Manaranjan Pradhan
manaranjan@enablecloud.com

*This notebook is given as part of **Data Science for everyone** workshop.*
(Forwarding this document to others is strictly prohibited.)

Classification - Supervised Learning

Predicting if a student will get admission or not

In [2]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
admission = pd.read_csv( "admission.csv" )
```

In [4]:

```
admission.head()
```

Out[4]:

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4

In [5]:

```
admission.columns = ["admit", "gre", "gpa", "ranking" ]
```

Does the ranking of the college impact the admissions and how much

In [6]:

```
pd.crosstab( admission.admit, admission.ranking )
```

Out[6]:

ranking	**1**	**2**	**3**	**4**
admit				
0	28	97	93	55
1	33	54	28	12

In [57]:

```
admit_by_rankings = pd.crosstab(
    admission.admit,
    admission.ranking ).apply( lambda x: x/x.sum(), axis = 0 )
```

In [8]:

```
admit_by_rankings
```

Out[8]:

ranking	**1**	**2**	**3**	**4**
admit				
0	0.459016	0.642384	0.768595	0.820896
1	0.540984	0.357616	0.231405	0.179104

In [9]:

```
admit_by_rankings = pd.DataFrame(
    admit_by_rankings.unstack() ).reset_index()
```

In [10]:

```
admit_by_rankings
```

Out[10]:

	ranking	admit	0
0	1	0	0.459016
1	1	1	0.540984
2	2	0	0.642384
3	2	1	0.357616
4	3	0	0.768595
5	3	1	0.231405
6	4	0	0.820896
7	4	1	0.179104

In [11]:

```
admit_by_rankings.columns = ["ranking", "admit", "total" ]
```

In [12]:

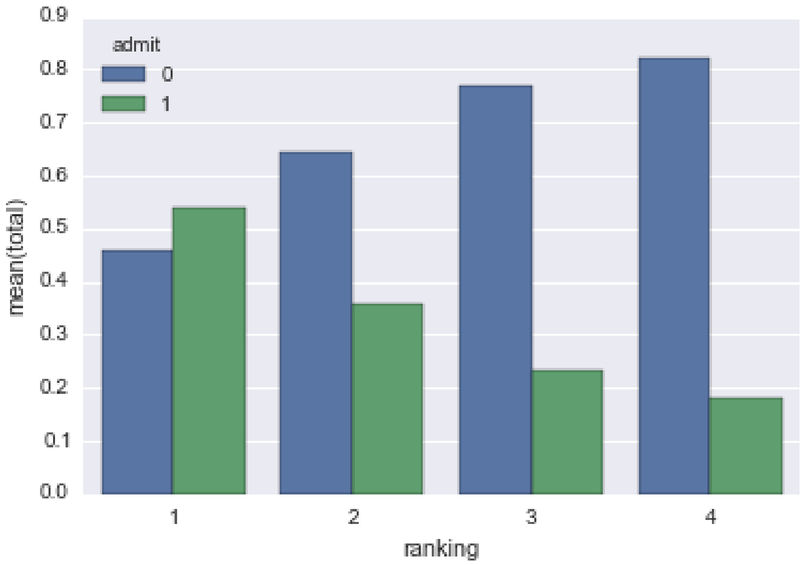
```
import matplotlib as plt
import seaborn as sn
%matplotlib inline
```

In [13]:

```
sn.barplot( admit_by_rankings.ranking, admit_by_rankings.total,
           hue = admit_by_rankings.admit )
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x91750f0>
```



Is the mean GRE and GPA score different for student who got admitted and who did not?

In [14]:

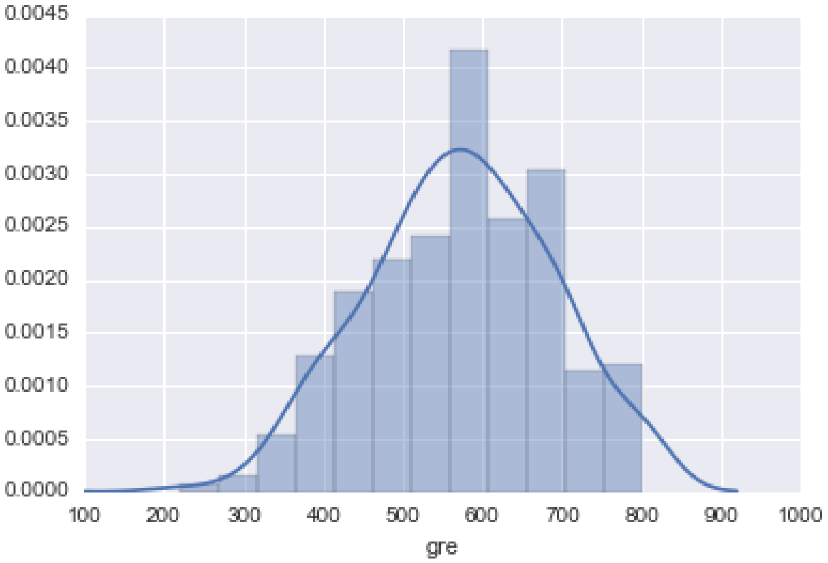
```
gre_0 = admission[admission.admit == 0]["gre"]
```

In [15]:

```
sn.distplot( gre_0 )
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x99034a8>
```



In [16]:

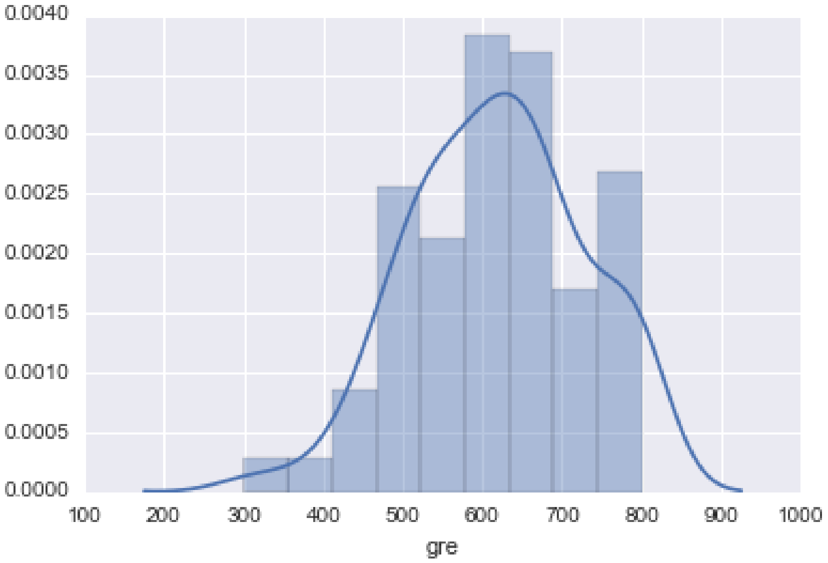
```
gre_1 = admission[admission.admit == 1]["gre"]
```

In [17]:

```
sn.distplot( gre_1 )
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x9996b70>
```

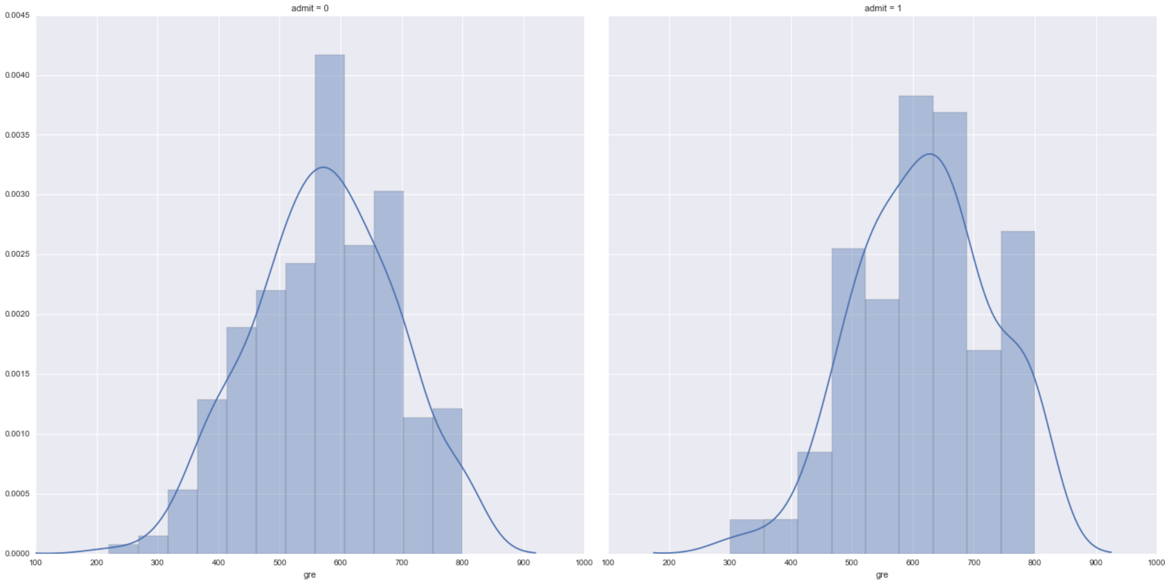


In [18]:

```
g = sn.FacetGrid(admission, col="admit", size = 10)
g.map(sn.distplot, "gre")
```

Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0xaa3f2e8>
```

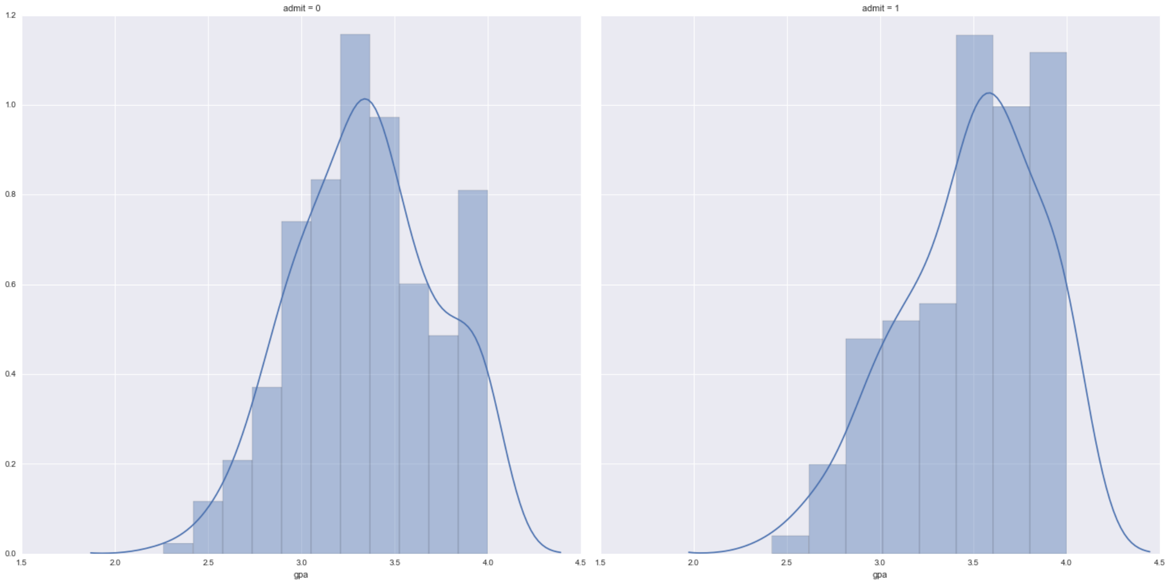


In [19]:

```
g = sn.FacetGrid(admission, col="admit", size = 10)
g.map(sn.distplot, "gpa")
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0xb048a58>
```

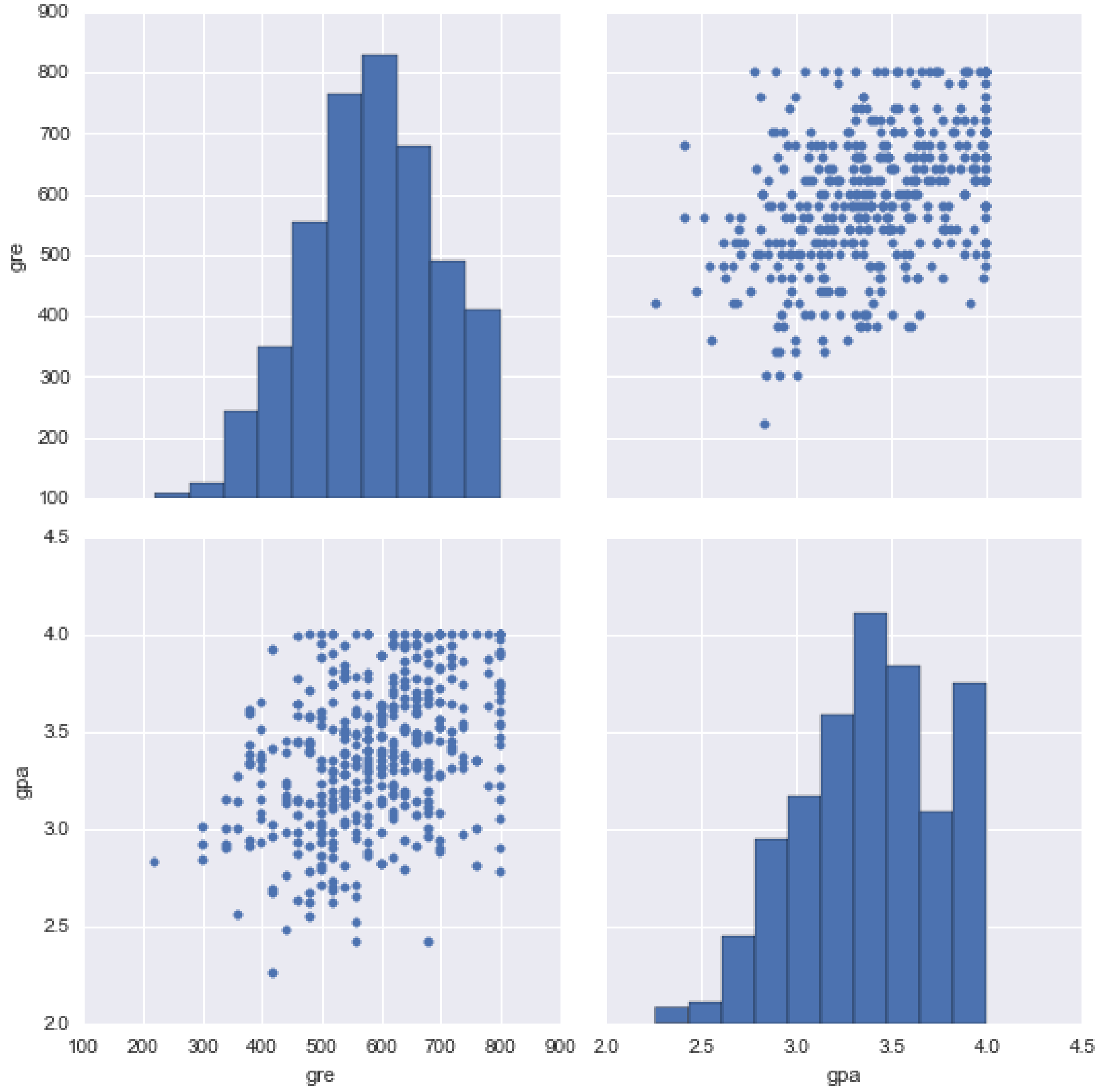


In [20]:

```
sn.pairplot( admission[["gre", "gpa"]], size = 4 )
```

Out[20]:

```
<seaborn.axisgrid.PairGrid at 0xb048d68>
```

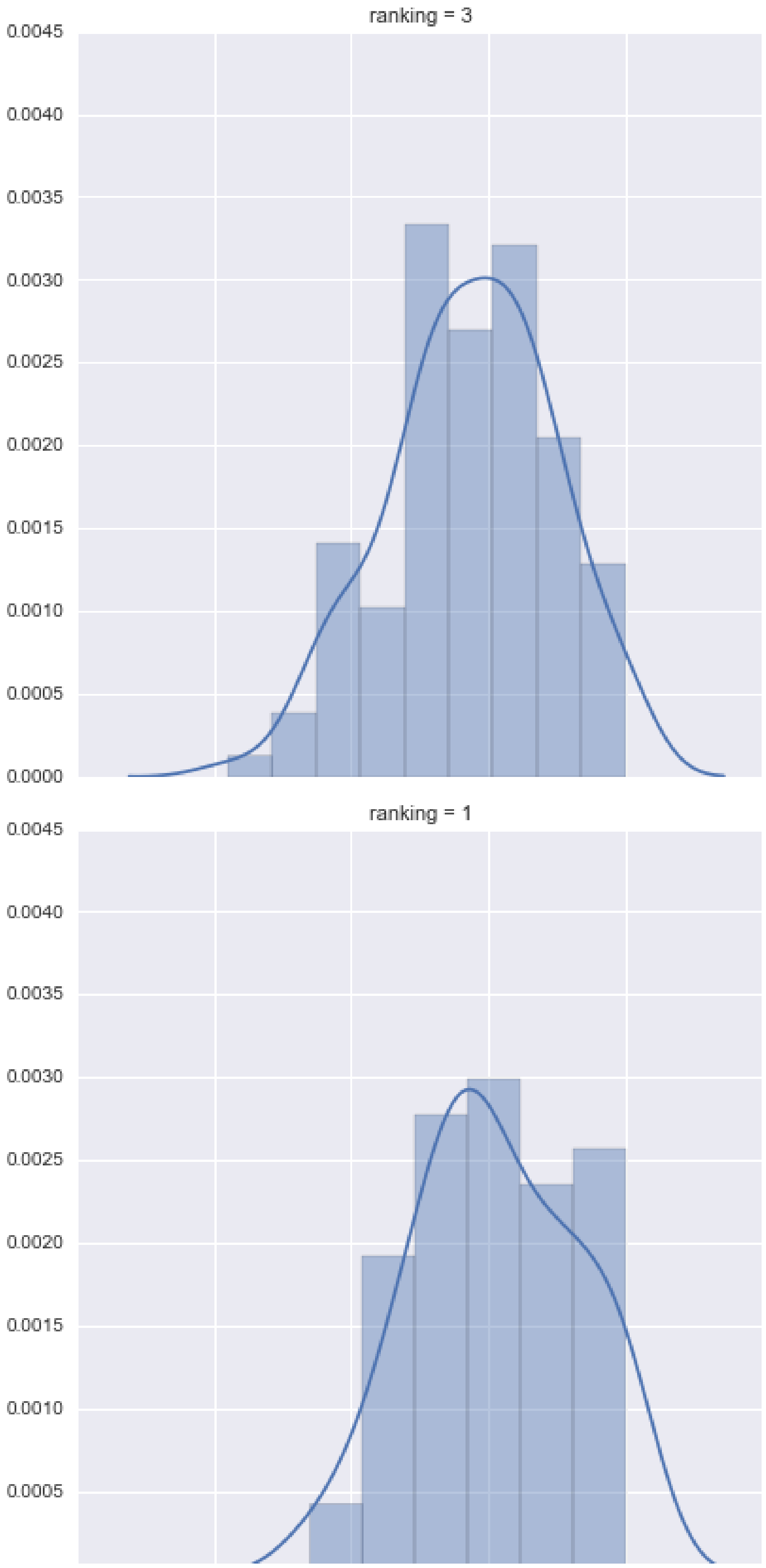


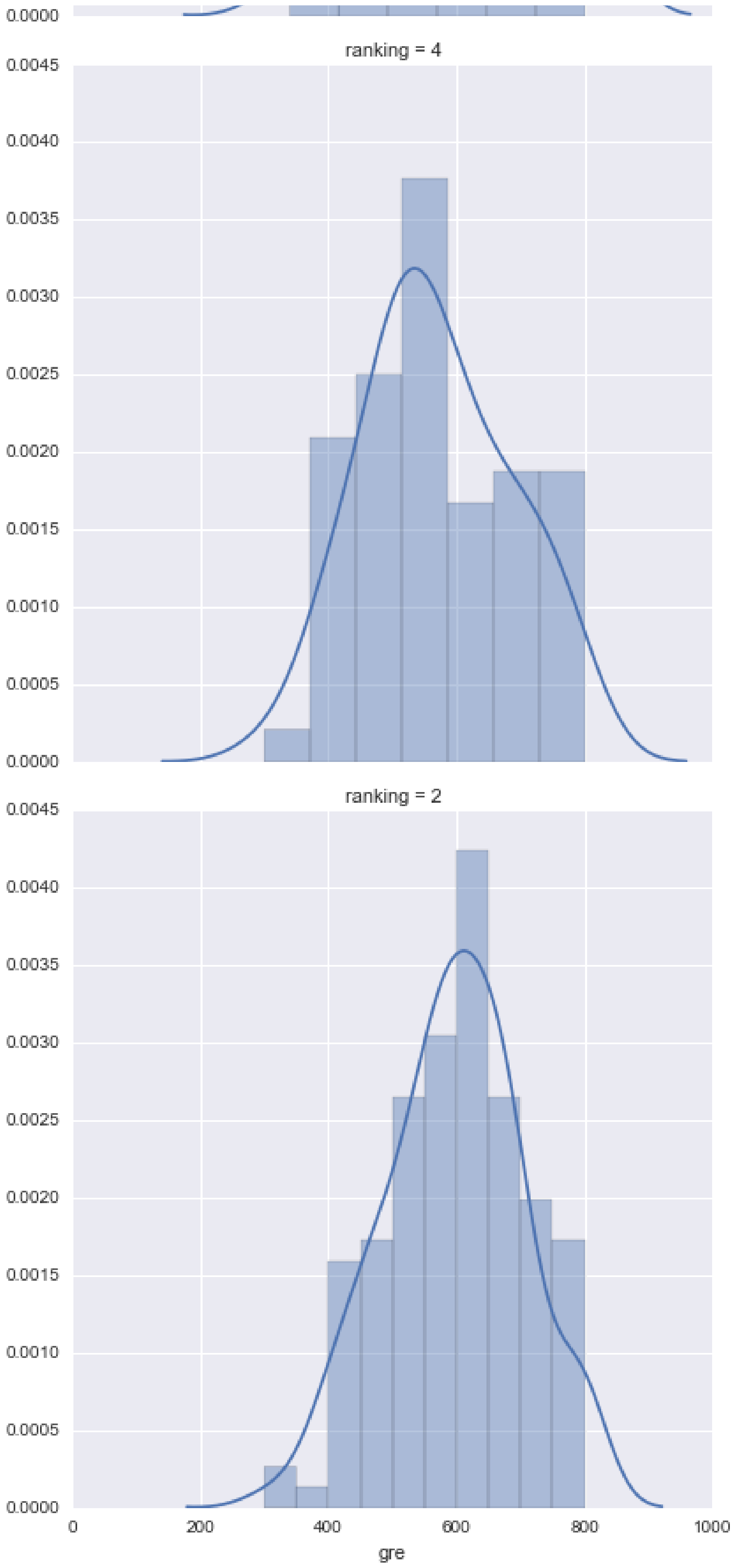
In [21]:

```
g = sn.FacetGrid(admission, row="ranking", size = 6)
g.map(sn.distplot, "gre")
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0xbbb62e8>
```



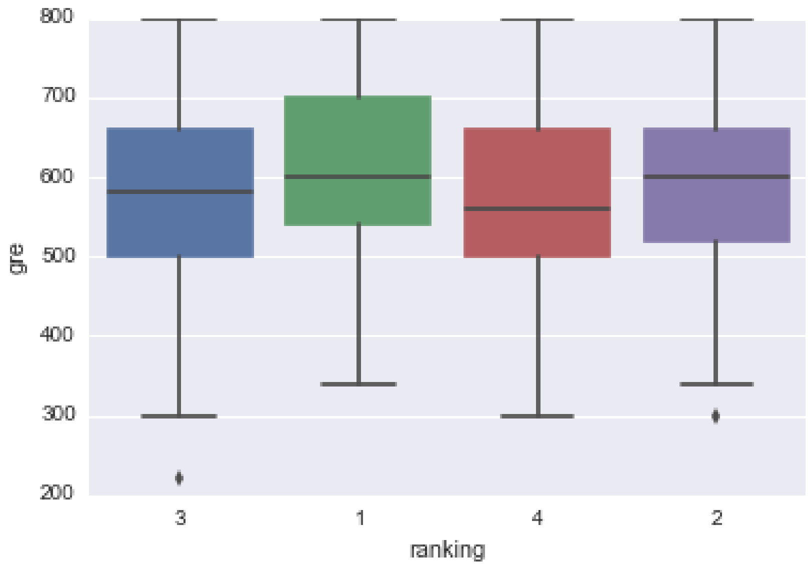

0.0000
ranking = 4
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
ranking = 2
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
200
400
600
800
1000
gre

In [22]:

```
sn.boxplot( "ranking", "gre", data = admission )
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xc052a20>
```



Building a Classification Model

Convert the categorical variables into dummy variables

In [23]:

```
def create_dummies( df, colname ):
    col_dummies = pd.get_dummies(df[colname], prefix=colname)
    col_dummies.drop(col_dummies.columns[0], axis=1, inplace=True)
    df = pd.concat([df, col_dummies], axis=1)
    df.drop( colname, axis = 1, inplace = True )
    return df
```

In [24]:

```
admission_new = create_dummies( admission, "ranking" )
```

In [25]:

```
admission_new.head()
```

Out[25]:

	admit	gre	gpa	ranking_2	ranking_3	ranking_4
0	0	380	3.61	0	1	0
1	1	660	3.67	0	1	0
2	1	800	4.00	0	0	0
3	1	640	3.19	0	0	1
4	0	520	2.93	0	0	1

In [26]:

```
from sklearn.linear_model import LogisticRegression
```

In [27]:

```
logreg = LogisticRegression()
```

In [28]:

```
admission_new.columns
```

Out[28]:

```
Index(['admit', 'gre', 'gpa', 'ranking_2', 'ranking_3', 'ranking_4'], d
type='object')
```

In [29]:

```
feature_cols = ['gre', 'gpa', 'ranking_2', 'ranking_3', 'ranking_4']
```

In [30]:

```
logreg.fit( admission_new[feature_cols], admission_new.admit )
```

Out[30]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.00
01,
          verbose=0)
```

In [31]:

```
list( zip( feature_cols, logreg.coef_ ) )
```

Out[31]:

```
[('gre',
  array([ 0.00181821,  0.24353836, -0.60583825, -1.1749243 , -1.3783986
1]))]
```

In [32]:

```
logreg.coef_
```

Out[32]:

```
array([[ 0.00181821,  0.24353836, -0.60583825, -1.1749243 , -1.3783986
1]])
```

In [33]:

```
logreg.intercept_
```

Out[33]:

```
array([-1.8727875])
```

In [34]:

```
admission_new["predicted_class"] = logreg.predict( admission_new[feature_cols] )
```

In [35]:

```
admission_new = pd.concat( [admission_new,
                            pd.DataFrame(
            logreg.predict_proba( admission_new[feature_cols] ) )], axis = 1 )
```

In [36]:

```
admission_new.head()
```

Out[36]:

	admit	gre	gpa	ranking_2	ranking_3	ranking_4	predicted_class	0
0	0	380	3.61	0	1	0	0	0.814212
1	1	660	3.67	0	1	0	0	0.721900
2	1	800	4.00	0	0	0	1	0.364488
3	1	640	3.19	0	0	1	0	0.787621
4	0	520	2.93	0	0	1	0	0.830918

Evaluating the model - Confusion Matrix

In [37]:

```
from sklearn import metrics
```

In [39]:

```
cm = metrics.confusion_matrix( admission_new.admit,
                              admission_new.predicted_class )
```

In [40]:

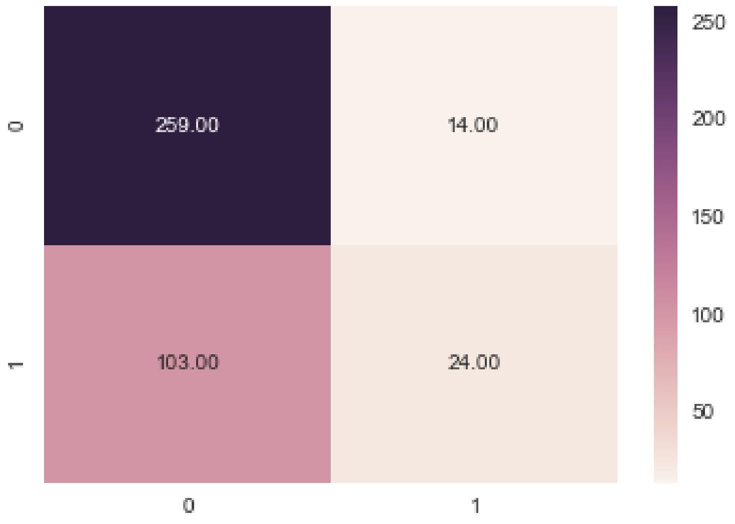
```
cm
```

Out[40]:

```
array([[259,  14],
       [103,  24]])
```

In [41]:

```
sn.heatmap(cm, annot=True,  fmt='.2f' );
```



Accuracy Score

In [42]:

```
score = metrics.accuracy_score( admission_new.admit,
                               admission_new.predicted_class )
```

In [45]:

```
score
```

Out[45]:

```
0.70750000000000002
```

ROC Curve

http://blog.yhathq.com/posts/roc-curves.html (http://blog.yhathq.com/posts/roc-curves.html)

In [46]:

```
auc_score = metrics.roc_auc_score( admission_new.admit,
                                  admission_new.predicted_class )
```

In [47]:

```
auc_score
```

Out[47]:

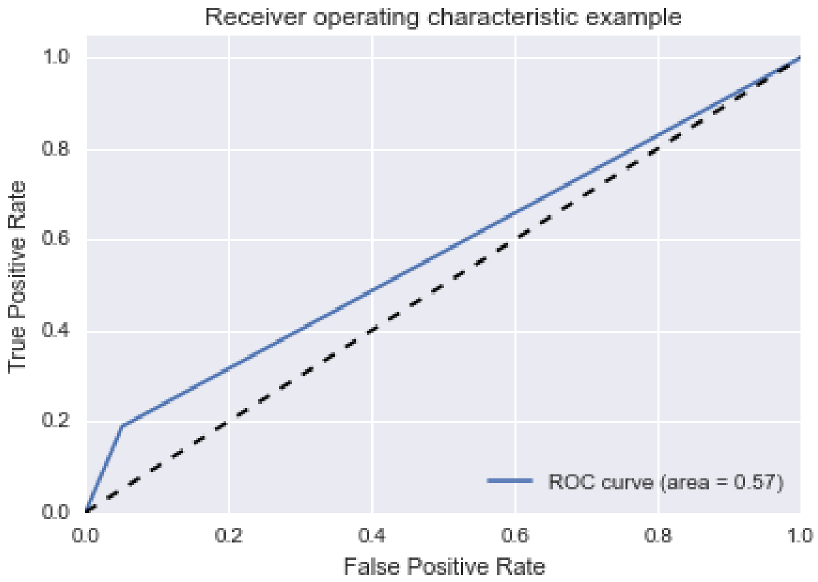
```
0.56884716333535223
```

In [49]:

```
fpr, tpr, _ = metrics.roc_curve( admission_new.admit,
                                admission_new.predicted_class )
```

In [50]:

```
import matplotlib.pyplot as pyplt
pyplt.figure()
pyplt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
pyplt.plot([0, 1], [0, 1], 'k--')
pyplt.xlim([0.0, 1.0])
pyplt.ylim([0.0, 1.05])
pyplt.xlabel('False Positive Rate')
pyplt.ylabel('True Positive Rate')
pyplt.title('Receiver operating characteristic example')
pyplt.legend(loc="lower right")
pyplt.show()
```



In [52]:

```
print( metrics.classification_report( admission_new.admit,
                                      admission_new.predicted_class ) )
```

```
             precision    recall  f1-score   support

          0       0.72      0.95      0.82       273
          1       0.63      0.19      0.29       127

avg / total       0.69      0.71      0.65       400
```

Cross validating the model

In [53]:

```
from sklearn.cross_validation import cross_val_score
```

In [54]:

```
logreg = LogisticRegression()
X = admission_new[feature_cols]
y = admission_new.admit
scores = cross_val_score(logreg, X, y, cv=10, scoring='accuracy')
```

In [55]:

```
scores
```

Out[55]:

```
array([ 0.82926829,  0.6097561 ,  0.73170732,  0.675     ,  0.725     ,
        0.675     ,  0.7       ,  0.66666667,  0.74358974,  0.6666666
7])
```

In [56]:

```
scores.mean()
```

Out[56]:

```
0.70226547842401499
```

Make note of lessons learnt in this exercise