Justin Curry

12/4/2019

Final Project Write-up

1. Introduction

My final project is a classification problem. I have a dataset of credit card loan defaults and my job is to try and create a model that can accurately predict which people are going to default on loans without creating too much loss of specificity. So, this project is one that is based in some statistical methods along with different metrics than simply accuracy. It is more harmful to give a loan a person can’t pay back than to refuse to give a loan someone can. The problem becomes not simply how to create an accurate model but how to create a very precise model. On top of that, I am trying to create a non-gender biased model. So, the model I use in the end is one that tries to minimize the models ability to predict gender. If the model is gender blind then it cannot classify default differently between men and women.

1. Background

This type of assignment is important because of businesses such as banks and investment firms. Banks need to decide who to trust with there money daily. If a bank knows what people are going to default (before they actually do) then preventative measures can be taken to minimize amount of money lost. There is little reason to keeping someone going on a credit line once you know that they are not going to pay it off. The dataset that I am using has 30,000 observations with 23 prediction variables and a binary response. The variables are things such as past payment delay, bill amounts, pay amounts, education, and marital status. Background knowledge needed for the analysis includes some basic statistical testing knowledge, dealing with imbalance in data sets, and different classification models.

1. Material & Methods

The materials I used for the analysis is python and JMP. I did all models in Python but I did all of the pre-analysis and statistical testing in JMP. I started by looking at the distributions of variables. The predictors or mostly skewed to the right because of people who have a large amount of payments to make. I then looked at a correlation matrix for the numeric variables and noticed that bill amounts are heavily correlated. However, oddly enough, the payment variables are not correlated with anything by all that much. I then used t-tests to try and learn if the response variable is related to any of the predictors by looking at the means. A Bill amounts up to 4 months prior were significant and all pay amounts are significant. However, due to sample size significance might be hard to interpret (sample so large almost everything significant) so I used the group means to compare by hand instead of simply looking at p-value. An example of this is Age who was significant but had group means of 34.7 and 34.4 (in years). So I decided this is not an important distinction.

The next thing I did was look to see what variables contained information on gender. Since I am trying to create a gender-neutral model, I should exclude any variables that contain a heavy amount of information for gender. As it turns out, the variable that seemed to contain gender was Age. So, from here I removed Age from the analysis any further since it showed no sign of being important to the predictor and being gender biased. I then began to look at clusters of the data. I ran K means clustering on the numeric variables to try and spot what variables are important. I ran clusters for K = 5 up to K=50. JMP gave the optimal number of clusters to be 44. However, after a quick glance I did not see any clusters that contained mostly 0’s or mostly 1’s to do any good comparisons.

At this point I ran univariate logistic regression to see if I could find single variables with good predictive power. However, all the variables had an accuracy of .78. At this point I realized that the class imbalance in the number of 0’s and 1’s was affecting all of the models. Out of the 30,000 samples there were only 6636 defaults. This meant that I needed to use statistical methods to try and balance out the classes. The methods I used are undersampling and oversampling. So I went into Python and split my data into training and validation sets (removing sex and age). I then created an 8-fold cross-validation function that oversamples the training folds on each iteration so that the model is built on folds with an equal number of 0’s and 1’s. I cross-validated tree models with varying depths (5,10,15), linear SVM, Logistic Regression, and Random Forests (100 trees and depth 5 or 10). I used the accuracy, roc\_accurary, sensitivity, and f1 scores as my metrics. I reran the models again after removing all bill amount variables except the most recent. The model results didn’t change so I removed then from the data. I then repeated this process with an oversampling 8-fold cross validation method to compare the differences.

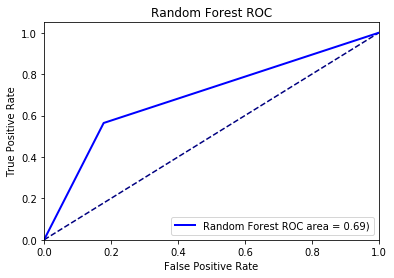
The last method I tried was a weighted classifier. Since false negatives are worse, I used the class weight attribute of models that had then to see if simply weighting the models works better than over or undersampling. So, I reran the tree, random forest, and logistic regression models penalizing false negatives more. However, this time I simply used the sklearn modules cross validation function and used 10 folds. I then compared the cross validated metrics in a table to try and pick the model that scored the best overall.

1. Results

The best model was the one that performed the best overall without being very gender biased. The following is the models I ran along with the cross validated metrics:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Recall | Roc | f1 |  |
| Tree(undersampled) | 0.61929 | 0.637891 | 0.62606 | 0.41689 |  |
| Tree(undersampled)(depth=10) | 0.71366 | 0.62636 | 0.680577 | 0.496805 |  |
| RF(under)(depth=10) | 0.71171 | 0.622266 | 0.679167 | 0.47984 |  |
| Logistic(balanced) | 0.6761 | 0.6155 | 0.7073 | 0.4581 |  |
| Tree(undersampled)(depth=5) | 0.73879 | 0.60253 | 0.68194 | 0.496805 |  |
| Tree(undersampled)(depth=15) | 0.664375 | 0.60253 | 0.68194 | 0.496805 |  |
| RF(under)(depth=5) | 0.738666 | 0.60234 | 0.68898 | 0.4965 |  |
| RF(under) | 0.728291 | 0.60097 | 0.6818 | 0.48552 |  |
| **RF(balanced)(depth=5)** | **0.7765** | **0.5964** | **0.77144** | **0.540877** | best!!!! |
| RF(balanced\_sub)(depth=5) | 0.774 | 0.5945 | 0.772189 | 0.540199 |  |
| Tree(oversampled)(depth=10) | 0.727666 | 0.59 | 0.677 | 0.4804 |  |
| SVM(oversampled) | 0.476625 | 0.58183 | 0.514964 | 0.242676 |  |
| SVM(undersampled) | 0.463333 | 0.575782 | 0.50431 | 0.24419 |  |
| RF(balanced\_sub)(depth=10) | 0.781 | 0.5733 | 0.76828 | 0.538896 |  |
| RF(balanced)(depth=10) | 0.779 | 0.5657 | 0.76921 | 0.53739 |  |
| Tree(oversampled)(depth=5) | 0.768333 | 0.557 | 0.69133 | 0.50683 |  |
| Tree(oversampled)(depth=15) | 0.71425 | 0.55292 | 0.655463 | 0.45227 |  |
| Tree(oversampled) | 0.726083 | 0.3916 | 0.6041 | 0.3788 |  |
| RF(balanced\_sub) | 0.803 | 0.325 | 0.772189 | 0.540199 |  |
| RF(balanced) | 0.803 | 0.3227 | 0.72935 | 0.4272 |  |
| Logistic(undersampled) | 0.695791 | 0.2492 | 0.533 | 0.2586 |  |
| Logistic(oversampled) | 0.699 | 0.2438 | 0.532685 | 0.2533 |  |
|  |  |  |  |  |  |

The way I qualified “best” was the one who had a good score in all 4 metrics and was hopefully the best at one of them. The balanced random forest model with depth=5 has the highest f1 score and is still good in all the other metrics. The bad news is that the random forest model has a 58% accuracy for predicting gender. However, other models had accuracy of around 56%. It might not be possible to get it to 50/50 because of the class imbalance of men and women (around 2/3 of the data is women). I then trained the random forest on all the models and go the diagnostics for the validation set. The diagnostics are as follows:



precision recall f1-score support

0 0.88 0.83 0.85 2360

1 0.47 0.57 0.52 640

accuracy 0.77 3000

Confusion Matrix:

[1949, 411]

[ 275, 365]

1. Discussion

There is a lot that could be changed in regard to the analysis of this problem. One example is that I could try to use different neural networks to classify. I could also try using SMOTE which is a balancing algorithm for data popularly used over the two I used. More feature engineering could also be done. Another option is data engineering looking closely at the outliers in the data to see if there is any pattern in the outliers with default. Another thing to do is to never even look at the classifications and simply try and estimate the probability of defaulting itself. My point is that there are a lot of different things that can be done, and time is a factor that must be considered. If given enough time to do all of these different things, then I would do them all. The results show that the random forest model works with a good amount of accuracy is also precise. The results are generalizable enough because the models were tested with cross validation. While there is a lot more that can be done, the results of my analysis are still significant. The only thing I would need to end this with is that when dealing with future predictions of the model use the probability of class rather than the classification itself. Businesses can then choose whatever threshold they want to use and classify themselves based on that.

1. Conclusion

The analysis that I have done here on this dataset shows that random forests are a good model to use to classify credit defaults. Again, the idea here is for banks to be able to look at circulating credit loans and be able to predict who will default so that they can take preventative measures to minimize loss. The random forest model has a recall of .57 for the validation set. This means that people who are going to default is correctly classified about 57% of the time. The model is also about 77% accurate. So, it classifies correctly at a relatively good rate. Using the probabilities generated by the model can save businesses money if they act in time.

UnderSampling Code:

import pandas as pd

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_validate

from imblearn.under\_sampling import RandomUnderSampler

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report

from sklearn.svm import LinearSVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import f1\_score

from sklearn.metrics import accuracy\_score

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import roc\_curve

from sklearn.metrics import auc as auc\_score

df = pd.read\_excel (r'default of credit card clients.xls')

pd.set\_option('display.max\_columns', 500)

pd.set\_option('display.width', 150)

new\_header = df.iloc[0] #grab the first row for the header

df.columns= new\_header

df.columns

df = df[1:] #take the data less the header row

df

y=df['SEX']

del df['AGE'] # Age is not an important predictor and is possibly gender biased

del df['SEX']

del df['BILL\_AMT2']

del df['BILL\_AMT3']

del df['BILL\_AMT4']

del df['BILL\_AMT5']

del df['BILL\_AMT6']

#y=df['default payment next month']

del df['default payment next month']

x=df

#Data Split

xTrain, xTest, yTrain, yTest = train\_test\_split(x, y, test\_size = 0.1, random\_state = 0)

x=xTrain

y=yTrain

rus= RandomUnderSampler(random\_state=20)

x\_res, y\_res =rus.fit\_resample(xTrain,yTrain.astype('int'))

def cross\_Validation(X,Y,estimator):

Folds=[]

y\_Folds=[]

results=[]

f1=[]

acc=[]

roc=[]

recall=[]

'''Creating the 8 folds through train test splits'''

split\_11, split\_12, y\_11,y\_12 = train\_test\_split(X,Y, test\_size=.5, random\_state=0)

split\_21, split\_22, y\_21,y\_22 = train\_test\_split(split\_11,y\_11, test\_size=.5, random\_state=0)

split\_23, split\_24, y\_23,y\_24 = train\_test\_split(split\_12,y\_12, test\_size=.5, random\_state=0)

split\_23, split\_24, y\_23,y\_24 = train\_test\_split(split\_12,y\_12, test\_size=.5, random\_state=0)

fold1, fold2, y\_fold1,y\_fold2 = train\_test\_split(split\_21,y\_21, test\_size=.5, random\_state=0)

fold3, fold4, y\_fold3,y\_fold4 = train\_test\_split(split\_22,y\_22, test\_size=.5, random\_state=0)

fold5, fold6, y\_fold5,y\_fold6 = train\_test\_split(split\_23,y\_23, test\_size=.5, random\_state=0)

fold7, fold8, y\_fold7,y\_fold8 = train\_test\_split(split\_24,y\_24, test\_size=.5, random\_state=0)

Folds.append(fold1)

y\_Folds.append(y\_fold1)

Folds.append(fold2)

y\_Folds.append(y\_fold2)

Folds.append(fold3)

y\_Folds.append(y\_fold3)

Folds.append(fold4)

y\_Folds.append(y\_fold4)

Folds.append(fold5)

y\_Folds.append(y\_fold5)

Folds.append(fold6)

y\_Folds.append(y\_fold6)

Folds.append(fold7)

y\_Folds.append(y\_fold7)

Folds.append(fold8)

y\_Folds.append(y\_fold8)

for i in range(0,len(Folds)):

test\_set=Folds[i]

y\_test= y\_Folds[i]

del Folds[i]

del y\_Folds[i]

train\_set= pd.concat(Folds)

y\_train= pd.concat(y\_Folds)

rus = RandomUnderSampler(random\_state=0)

x\_resampled, y\_resampled = rus.fit\_resample(train\_set, y\_train.astype('int'))

experiment=estimator.fit(x\_resampled,y\_resampled.astype('int'))

acc.append(accuracy\_score(yTest.astype('int'),experiment.predict(xTest)))

recall.append(recall\_score(yTest.astype('int'),experiment.predict(xTest)))

roc.append(roc\_auc\_score(yTest.astype('int'),experiment.predict(xTest)))

f1.append(f1\_score(yTest.astype('int'),experiment.predict(xTest)))

Folds.insert(i,test\_set)

y\_Folds.insert(i,y\_test)

accuracy=sum(acc)/len(acc)

results.append(accuracy)

rec=sum(recall)/len(recall)

results.append(rec)

auc=sum(roc)/len(roc)

results.append(auc)

f\_score=sum(f1)/len(f1)

results.append(f\_score)

return results

'''SVM= LinearSVC()

'''

'''tree=DecisionTreeClassifier(max\_depth=15) #this model shows to be a little bit biased can predidct gender with accuracy of .6

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='accuracy')

print(sum(CV['test\_score'])/10)

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='recall')

print(sum(CV['test\_score'])/10)

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='roc\_auc')

print(sum(CV['test\_score'])/10)

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='f1')

print(sum(CV['test\_score'])/10)

'''

'''forest= RandomForestClassifier(class\_weight="balanced")

'''

'''Average accuracy for CV Logistic is .695791 with avg TP .2492 and roc .533 and f1 .2586

Average Accuracy for Tree( max\_depth=10) is .71366 with avg TP .62636 and roc .680577 and f1 .482114

Average Accuracy for Tree is .61929 with avg TP .6378906 and roc .62606 and f1 .41689

Average Accuracy for Tree( max\_depth=5) is .73879 with avg TP .60253 and roc .681940 and f1 .496805

Average Accuracy for Tree( max\_depth=15) is .664375 with avg TP .633789 and roc .653229 and f1 .44618

Average Accuracy for SVM is .463333333 with avg TP .57578215 and roc .504310 and f1 .24419

Average Accuracy for RF is .728291 recall is .60097 and roc .6818 and f1 .48552

Average Accuracy for RF(depth=5) is .738666 recall is ..60234 and roc .68898 and f1 .4965

Average Accuracy for RF(depth=10 is .71171 recall is .622265625 and roc .679167 and f1 .479845

'''

#Pre Process Data and Create ROC Curve.

# code from https://medium.com/datadriveninvestor/computing-an-roc-graph-with-python-a3aa20b9a3fb

def runClassifiers(X, TestX, cfTest, CF):

RF= RandomForestClassifier(max\_depth=5, class\_weight='balanced')

RF.fit(X,CF.astype('int'))

predictProbForest= RF.predict(TestX.astype('int'))

#GET ROC DATA

fpr1, tpr1, thresholds1 = roc\_curve(cfTest.astype('int'), predictProbForest, pos\_label=1)

roc\_auc = auc\_score(fpr1, tpr1)

#GRAPH DATA

plt.figure()

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.plot([0, 1], [0, 1], color='navy', linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.title('Random Forest ROC')

plt.plot(fpr1, tpr1, color='blue', lw=2, label='Random Forest ROC area = %0.2f)' % roc\_auc)

plt.legend(loc="lower right")

plt.show()

if \_\_name\_\_=="\_\_main\_\_":

RF=RandomForestClassifier(max\_depth=5,class\_weight='balanced')

RF2=RandomForestClassifier(max\_depth=5,class\_weight='balanced\_subsample')

Oversampling code:

import pandas as pd

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_validate

#from sklearn.model\_selection import cross\_val\_score

from imblearn.over\_sampling import RandomOverSampler

import random

import warnings

warnings.filterwarnings('ignore')

from sklearn.metrics import roc\_curve

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report

from sklearn.svm import LinearSVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import f1\_score

from sklearn.metrics import accuracy\_score

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import recall\_score

df = pd.read\_excel (r'default of credit card clients.xls')

pd.set\_option('display.max\_columns', 500)

pd.set\_option('display.width', 150)

new\_header = df.iloc[0] #grab the first row for the header

df.columns= new\_header

df.columns

df = df[1:] #take the data less the header row

df

#y=df['SEX']

del df['AGE'] # Age is not an important predictor and is possibly gender biased

del df['SEX']

del df['BILL\_AMT2']

del df['BILL\_AMT3']

del df['BILL\_AMT4']

del df['BILL\_AMT5']

del df['BILL\_AMT6']

y=df['default payment next month']

del df['default payment next month']

x=df

#Data Split

xTrain, xTest, yTrain, yTest = train\_test\_split(x, y, test\_size = 0.1, random\_state = 0)

x=xTrain

y=yTrain

'''

tree=LogisticRegression(class\_weight='balanced') #this model shows to be a little bit biased can predidct gender with accuracy of .6

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='accuracy')

print(sum(CV['test\_score'])/10)

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='recall')

print(sum(CV['test\_score'])/10)

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='roc\_auc')

print(sum(CV['test\_score'])/10)

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='f1')

print(sum(CV['test\_score'])/10)'''

'''SVM= LogisticRegression(class\_weight='balanced')

CV= cross\_validate(,xTrain,yTrain.astype('int'),cv=10, scoring='f1')

print(sum(CV['test\_score'])/10)'''

'''tree=DecisionTreeClassifier(class\_weight='balanced', max\_depth=10) #this model shows to be a little bit biased can predidct gender with accuracy of .6

CV= cross\_validate(tree,xTrain,yTrain.astype('int'), cv = 10, scoring='roc\_auc')

print(sum(CV['test\_score'])/10)

'''

'''Average accuracy for CV Logistic is .580072 with avg TP .38465938?

Average Accuracy for Logistic(balanced) .6761 with recall .6155 and roc .7073 and f1 .4581

Average Accuracy for Tree( max\_depth=10) is .6825903 with avg TP .58658835

Average Accuracy for Tree is .60763340 with avg TP .40134973

Average Accuracy for Tree( max\_depth=5) is .69999021418370472 with avg TP .57457495

Average Accuracy for Tree( max\_depth=15) is .6584622830022163 with avg TP .522605

Average Accuracy for SVM is .5048803256241 with avg TP .51061456

Average Accuracy for Tree(max\_depth=10) only will bill variable .58 with avg TP .6 and "less gender biased accuracy of about .44 but can always just flip your choices so really .56"

Average Accuracy for SVM only will bill variable .5055 with avg TP .61 (mostly predicts 1's) and roc .5865 and f1 .0878

Average Accuracy for RF is .803 balanced and recall is .3227 and roc .7293590 and f1 .4272

Average Accuracy for RF(depth=5) is .77655 balanced and recall is .5964 and roc .771440 and f1 .5408766

Average Accuracy for RF(depth=10 is .779 balanced and recall is .5657 and roc .76921 and f1 .53739

Average Accuracy for RF is .803 balanced\_subsample and recall is .325 and roc .728453 and f1 .42989

Average Accuracy for RF(depth=5) is .774 balanced\_subsample and recall is .5945 and roc .772189 and f1 .540199

Average Accuracy for RF(depth=10) is .781 balanced\_subsample and recall is .5733 and roc .76828 and f1 .5388955

'''

'''

forest= RandomForestClassifier(class\_weight="balanced\_subsample", max\_depth=10)

CV=cross\_validate(forest,xTrain,yTrain.astype('int'),cv=10, scoring='f1')

print(sum(CV['test\_score'])/10)'''

'''

CV=cross\_validate(forest,xTrain,yTrain.astype('int'),cv=10, scoring='roc\_auc')

print(sum(CV['test\_score'])/10)

'''

def cross\_Validation(X,Y,estimator):

Folds=[]

y\_Folds=[]

f1=[]

acc=[]

roc=[]

recall=[]

results=[]

'''Creating the 8 folds through train test splits'''

split\_11, split\_12, y\_11,y\_12 = train\_test\_split(X,Y, test\_size=.5, random\_state=0)

split\_21, split\_22, y\_21,y\_22 = train\_test\_split(split\_11,y\_11, test\_size=.5, random\_state=0)

split\_23, split\_24, y\_23,y\_24 = train\_test\_split(split\_12,y\_12, test\_size=.5, random\_state=0)

split\_23, split\_24, y\_23,y\_24 = train\_test\_split(split\_12,y\_12, test\_size=.5, random\_state=0)

fold1, fold2, y\_fold1,y\_fold2 = train\_test\_split(split\_21,y\_21, test\_size=.5, random\_state=0)

fold3, fold4, y\_fold3,y\_fold4 = train\_test\_split(split\_22,y\_22, test\_size=.5, random\_state=0)

fold5, fold6, y\_fold5,y\_fold6 = train\_test\_split(split\_23,y\_23, test\_size=.5, random\_state=0)

fold7, fold8, y\_fold7,y\_fold8 = train\_test\_split(split\_24,y\_24, test\_size=.5, random\_state=0)

Folds.append(fold1)

y\_Folds.append(y\_fold1)

Folds.append(fold2)

y\_Folds.append(y\_fold2)

Folds.append(fold3)

y\_Folds.append(y\_fold3)

Folds.append(fold4)

y\_Folds.append(y\_fold4)

Folds.append(fold5)

y\_Folds.append(y\_fold5)

Folds.append(fold6)

y\_Folds.append(y\_fold6)

Folds.append(fold7)

y\_Folds.append(y\_fold7)

Folds.append(fold8)

y\_Folds.append(y\_fold8)

for i in range(0,len(Folds)):

test\_set=Folds[i]

y\_test= y\_Folds[i]

del Folds[i]

del y\_Folds[i]

train\_set= pd.concat(Folds)

y\_train= pd.concat(y\_Folds)

random.seed(25)

ros = RandomOverSampler(random\_state=0)

x\_resampled, y\_resampled = ros.fit\_resample(train\_set, y\_train.astype('int'))

experiment=estimator.fit(x\_resampled, y\_resampled.astype('int'))

acc.append(accuracy\_score(yTest.astype('int'),experiment.predict(xTest)))

recall.append(recall\_score(yTest.astype('int'),experiment.predict(xTest)))

roc.append(roc\_auc\_score(yTest.astype('int'),experiment.predict(xTest)))

f1.append(f1\_score(yTest.astype('int'),experiment.predict(xTest)))

Folds.insert(i,test\_set)

y\_Folds.insert(i,y\_test)

accuracy=sum(acc)/len(acc)

results.append(accuracy)

rec=sum(recall)/len(recall)

results.append(rec)

auc=sum(roc)/len(roc)

results.append(auc)

f\_score=sum(f1)/len(f1)

results.append(f\_score)

return results

if \_\_name\_\_ == '\_\_main\_\_':

print('main')

print(cross\_Validation(xTrain,yTrain,LinearSVC()))