PROGRESS REPORT FOR A HYBRID ATTENTION-BASED DEEP LEARNING MODEL TO IMPROVE CREDIT RISK PREDICTION

Arnav Talwani
Student# 1008233662
arnav.talwani@mail.utoronto.ca

Dhanishika Antony Jotheeswaran
Student# 1010131073
d.antony@mail.utoronto.ca

Dharuniga Antony Jotheeswaran
Student# 1010102112
dharuniga.antony@mail.utoronto.ca

Vishwajith Subhashraj
Student# 1010009688
vishwa.subhashraj@mail.utoronto.ca

ABSTRACT

This progress report outlines the development of a Hybrid Attention Network for credit risk prediction on structured financial data. The team completed data processing on the Lending Club dataset (855,969 records, 73 features), addressing missing values, class imbalance, and feature leakage. The final dataset contains 81 features and was split into training, validation, and test sets (70, 15, 15) using stratified sampling. A logistic regression model was implemented as the baseline, achieving ROC-AUC scores of 0.813 (train), 0.817 (validation), and 0.812 (test), with F1 scores around 0.740. Performance dropped in a reduced feature test subset (ROC-AUC 0.773, F1 0.703), revealing sensitivity to key features such as loan grade. Preliminary implementation of the Hybrid Attention Network is complete, including model architecture design and initial training. The model integrates multi-head attention and MLP streams, achieving improved performance over the baseline: ROC-AUC scores of 0.832 (train), 0.827 (validation), 0.822 (test), and 0.820 (subset), with F1 scores ranging from 0.773 to 0.778. Qualitative results show strong generalization and robustness to reduced feature sets. Feature attention analysis confirms effective adaptation to missing information, with highest weights assigned to temporal and employment-related features. Teamwork has been consistent and effective, with clear task delegation, shared GitHub and Colab environments, regular Discord check-ins, and transparent progress tracking through a shared logbook and responsibility tracker. All members have contributed equitably and met Phase 1 deadlines. The next steps are to target ROC-AUC > 0.85 and improve the robustness in missing-feature scenarios.
—-Total Pages: 10

1 INTRODUCTION

Credit risk prediction is critical for financial institutions to assess the likelihood of borrower default. Traditional models like logistic regression, decision trees, and ensemble methods use financial and demographic inputs (e.g., income, credit history, loan amount, debt-to-income ratio). However, these models often fail to capture non-linear relationships and complex feature interactions, especially with growing data complexity or limited information in underrepresented populations.

Accurate predictions can reduce default rates, improve loan allocation, and enhance financial stability. A reliable model can also reduce the time, effort, and subjectivity required by analysts, enabling them to focus on edge cases and improving operational efficiency.

To address these challenges, this project introduces a Hybrid Attention-Based Deep Learning Model that processes tabular loan applicant data to predict default risk. The model architecture (Figure 4) integrates feature embeddings, multi-layer perceptrons (MLPs), and attention mechanisms to learn both low and high-level patterns. The input is vectorized applicant data, which includes numerical

data (income, loan amount) and categorical features (loan grade), encoded via one-hot encoding (see Section 3.1). The model outputs a probability between 0 and 1 indicating the likelihood of default.

This hybrid deep learning approach offers several advantages over traditional methods:

- Non-linear modeling: Deep networks can capture complex interactions (e.g., how income modifies the impact of loan grade).
- Attention mechanisms: Help prioritize relevant features or feature groups, boosting performance and interpretability.
- Automatic feature learning: Embedding layers and MLPs reduce reliance on manual feature engineering.
- Group-wise attention: Offers transparency by highlighting how feature categories (e.g., loan vs. borrower info) influence predictions.

This report summarizes team progress so far, including data collection, baseline model and primary model development, and initial results. The following sections outline team contributions, data processing, baseline model evaluation, and primary model design, demonstrating that the team is on track to meet the project objectives.

2 Individual Contributions

The team uses the following tools for collaboration, development, and project tracking:

- Discord: primary platform for communication and coordination; used for sharing updates, asking questions, and resolving issues
- Google Colab: collaborative environment used for writing and testing code
- GitHub: version control for all code
- Excel logbook: tracks small daily tasks and personal progress
- Excel Responsibility tracker: documents major tasks, deadlines, and assignments

To ensure smooth collaboration and accountability, the team holds regular check-ins via Discord where members update progress and discuss blockers. The responsibility tracker and logbook are updated daily to maintain transparency. Version control with GitHub ensures that code changes are tracked and reviewed. So far, all members have contributed equally and met agreed deadlines without issue, and a summary of the work completed is provided in Table 1.

Table 1: Summary of team member contributions

Team Member	**Contributions**
Dharuniga	Handled data processing: feature cleaning, handling missing values, encoding, normalization, and data splitting. Wrote the data processing report section.
Dhanishika	Implemented and evaluated the baseline logistic regression model, including AUC-ROC and classification reports. Wrote the introduction, individual contributions, and baseline model sections.
Vishwa	Developed, trained, and tested the hybrid deep learning model. Computed metrics such as AUC-ROC, precision, recall, and accuracy. Analyzed feature attention for insights in the final report.
Arnav	Reviewed hybrid model code, assisted in model architecture decisions, and wrote the primary model section based on Vishwa's implementation.

The team is currently on track after completing all Phase 1 deliverables for the progress report ((see responsibility tracker). The team is now transitioning to training and fine-tuning the primary

deep learning model. Table 2 presents the updated project plan for Phase 2 and detailed task assignments. A more detailed plan can be found in the responsibility tracker (sheet 2). To ensure smooth progress and mitigate risks such as unexpected absences or delays, each person's set of tasks has been assigned at least one backup member who is familiar with the work and can step in if needed. Overall, the team collaborates effectively, with clear roles, timely updates, and transparent tracking of responsibilities and deliverables.

Table 2: Phase 2 project plan and work distribution with back-up member in parentheses

Task	**Dhanishika (DN)**	**Dharuniga (DR)**	**Vishwa (V)**	**Arnav (A)**	**Due Date**
Model Training & Tuning	Run and monitor training sessions, do hyperparameter tuning with random search and try architecture variations (test dropout, hidden units, varied regularization settings, layer depth, activation functions)				July 25, 2025
Model Evaluation & Analysis	Analyze ROC-AUC scores, plot trends (V)	Interpret class performance and results (DN)	Review error cases and data-label balance (A)	Analyze F1-scores and confusion matrices (DR)	Aug 1, 2025
Project Presentation	Baselines and Comparisons, Challenges, Lessons Learned (A)	Dataset, Features, and Preprocessing, Training Strategy, Hyperparameters (V)	Model Architecture Overview, Quantitative Metrics (DR)	Motivation and Problem Description, Model Evaluation Summary (DN)	Aug 15, 2025
Final Report Part A	Baseline Model, New Data Evaluation, Quantitative Results (DR)	Background, Data Processing, Qualitative Analysis (A)	Introduction, Model Training, Architecture, Illustration (DN)	Qualitative Results, Discussion, Ethics, Final Edits/Submit (V)	Aug 15, 2025
Final Report Part B	Individual contribution summary with task percentages and reflection				Aug 15, 2025

3 Notable Contributions

3.1 Data Processing

Our main primary dataset for our model is the Lending Club dataset (Mehta, 2020) which contains 855,969 records and 73 features, with severe class imbalance and multiple missing values.

3.1.1 Feature Cleaning

Table 3 summarizes features with missing values. Those with a large amount of missing/null values (over 98%) were removed entirely from the dataset. Features with moderate missingness were either imputed or a missing indicator feature was created when missingness was informative. A major challenge was dealing with missing values in some features that correlated strongly with the target variable (`default_ind`), for example `total_rev_hi_lim`. This was determined using:

```
df.groupby(df['total_rev_hi_lim'].isnull())['default_ind'].mean()
```

Listing 1: Code to determine if null values were significant

Table 4 shows that there is a higher occurrence of defaults in the null values which we wanted to preserve. Instead of only doing imputation, we created a missingness indicator feature to preserve this information. This is a new binary feature which represents if the value for the original feature was null. Other steps for feature cleaning and transformation include:

- Removing 27 features, including those with excessive missing data, and redundant categorical fields with high cardinality. Also Removing an additional 22 due to a data leakage issue.
- Converting date columns into numerical values by finding the number of years, months, and days since a chosen reference date (earliest date in the dataset), then normalized.
- Ordinal encoding features with ordered values like `grade` and one-hot encoding categorical features with low cardinality like `purose`.
- Normalizing numerical features using PowerTransformer (Yeo-Johnson) for improved model convergence. For highly varied data, where we could not use PowerTransformer, we used StandardScaler. Figure 1 shows the before and after of normalization for feature `total_acc`.

Table 3: Summary of missing values in dataset

Feature	Missing	Percentage
dti_joint	855,527	99.95%
il_util	844,360	98.64%
...	...	...
last_credit_pull_d	50	0.006%

Table 4: Default rate of null values in `total_rev_hi_lim`

total_rev_hi_lim Null	Default Rate
False (Not null)	0.0462
True (null)	0.1491

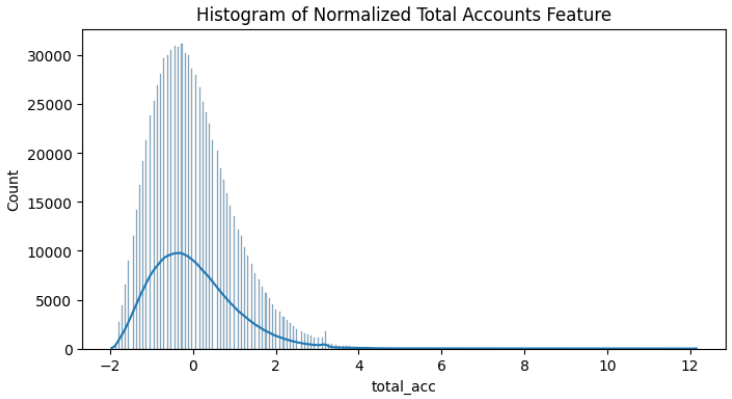

Figure 1: Normalization of `total_acc` feature where the curved line is after normalization.

The final dataset now has 47 features including the target `default_ind`.

3.1.2 Class Imbalance and Data Splitting

The original dataset was highly imbalanced so we applied random undersampling on the majority class to balance the dataset for training, which mitigates bias towards non-default predictions. After removing the target variable, the balanced dataset was split into training (70%), validation (15%), and test (15%) sets with stratified sampling to maintain class proportions:

- **Training set:** 65,053 samples (32,527 defaults, 32,526 non-defaults)
- **Validation set:** 13,940 samples (6,970 defaults, 6,970 non-defaults)
- **Test set:** 13,941 samples (6,970 defaults, 6,971 non-defaults)

Table 5 shows a sample record from the cleaned, processed training set after encoding and normalization. All features are numerical (scaled floats) or binary indicators, with no missing values.

3.1.3 Testing Plan

To validate the model, we will test on a separate dataset from Kaggle (Tse, 2020) which contains 32 581 records and 12 features. We will process the dataset similar to our current dataset by selecting overlapping features and applying consistent transformations, including handling missing values and scaling. We will also ensure proper indexing to align with our original data. Additionally, 15% of

Table 5: Cleaned data sample (first 5 rows with selected features)

Index	term_60 months	emp_length	pymnt_plan_y	loan_amnt	dti	default_ind
0	0	1.03	0	-1.157	0.547	0
1	1	-1.69	0	-1.453	-0.983	1
2	0	1.03	0	-1.465	-0.540	0
3	0	1.03	0	-0.563	0.108	0
4	1	-1.42	0	-1.394	-0.010	0

the original dataset is reserved as a backup holdout set. To further investigate the accuracy of our model in real world applications we removed some features like `loan_grade` from the backup set to simulate the missingness of features in real loan applications.

3.1.4 Challenges

Some of the challenges discussed throughout our data processing are summarized below.

1. Handling extreme missingness in features required removing many columns.
2. Null values for features were significant which required missing indicators.
3. Imbalanced target classes required undersampling.
4. Data leakage was identified when testing our baseline model (see section 3.2.3) which had an unusally high accuracy (99.7%) so we removed any information that would not be available until after an application is processed.

3.2 Baseline Model

We selected Logistic Regression (LR) as our baseline model due to its interpretability, simplicity, and common use in credit risk assessment (Stojiljković, 2024). As a binary classifier, LR produces probabilistic outputs indicating the model's confidence in each prediction. These probabilities support evaluation using the Receiver Operating Characteristic – Area Under the Curve (ROC-AUC), which measures the model's ability to distinguish between classes across all thresholds (scikit-learn developers, 2024). A higher ROC-AUC indicates better separability. LR also provides transparency through its learned coefficients, revealing which features most influence loan default predictions.

The baseline model was configured as follows:

- `penalty='l2'`: L2 regularization- penalizes large coefficients to prevent overfitting, default
- `C=1.0`: inverse strength of regularization- balances between under- and overfitting, default
- `class_weight='balanced'`: mitigates class imbalance by automatically adjusting weights inversely proportional to class frequencies to ensure minority classes aren't ignored
- `solver='liblinear'`: algorithm used to fit model, for small/medium datasets, default
- `max_iter=1000`: ensures convergence during training

`default_ind` is the binary target variable where 1 indicates a defaulted loan and 0 otherwise.

```
from sklearn.linear_model import LogisticRegression

logreg_model = LogisticRegression(class_weight='balanced', max_iter=1000)
logreg_model.fit(X_train, y_train)
```

Listing 2: LR model configuration and training (Stojiljković, 2024), full code in GitHub repository

3.2.1 Logistic Regression Model Quantitative Performance

Table 6 presents the results from the training, validation, test, and subset test, including ROC-AUC scores, accuracy, and F1 scores, which is the harmonic mean of precision (the proportion of positives

Table 6: Performance of logistic regression on different datasets

Dataset	ROC-AUC	Accuracy	F1 Score (Macro)
Training Set	0.813	0.738	0.738
Validation Set	0.817	0.741	0.741
Test Set	0.812	0.738	0.738
Subset Test (Limited Info)	0.773	0.705	0.703

that are actually positive) and recall (the proportion of positives that are correctly identified) (scikit-learn developers, 2024).

The baseline logistic regression model achieved stable ROC-AUC scores across splits: 0.813 (train), 0.817 (validation), and 0.812 (test) (Table 6). Precision, recall, and F1-scores were consistently around 0.740. However, on a limited-feature subset simulating missing credit history, performance dropped to an ROC-AUC of 0.773 and F1-score of 0.702, indicating sensitivity to missing features, particularly the absence of grade-related information. This suggests limitations in handling incomplete data. In contrast, a feedforward neural network can learn complex feature interactions and generalize better in such cases, potentially outperforming logistic regression by modeling latent structures and improving robustness.

3.2.2 Logistic Regression Model Qualitative Performance

The ROC curves below (Figure 2 and Figure 3) visualize the model's performance across the training, validation, and test datasets, as well as a subset test set where key features (e.g., grade and sub-grade) were removed to simulate limited customer information.

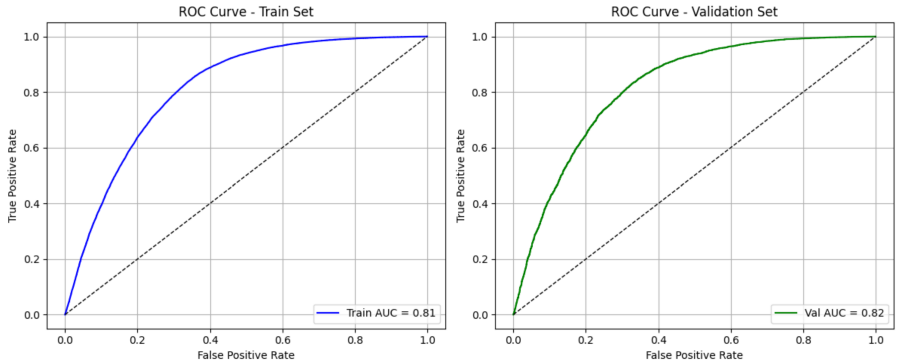

Figure 2: ROC Curves on Full Training and Validation Sets. Shows good performance on both.

The ROC curves show logistic regression performance across training, validation, test, and subset test sets. The training and validation curves (Figure 2) rise sharply toward the top-left, indicating strong separation and good generalization without overfitting. The full test ROC curve (Figure 3, left) closely matches, confirming model robustness on unseen data. In contrast, the subset test ROC curve (Figure 3, right), excluding grade and subgrade, is noticeably flatter, revealing reduced class separability. This highlights the importance of those features and suggests that more complex models, like deep learning, may be better suited for limited-information scenarios. For future testing, the data will be split into various samples (e.g., income $<$ \$80K and $>$ \$80K) and model performance will be compared.

3.2.3 Challenges

During the configuration and testing of our baseline model, we encountered several key challenges. One major issue was data leakage, where features containing future payment information (such as

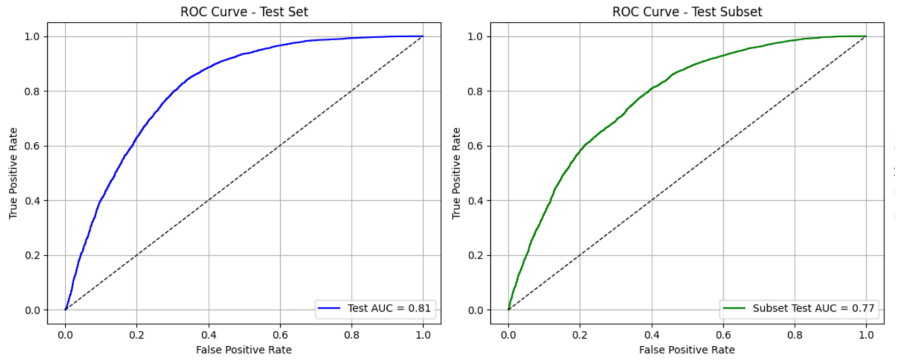

Figure 3: ROC Curves on Test Set and Test Subset with reduced features. Shows a noticeably flatter curve on the reduced test dataset.

`total_pymnt`) were initially included in the training data. This led to an unrealistically high ROC-AUC score (0.997) in early tests before those features were identified and removed.

We trained the model on the full dataset and tested it on a limited subset to assess generalization and performance in edge cases lacking full credit history. Aligning features in the limited subset required careful re-indexing to match the training structure, which we resolved using the following line of code to maintain feature alignment despite dropped columns.

```
# Make sure the columns are aligned in reduced test dataset
X_test_subset_aligned = X_test_subset.reindex(columns=X_train.columns,
    fill_value=0)
```

Listing 3: Fixes alignment issues after using drop to reduce dataset features

Logistic Regression served as a strong, interpretable benchmark with ROC-AUC consistently above 0.81 on full datasets and fair performance (AUC: 0.77) on limited feature sets. This establishes a good performance baseline so that future deep learning models can aim to outperform this baseline, especially in edge cases with incomplete feature datasets.

3.3 Primary Model

As mentioned in Section 1, the primary model design is that of a Hybrid Attention Network consisting of both a multi-head attention stream reminiscent of that employed in transformer models as well as a standard multi-layer perceptron (MLP) stream that form the basis of all deep learning models. Figure 4 shows the detailed architecture of the model, including the exact number of layers in each branch, the number of weights, and other specific parameters. In summary:

- The attention branch consists of one embedding layer with 32 weights and a bias of 32, resulting in a total of 64 parameters for this layer.
- The attention branch also contains a multi-head attention layer which consists of three linear layer streams, commonly denoted as V, K, and Q, and a "scaled dot product attention" layer set. Each of these consists of four layers, as the number of "heads" for this model is 4.
- The total number of parameters in the attention branch is 5408.
- The three hidden layers in the MLP stream contain 64, 32, and 16 neurons respectively, which means they have bias values of 64, 32, and 16 and weight values of $64 \times \texttt{input\_dim}$, $64 \times 32 = 2048$, and $32 \times 16 = 512$ respectively.
- The total number of parameters in the MLP is $64 \times \texttt{input\_dim} + 2672$.
- The total number of parameters in the fusion layer, which consists of one concatenation linear layer and one sigmoid linear layer, is $2048 \times \texttt{input\_dim} + 1153$.
- The model outputs a single prediction of the risk of a candidate defaulting on their credit.

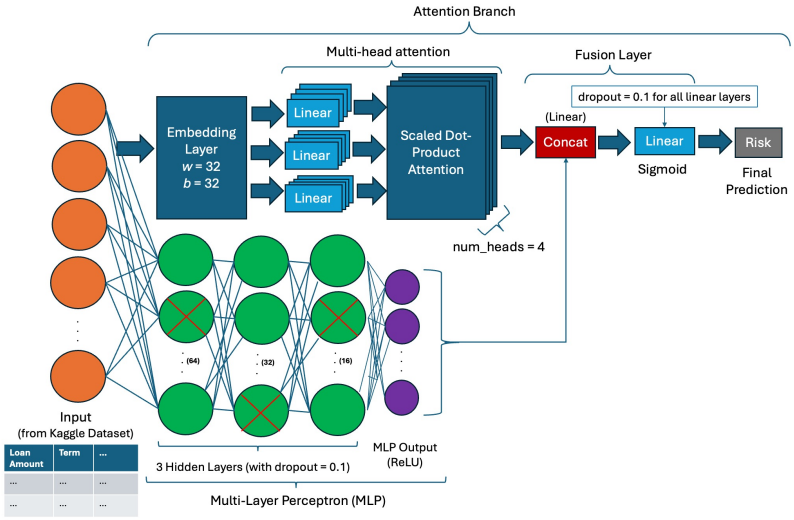

Figure 4: Detailed architecture of the model showing dataset input, the Hybrid Attention Network with parallel Multi-Head Attention and Multi-Layer Perceptron streams, and the output credit risk probability. "Linear" layers in the attention stream represent multiple fully-connected layers, abstracted for simplicity.

3.3.1 Primary Model Quantitative Performance

Table 7 presents the performance of the model on the training, validation, test, and subset of the test set, including ROC-AUC scores, accuracy, and F1 scores, which is the harmonic mean of precision (the proportion of positives that are actually positive) and recall (the proportion of positives that are correctly identified) (scikit-learn developers, 2024). From these results, it is clear that the primary model does indeed outperform the baseline model in all metrics by a significant margin with an AUC > 0.820; however, there is still room for improvement in the performance, which will be continuously explored as it is refined further.

Table 7: Performance of primary model on each dataset

Dataset	ROC-AUC	Accuracy	F1 Score (Macro)
Training Set	0.8324	0.7601	0.7779
Validation Set	0.8274	0.7577	0.7725
Test Set	0.8217	0.7531	0.7738
Subset Set	0.8195	0.7545	0.7758

3.3.2 Primary Model Qualitative Results

Figures 5 and 6 depict the learning curves of the primary model on the training, test, and validation set as well as on a specific small class of data known as the subset test set (which was defined in Section 3.2.2). As was for the baseline model, the training and validation curves rise sharply toward the top-left, indicating strong separation and good generalization without overfitting. The full test ROC curve closely matches this, confirming model robustness on unseen data, and this time, unlike for the baseline model, the subset test ROC curve also maintains the same sharper shape of the full test curve, confirming the efficacy of the deep learning approach on such limited-information scenarios, as was predicted earlier. In the future, further qualitative analysis will also be conducted

by examining specific cases from the subset test set to analyze the model's specific performance on these singular cases.

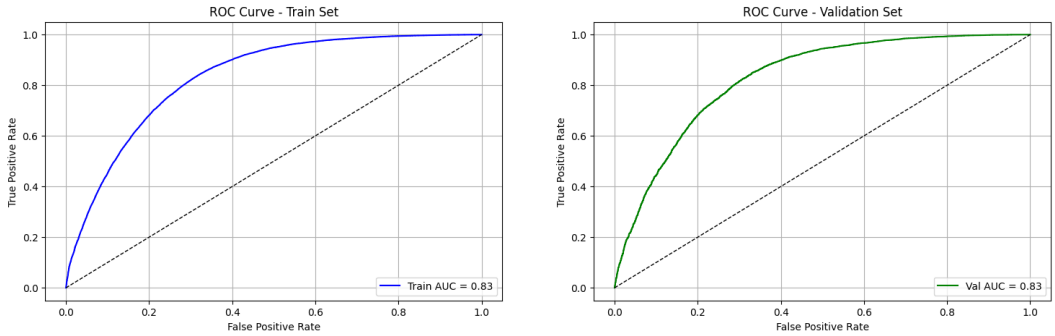

Figure 5: ROC Curves on Full Training and Validation Sets for the Primary Model

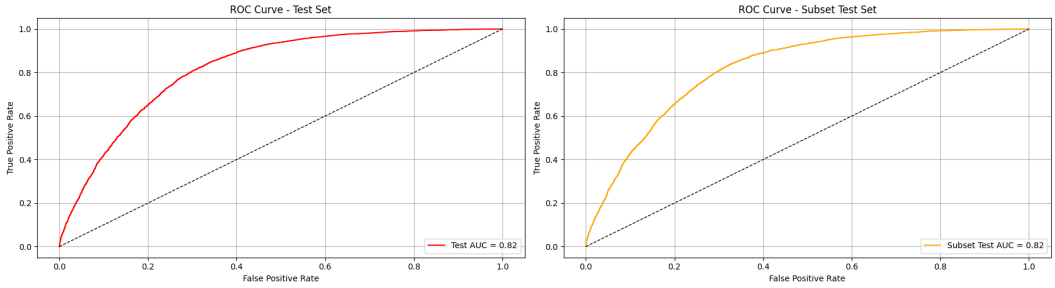

Figure 6: ROC Curves on Test Set and Test Subset with reduced features for the Primary Model

To further observe the model's performance on select features, a feature-attention analysis was conducted on the subset test, which concluded that the model attributed the most attention to the `earliest_cr_line_year`, `issued_year`, and `emp_length` features, which were not features that had been removed to form the subset test set, further verifying the model's ability to adapt well to limited information scenarios. Regardless, the model seemed to attribute lower attention (almost the same as `grade` and `sub_grade` to features such as `loan_amt` and `int_rate` (interest rate), indicating that, had these features been the only ones present in the dataset, the model may not have performed as well. The full analysis is provided in the Google Colab (end of page).

3.3.3 Challenges

Several challenges were faced during the development of the primary model; however, two key challenges stand out.

Firstly, the design of the attention mechanism, specifically for tabular data like the dataset employed here, proved challenging as most attention-based models are built for sequences (e.g., text, time series). Thus, applying attention to tabular data — where features are not ordered — required careful adaptation, since choosing to treat each feature as a token and applying self-attention across features was non-trivial. However, this was overcome through structural experimentation, and eventually an appropriate mechanism was achieved.

Secondly, the integration of both the MLP and multi-head attention branches to form the hybrid network proved difficult, as combining the outputs of both an attention branch and an MLP branch introduced critical design challenges, especially around dimensionality matching, fusion strategy, and training stability. Ensuring both branches contributed meaningfully (rather than one dominating) required tuning and a modular architecture, and employing these techniques allowed for this challenge to also be overcome.

REFERENCES

Ramesh Mehta. Credit risk analysis dataset. https://www.kaggle.com/datasets/rameshmehta/credit-risk-analysis, 2020. Accessed: 2025-06-12.

scikit-learn developers. roc_auc_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html, 2024. Accessed: 2025-07-10.

Mirko Stojiljković. Logistic regression in python. https://realpython.com/logistic-regression-python/, 2024. Accessed: 2025-07-10.

Lao Tse. Credit risk dataset. https://www.kaggle.com/datasets/laotse/credit-risk-dataset, 2020. Accessed: 2025-07-09.