Loan Predication Using Machine Learning

Nicolas Rubert

Rutgers University  
School of Engineering

Machine Learning for Engineers

ECE 445

[Nar126@rutgers.edu](mailto:Nar126@rutgers.edu)

**ABSTRACT**

Using Machine Learning, a data frame loaded with values will determine whether an applicant we get the approved for a loan. This will be done with Logistic Regression, K-Nearest Neighbors and Decision Tree algorithms in Python.

**CCS Concepts**

• **CCS ➝ Computing methodologies ➝ Machine learning ➝ Machine learning algorithms ➝ Feature selection**

**CCS ➝ Computing methodologies ➝ Machine learning ➝ Cross-validation**

**Keywords**

Machine Learning; Data Science; K-Nearest; Logistic Regression; Decision Tree; Python; NumPy; Pandas; Sklearn

# INTRODUCTION

The objective of this project is to apply Machine Learning techniques learned throughout the semester and apply it to real world application. In this project a dataset with several applicants trying to get a loan from a bank towards a new home. The system will then try to predict whether to approve the loan for the house using certain information from the applicant.

# SCOPE OF THE PROJECT

The scope of work for this project includes the following: Code that was used to complete the project, this project report and a short 10-minute video presenting the project. The goal is to complete the aforementioned deliverables in a timely manner. This will demonstrate what was learned during the fall 2021 Machine Learning for Engineering course.

# PRELIMINARY DESIGN

The design of this project starts off with the dataset. The dataset will have to be cleaned up and get ready for processing. This includes getting rid of *null* values and replacing them. This can then be analyzed and passed into a machine learning algorithm. The machine learning algorithm can then train. Once the algorithm can then test the data set for accuracy. The accuracy can be then cross validated to confirm.

# UNDERSTANDING THE DATA SET

Before applying the machine learning algorithms to predict which applicants are approved for the loan, we must first understand the data. The data will provide the necessary values needed for the system to calculate the outcome.

## Data Collection

The data was uploaded to the website, “Kaggle.com” in 2015. The person who uploaded the data goes by the username of Debdatta Chatterjee. The user did leave any explanation on how the data was collected. This may mean that the data was fabricated to emulate real data used by banking systems. The user did not specify the location where the data was collected if it was indeed not fabricated. This may affect certain variable values int the dataset i.e., Applicant’s income and what currency it is under.

Although there are certain questions that can be asked on how the collection method, the data can still be used to demonstrative the machine learning techniques learned thought this semester.

## Dataset Variables

The dataset used in the project contains the following twelve variables:

1. Loan\_ID
2. Gender
3. Married
4. Dependents
5. Self\_Employed
6. ApplicantIncome
7. CoapplicantIncome
8. Loan\_Amount
9. Loan\_Amount\_Term
10. Credit\_History
11. Property\_Area
12. Loan\_Status

Each variable in the dataset is either an integer or a string of characters. Variables (1 through 5, 11 and 12) are all variables using a string type. They are also the categorical variables. The numerical variables (6 through 9) are variables with integer type.

Loan\_ID (1) represents the applicant. Gender (2) is either “Male” or “Female”. Married (3) has values either “Married” or “Non-Married”. The Dependents (4) has four categories “0”, “1”, “2” and “3+”. Although the values represent numbers, they are indeed strings. “Urban”, “Semiurban” and “Rural” are the string values for the Property\_Area (11). Loan\_Status (12) has either “Y” or “N” for the values.

ApplicantIncome (6), CoapplicantIncome (7), Loan\_Amount (8), Loan\_Amount\_Term (9) and Credit\_History (10) have integers for their values.

## Dependent/Independent Variables

Dependent variables are defined as variables whose outcome is dependent on values from other variables. Independent variables are variables with values that are independent from other variables.

Apart from the “LoanStatus” variable, all the variables in the dataset are independent variables as they do not rely on each other.

In this dataset there is one dependent variable in the dataset, “LoanStatus”. The “LoanStatus” variable takes the values from all the other variables to determine whether or not the “LoanStatus” was approved or not.

## Pre-Processing The Data

No matter how meticulous we try to collect data three will also be cases where mistakes will be made. In this case there are certain instances where the value of a variable is *null.* This happens quite often in Data Science. Although we can choose to ignore the variables that contains *null* values for the sake of accuracy, we can fill in these *null* values with a “real-value”.

Here are all the variables along with the number of *null* values*:*

Text

Description automatically generated

For the variables: Gender (1), Married (2), Dependents (3) and Self Employed (4), the *null* value was replaced with the lowest value out of their respective variable. For an example: Gender (1) contained more Males than Females and had 13 *null* values. Those 13 *null* values where then replaced with Female to try to even the value entered. This was done to remove as much potential biases when we begin to analyze and train using the data.

Although this works great for the categorical variables this method of pre-processing can’t be used for all. To remove the *null* values in the numerical variables such as Loan Amount (8) a different approach was needed. In these instances, calculating the average value and getting the mean was used to fill in the *null* values.

**Equation 1**

Using Equation (1) all that needs to be specified is what variable i.e., Loan Amount (8) the *null* values can be replaced with an average value.

## Data Visualized

After the data is “cleaned-up” or “pre-processed” we are now able to visualize the data to understand it and prepare said data for training and testing.

The first thing we can get is the dimensions of the dataset. Using Pandas (A widely used data analysis framework for python) after reading the dataset into the program, the line “df.shape()” can print the dimension of the data which yields (614, 13).

Next, we can plot all the categorical variables using Pyplot (A framework used to visualize data).

Icon

Description automatically generated with low confidence

**Figure 1**

A picture containing chart

Description automatically generated

**Figure 2**

Chart

Description automatically generated

**Figure 3**

A picture containing icon

Description automatically generated

**Figure 4**

Chart, histogram

Description automatically generated

**Figure 5**

Chart, histogram

Description automatically generated

**Figure 6**

Chart, histogram

Description automatically generated

**Figure 7**

## Potential Biases

Because we don’t know how the data was collected there are some clear discrepancies with the data as alluded by the previous graphs. A clear example is **Figure 1. Figure 1** contains a plot with the amount of Male vs Female in the dataset. Although it should not matter in the outcome due to the value discrepancy there might be a bias more towards Male applicants versus their Female counterparts.

# FEATURE SELECTION AND ENGINNERING

To simplify the data, every value was made into binary. This means that every value that entered in the dataset was turned either a 0 or 1. Under the variable Gender (2) if a value was a “Male” the corresponding value will be turned into a 1, and 0 for “Female” respectively. This simplification method was applied to all categorical variables.

From the previous plots we can see that the data does not follow a “bell shape” due to skewness in the data frame. To address this log transformation **Equation 2** was done to all numerical variables.

**Equation 2**

Once the log transformation was done the values in the x-axis stays the same while the y becomes density. The newly transformed variables can be plotted with their corresponding values. As seen in **Figure 8** through **Figure 11** the new plot has a “bell-shape” plot. This also tries to remove as much of the skewness as possible. The plot was done using Seaborn as it provides an easier way to complete these distribution plots.

Shape

Description automatically generated

**Figure 8**

A picture containing shape

Description automatically generated

**Figure 9**

A picture containing logo

Description automatically generated

**Figure 10**

Logo

Description automatically generated

**Figure 11**

Another variable was created as well to compliment the data frame. The newly created variable is called “Total\_Income”. This variable is a dependent variable and is the sum of two original variables, ApplicantIncome (6) and CoapplicantIncome (7). The inclusion of this new variable was to determine whether the Total\_Income (13) of the applicants is a better decider in the loan approval or if the applicants individually had a better result.

Once the log transformation was done the next thing to do was drop unnecessarily variables. These variables were Loan\_ID (1), ApplicantIncome (6), CoapplicantIncome (7), LoanAmount (9), Loan\_Term\_Amount (9), Total\_Income (13) and CoapplicantIncomeLog (15). These variables were redundant as the values of the log transformation is going to be used. The LoanStatus (12) will be dropped after the correlation matrix is completed and when testing is about to take place.

# PATTERNS IN THE DATA

To see if there are any patterns in the dataset a correlation matrix can be made. It takes all the variables and makes a table which who’s the correlation of two variables. The closer the correlation the closer to the value 1 it becomes. There is a potential for negative numbers which simply means that they are not closely correlated.

Table

Description automatically generated

**Figure 12**

In the correlation matrix **Figure 8**, has all the variables placed out. Going down the matrix there are many values which includes negative and positive numbers. One prominent value is the correlation between LoanStatus (12) and CreditHistory (10). This has a correlation value of 0.54 or 54%. This means that the outcome of the LoanStatus (12) relies the heaviest on whether the applicant has an already established Credit History (10). This is also how a prominent bank practice.

# CLASSIFING ALGORITHMS

For this project three algorithms were chosen. Logistic Regression, K-Nearest Neighbors, and the Decision Tree algorithm. The variable LoanStatus (12) was dropped to try to predict the value during testing.

## Sklearn

To simplify the project, Sklearn was used (Sklearn is an opensource Machine learning framework in Python) to implement the algorithms. Sklearn offers a lot of features and includes some functions with the algorithms used. Sklearn also makes it easy to create both training and testing models. There are also inclusions for calculating the accuracy and cross validation.

## Logistic Regression

Logistic Regression is used to model binary dependent variables. This was chosen since the variables in the data frame were converted into binary values. Logistic Regression was also chosen due the value the project is trying to find. The LoanStatus (12) is a dependable variable.

## K-Nearest Neighbors

K-Nearest Neighbors or K-NN was also selected to classify the variable. In hindsight this algorithm should not have picked and instead of it should have been something like a Naïve Bayes Algorithm as it is used a lot in probability.

Although the KNN algorithm is used in classification problems which in theory works great for this project, KNN works better with larger amounts of data.

Each neighbor has a certain distance to each other. Variables with the same results end up with a closer distance. This means that loans that approved are closer together compared to loans that are denied. Variable values determine the distance and the output.

## Decision Tree Algorithm

This algorithm selection was the easiest. The decision tree looks and works kind of like a flowchart. Another way to visualize this is a bunch of if/else statements that tests on an attribute. For a simplified example using our dataset, does an applicant have an income? No, then no loan. This is done with all the variables and values. This type of algorithm is what some banks uses to determine whether a person receives a loan. For that reason, it was chosen.

## Results

**Table 1: Training Accuracy**

|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | Cross-Validation |
| Logistic Regression | **79.34 %** | **76.63 %** |
| KNN | **82.06 %** | **70.10 %** |
| Decision Tree | **100 %** | **72.28 %** |

**Table 2: Testing Accuracy**

|  |  |  |
| --- | --- | --- |
| Algorithm | Accuracy | Cross-Validation |
| Logistic Regression | **82.09%** | **81.62 %** |
| KNN | **81.86 %** | **76.74 %** |
| Decision Tree | **100 %** | **70.68 %** |

To confirm that the algorithms work we can test for accuracy using **Equation 3**. This takes the number of correct predictions divided by the total number of predictions. In the project Sklearn was also used as it can calculate the accuracy and give us a percentage.

**Equation 3**

**Table 1** contains a training model. The training model is used to “train” so when we put in real data it can give an accurate result. The accuracy in this table shows the outcome of the training. A more accurate result will be when the model is **Table 2** as it contains testing data.

In an ideal world you would like the accuracy to be 100 % correct as it means it gets the prediction right every time. To get to this more data can be used. In **Table 1 and 2** there is an error regarding the Logistic Regression Algorithm at it has an accuracy of 100%. This could be to either a problem with the way was “pre-processed” or the quality of the data. To validate the accuracy a cross validation was also ran to ensure the results. The cross validation takes random samples of the data and runs the model. This also gives a percentage for the results. The closer the cross validation and the accuracy percentages are the truer the model was.

Using the **Table 1 and 2**, the best algorithm for this task is the Logistic Regression Algorithm as it contains accuracy and cross validation values closest to each other.

# CONCLUSION

Using the techniques learned through out the semester. The application of these techniques was used to create a machine learning model that can predict if an applicant were to get approved for a loan. Using the Logistic Regression algorithm in the both the training and testing model we were able to achieve an accuracy of 82.09%! This means for every 10 applicants we can predict 8 of the applicant’s loan status!

# REFERENCES

1. Google’s Machine Learning Crash Course*.* <https://developers.google.com/machine-learning/crash-course>
2. Yates, R. and Goodman, D. 2015. *Probability and Stochastic Process Third Edition* Maryland at College Park.
3. James, G. Witten, D. Hastie, T. and Tibshirani, R. *An Introduction to Statistical Learning with Application in R Second Edition* 2021
4. Friedman, J. Hastie, T. and Tibshirani, R *The Elements of Statistical Learning*.
5. Seaborn Tutorial and API

Link: <https://seaborn.pydata.org/tutorial.html>

1. Scikit-Learn Course - Machine Learning in Python Tutorial

FreeCodeCamp link: [Scikit-Learn Course - Machine Learning in Python Tutorial](file:///Users/nico/Downloads/Scikit-Learn%20Course%20-%20Machine%20Learning%20in%20Python%20Tutorial)

1. Loan Prediction Problem Dataset

Link: [link: https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset](link:%20https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset)