"Wine Quality Prediction & Data Analysis"

Dataset downloaded from

https://github.com/EshitaNandy/Wine-Dataset-Prediction

The overall goal is :

"Wine Quality Prediction & Data Analysis"

- The datasets are related to red and white variants of the Portuguese "Vinho Verde" wine.
- For more details, the reference [Cortez et al., 2009].
- Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).
- These datasets can be viewed as classification or regression tasks.
- The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).
- Outlier detection algorithms could be used to detect the few excellent or poor wines.

Prerequisites : Jupyter Notebook, Pandas, Numpy and Seaborn.

Here are all the imports that we will require to build this model.

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
df = pd.read_csv("D:\DATA SCIENCE\Wine Dataset\WINE_QUALITY.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [4]:
```
df.describe()
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphate
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000	6497.000000	6497.000000	6488.000000	6493.00000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574	0.994697	3.218395	0.5312
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400	56.521855	0.002999	0.160748	0.1488
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.22000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.43000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.51000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.60000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.00000

In [5]:
```
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
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
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
```

In [7]:
```
df["fixed acidity"].value_counts()
```

Out[7]:
```
6.80     354
6.60     326
6.40     305
7.00     282
6.90     279
        ...
14.30      1
15.90      1
13.80      1
14.20      1
6.45       1
Name: fixed acidity, Length: 106, dtype: int64
```

In [8]:
```
mean = df["fixed acidity"].mean()
df["fixed acidity"].fillna(mean,inplace=True)
df["fixed acidity"].isnull().sum()
```

Out[8]: 0

In [9]:
```
mean2 = df["volatile acidity"].mean()
df["volatile acidity"].fillna(mean,inplace=True)
df["volatile acidity"].isnull().sum()
```

Out[9]: 0

In [10]:
```
mean3 = df["citric acid"].mean()
df["citric acid"].fillna(mean,inplace=True)
df["citric acid"].isnull().sum()
```

Out[10]: 0

In [11]:
```
mean4 = df["residual sugar"].mean()
df["residual sugar"].fillna(mean,inplace=True)
df["residual sugar"].isnull().sum()
```

Out[11]: 0

In [12]:
```
mean4 = df["chlorides"].mean()
df["chlorides"].fillna(mean,inplace=True)
df["chlorides"].isnull().sum()
```

Out[12]: 0

In [13]:
```
mean5 = df["pH"].mean()
df["pH"].fillna(mean,inplace=True)
df["pH"].isnull().sum()
```

Out[13]: 0

In [14]:
```
mean6 = df["sulphates"].mean()
df["sulphates"].fillna(mean,inplace=True)
df["sulphates"].isnull().sum()
```

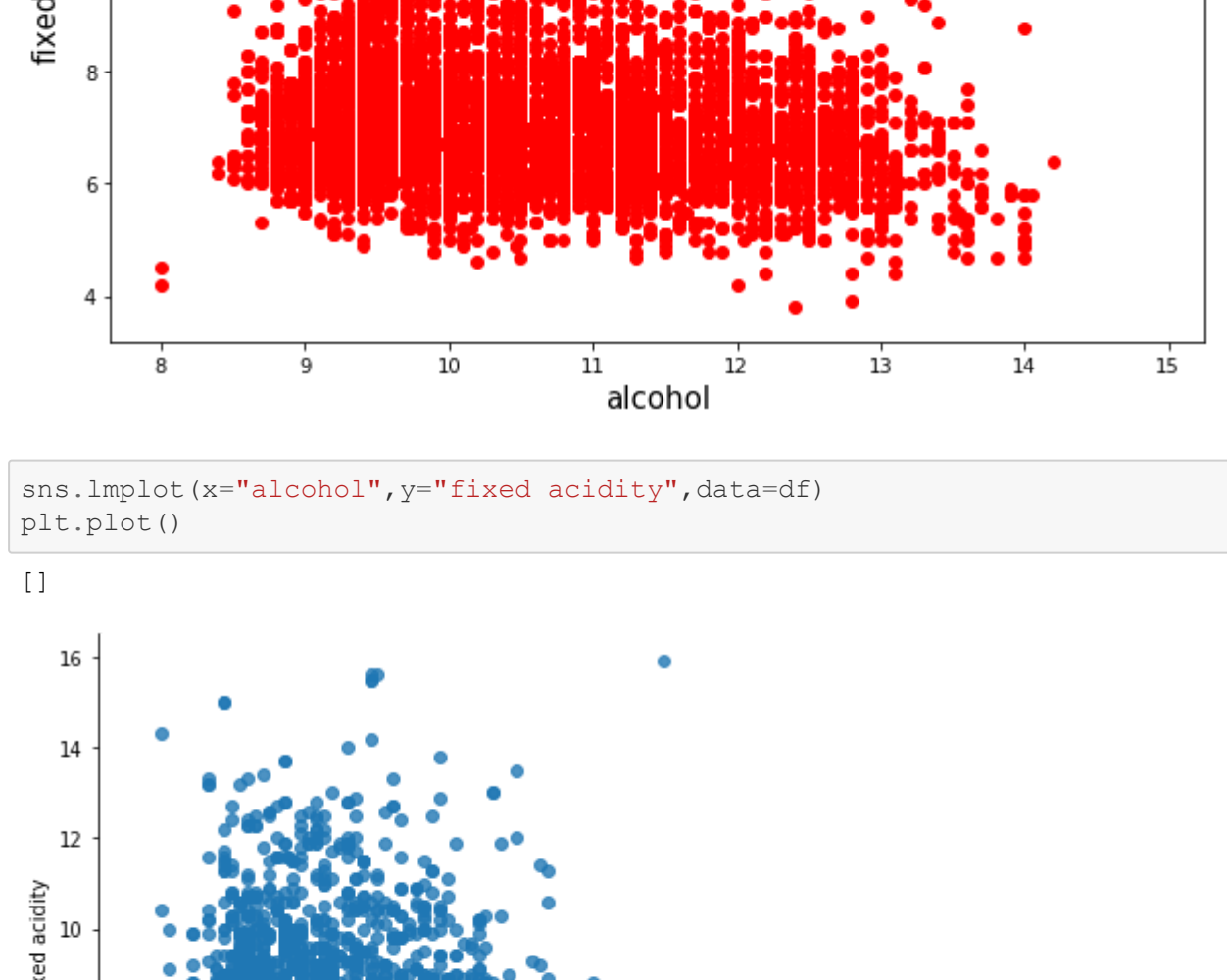
Out[14]: 0

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
type                    0
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
```

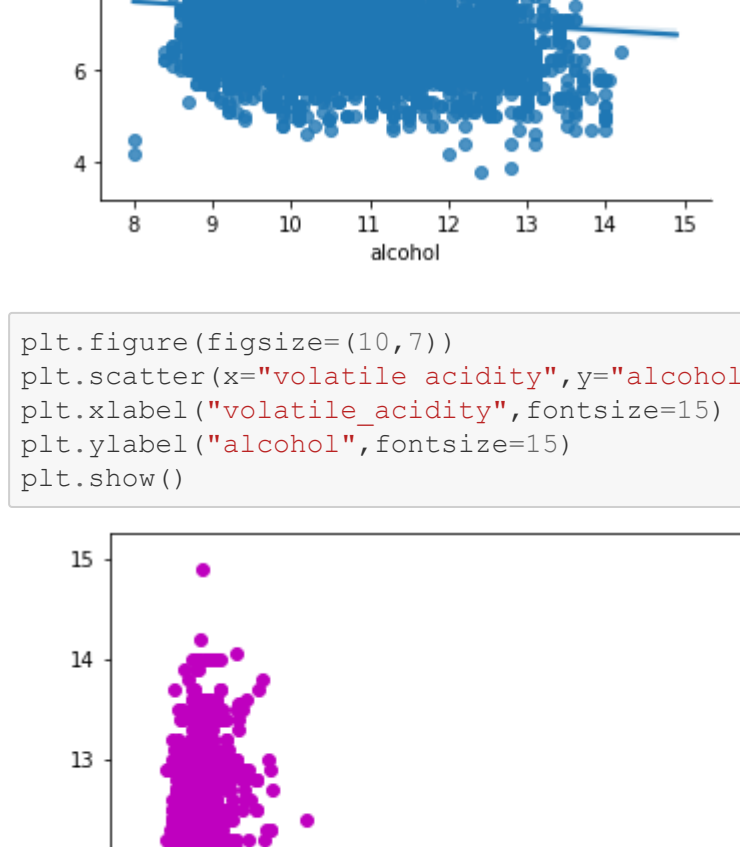
LET's VISUALISE THE DATA

In [16]:
```
plt.figure(figsize=(10,7))
plt.scatter(x="alcohol",y="fixed acidity",data =df,marker= 'o',c="r")
plt.xlabel("alcohol",fontsize=15)
plt.ylabel("fixed_acidity",fontsize=15)
plt.show()
```

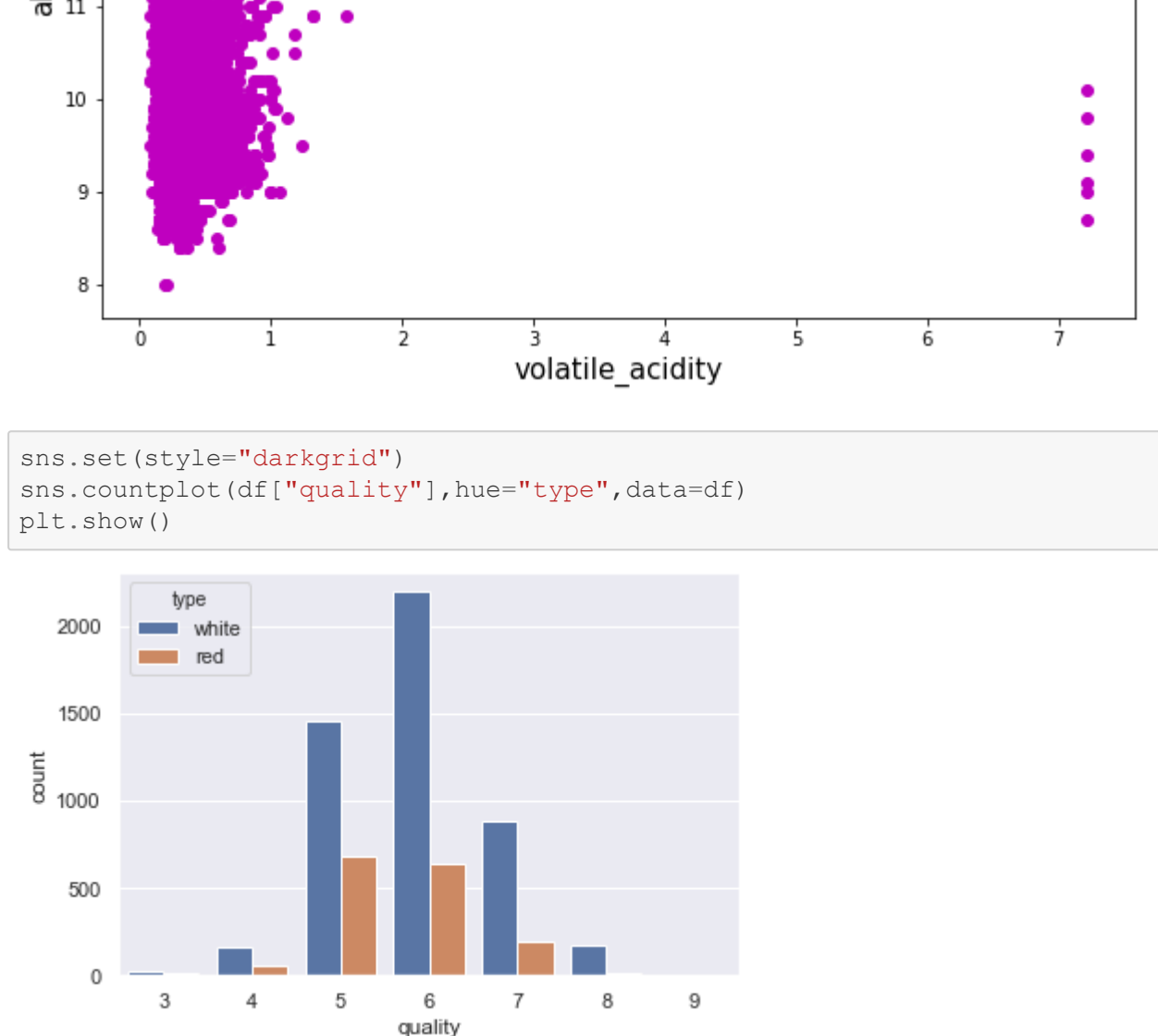


In [17]:
```
sns.lmplot(x="alcohol",y="fixed acidity",data=df)
plt.plot()
```

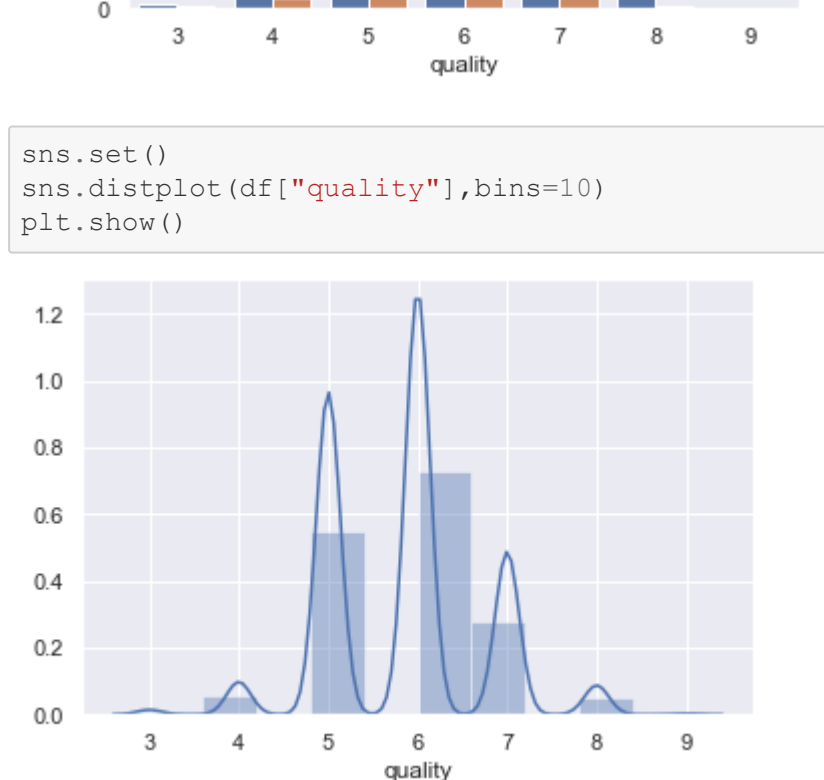
Out[17]: []



In [18]:
```
plt.figure(figsize=(10,7))
plt.scatter(x="volatile acidity",y="alcohol",data =df,marker= 'o',c="m")
plt.xlabel("volatile_acidity",fontsize=15)
plt.ylabel("alcohol",fontsize=15)
plt.show()
```

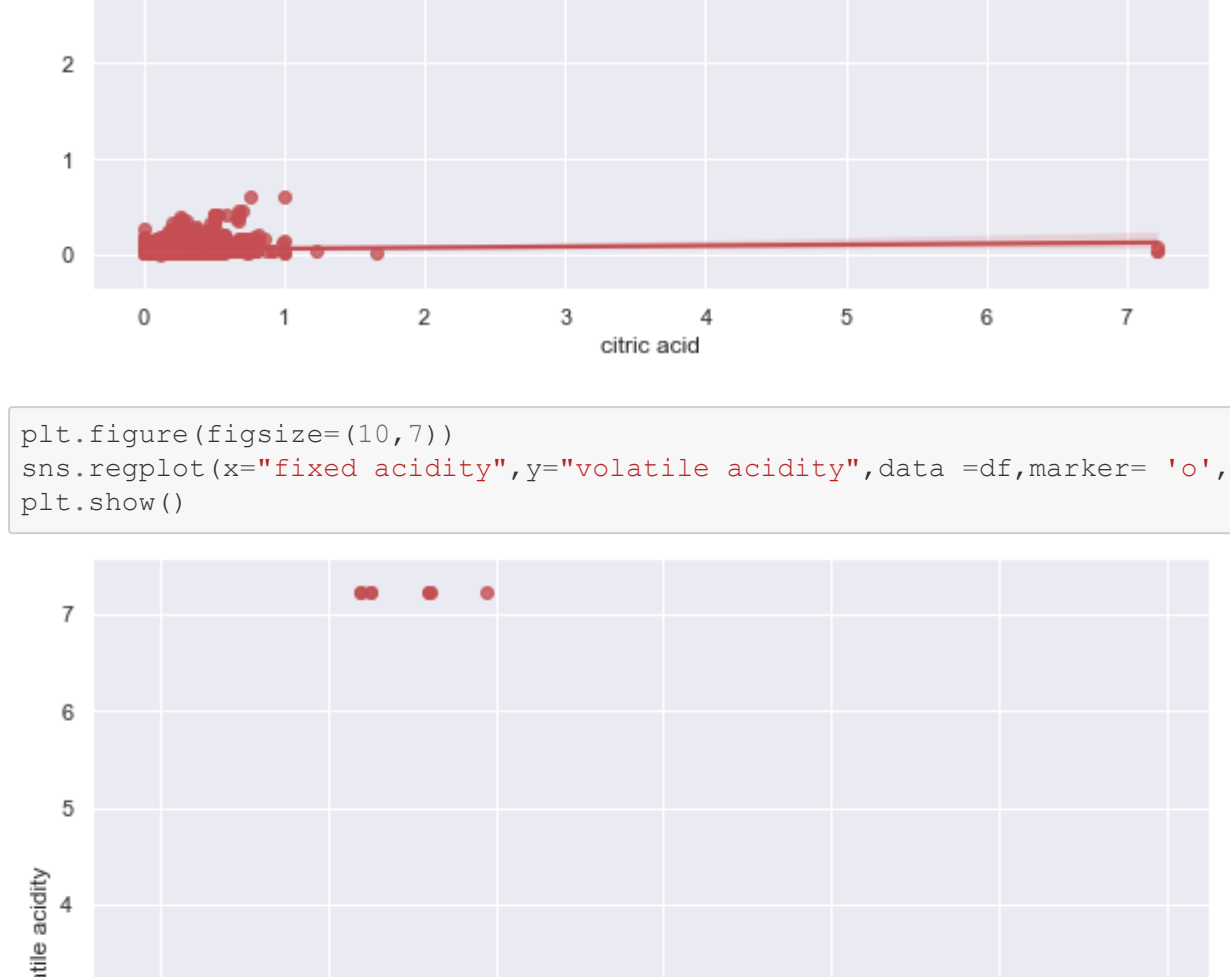


In [19]:
```
sns.set(style="darkgrid")
sns.countplot(df["quality"],hue="type",data=df)
plt.show()
```

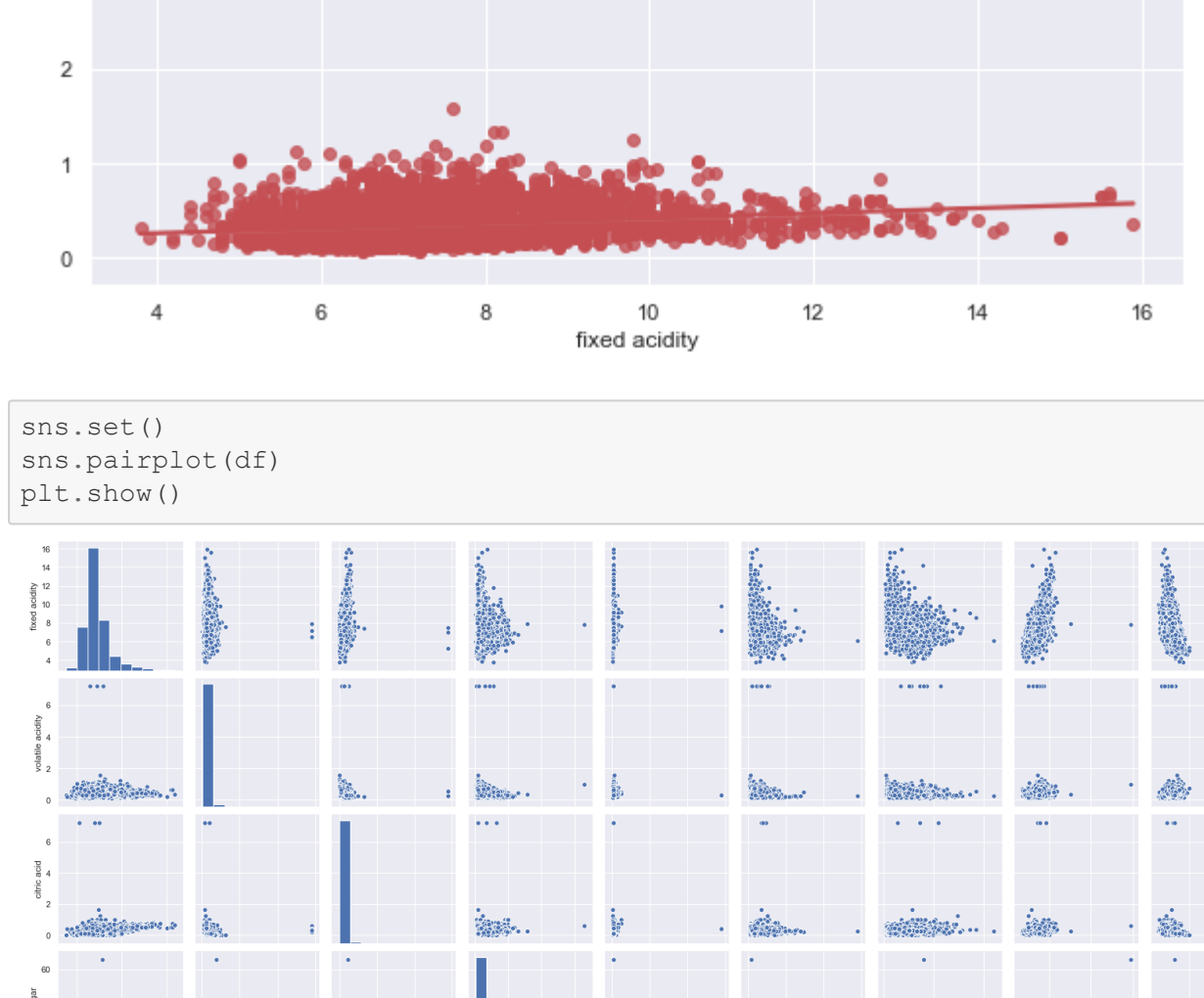


In [20]:
```
sns.set()
sns.distplot(df["quality"],bins=10)
plt.show()
```

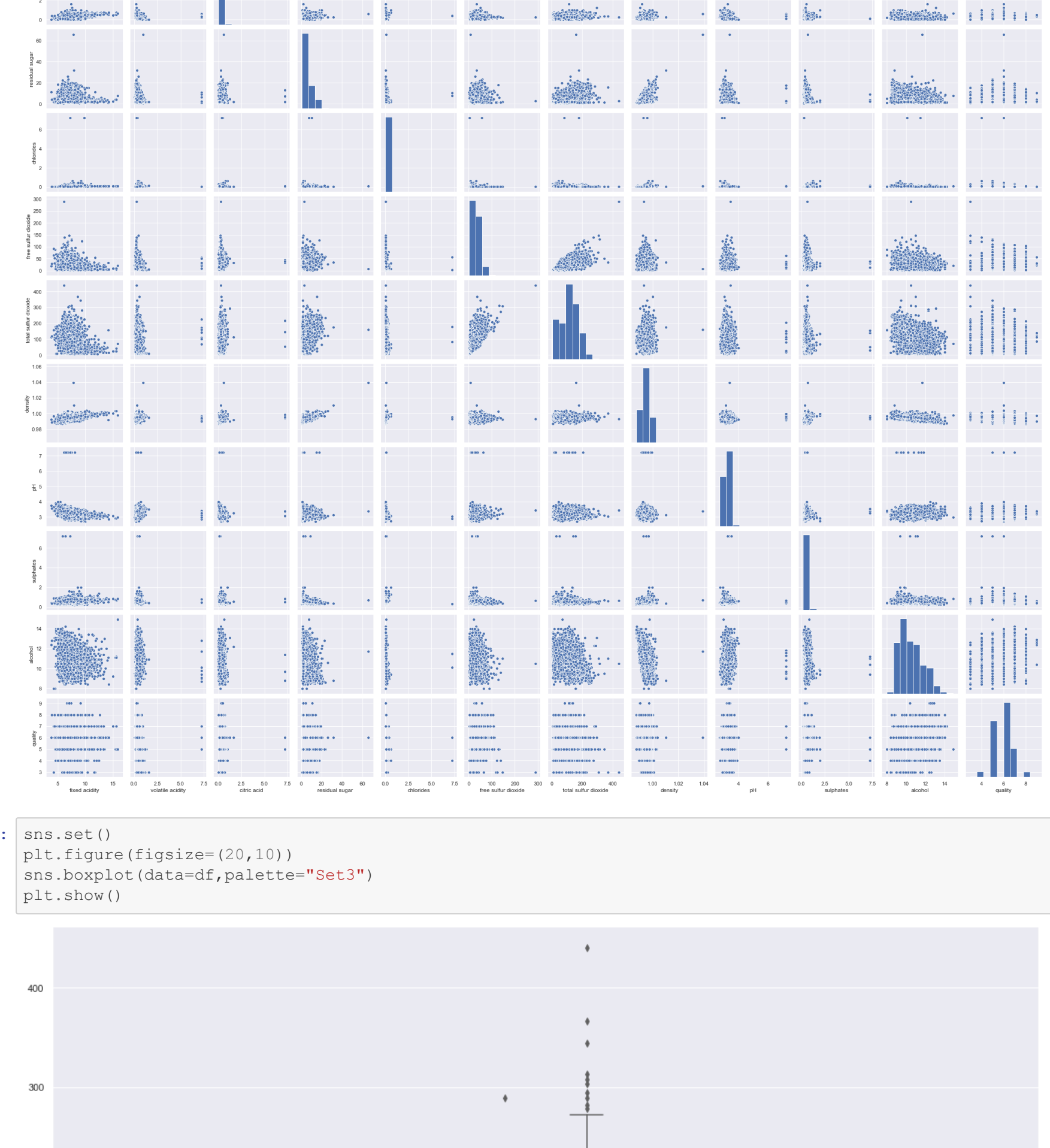
In [21]:
```
plt.figure(figsize=(10,7))
sns.regplot(x="citric acid",y="chlorides",data =df,marker= 'o',color="r")
plt.show()
```



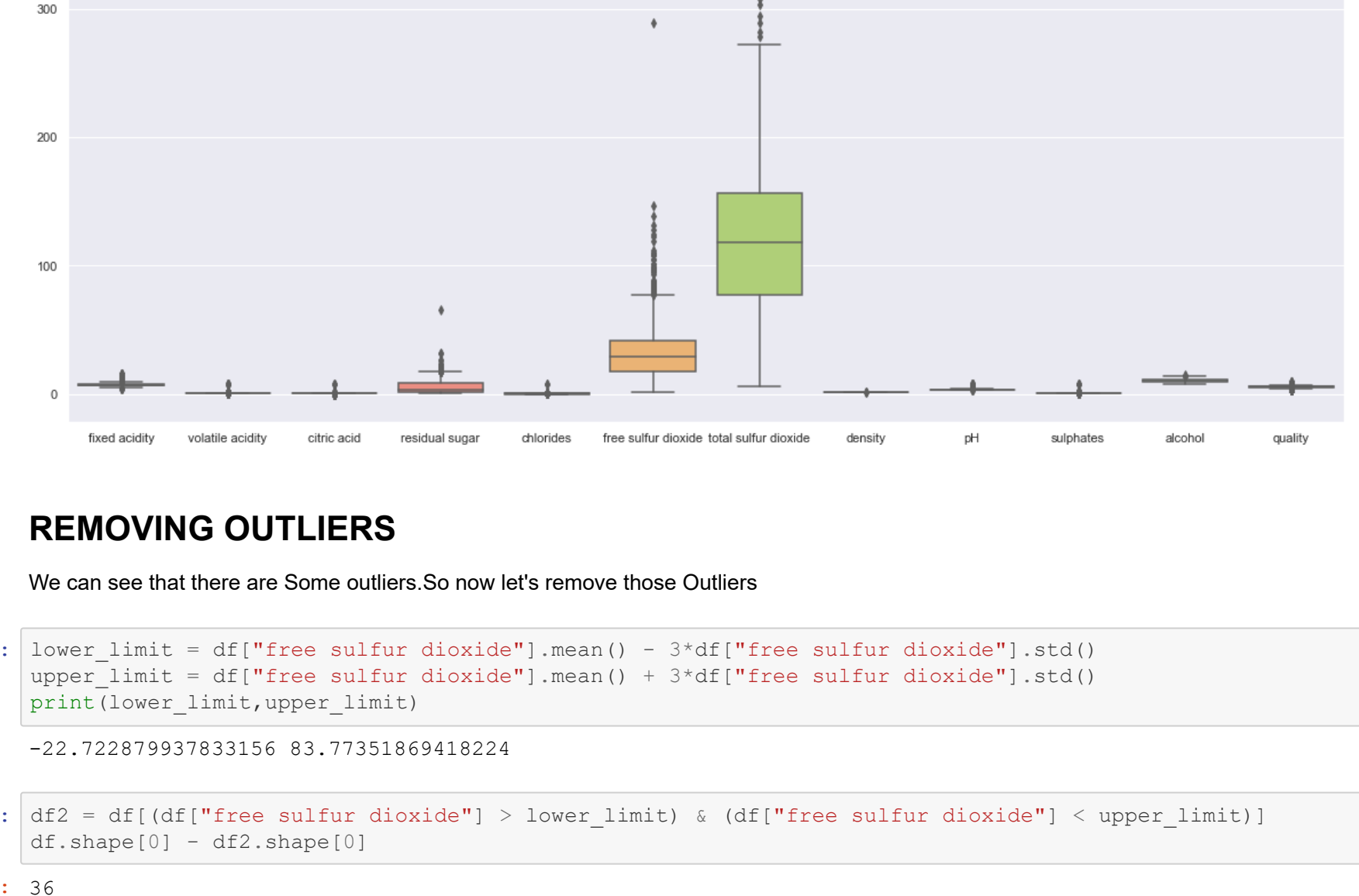
In [22]:
```
plt.figure(figsize=(10,7))
sns.regplot(x="fixed acidity",y="volatile acidity",data =df,marker= 'o',color="r")
plt.show()
```



In [23]:
```
sns.set()
sns.pairplot(df)
plt.show()
```



In [24]:
```
sns.set()
plt.figure(figsize=(20,10))
sns.boxplot(data=df,palette="Set3")
plt.show()
```



REMOVING OUTLIERS

We can see that there are Some outliers.So now let's remove those Outliers

In [25]:
```
lower_limit = df["free sulfur dioxide"].mean() - 3*df["free sulfur dioxide"].std()
upper_limit = df["free sulfur dioxide"].mean() + 3*df["free sulfur dioxide"].std()
print(lower_limit,upper_limit)
```

```
-22.722879937833156 83.77351869418224
```

In [26]:
```
df2 = df[(df["free sulfur dioxide"] > lower_limit) & (df["free sulfur dioxide"] < upper_limit)]
df.shape[0] - df2.shape[0]
```

Out[26]: 36

In [27]:
```
lower_limit = df2['total sulfur dioxide'].mean() - 3*df2['total sulfur dioxide'].std()
upper_limit = df2['total sulfur dioxide'].mean() + 3*df2['total sulfur dioxide'].std()
print(lower_limit,upper_limit)
```

```
-53.15243132839596 283.65436601342924
```

In [28]:
```
df3 = df2[(df2['total sulfur dioxide'] > lower_limit) & (df2['total sulfur dioxide'] < upper_limit)]
df3.head()
```

Out[28]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [29]:
```
df2.shape[0] - df3.shape[0]
```

Out[29]: 5

In [30]:
```
lower_limit = df3['residual sugar'].mean() - 3*df3['residual sugar'].std()
upper_limit = df3['residual sugar'].mean() + 3*df3['residual sugar'].std()
print(lower_limit,upper_limit)
```

```
-8.835623392315494 19.713297852856098
```

In [31]:
```
df4 = df3[(df3['residual sugar'] > lower_limit) & (df3['residual sugar'] < upper_limit)]
df4.head()
```

Out[31]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
5	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6

In [32]:
```
df3.shape[0] - df4.shape[0]
```

Out[32]: 26

In [33]:
```
df4.isnull().sum()
```

Out[33]:
```
type                    0
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
```

In [34]:
```
dummies = pd.get_dummies(df4["type"],drop_first=True)
```

In [35]:
```
df4 = pd.concat([df4,dummies],axis=1)
df4.drop("type",axis=1,inplace=True)
df4.head()
```

Out[35]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	white
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	1
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6	1

In [36]:
```
df4.quality.value_counts()
```

Out[36]:
```
6    2806
5    2116
7    1075
4     214
8     189
3      25
9       5
Name: quality, dtype: int64
```

In [37]:
```
df4.head()
```

Out[37]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	white
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	1
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6	1

Now lets Change the Categorical 'String' Variables into Numerical Variables

In [38]:
```
quaity_mapping = { 3 : "Low",4 : "Low",5: "Medium",6 : "Medium",7: "Medium",8 : "High",9 : "High"}
df4["quality"] =  df4["quality"].map(quaity_mapping)
```

In [39]:
```
df4.quality.value_counts()
```

Out[39]:
```
Medium    5997
Low        239
High       194
Name: quality, dtype: int64
```

In [40]:
```
df4.head()
```

Out[40]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	white
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	Medium	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	Medium	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	Medium	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	Medium	1
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	Medium	1

In [41]:
```
mapping_quality = {"Low" : 0,"Medium": 1,"High" : 2}
df4["quality"] =  df4["quality"].map(mapping_quality)
df4.head()
```

Out[41]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	white
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	1	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	1	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	1	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	1	1
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	1	1

Selecting the best Features for our Model

In [42]:
```
x = df4.drop("quality",axis=True)
y = df4["quality"]
```

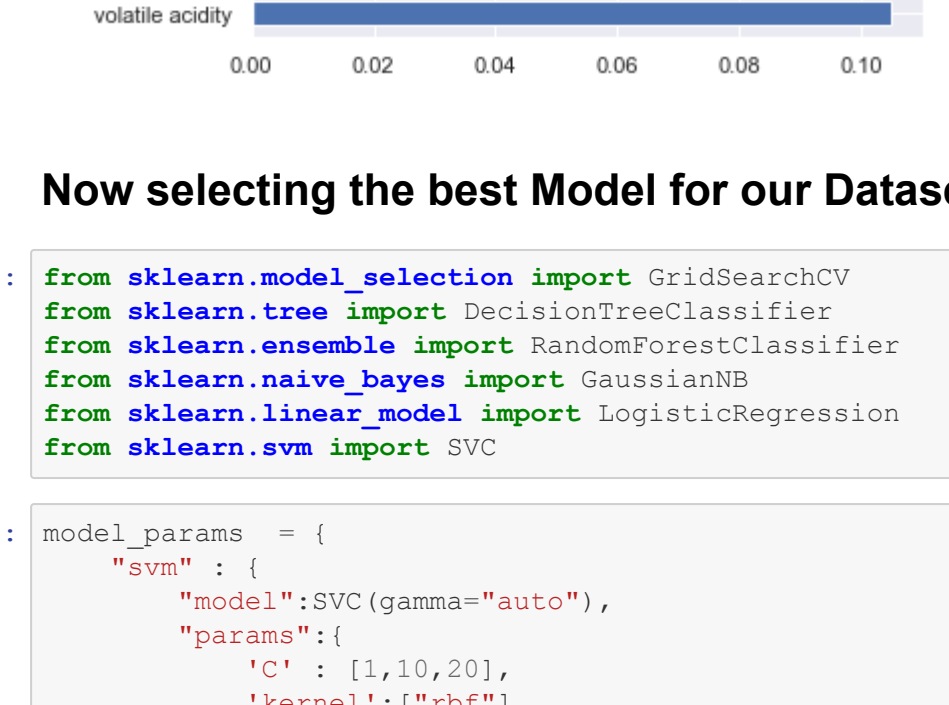
In [43]:
```
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
model.fit(x,y)
```

Out[43]:
```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100,
                     n_jobs=None, oob_score=False, random_state=None, verbose=0,
                     warm_start=False)
```

In [44]:
```
print(model.feature_importances_)
```

```
[0.08373092 0.10482106 0.08603915 0.08914575 0.08501738 0.10395321
 0.08564312 0.08663665 0.08155549 0.08990292 0.09602492 0.00752943]
```

In [45]:
```
feat_importances = pd.Series(model.feature_importances_,index =x.columns)
feat_importances.nlargest(9).plot(kind="barh")
plt.show()
```



Now selecting the best Model for our Dataset

In [46]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [47]:
```
model_params  = {
    "svm" : {
        "model":SVC(gamma="auto"),
        "params":{
            'C' : [1,10,20],
            'kernel':["rbf"]
        }
    },

    "decision_tree":{
        "model": DecisionTreeClassifier(),
        "params":{
            'criterion':["entropy","gini"],
            "max_depth":[5,8,9]
        }
    },

    "random_forest":{
        "model": RandomForestClassifier(),
        "params":{
            "n_estimators":[1,5,10],
            "max_depth":[5,8,9]
        }
    },
    "naive_bayes":{
        "model": GaussianNB(),
        "params":{}
    },

    'logistic_regression' : {
        'model' : LogisticRegression(solver='liblinear',multi_class = 'auto'),
        'params': {
            "C" : [1,5,10]
        }
    }

}
```

In [48]:
```
score=[]
for model_name,mp in model_params.items():
    clf = GridSearchCV(mp["model"],mp["params"],cv=8,return_train_score=False)
    clf.fit(x,y)
    score.append({
        "Model" : model_name,
        "Best_Score": clf.best_score_,
        "Best_Params": clf.best_params_
    })
```

In [49]:
```
df5 = pd.DataFrame(score,columns=["Model","Best_Score","Best_Params"])
```

In [50]:
```
df5.head()
```

Out[50]:

	Model	Best_Score	Best_Params
0	svm	0.932193	{'C': 1, 'kernel': 'rbf'}
1	decision_tree	0.923484	{'criterion': 'entropy', 'max_depth': 5}
2	random_forest	0.932348	{'max_depth': 5, 'n_estimators': 10}
3	naive_bayes	0.636770	{}
4	logistic_regression	0.932348	{'C': 1}

So we can see that, we are getting 93% accuracy for "SVM" & "Random Forest".

In [51]:
```
from sklearn.model_selection import cross_val_score
clf_svm = SVC(kernel="rbf",C=1)
scores = cross_val_score(clf_svm,x,y,cv=8,scoring="accuracy")
```

In [52]:
```
scores
```

Out[52]:
```
array([0.93283582, 0.93283582, 0.93283582, 0.93283582, 0.93283582,
       0.93159204, 0.93275218, 0.93275218])
```

In [53]:
```
scores.mean()
```

Out[53]: 0.9326594378667064

So we are getting "93% Accuracy for predicting the Quality of Wine".

In [54]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

In [55]:
```
clf_svm1 = SVC(kernel="rbf",C=1)
clf_svm1.fit(x_train,y_train)
```

Out[55]:
```
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [56]:
```
y_pred = clf_svm1.predict(x_test)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

Out[56]: 0.9339035769828927

Now Lets see the Real value and Predicted Value

In [57]:

```
accuracy_dataframe = pd.DataFrame({"y_test": y_test, "y_pred": y_pred})
accuracy_dataframe.head()
```

Out[57]:

	y_test	y_pred
1935	1	1
1106	2	1
2932	1	1
743	1	1
2230	1	1