# University of Toronto
# ECE496 Implementation Plan

<u>Credit Card Fraud Detection</u>

Team Number:  2022524

Supervisor: Zeb Tate

Shadman Kaif, Abdurrafay Khan, Krutarth Patel, Shanthosh Sivayogaligam

**Executive summary**

As outlined in our Project Proposal, our goal was to create a linearly classifiable ensemble ML algorithm that minimizes the number of false positives to at least 20-30% [1] in a credit card dataset. In order to work together on our solution, the team agreed to use Google Colab, a collaborative coding environment that all team members are familiar with. The team has also developed a baseline LSTM model to compare the performance of our proposed solution with. The team then considered three possible solutions: a Support Vector Machine (SVM) model, the ensemble learning method of Bagging as well as AdaBoost. Using a weighted decision matrix and considering factors such as accuracy, processing speed and the number of false positives detected, the team chose an Adaboost for our proposed solution. Throughout the project, the team faced both technical and project management-related challenges. Google Colab's GPU limitation restricted the team to only preprocess two-thirds of the IBM dataset in order for the program to remain functional and not crash while training. The team also found it difficult to schedule team meetings on a week-by-week basis while accommodating scheduling conflicts. To meet internal project deadlines in a timely manner and reduce the number of postponed meetings, the team decided to schedule a fixed meeting time of 8:00 p.m. every Monday.

The team designed system-level diagrams for all possible solutions highlighting the differences in each model and also created a detailed module-level diagram for the chosen AdaBoost solution. The module-level diagram for the proposed solution outlines how the data will be processed, how the AdaBoost learning method is implemented as well as how the results will be classified for a prediction. The AdaBoost solution will need to meet a 20-30% false positive rate, 95% model accuracy and process 65,000 transactions/s to be a successful model. The solution will be constrained to only historical transactional data and cannot be trained on real credit card data due to ethical concerns.

For project management, the team has created a Gantt chart which serves the purpose of keeping the team on track of completing internal tasks by or before their assigned deadlines. A possible risk to be considered is the free version of Google Colab running out of its limited 12GB memory. Potential solutions the team has to include storing a Keras H5 file containing the model to prevent re-building every run or purchasing Colab Pro to increase memory to 32GB. Another possible risk is an endless runtime for the program as Colab prohibits background execution of code. In order to mitigate this possibility, the team plans to either use less testing data but keep the ratio of fraud and non-fraud transactions consistent or purchase Colab Pro+ which enables background execution.

**Project Status and Report and Changes**

At this stage of the project, the team has made substantial progress into designing and developing a solution to reduce the high false positive rates that exist in banks for credit card transactions. As outlined in our Project Proposal, our goal was to create a linearly classifiable ensemble ML algorithm that minimizes the number of false positives to at least 20-30% [1] in a credit card dataset. To create such an algorithm, the team agreed upon using Google Colab as the collaborative coding environment. Google Colab offers limited but free GPU resources which helps with training our model in an efficient and timely manner. It is also an environment that the team is familiar with through labs and projects completed in other past courses. The team has also implemented an LSTM model as a baseline in order to relatively compare the results that our ensemble method solution would produce. It currently produces the best results of current non-ensemble learning algorithms for a credit card fraud detection application, making it an optimal choice for our baseline model. Moving forward, the team has decided to implement an Adaboost model for our proposed solution of the various alternatives we explored.

One of the major challenges faced by the team was the GPU limitations with Google Colab. Due to the size of the dataset with 24 million entries, any training beyond a certain iteration would cause the program to fail in Google Colab. As a result, the initial baseline model could not be trained to the desired parameters. In order to overcome this problem, the team had to reduce the dataset to a subset consisting of 16.5 million rows in order to fall within the limits set up in Google Colab. Another obstacle the team faced was meeting every week consistently to discuss project status updates. Initially the team decided to schedule meetings on a week-by-week basis in order to accommodate the various conflicts we all had with midterms and assignments throughout the semester. This caused meetings to be postponed which ultimately delayed project tasks and limited progress. To solve this issue the team decided to meet regularly every Monday at 8:00 p.m. which ensures every team member is available.

A significant change made to this project after the proposal was to use a new dataset sourced from IBM instead of the proposed Kaggle dataset. The Kaggle dataset only consisted of 556,000 entries of different transactions which was not enough data to train, validate and test solutions to the desired accuracy for the project [2]. The new IBM dataset has significantly more data at 24 million entries of various transactions that will mitigate this problem [3].

**Possible Solutions and Design Alternatives**

It is essential to briefly describe our problem again. False positives among credit card transactions are a major issue amongst financial institutions; research shows that many banks have a false positive rate above 90% [4]. Our goal is to reduce the false positive rate to 20-30%. In the process of solving this problem, we explored a plethora of solutions. Two design alternatives that we explored are: support vector machines (SVM) and ensemble learning with bagging. The final solution that we chose is: ensemble learning with boosting.

It is imperative to explain each design alternative and the proposed solution. Support vector machine (SVM) is a popular classification tool. This is because of its ability to produce significantly accurate results while using less computational power. The algorithm's objective is to find an N-dimensional hyperplane that distinctly classifies the data(fraud/not fraud in our case). This is done by tuning the parameters of the hyperplane so that it best segregates that data into its separate classes. Figure 1 illustrates the system-level diagram of how the support vector machine algorithm would be implemented for our use case.

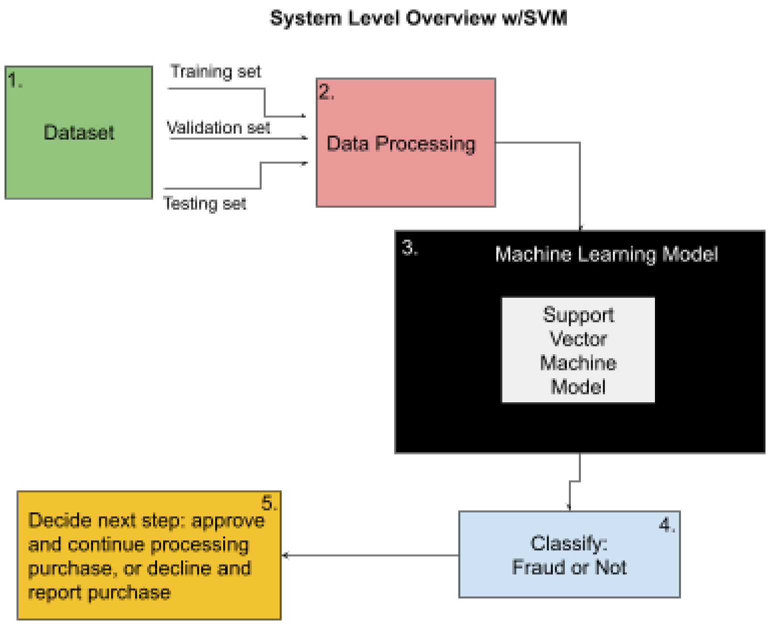

System Level Overview w/SVM
1.
Dataset
Training set
Validation set
Testing set
2.
Data Processing
3.
Machine Learning Model
Support Vector Machine Model
4.
Classify: Fraud or Not
5.
Decide next step: approve and continue processing purchase, or decline and report purchase

Figure 1. System Level Overview with SVM

A group of basic learners, or models, known as an ensemble, collaborate to make a superior final prediction. Due to excessive variation or strong bias, a single model, also known as a base or weak learner, may not perform effectively on its own. But when weak learners are combined, they become strong learners because the reduced bias or variance that results from their combination improves model performance [5]. Ensemble learning has been proven superior to traditional algorithms in other avenues such as malware detection and Twitter spam detection [6][7]. Two of the most popular ensemble learning algorithms are bagging and adaptive boosting (AdaBoost).

Bagging, also known as bootstrap aggregating, is an ensemble learning technique that improves the performance and accuracy of machine learning algorithms. It lowers the variance of a prediction model and is used to handle bias-variance trade-offs. Bagging is used in both regression and classification models to prevent overfitting of the data [8]. Figure 2 demonstrates a system level overview of how bagging would be utilized within our scope. After processing the data, bagging performs three main operations: bootstrapping, parallel training with the LSTM model, and aggregation. Bootstrapping is the process of generating bootstrapped samples from the given dataset. The data points are drawn at random with replacement to create the samples. Bootstrapping helps mitigate overfitting. The model performs well with the test data because it becomes resilient to generating errors when different sets of training data are utilized in the model. These bootstrap samples are then trained independently and in parallel with each other using weak or base learners [5], in our case, the LSTM model. Finally, in the aggregation stage, the majority of the predictions are taken to compute a more accurate estimate. As this is a classification task, the class with the highest majority of votes is accepted.

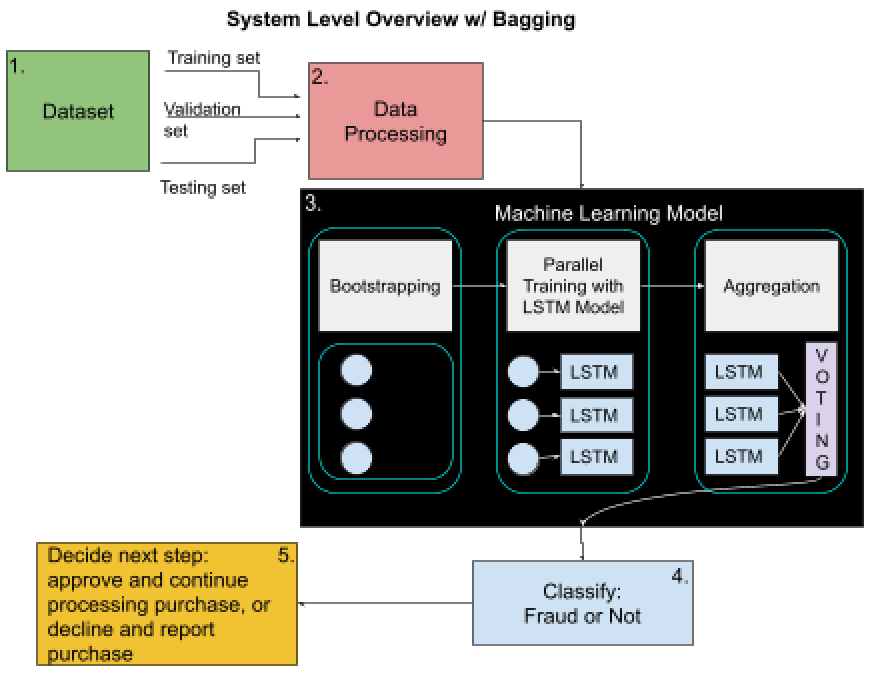

System Level Overview w/ Bagging
1. Dataset
Training set
Validation set
Testing set
2. Data Processing
3. Machine Learning Model
Bootstrapping
Parallel Training with LSTM Model
Aggregation
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
VOTING
4. Classify: Fraud or Not
5. Decide next step: approve and continue processing purchase, or decline and report purchase

Figure 2: System Level Overview with Bagging

Adaptive boosting, commonly known as AdaBoost, is another ensemble learning technique that improves the performance and accuracy of machine learning algorithms. It lowers the bias and variance of a prediction model and is used in both regression and classification models to prevent overfitting of the data [8]. Figure 3 demonstrates a system level overview of how AdaBoost would be utilized within our scope. After processing the data, AdaBoost uses the results of the other learning algorithms, or in our case, the LSTM model, and merges it to create a weighted total that represents the boosted classifier's final results. Initially all the training samples have the same weight. After each iteration, the weight distribution is updated such that misclassified samples have more weight. With the updated weight distribution, there are two ways of generating new training samples. In reweighting, the original training set is used, but each sample is associated with a new weight. This method is applicable to the learners that can handle weighted samples. In resampling, the new training set is constructed according to the weight distribution, where samples with more weights are more likely to be selected. Finally, in predicting a new sample, a weighted combination of multiple classifiers are used [9].

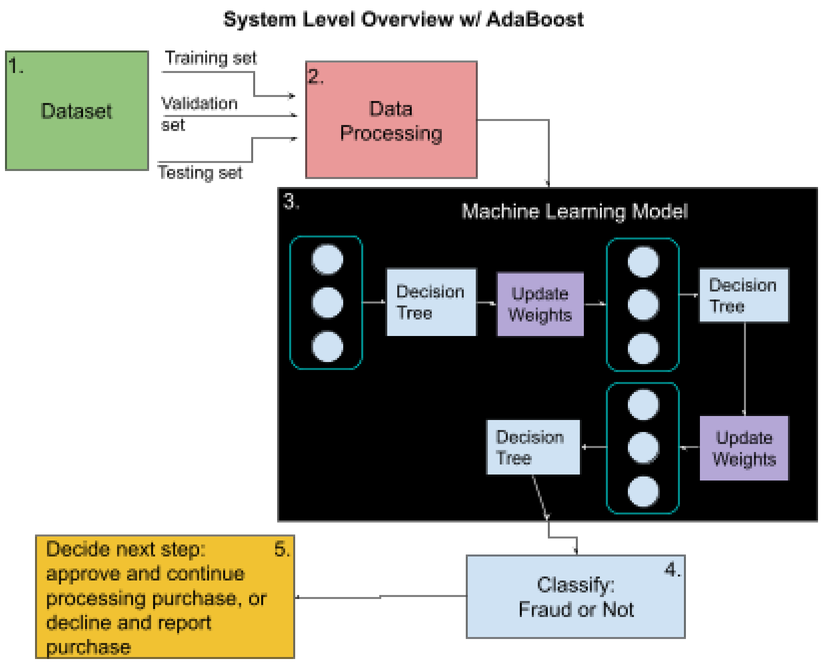


Figure 3: System Level Overview with AdaBoost

The main difference between the bagging and boosting learning methods is the way in which they are trained. In bagging, models are trained in parallel, but in boosting, they learn sequentially [5]. The weighted decision matrix was used to determine which of the three designs, SVM, bagging, and AdaBoost, was the most effective based on the objectives: minimizing false positives, and accuracy. Designs were graded out of ten after each objective was assigned a weight based on importance. The total rating of each design is the sum of each objective rating multiplied by its corresponding weight.

| Objective | Weight | Design 1 - SVM | Design 2 - Bagging | Design 3 - AdaBoost |
|---|---|---|---|---|
| Minimize the amount of false positives | 0.6 | 6 | 8 | 9 |
| Be accurate | 0.3 | 7 | 8 | 9 |
| Processing speed | 0.1 | 10 | 7 | 6 |

|  |  |  |  |  |
|---|---|---|---|---|
| *Total* | 1 | 6.7 | 7.9 | 8.7 |

Table 1: Weighted Decision Matrix of Three Design Alternatives

Our weighted decision matrix, Table 1, concludes that AdaBoost performs the best against our three objectives and thus will be the chosen solution. AdaBoost outperforms bagging in terms of accuracy and precision, which is a metric involving false positives [9][10]. Although bagging typically outperforms boosting, various renditions of boosting such as random-subspace boosting and AdaBoost provide capability to update weights sequentially, improving performance.

**Technical Design and Implementation**

Our chosen solution is the model with AdaBoost. Figure 3 illustrates the system-level overview of the solution with AdaBoost that we are planning to take to solve the problem. According to the diagram, the dataset that we will be using will be split into training, validation, and testing set. Afterwards, in the data processing step, the data is processed to make it interpretable and efficient for the computer to analyze and understand. After, that data will be used in the machine-learning model with AdaBoost to train and generate a model. Although outside of the scope, steps 4 and 5 of the diagram pertain to classifying the output from the model to determine a transaction as fraud/not fraud and decide the next steps/actions to take.

| **No** | **Module** | **Description** |
|---|---|---|
| 1 | Dataset | We will be using the IBM dataset and will split the data into training(50%), validation(30%), and testing(20%) sets. |
| 2 | Data Processing | The data from the dataset will be processed and information that contains texts such as transaction type, merchant name, etc will be converted to numerical data to make it easier and more efficient for the computer to analyze it. |
| 3 | Machine Learning Model | The data from the data processing block will be used as the input to the model. The Long Short-Term Memory (LSTM) model in association with the AdaBoost method will be used to train and tune the model. |

Table 2: Module Level Descriptions

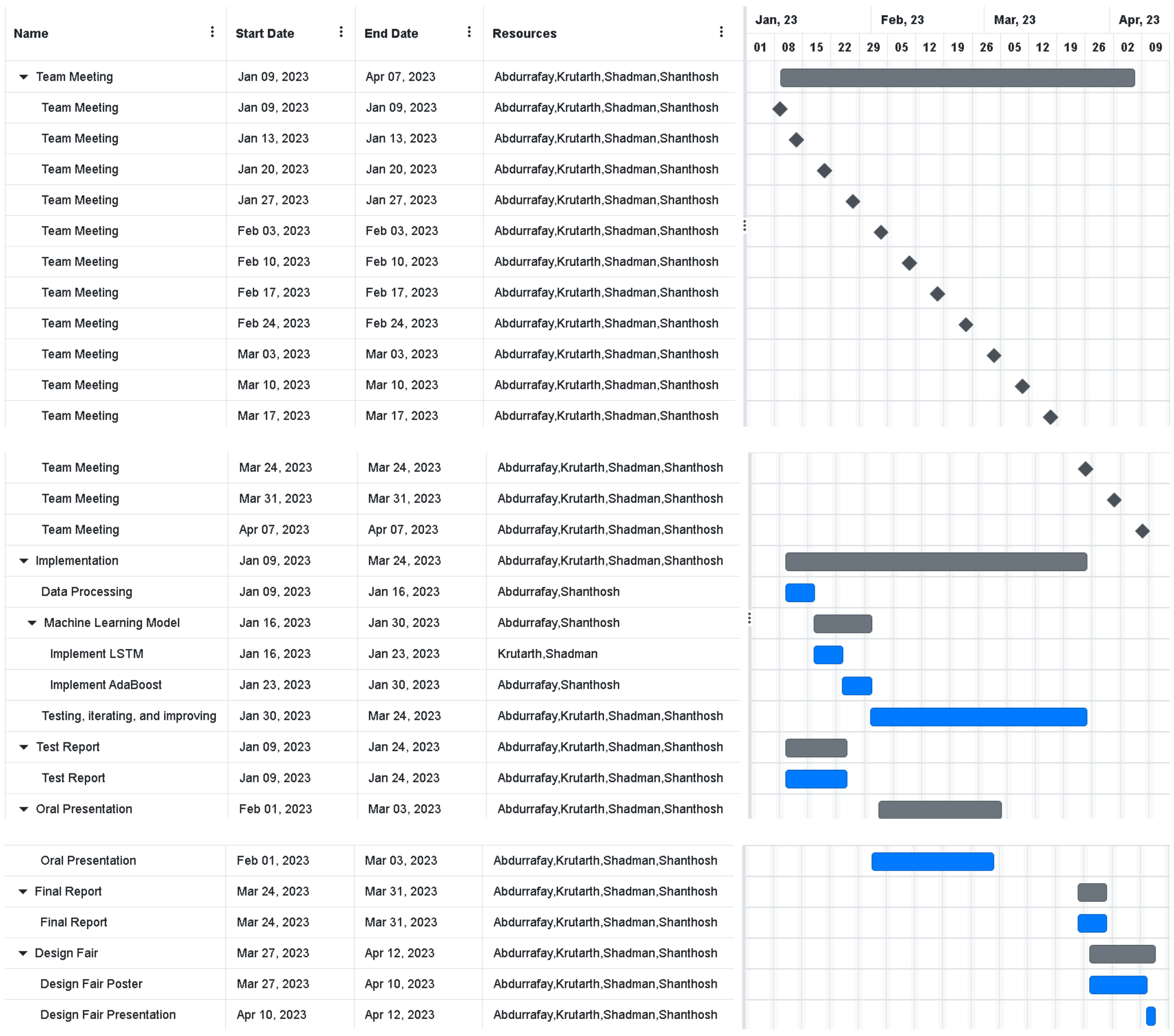
**Design Specifications**

The table below outlines aspects the team must consider when measuring the success of possible solutions, project limitations based on the gap and certain codes/regulations within financial institutions, and the functional basis of the project.

| No. | Item Type | Description | Metric | Goal |
|---|---|---|---|---|
| 1 | Function | Detect fraudulent credit card transactions using ensemble learning | False Positives (%) | 20-30% [1] using AdaBoost Ensemble Learning |
| 2 | Objective | Minimize the amount of false positives | False Positives (%) | 11[1] |
| 3 | Objective | Be accurate | Model Accuracy (%) | 95% [11] |
| 4 | Objective | Processing speed | Speed (transactions/s) | 65,000 [12] |
| 5 | Constraint (Functional) | Model cannot make a prediction for a real-time transaction when the transaction does not have historical data | Sequence Length | There must be a sequence length for our model to base previous transactions made by a card in order to predict the next transaction. |
| 6 | Constraint (Ethical) | No access to real data from credit card companies | Adherence to "Protection of Personal Information" [13] | Obtain adherence by resorting to a simulated dataset [14]. |

Table 3: Functions, Objectives, and Constraints Specifications

## Project Management



| Name | Start Date | End Date | Resources |
|---|---|---|---|
| Team Meeting | Jan 09, 2023 | Apr 07, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Jan 09, 2023 | Jan 09, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Jan 13, 2023 | Jan 13, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Jan 20, 2023 | Jan 20, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Jan 27, 2023 | Jan 27, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Feb 03, 2023 | Feb 03, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Feb 10, 2023 | Feb 10, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Feb 17, 2023 | Feb 17, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Feb 24, 2023 | Feb 24, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Mar 03, 2023 | Mar 03, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Mar 10, 2023 | Mar 10, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Mar 17, 2023 | Mar 17, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Mar 24, 2023 | Mar 24, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Mar 31, 2023 | Mar 31, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Team Meeting | Apr 07, 2023 | Apr 07, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Implementation | Jan 09, 2023 | Mar 24, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Data Processing | Jan 09, 2023 | Jan 16, 2023 | Abdurrafay,Shanthosh |
| Machine Learning Model | Jan 16, 2023 | Jan 30, 2023 | Abdurrafay,Shanthosh |
| Implement LSTM | Jan 16, 2023 | Jan 23, 2023 | Krutarth,Shadman |
| Implement AdaBoost | Jan 23, 2023 | Jan 30, 2023 | Abdurrafay,Shanthosh |
| Testing, iterating, and improving | Jan 30, 2023 | Mar 24, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Test Report | Jan 09, 2023 | Jan 24, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Test Report | Jan 09, 2023 | Jan 24, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Oral Presentation | Feb 01, 2023 | Mar 03, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Oral Presentation | Feb 01, 2023 | Mar 03, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Final Report | Mar 24, 2023 | Mar 31, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Final Report | Mar 24, 2023 | Mar 31, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Design Fair | Mar 27, 2023 | Apr 12, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Design Fair Poster | Mar 27, 2023 | Apr 10, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |
| Design Fair Presentation | Apr 10, 2023 | Apr 12, 2023 | Abdurrafay,Krutarth,Shadman,Shanthosh |

Figure 4: Gantt chart for the team’s task schedule.

## Risk Assessment

During a year-long engineering project, there are often threats to the project, ranging from the completion of certain modules, the possibility of missing deadlines, and software tools not working to expectations. Three credible risks to our project in particular are: running out of Google Colab memory, and endless runtime for making predictions over the test set.

A table summarizing the credible risks and brief description and mitigation plan is below.

| No. | Risk | Description | Mitigation Plan |
|---|---|---|---|
| 1 | Running out of Colab memory | There is a 12GB memory limit for free Colab GPU usage [15]. | Two potential resolutions:<br>1) Save a Keras H5 file containing the model to prevent building the model during every |

| | | | |
|---|---|---|---|
| | | | run<br>2) Purchase Colab Pro which increases memory to 32GB [16] |
| 2 | Endless runtime for making predictions over the test set | Obtaining metrics over the test data takes hours on end, and Colab prohibits background execution. | Two potential resolutions:<br>1) Use less testing data but still ensure we use a mix of fraud and non-fraud transactions<br>2) Purchase Colab Pro+ which allows background execution [16] |

Table 4: Credible Risks and Mitigation Plan

Google Colab is a hosted Jupyter notebook service that allows users to write and execute arbitrary Python code through the browser with GPUs for free [15]. Although it is a free service, the limitation is that users only get to use 12GB of memory at once. The downside of this limitation in our project is that we import a 2.3GB CSV file and perform various data processing operations, which use a large amount of the free allocated memory in itself. Furthermore, we must build the model over 10 epochs and perform testing over the dataset to obtain metrics against our specifications. Thus, we often run out of memory and are unable to continue testing. Two potential resolutions to this memory limitation are: purchasing Colab Pro and saving a Keras H5 file containing the model. Colab Pro increases the memory from 12GB to 32GB and allows us to continue the end-to-end run without being disrupted. Saving the model in Google Drive and importing it for the next run prevents the model build from allocating memory every time we open and run the script. Both or either of these solutions helps mitigate the Colab memory limitation that we are currently enduring.

The test data is 20% of the entire data, which amounts to approximately 5 million rows. Since our model is sequential, it relies on historical data to perform predictions on test data. Thus, running over the test data to make predictions and obtaining metrics for our objectives such as accuracy, false positives, and speed, is time-consuming. During our test run for the baseline LSTM model, the model took over 6 hours to obtain all the required metrics. We also had to reduce the 24 million total rows to 16.5M rows to get the results in 6 hours. As we train with more complicated models which involve ensemble machine learning, we anticipate longer runtimes. To mitigate long runtimes which sign us out of Colab's 24-hour window, we have two potential resolutions. The first is to purchase Colab Pro+ which allows background execution [16]. Background execution allows us to close the Colab notebook without having to worry about terminations. Another resolution is to use less testing data whilst still manually ensuring we use a combination of fraud and non-fraud transactions and are not just blindly removing one particular subset of data as it could skew and provide biased results.

## References

[1] F. Wallny, “False Positives in Credit Card Fraud Detection: Measurement and Mitigation,” 2022. [Online]. Available: https://scholarspace.manoa.hawaii.edu/. E. Esenogho, I. D. Mienye, T. G. Swart, K. Aruleba, and G. Obaido, “A neural network ensemble with feature engineering for improved credit card fraud detection,” IEEE Access, vol. 10, pp. 16400–16407, 2022.

[2] K. Shenoy, “Credit Card Transactions Fraud Detection Dataset,” *Kaggle*, 05-Aug-2020. [Online]. Available: https://www.kaggle.com/datasets/kartik2112/fraud-detection?select=fraudTrain.csv.

[3] IBM, “IBM/Tabformer: Code & data for ‘Tabular transformers for modeling multivariate time series’ (ICASSP, 2021),” *GitHub*. [Online]. Available: https://github.com/IBM/TabFormer. [Accessed: 09-Jan-2023].

[4] D. Holmes, “The Real Costs of False Positives for Banks,” *Feedzai*, 16-May-2022. [Online]. Available: https://feedzai.com/blog/the-real-costs-of-false-positives-for-banks.

[5] IBM, “What is bagging?,” *IBM*. [Online]. Available: https://www.ibm.com/topics/bagging. [Accessed: 09-Jan-2023].

[6] K. N. Khasawneh, M. Ozsoy, C. Donovick, N. Abu-Ghazaleh, and D. Ponomarev, “Ensemble learning for low-level hardware-supported malware detection,” *Research in Attacks, Intrusions, and Defenses*, pp. 3–25, 2015.

[7] S. Liu, Y. Wang, J. Zhang, C. Chen, and Y. Xiang, “Addressing the class imbalance problem in Twitter spam detection using Ensemble Learning,” *ScienceDirect*, 13-Dec-2016. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/S0167404816301754?token=39F0203C8C0EBCE1DCDE4495D153A4678D27E055F52E742164DF963E5A26E2FE66C0291FB67B1909BA70F32E70EABD4D&originRegion=us-east-1&originCreation=20220923030108.

[8] A. Biswal, “What is bagging in machine learning and how to perform bagging,” *Simplilearn.com*, 30-Nov-2022. [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/bagging-in-machine-learning#what_is_bagging_in_machine_learning. [Accessed: 09-Jan-2023].

[9] X. Sun, “Pitch accent prediction using Ensemble Machine Learning,” *7th International Conference on Spoken Language Processing (ICSLP 2002)*, 2002.

[10] Y. Zhu, C. Xie, G.-J. Wang, and X.-G. Yan, “Comparison of individual, Ensemble and Integrated Ensemble Machine Learning Methods to predict China’s SME credit risk in Supply Chain Finance,” *Neural Computing and Applications*, vol. 28, no. S1, pp. 41–50, 2016.

[11] R. Bin Sulaiman, V. Schetinin, and P. Sant, “Review of Machine Learning Approach on credit card fraud detection,” *Human-Centric Intelligent Systems*, vol. 2, no. 1-2, pp. 55–68, 2022.

[12] “Visanet: Global Electronic Payment Network,” *Visa*. [Online]. Available: https://www.visa.ca/en_CA/about-visa/visanet.html.

[13] “Protection of Personal Information,” *Bank of Canada*. [Online]. Available: https://www.bankofcanada.ca/privacy/protection-of-personal-information/.

[14] K. Shenoy, “Credit Card Transactions Fraud Detection Dataset,” *Kaggle*, 05-Aug-2020. [Online]. Available: https://www.kaggle.com/datasets/kartik2112/fraud-detection?select=fraudTrain.csv.

[15] *Google colab*. [Online]. Available: https://research.google.com/colaboratory/faq.html. [Accessed: 09-Jan-2023].

[16] David, “Alternative to colab pro: Comparing Google's Jupyter notebooks to gradient notebooks (updated!),” Paperspace Blog, 23-Dec-2022. [Online]. Available: https://blog.paperspace.com/alternative-to-google-colab-pro/#:~:text=Colab%20Pro%20limits%20RAM%20to,background%20execution%2C%20while%20Pro%2B%20does. [Accessed: 09-Jan-2023].