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

```
df = pd.read_csv("/content/winequality-red.csv")
df.head(50)
```

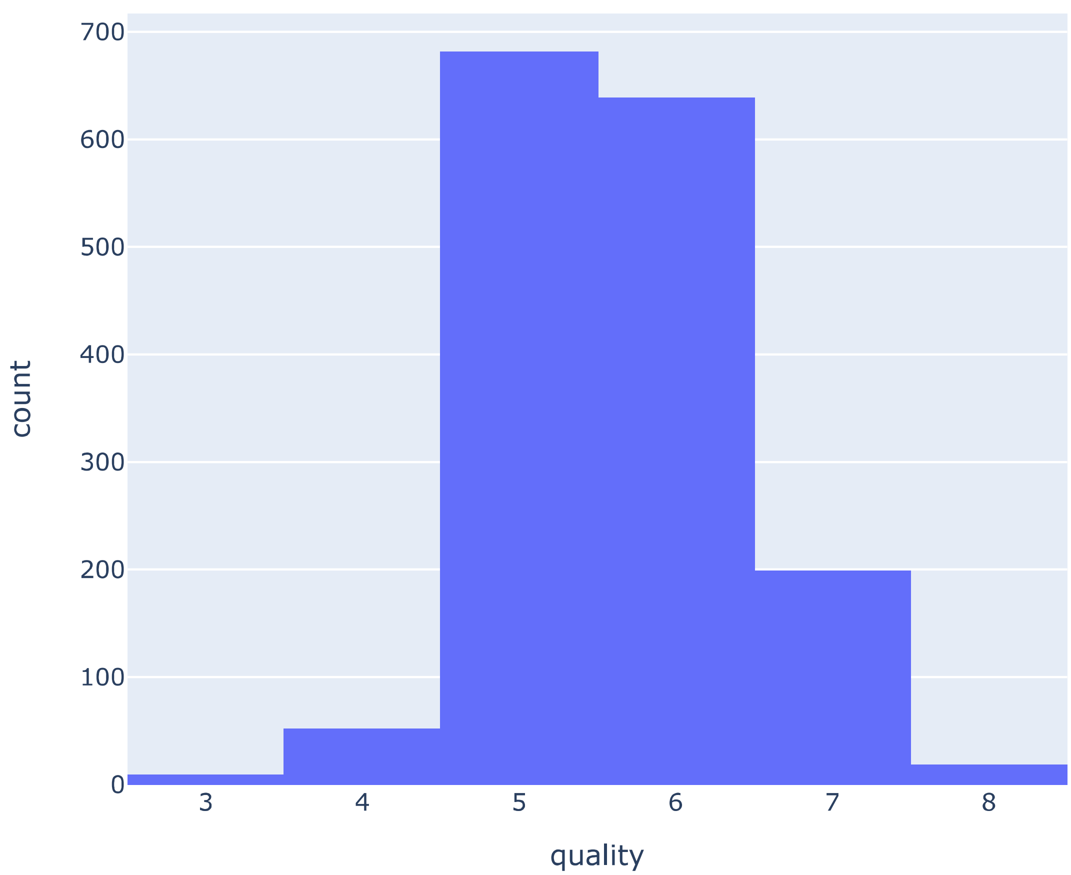
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sul
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	
5	7.4	0.660	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	
6	7.9	0.600	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	
7	7.3	0.650	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	
8	7.8	0.580	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	
9	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	
10	6.7	0.580	0.08	1.8	0.097	15.0	65.0	0.9959	3.28	
11	7.5	0.500	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	
12	5.6	0.615	0.00	1.6	0.089	16.0	59.0	0.9943	3.58	
13	7.8	0.610	0.29	1.6	0.114	9.0	29.0	0.9974	3.26	
14	8.9	0.620	0.18	3.8	0.176	52.0	145.0	0.9986	3.16	
15	8.9	0.620	0.19	3.9	0.170	51.0	148.0	0.9986	3.17	
16	8.5	0.280	0.56	1.8	0.092	35.0	103.0	0.9969	3.30	
17	8.1	0.560	0.28	1.7	0.368	16.0	56.0	0.9968	3.11	
18	7.4	0.590	0.08	4.4	0.086	6.0	29.0	0.9974	3.38	
19	7.9	0.320	0.51	1.8	0.341	17.0	56.0	0.9969	3.04	
20	8.9	0.220	0.48	1.8	0.077	29.0	60.0	0.9968	3.39	
21	7.6	0.390	0.31	2.3	0.082	23.0	71.0	0.9982	3.52	
22	7.9	0.430	0.21	1.6	0.106	10.0	37.0	0.9966	3.17	
23	8.5	0.490	0.11	2.3	0.084	9.0	67.0	0.9968	3.17	
24	6.9	0.400	0.14	2.4	0.085	21.0	40.0	0.9968	3.43	
25	6.3	0.390	0.16	1.4	0.080	11.0	23.0	0.9955	3.34	
26	7.6	0.410	0.24	1.8	0.080	4.0	11.0	0.9962	3.28	

```
print(df.isna().sum())
```

```
fixed acidity           0
volatile acidity        0
```

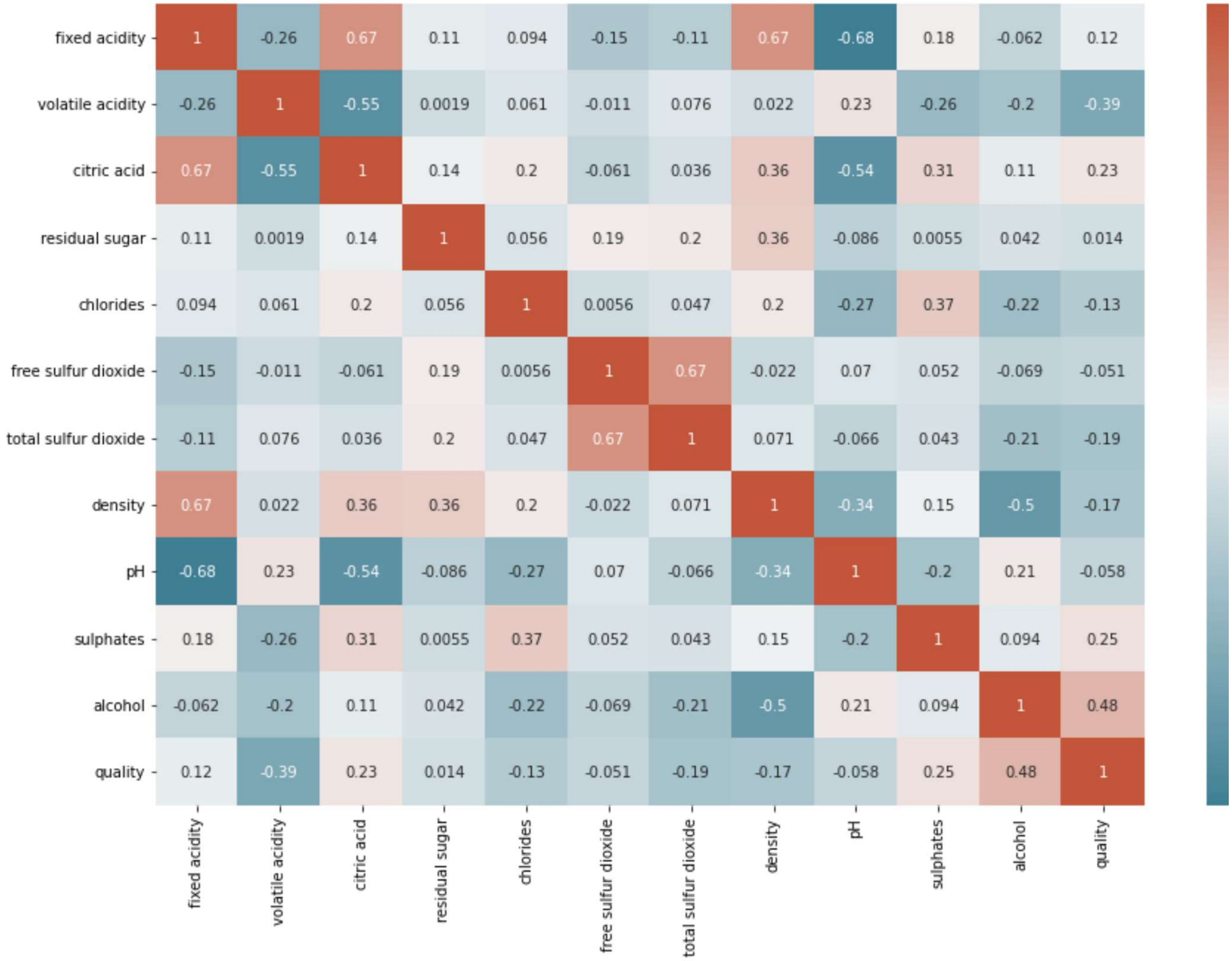
```
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

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




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7ff8ee26d0>
```




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

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

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

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

```
# Splitting the data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25, random_state=0)
```

```
X_test[0]
```

```
array([ 1.4250439 , -0.32301294,  0.81659759, -0.31132282,  1.7753969 ,
        1.06389977,  0.59395426,  0.77027994, -0.91431164,  0.60105502,
        0.35389538])
```

```
X_train
```

```
array([[ 0.04617083,  1.21326812, -0.82661719, ..., -0.78472608,
         0.95513348, -0.77251161],
       [-0.41345352, -0.546472  ,  0.09769112, ...,  0.57592232,
        -0.10710191, -0.86637886],
       [ 0.04617083,  0.17976995, -1.18607043, ..., -0.59034773,
        -1.28736344, -0.77251161],
       ...,
       [-0.24109439,  0.23563472,  0.20039205, ..., -0.13679827,
         0.18796348, -0.86637886],
       [ 2.68901088, -0.32301294,  1.12470036, ..., -0.07200549,
         0.1289504 ,  2.13737311],
       [ 0.85051346,  2.52609011,  0.25174251, ..., -0.39596939,
        -1.05131114, -0.96024611]])
```

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

from sklearn.tree import DecisionTreeClassifier
model1 = DecisionTreeClassifier(random_state=1)
model1.fit(X_train, y_train)
y_pred1 = model1.predict(X_test)
from sklearn.metrics import accuracy_score
print("DecisionTreeClassifier Accuracy: ", accuracy_score(y_test, y_pred1)*100, "%")
```

```
DecisionTreeClassifier Accuracy:  89.75 %
```

```
from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(random_state=1)
model2.fit(X_train, y_train)
y_pred2 = model2.predict(X_test)
from sklearn.metrics import accuracy_score
print("RandomForestClassifier Accuracy: ", accuracy_score(y_test, y_pred2)*100, "%")
```

```
RandomForestClassifier Accuracy:  92.25 %
```

```
from sklearn.svm import SVC
```

```
svc_model = SVC()
svc_model.fit(X_train, y_train)
svc_pred = svc_model.predict(X_test)
from sklearn import metrics
print("Support Vector Machine Classifier (SVM) Accuracy:", accuracy_score(y_test, svc_pred)*1
```

```
Support Vector Machine Classifier (SVM) Accuracy: 91.25 %
```

```
from sklearn.ensemble import AdaBoostClassifier
model3 = AdaBoostClassifier(random_state=1)
model3.fit(X_train, y_train)
y_pred3 = model3.predict(X_test)
from sklearn.metrics import accuracy_score
print("AdaBoostClassifier Accuracy: ", accuracy_score(y_test, y_pred3))
```

```
AdaBoostClassifier Accuracy:  89.0 %
```

```
from sklearn.ensemble import GradientBoostingClassifier
model4 = GradientBoostingClassifier(random_state=1)
model4.fit(X_train, y_train)
y_pred4 = model4.predict(X_test)
from sklearn.metrics import accuracy_score
print("GradientBoostingClassifier Accuracy: ", accuracy_score(y_test, y_pred4)*100, "%")
```

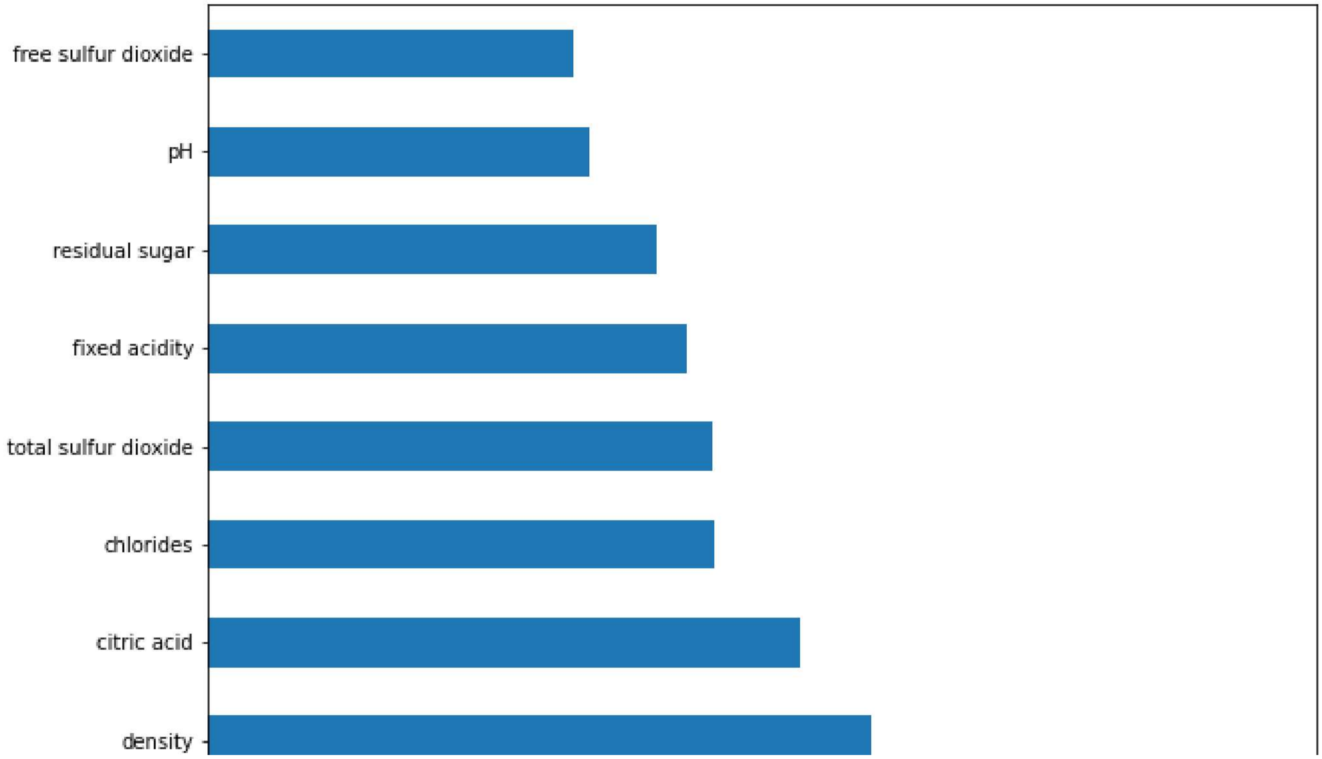
```
GradientBoostingClassifier Accuracy:  89.25 %
```

```
import xgboost as xgb
model5 = xgb.XGBClassifier(random_state=1)
model5.fit(X_train, y_train)
y_pred5 = model5.predict(X_test)
from sklearn.metrics import accuracy_score
print("xgboost Accuracy: ", accuracy_score(y_test, y_pred4)*100, "%")
```

```
xgboost Accuracy:  89.25 %
```

```
feat_importances = pd.Series(model2.feature_importances_, index=X_features.columns)
feat_importances.nlargest(25).plot(kind='barh',figsize=(10,10))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f7fecb21fd0>




```
# Filtering df for only good quality
df_temp = df[df['goodquality']==1]
df_temp.describe()
# 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	c
count	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382.000000	1382
mean	8.236831	0.547022	0.254407	2.512120	0.089281	16.172214	48
std	1.682726	0.176337	0.189665	1.415778	0.049113	10.467685	32
min	4.600000	0.160000	0.000000	0.900000	0.034000	1.000000	6
25%	7.100000	0.420000	0.082500	1.900000	0.071000	8.000000	23
50%	7.800000	0.540000	0.240000	2.200000	0.080000	14.000000	39
75%	9.100000	0.650000	0.400000	2.600000	0.091000	22.000000	65
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	165

```
import joblib
joblib_file = "wine_model.h5"
joblib.dump(model2, joblib_file)
```

```
['wine_model.h5']
```

```
import joblib
joblib_file1 = "wine_model_decission_tree.h5"
joblib.dump(model1, joblib_file1)
```

```
['wine_model_decission_tree.h5']
```

```
model1 = joblib.load(joblib_file1)
model1.predict([[7.8, 0.760, 0.04, 2.3, 0.092, 15.0, 54.0, 0.9970, 3.26, 0.65, 9.8]])[0]
```

```
1
```

```
model1 = joblib.load(joblib_file1)
model1.predict([[6.9, 0.685,  0.00, 2.5,  0.105,  22.0, 37.0, 0.9966, 3.46, 0.57, 10.6]])[0]
```

```
1
```

✓ 0s completed at 4:16 PM