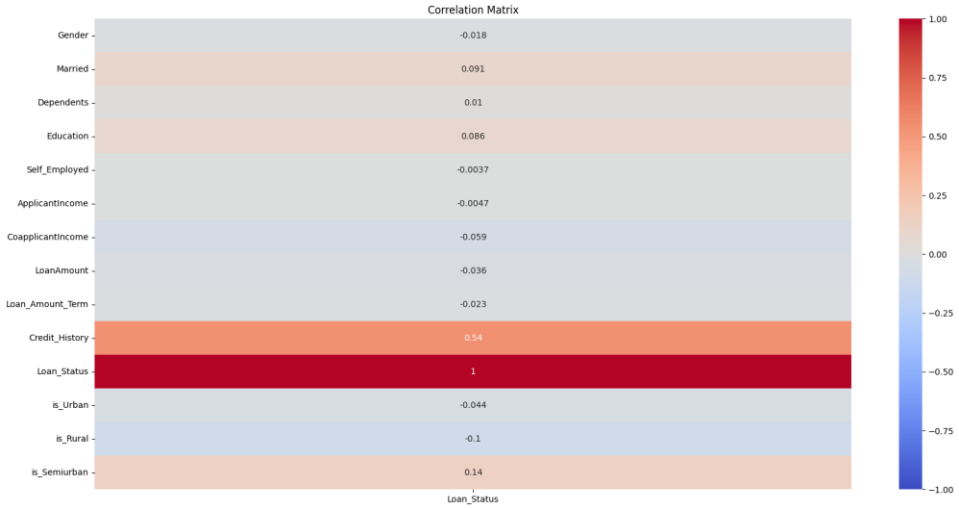
Predicting Loan Approval Using Machine Learning

Intro

Banks assess various factors of an individual's financial situation to decide whether they are likely to repay a loan. Our goal is to develop a machine learning model that can predict loan approval outcomes. This tool could be beneficial for potential borrowers to evaluate their chances of obtaining a loan. Additionally, it could assist banks in streamlining their loan approval process.

Data

Our dataset, hosted on Kaggle, includes 614 records with 13 attributes detailing individuals' financial status and their loan approval outcomes. Initially, we'll address missing values in several columns: Gender, Married, Dependents, Self_Employed, LoanAmount, Loan_Amount_Term, and Credit_History. For the continuous variable LoanAmount, we'll use the mean for imputation. For the others, being categorical or discrete, we'll employ the mode. Next, we'll convert categorical data into binary format for compatibility with our algorithms. This includes encoding Gender, Married, Education, Self_Employed, and Loan_Status as 0 and 1, while Property_Area will be split into three separate columns to reflect its non-sequential three-category nature. Additionally, we'll modify the Dependents column, changing '3+' to '3' and converting the entries from characters to integers.



After cleaning the data, we'll examine a correlation matrix to understand feature interrelationships. Notably, Credit_History shows the highest correlation with Loan_Status, at 0.54. We'll then divide the dataset into an 80/20 split for training and testing. We standardize both to enhance our models' ability to interpret the data relationships more effectively.

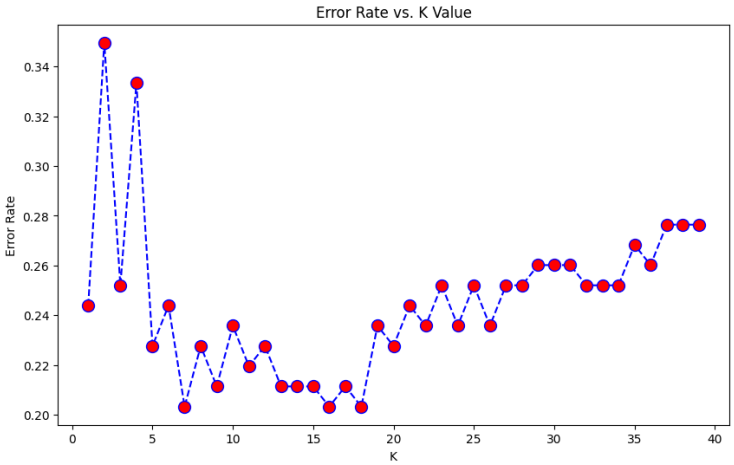
Results

Baseline

We began our modeling phase with a Naive Bayes classifier as our baseline model. This choice was influenced by the Naive Bayes classifier's reputation for simplicity and its ability to efficiently process categorical data, which is prevalent in our dataset. By employing the GaussianNB function from the Scikit-Learn Python library, we set up a standard Naive Bayes classifier. This preliminary model achieved an accuracy of around 78%.

K Nearest Neighbors

Next, we selected the K-Nearest Neighbors (KNN) classifier for our subsequent model, utilizing the KNeighborsClassifier from the Scikit-Learn Python package. The KNN approach was chosen for its ability to make predictions based on proximity to the nearest data points, which is particularly effective for datasets with clear groupings or clusters. Our initial KNN model, with the number of neighbors set to 3, achieved an accuracy of approximately 75%.



To optimize the number of neighbors, we plotted the error rate against different K values. Based on this analysis, we adjusted the model to use 7 neighbors, which improved our accuracy to around 80%

Neural Net

For our most advanced model, we implemented a neural network using Scikit-Learn's built-in MLPClassifier, a Python package for multi-layer perceptrons. We initially used the default settings of the MLPClassifier, modifying only the 'random_state' parameter for consistent results across multiple runs, and 'verbose' to monitor the progress during execution. This initial setup resulted in an accuracy of approximately 76%. To further refine the model, we then executed a grid search, experimenting with different parameter configurations.

'learning_rate_init': [0.001, 0.01, 0.1], - controls the step-size in updating the weights

'hidden_layer_sizes': [(50, 50, 50), (100, 50, 100), (100, 100, 100), (25, 50, 25)], - represents the hidden layers in the model and their sizes (size of 1st layer, size of second layer, size of third layer)

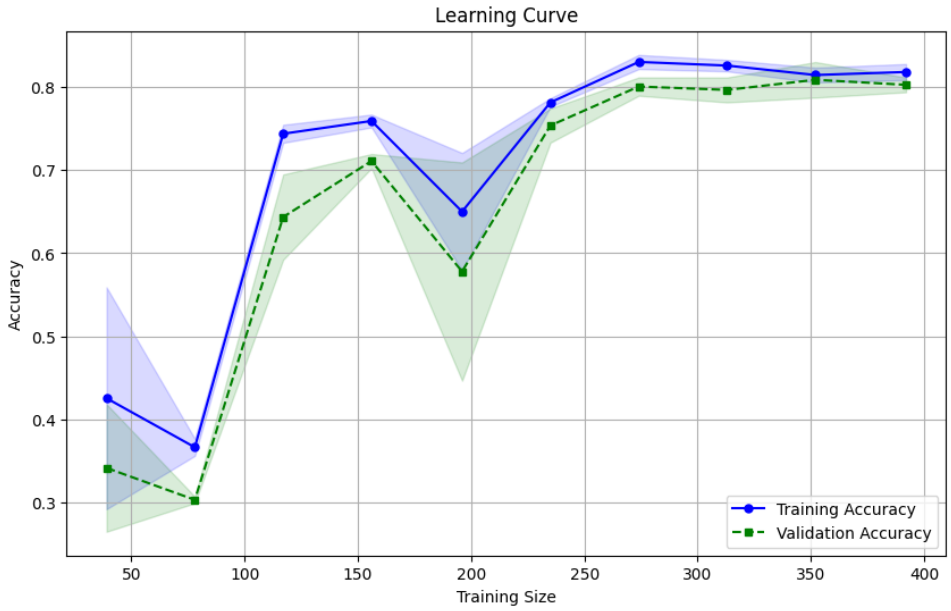
'learning_rate': ['adaptive', 'constant'], - constant learning rate remains learning_rate_init adaptive changes learning rate if training loss does not decrease.

'early_stopping': [True, False] - If True sets aside 10% of data for validation and ends training if validation score does not improve by at least the tolerance for optimization: default= 0.0001 for 10 epochs.

Of the parameters we tested the grid search returned these as the best
MLPClassifier(early_stopping=True, hidden_layer_sizes=(50, 50, 50),
learning_rate='adaptive', random_state=42,

verbose=True, learning_rate_init: 0.001)

Runing this model on our data set returned an accuracy of about 80% with this learning curve.



Conclusions

In summary, our advanced models yielded only a marginal improvement over our baseline model, with an increase in accuracy of about 2%, reaching 80%. While this level of accuracy makes the model a reasonably good tool for predicting loan approvals, it should be regarded as an aid rather than a definitive solution in the loan approval process. Both banks and customers might find it useful for expediting loan assessments, but it is important that a human expert reviews these cases, using the model's insights as a supplementary guide in the decision-making process.

Contributions

We all cleaned and implemented visualizations for the data. We all made the Naïve Bayes Model. Jonathan was the one who made the KNN model, while Jared and I implemented the NN. I made the presentation, and Jared and Jonathan made the report.

Here’s our GitHub: https://github.com/Jscott7227/CSC_5240_Project/blob/main/