UDC 519.85

Bobriiekhova K.M,[1]
Scientific supervisor Bocharov B.P.[2]

BINARY CLASSFICATION: CREDIT RISK PREDICTION

This thesis demonstrates how to perform cost-sensitive binary classification in Azure ML Studio to predict credit risk based on the information given on a credit application. The classification problem in this experiment is a cost-sensitive one because the cost of misclassifying the positive samples is five times the cost of misclassifying the negative samples.

In this experiment, we compare two different approaches for generating models to solve this problem:

— Training using the original data set;

— Training using a replicated data set.

In both approaches, we evaluate the models using the test data set with replication, to ensure that results are aligned with the cost function. We test two classifiers in both approaches: Two-Class Support Vector Machine and Two-Class Boosted Decision Tree.

We use the German Credit Card data set from the UC Irvine repository.

This data set contains 1000 samples with 20 features and 1 label. Each sample represents a person. The 20 features include both numerical and categorical features. The last column is the label, which denotes the credit risk and has only two possible values: high credit risk = 2, and low credit risk = 1.

The cost of misclassifying a low risk example as high is 1, whereas the cost of misclassifying a high risk example as low is 5.

We started by using the Metadata Editor module to add column names to replace the default column names with more meaningful names, obtained from the data set description on the UCI site. The new column names are provided as comma-separated values in the New column name field of Metadata Editor.

Next, we generated training and test sets used for developing the risk prediction model. We split the original data set into training and test sets of the same size using the Split module. To create sets of equal size, we set the option, Fraction of rows in the first output, to 0.5.

Because the cost of underestimating risk is high in the real world, we set the cost of misclassification as follows:

— For high risk cases misclassified as low risk: 5;

— For low risk cases misclassified as high risk: 1.

To reflect this cost function, we generated a new data set, in which each high risk example is replicated five times, whereas the number of low risk examples are kept as it is We split the data into training and test data sets before replication to prevent the same example from being in both the training and test sets.

[1] Student. O. M. Beketov National University of Urban Economy (NUUE) in Kharkiv
[2] Ass. Prof. of the KNIT Dep., O. M. Beketov NUUE in Kharkiv

To replicate the high risk data, we put the following R code into an Execute R Script module:

```
dataset <- maml.mapInputPort(1)
data.set <- dataset[dataset[,21]==1,]
pos <- dataset[dataset[,21]==2,]
for (i in 1:5) data.set <- rbind(data.set,pos)
row.names(data.set) <- NULL
maml.mapOutputPort("data.set")
```

Both the training and test data sets are replicated using the Execute R Script module.

Finally, we used the Descriptive Statistics module to compute statistics for all fields of the input data.

One of the machine learning algorithms requires that data to be normalized. Therefore, we used the Normalize Data module to normalize the ranges of all numeric features, using a tanh transformation. A tanh transformation converts all numeric features to values within a range of 0 - 1, while preserving the overall distribution of values.

The Two-Class Support Vector Machine module handles string features for us, converting them to categorical features and then to binary features having a value of 0 or 1, so there is no need to normalize these features.

In this experiment, we applied two classifiers: Two-Class Support Vector Machine (SVM) and Two-Class Boosted Decision Tree. Because we also used two datasets, we generated a total of four models:

— SVM, trained with original data;
— SVM, trained with replicated data;
— Boosted Decision Tree, trained with original data;
— Boosted Decision Tree, trained with replicated data.

We used the standard experimental workflow to create, train, and test the models:

Initialize the learning algorithms, using Two-Class Support Vector Machine and Two-Class Boosted Decision Tree

Use Train Model to fit the algorithm to the data and create the actual model.

Use Score Model to produce scores using the test examples.

The following diagram shows a portion of this experiment, in which the original and replicated training sets are used to train two different SVM models. Train Model is connected to the training set, whereas Score Model is connected to the test set.

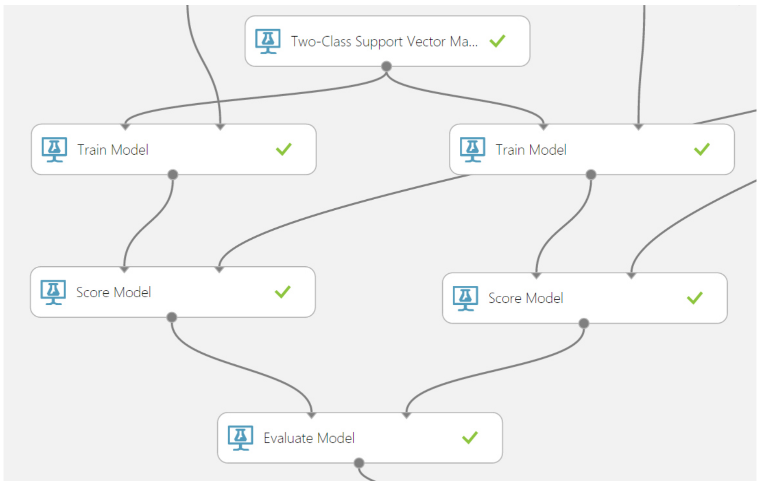


Figure 1 – A portion of experiment

In the evaluation stage of the experiment, we computed the accuracy of each of the four models. For this experiment, we have used Evaluate Model to compare examples that have the same misclassification cost.

The Evaluate Model module can compute the performance metrics for up to two scored models. Therefore, we used one instance of Evaluate Model to evaluate the two SVM models, and another instance of Evaluate Model to evaluate the two boosted decision tree models.

Notice that the replicated test data set is used as the input for Score Model. In other words, the final accuracy scores include the cost for getting the labels wrong.

The Evaluate Model module produces a table with a single row that contains various metrics. To create a single set of accuracy results, we first used Add Rows to combine the results into a single table, and then used the following simple R script in the Execute R Script module to add the model name and training approach for each row in the table of results.

```
dataset <- maml.mapInputPort(1)
a <- matrix(c("SVM","weighted",
 "SVM","unweighted",
"Boosted Decision Tree","weighted",
"Boosted Decision Tree","unweighted"),
nrow=4,ncol=2,byrow=T)
data.set <- cbind(a,dataset)
names(data.set)[1:2] <- c("Algorithm","Training")
maml.mapOutputPort("data.set")
```

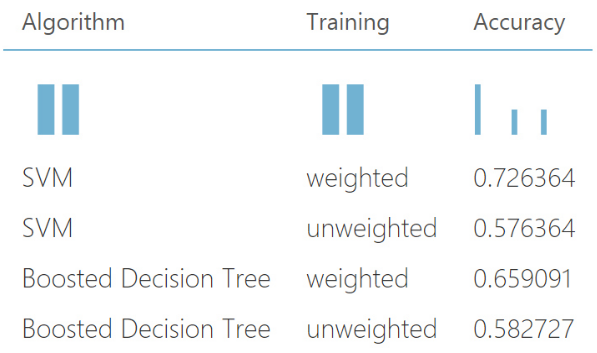
Finally, we removed the columns with non-relevant metrics using the Project Columns module.

To view the final results of the experiment, you can right-click the Visualize output of the last Project Columns module.

The first column lists the machine learning algorithm used to generate a model.

The second column indicates the type of the training set.

The third column contains the cost-sensitive accuracy value.



Algorithm	Training	Accuracy
SVM	weighted	0.726364
SVM	unweighted	0.576364
Boosted Decision Tree	weighted	0.659091
Boosted Decision Tree	unweighted	0.582727

Figure 2 – The final results of the experiment

From these results, you can see that the best accuracy is provided by the model that was created using Two-Class Support Vector Machine and trained on the replicated training data set [1][2].

REFERENCES

1. Tan P.N., Stembach M., Karpatne A. and Kumar V. “Introduction to data mining”. Pearson, 2nd edition. – 2018. – 864p.
2. Abu-Mostafa S., Magdon-Ismail M., Lin H-T. “Learning from data”. AMLBook. – 2012. - 213p.