Mystery Data Set

Hieu Nguyen, Nithya Devadoss, Ramesh Simhambhatla, Ramya Mandava Dec 2, 2018

Abstract

In this case study, we explore a mystery data set that has about **50 features** and **160,000** observations and the goal is to make a binary prediction. The business owner provided an open ended data set without much explanation or details about the features. We perform exploratory data analysis and mold the tidy data based on common sense approach. Though the business owner did not provide much details about the data, he did provide the impact of wrongful predictions – a **False Positive has -$500** and a **False Negative has a -$10,000** financial impact. Based on this expectation, '**Recall'** shall be the model metric to select the best model when there is a high cost associated with False Negative. Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (more True Positives, and less False Negatives are better). The approach to machine learning technique in this case study is of 2 stages to determine the best model and parameters using **ensemble learning** and then **stacking**. This helps improve machine learning results by combining several models; the predictions of the first models becomes the feature on the second level to train the second level model. This approach allows to produce a better predictive performance compared to a single model. However, at the end of our analysis we concluded that an independent Random Forest model with 5-fold cross validation yielded best results (**0.96** of **Recall** and **0.97 of roc_auc scores**).

1 Introduction

The goal of our project is to build a binary classification model which can predict the class label of the variable **y** given by our business user into classes of 0 or 1. Our dataset contains 160000 observations with 50 features, most of which are **numeric data** and few **categorical features**. We have cleaned our dataset before running through the model which is explained in detail. Our workflow is as follows:

- Import dataset into data-frame
- Exploratory Data Analysis
- Create Tidy dataset required for modelling
- Model Architecture - Stacking
- Model building / Hyperparameter optimization

- Evaluate Model performance and Report results.

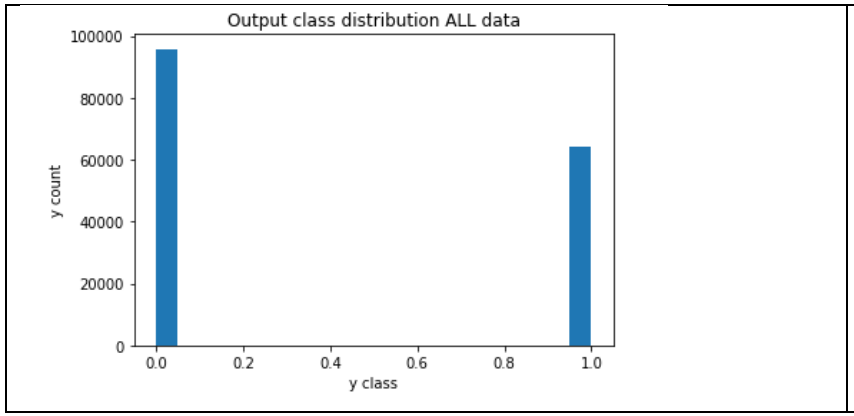
2 Modeling Work Flow

2.1 Data Set:

Our dataset has 160K observations with 50 attributes. It contains **~45 undefined numeric features** and categorical variables like **region, month, day of the week, percentage** information along with the output variable ‘y’. We have performed data cleanup as follows:

1. After observing the data sample, we have renamed the attributes 'x24', 'x29','x30','x32','x37' to ‘region', ‘month', ‘day', 'rate' and 'PL' (Profit/Loss) respectively.
2. Replaced all nulls to NaN
3. Replaced special characters like ‘%’ and ‘$’ in ‘rate’ and ‘PL’ fields respectively and converted them to float.

After summarizing the data, we found that there are approximately **30 data points** per feature that are **NaN’s**. These missing data points with NaN’s has no relation with other data points and are missing at random. Moreover, the distribution of the predictor variable remains the same with and without the data points containing these NaN’s. Hence we have decided to drop the data points and have verified the distribution as below:



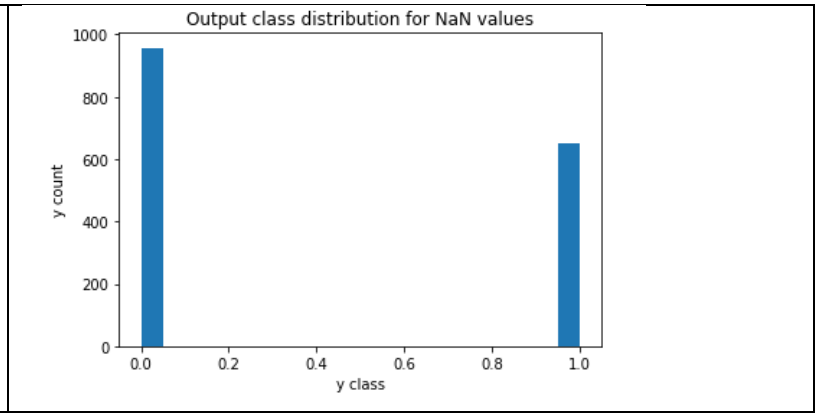


Table 1: Histogram of output variable

2.2. Exploratory Data Analysis:

We tried to analyze the distribution of the variables and identify any outliers using box plots before and after normalization which is shown in the **Table 2** below.
Normalization allowed us to scale the data to one standard and compare the effect of variables on the model without biasing.

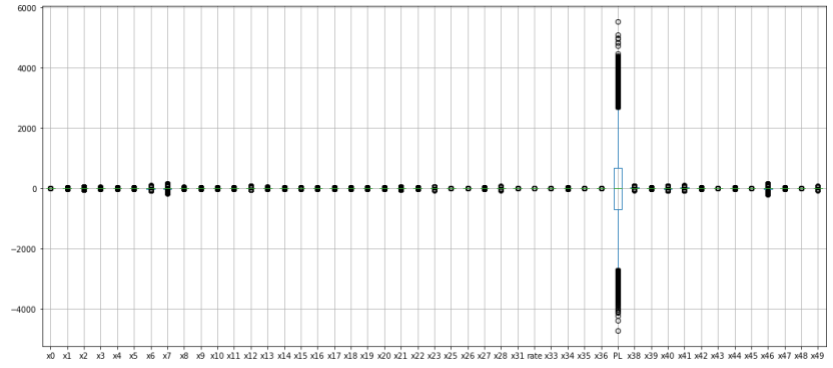
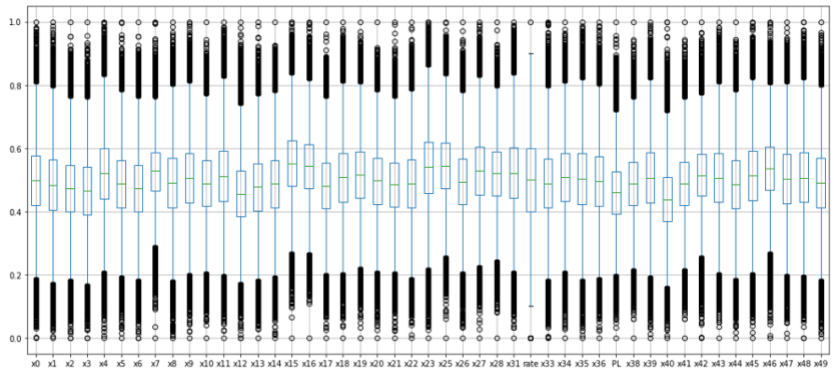
Distribution of raw data	Distribution after normalization
	

Table 2: Box plots of the data set

Coming back to our **categorical** variables which are region, day and month, we have performed one hot encoding to properly space them within our model. Now the normalized data set contains **158392** rows and **68** columns

Next we performed a **correlation plot** to find variables with high correlation. Based on the plot below in **Figure1** we find that variables **'x2'** has correlation with **‘x6’** and ‘**PL’** whereas **'x41'** is highly correlated with ‘**x39’**. Hence we have dropped both ‘x2’ and ‘x49’.

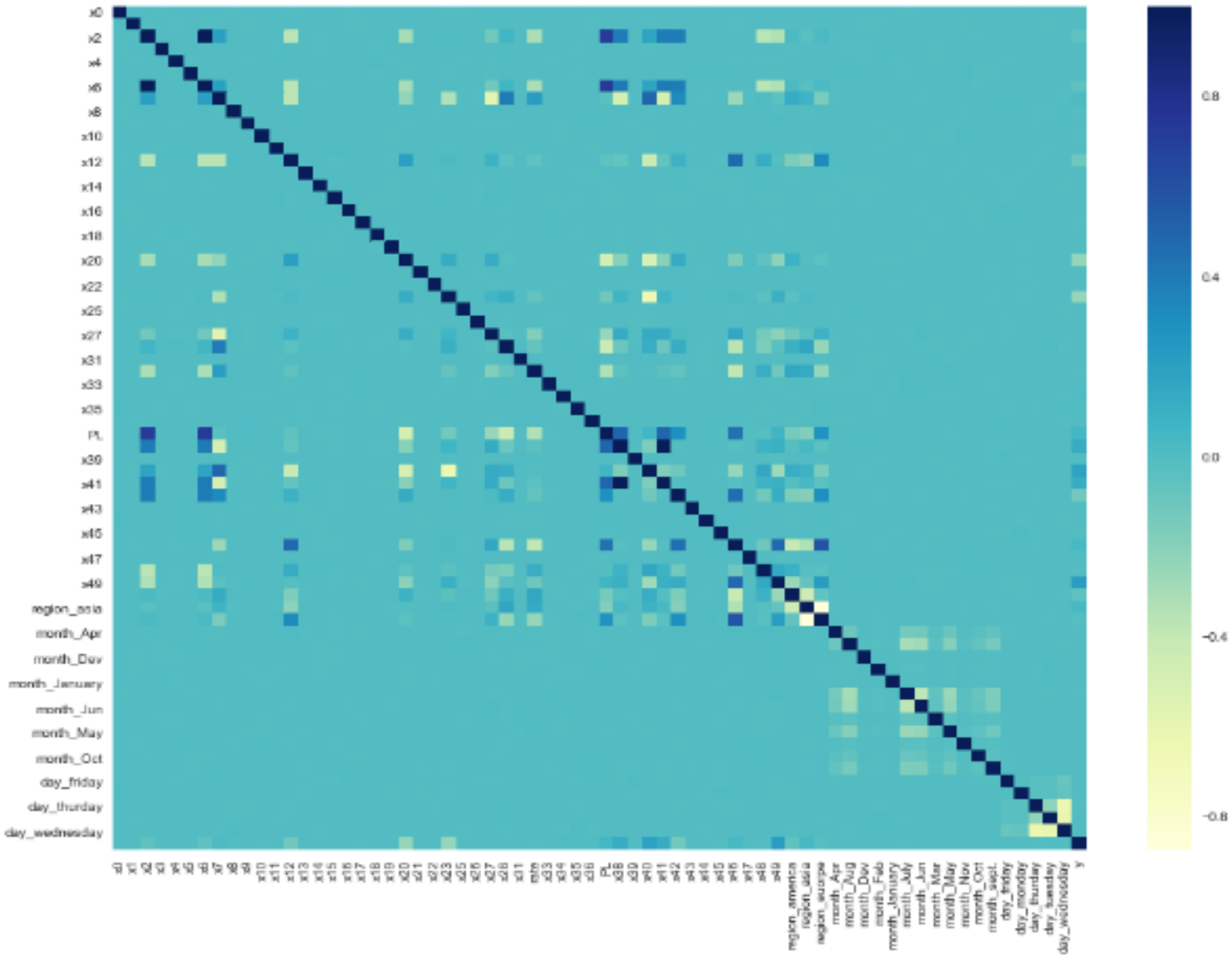


Figure 1: Correlation plot

2.3. Creating Tidy Data:

After dropping the NaN's with count of **1608** observations, the total number of good data in our data frame is around **158392**. So the clean data set which is ready to be fed to the model has 158392 observations with **66** attributes including the output variable '**y'**. All the attributes are float data types except for the imputed columns which are of unsigned int data type. The output variable is of datatype int64.

2.4. Model Architecture

Predictive accuracy of a model can be boosted by combining predictions of multiple machine learning models, generally referred to as an Ensemble method. Traditionally, 'weak' learners are combined and 'Voting classifiers' are used to predict the output. However, creating an ensemble of well-chosen group of strong and diverse models have proven to be more powerful in the recent times.

The simplest form of ensemble is achieved by averaging the predictions of the models, but a weighted average method yields better results in most cases according to literature. And an even better approach is to estimate the weights more intelligently by using another layer of learning algorithm. This method is referred to as 'model stacking'.

Model stacking is achieved by feeding the outputs of multiple Machine Learning algorithms to a second layer of learning algorithm as shown in the below Figure2 [2]:

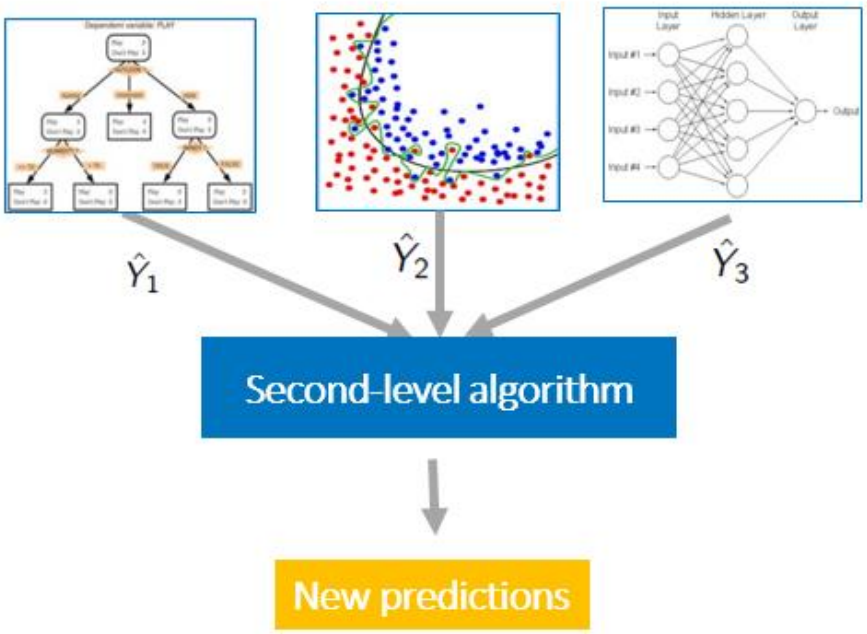


Figure2: Model Stacking with two layers

In our case study, Model stacking has been implemented where Random Forest and XGBoost have been employed in the first layer with optimized parameters for improving 'Recall' through a **Random Search** algorithm. These results are further fed to XGBoost in second layer to get to an ***accuracy*** of **0.92** and ***Recall*** of **0.92** as well.

Even though, 5 Machine Learning algorithms (Random Forest, XGBoost, GaussianNB, Logistic Regression) have been tested for layer 1, only RF and XGBoost have been selected for layer 1 implementation as their **accuracy** and **Recall** is above **90%.**

2.5 Model building / Hyper-parameter optimization

As explained in *section 2.4*, *Model stacking* has been implemented with ***layer-1*** having an ensemble of Random Forest and XG Boost models. Random forest model is implemented with hyper-parameter optimization and parameters are chosen in a way to maximize **Recall** score. Below Table 3 shows the performance of Random Forest with tuned parameters as well as XGBoost with default parameters:

Layer 1 Models	Parameters	Performance
Random Forest	n_estimators=250, min_samples_split = 4, max_depth=15, criterion='gini'	ROC_AUC = 0.97 (average of 5-fold CV) Recall = 0.96
XG Boost	Default parameters	ROC_AUC = 0.91 (average of 5-fold CV)

Table 3: Layer 1 Architecture

In ***layer-2***, the outputs of ***RandomForest*** and ***XGBoost*** from ***layer1*** have been fed to ***XGBoost*** but this time with tuned parameters ('su bsample': 1.0, 'min_child_weight': 1, 'max_depth': 4, 'gamma': 1.5, 'colsample_bytree': 1.0}) to optimize model performance on ***Recall*** and it yielded a result of **0.92**.

2.6 Evaluate Model performance and Report results.

Below **Table 4** shows all the models that were tested and their results. As can be noted, stacking yielded a score of 0.92 on Recall while Random Forest alone with hyper parameter optimization and 5-fold CV yielded a score of 0.97. Therefore, it is proposed to use Random forest alone for this particular dataset.

Models	Parameters	Performance	Layer	Dataset
Random Forest	n_estimators=250, min_samples_split = 4, max_depth=15, criterion='gini'	ROC_AUC = 0.97 (average of 5-fold CV) Recall = 0.96	Layer 1	Original Data set
XG Boost	Default parameters	ROC_AUC = 0.91 (average of 5-fold CV)	Layer 1	Original Data set

Logistic Regression	Default parameters	ROC_AUC = 0.76 (average of 5-fold CV)	Layer 1	Original Data set
GaussianNB	Default parameters	ROC_AUC = 0.74 (average of 5-fold CV)	Layer 1	Original Data set
Decision Tree	Default parameters	ROC_AUC = 0.84 (average of 5-fold CV)	Layer 1	Original Data set
XG Boost	{'subsample': 1.0, 'min_child_weight': 1, 'max_depth': 4, 'gamma': 1.5, 'colsample_bytree': 1.0}	ROC_AUC = 0.91(average of 5-fold CV) Recall = 0.92	Layer 2	Combined output of Random Forest and XGBoost from Layer 1

Table 4: Models performance and Results

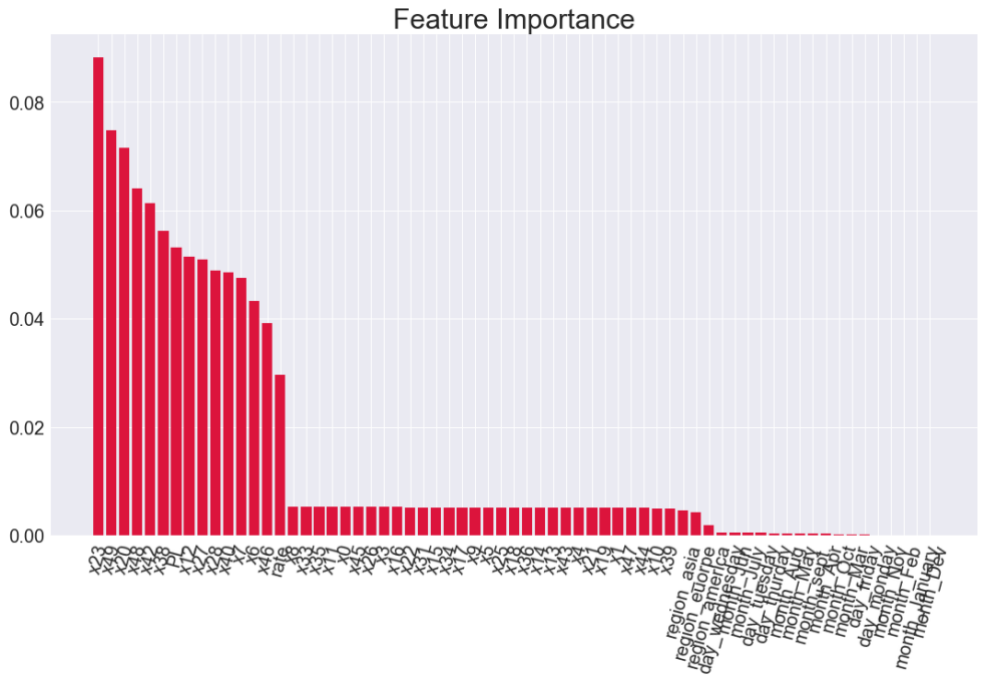


Figure 3: Feature Importance from Random Forest model

Above **Figure 3** shows Feature Importance from the Random Forest model. As can be seen, top 15 features are all numerical and masked (undefined features) except for **PL** (profit loss) and **rate** features that can be understood based on the sample data values.

3 Conclusion

Keeping our metric in mind (Recall), stacking has been performed with ensemble of Random Forest (with tuned parameters for optimizing our metric) and XGBoost in layer 1 and the output is fed to XGBoost (with tuned parameters) that yielded a score of 0.92. However, the stacking performance is lower than the individual Random Forest model performance with optimized parameters (0.97). Hence, it is suggested to use Random Forest for this particular data set.

With the tuned Random Forest model, we have obtained a recall value of 0.96, which means the model has correctly predicted the outcome for 96% of the instances. Since our objective is to further reduce the false negatives for the remaining 4% of predictions, we

would have to further tune the model to improve the true positive outcomes. This would help to reduce the negative dollar impact as per the objective of the Business Owner.

4 References

[1] Prof Slater's Jupyter Notebook sample and class material

[2] https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/

APPENDIX A

CODE:

Separate Jupyter notebook is being submitted as a reference to code.