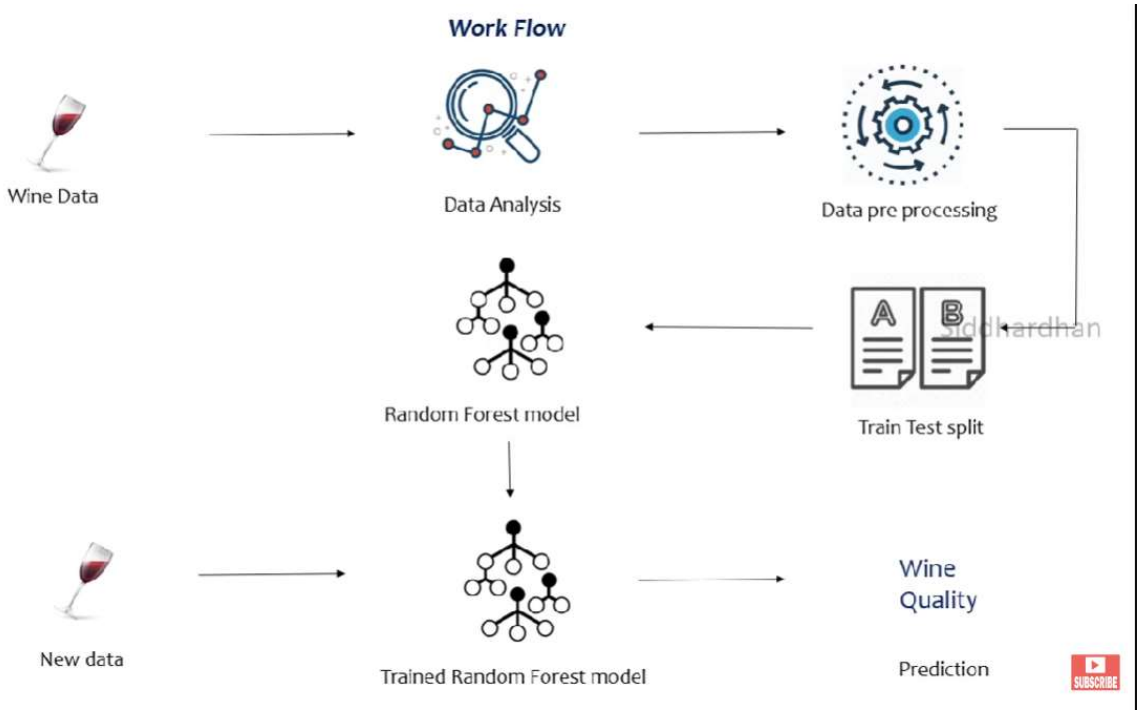
Wine Quality Predictor:



To check no. of rows and columns in a dataset:

train.shape

To print first 5 rows of dataframe

train.head()

To check for null values in each column

train.isnull().sum()

Note: both are functions and not attributes

Just in case there would be null values, we would have to deal with replacing all null values possible before insertion data into our model.

Once we are through that our data is free from NaN values, we start data visualization and statistical analysis.

Statistical Analysis of the data:

train.describe()

To find how many types of qualities are available and what is the count of each type of value_counts.

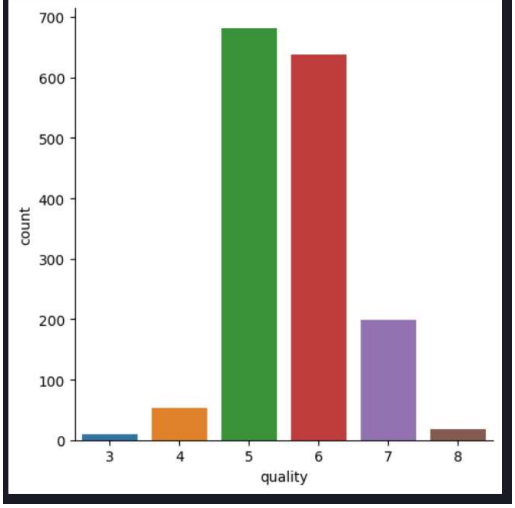
```
train['quality'].value_counts()
```
✓ 0.0s

```
quality
5    681
6    638
7    199
4     53
8     18
3     10
Name: count, dtype: int64
```

catplot:

```
sns.catplot(x='quality', data=train, kind='count')
```
✓ 0.6s

```
<seaborn.axisgrid.FacetGrid at 0x16aada87a60>
```



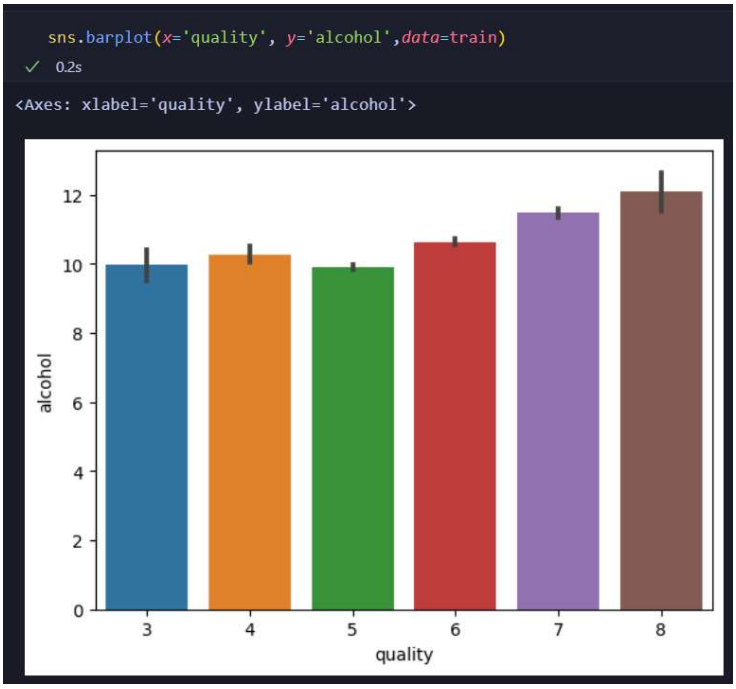
Note: Catplot was used when we were plotting the counts chart of quality.

To compare between two columns, (features of dataframe) use barplot.

Features of red wine plotted vs quality.

```
sns.barplot(x='quality', y='alcohol',data=train)
```
✓ 0.2s

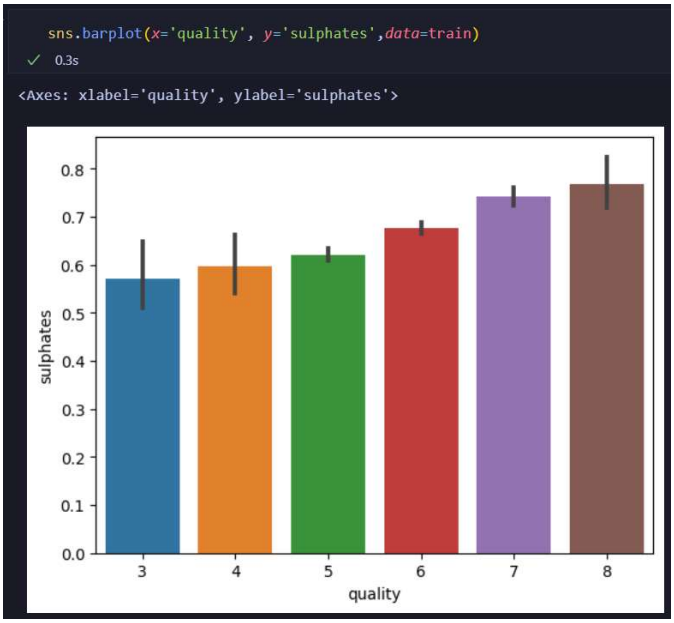
```
<Axes: xlabel='quality', ylabel='alcohol'>
```



quality directly proportional to alcohol

```
sns.barplot(x='quality', y='sulphates',data=train)
```
✓ 0.3s

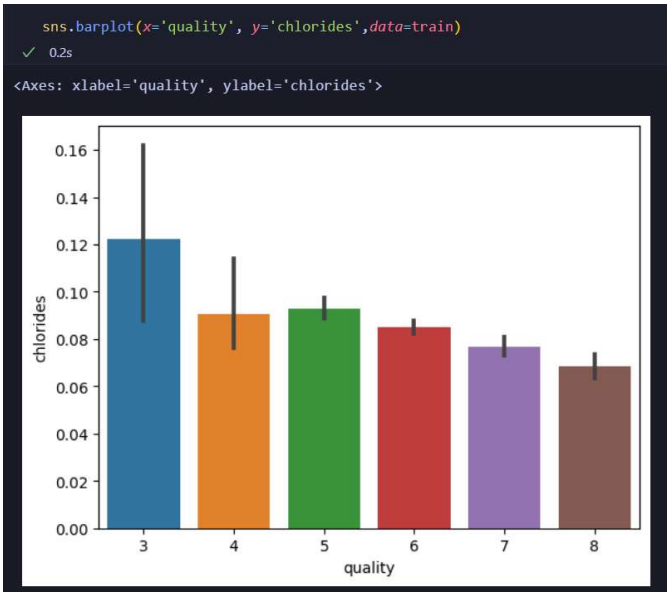
```
<Axes: xlabel='quality', ylabel='sulphates'>
```



quality directly proportional to sulphates

```
sns.barplot(x='quality', y='chlorides',data=train)
```
✓ 0.2s

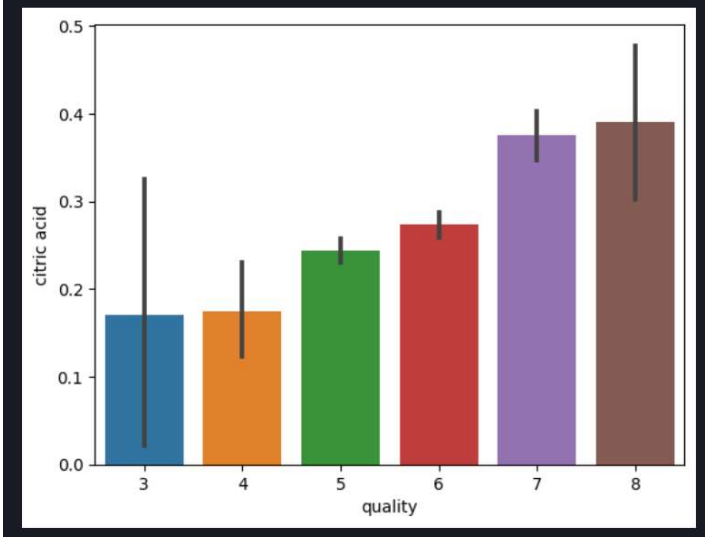
```
<Axes: xlabel='quality', ylabel='chlorides'>
```



quality inversely proportional to chlorides.

```
sns.barplot(x='quality', y='citric acid',data=train)
```
✓ 0.2s

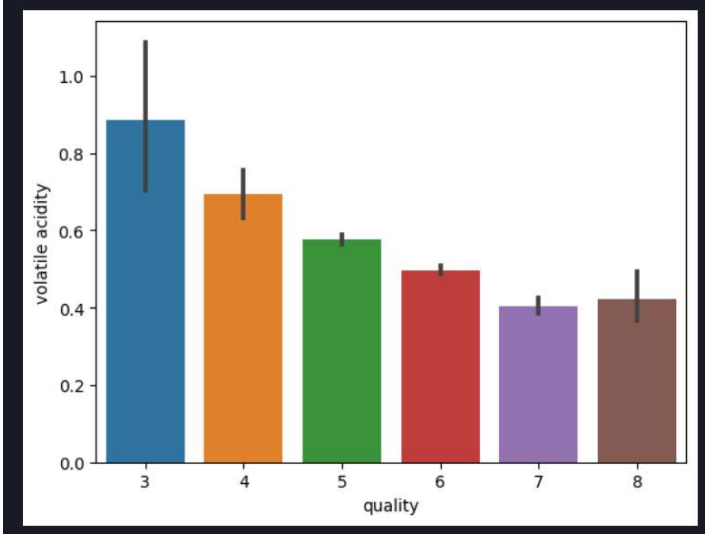
```
<Axes: xlabel='quality', ylabel='citric acid'>
```



quality directly proportional to citric acid

```
sns.barplot(x='quality', y='volatile acidity',data=train)
```
✓ 0.2s

```
<Axes: xlabel='quality', ylabel='volatile acidity'>
```



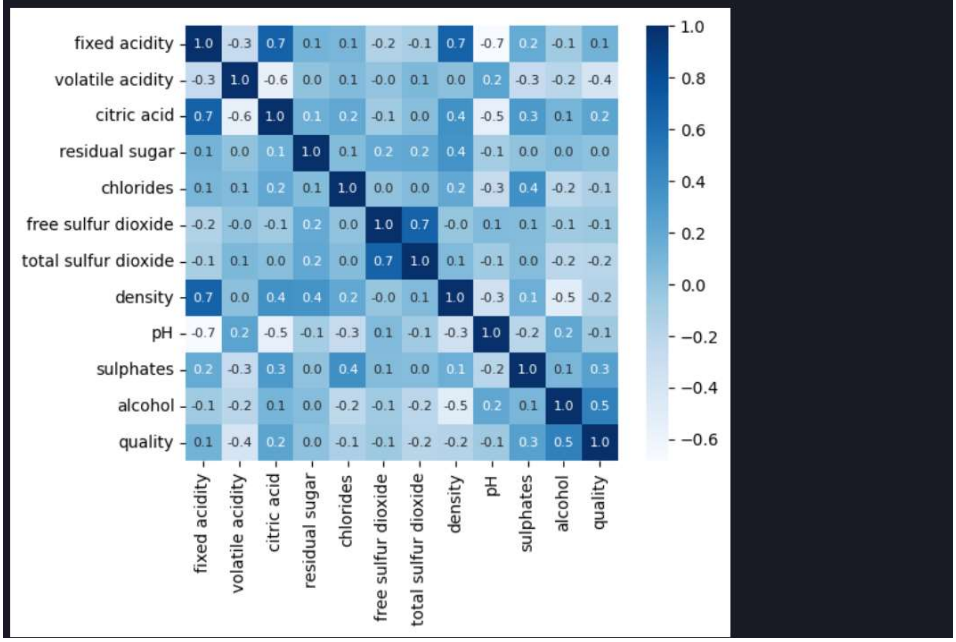
quality inversely proportional to volatile acidity.

The data visualization helps us in finding how different features are (if they are) related to quality of red wine.

Now we check the correlation between different features and quality.

```
sns.heatmap(correlation,cbar=True,square=True,fmt='.1f',annot=True,annot_kws={'size':8},cmap='Blues')
```
✓ 0.7s

```
<Axes: >
```



How to read this heatmap?

1. The darker the shade in the heatmap, the more directly correlated the values are, the lighter the value in the heatmap, the more inversely correlated the values are.
2. The centre diagonal is full of darkest values (correlation=1.0), because they are values correlated to themselves, which is 100%. Ignore this diagonal.
3. We are concerned with the parametre “quality”, so refer the row/column of quality.
4. Check the row column, the alcohol has a value of 0.5 (highest +ve) indicating that they are highly correlated, while volatile acidity has a value of -0.4 (highest -ve) indicating that they are highly inversely correlated.

Data Preprocessing:

To put the training data into the ML model, we need to separate the quality column from the training data, put them into separate variables.

Label Binarization:

~ Instead of having multiple values for quality we classify quality into two major groups, good or bad.

```
Y = train['quality'].apply(lambda y_value:1 if y_value>=7 else 0)
```

For these operations, we use lambda function in python

```
Y.value_counts()
✓ 0.0s

quality
0    1382
1     217
Name: count, dtype: int64
```

We can import train_test_split from sklearn library

Sklearn dependencies:

```
import sklearn
✓ 0.6s

from sklearn.model_selection import train_test_split
✓ 0.0s

from sklearn.ensemble import RandomForestClassifier
✓ 0.0s

from sklearn.metrics import accuracy_score
✓ 0.0s
```

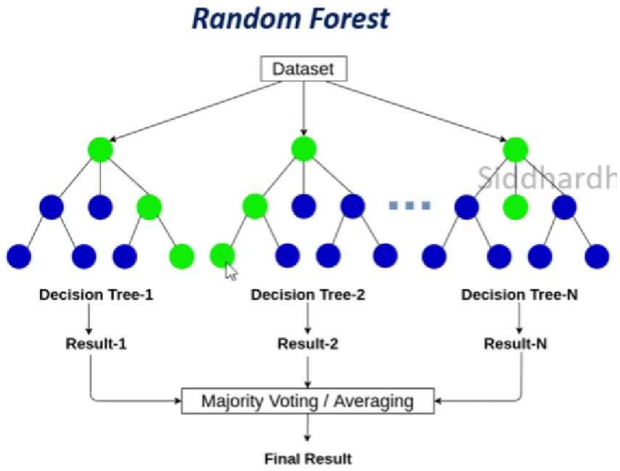
```
X_train, X_test, Y_train, Y_test =train_test_split(X,Y,test_size=0.2,random_state=2)
```

general convention:

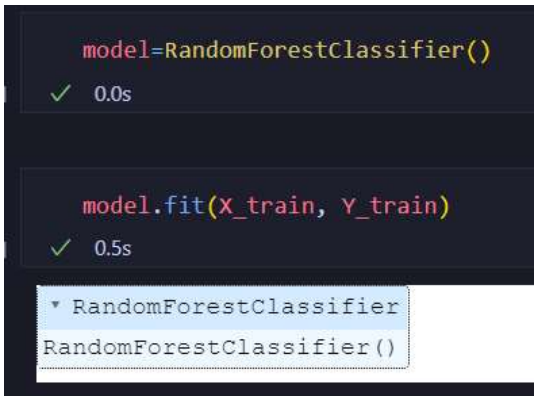
- test_size: test data is 10/20% of training data.
- random_state: The train and test sets directly affect the model's performance score. Because we get different train and test sets with different integer values for random_state in the train_test_split() function, the value of the random state hyperparameter indirectly affects the model's performance score.

Model we shall be using: Random Forest Classifier

This model is an ensemble model, meaning this model uses a combination of two or three models. (In this case, ensemble model for decision tree)

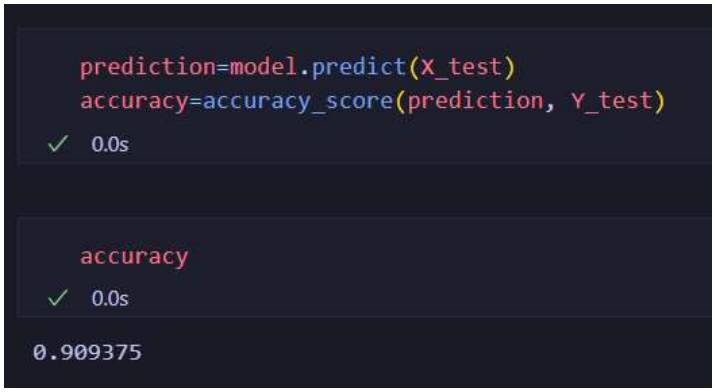


Model Training:



Model Evaluation

Initially we trained our model using train data and now we let the model predict values based on X_test values and compare the predictions with actual outputs to get accuracy/model evaluation.



Now we shall

Build a predicting system

```
input_data=(7.3,0.65,0.0,1.2,0.065,15.0,21.0,0.9946,3.39,0.47,10.0)
arr=np.asarray(input_data)
reshaped=arr.reshape(1,-1)
prediction1=model.predict(reshaped)
if(prediction1[0]==1):
    print('Good Wine!')
else:
    print('Bad Wine!')
```

✓ 0.0s

```
Good Wine!
```

Note:

While fitting the model, use .values to avoid a dataframe label warning!

```
model.fit(X_train.values, Y_train.values)
```

✓ 0.4s

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
prediction=model.predict(X_test.values)
accuracy=accuracy_score(prediction, Y_test)
```

✓ 0.0s