Machine Learning Models to Predict Social Impact Bonds Outcome

Serena Yin, Nadia Bozeman, Cyra Alesha, Sophie Hollowell

Background

Social Impact Bonds (SIBs) are a financing mechanism designed to fund social programs through a performance-based approach. Unlike traditional government-funded initiatives, SIBs transfer the financial risk of project failure from the public sector to private investors. In a typical SIB, private investors provide upfront capital to organizations implementing a social program. If pre-agreed outcomes are achieved—as validated by a third party—the government or an outcome funder repays the investors, often with a return. However, if outcomes are not met, investors may lose some or all of their investment.

Thus, SIBs enable experimentation and scaling of social programs while encouraging accountability, outcome tracking, and evidence-based decision-making. SIBs have been used to address a wide range of issues, from early childhood education to homelessness. Despite their promise, SIBs are complex to structure and evaluate, and relatively few have reached maturity globally

Problem Statement

The primary challenge facing the development and scalability of Social Impact Bonds lies in the uncertainty and difficulty of predicting program outcomes. These uncertainties impact multiple facets of SIB design—most notably, how to price the bond, assess the associated risk, and determine expected returns for investors. Policymakers, investors, and social organizations often lack robust, data-driven frameworks to estimate bond values and forecast success based on project and contextual features.

As such, for our project, we aim to leverage historical SIB data to:

1. Predict the likelihood of SIB success
2. Estimate the expected return profile of new SIBs based on the likelihood of success and project characteristics

Overall, we hope to develop a trading strategy that would help advise impact investors on whether or not they should invest in an SIB based on the likelihood of the success of the bond and the predicted price of the bond.

Data Overview

To support the analysis, we are utilizing the Indigo Impact Bond Dataset, curated by the Government Outcomes Lab at the University of Oxford. The dataset includes detailed records of 314 individual SIB projects from around the world and contains structured information on:

- **Project Characteristics**: Name, policy sector, development stage, investment amount, maximum return/loss, outcome payments, number of beneficiaries, associated SDG goals
- **Outcomes & Results**: Defined outcome metrics, targets, validation methods, and text description of outcomes achieved
- **Investments & Repayments**: Types of investment instruments used, repayment structures, total commitments, and outcome fund attributes.
- **Timeline & Status**: Start and end dates, project duration, and current status (e.g., implementation, completed).

However, many of the records within the dataset are incomplete. As such, we elected to generate synthetic data to facilitate predictive modeling. The goal was to preserve the structural integrity of the schema while embedding realistic statistical patterns that would support the development of machine learning models. This approach allowed us to simulate a robust dataset aligned with real-world logic, despite missing values in the original source.

We based our synthetic generation on four core tables modeled after the INDIGO schema: (1) Projects.csv, (2) Projects_Outcome_Metrics.csv, (3) Projects_Investments.csv, (4) Projects_Outcome_Payments.csv.

We retained the original project IDs and project names where available, to maintain schema consistency and allow simulated relationships across tables.

Below is table-specific generation:

- **Projects.csv:** Simulated each project's key metadata (dates, policy areas, populations), financials, and success-related attributes using structured logic and conditional distributions.
- **Projects_Outcome_Metrics.csv:** Each project was assigned a single outcome metric. The outcome target was set as a function of the number of beneficiaries, while pricing varied by policy sector.
- **Projects_Investments.csv:** The maximum potential outcome payment was computed as the product of outcome pricing and number of beneficiaries, plus minor noise.
- **Projects_Outcome_Payments.csv:** The payment amount was determined by the number of outcomes achieved (a function of success and IRR), multiplied by outcome pricing. This introduced temporal and financial dependencies.

All datasets were generated to contain no missing values and to support both classification and regression modeling. By embedding these relationships, the synthetic data became suitable for exploring real-world analytical scenarios, such as predicting project success, estimating IRR, and modeling outcome-based payments.

Methodology

To access the code and all other files for this project, please see here.

To develop a trading strategy that impact investors can utilize when determining whether to invest in a social impact bond, we broke the strategy into two-fold: (1) Predicting the success of the SIB overall and (2) Predicting the internal rate of return for successful SIBs.

As the government does not need to pay investors back if the outcomes of the social project are not deemed achieved, the risk is thereby shifted to the impact investors. As such, the first step for impact investors should be to determine whether or not the social project they are investing in is likely to succeed, as this would determine whether or not they will be paid back and earn a return on their investment.

Given the likelihood of success of the SIB overall, impact investors should then explore whether or not the price and return of each outcome tracked within the SIB would be within their budget and capital resources, as this would directly dictate whether or not impact investors would be able to achieve the return they are looking for while minimizing the risk.

As such, the trading strategy we develop will be broken into two steps to better address the various factors that impact the risk profile of each SIB.

I. Predicting the likelihood of SIB success

We implemented and compared three supervised classification models: Logistic Regression (with Lasso regularization), Random Forest, and XGBoost.

- **Logistic Regression** served as a baseline model due to its interpretability and ease of implementation.
- **Lasso Regularization** (L1 penalty) was applied to Logistic Regression to perform feature selection and mitigate multicollinearity.
- **Random Forest** was chosen for its ability to capture nonlinear interactions and handle noisy features with minimal tuning.

- **XGBoost** was used as a gradient boosting algorithm optimized for tabular data, providing improved performance through boosting and robust handling of class imbalance.

All models were trained on a common dataset derived from multiple sources, containing financial, contractual, and timing-related features for each SIB project.

To ensure that our model simulated a realistic pre-investment decision-making process, we strictly limited inputs to features that would be available before an investment was made.

- Included Features are as follows: Overall Project Finance – maximum potential loss, Overall project finance – maximum potential return, Maximum potential outcome payment, potential_return_to_loss, Service and beneficiaries - Targeted number of unique service users or beneficiaries (total) - (Value), roi_estimate
- Excluded Features (to avoid leakage): Actual IRR, outcome payments, investor repayment amounts, Actual number of service users, Duration metrics based on actual project completion.

By excluding outcome-based fields such as Actual IRR, we ensured that the models would not inadvertently "peek" at future outcomes when making predictions, successfully avoiding data leakage and overfitting.

During the development process, several challenges were encountered and addressed:

- **Data Leakage**: Initially, the target variable (project success) was defined using actual IRR, which was mistakenly included in the feature set. This led to artificially inflated performance metrics. The issue was resolved by removing all features derived from actual IRR values before model training.
- **Class Imbalance**: The dataset contained significantly more successful projects than unsuccessful ones. We addressed this using **stratified train-test splits**, **class weighting** in the models (Random Forest and XGBoost), and evaluating metrics beyond accuracy, such as **precision**, **recall**, and **ROC AUC**.
- **Multicollinearity**: Financial data contained several overlapping or highly correlated variables. We mitigated this by:
    - Dropping redundant fields (e.g., different currency representations of the same value)
    - Applying **Lasso regularization**, which effectively shrinks less informative coefficients to zero

Each model was trained using appropriate techniques to balance performance and interpretability:

- **Logistic Regression with Lasso**:
  Tuned using LogisticRegressionCV with 5-fold cross-validation to select the optimal regularization strength. Scaling was handled via a pipeline using StandardScaler.
- **Random Forest**:
  Used 100 estimators and default parameters, with class_weight='balanced' to account for class imbalance. Feature importance scores were used to interpret model behavior.
- **XGBoost**:
  Trained with n_estimators=100, max_depth=4, learning_rate=0.1, subsample=0.8, and a scale_pos_weight reflecting the ratio of class imbalance. Evaluation metric was set to logloss.

All models were evaluated using a common holdout test set and assessed with the following metrics: **Accuracy, Precision, Recall, F1-Score, ROC AUC Score, Strategy performance**, which was based on simulated investment decisions using a threshold on predicted probabilities (e.g., invest if predicted probability of success > 0.7).

II. Estimating the internal rate of return of SIBs

For the baseline method, we developed a naïve average predictor, which assumes that all Social Impact Bonds perform similarly to the historical mean. The naïve average predictor utilizes the mean internal rate of return in the training dataset and uses the mean as the prediction outcome for the test dataset.

We implemented and compared 7 supervised regression models: Linear Regression, Lasso Regression, Ridge Regression, Decision Tree Regressor, Decision Tree with Bagging, Random Forest Regressor, and XGBoost.

- **Linear Regression** served as a baseline model due to its interpretability and ease of implementation.
- **Lasso And Ridge Regularization** (L1 and L2 penalties) were applied to Linear Regression to perform feature selection and mitigate multicollinearity.
- **Decision Tree** was implement as a simple, interpretable nonlinear model to explore interactions between features without requiring feature scaling or transformation
- **Bagged Trees** was applied to the Decision Tree Model to address its tendency to overfit and potentially improve generalization performance
- **Random Forest** was chosen for its ability to capture nonlinear interactions and handle noisy features with minimal tuning.
- **XGBoost** was used as a gradient boosting algorithm optimized for tabular data, providing improved performance through boosting and robust handling of class imbalance.

All models were initially trained on a common dataset derived from multiple sources, containing financial, contractual, and timing-related features for each SIB project. Additionally, since payments are only made if the social project was successful, to predict the internal rate of return of an SIB, we filtered the data to only include the project that were successful, as defined by the "success" feature in the dataset.

To ensure that our model simulated a realistic pre-investment decision-making process, we strictly limited inputs to features that would be available before an investment was made.

Excluded Features (to avoid leakage): Outcome payments, Investor Repayment Amounts, Actual number of service users, and Duration metrics based on actual project completion.

This strict filtering ensured that the models would not inadvertently "peek" at future outcomes when making predictions. Additionally, of the remaining features, we used an ensemble/consensus feature selection method taking the union of the top 10 most important features across all models ran with all pre-investment features.

During the development process, several challenges were encountered and addressed:

- **Processing Categorical Variables for Regression**: Regression models typically expect numerical input, so categorical variables had to be carefully transformed. For categorical variables, one hot encoding was conducted for linear, lasso and ridge regression while label encoding was conducted for the tree-based regression models.
- **Predictive Power of Features**: We initially ran a Lasso Regression to help select features for our final models. However, only two features had non-zero coefficients and the overall accuracy of the model was still pretty low. Thus, we opted for an ensemble/consensus feature selection method to hopefully improve the accuracy of the model without overfitting.

All models were evaluated using a common holdout test set and assessed with the following metrics: Mean Squared Error, Root Mean Squared Error and R-squared Score (Coefficient of Determination).

Linear Regression:

- Purpose: Served as a foundational baseline model due to its simplicity, speed, and interpretability
- Preprocessing: Since the model is sensitive to feature scaling and magnitude differences, input features were standardized using StandardScaler in a pipeline when appropriate
- Configuration: Implement without regularization, using all selected features as-is
- Tuning: No hyperparameter tuning was required, as this model has no regularization term

Lasso Regression:

- Purpose: Applied to perform feature selection by shrinking less informative coefficients to exactly 0, thus improving model interpretability and reducing overfitting
- Preprocessing: features were standardized using *StandardScaler* within a pipeline to ensure regularization treated all features equally
- Tuning: Used *GridSearchCV* with 5-fold cross=validation to select the optimal value of the regularization parameter *alpha*, minimizing negative mean squared error

Ridge Regression:

- Purpose: Address potential multicollinearity among features to stabilize the model and improve generalization
- Preprocessing: features were standardized using *StandardScaler* within a pipeline to ensure regularization treated all features equally
- Tuning: Used *GridSearchCV* with 5-fold cross=validation to select the optimal value of the regularization parameter *alpha*, minimizing negative mean squared error

Decision Tree Regressor:

- Purpose: Capture nonlinear relationship and allow interpretable rule-based predictions
- Tuning: Used *GridSearchCV* with 5-fold cross-validation to select optimal values of *max_depth* and *min_samples_leaf*, minimizing negative mean squared error.

Bagged Trees:

- Purpose: Reduce overfitting and model variance by aggregating predictions from multiple trees trained on bootstrap samples
- Tuning: Evaluated different values of *n_estimators* and *max_samples* using *GridSearchCV*, with Decision Tree Regressor as the base estimator

Random Forest:

- Purpose: Built on bagging by introducing feature-level randomness, making individual trees less correlated and more robust
- Tuning: Evaluated different values of *n_estimators* and *min_samples_leaf* using *GridSearchCV*

XGBoost:

- Purpose: Leverages gradient boosting to sequentially fit new trees on the residuals of prior trees, reducing both bias and variance beyond what bagging methods achieve
- Tuning: Used GridSearchCV with 5-fold cross-validation to evaluate different *n_estimators*, *learning_rate*, *max_depth*, *subsample*, and *colsample_bytree*, optimizing for negative MSE

Please refer to Appendix F & G for additional information on parameter tuning with GridSearchCV.

Results

I. Predicting the likelihood of SIB success

To assess model performance, we evaluated our three classifiers using key classification metrics: accuracy, precision, recall, F1-score (for the positive class), and the ROC AUC score. These metrics provide a balanced view of how well each model performs in predicting successful outcomes, especially

considering the class imbalance present in the dataset. The table below summarizes the comparative results across these metrics.

Model	Accuracy	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	ROC AUC Score
Logistic Regression	**0.87**	**0.87**	**1.00**	**0.93**	**0.7477**
Random Forest	**0.89**	**0.89**	**1.00**	**0.94**	**0.5682**
XGBoost	**0.73**	**0.88**	**0.80**	**0.84**	**0.5159**

To evaluate the practical effectiveness of our models, we simulated investment decisions based on each classifier's predictions and compared them to three baseline strategies: Naive Classifier, Buy-and-Hold, and a Monte Carlo ROI Threshold heuristic.

The Naive Classifier: The Naive Classifier, which assumes every social impact bond will be successful, serves as a simple yet informative baseline. While it achieved a seemingly high accuracy of 87.30%, this is largely a reflection of the class imbalance in the dataset, where the majority of projects are in fact successful. Its confusion matrix reveals a complete inability to identify unsuccessful projects (0% precision and recall for the negative class), highlighting the importance of a model's ability to differentiate risk, not just maximize accuracy. The Naive Classifier invested indiscriminately, resulting in a relatively high estimated return ($4M) but offering no selective power. In contrast, our best-performing model, Logistic Regression, was nearly as profitable while being significantly more selective—investing in fewer projects and correctly identifying both successes and failures. This demonstrates that machine learning models can offer more informed, targeted decision-making compared to naive or blanket investment approaches.

Monte Carlo (ROI Threshold Strategy): This randomized strategy invested selectively based on a heuristic ROI threshold, achieving an average success rate of 87.21% and a much lower estimated return (~$1.2M). This underscores the importance of learning from project features rather than investing at random.

Logistic Regression: This model outperformed all baselines in both success rate (89.83%) and intelligent allocation, investing in 59 projects and correctly identifying 53 as successful. The estimated return is ~$3.57M. This reflects both high precision and recall in a real-world setting, especially given its higher ROC AUC score (0.7477) from earlier evaluations.

Random Forest: While slightly less selective (46 investments), it correctly classified 41 successful projects, achieving the highest accuracy (89.13%) of the tree-based models. However, it yielded a lower estimated return than Logistic Regression.

XGBoost: This model was the most conservative, investing in only 36 projects but still achieving a solid accuracy of 88.89%. Its conservative nature likely limited returns (approx. $2.23M), suggesting underconfidence in some true positives.

II. Estimating the internal rate of return of SIBs

To assess model performance, we used three standard regression metrics:

- **Mean Squared Error (MSE):** Evaluates the average squared difference between predicted and actual values—sensitive to outliers and penalizes large errors
- **Root Mean Squared Error (RMSE):** The square root of MSE, providing an interpretable error in the original units of the target variable
- **R-squared:** Measures the proportion of variance in the dependent variable explained by the model

Naïve Average Predictor:

- Naïve Baseline MSE: 0.000303
- Naïve Baseline RMSE: 0.017403
- Naïve Baseline R-squared: -0.038309

The results of the models implemented as follows (ordered by Test MSE):

Model	Train MSE	Train RMSE	Test MSE	Test RMSE	Train R-squared	Test R-squared
Bagging	0.000358	0.018918	0.000299	0.017305	0.043143	-0.026616
Random Forest	0.000366	0.019134	0.000302	0.017389	0.021125	-0.036647
Lasso	0.000365	0.019092	0.000308	0.017553	0.025400	-0.056338
Ridge	0.000340	0.018436	0.000309	0.017591	0.091290	-0.060918
XGBoost	0.000324	0.017993	0.000314	0.017708	0.134424	-0.075037
Decision Tree	0.000350	0.018713	0.000336	0.018317	0.063782	-0.150214
Linear	0.000283	0.016810	0.000443	0.021039	0.244540	-0.517514

Based on the evaluation metrics above, only the Bagging and Random Forest models outperformed the naïve predictor in terms of both MSE and RMSE. However, none of the models achieved an R-squared close to -1 or 1, indicating that overall explanatory power is low (see Appendix E).

While Bagging and Random Forest marginally outperformed the naïve baseline (MSE: 0.000303) in terms of prediction error, the consistently negative R-squared values across all models indicate fundamental challenges in predicting IRR.

This points to either insufficient signal in our feature set or complex relationships that our models couldn't capture effectively. Feature importance analysis from our tree-based models revealed that maximum potential return, number of beneficiaries, and project duration had the strongest influence on predictions, though their predictive power remained limited.

Interestingly, despite the sophistication of XGBoost, it did not outperform simpler ensemble methods like Bagging. This suggests that the performance limitation stems from the data characteristics rather than model complexity. The close performance between training and testing errors indicates our models did not overfit but rather struggled with capturing the underlying IRR determinants.

Conclusion

I. Predicting likelihood of success of SIBs

Our goal in this part of our two-fold strategy was to help impact investors make informed, risk-aware decisions by predicting whether a SIB is likely to succeed, which is a critical first step in their trading strategy given that investor repayment is contingent on successful project outcomes.

We approached achieving this goal by testing three classification models and found that Logistic Regression with Lasso regularization performed best and was able to balance interpretability, accuracy, and risk differentiation. Its higher precision meant that it was able to outperform baseline investment

strategies. While Random Forest and XGBoost offered more sophisticated modeling of nonlinear patterns, we did not find that they could outperform the simpler logistic model on our test dataset at least, and overall, these more complex models could not overcome weak signals in the data, highlighting the importance of feature quality.

Throughout this project, it was important for us to develop a pipeline that excluded post-investment information, so we could see our models' performance when subjected to a real-world investment decision, making it relevant to investors. Despite addressing challenges with class imbalance and early-stage data leakage with thorough pre-processing of the data, the results of this project highlighted the value of simpler models in working with limited data under real-world constraints. It also pushed us to consider that while our models achieved high accuracies when applying the investing strategy, there was the possibility that those metrics were misleading due to the nature of our dataset and other indicators like low precision.

Consequently, it could be crucial to develop uncertainty estimates to better represent the confidence investors should have in predictions as well as cost-sensitive learning that would better address false positives, providing fail safes to investors who bear the cost of failure should predictions be inaccurate. While there is currently a limited amount of data on SIBs and only 240 contracted SIBs across the globe, being able to expand our dataset would allow us to improve model generalization and determine true performance for underrepresented failed projects, so next steps might also include advocating for more and better quality data collection in this space.

II. Predicting the Internal Rate of Return of SIBs

Our objective in the second part of this project was to develop robust regression models capable of accurately estimating the internal rate of return (IRR) for Social Impact Bonds (SIBs), aiming to provide investors with valuable insights into potential financial outcomes. We implemented and evaluated a suite of regression models, ranging from linear and regularized models to tree-based ensembles like Bagging, Random Forest, and XGBoost. While Bagging and Random Forest outperformed the naïve baseline in terms of test error, none of the models demonstrated strong explanatory power as measured by R-squared values. This suggests that IRR is influenced by factors beyond the pre-investment variables available in our dataset.

The relatively low predictive performance across models highlights the inherent challenges posed by limited data availability and the subtlety of financial indicators within social impact bonds. This outcome likely reflects the complex, multi-faceted nature of social impact projects, where success and returns depend on numerous factors beyond quantitative metrics—including implementation quality, stakeholder relationships, and external socioeconomic conditions not captured in our dataset.

Despite these challenges, our work may provide valuable insights for impact investors. The marginal improvements offered by ensemble methods like Bagging suggest that combining multiple predictive approaches may yield better results than single models. Additionally, our feature importance analysis identified key variables that, while not strongly predictive in isolation, may warrant closer attention during investment evaluation. For instance, *the anticipated duration of service provision* is consistently the best predictor variable across all model types.

Future work should focus on incorporating qualitative factors and contextual information that may better explain IRR variations. Developing hybrid models that combine statistical prediction with domain expertise could prove more effective. As the SIB market matures and more real-world performance data becomes available, predictive models will likely improve, potentially enabling more informed investment decisions in this socially valuable financial instrument.

III. Overall Trading Strategy

Based on our insights and conclusions found above, we determined the following trading strategy impact investors can implement:

1. Use logistic regression with lasso regularization to predict success of the Social Impact Bond

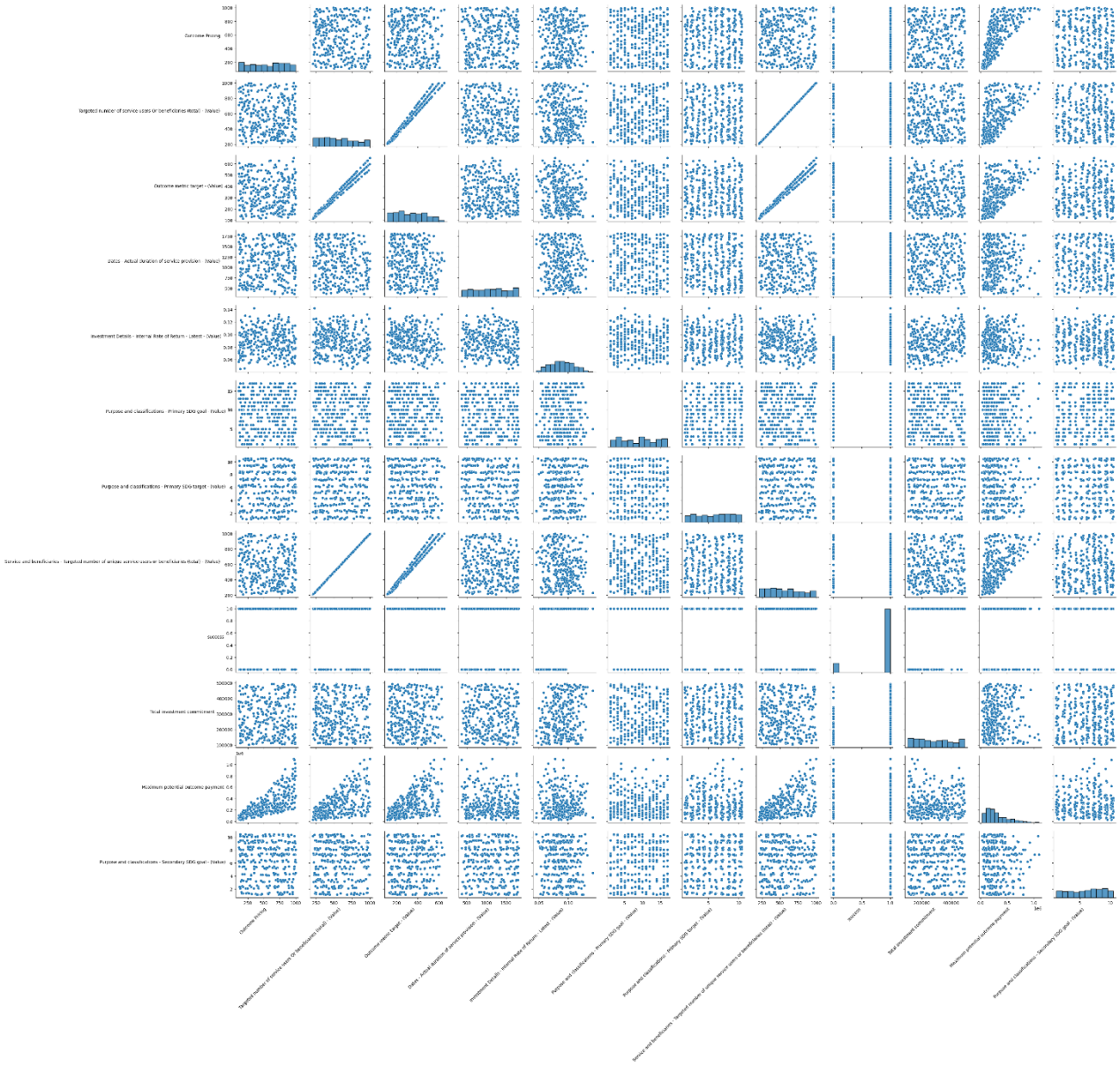
2. Utilize Bagging or Random Forest models to predict IRR. Otherwise, investors should assume the historical average for the IRR of SIB.
3. Based on investors' risk appetite, investors can choose to invest in the Social Impact Bond based on whether the bond would achieve their return goals.

Summary & Potential Next Steps

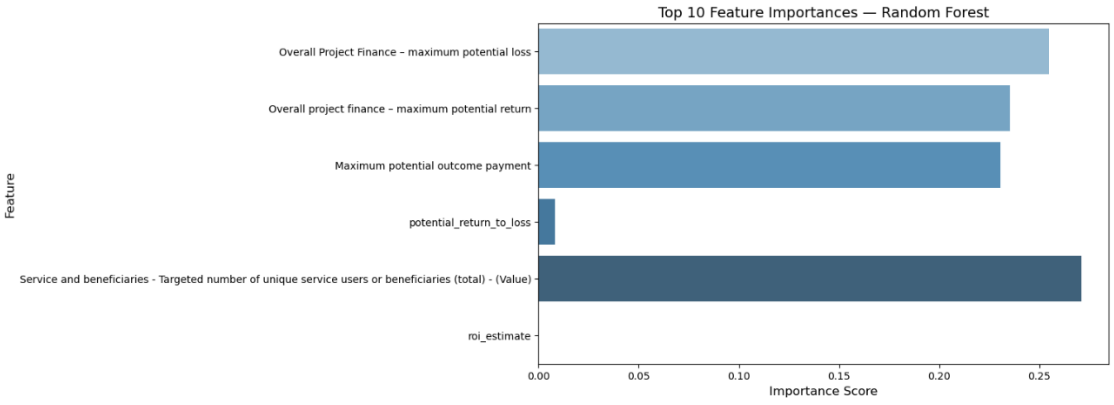
Overall, this project provided hands-on experience with imperfect real-world data, challenged us to address data quality and class imbalance across multiple models, and required us to approach development from an investor's perspective. Based on our learnings and reflections, we identified a few areas to continue exploring: quantifying prediction confidence, enriching the dataset, and bridging our models to real-world use with streamlined tool development. Out of these options, we believe that enriching the dataset with additional feature coverage will provide insights on relationships between features, which we did begin exploring as outlined in Appendix H.

Appendix

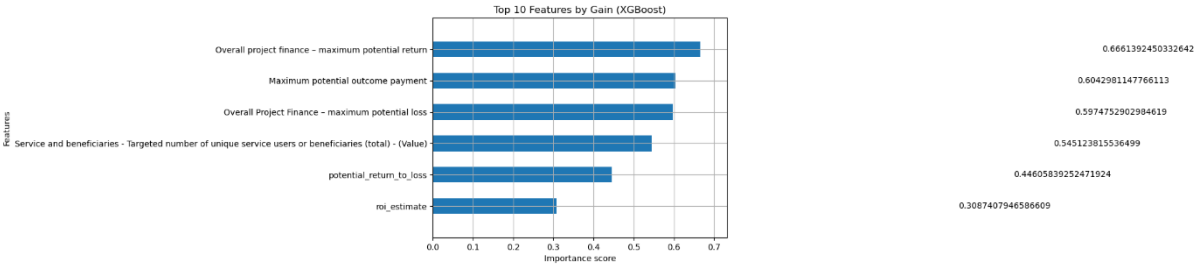
Appendix A: Visualizing Pairwise Relationships



Appendix B: Top Features Random Forest for SIB Success



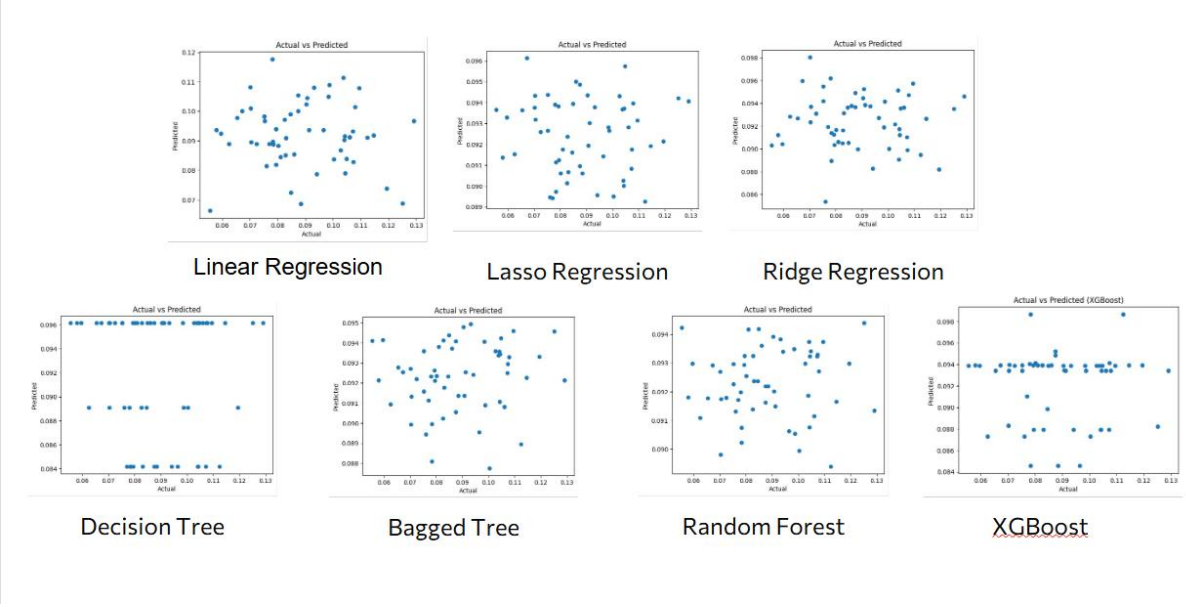
Appendix C: Top Features XGBoost for SIB Success



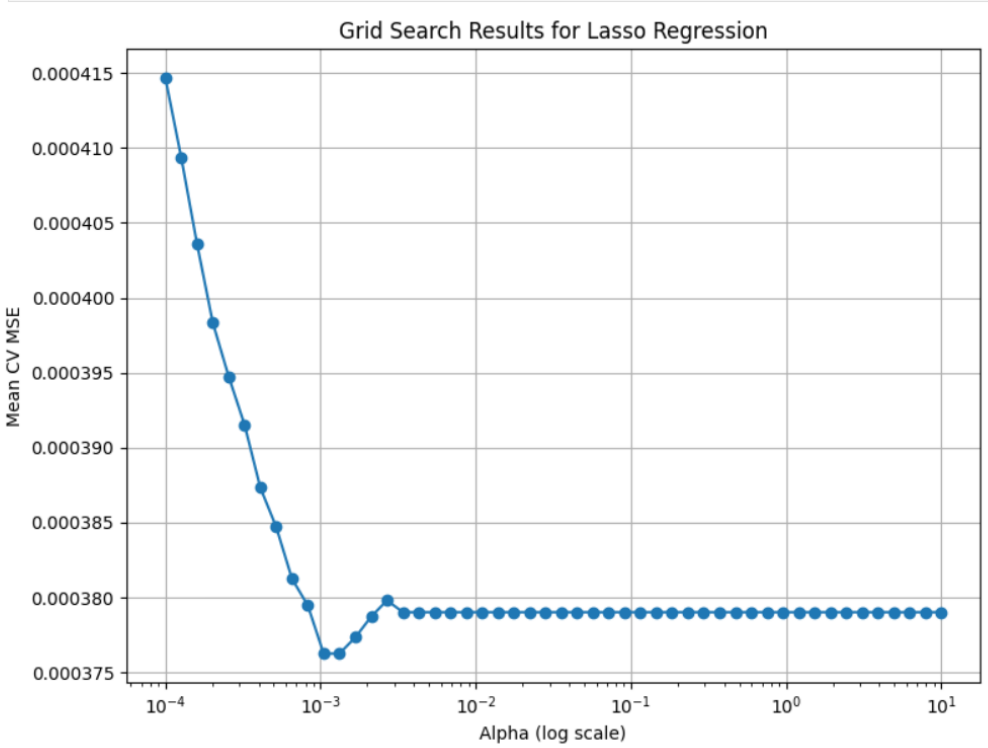
Appendix D: Investment Strategy Comparison for SIB Success

Model	Accuracy	No. of projects invested in	Estimated Return	Notes
Naïve Classifier	87.30%	63	$4M	High return and accuracy reflects the class imbalance in the dataset.
Monte Carlo	87.21%	18 (random)	~ $1.2M	Important to learn from project features rather than investing randomly.
Logistic Regression	89.83%	59	~$3.57M	Correctly identified 53 projects as successful.
Random Forest	89.13%	46	~$2.7M	Correctly identified 41 projects as successful.
XGBoost	88.89%	36	~$2.23M	Conservative nature likely limited returns.

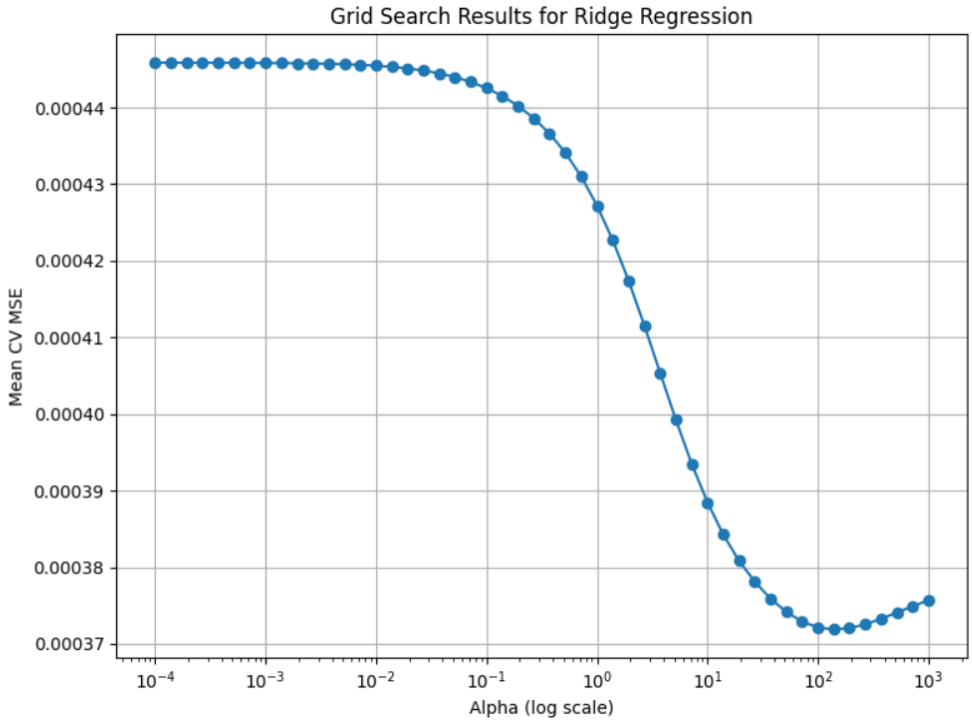
Appendix E: Actual vs. Predicted Plots for IRR Prediction Models



Appendix F: Grid Search CV Elbow Graph for Lasso Regression



Appendix G: Grid Search CV Elbow Graph for Lasso Regression



Appendix H: Enriching Our Dataset with a Calculated Feature

While we were unable to add additional features or data points to our current dataset, we did want to see if we could gather additional insights from looking at the input features effect on a calculated IRR

Achieved Ratio. The binary Success indicator in our dataset was determined by whether the target IRR was achieved, and since the dataset also gave us values for target IRR and actual IRR, which were dropped to prevent data leakage, we were able to calculate a field called irrAchievedRatio (actual IRR / target IRR) to provide more granularity on the performance of a SIB should it be successful. The rationale behind exploring a continuous irrAchievedRatio is to provide investors with greater knowledge on ROI quality beyond just the success or failure of an SIB, which we mainly focused on in our original two-fold strategy. Creating this field creates a normalized score of *how well* a project performed compared to expectations, allowing us to both enrich the dataset with an additional field and potentially derive a greater breadth of predictive knowledge for investors who want to generate the greatest return for minimized risks.

All Linear Regression and Neural Network Models were trained on the same validated and preprocessed data as our classification models to prevent the data leakage and class imbalance we found with our other models.

Each exploratory model was trained using appropriate techniques to balance performance and interpretability:

1. **OLS Linear Regression:** Due to its simplicity, OLS was mainly used as a baseline. We trained it on the preprocessed and selected features from our classification model listed above. No tuning was required as there was no regularization term.
2. **Lasso Regression (L1 Regularization):** Trained using LassoCV with 5-fold cross-validation to automatically select the optimal regularization strength (alpha). The model was incorporated into a pipeline with StandardScaler to ensure consistent feature scaling. L1 regularization helped with feature selection by shrinking less informative coefficients to zero.
3. **Ridge Regression (L2 Regularization):** Trained using RidgeCV with 5-fold cross-validation to select the optimal regularization strength (alpha). A pipeline with StandardScaler was used to standardize input features. L2 regularization helped reduce model variance and mitigate multicollinearity.
4. **Feedforward MLP for Regression:** Implemented using TensorFlow's Keras API as a simple feedforward multi-layer perceptron (MLP). The model architecture consisted of two hidden layers with 128 and 64 ReLU-activated units, respectively, followed by a single linear output neuron for regression. The model was compiled with the Adam optimizer and trained for 30 epochs using a mean squared error (MSE) loss and mean absolute error (MAE) as a secondary metric. Feature scaling was handled prior to input.
5. **MLP with Dropout Layer:** The model featured two hidden layers with 128 and 64 ReLU-activated units, each followed by a Dropout layer with a rate of 0.2 to prevent overfitting by randomly disabling a portion of neurons during training. The final output layer used a linear activation to produce continuous predictions for regression. The model was compiled with the Adam optimizer and trained for 30 epochs using MSE loss and MAE as a secondary metric. Prior feature scaling ensured effective training convergence. Dropout helped improve robustness by regularizing the model.

Linear Regression models were evaluated with the same testing set as the classification models and assessed by **MSE, RMSE, and R-squared.** We also looked at Strategy Performance and prediction accuracy based on a predicted IRR Ratio ≥ 1 (cutoff for success).

On the other hand, MLP was also evaluated with the same testing set and assessed by **MSE, MAE, and R-squared**.

Linear Regression

	Train MSE	**Train RMSE**	**Test MSE**	**Test RMSE**	**Train R-Squared**	**Test R-Squared**
Linear	0.028031	0.167425	0.028699	0.169407	0.136786	0.054612

Lasso	0.028411	0.168556	0.027655	0.166298	0.125084	0.088992
Ridge	0.028399	0.168518	0.027667	0.166334	0.125467	0.088599

Looking at the MSE and RMSE, these metrics indicate that there is limited overfitting as all three models perform nearly identically. However, the R-squared for all models is significantly low, which tells us that the Linear model can only account for ~9% of the variance in the target variable. The features available do not seem to have a strong enough signal to act as a driver for IRR, and a Linear approach does not seem appropriate to use in this case.

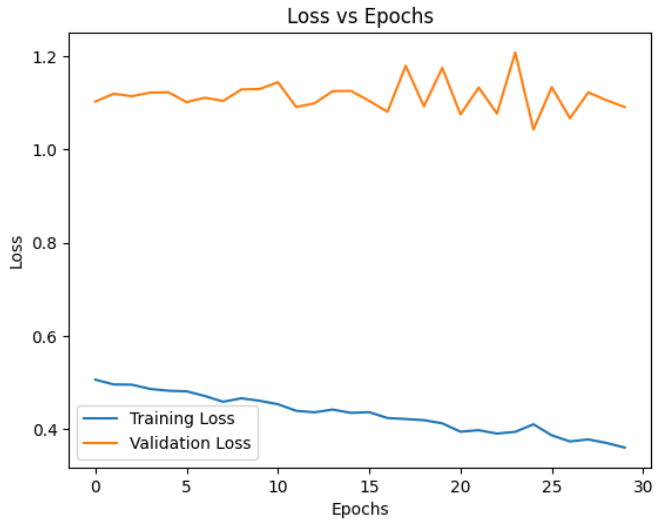
Interestingly, if we run the basic trading strategy using the Linear model, we do still get high accuracies with all Linear models investing in 63 projects with 54 of them being successful, resulting in an 85.71% accuracy. However, this accuracy likely derives from the high class imbalance of the dataset, with a majority of projects being successful.

Feedforward MLP

Since the Linear Models suggested either a weak signal or nonlinear relationship between the input features and target variable, we expanded to a MLP model, which has greater ability to learn nonlinear mappings between the inputs and the target.

	Feedforward MLP	***MLP with Dropout Layer***
Test MSE	0.8815	0.8997
Test MAE	0.7549	0.7690
R-Squared	0.0571	0.0375

Despite the MLP's increased capacity to model complex nonlinear relationships, neither architecture significantly outperformed linear models. In fact, Linear Regression slightly outperformed the two MLP models. The above metrics also show that adding a Dropout layer resulted in slightly worse performance across all metrics.



If we look at the training process, we can see that while training loss decreases overall, indicating that the model is learning from the data, the validation loss stays relatively constant with some fluctuations. This behavior tells us that the model may be underfitting the data, which explains why the Dropout layer harmed performance as it addresses overfitting. Furthermore, because the dataset consists of only 314 datapoints, it does make sense that the network struggles to determine meaningful patterns.

Exploratory Model Summary & Takeaways

While both Linear Regression and MLP had issues with significantly low R-squared values, they still achieved relatively low MSE values, which means that despite not explaining the variance well, their predications remained reasonably close to the actual outcomes.

To evaluate real-world implementation of these models, we tested a strategy where investors invested in the eight SIBs with the highest predicted IRR Ratios. This approach did lead to a notable improvement in the average return on investment per project. Investing in all projects based on a threshold for the Linear Model resulted in an average return of ~$64,000 whereas investing in the top SIBs resulted in an average return of ~$100,000. However, investing across the full test set understandably resulted in a higher overall return because investors invested in significantly more projects.

Several limitations to note with these results are the class imbalance of the dataset, which skews the models' learning process towards higher accuracy or favorable return results. Furthermore, the improved average return from the top predicted investments is not truly a reflection of model precision but more so represent the added benefit of having granular information through predicting IRR Ratio compared to the binary Success feature. Due to small dataset, underfitting, or unclear data patterns, the models might not be able to adequately learn complex, nonlinear relationships robustly, so it is very likely that the overlap between the model's ranking and the higher-performing projects is coincidence. Ultimately, narrow top-N selection can give the illusion of strong performance even if the overall predictive capability is weak.

However, by engineering and incorporating the IRR Achieved Ratio, we were able to enrich the dataset with a continuous and interpretable measure of investment performance that goes beyond the original binary success label. It is also significant to note that currently, while it would be ideal to collect more data points or new features that were not included in our dataset, there is a high barrier to incorporating this information due to the nature of SIBs, which are relatively new, often small in scale, and not consistently tracked through standardized data collection or long-term research. By creating a new feature from our current data, we were still able to gather additional insights on the degree to which a project met or missed its return target, enabling a more nuanced analysis of project outcomes. From an investor's perspective, this opens the door to risk-adjusted evaluation strategies, performance benchmarking, and prioritization of investments not just by likelihood of success, but by potential return magnitude.