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

```
df = pd.read_csv("/content/winequality-red.csv")
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	dens
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99
...	...	...	...	...	...	...	...	
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99

1599 rows × 12 columns

Exploratory Data Analysis

```
print("rows,columns:"+str(df.shape))
```

```
rows,columns:(1599, 12)
```

```
df.head()
```

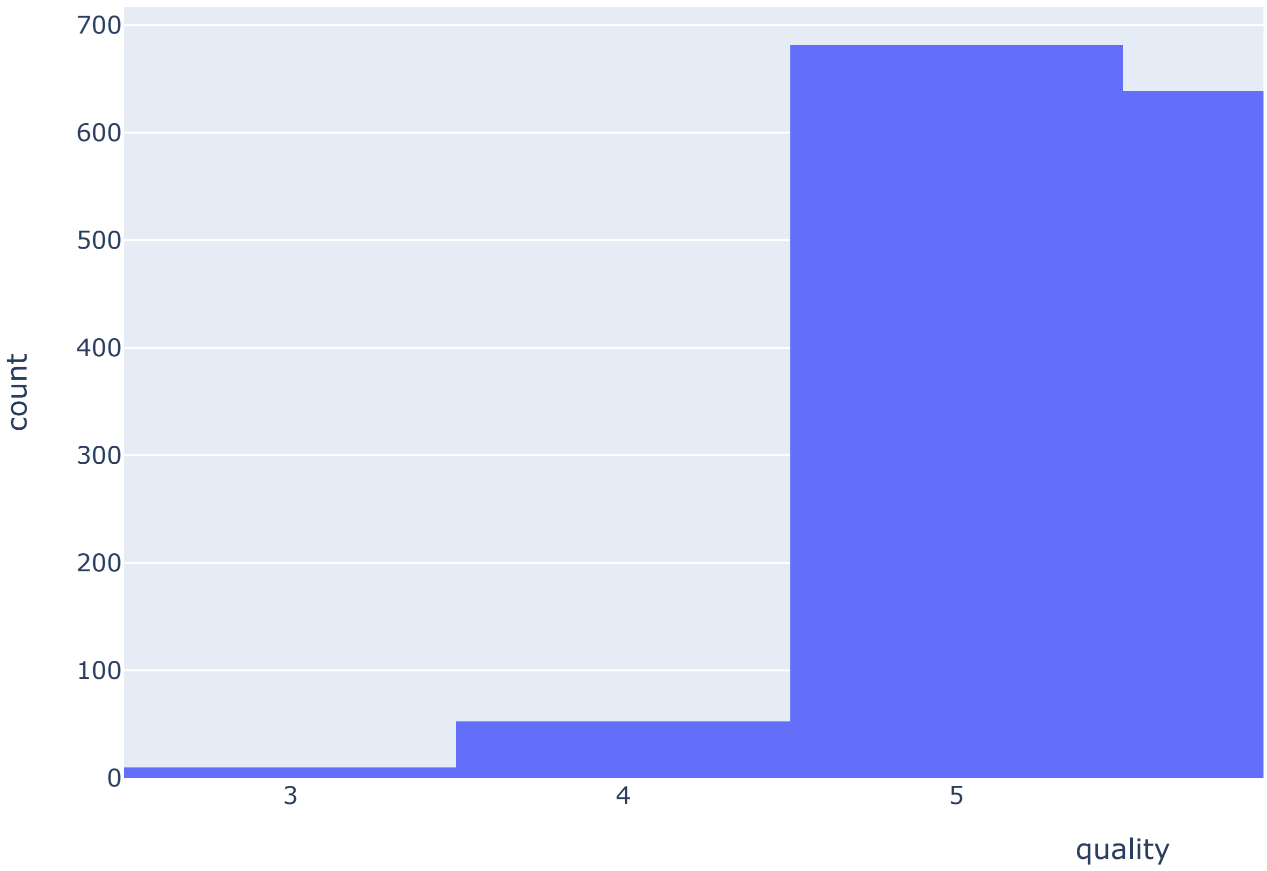
```
                                                          free         total
```

```
#missing variables
print(df.isna().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
dtype: int64
```

Exploring data using variables

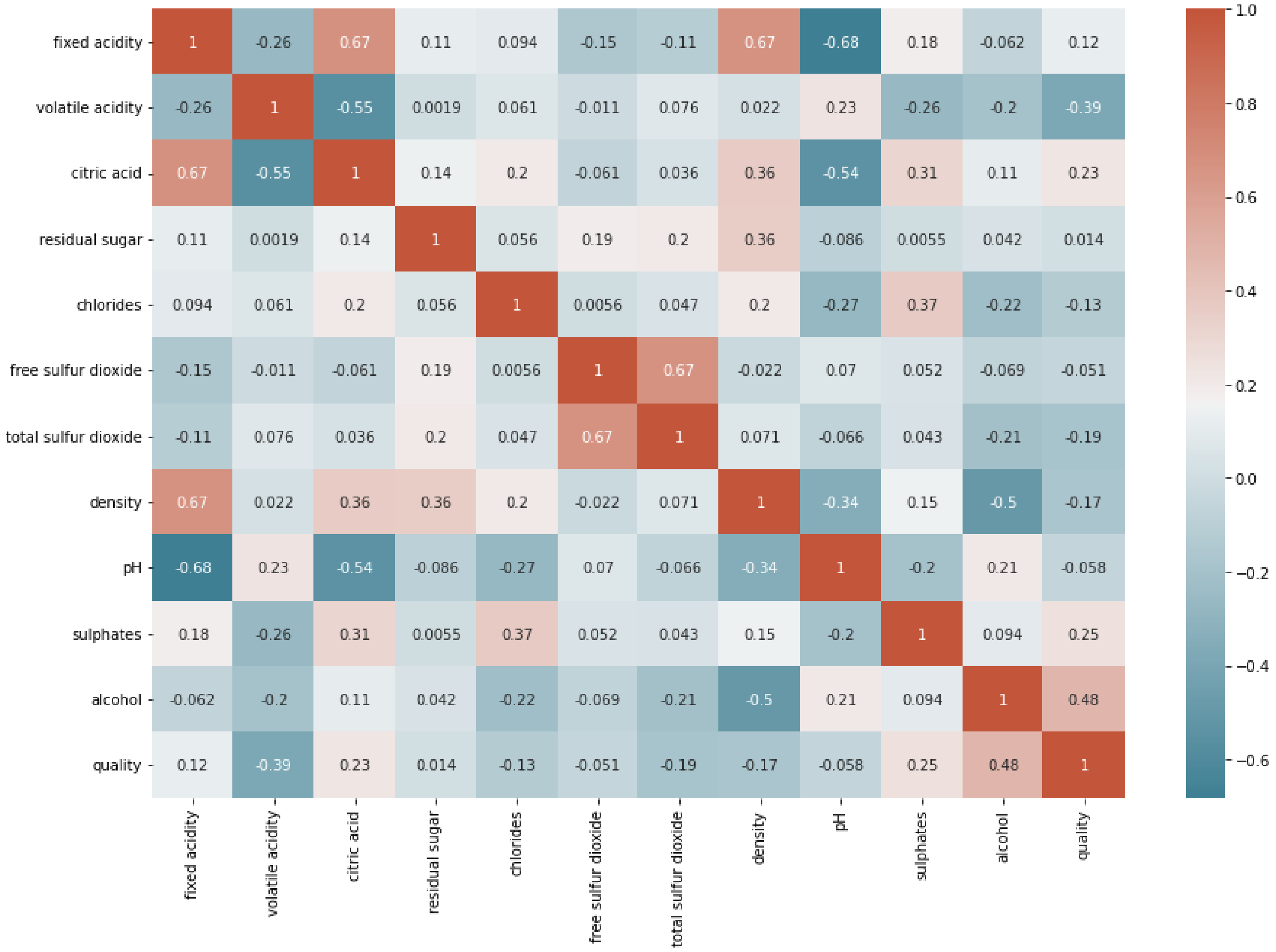
```
figure = px.histogram(df,x = 'quality')
figure
```



```
#to find the correlation between variables
```

```
corr = df.corr()
plt.figure(figsize = (15,10))
sns.heatmap(corr, xticklabels = corr.columns,yticklabels = corr.columns,annot = True,cmap
```

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7eff69a7a990>
```



	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

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

```
#see proportion of good wines vs bad wines
df['goodquality'].value_counts()
```

```
    0    1382
    1     217
    Name: goodquality, dtype: int64
```

```
#standardizing variables
from sklearn.preprocessing import StandardScaler
X_features = X
X = StandardScaler().fit_transform(X)
```

Split data

```
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 = 0)
```

```
from sklearn.ensemble import RandomForestClassifier
classifier1 = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_stat
classifier1.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier1.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[346   9]
 [ 23  22]]
0.92
```

```
from sklearn.svm import SVC
classifier2 = SVC(kernel = 'rbf', random_state = 0)
classifier2.fit(X_train, y_train)
```

```
SVC(random_state=0)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier2.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[349   6]
 [ 29  16]]
0.9125
```

```
import xgboost as xgb
model = xgb.XGBClassifier(random_state=1)
model.fit(X_train, y_train)
```

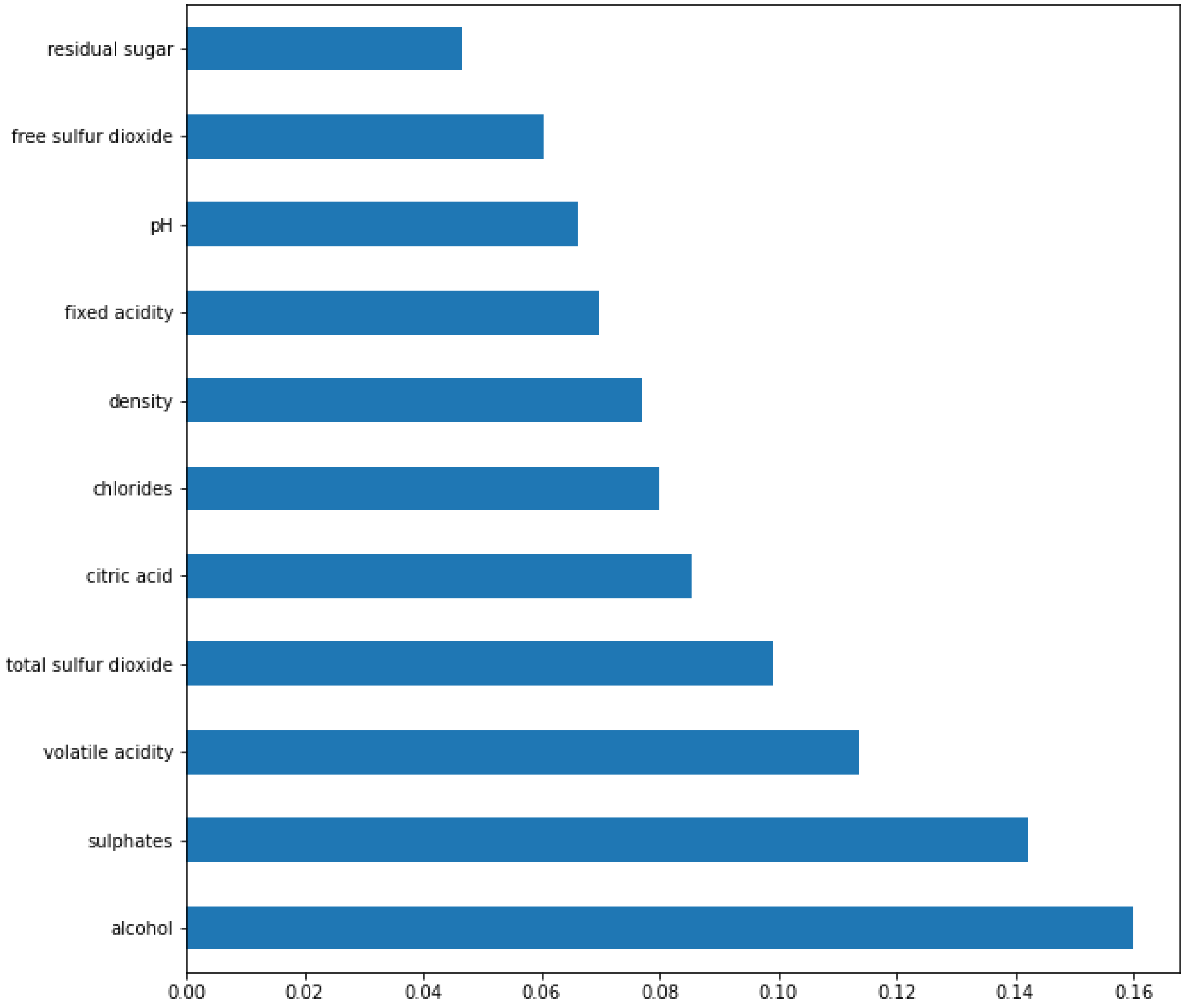
```
XGBClassifier(random_state=1)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[336  19]
 [ 20  25]]
0.9025
```

```
feat_importances = pd.Series(classifier1.feature_importances_, index=X_features.columns)
feat_importances.nlargest(25).plot(kind='barh',figsize=(10,10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7eff58c6f990>
```



```
# Filtering df for only good quality
df_temp = df[df['goodquality']==1]
df_temp.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	tota sulfu dioxi
count	217.000000	217.000000	217.000000	217.000000	217.000000	217.000000	217.00000
mean	8.847005	0.405530	0.376498	2.708756	0.075912	13.981567	34.8894
std	1.999977	0.144963	0.194438	1.363026	0.028480	10.234615	32.5722
min	4.900000	0.120000	0.000000	1.200000	0.012000	3.000000	7.00000
25%	7.400000	0.300000	0.300000	2.000000	0.062000	6.000000	17.00000
50%	8.700000	0.370000	0.400000	2.300000	0.073000	11.000000	27.00000
75%	10.100000	0.490000	0.490000	2.700000	0.085000	18.000000	43.00000

```
# Filtering df for only bad quality
df_temp2 = df[df['goodquality']==0]
df_temp2.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	
count	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	13
mean	8.236831	0.547022	0.254407	2.512120	0.089281	16.172214	
std	1.682726	0.176337	0.189665	1.415778	0.049113	10.467685	
min	4.600000	0.160000	0.000000	0.900000	0.034000	1.000000	
25%	7.100000	0.420000	0.082500	1.900000	0.071000	8.000000	
50%	7.800000	0.540000	0.240000	2.200000	0.080000	14.000000	
75%	9.100000	0.650000	0.400000	2.600000	0.091000	22.000000	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	1