Homework 3 - Code Outputs

Allison Collins

Document Overview

In this file, I will run my updated pipeline on the kaggle credit dataset. Given the number of thresholds we were asked to compare, I reduced the different values of each of the other parameters from my initial exploration, as I did not see huge differences in performance, to improve the readability and length of this document.

I run each model individually with the associated performance outputs to try to make it clearer given the volume of information. Additionally, the baseline is computed using the sklearn dummy classifier as a baseline metric.

***A few notes and citations:

-- I consulted lab materials both here and in the pipeline document (specifically for adjusting thresholds & calculations from predicted scores; looping over models modifying parameters; investigating feature importance.)

--At the end of each section per type of model, I include a chart which has all of the performance metrics across models. The date, threshold, and other (specific to the model) parameter values are included in the leftmost column.

--I will call out in the different sections some models with better performance; the policy memo will stick to the implications for implementations and avoid too much technical language as per the instructions.

Data cleaning and processing

We will import the pipeline file and a few other modules we might need in cleaning the data (specific to this dataset and the task at hand).

In [1]:
```
import pipeline_revised
import pandas as pd
import pydot
import numpy as np
import warnings
warnings.simplefilter('ignore')
```

In [2]:
```
#Read in the data
df = pipeline_revised.read_file('projects_2012_2013.csv')
```

In [3]:
```
#Check its contents
df.columns
```

Out[3]:
```
Index(['projectid', 'teacher_acctid', 'schoolid', 'school_ncesid',
       'school_latitude', 'school_longitude', 'school_city', 'school_st
ate',
       'school_metro', 'school_district', 'school_county', 'school_char
ter',
       'school_magnet', 'teacher_prefix', 'primary_focus_subject',
       'primary_focus_area', 'secondary_focus_subject', 'secondary_focu
s_area',
       'resource_type', 'poverty_level', 'grade_level',
       'total_price_including_optional_support', 'students_reached',
       'eligible_double_your_impact_match', 'date_posted', 'datefullyfu
nded'],
      dtype='object')
```

In [4]:
```
df.head(1)
```

Out[4]:

	projectid	teacher_acctid	
0	00001ccc0e81598c4bd86bacb94d7acb	96963218e74e10c3764a5cfb153e6fea	9f3f9f2c2da7edda5648cc

1 rows × 26 columns

For this particular dataset, our target variable is time to funding, so we will need to create an additional column which looks at the time required to get full funding (e.g. the difference between date posted and date fully funded, since this is not explicitly included in the dataset).

In [5]:
```
df['datefullyfunded'] = pd.to_datetime(df.datefullyfunded)
df['date_posted'] = pd.to_datetime(df.date_posted)
df['time_to_funding'] = (df.datefullyfunded - df.date_posted).dt.days
df.head(1)
```

Out[5]:

	projectid	teacher_acctid	
0	00001ccc0e81598c4bd86bacb94d7acb	96963218e74e10c3764a5cfb153e6fea	9f3f9f2c2da7edda5648cc

1 rows × 27 columns

For this particular dataset, we have a number of categorical columns; in this case, I will opt to drop NAs; with the information we have on hand, it doesn't make sense to impute categories such as a school's focus area or poverty level. I'm subsetting this to columns planned to include in the analysis -- so if there is a na in a different column, we will retain it.

In [6]:
```
print('Current shape:', df.shape)
```

```
Current shape: (124976, 27)
```

In [7]:
```
df = df.dropna(axis=0,subset=['time_to_funding','total_price_including_optional_support','students_reached'])
print('New shape:', df.shape)
```

```
New shape: (124917, 27)
```

We can now look at some summary stats for the numerical columns with the cleaned dataset we are going to be working with. We can see that the average time to funding is 48 days, with a median of 41, so it is skewed upwards. The average total price is about 670 dollars, and on averge 90 students are reached.

In [8]:
```
pipeline_revised.calc_summary_stats(df,cols_to_include=['time_to_funding','total_price_including_optional_support','students_reached'])
```

Out[8]:

	mean	std_dev	median	max_val	min_val
time_to_funding	48.349144	31.997898	41.00	120.00	5.0
total_price_including_optional_support	654.075199	1098.259478	510.51	164382.84	92.0
students_reached	95.445760	163.481912	30.00	12143.00	1.0

In addition, we will have to create a binary outcome "label" column which assigns 1 or 0 based on whether the time to funding was over or under 60. We will also convert a few of the categorical variables to dummy columns so that we can include those in our analysis as well.

In [9]:
```
#We will convert the time-to-funding column to be binary as required for
some of the models
df['time_tf'] = np.where(df['time_to_funding']>=60, 1, 0)

#We will convert a few columns to use in our analysis
df = pipeline_revised.create_binary_col(df, 'eligible_double_your_impact
_match', {'t':1,'f':0})
df = pipeline_revised.discretize_categorical(df, ['poverty_level','resou
rce_type','primary_focus_subject','school_metro'])

#Now we can look at columns to grab which we want as predictors
df.columns
```

Out[9]:
```
Index(['projectid', 'teacher_acctid', 'schoolid', 'school_ncesid',
       'school_latitude', 'school_longitude', 'school_city', 'school_st
ate',
       'school_district', 'school_county', 'school_charter', 'school_ma
gnet',
       'teacher_prefix', 'primary_focus_area', 'secondary_focus_subjec
t',
       'secondary_focus_area', 'grade_level',
       'total_price_including_optional_support', 'students_reached',
       'eligible_double_your_impact_match', 'date_posted', 'datefullyfu
nded',
       'time_to_funding', 'time_tf',
       'eligible_double_your_impact_match_binary',
       'poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty',
       'resource_type_Books', 'resource_type_Other', 'resource_type_Sup
plies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors', 'primary_focus_subject_Applied Science
s',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subjec
t_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Othe
r',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Sp
orts',
       'primary_focus_subject_Visual Arts', 'school_metro_rural',
       'school_metro_suburban', 'school_metro_urban'],
      dtype='object')
```

Now that the data is in the right format, we will work through the different models and look at their evaluation performance.

We can first start with the decision tree.

Decision Tree Classifier

In [10]:
```
start_list = ['2013-06-30', '2013-05-31', '2013-04-30']
end_list = ['2013-12-31', '2013-11-30','2013-10-31']
thresholds = [.01,.02,.05,.1,.2,.3,.5]
depth_list = [3, 4]
trees = []

for i, date in enumerate(start_list):
    end_date = end_list[i]
    for depth in depth_list:
        for threshold in thresholds:
            tree_name = 'tree_' + date + '_' + str(threshold) + '_' + st
r(depth)
            y_test, y_predict, y_score, features = pipeline_revised.crea
te_decision_tree(df, 'time_tf', depth ,500 ,tree_name, threshold=thresho
ld,
                                                    predictors=['t
otal_price_including_optional_support', 'students_reached','poverty_leve
l_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','re
source_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subj
ect_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject
_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Spo
rts',
       'primary_focus_subject_Visual Arts'],temporal=True, start_col='da
te_posted',end_col='datefullyfunded',start_date=date,end_date=end_date)

            trees.append((tree_name,y_test,y_predict, y_score, features
))
```

We will create a table which shows the performance of the different iterations of the model.

In [11]:
```
tree_metrics = pd.DataFrame(columns=['name','accuracy','precision','auc'
,'recall','f1'])

for i in range(len(trees)):
    name, test, predict, _, _ = trees[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_mod
el(test, predict)
    tree_metrics.loc[i,'name'] = name
    tree_metrics.loc[i,'accuracy'] = accuracy
    tree_metrics.loc[i,'precision'] = precision
    tree_metrics.loc[i,'auc'] = auc
    tree_metrics.loc[i,'recall'] = recall
    tree_metrics.loc[i,'f1'] = f1

tree_metrics
```

Out[11]:

	name	accuracy	precision	auc	recall	f1
0	tree_2013-06-30_0.01_3	0.168283	0.168283	0.5	1	0.288087
1	tree_2013-06-30_0.02_3	0.168283	0.168283	0.5	1	0.288087
2	tree_2013-06-30_0.05_3	0.168283	0.168283	0.5	1	0.288087
3	tree_2013-06-30_0.1_3	0.243857	0.179903	0.538117	0.981659	0.30408
4	tree_2013-06-30_0.2_3	0.39903	0.202386	0.588558	0.874236	0.328682
5	tree_2013-06-30_0.3_3	0.55585	0.221265	0.593661	0.650655	0.33023
6	tree_2013-06-30_0.5_3	0.831717	0	0.5	0	0
7	tree_2013-06-30_0.01_4	0.168283	0.168283	0.5	1	0.288087
8	tree_2013-06-30_0.02_4	0.168283	0.168283	0.5	1	0.288087
9	tree_2013-06-30_0.05_4	0.168283	0.168283	0.5	1	0.288087
10	tree_2013-06-30_0.1_4	0.243857	0.179903	0.538117	0.981659	0.30408
11	tree_2013-06-30_0.2_4	0.397972	0.202019	0.587713	0.873712	0.328161
12	tree_2013-06-30_0.3_4	0.560112	0.221988	0.593715	0.644367	0.330215
13	tree_2013-06-30_0.5_4	0.831717	0	0.5	0	0
14	tree_2013-05-31_0.01_3	0.158696	0.158696	0.5	1	0.273922
15	tree_2013-05-31_0.02_3	0.158696	0.158696	0.5	1	0.273922
16	tree_2013-05-31_0.05_3	0.158696	0.158696	0.5	1	0.273922
17	tree_2013-05-31_0.1_3	0.225811	0.168225	0.5331	0.98327	0.287297
18	tree_2013-05-31_0.2_3	0.395093	0.190094	0.584651	0.862346	0.311517
19	tree_2013-05-31_0.3_3	0.573651	0.204731	0.578138	0.58471	0.303273
20	tree_2013-05-31_0.5_3	0.841304	0	0.5	0	0
21	tree_2013-05-31_0.01_4	0.158696	0.158696	0.5	1	0.273922
22	tree_2013-05-31_0.02_4	0.158696	0.158696	0.5	1	0.273922
23	tree_2013-05-31_0.05_4	0.158696	0.158696	0.5	1	0.273922
24	tree_2013-05-31_0.1_4	0.217308	0.166936	0.528906	0.985388	0.285504
25	tree_2013-05-31_0.2_4	0.389212	0.188957	0.582358	0.865311	0.31018
26	tree_2013-05-31_0.3_4	0.573651	0.204731	0.578138	0.58471	0.303273
27	tree_2013-05-31_0.5_4	0.841304	0	0.5	0	0
28	tree_2013-04-30_0.01_3	0.157058	0.157058	0.5	1	0.271478
29	tree_2013-04-30_0.02_3	0.157058	0.157058	0.5	1	0.271478
30	tree_2013-04-30_0.05_3	0.157058	0.157058	0.5	1	0.271478
31	tree_2013-04-30_0.1_3	0.227879	0.166741	0.533746	0.979693	0.284979
32	tree_2013-04-30_0.2_3	0.437602	0.190625	0.583049	0.795105	0.307522
33	tree_2013-04-30_0.3_3	0.575237	0.208478	0.589167	0.609477	0.310683

	name	accuracy	precision	auc	recall	f1
34	tree_2013-04-30_0.5_3	0.842942	0	0.5	0	0
35	tree_2013-04-30_0.01_4	0.157058	0.157058	0.5	1	0.271478
36	tree_2013-04-30_0.02_4	0.157058	0.157058	0.5	1	0.271478
37	tree_2013-04-30_0.05_4	0.157058	0.157058	0.5	1	0.271478
38	tree_2013-04-30_0.1_4	0.219251	0.16549	0.529688	0.982296	0.283258
39	tree_2013-04-30_0.2_4	0.40579	0.189702	0.586846	0.85082	0.310234
40	tree_2013-04-30_0.3_4	0.579939	0.209642	0.589944	0.60453	0.311323
41	tree_2013-04-30_0.5_4	0.838281	0.284091	0.505179	0.0195262	0.0365408

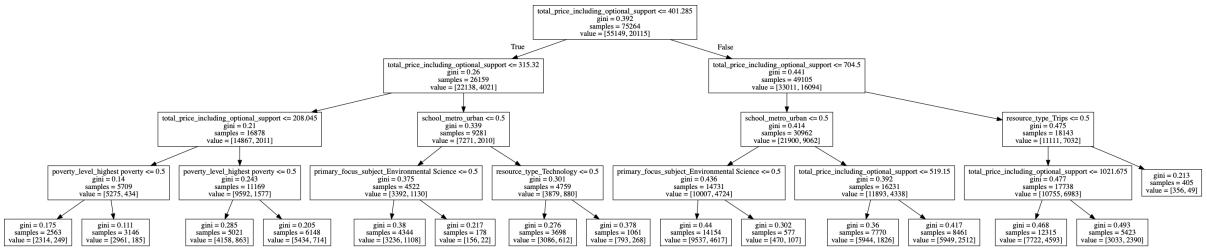
Some of trees (as will happen with later models) have precision / recall of 0 because they were lacking true positives. Let's visualize the one which has the highest area under the curve.

In [12]:
```
(graph,) = pydot.graph_from_dot_file('tree_2013-06-30_0.3_4')
```

In [13]:
```
graph.write_png('tree_1.png')
```

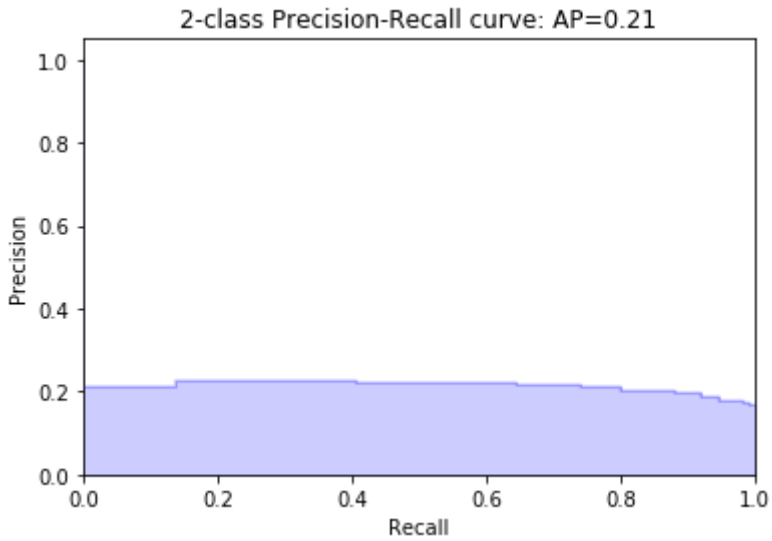
In [14]:
```
from IPython.display import Image
Image("tree_1.png")
```

Out[14]:



In [15]:
```
#Citation for precision recall curve - sklearn documentation
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
from sklearn.utils.fixes import signature
from sklearn.metrics import average_precision_score

_, test, _, score, _ = trees[12]
clean_score = [x[1] for x in score]

precision, recall, _ = precision_recall_curve(test, clean_score)
average_precision = average_precision_score(test, clean_score)

step_kwargs = ({'step': 'post'}
               if 'step' in signature(plt.fill_between).parameters
               else {})
plt.step(recall, precision, color='b', alpha=0.2,
         where='post')
plt.fill_between(recall, precision, alpha=0.2, color='b', **step_kwargs)

plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('2-class Precision-Recall curve: AP={0:0.2f}'.format(
          average_precision))
```

Out[15]:
```
Text(0.5, 1.0, '2-class Precision-Recall curve: AP=0.21')
```



This will be discussed in the policy write-up, but overall the performance across the models will be fairly similar. Investigating model documentation suggests that feature selection can be the most important -- selecting on different models with good or bad features will perform in line with that. Let's visualize the most important features for this classifier.

In [16]:
```
feature_scores = (trees[12][4])
feature_names = ['total_price_including_optional_support', 'students_reached','poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','resource_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subject_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Music',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Sports',
       'primary_focus_subject_Visual Arts']

d = {'Features': feature_names, "Importance": feature_scores}
feature_importance = pd.DataFrame(data=d)
feature_importance = feature_importance.sort_values(by=['Importance'], ascending=False)
feature_importance
```

Out[16]:

	Features	Importance
0	total_price_including_optional_support	0.887343
14	school_metro_urban	0.038917
10	resource_type_Trips	0.036004
23	primary_focus_subject_Environmental Science	0.017046
3	poverty_level_highest poverty	0.013041
9	resource_type_Technology	0.007648
25	primary_focus_subject_Foreign Languages	0.000000
26	primary_focus_subject_Gym & Fitness	0.000000
27	primary_focus_subject_Health & Life Science	0.000000
28	primary_focus_subject_Health & Wellness	0.000000
29	primary_focus_subject_History & Geography	0.000000
30	primary_focus_subject_Literacy	0.000000
31	primary_focus_subject_Literature & Writing	0.000000
34	primary_focus_subject_Nutrition	0.000000
32	primary_focus_subject_Mathematics	0.000000
33	primary_focus_subject_Music	0.000000
35	primary_focus_subject_Other	0.000000
36	primary_focus_subject_Parent Involvement	0.000000
37	primary_focus_subject_Performing Arts	0.000000
38	primary_focus_subject_Social Sciences	0.000000
39	primary_focus_subject_Special Needs	0.000000
40	primary_focus_subject_Sports	0.000000
24	primary_focus_subject_Extracurricular	0.000000
21	primary_focus_subject_Early Development	0.000000
22	primary_focus_subject_Economics	0.000000
1	students_reached	0.000000
2	poverty_level_high poverty	0.000000
4	poverty_level_low poverty	0.000000
5	poverty_level_moderate poverty	0.000000
6	resource_type_Books	0.000000
7	resource_type_Other	0.000000
8	resource_type_Supplies	0.000000
11	resource_type_Visitors	0.000000
12	school_metro_rural	0.000000

	Features	Importance
13	school_metro_suburban	0.000000
15	primary_focus_subject_Applied Sciences	0.000000
16	primary_focus_subject_Character Education	0.000000
17	primary_focus_subject_Civics & Government	0.000000
18	primary_focus_subject_College & Career Prep	0.000000
19	primary_focus_subject_Community Service	0.000000
20	primary_focus_subject_ESL	0.000000
41	primary_focus_subject_Visual Arts	0.000000

As we can see above, the most relevant features are the total price of the request; the school being in an urban area; and it being directed toward trips.

Nearest Neighbor Classifier

In [17]:
```
start_list = ['2013-06-30', '2013-05-31', '2013-04-30']
end_list = ['2013-12-31', '2013-11-30','2013-10-31']
neighbor_list = [10, 50]
thresholds = [.01,.02,.05,.1,.2,.3,.5]
metrics = ["euclidean", "minkowski"]
neighbors = []

for i, date in enumerate(start_list):
    end_date = end_list[i]
    for n in neighbor_list:
        for metric in metrics:
            for threshold in thresholds:
                model_name = 'knn_' + date + '_' + str(n) + '_' + str(th
reshold) +'_' + metric
                p_test, p_predict, p_score, params = pipeline_revised.kn
n_classify(df, 'time_tf', n, metric, threshold = threshold,

predictors=['total_price_including_optional_support', 'students_reached'
,'poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','re
source_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subj
ect_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject
_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Spo
rts',
       'primary_focus_subject_Visual Arts'],temporal=True, start_col='da
te_posted',end_col='datefullyfunded',start_date=date,end_date=end_date)
                neighbors.append((model_name, p_test, p_predict, p_score
, params))
```

We will create a table which shows the performance of the different iterations of the model.

In [18]:
```
knn_metrics = pd.DataFrame(columns=['name','accuracy','precision','auc',
'recall','f1'])

for i in range(len(neighbors)):
    name, test, predict, _, _ = neighbors[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_mod
el(test, predict)
    knn_metrics.loc[i,'name'] = name
    knn_metrics.loc[i,'accuracy'] = accuracy
    knn_metrics.loc[i,'precision'] = precision
    knn_metrics.loc[i,'auc'] = auc
    knn_metrics.loc[i,'recall'] = recall
    knn_metrics.loc[i,'f1'] = f1

knn_metrics
```

Out[18]:

	name	accuracy	precision	auc	recall	f1
0	knn_2013-06-30_10_0.01_euclidean	0.251293	0.177648	0.530118	0.950393	0.299343
1	knn_2013-06-30_10_0.02_euclidean	0.251293	0.177648	0.530118	0.950393	0.299343
2	knn_2013-06-30_10_0.05_euclidean	0.251293	0.177648	0.530118	0.950393	0.299343
3	knn_2013-06-30_10_0.1_euclidean	0.381188	0.190101	0.556654	0.821135	0.308728
4	knn_2013-06-30_10_0.2_euclidean	0.519753	0.201295	0.561581	0.624629	0.30447
5	knn_2013-06-30_10_0.3_euclidean	0.646649	0.216038	0.555592	0.418341	0.284932
6	knn_2013-06-30_10_0.5_euclidean	0.793504	0.233825	0.516807	0.099738	0.139831
7	knn_2013-06-30_10_0.01_minkowski	0.251293	0.177648	0.530118	0.950393	0.299343
8	knn_2013-06-30_10_0.02_minkowski	0.251293	0.177648	0.530118	0.950393	0.299343
9	knn_2013-06-30_10_0.05_minkowski	0.251293	0.177648	0.530118	0.950393	0.299343
10	knn_2013-06-30_10_0.1_minkowski	0.381188	0.190101	0.556654	0.821135	0.308728
11	knn_2013-06-30_10_0.2_minkowski	0.519753	0.201295	0.561581	0.624629	0.30447
12	knn_2013-06-30_10_0.3_minkowski	0.646649	0.216038	0.555592	0.418341	0.284932
13	knn_2013-06-30_10_0.5_minkowski	0.793504	0.233825	0.516807	0.099738	0.139831
14	knn_2013-06-30_50_0.01_euclidean	0.170488	0.168538	0.500907	0.998952	0.288416
15	knn_2013-06-30_50_0.02_euclidean	0.179718	0.169955	0.505899	0.997555	0.290429
16	knn_2013-06-30_50_0.05_euclidean	0.195267	0.172317	0.513993	0.99441	0.293734
17	knn_2013-06-30_50_0.1_euclidean	0.267607	0.182363	0.544662	0.962271	0.306618
18	knn_2013-06-30_50_0.2_euclidean	0.418342	0.200673	0.579895	0.823406	0.3227
19	knn_2013-06-30_50_0.3_euclidean	0.603909	0.219244	0.573857	0.528559	0.30993
20	knn_2013-06-30_50_0.5_euclidean	0.82134	0.26498	0.507626	0.0347598	0.0614577
21	knn_2013-06-30_50_0.01_minkowski	0.170488	0.168538	0.500907	0.998952	0.288416
22	knn_2013-06-30_50_0.02_minkowski	0.179718	0.169955	0.505899	0.997555	0.290429
23	knn_2013-06-30_50_0.05_minkowski	0.195267	0.172317	0.513993	0.99441	0.293734
24	knn_2013-06-30_50_0.1_minkowski	0.267607	0.182363	0.544662	0.962271	0.306618
25	knn_2013-06-30_50_0.2_minkowski	0.418342	0.200673	0.579895	0.823406	0.3227
26	knn_2013-06-30_50_0.3_minkowski	0.603909	0.219244	0.573857	0.528559	0.30993
27	knn_2013-06-30_50_0.5_minkowski	0.82134	0.26498	0.507626	0.0347598	0.0614577
28	knn_2013-05-31_10_0.01_euclidean	0.245942	0.167099	0.528139	0.94155	0.283826
29	knn_2013-05-31_10_0.02_euclidean	0.245942	0.167099	0.528139	0.94155	0.283826
...	...	...	...	...	...	...
54	knn_2013-05-31_50_0.3_minkowski	0.617442	0.205864	0.56722	0.493647	0.290558
55	knn_2013-05-31_50_0.5_minkowski	0.831625	0.234317	0.505159	0.0268954	0.0482523
56	knn_2013-04-30_10_0.01_euclidean	0.252004	0.165958	0.529732	0.934652	0.281867

	name	accuracy	precision	auc	recall	f1
57	knn_2013-04-30_10_0.02_euclidean	0.252004	0.165958	0.529732	0.934652	0.281867
58	knn_2013-04-30_10_0.05_euclidean	0.252004	0.165958	0.529732	0.934652	0.281867
59	knn_2013-04-30_10_0.1_euclidean	0.393891	0.177683	0.554261	0.788076	0.289984
60	knn_2013-04-30_10_0.2_euclidean	0.540522	0.188248	0.557136	0.581359	0.284404
61	knn_2013-04-30_10_0.3_euclidean	0.670469	0.20195	0.549056	0.372039	0.261794
62	knn_2013-04-30_10_0.5_euclidean	0.810435	0.225294	0.515248	0.0848737	0.123298
63	knn_2013-04-30_10_0.01_minkowski	0.252004	0.165958	0.529732	0.934652	0.281867
64	knn_2013-04-30_10_0.02_minkowski	0.252004	0.165958	0.529732	0.934652	0.281867
65	knn_2013-04-30_10_0.05_minkowski	0.252004	0.165958	0.529732	0.934652	0.281867
66	knn_2013-04-30_10_0.1_minkowski	0.393891	0.177683	0.554261	0.788076	0.289984
67	knn_2013-04-30_10_0.2_minkowski	0.540522	0.188248	0.557136	0.581359	0.284404
68	knn_2013-04-30_10_0.3_minkowski	0.670469	0.20195	0.549056	0.372039	0.261794
69	knn_2013-04-30_10_0.5_minkowski	0.810435	0.225294	0.515248	0.0848737	0.123298
70	knn_2013-04-30_50_0.01_euclidean	0.15947	0.157353	0.501113	0.999219	0.27189
71	knn_2013-04-30_50_0.02_euclidean	0.171941	0.159063	0.507452	0.996615	0.27434
72	knn_2013-04-30_50_0.05_euclidean	0.188543	0.161076	0.514651	0.990107	0.277076
73	knn_2013-04-30_50_0.1_euclidean	0.269913	0.170662	0.544698	0.945327	0.289127
74	knn_2013-04-30_50_0.2_euclidean	0.437929	0.188424	0.576993	0.779745	0.303506
75	knn_2013-04-30_50_0.3_euclidean	0.638862	0.206239	0.56452	0.456131	0.284047
76	knn_2013-04-30_50_0.5_euclidean	0.835214	0.227666	0.503784	0.0205676	0.0377268
77	knn_2013-04-30_50_0.01_minkowski	0.15947	0.157353	0.501113	0.999219	0.27189
78	knn_2013-04-30_50_0.02_minkowski	0.171941	0.159063	0.507452	0.996615	0.27434
79	knn_2013-04-30_50_0.05_minkowski	0.188543	0.161076	0.514651	0.990107	0.277076
80	knn_2013-04-30_50_0.1_minkowski	0.269913	0.170662	0.544698	0.945327	0.289127
81	knn_2013-04-30_50_0.2_minkowski	0.437929	0.188424	0.576993	0.779745	0.303506
82	knn_2013-04-30_50_0.3_minkowski	0.638862	0.206239	0.56452	0.456131	0.284047
83	knn_2013-04-30_50_0.5_minkowski	0.835214	0.227666	0.503784	0.0205676	0.0377268

84 rows × 6 columns

The top performing ones from an area under the curve perspective had number of neighbors of 50, threshold .3 -- but the distance mechanism didn't matter. We can check the characteristics for one of these if we want.

In [19]:
```
#neighbors[80][4]
```

Logistic Classifier

In [20]:
```
start_list = ['2013-06-30', '2013-05-31', '2013-04-30']
end_list = ['2013-12-31', '2013-11-30','2013-10-31']
thresholds = [.01,.02,.05,.1,.2,.3,.5]
penalties = ['l1','l2']
c_values = [.01, 1]
regressions = []

for i, date in enumerate(start_list):
    end_date = end_list[i]
    for t in thresholds:
        for penalty in penalties:
            for c in c_values:
                model_name = 'reg_' + date + '_' + str(t) + '_' + penalt
y + '_' + str(c)
                r_test, r_predict, r_score, coeffs = pipeline_revised.lo
gistic_classify(df, 'time_tf', penalty, c, threshold = t,

predictors=['total_price_including_optional_support', 'students_reached'
,'poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','re
source_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subj
ect_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject
_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Spo
rts',
       'primary_focus_subject_Visual Arts'],temporal=True, start_col='da
te_posted',end_col='datefullyfunded',start_date=date,end_date=end_date)
                regressions.append((model_name, r_test, r_predict, r_sco
re, coeffs))
```

We will create a table which shows the performance of the different iterations of the model.

In [21]:
```
pd.set_option("display.max_rows", 100)
log_metrics = pd.DataFrame(columns=['name','accuracy','precision','auc',
'recall','f1'])

for i in range(len(regressions)):
    name, test, predict, _, _ = regressions[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_mod
el(test, predict)
    log_metrics.loc[i,'name'] = name
    log_metrics.loc[i,'accuracy'] = accuracy
    log_metrics.loc[i,'precision'] = precision
    log_metrics.loc[i,'auc'] = auc
    log_metrics.loc[i,'recall'] = recall
    log_metrics.loc[i,'f1'] = f1

log_metrics
```

Out[21]:

	name	accuracy	precision	auc	recall	f1
0	reg_2013-06-30_0.01_l1_0.01	0.168283	0.168283	0.5	1	0.288087
1	reg_2013-06-30_0.01_l1_1	0.168577	0.168333	0.500177	1	0.288159
2	reg_2013-06-30_0.01_l2_0.01	0.168283	0.168283	0.5	1	0.288087
3	reg_2013-06-30_0.01_l2_1	0.168283	0.168283	0.5	1	0.288087
4	reg_2013-06-30_0.02_l1_0.01	0.168283	0.168283	0.5	1	0.288087
5	reg_2013-06-30_0.02_l1_1	0.169694	0.168463	0.500639	0.999476	0.288327
6	reg_2013-06-30_0.02_l2_0.01	0.168283	0.168283	0.5	1	0.288087
7	reg_2013-06-30_0.02_l2_1	0.168283	0.168283	0.5	1	0.288087
8	reg_2013-06-30_0.05_l1_0.01	0.168283	0.168283	0.5	1	0.288087
9	reg_2013-06-30_0.05_l1_1	0.170047	0.168503	0.500782	0.999301	0.288379
10	reg_2013-06-30_0.05_l2_0.01	0.168283	0.168283	0.5	1	0.288087
11	reg_2013-06-30_0.05_l2_1	0.16943	0.168457	0.50062	0.999825	0.288334
12	reg_2013-06-30_0.1_l1_0.01	0.168342	0.168293	0.500035	1	0.288101
13	reg_2013-06-30_0.1_l1_1	0.170488	0.168558	0.500977	0.999127	0.288452
14	reg_2013-06-30_0.1_l2_0.01	0.168283	0.168283	0.5	1	0.288087
15	reg_2013-06-30_0.1_l2_1	0.170076	0.168527	0.500869	0.999476	0.288422
16	reg_2013-06-30_0.2_l1_0.01	0.233392	0.176443	0.52695	0.969432	0.298548
17	reg_2013-06-30_0.2_l1_1	0.298001	0.184127	0.547817	0.924367	0.307085
18	reg_2013-06-30_0.2_l2_0.01	0.277278	0.181191	0.540096	0.936245	0.303623
19	reg_2013-06-30_0.2_l2_1	0.300206	0.184675	0.549351	0.924891	0.307876
20	reg_2013-06-30_0.3_l1_0.01	0.669048	0.22712	0.562648	0.402271	0.290325
21	reg_2013-06-30_0.3_l1_1	0.657554	0.231973	0.573921	0.44786	0.305638
22	reg_2013-06-30_0.3_l2_0.01	0.655791	0.228473	0.569656	0.439825	0.300729
23	reg_2013-06-30_0.3_l2_1	0.656379	0.2307	0.572587	0.446288	0.304167
24	reg_2013-06-30_0.5_l1_0.01	0.825367	0.21123	0.501687	0.0137991	0.0259059
25	reg_2013-06-30_0.5_l1_1	0.819694	0.20146	0.502386	0.0241048	0.0430577
26	reg_2013-06-30_0.5_l2_0.01	0.826132	0.240437	0.502773	0.0153712	0.0288951
27	reg_2013-06-30_0.5_l2_1	0.821458	0.200686	0.501984	0.0204367	0.0370958
28	reg_2013-05-31_0.01_l1_0.01	0.158696	0.158696	0.5	1	0.273922
29	reg_2013-05-31_0.01_l1_1	0.159368	0.158803	0.500399	1	0.274081
30	reg_2013-05-31_0.01_l2_0.01	0.158696	0.158696	0.5	1	0.273922
31	reg_2013-05-31_0.01_l2_1	0.158898	0.158728	0.50012	1	0.273969
32	reg_2013-05-31_0.02_l1_0.01	0.158696	0.158696	0.5	1	0.273922
33	reg_2013-05-31_0.02_l1_1	0.16004	0.158864	0.500627	0.999576	0.274156

	name	accuracy	precision	auc	recall	f1
34	reg_2013-05-31_0.02_l2_0.01	0.158696	0.158696	0.5	1	0.273922
35	reg_2013-05-31_0.02_l2_1	0.159805	0.158826	0.500487	0.999576	0.2741
36	reg_2013-05-31_0.05_l1_0.01	0.158696	0.158696	0.5	1	0.273922
37	reg_2013-05-31_0.05_l1_1	0.160208	0.158845	0.500555	0.999153	0.274111
38	reg_2013-05-31_0.05_l2_0.01	0.158696	0.158696	0.5	1	0.273922
39	reg_2013-05-31_0.05_l2_1	0.160175	0.158862	0.500621	0.999365	0.274145
40	reg_2013-05-31_0.1_l1_0.01	0.15873	0.158701	0.50002	1	0.27393
41	reg_2013-05-31_0.1_l1_1	0.160645	0.158914	0.500815	0.999153	0.274215
42	reg_2013-05-31_0.1_l2_0.01	0.158696	0.158696	0.5	1	0.273922
43	reg_2013-05-31_0.1_l2_1	0.160444	0.158882	0.500695	0.999153	0.274167
44	reg_2013-05-31_0.2_l1_0.01	0.239187	0.168271	0.532544	0.962304	0.286453
45	reg_2013-05-31_0.2_l1_1	0.30731	0.174819	0.549576	0.90449	0.293006
46	reg_2013-05-31_0.2_l2_0.01	0.291716	0.173084	0.545292	0.916773	0.291192
47	reg_2013-05-31_0.2_l2_1	0.307646	0.17489	0.549776	0.90449	0.293106
48	reg_2013-05-31_0.3_l1_0.01	0.687649	0.209308	0.550094	0.348581	0.26156
49	reg_2013-05-31_0.3_l1_1	0.673399	0.215619	0.562932	0.401101	0.280468
50	reg_2013-05-31_0.3_l2_0.01	0.678239	0.213307	0.558162	0.382253	0.273817
51	reg_2013-05-31_0.3_l2_1	0.67434	0.215464	0.562375	0.398348	0.279661
52	reg_2013-05-31_0.5_l1_0.01	0.836767	0.264808	0.503833	0.0160949	0.0303454
53	reg_2013-05-31_0.5_l1_1	0.832432	0.23494	0.504779	0.0247776	0.0448276
54	reg_2013-05-31_0.5_l2_0.01	0.837069	0.282759	0.504528	0.0173655	0.0327215
55	reg_2013-05-31_0.5_l2_1	0.832768	0.23431	0.504549	0.0237188	0.0430769
56	reg_2013-04-30_0.01_l1_0.01	0.157058	0.157058	0.5	1	0.271478
57	reg_2013-04-30_0.01_l1_1	0.15763	0.157091	0.500128	0.999479	0.271509
58	reg_2013-04-30_0.01_l2_0.01	0.157058	0.157058	0.5	1	0.271478
59	reg_2013-04-30_0.01_l2_1	0.15718	0.157077	0.500073	1	0.271506
60	reg_2013-04-30_0.02_l1_0.01	0.157058	0.157058	0.5	1	0.271478
61	reg_2013-04-30_0.02_l1_1	0.158121	0.157169	0.500419	0.999479	0.271624
62	reg_2013-04-30_0.02_l2_0.01	0.157058	0.157058	0.5	1	0.271478
63	reg_2013-04-30_0.02_l2_1	0.157712	0.157104	0.500176	0.999479	0.271528
64	reg_2013-04-30_0.05_l1_0.01	0.157058	0.157058	0.5	1	0.271478
65	reg_2013-04-30_0.05_l1_1	0.158243	0.15716	0.500386	0.999219	0.271601
66	reg_2013-04-30_0.05_l2_0.01	0.157058	0.157058	0.5	1	0.271478
67	reg_2013-04-30_0.05_l2_1	0.158202	0.157153	0.500361	0.999219	0.271592
68	reg_2013-04-30_0.1_l1_0.01	0.157098	0.157064	0.500024	1	0.271487

	name	accuracy	precision	auc	recall	f1
69	reg_2013-04-30_0.1_l1_1	0.158816	0.157222	0.500619	0.998959	0.271684
70	reg_2013-04-30_0.1_l2_0.01	0.157098	0.157064	0.500024	1	0.271487
71	reg_2013-04-30_0.1_l2_1	0.158734	0.157237	0.500677	0.999219	0.271717
72	reg_2013-04-30_0.2_l1_0.01	0.256052	0.168507	0.538277	0.949753	0.28623
73	reg_2013-04-30_0.2_l1_1	0.326096	0.175631	0.555886	0.890914	0.293419
74	reg_2013-04-30_0.2_l2_0.01	0.305201	0.172975	0.549424	0.905493	0.290463
75	reg_2013-04-30_0.2_l2_1	0.326014	0.175513	0.55552	0.890133	0.293212
76	reg_2013-04-30_0.3_l1_0.01	0.712504	0.208196	0.543166	0.296277	0.244547
77	reg_2013-04-30_0.3_l1_1	0.695453	0.220257	0.562922	0.369695	0.27605
78	reg_2013-04-30_0.3_l2_0.01	0.69852	0.214147	0.554467	0.344442	0.264098
79	reg_2013-04-30_0.3_l2_1	0.696802	0.220781	0.562981	0.367873	0.27595
80	reg_2013-04-30_0.5_l1_0.01	0.839426	0.295238	0.504481	0.0161416	0.0306097
81	reg_2013-04-30_0.5_l1_1	0.835868	0.259053	0.505655	0.0242124	0.0442857
82	reg_2013-04-30_0.5_l2_0.01	0.83914	0.289593	0.504523	0.0166623	0.0315116
83	reg_2013-04-30_0.5_l2_1	0.836277	0.263768	0.505685	0.0236917	0.0434783

Regression 21 had the highest area under the curve, and one of the higher accuracies; we can visualize the weights given to the different features if we wanted.

In [22]:
```
#regressions[21][4]
```

SVM

In [23]:
```
start_list = ['2013-06-30', '2013-05-31', '2013-04-30']
end_list = ['2013-12-31', '2013-11-30','2013-10-31']
thresholds = [.01,.02,.05,.1,.2,.3,.5]
c_values = [1, 10]
svm = []

for i, date in enumerate(start_list):
    end_date = end_list[i]
    for t in thresholds:
        for c in c_values:
            model_name = 'svm_' + date + '_' + str(t) + '_' + str(c)
            s_test, s_predict, s_score = pipeline_revised.SVM_classify(d
f, 'time_tf', c, threshold = t,

predictors=['total_price_including_optional_support', 'students_reached'
,'poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','re
source_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subj
ect_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject
_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Spo
rts',
       'primary_focus_subject_Visual Arts'],temporal=True, start_col='da
te_posted',end_col='datefullyfunded',start_date=date,end_date=end_date)
            svm.append((model_name, s_test, s_predict, s_score))
```

Let's now create a table showing the performance of the different metrics.

In [24]:
```
svm_metrics = pd.DataFrame(columns=['name','accuracy','precision','auc',
'recall','f1'])

for i in range(len(svm)):
    name, test, predict, _, = svm[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_mod
el(test, predict)
    svm_metrics.loc[i,'name'] = name
    svm_metrics.loc[i,'accuracy'] = accuracy
    svm_metrics.loc[i,'precision'] = precision
    svm_metrics.loc[i,'auc'] = auc
    svm_metrics.loc[i,'recall'] = recall
    svm_metrics.loc[i,'f1'] = f1

svm_metrics
```

Out[24]:

	name	accuracy	precision	auc	recall	f1
0	svm_2013-06-30_0.01_1	0.830952	0.259259	0.500516	0.00244541	0.00484513
1	svm_2013-06-30_0.01_10	0.409289	0.204798	0.593332	0.870742	0.331604
2	svm_2013-06-30_0.02_1	0.830952	0.259259	0.500516	0.00244541	0.00484513
3	svm_2013-06-30_0.02_10	0.41261	0.205243	0.593866	0.867074	0.331918
4	svm_2013-06-30_0.05_1	0.831099	0.266667	0.500465	0.00209607	0.00415945
5	svm_2013-06-30_0.05_10	0.424103	0.206792	0.59562	0.854148	0.33297
6	svm_2013-06-30_0.1_1	0.831158	0.243243	0.500291	0.00157205	0.00312392
7	svm_2013-06-30_0.1_10	0.442769	0.209289	0.597993	0.831965	0.334445
8	svm_2013-06-30_0.2_1	0.831217	0.206897	0.500118	0.00104803	0.00208551
9	svm_2013-06-30_0.2_10	0.479012	0.214432	0.601808	0.7869	0.337024
10	svm_2013-06-30_0.3_1	0.831335	0.217391	0.500119	0.000873362	0.00173974
11	svm_2013-06-30_0.3_10	0.516931	0.21885	0.601126	0.728035	0.336536
12	svm_2013-06-30_0.5_1	0.83154	0.3125	0.500242	0.000873362	0.00174186
13	svm_2013-06-30_0.5_10	0.586508	0.223885	0.588197	0.590742	0.324708
14	svm_2013-05-31_0.01_1	0.841304	0	0.5	0	0
15	svm_2013-05-31_0.01_10	0.787834	0.148786	0.497175	0.0713681	0.0964649
16	svm_2013-05-31_0.02_1	0.841304	0	0.5	0	0
17	svm_2013-05-31_0.02_10	0.788271	0.149021	0.497263	0.0709445	0.0961263
18	svm_2013-05-31_0.05_1	0.841304	0	0.5	0	0
19	svm_2013-05-31_0.05_10	0.78985	0.149336	0.497428	0.0690385	0.0944243
20	svm_2013-05-31_0.1_1	0.841304	0	0.5	0	0
21	svm_2013-05-31_0.1_10	0.792203	0.148629	0.497366	0.0654384	0.090869
22	svm_2013-05-31_0.2_1	0.841304	0	0.5	0	0
23	svm_2013-05-31_0.2_10	0.79637	0.149081	0.497695	0.060144	0.08571
24	svm_2013-05-31_0.3_1	0.841304	0	0.5	0	0
25	svm_2013-05-31_0.3_10	0.799798	0.150933	0.498272	0.0565438	0.0822678
26	svm_2013-05-31_0.5_1	0.841304	0	0.5	0	0
27	svm_2013-05-31_0.5_10	0.805411	0.145418	0.497483	0.0463787	0.0703276
28	svm_2013-04-30_0.01_1	0.842942	0	0.5	0	0
29	svm_2013-04-30_0.01_10	0.842942	0	0.5	0	0
30	svm_2013-04-30_0.02_1	0.842942	0	0.5	0	0
31	svm_2013-04-30_0.02_10	0.842942	0	0.5	0	0
32	svm_2013-04-30_0.05_1	0.842942	0	0.5	0	0
33	svm_2013-04-30_0.05_10	0.842942	0	0.5	0	0

	name	accuracy	precision	auc	recall	f1
34	svm_2013-04-30_0.1_1	0.842942	0	0.5	0	0
35	svm_2013-04-30_0.1_10	0.842942	0	0.5	0	0
36	svm_2013-04-30_0.2_1	0.842942	0	0.5	0	0
37	svm_2013-04-30_0.2_10	0.842942	0	0.5	0	0
38	svm_2013-04-30_0.3_1	0.842942	0	0.5	0	0
39	svm_2013-04-30_0.3_10	0.842942	0	0.5	0	0
40	svm_2013-04-30_0.5_1	0.842942	0	0.5	0	0
41	svm_2013-04-30_0.5_10	0.842942	0	0.5	0	0

As we can see above, model # 10, with threshold of .3, has the highest area under the curve, but mediocre accuracy.

Random Forest

Note - random forest models incorporate bagging; thus this serves for both the random forest and bagging model requirements outline in the assignment.

In [25]:
```
start_list = ['2013-06-30', '2013-05-31', '2013-04-30']
end_list = ['2013-12-31', '2013-11-30','2013-10-31']
thresholds = [.01,.02,.05,.1,.2,.3,.5]
max_depth = [3, 4]
forest = []

for i, date in enumerate(start_list):
    end_date = end_list[i]
    for t in thresholds:
        for d in max_depth:
            model_name = 'rf_' + date + '_' + str(t) + '_' + str(d)
            f_test, f_predict, f_score, features = pipeline_revised.rand
om_forest(df, 'time_tf', d, threshold = t,

predictors=['total_price_including_optional_support', 'students_reached'
,'poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','re
source_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subj
ect_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject
_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Spo
rts',
       'primary_focus_subject_Visual Arts'],temporal=True, start_col='da
te_posted',end_col='datefullyfunded',start_date=date,end_date=end_date)
            forest.append((model_name, f_test, f_predict, f_score, featu
res))
```

We can now produce a chart of the models' relative performances.

In [26]:
```
rf_metrics = pd.DataFrame(columns=['name','accuracy','precision','auc',
'recall','f1'])

for i in range(len(forest)):
    name, test, predict, _,_ = forest[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_mod
el(test, predict)
    rf_metrics.loc[i,'name'] = name
    rf_metrics.loc[i,'accuracy'] = accuracy
    rf_metrics.loc[i,'precision'] = precision
    rf_metrics.loc[i,'auc'] = auc
    rf_metrics.loc[i,'recall'] = recall
    rf_metrics.loc[i,'f1'] = f1

rf_metrics
```

Out[26]:

	name	accuracy	precision	auc	recall	f1
0	rf_2013-06-30_0.01_3	0.168283	0.168283	0.5	1	0.288087
1	rf_2013-06-30_0.01_4	0.168283	0.168283	0.5	1	0.288087
2	rf_2013-06-30_0.02_3	0.168283	0.168283	0.5	1	0.288087
3	rf_2013-06-30_0.02_4	0.168283	0.168283	0.5	1	0.288087
4	rf_2013-06-30_0.05_3	0.168283	0.168283	0.5	1	0.288087
5	rf_2013-06-30_0.05_4	0.168283	0.168283	0.5	1	0.288087
6	rf_2013-06-30_0.1_3	0.168283	0.168283	0.5	1	0.288087
7	rf_2013-06-30_0.1_4	0.168283	0.168283	0.5	1	0.288087
8	rf_2013-06-30_0.2_3	0.232569	0.177045	0.529033	0.975895	0.299716
9	rf_2013-06-30_0.2_4	0.280218	0.183267	0.5466	0.948122	0.307161
10	rf_2013-06-30_0.3_3	0.688948	0.242662	0.573706	0.4	0.302071
11	rf_2013-06-30_0.3_4	0.665697	0.234387	0.5738	0.435284	0.304701
12	rf_2013-06-30_0.5_3	0.831717	0	0.5	0	0
13	rf_2013-06-30_0.5_4	0.831717	0	0.5	0	0
14	rf_2013-05-31_0.01_3	0.158696	0.158696	0.5	1	0.273922
15	rf_2013-05-31_0.01_4	0.158696	0.158696	0.5	1	0.273922
16	rf_2013-05-31_0.02_3	0.158696	0.158696	0.5	1	0.273922
17	rf_2013-05-31_0.02_4	0.158696	0.158696	0.5	1	0.273922
18	rf_2013-05-31_0.05_3	0.158696	0.158696	0.5	1	0.273922
19	rf_2013-05-31_0.05_4	0.158696	0.158696	0.5	1	0.273922
20	rf_2013-05-31_0.1_3	0.158696	0.158696	0.5	1	0.273922
21	rf_2013-05-31_0.1_4	0.158696	0.158696	0.5	1	0.273922
22	rf_2013-05-31_0.2_3	0.160376	0.158917	0.500827	0.999576	0.274235
23	rf_2013-05-31_0.2_4	0.230852	0.167788	0.531285	0.97141	0.286151
24	rf_2013-05-31_0.3_3	0.817409	0.264101	0.519992	0.0842863	0.127789
25	rf_2013-05-31_0.3_4	0.641069	0.216933	0.577138	0.483482	0.299488
26	rf_2013-05-31_0.5_3	0.841304	0	0.5	0	0
27	rf_2013-05-31_0.5_4	0.841304	0	0.5	0	0
28	rf_2013-04-30_0.01_3	0.157058	0.157058	0.5	1	0.271478
29	rf_2013-04-30_0.01_4	0.157058	0.157058	0.5	1	0.271478
30	rf_2013-04-30_0.02_3	0.157058	0.157058	0.5	1	0.271478
31	rf_2013-04-30_0.02_4	0.157058	0.157058	0.5	1	0.271478
32	rf_2013-04-30_0.05_3	0.157058	0.157058	0.5	1	0.271478
33	rf_2013-04-30_0.05_4	0.157058	0.157058	0.5	1	0.271478

	name	accuracy	precision	auc	recall	f1
34	rf_2013-04-30_0.1_3	0.157058	0.157058	0.5	1	0.271478
35	rf_2013-04-30_0.1_4	0.157058	0.157058	0.5	1	0.271478
36	rf_2013-04-30_0.2_3	0.329735	0.17938	0.567472	0.914085	0.299906
37	rf_2013-04-30_0.2_4	0.301439	0.176258	0.560644	0.938558	0.296781
38	rf_2013-04-30_0.3_3	0.734217	0.210285	0.537721	0.251237	0.228944
39	rf_2013-04-30_0.3_4	0.712463	0.212159	0.547167	0.30617	0.250639
40	rf_2013-04-30_0.5_3	0.842942	0	0.5	0	0
41	rf_2013-04-30_0.5_4	0.842942	0	0.5	0	0

We can see that model 7 has the highest area under the curve, moderate accuract, and an average f1 (combined precision/recall) score among models.

In [27]:
```
#If we wanted to see the feature values (associated with predictors inpu
t)
#forest[7][4]
```

Gradient Boosting

In [28]:
```
start_list = ['2013-06-30', '2013-05-31', '2013-04-30']
end_list = ['2013-12-31', '2013-11-30','2013-10-31']
thresholds = [.01,.02,.05,.1,.2,.3,.5]
max_features = [3, 4]
gradient = []

for i, date in enumerate(start_list):
    end_date = end_list[i]
    for t in thresholds:
        for f in max_features:
            model_name = 'gradient_' + date + '_' + str(t) + '_' + str(f
)
            g_test, g_predict, g_score = pipeline_revised.gradient_boost
(df, 'time_tf', f, threshold = t,

predictors=['total_price_including_optional_support', 'students_reached'
,'poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','re
source_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subj
ect_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject
_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Spo
rts',
       'primary_focus_subject_Visual Arts'],temporal=True, start_col='da
te_posted',end_col='datefullyfunded',start_date=date,end_date=end_date)
            gradient.append((model_name, g_test, g_predict, g_score))
```

We can now produce a chart of the models' relative performances.

In [29]:
```
gradient_metrics = pd.DataFrame(columns=['name','accuracy','precision',
'auc','recall','f1'])

for i in range(len(gradient)):
    name, test, predict, _ = gradient[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_mod
el(test, predict)
    gradient_metrics.loc[i,'name'] = name
    gradient_metrics.loc[i,'accuracy'] = accuracy
    gradient_metrics.loc[i,'precision'] = precision
    gradient_metrics.loc[i,'auc'] = auc
    gradient_metrics.loc[i,'recall'] = recall
    gradient_metrics.loc[i,'f1'] = f1

gradient_metrics
```

Out[29]:

	name	accuracy	precision	auc	recall	f1
0	gradient_2013-06-30_0.01_3	0.168283	0.168283	0.5	1	0.288087
1	gradient_2013-06-30_0.01_4	0.168283	0.168283	0.5	1	0.288087
2	gradient_2013-06-30_0.02_3	0.168283	0.168283	0.5	1	0.288087
3	gradient_2013-06-30_0.02_4	0.168283	0.168283	0.5	1	0.288087
4	gradient_2013-06-30_0.05_3	0.168283	0.168283	0.5	1	0.288087
5	gradient_2013-06-30_0.05_4	0.168283	0.168283	0.5	1	0.288087
6	gradient_2013-06-30_0.1_3	0.171282	0.168692	0.501454	0.999127	0.288648
7	gradient_2013-06-30_0.1_4	0.177014	0.169471	0.504203	0.99738	0.289715
8	gradient_2013-06-30_0.2_3	0.38739	0.202113	0.590129	0.895721	0.329807
9	gradient_2013-06-30_0.2_4	0.390329	0.202229	0.589876	0.890655	0.329616
10	gradient_2013-06-30_0.3_3	0.597178	0.233554	0.602623	0.61083	0.337907
11	gradient_2013-06-30_0.3_4	0.585509	0.231642	0.603828	0.631441	0.338943
12	gradient_2013-06-30_0.5_3	0.831481	0.357143	0.500555	0.00174672	0.00347645
13	gradient_2013-06-30_0.5_4	0.831305	0.37037	0.501146	0.00349345	0.00692161
14	gradient_2013-05-31_0.01_3	0.158696	0.158696	0.5	1	0.273922
15	gradient_2013-05-31_0.01_4	0.158696	0.158696	0.5	1	0.273922
16	gradient_2013-05-31_0.02_3	0.158696	0.158696	0.5	1	0.273922
17	gradient_2013-05-31_0.02_4	0.158696	0.158696	0.5	1	0.273922
18	gradient_2013-05-31_0.05_3	0.158763	0.158707	0.50004	1	0.273938
19	gradient_2013-05-31_0.05_4	0.158763	0.158707	0.50004	1	0.273938
20	gradient_2013-05-31_0.1_3	0.171097	0.160388	0.506253	0.997247	0.276334
21	gradient_2013-05-31_0.1_4	0.184944	0.162251	0.512936	0.993435	0.278944
22	gradient_2013-05-31_0.2_3	0.393346	0.191758	0.58997	0.878018	0.314771
23	gradient_2013-05-31_0.2_4	0.391934	0.191329	0.588959	0.877594	0.314165
24	gradient_2013-05-31_0.3_3	0.61015	0.21742	0.589949	0.560356	0.313284
25	gradient_2013-05-31_0.3_4	0.611124	0.218125	0.590872	0.561203	0.314149
26	gradient_2013-05-31_0.5_3	0.841304	0.5	0.500344	0.000847099	0.00169133
27	gradient_2013-05-31_0.5_4	0.841506	0.75	0.500893	0.00190597	0.00380228
28	gradient_2013-04-30_0.01_3	0.157058	0.157058	0.5	1	0.271478
29	gradient_2013-04-30_0.01_4	0.157058	0.157058	0.5	1	0.271478
30	gradient_2013-04-30_0.02_3	0.157058	0.157058	0.5	1	0.271478
31	gradient_2013-04-30_0.02_4	0.157058	0.157058	0.5	1	0.271478
32	gradient_2013-04-30_0.05_3	0.157098	0.157064	0.500024	1	0.271487
33	gradient_2013-04-30_0.05_4	0.157098	0.157064	0.500024	1	0.271487

	name	accuracy	precision	auc	recall	f1
34	gradient_2013-04-30_0.1_3	0.183063	0.160567	0.512884	0.993752	0.276464
35	gradient_2013-04-30_0.1_4	0.183145	0.160581	0.512932	0.993752	0.276484
36	gradient_2013-04-30_0.2_3	0.404809	0.190622	0.589759	0.859412	0.312033
37	gradient_2013-04-30_0.2_4	0.406322	0.19103	0.590657	0.859412	0.31258
38	gradient_2013-04-30_0.3_3	0.650474	0.22138	0.583906	0.486852	0.304362
39	gradient_2013-04-30_0.3_4	0.626881	0.219473	0.590778	0.538141	0.311788
40	gradient_2013-04-30_0.5_3	0.842902	0.4	0.500188	0.000520698	0.00104004
41	gradient_2013-04-30_0.5_4	0.842902	0.333333	0.500082	0.000260349	0.000520291

We can see that model 11 has highest area under the curve and one of the higher accuracies.

Baseline Classifier

We will use sklearn's dummy classifier as a baseline for accuracy

In [30]:
```
start_list = ['2013-06-30', '2013-05-31', '2013-04-30']
end_list = ['2013-12-31', '2013-11-30','2013-10-31']
thresholds = [.01,.02,.05,.1,.2,.3,.5]
dummy = []

for i, date in enumerate(start_list):
    end_date = end_list[i]
    for t in thresholds:
        model_name = 'dummy_' + date + '_' + str(t)
        d_test, d_predict, d_score = pipeline_revised.dummy_baseline(df,
'time_tf', threshold = t,

predictors=['total_price_including_optional_support', 'students_reached'
,'poverty_level_high poverty', 'poverty_level_highest poverty',
       'poverty_level_low poverty', 'poverty_level_moderate poverty','re
source_type_Books', 'resource_type_Other', 'resource_type_Supplies',
       'resource_type_Technology', 'resource_type_Trips',
       'resource_type_Visitors','school_metro_rural',
       'school_metro_suburban', 'school_metro_urban','primary_focus_subj
ect_Applied Sciences',
       'primary_focus_subject_Character Education',
       'primary_focus_subject_Civics & Government',
       'primary_focus_subject_College & Career Prep',
       'primary_focus_subject_Community Service', 'primary_focus_subject
_ESL',
       'primary_focus_subject_Early Development',
       'primary_focus_subject_Economics',
       'primary_focus_subject_Environmental Science',
       'primary_focus_subject_Extracurricular',
       'primary_focus_subject_Foreign Languages',
       'primary_focus_subject_Gym & Fitness',
       'primary_focus_subject_Health & Life Science',
       'primary_focus_subject_Health & Wellness',
       'primary_focus_subject_History & Geography',
       'primary_focus_subject_Literacy',
       'primary_focus_subject_Literature & Writing',
       'primary_focus_subject_Mathematics', 'primary_focus_subject_Musi
c',
       'primary_focus_subject_Nutrition', 'primary_focus_subject_Other',
       'primary_focus_subject_Parent Involvement',
       'primary_focus_subject_Performing Arts',
       'primary_focus_subject_Social Sciences',
       'primary_focus_subject_Special Needs', 'primary_focus_subject_Spo
rts',
       'primary_focus_subject_Visual Arts'],temporal=True, start_col='da
te_posted',end_col='datefullyfunded',start_date=date,end_date=end_date)
       dummy.append((model_name, d_test, d_predict, d_score))
```

Now let's compute the performance metrics for the dummy variables generated.

In [31]:
```
dummy_metrics = pd.DataFrame(columns=['name','accuracy','precision','au
c','recall','f1'])

for i in range(len(dummy)):
    name, test, predict, _ = dummy[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_mod
el(test, predict)
    dummy_metrics.loc[i,'name'] = name
    dummy_metrics.loc[i,'accuracy'] = accuracy
    dummy_metrics.loc[i,'precision'] = precision
    dummy_metrics.loc[i,'auc'] = auc
    dummy_metrics.loc[i,'recall'] = recall
    dummy_metrics.loc[i,'f1'] = f1

dummy_metrics
```

Out[31]:

	name	accuracy	precision	auc	recall	f1
0	dummy_2013-06-30_0.01	0.652939	0.165088	0.496953	0.261834	0.202499
1	dummy_2013-06-30_0.02	0.651969	0.168851	0.50055	0.272314	0.20845
2	dummy_2013-06-30_0.05	0.657907	0.1727	0.504189	0.272489	0.211411
3	dummy_2013-06-30_0.1	0.654615	0.165612	0.497473	0.260611	0.202525
4	dummy_2013-06-30_0.2	0.653762	0.166336	0.498145	0.263581	0.20396
5	dummy_2013-06-30_0.3	0.653616	0.167124	0.498893	0.265677	0.20518
6	dummy_2013-06-30_0.5	0.65388	0.16882	0.500515	0.269345	0.207551
7	dummy_2013-05-31_0.01	0.662847	0.161006	0.502277	0.267048	0.200892
8	dummy_2013-05-31_0.02	0.658746	0.157244	0.498552	0.263871	0.197058
9	dummy_2013-05-31_0.05	0.659822	0.158832	0.500136	0.266201	0.198955
10	dummy_2013-05-31_0.1	0.65878	0.157782	0.499087	0.265142	0.197835
11	dummy_2013-05-31_0.2	0.664695	0.159959	0.501228	0.261753	0.19857
12	dummy_2013-05-31_0.3	0.655722	0.152092	0.493403	0.255612	0.190709
13	dummy_2013-05-31_0.5	0.662645	0.160883	0.502157	0.267048	0.200796
14	dummy_2013-04-30_0.01	0.668588	0.162124	0.504937	0.266337	0.201556
15	dummy_2013-04-30_0.02	0.659266	0.150482	0.493475	0.251757	0.188371
16	dummy_2013-04-30_0.05	0.664949	0.154359	0.497376	0.253059	0.191754
17	dummy_2013-04-30_0.1	0.664704	0.154322	0.497336	0.253319	0.1918
18	dummy_2013-04-30_0.2	0.669529	0.161208	0.504012	0.262692	0.199802
19	dummy_2013-04-30_0.3	0.665113	0.160817	0.503722	0.26842	0.201131
20	dummy_2013-04-30_0.5	0.662659	0.155385	0.498348	0.258787	0.194179

As we can see here, the performance of the dummy model on dimensions like AUC/accuracy does not achieve that of the "highest performing" models on those dimensions; however, it is not that far off. This will be described in the write-up in more depth.