Credit Card Fraud Transaction Detection

Module 2

Model Building – Optimization – Performance Evaluation – Tackling Imbalance Class

Group 6 Evan Day

BerkeleyMEng

Recap-Background

1/10 people in US has been a victim of credit card fraud (median amount of $399)

Fraud

Card-Present

- Lost/Stolen
- Counterfeited card
- Card not received

Card-Not-Present

- Data Leak
- Fraud Website

BerkeleyMEng

Recap-Background

Real World data can be CONFIDENTIAL : Privacy Issue

Transaction Data Frame:

TRANSACTION_ID, DATETIME, CUSTOMER_ID, Store_ID, AMOUNT, FRAUD

Customer Data Frame:

CUSTOMER_ID, location_x, location_y, Mean_trans, STD_trans, num_tran_day

Store Data Frame:

Store_ID, Station_location_x, Station_location_y, Mean_trans, STD_trans, num_tran_day

BerkeleyMEng

Feature Engineering of dataset (Improvement on Data set)

- Transaction during night, Transaction during weekend
- RFM Model (Recency, Frequency, Monetary Value) [Véronique, 2015]:
    - a. # of Transactions within a window [1, 7, 30]
    - b. Avg. amount within the window
- Risk Score: Avg. fraudulent transactions that occurred on a store [1, 7, 30]
- …

BerkeleyMEng

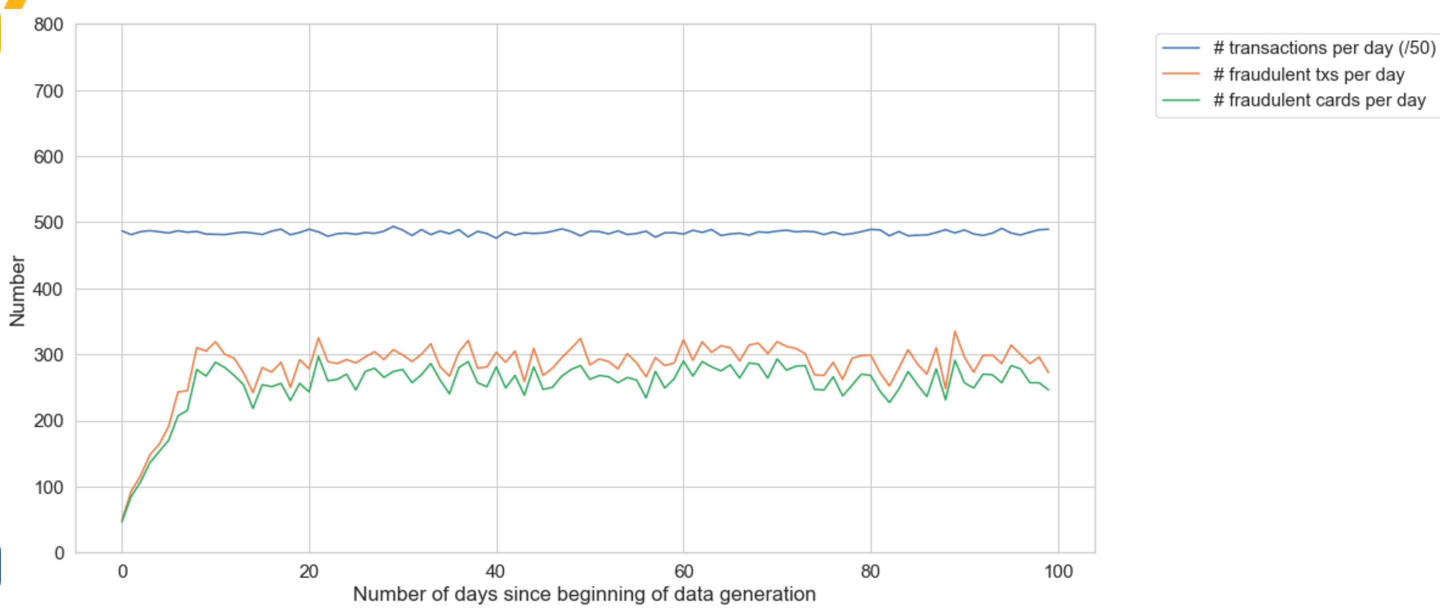
Final Data Set

```
transactions_df
```

[33] Python

	TRANSACTION_ID	DATETIME	CUSTOMER_ID	STORE_ID	AMOUNT	Trans_TIME_SECONDS	Trans_TIME_DAYS	Trans_FRAUD	Trans_DURING_WEEKEND	Trans_DURING_NIGHT	...	CUS
0	0	2022-01-01 00:00:17	5820	2647	17.99	17	0	0	1	1	...	
1	1	2022-01-01 00:00:17	6160	2980	44.48	17	0	0	1	1	...	
2	2	2022-01-01 00:00:30	356	752	86.87	30	0	0	1	1	...	
3	3	2022-01-01 00:00:31	1829	2266	16.65	31	0	0	1	1	...	
4	5	2022-01-01 00:00:31	6820	8046	128.04	31	0	0	1	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	
2421220	2421220	2022-04-10 23:58:31	3491	2534	144.45	8639911	99	0	1	0	...	
2421221	2421221	2022-04-10 23:59:24	4381	7285	80.93	8639964	99	0	1	0	...	
2421222	2421222	2022-04-10 23:59:27	3805	7758	39.65	8639967	99	0	1	0	...	
2421223	2421223	2022-04-10 23:59:38	4491	8719	142.47	8639978	99	0	1	0	...	
2421224	2421224	2022-04-10 23:59:48	9535	3388	38.94	8639988	99	0	1	0	...	

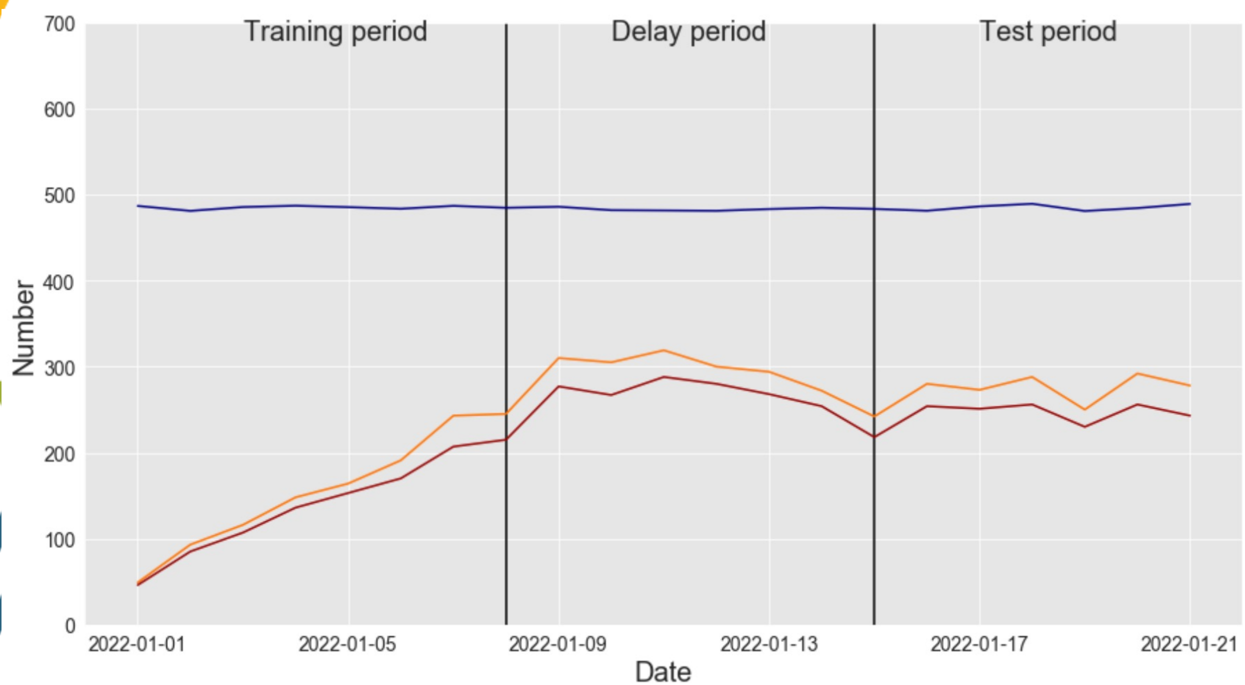
2421225 rows × 22 columns



BerkeleyMEng

How to Predict?

- It is important to note that we chose to run the test set one week after the last trade in the training set. In the context of fraud detection, this period of time that separates the training and test sets is called the ***delay time or feedback delay*** [Andrea, 2017]. It accounts for the fact that in real-world fraud detection systems, the label of a transaction, fraudulent or real, is known only after a customer complaint or as a result of a fraud investigation.



BerkeleyMEng

Model Building

- Decision Trees
- Logistic Regressions
- Random Forest
- XG-Boosting

Performance Metrics

- ROC/AUC:

The ROC curve is used to assess the overall diagnostic performance of a test and to compare the performance of two or more diagnostic tests. It is also used to select an optimal cut-off value for determining the presence or absence of a disease.

- Card Precision Top K (CP@K)

multiple fraudulent transactions from the same card should count as a single correct detection since investigators check all the recent transactions when contacting cardholders.

- Average Precision

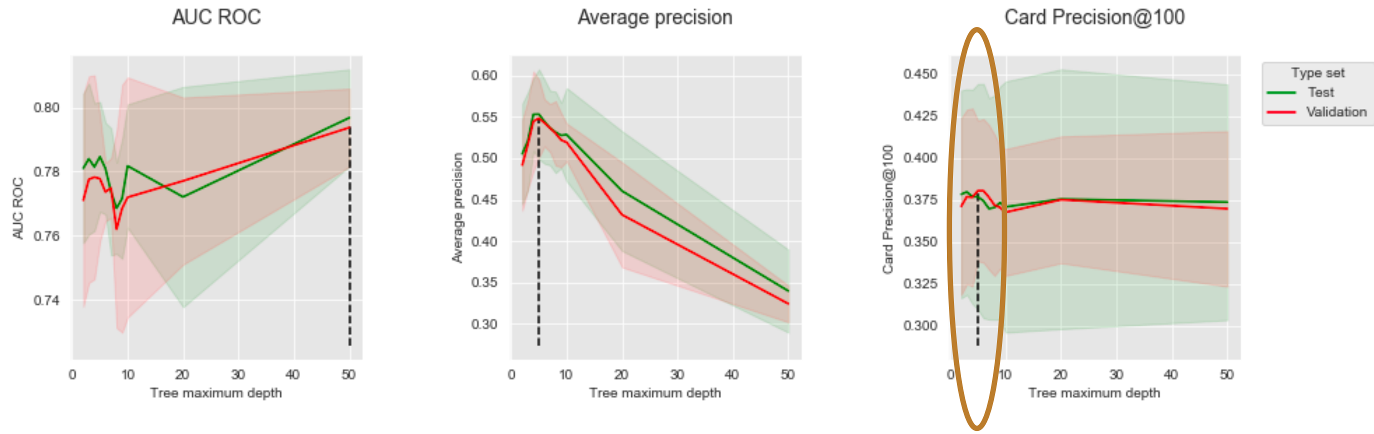
(Why choose these? Instead of traditional Accuracy? Imbalance Class, explained later)

BerkeleyMEng

Model Building

- Decision Trees

	AUC ROC	Average precision	Card Precision@100
Best estimated param	50	5	5
Validation performance	0.794+/-0.01	0.548+/-0.02	0.38+/-0.02
Test performance	0.797+/-0.01	0.553+/-0.03	0.377+/-0.03
Optimal parameters	50	5	3
Optimal test performance	0.797+/-0.01	0.553+/-0.03	0.38+/-0.03

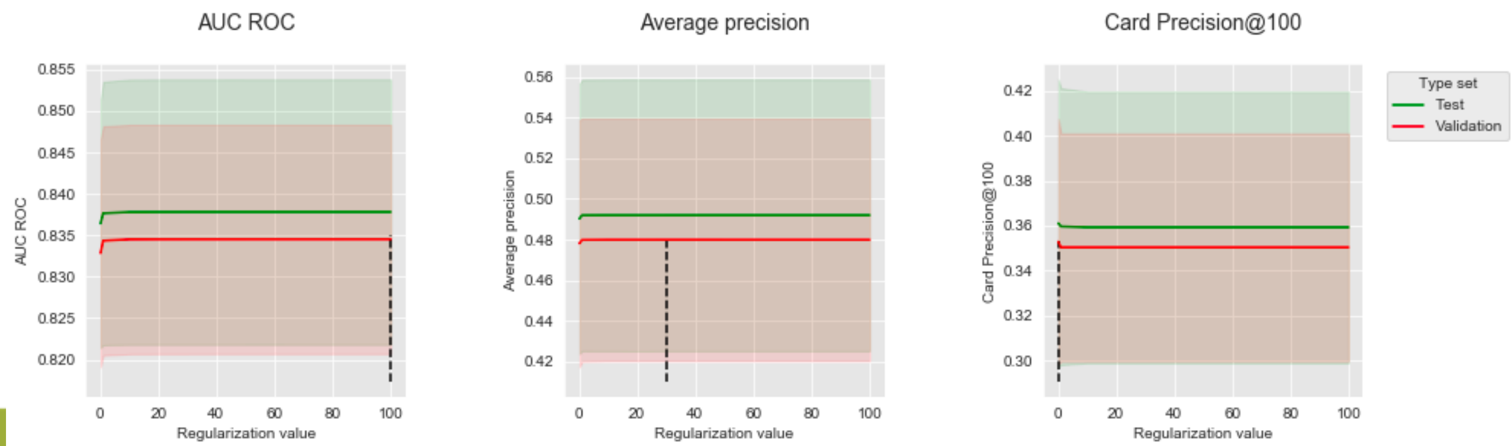


- Maximum depth are different according to different performance metrices
- The best parameters for the validation may not be the optimal parameter for the test set. CP@100.

Model Building

- Logistic Regressions → HP: regularization parameter

	AUC ROC	Average precision	Card Precision@100
Best estimated param	100.0	30.0	0.1
Validation performance	0.835+/-0.01	0.48+/-0.03	0.353+/-0.03
Test performance	0.838+/-0.01	0.492+/-0.03	0.361+/-0.03
Optimal parameters	100.0	100.0	0.1
Optimal test performance	0.838+/-0.01	0.492+/-0.03	0.361+/-0.03

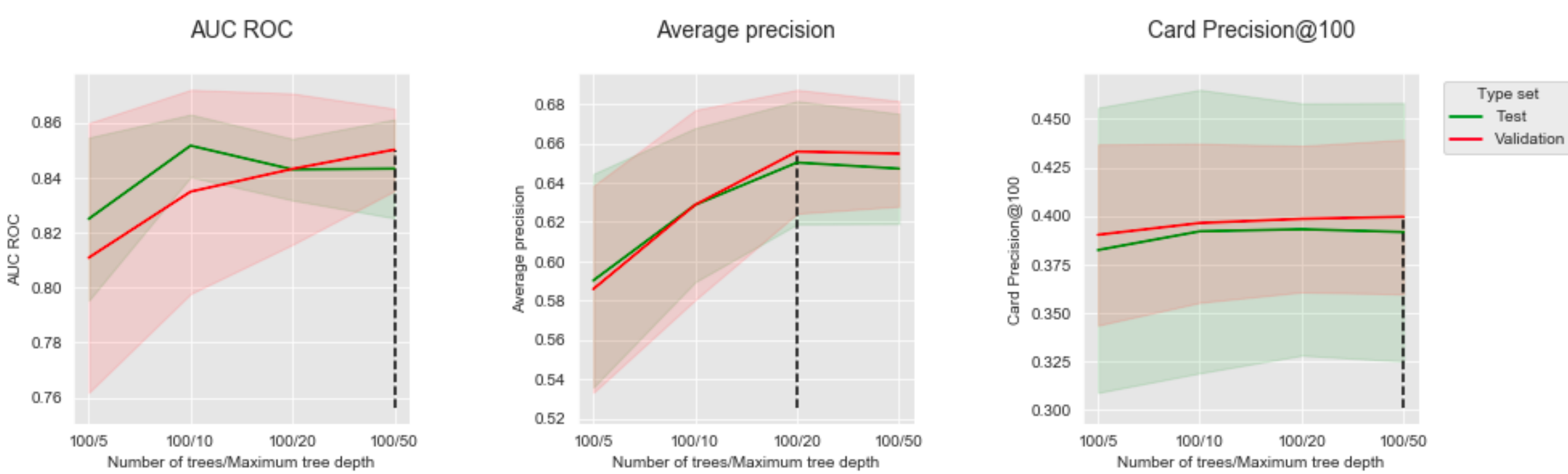


BerkeleyMEng

Model Building

- Random Forest → HP: MAX Depth and # of trees

	AUC ROC	Average precision	Card Precision@100
Best estimated param	100/50	100/20	100/50
Validation performance	0.85+/-0.01	0.656+/-0.02	0.4+/-0.02
Test performance	0.843+/-0.01	0.65+/-0.02	0.392+/-0.03
Optimal parameters	100/10	100/20	100/20
Optimal test performance	0.852+/-0.01	0.65+/-0.02	0.393+/-0.03

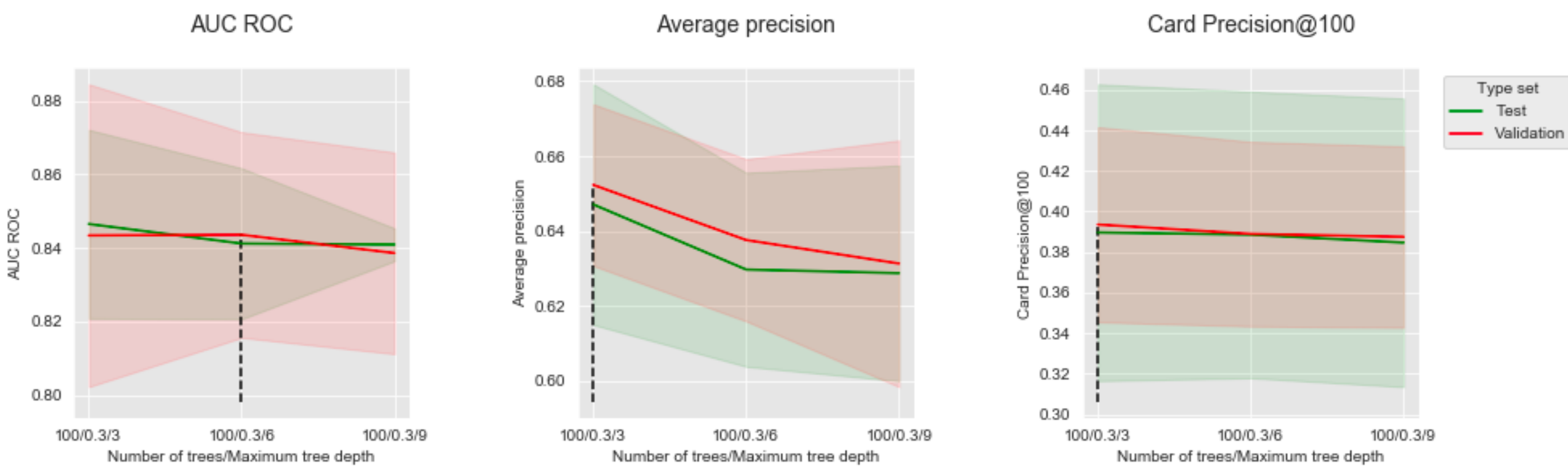


BerkeleyMEng

Model Building

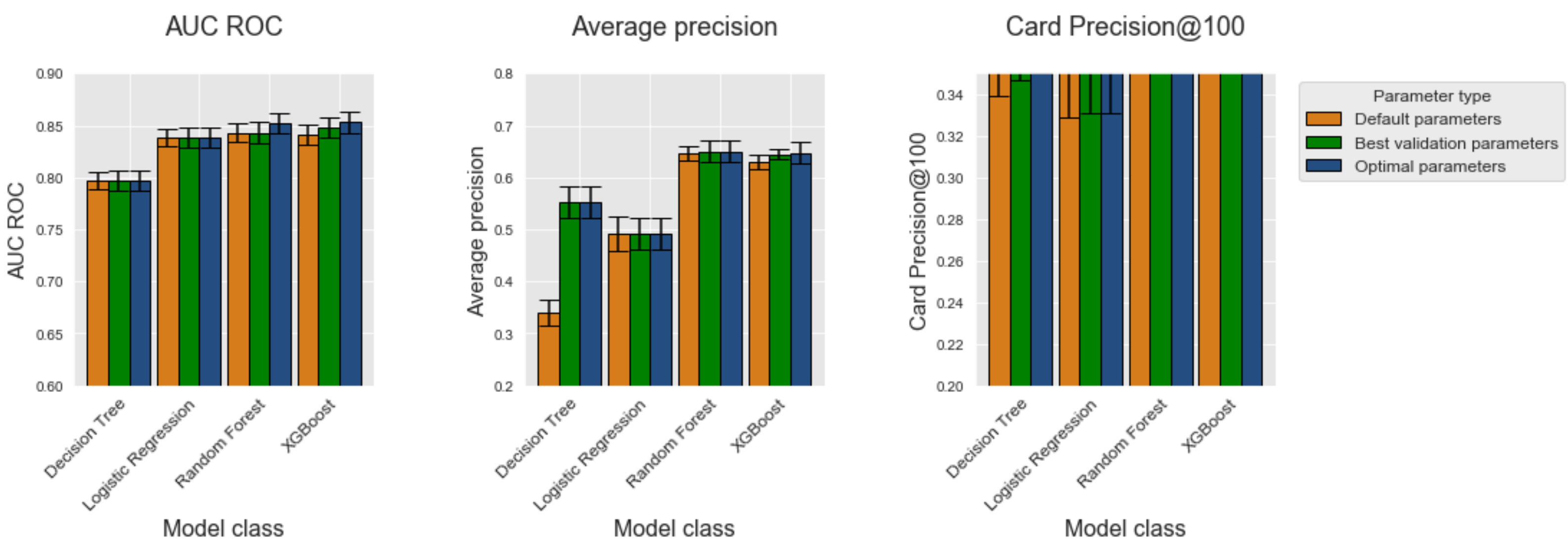
- XG Boosting → HP: MAX Depth, # of trees, learning rate

	AUC ROC	Average precision	Card Precision@100
Best estimated param	100/0.1/6	100/0.1/3	100/0.1/3
Validation performance	0.845+/-0.02	0.652+/-0.01	0.397+/-0.02
Test performance	0.848+/-0.01	0.644+/-0.01	0.389+/-0.03
Optimal parameters	100/0.1/3	100/0.3/3	100/0.3/3
Optimal test performance	0.853+/-0.01	0.647+/-0.02	0.39+/-0.04



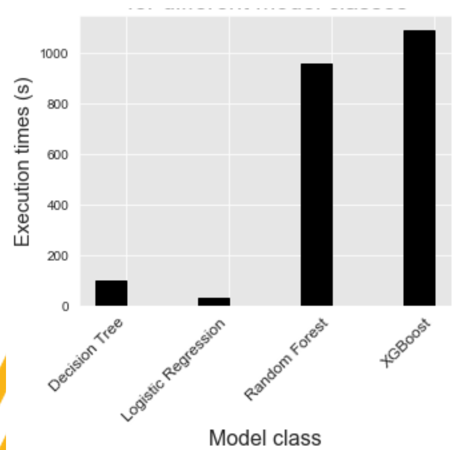
BerkeleyMEng

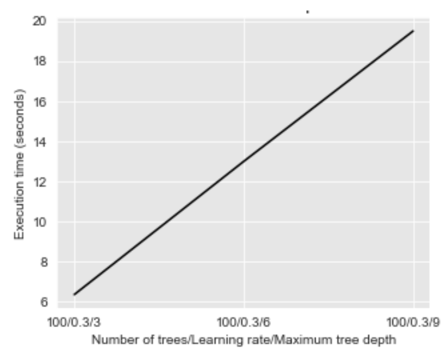
Model Selection:



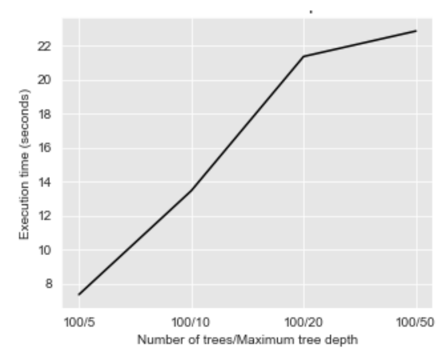
On this simulated dataset, XGBoost performs the best in terms of average accuracy and CP@100, followed by random forest, logistic regression, and finally decision tree. The gap in performance is most pronounced under the average precision metric. In terms of AUC ROC, the performance of logistic regression, random forest and XGBoost is very similar.

Model Selection - Time:





XG-Boosting



RF

It illustrates that model selection for complex models such as random forests or boosting usually requires more computational resources, since they require tuning a higher number of hyperparameters.

Further Optimization – Random Research:

```python
print("Total execution time for XGBoost with grid search: "+str(round(execution_time_boosting,2))+"s")
print("Total execution time for XGBoost with random search: "+str(round(execution_time_boosting_random,2))+"s")
```

[48] ✓ 0.0s Python

```
Total execution time for XGBoost with grid search: 1533.7s
Total execution time for XGBoost with random search: 1212.4s
```

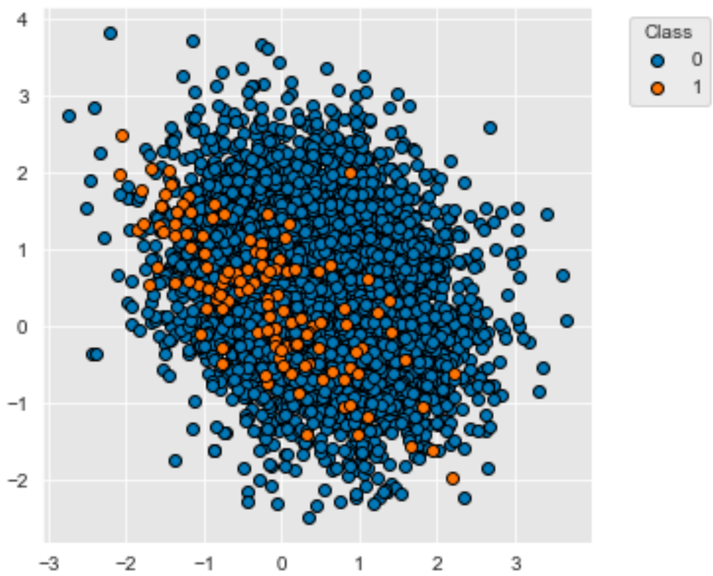
Imbalance Class Issue:

Only 3% fraud, the model can predict all the transaction are all the normal transaction, but we can still achieve an accuracy =97%! As a result, we should consider how to resolve the imbalance problem:

Using the **Cost Sensitive Learning: Used if the misclassification cost is not equal.**

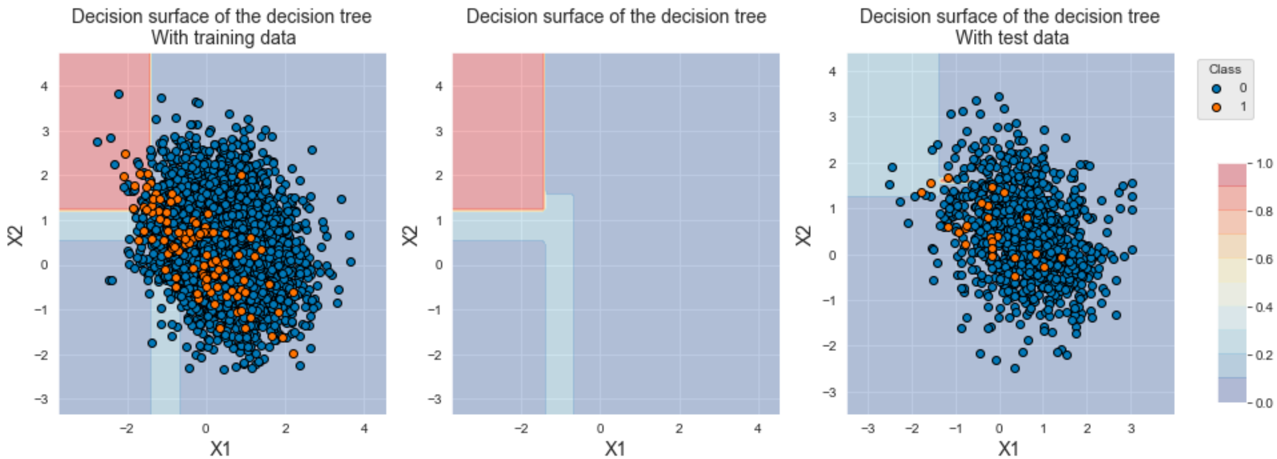
For visualization purpose, we retrieve the Top-2 PCA to represent the X-variables.

We can see from the plot that the distribution of two classes are overlapping, however, we can see some patterns here.

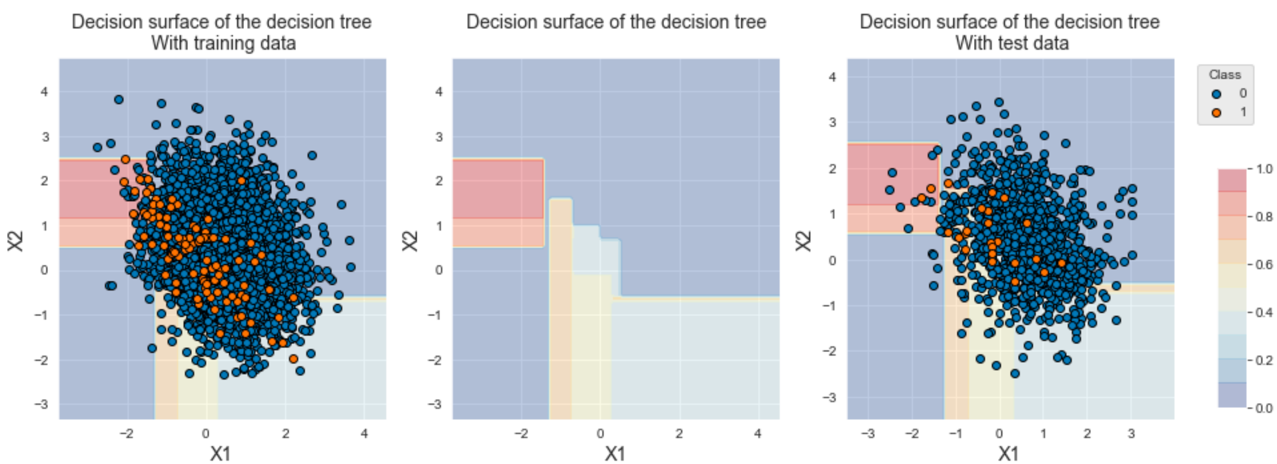


Imbalance Class Issue:

Normal Decision Tree:

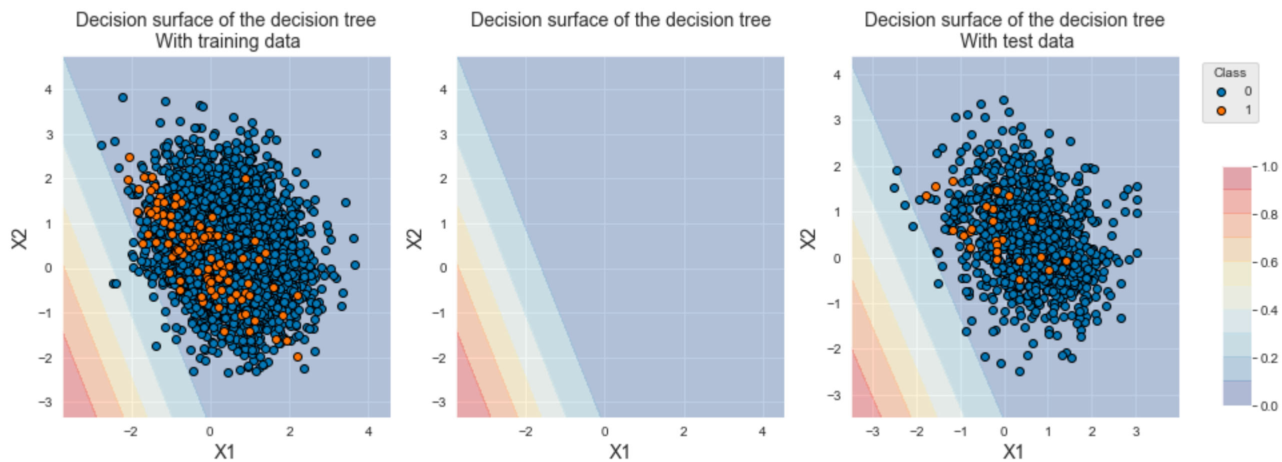


Adding the Cost-Sensitive learning

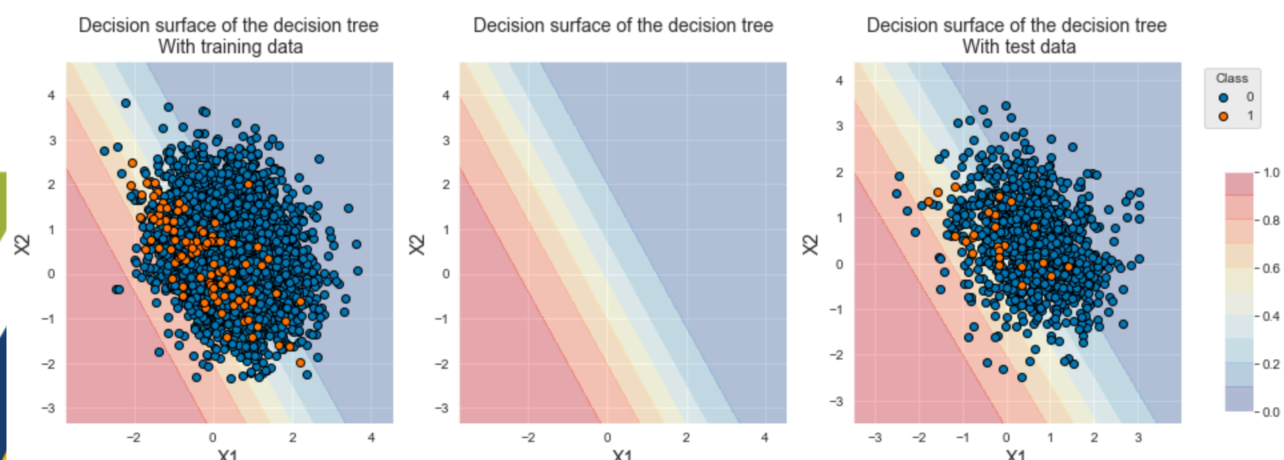


Imbalance Class Issue:

Normal Logistic Regression:



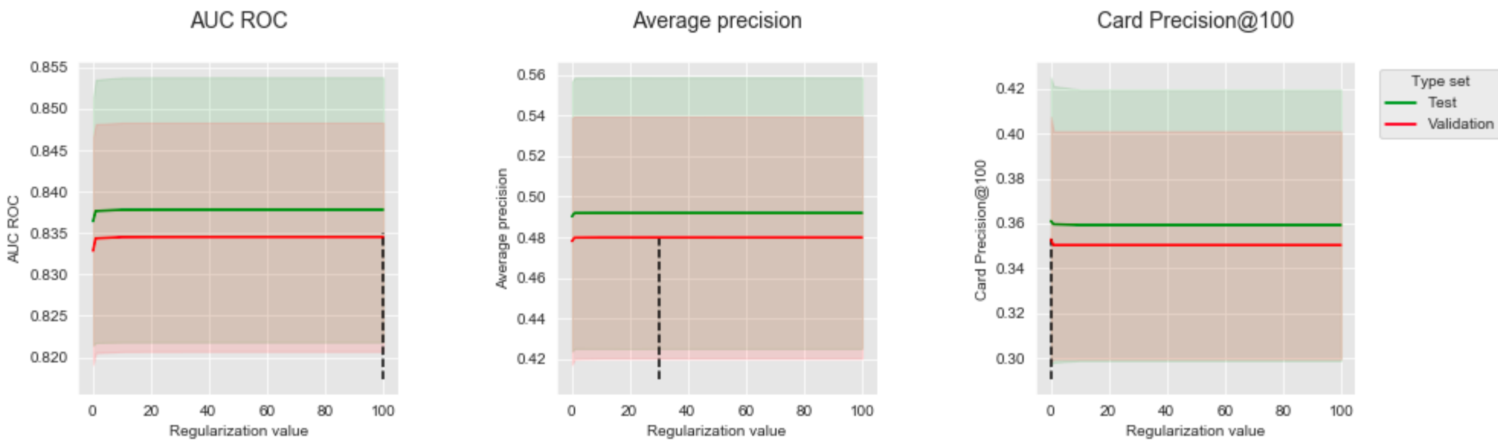
Adding the Cost-Sensitive learning



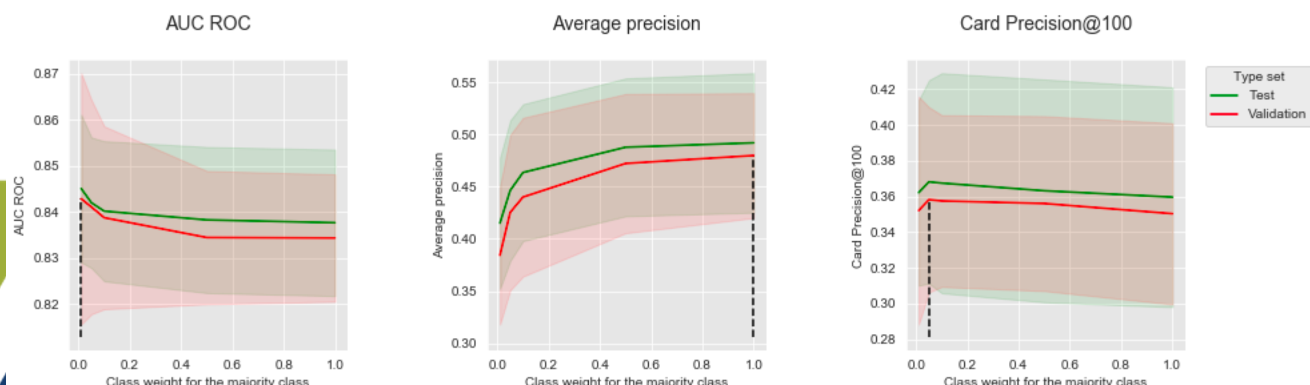
BerkeleyMEng

Improvement from Logistic Regression

Normal Logistic Regression:



Adding the Cost-Sensitive learning



BerkeleyMEng

Limitations

- Cost-sensitive learning is ambiguous, we cannot accurately define the “Cost”.
- Machine learning may not be the best choice because there are limited hyper-parameters we can control
- Data Limitation: Pure simulated data have many drawbacks
- …

Further Action

- Resampling methods e.g. SMOTE
- Adding deep learning in our further work, building layers to feed forward
- Instead of focusing on the overall performance, we are trying to focus on fraud class accuracy, because it is cheap to say a normal transaction is fraud.
- …

Q&A

Thanks

BerkeleyMEng

References

Charles X Ling and Victor S Sheng. Cost-sensitive learning and the class imbalance problem. *Encyclopedia of machine learning*, 2011:231–235, 2008.

Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Apate: a novel approach for automated credit card transaction fraud detection using network-based extensions. Decision Support Systems, 75:38–48, 2015.

Machine Learning Group: Reproducible Machine Learning for Credit Card Fraud detection - Practical handbook