Capstone Project

Adrian Lievano

Machine Learning Engineer Nanodegree

Definition

Project Overview:

Banco Santander, S.A, is a multinational Spanish commercial bank and financial services company with offices in extended parts of Latin America; these include, but are not limited to, Chile, Mexico, El Salvador, and Brazil. Being the ninth-largest financial services company, as measured by revenue, Banco Santander owns data on millions of customers across the globe: daily transactions, checking and savings daily balances, investment holdings, debt, credit scores, and income are some of the rich features that empower financial institutions to create better and more personalized financial products to help more customers assess their personal financial health and to reach their goals.

For example, financial savings are essential for growth: they finance productive investments, provide a safety net for the future, and are associated with long-term distributed economic prospects, meaning gains are more equitably shared and enjoyed by the populace. In Latin America, however, reports show that, relative to "emergent Asian economics," gross public savings are one-third while private savings are "69 percent of advanced economies."[1] According to a 2016 report from the Inter-American Development Bank (IADB), low levels of consumer trust in the financial sector, resulted in only "16 percent of the region's adults" owning a savings account; in addition, banks are more restrictive, loaning 30 percent of gross domestic product (GDP) to the private sector versus 80 to 100 percent in more mature markets.

Part of the solution to these systemic economic problems involves improving access to the financial system and creating additional products and services that encourage savings and investment. Banco Santander Financial assembled a curated dataset of over 200,000 customers that contains privatized, financial information for over 200 different categories and challenged the Kaggle data science community to predict which customers will make a specific transaction in the future based on this dataset.

Problem Statement:

Banco Santander, a financial institution, aims to predict the next transaction a customer might complete based on historical banking information. This is a binary classification problem where the input data contains 202 feature variables. The performance of our machine learning algorithm will be tested on a provided test set by Banco Santander.

Metrics:

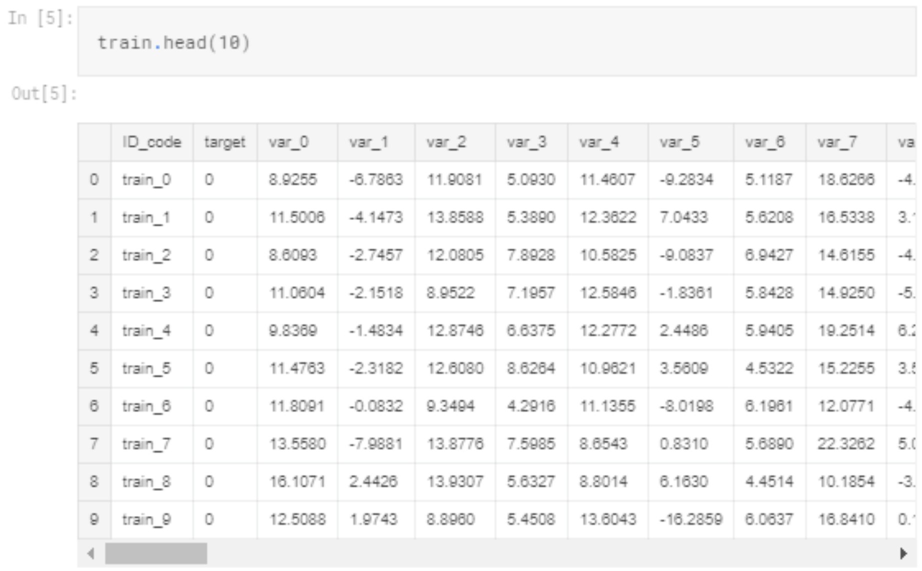
In a binary classification problem, the ability to precisely and repeatably reveal class A or class, respective of the two classes in question, is most important; for this machine learning problem, we will track three main variables:

1. Test Set Accuracy:
    a. Defined as the ratio of true positives and true negatives to the total amount of attempted predictions, accuracy results on the provided test dataset will determine how effective we were in classifier the next action customers take.

Analysis

Data Exploration:

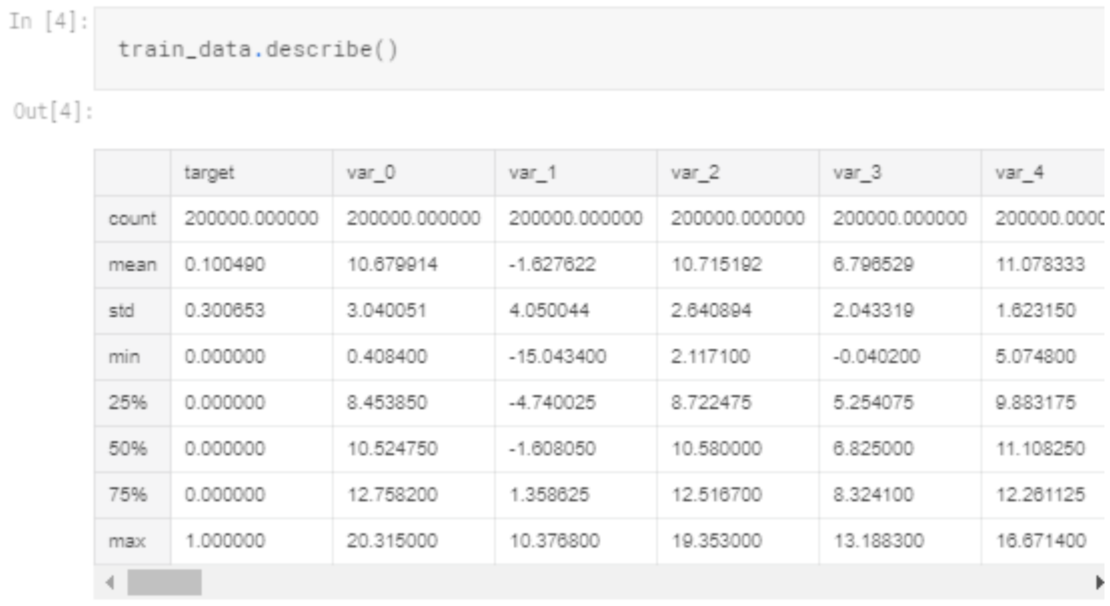
The provided train.csv file contains 200,000 rows with 200 features per row for each respective customer in their database. Given the large dataset, and the need to complete binary classification, there are many solutions to his problem: we could explore using a deep neural network, with some preprocessing using standard scaling techniques (set the mean to zero and the standard deviation to one for each feature), or other less computationally involved algorithms (decision trees, logistic regression, etc.).



```
In [5]:  train.head(10)
Out[5]:
```

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	va
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	-4.
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	3.
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	-5.
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	6.
5	train_5	0	11.4763	-2.3182	12.6080	8.6264	10.9621	3.5609	4.5322	15.2255	3.
6	train_6	0	11.8091	-0.0832	9.3494	4.2916	11.1355	-8.0198	6.1961	12.0771	-4.
7	train_7	0	13.5580	-7.9881	13.8776	7.5985	8.6543	0.8310	5.6890	22.3262	5.
8	train_8	0	16.1071	2.4426	13.9307	5.6327	8.8014	6.1630	4.4514	10.1854	-3.
9	train_9	0	12.5088	1.9743	8.8960	5.4508	13.6043	-16.2859	6.0637	16.8410	0.

Figure 1: A subsample of the Banco Santander training dataset. It shows a few (seven) of the 200 anonymized features associated with each ID_code, or customer. It is important to note that there are no categorical variables in the dataset; each feature is a continuous variable.



```
In [4]:  train_data.describe()

Out[4]:
```

	target	var_0	var_1	var_2	var_3	var_4
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.0000
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400

Figure 2: A statistical representation of a subsample of the training data. It shows the variation between the features in mean, standard deviation, minimum, and maximum values. If we chose to implement a black- or white-box machine learning algorithm, it would be more than likely be helpful if we preprocessed this input data.

Visualizations:

This binary classification challenge can be further understood after plotting the total number of examples of each class from the training data provided. Figure 3 shows that this problem is an unbalanced classification algorithm, leading us to take extra steps to ensure that if folds are created for multiple validation data sets, it is imperative that we include appropriate numbers of each class in each fold. Otherwise, we risk any implemented algorithm becoming biased to class 0 as opposed to class 1.

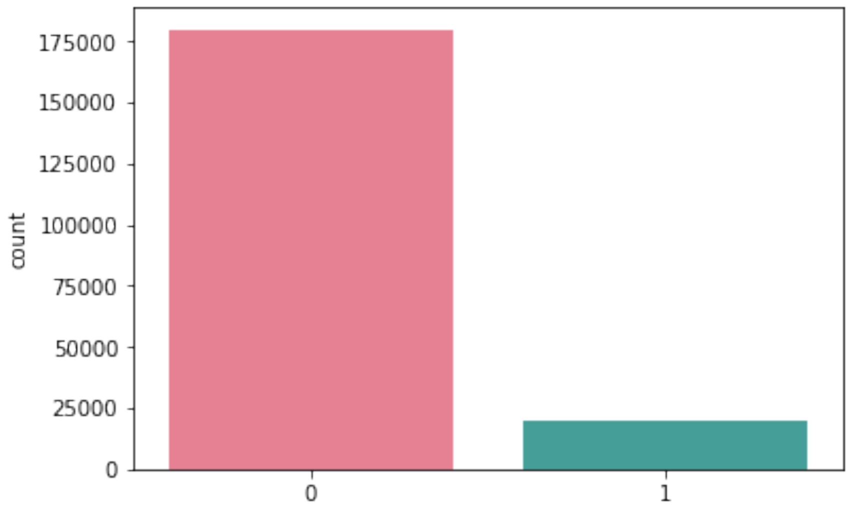


Figure 3: A bar graph showing the total number of instances for each target class that exists in the training data set. There are 175,000 examples of class 0 and 25,000 examples of class 1, creating a unbalanced binary classification problem that our algorithm will need to adjust for while training.

Algorithms and Techniques:

A variety of supervised learning algorithms were implemented and evaluated on the test set. Figure 4 shows the results, demonstrating that even computational intense techniques, like deep neural networks, did not significantly outcompete lesser models, such as logistic classification or a simple decision tree algorithm. To simplify the chart, we did not include all the tested hyperparameters for each supervised learning algorithm; instead, we describe some of the hyperparameters we optimized, if there were any, for performance.

Algorithm	Validation Set Accuracy	Test Set Accuracy Score (%)	Subset of Hyperparameters Optimized
Decision Tree Classifier	83%	56%	n/a
Logistic Regression w/o PCA	91%	62%	solver
Logistic Regression w/ PCA	90%	63%	solver
ADABoost (Ensemble)	90%	64%	algorithm, n_estimators
14-layer Neural Network w/ Batch Normalization	92%	65%	learning rate, depth of network, number of nodes
XGBoost (Ensemble)	92%	62%	learning rate, max_depth, n_estimators
Light GBM w/ Stratified Kfold	89%	90%	num_leaves, learning_rate, bagging_freq, feature_fraction, etc.

Figure 4: A chart describing all the attempted supervised learning algorithms and hyperparameters we tuned to obtain the respective validation and test set accuracies.

All the computation is performed on a Kaggle GPU server, providing enough RAM to run some of the more intense algorithms.

Benchmark Comparison:

To create an initial benchmark for the Banco Santander prediction problem, I implemented a simple decision tree classifier; the network trained in under five minutes and produced a score of 56% on the test set.

Methodology

Data Preprocessing:

From the dataset statistics, the mean and standard deviation between features varies; due to our data lacking a normal distribution and large variation in the mean and medians (including a large skew in the dataset), it is appropriate to apply non-linear scaling to reduce this skew. We can also scale the numerical features without changing the distribution by setting the mean to zero an standard deviation to one for each continuous variable (step 2).

The preprocessing in the Kaggle kernel consists of the following steps:

1. Randomize the training data
2. Scale features to have zero mean and standard deviation of one
3. (optional) for a subset of supervised learning algorithms, we explored implementing principal component analysis (PCA) on the initial training data to identify the minimum number of features required to capture 95% of the variation in the data. 95% was arbitrarily selected.
4. The training data is split into training and validation data sets.
5. The submission file is rebuilt from the predicted labels and submitted via Kaggle kernels.

Optional step (PCA Analysis):

For some of the supervised learning algorithms, we tested the impact of reducing the dimensional space of the original training dataset. We wanted to understand the most important feature variables when it came to classifying class 0 or class 1. The original dataset contained 200 feature variables; after deciding that we wanted to maintain 95% of the variance in the original training set, we implemented PCA from the sklearn.decomposition library and determined that we could reduce the number of features to 111. Although it was not a drastic reduction, it did provide insight that this dataset requires most of the feature variables to capture the nuances when it comes to this binary classification problem.

Implementation:

After preprocessing the input data and splitting it into training and validation sets, the assortment of classifiers are trained in a Kaggle GPU kernel with 16GB of available RAM. We took the following steps to run our code:

1. Load the training and validation data into memory, preprocess them as described above
2. Define the network architecture (whether it is a decision tree, logistic regression, neural network)
3. Train the algorithm
4. Test on validation set
5. Choose best parameters and submit results

Refinement:

As demonstrated in the algorithms and techniques section, the first algorithm we tried is a simple decision tree classifier that yielded 56% accuracy on the test set; this initial attempt did not use any feature preprocessing or PCA analysis. From there, we iterated on two major areas of the algorithms:

1. Preprocessing
2. Model selection

The final model derived was a Light GBM technique and trained in an iterative fashion, adjusting the parameters such as learning rate, num_leaves, bagging frequency, and much more. The final model we developed had a test set accuracy of 90%.

Results

Model Evaluation & Validation:

The final architecture and hyperparameters were chosen because they performed the best among the list of supervised learning algorithms. We however, choose to highlight the neural network implementation of our algorithm: it provides the most insight into understanding why larger, more complex supervised learning algorithms do not always produce the best results. After tuning (iterating among different combinations), the final architecture uses:

- Nodes = 1024
- Batch normalization
- ReLu activations
- Dropout

- Nodes = 512
- Batch normalization

- ReLu activations
- Dropout

- Nodes = 512
- Batch normalization
- ReLu activations
- Dropout

- Nodes = 254
- Batch normalization
- ReLu activations
- Dropout

- Nodes = 254
- Batch normalization
- ReLu activations
- Dropout

- Nodes = …
- Batch normalization
- ReLu activations
- Dropout

- Nodes = 2
- Batch normalization
- softmax activations
- Dropout
- Loss function = binary cross entropy

We ran the network through 300 epochs with a batch size of 300.

Justification:

Banco Santander is trying to predict future customer action based on historical data – that is anonymized for the purposes of this analysis and Kaggle competition. This application is useful for predicting with high accuracy whether a customer will perform action A or action B, defined in this binary classification problem.

Conclusion

Free-form Visualization:

```
dense_11 (Dense)          (None, 16)          272
```

```
_________________________________________________________________
dropout_9 (Dropout)          (None, 16)                0
_________________________________________________________________
batch_normalization_11 (Batc (None, 16)                64
_________________________________________________________________
dense_12 (Dense)             (None, 8)                 136
_________________________________________________________________
dropout_10 (Dropout)         (None, 8)                 0
_________________________________________________________________
batch_normalization_12 (Batc (None, 8)                 32
_________________________________________________________________
dense_13 (Dense)             (None, 8)                 72
_________________________________________________________________
dropout_11 (Dropout)         (None, 8)                 0
_________________________________________________________________
batch_normalization_13 (Batc (None, 8)                 32
_________________________________________________________________
dense_14 (Dense)             (None, 2)                 18
=================================================================
Total params: 1,255,410
Trainable params: 1,249,562
Non-trainable params: 5,848
_________________________________________________________________
```

Figure 5: The last 5 layers of the neural network algorithm. The total sum of trainable parameters for this dataset was nearly 1.3 million.

Reflection:

The process for solving the Banco Santander classification problem is summarized with the following steps:

1. Receive provided dataset from Banco Santander
2. Download and preprocess the data
3. Create a benchmark classification accuracy and measure test set accuracy
4. Try multiple supervised learning algorithms with and without PCA
5. Try different techniques for creating the validation data sets to balance the nature of this unbalanced classification problem to avoid training a biased algorithm
6. Train using ensemble techniques
7. Submit the predictions through Kaggle kernels.

I personally found step 5 to be the most difficult; initially, I thought I could just use iterate through a set of hyperparameters and supervised learning algorithms to see which one gives the highest accuracy. I unfortunately did not consider that having more examples of a class will inevitably lead to training a biased algorithm, which will generalize with poorer results. Although challenging, this portion of the project taught me the most; it doesn't matter how complex or powerful your choice of algorithm is if the preprocessing steps taken do not properly account for unbalanced classes.

Improvement:

This unbalanced binary classification problem requires clever creation of the validation sets and can be solved using a variety of supervised learning techniques. The best solution, however, relies on stratified kfold processing in combination with an ensemble technique like Light GBM or XGBoost. To improve performance in this model, I would like to try other preprocessing techniques and perhaps using stratified kfold processing with a neural network and Light GBM. Although the improvements would marginally improve the result and we more than adequately hit a satisfactory accuracy, it could yield an improvement in the overall prediction for Banco Santander. Instead of focusing on further optimization, I'd ask Banco Santander to think of more creative datasets we can combine with this one to determine additional products and services that could be of further use for their customers.

There were a few techniques that I'd like to explore: some of the public Kaggle kernels achieved high test set accuracy using Light GBM, an ensemble-based technique, along with stratified kfold. I think the best solution would involve preprocessing the features to have zero mean and standard deviation of one, applying PCA, and creating the appropriate validation sets that appropriately represents each class in each validation fold. From there, I'd implement a deep neural network (20-30-layer network with ReLu activations and batch normalization between nodes). After reviewing some of the kernels that achieved over 92% test set accuracy, most of them used clever preprocessing techniques and ensemble-based methods while exploring a large space of hyperparameters; some kernels needed to run their code on multiple GPUs and run for multiple days to find an optimal solution. Due to my limited computational resources, I capped myself at 8 hours runtime.

Acknowledgements:

I thank Kaggle, its contributors, and the publicly-available kernels for their help gathering the data, formulating potential solutions, and presenting these results to the Udacity review committee.

References:

1. Bahney, Anna. "40% Of Americans Can't Cover a $400 Emergency Expense." *CNNMoney*, Cable News Network, 2018, money.cnn.com/2018/05/22/pf/emergency-expenses-household-finances/index.html.
2. Santander, Bank. "Santander Customer Transaction Prediction." *Kaggle*, www.kaggle.com/c/santander-customer-transaction-prediction/leaderboard.
3. Taylor, Matthew. "Latin America's Savings Problem." *Council on Foreign Relations*, Council on Foreign Relations, 14 July 2016, www.cfr.org/blog/latin-americas-savings-problem.