Modeling & Statistical Evaluation

Introduction

In this step, our primary objective is to select the final model for deployment and evaluate its statistical results and implications. To achieve this, we have developed several cascading models that combine classification and regression techniques, applying the regression models to test data labeled by our classifier models. This approach follows the poor performance of our regression models in the previous step. By leveraging the classification model, we aim to reduce data dispersion and enhance our regression capabilities.

We have explored various combinations of these models and will analyze the results in this working paper to select the best cascading model.

In this paper, we will detail each step of implementing the cascading models, discuss the pitfalls encountered, and present our conclusions and future direction (see Step5.ipynb for complete code).

Breakdown of our Workflow

Classification Models- In the current phase, we are considering two classification models based on their performance in the previous phase: Logistic Regression (AUC 0.7) and XGBoost (AUC 0.91), as detailed in appendix 1. These models were selected after conducting forward sequential feature selection using features identified in an earlier step. In that step, we utilized grid search to determine the optimal hyperparameters for each model. This process involved cross-validation across multiple folds to systematically evaluate various hyperparameter combinations. Our evaluation metric for selecting the best hyperparameters for the classification models was the weighted F1 score, as specified in the appendix.

Addressing class imbalanced- Our data is divided into two classes that are not equally represented: 79.8% of the loans yield above 2%, while only 20.2% yield less than 2% see appendix . To prevent our classification models from becoming biased towards the majority class, we applied inverse proportional weighting based on class frequencies for Logistic Regression and manually applied the weights to the XGBoost see appendix 2 for details . This approach assigns higher weights to the minority class and lower weights to the majority class during training, ensuring adequate representation of the minority class and improving performance on the test data.

Classification Threshold tuning- We designated the less prevalent class as the True class (below 2%). Therefore in our step ensuring financial prudence involves prioritizing the minimization of false negatives over potentially losing loans that could be profitable. This strategy favors investing in loans that may yield less return but are more likely to be repaid.

Our aim is to optimize the classification threshold to maximize True Negatives (TN) — loans correctly predicted to have a return higher than 2% — while minimizing False Negatives (FN) — loans incorrectly predicted to exceed the 2% return threshold when they do not while keeping a reasonable amount of loans pool.

By fine-tuning these thresholds, we strike a balance that allows our models to accurately identify loans with positive returns while reducing the risk of misclassifying loans that will not be fully paid . This equilibrium is critical for enhancing the effectiveness and reliability of our loan performance predictions.

We've set a naive requirement to maintain at least 50,000 instances classified as True Negatives (TN) for our regression model to ensure sufficient data for training and evaluation.

Accordingly, we have set the Logistic Regression threshold at 0.31 and the XGBoost threshold at 0.21. These thresholds enable us to retain a sufficient number of True Negatives while managing the False Negative Rate within acceptable limits.

Despite Logistic Regression demonstrating superior overall performance metrics (such as F1 score, accuracy, and recall) in prior assessments, the XGBoost model at the 0.21 threshold better aligns with our specific requirement to minimize false negatives and optimize loan selection for potential profitability.

Regression models - We assessed two regression models, linear regression and random forest regression, based on their superior performance in earlier stages. The test and train datasets were prepared using predictions from these classifiers. Employing a grid search alongside cross-validation, we systematically explored diverse hyperparameter combinations to identify optimal settings for enhanced performance. Subsequently, we conducted experiments with varying thresholds to evaluate their influence on metrics such as MSE, R2, and learning curves (refer to the appendix for detailed results).

Cascading Models Results

After determining the optimal threshold for each of our classification models, we proceeded to examine the results of various cascading models. We tested four different combinations where the regression model is based on the results of the classification model. Specifically, we predicted by Random Forest Regression and Linear Regression the expected return of the loans that were classified as 0 (indicating a yield of more than 2% return) see appendix 4 for the results.

This process involved adding the classification results as a new column for the regression model. The combinations we examined were:

- Logistic Regression -> Linear Regression
- Logistic Regression -> Random Forest Regression
- XGBoost -> Linear Regression
- XGBoost -> Random Forest Regression

Our findings indicate that the models which utilized the XGBoost classification results for predicting expected returns consistently outperformed those based on Logistic Regression. This superior performance was evident across both regression models (Linear Regression and Random Forest Regression).

The ultimate model, which combined XGBoost with Linear Regression, demonstrated the highest R-squared value (0.141851) and the lowest Mean Squared Error (MSE=0.003848). This represents an almost 90% reduction in MSE compared to the best-performing regression model from previous steps, which was Linear Regression with an MSE of 0.017723.

Benchmark models see appendix 5 for results **:** In our cascading models, we used classification models to predict whether a loan would yield above or below 2%. For those predicted to yield above 2%, we calculated the expected return using regression models. To benchmark our cascading models, we created two different baseline models:

Grade-Based Baseline Model- Instead of using a classification model, we directly selected all loans in Grade A, which has the lowest percentage of loans yielding under 2%.We applied regression models to these loans to predict their expected returns.

Simple Benchmark Model- We filtered the dataset to include only loans predicted by XGBoost to yield above 2%.This model predicts the average expected return from the training data for all test instances.

By establishing these baseline models, we provide straightforward points of reference for evaluating the performance of our more complex cascading models. This allows us to better assess the added value and effectiveness of our cascading models in predicting loan returns.

For both benchmarks, our cascading models significantly outperformed in terms of R-squared (R^2) and Mean Squared Error (MSE), demonstrating their superior predictive power and accuracy.

Pitfalls

- **Sequential Model Bias** - Using linear regression on the results of logistic regression could introduce bias, as the data has already undergone extensive processing and analysis.
- **Selective Loan Filtering**- Our business approach may inadvertently filter out potentially good loans, limiting our pool of investment opportunities.
- **Dependence on Initial Estimates-** Our calculations heavily rely on our initial estimates of loan yields, potentially overlooking profitable opportunities.
- **Absence of Risk Assessment-** Risk assessment hasn't been integrated into our models yet, which could affect the accuracy of loan performance predictions.
- **Handling Class Imbalance**- Although you've applied techniques like weighted sampling to address class imbalance, ensuring that these methods effectively generalize to new data and maintain performance stability over time is crucial. Ignoring the evolving nature of class distributions can lead to degradation in model performance.

Summary:

In this stage, we aimed to further optimize the results obtained in the previous phase by combining our classification models' predictions with our regression models. We compared the outcomes of our models, including two benchmark models, and presented a comprehensive analysis of the results. The model that will continue with us to the next step for deployment is the XGBoost -> Linear Regression cascading model, as it demonstrated the highest performance in terms of R-squared value and Mean Squared Error (MSE).

Next Step

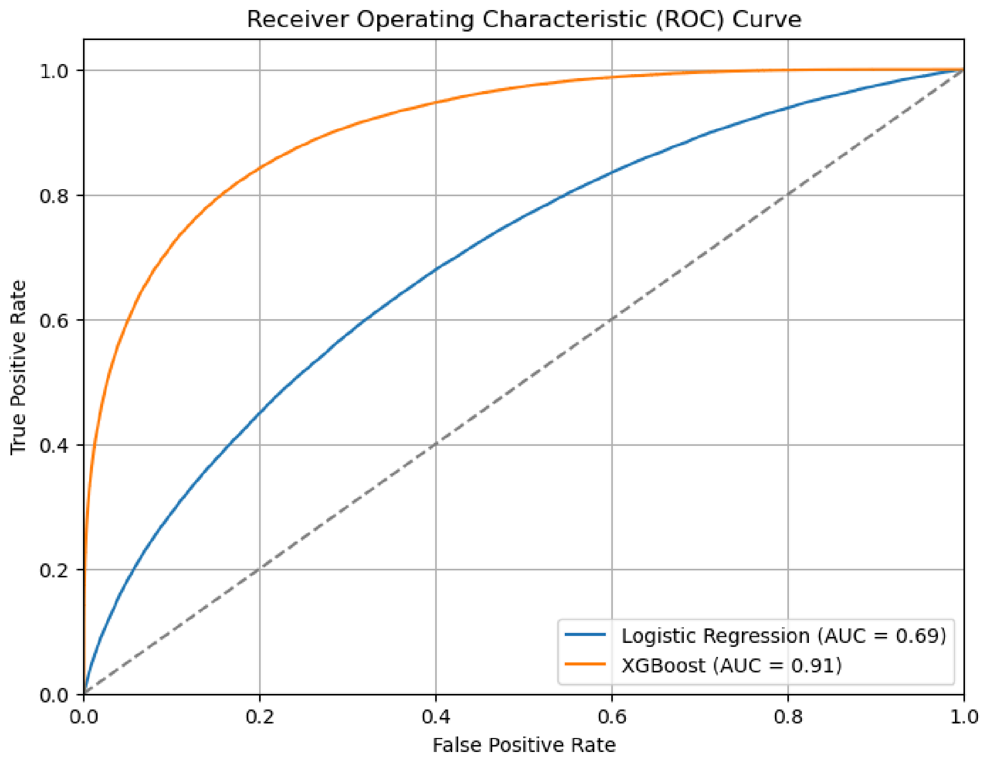
Moving forward, we will implement monetization based on our model's results and discuss the associated risks to fully assess the business implications of deploying our model.

Thanks for your time,

Team B

Appendixes

Appendix 1 - Evaluation of classification models from prior step

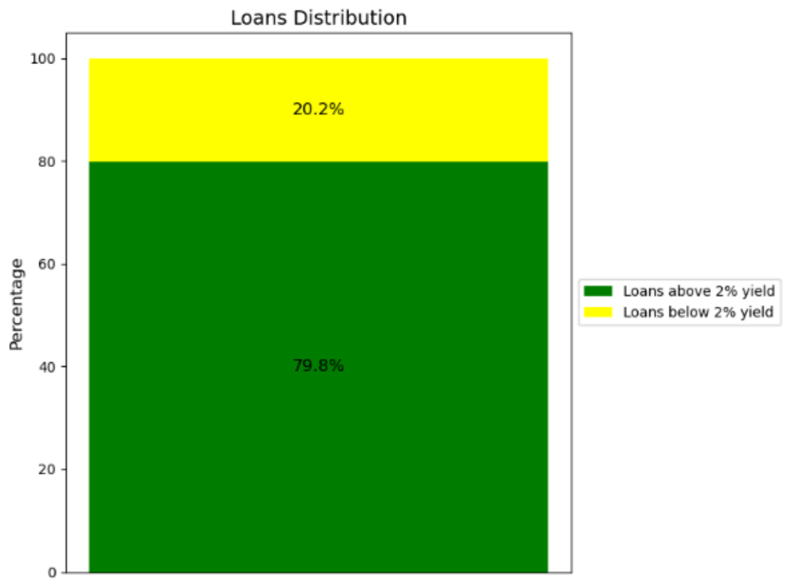


This table provides a clear comparison of the performance metrics and best hyperparameters for the Logistic Regression and XGBoost models.

Metric	Logistic Regression	XGBoost
Average F1 Score	0.6939402672129585	0.7034755364338819
Average Precision	0.7595586491602211	0.7598267661428789
Average Accuracy	0.6634183849606127	0.6755327495841572
Average Recall	0.6634183849606127	0.6755327495841572
Best Hyperparameters	{'C': 1.7575106248547894}	{'learning_rate': 0.3, 'max_depth': 9}
Confusion Matrix (TN, FP, FN, TP)	[[34738, 16261], [5188, 7539]]	[[35728, 15271], [5406, 7321]]

The table compares the performance metrics and best hyperparameters of the Logistic Regression and XGBoost models, showing that XGBoost slightly outperforms Logistic Regression in F1 score, accuracy, and recall, while both models have similar precision.

<u>Appendix 2-Addressing class imbalanced:</u>



We applied to each model a different balancing method to the train data. For the logistic regression we used its built in arguments :

solver='liblinear', penalty="l2", class_weight='balanced'

For XGboost we used the following technique:

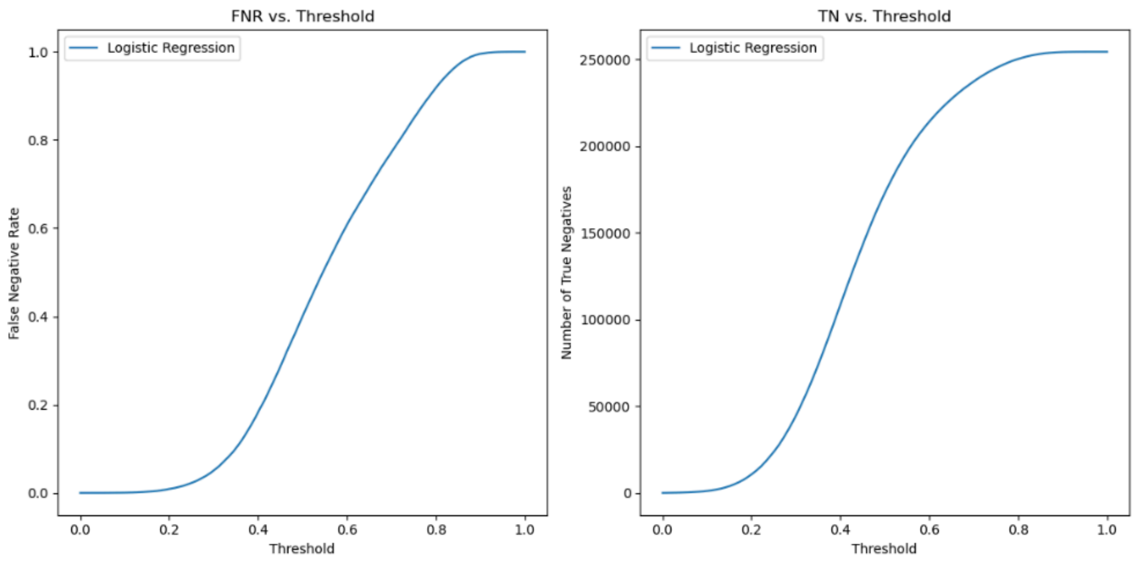
- For the `XGBClassifier`, we manually calculated class weights based on the training data distribution. The weights were computed as follows:

$$\text{class\_weights} = \left\{ 0 : \frac{\text{len}(y\_train)}{2 \cdot \sum(y\_train == 0)}, 1 : \frac{\text{len}(y\_train)}{2 \cdot \sum(y\_train == 1)} \right\}$$

- These weights were then applied during model fitting to ensure balanced learning.

Appendix 3 threshold tunning

logistic regression



Working Paper, Step 5, Group B

```
Logistic Regression Results:
Threshold       FNR     TN
0.15            0.0027  3945
0.16            0.0034  4928
0.17            0.0044  6117
0.18            0.0054  7505
0.19            0.0072  9151
0.20            0.0090  10981
0.21            0.0110  13060
0.22            0.0134  15391
0.23            0.0163  18107
0.24            0.0196  21116
0.25            0.0233  24348
0.26            0.0274  27795
0.27            0.0327  31800
0.28            0.0385  36087
0.29            0.0449  40626
0.30            0.0525  45631
0.31            0.0609  50996
0.32            0.0706  56620
0.33            0.0809  62447
0.34            0.0916  68650
```

As presented above, as the threshold increases, the number of instances classified as True (less than 2% return) decreases. This means fewer loans are predicted to yield less than 2%, resulting in a higher number of True Negatives (TN) and False Negatives (FN).

We set a requirement to maintain at least 50,000 instances classified as True Negatives (TN) for our regression model to ensure sufficient data for training and evaluation. According to the threshold tuning results:

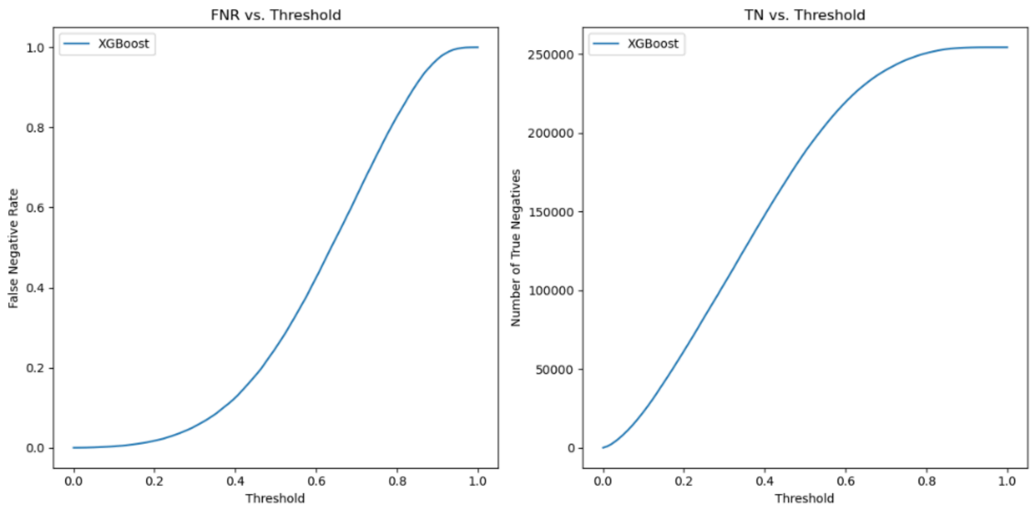
- At a threshold of 0.31, the number of True Negatives (TN) is 50,996 which aligns with our requirement.
- The False Negative Rate (FNR) at this threshold is 0.0609, indicating that about 6.09% of the actual high-return loans are misclassified.

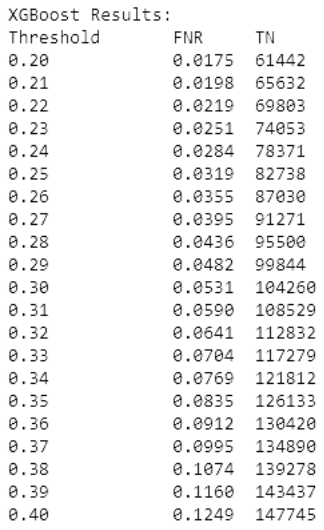
Based on these observations, we have chosen the threshold of 0.31. This threshold allows us to maintain an adequate number of True Negatives for our regression model while keeping the False Negative Rate at an acceptable level. The selection process is supported by the following data:

Logistic Regression Results at Threshold 0.31:

- Threshold: 0.31
- FNR: 0.0609
- TN: 50,996

<u>Results threshold- XGboost:</u>





```
XGBoost Results:
Threshold       FNR     TN
0.20            0.0175  61442
0.21            0.0198  65632
0.22            0.0219  69803
0.23            0.0251  74053
0.24            0.0284  78371
0.25            0.0319  82738
0.26            0.0355  87030
0.27            0.0395  91271
0.28            0.0436  95500
0.29            0.0482  99844
0.30            0.0531  104260
0.31            0.0590  108529
0.32            0.0641  112832
0.33            0.0704  117279
0.34            0.0769  121812
0.35            0.0835  126133
0.36            0.0912  130420
0.37            0.0995  134890
0.38            0.1074  139278
0.39            0.1160  143437
0.40            0.1249  147745
```

As the threshold increases, the number of instances classified as True (less than 2% return) decreases. This means fewer loans are predicted to yield less than 2%, resulting in a higher number of True Negatives (TN) and False Negatives (FN).

We set a requirement to maintain at least 50,000 instances classified as True Negatives (TN) for our regression model to ensure sufficient data for training and evaluation. According to the threshold tuning results:

- At a threshold of **0.21**, the number of True Negatives (TN) is **65,632**, which exceeds our requirement.
- The False Negative Rate (FNR) at this threshold is **0.0198**, indicating that about 1.98% of the actual high-return loans are misclassified.

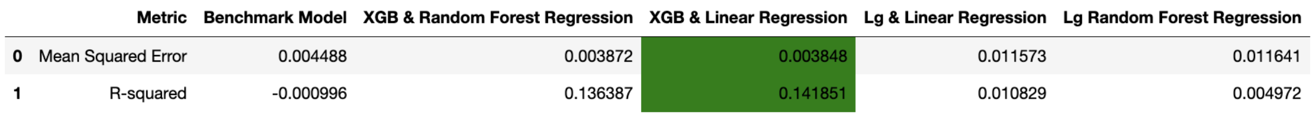
Based on these observations, we have chosen the threshold of **0.21**. This threshold allows us to maintain an adequate number of True Negatives for our regression model while keeping the False Negative Rate at an acceptable level. The selection process is supported by the following data:

XGBoost Results at Threshold 0.21:

- **Threshold**: 0.21
- **FNR**: 0.0198
- **TN**: 65,632

Appendix 4 - Cascading results

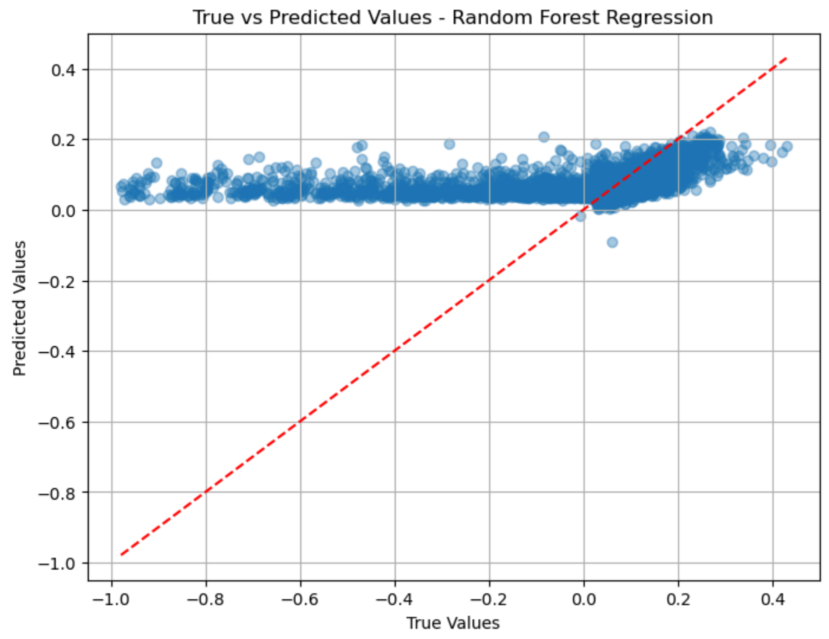
Linear regression based on Logistic regression's predictions:

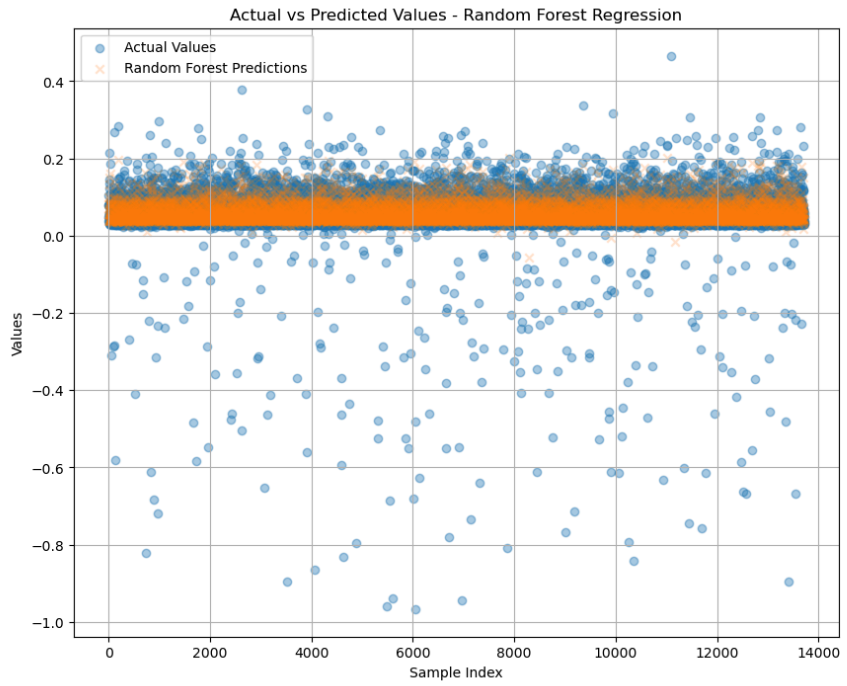


	Metric	Benchmark Model	XGB & Random Forest Regression	XGB & Linear Regression	Lg & Linear Regression	Lg Random Forest Regression
0	Mean Squared Error	0.004488	0.003872	0.003848	0.011573	0.011641
1	R-squared	-0.000996	0.136387	0.141851	0.010829	0.004972

Best model Prediction Vs True Value

Working Paper, Step 5, Group B





Based on the model prediction plots provided, here are two insights regarding the Random Forest Regression model's predictions:

Model Bias and Variance:

The "True vs Predicted Values" plot shows that the predictions from the Random Forest model are highly concentrated around 0. This indicates that the model has a strong bias toward predicting values close to zero. The predicted values do not vary much and fail to capture the actual range of the true values, especially for values that deviate significantly from zero. This suggests that the model is underfitting the data, as it does not learn the relationship between the features and the target variable well enough to make diverse predictions.

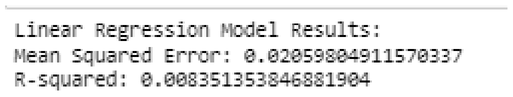
Prediction Spread and Errors:

The "Actual vs Predicted Values" plot further emphasizes the lack of variation in the model's predictions. The actual values (blue dots) show a wide spread across the range, while the predicted values (orange crosses) remain tightly clustered around zero. This clustering indicates that the model is not capturing the underlying patterns in the data, leading to significant errors for many samples. The predictions are not reflecting the true values accurately, suggesting that the model's ability to generalize is poor, and it fails to provide meaningful predictions for different inputs.

To improve the model we'll continue tuning the hyper parampers and threshold of the classification as Its evident that there is a section below 0 if the model fails to make good predictions. Its still important to mention that the model outperforms both of the baselines presented below.

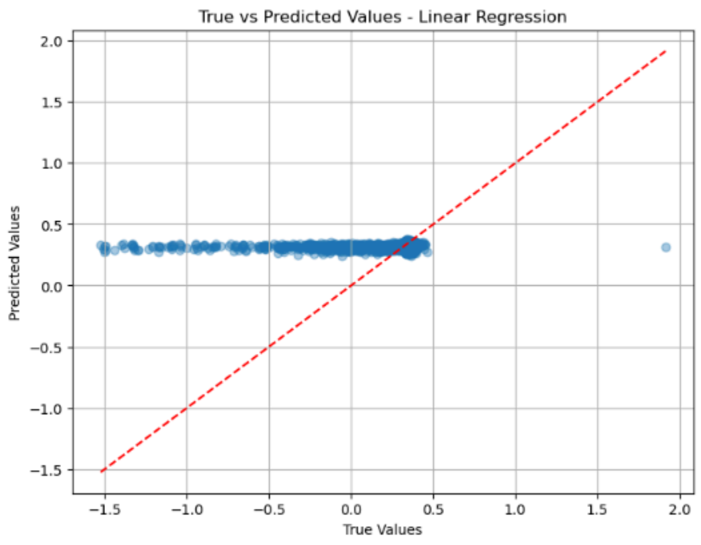
<u>Appendix 5- Benchmark results x</u>

<u>Grade-Based Baseline Model- linear regression</u>



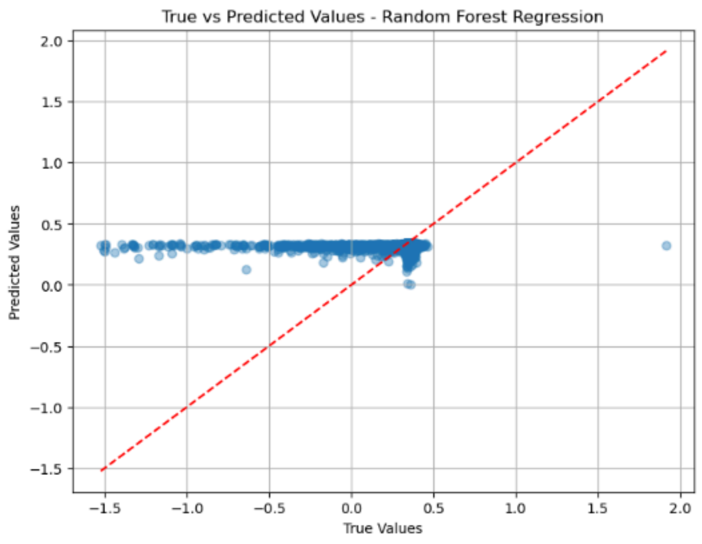
```
Linear Regression Model Results:
Mean Squared Error: 0.02059804911570337
R-squared: 0.00835135384681904
```

Working Paper, Step 5, Group B



<u>Grade-Based Baseline Model- random forest regression</u>

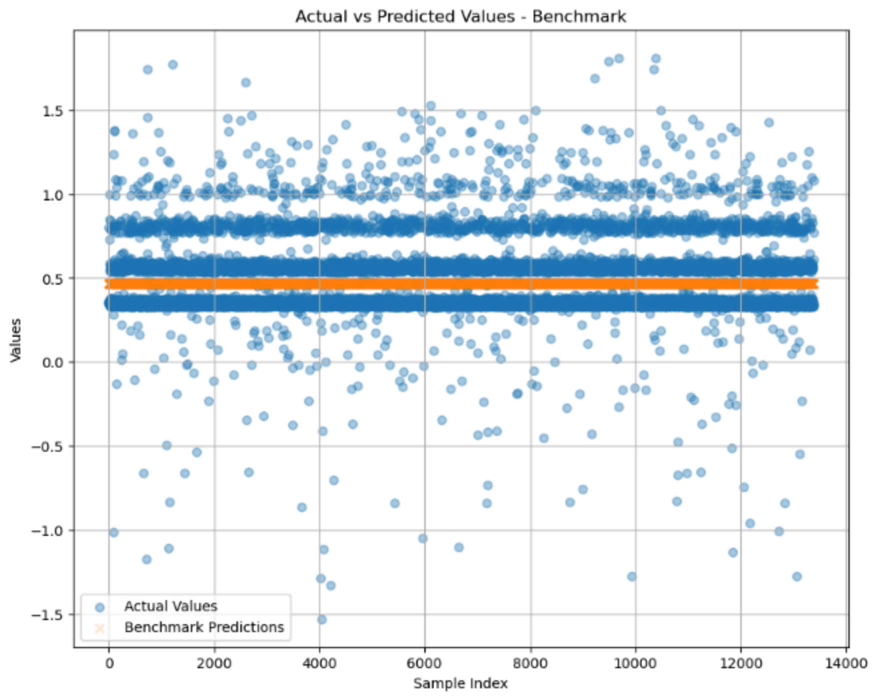
```
Random Forest Regression Model Results:
Mean Squared Error: 0.02068922073927068
R-squared: 0.003962092680921292
```



<u>Simple Benchmark Model:</u>

```
Benchmark Model Results:
Mean Squared Error: 0.043661325524282826
R-squared: -0.00011979722513189017
```

Working Paper, Step 5, Group B


Actual vs Predicted Values - Benchmark
Values
Sample Index
Actual Values
Benchmark Predictions