

Homework 3 - Code Outputs

Allison Collins

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
```

```
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 average 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 [11]: 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) + '_' + str(depth)

            y_test, y_predict, y_score, features = pipeline_revised.create_decision_tree(df, 'time_tf', depth, 500, tree_name, 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', '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'], temporal=True, start_col='date_posted', end_col='date_fully_funded', 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 [12]: 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_model(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
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.p  
y:1143: UndefinedMetricWarning: Precision is ill-defined and being set  
to 0.0 due to no predicted samples.  
    'precision', 'predicted', average, warn_for)  
/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.p  
y:1143: UndefinedMetricWarning: F-score is ill-defined and being set to  
0.0 due to no predicted samples.  
    'precision', 'predicted', average, warn_for)
```

Out[12]:

	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	0.354909	0.195151	0.575061	0.9069	0.321188
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	0.210611	0.174843	0.522381	0.992314	0.297302
11	tree_2013-06-30_0.2_4	0.381717	0.200423	0.586231	0.894498	0.327472
12	tree_2013-06-30_0.3_4	0.508201	0.217331	0.600267	0.739039	0.335887
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	0.235187	0.169531	0.537212	0.97967	0.289044
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	0.55883	0.207326	0.587887	0.630453	0.312038
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	0.202084	0.164834	0.52192	0.99047	0.282632
25	tree_2013-05-31_0.2_4	0.380407	0.188006	0.581077	0.875053	0.309513
26	tree_2013-05-31_0.3_4	0.506033	0.20228	0.591905	0.717704	0.315608
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	0.236997	0.168226	0.53852	0.978131	0.287079
32	tree_2013-04-30_0.2_3	0.404236	0.189285	0.585924	0.85082	0.309675
33	tree_2013-04-30_0.3_3	0.571394	0.207934	0.589323	0.615465	0.310848

	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	0.202527	0.163299	0.522416	0.988805	0.280306
39	tree_2013-04-30_0.2_4	0.404359	0.189354	0.586103	0.85108	0.309784
40	tree_2013-04-30_0.3_4	0.516887	0.201214	0.590993	0.699037	0.312482
41	tree_2013-04-30_0.5_4	0.842942	0	0.5	0	0

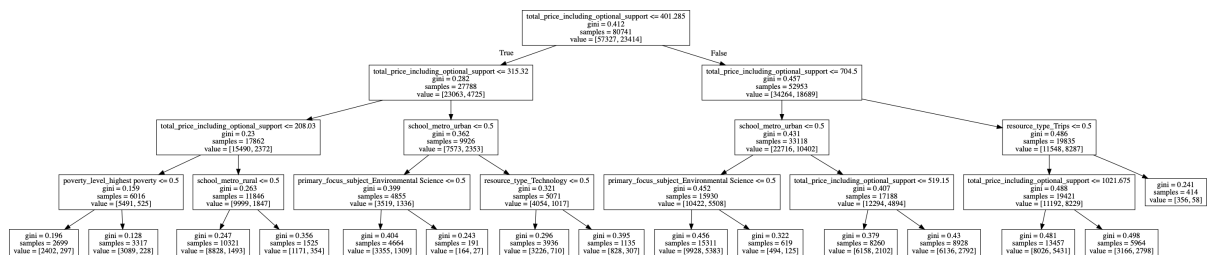
Some of trees (as will happen with later models) have precision / recall of 0 because they were lacking true positives. Let's visualize the one which has the highest area under the curve.

```
In [13]: (graph,) = pydot.graph_from_dot_file('tree_2013-06-30_0.3_4')
```

```
In [14]: graph.write_png('tree_1.png')
```

```
In [15]: from IPython.display import Image
Image("tree_1.png")
```

Out[15]:



```

In [16]: #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]

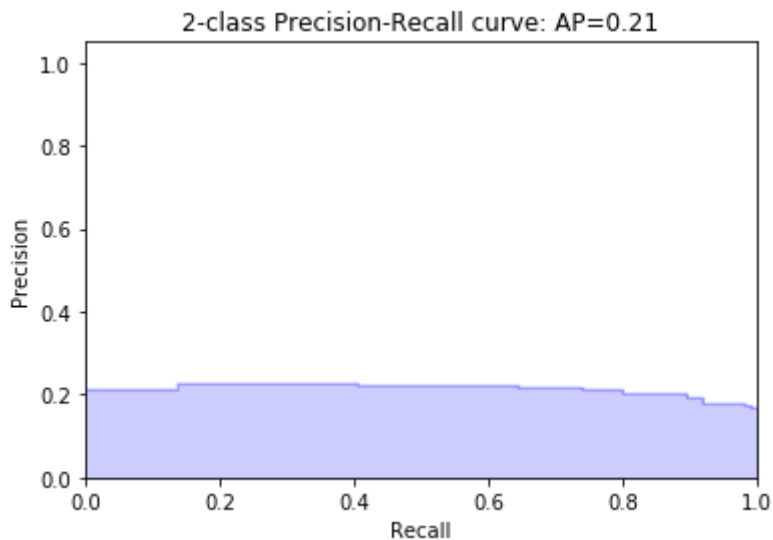
precision, recall, _ = precision_recall_curve(test, clean_score)
average_precision = average_precision_score(test, clean_score)

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[16]: 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 [17]: feature_scores = (trees[12][4])
feature_names = ['total_price_including_optional_support', 'students_rea
ched', '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']

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

```

Out[17]:

	Features	Importance
0	total_price_including_optional_support	0.883423
14	school_metro_urban	0.045634
10	resource_type_Trips	0.033358
23	primary_focus_subject_Environmental Science	0.017271
12	school_metro_rural	0.010401
9	resource_type_Technology	0.007316
3	poverty_level_highest poverty	0.002597
32	primary_focus_subject_Mathematics	0.000000
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
39	primary_focus_subject_Special Needs	0.000000
38	primary_focus_subject_Social Sciences	0.000000
33	primary_focus_subject_Music	0.000000
24	primary_focus_subject_Extracurricular	0.000000
34	primary_focus_subject_Nutrition	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
40	primary_focus_subject_Sports	0.000000
21	primary_focus_subject_Early Development	0.000000
22	primary_focus_subject_Economics	0.000000
11	resource_type_Visitors	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
13	school_metro_suburban	0.000000

	Features	Importance
1	students_reached	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 [18]: 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(threshold) + '_' + metric
                p_test, p_predict, p_score, params = pipeline_revised.knn_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', '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'], temporal=True, start_col='date_posted', end_col='date_fully_funded', 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 [19]: 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_model(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[19]:

	name	accuracy	precision	auc	recall	f1
0	knn_2013-06-30_10_0.01_euclidean	0.240182	0.176879	0.528106	0.962096	0.29882
1	knn_2013-06-30_10_0.02_euclidean	0.240182	0.176879	0.528106	0.962096	0.29882
2	knn_2013-06-30_10_0.05_euclidean	0.240182	0.176879	0.528106	0.962096	0.29882
3	knn_2013-06-30_10_0.1_euclidean	0.355026	0.188267	0.554581	0.855371	0.308609
4	knn_2013-06-30_10_0.2_euclidean	0.487507	0.199219	0.563235	0.67738	0.307888
5	knn_2013-06-30_10_0.3_euclidean	0.613639	0.210715	0.557135	0.471965	0.291352
6	knn_2013-06-30_10_0.5_euclidean	0.780482	0.230269	0.52103	0.129956	0.166146
7	knn_2013-06-30_10_0.01_minkowski	0.240182	0.176879	0.528106	0.962096	0.29882
8	knn_2013-06-30_10_0.02_minkowski	0.240182	0.176879	0.528106	0.962096	0.29882
9	knn_2013-06-30_10_0.05_minkowski	0.240182	0.176879	0.528106	0.962096	0.29882
10	knn_2013-06-30_10_0.1_minkowski	0.355026	0.188267	0.554581	0.855371	0.308609
11	knn_2013-06-30_10_0.2_minkowski	0.487507	0.199219	0.563235	0.67738	0.307888
12	knn_2013-06-30_10_0.3_minkowski	0.613639	0.210715	0.557135	0.471965	0.291352
13	knn_2013-06-30_10_0.5_minkowski	0.780482	0.230269	0.52103	0.129956	0.166146
14	knn_2013-06-30_50_0.01_euclidean	0.169929	0.168502	0.500781	0.999476	0.288385
15	knn_2013-06-30_50_0.02_euclidean	0.175779	0.169378	0.503879	0.998428	0.289623
16	knn_2013-06-30_50_0.05_euclidean	0.188477	0.171397	0.510886	0.996856	0.292502
17	knn_2013-06-30_50_0.1_euclidean	0.249941	0.180134	0.538501	0.97345	0.304012
18	knn_2013-06-30_50_0.2_euclidean	0.387654	0.197896	0.577749	0.864279	0.322052
19	knn_2013-06-30_50_0.3_euclidean	0.554674	0.214446	0.579997	0.618166	0.318427
20	knn_2013-06-30_50_0.5_euclidean	0.810788	0.252434	0.512707	0.0634061	0.101354
21	knn_2013-06-30_50_0.01_minkowski	0.169929	0.168502	0.500781	0.999476	0.288385
22	knn_2013-06-30_50_0.02_minkowski	0.175779	0.169378	0.503879	0.998428	0.289623
23	knn_2013-06-30_50_0.05_minkowski	0.188477	0.171397	0.510886	0.996856	0.292502
24	knn_2013-06-30_50_0.1_minkowski	0.249941	0.180134	0.538501	0.97345	0.304012
25	knn_2013-06-30_50_0.2_minkowski	0.387654	0.197896	0.577749	0.864279	0.322052
26	knn_2013-06-30_50_0.3_minkowski	0.554674	0.214446	0.579997	0.618166	0.318427
27	knn_2013-06-30_50_0.5_minkowski	0.810788	0.252434	0.512707	0.0634061	0.101354
28	knn_2013-05-31_10_0.01_euclidean	0.23307	0.166826	0.527792	0.959551	0.284236
29	knn_2013-05-31_10_0.02_euclidean	0.23307	0.166826	0.527792	0.959551	0.284236
...
54	knn_2013-05-31_50_0.3_minkowski	0.565787	0.201544	0.574065	0.586192	0.299957
55	knn_2013-05-31_50_0.5_minkowski	0.822047	0.228436	0.50926	0.0510377	0.0834343
56	knn_2013-04-30_10_0.01_euclidean	0.23757	0.165816	0.529962	0.956261	0.282625

	name	accuracy	precision	auc	recall	f1
57	knn_2013-04-30_10_0.02_euclidean	0.23757	0.165816	0.529962	0.956261	0.282625
58	knn_2013-04-30_10_0.05_euclidean	0.23757	0.165816	0.529962	0.956261	0.282625
59	knn_2013-04-30_10_0.1_euclidean	0.36212	0.174968	0.550033	0.824004	0.288646
60	knn_2013-04-30_10_0.2_euclidean	0.502085	0.185955	0.559228	0.642541	0.288436
61	knn_2013-04-30_10_0.3_euclidean	0.635631	0.200284	0.55646	0.441031	0.27547
62	knn_2013-04-30_10_0.5_euclidean	0.79551	0.215965	0.518576	0.114814	0.149924
63	knn_2013-04-30_10_0.01_minkowski	0.23757	0.165816	0.529962	0.956261	0.282625
64	knn_2013-04-30_10_0.02_minkowski	0.23757	0.165816	0.529962	0.956261	0.282625
65	knn_2013-04-30_10_0.05_minkowski	0.23757	0.165816	0.529962	0.956261	0.282625
66	knn_2013-04-30_10_0.1_minkowski	0.36212	0.174968	0.550033	0.824004	0.288646
67	knn_2013-04-30_10_0.2_minkowski	0.502085	0.185955	0.559228	0.642541	0.288436
68	knn_2013-04-30_10_0.3_minkowski	0.635631	0.200284	0.55646	0.441031	0.27547
69	knn_2013-04-30_10_0.5_minkowski	0.79551	0.215965	0.518576	0.114814	0.149924
70	knn_2013-04-30_50_0.01_euclidean	0.158325	0.157257	0.500752	1	0.271775
71	knn_2013-04-30_50_0.02_euclidean	0.163641	0.157788	0.50274	0.997136	0.272462
72	knn_2013-04-30_50_0.05_euclidean	0.178852	0.159874	0.510385	0.993752	0.275437
73	knn_2013-04-30_50_0.1_euclidean	0.247506	0.168171	0.537656	0.960687	0.286235
74	knn_2013-04-30_50_0.2_euclidean	0.397857	0.185095	0.574832	0.832856	0.302878
75	knn_2013-04-30_50_0.3_euclidean	0.584724	0.206743	0.582613	0.579537	0.304765
76	knn_2013-04-30_50_0.5_euclidean	0.824051	0.222356	0.50839	0.0481645	0.0791783
77	knn_2013-04-30_50_0.01_minkowski	0.158325	0.157257	0.500752	1	0.271775
78	knn_2013-04-30_50_0.02_minkowski	0.163641	0.157788	0.50274	0.997136	0.272462
79	knn_2013-04-30_50_0.05_minkowski	0.178852	0.159874	0.510385	0.993752	0.275437
80	knn_2013-04-30_50_0.1_minkowski	0.247506	0.168171	0.537656	0.960687	0.286235
81	knn_2013-04-30_50_0.2_minkowski	0.397857	0.185095	0.574832	0.832856	0.302878
82	knn_2013-04-30_50_0.3_minkowski	0.584724	0.206743	0.582613	0.579537	0.304765
83	knn_2013-04-30_50_0.5_minkowski	0.824051	0.222356	0.50839	0.0481645	0.0791783

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 [20]: #neighbors[80][4]
```

Logistic Classifier

```

In [21]: 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 [22]: 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_model(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[22]:

	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	0.16846	0.168313	0.500106	1	0.28813
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	0.168342	0.168293	0.500035	1	0.288101
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	0.169518	0.168472	0.500673	0.999825	0.288355
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	0.169312	0.168437	0.500549	0.999825	0.288305
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	0.170047	0.168522	0.500851	0.999476	0.288415
12	reg_2013-06-30_0.1_l1_0.01	0.16843	0.168308	0.500088	1	0.288123
13	reg_2013-06-30_0.1_l1_1	0.170223	0.168513	0.500818	0.999127	0.288386
14	reg_2013-06-30_0.1_l2_0.01	0.168607	0.168338	0.500194	1	0.288166
15	reg_2013-06-30_0.1_l2_1	0.170165	0.168503	0.500783	0.999127	0.288372
16	reg_2013-06-30_0.2_l1_0.01	0.188095	0.170677	0.508357	0.991092	0.291206
17	reg_2013-06-30_0.2_l1_1	0.240858	0.17703	0.528582	0.962271	0.299045
18	reg_2013-06-30_0.2_l2_0.01	0.221017	0.174459	0.520695	0.972402	0.295842
19	reg_2013-06-30_0.2_l2_1	0.240917	0.177083	0.528757	0.96262	0.299137
20	reg_2013-06-30_0.3_l1_0.01	0.596502	0.22223	0.581595	0.559127	0.318049
21	reg_2013-06-30_0.3_l1_1	0.592416	0.222548	0.583598	0.570306	0.320161
22	reg_2013-06-30_0.3_l2_0.01	0.592416	0.221409	0.581508	0.565066	0.318155
23	reg_2013-06-30_0.3_l2_1	0.592828	0.221926	0.582313	0.566463	0.31891
24	reg_2013-06-30_0.5_l1_0.01	0.819871	0.189522	0.501447	0.0214847	0.0385943
25	reg_2013-06-30_0.5_l1_1	0.813757	0.199015	0.503275	0.0352838	0.0599407
26	reg_2013-06-30_0.5_l2_0.01	0.819753	0.20292	0.502491	0.0242795	0.0433697
27	reg_2013-06-30_0.5_l2_1	0.813962	0.200397	0.503399	0.0352838	0.060003
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	0.158898	0.158728	0.50012	1	0.273969
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	0.158696	0.158696	0.5	1	0.273922
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	0.159738	0.158816	0.500447	0.999576	0.274084

	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	0.158864	0.158723	0.5001	1	0.273961
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	0.160175	0.158862	0.500621	0.999365	0.274145
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	0.15994	0.158848	0.500567	0.999576	0.274132
40	reg_2013-05-31_0.1_l1_0.01	0.15883	0.158717	0.50008	1	0.273954
41	reg_2013-05-31_0.1_l1_1	0.160477	0.158887	0.500715	0.999153	0.274175
42	reg_2013-05-31_0.1_l2_0.01	0.158965	0.158739	0.50016	1	0.273985
43	reg_2013-05-31_0.1_l2_1	0.160208	0.158867	0.500641	0.999365	0.274153
44	reg_2013-05-31_0.2_l1_0.01	0.183062	0.161563	0.510443	0.990047	0.277794
45	reg_2013-05-31_0.2_l1_1	0.243119	0.168164	0.53196	0.955104	0.285977
46	reg_2013-05-31_0.2_l2_0.01	0.222887	0.16591	0.525005	0.967598	0.283252
47	reg_2013-05-31_0.2_l2_1	0.241338	0.167783	0.53073	0.95468	0.285407
48	reg_2013-05-31_0.3_l1_0.01	0.617073	0.208085	0.571039	0.5036	0.294489
49	reg_2013-05-31_0.3_l1_1	0.608704	0.208884	0.575086	0.525837	0.298995
50	reg_2013-05-31_0.3_l2_0.01	0.609679	0.20752	0.572401	0.517789	0.296292
51	reg_2013-05-31_0.3_l2_1	0.608469	0.208172	0.573915	0.523295	0.297854
52	reg_2013-05-31_0.5_l1_0.01	0.833104	0.228889	0.503976	0.0218128	0.0398299
53	reg_2013-05-31_0.5_l1_1	0.827726	0.220222	0.505591	0.0336722	0.0584129
54	reg_2013-05-31_0.5_l2_0.01	0.832902	0.238494	0.504801	0.0241423	0.0438462
55	reg_2013-05-31_0.5_l2_1	0.829104	0.229508	0.50598	0.0326133	0.0571111
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	0.15718	0.157077	0.500073	1	0.271506
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	0.157098	0.157064	0.500024	1	0.271487
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	0.157753	0.157111	0.5002	0.999479	0.271538
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	0.15763	0.157119	0.500234	0.99974	0.27156
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	0.158202	0.157153	0.500361	0.999219	0.271592
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	0.157181	0.500467	0.999479	0.271643
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	0.15853	0.157205	0.500555	0.999219	0.271669
70	reg_2013-04-30_0.1_l2_0.01	0.157344	0.157103	0.50017	1	0.271545
71	reg_2013-04-30_0.1_l2_1	0.158407	0.157186	0.500483	0.999219	0.27164
72	reg_2013-04-30_0.2_l1_0.01	0.180855	0.159832	0.510197	0.990367	0.275243
73	reg_2013-04-30_0.2_l1_1	0.246974	0.167192	0.534269	0.953137	0.284482
74	reg_2013-04-30_0.2_l2_0.01	0.224444	0.164909	0.527366	0.969018	0.281852
75	reg_2013-04-30_0.2_l2_1	0.247138	0.167192	0.53426	0.952877	0.284471
76	reg_2013-04-30_0.3_l1_0.01	0.634936	0.208214	0.568864	0.472533	0.289059
77	reg_2013-04-30_0.3_l1_1	0.624959	0.210681	0.576292	0.505337	0.29738
78	reg_2013-04-30_0.3_l2_0.01	0.62684	0.211171	0.576455	0.502994	0.29746
79	reg_2013-04-30_0.3_l2_1	0.624346	0.210526	0.576246	0.506118	0.297361
80	reg_2013-04-30_0.5_l1_0.01	0.836564	0.259259	0.505114	0.0218693	0.0403361
81	reg_2013-04-30_0.5_l1_1	0.831166	0.231343	0.506149	0.0322833	0.0566598
82	reg_2013-04-30_0.5_l2_0.01	0.836114	0.263456	0.5058	0.0242124	0.0443491
83	reg_2013-04-30_0.5_l2_1	0.831412	0.229885	0.505871	0.0312419	0.055008

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 [23]: #regressions[21][4]
```

SVM

```

In [24]: 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(df, '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', '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'], temporal=True, start_col='date_posted', end_col='date_fully_funded', start_date=date, end_date=end_date)
            svm.append((model_name, s_test, s_predict, s_score))

```

[illegible]

```
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
tions.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: Converg
enceWarning: Liblinear failed to converge, increase the number of itera
```

tions.

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
```

```
"the number of iterations.", ConvergenceWarning)
```


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

```
In [25]: 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_model(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
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.p
y:1143: UndefinedMetricWarning: Precision is ill-defined and being set
to 0.0 due to no predicted samples.
    'precision', 'predicted', average, warn_for)
/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.p
y:1143: UndefinedMetricWarning: F-score is ill-defined and being set to
0.0 due to no predicted samples.
    'precision', 'predicted', average, warn_for)
```

Out[25]:

	name	accuracy	precision	auc	recall	f1
0	svm_2013-06-30_0.01_1	0.453233	0.207044	0.589446	0.79476	0.328508
1	svm_2013-06-30_0.01_10	0.831717	0	0.5	0	0
2	svm_2013-06-30_0.02_1	0.456408	0.207692	0.590379	0.792314	0.329113
3	svm_2013-06-30_0.02_10	0.831717	0	0.5	0	0
4	svm_2013-06-30_0.05_1	0.465785	0.209339	0.592324	0.783057	0.330361
5	svm_2013-06-30_0.05_10	0.831717	0	0.5	0	0
6	svm_2013-06-30_0.1_1	0.480188	0.210982	0.592762	0.762445	0.330507
7	svm_2013-06-30_0.1_10	0.831717	0	0.5	0	0
8	svm_2013-06-30_0.2_1	0.509994	0.215256	0.594797	0.72262	0.331703
9	svm_2013-06-30_0.2_10	0.831717	0	0.5	0	0
10	svm_2013-06-30_0.3_1	0.538301	0.219575	0.59586	0.68262	0.332271
11	svm_2013-06-30_0.3_10	0.831717	0	0.5	0	0
12	svm_2013-06-30_0.5_1	0.592563	0.224614	0.587378	0.579563	0.323755
13	svm_2013-06-30_0.5_10	0.831717	0	0.5	0	0
14	svm_2013-05-31_0.01_1	0.841304	0	0.5	0	0
15	svm_2013-05-31_0.01_10	0.806486	0.198487	0.508604	0.0722152	0.105901
16	svm_2013-05-31_0.02_1	0.841304	0	0.5	0	0
17	svm_2013-05-31_0.02_10	0.807125	0.19678	0.508038	0.0698856	0.103141
18	svm_2013-05-31_0.05_1	0.841304	0	0.5	0	0
19	svm_2013-05-31_0.05_10	0.809813	0.193991	0.506801	0.0628971	0.0949944
20	svm_2013-05-31_0.1_1	0.841304	0	0.5	0	0
21	svm_2013-05-31_0.1_10	0.812401	0.188406	0.50516	0.0550614	0.085218
22	svm_2013-05-31_0.2_1	0.841304	0	0.5	0	0
23	svm_2013-05-31_0.2_10	0.815728	0.178903	0.503014	0.0448962	0.0717792
24	svm_2013-05-31_0.3_1	0.841304	0	0.5	0	0
25	svm_2013-05-31_0.3_10	0.818552	0.180359	0.502887	0.040449	0.0660785
26	svm_2013-05-31_0.5_1	0.841304	0	0.5	0	0
27	svm_2013-05-31_0.5_10	0.823122	0.182884	0.502597	0.0330368	0.0559641
28	svm_2013-04-30_0.01_1	0.807736	0.185077	0.505915	0.0658683	0.0971582
29	svm_2013-04-30_0.01_10	0.807859	0.185484	0.505988	0.0658683	0.0972142
30	svm_2013-04-30_0.02_1	0.808431	0.186012	0.50601	0.0650872	0.096432
31	svm_2013-04-30_0.02_10	0.808431	0.186012	0.50601	0.0650872	0.096432
32	svm_2013-04-30_0.05_1	0.81019	0.184397	0.505358	0.0609216	0.0915851
33	svm_2013-04-30_0.05_10	0.810231	0.184044	0.505276	0.0606613	0.0912473

	name	accuracy	precision	auc	recall	f1
34	svm_2013-04-30_0.1_1	0.813297	0.180053	0.504023	0.0531112	0.0820265
35	svm_2013-04-30_0.1_10	0.813297	0.180053	0.504023	0.0531112	0.0820265
36	svm_2013-04-30_0.2_1	0.818736	0.180346	0.50333	0.0434783	0.070065
37	svm_2013-04-30_0.2_10	0.818858	0.180932	0.503403	0.0434783	0.0701092
38	svm_2013-04-30_0.3_1	0.822048	0.180225	0.502859	0.0374902	0.062069
39	svm_2013-04-30_0.3_10	0.822089	0.180451	0.502883	0.0374902	0.0620823
40	svm_2013-04-30_0.5_1	0.826873	0.168634	0.50106	0.0260349	0.045106
41	svm_2013-04-30_0.5_10	0.826914	0.168919	0.501084	0.0260349	0.0451162

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 [26]: 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.random_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','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'],temporal=True, start_col='date_posted',end_col='date_fully_funded',start_date=date,end_date=end_date)
            forest.append((model_name, f_test, f_predict, f_score, features))

```

[illegible]

[illegible]

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

```
In [27]: 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_model(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
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.p  
y:1143: UndefinedMetricWarning: Precision is ill-defined and being set  
to 0.0 due to no predicted samples.  
    'precision', 'predicted', average, warn_for)  
/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.p  
y:1143: UndefinedMetricWarning: F-score is ill-defined and being set to  
0.0 due to no predicted samples.  
    'precision', 'predicted', average, warn_for)
```

Out[27]:

	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	0.178777	0.169558	0.504497	0.995459	0.28976
9	rf_2013-06-30_0.2_4	0.198677	0.172406	0.514231	0.989869	0.293665
10	rf_2013-06-30_0.3_3	0.595738	0.225332	0.587545	0.575197	0.323811
11	rf_2013-06-30_0.3_4	0.589006	0.217323	0.575208	0.55441	0.312248
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	0.266913	0.172448	0.545156	0.952774	0.292039
23	rf_2013-05-31_0.2_4	0.256764	0.170649	0.539726	0.954257	0.289524
24	rf_2013-05-31_0.3_3	0.542531	0.206639	0.59143	0.663066	0.315085
25	rf_2013-05-31_0.3_4	0.594051	0.202555	0.568268	0.530496	0.293171
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	0.157058	0.157058	0.5	1	0.271478
37	rf_2013-04-30_0.2_4	0.198274	0.162167	0.518411	0.98516	0.278491
38	rf_2013-04-30_0.3_3	0.632115	0.205304	0.565178	0.467587	0.285328
39	rf_2013-04-30_0.3_4	0.589671	0.206668	0.580888	0.568081	0.303077
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 [28]: #If we wanted to see the feature values (associated with predictors input)
          #forest[7][4]
```

Gradient Boosting

```

In [29]: 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 [30]: 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_model(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[30]:

	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	0.169283	0.168393	0.500392	0.999476	0.288226
7	gradient_2013-06-30_0.1_4	0.170841	0.168637	0.501259	0.999301	0.288575
8	gradient_2013-06-30_0.2_3	0.348031	0.19556	0.577405	0.923144	0.322748
9	gradient_2013-06-30_0.2_4	0.363139	0.198304	0.583284	0.915109	0.325971
10	gradient_2013-06-30_0.3_3	0.548589	0.226115	0.606782	0.694498	0.341156
11	gradient_2013-06-30_0.3_4	0.547707	0.227155	0.609457	0.702533	0.343306
12	gradient_2013-06-30_0.5_3	0.831158	0.409524	0.50266	0.00751092	0.0147513
13	gradient_2013-06-30_0.5_4	0.831246	0.416667	0.502504	0.0069869	0.0137433
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	0.158696	0.158696	0.5	1	0.273922
19	gradient_2013-05-31_0.05_4	0.15873	0.158701	0.50002	1	0.27393
20	gradient_2013-05-31_0.1_3	0.16125	0.15901	0.501174	0.999153	0.274358
21	gradient_2013-05-31_0.1_4	0.166157	0.159635	0.503489	0.99767	0.275232
22	gradient_2013-05-31_0.2_3	0.334767	0.182588	0.571393	0.918043	0.304595
23	gradient_2013-05-31_0.2_4	0.359637	0.186472	0.579902	0.902584	0.309087
24	gradient_2013-05-31_0.3_3	0.560242	0.210683	0.594568	0.644854	0.317601
25	gradient_2013-05-31_0.3_4	0.550496	0.210544	0.597539	0.666455	0.319996
26	gradient_2013-05-31_0.5_3	0.840598	0.40708	0.503533	0.00974163	0.0190279
27	gradient_2013-05-31_0.5_4	0.840733	0.428571	0.504042	0.0108005	0.02107
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	0.157058	0.157058	0.5	1	0.271478
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	0.161351	0.157707	0.502441	0.99974	0.272437
35	gradient_2013-04-30_0.1_4	0.16589	0.15826	0.504497	0.998178	0.273203
36	gradient_2013-04-30_0.2_3	0.367558	0.185307	0.580586	0.891174	0.306817
37	gradient_2013-04-30_0.2_4	0.372628	0.186731	0.584123	0.892476	0.308843
38	gradient_2013-04-30_0.3_3	0.57446	0.211714	0.596121	0.627701	0.316633
39	gradient_2013-04-30_0.3_4	0.571148	0.213812	0.601782	0.646446	0.321341
40	gradient_2013-04-30_0.5_3	0.841716	0.37069	0.503827	0.011195	0.0217336
41	gradient_2013-04-30_0.5_4	0.841061	0.346667	0.504392	0.0135381	0.0260586

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 [31]: 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 [32]: dummy_metrics = pd.DataFrame(columns=['name', 'accuracy', 'precision', 'auc', 'recall', 'f1'])

for i in range(len(dummy)):
    name, test, predict, _ = dummy[i]
    accuracy, precision, auc, recall, f1 = pipeline_revised.evaluate_model(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[32]:

	name	accuracy	precision	auc	recall	f1
0	dummy_2013-06-30_0.01	0.635567	0.166183	0.497796	0.290131	0.211323
1	dummy_2013-06-30_0.02	0.64221	0.169554	0.501302	0.288908	0.213695
2	dummy_2013-06-30_0.05	0.639124	0.167479	0.499168	0.28821	0.211851
3	dummy_2013-06-30_0.1	0.636684	0.165676	0.497283	0.287162	0.210123
4	dummy_2013-06-30_0.2	0.636772	0.165456	0.497057	0.286463	0.209759
5	dummy_2013-06-30_0.3	0.643563	0.172985	0.504831	0.295721	0.218283
6	dummy_2013-06-30_0.5	0.639124	0.167479	0.499168	0.28821	0.211851
7	dummy_2013-05-31_0.01	0.652361	0.165437	0.507128	0.294367	0.211826
8	dummy_2013-05-31_0.02	0.644766	0.1547	0.495741	0.277425	0.198635
9	dummy_2013-05-31_0.05	0.649235	0.162274	0.50381	0.290767	0.208299
10	dummy_2013-05-31_0.1	0.6449	0.160785	0.502265	0.293308	0.207708
11	dummy_2013-05-31_0.2	0.64648	0.158719	0.500025	0.285472	0.204011
12	dummy_2013-05-31_0.3	0.647387	0.16202	0.503571	0.292884	0.208629
13	dummy_2013-05-31_0.5	0.645673	0.153718	0.494734	0.273613	0.196846
14	dummy_2013-04-30_0.01	0.648348	0.155594	0.498439	0.279875	0.2
15	dummy_2013-04-30_0.02	0.648593	0.15712	0.500067	0.28352	0.202191
16	dummy_2013-04-30_0.05	0.64614	0.154883	0.497658	0.281177	0.199741
17	dummy_2013-04-30_0.1	0.644464	0.156913	0.499842	0.288987	0.20339
18	dummy_2013-04-30_0.2	0.649207	0.157659	0.500643	0.284041	0.202769
19	dummy_2013-04-30_0.3	0.648757	0.158394	0.501435	0.286644	0.20404
20	dummy_2013-04-30_0.5	0.642501	0.150057	0.492428	0.273627	0.193822

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.