Fuzzy Name Match Predictor

Final Report

By Michael Ward

# Problem Statement

MPW Financial Corporation is a Fintech company that processes payment transactions. The company needs to identify whether a person listed on a financial transaction is the same individual that is documented on government sanctions lists in order to prevent payments related to drug cartel members, weapons dealers, terrorist organization members, etc. Programmatic comparisons of individuals' names is a very difficult business problem. People can enter their names in different formats. For example, they can use shortened names such as “Mike” for “Michael”. People also have hyphenated surnames and could use a portion of the surname. In Latin and Arabic populations, it is common to have multiple first names and last names such as “Jose Miguel Castillo Diaz” and they may only enter a portion of their name such as “Miguel Diaz”. There is also the possibility of misspellings, transposed letters, etc. MPW Financial needs a machine learning module that can confidently match two inputted names to the same individual so that they can review the payment if the person is likely to be a sanctioned individual and stop the payment if necessary.

I was able to create a model that predicts whether a pair of first and last names are the same based on fuzzy variations. Multiple classification models were evaluated using a combination of features and data. The XGBoost Regressor model proved to be the best predictor of name matches with precision and recall of 98%.

# Data Wrangling

I was able to obtain a file of real names from an actual company’s transactions. The names were joined with other transactions based on a common hashed key. The names were filtered where either the first names or last names did not match. This data is confidential and is not uploaded to GitHub or exposed in the notebooks.

I also created a file of synthetic names to be used in the analysis which are not confidential. The first names were obtained from a GitHub repository <https://github.dev/Christopher-Thornton/hmni>. This file has a list of over 17,000 first name pairs which has the first name as well as alternative variations of each name. I was also able to obtain a file of common last names from a GitHub repository <https://github.com/arineng/arincli/blob/master/lib/last-names.txt>. I combined these files to create a dataset of name pairs which includes the first name, alternative variation, and last name.

I dropped records where the any of the names were missing or invalid. This included names that were all numeric. I also removed any of the real names where the key was not hashed so as to not expose and personal information. All of the names were converted to upper case in order to ensure the data was standardized for matching.

I found that many of the real data had names that had the same key, but the names were completely different. This seemed to be test data. Due to this issue, I used the FuzzyWuzzy package to create scores for matching the first names and last names. A perfect match would have a score of 2.0 (1.0 for first name exact match plus 1.0 for last name exact match). The average score was just over 0.5 which means that most of the names were completely different. The standard deviation was around 0.4. Based on this, I updated all records that were less than 2 standard deviations to a label of “No-Match”. These records had a match score of less than ~1.4. Records that had a score of at least 2 standard deviations were marked as “Match”. This resulted in over 400,000 records as “No-Match” and ~33,000 records as “Match”. I then grouped the records by key along with the maximum match score and removed all other records. This resulted in ~33,000 “Match” records and ~29,000 “No-Match” records for the real data.

For the synthetic data, I utilized the first name pairs from the GitHub repository and then added the same last name for each first name pair from the file obtained from the other GitHub repository. These were labeled as the “Match” records and had a total over ~17,500 records. I then copied this data and shuffled the first names and last names in order to create ~17,500 “No Match” records.

I combined the real and synthetic records in order to create the final dataset which contained 97,110 records and included the key along with the first and last names from “the list” and the first and last names from the transactions. Each record was labeled with “is\_match” as 0 (false) or 1 (true). Each record was also labeled as “confidential” based on whether it was real or synthetic data.

# Exploratory Data Analysis

The “is\_match” field is the target feature to predict. I removed a small number of the synthetic data that had a first name of “Nan” as this was interpreted as a numeric “Not a number” field in python. This resulted in the following statistics of “Match” and “No-Match” data for the real and synthetic data.

Chart, bar chart

Description automatically generated

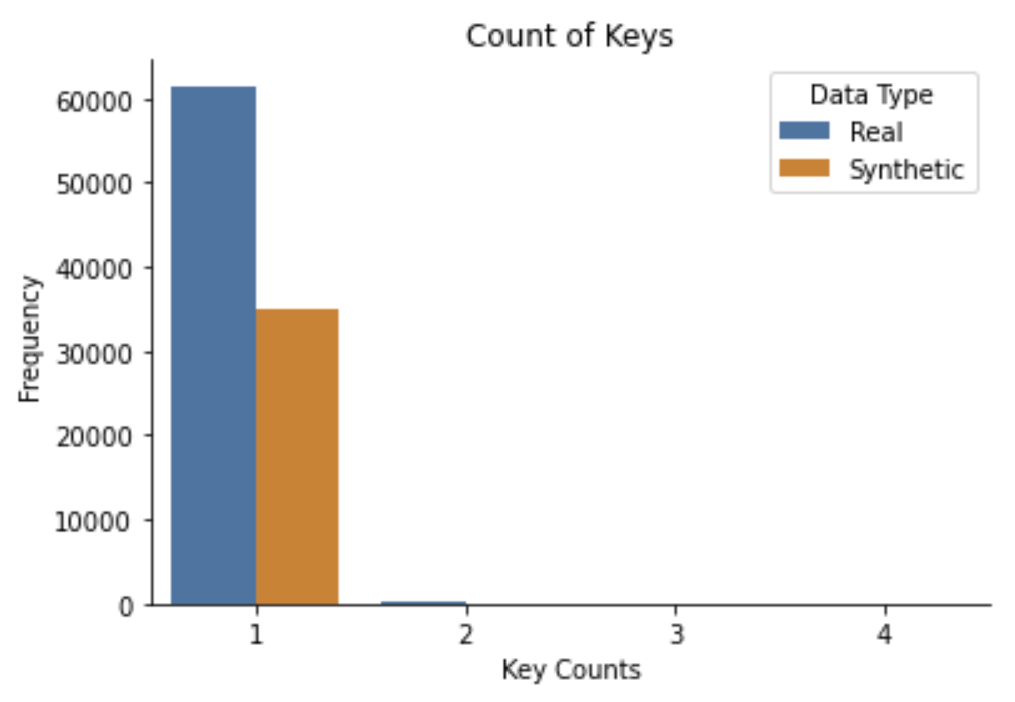
I explored how many records had exact matches on the first names and last names. The synthetic data does not have any first name matches as the match records were duplicated and then the names shuffled. Most the real data have first names that do not match. In the cases where the real data first names do match, the last names should not be exact matches. The synthetic data has exactly the same number of matches and no matches. This is because the matches only contain variations of first names. Most the real data have last names that do not match. In the cases where the real data last names do match, the first names should not be exact matches.

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

I also explored the number of duplicate keys. This can occur if the real transactions had the same key and the first and last name matches had the same FuzzyWuzzy similarity score. Most of the keys are unique, but some have 2-4 duplicates. The synthetic data does not have any duplicate keys. The real data does have approximately 300 records with duplicate keys. This can occur if the same individual had multiple transactions where the names provided were not exactly the same.

Chart, bar chart

Description automatically generated

The length of the names were also analyzed. The length of first names on the list range from 1 (first initial only) to >30 with a mean of 6 letters. The length of first names on the transactions were similar to the list first names with a range from 1 (first initial only) to >30 and a mean of 6 letters.

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

The length of last names on the list range from 1 (initial only) to >35 with a mean of 6-7 letters. The length of last names on the transactions were similar to the list last names with a range from 1 (initial only) to >35 with a mean of 7-8 letters.

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Finally, the names were analyzed to count where the first and last names contained multiple words (i.e. Miguel Luiz Rodriguez Diaz). The real data have approximately 4000 first names on the list that contain multiple words. The synthetic data does not have any first names on the list that contain multiple words. The real data have a small number of first names on the transactions that contain multiple words. The synthetic data does not have any first names on the transactions that contain multiple words.

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

The real data have approximately 2000 last names on the list that contain multiple words. The synthetic data does not have any last names on the list that contain multiple words. The real data have just under 10,000 last names on the transactions that contain multiple words. The synthetic data does not have any last names on the transactions that contain multiple words.

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

# Feature Engineering

First and last names are not features that can be utilized by machine learning models. In order to engineer features that could be used by the models, numeric features needed to be created based on the first and last names. In order to build these features, I leveraged code from <https://github.com/Christopher-Thornton/hmni>.

I utilized a “syllables” function to decompose each name into a list of syllables. The FuzzyWuzzy library was then used to convert the syllables into the following features for both the first and last names.

* Partial Ratio
* Token Sort Ratio
* Token Set Ratio

The Abydos NLP/IR library provides phonetic algorithms, string distance measures & metrics, stemmers, and string fingerprinters. This library was used to create the following features for the first and last names.

* Sum IPA (International Phonetic Alphabet) features
* PSHP Soundex First
* Iterative Substring
* BI-SIM similarity
* Discounted Levenshtein
* Prefix Distance
* Longest Common Substring (LCSstr)
* Modified Language-Independent Product Name Search Distance (MLIPNS)
* Strcmp95 Distance
* Match Rating Algorithm (MRA) Comparison
* Editex
* Syllable Alignment Pattern Searching (SAPS) Similarity
* FlexMetric Distance
* Jaro-Winkler Distance
* Higuera-Mico contextual normalized edit distance
* Sift4 Distance
* Eudex Distance
* ALINE Distance
* Covington Distance
* Phonetic Edit Distance

Finally, I leveraged the HMNI (Hello My Name Is) library to engineer HMNI Similarity features for the first and last names.

I opted to skip the Covington Distance feature because it was taking many hours to create these features.

# Model Selection

This project required a classification model as the target feature of “is\_match” is a categorical variable. I tested three different models in order to determine which was most useful for this problem. The three models that I tested were:

* Decision Tree Classifier (DecisionTreeClassifier)
* Logistic Regression Classifier (LogisticRegression)
* Extreme Gradient Boosting Classifier (XGBClassifier)

The metrics that I focused on while training and testing the models was the accuracy scores using precision and recall. I chose this because I wanted to maximize the precision of identifying name matches while also ensuring that the model had high recall.

In order to optimize the hyper parameters, I used a Randomized Search with 5-fold Cross-Validation. Randomized Search was chosen over Grid Search in order to reduce processing due to the size of the data. The following hyperparameters were tested for each model.

* Decision Tree (DecisionTreeRegressor)
  + Splitter: "best","random"
  + max\_depth: 1,4,8,12
  + min\_samples\_leaf: 3,5,7,9
  + min\_weight\_fraction\_leaf: 0,0.5
  + max\_features: "auto","log2","sqrt",None
  + max\_leaf\_nodes: None,10,30,50,70,90
* Logistic Regression (LinearRegression)
  + logspace: -4, 4, 20
* Extreme Gradient Boosting (XGBRegressor)
  + n\_estimators: 10, 50, 100, 500, 1000
  + max\_depth: 2, 3, 5, 10, 15
  + learning\_rate: 0.05, 0.1, 0.15, 0.20
  + min\_child\_weight: 1, 2, 3, 4

The following results were provided by the cross-fold validation for each model.

## XGBoost Classifier

Table

Description automatically generated

Chart

Description automatically generated

## Logistic Regression Classifier

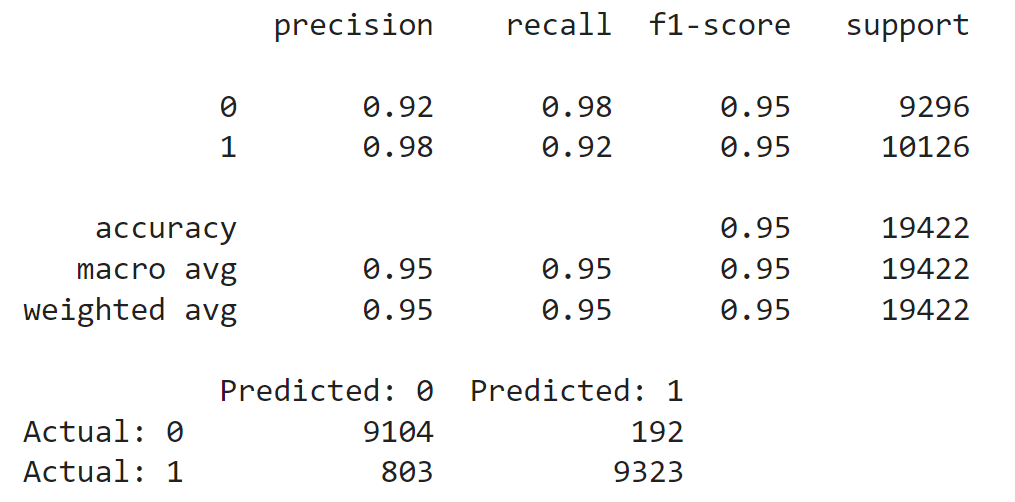
Table

Description automatically generated

Chart

Description automatically generated

## Decision Tree Classifier



Chart

Description automatically generated

The Extreme Gradient Boosting had slightly better results than the other three models.

# Conclusion

I decided to select the **Extreme Gradient Boosting (XGB)** model with the following hyperparameters as this resulted in the highest precision (0.98), recall (0.98), ROC/AUC score (0.998).

* n\_estimators: 500
* max\_depth: 10
* learning\_rate: 0.05
* min\_child\_weight: 1

The following features were determined to be the most important for the XGBoost Classifier model.

Graphical user interface, chart

Description automatically generated

The XGB model did take much longer to train than the Logistic Regression and Decision Tree Classifier models. If training time is an important factor, then the Logistic Regression model is a good alternative with a precision of 0.96, recall of 0.97, and AUC of 0.995.

The model can provide MPW Financial with the ability to identify potential to stop financial transactions where the individual is documented on government sanctions lists in order to prevent payments related to drug cartel members, weapons dealers, terrorist organization members, etc.

# Future Model Improvements

While the model performed well, potential improvements can be made.

* The real data had some quality issues where the names were not similar even though they had the same key. Due to this issue, the “is\_match” label had to be manufactured based on the FuzzyWuzzy similarity scores.
* The synthetic data had a comprehensive list of first name pairs that helped to identify variations such as “Richard” being the same as “Dick”. However, the last names had to be created based on a separate list of common last names. This resulted in each first name pair having the exact same last name.
* Acquiring more accurate, real data would help to build a better model.
* The name data was primarily from the US. Adding more international names such as Latin, Asian, Arabic, and Eastern European names would improve the model.
* The Covington Distance feature was excluded due to the length of time that it took to build the features. This feature could be added to improve the model.
* Other models could be tested and analyzed in order to improve the results. This could include other classification and deep learning models such as Siamese networks.