

Internship report

FLAPMAX | FAI INSTITUTE | WENTORS



TOPIC: CUSTOMER CHURN PREDICTION

BY ABARUGO GOODNESS C.

GROUP B

Contents

[INTRODUCTION 1](#_Toc143734491)

[PROJECT OVERVIEW 2](#_Toc143734492)

[Objective 2](#_Toc143734493)

[Project Data: 2](#_Toc143734494)

[Type of Learning 2](#_Toc143734495)

[Project Approach and Tools: 2](#_Toc143734496)

[Significance 3](#_Toc143734497)

[REPORT STRUCTURE 4](#_Toc143734498)

[Data Preprocessing 4](#_Toc143734499)

[Data Cleaning and Feature Engineering 4](#_Toc143734500)

[Modelling 5](#_Toc143734501)

[Data Splitting 6](#_Toc143734502)

[Models 6](#_Toc143734503)

[RESULTS AND INSIGHTS 7](#_Toc143734504)

[Stock Environment 7](#_Toc143734505)

[First Training 7](#_Toc143734506)

[With Hyperparameter Tuning 7](#_Toc143734507)

[Intel Environment 9](#_Toc143734508)

[First Training: 9](#_Toc143734509)

[With Hyoerparameter Tuning 10](#_Toc143734510)

[Logistic Regression 12](#_Toc143734511)

[Insights 13](#_Toc143734512)

[Comparing Model Training Time 13](#_Toc143734513)

[Comparing Model Classification Results 15](#_Toc143734514)

[Summary Environment Comparison 17](#_Toc143734515)

[CHALLENGES, SOLUTIONS & RECOMMENDATIONS 18](#_Toc143734516)

[Challenges and Solutions 18](#_Toc143734517)

[Recommendations 18](#_Toc143734518)

[CONCLUSION 20](#_Toc143734519)

[ACKNOWLEDGEMENTS 21](#_Toc143734520)

[References 22](#_Toc143734521)

# INTRODUCTION

This report documents my internship experience at Flapmax as part of the Women in AI (WAI) Initiative. During this period, my group members and I worked on customer churn prediction project using artificial Intelligence (AI) techniques. The project aimed at predicting which customer may churn based on the pattern learnt from historical data, in order to prevent and discourage churn. This report provides an overview of the project, its methodologies, objectives, challenges and outcomes.

# PROJECT OVERVIEW

## Objective

The primary objective of this project is to develop a machine learning model for predicting customer churn using historical data.

## Project Data:

The dataset is a structured telecom internet subscriber data sourced from Kaggle (https://www.kaggle.com/datasets/mehmetsabrikunt/internet-service-churn) and contains 72274 rows, and 11 columns of customer data. Each entry contains the following information;

1. Id:This is customers unique identification number.
2. is\_tv\_subcriber: These are customers that subscribes to tv packages; 1 means subscriber where 0 means no subscriber
3. is\_movie\_package\_subscriber: Customers that subscribes to movie packages; 1 means subscriber where 0 means no subscriber
4. subscription\_age: Age of the subscribers.
5. bill\_avg: customers billing average.
6. reamining\_contract: shows the units left of customer subscription
7. service\_failure\_count: This counts the number of times customer call to call center for service failure for last 3 months
8. download\_avg: average downloads made by customer
9. upload\_avg: average uploads made by customer
10. download\_over\_limit: limit of download for each customer
11. churn: describes the loss of customers who don't resign their contract at the time of their renewal

## Type of Learning

Supervised learning Models: Logistic regression, random forest classification Output: Yes or No if customer churn is predicted.

## Project Approach and Tools:

This project was executed using the following tools:

* Python 3 (Anaconda)
* Libraries (numpy, pandas, Scikit-learn, Intel Extension Scikit-learn)
* Jupyter Notebook
* Git Bash

The project involves the use of different Python environments for model training and testing. The environments are:

* Stock Environment (Regular Scikit-learn Python Library)
* Intel OneAPI Powered Environment (Intel Python and Intel improved Scikit-learn Library)

The goal of this approach is to compare these environments in terms of key performance metrics such as accuracy, speed, and time efficiency.

## Problem Statement

The objective of this project is to leverage a dataset to develop predictive models using two distinct algorithms for the purpose of assisting a company in predicting customer churn within the telecom industry. These models will be created and tested in different computational environments. The primary aim is to assess and compare the performance of these predictive models under varying conditions, identifying challenges, potential improvements, and insights that can inform strategies for reducing customer churn.

## Significance

Predicting customer churn, or the likelihood of customers leaving, is of paramount importance to businesses across diverse sectors. In a global marketplace characterized by fierce competition, retaining existing customers is often more cost-effective than acquiring new ones. Churn prediction equips companies with the means to foresee which customers are likely to leave. This foresight enables businesses to implement proactive strategies to retain valuable customers, addressing their concerns and needs effectively. By reducing churn, companies can maintain a loyal customer base and improve overall profitability.

In the context of this internship, our project challenged our data analysis and machine learning skills to predict customer churn in the telecommunications industry. In this industry, the significance of churn prediction and data-driven decision-making increases significantly. This sector is highly dynamic and competitive, with customers frequently exploring alternative service providers. Accurate churn prediction in telecoms empowers companies to identify and target customers at risk of leaving. By analyzing customer behavior and historical data, telecom providers can tailor retention strategies, offering personalized incentives, resolving issues promptly, and ensuring a seamless customer experience. The implications extend beyond customer retention; they encompass cost savings, revenue growth, and gaining competitive edge in the telecom market.

# REPORT STRUCTURE

This report covers all the activities carried out individually and with my team, such as data preprocessing, model development, results, challenges, and recommendations.

## Data Preprocessing

Data description

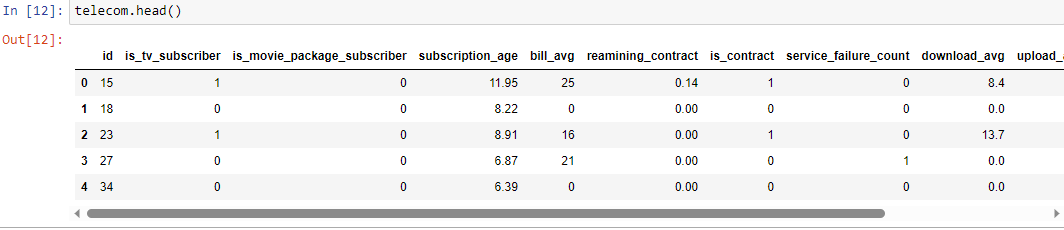
* id: A unique identifier for each customer.
* is\_tv\_subscriber: Indicates whether the customer is a TV subscriber (1 for yes, 0 for no).
* is\_movie\_package\_subscriber: Indicates whether the customer subscribes to a movie package (1 for yes, 0 for no).
* subscription\_age: Represents the age of the subscription.
* bill\_avg: Average billing amount.
* remaining\_contract: Remaining contract duration.
* service\_failure\_count: Count of service failures.
* download\_avg: Average download speed.
* upload\_avg: Average upload speed.
* download\_over\_limit: Indicates if downloads exceeded a limit (1 for yes, 0 for no).
* churn: Indicates customer churn (1 for churned, 0 for retained).

### Data Cleaning and Feature Engineering

We carried out the following during data cleaning:

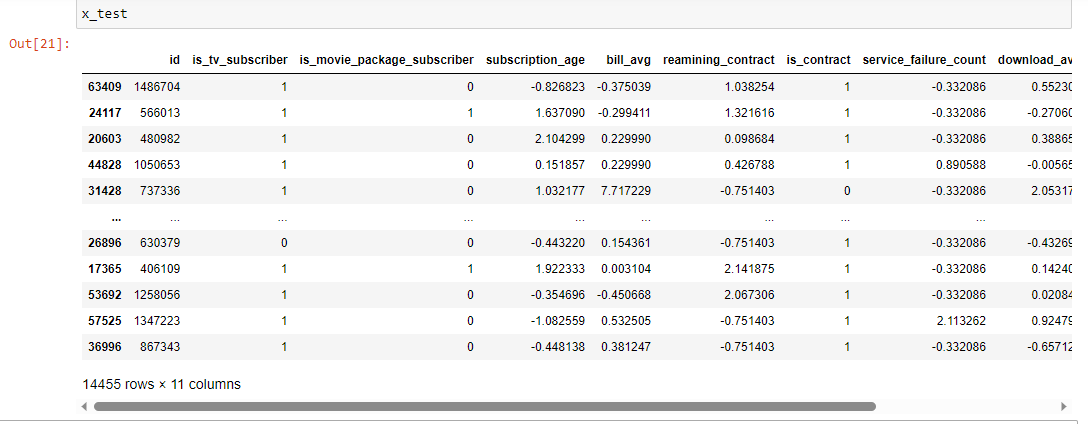
* **Creating 'is\_contract' Column**: We created a new column named 'is\_contract' based on the 'reamining\_contract' column. This column appears to represent whether a contract exists or not (0 for no contract, 1 for an active contract).
* **Imputing Null Values**: Null values in the 'reamining\_contract' column were replaced with 0, indicating no remaining contract duration.
* **Rearranging Columns:** The order of columns in the DataFrame was rearranged to match the desired sequence of features. This can enhance clarity and organization.
* **Handling Missing Values**: We replaced empty strings in the 'download\_avg' and 'upload\_avg' columns with NaN values.
* **Drop Rows with Missing Values**: Rows with missing values in 'download\_avg' and 'upload\_avg' columns were dropped from the DataFrame.

After data cleaning, our dataset looks like the image below:



* **Feature Scaling:** Numerical features such as 'subscription\_age,' 'reamining\_contract,' 'download\_avg,' 'upload\_avg,' 'download\_over\_limit,' 'bill\_avg,' and 'service\_failure\_count' were standardized using the StandardScaler. This scales these features to have a mean of 0 and a standard deviation of 1, making them suitable for machine learning algorithms.

After scaling, our dataset looks like the image below:



The entire preprocessing activity ensures that we had standardized features and that helps us to build a more robust predictive model.

## Modelling

Our project involves a supervise machine learning. For that reason, we had to categorize the data and split them before modelling. After visually preprocessing the data, we switched to completing the entire process from preprocessing to modelling and tuning using automation scripts (utils.py and training.py) on our different environments.

### Data Splitting

First, we separated features from labels. So, all columns remain features except the churn column which serves as the label. The dataset was split into training and testing sets using the train\_test\_split function. The split ensures that the models are not exposed to test data during training, to avoid overfitting. This is a critical step to evaluate the machine learning model's performance. On jupyter notebook, we manually assigned the split portion to 20% of the dataset which is also the default argument in the automation script

### Models

For our project, we used the Logistic Regression and Random forest algorithms to train our models in our two environments – stock and intel. After training, we tuned the hyperparameters using set values.

# RESULTS AND INSIGHTS

## Stock Environment

Following are the results gotten from the modelling process.

### First Training

INFO:\_\_main\_\_:[RandomForestClassifier] Training time: 21.595000 secs

INFO:\_\_main\_\_:[LogisticRegression] Training time: 0.408837 secs

### With Hyperparameter Tuning

**Random Forest**

INFO:\_\_main\_\_:[RandomForestClassifier] Training Time with Hyper parameter Tuning: 104.896959 secs

INFO:\_\_main\_\_:[RandomForestClassifier] Best parameters {'max\_depth': None, 'max\_leaf\_nodes': 45, 'n\_estimators': 100}

INFO:\_\_main\_\_:[RandomForestClassifier] Best Accuracy score 0.940988288938414

INFO:\_\_main\_\_:[RandomForestClassifier] Training Time with best hyper parameters: 9.861994 secs

**Logistic Regression**

INFO:\_\_main\_\_:[LogisticRegression] Training Time with Hyper parameter Tuning: 8.489990 secs

INFO:\_\_main\_\_:[LogisticRegression] Best parameters {'fit\_intercept': True}

INFO:\_\_main\_\_:[LogisticRegression] Best Accuracy score 0.44291824565546295

INFO:\_\_main\_\_:[LogisticRegression] Training Time with best hyper parameters: 0.076013 secs

Summary Table

|  |  |  |
| --- | --- | --- |
| Algorithm | Training Time | Accuracy |
| RandomForestClassifier | 21.595000 secs | - |
| Logistic Regression | 0.408837 secs | - |
| Random Forest (Tuned) | 104.896959 secs | 94.1% |
| Logistic Regression (Tuned) | 8.489990 secs | 44.3% |

Testing

INFO:utils:[Data] DataPreparation Time Taken in seconds --> 0.918993 secs

INFO:utils:[Data] Total Data samples ---> 71893

INFO:\_\_main\_\_:[RandomForestClassifier] Time taken for Batch inference of size 3000 is: 0.121991 secs

INFO:\_\_main\_\_:[RandomForestClassifier] Classification Report

|  |  |  |  |
| --- | --- | --- | --- |
| **precision** | **recall** | **f1-score** | **support** |
| 0.94 | 0.95 | 0.95 | 1353 |
| 0.96 | 0.95 | 0.95 | 1647 |
| accuracy | 0.95 | 3000 |  |
| macro avg | 0.95 | 0.95 | 0.95 |
| weighted avg | 0.95 | 0.95 | 0.95 |

Confusion Matrix

|  |  |  |
| --- | --- | --- |
|  | Predicted Negative | Predicted Positive |
| Actual Negative | 1289 | 64 |
| Actual Positive | 85 | 1562 |

INFO:\_\_main\_\_:[RandomForestClassifier] Accuracy is: 0.950333

INFO:\_\_main\_\_:[RandomForestClassifier] Average Real Time inference: 0.036580 secs

INFO:utils:[Data] DataPreparation Time Taken in seconds --> 0.683984 secs

INFO:utils:[Data] Total Data samples ---> 71893

INFO:\_\_main\_\_:[LogisticRegression] Time taken for Batch inference of size 3000 is: 0.008998 secs

|  |  |  |  |
| --- | --- | --- | --- |
| **precision** | **recall** | **f1-score** | **support** |
| 0.45 | 1 | 0.62 | 1353 |
| 0 | 0 | 0 | 1647 |
| accuracy | 0.45 | 3000 |  |
| macro avg | 0.23 | 0.5 | 0.31 |
| weighted avg | 0.2 | 0.45 | 0.28 |

Confusion matrix

|  |  |  |
| --- | --- | --- |
|  | Predicted Negative | Predicted Positive |
| Actual Negative | 1353 | 0 |
| Actual Positive | 1647 | 0 |

INFO:\_\_main\_\_:[LogisticRegression] Accuracy is: 0.451000

INFO:\_\_main\_\_:[LogisticRegression] Average Real Time inference: 0.005649 secs

Summary Table of Model Performance

|  |  |  |
| --- | --- | --- |
| Algorithm | Testing Time | Accuracy |
| RandomForestClassifier | 0.036580 secs | 95.0% |
| Logistic Regression | 0.005649 secs | 45.1% |

## Intel Environment

### First Training:

Randon Forest

INFO:utils:[Data] DataPreparation Time Taken in seconds --> 1.541863 secs

INFO:utils:[Data] Total Data samples ---> 71893

INFO:root:sklearn.ensemble.RandomForestClassifier.fit: running accelerated version on CPU

INFO:\_\_main\_\_:[RandomForestClassifier] Training time: 5.906224 secs

Logistic Regression

INFO:utils:[Data] DataPreparation Time Taken in seconds --> 0.635003 secs

INFO:utils:[Data] Total Data samples ---> 71893

INFO:root:sklearn.linear\_model.LogisticRegression.fit: running accelerated version on CPU

INFO:\_\_main\_\_:[LogisticRegression] Training time: 1.210351 secs

### With Hyoerparameter Tuning

Random forest

INFO:root:sklearn.ensemble.RandomForestClassifier.fit: running accelerated version on CPU

INFO:\_\_main\_\_:[RandomForestClassifier] Training Time with Hyper parameter Tuning: 67.633824 secs

INFO:\_\_main\_\_:[RandomForestClassifier] Best parameters {'max\_depth': None, 'max\_leaf\_nodes': 45, 'n\_estimators': 150}

INFO:\_\_main\_\_:[RandomForestClassifier] Best Accuracy score 0.9397016414532885

INFO:root:sklearn.ensemble.RandomForestClassifier.fit: running accelerated version on CPU

INFO:\_\_main\_\_:[RandomForestClassifier] Training Time with best hyper parameters: 2.223990 secs

Logistic Regression

INFO:root:sklearn.linear\_model.LogisticRegression.fit: running accelerated version on CPU

INFO:\_\_main\_\_:[LogisticRegression] Training Time with Hyper parameter Tuning: 8.968993 secs

INFO:\_\_main\_\_:[LogisticRegression] Best parameters {'fit\_intercept': True}

INFO:\_\_main\_\_:[LogisticRegression] Best Accuracy score 0.44291824565546295

INFO:root:sklearn.linear\_model.LogisticRegression.fit: running accelerated version on CPU

INFO:\_\_main\_\_:[LogisticRegression] Training Time with best hyper parameters: 0.236005 secs

|  |  |  |
| --- | --- | --- |
| Algorithm | Training Time | Accuracy |
| RandomForestClassifier | 5.906224 secs | - |
| Logistic Regression | 1.210351 secs | - |
| Random Forest (Tuned) | 67.633824 secs | 94.0% |
| Logistic Regression (Tuned) | 8.968993 secs | 44.3% |

Testing

Random Forest

INFO:root:sklearn.ensemble.RandomForestClassifier.predict: running accelerated version on CPU

INFO:\_\_main\_\_:[RandomForestClassifier] Time taken for Batch inference of size 3000 is: 0.132000 secs

INFO:\_\_main\_\_:[RandomForestClassifier] Classification Report

INFO:\_\_main\_\_:[RandomForestClassifier] Classification Report

|  |  |  |  |
| --- | --- | --- | --- |
| **precision** | **recall** | **f1-score** | **support** |
| 0.94 | 0.95 | 0.95 | 1353 |
| 0.96 | 0.95 | 0.95 | 1647 |
| accuracy | 0.95 | 3000 |  |
| macro avg | 0.95 | 0.95 | 0.95 |
| weighted avg | 0.95 | 0.95 | 0.95 |

Confusion Matrix

|  |  |  |
| --- | --- | --- |
|  | Predicted Negative | Predicted Positive |
| Actual Negative | 1285 | 68 |
| Actual Positive | 84 | 1563 |

INFO:\_\_main\_\_:[RandomForestClassifier] Accuracy is: 0.949333

### Logistic Regression

INFO:\_\_main\_\_:[LogisticRegression] Time taken for Batch inference of size 3000 is: 0.007002 secs

INFO:\_\_main\_\_:[LogisticRegression] Classification Report

|  |  |  |  |
| --- | --- | --- | --- |
| **precision** | **recall** | **f1-score** | **support** |
| 0.45 | 1 | 0.62 | 1353 |
| 0 | 0 | 0 | 1647 |
| accuracy | 0.45 | 3000 |  |
| macro avg | 0.23 | 0.5 | 0.31 |
| weighted avg | 0.2 | 0.45 | 0.28 |

Confusion matrix

|  |  |  |
| --- | --- | --- |
|  | Predicted Negative | Predicted Positive |
| Actual Negative | 1353 | 0 |
| Actual Positive | 1647 | 0 |

INFO:\_\_main\_\_:[LogisticRegression] Accuracy is: 0.451000

Summary Table for Intel Hyperparamter Tuning

|  |  |  |
| --- | --- | --- |
| Algorithm | Testing Time | Accuracy |
| RandomForestClassifier | 0.036580 secs | 95.0% |
| Logistic Regression | 0.005649 secs | 45.1% |

## Insights

### Comparing Model Training Time

|  |  |  |
| --- | --- | --- |
| Table showing Model Training Speeds | | |
|  |  |  |
|  | Stock | Intel |
| RandomForestClassifier | 21.60 | 5.91 |
| Logistic Regression | 0.41 | 1.21 |
| Random Forest (Tuned) | 104.90 | 67.63 |
| Logistic Regression (Tuned) | 8.49 | 8.97 |

The table and chart above present a comprehensive comparison of the training speeds for the two machine learning models in our two distinct environments: "Stock" and "Intel." Training speed is a crucial metric in machine learning, as it directly impacts the time required to develop, validate, and deploy predictive models.

Random Forest Classifier: In the "Stock" environment, the Random Forest Classifier model took approximately 21.60 units of time for training. In the "Intel" environment, the training speed improved significantly, reducing the training time to approximately 5.91 units. This improvement suggests that the "Intel" environment offers superior computational resources for this particular model. Logistic Regression: The Logistic Regression model's training speed is also compared in both environments. In the "Stock" environment, the training time is approximately 0.41 units. In the "Intel" environment, the training time increases to approximately 1.21 units. This difference indicates that the "Intel" environment, while more capable for Random Forest Classifier, may not necessarily be faster for all types of machine learning models.

Random Forest (Tuned): The "Random Forest (Tuned)" model refers to the Random Forest Classifier with specific hyperparameters (n-estimators, max\_depth and batch size) optimized for performance. In the "Stock" environment, the tuned Random Forest model required around 104.90 units of training time. In the "Intel" environment, the training time is approximately 67.63 units. This suggests that, even though the "Intel" environment is faster for this tuned Random Forest model, the training time remains substantial.

Logistic Regression (Tuned): Similarly, the "Logistic Regression (Tuned)" model represents a Logistic Regression model with optimized hyperparameters. In the "Stock" environment, it took approximately 8.49 units of time for training. In the "Intel" environment, the training time is nearly identical at approximately 8.97 units. This indicates that for this specific tuned Logistic Regression model, there is no significant difference in training speed between the two environments.

In the comparison of training speeds for various machine learning models in the "Stock" and "Intel" environments, some notable patterns emerge. **The "Intel" environment demonstrates its superiority in terms of training speed for complex models such as the Random Forest Classifier and the tuned Random Forest model. These models benefit significantly from the computational resources provided by the Intel library, resulting in significantly faster training times. On the other hand, for simpler models like Logistic Regression and the tuned Logistic Regression model, the "Stock" environment emerges as the preferred choice, offering comparable training speeds to the "Intel" environment.**

### Comparing Model Classification Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Stock Log-R | Intel Log-R | Stock RFC | Intel RFC |
| precision (no) | 0.45 | 0.45 | 0.94 | 0.94 |
| Recall (no) | 1.00 | 1.00 | 0.95 | 0.95 |
| F1-score (no) | 0.62 | 0.62 | 0.95 | 0.95 |
| precision (yes) | 0.00 | 0.00 | 0.96 | 0.96 |
| Recall (yes) | 0.00 | 0.00 | 0.95 | 0.95 |
| F1-score (yes) | 0.00 | 0.00 | 0.95 | 0.95 |
| Accuracy | 45% | 45% | 95% | 95% |

The table and chart above present the metrics presented for the different models in the "Stock" and "Intel" environments:

**Logistic Regression (Log-R)**

- Precision (no): Both in the "Stock" and "Intel" environments, Logistic Regression demonstrates identical precision for the negative class (no), which is 0.45. This metric indicates that around 45% of the predictions for the negative class were correct.

- Recall (no): Similarly, the recall for the negative class is identical at 1.00 in both environments. This means that the model successfully identified all actual instances of the negative class.

- F1-score (no): The F1-score for the negative class is also identical in both environments, standing at 0.62. The F1-score balances precision and recall, offering a measure of the model's overall performance for the negative class.

- Precision (yes), Recall (yes), and F1-score (yes): For the positive class (yes), the Logistic Regression model in both environments yields precision, recall, and F1-score values of 0.00. These values indicate that the model failed to correctly predict any instances of the positive class.

- Accuracy: Both environments yield the same accuracy of 45% for Logistic Regression, which means that overall, the model is not performing well in correctly classifying instances from both classes.

**Random Forest Classifier (RFC):**

- Precision (no): In both the "Stock" and "Intel" environments, the Random Forest Classifier excels with a precision of 0.94 for the negative class, indicating that 94% of the negative predictions are accurate.

- Recall (no): The recall for the negative class is 0.95 in both environments, demonstrating that the model effectively identifies 95% of the actual negative instances.

- F1-score (no): The F1-score for the negative class is 0.95 in both environments, reflecting the model's excellent balance of precision and recall for the negative class.

- Precision (yes), Recall (yes), and F1-score (yes): For the positive class (yes), the Random Forest Classifier also performs remarkably well in both environments, yielding precision, recall, and F1-score values of 0.96.

- Accuracy: The Random Forest Classifier achieves a high accuracy of 95% in both environments, indicating its strong overall performance in correctly classifying instances from both classes.

### Summary Environment Comparison

In comparing Logistic Regression and the Random Forest Classifier in both environments we note the following:

- The Random Forest Classifier outperforms Logistic Regression significantly in terms of precision, recall, and F1-score for both classes (yes and no).

- Logistic Regression shows poor performance for the positive class (yes), with precision, recall, and F1-score all being 0.00, indicating it struggles to identify positive instances.

- Both models achieve identical accuracy (45%) for Logistic Regression and (95%) for the Random Forest Classifier in both environments.

**Overall, the Random Forest Classifier demonstrates superior performance, achieving high precision, recall, and F1-score values for both classes, while Logistic Regression struggles particularly with positive class predictions in both environments.**

Interestingly, the benchmark solution to this project recorded an effective Logistic Regression model. This goes to show, that given same parameter, system hardware or any other factors may contribute to the performance of any model.

# CHALLENGES, SOLUTIONS & RECOMMENDATIONS

## Challenges and Solutions

The first roadblock I faced was setting up the environment for the project. While installing Anaconda and using it was not a problem, creating and using virtual environments in python would not work.

We tried using Intel DevCloud but that did not help either. My team and I expressed our concern during regular check-in and we were guided on how to set up the environment using Git Bash. We were also made to know that low computer resources could be a reason for slow modelling on local computers.

## Recommendations

From the results of this project, it is safe to say that there is significant improvement in modelling speed with modelling intel Scikit Learn library. However, the speed actually depends on the model used and the hyperparameters used to tune the model. Different models would perform differently based on the data and other parameters set at the time of modelling. Results of each modelling process is almost not affected by any set environment.

For the models, Logistic regression behaved poorly with 45% accuracy. The model clearly could not predict positive class records. This could be for so many reasons:

1. **Model Complexity**: While Random Forest Classifier is an ensemble model combing the powers of multiple decision trees, Logistic regression learns the linear relationship between features. The ensemble model handles complexity well while the other does not understand complex relationships.
2. **Handling imbalanced Data**: Logistic regression is often sensitive to data imbalance but the ensemble nature of the RFC means that some decision trees in its ensemble can capture some positive classes. The trees that encounter the positive classes are likely to carry more votes, as RFC can assign higher importance to underrepresented classes. When logistic regression model encounters very few members of a class, it has difficulties recalling that class.
3. **Hyperparameter Tuning**: The Random Forest Classifier is optimized through hyperparameter tuning, which can significantly enhance its performance. Tuning parameters like the maximum depth of trees and the number of estimators made the model more effective.

Although we fine-tuned the fit\_intercept parameter and tried to scale and cross-validate using GridSearch, there are other things we can do to improve the performance of the Logistic Regression model. They include:

1. **Feature Selection**: Reevaluate the features included in the model. It's possible that some features are not informative or are causing the issue. Feature selection techniques like recursive feature elimination or feature importance analysis can help to identify which features are contributing to the problem.
2. **Resampling Technique**: Logistic regression model failed to identify the positive class which means that it may have encountered very few of them. To mitigate this, resampling techniques like SMOTE (Synthetic Minority Over-sampling Technique) to create synthetic examples of the minority class. This can help balance the class distribution, giving the model enough instances of those features to learn from.

# CONCLUSION

In conclusion, this project represents a comprehensive exploration of machine learning's potential in predicting customer churn, with a specific focus on the telecom industry. We used a dataset sourced from Kaggle, encompassing over 70,000 records, and employing two different Python environments for model development, we built two predictive models. Notably, our challenges and eventual solutions for different environments underscored the importance of selecting the right computational setup to achieve optimal model performance.

The significance of this project and the knowledge gained from it extends beyond its immediate scope. It underscores the critical role of predictive analytics in modern business strategy, especially within the telecom sector, where customer retention is very important. Armed with machine learning tools, businesses can proactively identify and address customer churn, thereby improving service quality and enhancing overall customer satisfaction. While this project showcased the impressive capabilities of models like the Random Forest Classifier, it also acknowledged the need for tailored solutions, as Logistic Regression exhibited limitations in handling imbalanced datasets. Ultimately, this project serves as a testament to the power of data-driven decision-making, demonstrating the potential to revolutionize industries by leveraging predictive analytics for improved customer relations and operational efficiency.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Flapmax, FAI, and Wentors for this Women in AI internship program, and for providing me with the opportunity to embark on this insightful journey. Your dedication to fostering a nurturing learning environment and challenging us to do hard things is greatly appreciated.

I extend my heartfelt thanks to my mentors, whose guidance and expertise were invaluable throughout this project. Your patience, wisdom, and willingness to share knowledge have been instrumental in shaping my understanding of machine learning and data science.

I am indebted to my fellow interns for their camaraderie and collaborative spirit. Our discussions and shared challenges and solutions enriched the learning process and made this internship a truly rewarding endeavor.

Special thanks go to Joy and Mary, my team members for their unwavering support, teamwork, and dedication. We moved from trying to find our feet to marching in strides Your contributions were essential and invaluable in the successful execution of this project.

Lastly, I would like to express my appreciation to all around me those who provided insights, feedback, and encouragement along the way. Your collective support has played a pivotal role in the completion of this project.

# References

Primary Project Task Repository: [ml classification result table - Search (github.com)](https://github.com/oneapi-src/customer-churn-prediction)

Dataset Repository: <https://www.kaggle.com/datasets/mehmetsabrikunt/internet-service-churn>

<https://towardsdatascience.com/visualize-machine-learning-metrics-like-a-pro-b0d5d7815065>

<https://www.researchgate.net/publication/357054206_CUSTOMER_CHURN_PREDICTION_ANALYSIS_IN_A_TELECOMMUNICATION_COMPANY_WITH_MACHINE_LEARNING_ALGORITHMS>