CSCI: 83 2017: Predicting a Banking Customer

Course: Fundamentals of Data Science – Harvard University – Division of Continuing Education (DCE).

Presented by Ayindri Banerjee – Founder and CEO Neon Inc.

Date: 12th May 2017.

Executive Summary

ABC Portugal is a Portuguese banking institution which offers various financial products such as home and auto loans, long term deposits and so on. In 2017, they are projecting to increase their long term deposit subscribers by 25%. To support this initiative, the company has engaged a specialized Data Science Consulting firm – Neon Inc, to help the bank better target future customers.

The bank has provided some historical dataset to the company which has demographic and existing product information about their customers, along with some campaign history. The goal is to find a way to identify potential long term deposit subscribers based on this data.

ABC Portugal will use this information to define their marketing and advertising strategy and budget for Q3.

Neon Inc, produced a predictive web service, which ABC Portugal will integrate into their digital channels as well their call centers, to know "real-time" whether the customer they are targeting will be a long term deposit subscriber.

Neon, found that the data was highly imbalanced with only 10% of the data who are long term deposit customers. They compared the predictive performance of various models, and out of all the Two-Class Boosted Decision Tree performed the best. They also used a technique known as SMOTE to improve the performance of the model, in order to manage the data imbalance. Following are the key results:

Model	AUC	Accuracy	Recall	Precision
Two Class Logistic Regression	0.781	0.902	0.226	0.674
Two Class Boosted Decision Tree	0.791	0.897	0.350	0.555
Two Class Decision Forest	0.743	0.892	0.276	0.522
Two Class Boosted Decision Tree - with SMOTE (Best Model)	**0.953**	**0.908**	**0.859**	**0.899**

When ABC Portugal tested this "predictive web service" they found that it was 88% accurate with identifying their potential customers, on out-of-sample data.

After such a dramatic success of the pilot project, ABC Portugal is awarding Neon Inc. with a $100 million deal over the next 5 years to help ABC shape the future with innovative ways of using data science to improve all aspects of their business.

Background

ABC Portugal is a Portuguese banking institution. The data is related to direct marketing calls the bank makes to potential or existing customers to upsell various banking products. These marketing campaigns were based on 'phone calls'. Often more than one contact to a client was required.

Our goal is to predict which of the customers of ABC Portugal will subscribe to a 'term deposit'.

What are term deposits?

A term deposit is a deposit held at a financial institution that has a fixed term. These are generally short-term with maturities ranging anywhere from a month to a few years. When a term deposit is purchased, the lender (the customer) understands that the money can only be withdrawn after the term has ended or by giving a predetermined number of days' notice. These types of financial products are sold by banks, thrift institutions and credit unions.

(**Source:** http://www.investopedia.com/terms/t/termdeposit.asp)

(Data Source: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing)

About the Data:

The dataset has 41,188 records with 20 columns with data divided into 3 categories.

1. Bank client data
2. Marketing Campaign Data
3. Contextual – economic data

'y' is the label which identifies if a customer subscribed to a 'term deposit' or not.

Here we are also splitting this dataset into training and test set, before we start any work on it. We are also exporting them to 'training.csv' and 'test.csv'.

Below is the code:

```
In [3]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split

        df= pd.read_csv('.//bank-additional//bank-additional-full.csv', sep=';')
        df.columns

(Index([u'age', u'job', u'marital', u'education', u'default', u'housing',
       u'loan', u'contact', u'month', u'day_of_week', u'duration', u'campaign',
       u'pdays', u'previous', u'poutcome', u'emp.var.rate', u'cons.price.idx',
       u'cons.conf.idx', u'euribor3m', u'nr.employed', u'y'],
      dtype='object'), (41188, 21))

In [8]: df.shape

Out[8]: (41188, 21)

In [9]: #split training and test set and export to csv

        train, test = train_test_split(df, test_size = 0.2)
        test.to_csv('test.csv')
        train.to_csv('train.csv')
        len(test), len(train)

Out[9]: (8238, 32950)
```

As you see above, we are splitting the training and test set into 80:20 split, and now the training data has 32,950 records and the test data as 8,238 records.

Next step is to explore the training dataset to uncover any relationships in the data, which

Data Exploration & Cleaning

Let's explore the training dataset now. We will perform some EDA (Exploratory Data Analysis) with visualizations and perform feature engineering based of the inferences from the visualizations.

The below tells us which features are numeric and which are categorical. Also it gives us an idea if there are any 'null' values which we need to treat.

- Looks like there are no null values, but there are some fields with value 'unknown' in the categorical variables.
- We can also find some '999' s in the 'pdays' numeric feature.

```
In [10]: df = train
         df.head()
```

Out[10]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome
22534	32	technician	married	university.degree	no	yes	no	cellular	aug	fri	...	2	999	0	nonexistent
41024	37	unemployed	single	professional.course	no	no	no	cellular	oct	fri	...	4	6	1	success
38347	50	technician	married	professional.course	no	unknown	unknown	cellular	oct	fri	...	1	8	2	failure
9105	54	technician	married	basic.9y	no	no	no	telephone	jun	fri	...	9	999	0	nonexistent
26519	45	services	divorced	high.school	no	no	no	cellular	nov	thu	...	2	999	1	failure

5 rows × 21 columns

```
In [11]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32950 entries, 22534 to 24393
Data columns (total 21 columns):
age             32950 non-null int64
job             32950 non-null object
marital         32950 non-null object
education       32950 non-null object
default         32950 non-null object
housing         32950 non-null object
loan            32950 non-null object
contact         32950 non-null object
month           32950 non-null object
day_of_week     32950 non-null object
duration        32950 non-null int64
campaign        32950 non-null int64
pdays           32950 non-null int64
previous        32950 non-null int64
poutcome        32950 non-null object
emp.var.rate    32950 non-null float64
cons.price.idx  32950 non-null float64
cons.conf.idx   32950 non-null float64
euribor3m       32950 non-null float64
nr.employed     32950 non-null float64
y               32950 non-null object
dtypes: float64(5), int64(5), object(11)
memory usage: 5.5+ MB
```

Categorical Features

Let us try to understand which features have ‘unknown’ as a categorical variables and what percentage of them have ‘unknown’ values.

Below code tells us which categorical features have ‘unknown’ values and what percent of the records have this value. Looks like feature ‘default’ as the highest no of ‘unknown’ records about 20%, while the other columns are really low.

As a starting strategy we will make a copy of this dataset, and drop these records.

We can come back to these records later and various methods to treat them to create another version of our model to see if the performance is any better.

Below code is where we are reviewing this data and determining % of records with ‘unknown’.

```
In [20]: cat_cols = df.select_dtypes(exclude=['int64','float64'])
         for col in cat_cols:
             count_unk = len(df[df[col] == 'unknown'])
             if count_unk > 0:
                 print(col, count_unk, len(df), round((float(count_unk)/len(df)) * 100,1))
                 print(df[col].unique())

         ('job', 264, 32950, 0.8)
         ['technician' 'unemployed' 'services' 'blue-collar' 'admin.' 'housemaid'
          'entrepreneur' 'management' 'retired' 'student' 'self-employed' 'unknown']
         ('marital', 67, 32950, 0.2)
         ['married' 'single' 'divorced' 'unknown']
         ('education', 1383, 32950, 4.2)
         ['university.degree' 'professional.course' 'basic.9y' 'high.school'
          'basic.6y' 'unknown' 'basic.4y' 'illiterate']
         ('default', 6887, 32950, 20.9)
         ['no' 'unknown' 'yes']
         ('housing', 793, 32950, 2.4)
         ['yes' 'no' 'unknown']
         ('loan', 793, 32950, 2.4)
         ['no' 'unknown' 'yes']
```

Here we create a copy of the dataset where we drop records with ‘unknown’

```
In [33]: # lets drop all records with 'unknown' and see if there are enough records
         df1 = df.copy()
         for m in cat_cols:
             c = df1[df[m].str.contains('unknown')]
             print(m,len(c), len(df1))
             df1 = df1[~df[m].str.contains('unknown')]
             print('after: ', len(df1))
         len(df), len(df1), round(100*float(len(df1))/len(df),1),'%'

         ('job', 255, 32950)
         ('after: ', 32695)
         ('marital', 57, 32695)
         ('after: ', 32638)
         ('education', 1265, 32638)

         C:\ProgramData\Anaconda3\envs\python2\lib\site-packages\ipykernel_launcher.py:4: UserWarning: Boolean Series key will be reinde
         xed to match DataFrame index.
           after removing the cwd from sys.path.
         C:\ProgramData\Anaconda3\envs\python2\lib\site-packages\ipykernel_launcher.py:6: UserWarning: Boolean Series key will be reinde
         xed to match DataFrame index.

         ('after: ', 31373)
         ('default', 6424, 31373)
         ('after: ', 24949)
         ('housing', 595, 24949)
         ('after: ', 24354)
         ('loan', 0, 24354)
         ('after: ', 24354)
         ('contact', 0, 24354)
         ('after: ', 24354)
         ('month', 0, 24354)
         ('after: ', 24354)
         ('day_of_week', 0, 24354)
         ('after: ', 24354)
         ('poutcome', 0, 24354)
         ('after: ', 24354)
         ('y', 0, 24354)
         ('after: ', 24354)

Out[33]: (32950, 24354, 73.9, '%')
```

This led to 27% reduction in the training dataset. This is significant, and in real data this kind of data will be present. Based on this finding, and to make the model more generic, we want to build the ‘unknown’ scenario into the data.

Hence we will leave the data as is, and move forward.

Numeric Features

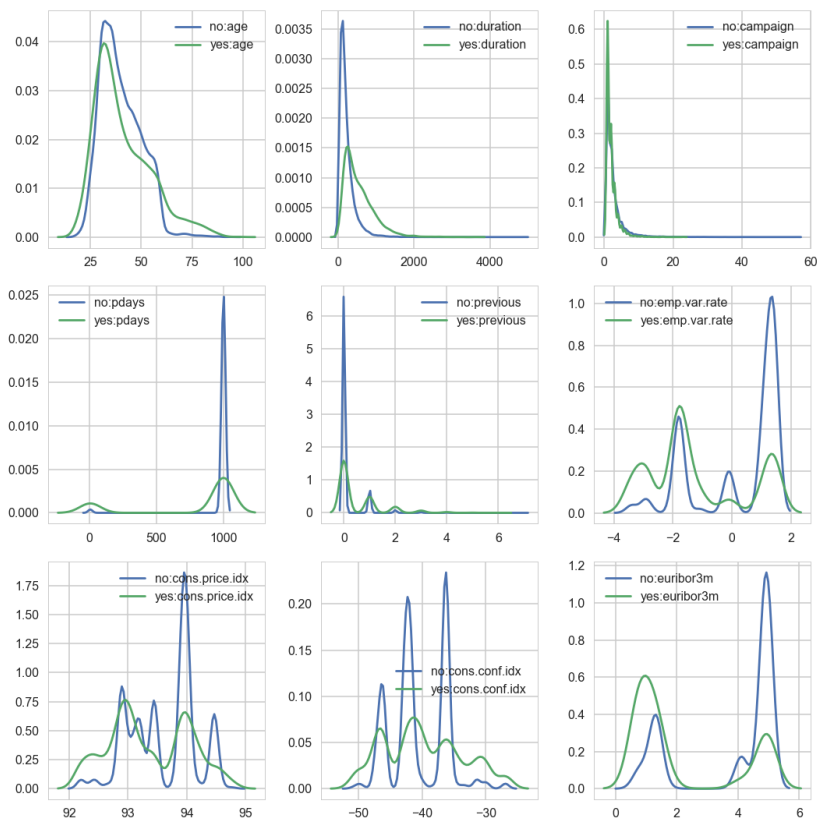
Now let us examine at the numeric features. The below information is there is a lot of variance 'duration','pdays'. If we still decide to use these features we have to make sure that we are eliminating outliers, or may be convert them to categorical features.

```
In [34]: df.describe()
```

Out[34]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	32950.000000	32950.000000	32950.000000	32950.000000	32950.000000	32950.000000	32950.000000	32950.000000	32950.000000	32950.000000
mean	39.978634	257.299029	2.565341	961.660577	0.175205	0.084398	93.577439	-40.489563	3.624134	5167.013806
std	10.424472	254.357894	2.775082	188.906443	0.499827	1.570494	0.578272	4.624910	1.734000	72.361978
min	17.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.000000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.000000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.000000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.000000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

In the following KDE plots we're looking for variables with very little overlap between the positive and negative results. Some examples are cons.price.idx and age.



Call duration represented by the feature 'duration' is a very strong predictor of whether a customer will subscribe for a long term deposit or not. Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model. So we will drop this feature.

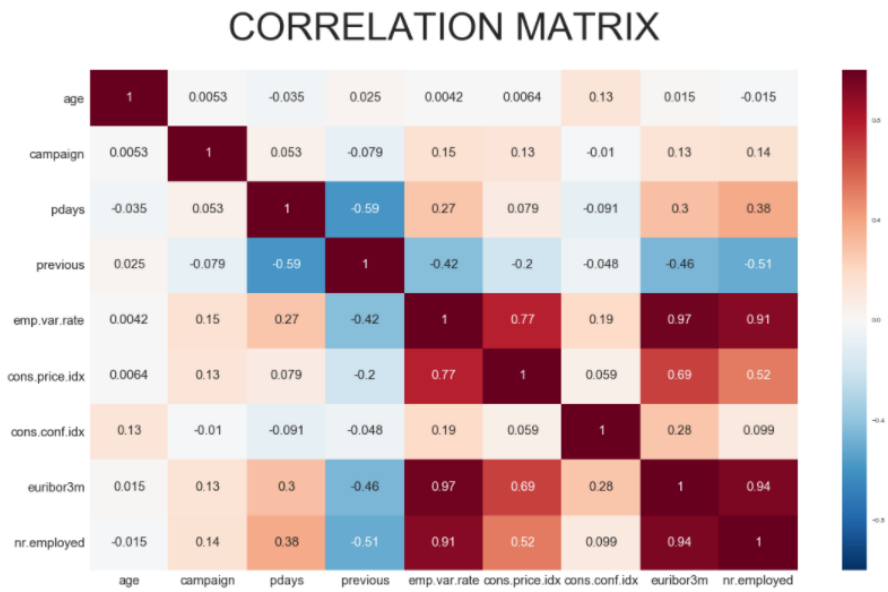
```
In [68]: del df['duration']
```

Correlation – eliminating collinear features:

Next step is to identify and eliminate collinear features. Below is the collinearity matrix for all features.

```
In [76]: int_cols = df.select_dtypes(include=['int64','float64']).columns
         sns.set_context("paper", rc={"font.size":15,"axes.titlesize":15,"axes.labelsize":20,"font.scale":1.5})
         plt.figure(figsize=(15, 10))
         corr = df[int_cols].corr()
         b = sns.heatmap(corr, annot=True,
                     xticklabels=corr.columns.values,
                     yticklabels=corr.columns.values)
         b.figure.savefig("allCorr.png")
         b.axes.set_title("CORRELATION MATRIX",fontsize=45,y=1.05)
         #b.set_xlabel("X Label",fontsize=20,labelpad=10)
         #b.set_ylabel("Y Label",fontsize=20,labelpad=10)
         b.tick_params(labelsize=15)
         plt.tight_layout(pad=0.7, w_pad=0.5, h_pad=1.0)
         plt.yticks(rotation=1)
         plt.yticks(rotation=0)
         #sns.plt.show()

Out[76]: (array([ 0.5,  1.5,  2.5,  3.5,  4.5,  5.5,  6.5,  7.5,  8.5]),
          <a list of 9 Text yticklabel objects>)
```

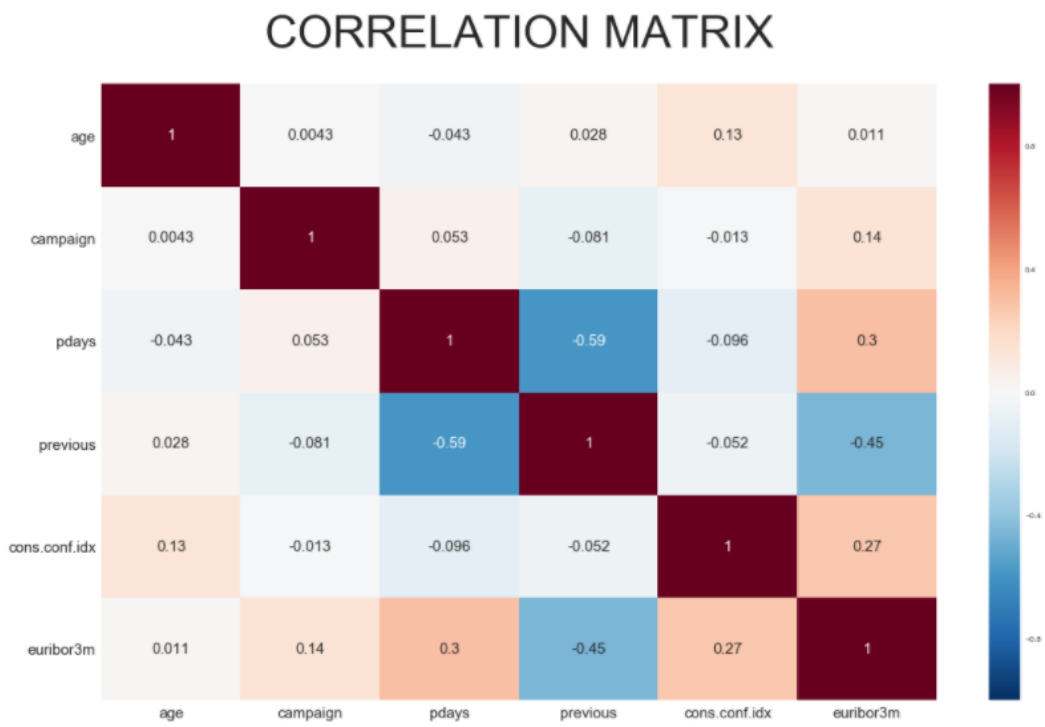


From above, we can see that nr.employed, conf.price.idx, emp.var.rate and euribor3m are highly correlated. So we will use any of the 4 features. We will go with euribor3m and drop the remaining 3 features.

```
In [109]: del df['nr.employed']
          del df['cons.price.idx']
          del df['emp.var.rate']
          df.columns

Out[109]: Index([u'age', u'job', u'marital', u'education', u'default', u'housing',
                 u'loan', u'contact', u'month', u'day_of_week', u'campaign', u'pdays',
                 u'previous', u'poutcome', u'cons.conf.idx', u'euribor3m', u'y'],
                dtype='object')
```

The new correlation matrix, below is cleaner, with no strong correlations.



<u>'pdays' and the value '999'</u>

Let us dive deeper into the numeric feature 'pdays'. 'pdays' is the number of days that passed by after the client was last contacted from a previous campaign. The numeric value '**999'** means client was not previously contacted). From below, we see that 96% of the customers were not contacted before.

```
In [113]: 100*float(len(df[df['pdays'] == 999]))/len(df)
Out[113]: 96.30652503793627
```

However, those who were contacted, leaves a very strong impression on the outcome. So we will convert this into categorical labels such as "0-5", "5-10", "10-20", "20-30", "30-40", "Not Contacted", and "Other". Also we will drop ‘pdays’ , since it will be represented by ‘pdays_class’.

In [117]:
```
df['pdays_class'] = pd.cut(np.array(df['pdays']), bins=[0,5,10,20,30,40,999,1000],\
                           labels=["0-5","5-10","10-20","20-30","30-40","Not Contacted","Other"])
df[['pdays_class','pdays']].head(), len(df[df['pdays_class'] == "Not Contacted"]), \
df.head(2)
```

Out[117]:
```
(         pdays_class  pdays
 28652  Not Contacted    999
 3325   Not Contacted    999
 1292   Not Contacted    999
 21159  Not Contacted    999
 2483   Not Contacted    999,
 31733,
        age           job   marital    education default housing loan  \
 28652   54  entrepreneur  divorced      unknown      no      no  yes
 3325    53       retired  divorced  high.school      no      no  yes

          contact month day_of_week  campaign  pdays  previous     poutcome  \
 28652   cellular   apr         thu         2    999         1      failure
 3325   telephone   may         thu         5    999         0  nonexistent

        cons.conf.idx  euribor3m   y    pdays_class
 28652          -47.1       1.41  no  Not Contacted
 3325           -36.4       4.86  no  Not Contacted  )
```

In [115]:
```
print(len(df))
pd.crosstab(df.pdays_class, df.y, margins=True)
```

```
32950
```

Out[115]:

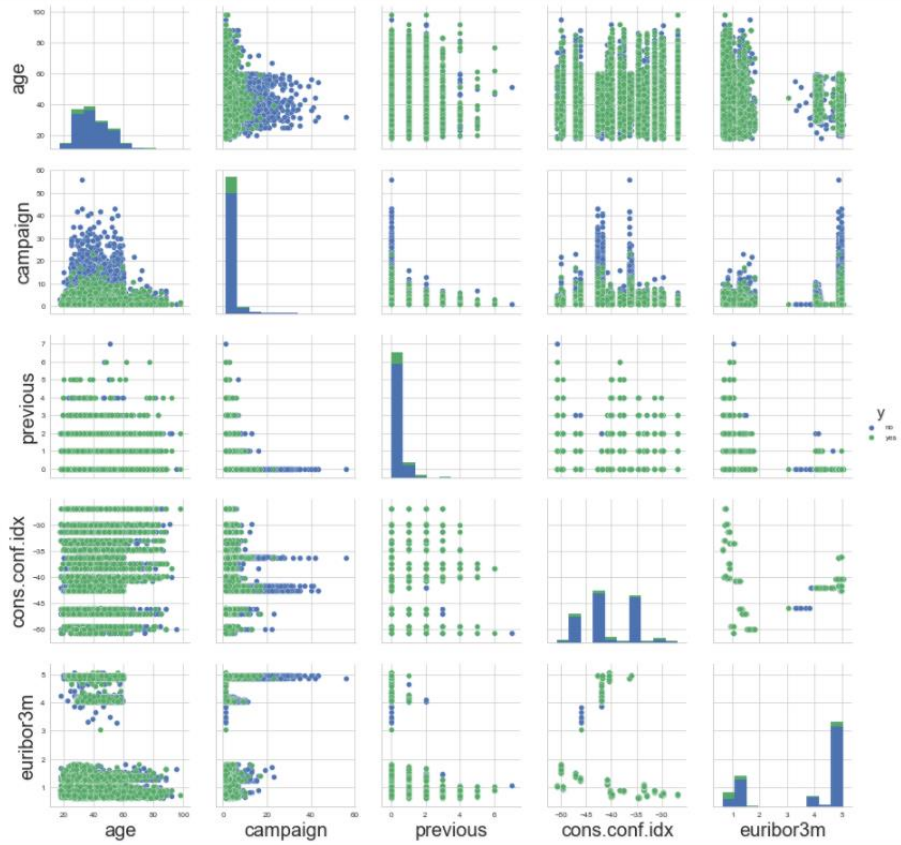
y	no	yes	All
pdays_class			
0-5	203	349	552
5-10	164	327	491
10-20	67	86	153
20-30	1	7	8
30-40	0	0	0
Not Contacted	28808	2925	31733
Other	0	0	0
All	29243	3694	32937

In [118]:
```
del df['pdays']
```

We will use pairplots below to explore any other relationships.

We find below that across all age groups, if customers are contacted too many times during a campaign, they are less likely to subscribe for a 'long term contract'.

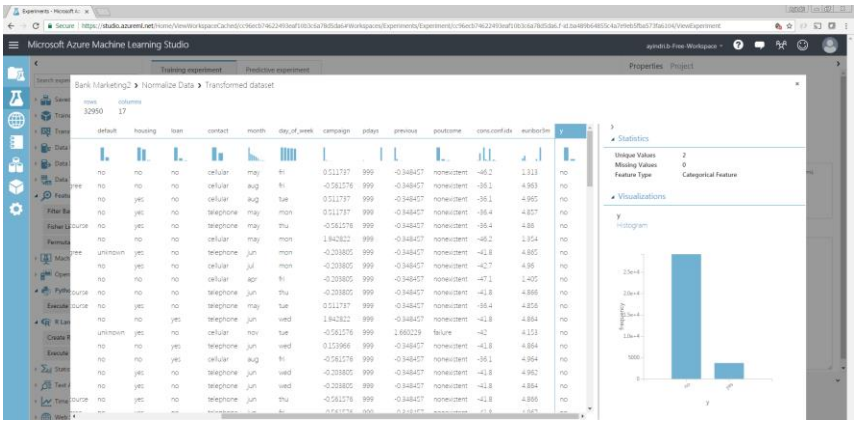
```
In [120]: z = sns.pairplot(df,hue='y')
```



After this we will move to building the model in Azure ML Studio, and using some the modules in it to tune the features further.

Predictive Models

Data is imbalanced with a 90:10 proportion of negative and positive 'y' values. So we have to treat the data imbalance, to optimize our model.



Change Categorical Data

We will use Azure ML to 'Edit Metadata' to convert the string features to categorical features.

Normalize Numeric Feature

We will normalize the numeric features using 'ZScore' method.

Which features are more important than others?

Per the below screenshot, looks like the feature 'eurobor3m' is the strongest predictor of whether a customer will subscriber to a long term deposit. The other important features are 'campaign' – of no of times a customer is contacted during a campaign or not, followed by 'age','month' of campaign and so on.

Bank Marketing2 › Permutation Feature Importance › Feature importance

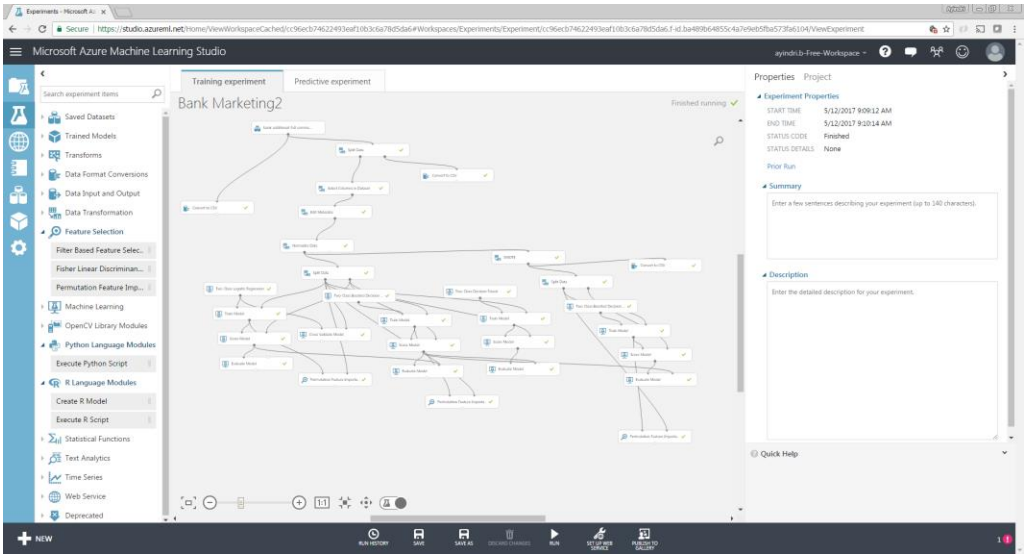
rows 16 columns 2

view as

Feature	Score
euribor3m	0.132927
campaign	0.1027
age	0.09376
month	0.02936
cons.conf.idx	0.006966
day_of_week	0.00592
pdays	0.005172
poutcome	0.005113
job	0.004634
contact	0.004126
education	0.003229
marital	0.001764
loan	0.001585
housing	0.001196
default	0.000957
previous	0.000807

<u>Baseline Model</u>

Based on above analysis, we are finally ready to build our baseline model. We will start with building a 'Two Class Logistic Classification' model.



Baseline Model

We see below the results from the baseline model, in our case the 2 class logistic regression classifier. The AUC of .78 and Accuracy of 0.9, look ok in the beginning, but when we see Recall, it's pretty low and Precision can also be improved further.

Clearly there is room to improve, and some of it could be attributed to the imbalance in the dataset.

We use the 'Cross validate' module to understand variance of the AUC and Accuracy across various folds. The below numbers tell us that across the folds, the AUC, Accuracy, Recall and Precision are pretty consistent.

Bank Marketing2 › Cross Validate Model › Evaluation results by fold

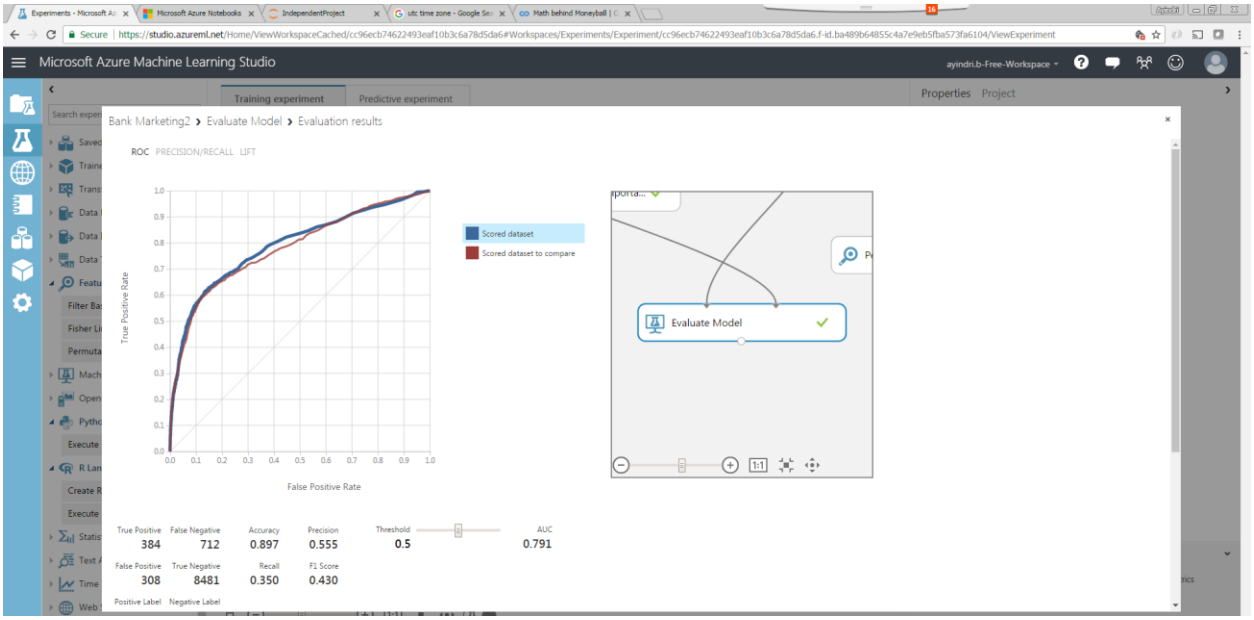
rows 12 columns 10

view as

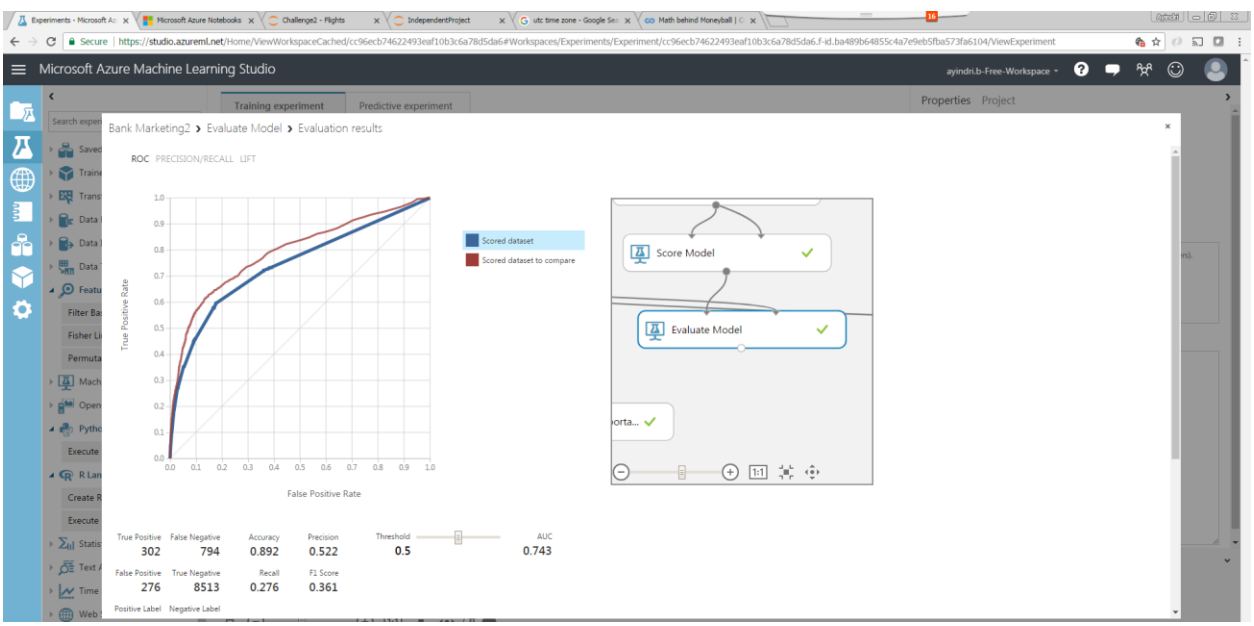
Fold Number	Number of examples in fold	Model	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss	Training Log Loss
0	2306	Logistic Regression	0.901561	0.655914	0.238281	0.34957	0.786451	0.274908	21.146683
1	2306	Logistic Regression	0.899393	0.704082	0.253676	0.372973	0.789751	0.282786	22.060342
2	2306	Logistic Regression	0.882914	0.655914	0.204013	0.311224	0.795728	0.305535	20.793216
3	2306	Logistic Regression	0.905898	0.727273	0.222222	0.340426	0.798405	0.272618	20.982012
4	2307	Logistic Regression	0.898136	0.62766	0.227799	0.334278	0.806544	0.271299	22.757243
5	2307	Logistic Regression	0.899436	0.642105	0.235521	0.344633	0.798193	0.273214	22.212088
6	2306	Logistic Regression	0.902862	0.685714	0.192	0.3	0.795476	0.27251	20.593577
7	2307	Logistic Regression	0.901604	0.651685	0.228346	0.338192	0.780775	0.279518	19.383086
8	2307	Logistic Regression	0.895102	0.54878	0.18	0.271084	0.788104	0.28106	18.078671
9	2307	Logistic Regression	0.898136	0.666667	0.199234	0.306785	0.785032	0.28236	20.015495
Mean	23065	Logistic Regression	0.898504	0.656579	0.218109	0.326917	0.792446	0.279581	20.802241
Standard Deviation	23065	Logistic Regression	0.006229	0.048185	0.023285	0.029449	0.007749	0.010128	1.399917

Model Selection

In the below screenshot we see the comparison of the Two-Class Boosted Decision Tree in blue, with the base line model in red. We see the AUC improved a little, Accuracy deteriorated but Recall improved.

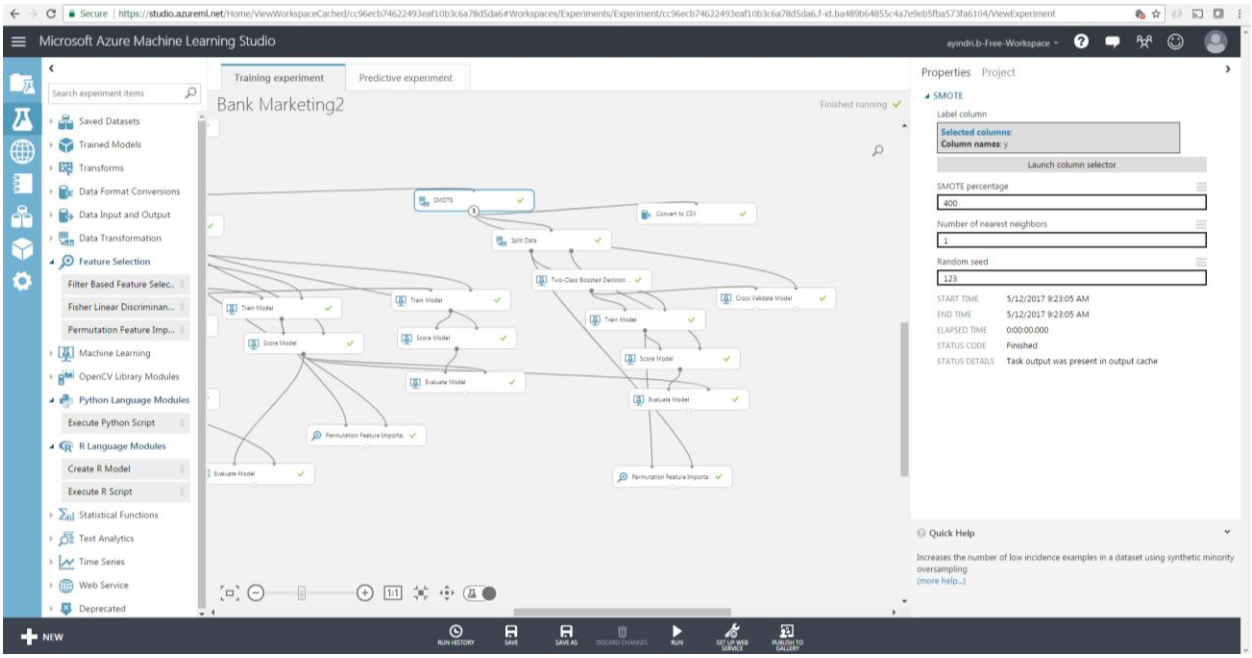


Let's compare with another model, Two-Class Decision Forest and see if find any improvements. As we see below, the Two-Class Decision Forest is performing worse than the Two-Class Boosted Decision Tree model.

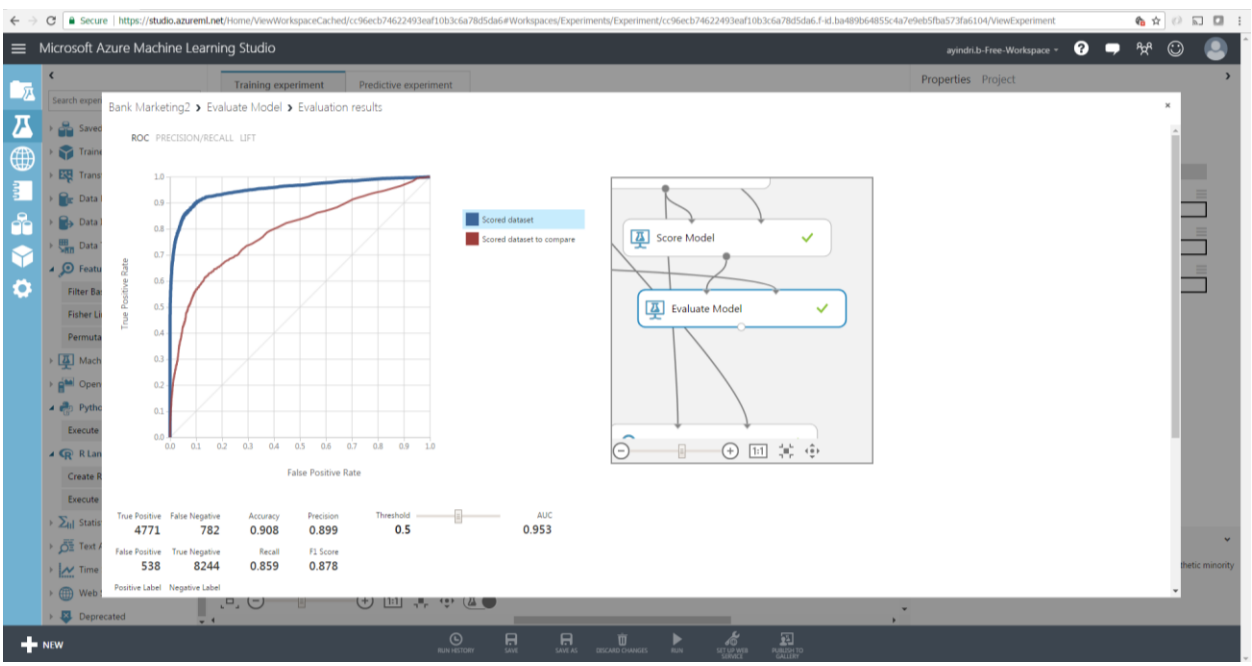


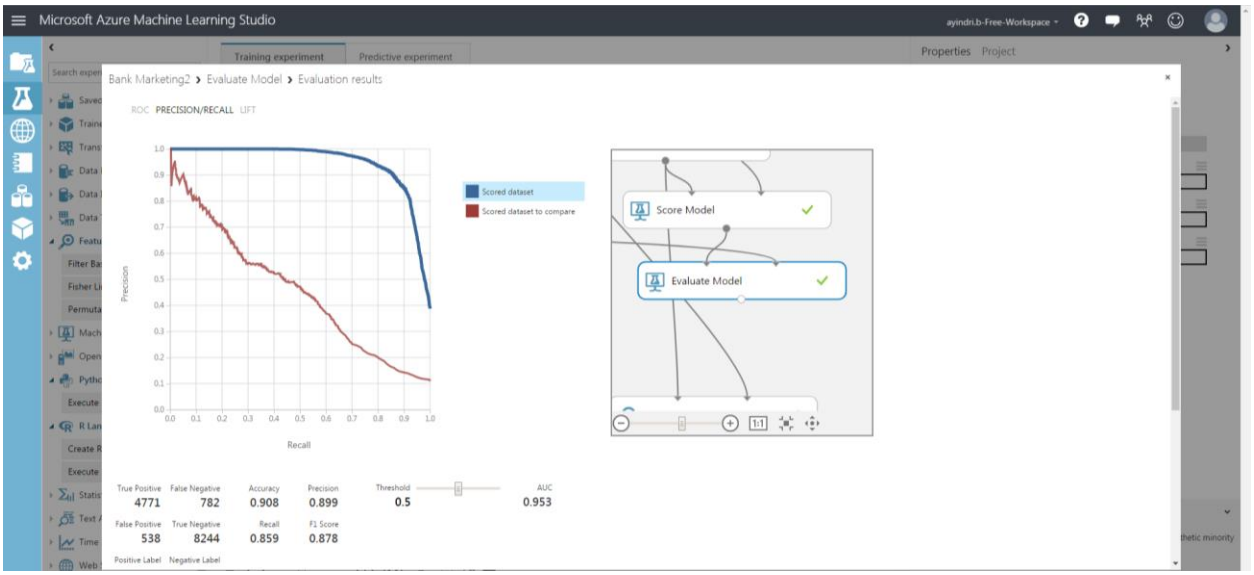
Best Model is Two-Class Boosted Decision Tree.

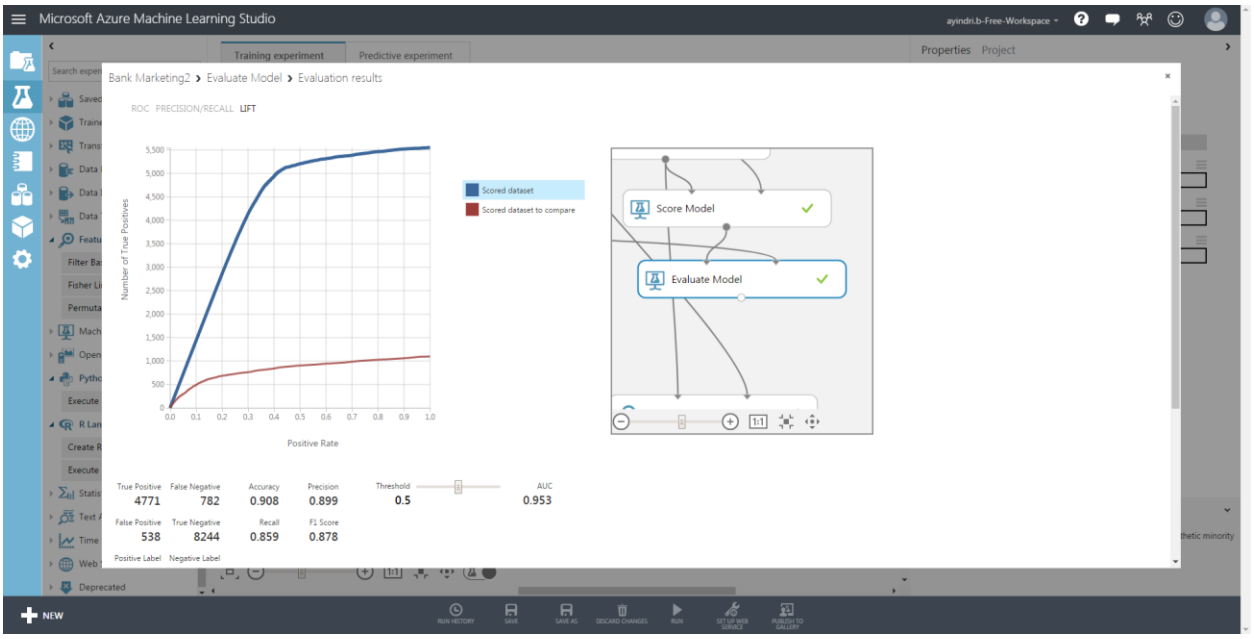
However, there is one important aspect we had discussed in the beginning, about data imbalance. Let's now try to apply SMOTE (Synthetic Minority Over-Sampling Technique) to the best model and see if that improves the performance. When we do this the number of minority records increase, this may eventually improve results. Below we make the SMOTE % to 400, this increases the ratio from 90:10 to 60:40.



Let's now compare the results with the best model so far. We see that with SMOTE, the improvement to our model is pretty dramatic. An AUC of 0.95, Accuracy 0.9 and Recall – 0.85 are massive improvements.





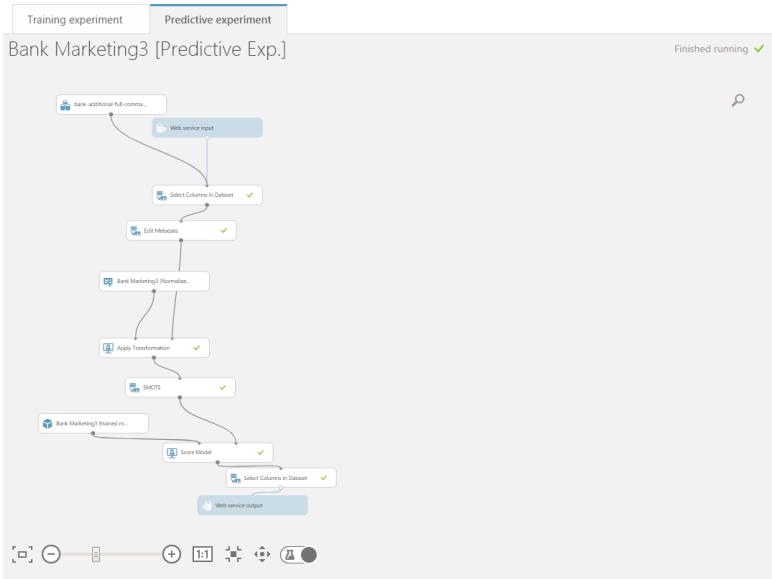


Model	AUC	Accuracy	Recall	Precision
Two Class Logistic Regression	0.781	0.902	0.226	0.674
Two Class Boosted Decision Tree	0.791	0.897	0.350	0.555
Two Class Decision Forest	0.743	0.892	0.276	0.522
Two Class Boosted Decision Tree -	**0.953**	**0.908**	**0.859**	**0.899**

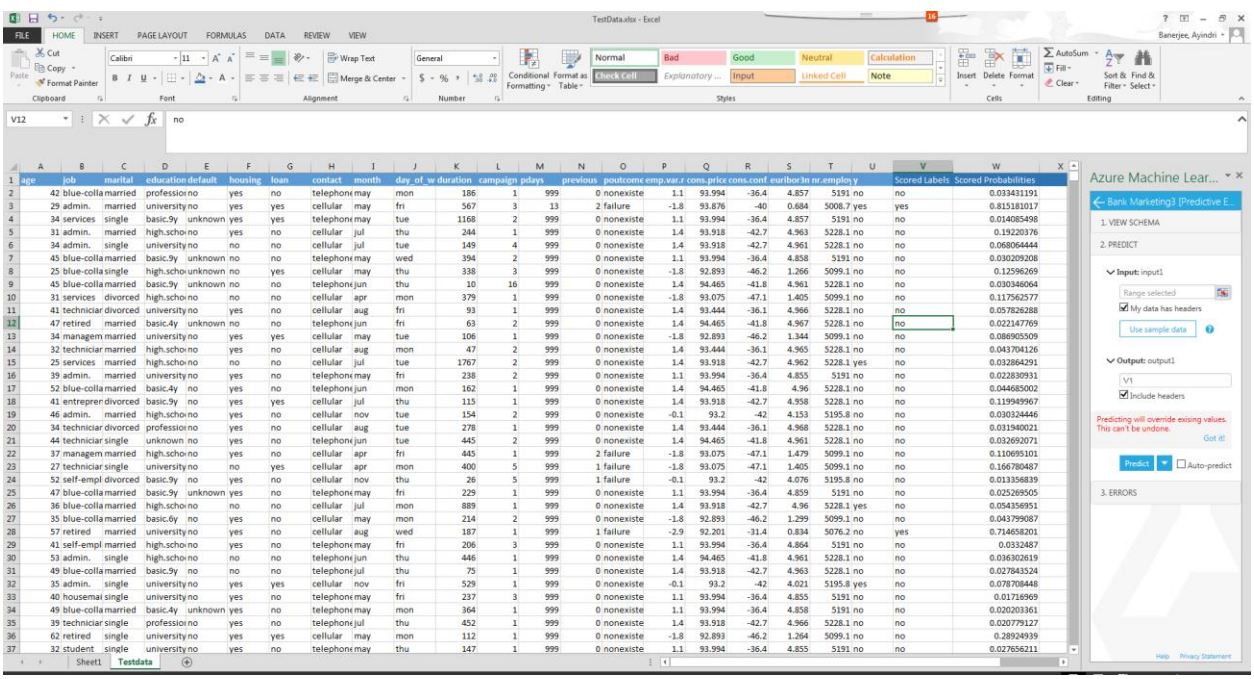
with SMOTE
(Best Model)

Operationalizing the Model

Now that we are satisfied with the model, we will operationalize the model by creating a “Predictive Web Service”.



After creating the web service and running the test data from excel, we found 88% accuracy with out-of-sample data.



Conclusion & Action Items

We found that for this data set Two-Class Boosted Decision Tree with SMOTE technique is the best performing model.

Appendix

Data source: Bank Marketing Total Instances: 45,211

Abstract:

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Source:

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

Data Set Information:

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

There are four datasets:

- 1) bank-additional-full.csv with all examples (41188) and 20 inputs, ordered by date (from May 2008 to November 2010), very close to the data analyzed in [Moro et al., 2014]
- 2) bank-additional.csv with 10% of the examples (4119), randomly selected from 1), and 20 inputs.
- 3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this dataset with less inputs).
- 4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this dataset with less inputs).

The smallest datasets are provided to test more computationally demanding machine learning algorithms (e.g., SVM).

The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

Attribute Information:

Input variables:

bank client data:

1 - age **(numeric)**
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')

related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds **(numeric).** Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

other attributes:

12 - campaign: number of contacts performed during this campaign and for this client **(numeric, includes last contact)**
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign **(numeric; 999 means client was not previously contacted)**
14 - previous: number of contacts performed before this campaign and for this client **(numeric)**
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

social and economic context attributes

16 - emp.var.rate: employment variation rate - quarterly indicator **(numeric)**
17 - cons.price.idx: consumer price index - monthly indicator **(numeric)**
18 - cons.conf.idx: consumer confidence index - monthly indicator **(numeric)**
19 - euribor3m: euribor 3 month rate - daily indicator **(numeric)**
20 - nr.employed: number of employees - quarterly indicator **(numeric)**

Output variable (desired target):

21 - y - has the client subscribed a term deposit? (binary: 'yes','no')