Foundations of Data Science

Final Project

Loan Default Prediction: A Data Analysis Report

Ahmed Essa Siddiqui	**26753**
Aminah Siddiqui	**26871**
Maryam Moin	**25207**
Nancy Charyani	**27181**

Abstract

This report investigates the application of machine learning algorithms for predicting loan approvals based on applicant information. Various algorithms, including Decision Trees, Naïve Bayes, and K-Nearest Neighbors (KNN), were employed and evaluated using metrics like Accuracy, Precision, and Recall to determine the best algorithm. The research methodology involved preprocessing techniques such as handling missing values, feature extraction, and data normalization. Limitations such as long training times and susceptibility to overfitting were identified, and potential solutions were proposed, including hyperparameter optimization and data enhancement strategies.

Problem Description

Financial institutions struggle with the challenge of approving loans to qualified borrowers. This is because they need to assess the risk of a borrower defaulting on the loan, meaning failing to repay it. Predicting loan default allows lenders to make informed decisions and reduce financial losses. Several factors can be considered while making the prediction such as:

Applicant Demographics: Age, marital status, education level

Financial Information: Income, employment status, debt-to-income ratio

Credit History: Credit score, history of late payments, bankruptcies.

By analyzing vast amounts of past loan data with known default outcomes, machine learning algorithms can identify patterns that correlate with loan defaults. This information can then be used to create a predictive model that assesses the likelihood of a new applicant defaulting on a loan.

Softwares Used

MS Excel

Our dataset was in the form of a CSV file. The non-accounted for values were replaced with empty cells to indicate missing values and help preprocessing in Jupyter software.

Jupyter

We applied the classification algorithms and evaluated their respective scores in Jupyter.

The following classification algorithms were applied:

1. Decision Tree
2. Logistic Regression Classifier
3. Random Forest Classifier
4. KNN
5. SVM
6. Naive Bayes
7. ANN

They were evaluated with different preprocessing techniques to deduce the most accurate set.

Data Understanding

The data analytics project falls under the category of '**Classification**'. The objective of the project is to predict whether the person is likely to get a loan based on the variables such as Applicant Demographics, Financial Information and Credit History.

Since it is a classification project, different classification algorithms can be applied to achieve the results. The classification algorithms applied for this project include Decision Tree Model, Naïve Bayes Model, K-Nearest Neighbor Classifier.

The evaluation metric used for this project is Accuracy, Precision,Recall, and based on different scores, the result with the highest accuracy was chosen to be the configuration for the predictive purpose.

The dataset consists of a total of 12 features, which was divided into train and test sets with a relatively 70:30 ratio. The vast attributes available were used to predict loans in terms of 0 and 1, where 0 representing 'No' and 1 representing 'Yes.'

Data Preprocessing

The data preparation consisted of several alterations to proceed to the modeling stage of the CRISP DM cycle. This step converted the raw stroke data set to a clean one. Data preprocessing helps improve the accuracy score for the data and makes it capable of being fed into the algorithm. The following operations were applied to the data with the respective nodes:

1. Handling Missing Values (Missing Value node)
2. Feature Extraction (Column Filter node)
3. Conversion into numerical Data (Category to Number node)
4. Partitioning (Partitioning node)
5. Data Normalization (Normalizer node)

Handling Missing Values

We replaced the missing values in the categorical columns with the mode of that column because it preserves the distribution of the data, also we converted the value 3+ as 3 in the dependents column and converted the column into a float. Then we did label encoding to convert these categorical columns into numerical.

Feature Extraction

In the final loan data set we did not have an id column. Therefore, to avoid the same attribute occurring twice the ID column was filtered out.

Conversion into numerical Data

There were six categorical attributes in the data required by the algorithm to predict the possibility of a loan. The category to number node transformed the data into binary values of 0 and 1. This included the conversion of following attributes:

Gender	Marital Status	Dependants	Education	Self Employed	Property Area

Partitioning

The stroke data set was a complete collaborative file of the test and train data. Through the partitioning node, the data was divided into 2 portions: 70% train data and 30% test data.

Data Normalization

The data consisted of attributes with carrying scales which needed adjustment. This is useful for the classification algorithms as it does not let any of the attributes overwhelm others, resulting in poor accuracy statistics.

decimal scaling normalization techniques were applied to see the difference in the accuracy scores. decimal scaling produced different results when applied to only train data

Data Modeling

Decision Tree Model

The decision tree model was used on the preprocessed data, one hot encoding and normalization were used, with changes being made in the normalization technique used, since our data was already preprocessed. This limits the tree to a maximum depth of 20 and specifies the minimum number of samples required to split an internal node. A chosen value of 2 means that each node that represents more than one sample can potentially be split, allowing a detailed splitting of the tree. The minimum number of samples a leaf node must have. A value of 1 means that each leaf can represent only one sample, which facilitates a fine-grained classification with the risk of introducing noise into the dataset. By using the same random seed (42 in this case) you ensure that the splits distributed in the tree are deterministic and the experiment is reproducible.

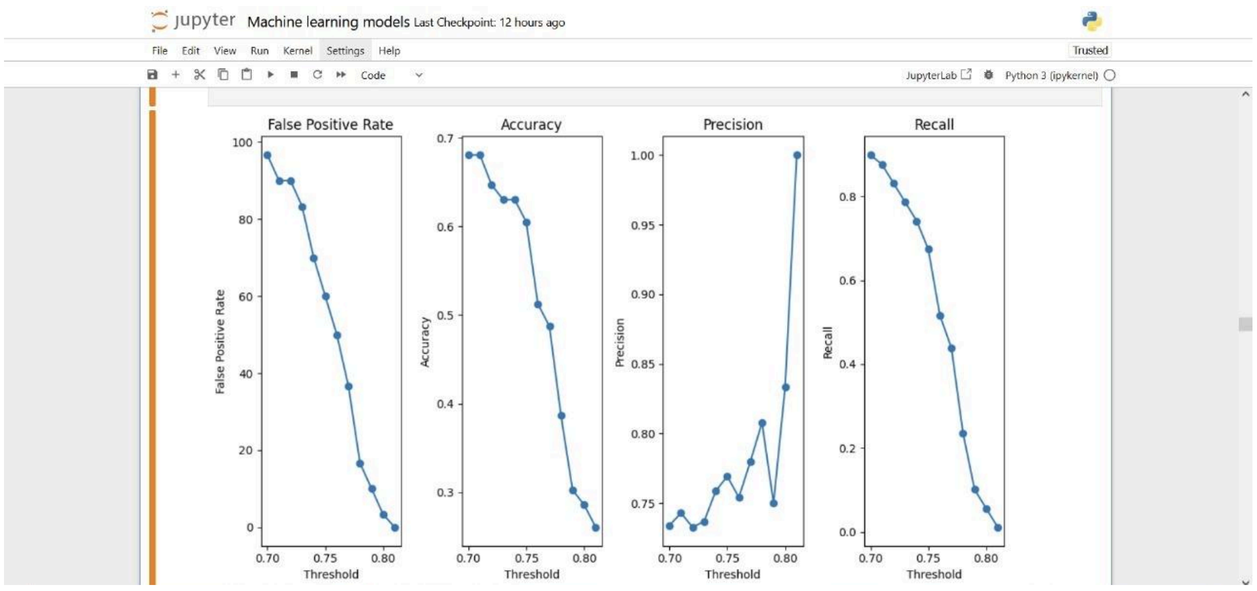
K-Nearest neighbor (KNN)

The KNN algorithm was applied to the loan data set. All the requirements for the algorithm were met for the prediction of the loan. The data consisted of well-labeled records. The proximity metric to compute distance/ similarity between the records was done through the KNN node in Jupyter. The best hyperparameters were checked and the seed was locked just to make sure the value stayed the same every time. Then we checked different values of hyperparameters such as N neighbors and we found out the best value was 7 and the weight was distance. This was done to reduce the false positive rate and achieve better precision accuracy and recall.

Logistic Regression Classifier

It imports the logistic regression model, a popular linear model used for binary classification tasks.

By looking at the following screenshot, we deduce that the best value for the threshold should be set at 0.75.



SVM

The best SVM model was fitted with a radial basis function (rbf) kernel. The C parameter, set to 22, trades off the maximization of the decision margin for correct classification. "Gamma" set to "scale" will automatically adjust itself to the number of features. Once the optimal settings were determined, our model was trained on the training dataset and used to make predictions on the test set.

Random Forest

In the configuration of the Random Forest classifier used in the loan prediction project, several key hyperparameters were fine-tuned to optimize performance. The model was specified with n_estimators=250, which refers to the number of trees in the forest. Using a larger number of trees generally helps to reduce the risk of overfitting, although it increases the computational load. Max_depth was limited to 20, which controls the maximum number of levels in each decision tree to effectively manage model complexity. In addition, the parameters min_samples_split=2 and min_samples_leaf=1 were chosen to dictate the minimum number of samples required to split an internal node and remain in a leaf node, respectively. These settings are important because they help avoid overly complex trees and thus improve the generalizability of the model.

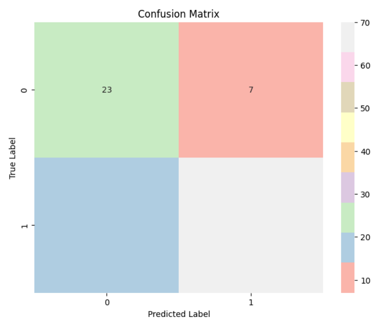
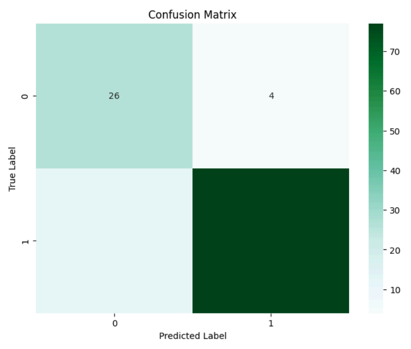
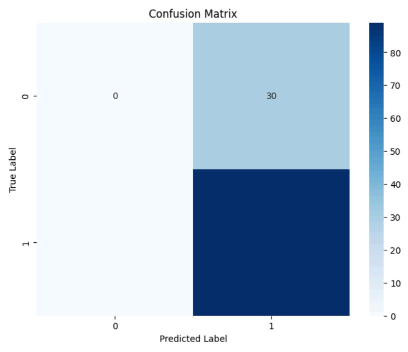
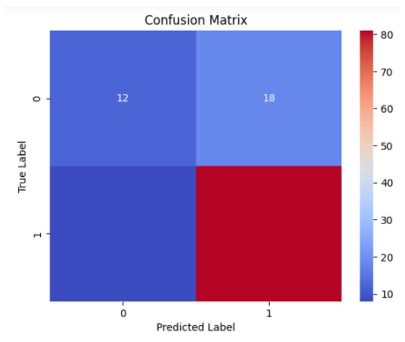
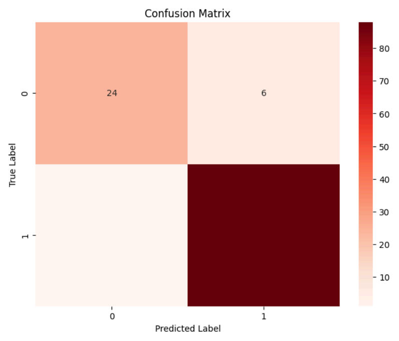
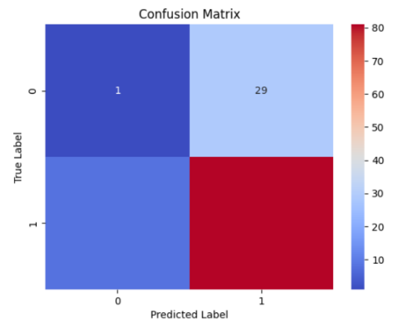
Naïve Bayes

The Naïve Bayes classifier was used to interpret the differing results of the accuracy statistics. The data was prepared for this algorithm by reduction of the ID column which would have violated the class conditional independence assumption of the classifier. The conditional independence assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. There are no hyperparameters.

Summary of our Findings

Algorithm	False Positive Rate	Precision	Accuracy	Recall
Decision Tree	23.33%	0.91%	0.78%	0.79%
Logistic Regression Classifier	60%	0.61%	0.77%	0.67%
Random Forest Classifier	20%	0.94%	0.94%	0.99%
KNN	13.3%	0.95%	0.87%	0.87%
SVM	60%	0.82%	0.78%	0.91%
Naive Bayes	96.67%	0.74%	0.69%	0.91%
ANN	63.33%	-	-	-

The following confusion matrices were obtained:

Algorithm	Confusion Matrix	Algorithm	Confusion Matrix
Decision Tree		**KNN**	
Logistic Regression Classifier		**SVM**	
Random Forest Classifier		**Naive Bayes**	

The model that predicted the best results was **Random Forest Classifier**.

Limitations of the Approaches

When examining the loan approval dataset with various preprocessing techniques and algorithms, it became apparent that classification algorithms present notable limitations that hinder the efficacy of the results for drawing definitive conclusions.

Decision Tree

The decision tree algorithm's lengthy training time and resource-intensive nature make it impractical for timely loan approval decisions in organizations like financial institutions. Its complex calculations and susceptibility to compromised outcomes with minor dataset alterations pose significant limitations to drawing reliable conclusions.

Support Vector Machine (SVM)

SVMs can be slow and require a lot of computational resources, making them less suitable for real-time applications. Additionally, SVMs can struggle with large datasets and may not perform well when the data is noisy or contains overlapping classes.

K-Nearest Neighbors (KNN)

KNN can be computationally expensive, especially with large datasets or high values of k. Additionally, KNN is sensitive to the choice of distance metric and may not perform well with high-dimensional data.

Naive Bayes:

Naive Bayes assumes that features are independent, which may not hold true in many real-world datasets. This can lead to suboptimal performance, especially when there are strong dependencies between features. Additionally, Naive Bayes can be sensitive to imbalanced datasets.

Random Forest

Although Random Forests generally perform well across various datasets, they can be prone to overfitting, especially with noisy data or datasets with a large number of irrelevant features. Random Forests can also be computationally expensive, particularly with a large number of trees in the ensemble.

Logistic Regression

Logistic Regression assumes a linear relationship between the features and the log-odds of the target variable, which may not always be the case in real-world datasets. Additionally, Logistic Regression may struggle with non-linear relationships between features and the target variable.

Possible Solutions to Counter the Limitations

To address the limitations of these algorithms, several approaches can be considered. For slow or resource-intensive algorithms like SVM and KNN, optimizing hyperparameters or using dimensionality reduction techniques can help improve efficiency. Additionally, for algorithms like Naive Bayes, addressing feature dependencies through feature engineering or selecting more appropriate models for the data can enhance performance. To mitigate overfitting in Random Forest, techniques such as feature selection, pruning, or ensemble regularization can be employed. Finally, for Logistic Regression, exploring more flexible models like polynomial regression or incorporating interaction terms can help capture non-linear relationships in the data more effectively. Regularization techniques can also be applied to prevent overfitting.

Overall, careful consideration of the specific limitations of each algorithm and appropriate adjustments or enhancements can lead to improved performance and more reliable results in practical applications.

We can enhance our model's optimization and improve predictions by gathering additional records with fewer missing values, ensuring a more comprehensive dataset.