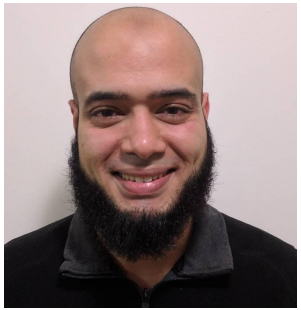
Machine Learning

Credit Card Fraud Detection

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Credit Card Fraud Detection

- Credit Card Fraud Detection Problem has 2 properties:
  - Binary classification: Is Fraud transaction?
  - Imbalanced dataset: There are very small fraud transaction (< 1%)
- Kaggle has some contest that will utilize
  - Most of features are anonymous, hence you can’t use domain knowledge
  - 2 clear features: time and amount of the transaction

Project Goals

- Your goal in this project is to build the best model that optimize **F1-Score**
  - You will also report the PR-AUC as a metric we care for, but F1 is the priority
  - Take it seriously, I will keep **internal scores** for what folks achieved
- As an imbalance dataset
  - Explore all you can to see if we can do something for this problem
  - Some examples: SMOTE, RandomOverSampler, RandomUnderSampler, etc
- Models of interest to explore:
  - LogisticRegression [we studied]
  - RandomForestClassifier [black-box]
  - Voting Classifier of both: LogisticRegression + RandomForestClassifier

Project input and output

- Find the data to use from here (not from kaggle)
  - You can use train.csv, val.csv or the combined file trainval.csv
- Your output
  - Save a pickle file that has dictionary for the model and your best threshold as below
  - There will be a separate test program that evaluate a private test.csv

```
model_dict = {
    "model": model,
    "threshold": threshold,
    "model_name": model_save_name
}

with open(os.path.join(root_dir, 'model.pkl'), 'wb') as file:
    pickle.dump(model_dict, file)
```

Deliverables

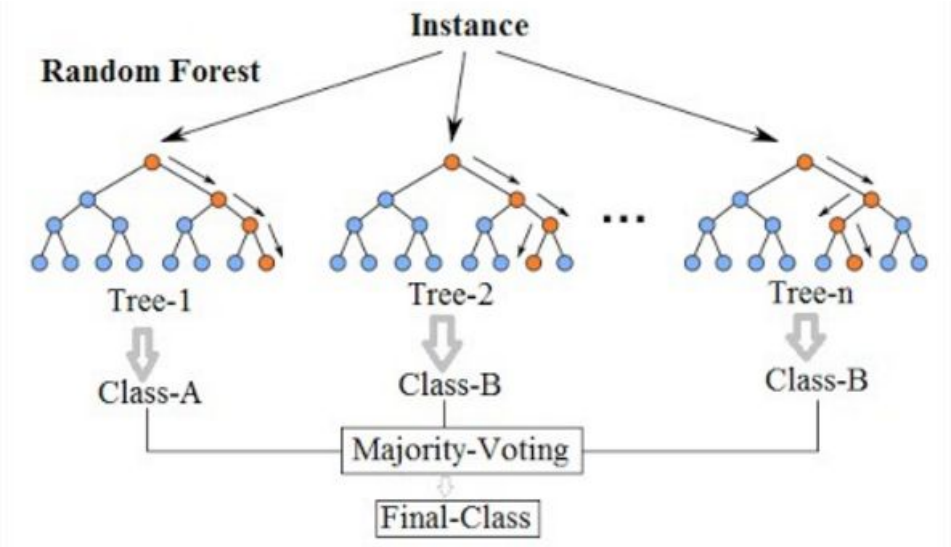
- 1) Prepare a jupyter notebook to explore and conclude about the train data
- 2) Create a project with 3 files:
  - credit_fraud_train.py:
    - The main entry of your program to load and train models
    - Use argparse to allow input entering (e.g. data path, which model, where to save, etc)
  - credit_fraud_utils_data.py
    - Contains any utilities that support data loading and processing
  - credit_fraud_utils_eval.py
    - Contains any utilities that support model evaluation
- 3) **model.pkl file**
  - Contains the dictionary of your best model
  - We will use it to evaluate against a hidden test set (who will get the highest score!)
  - When you are ready, let me know

Ensemble Models

- Techniques that **combine** multiple individual models to make predictions and average or vote among them to boost the performance
- **Voting Technique**: multiple different models trained independently and then combined (majority voting in classification or averaging n regression)
  - The same data - different models: quality of data + models are important
- **Bagging Technique**: Similar to voting, but each model is trained on a **random subset** of the data with replacement
  - So datasets are a bit different. Reduce variance and good for noisy data
  - Random Forest is a bagging approach

Ensemble Models

- Random Forest Classifier
  - The "forest" in Random Forest refers to a collection of **decision trees**, where each tree is built using a random subset of the training data and a random subset of the input features.



Ensemble Models: Boosting techniques

- **Boosting**: models are trained **sequentially**, with each subsequent model focusing on the examples that were **misclassified** by previous models
- AdaBoost: one of the earliest models
- Gradient Boosting: It aims to **minimize a loss function** by iteratively adding **weak models**: **XGBoost, LightGBM, and CatBoost**
  - Most of the market ML between deep learning and gradient boost techniques
  - Vert popular on kaggle. In future, put effort to study
  - **XGBoost**: highly optimized to improve performance and model accuracy
  - **LightGBM**: efficient in terms of memory usage and training speed
  - **CatBoost**: handle categorical features without requiring explicit encoding
- *Notes by chatgpt*

Using RandomForestClassifier

- In this project, you will explore the RandomForestClassifier as blackbox
  - You will explore 2 parameters:
  - max_depth: The maximum depth of the tree. You may explore in range [3-10]
  - n_estimators: The number of trees in the forest. You may explore in range [5-50]
- Optional: You are encouraged to educate yourself little more about
  - Decision Tree and Random Forest
  - Then, you may explore more parameters
- The usage of this model, is like **any classification** model in sklearn

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(max_depth=3, n_estimators=6)
```

Using

- Create a voting classifier on your best 2 models for random forest and logistic regression. Explore the 2 passed parameters below

```
from sklearn.ensemble import VotingClassifier

voting_model = VotingClassifier(
    estimators=[
        ('randomforest', ToDo),
        ('logstic', ToDo),
    ],
    voting='soft' , weights=[2, 1.5])
```

Project Report

- Create an online google doc for this project
  - Most of it mainly writing or little visualizations
- Focus on your findings
- Share with us some lessons you learned
- No specific format. Be smart in presenting yourself

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."