

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings('ignore')

```

```

df = pd.read_csv('/content/winequality-red.csv')
print(df.head())

```

```

      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides \
0              7.4              0.70          0.00              1.9         0.076
1              7.8              0.88          0.00              2.6         0.098
2              7.8              0.76          0.04              2.3         0.092
3             11.2              0.28          0.56              1.9         0.075
4              7.4              0.70          0.00              1.9         0.076

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates \
0                11.0              34.0    0.9978  3.51         0.56
1                25.0              67.0    0.9968  3.20         0.68
2                15.0              54.0    0.9970  3.26         0.65
3                17.0              60.0    0.9980  3.16         0.58
4                11.0              34.0    0.9978  3.51         0.56

      alcohol  quality
0         9.4        5
1         9.8        5
2         9.8        5
3         9.8        6
4         9.4        5

```



```
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().T
```

	count	mean	std	min	25%	50%	75%	max	
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000	
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000	
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000	
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000	
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100	
free sulfur dioxide	1599.0	15.871922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000	

df.isnull().sum()

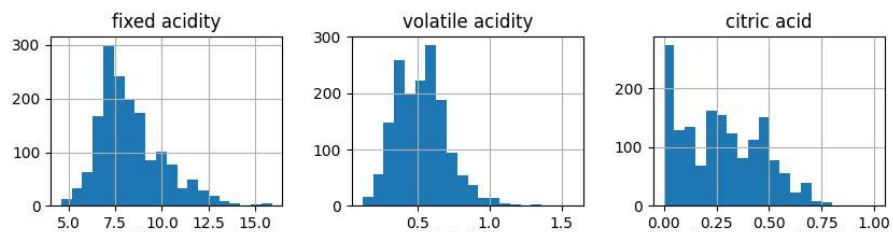
```
fixed acidity      0
volatile acidity   0
citric acid        0
residual sugar     0
chlorides          0
free sulfur dioxide 0
total sulfur dioxide 0
density            0
pH                 0
sulphates          0
alcohol            0
quality            0
dtype: int64
```

```
for col in df.columns:
    if df[col].isnull().sum() > 0:
        df[col] = df[col].fillna(df[col].mean())
```

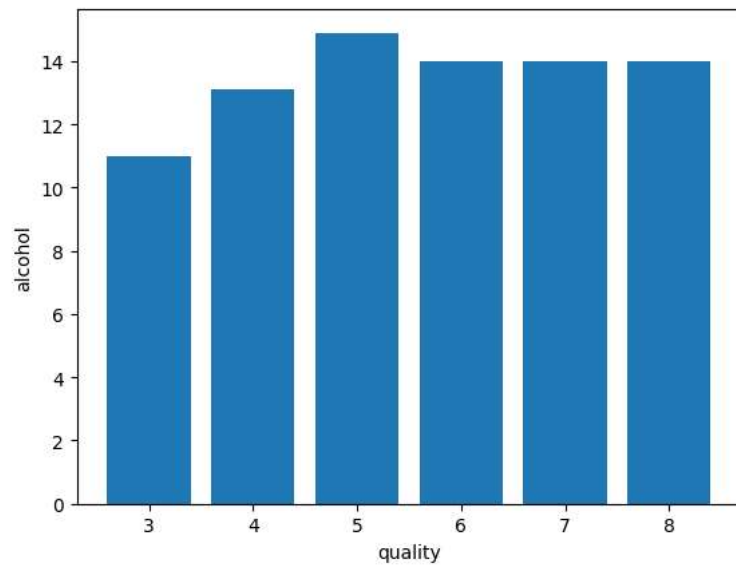
df.isnull().sum().sum()

0

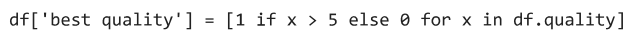
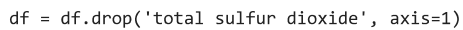
```
df.hist(bins=20, figsize=(10, 10))
plt.show()
```



```
plt.bar(df['quality'], df['alcohol'])
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.show()
```



```
plt.figure(figsize=(12, 12))
sb.heatmap(df.corr() > 0.7, annot=True, cbar=False)
plt.show()
```



```
features = df.drop(['quality', 'best quality'], axis=1)
target = df['best quality']
```

```
xtrain.shape, xtest.shape
```

```
norm = MinMaxScaler()
xtrain = norm.fit_transform(xtrain)
xtest = norm.transform(xtest)
```

```
for i in range(3):
    models[i].fit(xtrain, ytrain)

    print(f'{models[i]} : ')
    print('Training Accuracy : ', metrics.roc_auc_score(ytrain, models[i].predict(xtrain)))
    print('Validation Accuracy : ', metrics.roc_auc_score(
        ytest, models[i].predict(xtest)))
    print()
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
```

```

max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan, monotone_constraints=None,
multi_strategy=None, n_estimators=None, n_jobs=None,
num_parallel_tree=None, random_state=None, ...) :
Training Accuracy : 1.0
Validation Accuracy : 0.8321733317604498

SVC() :
Training Accuracy : 0.7699408577589806
Validation Accuracy : 0.7930675160237505

```

"""Model Evaluation

From the above accuracies we can say that Logistic Regression and SVC()
 classifier performing better on the validation data with less difference between the validation and training data.
 Let's plot the confusion matrix as well for the validation data using the Logistic Regression model."""

'Model Evaluation\nFrom the above accuracies we can say that Logistic Regression and SVC()\nclassifier performing better on the validation
 confusion matrix as well for the validation data using the Logistic Regression model.'

```

print(metrics.classification_report(ytest,
                                   models[1].predict(xtest)))

```

	precision	recall	f1-score	support
0	0.80	0.84	0.82	147
1	0.86	0.82	0.84	173
accuracy			0.83	320
macro avg	0.83	0.83	0.83	320
weighted avg	0.83	0.83	0.83	320