CS361: Credit Risk Assessment

Bhogi Sai Sathwik - 210101031 Bussa Sai Santhosh - 210101033 Himangshu Deka - 210101050
Naladala Navadeep - 210101072

Abstract

Credit risk analysis is a critical task in the financial industry, influencing lending decisions and risk management strategies. This project proposes a machine learning-based approach to credit risk analysis, aiming to predict loan statuses (e.g., completed, charged off, defaulted) based on borrower and loan attributes. Leveraging a comprehensive dataset containing borrower demographics, credit scores, employment status, loan amounts, and interest rates, our objective is to develop accurate predictive models capable of identifying high-risk borrowers and minimizing potential financial losses. The project involves exploratory data analysis, data cleaning, feature engineering, and model building techniques to extract insights and patterns from the data. Several machine learning algorithms, including logistic regression, decision trees, and Naive Bayes, will be trained and evaluated to determine the most effective model for credit risk assessment. The ultimate goal is to provide financial institutions with a reliable tool for credit risk management, enhancing lending practices and promoting financial stability.

1. Introduction

1.1. Motivation

The target problem of this project revolves around addressing credit risk analysis in the context of financial lending, particularly focusing on loans extended to individuals or small businesses. The motivation behind this endeavor stems from the significance of credit risk assessment in ensuring responsible lending practices and safeguarding financial institutions against potential defaults. In the Indian context, initiatives like the *Jan Dhan Yojana*, aimed at promoting financial inclusion by providing access to banking services to all citizens, underscore the importance of effective credit risk management to mitigate risks associated with lending to diverse and often under-served populations.

he primary objective of this project is to develop and deploy machine learning models capable of predicting loan statuses based on various borrower and loan attributes. By leveraging historical loan data, including borrower demographics, credit scores, employment status, loan amounts, and interest rates, the project aims to build predictive models that can accurately classify loan statuses such as completed, charged off, or defaulted. The overarching goal is to provide financial institutions with actionable insights for assessing credit risk, enabling them to make informed lending decisions and tailor financial products to meet the needs of diverse customer segments while minimizing potential financial losses. Through this project, we seek to contribute to the advancement of responsible lending practices, financial inclusion initiatives, and overall economic development.

1.2. Target Problem

Our objective is to develop various machine learning models for the classification problem where our goal is to predict whether the loan status of a loan applicant shall be completed or not.Our dataset includes 81 columns and around 113,000 data points which includes some seemingly important features such as Borrower APR,Borrower Rate,Prosper Rating,Employment Status,Occupation,Prosper Score etc.

1.3. Outline of the project

In this report we table the efforts that we have put so far

- Exploratory Data Analysis
- Data Preprocessing
- PCA
- Gaussian Naive Bayes and Model evaluation
- Logistic Regression and Model evaluation
- SVM and Model evaluation
- Decision Tree and Model evaluation
- Ensemble Methods and Model evaluation

2. Methods

2.1. Exploratory Data Analysis

Removing columns with a high proportion of null values enhances model performance, simplifies data handling, improves interpretability, and reduces computational complexity. This streamlines data preprocessing, leading to more accurate and efficient analysis.

2.2. Preprocessing

2.2.1. Label Encoding

Label Encoding is a technique used to convert categorical data into numerical format, where each category is assigned a unique integer label. It's useful because many machine learning algorithms require numerical input, and Label Encoding provides a simple way to transform categorical features into a format that these algorithms can understand.

2.2.2. One Hot Encoding

One-Hot Encoding is a technique used to convert categorical data into a binary format, where each category becomes a binary feature column. It's useful because it preserves the categorical nature of the data without introducing ordinality, which could mislead the model. Each category is represented by a binary column, where a value of 1 indicates the presence of the category, and 0 indicates absence. This method ensures that each category is treated equally by the model and is particularly beneficial when dealing with nominal categorical variables with no inherent order.

2.3. Standardization

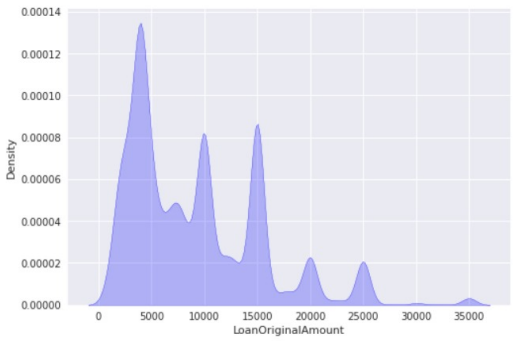

Figure 1. The largest concentration of loan amount are for 5000,10000 and 15000 $

Standardization transforms numerical data to have a mean of 0 and a standard deviation of 1. It equalizes feature scales, ensuring all features contribute equally to the analysis and preventing domination by those with larger scales. This preprocessing technique is vital for many Machine Learning algorithms.

2.4. PCA

High-dimensional data often contains noise or irrelevant information. PCA focuses on the directions in which the data varies the most, effectively filtering out noise and enhancing the signal-to-noise ratio. This can lead to better performance

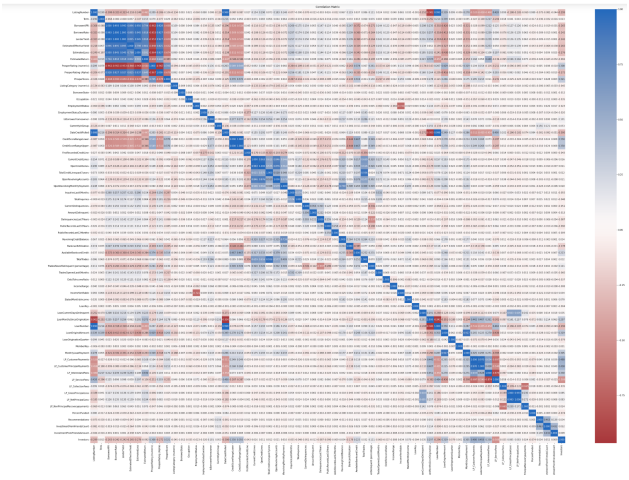

Figure 2. The correlation matrix of the columns

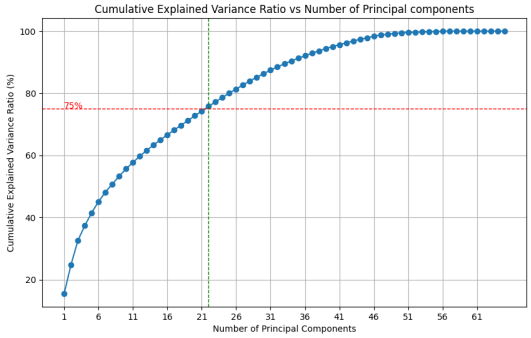

Figure 3. Explained variance ratio v/s Principal Components

in predictive modeling tasks by reducing overfitting but not at cost of much data loss. It simplifies data, eliminates noise, and identifies variable patterns. By identifying principal components, PCA captures maximum variance directions, providing a compact data representation.

Choosing 21 principal components out of 65 strikes a balance between retaining sufficient information and reducing dimensionality effectively. With this selection:

High Variance Retention: By retaining 21 components, we capture a substantial portion of the total variance in the data, ensuring that important patterns and structures are preserved.

Computational Efficiency: This choice optimizes computational resources, balancing dimensionality reduction with manageable computational complexity for subsequent analyses.

Interpretability: A smaller number of components facilitates interpretation, making it easier to identify and understand the most significant patterns and relationships within the data.

Mitigating Overfitting: Dimensionality reduction to 21 components mitigates the curse of dimensionality, reducing the risk of overfitting and improving model generalization.

2.5. Algorithms

2.6. SMOTE

```
In [75]: Y.value_counts()
Out[75]: 0    65189
         1    19664
         Name: LoanStatus, dtype: int64
```

Figure 4. Before SMOTE

```
In [83]: Y_train.value_counts()
Out[83]: 0    52094
         1    52094
         Name: LoanStatus, dtype: int64
```

Figure 5. After SMOTE

SMOTE (Synthetic Minority Over-sampling Technique) is a method for addressing class imbalance by generating synthetic samples for the minority class. It works by interpolating between existing minority class instances to create new synthetic examples. This technique helps to balance class distribution and improve model performance, particularly in scenarios with imbalanced classes.

2.7. Machine Learning Algorithms

2.7.1. Logistic Regression

It is crucial for modeling the probability of a binary outcome, making it essential for predicting loan defaults and assessing credit risk in our project.

2.8. Gaussian Naive Bayes

Bayes classification is important for its simplicity and efficiency in handling large datasets, making it invaluable for processing extensive borrower information and predicting loan outcomes.

	precision	recall	f1-score	support
0	0.96	0.83	0.89	6533
1	0.61	0.89	0.72	1953
accuracy			0.84	8486
macro avg	0.79	0.86	0.81	8486
weighted avg	0.88	0.84	0.85	8486

Figure 6. Logistic Regression scores

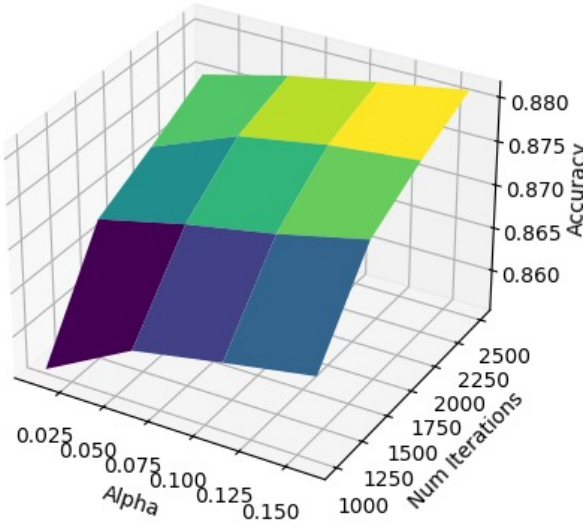

Figure 7. Logistic Regression Fine Tuning Graph

2.9. Support Vector Machine

SVM is significant for its ability to handle complex decision boundaries and high-dimensional data, making it well-suited for classifying borrowers into risk categories and optimizing lending strategies.

2.9.1. Decision Trees

Decision trees are vital for understanding the hierarchical relationships between different features, enabling us to identify key factors influencing loan approval or denial.

2.9.2. Ensemble Methods

Ensemble methods, such as Random Forests or Gradient Boosting, were employed to combine the strengths of multiple models and improve overall predictive accuracy.

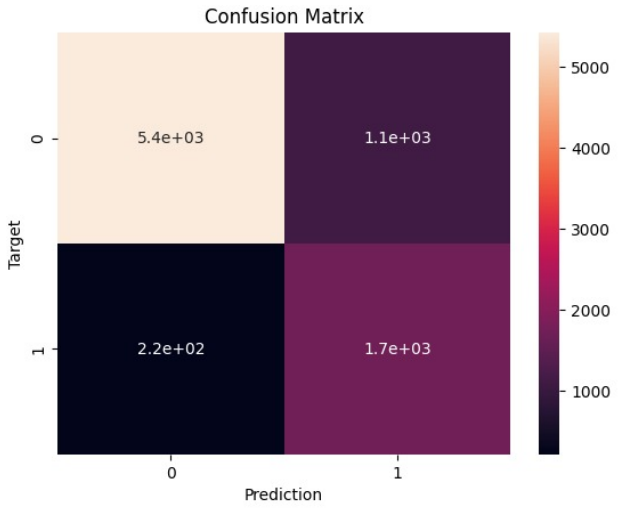

Figure 8. Confusion Matrix For Logistic Regression

```
[15]: from sklearn.metrics import classification_report, confusion_matrix
      print(classification_report(Y_test,rf_predictions))
```

	precision	recall	f1-score	support
0	0.98	0.94	0.96	6533
1	0.82	0.94	0.87	1953
accuracy			0.94	8486
macro avg	0.90	0.94	0.92	8486
weighted avg	0.94	0.94	0.94	8486

Figure 9. Random Forest Results

	precision	recall	f1-score	support
0	0.96	0.83	0.89	6533
1	0.61	0.89	0.72	1953
accuracy			0.84	8486
macro avg	0.79	0.86	0.81	8486
weighted avg	0.88	0.84	0.85	8486

Figure 10. Logistic Regression scores

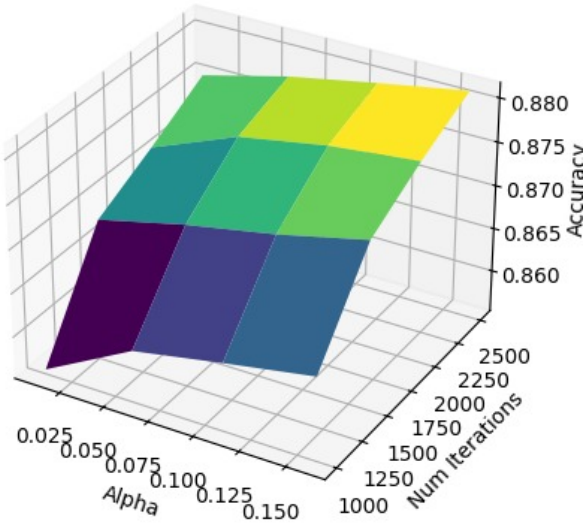

Figure 11. Logistic Regression Fine Tuning Graph

3. Experiements

3.1. Data Cleaning-Handling Null Values

We found out the percentage of entries are null values in the columns of our database. Then we proceeded to remove those columns that possess null values in excess of 50%. We then proceeded to find the data types of the columns and segregated the column space depending on whether the data types were 'object' or 'numeric'.This enabled us to fill the null values in such columns with the mode of that column in case it is of data type 'object' or else for those columns of data types being numeric. Now we have with all sanity regarded the column "Loan Status" to be our target column. So we then proceed to remove all such entries in the database(i.e. the rows) which have a null entry corresponding to the aforementioned column.

3.2. Exploratory Data Analysis

We found that most loan statuses are still current as depicted in , the largest amount of loans in the borrowed amount goes to the average of 5,000, 10,000, and 15,000 and that most of the loans fall within the three-year range as depicted in, followed by the five-year range .

We found that most of the people are employed however more than half lack possession of their own house . Even though on average a client is likely to have 6 accounts as depicted in Figure-8 the average monthly disbursement stands at around 1000 . We also concluded from the data at hand that most of the reasons for loans were among Debt Consolidation, Home, Business, and Others0.

3.3. Preprocessing

3.3.1. Label Encoding

For columns with labeled data, we utilized Labelled Encoding to encode the categorical columns. We endeavored to make the "Loan Status" column as our target variable.

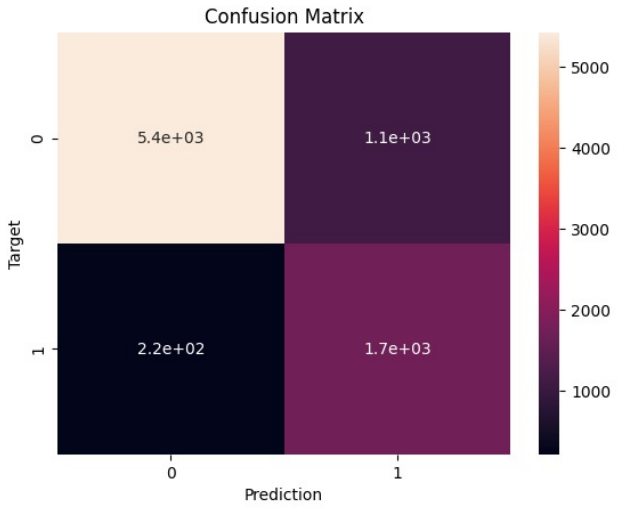

Figure 12. Confusion Matrix For Logistic Regression

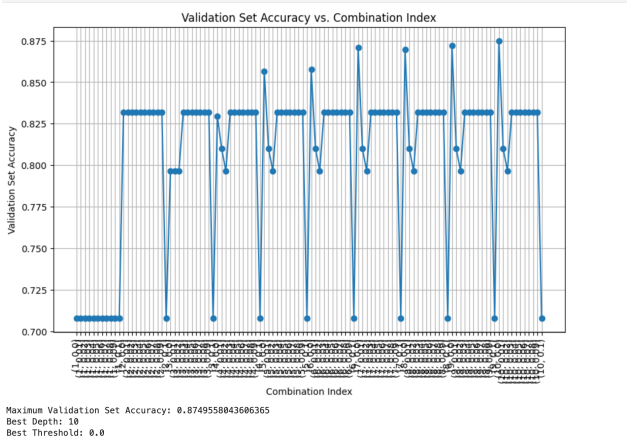

Figure 13. Hyper parameter tunin g for decision tree

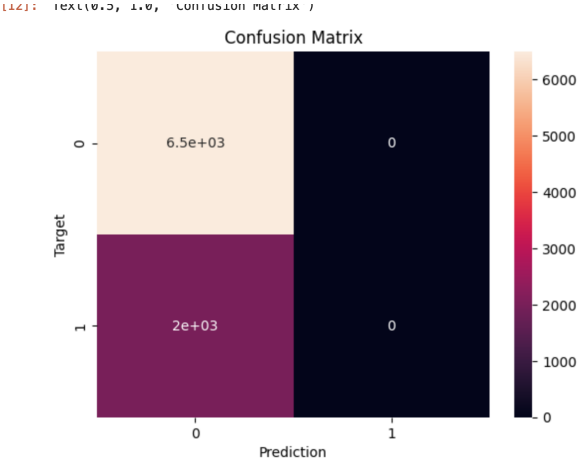

Figure 14. GNB confusion Matrix

3.3.2. One Hot encoding

As the ”Loan Status” column is categorical in nature, hence we used One Hot Encoding to categorize the entries as columns. As we wish to proceed with our problem as two class problem we shall only keep the ”Completed” column.

3.3.3. Train Val Test Split

Then we split our data set into train ,cv and test sets with the val set and test set size being 10% each.In this process, we considered the ”Loan Status” Column as the Target column i.e.”Y” and accordingly accorded these as Y_train,Y_val and Y_test.

3.3.4. standardisation

We standardized both the training and test data to ensure that they have a mean of 0 and a standard deviation of 1. This process is crucial for preparing data for proceeding later, as it helps in achieving better model performance and stability. We employed a function that calculates the mean and standard deviation from the training data and applies the same transformation to both the training and test sets. It returns the standardized training and test data, allowing for consistency in data preprocessing across different datasets.

3.4. PCA and Correlation Matrix

We plotted the correlation matrix as depicted in Figure-2 for the columns.This shows that there are many columns/features with high correlation among them ,so we proceed to perform PCA and select requisite no of features so that variance in data can be reduced and also decrease the time required to run the models,as we had higher no. of features which could have lead to higher variance if left untouched. As depicted in Figure-3 , we decided to put a threshold of 75% on explained variance ratio and selected 20 as the No. of Principal Components.

3.5. Data Imbalance

We applied SMOTE after applying SMOTE to the training set ,our results are shown in Figure-5.

3.6. Logistic Regression

We wrote the code for logisitic regression ,with epochs and learning rate being the hyper parameters.Thereafter we proceeded to plotting the val set accuracy against learning rate.

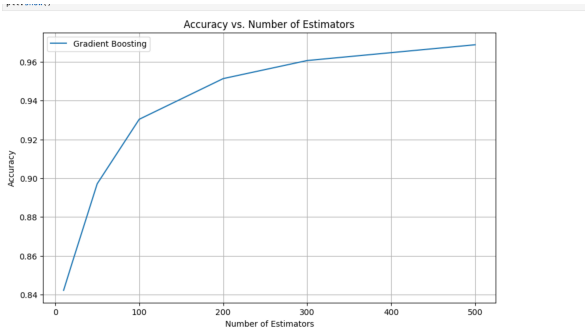

Figure 15. Gradient Boosting Results

```
[13]: from sklearn.metrics import classification_report, confusion_matrix
      print(classification_report(Y_test,gnb_predictions))
```

	precision	recall	f1-score	support
0	0.77	1.00	0.87	6498
1	0.00	0.00	0.00	1988
accuracy			0.77	8486
macro avg	0.38	0.50	0.43	8486
weighted avg	0.59	0.77	0.66	8486

Figure 16. GNB Results

3.7. Gaussian Naive Bayes

We wrote the code for GNB ,with alpha being our hyperparameter,where alpha parameter in Gaussian Naive Bayes serves as a smoothing factor to prevent zero probabilities and enhance the robustness of the model.

3.8. Supoort Vector Machine

We wrote the code for SVM ,with lambda and learning rate being our hyperparameters ,where lambda being the constant that multiplies epsilon values,in soft margin SVM classifier

3.9. Decision Tree

We wrote the code for Decision Tree ,with depth and threshold entropy being our hyperparameters However due to the sheer magnitude of our data we were unable to run our custom built algorithm,with the challenges and problems we faced being detailed below,we proceeded with inbuilt library. and did hyperparameter tuning.

3.10. Ensemble Methods

We used RandomForestClassifier and GradientBoostingClassifier ,as available in sklearn our datasets,our hyperparameters in this case was 'No. of Estimators',we plotted the Val Set Accuracy v/s No of Estimators.

```
[40]: #Import scikit-learn metrics module for
      from sklearn import metrics

      # Model Accuracy: how often is the clas
      print("Accuracy:",metrics.accuracy_scor

      Accuracy: 0.9013553329404832

[41]: y_pred_test = clf.predict(X_test_pca_df

[42]: # Model Accuracy: how often is the clas
      print("Accuracy:",metrics.accuracy_scor

      Accuracy: 0.9117369785529107

[45]:

      from sklearn.metrics import f1_score
      from sklearn.metrics import accuracy_sc

[46]: from sklearn.metrics import f1_score, r

      # Calculate F1 score
      f1 = f1_score(Y_val, y_pred)

      # Compute ROC curve and AUC
      fpr, tpr, thresholds = roc_curve(Y_val,
      roc_auc = auc(fpr, tpr)

      print("F1 Score:", f1)
      print("ROC AUC Score:", roc_auc)

      F1 Score: 0.8089477288290345
      ROC AUC Score: 0.9084677803448621
```

Figure 17. SVM results

3.11. Algorithms

4. Analysis

4.1. Outline

- Gaussian Naive Bayes
- Logistic Regression
- Support Vector Machine
- Decision Tree

Our results obtained for the above models are detailed in this report

Algorithm 1 Logistic Regression

Input: Training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, learning rate α, number of iterations N
Initialize weights w and bias b
for $k = 1$ **to** N **do**
  for each training example $(x^{(i)}, y^{(i)})$ **do**
    Compute predicted output $\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$
    Compute loss $L^{(i)} = -y^{(i)} \log(\hat{y}^{(i)}) - (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$
    Compute gradients $\nabla_w L^{(i)} = (\hat{y}^{(i)} - y^{(i)}) x^{(i)}$ and $\nabla_b L^{(i)} = \hat{y}^{(i)} - y^{(i)}$
    Update weights $w \leftarrow w - \alpha \nabla_w L^{(i)}$ and bias $b \leftarrow b - \alpha \nabla_b L^{(i)}$
  end for
end for

Algorithm 2 Decision Tree

Input: Training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, max depth d
Output: Trained decision tree T
Define SPLITENTROPY(D, f, t) to calculate split entropy based on feature f and threshold t
Define FINDBESTSPLIT(D) to find feature and threshold with lowest split entropy
Initialize decision tree T
function BUILDTREE(D, depth):
  If depth $>$ max depth or D is pure, create leaf node with majority class
  Else, find best split feature f and threshold t
  Split D into subsets D_{left} and D_{right}
  Create internal node with split rule (f, t) and attach left and right child nodes
$T \leftarrow$ BUILDTREE(training data, 0)

Algorithm 3 Gaussian Naive Bayes

Input: Training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, classes C
Separate dataset by class: $\{X_c\}_{c \in C}$ where X_c contains samples with class c
for $c \in C$ **do**
  Estimate class prior: $P(y = c) = \frac{|X_c|}{n}$
  Estimate class-conditional mean and variance for each feature: $\mu_{c,j} = \frac{1}{|X_c|} \sum_{\mathbf{x} \in X_c} x_j$, $\sigma^2_{c,j} = \frac{1}{|X_c|} \sum_{\mathbf{x} \in X_c} (x_j - \mu_{c,j})^2$
end for
Output: Trained Gaussian Naive Bayes classifier with class priors $P(y = c)$ and class-conditional means $\mu_{c,j}$ and variances $\sigma^2_{c,j}$

5. Challenges Faced

5.1. Data Imbalance

The Loan Status of customers in our dataset are under many various columns such as "Completed","Charged Off","Defaulted" etc,we took the conscious decision of converting all columns except "Completed" as category-0 and the aforementioned column as category-1;this creates a data imbalance for that we used "SMOTE";our dataset contains around 75% class-0 labels and 25% class-1 labels.

5.2. PCA

Upon running our models on a reduced dataset with 40 principal columns(95% threshold) but this yielded to overfitting when tested with Logistic Regression and Support Vector Machine.We made a conscious decision to select 20 PCs corresponding to 75% threshold

5.3. Inherent Bias in Data

Because of the class imbalance after train-test-split we ran our models on the test data thus generated,we got a test-set accuracy of 81% on logistic regression ,with an f1 score of 91% which pointed to a bias,we were able to identify this and used smote to resolve this class imbalance and thus resolved this bias problem.

5.4. Writing Algorithms

5.4.1. DECISION TREE

As we were dealing with huge amount of data(15 columns* 100,000 rows after PCA) running a custom Decision Tree algorithm was rendered infeasible.For a threshold entropy of 0.05 and a tree depth of 5 ,after running for 3 hours we got an test set accuracy of 77% , Hyperparameter tuning was thus rendered infeasible as there was no way we could plot the accuracy v depth and accuracy vs threshold entropy in feasible time.So we utilised sklearn library to build this model.

5.4.2. SVM

As we were dealing with huge amount of data(15 columns* 100,000 rows after PCA) running a custom SVM algorithm was rendered infeasible.So we endeavoured to use inbuilt SKlearn library and didn't perform hyperparameter tuning due to the dearth of resources and time at our end.

6. Conclusion

We can conclude from the results of our model that Gradient Boosting gives the best results in terms of F1 score.

References

Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2000.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (eds.). *An Introduction to Statistical Learning with Applications in R*. Springer Texts in Statistics.

Tian Zhenya, Xiao Jialiang, F. H. W. Y. Credit risk assessment based on gradient boosting decision tree. Elsevier B.V, 2019.

Yu, L. Credit risk prediction based on machine learning methods. The 14th International Conference on Computer Science Education (ICCSE 2019), 2019.

Dataset [https://www.kaggle.com/code/suwarnabaraskar/prosperloan-patch-3suwarna/input]