In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
file = pd.read_csv("winequality-red.csv")
```

In [3]:
```
file
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99548	3.39	0.66	11.0	6

1599 rows × 12 columns

In [4]:
```
file.isnull().sum()
```

Out[4]:
```
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 [5]:
```
file.columns
```

Out[5]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [6]:
```
file.describe()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001897	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

In [7]:
```
file.quality = file['quality'].astype('float')
```

In [8]:
```
file.dtypes
```

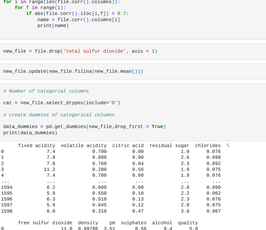
Out[8]:
```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                 float64
dtype: object
```

In [9]:
```
file.hist(bins=25 , figsize=(12,12))
plt.show()
```

Out[9]:

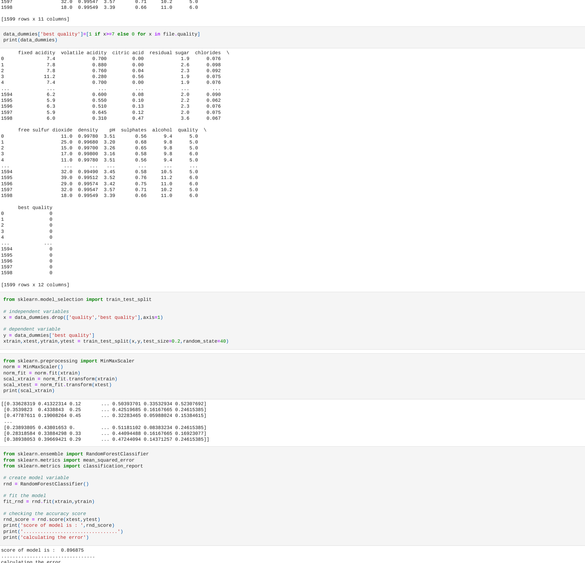
In [10]:
```
plt.figure(figsize=(12,6))
plt.bar(file['quality'],file['alcohol'],color="purple")
plt.xlabel('quality')
plt.ylabel('alcohol')
```

Out[10]:
```
Text(0, 0.5, 'alcohol')
```



In [11]:
```
plt.figure(figsize=(20,10),facecolor='white')
sns.heatmap(file.corr(),annot= True)
```

Out[11]:
```
<AxesSubplot:>
```



In [12]:
```
file.corr()
```

Out[12]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	-0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000

In [13]:
```
for i in range(len(file.corr().columns)):
    for f in range(i):
        if abs(file.corr().iloc[i,f]) > 0.7:
            name = file.corr().columns[i]
            print(name)
```

In [14]:
```
new_file = file.drop('total sulfur dioxide', axis = 1)
```

In [15]:
```
new_file.update(new_file.fillna(new_file.mean()))
```

In [16]:
```
# Number of categorial columns

cat = new_file.select_dtypes(include='O')

# create dummies of categorical columns

data_dummies = pd.get_dummies(new_file,drop_first = True)
print(data_dummies)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
...             ...               ...          ...             ...        ...
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  density    pH  sulphates  alcohol  quality
0                    11.0  0.99780  3.51       0.56      9.4        5
1                    25.0  0.99680  3.20       0.68      9.8        5
2                    15.0  0.99700  3.26       0.65      9.8        5
3                    17.0  0.99800  3.16       0.58      9.8        6
4                    11.0  0.99780  3.51       0.56      9.4        5
...                   ...      ...   ...        ...      ...      ...
1594                 32.0  0.99490  3.45       0.58     10.5        5
1595                 39.0  0.99512  3.52       0.76     11.2        6
1596                 29.0  0.99574  3.42       0.75     11.0        6
1597                 32.0  0.99547  3.57       0.71     10.2        5
1598                 18.0  0.99549  3.39       0.66     11.0        6

[1599 rows x 11 columns]
```

In [17]:
```
data_dummies['best quality']=[1 if x==7 else 0 for x in file.quality]
print(data_dummies)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
...             ...               ...          ...             ...        ...
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  density    pH  sulphates  alcohol  quality  \
0                    11.0  0.99780  3.51       0.56      9.4        5
1                    25.0  0.99680  3.20       0.68      9.8        5
2                    15.0  0.99700  3.26       0.65      9.8        5
3                    17.0  0.99800  3.16       0.58      9.8        6
4                    11.0  0.99780  3.51       0.56      9.4        5
...                   ...      ...   ...        ...      ...      ...
1594                 32.0  0.99490  3.45       0.58     10.5        5
1595                 39.0  0.99512  3.52       0.76     11.2        6
1596                 29.0  0.99574  3.42       0.75     11.0        6
1597                 32.0  0.99547  3.57       0.71     10.2        5
1598                 18.0  0.99549  3.39       0.66     11.0        6

      best quality
0                0
1                0
2                0
3                0
4                0
...            ...
1594             0
1595             0
1596             0
1597             0
1598             0

[1599 rows x 12 columns]
```

In [18]:
```
from sklearn.model_selection import train_test_split

# independent variables
x = data_dummies.drop(['quality','best quality'],axis=1)

# dependent variable
y = data_dummies['best quality']
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=40)
```

In [19]:
```
from sklearn.preprocessing import MinMaxScaler
norm = MinMaxScaler()
norm_fit = norm.fit(xtrain)
scal_xtrain = norm_fit.transform(xtrain)
scal_xtest = norm_fit.transform(xtest)
print(scal_xtrain)
```

```
[[0.33678319 0.41322314 0.12       ... 0.56393701 0.33532934 0.52307692]
 [0.35726223 0.4338843  0.25       ... 0.42519685 0.16167665 0.24615385]
 [0.47787611 0.19008264 0.45       ... 0.32283465 0.05988024 0.15384615]
 ...
 [0.23893805 0.43801653 0.         ... 0.51181102 0.08383234 0.24615385]
 [0.28318584 0.33884298 0.33       ... 0.44094488 0.16167665 0.16923077]
 [0.38938053 0.39669421 0.29       ... 0.47244094 0.14371257 0.24615385]]
```

In [20]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report

# create model variable
rnd = RandomForestClassifier()

# fit the model
fit_rnd = rnd.fit(xtrain,ytrain)

# checking the accuracy score
rnd_score = rnd.score(xtest,ytest)
print('score of model is : ',rnd_score)
print('.....................................')
print('calculating the error')

score of model is :  0.896875
.....................................
calculating the error
```

In [21]:
```
x_predict = list(rnd.predict(xtest))
df = {'predicted':x_predict,'original':ytest}
pd.DataFrame(df).head(15)
```

Out[21]:

	predicted	original
1035	0	1
49	0	0
709	0	0
538	0	1
665	0	0
990	0	0
398	0	0
1068	0	1
1155	0	0
468	0	0
1377	0	0
1345	0	0
494	0	0
1367	1	1
1307	0	0

In [ ]:
```

```