COSC 522 Final Project
Loan Default Prediction

Group 3

Andrew Penny, Cody Viscardis, Peter Mansfield, Soe Thet Ko

Abstract—**In the financial sector, accurately predicting loan defaults is crucial for loan providers to make informed decisions and minimize risks. The goal of this project is to build a machine learning model that can predict potential loan defaults, based on customer demographics of loan applicants, with over 90% accuracy. In this project, we trained four different models utilizing Logistic Regression, Neural Network, Gradient Boosting and Random Forest architectures. The performance of each model was evaluated on the holdout set using the following metrics: Accuracy, Precision, Recall, F1, Confusion Matrix and ROC AUC. In addition, we analyzed features correlation as well as feature importance and discussed the advantages and limitations of each model.**

I. INTRODUCTION

Inaccurate risk assessment on loan applications not only leads to financial losses for lending institutions but also hinders financial inclusion for qualified borrowers. Traditional loan assessment methods often rely heavily on credit scores and financial history such as income and debt-to-income ratio, overlooking other customer demographics that may influence loan repayment behavior. This limited scope can lead to inaccurate evaluations for loan applicants with limited credit history who fall outside the traditional credit scoring models.

To address these limitations, we are motivated to explore the potential of binary classification machine learning models that will assist in identifying potential defaulters by learning the hidden patterns and structures in customer demographics of loan applicants and mapping to the corresponding target class with high accuracy. With the ability to handle complex data and intricate relationships between the feature variables and the target, these machine learning models can potentially create more nuanced and accurate assessment of loan applications by continuously learning and improving their predictive capabilities over time.

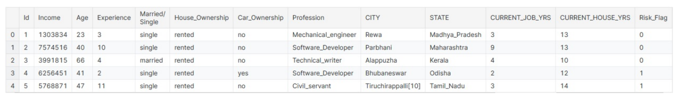
We are eager to research and exploit the capabilities of machine learning in loan default prediction since a robust and accurate risk assessment solution will bring tremendous value to the financial sector. The successful development and deployment of a highly accurate model will not only help mitigate risks for financial institutions but also help increase loan access for qualified borrowers.

II. RELATED WORK

Credit risk assessment is a crucial part of the finance industry. Previously credit risk assessment has utilized credit score models like FICO, but with the rapid development of machine learning, research has been performed to determine whether machine learning can predict if an individual may or may not be a credit risk. Researchers have utilized various algorithms such as support vector machines, logistic regression, neural networks, decision trees, and random forests. Random forests have typically outperformed all other models utilized to determine credit risk which can be seen in papers like "A Comparative Assessment of Credit Risk Model Based on Machine Learning" where they used KNN, decision trees, random forests, naive bayesian, and logistic regression. Our goal is to determine if random forests will reign supreme in credit risk assessment or will other algorithms like gradient boosting, logistic regression, or neural networks be able to more accurately classify credit risks.

III. DATA

To train our models with a supervised learning approach, we used the "Loan Prediction Based on Customer Behavior" dataset from Kaggle. The dataset comprises 252,000 borrowers, with 11 features (excluding 'Id') and a target variable. The features include 'Income', 'Age', 'Experience', 'Married/Single', 'House Ownership', 'Car Ownership', 'Profession', 'City', 'State', 'Current Job Yrs, and 'Current House Yrs'. The target 'Risk Flag' indicates whether the customer defaulted or not.



	Id	Income	Age	Experience	Married/ Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh	3	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	9	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	4	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	2	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu	3	14	1

Figure 1: Dataset Snippet

As we performed Exploratory Data Analysis on the dataset, we found it to be clean with no NULL values but the distribution is heavily skewed towards the negative target class (no loan default).

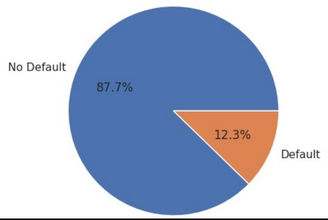
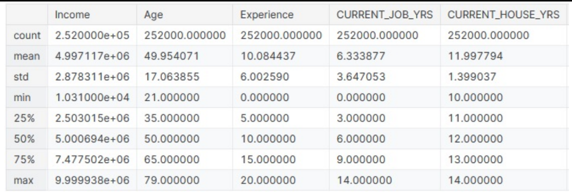


Figure 2: Loan Default Proportions

A statistical description of the numerical features indicates the dataset is relatively free of outliers.



	Income	Age	Experience	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS
count	2.520000e+05	252000.000000	252000.000000	252000.000000	252000.000000
mean	4.997117e+06	49.954071	10.084437	6.333877	11.997794
std	2.878311e+06	17.063855	6.002590	3.647053	1.399037
min	1.031000e+04	21.000000	0.000000	0.000000	10.000000
25%	2.503015e+06	35.000000	5.000000	3.000000	11.000000
50%	5.000694e+06	50.000000	10.000000	6.000000	12.000000
75%	7.477502e+06	65.000000	15.000000	9.000000	13.000000
max	9.999938e+06	79.000000	20.000000	14.000000	14.000000

Figure 3: Statistics of Numerical Features

We used Kernel Density Estimate plots to visualize the distribution of the target class for each numerical feature.

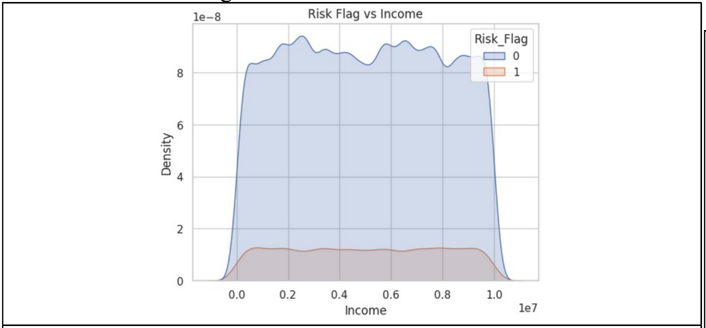


Figure 4: Target Distribution for Income

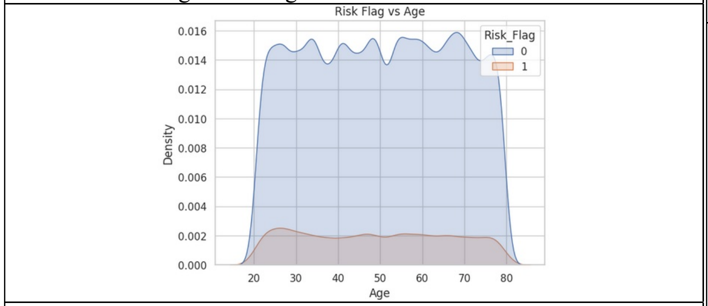


Figure 5: Target Distribution for Age

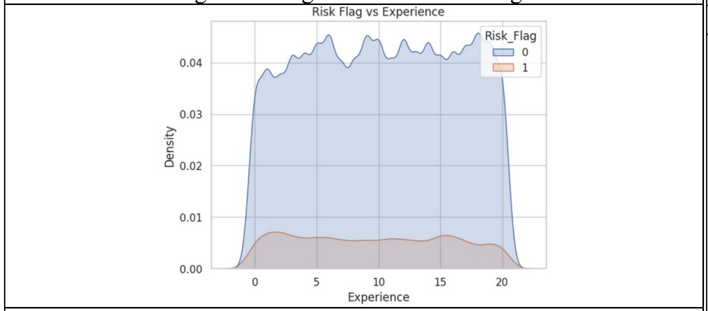


Figure 6: Target Distribution for Experience

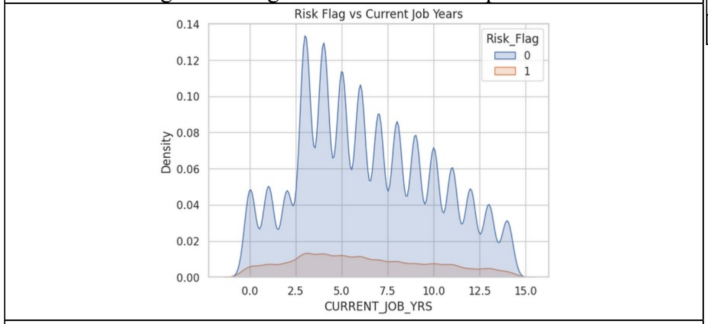


Figure 7: Target Distribution for Current Job Years

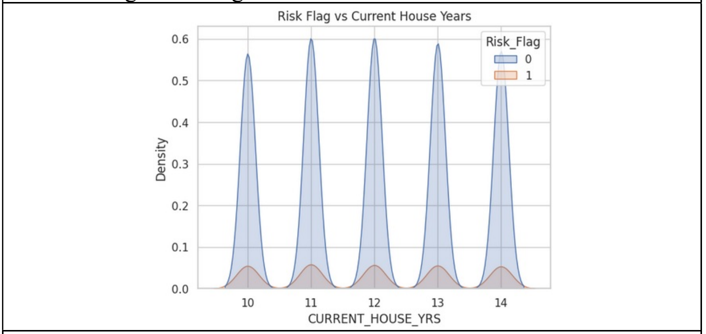


Figure 8: Target Distribution for Current House Years

For the categorical features with low cardinality, we used bar plots to visualize the ratio of loan defaults for each category of each feature.

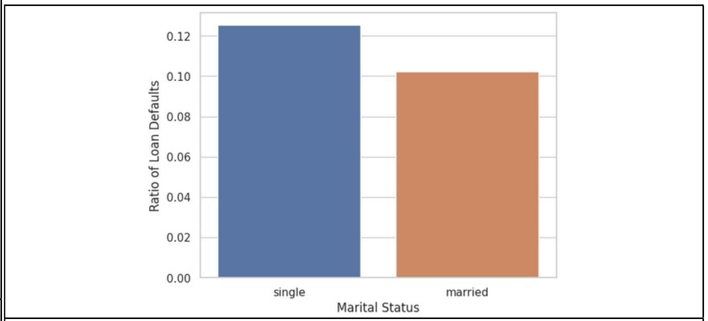


Figure 9: Loan Default Ratio for Marital Status

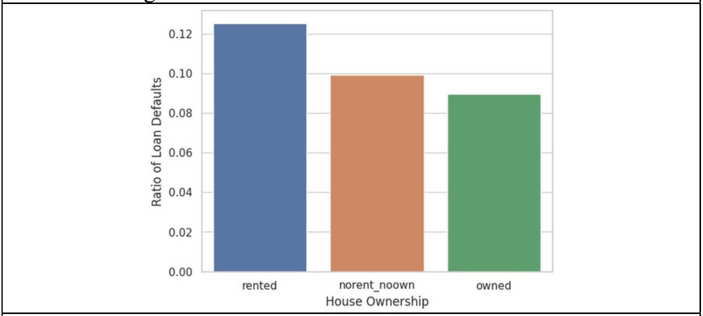


Figure 10: Loan Default Ratio for House Ownership

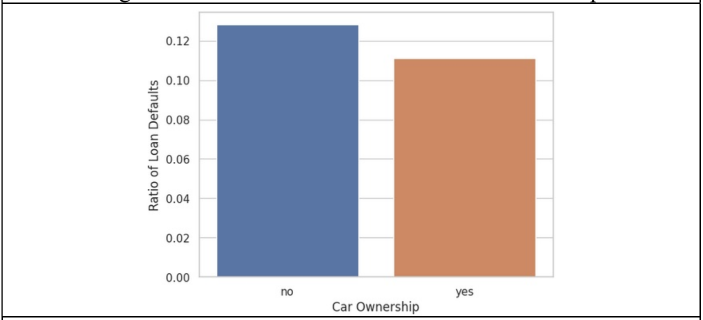


Figure 11: Loan Default Ratio for Car Ownership

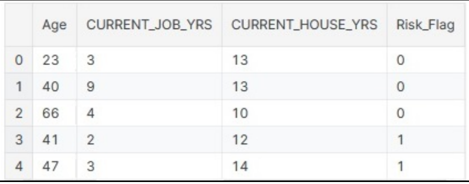
The ‘Profession’, ‘City’ and ‘State’ are also categorical features but have high cardinality. So, we just used the “groupby” option for each feature and averaged ‘Risk_Flag’ to get the ratio of loan defaults for each category of the feature. The ‘Profession’ variable has 51 categories with the ratio of loan defaults ranging from 0.16 to 0.08. The ‘City’ variable has 317 categories with the ratio of loan defaults ranging from 0.33 to 0.03. The ‘State’ variable has 29 categories with the ratio of loan defaults ranging from 0.22 to 0.05.

IV. METHODS

A. *Logistic Regression*

- Data Transformation

Upon initial review of the data within the context of the logistic regression model, a few checks and manipulations were performed. First, a basic analysis of the features was conducted including rechecking for null values and shape of the data. 3 categorical variables - “married/Single”, “House Ownership”, and “Car Ownership” - were transformed to binary classifications. Upon realization it was a clean but heavily skewed dataset, several features were strategically removed throughout as well as substantial rebalancing of the input data sets via over and under sampling. This was done to varying degrees during different iterations of the model application to assess accuracy changes. The final input was composed of three primary features with the target variable being "risk_flag" (the binary variable for loan default) and an under-sampling of the majority class by a factor 4.



	Age	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
0	23	3	13	0
1	40	9	13	0
2	66	4	10	0
3	41	2	12	1
4	47	3	14	1

Figure 12: Final Chosen Features and the Target

- Model Architecture

This analysis was based on the LogisticRegression model from sklearn library. We applied standard application techniques under the following manipulations: the full dataset; transformed with strategic removal of features; oversampling of the minority class by 2-6 times; undersampling of the majority class by factors of 2-6.

- Evaluation Plan

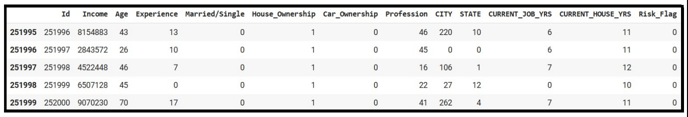
The model was evaluated with the primary metric of accuracy and secondary metrics of precision, recall, F1 score, and a confusion matrix for distribution analysis of result.

B. Neural Network

- Data Transformation

For one of our methods, we utilized a neural network to analyze the loan risk assessment. The label for this model was a risk flag with binary values and its features were: income, age, experience, marital status, house ownership, car ownership, profession, city, state, how long they have worked at their job, and how long they have been in their home.

To transform the data, we utilized label encoding where each categorical value corresponds to an integer value. Marital status and car ownership were mapped to the binary values of 0 and 1. House ownership became 0 for not renting or owning a home, 1 for renting, and 2 for owning their home. Profession values were mapped to an integer range of 0 to 50, city was mapped to an range from 0 to 316, and state was mapped to a range from 0 to 29. For the data, we utilized a 80/10/10 train/test/validation split and used z-score normalization to scale values to a common range.

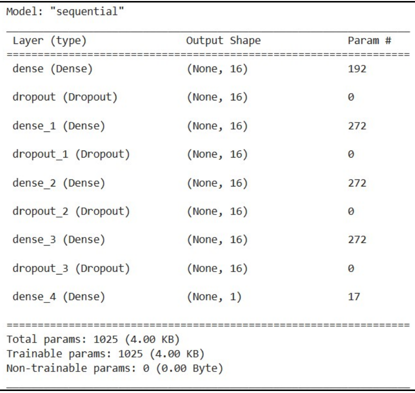


	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
251995	251996	8154883	43	13	0	1	0	46	220	10	6	11	0
251996	251997	2843572	26	10	0	1	0	45	0	0	6	11	0
251997	251998	4522448	46	7	0	1	0	16	106	1	7	12	0
251998	251999	6507128	45	0	0	1	0	22	27	12	0	10	0
251999	252000	9070230	70	17	0	1	0	41	262	4	7	11	0

Figure 13: Transformed Data

- Model Architecture

For our model we utilized the Adam optimizer with a learning rate of 0.005. Our loss function was binary cross-entropy and our performance metric was accuracy. For our model architecture we used 4 16 unit dense layers with each followed by a dropout layer with a dropout rate of 0.2. A 1 unit dense layer followed the other dense and dropout layers. The 16 unit layers had an activation function of a rectified linear unit and the final dense layer had a sigmoid activation function. The batch size was 512 and the model was trained for 100 epochs but contained early stopping when changes in validation loss were miniscule.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                192

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 16)                272

 dropout_1 (Dropout)         (None, 16)                0

 dense_2 (Dense)             (None, 16)                272

 dropout_2 (Dropout)         (None, 16)                0

 dense_3 (Dense)             (None, 16)                272

 dropout_3 (Dropout)         (None, 16)                0

 dense_4 (Dense)             (None, 1)                 17

=================================================================
Total params: 1025 (4.00 KB)
Trainable params: 1025 (4.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 14: Neural Network Architecture

- Evaluation Plan

The evaluation was to fit the model with an early stopping callback to stop training after 5 epochs of miniscule changes of validation loss. Using the built-in functions from Keras to evaluate our model we would be able to see its performance metrics and evaluate whether the model was successful or whether it was overfitting on the training data. High validation accuracy with low validation loss was our goal to reach so that we would have a successful model.

C. Gradient Boosting

- Data Transformation

Upon inspection of the data, it was found to be cleaned and ready for analysis, i.e. no null values or junk data. The target variable was identified as 'Risk_Flag' in which case the positive class would indicate a historical default of that particular record. In order to train a model, the data would need to be transformed into numeric representations. Several features, such as 'Profession,' 'City,' and 'State' included a large set of possible nominal categorical variables, so these were encoded using Target Encoding. The large scale of numerical values led to skewed classifications in the target variable, so a scaling method was employed on the data set, specifically using Standard Scaler. After transformation, the data was undersampled and oversampled, and it was discovered that over sampling using SMOTE was more effective, according to some initial explorations using a basic Logistic Regression model.

- Model Architecture

Initially the Gradient Boosting model was used and performed slightly better than average. So, GridSearch was employed to iterate over possible combinations of hyper-parameters. The GridSearch resulted in a 'best' model with parameters of: learning rate - 0.2, max-depth - 7, min_samples_split -4, n_estimators - 150.

- Evaluation Plan

The model was to be evaluated using metrics such as Accuracy, Precision, Recall, F1 Score, and a confusion matrix for visualization.

D. Random Forest

- Data Transformation

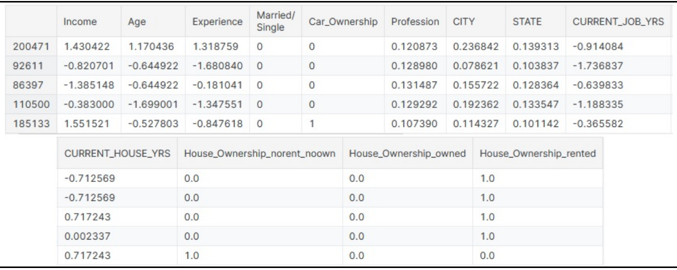
Before doing any encoding or normalization of features, the dataset was separated into train and test sets with 80-20 split.

Since the numerical features, 'Income', 'Age', 'Experience', 'Current_Job_Yrs', and 'Current_House_Yrs', have different value ranges, they were normalized using "StandardScalar". We used the “fit_transform” method on the train set and the “transform” method on the test set.

The ‘Married/Single’ feature was binary encoded by mapping ‘married’ to 1 and ‘single’ to 0. The same was done for the ‘Car_Ownership’ feature, mapping ‘yes’ to 1 and ‘no’ to 0.

The ‘House_Ownership’ feature, which has three possible values: ‘rented’, ‘owned’, ‘norent_noown’, was one-hot encoded using “OneHotEncoder”. We used the “fit_transform” method on the train set and the “transform” method on the test set.

The other categorical features, ‘Profession’, ‘City’ and ‘State’ have high cardinality. So, they were encoded using “TargetEncoder”, which basically gives the mean value of ‘Risk_Flag’ target for each possible category of each feature. We used the “fit_transform” method on the train set and the “transform” method on the test set.



	Income	Age	Experience	Married/ Single	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS
200471	1.430422	1.170436	1.318759	0	0	0.120873	0.236842	0.139313	-0.914084
92611	-0.820701	-0.644922	-1.680840	0	0	0.128980	0.078621	0.103837	-1.736837
86397	-1.385148	-0.644922	-0.181041	0	0	0.131487	0.155722	0.128364	-0.639833
110500	-0.383000	-1.699001	-1.347551	0	0	0.129292	0.192362	0.133547	-1.188335
185133	1.551521	-0.527803	-0.847618	0	1	0.107390	0.114327	0.101142	-0.365582

CURRENT_HOUSE_YRS	House_Ownership_norent_noown	House_Ownership_owned	House_Ownership_rented
-0.712569	0.0	0.0	1.0
-0.712569	0.0	0.0	1.0
0.717243	0.0	0.0	1.0
0.002337	0.0	0.0	1.0
0.717243	1.0	0.0	0.0

Figure 15: Transformed Data

- Model Architecture

The “RandomForestClassifier” from “sklearn” performs really well straight out of the box. Before we finalize our model, we tried optimization with “GridSearchCV’ on various value ranges for these hyperparameters: 'max_depth', 'n_estimators', 'min_samples_split', 'min_samples_leaf' and 'max_features'. However, we got no significant performance improvement. Throughout all iterations of our model training, we just used the default architecture except for the “random_state” parameter.

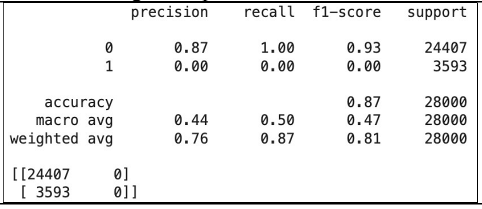
- Evaluation Plan

We evaluated the performance of different iterations of our model on the test set with the “classification_report” metric which includes accuracy, precision, recall and f1-score. Also, we used “confusion_matrix” for quick interpretation and “roc_auc_socre” to rigorously validate the performance.

V. RESULTS

A. Logistic Regression

Ultimately, the logistic regression model proved oversensitive to the negative class (majority) and would underpredict positive cases to the point of inviability. The model was a nonviable solution for this dataset as its accuracy values over the various iterations converged to the distribution of positive to negative cases - approximately 12.5% to 87.5% respectively - regardless of feature removal and rebalancing attempts.



```
              precision    recall  f1-score   support

           0       0.87      1.00      0.93     24407
           1       0.00      0.00      0.00      3593

    accuracy                           0.87     28000
   macro avg       0.44      0.50      0.47     28000
weighted avg       0.76      0.87      0.81     28000

[[24407     0]
 [ 3593     0]]
```

Figure 16: Evaluation Metrics for Logistic Regression

We believe that this was due primarily to the severely imbalanced nature of the input dataset as well as the multiple continuous and categorical feature types within the dataset as logistic regression tends to perform best when features are naturally binary values. Per the ROC curve below, results were no better than random organization.

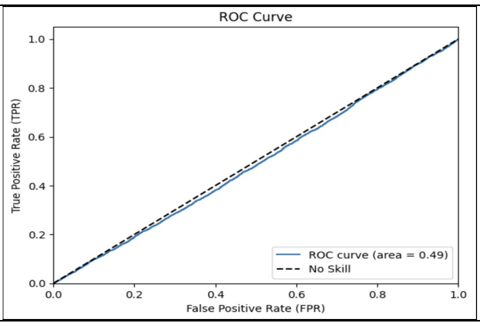
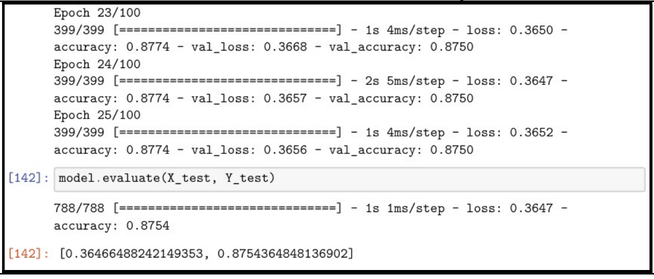


Figure 17: ROC Curve for Logistic Regression

B. Neural Network

For the results of our neural network, the model stopped fitting at the epoch 25, due to repeated epochs only containing minor differences in validation loss. The final epoch resulted in a loss value of 36.43%, 87.74% accuracy, 36.57% validation accuracy, and 87.5% validation accuracy. Upon evaluating the model against the test set, it resulted in loss of 36.36% and accuracy of 87.54%.



```
      Epoch 23/100
      399/399 [==============================] - 1s 4ms/step - loss: 0.3650 -
      accuracy: 0.8774 - val_loss: 0.3668 - val_accuracy: 0.8750
      Epoch 24/100
      399/399 [==============================] - 2s 5ms/step - loss: 0.3647 -
      accuracy: 0.8774 - val_loss: 0.3657 - val_accuracy: 0.8750
      Epoch 25/100
      399/399 [==============================] - 1s 4ms/step - loss: 0.3652 -
      accuracy: 0.8774 - val_loss: 0.3656 - val_accuracy: 0.8750

[142]: model.evaluate(X_test, Y_test)

      788/788 [==============================] - 1s 1ms/step - loss: 0.3647 -
      accuracy: 0.8754

[142]: [0.36466488242149353, 0.8754364848136902]
```

Figure 18: Neural Network Training and Evaluation

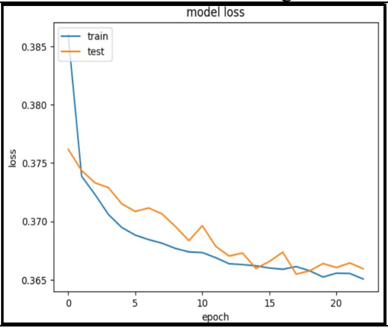


Figure 19: Loss Curves for Neural Network

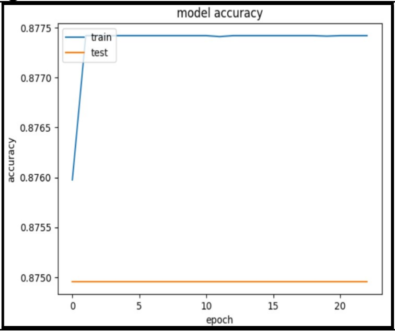


Figure 20: Accuracy Curves for Neural Network

C. *Gradient Boosting*

The initial model achieved an accuracy of 63%, leaving room for improvement.

Accuracy: 0.6291636422292121
Precision: 0.6316717422663889
Recall: 0.6314693158147733
F1 Score: 0.6315705128205128
Confusion Matrix:
[[3860 2298]
 [2300 3941]]

The final model performed well with the 'best parameters' showing an accuracy of 85%. Precision and recall were 86% and 83% respectively, showing a decent accuracy among predicting the positive class.

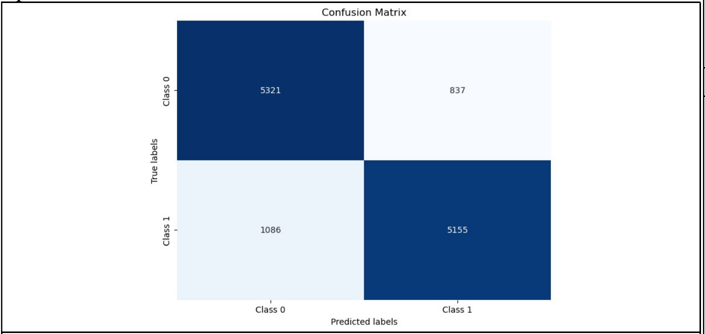


Figure 21: Confusion Matrix for Gradient Boosting

D. *Random Forest*

- 1st Model

In the initial model, we trained our Random Forest Classifier using all features, with no rebalancing technique applied on the dataset. Since data distribution is highly skewed towards the negative target class, the model had very bad performance for the positive target class. The distribution of train and test sets and the evaluation metrics of the model can be seen in the figures below.

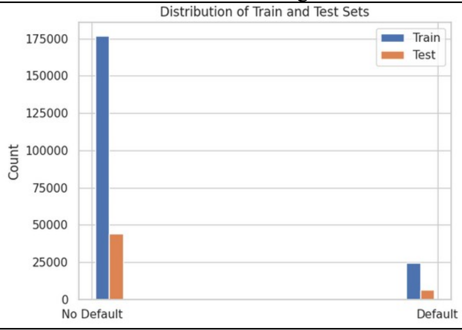
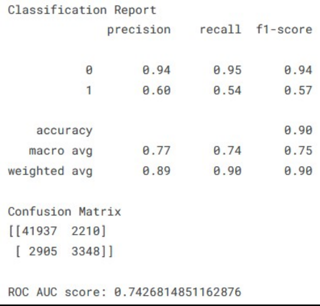


Figure 22: Target Class Distribution for Train and Test Sets



```
Classification Report
              precision    recall  f1-score

           0       0.94      0.95      0.94
           1       0.60      0.54      0.57

    accuracy                           0.90
   macro avg       0.77      0.74      0.75
weighted avg       0.89      0.90      0.90

Confusion Matrix
[[41937  2210]
 [ 2905  3348]]

ROC AUC score: 0.7426814851162876
```

Figure 23: Evaluation Metrics for Random Forest

- 2nd Model

For this stage, we tried undersampling the majority target class to rebalance the dataset, using "RandomUnderSampler". We observed significant improvement in the model's overall performance. The distribution of train and test sets and the evaluation metrics of the model can be seen in the figures below.

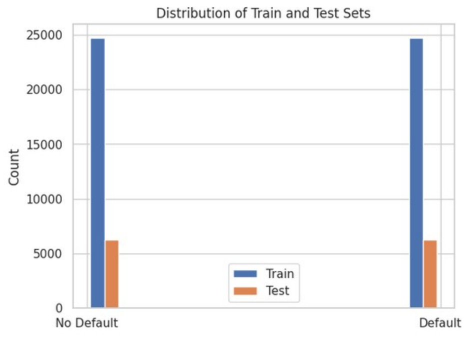
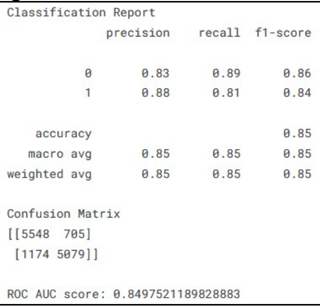


Figure 24: Target Class Distribution for Train and Test Sets



```
Classification Report
              precision    recall  f1-score

           0       0.83      0.89      0.86
           1       0.88      0.81      0.84

    accuracy                           0.85
   macro avg       0.85      0.85      0.85
weighted avg       0.85      0.85      0.85

Confusion Matrix
[[5548  705]
 [1174 5079]]

ROC AUC score: 0.8497521189828883
```

Figure 25: Evaluation Metrics for Random Forest

- 3rd Model

In the third iteration, we tried oversampling the minority target class to rebalance the dataset, using "SMOTE". The model performed even better than the previous case, where we had reduced data samples with undersampling. The distribution of train and test sets and the evaluation metrics of the model can be seen in the figures below.

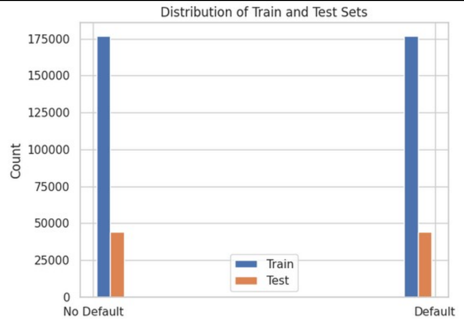
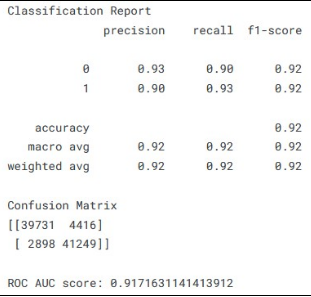


Figure 26: Target Class Distribution for Train and Test Sets



```
Classification Report
              precision    recall  f1-score

           0       0.93      0.90      0.92
           1       0.90      0.93      0.92

    accuracy                           0.92
   macro avg       0.92      0.92      0.92
weighted avg       0.92      0.92      0.92

Confusion Matrix
[[39731  4416]
 [ 2898 41249]]

ROC AUC score: 0.9171631141413912
```

Figure 27: Evaluation Metrics for Random Forest

- 4th Model

For this stage, we still applied the oversampling technique. We analyzed features correlation and found 'Experience' and 'Current_Job_Yrs' to be highly correlated while 'State' and 'City' are somewhat correlated. So, we dropped 'Current_Job_Yrs' and ‘City’ features. As expected, the performance stayed about the same as the previous stage. The heatmap of features correlation and the evaluation metrics of the model can be seen in the figures below.

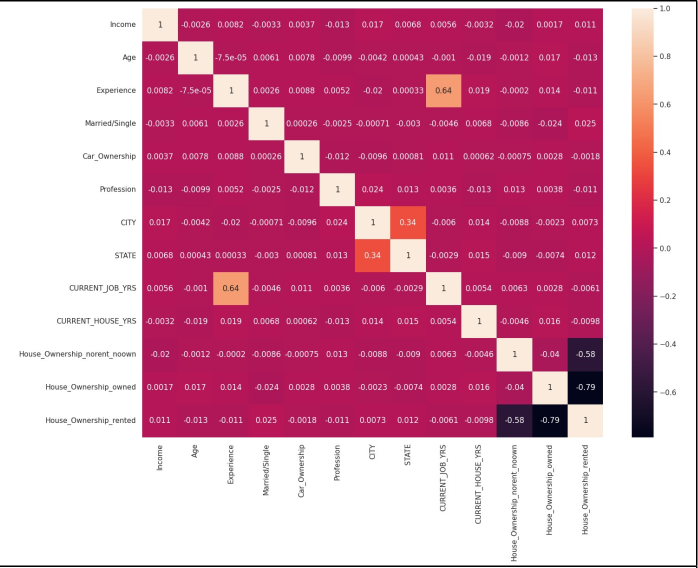
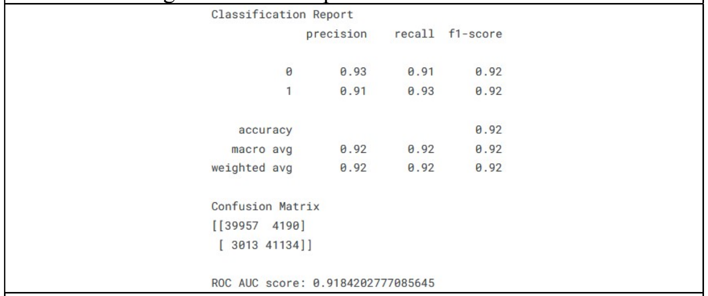


Figure 28: Heatmap of Features Correlation



```
Classification Report
              precision    recall  f1-score

           0       0.93      0.91      0.92
           1       0.91      0.93      0.92

    accuracy                           0.92
   macro avg       0.92      0.92      0.92
weighted avg       0.92      0.92      0.92

Confusion Matrix
[[39957  4190]
 [ 3013 41134]]

ROC AUC score: 0.9184202777085645
```

Figure 29: Evaluation Metrics for Random Forest

- Final Model

In the final run, we did feature importance analysis on the remaining features. It revealed that 'Car_Ownership', 'Married/Single' and 'House_Ownership' had relatively low importance. So, we dropped those features and the model’s performance even improved a bit. The heatmap of features importance analysis and the evaluation metrics and plots of the model can be seen in the figures below.

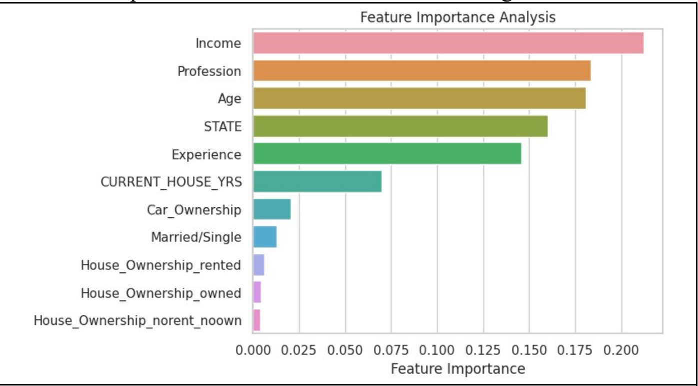
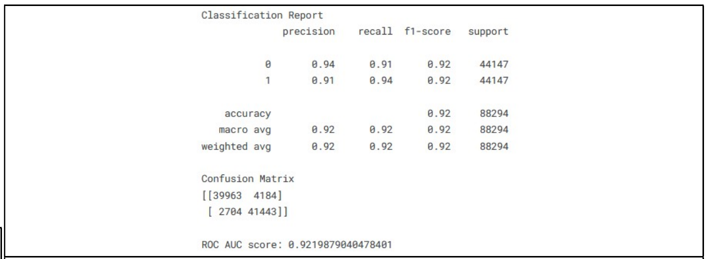


Figure 30: Analysis of Features Importance



```
Classification Report
              precision    recall  f1-score   support

           0       0.94      0.91      0.92     44147
           1       0.91      0.94      0.92     44147

    accuracy                           0.92     88294
   macro avg       0.92      0.92      0.92     88294
weighted avg       0.92      0.92      0.92     88294

Confusion Matrix
[[39963  4184]
 [ 2704 41443]]

ROC AUC score: 0.9219879040478401
```

Figure 31: Evaluation Metrics for Random Forest

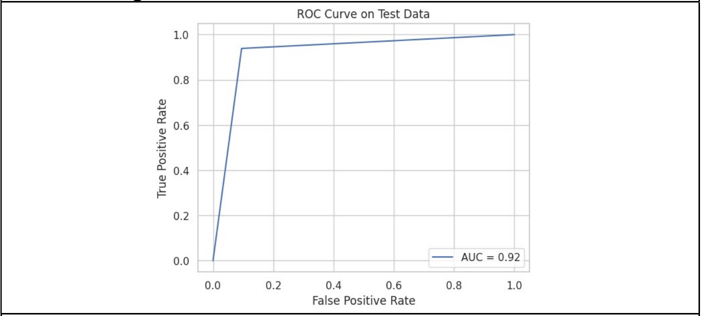


Figure 32: ROC Curve for Random Forest

VI. DISCUSSION

Ultimately, we were able to achieve our prediction accuracy goal of 90% or better. However, if this were a real-world scenario project, we would have liked to do some more feature engineering like creating new features from aggregation, etc. This would be possible by performing more research and gaining domain knowledge around the finance loan sector. Another approach would be to augment the data using more data sources. Lastly, when dealing with predictions that center around individuals and their situations, some sort of interpretation of that data would be ideal in terms of ensuring fairness, ethics and reducing bias towards certain groups of people.

VII. CONCLUSION

In conclusion, the random forest model was our most effective solution. It reached over the 90% benchmark, and we believe it to be an effective method for this actuarial analysis. Our other models were not suited to this dataset as they could not achieve our accuracy requirements. This effort revealed the necessity of rebalancing the input data distribution as this dataset was significantly skewed toward the negative class. Additionally, critical feature analysis was essential as results were improved after irrelevant or noisy features were removed from the analysis. Going forward, we would like to apply this method to a larger dataset with additional features that could potentially increase the accuracy when analyzed in unison as well as training the method on a variety of credit risk types to include but not limited to automotive, revolving lines, SBA, or similar.

REFERENCES

[1] "Loan Prediction Based on Customer Behavior," kaggle.com. https://www.kaggle.com/datasets/subhamjain/loan-prediction-based-on-customer-behavior

[2] Wang, Y., Zhang, Y., Lu, Y., & Yu, X. (2020). A Comparative Assessment of Credit Risk Model Based on Machine Learning——a case study of bank loan data. Procedia Computer Science, 174, 141-149.