International Journal of Engineering in Computer Science

E-ISSN: 2663-3590
P-ISSN: 2663-3582
IJECS 2020; 2(1): 32-37
Received: 13-11-2019
Accepted: 17-12-2019

Veeraballi Nagajyothi
Department of Computer Science, Sri Venkateswara University, Tirupati, India

Correspondence
Veeraballi Nagajyothi
Department of Computer Science, Sri Venkateswara University, Tirupati, India

Loan approval prediction using KNN, decision Tree and Naïve Bayes models

Veeraballi Nagajyothi

DOI: https://doi.org/10.33545/26633582.2020.v2.i1a.30

Abstract
In this modern world, financial institutions are playing a very crucial role. Nowadays, banks are developing their financial reserves by providing different kinds of loans to people who are in need. At the same time, there is also a massive increase in the count of individuals requesting loans. However, banks cannot provide loans for everyone as there are only limited reserves associated with each of them. So, banks must follow some stringent verification process to approve the loan, because if the one who got his/her loan approved failed to pay back his loan it may have a direct impact on the financial reserves of the bank and also onto the banking sector. So, banks started to provide loans only for a limited set of people who are capable of repaying their loans. But finding out who is eligible for the loan is a much typical and risky process. In this project, we will develop a model to predict who is eligible for a loan in order to reduce the risk associated with the decision process and to modify the typical loan approval process into a much easier one. Moreover, we will make use of previous data of loan decisions made by the company and with the help of various data mining techniques, we will develop a loan approval decision predicting model which can draw decisions for each individual based on the information provided by them. We will use a machine-learning-based KNN, Decision-tree, Naïve Bayes algorithms to train the model. This project primary goal is to develop a loan prediction model with a better accuracy rate.

Keywords: Financial, Prediction, Reserves, Repaying, Institutions

1. Introduction
Distribution of the loans is the core business part of almost every banks. The main portion the bank's assets are directly coming from the profit earned from the loans distributed by the banks. The prime objective in banking environment is to invest their assets in safe hands where it is. Today many banks/financial companies approve loan after a regress process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants. Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of features is automated by machine learning technique. The disadvantage of this model is that it emphasizes different weights to each factor but in real life sometime loan can be approved on the basis of single strong factor only, which is not possible through this system. Loan Prediction is very helpful for employee of banks as well as for the applicant also. The aim of this Paper is to provide quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. The Loan Prediction System can can automatically calculate the weight of each features taking part in loan processing and on new test data same features are processed with respect to their associated weight. A time limit can be set for the applicant to check whether his/her loan can be sanctioned or not. Loan Prediction System allows jumping to specific application so that it can be check on priority basis. This Paper is exclusively for the managing authority of Bank/finance company, whole process of prediction is done privately no stakeholders would be able to alter the processing. Result against particular Loan Id can be send to various department of banks so that they can take appropriate action on application. This helps all others department to carried out other formalities.

2. Literature Survey
In paper [1] researchers used various machine learning algorithms to develop a machine learning model. They designed an individual model for every machine learning algorithm used in developing a protection model. They made use of different machine algorithms like

decision tree algorithm, Bayes algorithm, random forest algorithm. Among all the models the model built with the decision tree algorithm has scored the highest accuracy rate of 81%. It is followed by random forest algorithm with an accuracy rate of 77% and the Bayes algorithm with an accuracy rate of 69%.

In paper [2] researchers used k means clustering technique for developing a loan prediction model based on risk percentage. This model works on the principle of risk associated with each loan applicant if the model predicted that the applicant is having a low-risk percentage which means that he/she can pay back the loan, his/her loan will be approved. By using K means clustering they divided all the observation in the data into 5 clusters where each cluster is associated with some risk percentage. By using this model, they have scored an accuracy rate of 84.56%.

In paper [3] researchers used different machine learning algorithms like KNN, Decision tee, Random forest, SVM for building a model for accuracy predictor loan risk. This model deals with predicting the risk associated with each applicant which means the possible chance of loan repayment by a customer. In this paper [3] they used R language to predict the accuracy of several models. Here random forest has scored the highest possible accuracy of 82.64% in run 3 and it is followed by an accuracy rate of 81.94% using SVM in the run 3.

3. Proposed Work

The training data set is now supplied to machine learning model, on the basis of this data set the model is trained. Every new applicant detail filled at the time of application form acts as a test data set. After the operation of testing, model predict whether the new applicant is a fit case for approval of the loan or not based upon the inference it concludes on the basis of the training data sets.

3.1 K-Nearest Neighbor Algorithm

An approach of classifying the data which would be used in estimating the likelihood of a data point in being a member of one group or the other based upon the nearest available group of data points can be described as a k nearest neighbor algorithm, which is often called as KNN algorithm.

KNN algorithm usually will not construct a model until a query is imposed on the dataset, which makes K-nearest neighbor a predominant example of a "lazy learning" algorithm.

In knn algorithm, if we need to determine whether a point will come under either group A or B, the algorithm will look at the nearest data points and the group does they belong to. If we consider a sample of data, the range is randomly determined. in case, if the majority of the points belong to group B, in such instance the data point will be having the most likelihood of being a member of group B and vice versa.

Dataset which is used for the implementation of prediction of process is extracted from Kaggle website. This data set is used to train the model and to make predictions by splitting it into training data and testing data using various modules. The dimensions of the dataset are:

Total number of Applicants: 615
Total number of Fields: 13

A1 | Loan_ID

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Loan_ID	Gender	Married	Dependen	Education	Self_Empl	ApplicantI	Coapplica	LoanAmou	Loan_Amo	Credit_His	Property_	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
8	LP001013	Male	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurbar	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurbar	N
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y
13	LP001027	Male	Yes	2	Graduate		2500	1840	109	360	1	Urban	Y
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y
18	LP001034	Male	No	1	Not Gradu	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N
20	LP001038	Male	Yes	0	Not Gradu	No	4887	0	133	360	1	Rural	N
21	LP001041	Male	Yes	0	Graduate		2600	3500	115		1	Urban	Y
22	LP001043	Male	Yes	0	Not Gradu	No	7660	0	104	360	0	Urban	N
23	LP001046	Male	Yes	1	Graduate	No	5955	5625	315	360	1	Urban	Y
24	LP001047	Male	Yes	0	Not Gradu	No	2600	1911	116	360	0	Semiurbar	N
25	LP001050		Yes	2	Not Gradu	No	3365	1917	112	360	0	Rural	N
26	LP001052	Male	Yes	1	Graduate		3717	2925	151	360		Semiurbar	N
27	LP001066	Male	Yes	0	Graduate	Yes	9560	0	191	360	1	Semiurbar	Y

train_LoanPrediction

Fig 1: Dataset used for training the model we need various libraries to be imported for designing a code that performs training and testing tasks, Predictions, array operations etc. they are as follows:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

Fig 2: Libraries used in Code

We must load the training data and then we must search for blank values as they will lead to uncertainties in the predictions.

```
train = pd.read_csv('train_LoanPrediction.csv')
print(train.isnull().sum())
```

Fig 3: Finding no. of blank fields

After finding about No. of blank fields present in the dataset then we must replace them with values which are derived by statistical methods such as mean, mode, mean for both numerical and categorical attributes present in the dataset and must check for null values to make sure that there are no blank fields in the dataset. We can also replace the irrelevant or noisy data with the precise ones so that it will not show any impact on the training process and to make predictions.

```
train['Dependents'].replace('3+', 3,inplace=True)
test['Dependents'].replace('3+', 3,inplace=True)
# Train Categorical Variables Missisng values
train['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
train ['Married'].fillna(train['Married'].mode()[0],inplace=True)
train['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
train['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
# Train Numerical Variables Missing Values
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)
train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
```

Fig 4: removing Noisy data and Blank fields

After cleaning the data, we must search for outliers present in the data and we must remove them because outliers will lead to the faulty predictions or biased predictions. After removing the outliers from the data we must divide the data into independent and dependent variables which means we must split first 12 attributes variables into one group of array elements and the final status attribute variables into other as they are dependent on the other attributes of the dataset.

```
# Outlier treatment
train['LoanAmount'] = np.log(train['LoanAmount'])
test['LoanAmount'] = np.log(test['LoanAmount'])

# Separting the Variable into Independent and Dependent
X = train.iloc[:, 1:-1].values
y = train.iloc[:, -1].values
```

Fig 5: outlier treatment and splitting variables

After splitting the variables into two groups then we must transform all the categorical data variables into the machine understandable format. So that we will convert them into some dummy variables. Here we will use LabelEncoder( ), OneHotEncoder( ), fitTransform( ) functions for transformation.

```
# Converting Categorical variables into dummy
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
labelencoder_X = LabelEncoder()

# Gender
X[:,0] = labelencoder_X.fit_transform(X[:,0])

# Marraige
X[:,1] = labelencoder_X.fit_transform(X[:,1])

# Education
X[:,3] = labelencoder_X.fit_transform(X[:,3])

# Self Employed
X[:,4] = labelencoder_X.fit_transform(X[:,4])

# Property Area
X[:,-1] = labelencoder_X.fit_transform(X[:,-1])

# Dummy Varaibles
onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()
```

Fig 6: Code for conversion of Categorical variables to machine understandable ones

After converting all the categorical data into dummy variables and loading it into again the same variable 'X', we must split both the data variables 'X' and 'Y' into train and test data using train_test_split module available from scikit-learn. Thereafter we must fit the split data using StandardScaler. Following that we must use KNeighborsClassifier module for the data classification. Finally, we must use Accuracy_Score module to calculate the accuracy score for the prediction made by the model.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.45, random_state = 6)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import accuracy_score
m=accuracy_score(y_test,y_pred)
```

Fig 7: Code for splitting and K-NN Classification

4. Results and discussions

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Fig 8: Blank Fields in Train dataset before data pre-processing

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

Fig 9: Training dataset with no blank fields after data pre-processing

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

Fig 10: Blank Fields in testing dataset before data pre-processing

```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
dtype: int64
```

Fig 11: Testing dataset with no blank fields after data pre-processing

```
Accuracy score of the prediction: 0.8339350180505415
Final Accuracy rate of the model: 83.39350180505414
```

Fig 12: Accuracy rate of the KNN model

```
Accuracy rate for decision tree algorithm: 0.7256317689530686
Final Accuracy rate of the decision tree model: 72.56317689530685
```

Fig 13: Accuracy rate of decision tree model

```
Accuracy rate for Naive bayes algorithm: 0.8303249097472925
Final Accuracy rate of the naive bayes model: 83.03249097472924
```

Fig 14: Accuracy rate of Naïve Bayes model

```
prediction for Test Dataset:
['Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'N' 'N' 'N' 'Y' 'N' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'N' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'N' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y'
 'Y' 'Y' 'Y' 'Y' 'Y' 'Y' 'Y']
```

Fig 15: Loan Approval Predictions made for Test Dataset

The trained model had scored an accuracy rates of:
Accuracy = 0.8339350180505414 for KNN model
Accuracy = 0.7256317689530686 for Decision-Tree model
Accuracy = 0.8303249097472925 for Naïve Bayes model

Which means that around 83.39 predictions made by the KNN model, 72.56 predictions made by Decision-Tree model, 83.03 predictions made by Naïve Byes model are correct when compared to original loan approval decisions.

5. Conclusion

Cleaning the data, processing the data, imputing missing values, exploratory analysis on data, model construction and its evaluation are the basic steps followed in the analytical process. 0.834 is the greatest attainable accuracy by using the KNN model with the given public dataset. Some of the following insights are developed about loan approval by examining the results. Applicants with a '0' credit history failed to get their loans approved, as '0' credit history very less liability of the applicant which means he's/she's having a less chance of repaying the loan. Moreover, people with few dependents and more salary followed by less amount of loan requested got their loan sanctioned. Details such as the gender of the applicant and education of the applicant were considered as the least important attributes.

6. References

1. Cowell RG AP, Lauritez SL DJ. Graphical models and Expert Systems. Berlin: Springer. This is a good introduction to probabilistic graphical models, 1999.
2. Kumar Arun, Garg Ishan, Kaur Sanmeet. MayJun. Loan Approval Prediction based on Machine Learning Approach, IOSR Journal of Computer Engineering (IOSR-JCE), 2016.
3. Wei Li, Shuai Ding, Yi Chen, Shanlin Yang. Heterogeneous Ensemble for Default Prediction of Peer-to-Peer Lending in China, Key Laboratory of Process Optimization and Intelligent DecisionMaking, Ministry of Education, Hefei University of Technology, Hefei 23009, China, 2016.
4. Clustering Loan Applicants based on Risk Percentage using K-Means Clustering Techniques, Dr. K. Kavitha, International Journal of Advanced Research in Computer Science and Software Engineering.
5. Research on bank credit default prediction based on data mining algorithm, The International Journal of Social Sciences and Humanities Invention. 2018; 5(06):4820-4823,
6. Short-term prediction of Mortgage default using ensembled machine learning models, Jesse C. Sealand on july 20, 2018.