```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.model_selection import cross_val_score
6 from sklearn.metrics import accuracy_score
```

```
1 df_red=pd.read_csv(r'https://archive.ics.uci.edu/ml/machine-learning-databases/v
2 df_white=pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases,
3 display(df_red)
4 display(df_white)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780

```
# dropping duplicates
df_red = df_red.drop_duplicates()
df_white = df_white.drop_duplicates()
```

2	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800

```
#Checking variable inforamtion
display(df_red.info())
display(df_white.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1359 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1359 non-null   float64
 1   volatile acidity      1359 non-null   float64
 2   citric acid           1359 non-null   float64
 3   residual sugar        1359 non-null   float64
 4   chlorides             1359 non-null   float64
 5   free sulfur dioxide   1359 non-null   float64
 6   total sulfur dioxide  1359 non-null   float64
 7   density               1359 non-null   float64
 8   pH                    1359 non-null   float64
 9   sulphates             1359 non-null   float64
 10  alcohol               1359 non-null   float64
 11  quality               1359 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 138.0 KB
None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3961 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         3961 non-null   float64
 1   volatile acidity      3961 non-null   float64
 2   citric acid           3961 non-null   float64
 3   residual sugar        3961 non-null   float64
 4   chlorides             3961 non-null   float64
 5   free sulfur dioxide   3961 non-null   float64
 6   total sulfur dioxide  3961 non-null   float64
 7   density               3961 non-null   float64
 8   pH                    3961 non-null   float64
 9   sulphates             3961 non-null   float64
 10  alcohol               3961 non-null   float64
 11  quality               3961 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 402.3 KB
None
```

```
# For red wine calculating unique and null values
for col in df_red.columns.values:
```

```
  list_vals = pd.unique(df_red[col]) #list of unique values
  print(col + ' has ' + str(len(list_vals))  +' unique values, and ' + str(df_r
  if len(list_vals) < 10:
    list_str = ''
    for n in range(0, len(list_vals)):
      list_str = list_str + str(list_vals[n]) + ', '
    print(' These are: '+list_str[0:len(list_str) - 2])

# For white wine calculating unique and null values
for col in df_white.columns.values:
  list_vals = pd.unique(df_white[col]) #list of unique values
  print(col + ' has ' + str(len(list_vals))  +' unique values, and ' + str(df_wh
  if len(list_vals) < 10:
    list_str = ''
    for n in range(0, len(list_vals)):
      list_str = list_str + str(list_vals[n]) + ', '
    print(' These are: '+list_str[0:len(list_str) - 2])
```

```
fixed acidity has 96 unique values, and 0 null entries
volatile acidity has 143 unique values, and 0 null entries
citric acid has 80 unique values, and 0 null entries
residual sugar has 91 unique values, and 0 null entries
chlorides has 153 unique values, and 0 null entries
free sulfur dioxide has 60 unique values, and 0 null entries
total sulfur dioxide has 144 unique values, and 0 null entries
density has 436 unique values, and 0 null entries
pH has 89 unique values, and 0 null entries
sulphates has 96 unique values, and 0 null entries
alcohol has 65 unique values, and 0 null entries
quality has 6 unique values, and 0 null entries
 These are: 5, 6, 7, 4, 8, 3
fixed acidity has 68 unique values, and 0 null entries
volatile acidity has 125 unique values, and 0 null entries
citric acid has 87 unique values, and 0 null entries
residual sugar has 310 unique values, and 0 null entries
chlorides has 160 unique values, and 0 null entries
free sulfur dioxide has 132 unique values, and 0 null entries
total sulfur dioxide has 251 unique values, and 0 null entries
density has 890 unique values, and 0 null entries
pH has 103 unique values, and 0 null entries
sulphates has 79 unique values, and 0 null entries
alcohol has 103 unique values, and 0 null entries
quality has 7 unique values, and 0 null entries
 These are: 6, 5, 7, 8, 4, 3, 9
```

Observations:

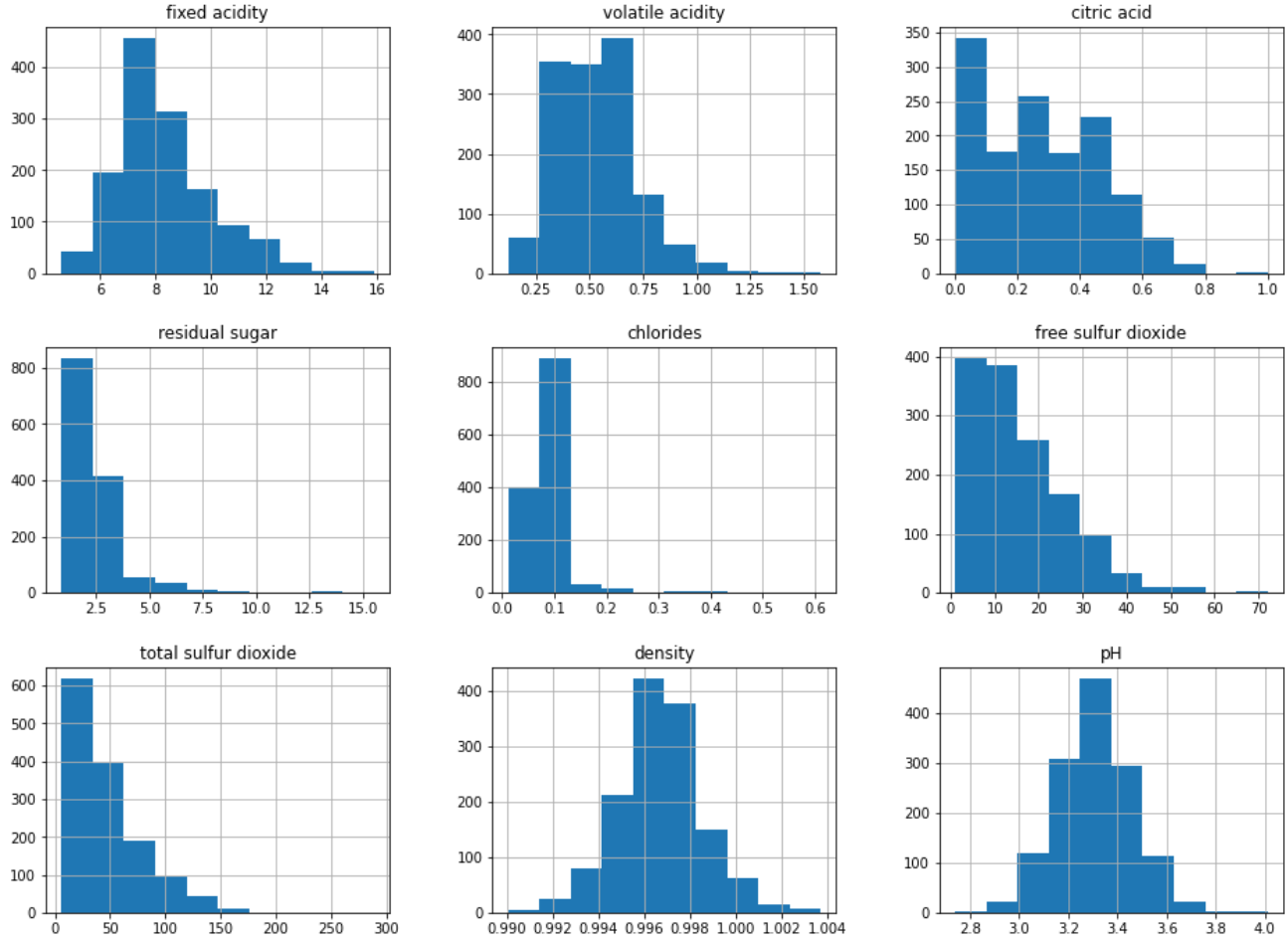
1: No column has null entry.

```
#Visualisation of similarity between two variable in wine data by checking corre
df_red.corr()
```

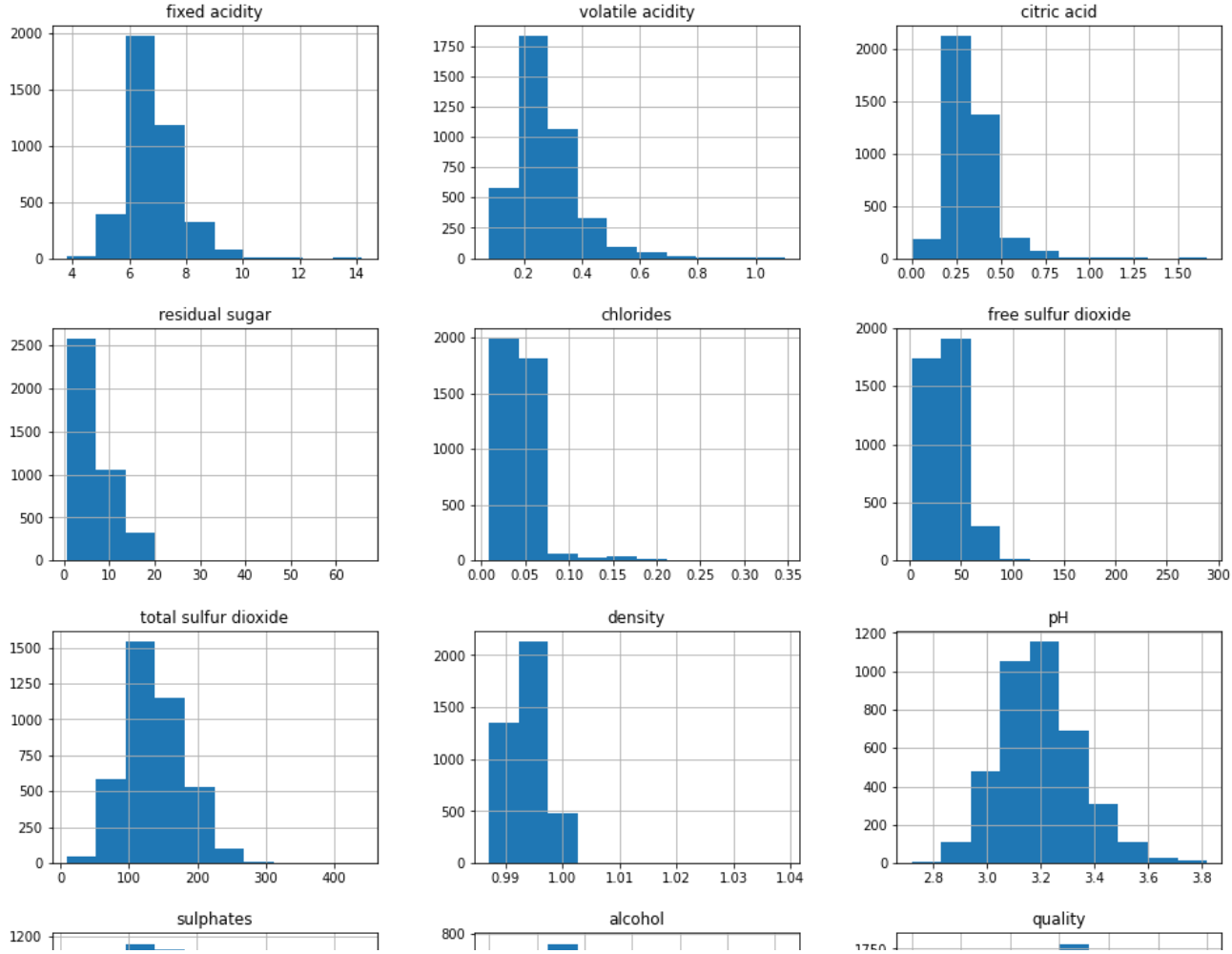
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
fixed acidity	1.000000	-0.255124	0.667437	0.111025	0.085886	-0.140580	-0.103777
volatile acidity	-0.255124	1.000000	-0.551248	-0.002449	0.055154	-0.020945	0.071701
citric acid	0.667437	-0.551248	1.000000	0.143892	0.210195	-0.048004	0.047358
residual sugar	0.111025	-0.002449	0.143892	1.000000	0.026656	0.160527	0.201038
chlorides	0.085886	0.055154	0.210195	0.026656	1.000000	0.000749	0.045773
free sulfur dioxide	-0.140580	-0.020945	-0.048004	0.160527	0.000749	1.000000	0.667246
total sulfur dioxide	-0.103777	0.071701	0.047358	0.201038	0.045773	0.667246	1.000000
density	0.670195	0.023943	0.357962	0.324522	0.193592	-0.018071	0.078141

```
#Visualisation of data through histogams
#Checking if the data is skewed
df_red.hist(bins=10, figsize=(16,16))
plt.figure(1)
plt.show()

```




```
#Visualisation of white wine data through histogams
#Checking if the data is skewed
df_white.hist(bins=10, figsize=(16,16))
plt.figure(1)
plt.show()
```

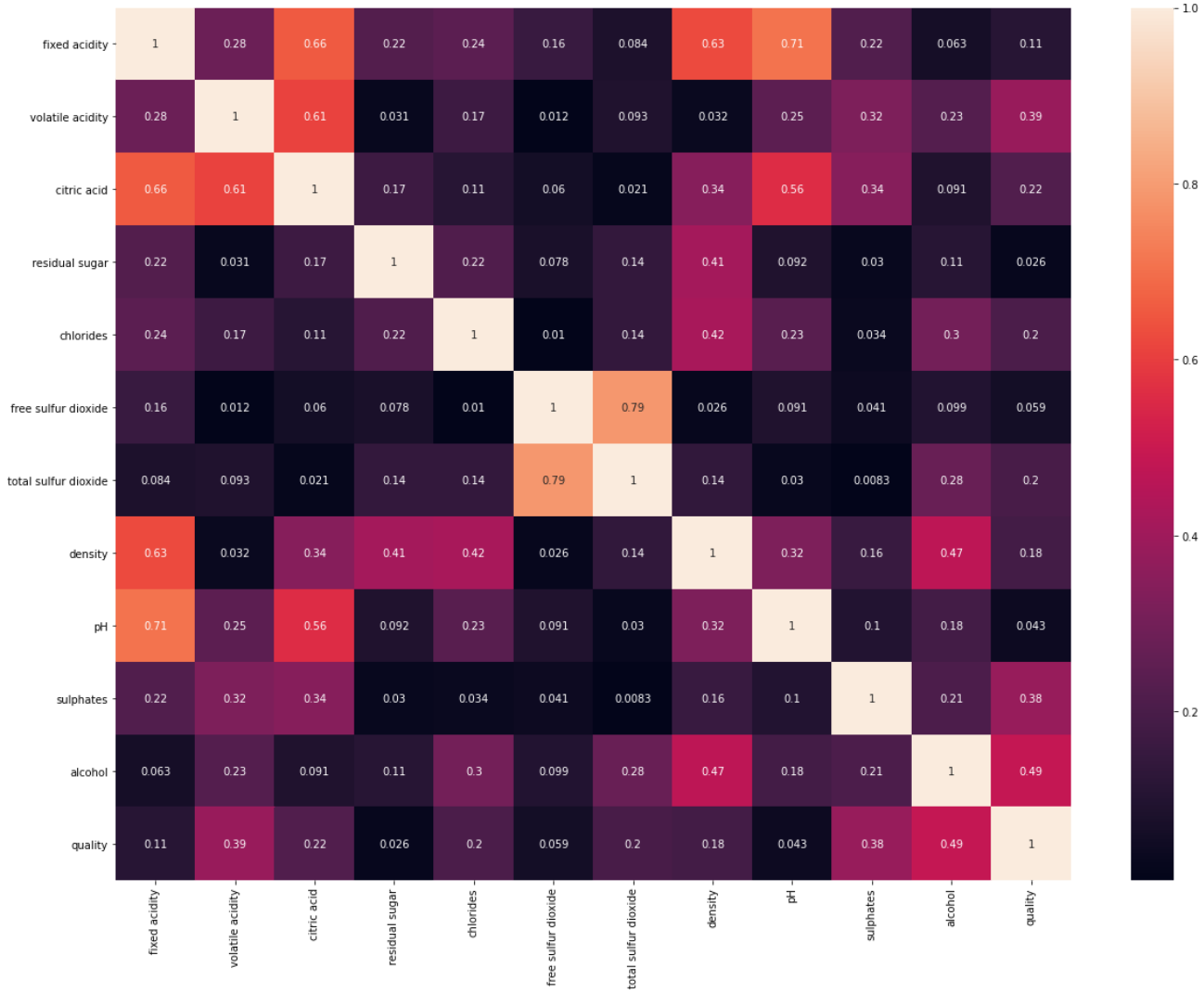


Obs:

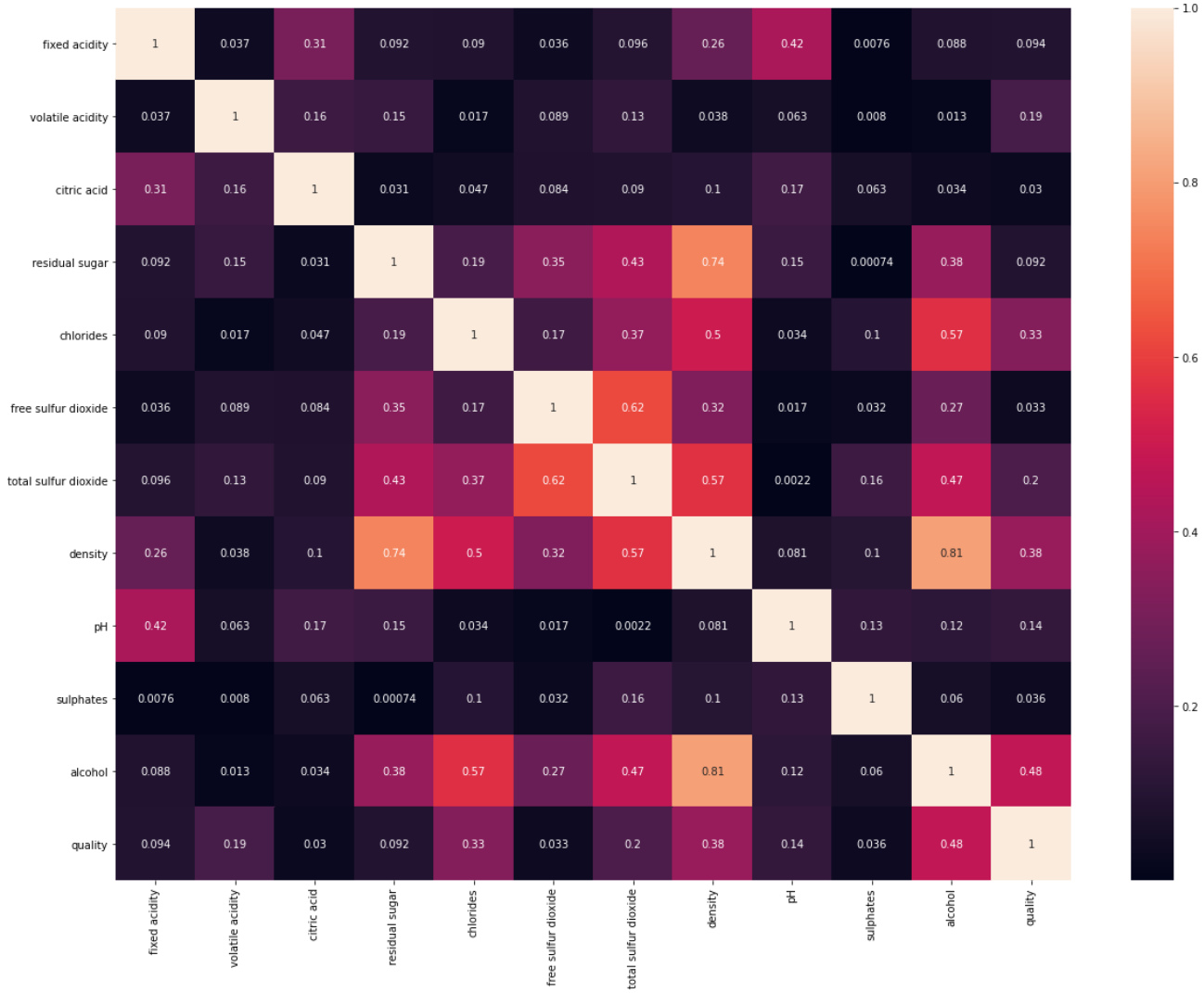
'acohol', 'sulphates','residual sugar','chlorides', 'free sulphur dioxide', 'total sulphur dioxide' hist seems to be positively skewed.

```
# visualization of correlation by heatmap for red wine

corr = df_red.corr(method = 'spearman')
fig, ax = plt.subplots(figsize = (20,15))
sns.heatmap(abs(corr), annot = True)
plt.show()
```

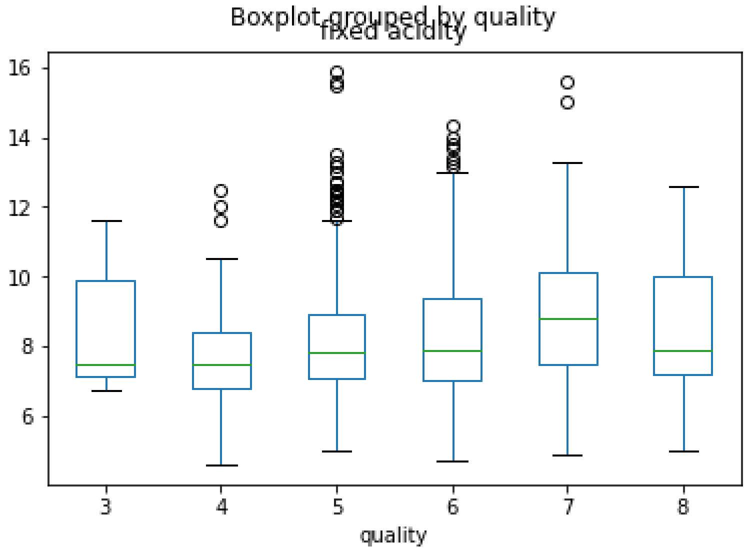


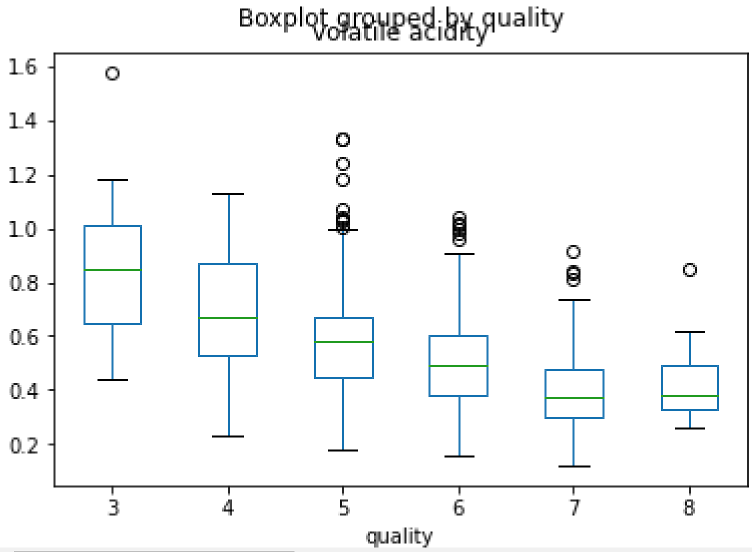

```
1 # visualization of correlation by heatmap for White wine
2
3 corr = df_white.corr(method = 'spearman')
4 fig, ax = plt.subplots(figsize = (20,15))
5 sns.heatmap(abs(corr), annot = True)
6 plt.show()
```




```
1 # boxplot for red wine
2 for col in df_red.columns.values:
3   df_red.boxplot(by = 'quality', column = col, grid = False)
```

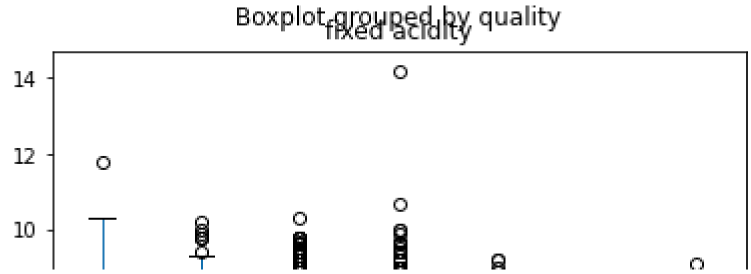
```
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: VisibleDe
  return array(a, dtype, copy=False, order=order)
```





```
# boxplot for white wine
for col in df_white.columns.values:
  df_white.boxplot(by = 'quality', column = col, grid = False)
```

```
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.6/dist-packages/numpy/core/_asarray.py:83: Visible
  return array(a, dtype, copy=False, order=order)
```



Preparation of data

```
#x = df_red.drop('free sulfur dioxide',axis=1)
```

```
#forred wine data and label
x = df_red.drop('quality',axis=1) # keeping all the coumns except quality, inpu
y = df_red['quality']  #'quality' as target var
```

```
a = df_white.drop('quality',axis=1) # keeping all the coumns except quality, in
b = df_white['quality']  #'quality' as Label
```

```
from sklearn.model_selection import train_test_split
```

```
#dividing the data into two parts train and test data (x,y)for red wine, (a,b)f
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_
a_train, a_test, b_train, b_test = train_test_split(a, b, test_size=0.3, random_
```

```
#scaling of data
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

#for white wine
a_train = sc.fit_transform(a_train)
a_test = sc.fit_transform(a_test)
```

LASSO L1 Regression

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.metrics import accuracy_score
lasso_reg=Lasso()

#hyperparameter tuning and cross validation using grid

parameter_grid_lasso=({'alpha':[0.001,0.01,0.1,1,10,100,1000]}) #setting dictio

grid_search = GridSearchCV(lasso_reg, parameter_grid_lasso, cv=5,scoring='neg_m
white_grid_search = GridSearchCV(lasso_reg, parameter_grid_lasso, cv=5,scoring=

grid_search.fit(x_train,y_train) #training and fitting data for red wine

white_grid_search.fit(a_train,b_train)   #training and fitting data for white w

print('')
print('Best parameter for red wine is',grid_search.best_params_) #for red wine
print('')
print('')
print('Best parameter for white wine is',white_grid_search.best_params_) #for wh
print('')


```

```
Best parameter for red wine is {'alpha': 0.01}


Best parameter for white wine is {'alpha': 0.01}
```

Testing the model

```
#for red wine testing
#Lasso best model metrics
lasso_final_model=grid_search.best_estimator_
lasso_final_prediction=lasso_final_model.predict(x_test)
```

```
5
6 mse_lasso = mean_squared_error(y_test,lasso_final_prediction)
7 rmse = np.sqrt(mse_lasso)
8 print('Tesying parameters for Red wine are')
9 print("\nLasso Regression RMSE:",rmse)
10 MAE=mean_absolute_error(y_test, lasso_final_prediction)
11 print("\nLasso Regression MAE:",MAE)
12 print("\nLasso Regression MSE:",mse_lasso)
13
14 lasso_scores = cross_val_score(lasso_reg, x_train, y_train,scoring="neg_mean_squ
15 print('\nLasso CV score is ',lasso_scores.mean())
16
17 print('\ncoefficients of bestestimator are ',lasso_final_model.coef_)
```

```
Tesying parameters for Red wine are

Lasso Regression RMSE: 0.669583214580127

Lasso Regression MAE: 0.5225142886509236

Lasso Regression MSE: 0.44834168124745644

Lasso CV score is  -0.6738527102028826

coefficients of bestestimator are  [-0.         -0.18822622 -0.00573779  0.00
 -0.04315893 -0.01305741 -0.0597066   0.15821998  0.30976761]
```

```
1 #Testing the model for white wine
2 lasso_final_model=white_grid_search.best_estimator_
3 lasso_final_prediction=lasso_final_model.predict(a_test)
4
5 mse_lasso = mean_squared_error(b_test,lasso_final_prediction)
6 rmse = np.sqrt(mse_lasso)
7
8 print('Tesying parameters for white wine are')
9 print("\nLasso Regression RMSE:",rmse)
10 MAE=mean_absolute_error(b_test, lasso_final_prediction)
11 print("\nLasso Regression MAE:",MAE)
12 print("\nLasso Regression MSE:",mse_lasso)
13
14 lasso_scores = cross_val_score(lasso_reg, a_train, b_train,scoring="neg_mean_squ
15 print('\nLasso CV score is ',lasso_scores.mean())
16
17 print('\ncoefficients of bestestimator are ',lasso_final_model.coef_)
```

```
Tesying parameters for white wine are

Lasso Regression RMSE: 0.7399127620760989

Lasso Regression MAE: 0.5718953963489289

Lasso Regression MSE: 0.5474708954830817

Lasso CV score is  -0.8054583580436189
```

```
coefficients of bestestimator are  [-0.01099256 -0.16203032  0.01450484  0.14
 -0.01935909 -0.13611277  0.04855118  0.05223789  0.38198165]
```

Reducing the value of alpha reduces the error of the model. The parameter α is used for feature reduction, estimating the sparse coefficients. α = 0 is similar to linear regression Calculate errors

Testing the red dataset with white model or vice versa: Not able to get the result as both are having different dimensions.

```
1
```

L1 L2 ElasticNet

training and testing for Red wine of L1L2 Enet

```
#rigorous training and testing for red wine
from sklearn.linear_model import ElasticNet
Enet=ElasticNet()
enet_param_grid = [{'alpha':[0.001, 0.01, 0.1, 1, 10, 100, 1000],'l1_ratio':[0.
En_param_grid_search = GridSearchCV(Enet, enet_param_grid, cv=5, scoring='neg_m
En_param_grid_search.fit(x_train,y_train)
print('')
print('Best parameter is',En_param_grid_search.best_params_)
print('')

#ENET best model metrics
Enet_final_model=En_param_grid_search.best_estimator_
Enet_final_prediction=Enet_final_model.predict(x_test)

mse_Enet = mean_squared_error(y_test,Enet_final_prediction)
rmse = np.sqrt(mse_Enet)
print('Tesying parameters for RED wine are')
print("\nENET Regression RMSE:",rmse)
MAE=mean_absolute_error(y_test, Enet_final_prediction)
print("\nENET Regression MAE:",MAE)
print("\nENET Regression MSE:",mse_Enet)

Enet_scores = cross_val_score(Enet, x_train, y_train,scoring="neg_mean_squared_
print('\nENET  coefficients of bestestimator are ',Enet_final_model.coef_)
```

```
Best parameter is {'alpha': 0.01, 'l1_ratio': 0.7}

Tesying parameters for RED wine are

ENET Regression RMSE: 0.6701157828877404

ENET Regression MAE: 0.5226655358624054
```

```
ENET Regression MSE: 0.4490551624752493

ENET  coefficients of bestestimator are  [-0.         -0.19081555 -0.01430124
 -0.05173179 -0.01772548 -0.06902841  0.16419338  0.30841417]
```

training and testing for **White** wine of L1L2 Enet

```
#training and testing of model for white wine
enet_param_grid = [{'alpha':[0.001, 0.01, 0.1, 1, 10, 100, 1000],'l1_ratio':[0.
white_En_param_grid_search = GridSearchCV(Enet, enet_param_grid, cv=5, scoring=
white_En_param_grid_search.fit(a_train,b_train)

print('Tesying parameters for white wine are')
print('')
print('Best parameter for white wine data is',white_En_param_grid_search.best_pa
print('')

#ENET best model metrics
Enet_final_model=white_En_param_grid_search.best_estimator_
Enet_final_prediction=Enet_final_model.predict(a_test)

mse_Enet = mean_squared_error(b_test,Enet_final_prediction)
rmse = np.sqrt(mse_Enet)

print("\nENET Regression RMSE:",rmse)
MAE=mean_absolute_error(b_test, Enet_final_prediction)
print("\nENET Regression MAE:",MAE)
print("\nENET Regression MSE:",mse_Enet)

Enet_scores = cross_val_score(Enet, a_train, b_train,scoring="neg_mean_squared_
print('\nENET  coefficients of bestestimator are ',Enet_final_model.coef_)
```

```
Tesying parameters for white wine are

Best parameter for white wine data is {'alpha': 0.01, 'l1_ratio': 0.1}


ENET Regression RMSE: 0.7377310817794659

ENET Regression MAE: 0.5697110989166195

ENET Regression MSE: 0.544247149023501

ENET  coefficients of bestestimator are  [ 0.01087688 -0.16024106  0.03022681
 -0.03806477 -0.27439823  0.08116125  0.06891974  0.31931421]
```

```
1
```

Random Forest

Training and Testing of RED Wine DATA

```
#training and testing of model for RED wine
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
forest_reg = RandomForestRegressor()

#parameter grid
forest_param_grid = [{'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]}

grid_search = GridSearchCV(forest_reg, forest_param_grid, cv=5,scoring='neg_mean

grid_search.fit(x_train,y_train)
grid_search.best_params_
print('The parameters for fitting and testing of RED wine are')
print('Best parameter is',grid_search.best_params_)
print('')

#ENET best model metrics
forest_final_model=grid_search.best_estimator_
forest_final_prediction=forest_final_model.predict(x_test)

mse_forest = mean_squared_error(y_test,forest_final_prediction)
rmse = np.sqrt(mse_forest)

print("\nRandomForest Regression RMSE:",rmse)
MAE=mean_absolute_error(y_test, forest_final_prediction)
print("\nRandomForest Regression MAE:",MAE)
print("\nRandomForest Regression MSE:",mse_forest)

forest_scores = cross_val_score(forest_reg, x_train, y_train,scoring="neg_mean_s
print('\nRF cross validation score is',forest_scores.mean())
```

```
The parameters for fitting and testing of RED wine are
Best parameter is {'max_features': 4, 'n_estimators': 30}


RandomForest Regression RMSE: 0.6583653917899648

RandomForest Regression MAE: 0.5110294117647058

RandomForest Regression MSE: 0.4334449891067538

RF cross validation score is -0.4240906907894736
```

```
1
```

Training and TESTING for WHITE wine for RANDOM Forest model

```
#training and testing of model for White wine

white_grid_search = GridSearchCV(forest_reg, forest_param_grid, cv=5,scoring='n

white_grid_search.fit(a_train,b_train)
white_grid_search.best_params_
print('\nThe parameters for fitting and testing of White wine are')
print('\nBest parameter is',grid_search.best_params_)
print('')

#ENET best model metrics
forest_final_model=white_grid_search.best_estimator_
forest_final_prediction=forest_final_model.predict(a_test)

mse_forest = mean_squared_error(b_test,forest_final_prediction)
rmse = np.sqrt(mse_forest)

print("\nRandomForest Regression RMSE:",rmse)
MAE=mean_absolute_error(b_test, forest_final_prediction)
print("\nRandomForest Regression MAE:",MAE)
print("\nRandomForest Regression MSE:",mse_forest)

forest_scores = cross_val_score(forest_reg, a_train, b_train,scoring="neg_mean_s
print('\nRF cross validation score is',forest_scores.mean())
```

```
The parameters for fitting and testing of White wine are

Best parameter is {'max_features': 4, 'n_estimators': 30}


RandomForest Regression RMSE: 0.7093709726929592

RandomForest Regression MAE: 0.5443790299971966

RandomForest Regression MSE: 0.5032071768993551

RF cross validation score is -0.5166170032205282
```

The parameter n estimator in RFR is used to decide the number of trees in the regressor to be used for modeling the data.

```
1
```

SVR

Training and testing of data for RED wine model

```
#Support vector regression training and testing for RED Wine
from sklearn.svm import SVR
#parameters grid code courtesy stackexchange
parameters = {'kernel': ('linear', 'rbf','poly'), 'C':[1.5, 10],'gamma': [1e-7,
modelsvr = SVR()
clf = GridSearchCV(modelsvr,parameters,cv=5)
clf.fit(x_train,y_train)
print('\nbest parameters for SVR are :',clf.best_params_)

print('')

#SVR best model metrics
SVR_final_model=clf.best_estimator_
SVR_final_prediction=SVR_final_model.predict(x_test)

mse_SVR = mean_squared_error(y_test,SVR_final_prediction)
rmse = np.sqrt(mse_SVR)

print("\nSupport vector Regression RMSE:",rmse)
MAE=mean_absolute_error(y_test, SVR_final_prediction)
print("\nSupport vector Regression MAE:",MAE)
print("\nSupport vector Regression MSE:",mse_SVR)

SVR_scores = cross_val_score(modelsvr, x_train, y_train,scoring="neg_mean_squar
print('\nSVR cross val score',SVR_scores)
print('\nSupport vector  coefficients of bestestimator are ',SVR_final_model.co
```

```
best parameters for SVR are : {'C': 1.5, 'epsilon': 0.2, 'gamma': 1e-07, 'ker


Support vector Regression RMSE: 0.6719473647525518

Support vector Regression MAE: 0.5206050473785004

Support vector Regression MSE: 0.45151326099789885

SVR cross val score [-0.45508625 -0.44595992 -0.45764429 -0.44372559 -0.36624

Support vector  coefficients of bestestimator are  [[ 0.06851271 -0.17575744
  -0.06977631 -0.06198414 -0.0432512   0.16462342  0.32358286]]
```

Training and testing model for WHITE wine data for SVR

```
#training and testing for white wine for SVR
```

```
white_clf = GridSearchCV(modelsvr,parameters,cv=5)
white_clf.fit(a_train,b_train)
print('\nbest parameters for SVR are :',white_clf.best_params_)

print('')

#SVR best model metrics
SVR_final_model=white_clf.best_estimator_
SVR_final_prediction=SVR_final_model.predict(a_test)

mse_SVR = mean_squared_error(b_test,SVR_final_prediction)
rmse = np.sqrt(mse_SVR)

print("\nSupport vector Regression RMSE:",rmse)
MAE=mean_absolute_error(b_test, SVR_final_prediction)
print("\nSupport vector Regression MAE:",MAE)
print("\nSupport vector Regression MSE:",mse_SVR)

SVR_scores = cross_val_score(modelsvr, a_train, b_train,scoring="neg_mean_squar
print('\nSVR cross val score',SVR_scores)
print('\nSupport vector  coefficients of bestestimator are ',SVR_final_model.co
```

```
best parameters for SVR are : {'C': 1.5, 'epsilon': 0.5, 'gamma': 1e-07, 'ker


Support vector Regression RMSE: 0.7408378198484619

Support vector Regression MAE: 0.574043228383877

Support vector Regression MSE: 0.5488406753178221

SVR cross val score [-0.51651117 -0.57723626 -0.53834078 -0.48924912 -0.49889

Support vector  coefficients of bestestimator are  [[ 0.05039598 -0.13283392
  -0.03570399 -0.434043    0.08137242  0.0751687   0.23362164]]
```

In SVR, the 'c' parameter trades off correct classification of training examples against maximization of the decision function's margin.

```
1
```

Determining importance of each variable

#source for statistical importance calculation:Here

#factors for measuring quality of wine are residual suagr and alcohol contenthere

```
#For Lasso L1 regression
num=[]
for col in x.columns:
```