In [316]:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
from scipy import stats
import plotly
import plotly.offline as py
import plotly.graph_objs as go
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

pd.set_option("display.max_columns", None)
#reading the data as the delimiter is different
data_set_red=pd.read_csv("winequality-red.csv",delimiter=';')
data_set_white=pd.read_csv("winequality-white.csv",delimiter=';')
```

In [317]:

```
print("Rows and Columns in Red Wine Dataset: ")
data_set_red.shape
```

```
Rows and Columns in Red Wine Dataset:
```

Out[317]:

```
(1599, 12)
```

In [318]:

```
print("Rows and Columns in White Wine Dataset: ")
data_set_white.shape
```

```
Rows and Columns in White Wine Dataset:
```

Out[318]:

```
(4898, 12)
```

In [319]:

```
#DATA SET 10 ROWS for Dataset Red wine quality
data_set_red.head(10)
```

Out[319]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	5
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	5
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	7
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	7
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	5

In [320]:

```
#DATA SET 10 ROWS for Dataset White wine quality
data_set_white.head(10)
```

Out[320]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
6	6.2	0.32	0.16	7.0	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6
7	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
8	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
9	8.1	0.22	0.43	1.5	0.044	28.0	129.0	0.9938	3.22	0.45	11.0	6

In [321]:

```
data_set_red.columns
data_set_white.columns
```

Out[321]:

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [ ]:

```

```

In [322]:

```
val_set_red=data_set_red['quality'].value_counts()
val_set_red
```

Out[322]:

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

In [323]:

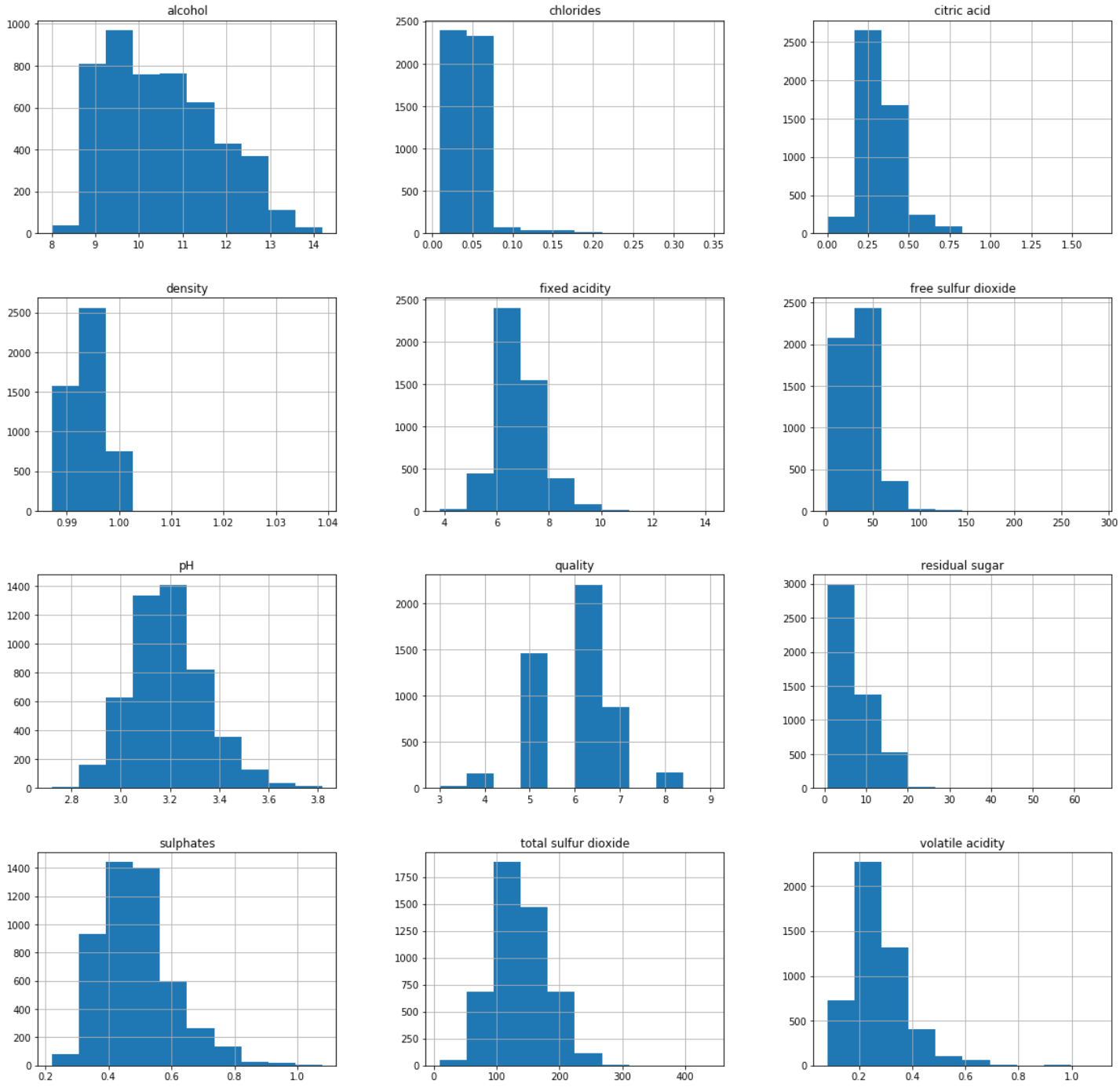
```
val_set_white=data_set_white['quality'].value_counts()
val_set_white
```

Out[323]:

```
6    2198
5    1457
7     880
8     175
4     163
3      20
9       5
Name: quality, dtype: int64
```

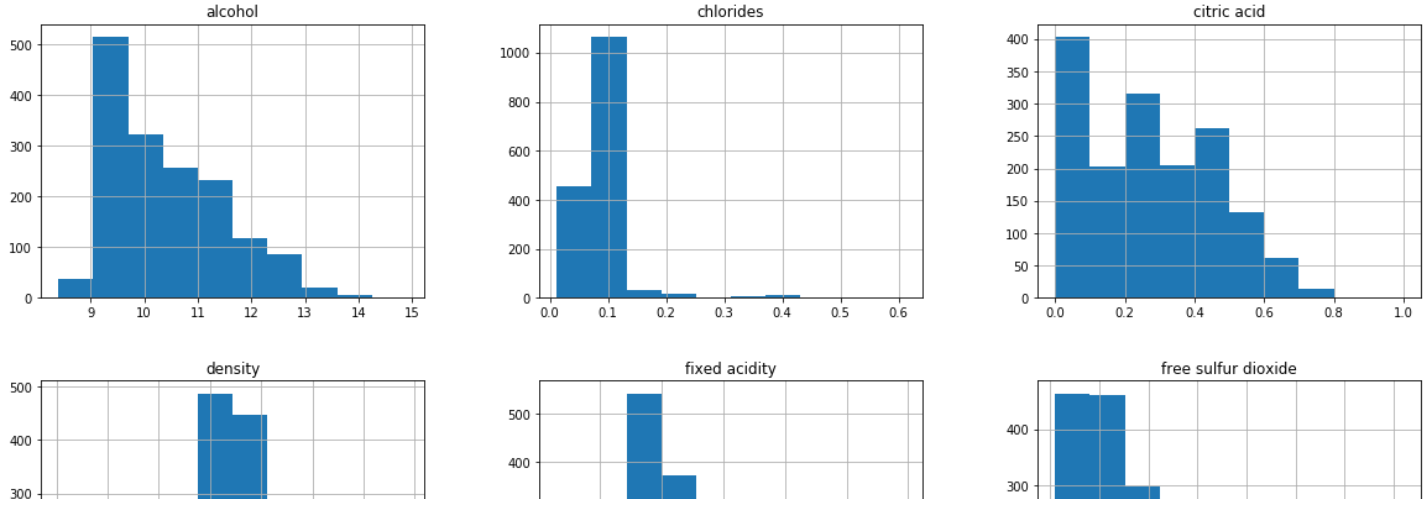
In [324]:

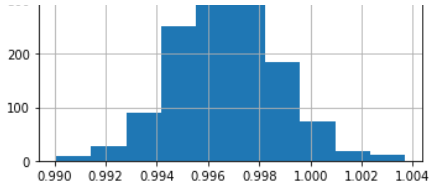
```
data_set_white.hist(bins=10,figsize=(20,20))
plt.show()
```

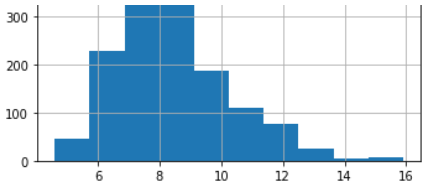


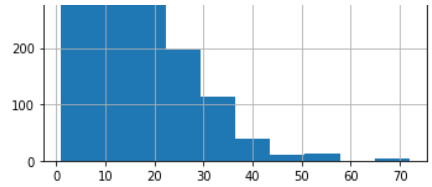
In [325]:

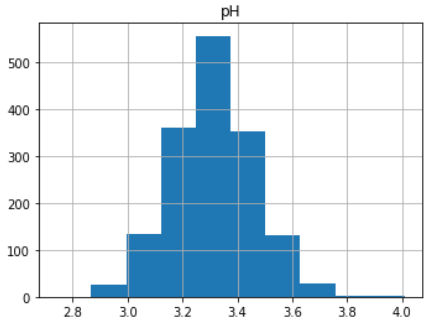
```
data_set_red.hist(bins=10,figsize=(20,20))
plt.show()
```

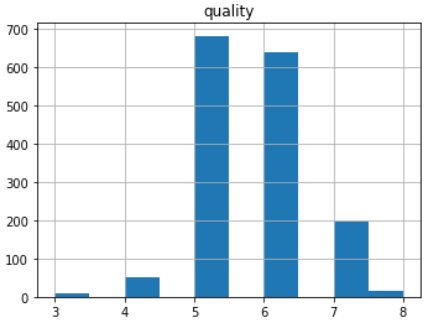


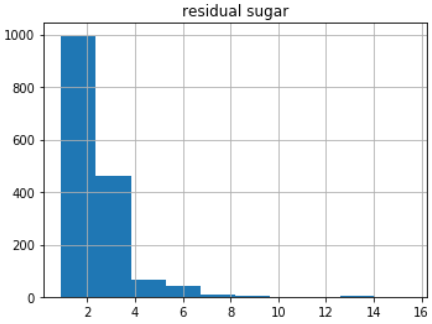


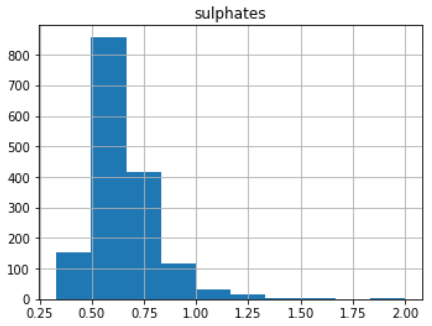


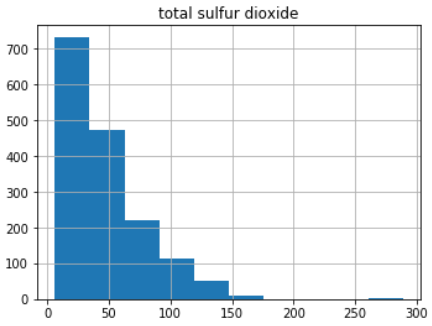


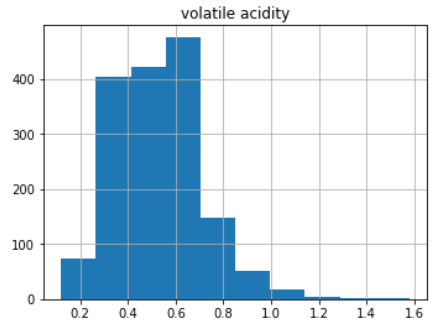










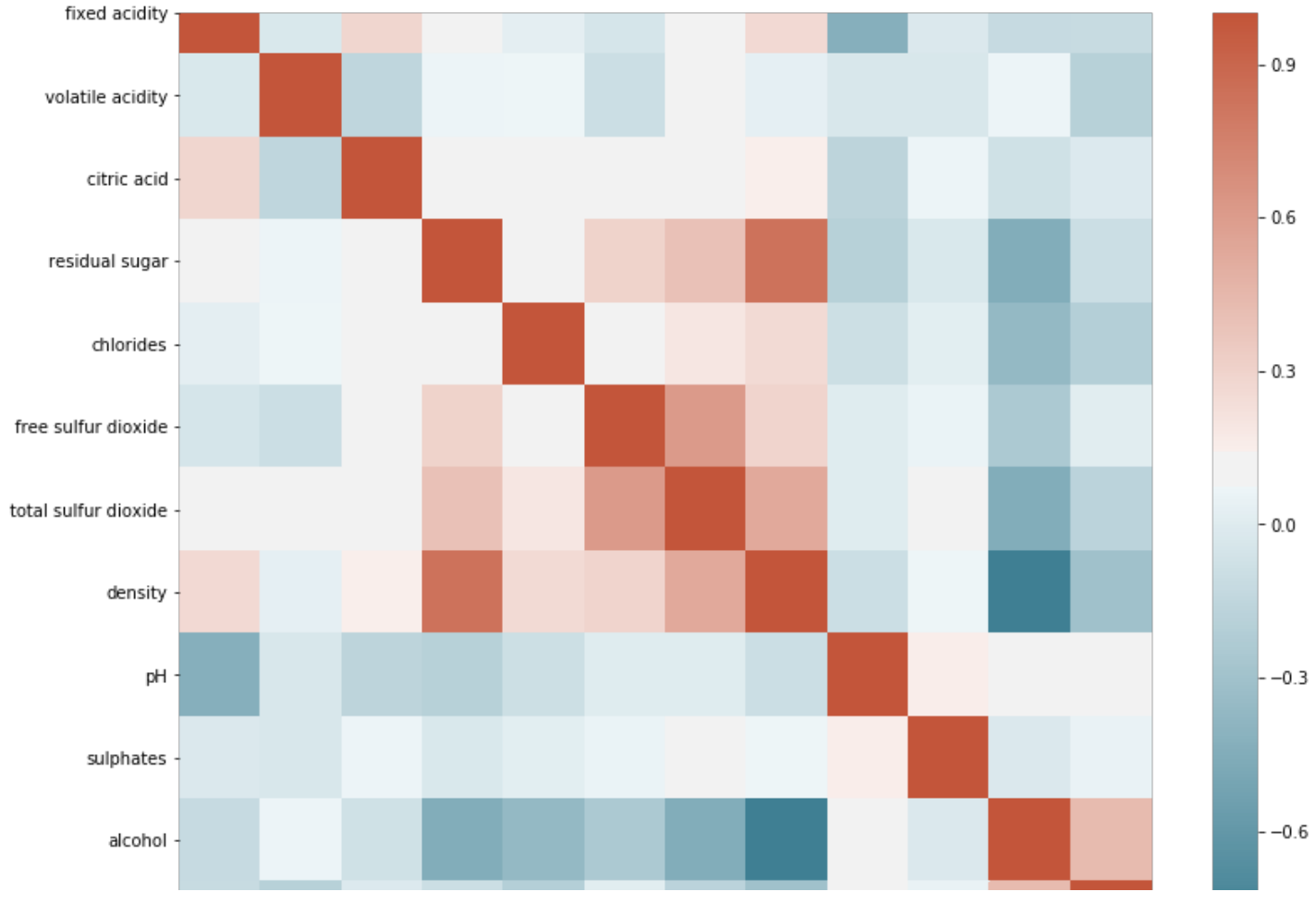


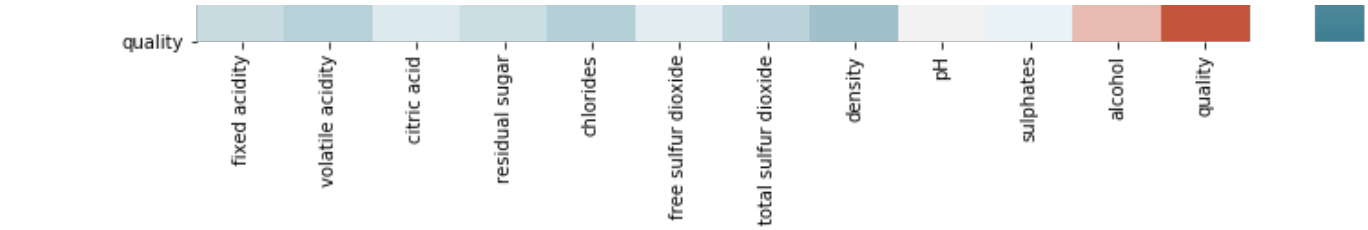
In [326]:

```
corr_white = data_set_white.corr()
plt.subplots(figsize=(13,10))
sns.heatmap(corr_white, xticklabels=corr_white.columns, yticklabels=corr_white.columns,c
map=sns.diverging_palette(220,20, as_cmap=True))
```

Out[326]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18a92725048>
```



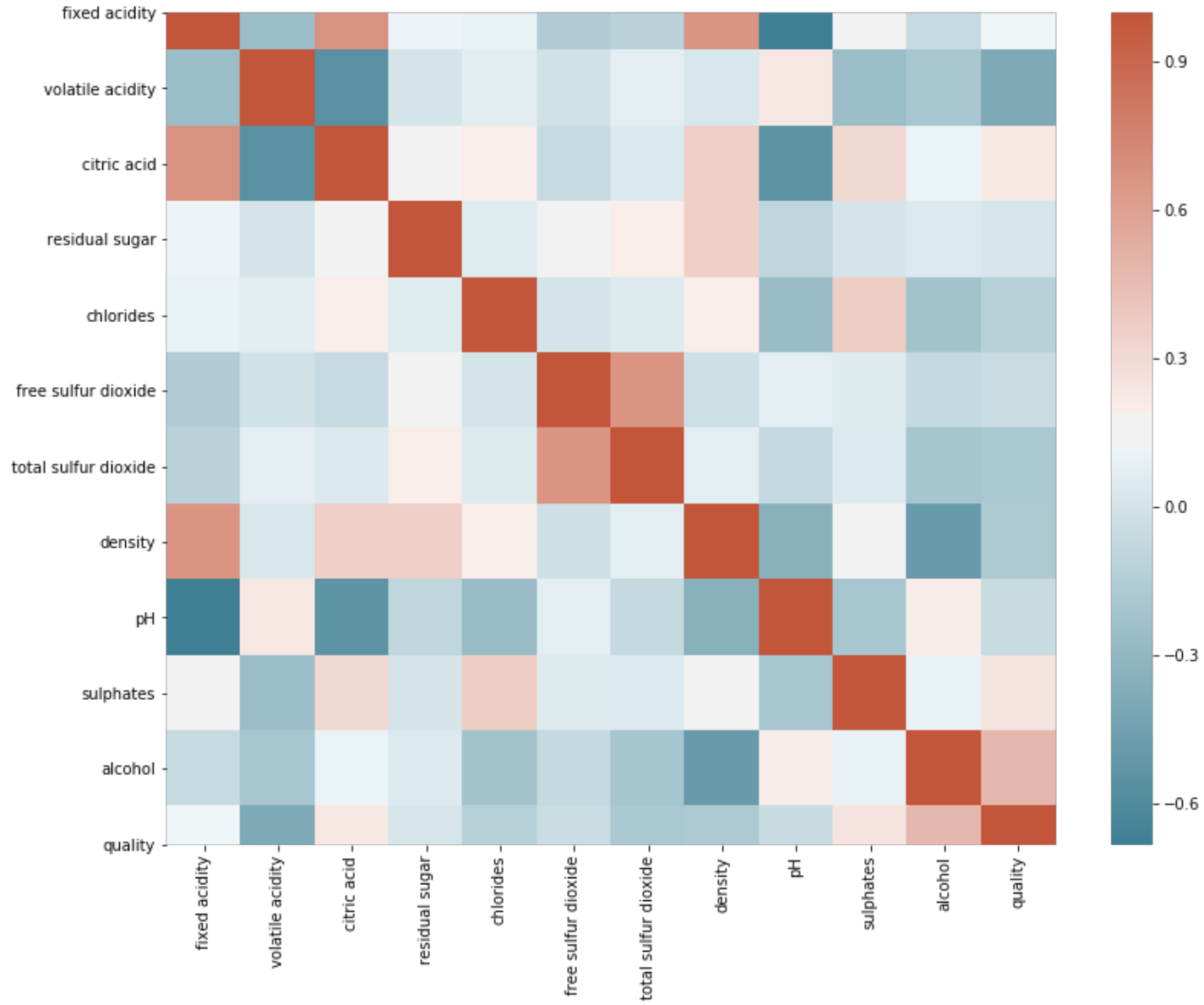


In [327]:

```
corr_red = data_set_red.corr()
plt.subplots(figsize=(13,10))
sns.heatmap(corr_red, xticklabels=corr_red.columns, yticklabels=corr_red.columns,cmap=sns
.diverging_palette(220,20, as_cmap=True))
```

Out[327]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18a973a40c8>
```



In [328]:

```
corr_white['quality'].sort_values(ascending=False)
```

Out[328]:

```
quality                1.000000
alcohol                0.435575
pH                     0.099427
sulphates              0.053678
free sulfur dioxide    0.008158
citric acid           -0.009209
residual sugar        -0.097577
fixed acidity         -0.113663
```

```
fixed acidity           0.113663
total sulfur dioxide   -0.174737
volatile acidity       -0.194723
chlorides              -0.209934
density                -0.307123
Name: quality, dtype: float64
```

In [329]:

```
corr_red['quality'].sort_values(ascending=False)
```

Out[329]:

```
quality                 1.000000
alcohol                 0.476166
sulphates               0.251397
citric acid             0.226373
fixed acidity           0.124052
residual sugar          0.013732
free sulfur dioxide    -0.050656
pH                     -0.057731
chlorides              -0.128907
density                -0.174919
total sulfur dioxide   -0.185100
volatile acidity       -0.390558
Name: quality, dtype: float64
```

Observed that from this data that the correlation of alcohol, pH, sulphastes and free sulgur dioxide is more in White wine (top 4) Observed that from this data that the correlation of alcohol, sulphastes, citric acid and fixed acidity is more in red wine (top 4)

In [274]:

```
#dropping unused features in both red and white wine
data_set_white_up=data_set_white.copy()
data_set_white_up.drop(['citric acid','residual sugar','fixed acidity','total sulfur dio
xide','volatile acidity','chlorides','density'],axis=1,inplace=True)
data_set_white_up.head()
```

Out[274]:

	free sulfur dioxide	pH	sulphates	alcohol	quality
0	45.0	3.00	0.45	8.8	6
1	14.0	3.30	0.49	9.5	6
2	30.0	3.26	0.44	10.1	6
3	47.0	3.19	0.40	9.9	6
4	47.0	3.19	0.40	9.9	6

In [ ]:

```

```

In [275]:

```
py.init_notebook_mode(connected=True)
# free sulfur dioxide
slope_sulf,intercept_sulf, r_value, p_value, std_err = stats.linregress(data_set_white_u
p['free sulfur dioxide'], data_set_white_up['quality'])
# pH
slope_pH, intercept_pH, r_value, p_value, std_err = stats.linregress(data_set_white_up['
pH'], data_set_white_up['quality'])
# sulphates
slope_sulp, intercept_sulp, r_value, p_value, std_err = stats.linregress(data_set_white_
up['sulphates'], data_set_white_up['quality'])
# alcohol
slope_alcohol, intercept_alcohol, r_value, p_value, std_err = stats.linregress(data_set_
white_up['alcohol'], data_set_white_up['quality'])
```

In [276]:

```
bins= [2,6,9]
group_names = ['bad','good']
data_set_white_up['quality'] = pd.cut(data_set_white_up['quality'], bins = bins, labels
= group_names)
label_quality = LabelEncoder()
data_set_white_up['quality'] = label_quality.fit_transform(data_set_white_up['quality'])
```

In [277]:

```
X = data_set_white_up.drop('quality', axis=1)
y = data_set_white_up['quality']
X_train, X_test, y_train, y_test = train_test_split(X, y)
y_train
data_set_white_up.isnull().any()
```

Out[277]:

```
free sulfur dioxide    False
pH                     False
sulphates              False
alcohol                False
quality                False
dtype: bool
```

In [278]:

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
log_reg_pred = log_reg.predict(X_test)
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureW
arning:

Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
```

In [279]:

```
print(classification_report(y_test, log_reg_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.98      0.88       961
           1       0.63      0.16      0.25       264

    accuracy                           0.80      1225
   macro avg       0.72      0.57      0.57      1225
weighted avg       0.77      0.80      0.75      1225
```

In [280]:

```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier(loss='log') # the 'log' loss gives logistic regression
sgd.fit(X_train, y_train)
sgd_pred = sgd.predict(X_test)
```

In [281]:

```
print(classification_report(y_test, sgd_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.95      0.88       961
           1       0.59      0.28      0.38       264

    accuracy                           0.80      1225
   macro avg       0.71      0.61      0.63      1225
```

```
weighted avg       0.78      0.80      0.78      1225
```

In [282]:

```
from sklearn.ensemble import GradientBoostingClassifier

gradient = GradientBoostingClassifier()
gradient.fit(X_train, y_train)
gradient_pred = gradient.predict(X_test)
```

In [283]:

```
print(classification_report(y_test, gradient_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90       961
           1       0.71      0.34      0.46       264

    accuracy                           0.83      1225
   macro avg       0.77      0.65      0.68      1225
weighted avg       0.81      0.83      0.80      1225
```

In [284]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
#Let's see how our model performed
print(classification_report(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.89      0.96      0.92       961
           1       0.78      0.57      0.66       264

    accuracy                           0.87      1225
   macro avg       0.83      0.76      0.79      1225
weighted avg       0.87      0.87      0.86      1225
```

In [285]:

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
pred_svc = svc.predict(X_test)
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning:

The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this war
ning.
```

In [286]:

```
print(classification_report(y_test, pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.82      0.96      0.88       961
           1       0.60      0.22      0.32       264

    accuracy                           0.80      1225
   macro avg       0.71      0.59      0.60      1225
weighted avg       0.77      0.80      0.76      1225
```

In [292]:

```
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

model1 = DecisionTreeClassifier(random_state=1)
model1.fit(X_train, y_train)
y_pred1 = model1.predict(X_test)
```

In [293]:

```
print(classification_report(y_test, y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.89      0.89      0.89       961
           1       0.60      0.61      0.60       264

    accuracy                           0.83      1225
   macro avg       0.75      0.75      0.75      1225
weighted avg       0.83      0.83      0.83      1225
```

In [299]:

```
data_set_red_up=data_set_red.copy()
data_set_red_up.drop(['free sulfur dioxide','pH','chlorides','total sulfur dioxide','vola
tile acidity','density'],axis=1,inplace=True)
data_set_red_up.head()
```

Out[299]:

	fixed acidity	citric acid	residual sugar	sulphates	alcohol	quality
0	**7.4**	**0.00**	**1.9**	**0.56**	**9.4**	**5**
1	**7.8**	**0.00**	**2.6**	**0.68**	**9.8**	**5**
2	**7.8**	**0.04**	**2.3**	**0.65**	**9.8**	**5**
3	**11.2**	**0.56**	**1.9**	**0.58**	**9.8**	**6**
4	**7.4**	**0.00**	**1.9**	**0.56**	**9.4**	**5**

In [300]:

```
py.init_notebook_mode(connected=True)
# fixed acidity
slope_acid, intercept_acid, r_value, p_value, std_err = stats.linregress(data_set_red_up
['fixed acidity'], data_set_red_up['quality'])
# citric acid
slope_cit, intercept_cit, r_value, p_value, std_err = stats.linregress(data_set_red_up['
citric acid'], data_set_red_up['quality'])
# residual sugar
slope_sugar, intercept_sugar, r_value, p_value, std_err = stats.linregress(data_set_red_
up['residual sugar'], data_set_red_up['quality'])
# sulphates
slope_sulp, intercept_sulp, r_value, p_value, std_err = stats.linregress(data_set_red_up
['sulphates'], data_set_red_up['quality'])
# alcohol
slope_alcohol, intercept_alcohol, r_value, p_value, std_err = stats.linregress(data_set_
red_up['alcohol'], data_set_red_up['quality'])
```

In [301]:

```
bins= [2,6,9]
group_names = ['bad','good']
data_set_red_up['quality'] = pd.cut(data_set_red_up['quality'], bins = bins, labels = gr
oup_names)
label_quality = LabelEncoder()
data_set_red_up['quality'] = label_quality.fit_transform(data_set_red_up['quality'])
```

In [302]:

```
data_set_red_up
```

Out[302]:

	fixed acidity	citric acid	residual sugar	sulphates	alcohol	quality
0	7.4	0.00	1.9	0.56	9.4	0
1	7.8	0.00	2.6	0.68	9.8	0
2	7.8	0.04	2.3	0.65	9.8	0
3	11.2	0.56	1.9	0.58	9.8	0
4	7.4	0.00	1.9	0.56	9.4	0
...	...	...	...	...	...	...
1594	6.2	0.08	2.0	0.58	10.5	0
1595	5.9	0.10	2.2	0.76	11.2	0
1596	6.3	0.13	2.3	0.75	11.0	0
1597	5.9	0.12	2.0	0.71	10.2	0
1598	6.0	0.47	3.6	0.66	11.0	0

1599 rows × 6 columns

In [304]:

```
X = data_set_red_up.drop('quality', axis=1)
y = data_set_red_up['quality']
X_train, X_test, y_train, y_test = train_test_split(X, y)
y_train
data_set_red_up.isnull().any()
```

Out[304]:

```
fixed acidity     False
citric acid       False
residual sugar    False
sulphates         False
alcohol           False
quality           False
dtype: bool
```

In [305]:

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
log_reg_pred = log_reg.predict(X_test)
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureW
arning:

Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warni
ng.
```

In [306]:

```
print(classification_report(y_test, log_reg_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       341
           1       0.43      0.05      0.09        59

    accuracy                           0.85       400
   macro avg       0.64      0.52      0.50       400
weighted avg       0.79      0.85      0.80       400
```

In [307]:

```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier(loss='log') # the 'log' loss gives logistic regression
sgd.fit(X_train, y_train)
sgd_pred = sgd.predict(X_test)
```

In [308]:

```
print(classification_report(y_test, sgd_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       341
           1       0.62      0.08      0.15        59

    accuracy                           0.86       400
   macro avg       0.74      0.54      0.54       400
weighted avg       0.83      0.86      0.81       400
```

In [309]:

```
from sklearn.ensemble import GradientBoostingClassifier

gradient = GradientBoostingClassifier()
gradient.fit(X_train, y_train)
gradient_pred = gradient.predict(X_test)
```

In [310]:

```
print(classification_report(y_test, gradient_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.96      0.93       341
           1       0.63      0.41      0.49        59

    accuracy                           0.88       400
   macro avg       0.77      0.68      0.71       400
weighted avg       0.86      0.88      0.87       400
```

In [311]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
#Let's see how our model performed
print(classification_report(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.95       341
           1       0.77      0.51      0.61        59

    accuracy                           0.91       400
   macro avg       0.84      0.74      0.78       400
weighted avg       0.90      0.91      0.90       400
```

In [312]:

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
pred_svc = svc.predict(X_test)
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning:
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning:

The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this war
ning.
```

In [313]:

```
print(classification_report(y_test, pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.87      0.98      0.92       341
           1       0.60      0.15      0.24        59

    accuracy                           0.86       400
   macro avg       0.74      0.57      0.58       400
weighted avg       0.83      0.86      0.82       400
```

In [314]:

```
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

model1 = DecisionTreeClassifier(random_state=1)
model1.fit(X_train, y_train)
y_pred1 = model1.predict(X_test)
```

In [315]:

```
print(classification_report(y_test, y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       341
           1       0.61      0.56      0.58        59

    accuracy                           0.88       400
   macro avg       0.77      0.75      0.76       400
weighted avg       0.88      0.88      0.88       400
```

In [ ]:

```

```

In [ ]:

```

```