Singapore Institute of Technology - University of Glasgow
Joint Degree in Computing Science Degree Programme

CSC3001 Capstone Project

Please complete the following form and attach it to the Capstone Report submitted.

**Capstone Period:** **06 Sep 2021** **to** **22 Jul 2022**
**Assessment Trimester:** **Final Trimester**
**Project Type:** **Academic**

**Academic Supervisor Details:**

Name: **Vivek Balachandran**
Designation: **Asst/Professor (ICT Information Security)**
Email Address: **vivek.b@SingaporeTech.edu.sg**
Contact Number: **96302329**

**Student Particulars & Declaration:**

Name of Student: **Tan Rui Han Ivan**
Student ID: **1901832**

I hereby acknowledge that I have engaged and discussed with my **Academic Supervisor** on the contents of this Final Capstone Report and have sought approval to release the report to the Singapore Institute of Technology and the University of Glasgow.

*Signature*

**Date:** **22 Jul 2022**

**END OF FORM**


Singapore Institute of Technology - University of Glasgow
Joint Degree in Computing Science Degree Programme

# Final Capstone Report
# “Credit Scoring for Loans”

For **Final Trimester** from 06 Sep 2021 to 22 Jul 2022

*Tan Rui Han Ivan*
*Student ID: 1901832*

Academic Supervisor: *Vivek Balachandran*

Submitted as part of the requirement for CSC3001 Capstone Project

# Table of Contents


1 Introduction 1

1.1 Capstone Motivation 1

1.2 Purpose of Capstone 1

1.3 Problem Formulation 2

1.3.1 Sub-Problem 1: Determining Data Preparation Needs of Raw Loan Dataset 2

1.3.2 Sub-Problem 2: Lack of LR Model Tuning and Feature Scaling Inclusions 2

1.4 Project Objectives/Project Specifications 3

2 Literature Review 3

2.1 Determining Data Pre-Processing Needs (Sub-Problem 1) 4

2.1.1 Cleaning of Dataset (Benefits) 4

2.1.2. Feature Selection (Benefits) 5

2.1.3. Weight of Evidence Binning (Benefits) 6

2.2 Lack of LR Model Tuning AND Feature Scaling (Sub-Problem 2) 7

2.2.1 Lack of LR Model Tuning 7

2.2.2 LR Model Tuning (Benefits) 8

2.2.3 Feature Scaling (Benefits) 9

2.3 Determining Data Pre-Processing Needs AND Lack of LR Model Tuning and Feature Scaling (Sub-Problem 1 AND 2) 10

2.3.1 Weight of Evidence Binning AND Lack of Feature Scaling 10

2.4 Essentials: Scorecard (Post-Model Training) 10

2.4 Conclusion (Literatures Review) 11

3. Methodology / Proposed Design 12

3.1 Dataset Pre-Processing 12

3.1.1 Removing Empty Features 13

3.1.2 Splitting Training/Testing Dataset 14

3.1.3 Data Transformations Methods (on Training Dataset) 14

3.2 Feature Selection (on Training Dataset) 16

3.3 One-Hot Encoding (on Training Dataset) 17

3.4 Applying Data Transformations (on Test Dataset) 18

3.5 Weight of Evidence (WOE) and Information Value (IV) 18

3.6 WOE Binning 22

3.7 Logistic Regression Pipelines: Hyper-Parameter Tuning and Feature Scaling 24

3.7.1 Feature Scaling 24

3.7.2 LR Model Tuning (Hyper-Parameter Tuning) 24

3.8 (Credit) Scorecard 26

4. Results and Analysis 27

4.1 Results 27

4.2 Analysis 29
4.2.1 Evaluation of Feature Scaling (GAP 1) 29
4.2.2 Evaluation of LR Model Tuning (GAP 2) 29
5. Scorecard (Utilization) 31
6. Conclusion 36
7. References 36
[1] Maldonado, S., Peters, G., & Weber, R. (2020). Credit scoring using three-way decisions with probabilistic rough sets. Information Sciences, 507, 700-714. https://doi.org/10.1016/j.ins.2018.08.001 36
[2] Guo, G., Zhu, F., Chen, E., Liu, Q., Wu, L., & Guan, C. (2016). From footprint to evidence. ACM Transactions on the Web, 10(4), 1-38. https://doi.org/10.1145/2996465 36
[3] Nikolic, N., Zarkic-Joksimovic, N., Stojanovski, D., & Joksimovic, I. (2013, May 11). The application of brute force logistic regression to corporate credit scoring models: Evidence from Serbian financial statements. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S0957417413003084 36
[4] Li, Y., & Chen, W. (2020, October 13). A comparative performance assessment of ensemble learning for credit scoring. Retrieved July 22, 2022, from https://www.mdpi.com/2227-7390/8/10/1756 36
[5] Bensic, M., Sarlija, N., & Zekic-Susac, M. (2005). Modelling small-business credit scoring by using logistic regression, neural networks and decision trees. Intelligent Systems in Accounting, Finance and Management, 13(3), 133-150. https://doi.org/10.1002/isaf.261 36
[6] Tae, K. H., Roh, Y., Oh, Y. H., Kim, H., & Whang, S. E. (2019). Data cleaning for accurate, fair, and robust models. Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning - DEEM'19. https://doi.org/10.1145/3329486.3329493 37
[7] Gudivada, V. N., Apon, A., & Ding, J. (2017, July 22). Data quality considerations for Big Data and machine learning: Going ... Retrieved July 22, 2022, from https://www.researchgate.net/publication/318432363_Data_Quality_Considerations_for_Big_Data_and_Machine_Learning_Going_Beyond_Data_Cleaning_and_Transformations 37
[8] Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), 20-29. https://doi.org/10.1145/1007730.1007735 37
[9] Provost, F. (2000). Machine learning from imbalanced data sets 101. Retrieved July 22, 2022, from https://www.aaai.org/Papers/Workshops/2000/WS-00-05/WS00-05-001.pdf 37
[10] Miron, M., Tolan, S., Gómez, E., & Castillo, C. (2020, June 07). Evaluating causes of algorithmic bias in juvenile criminal recidivism - artificial intelligence and law. Retrieved July 22, 2022, from https://link.springer.com/article/10.1007/s10506-020-09268-y 37
[11] Al-Najjar, H., Pradhan, B., Kalantar, B., Sameen, M., Santosh, M., & Alamri, A. (2021, August 19). Landslide susceptibility modeling: An integrated novel method based on machine learning feature transformation. Retrieved July 22, 2022, from https://doi.org/10.3390/rs13163281 37
[12] Schratz, P., Muenchow, J., Iturritxa, E., Richter, J., & Brenning, A. (2019). Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using Spatial Data. Ecological Modelling, 406, 109-120. https://doi.org/10.1016/j.ecolmodel.2019.06.002 37
[13] Palekar, V., Ambatkar, S., Kamble, K., Ali, S., Zade, H., & Kharade, S. (2020, November 11). IRJET- credit card fraud detection using isolation forest. Retrieved July 22, 2022, from

https://issuu.com/irjet/docs/irjet-v7i3710 37

[14] L, R. N. (2018). Machine learning for the prevention and prognosis of pediatric head injury in Sport. Retrieved July 22, 2022, from https://etd.ohiolink.edu/apexprod/rws_olink/r/1501/10?clear=10&p10_accession_num=ucin1535633637793776 37

[15] Seger, C. (2018, October 26). An investigation of categorical variable encoding techniques in machine learning: Binary versus one-hot and feature hashing. Retrieved July 22, 2022, from https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1259073&dswid=-8025 37

[16] Pargent, F., Pfisterer, F., Thomas, J., & Bischl, B. (2022). Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. Computational Statistics. https://doi.org/10.1007/s00180-022-01207-6 37

[17] Urbanowicz, R., Olson, R., Schmitt, P., Meeker, M., & Moore, J. (2018, July 17). Benchmarking relief-based feature selection methods for Bioinformatics Data Mining. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S1532046418301412 37

[18] Kamalov, F., & Thabtah, F. (2017). A feature selection method based on ranked vector scores of features for classification. Annals of Data Science, 4(4), 483-502. https://doi.org/10.1007/s40745-017-0116-1 37

[19] Siddiqi, N. (2017). Intelligent credit scoring. Retrieved July 22, 2022, from https://books.google.com.sg/books?id=q-qoDQAAQBAJ&printsec=frontcover 37

[20] Baesens, B., Roesch, D., & Scheule, H. (2016). Credit risk analytics. Retrieved July 22, 2022, from https://books.google.com.sg/books?id=ornsDAAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r 37

[21] Balabaeva, K., & Kovalchuk, S. (2019, September 26). Comparison of temporal and non-temporal features effect on machine learning models quality and interpretability for chronic heart failure patients. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S1877050919311020 38

[22] Ay, O. (1970, January 01). Price prediction using Machine Learning Techniques: An application to vacation rental properties. Retrieved July 22, 2022, from http://openaccess.mef.edu.tr/xmlui/handle/20.500.11779/1702 38

[23] Zahedi, L., Mohammadi, F., Rezapour, S., Ohland, M., & Amini, M. (2021, April 29). Search algorithms for automated hyper-parameter tuning. Retrieved July 22, 2022, from https://arxiv.org/abs/2104.14677 38

[24] Mesafint, D., & H, M. D. (2021, September). Grid search in hyperparameter optimization of machine learning models ... Retrieved July 22, 2022, from https://www.researchgate.net/publication/354541720_Grid_search_in_hyperparameter_optimization_of_machine_learning_models_for_prediction_of_HIVAIDS_test_results 38

[25] A, V. B., & N, S. (2012). Building loss given default scorecard using weight of evidence bins in. Retrieved July 22, 2022, from https://www.researchgate.net/profile/Naeem-Siddiqi/publication/276956927_Building_Loss_Given_Default_Scorecard_Using_Weight_of_Evidence_Bins_in_SASR_Enterprise_Miner_Loss_Given_Default/links/555c91bd08ae86c06b5d3976/Building-Loss-Given-Default-Scorecard-Using-Weight-of-Evidence-Bins-in-SASR-Enterprise-Miner-Loss-Given-Default.pdf 38

[26] Peussa, A. (1970, January 01). Credit risk scorecard estimation by logistic regression. Retrieved July 22, 2022, from https://helda.helsinki.fi/handle/10138/163135 38

[27] Tae, K. H., Roh, Y., Oh, Y., Kim, H., & Whang, S. E. (2019, June 01). Data cleaning for

accurate, fair, and robust models: Proceedings of the 3rd International Workshop on Data Management for end-to-end machine learning. Retrieved July 22, 2022, from https://dl.acm.org/doi/10.1145/3329486.3329493 38

[28] Vannoy, T. C., Scofield, T. P., Shaw, J. A., Logan, R. D., Whitaker, B. M., & Rehbein, E. M. (2021, October). Detection of insects in class-imbalanced Lidar Field Measurements. Retrieved July 22, 2022, from https://ieeexplore.ieee.org/abstract/document/9596143/ 38

[29] Bonev, B. (2010, June 29). Feature selection based on information theory. Retrieved July 22, 2022, from http://rua.ua.es/dspace/handle/10045/18362 38

[30 Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., . . . Guevara, N. (2015, December 01). Experimental study with real-world data for Android App Security Analysis Using Machine Learning: Proceedings of the 31st Annual computer security applications conference. Retrieved July 22, 2022, from https://dl.acm.org/doi/abs/10.1145/2818000.2818038 38

[31] Abdou, H. A., & Pointon, J. (2011, April). (PDF) credit scoring, statistical techniques and evaluation criteria: A ... Retrieved July 22, 2022, from https://www.researchgate.net/publication/220613924_Credit_Scoring_Statistical_Techniques_and_Evaluation_Criteria_A_Review_of_the_Literature 38

[32] Louzada, F., Ferreira-Silva, P., & Diniz, C. (2012, February 01). On the impact of disproportional samples in credit scoring models: An application to a Brazilian bank data. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S0957417412001522?ref=cra_js_challenge&fr=njs 38

[33] Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., . . . Guevara, N. (2015, December 01). Experimental study with real-world data for Android App Security Analysis Using Machine Learning: Proceedings of the 31st Annual computer security applications conference. Retrieved July 22, 2022, from https://dl.acm.org/doi/abs/10.1145/2818000.2818038 38

[34] Siddiqi, N. (2017, January). Intelligent credit scoring. Retrieved July 22, 2022, from https://books.google.com.sg/books?id=q-qoDQAAQBAJ 39

[35] Cao, N. T., Tran, L. H., & Ton-That, A. H. (2021, December). Using machine learning to create a credit scoring model in banking and Finance. Retrieved July 22, 2022, from https://ieeexplore.ieee.org/abstract/document/9718414 39

[36] Książek, W., Gandor, M., & Pławiak, P. (2021, May 11). Comparison of various approaches to combine logistic regression with genetic algorithms in survival prediction of hepatocellular carcinoma. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S0010482521002250 39

8. Knowledge and Training Requirements 39

8.1 Applicable Knowledge from the Degree Programme 39

8.2 Additional Knowledge, Skillsets, or Certifications Required 39

Appendices 1

Appendix A: 1

## Acknowledgments

This project would not have been possible without the support of many people. I would like to thank my Academic Supervisor, Professor Vivek Balachandran, for his guidance, patience, encouragement, and committed assistance during my Capstone project. I would also like to thank my Capstone Examiner, Professor Sye Loong Keoh, for his patience and constructive Feedback during the Middle Capstone Interview with regards to this project which has allowed me to understand the significance of the additions needed for this Capstone. I also wish to thank Asad Mumtaz for answering my queries while providing the base code work need for this Capstone to be built on top of, allowing me to implement the Missing GAPs identified in Credit Scoring Research Studies.

# 1 Introduction

## 1.1 Capstone Motivation

This Capstone's environmental scope involves one of the Company's many services, their "Loan Plan" service, which is provided to new eligible customers seeking a monetary loan. Before granting such loans, the Company would first need to evaluate all new applicants based on their submitted loan registration details, such as loan history, assets currently available (e.g. Owned/Rented Properties), and more. These details allow the Company to tabulate the probability of new applicants defaulting on their applied loans while referencing the Company's historical loan lending data of all existing customers to date with their loan outcomes (i.e. whether they have defaulted or not). From there, the Company can then grant customers displaying lower probabilities of defaulting on their requested loans while considering the Company's currently available funding for provisioning the new loans.

This prediction process is known as Credit Scoring, which is essential to companies providing (monetary) loans, where "approving… clients who fail to repay, may lead to affordable losses in the best case but may also jeopardize a lender's business" [1]. Likewise, due to "the volatile and dynamic nature of the personal credit scoring problem… [it] is extremely hard to predict" [2] and would be overbearing, especially when done through manual human interventions. Therefore, by digitalizing and automating an efficient Credit Scoring System, the Company would be able to reduce any unnecessary operating expenditure(s) caused by human intervention (i.e. Credit Scoring mistakes, slow Score tabulations, and other human-related errors).

## 1.2 Purpose of Capstone

This Capstone aims to provide a tangible, real-world means of creating a predictive Credit Scoring System for the Company to use in their daily operations, all while maximizing its Scoring efficiency through Machine Learning (ML) Algorithms.

This system allows the Company to determine if new loan applicants are eligible using a trained Machine Learning Model based on the Company's customer-based Loan Transactions with the Defaulting Outcomes (i.e. whether customers had defaulted or not) during the Loan Plan's operations. Despite it being possible to curate a logical, algorithmic system for such a purpose, the Company has personally requested the use of Machine Learning, particularly the Logistic Regression (LR) algorithm, while using a measurable Scorecard for more straightforward Credit Scoring interpretations.

It should be noted that this Capstone aims to continue building on the research done by Nikolic et al. [3] by adapting the Brute Force (model) Tuning Approach for LR to improve upon the studies by Li and Chen [4] and Bensic et al. [5] which excluded LR Tuning during their experiments. In addition, the Capstone also aims to challenge the same Studies [[3],[4],[5]] which had not included the use of Feature Scaling as a means of improving LR performance.

As such, this Capstone focuses on improving LR (model) efficiency through Model Tuning and Feature Scaling Approaches, together with producing and utilizing a Scorecard with the relevant evaluation metrics, after which anything else will be excluded within this project's scope.

### 1.3 Problem Formulation

The feat of procuring a Credit Scoring System while maximizing Scoring accuracy for the Company's daily use **(Section 1.1)** can be broken down into two sub-problems: (**Sub-Problem 1**) Rectifying the Raw and Noisy Loan Dataset; and (**Sub-Problem 2**) Maximizing the LR model's performance using Model Tuning and Feature Scaling, previously omitted by the 3 studies **(Section 1.2**).

Correcting the Loan Dataset prior to its use with the LR model and maximizing the model's performance through Tuning and Feature Scaling can significantly reduce future Credit Scoring errors. This is a much-needed solution in response to the Company's existing operating expenditures caused by human-related Credit Scoring mistakes/errors (**Section 1.2**).

#### 1.3.1 Sub-Problem 1: Determining Data Preparation Needs of Raw Loan Dataset

As with all raw datasets, including Loan-related ones, there is a high possibility that they may contain redundant data elements that could skew the accuracy of the ML model in question. Such data elements are known as "outliers in [a] training dataset [which] can cause instability or non-convergence… [while] incomplete, inconsistent, and missing data can lead to drastic degradation in [model] prediction" **[7]**. Therefore, datasets are essential when training ML Models since "machine learning is only as good as its data, and no matter how good the training algorithm, the ultimate problem may lie in the data itself", which places much importance on correctly processing any dataset to reduce the possibility of avoidable degradative Model performance.

Standardized data preparation techniques can thus be applied to most datasets, including Loan-related ones. A dataset may also contain a "class imbalance problem… [which] is often reported as an obstacle to the induction of good classifiers by Machine Learning (ML) algorithms" **[8]** and "a common [data preparation technique] for dealing with [such] imbalanced data sets is to rebalance them artificially… [through] "upsampling" (replicating cases from the minority) and "downsampling" (ignoring cases from the majority)" **[9]**. Another typical raw dataset issue involves the need to "remove any undesired numerical relation between the [dataset's] categories… for a single feature" **[10]** which can be resolved through the use of "one-hot encoding transformation [which was found to be] more valued when it comes to categorical condition factors" **[10],[11]**.

However, with such promising preparation techniques, not all of them should be implemented. Upon closer inspection of the respective dataset's raw state, it would reveal that only specific data preparation technique(s) are needed to boost its usability with the LR Model later. Thus, the need to identify and resolve the dataset's respective data-cleaning need(s) is worth working on, as it reduces the degradative impact on Model performance.

#### 1.3.2 Sub-Problem 2: Lack of LR Model Tuning and Feature Scaling Inclusions

However, it was observed that no specified hyper-parameter tuning options were made to the LR model during such experiments. These experiments' LR Models resorted to using their default untuned state(s), which failed to reveal the LR Model's potentially positive tuning impact on performance since "default hyperparameter settings cannot guarantee an optimal performance of machine-learning techniques" **[13]**. This sparks the rise of two GAPs in the Credit Scoring research found **[3]**,**[4]**,**[5]** (further mentioned in Section 2.2) , which this Capstone builds upon. These Gaps reveal how LR Models' Credit Scoring accuracy could have been further maximized should there be an implementation of (**Gap 1**) Feature Scaling Techniques and (**Gap 2**) LR Model Tuning.

Therefore, the need to work on utilizing Tuning Options with Feature Scaling is worth working on as it ensures that the Credit Scoring System outperforms Scoring done manually by humans in terms of Model Scoring accuracy.

### 1.4 Project Objectives/Project Specifications

The outline of the deliverables (found in **Section 3** and **Section 4**) for this Capstone is as follows; (1) To resolve the problems posed by the given (Credit Scoring) Loan dataset, prior to training our ML Model with it, by initializing several data preparation techniques based on our dataset's initial raw state; after which, (2) the resulting transformed dataset would be analyzed and then separated into relevant categories using Weight of Evidence (WOE) Binning; before (3) setting up and training four LR Model Pipelines, with the "binned" dataset, which would help exhibit the positive effects of Model Tuning and Feature Scaling as compared to without them; and in order to provide the Company with a tangible understanding of the Credit Scoring System generated (4) assess the accuracy/efficacy of the four Model-based Scorecards through various testing metrics (i.e AUROC and GINI Coefficient, AUC/ROC Curves, Confusion Matrixes); before finally (5) providing a walkthrough on how the ideally chosen LR model can be utilized to create a Scorecard for the Company's daily operations (found in **Section 5**).

## 2 Literature Review

This section covers the Literature review of both the techniques and the research on the two Gaps to be tackled/built on by this Capstone (i.e. Lack of LR Model Tuning and Feature Scaling).

Each ***section***, e.g '*2.1 Determining Data Pre-Processing Needs (Sub-Problem 1)*', tackles one/more of the *Sub-Problem (Number)* that the Capstone aims to investigate and rectify with the relevant Literature(s) discovered.
As a note, the '*Essentials: Scorecard (Post-Model Training)'* **section** does not tackle any *Sub-Problem(s)*. Instead, it contains essential Literature(s) covering techniques that ensure the **wholeness/wrap-up** of this Capstone through the transformation of the ML Model/Pipeline into a usable Credit Scoring Scorecard for the Company's daily operations.

Each respective *section* has a ***sub-section***, e.g. ***2.1.1 Cleaning of Dataset (Benefits)***, that contains either the technique's **benefit(s)** with/without downsides or the **GAP(s**) in the Literature (mentioned in **Section 1.3**) to be built-on by this Capstone.

The Outline for Each *sub*-section consists of a summary of one research article cited by [ ] in the format of: (1) the study's ***Focus/Aim***, (2) its ***Methodology***, (3) ***Conclusion***, and (4) ***Relevance*** in the area covered by the Capstone with justifications for the respective rejection/acceptance, as purposed by this Capstone's aim(s).

Lastly, the Conclusion (in **Section 2.4**) encapsulates the Literature findings. It discusses the Relevant Techniques to be adopted for implementation, Outlining the Capstone's direction for the remaining sections (i.e. *Methodology*, *Results and Analysis*, and more).

## 2.1 Determining Data Pre-Processing Needs (Sub-Problem 1)

### 2.1.1 Cleaning of Dataset (Benefits)

Palekar et al. [13] aimed to identify credit card transactions flagged as high fraud risk through explorational analysis using Machine Learning (ML). They pre-processed their dataset before training/testing with the ML Isolation Forest (IF) algorithm, which began with data exploration via a graph and dataset analysis, before removing null/empty values and splitting their transformed data into training/splitting sets. By doing so, the resulting "entries are provided randomly to amplify the quality and thus more realistic data is generated". Next, IF was trained and predicted classifications before analyzing any Correlated Values with a heatmap, value amounts with a histogram, and finally evaluating Model performance via confusion matrixes. It concluded with IF being ideal for mass-scale deployment for fraudulent detection.
Although the IF algorithm was utilized, the study still proved how these data pre-processing methods: visual chart analysis of the dataset, removing null data points and splitting the resulting dataset into training/validation sets with randomization techniques could further reduce biases and boost data quality when being fed later into the LR model.

Richards et al. [14] focussed on the positive effects and obstacles faced when using Machine Learning to prevent and predict concussions requiring extended recovery time, particularly in sports. It began with data preparation by removing outliers and applying Cross-Validation (CV) with stratification (80:20, Training: Testing) splitting to prevent sampling biases. The ML models' (Logistic Regression, Support Vector Classification, Deep Neural Network) Hyper-Parameters were tuned before evaluating their performance with metrics such as; Accuracy, AUROC (Area under the Receiver Operating Characteristics Curve) and F1 Score. It concluded that Deep Neural Network (DNN) outperformed the interpretable Logistic Regression and Support Vector Machines revealed through the applied performance metrics. It warned against trusting DNN's black-box models which are uninterpretable, despite the interpretable ones' (Logistic Regression, Support Vector Classification) having weaker predictive powers.
As such, despite DNN superseding other ML models, including LR, through performance metric testing, the study also noted that DNN's closed architecture made it unfit for this Capstone project's requirements since it requires an interpretable scoring mechanism for the Company to evaluate openly. This study's implementation of randomizing the dataset through removing outliers and utilizing stratification to prevent sampling biases is like that of [13], giving weight to these data preparation techniques in removing unnecessary biases found originally in the dataset.

Seger [15] aimed to compare the performance of ML models utilizing various Feature Encoding strategies (One-Hot, Binary, Feature Hashing). Using Area Under Curve (AUC) and Precision-Recall (PR) as evaluation metrics, it graded performance on the models such as Artificial Neural Networks (ANN) and LR Models. It concluded that One-Hot Encoding has more dimensional vectors than other encoders, achieving the highest evaluation scores for each Model. Lastly, the study's future improvements include using Cross-Validation and increasing the number of (model) epochs/rounds

while using more varied datasets. As a possible future enhancement, it suggested investigating the encoders' execution speed and encoding only data with high cardinality features.
Testing various Feature Encoding strategies, such as One-Hot Encoding, proved to be most efficient through various testing metrics, such as AUC and PR. This made it the most suitable among the models mentioned above, including LR, which is also part of our intended use case for this Capstone.

Pargent et al. [16] aimed to evaluate various encoding techniques (One-Hot, Dummy, Hash, Integer, and more) on five ML Algorithms (some of which include LR and K Nearest Neighbour). Each algorithm was given specific encoders based on the available High Cardinality Threshold, except for the control condition(s). Their Hyper-Parameters were tuned with 5-fold Cross-Validation where performance evaluation was determined through; Root Mean Squared Error (RMSE), Area Under Curve (AUC), and AUNU (i.e. the multiclass version of AUC). The results revealed that One-Hot encoding outperformed dummy encoding by a slight difference but concluded that target encoding still outperformed all encoders, despite its longer execution timeframe.
Target Encoding was performed on five ML algorithms with encoding strategies, and Model Tuning configurations performed the best overall in terms of performance. However, One-Hot encoding showed to overtake Dummy Encoding after being verified with evaluation metrics such as RMSE, AUC and AUNU. Despite Target Encoding outperforming One-Hot Encoding, One-Hot Encoding overtook Dummy Encoding in terms of performance. One-Hot Encoding was also well-used in the Data Science Space, as resounded by the other study [15]. It, therefore, proves to be effective for this Capstone in reducing biases within the Credit Scoring dataset.

### 2.1.2. Feature Selection (Benefits)

Urbanowicz et al. [17] focussed on comparing existing feature selection algorithms, including Chi-Squared, ANOVA F-Test and Information Gain (IG), against their proposed Relief-Based Algorithm (RBA) for Feature Assessment while using the Decision Tree model for Modern Biomedical Data Mining.
Despite mainly focussing on various implementations of RBAs, it also included the other Feature Selection tools (i.e. Chi-Squared, ANOVA F-Test, IG) were all were each evaluated on their ability to accommodate differing data type-related issues, such as Binary/Multi-Class Classification, Discrete/Continuous/Mixed Feature types, missing data, and Class Imbalances.
It mentioned that the Chi-Squared test was limited to discrete variables. In contrast, ANOVA F-Test allowed multi-class (including non-discretized ones). It found that Chi-Squared and ANOVA F-Test successfully recognized the main features within the dataset, except for cases when the data consisted of missing instances. However, it concluded that these two feature selection algorithms failed to compare with their better-performing Relief-Based Algorithm on generalized problem types.
Despite RBA outperforming the other two Feature Selection Algorithms, it is still considered a recent publication in 2018 where more research on this newly proposed algorithm is needed. With that, the well-researched Chi-Squared Algorithm allows for selecting discrete variables, while ANOVA F-Test allows both Discrete and Continuous

ones. Given that both successfully identified the dataset's main Features, this Capstone should thus utilize these two Algorithms to recognize relevant Features within the dataset while discarding the unnecessary ones.

Kamalov and Thabtah [18] propose a new method of normalizing and combining Scores of Feature Selection Methods, such as Information Gain (IG), Chi-Squared and Correlation Feature Selection (CFS), to maximize Score stability without diminishing the accuracy of the Predictive Classification model, as compared to solely using methods like IG, Chi-Squared or CFS. The study noted the use of various Classification Models attained through Java's (ML) WEKA Software.
The study intensively focused on their proposed RBA configuration methods, which are not limited to the creation of Vector Scores from the combination of IG and Chi-Squared before utilizing them with CFS to attain a more robust Feature Selection (Ranking) measure. The RBA, IG and Chi-Squared methods were also each tested separately in conjunction with the predictive models available in WEKA, such as eDRI and PART.
As a result, all individual methods (Chi-Squared, IG, CFS) had shown improvements in data dimensionality reduction and classification performance. However, it was found that RBA had reduced feature size while retaining classification accuracy. The study concluded that RBA helped reduce the number of rules in rule-based Classification models without negatively affecting classification power, unlike the previously mentioned individual method(s).
Like [17], this 2017 study with their proposed method was recently published; thus, more research would be needed to verify its efficacy, despite having performed better than Chi-Squared, IG and Inter-Correlation in this study. The study has also acknowledged previous scholars' wide applications/evaluations of the individual methods, such as Chi-Squared due to its reaffirmed efficiency/credibility in the industry, especially when supported by the other study [17].

### 2.1.3. Weight of Evidence Binning (Benefits)

Siddiqi's book [19] covered various Credit Scoring Techniques. Some notable guidelines include applying WOE Binning to include missing values, having each binned Feature within the dataset account for at least five percent of all observations while having at least one (data) record of the defaulting or non-defaulting type within each Binned Feature served as helpful guidelines to WOE Binning. Lastly, Saddiq mentioned that the number of outlined attributes should be restricted to prevent Model overfitting.
Likewise, to prevent the down-sampling of records in the dataset due to missing values, WOE Binning could potentially be useful for processing the missing components/Features within this Capstone's dataset.

## 2.2 Lack of LR Model Tuning AND Feature Scaling (Sub-Problem 2)

### 2.2.1 Lack of LR Model Tuning

Bensic et al. [5] intended to compare the Credit Scoring prediction efficiency of a few Machine Learning Models (i.e. LR, Neural Networks, Decision Trees) trained on a business credit lending dataset. It tuned each model's Hyper-Parameters, except for that of LR. The Decision Trees' Model tuning began by "pruning on the misclassification error as the procedure for selecting subtrees". The four Neural Networks (NN) Models' tuning involved optimizing their "learning rate and the momentum for the EDBD learning rule... set" while having overtraining "avoided by [including] the 'save best' cross-validation procedure". It concluded with a justification that their experiment was "dealing with small datasets [of 160 records] and a large number of [30] variables" prevented the Model Tuning of LR and thus hindered it "from being successful". As a result, NN showed more success through increased hit rates, and the study's future enhancement suggested that more data and NN algorithms be included.
In conclusion, this study not only excludes Feature Scaling within their Credit Scoring dataset but by leaving their LR Model untuned. However, it still concluded LR's inefficiency when compared to NNs, with their respective justifications with the dataset. As such, this study (mentioned in **Section 1.3.2**) would be built upon in this Capstone, with the addition of model Tuning the LR with the respective Feature Scaling methods to boost its efficacy compared to without such methods.

Li and Chen [4] focussed on comparing the model performance of various ensemble and baseline classifiers, including LR, when using a Credit Scoring dataset from the US-based Lending Club™. It used Grid Search and pre-defined search spaces to determine the models' optimal Hyper-Parameters. However, Logistic Regression only had its C values tuned, omitting most of LR's available parameter spaces (observed in '**Table 1**'). It employed 5-fold Cross-Validation to resist overfitting the Models and split the dataset into an 80:20 (Training: Testing) ratio. Their chosen evaluation metrics included Area under Curve (AUC), Kolmogorov-Smirnov (KS) statistic, Brier Score, Accuracy and Model procedure time. It concluded that LR outperformed other classifiers on most evaluation metrics, although Random Forest attained the best results on all metrics employed. It noted that a future enhancement should include more credit scoring data with the potential of using Bayesian Hyper-Parameter Tuning.
Like [5], with the omission of Feature Scaling within this study's Models, the study had selectively Model Tuned LR's parameters (with only its 'C' value) while excluding the rest within it. Despite this, LR could still overtake other classifiers in the study, despite having the Tuned NN Model perform the best overall. As such, this study (as mentioned in **Section 1.3.2**) would be built upon in this Capstone by including more of LR's parameters when Model Tuning, with the inclusion of Feature Scaling, to assist in enhancing LR's performance with Credit Scoring data.

### 2.2.2 LR Model Tuning (Benefits)

Mesafint and Manjalah [24] focussed on comparing the effects of the 'Brute-Force' Grid Search Hyper-Parameter Optimization (GSHPO) Algorithm on 8 ML Models (i.e. LR, RF, Decision Tree, K-Nearest Neighbour, Support Vector Machine, Extra-Tree, AdaBoost, Gradient Boosting) to predict the outcomes of HIV tests from a dataset by the Ethiopian Demographic and Health Survey. The study was split into two 'phases'; the first implemented GSHPO on all ML Models, while the second acted separately as a control setup, by having all 8 Models un-optimized (i.e. using Default Hyper-Parameter Values). First, data Pre-Processing techniques were applied (i.e. identification and removal of incorrect/duplicated data points, imputation of any missing values, using Min-Max Feature Scaling, utilizing Feature Selection, splitting the dataset into an 80:20 Train: Test configuration). From there, GSHPO tuned all eight models' Hyper-Parameters, including LR which had some parameters for tuning such as 'penalty, 'C', and 'Max Iterations'. The models had 10-Fold Cross-Validation implemented to resolve any biases during model training. The models were then evaluated with performance metrics (i.e. Accuracy, Precision, Recall, F1-Score, AUC-ROC, Mean-Absolute-Error or MAE, Root-Mean-Squared-Error or RMSE, R-Squared, Confusion Matrix plots). The resulting models' performances were shown in Table 4 (Untuned) and Table 5 (Tuned), where LR with GSHPO showed better overall performance metrics than without the Optimizer. In contrast, the remaining ML models showed improved/equal performance with GSHPO. The study concluded that the GSHPO technique could increase Model prediction power and as future research, to use more extensive datasets and/or more demanding Algorithms.

From the results, Grid Search Optimization proved efficient in determining LR's optimal Hyper-Parameters, which is especially useful for the Capstone's use-case of attaining the best LR Model (as mentioned in **Section 1.3.2**).

Zahedi et al. [23] directed their approach in reinforcing the efficacy of Automated Hyper-Parameter Optimization Algorithms (i.e. Grid Search, Random Search) over Manual Search Algorithms (which require a deep understanding of the model). This was tested against various ML models (i.e. LR, Decision Tree, Random Forest, Naïve Bayes, XGBoost, Support Vector Machine, K-Nearest Neighbour) with an education-based dataset. The study's dataset was derived from MINDFIELD™, which consisted of real-world students' educational data logs (i.e. graduation rates, 'on-time completion', and GPA). The study used a subset of the entire dataset to attain only Computing-related students and then further filtered students based on specific majors. The dataset subset was then normalized and cleaned by removing corrupt/invalid data points, including Features with more than 60% missing values. In addition, it included Feature Engineering with One-Hot Encoding to encode Categorical data values. The Pre-processed dataset was then trained on the ML models with 3-Fold Cross-Validation to prevent Overfitting. Finally, the trained and untrained models were evaluated and compared with the *Accuracy* Performance Metric. The results revealed that all ML models had improved predictive performance with Grid Search or Random Search Optimization than without (i.e. Default Parameter Values). However, it concluded that despite the improvements brought about by the Automated Optimizations, it led to high model Tuning time depending on the data scale, size of its Search Space (i.e. Specified model Parameters' Range) and the number of Hot-Encoded Features.

Despite the high time complexity required by Grid Search, the results revealed that it increased Accuracy on the LR model, which was also resonated through [24], supporting its effectiveness in the Capstone's pursuit for a more error-resistant Credit Scoring LR model.

### 2.2.3 Feature Scaling (Benefits)

Balabaeva and Kovalchuk [21] focused on determining if historical Chronic Heart Failure (CHF) datasets could improve the classification of various Machine Learning models in predicting heart failure as compared to using more recent datasets to date. The study began by pre-processing the raw data. Its scope included several types of Feature Scaling Methods, such as Standard Scalar, Min-Max, Max Abs, Robust, and Quantile Transformers. They were then applied to ML Model Algorithms (i.e. LR, Tree-based) before fine-tuning them with Cross-Validation. Finally, the Shapley Additive exPlanations (SHAP) Algorithm, a type of Co-operative Game Theory, was used to interpret the ML models' predictions. The predictions were evaluated with the Performance metric, *F1-Score* with *Macro Averaging*, to compensate for any available imbalanced classes. The results showed that of all the Feature Scaling Methods used, only the LR Model showed the best performance with Standard Scaler (with reference to the study's Table 2 labelled "Scaling methods performance for CHF identification task"). The remaining Feature Scaling Methods showed degraded/equal performance when no Feature Scaling was applied to the Tree-based Algorithms. The results showed that "Logistic regression [had] boosted its performance using [the] standard scaler [method]" while mentioning that Scaling when used with Tree-based algorithms, despite the results, may, in other cases, show improvements.
Based on the results regarding our Capstone requirements of creating a Credit Scoring Model with LR, Standard Scaler has proved to be more efficient for such a model as compared to other Feature Scaling Methods mentioned within the study.

Oğuz Ay [22] aimed at predicting Rental Property prices using ML Algorithms, such as Linear Regression, Ridge Regression, Support Vector Regression, Random Forest Regressor, Light Gradient Boosting Machine Regressor, and eXtreme Gradient Boosting Regressor. He then Hyper-Parameter tuned them with the inclusion of Feature Scaling Methods (i.e Standard Scaler, MinMax Scaler, Normalizer). After which, he evaluated their performance using Mean Squared Error (MSE) and Adjusted r-Squared Values. All ML Models resulted in the best performance (lowest Mean Squared Error and highest R-Squared Values) with Standard Scaler and/or MinMax Scaler (regarding “Table 5: Mean Squared Error Performance of Algorithms”). It concluded that Feature Scaling with LGBM regressor was the best performing algorithm, with the inclusion of Hyper-Parameter tuning.
Despite this study’s scope excluding the use of Logistic Regression (i.e. LR) which the Company requested, the experiment’s results still revealed that Standard Scalar, including Min-Max, is the most suitable for Feature Scaling Methods, even among various other ML Models.

## 2.3 Determining Data Pre-Processing Needs AND Lack of LR Model Tuning and Feature Scaling (Sub-Problem 1 AND 2)

### 2.3.1 Weight of Evidence Binning AND Lack of Feature Scaling

Nikolic et al. [3] aimed to utilize LR with Weight of Evidence (WOE) Binning for predicting Credit Scores based on five years of financial data. This dataset was then split into a (Training: Testing) ratio instead of utilizing 100% of the dataset for training which may result in the "credit scoring model… doing well on train sample, but perform poorly in practice [or test dataset]", prior to being transformed through the WOE Binning approach, by adapting Siddiqi's [19] guidelines to determine each Feature's defaulting rate/relation, which includes: 1) binning missing values separately, 2) having each bin account for at least five percent of all observations, and finally, 3) ensuring that each bin have at least one record of defaulting and/or non-defaulting type(s). Nikolic et al. [3] continued with Siddiqi's guidelines ensuring that the number of outlined bins should be as few as possible since too many of them can cause greater complexity and increase the potential for model overfitting. In contrast, Siddiqi's guidelines mentioned that too few attribute classes could also distort valuable information. After which, "manual adjustment[s] of attribute [or bin's] boundaries [known as the minimum and maximum ranges] would be required in order to preserve the economic logic of each variable [or Feature] and at the same time [help] keep the [variable's] predictive power". Nikolic et al. adhered to Siddiqi's WOE Binning guidelines by binning the values and then training them through a brute-force approach (like Grid Search Hyper-Parameter Tuning). They attained 14 million different LR Model parameter configurations before finally settling on the best performing one through Model evaluation metrics such as ROC (i.e. defining the relation between the Model's TPR and FPR) and AUC. The study concluded that there were improvements in predictive powers with the implementing of WOE data transformations, which separated the financial dataset into the relevant sections, effectively eradicating problems caused by unique values and/or outliers within the Feature-filled dataset.

Although the results did not utilize Feature Scaling within the dataset, it was noted that its dataset contained "financial ratios" that align with the Credit Scoring scope as with this Capstone. With the improvements in the ideal LR Model's predictive power despite the need to generate millions of LR Model combinations using Grid Search Optimization, this study (mentioned in Section 1.3.2) could be adapted and built upon through this Capstone. This adaptation could include the preparation of the Capstone's Credit Scoring dataset through WOE Binning to counter any invalid/problematic values/outliers. Afterwards, the WOE Binned values could be used to train and attain the best LR Model parameters through the brute-force Model Tuning approach. From there, evaluation metrics such as ROC and AUC could help determine the most accurate LR Model overall.

## 2.4 Essentials: Scorecard (Post-Model Training)

Berkel et al. [25] aimed to explain how a Scorecard (for Loan Default detection) is formed using WOE Binning. The study explained how the Scorecard "serves to increase [the] understanding of not just what is predictive [such as the Model's Risk-Ranking abilities], but additionally, how they are predictive" in allowing "the user to validate existing policy rules and strategies".

It concluded that the scorecard can be easily examined, understood, and utilized due to its binned structure which makes it "business friendly" as "compared to more complex [ML] models", such as uninterpretable Neural Networks.
Therefore, with an interpretable Scorecard implemented within the Capstone, the Company can validate the ML Model's Scoring decision(s) through the scorecard when scoring their new Customers (i.e. Loan Applicants).

Peussa [26] aimed to explain the theory/implementations of Scorecards followed by the description of measurable analysis techniques. Using the study's experimental results, he replicated the process with a dataset from a consumers' credit bank and an LR Model. The study mentioned that the 'credit scorecard is a statistical model, which predicts a probability of default for an applying customer with certain characteristics'.' In contrast, Credit Scoring is based on the 'behaviour of previously accepted accounts… to determine the initial credit limit a customer is offered', as derived from the customers' Loan application form, which contained their respective particulars. The LR model was trained with the dataset and then evaluated for performance using metrics (i.e. GINI Coefficient, Akaike Information Criterion, Bayesian Information Criterion). Finally, he creates the Scorecard using the Model's estimation coefficients before clustering and validating the 800 data observations via a dot plot. It concluded with the study's generated Model's resulting slow (significant) coefficient estimate being less than three percent. This decline mentioned was concluded to be due to a "poor [dataset] quality" where the "unrestricted responses" (i.e. Typos, misunderstanding of questions within the Application Form) resulted in a lack of informative/credible effects. Despite the dataset's biases given by its unstructured ambiguity, the theory/implementation process (i.e. the use of LR's Coefficients to create a Scorecard) could provide a helpful start in implementing an interpretable Scorecard within this Capstone. Likewise, a better structured (quality) dataset with the necessary transformations is also necessary.

### 2.4 Conclusion (Literatures Review)

Research, related/unrelated to Credit Loans, have shown to have employed dataset pre-processing techniques such as exploratory Dataset Analysis [13], removing missing/outlier values [13],[14],[23], performing One-Hot Encoding of Categorical Values within the dataset [15],[16],[23], and Random Stratification [13],[14] when splitting the dataset into training and testing sets [3],[4],[13],[14],[24].

After which, commonly implemented Feature Selection algorithms, such as Chi-Squared [17],[18] and ANOVA F-Test [17] are shown to assist in identifying useful Categorical and Continuous Features, respectively, in preparation for WOE Binning in the later section.

After selecting important features, Weight of Evidence (WOE) Binning can be used to bin specific variables together based on their IV and Total Number of Observations while handling missing Values/Value-Ranges [3],[19] without affecting the later processes (i.e. Model Training/Testing, Scorecard).

After binning the significant features, Feature Scaling research points out that Standard Scaler, prior to LR Model training, can aid in improved Model performance [21],[22], which was left out of Credit Scoring studies [3],[4],[5].

Next, with the positive effects of Cross-Validation [4],[5],[14],[16],[21],[23],[24] applied to the LR model training, together with Hyper-Parameter Tuning through Grid Search, this may boost LR's performance [23],[24] should it be tuned properly [4],[5].
The LR Model(s) can then be evaluated with performance metrics such as AUROC

[[14],[24]], ROC Curves [3],AUC [[3],[4],[15],[16]], Confusion Matrix Plots [24], and GINI Coefficient [26].

Lastly, a Scorecard can be extracted after the dataset has been WOE Binned and trained with an LR model. This is can be helpful to companies looking to understand the Scoring done by the ML Model [[25],[26]], through a transparent and interpretable Scorecard.
In summary, the Company's Credit Scoring goals can be attained through Feature Selection prior to WOE Binning and be later fed into an LR Model with the Grid Search model tuning. The various models generated will then be evaluated to find one with the highest *Accuracy* before being extracted as a Scorecard for the business to utilize for daily operations.

## 3. Methodology / Proposed Design

With the Literature Review(s) mentioned above consisting of Research done with regards to the effectiveness of various types of Credit Scoring System Implementations currently available, the *Methodology* section includes detailed implementation(s) of the agreed methods, decided on through the said Literature Review(s). Furthermore, it should be noted that the Research Citations within this section covers previously/newly cited articles and/or resources to further expound on the technical implementations needed for the respective methods previously decided from within the *Literature Review* section.
The outline of this section would thus traverse through the previously agreed methods within the *Literature Reviews* section, such as (3.1) Data Pre-Processing of the Raw Dataset; together with the determining of the Dataset's Feature relevancy through (3.2) Feature Selection tools followed by having its Training Dataset undergoing (3.3) One-Hot Encoding before (3.4) Applying the previously mentioned Data Transformations on the Test Dataset. After which, both Datasets will proceed to be (3.5) Analysed through Weight-Of-Evidence and Information Value to determine their respective Predictive Powers and finally be (3.6) Weight-Of-Evidence Binned. The next step would require the tackling of the two previously identified GAPs from Research [3],[4],[5] through the initial (3.7) Preparation of the respective LR Model Pipelines containing individual/collective aspects of Feature Scaling and/or Hyper-Parameter Tuning (i.e. model tuning). After which, a (3.8) Walkthrough of the technical understandings and procedures of creating a Credit Scorecard will be done so that the Company can build its own Scorecard.
The Software Tools/Libraries used in this Capstone Implementation include Uses: Jupyter Notebooks, Python, Pandas, Numpy, Sklearn, Scipy and Matplotlib.

### 3.1 Dataset Pre-Processing

The dataset utilized for the following methodological experiment(s) was attained from the LendingClub™ (https://www.lendingClub.com), a US P2P Lender, containing 2 million records of funded loans with 145 Features (Columns) from the year 2007 to Quarter 2018. This dataset was chosen due to its relevancy towards the Capstone's purpose of Credit Scoring real-world Loans. A Subset of Dataset Values can be seen in **Figure 5**. From here, Exploratory Dataset Analysis [13] can begin on the given Dataset.

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | installment | grade | sub_grade | emp_title | emp_length | home_ownership | annı |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | NaN | 2500 | 2500 | 2500.0 | 36 months | 13.56 | 84.92 | C | C1 | Chef | 10+ years | RENT | 5 |
| 1 | NaN | NaN | 30000 | 30000 | 30000.0 | 60 months | 18.94 | 777.23 | D | D2 | Postmaster | 10+ years | MORTGAGE | 9 |
| 2 | NaN | NaN | 5000 | 5000 | 5000.0 | 36 months | 17.97 | 180.69 | D | D1 | Administrative | 6 years | MORTGAGE | 5 |
| 3 | NaN | NaN | 4000 | 4000 | 4000.0 | 36 months | 18.94 | 146.51 | D | D2 | IT Supervisor | 10+ years | MORTGAGE | 9 |
| 4 | NaN | NaN | 30000 | 30000 | 30000.0 | 60 months | 16.14 | 731.78 | C | C4 | Mechanic | 10+ years | MORTGAGE | 5 |

*Figure 5: Subset of Feature Values (within Dataset)*

### 3.1.1 Removing Empty Features

Before performing Feature Selection on the data, some data pre-processing is needed with the current dataset. On closer examination, several variables with more than 80% null values are present (as seen in **Figure 6**). The values range from 0 to 1, where 0 equates the Feature to having 0% null values while 1.0 reveals that it contains 100% null values.

| | |
|---|---|
| id | 1.000000 |
| member_id | 1.000000 |
| url | 1.000000 |
| desc | 0.944235 |
| mths_since_last_record | 0.841128 |
| annual_inc_joint | 0.946604 |
| dti_joint | 0.946606 |
| verification_status_joint | 0.948807 |
| revol_bal_joint | 0.952218 |
| sec_app_earliest_cr_line | 0.952217 |
| sec_app_inq_last_6mths | 0.952217 |
| sec_app_mort_acc | 0.952217 |
| sec_app_open_acc | 0.952217 |
| sec_app_revol_util | 0.953030 |
| sec_app_open_act_il | 0.952217 |
| sec_app_num_rev_accts | 0.952217 |
| sec_app_chargeoff_within_12_mths | 0.952217 |
| sec_app_collections_12_mths_ex_med | 0.952217 |
| sec_app_mths_since_last_major_derog | 0.984101 |
| hardship_type | 0.995305 |
| hardship_reason | 0.995305 |
| hardship_status | 0.995305 |
| deferral_term | 0.995305 |
| hardship_amount | 0.995305 |
| hardship_start_date | 0.995305 |
| hardship_end_date | 0.995305 |
| payment_plan_start_date | 0.995305 |
| hardship_length | 0.995305 |
| hardship_dpd | 0.995305 |

*Figure 6: Proportion of Features with >80% Null Values*

Thus, there is a need to remove the respective (>80%) null-valued columns from the training dataset since unnecessary empty columns could degrade prediction performance [[13],[14],[23],[27]], after which 97 Features remain within the Dataset, having less than 80% null values.

After removing Features containing mostly null values (>80%), each row's output within the Dataset would need to be labelled (i.e. whether each Loan row/record was recognized as a "Defaulted" or "Non-Defaulted")
This labelling comes from the Dataset's Column *loan_status* containing the row's Default Outcome in a String/Text format (**Figure 7.1**) and will need to be converted to integer-type Values. This column would contain '0' to indicate the row's Defaulted outcome and '1' for a non-Defaulted one (as executed in **Figure 7.2**) to be better processed by the Splitting of the Dataset and for ML Model Training/Testing later.

```
In [116]: # Check the unique values in loan_status
          loan_data['loan_status'].value_counts(normalize = True)

Out[116]: Fully Paid                                            0.460904
          Current                                               0.406824
          Charged Off                                           0.115742
          Late (31-120 days)                                    0.009686
          In Grace Period                                       0.003960
          Late (16-30 days)                                     0.001653
          Does not meet the credit policy. Status:Fully Paid    0.000879
          Does not meet the credit policy. Status:Charged Off   0.000337
          Default                                               0.000014
          Name: loan_status, dtype: float64
```

*Figure 7.1 String/Text Labels of each row's Defaulting Outcome*

```
In [119]: # Create new column based on the loan_status
          loan_data['good_bad'] = np.where(loan_data.loc[:, 'loan_status'].isin(['Charged Off', 'Default', 'Late (31-120 days)',
                                                                                 'Does not meet the credit policy. Status:Charged Off']), 0
          # Remove the 'loan_status' column
          loan_data.drop(columns = ['loan_status'], inplace = True)
```

*Figure 7.2 Creating a new Column containing '1'/'0' as a integer label for each row's Defaulting Outcome*

### 3.1.2 Splitting Training/Testing Dataset

Next, like Nikolic et al. [3], Palekar et al. [13], Richards et al. [14], Li and Chen [4], and Mesafint and Manjalah [24], the Capstone's Dataset will also be split through an 80:20 split, with Random Stratification [[13],[14]], to have a well-mixed/diverse Train and Test Datasets, removing any biases/unbalanced dataset groups. As seen in **Figure 8**, our Dataset has also employed the 80:20 Train: Test Dataset Split with Stratification and a specified random state to allow easy replication by future researchers if needed.

```
In [120]: # Split the data using an 80/20 split
          X = loan_data.drop('good_bad', axis = 1)
          y = loan_data['good_bad']
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42, stratify = y)

          X_train, X_test = X_train.copy(), X_test.copy()
```

*Figure 8 Splitting the Dataset with a (Train:Test split ratio of 80:20)*

### 3.1.3 Data Transformations Methods (on Training Dataset)

After splitting the dataset, data cleaning (method) functions will be created to properly format the values in the dataset into usable data types. The training Dataset will be cleaned first, followed by the test Dataset.

Training dataset's Features/Columns that need to be cleaned include; 1) the number of years found in column "emp_length" (post-processing in **Figure 9**), 2) columns with dates that are of the string (text) type format such as "earliest_cr_line", "issue_d", "last_payment_d", last_credit_pull_d" (post-processing in **Figure 9.2**), and similar to the column "emp_length", the column "term" (as seen in **Figure 9.1**) which contains the number of months, via a combination of strings and numbers. Creating individual data-cleaning python methods can allow accessible data applications to the entire training dataset, allowing for faster reusing of the data-transforming methods for transforming the Test Dataset later on.

The "emp_length" column contains the number of years with the 'year' string attached to it (e.g. "< 1 years"). Thus, removing the string and retaining the number of the equivalent year(s) in an integer data type could allow the Machine Learning Model to utilize such numerical values more meaningfully than a categorical string (e.g. '10+ years'). The Transformation execution logic can be seen in **Figure 9**.

```
In [121]: # Create a function to clean the emp_length data (assign 0 to Nulls, and convert to numbers)
          def emp_length_converter(df, column):
              df[column] = df[column].str.replace('\+ years', '')
              df[column] = df[column].str.replace('< 1 year', str(0))
              df[column] = df[column].str.replace(' years', '')
              df[column] = df[column].str.replace(' year', '')
              df[column] = pd.to_numeric(df[column])
              df[column].fillna(value = 0, inplace = True)

          # clean X_train dataset
          emp_length_converter(X_train, 'emp_length')

          # Check the resulting data
          X_train['emp_length'].unique()

          C:\Users\I\AppData\Local\Temp/ipykernel_15152/740673079.py:3: FutureWarning: The default val
          o False in a future version.
            df[column] = df[column].str.replace('\+ years', '')

Out[121]: array([ 0., 10.,  9.,  3.,  8.,  2.,  6.,  7.,  4.,  5.,  1.])
```

*Figure 9: Result of transforming the raw Feature emp_length*'s data type *from String to an Integer-based value*

The fields containing dates in a string format (e.g. “earliest_cr_line”, “issue_d”, “last_payment_d”, last_credit_pull_d”) will be converted to an integer data type, containing the number of months (as seen in **Figure 9.1**) which would allow the model to recognize and predict better as compared to it being in a Calendar date types.

```
In [124]: # create function to remove 'months' label from the records
          # and convert the resulting string to a number
          def loan_term_converter(df, column):
              df[column] = pd.to_numeric(df[column].str.replace(' months', ''))

          loan_term_converter(X_train, 'term')
```

*Figure 9.1 Feature ‘Term’ converting String Type to Integer Type (with replacing ‘months’ String)*

```
In [123]: # convert date columns to datetime format and create a new column as a difference between today and the respective date
          def date_columns(df, column):
              # store current month
              today_date = pd.to_datetime('2022-01-31')
              # convert to datetime format
              df[column] = pd.to_datetime(df[column], format = "%b-%Y")
              # calculate the difference in months and add to a new column
              df['mths_since_' + column] = round(pd.to_numeric((today_date - df[column]) / np.timedelta64(1, 'M')))
              # make any resulting -ve values to be equal to the max date
              df['mths_since_' + column] = df['mths_since_' + column].apply(lambda x: df['mths_since_' + column].max() if x < 0 else x)
              # drop the original date column
              df.drop(columns = [column], inplace = True)

          # apply to X_train
          date_columns(X_train, 'earliest_cr_line')
          date_columns(X_train, 'issue_d')
          date_columns(X_train, 'last_pymnt_d')
          date_columns(X_train, 'last_credit_pull_d')

          # let's check these new columns
          print(X_train['mths_since_earliest_cr_line'].describe())
          print(X_train['mths_since_issue_d'].describe())
          print(X_train['mths_since_last_pymnt_d'].describe())
          print(X_train['mths_since_last_credit_pull_d'].describe())
```

*Figure 9.2 date_columns* function to convert Features with date types into the number of days, from the value’s date till the Current (Now) Date

### 3.2 Feature Selection (on Training Dataset)

Next, the training dataset will be split between 'object' and 'number' data types as categorical and continuous data types, respectively. After which, Feature Selection can begin where by applying Chi-Squared Statistic, which only accepts discrete categorical values [17],[18], and ANOVA F-Test [17] for the respective categorical ones.

As such, Chi-Squared may help identify variables that are highly related/dependent on one another and, therefore, assists the analyst in removing such directly correlated variables. This protects the model from "redundant information… [which would] not help the learning algorithm [Model] to achieve a better understanding of the data" [29]. To add on, Chi-Squared produces **p-values** for Features, which determine their significance, and can be used to rank in ascending order, where the smaller **p-values**, the more statistically significant the features are. Each feature's statistically significant **p-values** will be sorted in ascending order according to their **p-values** (**Figure 9.3**). The top 4 Discrete Features will be selected for further consideration for WOE Binning later as a guide.

| | Feature | p-value |
|---|---|---|
| 0 | grade | 0.0 |
| 1 | home_ownership | 0.0 |
| 2 | verification_status | 0.0 |
| 3 | pymnt_plan | 0.0 |
| 4 | purpose | 0.0 |
| 5 | addr_state | 0.0 |
| 6 | initial_list_status | 0.0 |
| 7 | application_type | 0.0 |
| 8 | hardship_flag | 0.0 |
| 9 | disbursement_method | 0.0 |
| 10 | debt_settlement_flag | 0.0 |

*Figure 9.3: Chi-Squared Performed on Discrete Features*

The same procedure will be done to the columns with the continuous (numerical) data type by using the ANOVA F-statistic instead since it handles multi-class variables, including that of Continuous data types [17] (as seen in **Figure 9.4**). Likewise, as a guide, the top 20 Continuous/Numerical Features will be selected for WOE Binning later.

| | Numerical_Feature | F-Score | p values |
|---|---|---|---|
| 0 | mths_since_last_pymnt_d | 101279.076110 | 0.000000 |
| 1 | int_rate | 79891.093938 | 0.000000 |
| 2 | mths_since_last_credit_pull_d | 64883.628239 | 0.000000 |
| 3 | last_pymnt_amnt | 63017.806787 | 0.000000 |
| 4 | out_prncp | 60841.859961 | 0.000000 |
| ... | ... | ... | ... |
| 80 | mths_since_recent_bc_dlq | 9.124605 | 0.002522 |
| 81 | total_bal_il | 3.567721 | 0.058913 |
| 82 | mths_since_earliest_cr_line | 2.078893 | 0.149349 |
| 83 | tot_coll_amt | 0.525834 | 0.468364 |
| 84 | policy_code | NaN | NaN |

*Figure 9.4: ANOVA F-Test Performed on Continuous Features*

To further facilitate the process of Feature Selection, the removal of correlated Features can be done by placing each Feature with continuous data types in a correlation Heatmap matrix (i.e. paired against other continuous Features, as seen in **Figure 10**) [13]. This matrix plot reveals correlated Features "out_prncp_inv" and "total_pymnt_inv" as correlated with one another, thus requiring them to be removed to prevent the ML model from valuing similarly Correlated Features that could negatively skew the model's predictions.

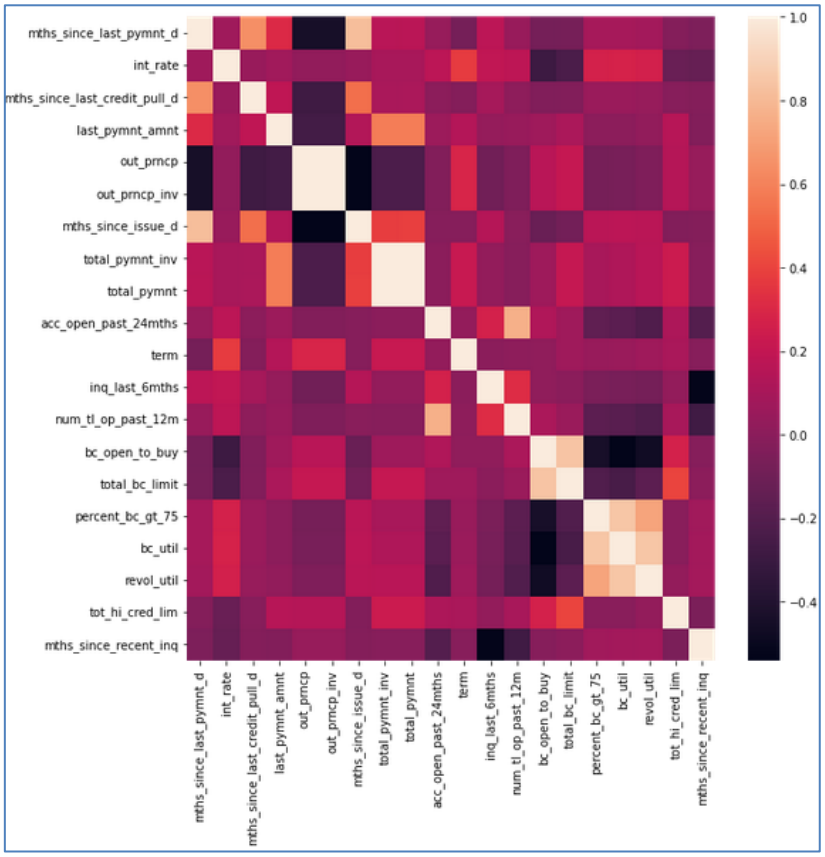


*Figure 10: Multicollinearity Comparison of Continuous Features*

To conclude the Feature Selection process, the Features of the top 4 Categorical (**Figure 9.3**) and top 20 Continuous (**Figure 9.4**) data types are kept due to them being the most significant (lower **p-values**). In contrast, the remaining Features are removed from the training dataset (Features Retained shown in **Section 3.5's Table 1)**.

### 3.3 One-Hot Encoding (on Training Dataset)

Next, one-Hot Encoding being more efficient [[15],[16],[23]] when dealing with Categorical Features, was performed on the four selected Categorical Features within the Training Dataset (seen in **Figure 11**). This Encoding strategy splits all the Feature's values into individual columns (one column for each value) containing a binary '1' or '0', thus eliminating any Relational (Value) Biases during ML Model Training.

In [135]:
```
X_train
```

Out[135]:

| | grade:A | grade:B | grade:C | grade:D | grade:E | grade:F | grade:G | home_ownership:ANY | home_ownership:MORTGAGE | home_ownership:NONE | home_ownership:OTI |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| ) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| ) | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| ) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| . | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| ) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| ) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |
| ) | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | |

*Figure 11 One-Hot Encoding performed on the Selected Top 4 Categorical Features of Training Dataset (Features: 'grade', 'home_ownership', 'verification_status','pymnt_plan')*

## 3.4 Applying Data Transformations (on Test Dataset)

With the completion of the previously mentioned data transformations (in **Section 3.1.3** and **Section 3.3**), on the Training Dataset, the same would be done to the test Dataset with the respective re-indexing to ensure that the columns in the Test Dataset match that of the Training one; by 1) using the mentioned python methods created such as "date_columns()", "loan_term_converter()"; and 2) remove the same Categorical and Continuous Features flagged out with higher **p-values** according to the previously established Training Dataset; before finally 3) performing One-Hot Encoding on the four statistically significant categorical Features, via the "dummy_creation()" method.
In order to do so, the transformation of the Test Dataset can be performed via the individual Python Methods used on the Training Dataset (as seen in **Figure 12**).

In [136]:
```
emp_length_converter(X_test, 'emp_length')
date_columns(X_test, 'earliest_cr_line')
date_columns(X_test, 'issue_d')
date_columns(X_test, 'last_pymnt_d')
date_columns(X_test, 'last_credit_pull_d')
loan_term_converter(X_test, 'term')
col_to_drop(X_test, drop_columns_list)
X_test = dummy_creation(X_test, ['grade', 'home_ownership', 'verification_status','pymnt_plan'
                                 #, 'purpose'
                                ])

# Use the Re-index function to ensure that the columns in the test dataset are also present in the training dataset
X_test = X_test.reindex(labels=X_train.columns, axis=1, fill_value=0)
```

*Figure 12 Using the Data-Transformation Python Methods on the Test Dataset*

## 3.5 Weight of Evidence (WOE) and Information Value (IV)

In order to bin Each Feature through Weight of Evidence (WOE) Binning (in **Section 3.6**), there is a need first to analyze the remaining Features' Information Values (IV), which are calculated from their respective WOE values.
WOE for Each Feature/Variable is calculated by the Equation (in **Figure 13**) by having the **Proportion of the Number of Non-Defaulting Records Divided By** the **Proportion of the Number of Defaulting Records**. A Positive WOE would mean a Higher Proportion of Non-Defaulting for that Feature, while a Negative one reveals a Higher Proportion of Defaulting for that same Feature.

$$WoE = \ln\left(\text{goods}_i / \sum_{i=1}^{n} \text{goods}_i\right) - \ln\left(\text{bads}_i / \sum_{i=1}^{n} \text{bads}_i\right)$$

*Figure 13: WOE calculation (taken from Nikolic et al's study [3])*

Pseudo-Code of ***Number of Observations ['n_obs' column]*** ***AND*** ***Proportion of Non-Defaulting ['prop_good' column]*** *in* ***Section 3.6*****:**

- For Each Feature (e.g. 'Grade' Feature):
    - For Each Feature's Unique Value/Value-Range (e.g. Categorical Value of *'A'/'B'/'C'/'D'* **OR** Continuous Value-Range of *(5.284, 6.594] / (6.594, 7.878]* )
        - Calculate the **Number of Observations (**i.e. ***'n_obs' column***) Get the **Number of Non-Defaulting Records** that use that Feature's Value/Value-Range (e.g. Total of 12,000 Defaulting Cases)
        - Calculate the **Proportion of Non-Defaulting (**i.e. **'prop_good' column**) Get the **Proportion of the Number of Non-Defaulting Records** that use the Feature's Categorical/Continuous Value/Value-Range (e.g. A proportion of 0.6 for Grade 'B' means that 60% of All Records that uses the Feature's Categorical/Continuous Value/Value-Range of Grade 'B' are non-Defaulting Outcomes)

Pseudo-Code of ***WOE*** *(i.e.* ***'WoE' column*** *in* ***Section 3.6*****):**

- For Each Feature (e.g. 'Grade' Feature):
    - For Each Feature's Unique Value/Value-Range (e.g. Categorical Value of *'A'/'B'/'C'/'D'* **OR** Continuous Value-Range of *(5.284, 6.594] / (6.594, 7.878]* )
        - Calculate the **Number of Observations**
        - Calculate the **Proportion of Non-Defaulting Records**
        - Calculate the **Proportion of Defaulting Records** (i.e. Same Procedure as **Proportion of Non-Defaulting Records** but by using Defaulting Outcomes**)**
        - Calculate the **Number of Non-Defaulting Records** (i.e. '**n_good' column**: **Proportion of Non-Defaulting Records MULTIPLY Number of Observations**)
        - Calculate the **Number of Defaulting Records** (i.e. '**n_bad' column**: [1 **MINUS Proportion of Defaulting Records**] ***MULTIPLY*** **Number of Observations**)
        - Calculate the **Proportion of Non-Defaulting Records** (i.e. '**prop_n_good' column**: **Number of Non-Defaulting Records** ***DIVIDED BY*** **Summation of All Non-Defaulting Records in Feature**)
        - Calculate the **Proportion of Defaulting Records** (i.e. '**prop_n_bad' column**: **Number of Defaulting Records** ***DIVIDED BY*** **Summation of All Defaulting Records in Feature**)
    - Calculate the **WOE** of Each Feature's Categorical/Continuous Value/Value-Range (**Proportion of Non-Defaulting Records DIVIDED BY Proportion of Defaulting Records)**

Categorical Features (e.g. 'grade') have their unique ***Value*****s** (e.g. 'A'/'B'/'C'/'D' as seen by the 'grade' Column in **Figure 14**, but Continuous Features like 'int_rate_factor' are required to undergo 20 Sets of Quartile Cutting (code work in **Figure 16**) where their Features' Value-Ranges are split into 20 Value Ranges to have unique '***Values***'. Examples of such Value Ranges are *(5.284, 6.594] and (6.594, 7.878]*, seen in **Figure 15** (*"int_rate_factor"* Column).

| | grade | n_obs | prop_good | prop_n_obs | n_good | n_bad | prop_n_good | prop_n_bad | WoE | diff_prop_good | diff_WoE | IV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | G | 9697 | 0.602764 | 0.005362 | 5845.0 | 3852.0 | 0.003697 | 0.016934 | -1.521816 | NaN | NaN | 0.483175 |
| 1 | F | 33507 | 0.640314 | 0.018527 | 21455.0 | 12052.0 | 0.013570 | 0.052982 | -1.362083 | 0.037550 | 0.159733 | 0.483175 |
| 2 | E | 108512 | 0.721238 | 0.060000 | 78263.0 | 30249.0 | 0.049500 | 0.132977 | -0.988198 | 0.080924 | 0.373885 | 0.483175 |
| 3 | D | 259341 | 0.800371 | 0.143398 | 207569.0 | 51772.0 | 0.131285 | 0.227594 | -0.550195 | 0.079133 | 0.438003 | 0.483175 |
| 4 | C | 519664 | 0.859985 | 0.287340 | 446903.0 | 72761.0 | 0.282661 | 0.319864 | -0.123648 | 0.059614 | 0.426547 | 0.483175 |
| 5 | B | 531178 | 0.915789 | 0.293706 | 486447.0 | 44731.0 | 0.307672 | 0.196641 | 0.447651 | 0.055805 | 0.571300 | 0.483175 |
| 6 | A | 346635 | 0.965214 | 0.191666 | 334577.0 | 12058.0 | 0.211616 | 0.053008 | 1.384329 | 0.049425 | 0.936678 | 0.483175 |

*Figure 14: WOE and IV calculated values for "grade" Categorical Feature (Section 3.5)*

| | int_rate_factor | n_obs | prop_good | prop_n_obs | n_good | n_bad | prop_n_good | prop_n_bad | WoE | diff_prop_good | diff_WoE | IV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | (5.284, 6.594] | 99983 | 0.978646 | 0.055284 | 97848.0 | 2135.0 | 0.061888 | 0.009386 | 1.886139 | NaN | NaN | 0.449621 |
| 1 | (6.594, 7.878] | 145963 | 0.967471 | 0.080708 | 141215.0 | 4748.0 | 0.089317 | 0.020873 | 1.453750 | 0.011175 | 0.432389 | 0.449621 |
| 2 | (7.878, 9.162] | 154733 | 0.936006 | 0.085557 | 144831.0 | 9902.0 | 0.091604 | 0.043530 | 0.744021 | 0.031465 | 0.709729 | 0.449621 |
| 3 | (9.162, 10.446] | 166225 | 0.931490 | 0.091911 | 154837.0 | 11388.0 | 0.097932 | 0.050063 | 0.671003 | 0.004515 | 0.073018 | 0.449621 |
| 4 | (10.446, 11.73] | 217977 | 0.906137 | 0.120527 | 197517.0 | 20460.0 | 0.124927 | 0.089944 | 0.328543 | 0.025354 | 0.342460 | 0.449621 |
| 5 | (11.73, 13.014] | 201025 | 0.880363 | 0.111154 | 176975.0 | 24050.0 | 0.111934 | 0.105726 | 0.057064 | 0.025774 | 0.271479 | 0.449621 |
| 6 | (13.014, 14.298] | 187860 | 0.862302 | 0.103874 | 161992.0 | 25868.0 | 0.102458 | 0.113718 | -0.104270 | 0.018061 | 0.161333 | 0.449621 |
| 7 | (14.298, 15.582] | 142739 | 0.851183 | 0.078925 | 121497.0 | 21242.0 | 0.076845 | 0.093382 | -0.194901 | 0.011119 | 0.090631 | 0.449621 |
| 8 | (15.582, 16.866] | 118746 | 0.819101 | 0.065659 | 97265.0 | 21481.0 | 0.061519 | 0.094432 | -0.428539 | 0.032082 | 0.233639 | 0.449621 |
| 9 | (16.866, 18.15] | 113345 | 0.804023 | 0.062672 | 91132.0 | 22213.0 | 0.057640 | 0.097650 | -0.527179 | 0.015078 | 0.098639 | 0.449621 |
| 10 | (18.15, 19.434] | 79641 | 0.773647 | 0.044036 | 61614.0 | 18027.0 | 0.038970 | 0.079248 | -0.709791 | 0.030376 | 0.182613 | 0.449621 |
| 11 | (19.434, 20.718] | 48931 | 0.768409 | 0.027056 | 37599.0 | 11332.0 | 0.023781 | 0.049816 | -0.739463 | 0.005238 | 0.029672 | 0.449621 |
| 12 | (20.718, 22.002] | 39421 | 0.758631 | 0.021797 | 29906.0 | 9515.0 | 0.018915 | 0.041829 | -0.793620 | 0.009777 | 0.054157 | 0.449621 |
| 13 | (22.002, 23.286] | 21935 | 0.736859 | 0.012129 | 16163.0 | 5772.0 | 0.010223 | 0.025374 | -0.909104 | 0.021772 | 0.115484 | 0.449621 |
| 14 | (23.286, 24.57] | 21226 | 0.734759 | 0.011737 | 15596.0 | 5630.0 | 0.009864 | 0.024750 | -0.919905 | 0.002100 | 0.010801 | 0.449621 |
| 15 | (24.57, 25.854] | 20624 | 0.716641 | 0.011404 | 14780.0 | 5844.0 | 0.009348 | 0.025691 | -1.010950 | 0.018118 | 0.091046 | 0.449621 |
| 16 | (25.854, 27.138] | 10973 | 0.724506 | 0.006067 | 7950.0 | 3023.0 | 0.005028 | 0.013289 | -0.971888 | 0.007865 | 0.039063 | 0.449621 |
| 17 | (27.138, 28.422] | 4060 | 0.753695 | 0.002245 | 3060.0 | 1000.0 | 0.001935 | 0.004396 | -0.820395 | 0.029189 | 0.151493 | 0.449621 |
| 18 | (28.422, 29.706] | 5406 | 0.720681 | 0.002989 | 3896.0 | 1510.0 | 0.002464 | 0.006638 | -0.990969 | 0.033014 | 0.170574 | 0.449621 |
| 19 | (29.706, 30.99] | 7721 | 0.697578 | 0.004269 | 5386.0 | 2335.0 | 0.003407 | 0.010265 | -1.103019 | 0.023103 | 0.112050 | 0.449621 |

*Figure 15: WOE and IV Values for "int_rate_factor" Continuous Feature (for Section 3.5)*

```
# fine classing
X_train_prepr['int_rate_factor'] = pd.cut(X_train_prepr['int_rate'], 20)
df_temp = woe_ordered_continuous(X_train_prepr, 'int_rate_factor', y_train_prepr)
df_temp
```

*Figure 16: Quartile Cutting Code of "int_rate_factor" Continuous Feature with 20 Value Ranges (Resulting in Figure 2's DataFrame)*

With the Equation in **Figure 17**, the Information Value (IV) for a Feature/Variable is calculated by having the [Proportion of the Feature's Number of Non-Defaulting Records (i.e. having the Final Labelled Outcome of Non-Defaulting of that Feature) **MINUS** the Proportion of the Feature's Number of Non-Defaulting Records] **MULTIPLY BY** the Feature's Summed WOE (of the Feature's Values/Value-Ranges).

$$IV = \sum_{i=1}^{n} \left( goods_i / \sum_{i=1}^{n} goods_i - bads_i / \sum_{i=1}^{n} bads_i \right) \times WoE_i$$

*Figure 17: IV calculation (taken from Nikolic et al's study [3])*

Pseudo-Code of ***Information Value*** (i.e. **'IV' column**)**:**

- For Each Feature (e.g. 'Grade' Feature):
  - For Each Feature's Unique Categorical/Continuous Value/Value-Range (e.g. Value of 'A'/'B'/'C'/'D'):
    - Calculate the Feature Categorical/Continuous Value/Value-Range's **IV** (i.e. [**Proportion of Number of Non-Defaulting Records *MINUS* Proportion of Number of Defaulting Records**] ***MULTIPLY BY*** **WOE** of Each Feature's Categorical/Continuous Value/Value-Range)
  - Calculate the Feature's **Total IV** (i.e. Summation of All Categorical/Continuous Values'/Value-Ranges' **IV** within that Feature)

After the IV has been calculated from the Feature's WOEs, the IV is used "to rank and compare by their predictive power to come up with a short list of financial ratios, [also known as significant data variables]". Likewise, any Features having IVs outside of the acceptable IV range, such as an IV of less than 0.02 or more than 0.5, as seen in (**Figure 18**), can be safely discarded due to having too Low or High a predictive power toward the actual outcome (Defaulting/Non-Defaulting), respectively.

| Information Value | Variable Predictiveness |
|---|---|
| Less than 0.02 | Not useful for prediction |
| 0.02 to 0.1 | Weak predictive Power |
| 0.1 to 0.3 | Medium predictive Power |
| 0.3 to 0.5 | Strong predictive Power |
| >0.5 | Suspicious Predictive Power |

*Figure 18: Each Feature's Predictive Influence on the Defaulting/Non-Defaulting Outcome based on its Information Value (Adapted from Siddiq [19])*

Python Methods were created to determine the Features' WOE (i.e. "woe_discrete" for Categorical Data Types, "woe_continuous" for Continuous Data Types) and the IV based on the respective equations/calculations previously mentioned in this section to ensure more accessible WOE/IV applications on all **p-value** selected Features. These methods will then display a temporary DataFrame for the Feature(s) applied to allow the analyst to determine the Feature's Eligibility for WOE Binning.

Due to space constraints, **Table 1** reveals the respective (Capstone) Dataset Features accepted/rejected for WOE Binning with their respective reason(s), while a Snapshot of the

Features' WOE/IV calculated Values for 'grade' (**Figure 19**) and 'int_rate_factor' (**Figure 20**), can be found respectively.

| **Feature(s)** | **Feature Type** | **Included in WOE Binning?** | **Reason** |
|---|---|---|---|
| grade | Discrete | Yes | IV within Stated Range (0.02 to 0.5) |
| home_ownership | | | |
| verification_status | | | |
| pyment_plan | | No | IV <0.02 |
| term | Continuous | Yes | IV within Stated Range (0.02 to 0.5) |
| int_rate_factor | | | |
| last_pymnt_amnt | | | |
| total_pymnt | | | |
| acc_open_past_24mths | | | |
| inq_last_6mths | | | |
| revol_util | | | |
| num_tl_op_past_12m | | | |
| bc_open_to_buy | | | |
| total_bc_limit | | | |
| percent_bc_gt_75 | | | |
| bc_util | | | |
| tot_hi_cred_lim | | | |
| mths_since_last_credit_pull_d | | | |
| mths_since_issue_d | | | |
| mths_since_recent_inq | | | |
| percent_bc_gt_75 | | | |
| total_pymnt_inv | | No | Multicollinearity |
| out_prncp | | | IV>0.5 |
| mths_since_last_pymnt_d_factor | | | |

*Table 1: Features (Selected based on p-values in Section 3.2) Accepted/Rejected for WOE Binning*

### 3.6 WOE Binning

After evaluating the Features' IV Values, those with Eligible IVs would proceed to the WOE Binning Process. This process begins with another Segmentizing of Features (e.g. grade) Values/Value-Ranges (e.g. Value of 'A'/'B'/'C'/'D') according to their Total Number of Observations, where each bin may contain as many Values/Value-Ranges of at least 5% of the Total Number of Observations [3],[19]. Each Categorical Feature, such as 'grade' (in **Figure 19**) for example, can have its unique Values 'G', 'F', 'E' grouped under one bin due to its Values collective Total Number of Observations of at least 5% Total Number of Observations (or <0.05 under the '**prop_n_obs**' column name). At the same time, the remaining Values 'D', 'C', 'B' and 'A' are each required to have their separate bin since each has more than 5% of Total Observations.

| | grade | n_obs | prop_good | prop_n_obs | n_good | n_bad | prop_n_good | prop_n_bad | WoE | diff_prop_good | diff_WoE | IV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | G | 9697 | Bin 1 | 0.005362 | 5845.0 | 3852.0 | 0.003697 | 0.016934 | -1.521816 | NaN | NaN | 0.483175 |
| 1 | F | 33507 | | 0.018527 | 21455.0 | 12052.0 | 0.013570 | 0.052982 | -1.362083 | 0.037550 | 0.159733 | 0.483175 |
| 2 | E | 108512 | | 0.060000 | 78263.0 | 30249.0 | 0.049500 | 0.132977 | -0.988198 | 0.080924 | 0.373885 | 0.483175 |
| 3 | D | 259341 | Bin 2 | 0.143398 | 207569.0 | 51772.0 | 0.131285 | 0.227594 | -0.550195 | 0.079133 | 0.438003 | 0.483175 |
| 4 | C | 519664 | Bin 3 | 0.287340 | 446903.0 | 72761.0 | 0.282661 | 0.319864 | -0.123648 | 0.059614 | 0.426547 | 0.483175 |
| 5 | B | 531178 | Bin 4 | 0.293706 | 486447.0 | 44731.0 | 0.307672 | 0.196641 | 0.447651 | 0.055805 | 0.571300 | 0.483175 |
| 6 | A | 346635 | Bin 5 | 0.191666 | 334577.0 | 12058.0 | 0.211616 | 0.053008 | 1.384329 | 0.049425 | 0.936678 | 0.483175 |

*Figure 19: Categorical Feature 'grade' Binning Layout during WOE Binning*

Unlike the unique Non-Empty Values found in Categorical Features such as 'grade', Continuous Features such as '**int_rate_factor**' (in **Figure 20**) have the potential of receiving Continuous Values outside of the Value-Range. However, WOE Binning allows the Binning of such missing values separately according to Siddiqi's [19] guidelines, resulting in the 1st Bin being reserved for missing values containing *NULL* Data. Subsequently, the proportion of the Number of Observations ('**prop_n_obs**' column) for each Value, except for those in Bin 12 and 13, can be Binned separately since each reached at least 5% of Total Number of Observations for the '**int_rate_factor**' Feature. Finally, the last bin (Bin 13) contains Values >20.718 which starts from its initial Value-Range to that of any future outliers beyond it.

| | int_rate_factor | n_obs | | prop_n_obs | n_good | n_bad | prop_n_good | prop_n_bad | WoE | diff_prop_good | diff_WoE | IV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | (5.284, 6.594] | 99983 | Bin 1 | 0.055284 | 97848.0 | 2135.0 | 0.061888 | 0.009386 | 1.886139 | NaN | NaN | 0.449621 |
| 1 | (6.594, 7.878] | 145963 | Bin 2 | 0.080708 | 141215.0 | 4748.0 | 0.089317 | 0.020873 | 1.453750 | 0.011175 | 0.432389 | 0.449621 |
| 2 | (7.878, 9.162] | 154733 | Bin 3 | 0.085557 | 144831.0 | 9902.0 | 0.091604 | 0.043530 | 0.744021 | 0.031465 | 0.709729 | 0.449621 |
| 3 | (9.162, 10.446] | 166225 | Bin 4 | 0.091911 | 154837.0 | 11388.0 | 0.097932 | 0.050063 | 0.671003 | 0.004515 | 0.073018 | 0.449621 |
| 4 | (10.446, 11.73] | 217977 | Bin 5 | 0.120527 | 197517.0 | 20460.0 | 0.124927 | 0.089944 | 0.328543 | 0.025354 | 0.342460 | 0.449621 |
| 5 | (11.73, 13.014] | 201025 | Bin 6 | 0.111154 | 176975.0 | 24050.0 | 0.111934 | 0.105726 | 0.057064 | 0.025774 | 0.271479 | 0.449621 |
| 6 | (13.014, 14.298] | 187860 | Bin 7 | 0.103874 | 161992.0 | 25868.0 | 0.102458 | 0.113718 | -0.104270 | 0.018061 | 0.161333 | 0.449621 |
| 7 | (14.298, 15.582] | 142739 | Bin 8 | 0.078925 | 121497.0 | 21242.0 | 0.076845 | 0.093382 | -0.194901 | 0.011119 | 0.090631 | 0.449621 |
| 8 | (15.582, 16.866] | 118746 | Bin 9 | 0.065659 | 97265.0 | 21481.0 | 0.061519 | 0.094432 | -0.428539 | 0.032082 | 0.233639 | 0.449621 |
| 9 | (16.866, 18.15] | 113345 | Bin 10 | 0.062672 | 91132.0 | 22213.0 | 0.057640 | 0.097650 | -0.527179 | 0.015078 | 0.098639 | 0.449621 |
| 10 | (18.15, 19.434] | 79641 | Bin 11 | 0.044036 | 61614.0 | 18027.0 | 0.038970 | 0.079248 | -0.709791 | 0.030376 | 0.182613 | 0.449621 |
| 11 | (19.434, 20.718] | 48931 | Bin 12 | 0.027056 | 37599.0 | 11332.0 | 0.023781 | 0.049816 | -0.739463 | 0.005238 | 0.029672 | 0.449621 |
| 12 | (20.718, 22.002] | 39421 | | 0.021797 | 29906.0 | 9515.0 | 0.018915 | 0.041829 | -0.793620 | 0.009777 | 0.054157 | 0.449621 |
| 13 | (22.002, 23.286] | 21935 | Bin 13 | 0.012129 | 16163.0 | 5772.0 | 0.010223 | 0.025374 | -0.909104 | 0.021772 | 0.115484 | 0.449621 |
| 14 | (23.286, 24.57] | 21226 | | 0.011737 | 15596.0 | 5630.0 | 0.009864 | 0.024750 | -0.919905 | 0.002100 | 0.010801 | 0.449621 |
| 15 | (24.57, 25.854] | 20624 | | 0.011404 | 14780.0 | 5844.0 | 0.009348 | 0.025691 | -1.010950 | 0.018118 | 0.091046 | 0.449621 |
| 16 | (25.854, 27.138] | 10973 | | 0.006067 | 7950.0 | 3023.0 | 0.005028 | 0.013289 | -0.971888 | 0.007865 | 0.039063 | 0.449621 |
| 17 | (27.138, 28.422] | 4060 | | 0.002245 | 3060.0 | 1000.0 | 0.001935 | 0.004396 | -0.820395 | 0.029189 | 0.151493 | 0.449621 |
| 18 | (28.422, 29.706] | 5406 | | 0.002989 | 3896.0 | 1510.0 | 0.002464 | 0.006638 | -0.990969 | 0.033014 | 0.170574 | 0.449621 |
| 19 | (29.706, 30.99] | 7721 | | 0.004269 | 5386.0 | 2335.0 | 0.003407 | 0.010265 | -1.103019 | 0.023103 | 0.112050 | 0.449621 |

*Figure 20: WOE Binning: Continuous Features ("int_rate [factor]')*

A Comprehensive List of the Eligible Features' **Original** Names in the initial Dataset, their **WOE Binned** Naming Conventions and their respective Values/Value-Ranges will be located within **Appendix A: Table 1**. This relocation is due to the repetitive WOE Binning Nature (that follow Siddiqi's guidelines) which includes checking that the Number of Observations of Each Feature at least 5% of the Total Observations.

```
# create a list of all the reference categories, i.e. one category from each of the global features
ref_categories = ['int_rate:>20.718','last_pymnt_amnt:>12657.615','total_pymnt:>28483.595','acc_open_past_24mths:>9.6',
                  'inq_last_6mths:>1.6','num_tl_op_past_12m:>4.8','bc_open_to_buy:>35557.0','total_bc_limit:>55275.0',
                  'bc_util:>84.9','tot_hi_cred_lim:>499999.95','mths_since_last_credit_pull_d:>56.3',
                  'mths_since_issue_d:>93.2','mths_since_recent_inq:>18.75']
```

*Figure 21: List of all Outliers (exceeded values of each Feature)*

As mentioned, apart from missing Values within Each Feature having its own bin [3], any future Continuous Feature Values that exceed the Maximum Value-Range of the WOE Binned Dataset would be classified as Outlier Values [3] and be dropped (list of Outliers in **Figure 21**) during WOE Binning of any future dataset. After establishing the WOE Bins for Each Feature by specifying the Categorical/Continuous Value/Value-Ranges together with the dropping of the Outlier Values, these Specific Binning *steps* (WOE Binning, Dropping of Outliers) can then be easily WOE Bin future Datasets before predicting Credit Scores with the ideally trained LR Model.

## 3.7 Logistic Regression Pipelines: Hyper-Parameter Tuning and Feature Scaling

### 3.7.1 Feature Scaling

Similar to findings by Oğuz Ay [22] and Balabaeva and Kovalchuk [21] on Standard Scaler's effectiveness before LR Model Training, Standard Scalar will be implemented as a tool for Feature Scaling. The inclusion of Standard Scaler would help build on the GAP for the Credit Scoring Studies [[3],[4],[5]] that this Capstone initially set out to tackle.

### 3.7.2 LR Model Tuning (Hyper-Parameter Tuning)

To continue building on another GAP missing from the Credit Scoring Studies [[3],[4],[5]], Hyper-Parameter Tuning (i.e. Model Tuning) will be included due to the efficacy displayed through Nikolic et al. which adapted the *Brute Force (model) Tuning* Approach using Grid-Search with Cross-Validation, which would assist in reducing Dataset Selection Biases [[4],[5],[14],[16],[21],[23],[24]], before determining the Best possible LR Model Parameters that yield the Highest Accuracy possible. After Model Training, each LR Model's Accuracy would then be evaluated using metrics such as ROC Curves [3] to visually analyse its performance via the True Positive Rates (TPR) against the False Positive Rates (FPR), according to Roy et al. [30]. Should the visual ROC Curves be unclear in determining its Accuracy, other metrics such as AUROC **[[14],[24]]** and GINI Coefficient [26], can be used to provide a more in-depth review of the Accuracy by providing more distinguishable Numerical Scores for evaluating Accuracy, together with Confusion Matrix Plots [24] which can assist in providing Numerical TPR/FPR Values.

To further expound on TPR/FPR assisting in Accuracy Evaluation, Roy et al.'s study [30] on Machine Learning Evaluation Metrics for Malware Detection used a "standard confusion metrics" plot format (**Figure 22**), which lays out the plotting of TP, FP, FN, and TN values. To add on, Louzada et al. [32] explain that Model performance evaluation compares "its prediction and the real classification of a client", providing four possible scenarios for Model classification accuracy where: (1) if the Model predicts a true-positive (TP) then it predicted that the "client is a bad payer, which, in fact, he actually is"; (2) while if the Model predicts a false-positive (FP) "the model accuses the client of being a bad payer, which, in fact, he actually is not"; (3) when the Model predicts a false-negative (FN) "the model classifies him as a good payer, which, in fact, he actually is not"; and lastly (4) should the Model predict a true-negative (TN), classifying "the client as a good payer, which, in fact, he actually is". These four scenarios show that TP and TN "can be seen as 'indicating that the proposed model "got it right" and are important measures of financial behavior'", thus higher levels of these two measurements and lower levels of FN and FP are ideal in any Model, regarding classification efficiency. Finally, the Model's True Positive Rates (TPR), which are derived as the "actual positives which are correctly identified" [33] , and the False Positive Rates (FPR), which are "the proportion of actual negatives which are falsely identified as positive" [33] are

rates that a Model can predict TPs and FPs, at any given time where higher TPR and lower FPR will reveal better Model classification efficiency.

| | actual positive (malware app) | actual negative (benign app) |
|---|---|---|
| predicted positive | true positive (TP) | false positive (FP) |
| predicted negative | false negative (FN) | true negative (TN) |

*Figure 22: Confusion Matrix Layout from Roy et al's study [30]*

As for the Numerical AUROC/GINI-Coefficient's Evaluations, "the auROC represents measure of the area under the ROC curve; [where] the closer the value of auROC getting to 1, the better the performance of the classifier" [30] , while Nikolic et al.'s study continues with GINI coefficient being also used to "calculate the model performance" by corresponding "to twice the area under the ROC curve…" and is calculated as "GINI = 2 x AUC – 1" while Abdou and Pointon's study [31] explains that the "GINI coefficient... [is] a measure of the inequality of a distribution and summarizes the predictive performance" from a range of '0' to '1'. Siddiqi [19] pointed out that if this GINI coefficient is close to 0, the model is not efficient in discerning between the defaulting and non-defaulting, while a GINI nearing '1' reveals a Model with better classification proficiency. Since ROC Curves rely on TPR/FPR Values, subsequent metrics (AUROC, GINI Coefficient), despite building on ROC, and adopting the same TPR/FPR Values, may still be able to provide further Model insights due to their differing Formula Computations.
The resulting Feature Scaling and Hyper-Parameter Tuning options, together with WOE Binning, can be grouped in a *Pipeline* to ensure easily accessible applications in Training/Testing our prepared Credit Scoring Dataset (and any other ones in the future).

There are **4 ML Model Pipelines** with the Following Layouts:
- **Pipeline 1:** WOE Binning > Logistic Regression (Un-Tuned)
- **Pipeline 2:** WOE Binning > Logistic Regression (Tuned)
- **Pipeline 3:** WOE Binning > Standard Scaler > Logistic Regression (Un-Tuned)
- **Pipeline 4:** WOE Binning > Standard Scaler > Logistic Regression (Tuned)

Each pair (**Pair 1**: Pipelines 1 and 2; **Pair 2**: Pipelines 3 and 4) will be evaluated side-by-side in **Section 4** to reveal the Performance Effects of Hyper-Parameter Tuning.
While Pairs (**Pair 1**: Pipelines 1 and 3), (**Pair 2**: Pipelines 2 and 4) will be evaluated in **Section 4** for the Effects of Feature Scaling (using Standard Scaler).

Due to the Exhaustive Model-Discovering Nature of GridSearch, only a few LR parameters (such as 'solver', 'C', 'penalty' and 'max iteration') and their respective Values are selected for the experiment phase, with the omission of those left out, due to the constraints in Time and Computational Power. As such, a few chosen LR parameters were adapted from Książek et al., including 'Penalty' which "adds bias to the model when it is suffering from high variance," 'C' where "higher values generalize the model, whereas the smaller values constrain it more," 'Max iteration' which is "the maximum number of iterations done to converge the model" and 'solver' which is a type of an algorithm solver." The respective Values and Ranges for these parameters for this Capstone are also partly derived from Książek et al., are as follows:

- **C:** Log (-4 to 4)
- **Max iteration:** '1000'
- **Penalty:** L2
- **Solver:** 'newton-cg', 'lbfgs', 'liblinear'

Grid Search Optimization was Setup with the Respective Settings (Snapshot in **Appendix A: Figure 4**) :

- Cross-Validation Settings
  - Repeated Stratification K Fold
    - 10 Splits
    - 3 Number of Repeats
    - Random State of '1'
- Optimizer's Scoring metric of 'auc_roc' (i.e. AUROC)
- Error Score of '0'

### 3.8 (Credit) Scorecard

After settling on a Model Pipeline with the Highest Accuracy, a Scorecard can be created to allow the business to predict their customers' Credit Scores during their day-to-day operations. This Pipeline containing the LR Model's coefficients (including its Intercept and parameters) is further built upon using 'scaling', which is the process of creating a "range and format of scores in a scorecard and the rate of change in odds for increases in score" [34]. The "assign[ing] [of Score] points to each attribute [or feature]... [allows for the calculation of] the total score (when you add up the attributes across characteristics) [which] can then be interpreted to mean... probabilities of default [also known as the 'odds']" [34]. These Scorecard's Score(s) would need an operational range. "The Fair Isaac Corporation (FICO) developed the [FICO scoring] formula used by all three major credit reporting agencies in the U.S" with "FICO credit scores… calculated on a scale from 300 to 850, with 850 [being] the maximum possible score" [5]. Any predicted Credit Score(s) outside of this range would be considered an irrelevant Credit Score.

$$Score = Offset + Factor * ln(Odds)$$

*Figure 23: Score Calculation Equation of each 'Binned' Feature, adapted from Saddiq [19]*

$$Factor = \frac{Max\ FICO\ Score - Min\ FICO\ Score}{\sum(Max\ Coef\ of\ Main\ Feature) - \sum(Min\ Coef\ of\ Main\ Feature)}$$

*Figure 24: Equation for Factor variable in Figure 23 ("Approval Rate")*

The Score of each WOE 'Binned' Feature can be calculated by the equation in **Figure 23**. The 'Odds' variable for each Feature is derived from the probability of an attribute being "good or bad" (non-default or default) [19] which is asserted by the coefficient of each WOE 'Binned' Feature, generated by the Trained LR Model. The 'Factor' variable for each 'Binned' Feature "depend[s] on the approval rate" [19] derived by the equation in **Figure 24**, which divides the value difference of the maximum and minimum FICO scores (300 to 850) with the value difference of the Summation of the maximum and minimum coefficients of each WOE Binned Feature (e.g. 'grade:A' /' grade:B' / 'grade:C') based off the Main Feature (e.g. 'grade'). In **Figure 23**, the 'Offset' variable is optional. It comes into play when the maximum and minimum scores (multiplication of Factors and Odds) are outside the FICO range (300 to 850). When implemented, the 'Offset' would be added/subtracted from both the multiplied values (maximum and minimum scores of the Model), resulting in a corrected *Score* predicted for each 'Binned' Feature. The Customer's (FICO-based) Credit Score is tabulated by adding all Features' Scores together.

## 4. Results and Analysis

The Results Sub-Section will cover the Results of implementing the Methodological Process of Creating a Model-Based Scorecard (based on **Section 3.8**). The **Analysis** Sub-Section will then explain any reasons/assumptions that give rise to the respective Results gathered.

### 4.1 Results

After Grid Search Optimization, the best Hyper-Parameters obtained for the 2 **Tuned** Models (Pipeline 2 in **Figure 25** and Pipeline 4 in **Figure 26**).

```
Best: 0.907945 using {'model__C': 21.54434690031882, 'model__penalty': 'l2', 'model__solver': 'lbfgs'}
```

*Figure 25: Pipeline 2's Ideal Parameters after Grid Search Optimization/Tuning*

```
Best: 0.907945 using {'model__C': 10000.0, 'model__penalty': 'l2', 'model__solver': 'lbfgs'}
```

*Figure 26: Pipeline 4's Ideal Parameters after Grid Search Optimization/Tuning*

The 4 Modelled Pipelines (2 **Tuned**, 2 **Untuned**) were then Plotted via Confusion Matrix Plots against one another for Comparing TP, FP, TN, and FN.
**Table 2** below shows the results attained from the four pipeline's Confusion Matrixes (Original Confusion Matrixes in **Appendix A: Figure 1** and **Appendix A: Figure 2**), the GINI AND AUROC scores (***Original*** Cell Output in **Appendix A: Figure 3**), while the ***Original*** ROC Plot Curves are given later in this section.

| **Pipeline No.** | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **LR Tuned?** | No | Yes | No | Yes |
| **Feature Scaling?** | No | No | Yes | Yes |
| **GINI Score** | 0.8158713758013114 | 0.8158900854661821 | 0.8158908685747437 | 0.8158908685747437 |
| **AUROC Score** | 0.9079356879006557 | 0.9079450427330911 | 0.9079454342873718 | 0.9079454342873718 |
| **True Positive (TP)** | 26107 | 26121 | 26120 | 26122 |
| **True Negative (TN)** | 387327 | 387327 | 387328 | 387328 |
| **False Positive (FP)** | 30762 | 30748 | 30749 | 30747 |
| **False Negative (FN)** | 7938 | 7938 | 7937 | 7937 |
| **Confusion Matrix *Original* Plot Found in** | *Appendix A: Figure 1* | *Appendix A: Figure 2* | *Appendix A: Figure 1* | *Appendix A: Figure 2* |
| **Confusion Matrix Plot *Name* (in Figure)** | *LG (No Scalar)* | *LG (No Scalar) Tuned* | *LG & Scalar* | *LG & Scalar Tuned* |
| **ROC *Original* Plot Found in** | *Figure 27* LG without Scalar (Not Tuned) | *Figure 28* LG without Scalar (Tuned) | *Figure 27* LG with Scalar (Not Tuned) | *Figure 28* LG with Scalar (Tuned) |

*Table 2: Results of 4 Pipelines (Confusion Matrixes' Positives/Negatives, GINI, AUROC)*

From **Table 2**, the 4 Pipelines' Confusion Matrixes reveal that:

- Pipeline 1 received 26107 TP, 387327 TN, 30762 FP, 7938 FN.
- Pipeline 2 received 26121 TP, 387327 TN, 30748 FP, 7938 FN.
- Pipeline 3 received 26120 TP, 387328 TN, 30749 FP, 7937 FN.
- Pipeline 4 received 26122 TP, 387328 TN, 30747 FP, 7937 FN.

Likewise, the 4 Pipelines' AUROC and GINI values (in **Table 2**):

- Pipeline 1 received an AUROC score of 0.9079356879006557 and a GINI score of 0.8158713758013114.
- Pipeline 2 received an AUROC score of 0.9079450427330911 and a GINI score of 0.8158900854661821.
- Pipeline 3 received an AUROC score of 0.9079454342873718 and a GINI score of 0.8158908685747437.
- Pipeline 4 received an AUROC score of 0.9079454342873718 and a GINI score of 0.8158908685747437.

The ***Original*** ROC Plot for Pipelines 1 and 3 (in **Figure 27**) showed minor graphical differences between each Line representing their TPRs and FPRs. At the same time, variations in the Plot for Pipelines 2 and 4 (in **Figure 28**) are too minuscule for further visual comparisons.

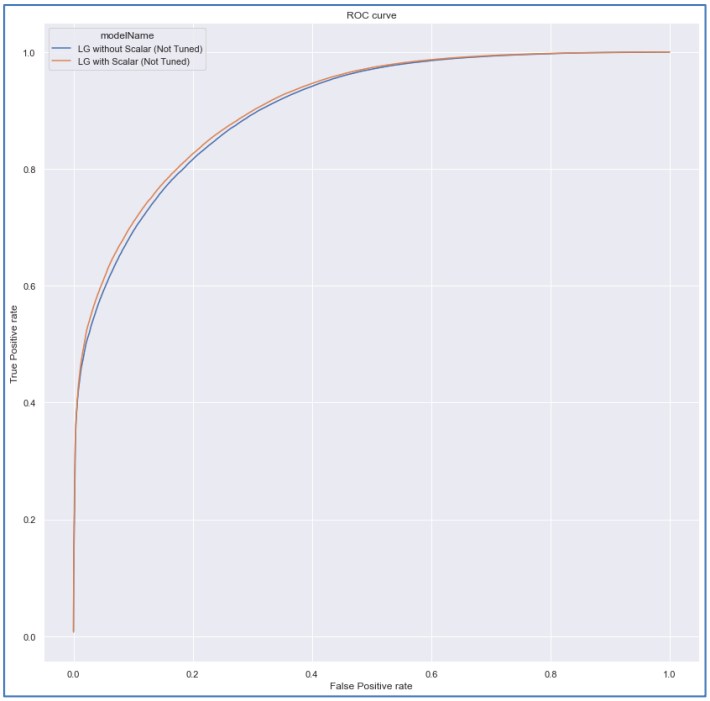


*Figure 27: ROC Curve for Pipeline 1 and 3, Without and With Feature Scaling Respectively*

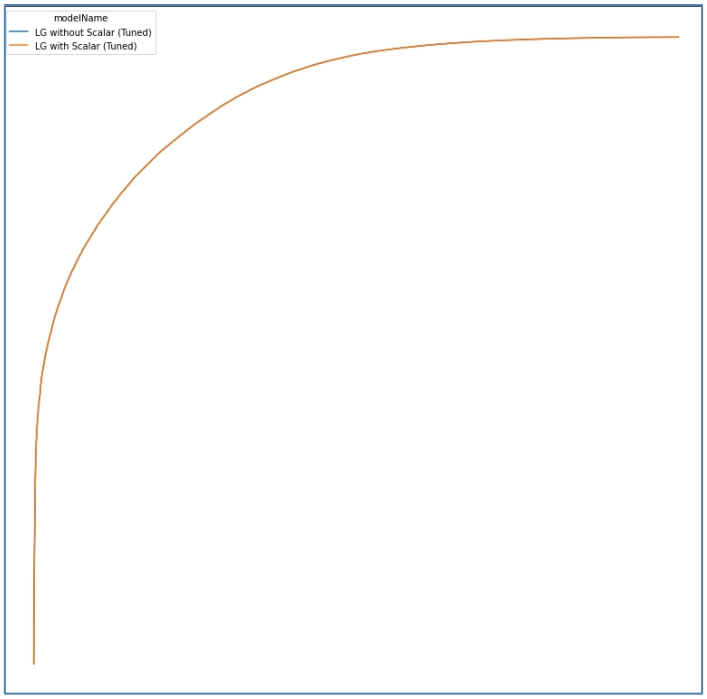


*Figure 28: ROC Curve for Pipeline 2 and 4, Without and With Feature Scaling Respectively*

## 4.2 Analysis

### 4.2.1 Evaluation of Feature Scaling (GAP 1)

This Sub-Section's Primary Focus is on the Effects of **Feature Scaling** measured by the Accuracy of the LR Pipelines in **Table 2**, which may, in some cases include Model Tuning (GAP 2) to further aid this Sub-Section's Primary Goal:

Models with **Without Tuning** (Pipeline 1 and 3):

- Pipeline 3 (**Feature Scaled** and *Without* Tuning) had 13 TP and 1 TN (how the model "got it right") Values more than Pipeline 1 (**No Feature Scaling** and *Without* Tuning).
- Pipeline 3 had 13 FP and 1 FN (how the model "got it wrong") Values Lesser than Pipeline 1.
- **Figure 27's** ROC Curve showed Pipeline 3 (**With Feature Scaling**) had a **Bigger** (TPR) **Area** covered than Pipeline 1 (**Without Scaling**), and on closer inspection of the GINI/AUROC Scores within **Table 2**:
  - Pipeline 3 has a Higher AUROC Score when compared to Pipeline 1.
  - Pipeline 3 has a Higher GINI Score than Pipeline 1.

Models with **Tuning** (Pipeline 2 and 4):

- Pipeline 4 (**Feature Scaled** and With Tuning) also had 1 TP and 1 TN Values more than the Pipeline 2 (**No Feature Scaling** and With Tuning).
- Pipeline 4 had 1 FP and 1 FN Values Lesser than Pipeline 2.
- **Figure 28's** ROC Curve show **Similar** (TPR/FPR) Area covered by Pipeline 2 and 4, and on closer inspection of GINI/AUROC Scores on **Table 2**:
  - Pipeline 4 has a Higher AUROC Score when compared to Pipeline 2.
  - Pipeline 4 has a Higher GINI Score than Pipeline 2.

These findings approve **GAP 1** in that **Feature Scaling** (i.e. Standard Scaler) improves LR Model Performance with/without any Model Tuning. This is seen by how **Feature Scaled** Pipelines 3 and 4 were able to better Correctly Classify Defaulting/Non-Defaulting Outcomes through an Increase of TP and TN while resisting more Misclassification through Decreased FP and FN Values when compared to the **Unscaled** Pipelines 1 and 2 via the Confusion Matrixes. Likewise, the **Scaled** Pipelines' Increased Classification Efficacy are also seen by their Higher AUROC, GINI Scores and the Larger ROC (TPR) Area achieved in the ROC Plot, unlike their **Unscaled** Pipeline Counterparts.

### 4.2.2 Evaluation of LR Model Tuning (GAP 2)

Similarly, this Sub-Section's Main Focus is on the Effects of **LR Model Tuning** measured by the Accuracy of the LR Pipelines in **Table 2**, which may, in some cases include Feature Scaling (GAP 2) to further aid this Sub-Section's Main Goal:

Models that are **Feature Un-Scaled** (Pipeline 1 and 2):

- Pipeline 2 (**Tuning** and *Without* Feature Scaling) had 14 TP Values More than Pipeline 1 (**No Tuning** and *Without* Feature Scaling).
- Pipeline 2 had 14 FP (the model "got it wrong") Values Lesser than Pipeline 1.
- The GINI/AUROC Scores within **Table 2**:
  - Pipeline 2 has a Higher AUROC Score when compared to Pipeline 1.
  - Pipeline 2 has a Higher GINI Score when compared to Pipeline 1.

Models that are **Feature Scaled** (Pipeline 3 and 4):

- Pipeline 4 (Feature Scaled with **Tuning**) had 2 TP Values More than Pipeline 3 (Feature Scaled with **No Tuning**).
- Pipeline 4 had 2 FP Values Lesser than Pipeline 3.
- Pipeline 4 has the Same AUROC score as Pipeline 3.
- Pipeline 4 has a Same GINI score as Pipeline 3.

These Results further approve **GAP 2** in that **LR Model Tuning** (i.e. Grid Search) indeed boosts LR Model Performance with/without Feature Scaling. This is seen by **Tuned LR Model** Pipelines 2 and 4 having better Classification Outcomes (Increased TP Values, Reduced FP Values), as compared to the **Untuned** Pipelines 1 and 3 via the Confusion Matrixes. Although the AUROC and GINI scores are Higher on the **Tuned** (Without Feature Scaling) Pipeline 2 than on the **Untuned** (Without Feature Scaling) Pipeline 1, however, the **Untuned** Pipeline 3 and **Tuned** Pipeline 4 both resulted in the Same AUROC and GINI values. The Similar AUROC and GINI Scores attained by Pipelines 3 and 4 may have resulted from the Smaller variance in their True/False Positive Values (2 TP and 2 FP). But compared to the Larger difference found in Pipeline 1 and 3's True/False Positive Values (14 TP and 14 FP), this might have subsequently varied the Pipelines' GINI/AUROC Scorings, which primarily rely on TPR/FPR Values, similar to ROC Curves.
Despite the similar AUROC/GINI Values, **GAP 2** may still be relevant in revealing how LR Model Tuning positively affects Classification Accuracy by comparing the base TP and FP Values via their respective Confusion Matrixes.

## 5. Scorecard (Utilization)

This Scorecard Section aims to assist the Company in utilizing its generated Scorecard (after attaining the ideal ML Pipeline/Model) for daily Credit Scoring operations.
Firstly, with the Ideal Pipeline's Parameters, create a Model Pipeline accordingly. In this Example, the Chosen Pipeline's (in **Figure 29**) sequence is the *WOE Binning* Python Method, followed by Standard Scaler (Feature Scaling) and finally Logistic Regression's Tuned Parameters (attained from Grid Search). Next, this **Model Pipeline** is fitted with the Testing/Training Dataset (**Figure 30)**.

```
In [30]: from sklearn.preprocessing import StandardScaler
         sc= StandardScaler()
         woe_transform = WoE_Binning(X)
         pipelineTunedScaled = Pipeline(steps=[('woe', woe_transform),('scale',sc), ('model',LogisticRegression(max_iter=1000, C=1000
                                                                                    penalty='l2',solver='lbfgs'))])
```

*Figure 29: Example of Chosen Pipeline (WOE Binning>StandardScaler>Tuned LR)*

```
In [32]: #Pipeline Fitting
         import pandas as pd
         X_train = pd.read_csv('../continue/X_train.csv')
         y_train = pd.read_csv('../continue/y_train.csv')
         X_test = pd.read_csv('../continue/X_test.csv')
         y_test = pd.read_csv('../continue/y_test.csv')
         X = pd.read_csv('../continue/X.csv')

         pipelineTunedScaled.fit(X_train, y_train)
```

```
2260666          65238.0         22000.0                   81975.0
2260667          25760.0         19500.0                       0.0

        hardship_flag  disbursement_method  debt_settlement_flag
0                   N                 Cash                     N
1                   N                 Cash                     N
2                   N                 Cash                     N
3                   N                 Cash                     N
4                   N                 Cash                     N
...               ...                  ...                   ...
2260663             N                 Cash                     N
2260664             N                 Cash                     N
2260665             N                 Cash                     N
2260666             N                 Cash                     N
2260667             N                 Cash                     N

[2260668 rows x 96 columns])),
                ('scale', StandardScaler()),
                ('model', LogisticRegression(C=10000.0, max_iter=1000
```

*Figure 30: Pipeline Fitted*

After Fitting the Pipeline, a new DataFrame would be created containing the WOE Binned Column Names transformed by the *WOE Binning* Python Method. The **LR Model Pipeline's** Coefficients will then be appended according to the WOE Binned Column Names (resulting in DataFrame within **Figure 31**).

| | Feature name | Coefficients |
|---|---|---|
| 0 | Intercept | 3.928058 |
| 1 | grade:A | 0.393112 |
| 2 | grade:B | 0.340824 |
| 3 | grade:C | 0.232294 |
| 4 | grade:D | 0.168589 |
| 5 | home_ownership:OTHER_NONE_RENT | 0.006512 |
| 6 | home_ownership:OWN | 0.015960 |
| 7 | home_ownership:MORTGAGE_ANY | -0.016434 |
| 8 | int_rate:missing | 0.000000 |
| 9 | int_rate:<6.594 | 0.510612 |
| 10 | int_rate:6.594-7.878 | 0.464779 |

*Figure 31: WOE Binning with Dataset, Retaining the Binned Feature Names and appending the Pipeline Coefficients accordingly within a DataFrame*

```
In [34]: # We create a new dataframe with one column.
         df_ref_categories = pd.DataFrame(ref_categori
         # We create a second column, called 'Coeffici
         df_ref_categories['Coefficients'] = 0
         df_ref_categories
```

Out[34]:

| | Feature name | Coefficients |
|---|---|---|
| 0 | int_rate:>20.718 | 0 |
| 1 | last_pymnt_amnt:>12657.615 | 0 |
| 2 | total_pymnt:>28483.595 | 0 |
| 3 | acc_open_past_24mths:>9.6 | 0 |
| 4 | inq_last_6mths:>1.6 | 0 |
| 5 | revol_util:>44.615 | 0 |
| 6 | num_tl_op_past_12m:>4.8 | 0 |
| 7 | bc_open_to_buy:>35557.0 | 0 |
| 8 | total_bc_limit:>55275.0 | 0 |
| 9 | percent_bc_gt_75:>95.0 | 0 |
| 10 | bc_util:>84.9 | 0 |
| 11 | tot_hi_cred_lim:>499999.95 | 0 |
| 12 | mths_since_last_credit_pull_d:>56.3 | 0 |
| 13 | mths_since_issue_d:>93.2 | 0 |
| 14 | mths_since_recent_inq:>18.75 | 0 |
| 15 | grade:E_F_G | 0 |

*Figure 32: Add the Reference Categories from Figure 21*

To begin creating a Scorecard, another DataFrame is created (**Figure 32)** with one column named as 'Feature name" Column, consisting of the Reference Categories' Naming Conventions (from *WOE Binning*), before making another column called 'Coefficients' with an initial value of '0'. The two DataFrames mentioned so far can then be joined/concatenated with their index reset since their Columns have the same Names/Shape.

| | index | Feature name | Coefficients |
|---|---|---|---|
| 0 | 0 | Intercept | 3.928058 |
| 1 | 1 | grade:A | 0.393112 |
| 2 | 2 | grade:B | 0.340824 |
| 3 | 3 | grade:C | 0.232294 |
| 4 | 4 | grade:D | 0.168589 |
| ... | ... | ... | ... |
| 103 | 11 | tot_hi_cred_lim:>499999.95 | 0.000000 |
| 104 | 12 | mths_since_last_credit_pull_d:>56.3 | 0.000000 |
| 105 | 13 | mths_since_issue_d:>93.2 | 0.000000 |
| 106 | 14 | mths_since_recent_inq:>18.75 | 0.000000 |
| 107 | 15 | grade:E_F_G | 0.000000 |

108 rows × 3 columns

*Figure 33: Two DataFrames joined into one (containing Coefficients and WOE Names)*

This results in one combined DataFrame (**Figure 33**) with the Indexes, Coefficients and Binned Feature Names (of Column-Names/Values of the WOE Binned Features and Reference Categories).

```
In [36]: # create a new column, called 'Original feature name', which contains the value of the 'Fe
         df_scorecard['Original feature name'] = df_scorecard['Feature name'].str.split(':').str[0]
         df_scorecard
```

Out[36]:

| | index | Feature name | Coefficients | Original feature name |
|---|---|---|---|---|
| 0 | 0 | Intercept | 3.928058 | Intercept |
| 1 | 1 | grade:A | 0.393112 | grade |
| 2 | 2 | grade:B | 0.340824 | grade |
| 3 | 3 | grade:C | 0.232294 | grade |
| 4 | 4 | grade:D | 0.168589 | grade |
| ... | ... | ... | ... | ... |
| 103 | 11 | tot_hi_cred_lim:>499999.95 | 0.000000 | tot_hi_cred_lim |
| 104 | 12 | mths_since_last_credit_pull_d:>56.3 | 0.000000 | mths_since_last_credit_pull_d |
| 105 | 13 | mths_since_issue_d:>93.2 | 0.000000 | mths_since_issue_d |
| 106 | 14 | mths_since_recent_inq:>18.75 | 0.000000 | mths_since_recent_inq |
| 107 | 15 | grade:E_F_G | 0.000000 | grade |

108 rows × 4 columns

*Figure 34: Adding the Original Feature Name via the Splitting by Semi-Colon (get 1st Value)*

From here, this Combined DataFrame would regain its Original Feature Names (in "*Original feature name*" Column prior to *WOE Binning* according to its '*Feature name'* Column's Values (**Figure 34**).

Next, an initial Credit Score is calculated based on the Pipeline's Coefficients for Each Binned Feature with regards to the Scoring Equations in **Figure 23** and **Figure 24**. The Features' Scores (*Score – Calculation* column) will be in a Continuous (Non-Integer) Data Type. From here, a Scorecard would begin to form after rounding the Scores to the nearest Whole Number (*Score – Preliminary* column in **Figure 35**).

```
In [37]: # Define the min and max threshholds for our scorecard
         min_score = 300
         max_score = 850

In [38]: # calculate the sum of the minimum coefficients of each category within the original feature name
         min_sum_coef = df_scorecard.groupby('Original feature name')['Coefficients'].min().sum()
         # calculate the sum of the maximum coefficients of each category within the original feature name
         max_sum_coef = df_scorecard.groupby('Original feature name')['Coefficients'].max().sum()
         # create a new columns that has the imputed calculated Score based on the multiplication of the coeff
         # maximum & minimum score and maximum & minimum sum of cefficients.
         df_scorecard['Score - Calculation'] = df_scorecard['Coefficients'] * (max_score - min_score) / (max_s
         # update the calculated score of the Intercept (i.e. the default score for each loan)
         df_scorecard.loc[0, 'Score - Calculation'] = ((df_scorecard.loc[0,'Coefficients'] - min_sum_coef) / (
         # round the values of the 'Score - Calculation' column and store them in a new column
         df_scorecard['Score - Preliminary'] = df_scorecard['Score - Calculation'].round()
         df_scorecard
```

Out[38]:

| | index | Feature name | Coefficients | Original feature name | Score - Calculation | Score - Preliminary |
|---|---|---|---|---|---|---|
| 0 | 0 | Intercept | 3.928058 | Intercept | 599.362503 | 599.0 |
| 1 | 1 | grade:A | 0.393112 | grade | 25.982462 | 26.0 |
| 2 | 2 | grade:B | 0.340824 | grade | 22.526535 | 23.0 |
| 3 | 3 | grade:C | 0.232294 | grade | 15.353307 | 15.0 |
| 4 | 4 | grade:D | 0.168589 | grade | 11.142754 | 11.0 |
| ... | ... | ... | ... | ... | ... | ... |
| 103 | 11 | tot_hi_cred_lim:>499999.95 | 0.000000 | tot_hi_cred_lim | 0.000000 | 0.0 |
| 104 | 12 | mths_since_last_credit_pull_d:>56.3 | 0.000000 | mths_since_last_credit_pull_d | 0.000000 | 0.0 |
| 105 | 13 | mths_since_issue_d:>93.2 | 0.000000 | mths_since_issue_d | 0.000000 | 0.0 |
| 106 | 14 | mths_since_recent_inq:>18.75 | 0.000000 | mths_since_recent_inq | 0.000000 | 0.0 |
| 107 | 15 | grade:E_F_G | 0.000000 | grade | 0.000000 | 0.0 |

*Figure 35: Converting the Pipeline Coefficients into Credit Scores within FICO Range and Displaying it in a DataFrame*

```
In [39]: # check the min and max possible scores of our scorecard
         min_sum_score_prel = df_scorecard.groupby('Original feature name')['Score - Preliminary'].min().sum()
         max_sum_score_prel = df_scorecard.groupby('Original feature name')['Score - Preliminary'].max().sum()
         print(min_sum_score_prel)
         print(max_sum_score_prel)
         #Guideline from FICO: from 300 to 850

         301.0
         849.0
```

*Figure 36: Scorecard's Score Range*

With the Scores (generated from the Coefficients) in Whole Numbers, there is a need to find the Minimum and Maximum Scores in our Scorecard to determine if its current Score Range is within the FICOTM Score Range. For example, in **Figure 36**, the Scorecard's Score Range is 301 to 849, which is within FICO's Range.

To first utilize the Scorecard, the Customers' Data (referenced by **X_test** variable) collected during the Company's Loan Application Process would then need to be transformed via ***WOE Binning*** (resulting in **Figure 37**). From here, an *Intercept* column is added with initial Default Values of '1' before displaying the Binned Result in a DataFrame.

| | Intercept | grade:A | grade:B | grade:C | grade:D | home_ownership:OTHER_NONE_RENT | home_ownership:OWN | home_ownership:MORTGAGE_ANY | int_rate:missing |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 3 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

5 rows × 92 columns

*Figure 37: Result of WOE Binning and Intercept Inclusion on the Customers' Data*

In continuation of the example, the Scores for Each WOE Binned Feature in the Scorecard ('Score – Preliminary' column in **Figure 35**) would be checked against that of the WOE Transformed Customers' Loan Application Data (in **Figure 37**) by measuring the Shape of both DataFrames. From **Figure 38**, the WOE Transformed Customers Dataset's Shape of (…, 92) would then be Re-Shaped to that of the generated Scorecard's '*Score – Preliminary'* Column (containing the Features' Scores in **Figure 35**), which is (108,…).

```
In [81]: # get the list of our final scorecard scores
         scorecard_scores = df_scorecard['Score - Preliminary']
         # check the shapes of test set and scorecard before doing
         print(X_test_woe_transformed.shape)
         print(scorecard_scores.shape)

         (452134, 92)
         (108,)

In [68]: # the test set has 13 less columns than the rows in score
         # since the reference categories will always be scored as
         X_test_woe_transformed = pd.concat([X_test_woe_transforme

         # Need to reshape scorecard_scores so that it is (108,1)
         scorecard_scores = scorecard_scores.values.reshape(108,1)
         print(X_test_woe_transformed.shape)
         print(scorecard_scores.shape)

         (452134, 108)
         (108, 1)
```

*Figure 38: Checking the Shapes of the Scorecard and WOE Transformed Customers' Data*

This Re-Shaping allows for Dot-Matrix Multiplication of the Customers' Dataset and only the Scorecard's *Score – Preliminary* Column to tabulate the Customers' Final Credit Scores (**Figure 39**) based on the Scorecard.

```
In [46]: # matrix dot multiplication of test set with scorecard scores
         y_scores = X_test_woe_transformed.dot(scorecard_scores)
         y_scores.head()

Out[46]:
```

| | 0 |
|---|---|
| 0 | 623.0 |
| 1 | 421.0 |
| 2 | 414.0 |
| 3 | 643.0 |
| 4 | 506.0 |

*Figure 39: Performing Dot-Matrix Multiplication to Attain Usable Credit Scores*

## 6. Conclusion

This Capstone was able to fulfil the Company's need for an Interpretable Scorecard that was complex enough to be trained via the Logistic Regression (LR) Model with Grid Search Optimization to maximize the Predicted Credit Scores' Accuracy of its new Loan Applicants. The Scorecard and trained Model's Pipeline was then conformed to the FICOTM Credit Scoring Range (300 to 850), allowing the Company to analyze further if the Predicted Scores are still relevant/usable by being within the FICOTM Range.

The design included initial Data Pre-Processing of 145 Features which includes: removing empty Features; preparing the Model Labels for Each Record's Default Outcome; ensuring high variance (reduced biases) through Dataset Splitting, and transforming the Train/Test Datasets' Data into Numerical Values for compatibility with the Machine Learning Model later. Feature Selection (i.e. ANOVA F-Test and Chi-Squared) was then used to determine the Most Significant Features within the Dataset, where the top 4 Categorical and 20 Continuous Features were selected for WOE Binning. Next, One-Hot Encoded helped eliminate Relational Value Biases within Each Feature. From here, the Resulting Dataset was then Analyzed based on its Weight of Evidence (WOE) and Information Values prior to being WOE Binned, respectively. The Capstone's Main Aim in resolving the Gaps, such as how Model Tuning and Feature Scaling could Improve ML Model Performance, was identified in previous Credit Scoring Studies [**[3]**,**[4]**,**[5]**]. These resolutions were accomplished by employing Grid Search Optimization (i.e. Model Tuning) on the LR Model with the inclusion of Standard Scaler (i.e. Feature Scaling). The accuracy of the four differing ML Experimental Pipeline Configurations was evaluated through Metrics like ROC Curves, AUROC, GINI Coefficient and TPR/FPR Confusion Matrixes. The results reveal that the Pipeline with the Highest Accuracy involved the combined use of Grid Search and Standard Scalar, together with the standard WOE Binning performed on all 4 Pipelines. This Efficient Pipeline attained the Most True-Positives and True-Negatives, with the Lowest False-Negatives and False-Positives across all other Experimental Pipelines Configured within this Capstone. Apart from resolving the GAPs identified within the Studies [**[3]**,**[4]**,**[5]**], the Capstone's Final task in meeting the Company's needs included detailed instructions on the Creation of a Credit Scorecard with the Chosen Pipeline producing Credit Scores based on its Customers' Loan Application Data. One Future Improvement for this Capstone would include the use of Model Tuning *Bayesian Search Optimizer* (with ROC evaluation for TPR/FPR Values), similar to Li and Chen **[4]**'s future study improvements. This change could replace Grid Search and thus reduce overall Time Complexity and Computation Overhead, that Grid Search currently faces, while still providing the respective LR Parameters that produce the Highest possible Model *Accuracy*.

## 7. References


[1] Maldonado, S., Peters, G., & Weber, R. (2020). Credit scoring using three-way decisions with probabilistic rough sets. Information Sciences, 507, 700-714. https://doi.org/10.1016/j.ins.2018.08.001

[2] Guo, G., Zhu, F., Chen, E., Liu, Q., Wu, L., & Guan, C. (2016). From footprint to evidence. ACM Transactions on the Web, 10(4), 1-38. https://doi.org/10.1145/2996465

[3] Nikolic, N., Zarkic-Joksimovic, N., Stojanovski, D., & Joksimovic, I. (2013, May 11). The application of brute force logistic regression to corporate credit scoring models: Evidence from Serbian financial statements. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S0957417413003084

[4] Li, Y., & Chen, W. (2020, October 13). A comparative performance assessment of ensemble learning for credit scoring. Retrieved July 22, 2022, from https://www.mdpi.com/2227-7390/8/10/1756

[5] Bensic, M., Sarlija, N., & Zekic-Susac, M. (2005). Modelling small-business credit scoring by using logistic regression, neural networks and decision trees. Intelligent

Systems in Accounting, Finance and Management, 13(3), 133-150. https://doi.org/10.1002/isaf.261
[6] Tae, K. H., Roh, Y., Oh, Y. H., Kim, H., & Whang, S. E. (2019). Data cleaning for accurate, fair, and robust models. Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning - DEEM'19. https://doi.org/10.1145/3329486.3329493
[7] Gudivada, V. N., Apon, A., & Ding, J. (2017, July 22). Data quality considerations for Big Data and machine learning: Going ... Retrieved July 22, 2022, from https://www.researchgate.net/publication/318432363_Data_Quality_Considerations_for_Big_Data_and_Machine_Learning_Going_Beyond_Data_Cleaning_and_Transformations
[8] Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), 20-29. https://doi.org/10.1145/1007730.1007735
[9] Provost, F. (2000). Machine learning from imbalanced data sets 101. Retrieved July 22, 2022, from https://www.aaai.org/Papers/Workshops/2000/WS-00-05/WS00-05-001.pdf
[10] Miron, M., Tolan, S., Gómez, E., & Castillo, C. (2020, June 07). Evaluating causes of algorithmic bias in juvenile criminal recidivism - artificial intelligence and law. Retrieved July 22, 2022, from https://link.springer.com/article/10.1007/s10506-020-09268-y
[11] Al-Najjar, H., Pradhan, B., Kalantar, B., Sameen, M., Santosh, M., & Alamri, A. (2021, August 19). Landslide susceptibility modeling: An integrated novel method based on machine learning feature transformation. Retrieved July 22, 2022, from https://doi.org/10.3390/rs13163281
[12] Schratz, P., Muenchow, J., Iturritxa, E., Richter, J., & Brenning, A. (2019). Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using Spatial Data. Ecological Modelling, 406, 109-120. https://doi.org/10.1016/j.ecolmodel.2019.06.002
[13] Palekar, V., Ambatkar, S., Kamble, K., Ali, S., Zade, H., & Kharade, S. (2020, November 11). IRJET- credit card fraud detection using isolation forest. Retrieved July 22, 2022, from https://issuu.com/irjet/docs/irjet-v7i3710
[14] L, R. N. (2018). Machine learning for the prevention and prognosis of pediatric head injury in Sport. Retrieved July 22, 2022, from https://etd.ohiolink.edu/apexprod/rws_olink/r/1501/10?clear=10&p10_accession_num=ucin1535633637793776
[15] Seger, C. (2018, October 26). An investigation of categorical variable encoding techniques in machine learning: Binary versus one-hot and feature hashing. Retrieved July 22, 2022, from https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1259073&dswid=-8025
[16] Pargent, F., Pfisterer, F., Thomas, J., & Bischl, B. (2022). Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. Computational Statistics. https://doi.org/10.1007/s00180-022-01207-6
[17] Urbanowicz, R., Olson, R., Schmitt, P., Meeker, M., & Moore, J. (2018, July 17). Benchmarking relief-based feature selection methods for Bioinformatics Data Mining. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S1532046418301412
[18] Kamalov, F., & Thabtah, F. (2017). A feature selection method based on ranked vector scores of features for classification. Annals of Data Science, 4(4), 483-502. https://doi.org/10.1007/s40745-017-0116-1
[19] Siddiqi, N. (2017). Intelligent credit scoring. Retrieved July 22, 2022, from https://books.google.com.sg/books?id=q-qoDQAAQBAJ&printsec=frontcover
[20] Baesens, B., Roesch, D., & Scheule, H. (2016). Credit risk analytics. Retrieved July 22,

2022, from https://books.google.com.sg/books?id=ornsDAAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r

[21] Balabaeva, K., & Kovalchuk, S. (2019, September 26). Comparison of temporal and non-temporal features effect on machine learning models quality and interpretability for chronic heart failure patients. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S1877050919311020

[22] Ay, O. (1970, January 01). Price prediction using Machine Learning Techniques: An application to vacation rental properties. Retrieved July 22, 2022, from http://openaccess.mef.edu.tr/xmlui/handle/20.500.11779/1702

[23] Zahedi, L., Mohammadi, F., Rezapour, S., Ohland, M., & Amini, M. (2021, April 29). Search algorithms for automated hyper-parameter tuning. Retrieved July 22, 2022, from https://arxiv.org/abs/2104.14677

[24] Mesafint, D., & H, M. D. (2021, September). Grid search in hyperparameter optimization of machine learning models ... Retrieved July 22, 2022, from https://www.researchgate.net/publication/354541720_Grid_search_in_hyperparameter_optimization_of_machine_learning_models_for_prediction_of_HIVAIDS_test_results

[25] A, V. B., & N, S. (2012). Building loss given default scorecard using weight of evidence bins in. Retrieved July 22, 2022, from https://www.researchgate.net/profile/Naeem-Siddiqi/publication/276956927_Building_Loss_Given_Default_Scorecard_Using_Weight_of_Evidence_Bins_in_SASR_Enterprise_Miner_Loss_Given_Default/links/555c91bd08ae86c06b5d3976/Building-Loss-Given-Default-Scorecard-Using-Weight-of-Evidence-Bins-in-SASR-Enterprise-Miner-Loss-Given-Default.pdf

[26] Peussa, A. (1970, January 01). Credit risk scorecard estimation by logistic regression. Retrieved July 22, 2022, from https://helda.helsinki.fi/handle/10138/163135

[27] Tae, K. H., Roh, Y., Oh, Y., Kim, H., & Whang, S. E. (2019, June 01). Data cleaning for accurate, fair, and robust models: Proceedings of the 3rd International Workshop on Data Management for end-to-end machine learning. Retrieved July 22, 2022, from https://dl.acm.org/doi/10.1145/3329486.3329493

[28] Vannoy, T. C., Scofield, T. P., Shaw, J. A., Logan, R. D., Whitaker, B. M., & Rehbein, E. M. (2021, October). Detection of insects in class-imbalanced Lidar Field Measurements. Retrieved July 22, 2022, from https://ieeexplore.ieee.org/abstract/document/9596143/

[29] Bonev, B. (2010, June 29). Feature selection based on information theory. Retrieved July 22, 2022, from http://rua.ua.es/dspace/handle/10045/18362

[30 Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., . . . Guevara, N. (2015, December 01). Experimental study with real-world data for Android App Security Analysis Using Machine Learning: Proceedings of the 31st Annual computer security applications conference. Retrieved July 22, 2022, from https://dl.acm.org/doi/abs/10.1145/2818000.2818038

[31] Abdou, H. A., & Pointon, J. (2011, April). (PDF) credit scoring, statistical techniques and evaluation criteria: A ... Retrieved July 22, 2022, from https://www.researchgate.net/publication/220613924_Credit_Scoring_Statistical_Techniques_and_Evaluation_Criteria_A_Review_of_the_Literature

[32] Louzada, F., Ferreira-Silva, P., & Diniz, C. (2012, February 01). On the impact of disproportional samples in credit scoring models: An application to a Brazilian bank data. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S0957417412001522?ref=cra_js_challenge&fr=njs

[33] Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., . . . Guevara, N. (2015, December 01). Experimental study with real-world data for Android App Security Analysis Using Machine Learning: Proceedings of the 31st Annual computer security applications conference. Retrieved July 22, 2022, from https://dl.acm.org/doi/abs/10.1145/2818000.2818038

[34] Siddiqi, N. (2017, January). Intelligent credit scoring. Retrieved July 22, 2022, from https://books.google.com.sg/books?id=q-qoDQAAQBAJ
[35] Cao, N. T., Tran, L. H., & Ton-That, A. H. (2021, December). Using machine learning to create a credit scoring model in banking and Finance. Retrieved July 22, 2022, from https://ieeexplore.ieee.org/abstract/document/9718414
[36] Książek, W., Gandor, M., & Pławiak, P. (2021, May 11). Comparison of various approaches to combine logistic regression with genetic algorithms in survival prediction of hepatocellular carcinoma. Retrieved July 22, 2022, from https://www.sciencedirect.com/science/article/pii/S0010482521002250

## 8. Knowledge and Training Requirements

### 8.1 Applicable Knowledge from the Degree Programme

The prerequisite knowledge and skillsets from the degree programme that was necessary for the capstone projects are as follows:

| No | Module(s) | Knowledge(s) Applied |
|---|---|---|
| 1 | CSC3005: Data Analytics | Despite having taken this Module where some Evaluation Metrics may have been covered during this Module, much of it were not referenced during this Capstone. However, this Module did provide a Steppingstone into allowing me to understand the idea behind True Positive and False Positive Values that determine a Model's Classification Accuracy. |
| 2 | CSC1009: Object Oriented Programming | This Module was able to expose me more deeply in the Pythonic Programming Language Syntax that was mainly used for this Capstone Project (with the use of *Jupyter Notebooks*). |
| 3 | CSC3009: Machine Learning | Although this Module arrived after the Completion of this Capstone's Machine Learning code work, it was a good refresher on the True Positive Rates (TPR) and False Positive Rates (FPR) which were previously discovered from ***CSC3005: Data Analytics*** together with the Accuracy Evaluation via a Confusion Matrix Plot. |

### 8.2 Additional Knowledge, Skillsets, or Certifications Required

| No | Additional Requirement(s) | Knowledge(s) Applied |
|---|---|---|
| 1 | Logistic Regression ML Model | Knowledge required to understand the Logistic Regression Model's Function with its respective Coefficients and use-cases. |
| 2 | ML Model Tuning Options | Knowledge on the Available Parameters that the Logistic Regression Model allows are needed, together with their respective Maximum and Minimum Ranges. Likewise, the Model Tuning Options available for Logistic Regression were also required in finding the right Model Optimizer which suits the Capstone's *Logistic Regression* Credit Scoring use case. |
| 3 | Accuracy Metrics Evaluations | Evaluation Metrics were also need in being investigated for the scope of Credit Scoring Studies within the Scientific Community, with their respective citations to other Researchers and their experiment findings. |
| 4 | Credit Scorecard Development/ Formulations | Scorecard Formulation understanding are also needed to understand the way ML Models' Coefficients can be utilized together with Credit Scorecards in developing one that is functional for future Company Operations. |

# Appendices

## Appendix A:

*Table 1 – WOE Binning Overview of Eligible Features (as mentioned in Section 3.6)*

| **(Original) Feature Name** | **(Binned) Feature Name** | **Value / Value-Ranges** |
|---|---|---|
| 'grade' | 'grade:A' | 'A' |
| | 'grade:B' | 'B' |
| | 'grade:C' | 'C' |
| | 'grade:D' | 'D' |
| | 'grade:E_F_G' | 'E', 'F', 'G' |
| 'home_ownership' | 'home_ownership:OTHER_NONE_RENT' | 'OTHER', 'NONE', 'RENT' |
| | 'home_ownership:OWN' | 'OWN' |
| | 'home_ownership:MORTGAGE_ANY' | 'MORTGAGE', 'ANY' |
| 'verification_status' | 'verification_status:Verified' | Verified' |
| | 'verification_status:Source Verified' | 'Source Verified' |
| | 'verification_status:Not Verified' | 'Not Verified' |
| 'int_rate_factor' | 'int_rate_factor:missing' | ***(NULL Value)*** |
| | 'int_rate_factor:<6.594' | <6.594 |
| | 'int_rate_factor_factor:6.594-7.878' | 6.594-7.878 |
| | 'int_rate_factor_factor:7.878- 9.162' | 7.878- 9.162 |
| | 'int_rate_factor_factor:9.162-10.446' | 9.162-10.446 |
| | 'int_rate_factor:10.446-11.73' | 10.446-11.73 |
| | 'int_rate_factor:11.73-13.014' | 11.73-13.014 |
| | 'int_rate_factor:13.014-14.298' | 13.014-14.298 |
| | 'int_rate_factor:14.298-15.582' | 14.298-15.582 |
| | 'int_rate_factor:15.582-16.866' | 15.582-16.866 |
| | 'int_rate_factor:16.866-18.15' | 16.866-18.15 |
| | 'int_rate_factor:18.15-20.718' | 18.15-20.718 |
| | 'int_rate_factor:>20.718' | >20.718 |
| last_pymnt_amnt | 'last_pymnt_amnt:missing' | ***(NULL Value)*** |
| | 'last_pymnt_amnt:<2109.602' | <2109.602 |
| | 'last_pymnt_amnt:2109.602-4219.205' | 2109.602-4219.205 |
| | 'last_pymnt_amnt:4219.205-8438.41' | 4219.205-8438.41 |
| | 'last_pymnt_amnt:8438.41-12657.615' | 8438.41-12657.615 |
| | 'last_pymnt_amnt:>12657.615' | >12657.615 |
| total_pymnt | 'total_pymnt:missing' | ***(NULL Value)*** |
| | 'total_pymnt:<3164.844' | <3164.844 |
| | 'total_pymnt:3164.844-6329.688' | 3164.844-6329.688 |
| | 'total_pymnt:6329.688-9494.532' | 6329.688-9494.532 |
| | 'total_pymnt:9494.532-12659.376' | 9494.532-12659.376 |
| | 'total_pymnt:12659.376-15824.219' | 12659.376-15824.219 |
| | 'total_pymnt:15824.219-18989.063' | 15824.219-18989.063 |
| | 'total_pymnt:18989.063-22153.907' | 18989.063-22153.907 |
| | 'total_pymnt:22153.907-28483.595' | 22153.907-28483.595 |
| | 'total_pymnt:>28483.595' | >28483.595 |

| | | |
|---|---|---|
| acc_open_past_24mths | 'acc_open_past_24mths:missing' | ***(NULL Value)*** |
| | 'acc_open_past_24mths:<3.2' | <3.2 |
| | 'acc_open_past_24mths:3.2-6.4' | 3.2-6.4 |
| | 'acc_open_past_24mths:6.4-9.6' | 6.4-9.6 |
| | 'acc_open_past_24mths:>9.6' | >9.6 |
| inq_last_6mths | 'inq_last_6mths:missing' | ***(NULL Value)*** |
| | 'inq_last_6mths:<1.6' | <1.6 |
| | 'inq_last_6mths:>1.6' | 1.6 |
| revol_util | 'revol_util:missing' | ***(NULL Value)*** |
| | 'revol_util:<44.615' | <44.615 |
| | 'revol_util:>44.615' | >44.615 |
| num_tl_op_past_12m | 'num_tl_op_past_12m:missing' | ***(NULL Value)*** |
| | 'num_tl_op_past_12m:<1.6' | <1.6 |
| | 'num_tl_op_past_12m:1.6-3.2' | 1.6-3.2 |
| | 'num_tl_op_past_12m:3.2-4.8' | 3.2-4.8 |
| | 'num_tl_op_past_12m:>4.8' | >4.8 |
| bc_open_to_buy | 'bc_open_to_buy:missing' | ***(NULL Value)*** |
| | 'bc_open_to_buy:<35557.0' | <35557.0 |
| | 'bc_open_to_buy:>35557.0' | >35557.0 |
| total_bc_limit | 'total_bc_limit:missing' | ***(NULL Value)*** |
| | 'total_bc_limit:<55275.0' | <55275.0 |
| | 'total_bc_limit:>55275.0' | >55275.0 |
| percent_bc_gt_75 | 'percent_bc_gt_75:missing' | ***(NULL Value)*** |
| | 'percent_bc_gt_75:<5.0' | <5.0 |
| | 'percent_bc_gt_75:5.0-15.0' | 5.0-15.0 |
| | 'percent_bc_gt_75:15.0-20.0' | 15.0-20.0 |
| | 'percent_bc_gt_75:20.0-30.0' | 20.0-30.0 |
| | 'percent_bc_gt_75:30.0-35.0' | 30.0-35.0 |
| | 'percent_bc_gt_75:35.0-50.0' | 35.0-50.0 |
| | 'percent_bc_gt_75:50.0-70.0' | 50.0-70.0 |
| | 'percent_bc_gt_75:70.0-95.0' | 70.0-95.0 |
| | 'percent_bc_gt_75:>95.0' | >95.0 |
| bc_util | 'bc_util:missing' | ***(NULL Value)*** |
| | 'bc_util:<16.98' | <16.98 |
| | 'bc_util:16.98-33.96' | 16.98-33.96 |
| | 'bc_util:33.96-50.94' | 33.96-50.94 |
| | 'bc_util:50.94-67.92' | 50.94-67.92 |
| | 'bc_util:67.92-84.9' | 67.92-84.9 |
| | 'bc_util:>84.9' | >84.9 |
| tot_hi_cred_lim | 'tot_hi_cred_lim:missing' | ***(NULL Value)*** |
| | 'tot_hi_cred_lim:<499999.95' | <499999.95 |
| | 'tot_hi_cred_lim:>499999.95' | >499999.95 |
| 'mths_since_last_credit_pull_d | 'mths_since_last_credit_pull_d:missing' | ***(NULL Value)*** |
| | 'mths_since_last_credit_pull_d:<42.1' | <42.1 |
| | 'mths_since_last_credit_pull_d:42.1-49.2' | 42.1-49.2 |
| | 'mths_since_last_credit_pull_d:49.2-56.3' | 49.2-56.3 |
| | 'mths_since_last_credit_pull_d:>56.3' | >56.3 |
| mths_since_issue_d | 'mths_since_issue_d:missing' | ***(NULL Value)*** |
| | 'mths_since_issue_d:<44.9' | <44.9 |
| | 'mths_since_issue_d:44.9-51.8' | 44.9-51.8 |
| | 'mths_since_issue_d:51.8-58.7' | 51.8-58.7 |
| | 'mths_since_issue_d:58.7-65.6' | 58.7-65.6 |

| | | |
|---|---|---|
| | 'mths_since_issue_d:65.6-72.5' | 65.6-72.5 |
| | 'mths_since_issue_d:72.5-79.4' | 72.5-79.4 |
| | 'mths_since_issue_d:79.4-86.3' | 79.4-86.3 |
| | 'mths_since_issue_d:86.3-93.2' | 86.3-93.2 |
| | 'mths_since_issue_d:>93.2' | >93.2 |
| mths_since_recent_inq | 'mths_since_recent_inq:missing' | ***(NULL Value)*** |
| | 'mths_since_recent_inq:<1.25' | <1.25 |
| | 'mths_since_recent_inq:1.25-2.5' | 1.25-2.5 |
| | 'mths_since_recent_inq:2.5-3.75' | 2.5-3.75 |
| | 'mths_since_recent_inq:3.75-5.0' | 3.75-5.0 |
| | 'mths_since_recent_inq:5.0-6.25' | 5.0-6.25 |
| | 'mths_since_recent_inq:6.25-7.5' | 6.25-7.5 |
| | 'mths_since_recent_inq:7.5-10.0' | 7.5-10.0 |
| | 'mths_since_recent_inq:10.0-12.5' | 10.0-12.5 |
| | 'mths_since_recent_inq:12.5-15.0' | 12.5-15.0 |
| | 'mths_since_recent_inq:15.0-18.75' | 5.0-18.75 |
| | 'mths_since_recent_inq:>18.75' | >18.75 |

## Grid Search Optimizer

### Confusion Matrix Plot

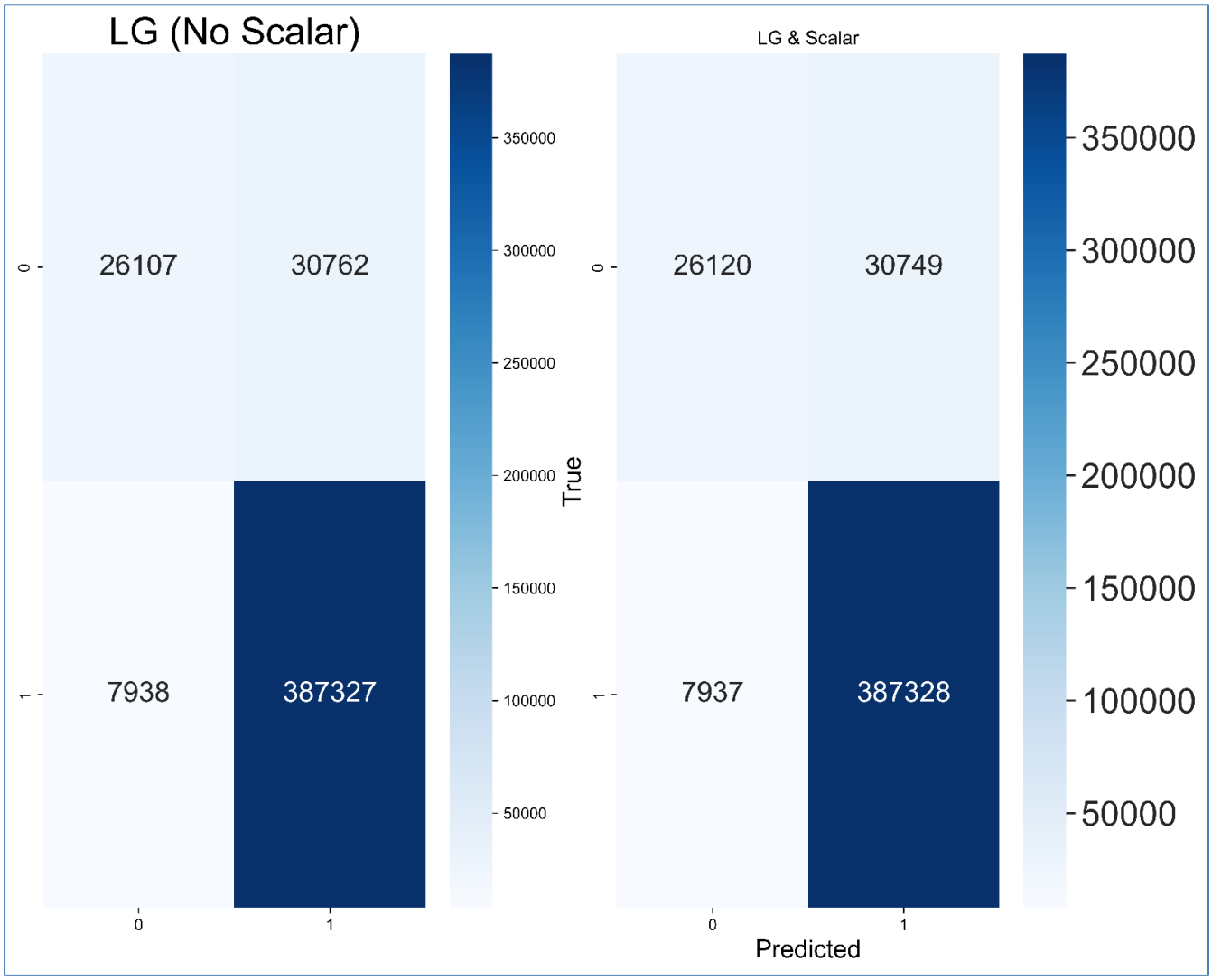


*Figure 1: Confusion Matrixes for Untuned Models (Pipeline 1 and 3)*

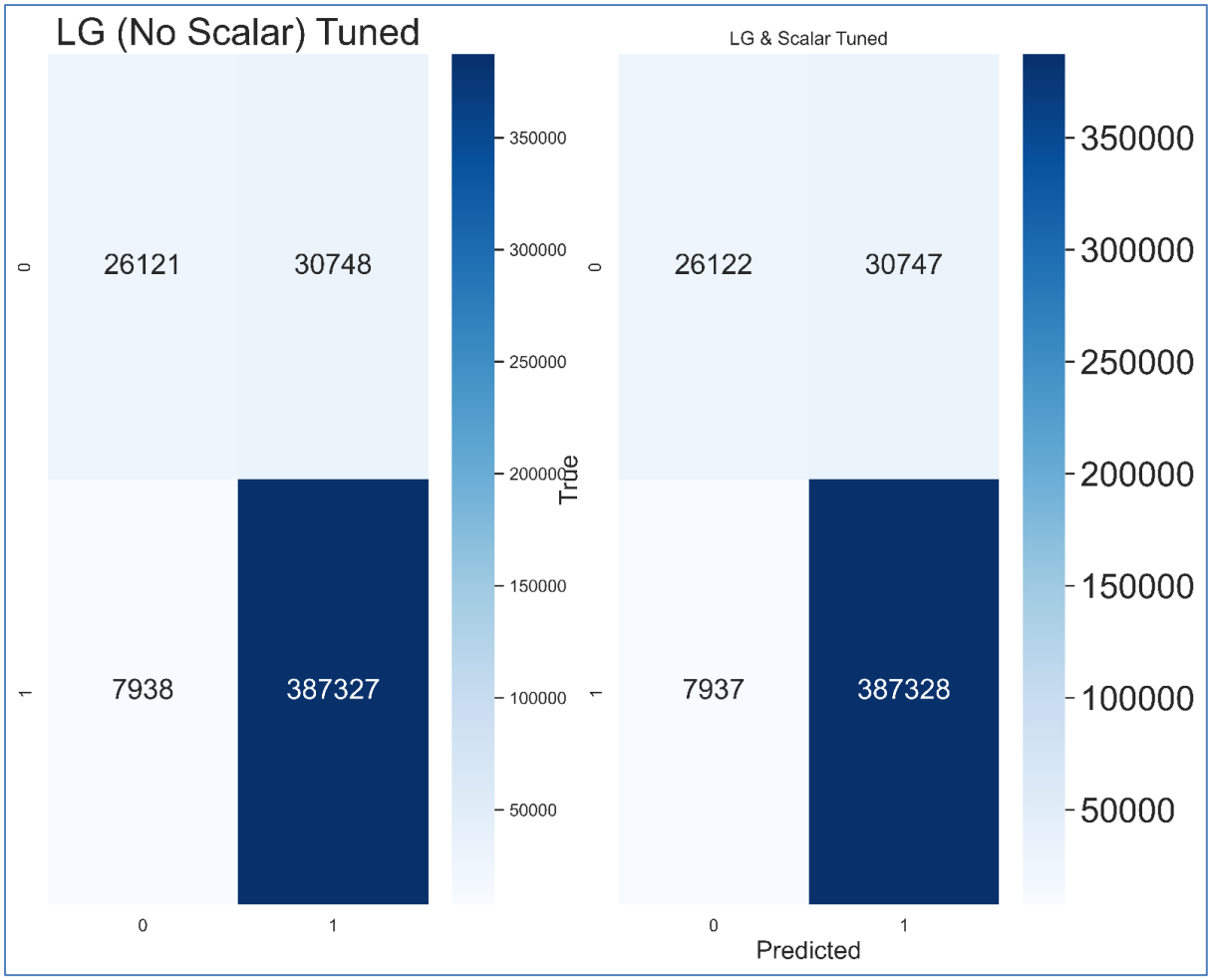


*Figure 2: Confusion Matrixes for Tuned Models (Pipeline 2 and 4)*

### GINI/AUROC Scores for All Pipelines

```
Pipeline 1 LG Only (Not Tuned)-> GINI: 0.8158713758013114  | AUC/ROC: 0.9079356879006557
Pipeline 2 LG Only (Tuned)-> GINI: 0.8158900854661821  | AUC/ROC: 0.9079450427330911
Pipeline 3 LG & scaler(Not Tuned)-> GINI: 0.8158908685747437  | AUC/ROC: 0.9079454342873718
Pipeline 4 LG & scaler(Tuned)-> GINI: 0.8158908685747437  | AUC/ROC: 0.9079454342873718
```

*Figure 3: GINI and AUROC Scores for Pipeline 1-4*
*(focus on Pipeline Number for easier referencing)*

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
reg = LogisticRegression(max_iter=1000)
woe_transform = WoE_Binning(X)
pipelineScal = Pipeline(steps=[('woe', woe_transform),('sca',sc),('model',reg)])
param = {
    "model__penalty":['l2'],
    "model__solver": ['newton-cg', 'lbfgs', 'liblinear'],
    "model__C": np.logspace(-4, 4, 4)
}
# define grid search
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_searchScal = GridSearchCV(estimator=pipelineScal, param_grid=param, n_jobs=1,
                               cv=cv, scoring='roc_auc',error_score=0)
grid_resultScal = grid_searchScal.fit(X_train, y_train)
# summarize results
print("Best: %f using %s" % (grid_resultScal.best_score_, grid_resultScal.best_params_))
means = grid_resultScal.cv_results_['mean_test_score']
stds = grid_resultScal.cv_results_['std_test_score']
params = grid_resultScal.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("Mean: %f | STD: %f | Parameter: %r" % (mean, stdev, param))

#source: https://blog.dominodatalab.com/hyperopt-bayesian-hyperparameter-optimization
```

*Figure 4: Grid Search Optimizer Settings*

**END OF REPORT**