Loan Approval Prediction using Classification

Abhishek Sonawane (1225469895) Darshil Vora (1225542201)
Gaurav Kulkarni (1225477253) Kashish Khullar (1222111059)
Vishnu Vuggepalli (1225427671)

December 7, 2022

Abstract

Credit lending is a major source of economic upheaval across the world. We can streamline the borrowing process using Machine Learning. In this project we have utilized ML models for classification to predict if a borrower will fail to pay off the loan or not. We have used a vast data set composed of past loans and visualized the data to recognize its scope and fallacies. Then we cleaned the data and identified the dominant features. After feature selection, we have executed model fitting. We have utilized k Nearest Neighbors, Naïve Bayes, Logistic Regression, Support Vector Machine and Decision Tree models. Further we have implemented hyperparameter tuning to achieve the maximum accuracy on the test data. This project achieves a 100% accuracy using Decision tree models and very high accuracy for Support Vector Machines and kNN, much higher than most of the existing implementations published. This project could be utilized by financial institutions for the primary screening of loan applications, making the process straightforward, hassle-free, and unbiased.

1 Introduction and Motivation

Loans are as old as the invention of currency. They are also the most vital driving factor for the global economy. Debt controls the supply and demand of money in the economy, leading to competition in the market, social and economic growth, and overall development of the world. According to the International Monetary Fund, post covid-19 global debt surged by 28% to $226 trillion in 2020, the largest surge in 50 years[1]. Further, throughout history, we have seen a series of debt crises that have transitioned the social, economic, and political structure of the global society. Some of the significant such crises are the Great Recession of 2008, the Greek and Venezuelan debt crises, the Chinese Property Sector Crisis for 2020, the Sri Lankan economic disaster of 2022 [2].

Further, there is an abundance of structured data maintained by the financial institutions for decades, even centuries in certain cases. Most of this data is publicly available and can be easily deciphered by machine learning models. Lastly, the field has an immense scope for improvement through the intervention of smart automation and optimizations. The lending process is plagued with institutional delays, frauds, and a bias towards socially and politically marginalized communities[3].

Therefore, we decided to utilize Machine Learning techniques to predict if a loan applicant has a high probability of defaulting on his loan based upon historical data and the applicant characteristics. We utilized a large-scale dataset of historical loan applicants and their descriptive characteristics and used classification models to classify the final status of the loan – if it is completely paid off or has it defaulted, with the borrower unable to pay back the loan. We have utilized classification models covered in the syllabus namely, logistic regression, Naïve bayes, Support Vector Machines, which were apt for the loan approval classification

task. We further utilized k Nearest Neighbors and Decision trees to improve the accuracy of previous models. We also tuned the hyper parameters to achieve the highest possible accuracy. We verified the accuracy of our models with confusion matrix, classification report and accuracy score. Finally, we have compared our models with the similar loan approval implementations in academia.

2 Problem Statement

The main task for this project was to achieve an optimally accurate machine learning model which can predict if a loan should be approved or not. This can be restated as - is there a high probability that the applicant will successfully pay off all the loan and interest within the dictated time? We decided to utilize classification techniques to achieve this. Due to the abundance of dataset and the manifold applications of this project, the primary goal for the project was to achieve a high accuracy.

First, we decided to utilize a large-scale dataset. With a large data set, it is easier to extract insights on the features of the data[4]. We can easily detect intra collinearity amongst the distinct features and check for repetition of data. We can also easily determine the dominant features that have a high impact on the target and independent features which do not have any effect on the results. Finally, the scale of data ensures the soundness of our analysis. Therefore, we selected the Loan Default data set from Kaggle. This data set comprises 34 features and almost 150 thousand entries. The features plot the profile of the loan applicant including features like income, gender, credit score and age. It also contains details about the loan including loan type, amount, rate of interest, upfront amount, and security type for the loan. Finally, the target vector is represented by status which indicates if the loan was successfully paid off in full or not.

After that we had to get familiar with the technological implementation of our plan. During the proposal stage we had decided to use python along with sklearn, pandas, matplotlib for the project. We had to get familiar with the implementation and research about classification models and their execution. The next task for the project was to check the soundness of the data. Using visualization, we plot the various features of data and their counts. We also checked the intra collinearity of the features using a heat map. The visualization indicated that a high percentage of certain features were missing. We needed to clean the data and fill the null values.

We also observed during visualization that certain features like loan amount, rate of interest displayed significant outliers. But since this is a large dataset representing a large period, the outliers are expected. Loan amounts can differ for distinct applicants. The rate of interest is determined by the federal bank and varies significantly over time. Therefore, the data with the outlier was determined as sound and utilized as it was. The next task was to select the dominant features and delete some of the unimportant features that had no effect on the target status. Post feature selection we had to encode the data and divide it into training the testing data.

The next stage of the problem required fitting the now cleaned and processed data with appropriate models. Based upon our knowledge from the class and the technological learnings we had to execute the classification models and optimize them. We started with Logistic Regression and Naïve Bayes and Support Vector Machine. We also had to tune the hyper parameters for these models to optimize the accuracy. We verified various metrics for accuracy and chose the optimum weights. We followed that with newer models like k Nearest Neighbors and Decision Trees that we had learned about during the research phase. The final task in the problem was to analyze the accuracy of each model and compare it against the published results.

3 Methodology and Results

3.1 Preprocessing

The preprocessing step in any machine learning project is the most important step. A well cleaned, organized and processed dataset can provide much better results even if we don't have a model with high complexity. In this project we have used various preprocessing steps to clean

and improve our dataset before training models on them.

Dataset information : There are a total 34 columns and 148670 rows in our dataset with 21 categorial and 13 numerical features.

Missing or Duplicate Values : This dataset had a lot of missing values where some features had more than 30,000 missing values. However, there were no duplicate values in our dataset. To handle numerical missing values, we filled them with the computed mean value for those features. Since categorical missing values were high in number, we decided to drop those rows as we had enough rows to train our model.

Feature Selection : This is an important part of preprocessing where we choose features that positively impact the performance of the model and remove the features that cause degrade the performance of our model.

Remove unnecessary features : We visualized the categorical features with respect to our target variable and observed that some of the features only had no values of the target variable, as a result it was not contributing at all to predict the target variable. Therefore we decided to remove features 'open_credit', 'construction_type','Secured_by','total_units','Security_Type' from our dataset.

Next we also drop the features 'ID' and' year' since the 'ID' was different for each row and didn't contribute towards the classification, moreover, 'year' was constant for all rows.

Correlation Matrix and Multicollinearity : To further select features for our training, we used correlation matrix to detect high collinearity among our features and remove multicollinearity.

We chose a threshold of 0.7 Peason correlation for removing features that have high correlation as per [5]. None of the numerical features had high correlation that crossed our threshold so we decided to keep all the remaining numerical features.

Next we checked for columns that had no correlation with the target variable. We dropped 4 such columns that had zero correlation with "Status" namely "Interest_rate_spread", "Upfront_charges", "term" and "Credit_Score". Since Pearson correlation is only applied on numerical data [6], we used the Chi-Square test for detecting multicollinearity among our categorical features. We performed the H0 test and calculated the p-value for all our categorical features. None of our features had a value greater or equal to the chosen value of alpha = 0.05, therefore we rejected the H0 hypothesis and no features were removed here.

Encoding Categorical Features and Multicollinearity : We encoded all our categorical features using one hot encoder. To reduce multicollinearity we dropped the first feature while encoding and also dropped feature business_or_commercial for nob/c value, since it had high -1 Pearson correlation with feature loan_type for type2 value.

Splitting Dataset and Feature Scaling : We split our entire dataset into training and testing dataset in 70:30 ratio. Since we had a mixture of categorical and numeric features in our dataset, we scaled the dataset using mean and standard deviation. After performing all these steps our dataset was ready to be trained on various models.

Once the preprocessing was completed we trained different supervised models on the resulting dataset which have been described in the sections that follow.

3.2 Logistic Regression

Logistic regression uses sigmoid function to estimate the probability of an event. The dot product of parameter vector θ and input vector is given as input to the sigmoid function and the corresponding output represents the estimated probability of success, in our case, approval of the loan application. A threshold value can be set to classify an example depending on the value of the estimated probability, usually the value of this threshold is 0.5.

During the training phase, the model learns the parameter vector based on the values in the training dataset. The cost function is a convex function with respect to parameters θ. Gradient descent can be used to compute the parameter vector but the algorithm involves scanning the entire dataset before taking a step towards the global minimum which makes it inefficient while working on a large dataset like the one in our project. Stochastic gradient descent is a feasible alternative in which we consider one training example to update the value

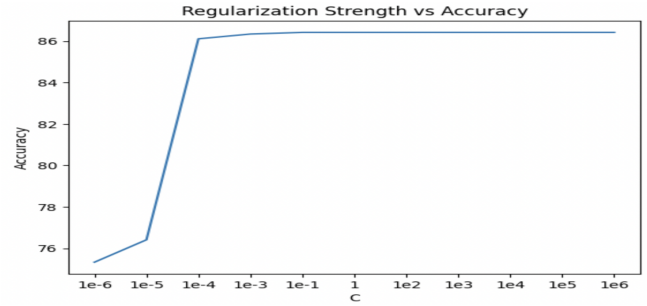

Figure 1: Regularization vs Accuracy for logistic regression

of the parameter vector. But both these methods require selection of a learning rate which determines the step size each time the parameter vector is updated. Schmidt et al. in [7] have proposed an enhanced version of the stochastic gradient descent algorithm which has faster convergence. Moreover, their approach uses line search techniques to find the right value of the learning rate thereby making it easier to learn the right model.

In this approach the parameter vector is updated by using the average values of gradients of all examples in the training data. In each iteration, the algorithm randomly updates the gradients of just one training example, due to this the average of all gradients can be computed in constant time while updating the parameter vector. Thus, in terms of asymptotic time complexity the stochastic average gradient technique is as efficient as stochastic gradient descent. The graph 1 illustrates the results obtained for different values of regularization strength, C, which is inversely proportional to L2-regularization parameter λ. It was observed that for lower values C a larger penalty was applied on the value of parameters due to which the model did not fit the data well whereas as C was increased the model was able to choose larger parameter values thus being able to fit the dataset better.

Performance metrics for C = 10^6

Class	Precision	Recall
0	0.85	0.99
1	0.94	0.48

Accuracy : 86.42%

3.3 Decision Trees

A decision tree classifier constructs a binary tree which has two types of nodes, leaf nodes and decision or internal nodes. Each decision node uses a criterion to decide which of its two branches must be followed while classifying an example. All examples that reach a particular leaf node are assigned the same class label. Building an optimal decision tree is an NP-hard problem [8], therefore, a greedy algorithm is used to find the best possible decision tree in feasible time.

Greedy construction of a decision tree starts by considering the entire dataset at the root node. At each node the algorithm finds the best criteria to split the set of data points at that level. The algorithm stops when it reaches a node which has all points that belong to the same class or some stopping condition like maximum depth or minimum split is reached. Information gain is used as a metric to select the best criterion out of all the criteria that are possible at a node. The best criterion is the one which separates the data points into two sets such that each of those two sets contains points belonging to the same class. The criterion which gives highest information gain is selected for doing the split. The value of information gain is the difference between the impurity of the parent node and the combined impurities of the child nodes weighted as per the size of child nodes as compared to the parent node. Impurity is a measure of homogeneity of the node [9]. Nodes that have all labels belonging to the same class have lower impurity. Two commonly used measures of impurity are entropy and gini index, gini index $= 1 - \sum_j p_j^2$ and entropy $= -\sum_j p_j Log(p_j$ where p_j is the probability of class j.

Both measures give practically identical results [10] but gini index is computationally less expensive since it doesn't have the Logarithm term. 2 and 3 show the decision trees that were generated.

Performance metrics for both trees

Class	Precision	Recall
0	1.0	1.0
1	1.0	1.0

Accuracy : 100%

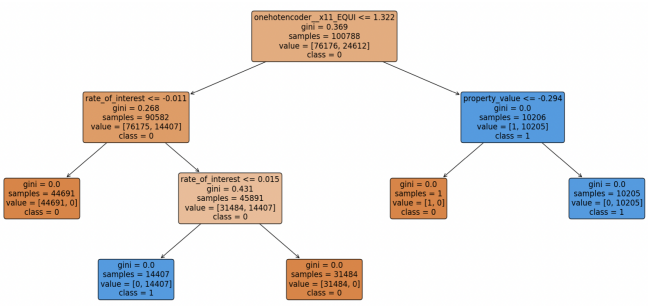

Figure 2: Decision Tree with Gini Index

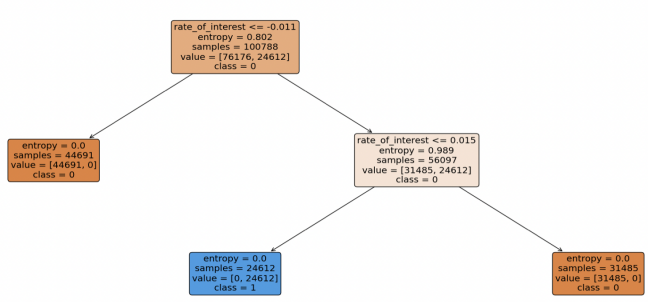

Figure 3: Decision Tree with Gini Index

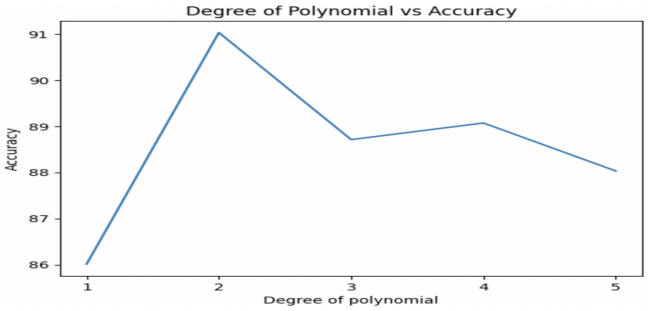

Figure 4: Polynomial kernel with different degrees

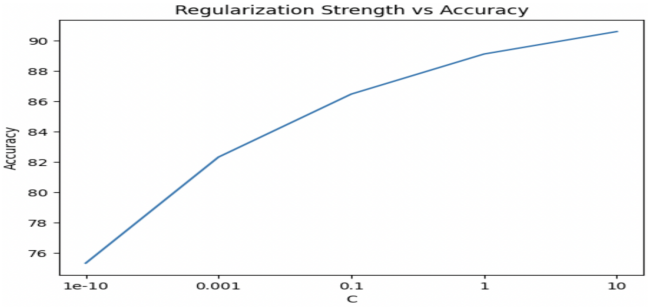

Figure 5: RBF kernel with different regularization strength

3.4 Support Vector Machines

Support Vector Classifier selects a hyperplane which can separate the examples belonging to different classes such that the distance between the closest training example and the separating hyperplane is as large as possible, resulting in a hard margin classifier. This approach could lead to overfitting and sensitivity to outliers in order to avoid this some misclassifications are allowed to get a separating hyperplane which can generalize well giving us a soft margin classifier. In many cases, the data might not be linearly separable to find non-linear decision boundaries, Support Vector Machines make use of kernel functions to transform the data so that it can be classified by a support vector classifier. The kernel computes relationships between every two transformed data points, these relationships are used to find support vector classifiers. We used the following two kernel functions.

1. Polynomial Kernel : $(a*b+r)^d$, a and b are two different observations in the dataset, r is the coefficient of the polynomial, d is the degree of the polynomial. 4 shows that quadratic kernel gave better performance than other higher order polynomial kernels.

2. Gaussian Radial Basis Function (RBF) Kernel : $e^{-\gamma(a-b)^2}$ a and b are two different examples and γ is a scaling factor. 5 shows that with lower values of regularization strength, C, the margin was wider hence, more misclassifications were allowed. With higher value of, C, narrower margin was achieved and performance improved.

Since the kernel function computes relationships for every two pairs of points, the SVM model took more time to learn with a large dataset as compared to the other models we implemented.

Performance metrics for quadratic kernel with C = 10

Class	Precision	Recall
0	0.91	0.97
1	0.90	0.72

Accuracy : 91.03%

3.5 k Nearest Neighbors (KNN)

KNN is among the simplest machine learning algorithms, being non-parametric and lazy in nature. Non-parametric means that no assumptions are made about the underlying data distribution, so the model structure follows the dataset. The Lazy or instance-based learning method simply means that the training data is not required for model generation and the whole training set of data is applied in the testing phase.

The KNN algorithm is performed in 2 steps:-

1. 1st Step:- In this step, each sample in the training set has its K nearest neighbor computed and stored in the system.

2. 2nd Step:- Then among these k-nearest neighbors, it predicts the class through voting.

Generally KNN in supervised learning can be used for both classification and Regression but for our dataset, we wanted to use it for classification. Here in our case, we have used Scikit Learn library's KNN classifier to perform the algorithm and get the best accuracy.

Since our dataset was very large and contained many features, we needed to first find the value of K, for which it gives the maximum accuracy. If we do it manually, it will become too much of a computationally heavy task. So, that is why to solve this problem, we have used the GridSearchCV algorithm provided by the Scikit Learn library to compute the value of K.

Performance metrics with K = 17

Class	Precision	Recall
0	0.86	0.99
1	0.92	0.51

Accuracy : 86.81%

3.5.1 GridSearchCV

It is available in the sklearn.model_selection.GridSearchCV module. GridSearchCV is the methodology for determining optimal values for a set of hyperparameters in a given model. Here there is no way to know the best value of hyperparameters in advance, so we need to try all possible values to determine which is best for our model.

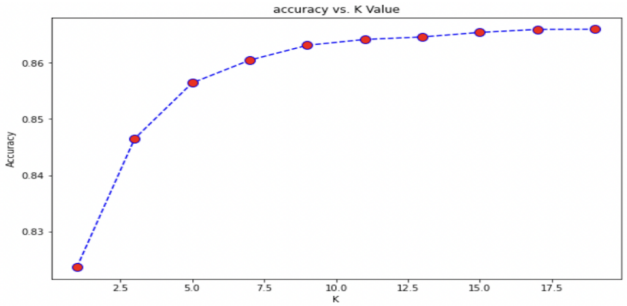

Figure 6: Performance of KNN with different values of K

So, we can pass a range or a list of parameters, in GridSearchCv and it will give us the best parameters by generating the model and checking the accuracy for each hyperparameter. So, this is how we found the best value of K at K=17. We have also plotted the k vs accuracy graph to verify the result and we get the best accuracy at $K = 17$, which confirms the result of the GridSearchCV algorithm.

So as we can see in 6, we get the best accuracy at K=17 and it decreases gradually after that.

We know that generally, for a large dataset which contains many features like ours, RandomizedSearchCV tends to give better results. Because it does not consider all the values from a given range, it randomly chooses the values based on how many values we want to consider. But with RandamizedSearchCV there was one problem, that it does not always guarantee the best results. Although it was less time consuming, there was not much difference in running time of RandomizedSearchCV and GridSearchCV. Since we wanted to get the best results, we used GridSearchCV instead of RandomizedSearchCV.

3.6 Gaussian Naive Bayes

We wanted to test our dataset by building a Gaussian Naive Bayes model, because it considered all the features independent of each other. So, it generally gives better accuracy, for large datasets.

The GNB(Gaussian Naive Bayes) model takes into account continuous valued features and models them as being distributed according to a normal(gaussian) distribution. By assuming a Gaussian distribution with independent dimensions, a simple model can be created by finding mean and standard deviation of the points within each label.

Here we have used Scikit Learn library's sklearn.naive_bayes.GaussianNB to build the Naive Bayes classifier. We have also used the GridSearchCV algorithm similar to KNN, with different parameters to get the best accuracy for our model. Here we have used the var_smoothing parameter for our model. Var_smoothing is Portion of the largest variance of all features that is added to variances for calculation stability. So, the default value of this is 1e-9, but we get the best results at 1e-30. The result did not change much after that and we were already getting around 99% accuracy, therefore, we didn't try smaller values.

Performance metrics with Laplace smoothing = 1e-30

Class	Precision	Recall
0	1.0	1.0
1	1.0	0.99

Accuracy : 99.99%

4 Related Work

Ref	Log. Reg.	D. Tree	SVM	KNN	GNB
[11]	80	91.1	84.44	93.32	-
[12]	69.1	60	95	89.1	-
[13]	85.2	82.1	95.6	90.2	91
[14]	81.17	97.26	83.17	-	83.71
Us	86.42	100	90.45	86.59	99.99

Accuracy in (%) published in different papers

As our project focuses on increasing the accuracy of the loan approval model we compared our accuracy with few other papers who have performed similar methods and we were able to achieve better accuracy. 4 summarizes the performance of our work with that of others

5 Conclusion

We were able to train and analyze the performance of all the models which were outlined in our proposal. The decision tree classifier gave best performance among all the models we implemented. A reason behind this could be that the nature in which humans process loan application aligns well with the framework of decision trees. For instance, if credit score is below a particular value, the loan application is rejected, such instances are captured by a decision tree effectively. We recognized that performing an effective data preprocessing task is an essential step and invested a lot of our time in learning different data cleaning, visualization, and statistical analysis techniques which helped us in getting a dataset which gave high accuracy when trained on different models later.

Such machine learning techniques can help transform the banking sector especially in developing and under-developed economies where the process of approving loan applications is slow and biased. Moreover, it can also help the bankers cut down on their human resources expenditure by utilizing autonomous decision making systems like ours.

Our analysis was biased towards accuracy. Next time we will keep the imbalanced nature of the dataset into account and try to judge models on the basis of their precision and recall scores putting more emphasis on reducing false positives since such errors are more harmful for the banks.

5.1 Future Work

- Train and analyze the performance of neural network and random forest classifiers on the dataset.
- Use PCA or other dimensionality reduction techniques to improve the performance.
- To use elastic net regression instead of L2 Regression and observe any performance changes.

References

[1] V. Gaspar, P. Medas, and R. Perelli, "Global debt reaches a record $226 trillion," *IMF Blog*.

[2] B. Eichengreen, "Historical research on international lending and debt," *Journal of Economic Perspectives*, vol. 5, pp. 149–169, June 1991.

[3] A. S. Messai and F. Jouini, "Micro and macro determinants of non-performing loans," *International Journal of Economics and Financial Issues*, vol. 3, p. 852–860, Sep. 2013.

[4] A. Althnian, D. AlSaeed, H. Al-Baity, A. Samha, A. B. Dris, N. Alzakari, A. Abou Elwafa, and H. Kurdi, "Impact of dataset size on classification performance: An empirical evaluation in the medical domain," *Applied Sciences*, vol. 11, no. 2, 2021.

[5] D. Nettleton, "Chapter 6 - selection of variables and factor derivation," in *Commercial Data Mining* (D. Nettleton, ed.), pp. 79–104, Boston: Morgan Kaufmann, 2014.

[6] R. Arnab, "Chapter 19 - complex surveys: Categorical data analysis," in *Survey Sampling Theory and Applications* (R. Arnab, ed.), pp. 645–671, Academic Press, 2017.

[7] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1, pp. 83–112, 2017.

[8] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is np-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.

[9] J. Upendar, C. P. Gupta, and G. K. Singh, "Statistical decision-tree based fault classification scheme for protection of power transmission lines," *International Journal of Electrical Power & Energy Systems*, vol. 36, no. 1, pp. 1–12, 2012.

[10] "Decision trees: Gini vs entropy." `https://quantdare.com/decision-trees-gini-vs-entropy/`.

[11] U. E. Orji, C. H. Ugwuishiwu, J. C. N. Nguemaleu, and P. N. Ugwuanyi, "Machine learning models for predicting bank loan eligibility," in *2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*, pp. 1–5, 2022.

[12] N. Pandey, R. Gupta, S. Uniyal, and V. Kumar, "Loan approval prediction using machine learning algorithms approach," in *International Journal Of Innovative Research In Technology*, pp. 898–902, June 2021.

[13] M. A. Sheikh, A. K. Goel, and T. Kumar, "An approach for prediction of loan approval using machine learning algorithm," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 490–494, 2020.

[14] K. Pathak and S. Shaikh, "Loan approval prediction using machine learning," in *International Journal Of Innovative Research In Technology*, pp. 1897–1900, September 2021.