Fraud? It’s more likely than you think!

CS3244 Team 05.

Ethan Wong Goon Hong, Nikita Goh Mei Xian,
Shawn Tan Jinhui, Ye JiaYing, Liang Zhengxin

01

Introduction

Background & Motivation

Dataset: Fraud Detection in Electricity and Gas Consumption Challenge

The Tunisian Company of Electricity and Gas (STEG) is a public and a non-administrative company responsible for delivering electricity and gas across Tunisia. The company suffered tremendous losses in the order of 200 million Tunisian Dinars (~86 mil SGD) due to fraudulent manipulations of meters by consumers.

Motivation:

- Electricity fraud is a significant issue in developing countries
- E.g. 20% of total electricity production in India is lost due to electricity fraud and theft
- Obstruct the economic growth of these developing countries - lose $58.7 billion per year due to electricity theft
- By eliminating electricity and gas fraud, economic growth can improve and in turn reduce poverty and increase quality of life.

Problem Statement

There is a need to detect the patterns that are associated with fraud in Tunisia. However, there are a lot of challenges that complicate the detection of electricity and gas fraud. Therefore, we aim to use Machine Learning algorithms to detect and identify clients committing fraud against the Tunisia company of Electricity and Gas.

02

Dataset Analysis

Overview of Dataset

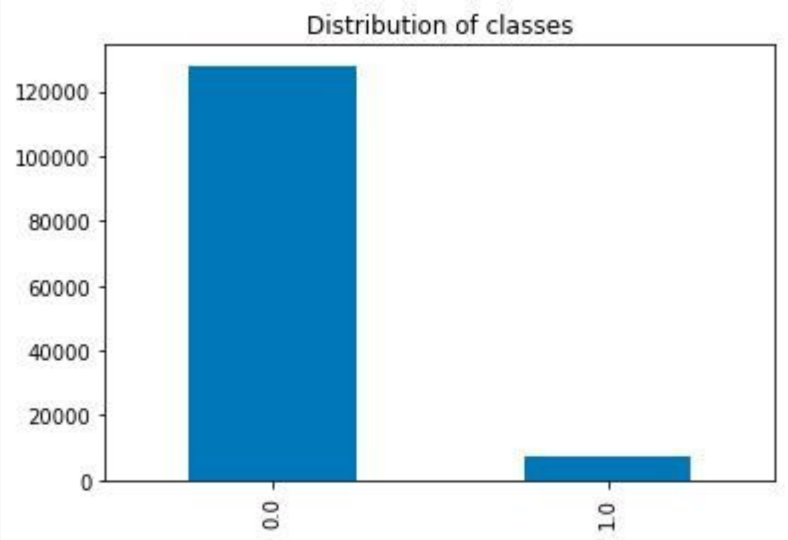
Data was split across two csv files:

client:
Each instance of the client csv file was unique to each client and contained relevant information with features such as client_id, region, creation_date and more importantly target (which classified the instance according to fraud or no fraud committed).

invoice:
In the invoice csv file, each client could have multiple instances as identified by client_id and they contained invoice information with features such as invoice_date, counter_number, consumption_level, counter_type, old_index and new_index.

Problems With the Dataset

Heavily Imbalanced Dataset



Unclear/Inconsistent Features

Multiple invoices share a client, labels associated with clients
Consumption_level divided across different levels

Lacked Proper Test Set

Original dataset designed for competition
Had to improvise and section out training set

Feature Engineering

43 Features!

client_id, district, client_catg, region, region_group, creation_date_day, creation_date_month, creation_date_year, no_months_as_client, services_consumed, target, months_of_service, number_of_counter, number_of_instances, number_of_person_counting, min_reading_remark, max_reading_remark, mean_reading_remark, mean_difference_index, sum_difference_index, min_difference_index, max_difference_index, min_counter_statue, max_counter_statue, mean_counter_statue, mean_diff_counter_cofficient, sum_counter_coefficient, mean_counter_coefficient, min_counter_coefficient, max_counter_coefficient, sum_total_consumption, mean_total_consumption, min_total_consumption, max_total_consumption, tally_check_true, tally_check_false, sum_tally_value, min_tally_value, max_tally_value, mean_tally_value, counter_type, last_year, last_month, last_day, last_day_is_weekday

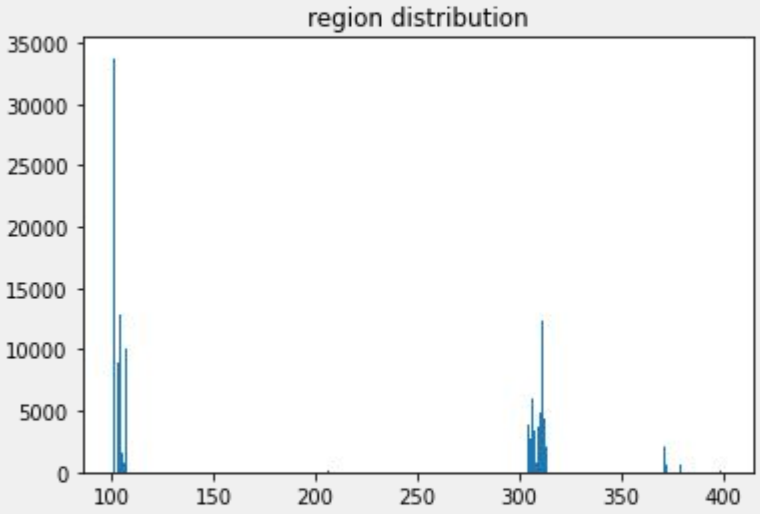
Key Additions

- client_id: To group all invoices done by the same client
- min/max/mean/sum of various attributes (under the same client)
- Last year/month/day
- classified region into groups

Notes

- Fraud -> Positive, No Fraud -> Negative
- Punish False Negatives over False Positives

Feature Engineering



Regions seem to cluster into 3 groups

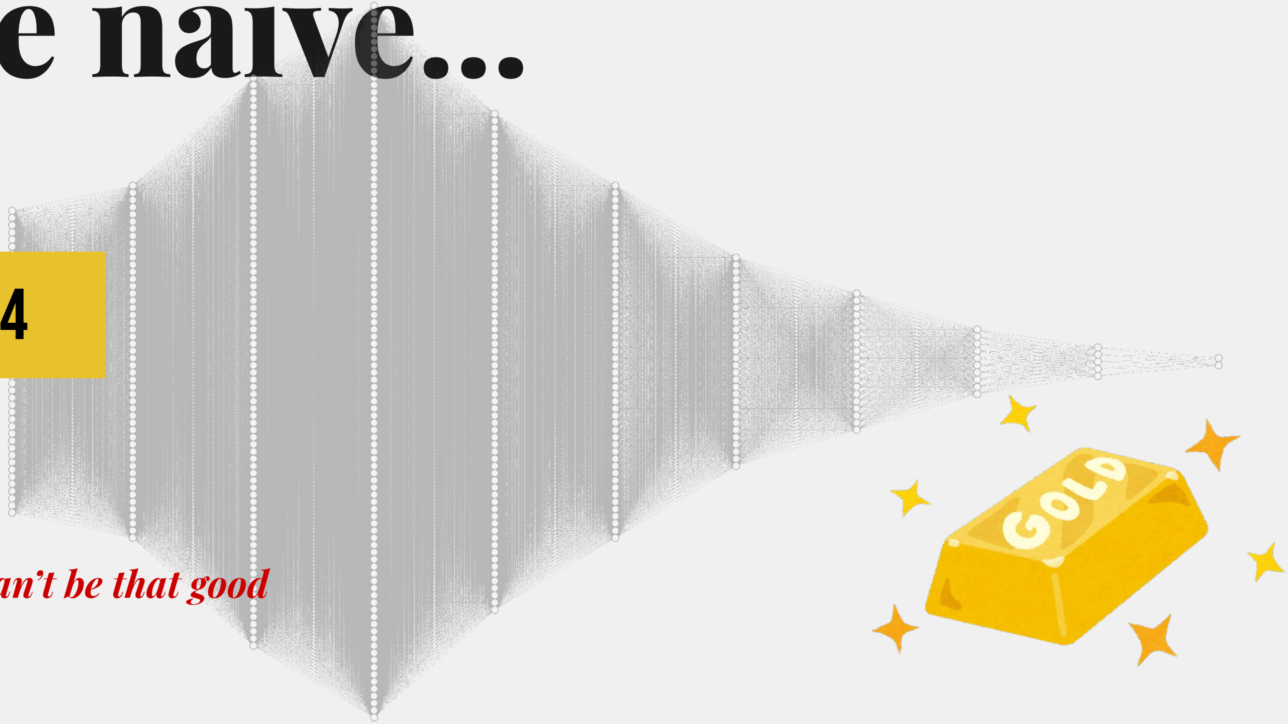
Creating a region type feature that discretizes the value of region into 3 values helps reduce the dimensionality of data.

03

Learning Models

kNN, SVM, Decision Tree, XGBoost, and MLP

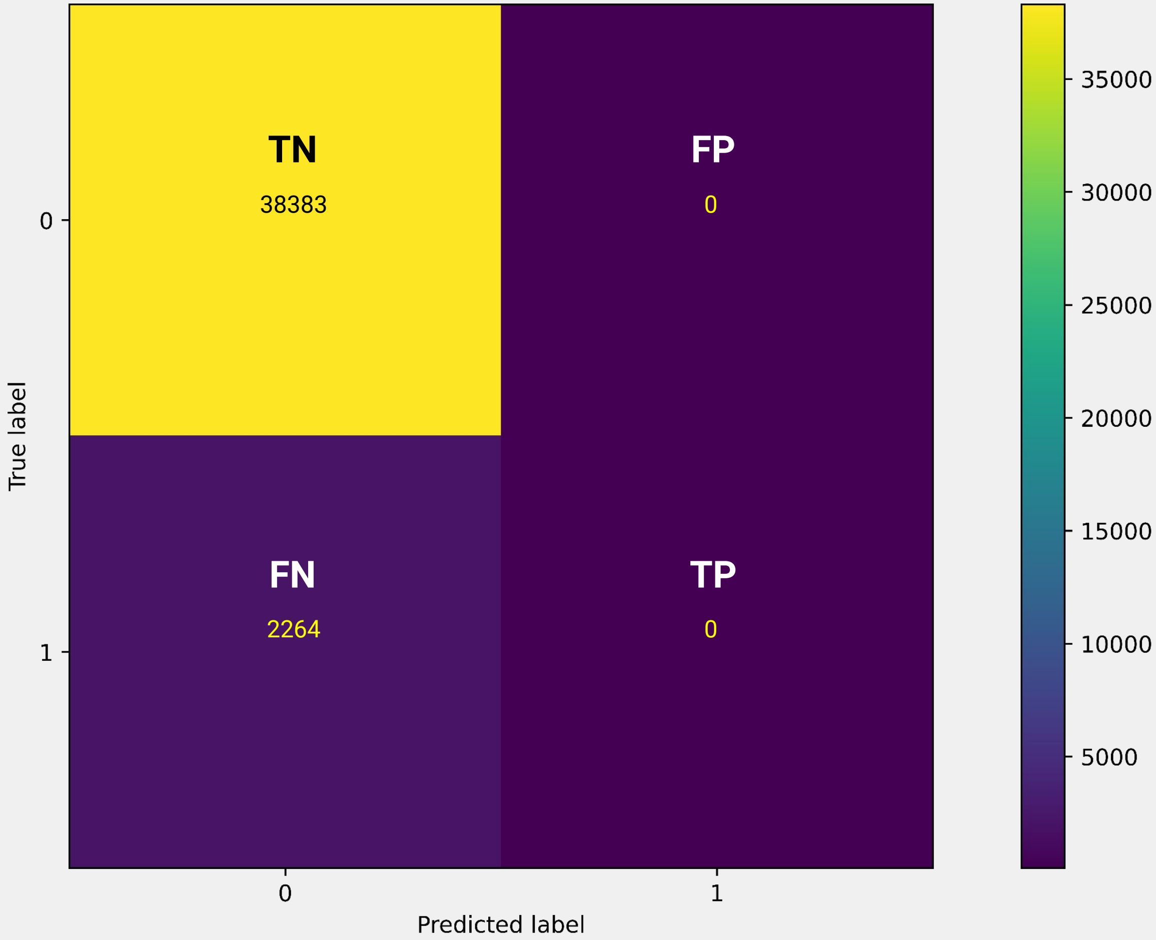
In the beginning, we were naive...



ACCURACY: ~0.944

But, we soon realised it can't be that good

LeNail, (2019). NN-SVG: Publication-Ready Neural Network Architecture Schematics. Journal of Open Source Software, 4(33), 747, https://doi.org/10.21105/joss.00747


TN
38383
FP
0
FN
2264
TP
0
0
1
True label
0
1
Predicted label
35000
30000
25000
20000
15000
10000
5000

K-Nearest Neighbors

Initial Test

- Default design (k = 5, p = 2)
- 94% accuracy, many false negatives
- Biased towards predicting negative as dataset was unbalanced

Improvements Made

- Used SMOTE to balance data
- Increase k to reduce bias
- Too high -> Reduced accuracy
- Drastically decreased false negatives (~2k -> ~1k)

Final Evaluation

- Only slight increase in f2 scores (only reached ~0.25)
- Found an James Thorn's article to help visualise curse of dimensionality
- Higher dimensions causes our concept of "proximity" to lose meaning

The Pros & Cons of kNN

Pros

- Uses very simple concepts
- O(1) training time, very fast

Cons

- Suffers heavily from curse of dimensionality
- O(training data * test data) testing time (each run took ~20 mins)
- Difficult to optimise other than brute forcing different hyper-parameters

Multi-Layer Perceptron

Pros

- It works on complex datasets
- A flexible model that can be used for both Classification or Regression

Cons

- While “adam” was used, the training time was long.
- Difficult to determine the actual number of hidden layers and nodes required.

A NEURAL NETWORK THAT CLASSIFIES BASED ON DEFINED HIDDEN LAYERS

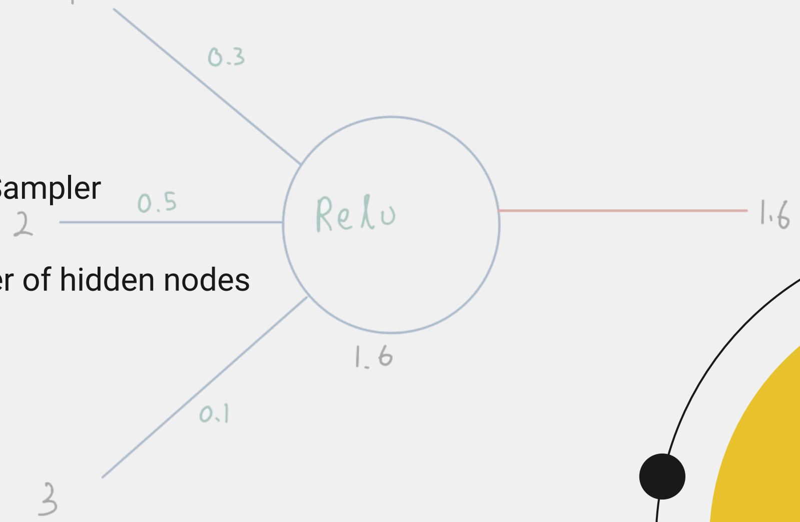
Training & Evaluation

Baseline:

- `Hidden Layers: (50, 80, 100, 70, 50, 30, 20, 10, 5)`
- `Accuracy = 0.944`

Improvements made:

- Oversampled with sklearn's RandomOverSampler
- Reduced the number of Hidden Layers
- Tried various models with different number of hidden nodes



Results:

`Hidden Layers: (14,13)`

- **F2 Score - 0.417**
- **Precision - 0.168**
- **Recall - 0.663**
- **Accuracy - 0.798**

Support Vector Machines

Baseline:

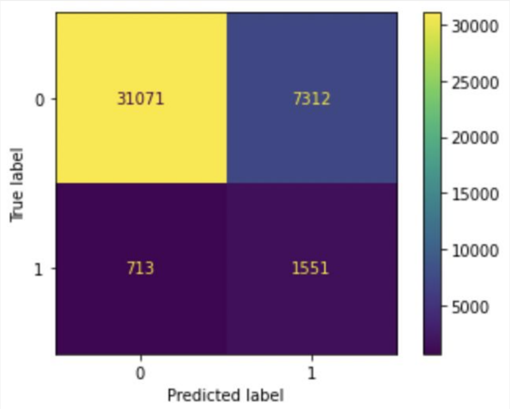
- Hard margin (C=1e10)
- L2 Regularisation (Standardized features)

Improvements made:

- Soft margin (C=1.0)
- L1 Regularisation
- Stochastic Gradient Descent (α=0.01)
- Hyperparameter tuning
  (Different C values, learning rate schedule)

Results:

- Accuracy = 0.802
- Precision = 0.177
- Recall = 0.699
- F2 score = 0.439



The Pros & Cons of SVM

Pros

- Effective in high-dimensional spaces
- Works well when there is a clear margin of separation between classes

Cons

- Not suitable for large data sets due to the high training time
- Does not work well when there is more noise (overlapping classes)

Decision Tree

- Concludes if an instance is fraud based on the features of the tree and path taken

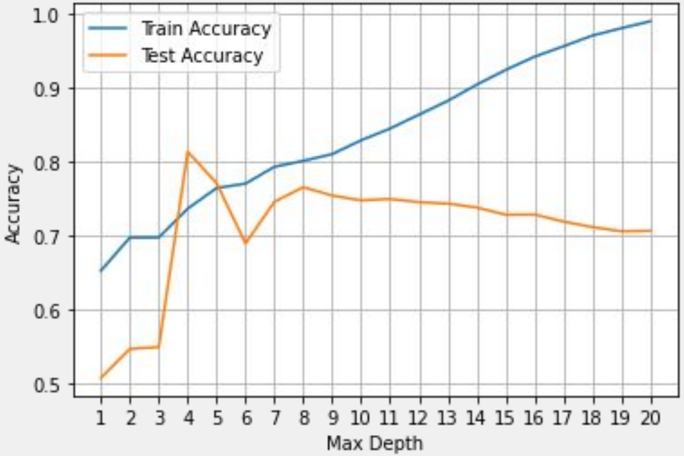
Baseline:

- Default decision tree builder using Gini Coefficient:
  - Training Accuracy: 1.00
  - Test Accuracy: 0.703
  - Overfitting

Training & Evaluation

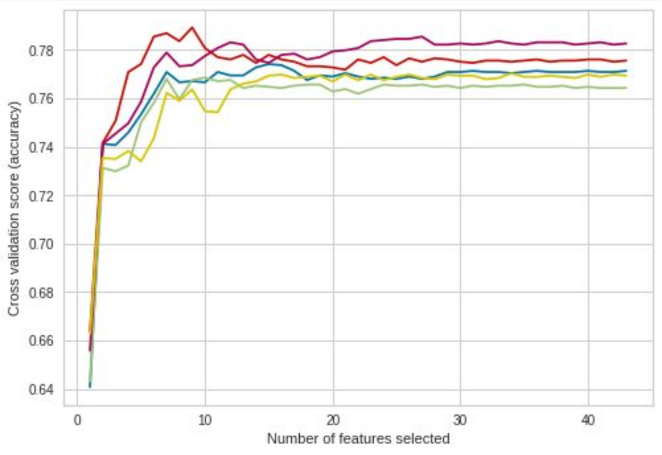
Variation

- Using Entropy Loss as criterion
- Overfitting → Pruning by changing depth of tree, optimal max depth is at 8 before it starts overfitting
- Curse of dimensionality → Using RFE to reduce features to only 7 + 5-fold CV
- More costly to have False Negatives → Increasing weights of Fraud class (performance decreased!!)



Final results

- Test accuracy: 0.762787
- Precision: 0.162982
- Recall: 0.787986
- F2 score: 0.445955



XGBoost

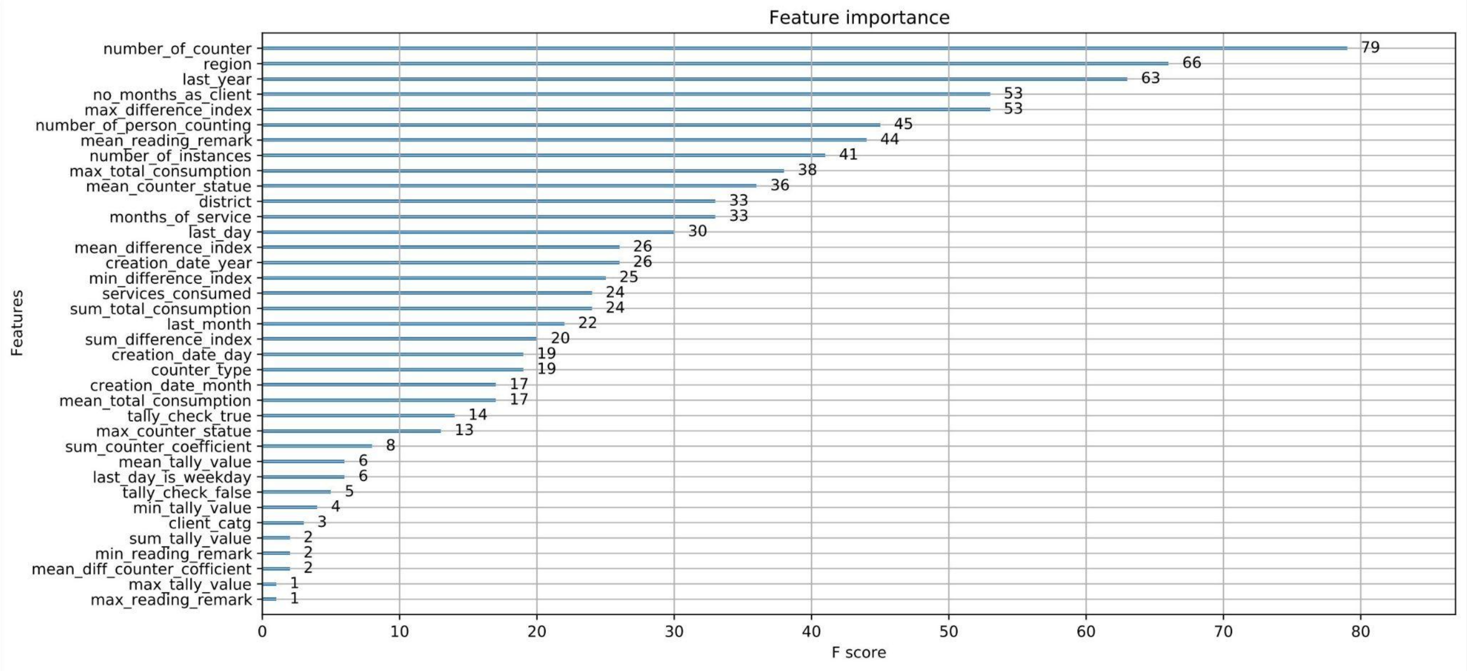
- Is an open source implementation of gradient boosted decision trees that has recently been dominating applied machine learning and competitions for structured or tabular data.
- Presented an interesting opportunity to see how a popular model would perform on our dataset.
- Library functionalities also provided useful data analysis tools.

dmlc XGBoost

https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7

XGBoost

Feature importance chart generated using XGBoost



Improving XGBoost

- Used hyperparameter tuning by iterating through 900 candidates with different hyperparamters e.g:
```
{'colsample_bytree': 0.9796084316530052, 'gamma': 0.05125486399533963, 'learning_rate': 0.3111685461709353, 'max_depth': 5, 'n_estimators': 135, 'subsample': 0.6271348236420687}
```
- Used hyperparameters that gave best performance:

Results:

- Accuracy = 0.843
- Precision = 0.213
- Recall = 0.676
- F2 score = 0.471

The Pros & Cons of XGBoost

Pros

- Fast
- Performant
- Easy to use

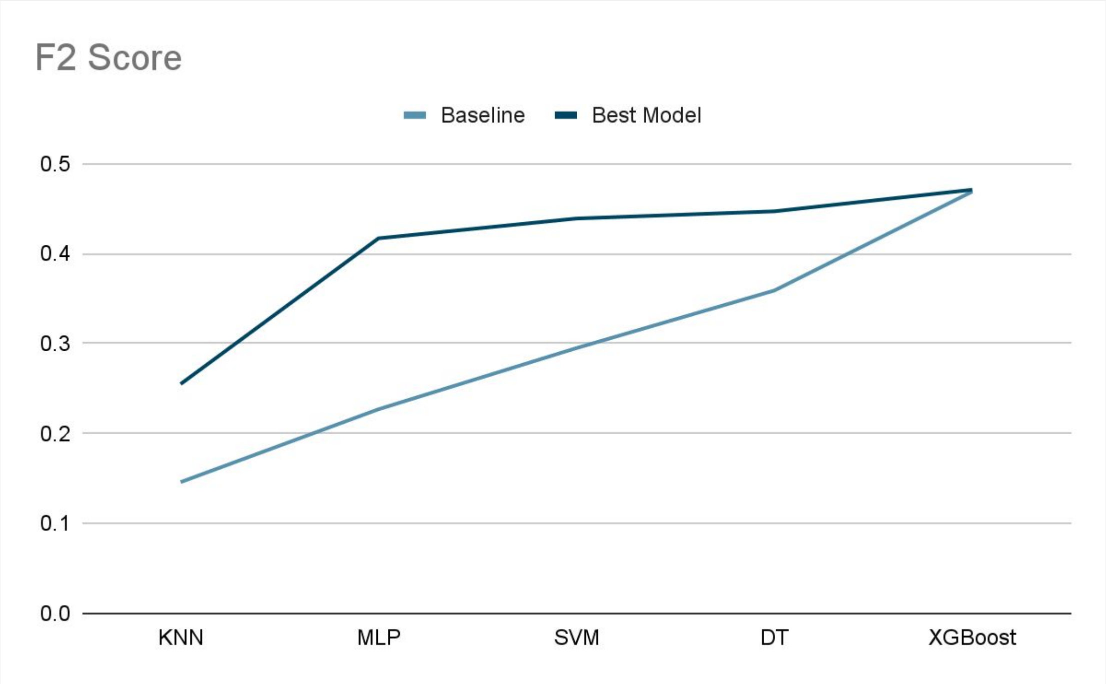
Cons

- Behaves like a black box model
- Loses interpretability due to ensembling many trees

04

Discussion

Evaluation Method



$$f_2 = \frac{5 \text{ Precision Recall}}{4 \text{ Precision} + \text{Recall}}$$

Undersampling

- Randomly discarded non-fraud data points
- Undersampled data may not be representative

Oversampling

- Randomly duplicated fraud instances
- Distinct number of fraud instances too small
- Increased computation

Ideal

Combine both ideas!

Future Improvements

- Better data resampling techniques
- Collect more instances of fraudulent data
- Allocate more computational time for more powerful models
- Ensembling between different models

Learning Outcomes

- Data cleaning/resampling techniques
- Usage of Python libraries such as SkLearn, Pandas
- XGBoost provides the best predictions

Acknowledgement

We would like to thank the NUS IT Research Computing team for providing us with the NUS High Performance Computing to compute our various Machine Learning models.

Moreover, we would like to thank Tian Fang for his continuous support throughout the semester.

Thanks!

We hope you enjoyed :)

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution