	0.2	-0.061	0.036	0.36	-0.54	0.31	0.11	0.23
residual sugar	0.11	0.0019	0.14	1	0.056	0.19	0.2	0.36	-0.086	0.0055	0.042	0.014
chlorides	0.094	0.061	0.2	0.056	1	0.0056	0.047	0.2	-0.27	0.37	-0.22	-0.13
free sulfur dioxide	-0.15	-0.011	-0.061	0.19	0.0056	1	0.67	-0.022	0.07	0.052	-0.069	-0.051
total sulfur dioxide	-0.11	0.076	0.036	0.2	0.047	0.67	1	0.071	-0.066	0.043	-0.21	-0.19
density	0.67	0.022	0.36	0.36	0.2	-0.022	0.071	1	-0.34	0.15	-0.5	-0.17
pH	-0.68	0.23	-0.54	-0.086	-0.27	0.07	-0.066	-0.34	1	-0.2	0.21	-0.058
sulphates	0.18	-0.26	0.31	0.0055	0.37	0.052	0.043	0.15	-0.2	1	0.094	0.25
alcohol	-0.062	-0.2	0.11	0.042	-0.22	-0.069	-0.21	-0.5	0.21	0.094	1	0.48
quality	0.12	-0.39	0.23	0.014	-0.13	-0.051	-0.19	-0.17	-0.058	0.25	0.48	1



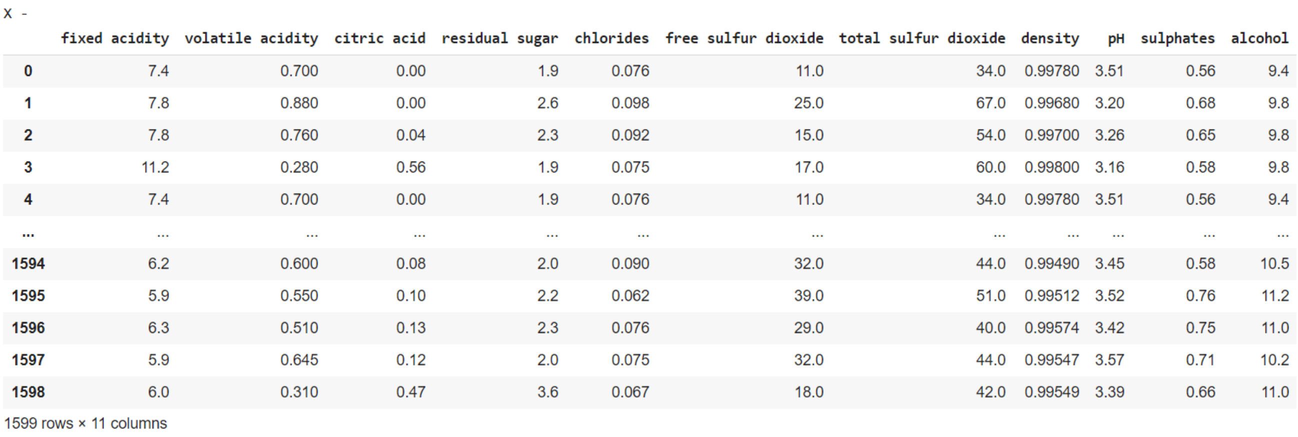
```
reviews = []
for i in df['quality']:
    if i >= 1 and i <= 3:
        reviews.append('1')
    elif i >= 4 and i <= 7:
        reviews.append('2')
    elif i >= 8 and i <= 10:
        reviews.append('3')
df['Reviews'] = reviews
df.head()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Reviews
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	2
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
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	2
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	2



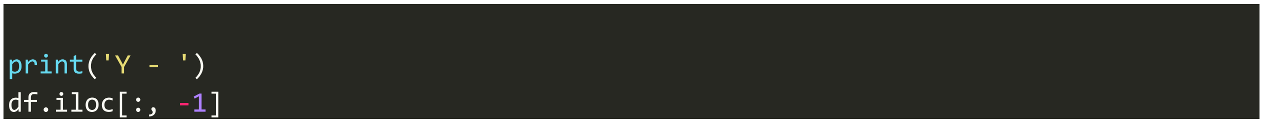
```
X = df.iloc[:, 0:-2].values
y = df.iloc[:, -1].values
print('X - ')
df.iloc[:, 0:-2]

# Split into training and testing
```



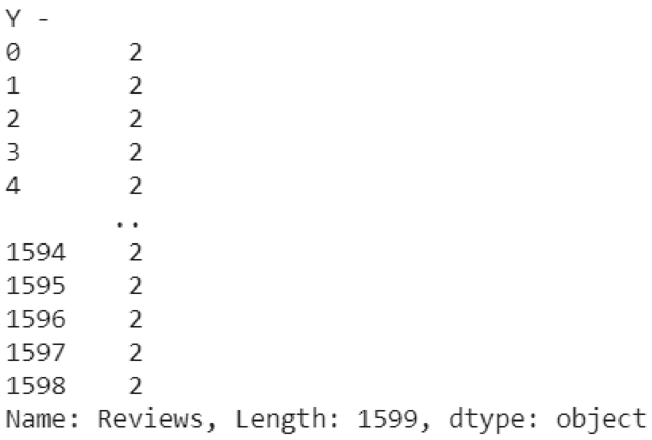
x -

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

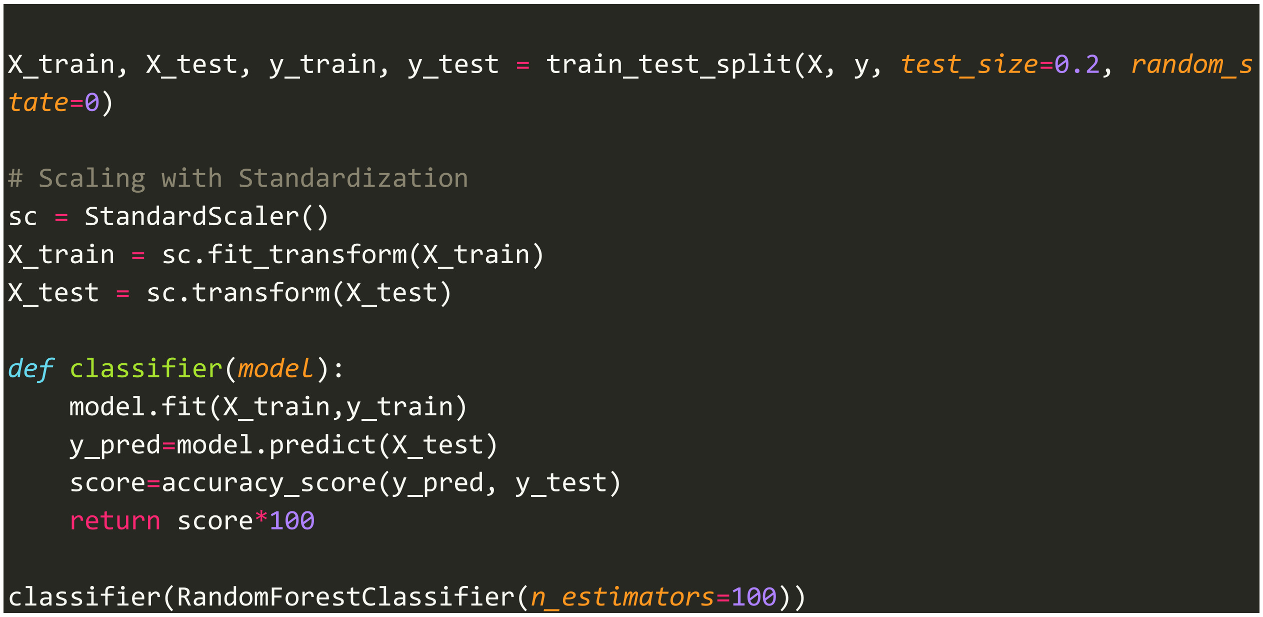
1599 rows × 11 columns



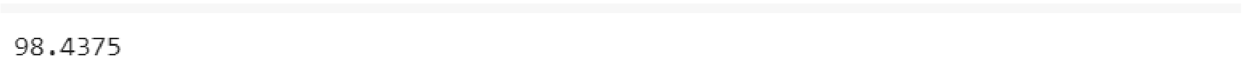
```
print('Y - ')
df.iloc[:, -1]
```



```
Y -
0       2
1       2
2       2
3       2
4       2
       ..
1594    2
1595    2
1596    2
1597    2
1598    2
Name: Reviews, Length: 1599, dtype: object
```

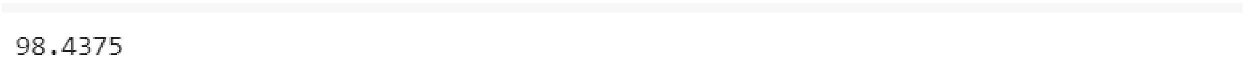


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
tate=0)

# Scaling with Standardization
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

def classifier(model):
    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    score=accuracy_score(y_pred, y_test)
    return score*100

classifier(RandomForestClassifier(n_estimators=100))
```



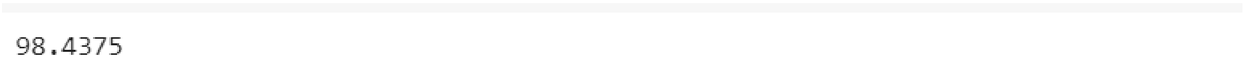
```
98.4375
```

```
classifier(LogisticRegression())
```



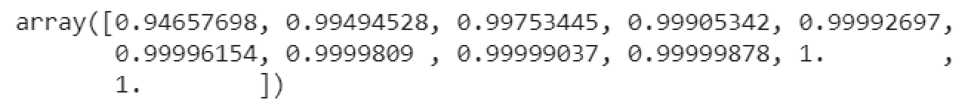
```
98.4375
```

```
classifier(GaussianNB())
```



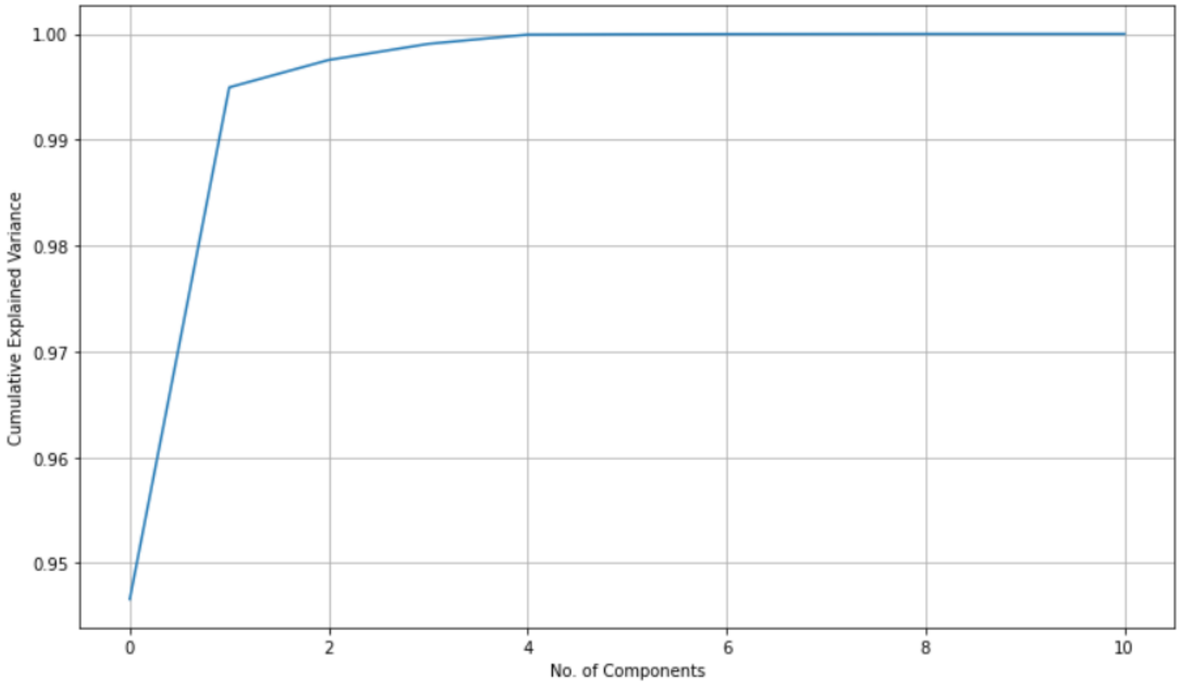
```
98.4375
```

```
# Check for Dimensionality Reduction

pca = PCA()
X_PCA = pca.fit_transform(X)
np.cumsum(pca.explained_variance_ratio_)
```



```
array([0.94657698, 0.99494528, 0.99753445, 0.99905342, 0.99992697,
       0.99996154, 0.9999809 , 0.99999037, 0.99999878, 1.        ,
       1.        ])
```

```
plt.figure(figsize=(12,7))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('No. of Components')
plt.ylabel('Cumulative Explained Variance');
plt.grid()
```



```
pca = PCA(n_components = 2)  # Reducing the data set from 11 columns to 2 columns
.
X_train2 = pca.fit_transform(X_train)
X_test2 = pca.transform(X_test)
X_train2.shape
```

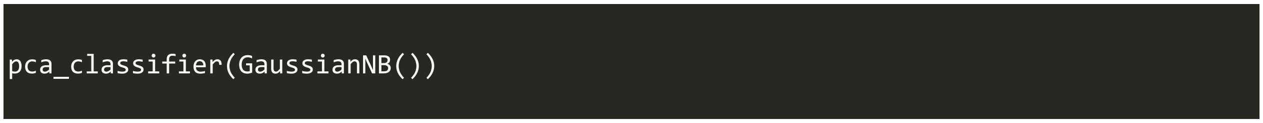
```
(1279, 2)
```

```
def pca_classifier(model):
    model.fit(X_train2,y_train)
    print(X_train2.shape)
    y_pred2=model.predict(X_test2)
    score=accuracy_score(y_pred2, y_test)
    return score*100

pca_classifier(RandomForestClassifier(n_estimators=100))
```

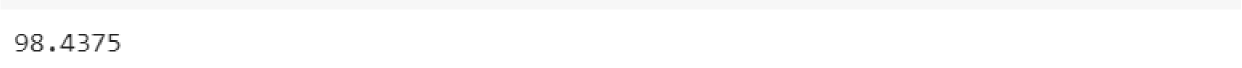
```
98.4375
```

```
pca_classifier(LogisticRegression())
```

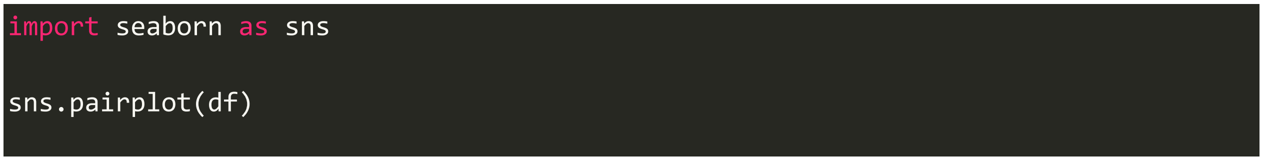
```
98.4375
```



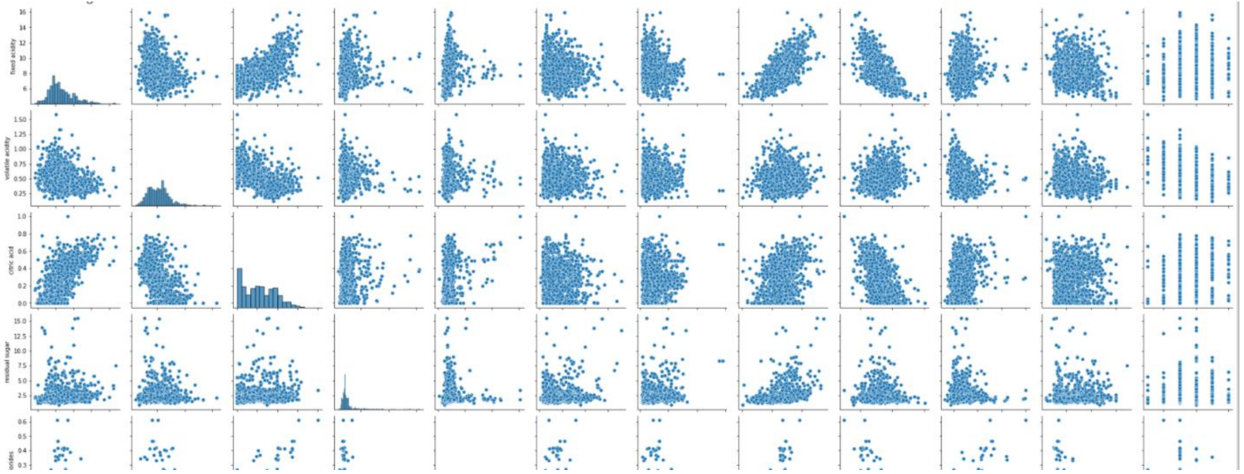
```
pca_classifier(GaussianNB())
```

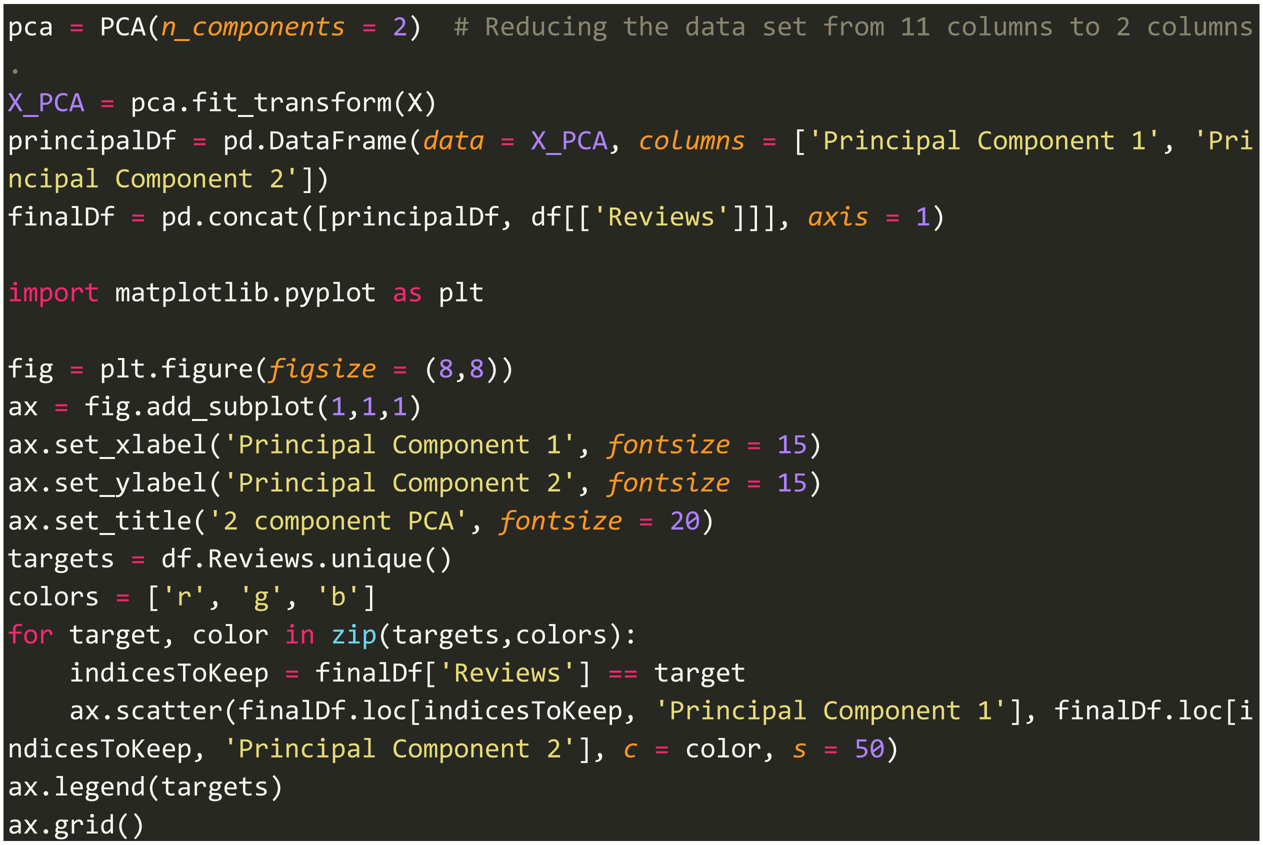


```
98.4375
```

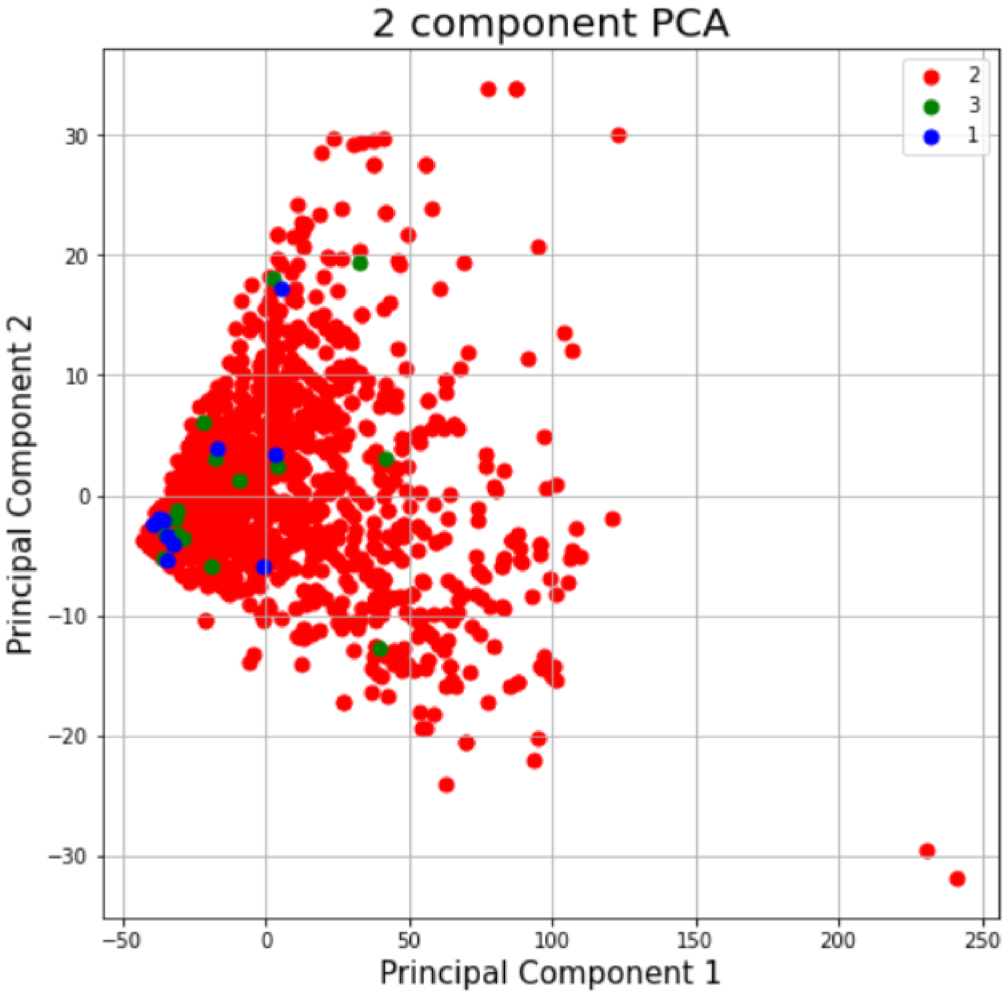


```
import seaborn as sns

sns.pairplot(df)
```





```
pca = PCA(n_components = 2)  # Reducing the data set from 11 columns to 2 columns
.
X_PCA = pca.fit_transform(X)
principalDf = pd.DataFrame(data = X_PCA, columns = ['Principal Component 1', 'Pri
ncipal Component 2'])
finalDf = pd.concat([principalDf, df[['Reviews']]], axis = 1)

import matplotlib.pyplot as plt

fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = df.Reviews.unique()
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['Reviews'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'Principal Component 1'], finalDf.loc[i
ndicesToKeep, 'Principal Component 2'], c = color, s = 50)
ax.legend(targets)
ax.grid()
```



Conclusion:

The model works pretty good for quality index of 5,6 and 7. Therefore we have reduced the target column from 10 different values to 3 different values – 1,2 and 3.

This is the primary reason that all the classification models work well even with 2 dimensions after applying PCA