In [192]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

In [193]:

```
df=pd.read_csv('./winequality-total.csv')
df.head()
```

Out[193]:

	Color	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide
0	Red	7.4	0.70	0.00	1.9	0.076	11.0
1	Red	7.8	0.88	0.00	2.6	0.098	25.0
2	Red	7.8	0.76	0.04	2.3	0.092	15.0
3	Red	11.2	0.28	0.56	1.9	0.075	17.0
4	Red	7.4	0.70	0.00	1.9	0.076	11.0

In [194]:

```
red=df.loc[df.Color=='Red']
red.head()
```

Out[194]:

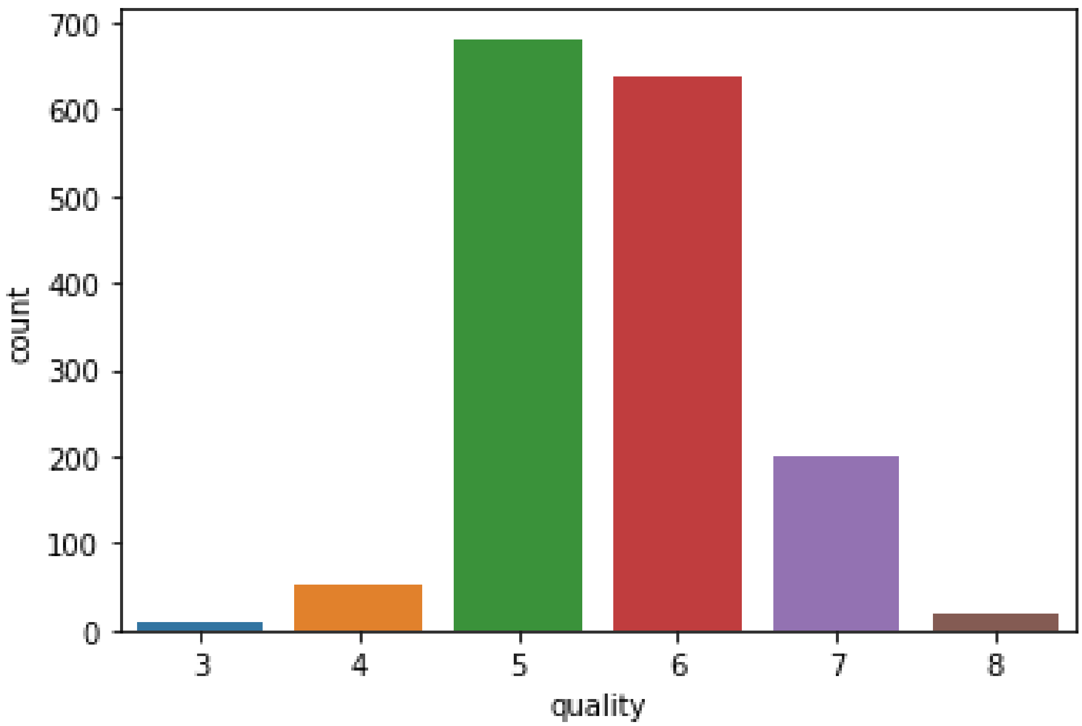
	Color	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide
0	Red	7.4	0.70	0.00	1.9	0.076	11.0
1	Red	7.8	0.88	0.00	2.6	0.098	25.0
2	Red	7.8	0.76	0.04	2.3	0.092	15.0
3	Red	11.2	0.28	0.56	1.9	0.075	17.0
4	Red	7.4	0.70	0.00	1.9	0.076	11.0

In [195]:

```
sns.countplot(x='quality',data=red)
```

Out[195]:

```
<matplotlib.axes._subplots.AxesSubplot at 0
x1a27f269d0>
```

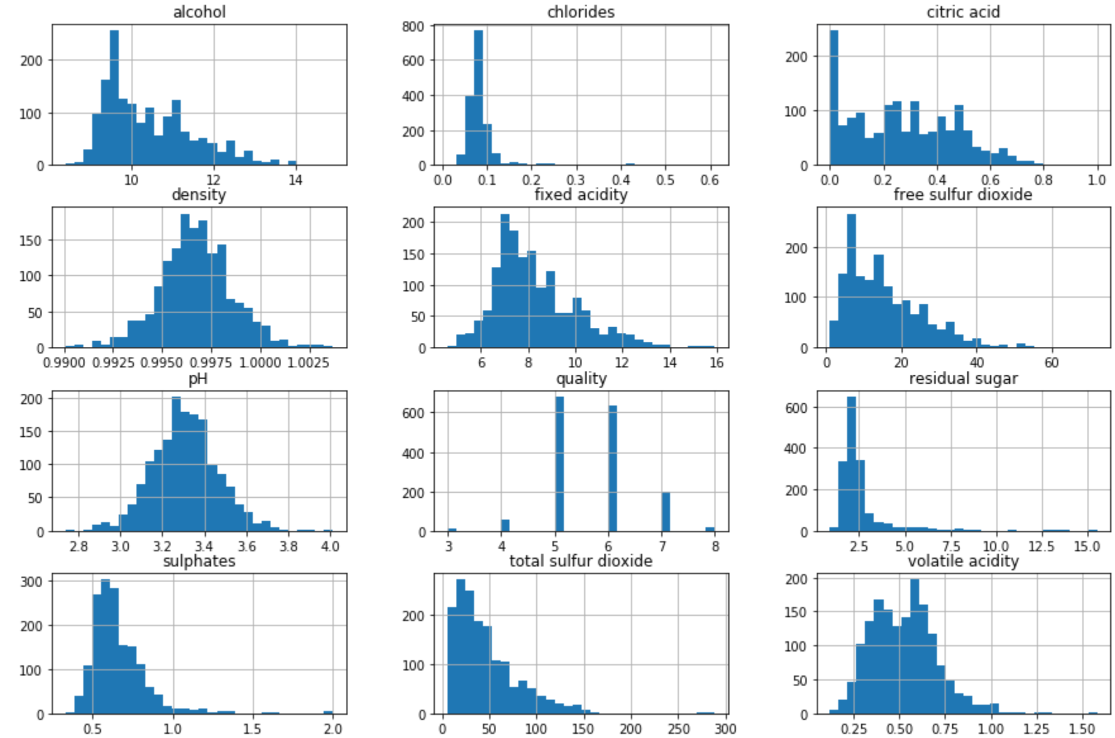


In [196]:

```
red.hist(bins=30, figsize=(15, 10))
```

Out[196]:

```
array([[<matplotlib.axes._subplots.AxesSubp
lot object at 0x1a27f2d150>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a28456fd0>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a28534810>],
       [<matplotlib.axes._subplots.AxesSubp
lot object at 0x1a28568f90>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a285ab850>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a285eabd0>],
       [<matplotlib.axes._subplots.AxesSubp
lot object at 0x1a2861f890>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a2865f0d0>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a2865fc10>],
       [<matplotlib.axes._subplots.AxesSubp
lot object at 0x1a2869e5d0>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a28a2c910>,
        <matplotlib.axes._subplots.AxesSubp
lot object at 0x1a28a68c90>]],
      dtype=object)
```

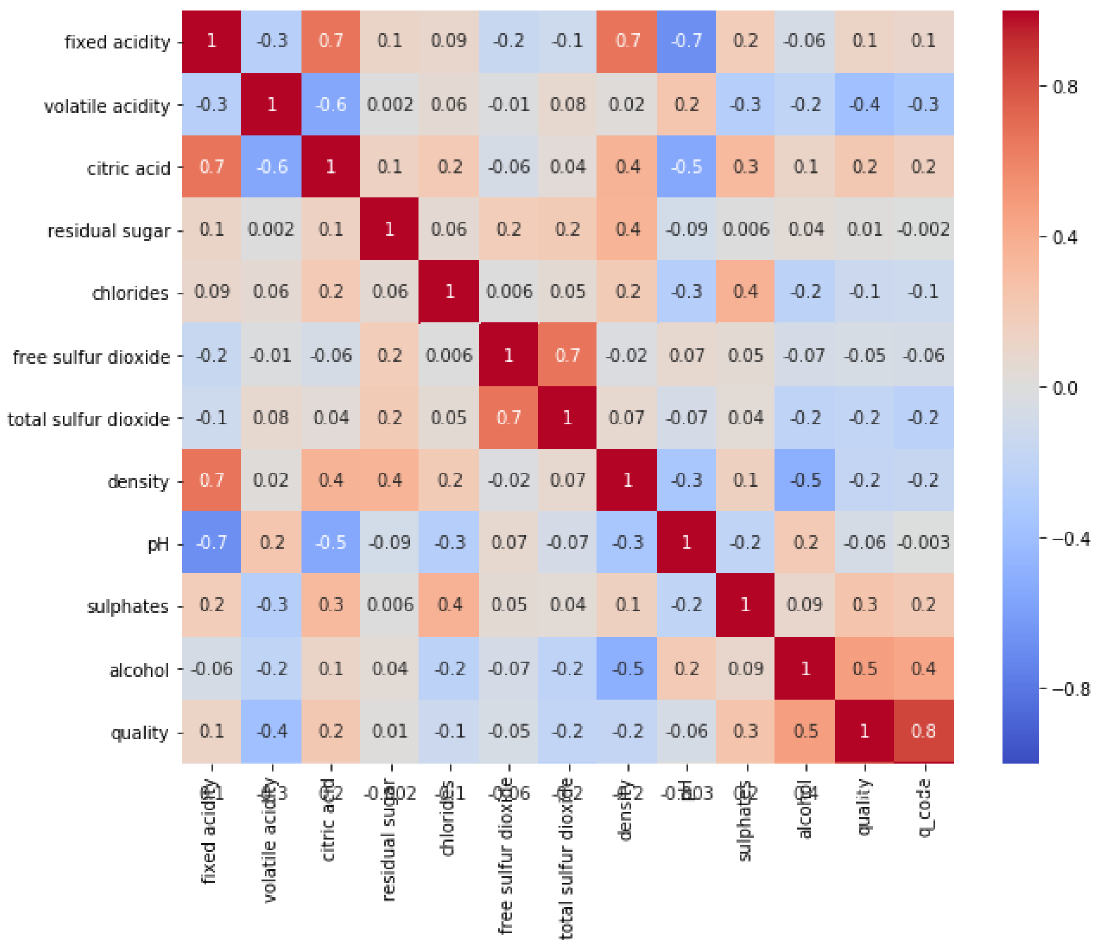

alcohol
chlorides
citric acid
density
fixed acidity
free sulfur dioxide
pH
quality
residual sugar
sulphates
total sulfur dioxide
volatile acidity

In [231]:

```
plt.figure(figsize=(10, 8))
ax=sns.heatmap(red.corr(), annot = True,vmin=-1, vmax=1,f
ax.get_ylim()
(5.5, 0.5)
ax.set_ylim(12, 0)
```

Out[231]:

```
(12, 0)
```



In [232]:

```
def qcode(quality):
    if quality>=6:
        return 1
    else:
        return 0
```

In [233]:

```
red['q_code']=red['quality'].apply(qcode)
```

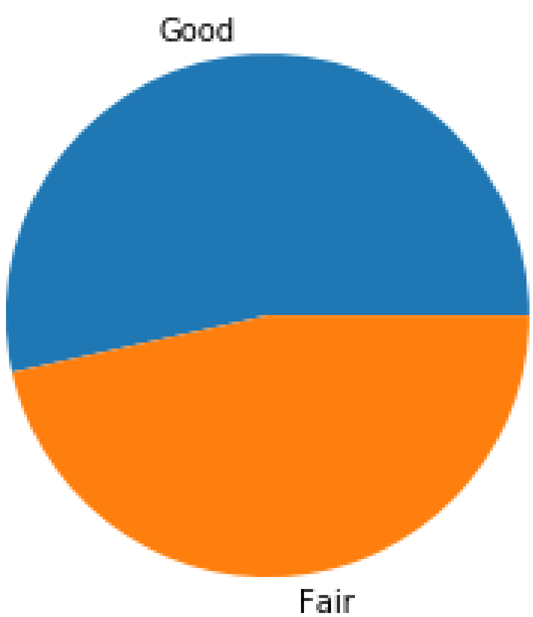
```
/Users/petergu/opt/anaconda3/lib/python3.7/
site-packages/ipykernel_launcher.py:1: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a
slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = v
alue instead

See the caveats in the documentation: htt
p://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-ver
sus-a-copy (http://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#retur
ning-a-view-versus-a-copy)
  """Entry point for launching an IPython k
ernel.
```

In [234]:

```
fig = plt.figure(figsize =(6, 4))
plt.pie(red['q_code'].value_counts(),labels=['Good','Fair
```

Out[234]:

```
([<matplotlib.patches.Wedge at 0x1a2a87a510
>,
  <matplotlib.patches.Wedge at 0x1a2a87a850
>],
 [Text(-0.11970888023405774, 1.093466864606
8377, 'Good'),
  Text(0.11970898261181326, -1.093466853398
8787, 'Fair')])
```



In [235]:

```
x=red.drop(['quality','Color','q_code'],axis=1).values
y=red['q_code'].values
```

In [236]:

```
x=StandardScaler().fit_transform(x)
```

In [237]:

```
from sklearn.decomposition import PCA
```

In [238]:

```
cov_mat = np.cov(x.transpose())
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)
print('\nEigenvalues \n%s' % eigen_vals)
print('\nEigenvectors \n%s' % eigen_vecs)
```

```
Eigenvalues
[3.10107182 1.92711489 1.55151379 1.2139917
5 0.95989238 0.05959558
 0.18144664 0.34485779 0.42322138 0.5841565
5 0.66002104]

Eigenvectors
[[ 0.48931422 -0.11050274 -0.12330157 -0.22
961737 -0.08261366 -0.63969145
  -0.24952314  0.19402091 -0.17759545 -0.35
022736  0.10147858]
 [-0.23858436  0.27493048 -0.44996253  0.07
895978  0.21873452 -0.0023886
   0.36592473 -0.1291103  -0.07877531 -0.53
37351   0.41144893]
 [ 0.46363166 -0.15179136  0.23824707 -0.07
941826 -0.05857268  0.0709103
   0.62167708 -0.38144967 -0.37751558  0.10
549701  0.06959338]
 [ 0.14610715  0.27208024  0.10128338 -0.37
279256  0.73214429 -0.18402996
   0.09287208  0.00752295  0.29984469  0.29
066341  0.04915555]
 [ 0.21224658  0.14805156 -0.09261383  0.66
619476  0.2465009  -0.05306532
  -0.21767112  0.11133867 -0.35700936  0.37
041337  0.30433857]
 [-0.03615752  0.51356681  0.42879287 -0.04
353782 -0.15915198  0.05142086
   0.24848326  0.63540522 -0.2047805  -0.11
```

```
659611 -0.01400021]
 [ 0.02357485  0.56948696  0.3224145  -0.03
457712 -0.22246456 -0.0687016
  -0.37075027 -0.59211589  0.01903597 -0.09
366237  0.13630755]
 [ 0.39535301  0.23357549 -0.33887135 -0.17
449976  0.15707671  0.5673319
  -0.23999012  0.02071868 -0.23922267 -0.17
048116 -0.3911523 ]
 [-0.43851962  0.00671079  0.05769735 -0.00
378775  0.26752977 -0.3407109
  -0.0109696  -0.16774589 -0.56139075 -0.02
513762 -0.52211645]
 [ 0.24292133 -0.03755392  0.27978615  0.55
087236  0.22596222 -0.06955538
   0.11232046 -0.05836706  0.37460432 -0.44
746911 -0.38126343]
 [-0.11323207 -0.38618096  0.47167322 -0.12
218109  0.35068141  0.31452591
  -0.3030145   0.03760311 -0.21762556 -0.32
76509   0.36164504]]
```

In [239]:

```
tot=sum(eigen_vals)
var_exp=[(i/tot)for i in sorted(eigen_vals, reverse=True)
cum_var_exp=np.cumsum(var_exp)
print('\nVariance explained\n%s'%var_exp)
print('\nCumulative variance explained\n%s'%cum_var_exp)
```

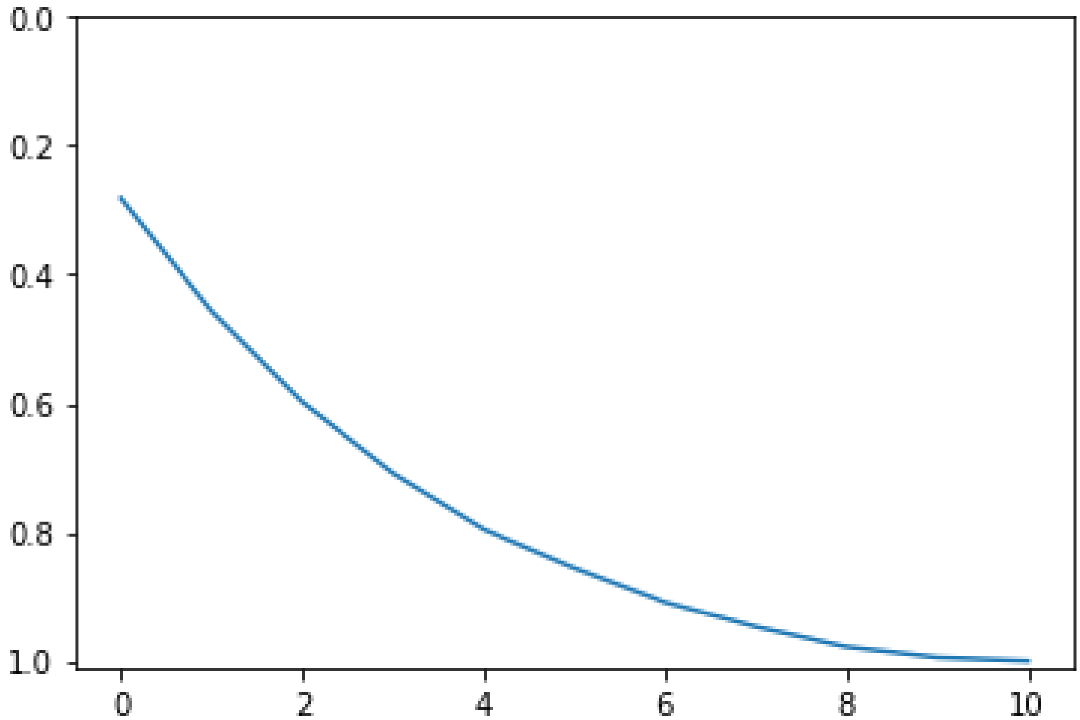
```
Variance explained
[0.28173931278845166, 0.17508269905390725,
0.1409584989800498, 0.11029386641613921, 0.
0872083701224993, 0.05996438771524651б, 0.0
53071929017561525, 0.038450609059750125, 0.
03133110152896331, 0.01648483332441732, 0.0
05414391993013964]

Cumulative variance explained
[0.28173931 0.45682201 0.59778051 0.7080743
8 0.79528275 0.85524714
 0.90831906 0.94676967 0.97810077 0.9945856
1 1.        ]
```

In [240]:

```
plt.plot(cum_var_exp)
plt.ylim(1.01, 0)
```

Out[240]:

```
(1.01, 0)
```



In [241]:

```
pca = PCA(n_components=5)
principalComponents = pca.fit_transform(x)
PCAdf = pd.DataFrame(data = principalComponents
, columns = ['PC1', 'PC2','PC3','PC4','PC5'])
```

In [258]:

```
print(PCAdf)
pca.explained_variance_ratio_
```

```
             PC1       PC2       PC3       PC
4       PC5
0    -1.619530  0.450950 -1.774454  0.04374
0  0.067014
1    -0.799170  1.856553 -0.911690  0.54806
6 -0.018392
2    -0.748479  0.882039 -1.171394  0.41102
1 -0.043531
3     2.357673 -0.269976  0.243489 -0.92845
0 -1.499149
4    -1.619530  0.450950 -1.774454  0.04374
0  0.067014
...        ...       ...       ...
...      ...
1594 -2.150500  0.814286  0.617063  0.40768
7 -0.240936
1595 -2.214496  0.893101  1.807402  0.41400
3  0.119592
1596 -1.456129  0.311746  1.124239  0.49187
7  0.193716
1597 -2.270518  0.979791  0.627965  0.63977
0  0.067735
1598 -0.426975 -0.536690  1.628955 -0.39171
6  0.450482

[1599 rows x 5 columns]
```

Out[258]:

```
array([0.28173931, 0.1750827 , 0.1409585 ,
0.11029387, 0.08720837])
```

In [243]:

```
pca_red=pd.concat([PCAdf,red['q_code']],axis=1)
```

In [244]:

```
print(pca_red)
```

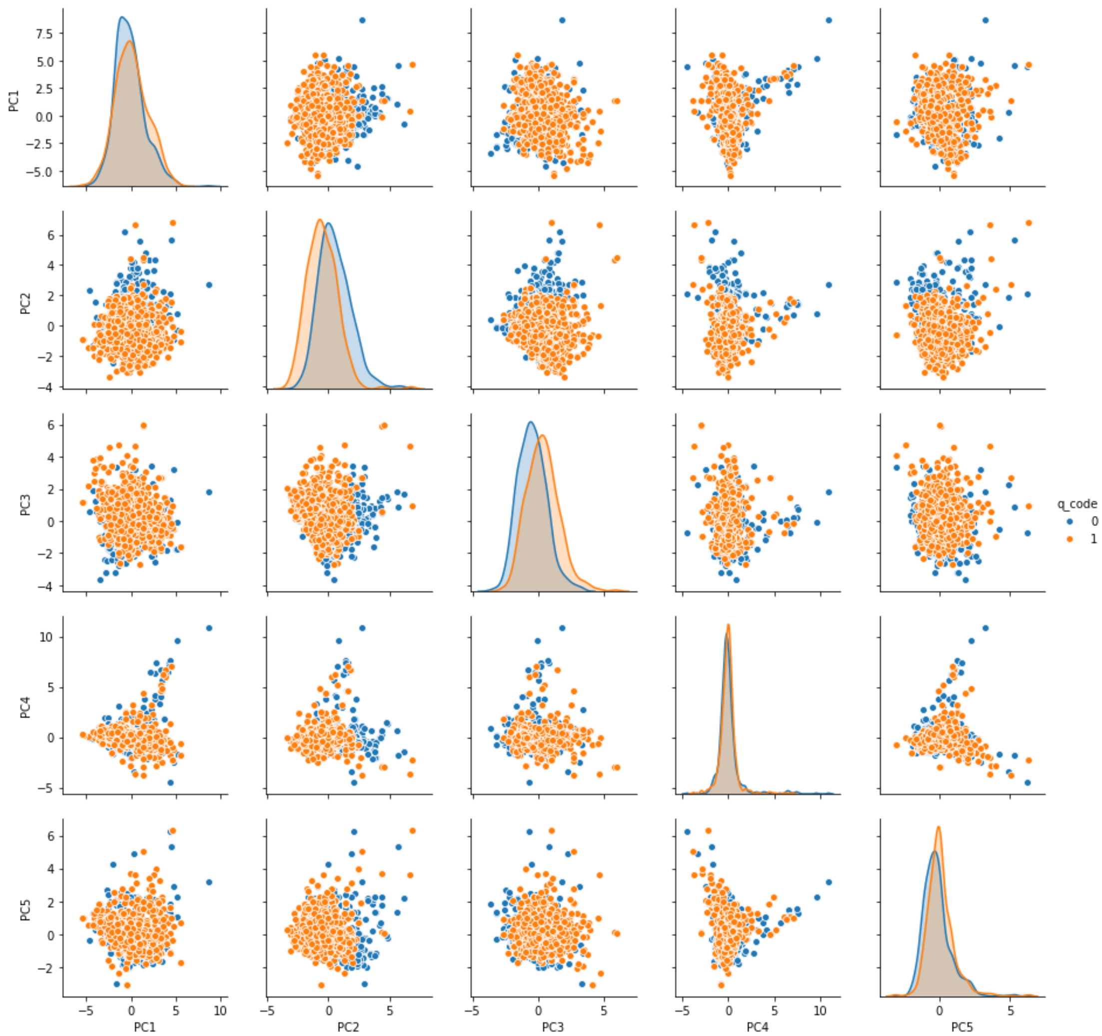
```
           PC1       PC2       PC3       PC
4      PC5  q_code
0    -1.619530  0.450950 -1.774454  0.04374
0  0.067014       0
1    -0.799170  1.856553 -0.911690  0.54806
6 -0.018392       0
2    -0.748479  0.882039 -1.171394  0.41102
1 -0.043531       0
3     2.357673 -0.269976  0.243489 -0.92845
0 -1.499149       1
4    -1.619530  0.450950 -1.774454  0.04374
0  0.067014       0
...        ...       ...       ...
...        ...     ...
1594 -2.150500  0.814286  0.617063  0.40768
7 -0.240936       0
1595 -2.214496  0.893101  1.807402  0.41400
3  0.119592       1
1596 -1.456129  0.311746  1.124239  0.49187
7  0.193716       1
1597 -2.270518  0.979791  0.627965  0.63977
0  0.067735       0
1598 -0.426975 -0.536690  1.628955 -0.39171
6  0.450482       1

[1599 rows x 6 columns]
```

In [259]:

```
sns.pairplot(vars=['PC1','PC2','PC3','PC4','PC5'], data=p
```

Out[259]:

```
&lt;seaborn.axisgrid.PairGrid at 0x1a2a055610&gt;
```



In [246]:

```
x=pca_red.drop(['q_code'],axis=1).values
y=pca_red['q_code'].values
```

In [247]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
```

In [248]:

```
from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)
pred_redlr=logisticRegr.predict(x_test)
```

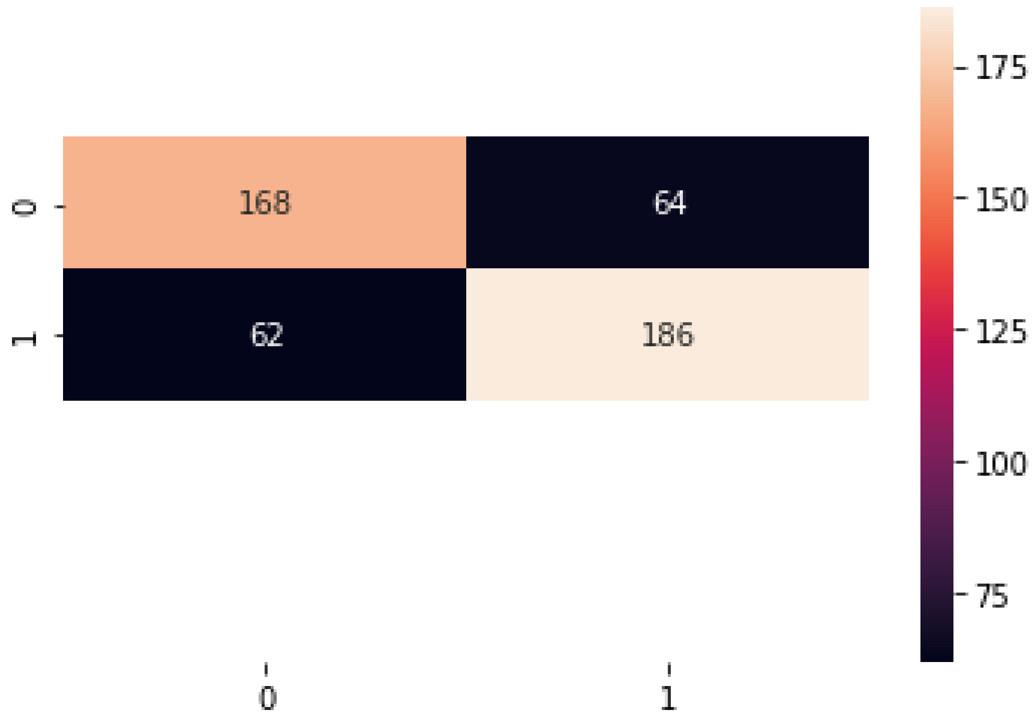
```
/Users/petergu/opt/anaconda3/lib/python3.7/
site-packages/sklearn/linear_model/logisti
c.py:432: FutureWarning: Default solver wil
l be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

In [249]:

```
from sklearn.metrics import confusion_matrix, classificat
logcon=confusion_matrix(pred_redlr, y_test)
ax=sns.heatmap(logcon, annot=True,fmt="d")
ax.get_ylim()
(5.5, 0.5)
ax.set_ylim(4, -1)
```

Out[249]:

```
(4, -1)
```



In [250]:

```
from sklearn.discriminant_analysis import LinearDiscrimina
```

In [251]:

```
lda=LinearDiscriminantAnalysis()
model=lda.fit(x_train, y_train.ravel())
pred_lday=model.predict(x_test)
print(model.priors_)
```

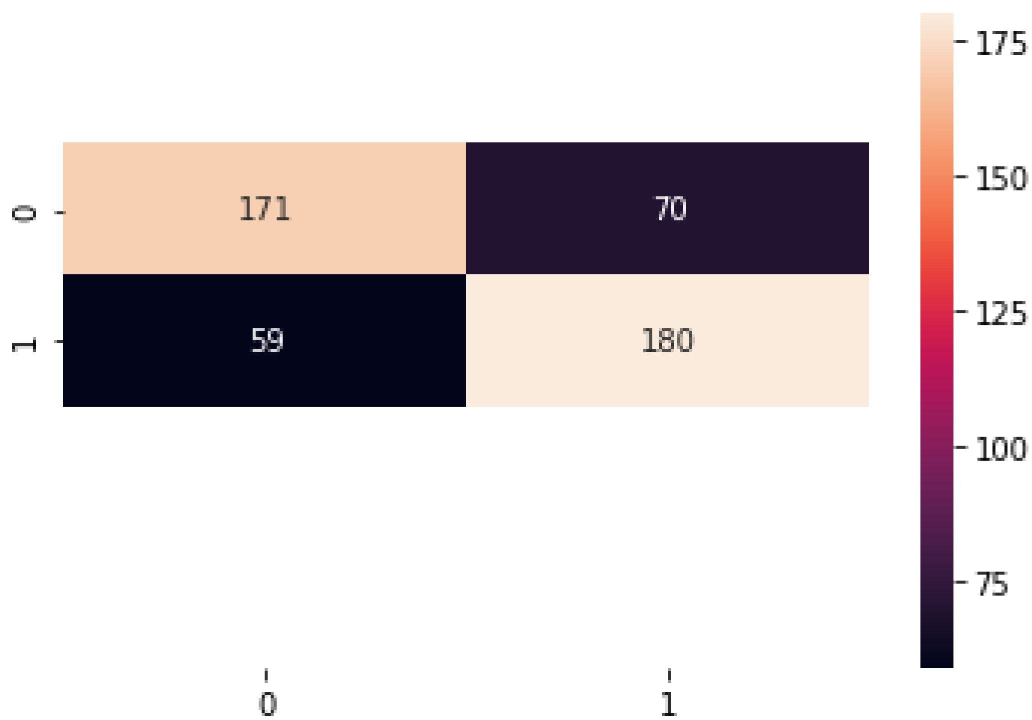
```
[0.4593387 0.5406613]
```

In [252]:

```
ldacon=confusion_matrix(pred_lday, y_test)
ax=sns.heatmap(ldacon, annot=True,fmt="d")
ax.get_ylim()
(5.5, 0.5)
ax.set_ylim(4, -1)
```

Out[252]:

```
(4, -1)
```



In [253]:

```
print(classification_report(y_test, pred_lday, digits=3))
```

```
              precision    recall  f1-score
support

           0      0.710     0.743     0.726
230
           1      0.753     0.720     0.736
250

    accuracy                          0.731
480
   macro avg      0.731     0.732     0.731
480
weighted avg      0.732     0.731     0.731
480
```

In [254]:

```
qda=QuadraticDiscriminantAnalysis()
model2=qda.fit(x, y.ravel())
pred_qday=model2.predict(x_test)
print(model.priors_)
```

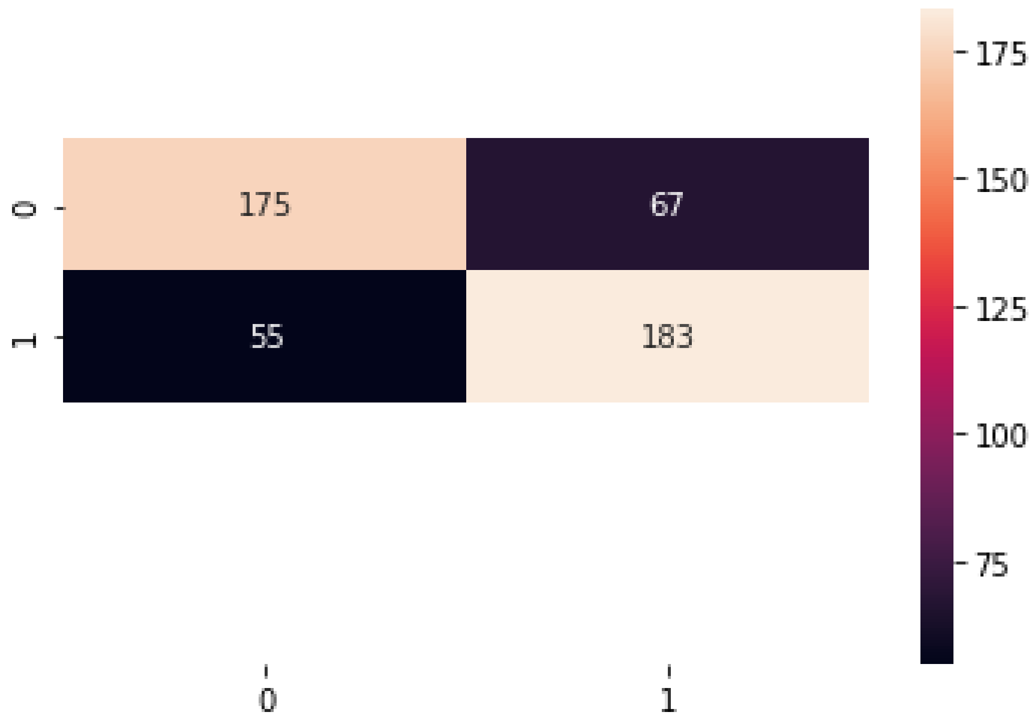
```
[0.4593387 0.5406613]
```

In [255]:

```
qdacon=confusion_matrix(pred_qday, y_test)
ax=sns.heatmap(qdacon, annot=True,fmt="d")
ax.get_ylim()
(5.5, 0.5)
ax.set_ylim(4, -1)
```

Out[255]:

```
(4, -1)
```



In [ ]:

```
logisticRegr.score(test_img, test_lbl)ß
```

In [ ]:

```

```

In [ ]:

```

```