Wine Quality Prediction Using Machine Learning

```
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/winequalityN.csv')
print(df.head())

```

```
    type  fixed acidity  volatile acidity  citric acid  residual sugar  \
0  white            7.0              0.27         0.36            20.7
1  white            6.3              0.30         0.34             1.6
2  white            8.1              0.28         0.40             6.9
3  white            7.2              0.23         0.32             8.5
4  white            7.2              0.23         0.32             8.5

   chlorides  free sulfur dioxide  total sulfur dioxide  density    pH  \
0      0.045                 45.0                 170.0   1.0010  3.00
1      0.049                 14.0                 132.0   0.9940  3.30
2      0.050                 30.0                  97.0   0.9951  3.26
3      0.058                 47.0                 186.0   0.9956  3.19
4      0.058                 47.0                 186.0   0.9956  3.19

   sulphates  alcohol  quality
0       0.45      8.8        6
1       0.49      9.5        6
2       0.44     10.1        6
3       0.40      9.9        6
4       0.40      9.9        6
```

```
df.info()

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   type                  6497 non-null   object
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

```
df.describe().T

```

	count	mean	std	min	
fixed acidity	6487.0	7.216579	1.296750	3.80000	
volatile acidity	6489.0	0.339691	0.164649	0.08000	
citric acid	6494.0	0.318722	0.145265	0.00000	
residual sugar	6495.0	5.444326	4.758125	0.60000	
chlorides	6495.0	0.056042	0.035036	0.00900	
free sulfur dioxide	6497.0	30.525319	17.749400	1.00000	1
total sulfur dioxide	6497.0	115.744574	56.521855	6.00000	7
density	6497.0	0.994697	0.002999	0.98711	

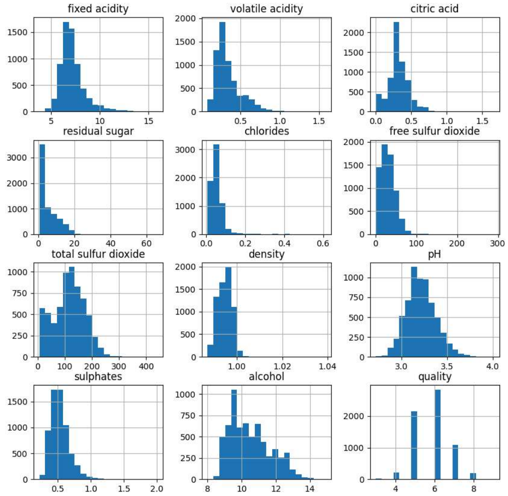
```
df.isnull().sum()

```

```
type                    0
fixed acidity          10
volatile acidity        8
citric acid             3
residual sugar          2
chlorides               2
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      9
sulphates               4
alcohol                 0
quality                 0
dtype: int64
```

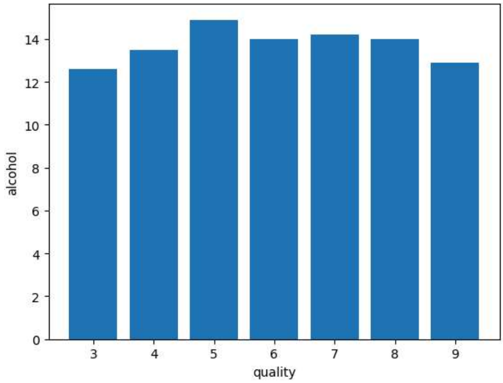
```
for col in df.columns:
 if df[col].isnull().sum() > 0:
     df[col] = df[col].fillna(df[col].mean())

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()

```



```
df['best quality'] = [1 if x > 5 else 0 for x in df.quality]

```

```
df.replace({'white': 1, 'red': 0}, inplace=True)

```

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

xtrain, xtest, ytrain, ytest = train_test_split(
    features, target, test_size=0.2, random_state=40)

xtrain.shape, xtest.shape

```

```
((5197, 11), (1300, 11))
```

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

```

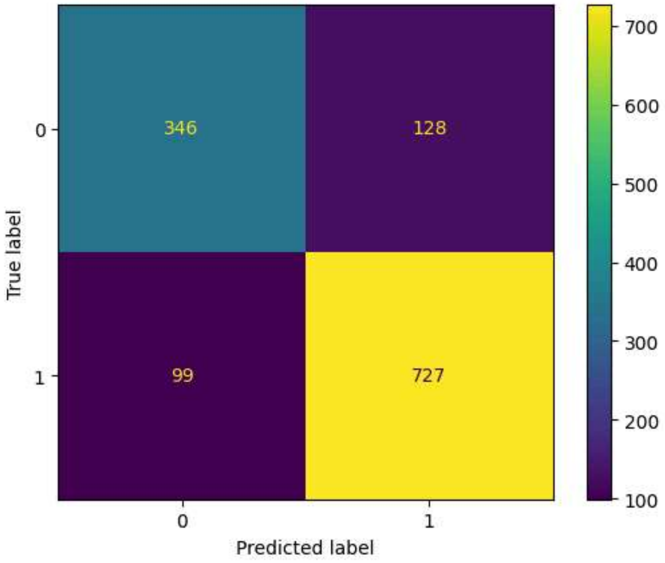
```
models = [LogisticRegression(), XGBClassifier(), SVC(kernel='rbf')]

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

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

```

```
LogisticRegression() : 
Training Accuracy :  0.7019709565048414
Validation Accuracy :  0.6937888865050418

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 :  0.9735567052182403
Validation Accuracy :  0.8050515421787681

SVC() : 
Training Accuracy :  0.7069199304892986
Validation Accuracy :  0.695796426272719
```

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

```

```
              precision    recall  f1-score   support

           0       0.78      0.73      0.75       474
           1       0.85      0.88      0.86       826

    accuracy                           0.83      1300
   macro avg       0.81      0.81      0.81      1300
weighted avg       0.82      0.83      0.82      1300
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# ... (rest of your code)

cm = confusion_matrix(ytest, models[1].predict(xtest))
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```


346
128
99
727
0
1
True label
0
1
Predicted label
700
600
500
400
300
200
100