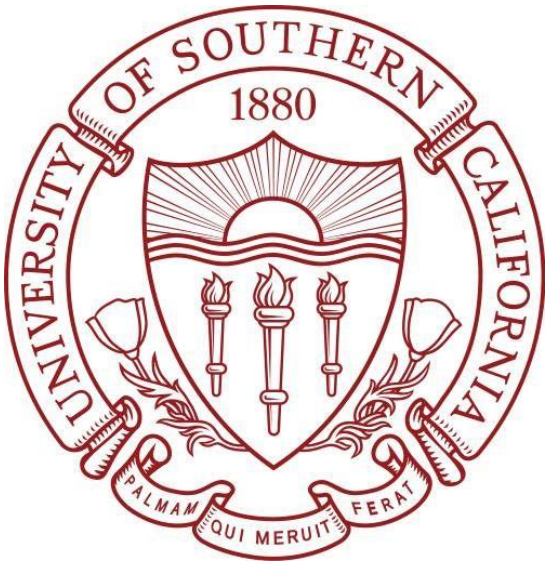
ISE-529 - Friday Section
Final Project
Santander Customer Transaction Prediction



Group:	**Hsuan-Hsuan Wu**	**4492-1547-13;**	**Sean Eskew**	**8439-7167-36**
	Junteng Zheng	**6496-1591-96;**	**Boxue Li**	**9814-3331-87**

Project Introduction:

For our project, we selected the Kaggle Competition, "Santander Customer Transaction Prediction" with the following problem statement. *"At Santander, our mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals[...] In this challenge, we invite Kagglers to help us identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted."*

This challenge is a categorical problem, where we are given information regarding a consumer and must predict if they will be a returning customer at Santander. We are given two sets of data, our "train" set that we will use to build, test, and refine our models; and a "test" set to be used to score our predictions by Kaggle.

Data Exploration/Preprocessing:

Our data pre-processing was divided into four stages: exploration, data-balancing, feature engineering, and feature normalization.

Our data exploration was used to conduct an initial analysis of our dataset to find characteristics that will affect our modeling. In this search, we found that our data consisted of 200 numeric features, one binary response variable, and an identifier column; our "train" set had 200,000 rows and our "test" set also had 200,000 rows that we would use to predict. We also discovered that our response variable, 'target', was very unbalanced within the train set with 90% of the responses being 'target 0' and only 10% 'target 1'. Next, we examined the correlation between our features, finding a very low correlation with a maximum value of 0.00984. This showed that values have little influence on each other and no features should be eliminated immediately.

For data-balancing; we originally attempted to create our model on the unbalanced data. However, we found that with the unbalanced dataset our models performed poorly, too often predicting 'target 0'. To rebalance our data, we under-sampled the training dataset randomly as 50% 'target 0', 50% 'target 1' and ultimately found that it improved the accuracy of prediction. This rebalancing also had the effect of decreasing our data from 200,000 rows to 40,196.

Through our research in the notebooks of past participants; we found that limited feature engineering will help the performance of our models. In "Santander EDA and Prediction" by Gabriel Preda, we found that adding new features based on descriptive statistics of each row of our data set provides a better prediction. With that, we added 8 new features based on the provided 200 features that included sum, minimum, maximum, mean, standard deviation, skewness, kurtosis, and median.

For feature normalization, we tested three normalization algorithms for our data. Our three methods were MinMaxScaler, QuantileTransformer, and StandardScaler. Of those,

StandardScaler provided us the best performance. StandardScaler centers each feature to a mean of 0 with a normal distribution and standard deviation based on the data set. Once our preprocessing was complete, we split the data into a train and test set and began our modeling.

Data Modeling:

We conducted our modeling in three phases. Our first phase was executing multiple untuned models to identify which models appear to perform the best in our data set. The second phase was tuning our top models to increase their performance. Our final phase was incorporating lessons learned from previous competitors' notebooks to continue our improvement.

For the first phase, we picked models from a variety of different categories to test against our data. Our initial trials included tests with Linear Models, Support Vector Machines, Nearest Neighbors, Naive Bayes, Decision Trees, Ensemble Models, and Neural Networks. Of these, the top performers were our Gaussian Naive Bayes, Linear Discriminant Analysis, Logistic Regression, Extreme Gradient Boosting Model, and Light Gradient Boosting Model.

We now began tuning our hyperparameters. Our Linear Models and Naive Bayes models had limited hyperparameters to be modified and we found that our performance remained roughly the same through tuning. Our XGB had many more parameters that could be adjusted; however, with the size of our data, it often took hours to run GridSearchCV. We also used Amazon Web Services to tap their higher power GPUs, but this had only a slight impact on the speed of our tuning. We were still able to conduct our tuning but limited our cross folds and the number of hyperparameters tested.

Finally, to increase our score further we researched the open-source notebooks of past competitors. Inspired by "Santander Magic LGB 0.901", we utilized the Light Gradient Boosting model featured within their project. Light GBM is a fast, distributed, high-performance gradient Boosting framework based on a decision tree algorithm. With the learning of the Light Gradient Boosting Model, we were able to increase our accuracy rate by almost 10%.

In an attempt to improve beyond the Light GBM; we create a voting model combining our XGB and LGB models, which can be seen in Appendix B. Unfortunately, this regressed to lower scores and the pure LGB model remained our top performer.

Results:

A detailed list of our Kaggle submission results can be found in Appendix A. We saw the best results with the Light Gradient Boost Model, followed by the voting model with LGB+XGB, then the pure XGB. Our Light GBM pure model earned us the accuracy score of 0.89733, the maximum accuracy achieved on the leaderboard was 0.92573.

Appendix A:

The score of attempts

All **Successful** Selected

Submission and Description	Private Score	Public Score	Use for Final Score
vote_sub2.csv 3 hours ago by Junteng Zheng Vote	0.80257	0.80657	☐
vote_sub.csv 3 hours ago by Junteng Zheng Vote	0.80041	0.79921	☐
sub11.csv 3 hours ago by Junteng Zheng LDA method	0.50000	0.50000	☐
sub10.csv 4 hours ago by Junteng Zheng LightGBM	0.89674	0.89733	☐
sub9.csv 4 hours ago by Junteng Zheng tuned XGBoosting	0.79567	0.79589	☐
sub8.csv a day ago by Junteng Zheng tuned XGBoosting	0.79209	0.79859	☐
sub7.csv a day ago by Junteng Zheng XGBoosting	0.78648	0.78425	☐
sub6.csv a day ago by Junteng Zheng Gaussian	0.65207	0.65986	☐

Appendix B:

The vote combined LGB and XGB

In [65]:
```
sub5 = sub6.round({'target': 0})
```

Out[65]:

	ID_code	target
0	test_0	1.0
1	test_1	1.0
2	test_2	1.0
3	test_3	1.0
4	test_4	0.0

In [72]:
```
vote_df = pd.DataFrame()
```

In [73]:
```
vote_df['XGB'] = yp5
vote_df['LGBM'] = sub5.target
```

In [74]:
```
vote_df['SUM'] = vote_df.sum(axis=1)
vote_df.loc[vote_df.SUM <= 1, 'Blend'] = 0
vote_df.loc[vote_df.SUM > 1, 'Blend'] = 1
```

In [75]:
```
vote_df[:5]
```

Out[75]:

	XGB	LGBM	SUM	Blend
0	0	1.0	1.0	0.0
1	1	1.0	2.0	1.0
2	1	1.0	2.0	1.0
3	1	1.0	2.0	1.0
4	0	0.0	0.0	0.0

In [76]:
```
vote_sub = pd.DataFrame({"ID_code":data_test["ID_code"].values})
vote_sub["target"] = vote_df.Blend
vote_sub.to_csv("vote_sub2.csv", index=False)
```

In [3]:
```
# the score of vote is 0.80657.
# we selcet LGB as our final model
```