EE 510: Machine Learning Theory & Algorithms **Spring 2022**

Mini Project 2
Due: May 13, 2022, 11:59PM PT

Student Name: Martín Rodriguez *Instructor Name: John Lipor*

1 Problem description

Statistical analysis plays a large part in insurance claim policies. Insurance companies often want to know the risk associated with an individual when offering them insurance packages or evaluating their current policy. There is much room for improvement in how these companies use machine learning to build models from customer data. The goal for this project is to take a set of insurance customer data provided as part of a Kaggle competion, and use the data to train a model to determine whether an individual policy holder will likely file a claim within the next year [1]. The XGBoost (Extreme Gradient Boosting) algorithm was used to train the model [2].

2 Exploratory data analysis (EDA)

This dataset consists of a training set with 595212 entries and test set with 892816 entries. The training set has 59 columns, one of which is the target feature, a binary value: 1 if the individual filed an insurance claim and 0 if they did not. The test set has 58 columns and lacks the target feature. Of the 595212 samples in the training set, 573518 samples are classified as 0 and 21694 are classified as 1. As a percentage, there are 96.36% negatives compared to 3.64% positives, indicating that this dataset is heavily imbalanced. Null values in this dataset are indicated by `-1`.
The features are grouped into four categories: `'ind'`, `'reg'`, `'car'`, and `'calc'`, and the results of a preliminary analysis are shown in table 1.

Category	Data type	Count	Total
`'ind'`	binary	11	18
	categorical	3	
	continuous/ordinal	4	
`'reg'`	binary	0	3
	categorical	0	
	continuous/ordinal	3	
`'car'`	binary	0	16
	categorical	11	
	continuous/ordinal	5	
`'calc'`	binary	6	20
	categorical	0	
	continuous/ordinal	14	

Table 1: Data grouping

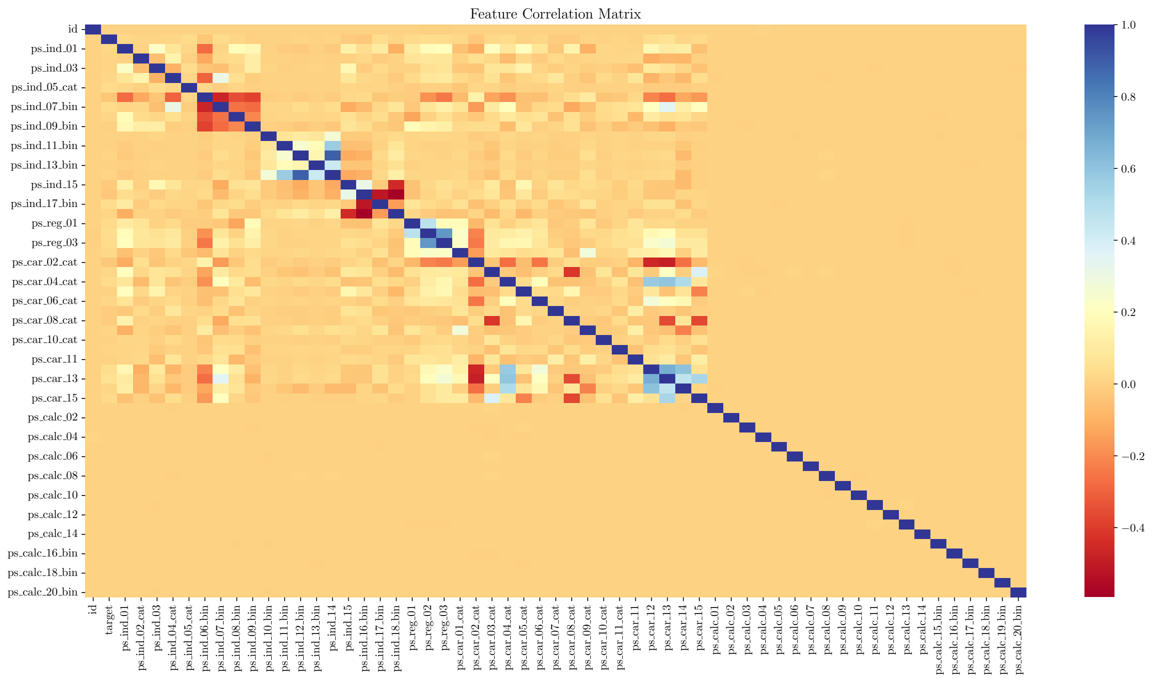


Figure 1: Feature correlation matrix

The feature correlation matrix shown in Figure 1 indicates that features in the `'calc'` category have no correlation with the target feature, so these were removed from the feature space. Figure 2 shows the updated feature correlation matrix.

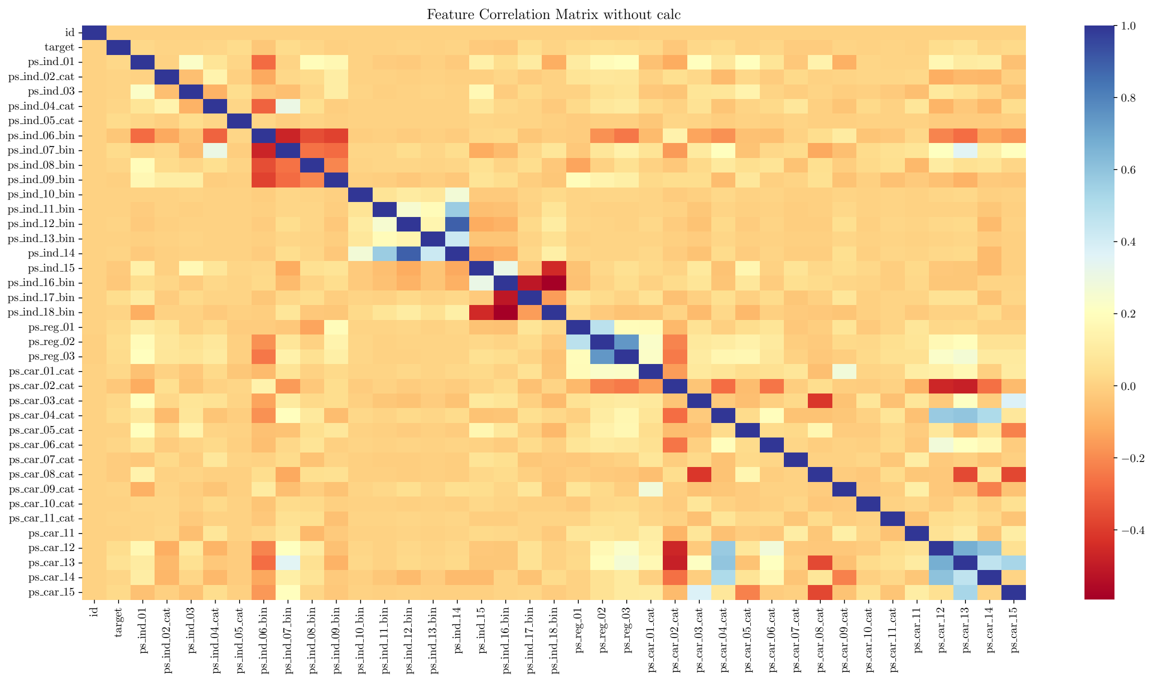


Figure 2: Feature correlation matrix without 'calc' features

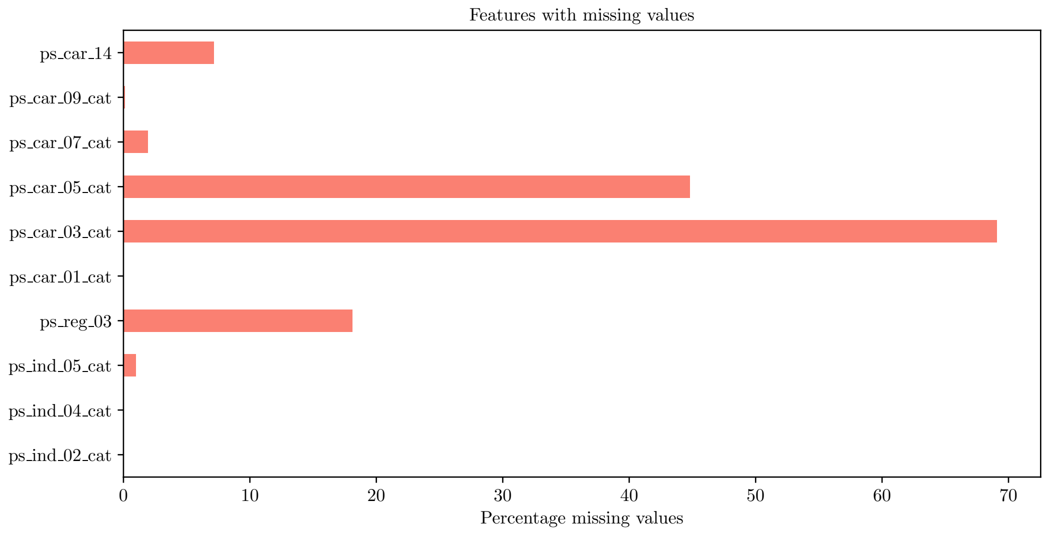


Figure 3: Features containing missing values

Since `'ps_car_03_cat'` and `'ps_car_05_cat'` both have a large percentage of missing values, they should be dropped as well.

3 Challenges

This dataset proved to be quite challenging to interpret. Because of the labeling scheme for the features, it was not possible to form an intuition about how the features might depend on one another. It turned out that several of the features did not contribute at all to the model, so these had to be disposed of. Another challenge was the fact that the data was heavily imbalanced, creating more of an "anomaly detection" problem. To deal with an imbalanced dataset, the training set can either be subsampled, oversampled, or a combination of the two such that there is an even split between negative and positive examples. Subsampling reduces the number in the of majority class and thus incurs information loss, whereas oversampling creates redundant or similar samples to those in the minority class [3].

4 Approach

Here is the general approach followed for this dataset:

1. Exploratory data analysis
   (a) Look at correlation, collinearity, class imbalance, etc.
2. Preprocessing
   (a) Remove uncorrelated features
   (b) Remove features with the most missing values
   (c) Fill missing values
   (d) One-hot-encode categorical features
3. Training
   (a) Shuffle and scale the data

(b) Create training and validation sets

(c) Subsample the data for a 50/50 class split

(d) Train the model

(e) Validate the model

It was observed that the features in the `'calc'` category had no correlation with the target, so these were removed, along with `'ps_car_03_cat'` and `'ps_car_05_cat'`, which had the most missing values. The categorical and binary missing values were filled with their column mode. Other missing values were filled with their column median. Highly collinear features do not add much value to a potential model, so methods of removing these were considered [4]. However, after removing collinear features with a Pearson Coefficient higher than 0.98, no performance increase was observed, so these were kept in the final model [5]. The categorical features were one-hot-encoded for easier interfacing with `scikit-learn`'s XGBoost API. Due to time constraints, a validation set with a test size of 20% was used to validate the model because this was the easiest to implement. Also for lack of time, the simplest subsampling approach was chosen, which just takes a random subset of the majority class such that the ratio between the classes is equal for training. The model was trained on this subsampled data using XGBoost with a binary logistic objective function, 100 boosting rounds, a maximum tree depth of 3, and a learning rate of 0.1. The default parameters for `scikit-learn`'s implementation actually gave a good tradeoff between performance and computation time. After training, the model was then validated using the non-resampled validation set.

5 Evaluation and summary

Because the dataset is heavily imbalanced toward the negative class, accuracy is not a good metric to use to determine the validity of our model [6]. If a model simply guessed '0' for every data point, it would perform quite well by this metric due to the overwhelming ratio of negatives to positives. For an anomaly detection problem such as this, other metrics have to be considered, such as the Precision (percentage of correctly identified positives out of all predicted positives), Recall score (percentage of actual positives found by our model), and F1 score, which combines the Precision and Recall score by taking their harmonic mean [7]. We can also look at the Receiver-Operator Characteristic (ROC) curve, which shows the balance between the True Positive Rate and the False Positive Rate for our model.

The Kaggle competition uses the Normalized Gini Coefficient to evaluate submissions. The Gini Coefficient can be obtained from the area under the ROC curve [8]:

$$\text{Gini} = 2 \times \text{AUC} - 1.$$

The final "late submission" to the Kaggle competition achieved a score of 0.2676. All metrics are summarized in Table 2.

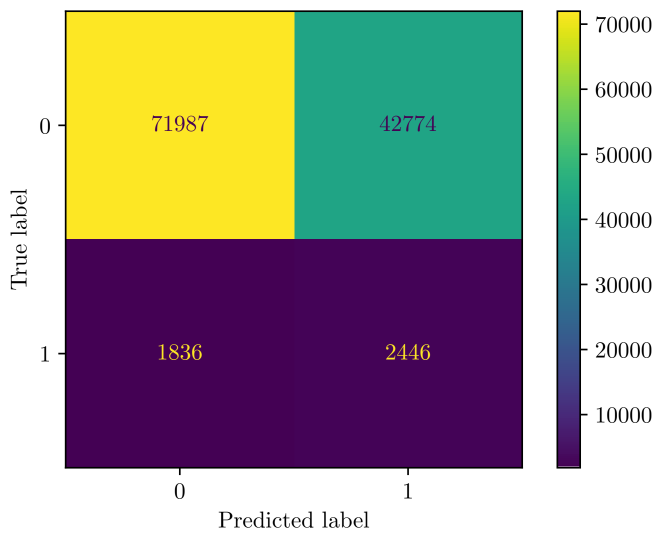


Figure 4: Confusion matrix for validation set

Metric	Score
Accuracy	0.6206
Precision	0.05516
Recall	0.5731
F1	0.1006
Gini	0.2676

Table 2: Summary of metrics on validation set

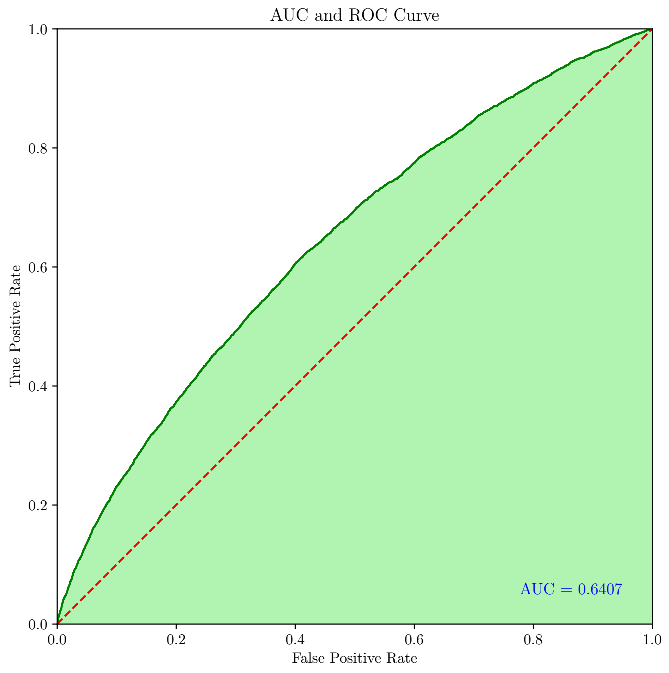
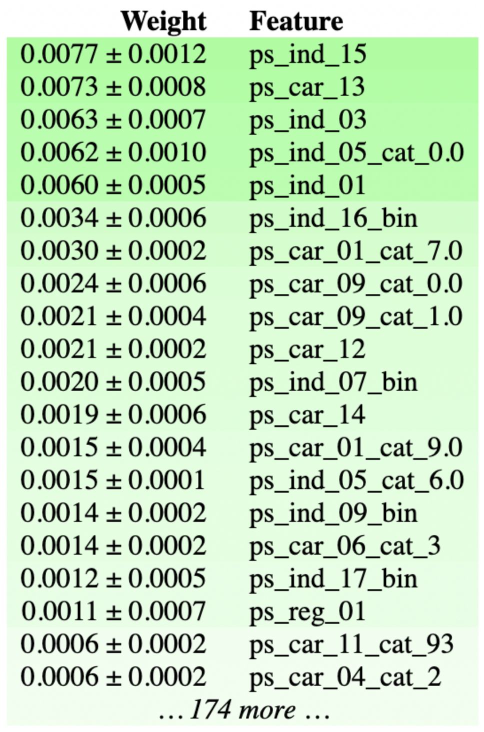


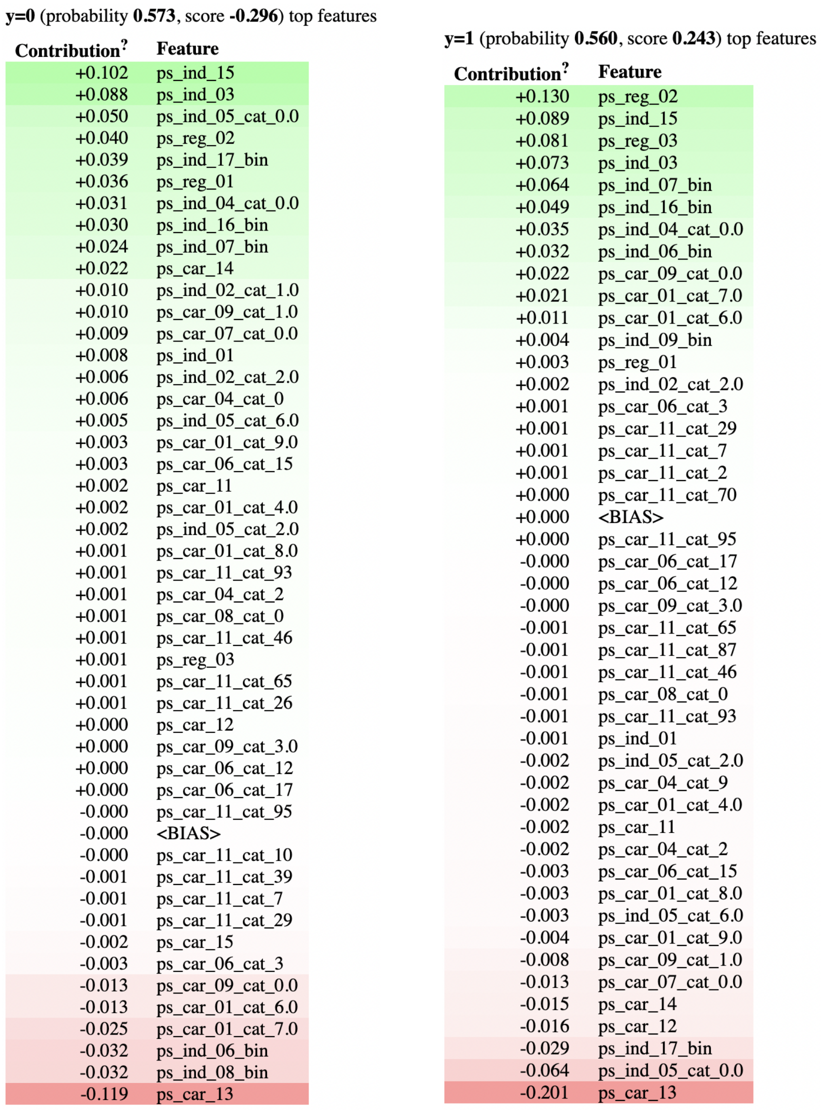
Figure 5: AUC and ROC curve for validation set

The results in Table 2 show how misleading the accuracy is for this dataset. Because the precision is so low (many negatives were incorrectly identified as positive), the F1 score suffers, but is a more helpful metric in this regard. For the sake of the competition, the Gini Coefficient was used for model validation and hyperparameter tuning. Although limited by time for this project, further study into various methods of subsampling and oversampling as well as cross-validation could increase model performance.



Weight	Feature
0.0077 ± 0.0012	ps_ind_15
0.0073 ± 0.0008	ps_car_13
0.0063 ± 0.0007	ps_ind_03
0.0062 ± 0.0010	ps_ind_05_cat_0.0
0.0060 ± 0.0005	ps_ind_01
0.0034 ± 0.0006	ps_ind_16_bin
0.0030 ± 0.0002	ps_car_01_cat_7.0
0.0024 ± 0.0006	ps_car_09_cat_0.0
0.0021 ± 0.0004	ps_car_09_cat_1.0
0.0021 ± 0.0002	ps_car_12
0.0020 ± 0.0005	ps_ind_07_bin
0.0019 ± 0.0006	ps_car_14
0.0015 ± 0.0004	ps_car_01_cat_9.0
0.0015 ± 0.0001	ps_ind_05_cat_6.0
0.0014 ± 0.0002	ps_ind_09_bin
0.0014 ± 0.0002	ps_car_06_cat_3
0.0012 ± 0.0005	ps_ind_17_bin
0.0011 ± 0.0007	ps_reg_01
0.0006 ± 0.0002	ps_car_11_cat_93
0.0006 ± 0.0002	ps_car_04_cat_2
… 174 more …	

Figure 6: Feature importance for validation set



y=0 (probability **0.573**, score **-0.296**) top features

Contribution?	Feature
+0.102	ps_ind_15
+0.088	ps_ind_03
+0.050	ps_ind_05_cat_0.0
+0.040	ps_reg_02
+0.039	ps_ind_17_bin
+0.036	ps_reg_01
+0.031	ps_ind_04_cat_0.0
+0.030	ps_ind_16_bin
+0.024	ps_ind_07_bin
+0.022	ps_car_14
+0.010	ps_ind_02_cat_1.0
+0.010	ps_car_09_cat_1.0
+0.009	ps_car_07_cat_0.0
+0.008	ps_ind_01
+0.006	ps_ind_02_cat_2.0
+0.006	ps_car_04_cat_0
+0.005	ps_ind_05_cat_6.0
+0.003	ps_car_01_cat_9.0
+0.003	ps_car_06_cat_15
+0.002	ps_car_11
+0.002	ps_car_01_cat_4.0
+0.002	ps_ind_05_cat_2.0
+0.001	ps_car_01_cat_8.0
+0.001	ps_car_11_cat_93
+0.001	ps_car_04_cat_2
+0.001	ps_car_08_cat_0
+0.001	ps_car_11_cat_46
+0.001	ps_reg_03
+0.001	ps_car_11_cat_65
+0.001	ps_car_11_cat_26
+0.000	ps_car_12
+0.000	ps_car_09_cat_3.0
+0.000	ps_car_06_cat_12
+0.000	ps_car_06_cat_17
-0.000	ps_car_11_cat_95
-0.000	<BIAS>
-0.000	ps_car_11_cat_10
-0.001	ps_car_11_cat_39
-0.001	ps_car_11_cat_7
-0.001	ps_car_11_cat_29
-0.002	ps_car_15
-0.003	ps_car_06_cat_3
-0.013	ps_car_09_cat_0.0
-0.013	ps_car_01_cat_6.0
-0.025	ps_car_01_cat_7.0
-0.032	ps_ind_06_bin
-0.032	ps_ind_08_bin
-0.119	ps_car_13

(a) Negative example

y=1 (probability **0.560**, score **0.243**) top features

Contribution?	Feature
+0.130	ps_reg_02
+0.089	ps_ind_15
+0.081	ps_reg_03
+0.073	ps_ind_03
+0.064	ps_ind_07_bin
+0.049	ps_ind_16_bin
+0.035	ps_ind_04_cat_0.0
+0.032	ps_ind_06_bin
+0.022	ps_car_09_cat_0.0
+0.021	ps_car_01_cat_7.0
+0.011	ps_car_01_cat_6.0
+0.004	ps_ind_09_bin
+0.003	ps_reg_01
+0.002	ps_ind_02_cat_2.0
+0.001	ps_car_06_cat_3
+0.001	ps_car_11_cat_29
+0.001	ps_car_11_cat_7
+0.001	ps_car_11_cat_2
+0.000	ps_car_11_cat_70
+0.000	<BIAS>
+0.000	ps_car_11_cat_95
-0.000	ps_car_06_cat_17
-0.000	ps_car_06_cat_12
-0.000	ps_car_09_cat_3.0
-0.001	ps_car_11_cat_65
-0.001	ps_car_11_cat_87
-0.001	ps_car_11_cat_46
-0.001	ps_car_08_cat_0
-0.001	ps_car_11_cat_93
-0.001	ps_ind_01
-0.002	ps_ind_05_cat_2.0
-0.002	ps_car_04_cat_9
-0.002	ps_car_01_cat_4.0
-0.002	ps_car_11
-0.002	ps_car_04_cat_2
-0.003	ps_car_06_cat_15
-0.003	ps_car_01_cat_8.0
-0.003	ps_ind_05_cat_6.0
-0.004	ps_car_01_cat_9.0
-0.008	ps_car_09_cat_1.0
-0.013	ps_car_07_cat_0.0
-0.015	ps_car_14
-0.016	ps_car_12
-0.029	ps_ind_17_bin
-0.064	ps_ind_05_cat_0.0
-0.201	ps_car_13

(b) Positive example

Figure 7: Prediction results for two examples

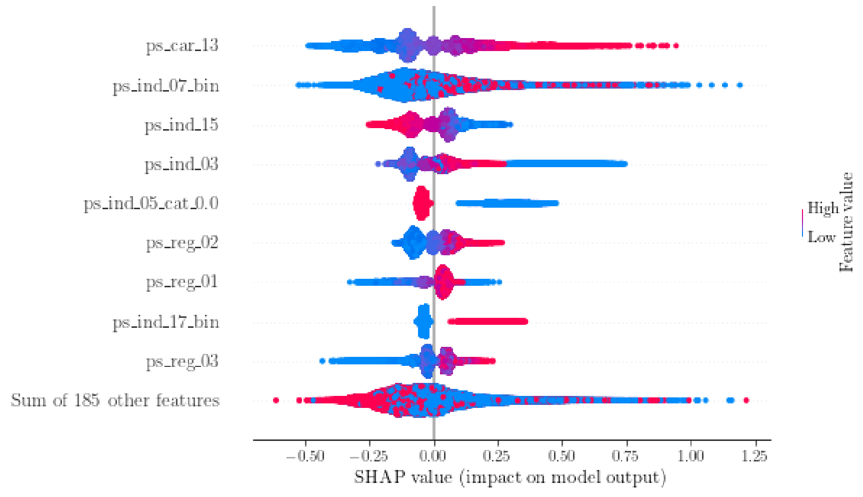


Figure 8: Beeswarm SHAP plot over all predictions

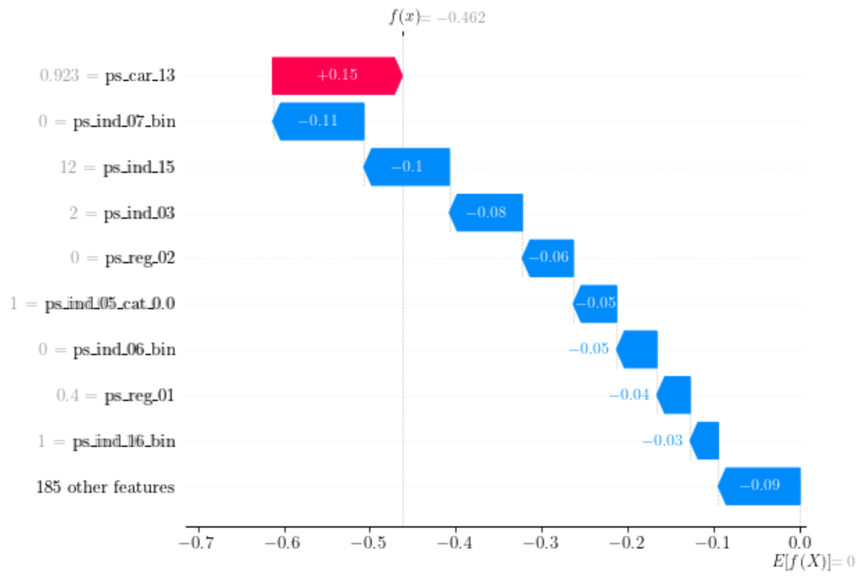


Figure 9: SHAP plot for a negative prediction

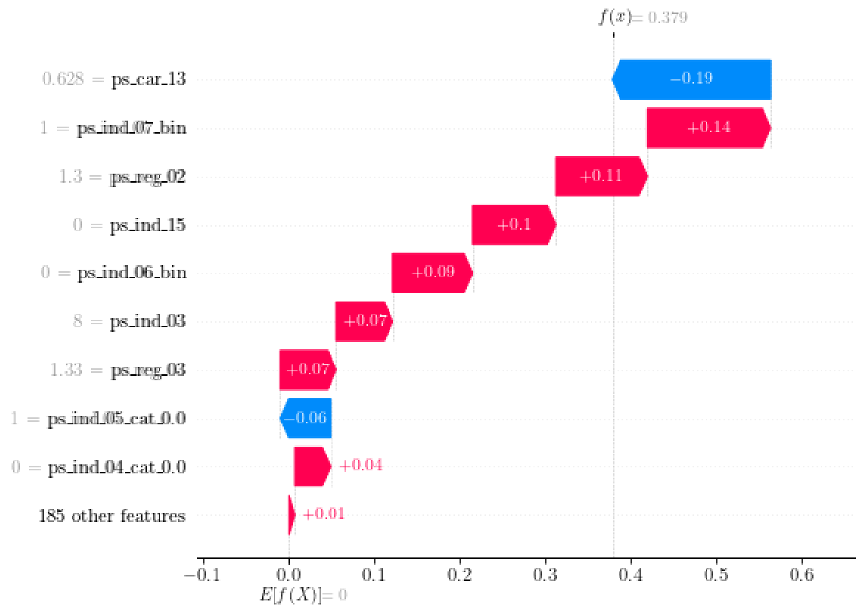


Figure 10: SHAP plot for a positive prediction

Figures 6, 7, 8, 9, and 10 are all different methods for looking into why the model made certain predictions on the validation set. Figure 6 indicates that the most importance features using permutation importance are `ps_ind_15`, `ps_car_13`, `ps_ind_03`, `ps_ind_05_cat_0.0` (one of the one-hot-encoded categorical variables), and `ps_ind_01` [9]. Figure 7 shows the feature importance for two difference predictions; (a) shows a negative prediction, and (b) shows a positive prediction. We can see that the greatest contributor to the negative (correct) prediction was `ps_ind_15` and the greatest contributor to the positive (incorrect) prediction was `ps_reg_02`. This indicates that `ps_reg_02` may be a problematic feature for classification. Figure 8 shows an overall "beeswarm" SHAP (SHapley Additive exPlanations) plot [10]. This plot indicates, for example, that a higher value for `ps_car_13` increases the predicted output and that a lower value for `ps_ind_15` decreases the predicted output. Figures 9 and 10 give another look at the same negative and positive predictions as in Figure 7. These plots show the force of each feature toward pushing the prediction away from the average model output; for this dataset, $\mathbb{E}[f(X)] = 0$. Figure 9 shows that `ps_car_13` works to push the predicted value in the positive direction, but it isn't enough to overcome all of the other features pushing it in the negative direction. Figure 10 shows the value of that same feature working to push the predicted value in the negative direction, but the other features keep it positive, resulting in the incorrect prediction.

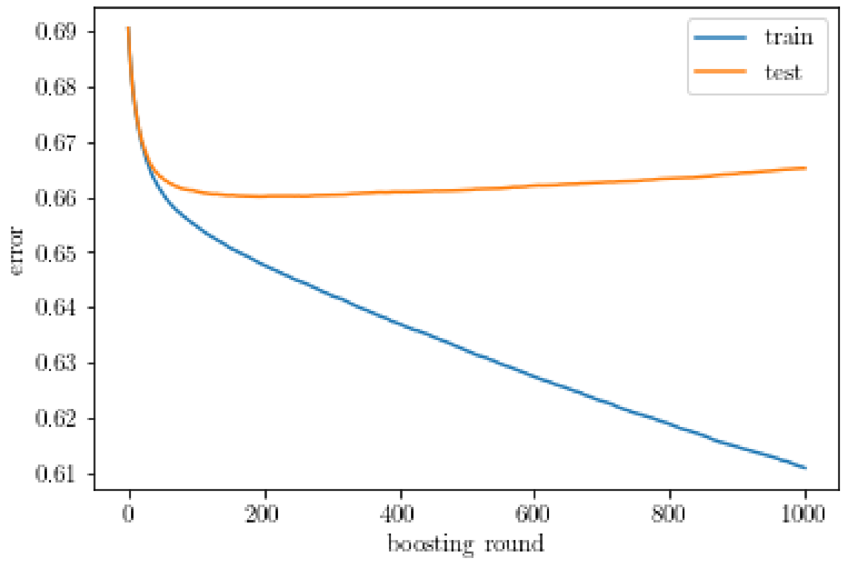


Figure 11: Training vs test error versus 1000 boosting rounds

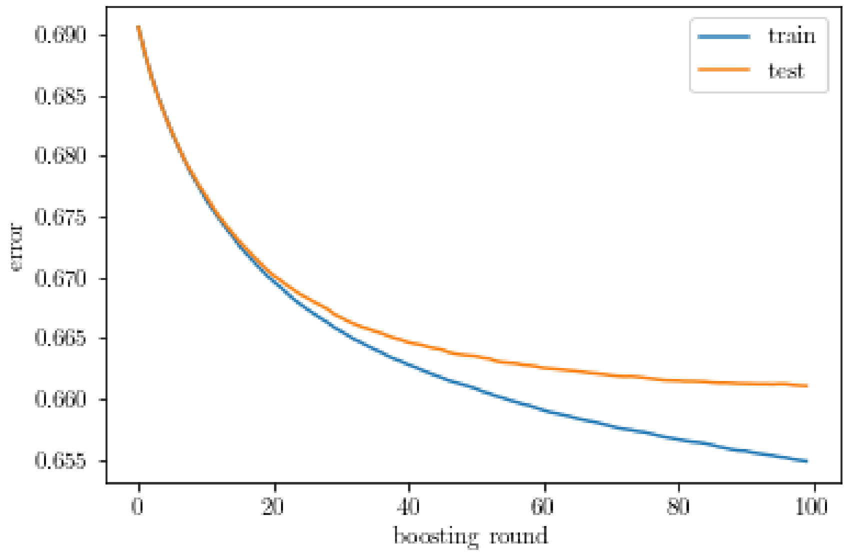


Figure 12: Training vs test error versus 100 boosting rounds

I ended up choosing 100 boosting rounds because of the increase in test error seen after 100-200 boosting rounds.

6 What I learned

I reviewed [11] and [12] to find out how to approach a dataset when its labeling does not permit easy intuition. Although I didn't end up using most of its tools in the final model, [4] was helpful in analyzing feature importance and discovering which features could potentially be dropped. [6] and [3] were valuable in learning about anomaly detection and handling imbalanced datasets. When evaluating the effectiveness of my approach on this project using the techniques outlined in [9], [13] and [10] allowed me to see the effect each feature had on specific predictions. I did not use these model analysis techniques on the first project, but I will definitely come back to these on future projects. The area which I could probably stand to improve upon the most is in hyperparameter tuning. For this problem I found that deviating from XGBoost's default parameters did not lead to an improvement in performance, although that may be due to my approach as well.

References

[1] (2018) Porto seguro's safe driver prediction. [Online]. Available: https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data?select=train.csv

[2] (2021) Xgboost documentation. [Online]. Available: https://xgboost.readthedocs.io/en/latest/index.html

[3] (2018) Undersampling and oversampling imbalanced data. [Online]. Available: https://www.kaggle.com/code/residentmario/undersampling-and-oversampling-imbalanced-data/notebook

[4] (2018) Feature selector: Simple feature selection in python. [Online]. Available: https://github.com/WillKoehrsen/feature-selector/blob/master/Feature%20Selector%20Usage.ipynb

[5] (2022) Spss tutorials: Pearson correlation. [Online]. Available: https://libguides.library.kent.edu/SPSS/PearsonCorr

[6] (2019) Credit fraud —— dealing with imbalanced datasets. [Online]. Available: https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets/notebook

[7] (2022) Scikit learn metrics - f1 score. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[8] E. Schechtman and G. Schechtman, "The relationship between gini terminology and the roc curve," *METRON*, no. 77, p. 171–178, 2019.

[9] (2019) Ml explainability: Deep dive into ml model! [Online]. Available: https://www.kaggle.com/niyamatalmass/ml-explainability-deep-dive-into-the-ml-model

[10] (2022) Shap (shapley additive explanations). [Online]. Available: https://github.com/slundberg/shap

[11] (2020) Exploratory data analysis (eda) using python. [Online]. Available: https://www.youtube.com/watch?v=-o3AxdVcUtQ

[12] (2020) Simple predicting insurance claim. [Online]. Available: https://www.kaggle.com/code/nkemjika/simple-predicting-insurance-claim/notebook

[13] (2017) Eli5 documentation. [Online]. Available: https://eli5.readthedocs.io/en/latest/index.html