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

from sklearn.metrics import mean_squared_error, accuracy_score, f1_score, confusio
n_matrix, roc_auc_score, recall_score, classification_report
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, learning_curve, KFold, cross
_val_score, cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import linear_model
```

In [20]:
```
#load the data
student_data = pd.read_csv('/Users/ilmasheriff/Desktop/cleaned_dataset.csv',header
= 0)
```

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

Out[3]:

	Student Number2	Ending YRTR	Start YRTR	Ending Term	Ending Term Type	Admission Category	Male	Female	Gender	Student of Color Value	...	Transfer Time in Years	Transf Tir Catego
0	0	0	8	0	0	1	0	1	0	0	...	1	
1	1	6	34	3	0	0	0	1	0	0	...	1	
2	2	0	25	0	0	0	1	0	1	0	...	1	
3	3	1	26	5	1	0	0	1	0	0	...	1	
4	4	1	4	5	1	0	0	1	0	0	...	1	

5 rows × 112 columns

In [4]:
```
print(student_data.shape)
student_data.dtypes
```

```
(21014, 112)
```

Out[4]:
```
Student Number2                        int64
Ending YRTR                            int64
Start YRTR                             int64
Ending Term                            int64
Ending Term Type                       int64
                                        ...
Transfer Instituion In/Out State       int64
Transfer Instituion Public/Private     int64
Transfer ZIP                           int64
Transfer Region                        int64
Transfer = Home Region Value           int64
Length: 112, dtype: object
```

In [5]:
```
target = student_data["Student Outcome"]
```

In [6]:
```
target.head()
```

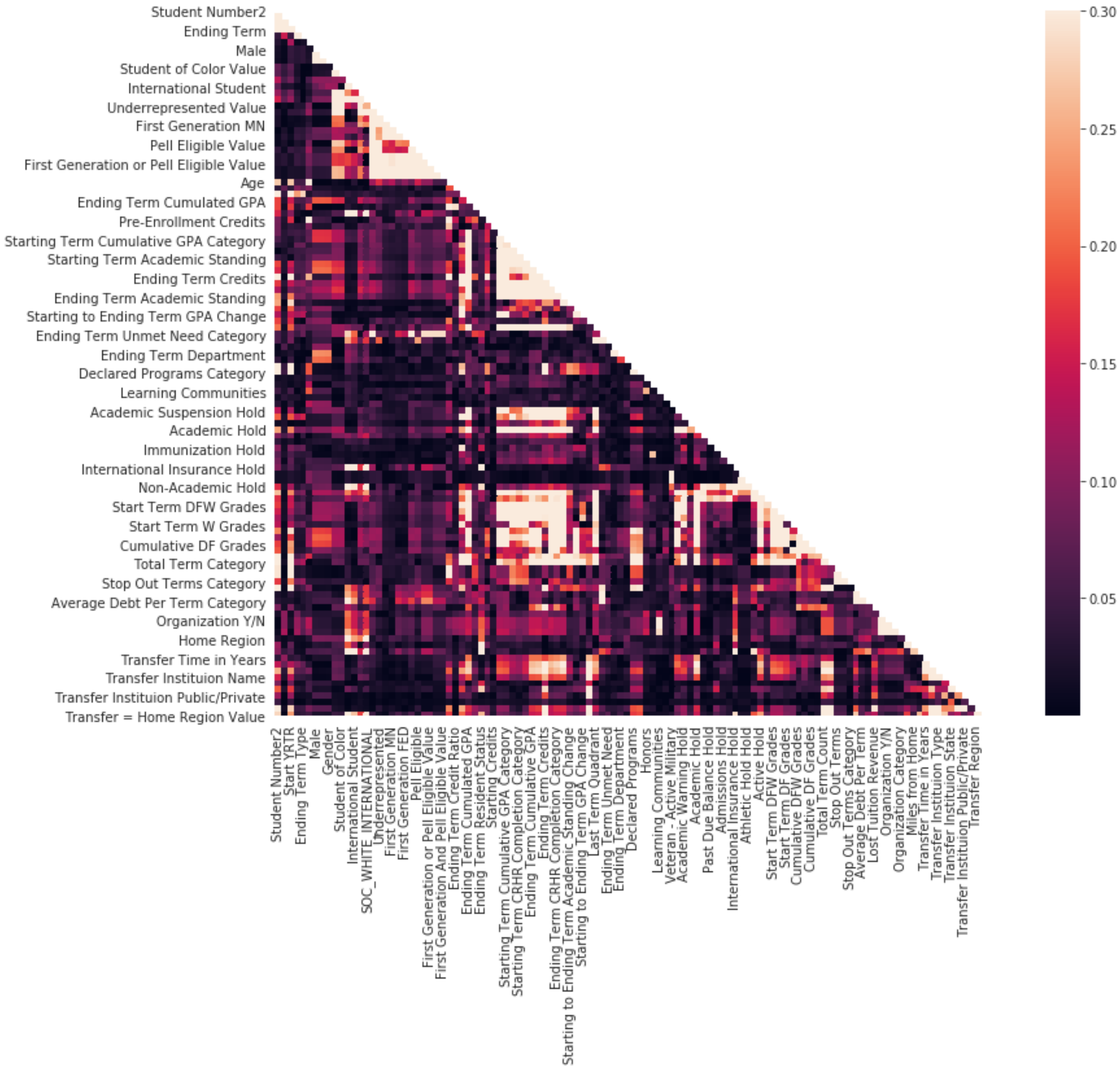
Out[6]:
```
0    0
1    0
2    0
3    0
4    1
Name: Student Outcome, dtype: int64
```

In [7]:
```
corr = student_data.corr(method='pearson').abs()
print(corr)
```

```
                                     Student Number2  Ending YRTR  Start YRTR  \
Student Number2                             1.000000     0.561451    0.852468
Ending YRTR                                 0.561451     1.000000    0.561209
Start YRTR                                  0.852468     0.561209    1.000000
Ending Term                                 0.323742     0.637992    0.328152
Ending Term Type                            0.040181     0.150363    0.046004
...                                              ...          ...         ...
Transfer Instituion In/Out State            0.195148     0.025712    0.173453
Transfer Instituion Public/Private          0.116781     0.002932    0.113484
Transfer ZIP                                0.046341     0.002231    0.036390
Transfer Region                             0.299153     0.015072    0.279722
Transfer = Home Region Value                0.283298     0.002774    0.238203

                                     Ending Term  Ending Term Type  \
Student Number2                         0.323742          0.040181
Ending YRTR                             0.637992          0.150363
Start YRTR                              0.328152          0.046004
Ending Term                             1.000000          0.857219
Ending Term Type                        0.857219          1.000000
...                                          ...               ...
Transfer Instituion In/Out State        0.031108          0.022744
Transfer Instituion Public/Private      0.011002          0.012164
Transfer ZIP                            0.000689          0.002377
Transfer Region                         0.043475          0.045737
Transfer = Home Region Value            0.031723          0.038873

                                     Admission Category      Male    Female  \
Student Number2                                0.029031  0.017836  0.015533
Ending YRTR                                    0.011128  0.006356  0.005976
Start YRTR                                     0.126573  0.042617  0.041310
Ending Term                                    0.008552  0.031806  0.031657
Ending Term Type                               0.018421  0.036584  0.036647
...                                                 ...       ...       ...
Transfer Instituion In/Out State               0.055574  0.019672  0.020069
Transfer Instituion Public/Private             0.014161  0.060708  0.060190
Transfer ZIP                                   0.017543  0.004467  0.004171
Transfer Region                                0.089288  0.069365  0.070033
Transfer = Home Region Value                   0.049062  0.020704  0.020043

                                       Gender  Student of Color Value  ...  \
Student Number2                      0.013128               0.077026  ...
Ending YRTR                          0.005553               0.044031  ...
Start YRTR                           0.039695               0.070896  ...
Ending Term                          0.031265               0.022199  ...
Ending Term Type                     0.036427               0.000945  ...
...                                       ...                    ...  ...
Transfer Instituion In/Out State     0.020308               0.005601  ...
Transfer Instituion Public/Private   0.059212               0.013586  ...
Transfer ZIP                         0.003846               0.014866  ...
Transfer Region                      0.070155               0.021465  ...
Transfer = Home Region Value         0.019233               0.046725  ...

                                     Transfer Time in Years  \
Student Number2                                    0.036595
Ending YRTR                                        0.108032
Start YRTR                                         0.032916
Ending Term                                        0.042966
Ending Term Type                                   0.017082
...                                                     ...
Transfer Instituion In/Out State                   0.166497
Transfer Instituion Public/Private                 0.122214
Transfer ZIP                                       0.016602
Transfer Region                                    0.237014
Transfer = Home Region Value                       0.169980
```

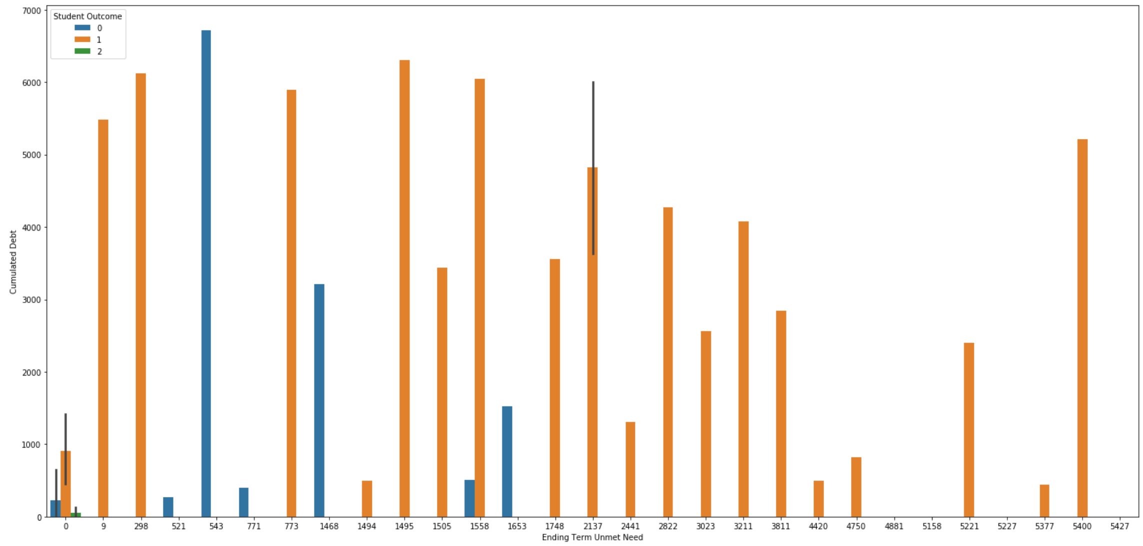
In [8]:

```
# corr = np.corrcoef(np.random.randn(10, 200))
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(16, 10))
    ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True)
```



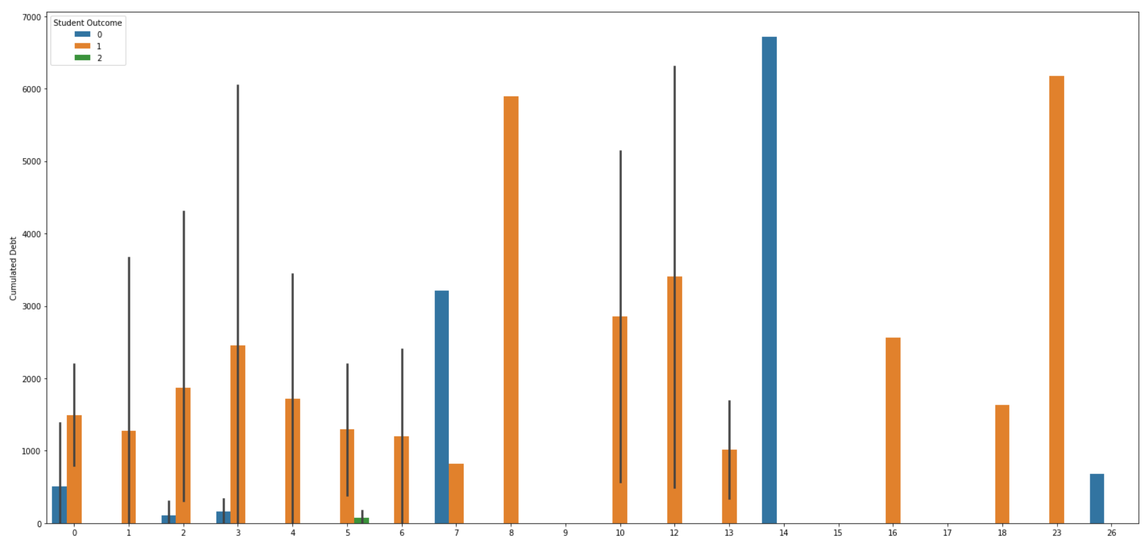
In [9]:
```
plt.figure(figsize=(25,12))

sns.barplot(data=student_data.head(100), x= "Ending Term Unmet Need", y="Cumulated Debt", hue="Student Outcome")
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x122cb20d0>



In [10]:
```
plt.figure(figsize=(25,12))

sns.barplot(data=student_data.head(100), x= "Cumulative DFW Grades", y="Cumulated Debt", hue="Student Outcome")
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1263493d0>



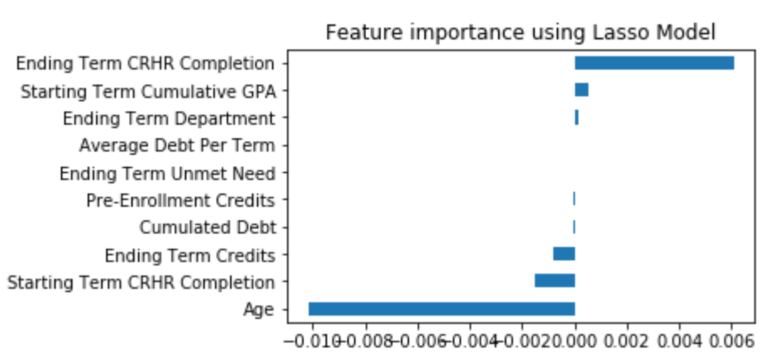
In [21]:
```
del student_data["Student Outcome"]
del student_data["Transfer Instituion State"]
del student_data["Transfer Instituion Name"]
del student_data["Transfer Region"]
del student_data['Transfer = Home Region Value']
del student_data["Transfer ZIP"]
del student_data["Student Number2"]
del student_data["Lost Tuition Revenue"]
del student_data["Miles from Home"]
del student_data["Ending Term Cumulated GPA"]
del student_data["Ending Term Cumulative GPA"]
del student_data["Transfer Time in Years"]
reg = linear_model.LassoCV()
reg.fit(student_data, target)
print("Best alpha using built-in LassoCV: %f" % reg.alpha_)
print("Best score using built-in LassoCV: %f" %reg.score(student_data,target))
coef = pd.Series(reg.coef_, index = student_data.columns)

print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other
" +  str(sum(coef == 0)) + " variables")
```

```
Best alpha using built-in LassoCV: 0.158401
Best score using built-in LassoCV: 0.112725
Lasso picked 10 variables and eliminated the other 90 variables
```

In [55]:
```
imp_coef = coef.sort_values()
plt.rcParams['figure.figsize'] = (5.0, 3.0)
# imp_coef.plot(kind = "barh")
# print(imp_coef)
#Selecting highly correlated features
relevant_features = imp_coef[imp_coef!=0]
print(relevant_features)
relevant_features.plot(kind = "barh")
plt.title("Feature importance using Lasso Model")
```

```
Age                              -0.010149
Starting Term CRHR Completion    -0.001535
Ending Term Credits              -0.000777
Cumulated Debt                   -0.000056
Pre-Enrollment Credits           -0.000053
Ending Term Unmet Need            0.000008
Average Debt Per Term             0.000044
Ending Term Department            0.000180
Starting Term Cumulative GPA      0.000550
Ending Term CRHR Completion       0.006087
dtype: float64
```

Out[55]: Text(0.5, 1.0, 'Feature importance using Lasso Model')



In [23]:
```
data = student_data[[
'Age',
"Starting Term CRHR Completion",
"Pre-Enrollment Credits",
"Ending Term Credits",
    "Cumulated Debt",
    'Ending Term Department',
"Ending Term Unmet Need",
"Average Debt Per Term",
"Starting Term Cumulative GPA",
"Ending Term CRHR Completion"
    ]]
data.head()
```

Out[23]:

	Age	Starting Term CRHR Completion	Pre-Enrollment Credits	Ending Term Credits	Cumulated Debt	Ending Term Department	Ending Term Unmet Need	Average Debt Per Term	Starting Term Cumulative GPA	Ending Term CRHR Completion
0	45	99	15	55	0	21	0	0	379	84
1	53	99	0	13	0	49	0	0	181	97
2	33	93	0	548	0	55	0	0	325	91
3	35	99	0	96	0	18	0	0	379	97
4	31	99	0	252	1241	43	0	271	260	97

In [24]:
```
data.corr(method="pearson")
```

Out[24]:

	Age	Starting Term CRHR Completion	Pre-Enrollment Credits	Ending Term Credits	Cumulated Debt	Ending Term Department	Ending Term Unmet Need	Average Debt Per Term	S Cum
Age	1.000000	-0.041120	0.369302	0.408657	0.049916	0.045945	-0.097780	-0.082673	0.
Starting Term CRHR Completion	-0.041120	1.000000	0.081141	0.178570	0.096770	-0.023734	0.006079	0.026575	0.
Pre-Enrollment Credits	0.369302	0.081141	1.000000	0.377343	-0.116084	0.048413	0.007623	-0.025935	0.
Ending Term Credits	0.408657	0.178570	0.377343	1.000000	0.318665	-0.025159	-0.061321	-0.077080	0.
Cumulated Debt	0.049916	0.096770	-0.116084	0.318665	1.000000	-0.019488	0.052788	0.725513	0.
Ending Term Department	0.045945	-0.023734	0.048413	-0.025159	-0.019488	1.000000	0.029720	0.023081	0.
Ending Term Unmet Need	-0.097780	0.006079	0.007623	-0.061321	0.052788	0.029720	1.000000	0.003498	0.
Average Debt Per Term	-0.082673	0.026575	-0.025935	-0.077080	0.725513	0.023081	0.003498	1.000000	-0.
Starting Term Cumulative GPA	0.139361	0.377786	0.157378	0.346839	0.039784	0.060338	0.012327	-0.097976	1.
Ending Term CRHR Completion	0.075948	0.667071	0.121404	0.466565	0.132524	0.020001	-0.017755	-0.024434	0.

In [25]:
```
data.corr(method="spearman")
```

Out[25]:

	Age	Starting Term CRHR Completion	Pre-Enrollment Credits	Ending Term Credits	Cumulated Debt	Ending Term Department	Ending Term Unmet Need	Average Debt Per Term	S Cum
Age	1.000000	-0.059871	0.300978	0.665096	0.171029	-0.043123	-0.129674	-0.069822	0.
Starting Term CRHR Completion	-0.059871	1.000000	0.038337	0.130549	0.047567	-0.012523	0.015054	-0.008154	0.
Pre-Enrollment Credits	0.300978	0.038337	1.000000	0.303864	-0.054486	0.016617	0.041302	0.005142	0.
Ending Term Credits	0.665096	0.130549	0.303864	1.000000	0.267024	-0.041319	-0.071729	-0.043119	0.
Cumulated Debt	0.171029	0.047567	-0.054486	0.267024	1.000000	-0.006115	0.196521	0.836440	-0.
Ending Term Department	-0.043123	-0.012523	0.016617	-0.041319	-0.006115	1.000000	0.033834	0.023458	0.
Ending Term Unmet Need	-0.129674	0.015054	0.041302	-0.071729	0.196521	0.033834	1.000000	0.168422	-0.
Average Debt Per Term	-0.069822	-0.008154	0.005142	-0.043119	0.836440	0.023458	0.168422	1.000000	-0.
Starting Term Cumulative GPA	0.145922	0.289793	0.181582	0.312746	-0.048828	0.070011	-0.002779	-0.128024	1.
Ending Term CRHR Completion	0.054467	0.520685	0.171830	0.388131	-0.014795	0.057145	0.010769	-0.079120	0.

In [26]:
```
#Logistic Regression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data,
           target, test_size=0.30,
            random_state=101)
X_test.shape
```

Out[26]: (6305, 10)

In [27]:
```
from sklearn.linear_model import LogisticRegression#create an instance and fit the
model
logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
```

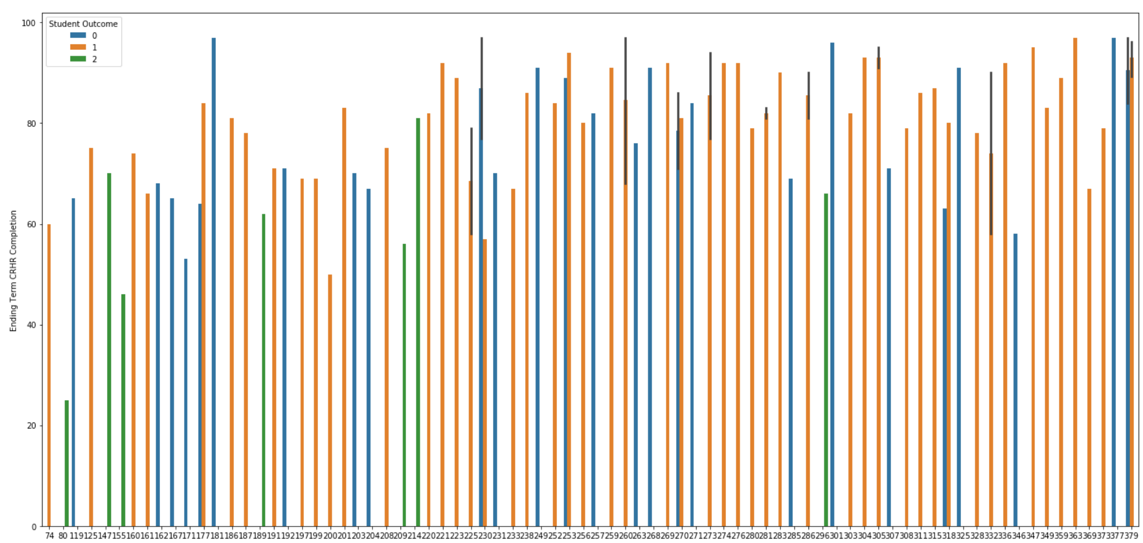
```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sk
learn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[27]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [28]:
```
#predictions
predictions = logmodel.predict(X_test)
```

In [29]:
```
plt.figure(figsize=(25,12))
sns.barplot(data=student_data.head(100), x= "Starting Term Cumulative GPA", y="End
ing Term CRHR Completion", hue=target)
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x122ced990>
```



In [30]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.57      0.04      0.08      1040
           1       0.87      0.98      0.93      3643
           2       0.65      0.86      0.74      1622

    accuracy                           0.80      6305
   macro avg       0.70      0.63      0.58      6305
weighted avg       0.77      0.80      0.74      6305
```

In [31]:
```
def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = preprocessing.LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)
    return roc_auc_score(y_test, y_pred, average=average)
```

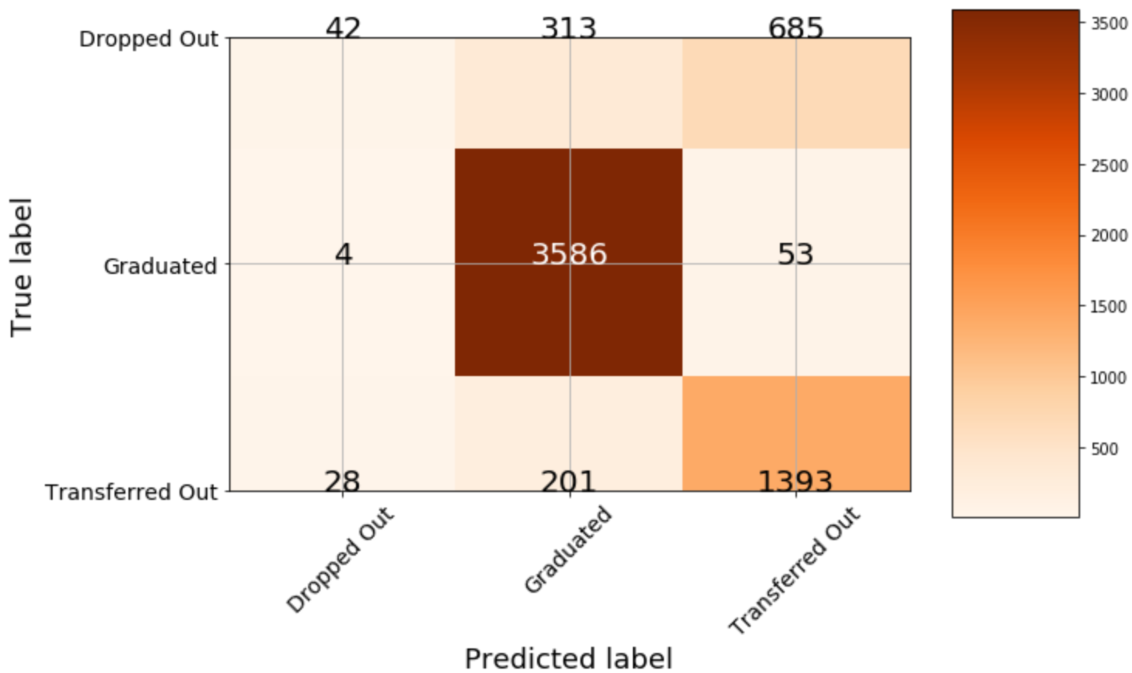
In [32]:
```
print("Accuracy", accuracy_score(y_test, predictions))
print("ROC", multiclass_roc_auc_score(y_test, predictions))
```

```
Accuracy 0.7963521015067406
ROC 0.7544662303561696
```

In [33]:
```
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          cmap=plt.cm.Oranges):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
#         print('Confusion matrix, without normalization')

        print(cm)

    # Plot the confusion matrix
    plt.figure(figsize = (10, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
#     plt.title(title, size = 24)
    plt.colorbar(aspect=4)
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45, size = 14)
    plt.yticks(tick_marks, classes, size = 14)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.

    # Labeling the plot
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt), fontsize = 20,
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.grid(None)
    plt.tight_layout()
    plt.ylabel('True label', size = 18)
    plt.xlabel('Predicted label', size = 18)
```

In [34]:
```
# Confusion matrix
cm = confusion_matrix(y_test, predictions)
plot_confusion_matrix(cm, classes = ["Dropped Out", "Graduated","Transferred Out"
])
```

```
[[  42  313  685]
 [   4 3586   53]
 [  28  201 1393]]
```



In [35]:
```
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

In [36]:
```
print("Accuracy:",accuracy_score(y_test, y_pred))
print("ROC", multiclass_roc_auc_score(y_test, y_pred))
```

```
Accuracy: 0.7679619349722443
ROC 0.7730238561557906
```

In [37]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.39      0.41      0.40      1040
           1       0.93      0.93      0.93      3643
           2       0.66      0.64      0.65      1622

    accuracy                           0.77      6305
   macro avg       0.66      0.66      0.66      6305
weighted avg       0.77      0.77      0.77      6305
```

In [38]:
```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = data.columns)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
# graph.write_png('test2.png')
Image(graph.create_png())
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sk
learn/externals/six.py:31: FutureWarning: The module is deprecated in version 0.
21 and will be removed in version 0.23 since we've dropped support for Python 2.
7. Please rely on the official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
```

Out[38]:

In [39]:
```
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_prediction = clf.predict(X_test)
```

In [40]:
```
print("Accuracy:",accuracy_score(y_test, y_prediction))
print("ROC", multiclass_roc_auc_score(y_test, y_prediction))
```

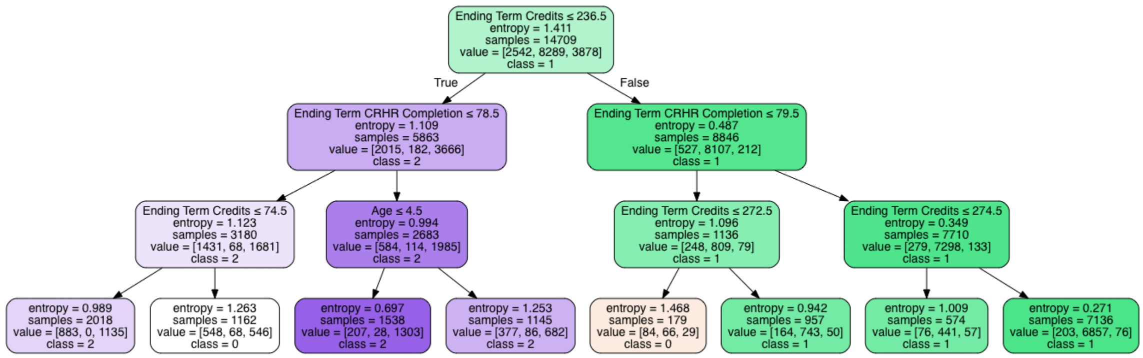
```
Accuracy: 0.8139571768437748
ROC 0.7938323027769213
```

In [41]:
```
print(classification_report(y_test,y_prediction))
```

```
              precision    recall  f1-score   support

           0       0.50      0.27      0.35      1040
           1       0.93      0.97      0.95      3643
           2       0.68      0.81      0.74      1622

    accuracy                           0.81      6305
   macro avg       0.70      0.68      0.68      6305
weighted avg       0.79      0.81      0.80      6305
```

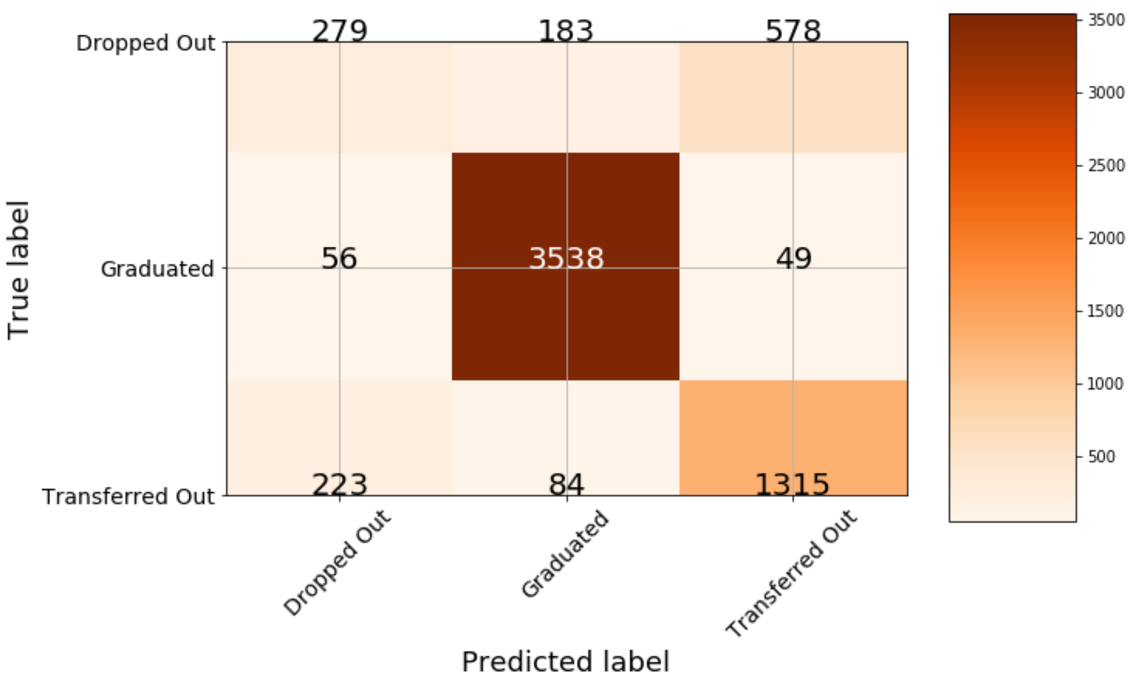
In [42]:
```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = data.columns,class_names=
['0','1','2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
# graph.write_png('test1.png')
Image(graph.create_png())
```

Out[42]:



In [43]:
```
cm = confusion_matrix(y_test, y_prediction)
plot_confusion_matrix(cm, classes = ["Dropped Out", "Graduated","Transferred Out"
])
```

```
[[ 279  183  578]
 [  56 3538   49]
 [ 223   84 1315]]
```



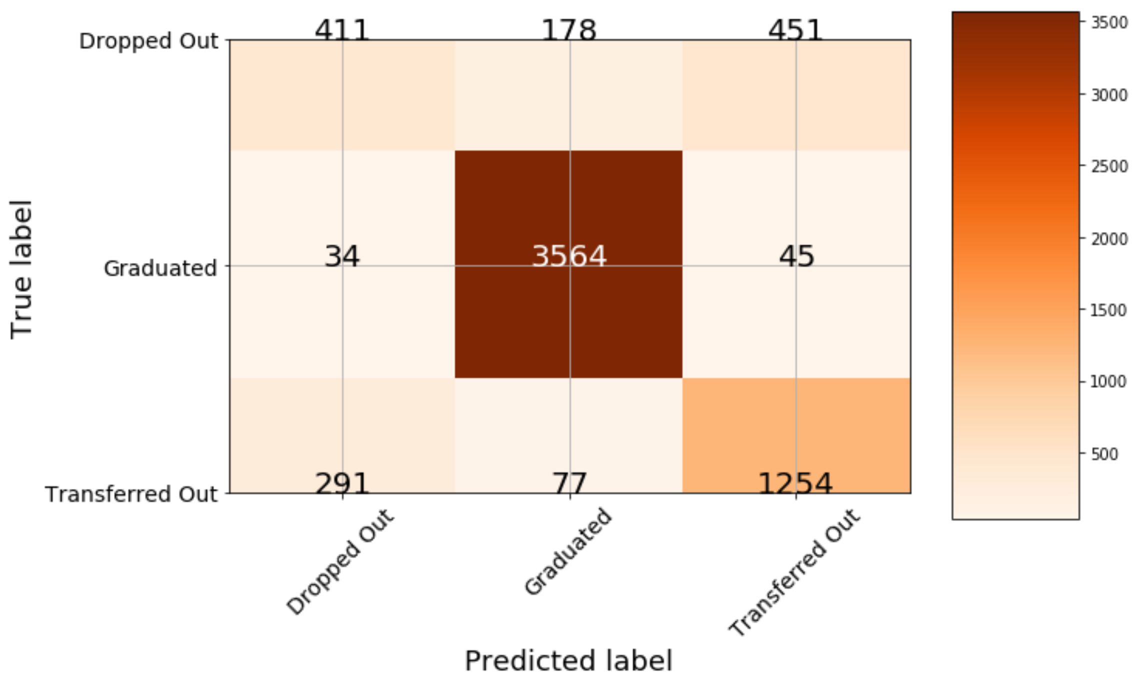
In [44]:
```
from sklearn.ensemble import RandomForestClassifier #
# 1. Choose the Random Forest Model
model = RandomForestClassifier(n_estimators=1000)
# 2. instantiate model
model.fit(X_train, y_train)
# 3. fit model to data
y_model = model.predict(X_test)
# 4. predict on new data
```

In [45]:
```
print("Accuracy", accuracy_score (y_test, y_model))
print("ROC", multiclass_roc_auc_score(y_test, y_model))
```

```
Accuracy 0.8293417922283902
ROC 0.8138650741825675
```

In [46]:
```
# Confusion matrix
cm = confusion_matrix(y_test, y_model)
plot_confusion_matrix(cm, classes = ["Dropped Out", "Graduated","Transferred Out"
])
```

```
[[ 411  178  451]
 [  34 3564   45]
 [ 291   77 1254]]
```



In [47]:
```
print(classification_report(y_test,y_model))
```

```
              precision    recall  f1-score   support

           0       0.56      0.40      0.46      1040
           1       0.93      0.98      0.96      3643
           2       0.72      0.77      0.74      1622

    accuracy                           0.83      6305
   macro avg       0.74      0.72      0.72      6305
weighted avg       0.82      0.83      0.82      6305
```

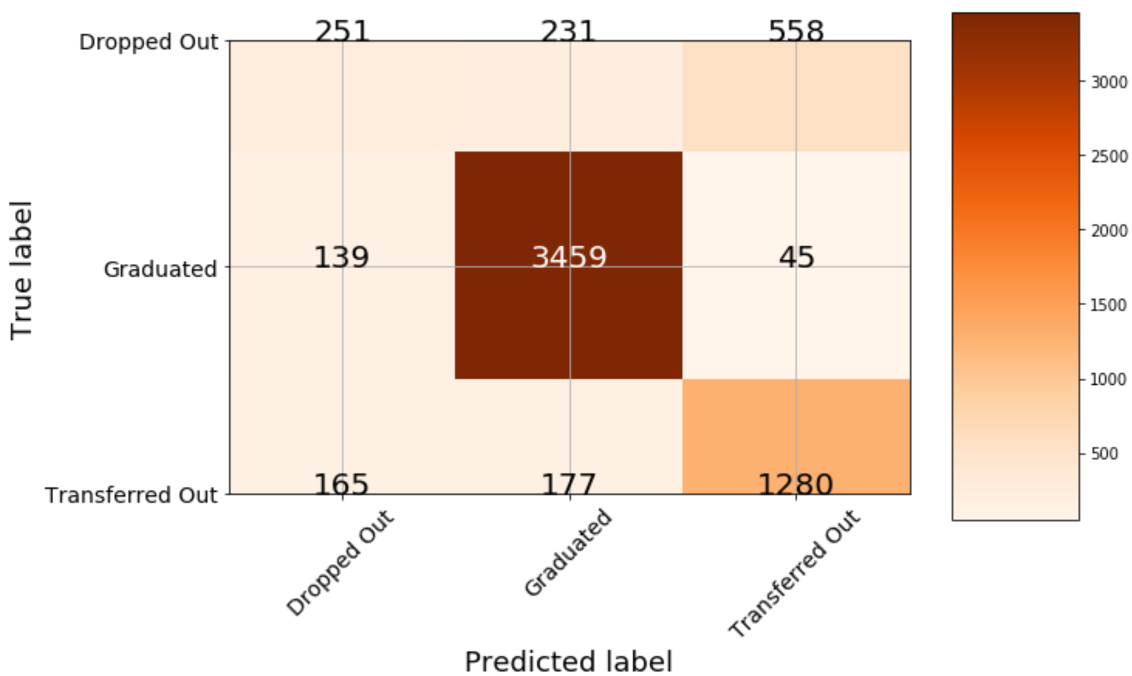
In [48]:
```
from sklearn.naive_bayes import GaussianNB # 1. choose model class
model = GaussianNB() # 2. instantiate model
model.fit(X_train, y_train) # 3. fit model to data
y_p = model.predict(X_test) # 4. predict on new data
```

In [49]:
```
print("ROC", multiclass_roc_auc_score(y_test, y_p))
print("Accuracy", accuracy_score (y_test, y_p))
```

```
ROC 0.7733693176030481
Accuracy 0.7914353687549563
```

In [52]:
```
# Confusion matrix
cm = confusion_matrix(y_test, y_p)
plot_confusion_matrix(cm, classes = ["Dropped Out", "Graduated","Transferred Out"
])
```

```
[[ 251  231  558]
 [ 139 3459   45]
 [ 165  177 1280]]
```



In [53]:
```
print(classification_report(y_test,y_p))
```

```
              precision    recall  f1-score   support

           0       0.45      0.24      0.31      1040
           1       0.89      0.95      0.92      3643
           2       0.68      0.79      0.73      1622

    accuracy                           0.79      6305
   macro avg       0.68      0.66      0.66      6305
weighted avg       0.77      0.79      0.77      6305
```