Final Case Study

MSDS Fall 2019

7333 Quantifying the World

Brandon de la Houssaye, Bruce Granger, Daniel Serna

# Abstract

In this paper, we demonstrate the use of multiple machine learning techniques to garner insight from an obfuscated data set. Despite being provided little context into the provided data, we are able to produce business value by leveraging a variety of traditional classifiers along with neural networks. Through comparison of different approaches, we recommend an algorithm that achieves effective accuracy metrics while minimizing false positives and false negatives.

# 1 Introduction

We have been approached with a dataset and an interesting problem to solve. Our business stakeholders would like us to consume a dataset that has had domain context removed. All feature column names have been obfuscated, leaving it up to the team to garner insight as best we can. Ultimately, we have been tasked with building a model that can accurately predict the value of a column labeled “y.”

Through consultation with our business stakeholders, we have gained some minor context around the provided data set. At a high level, we know the data relates to insurance claims in some manner. We also know that our model predictions have monetary impact to the business. Most importantly, bad predictions cost the business money. Additionally, we have also been informed that false positives incur a 10x greater cost than false negatives. Thus, while achieving high accuracy with our model is desirable, we must ensure we minimize our false predictions and specifically focus on limiting false positives as much as possible. Because of this constraint, the recall metric of our models becomes important for analysis.

The rest of this paper is organized as follows. We will first discuss the data provided to us in section two with an explanation of data cleansing methods following in section three. Section four gives an overview of the models and techniques we applied against the data. Our results follow in section five. Lastly, our final conclusions are presented in section six.

# 2 Data

As mentioned above, the data provided has been obfuscated by having all feature column names removed. The data contains fifty feature columns labeled x0 through x49 and one prediction column labeled y. Of the fitty feature columns, forty-seven are numeric variables and three are categorical variables. We are dealing with a binary prediction variable as column y only contains the values 1 or 0.

The dataset contains 160,000 records and for the most part is well formed. Fortunately, we do not have much missing data to deal with as can be seen in Figure 1 (note we have renamed some of the original columns).



Figure – Count of missing values in original data

As can be seen from Figure 1, all columns have less than fifty missing data points. When compared to the total record count of 160,000, this is almost a negligible number of missing values (<0.03% for each column). We discuss our handling of the missing values in the Data Cleansing section.

Our data also appears to follow normal distributions and no significant deviations from normality were observed as can be from the boxplots in Figure 2.



Figure - Boxplots

As can be seen in the figure of above, we do not see evidence of any left or right skewness in the distributions. Two columns (x6 and x47) appear to have a wider distribution than the others, but still appear to follow a normal distribution. The boxplots also show little evidence of outliers. Thus, no transformations of the original data were deemed necessary.

We do have some collinearity in the data as can be seen from the Pearson correlation matrix in Figure 3.



Figure - Pearson correlation

As can be seen in Figure 3, columns x2 and x6 are highly correlated with each other and columns x38 and x41 are highly correlated with each other. We will discuss our handling of these correlated columns in the Data Cleansing section.

# 3 Data Cleansing

We first decided to deal with our correlated columns (x2 and x6, x38 and x41). Since these columns were perfectly correlated with each other (correlation coefficient = 1), we decided to simply drop the x2 and x38 columns.

Through exploratory data analysis, we were able to gain insight into the meaning of some of our columns. Column x24 contained the unique values of euorpe, asia, and america. Thus, it was clear to us this was data specifying a continent and we renamed the column to “Continent.” We also cleaned up the existing data by rectifying the spelling mistake of Europe and capitalizing the existing data.

Column x29 contained the unique values of July, Aug, Jun, May, sept., Apr, Nov, Oct, Mar, Feb, Dev, and January. Clearly, this column represents month data so we renamed the column to “Month.” We also standardized the existing values to a three-letter abbreviation and rectified the spelling mistake of Dev.

Column x30 contained the unique values of monday, tuesday, wednesday, thurday, and friday. It is evident this column represents the day of the week so we renamed the column to “DayOfWeek.” We also standardized the existing values to a three-letter abbreviation (while accounting for the misspelling of Thursday).

All values in column x37 had ‘$’ as the first character value, so this column was renamed to “Money.” We also stripped off the dollar sign and converted existing values to float datatypes so this column could be treated as a numerical variable. Similarly, all values in column x32 had ‘%’ as the last character value, so we stripped off the percent sign and converted existing values to float datatypes as well.

With our data standardized, money and percent columns converted to floats, we next handled our missing values. As mentioned previously, the percentage of missing values we are dealing with is very small (< 0.03% for each column). We decided to impute the missing values for our numeric variables with their respective column means. After this imputation, the only remaining columns left with missing values were the categorical columns as seen in Figure 4.



Figure – Count of missing values after mean imputation

As we can see in Figure 4, we still have 28 missing values in Continent, and 30 missing values in both Month and DayOfWeek. Since these represent such a low percentage of our total dataset, we decided to simply drop these records. We felt attempting to come up with a placeholder for these missing values could do more harm to our analysis than simply dropping the records.

# 4 Methods

We decided to employ multiple different modeling approaches against our data with a desire to compare the results to determine the optimal model choice. We utilized traditional classifiers, neural networks, and ensemble approaches as part of our model experimentation. All of our model runs employed an 80/20 train/test split.

We first attempted a number of Naïve Bayes classifiers including Gaussian, Multinomial, Bernoulli, and Complement variations. For the Complement and Multinomial variations, we utilized a Min-Max scaler between 0 and 1 on the data prior to running the models.

We next attempted a K-Nearest Neighbors model combined with a custom grid search method to determine optimal parameter choices. Our custom grid search method experimented with the number of neighbors as well as different weights (uniform and distance) and powers (1 and 2).

Since we are dealing with a binary prediction variable, a Logistic Regression model should work well and was attempted next.

We next utilized a Random Forest classifier with 100 estimators against the data.

Next, an XGBoost classifier was used.

We next utilized Principle Component Analysis and leveraged this with both a Support Vector Machine model and neural network implementation. Our neural network implementation consisted of eight layers utilizing both dense and dropout layers.

Finally, we attempted an ensemble approach. For the first level of stacking, we utilized Gaussian Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Complement Naïve Bayes, and Random Forest. The predictions from this level were fed into a subsequent level consisting of Support Vector Machine and XGBoost models. This output was fed into a final level consisting of a four-layer neural network.

# 5 Results

Given the constraint that false positives incur a 10x greater cost than false negatives, it is important that we look at both precision and recall metrics when determining the optimal model. Because both precision and recall are important for our problem at hand, we utilized model F1 score as the gauge for model effectiveness.

Figure 5 shows the results from our Logistic Regression model.

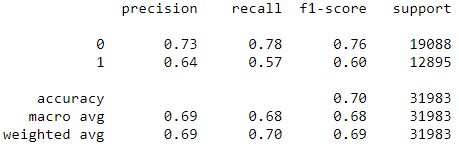


Figure - Logistic Regression Results

It looks like Logistic Regression model performs fairly well at predicting the 0 class with a 0.76 F1-score, however, it performs poorer at predicting the 1 class only achieving an F1-score of 0.60.

As mentioned, for our K Nearest Neighbors model, we utilized a custom grid search method to find optimal neighbors, weight function, and power parameters. Our results for optimal F1-scores for class 0 prediction are shown in Figure 6 and optimal F1-scores for class 1 prediction are shown in Figure 7.

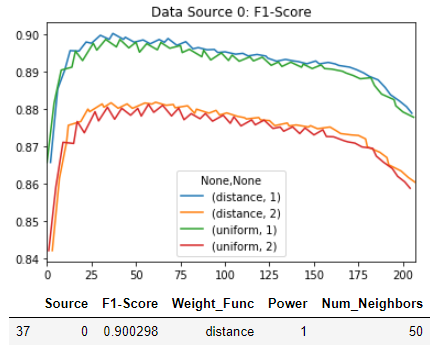


Figure - KNN Optimal Class 0 Prediction

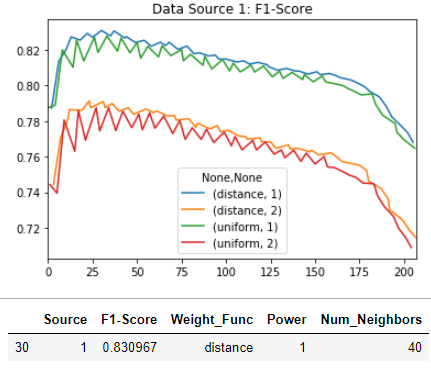


Figure - KNN Optimal Class 1 Prediction

As we can see from Figure 6 and Figure 7, we had much greater success with our KNN approach. We achieved an F1-score of .90 for predicting class 0 utilizing 50 neighbors, power of 1, and distance weight function. For predicting class 1, we achieved an F1 score of 0.831 using 40 neighbors, power of 1, and distance weight function. It appears a power of 1 and distance weight function is the optimal choice regardless of the class we are trying to predict.

After assessing all of the model results, the [optimal model for class 0] performed the best at classifying class 0 with an F1 score of [optimal F1 score for class 0]. The [optimal model for class 1] performed the best at classifying class 1 with an F1 score of [optimal F1 score for class 1].

# 6 Conclusions