A Comparative Study to Bank Loan Prediction

Pritam Sarkar, Pratyush Sarkar

(B.Tech , CSE , Academy of Technology,2015-19)(B.Tech , CSE , Academy of Technology,2015-19)

Under the guidance of Dr. Soumadip Ghosh

HOD of Information Technology, Academy of Technology, Aedconagar, Hoogly-712121, West Bengal, India

Abstract

Bank loan prediction is one of the important applications of machine learning field. In modern days, different bank organizations generate huge amount of data. And the huge amount of data is used in analytical purpose. The data is fed to different efficient machine learning algorithms to generate meaningful information as output. Prediction ability of those algorithms are used to improve decision making capability in different sectors of banks. This effectively reduces the time consumption and corruption and communication complexity problem in decision making due to the involvement of active human resources. In Loan prediction case one can easily check whether he or she is capable of getting the loan using some proper web-based user-interfaces, sitting in home without running to different authority holders of different banks. In our case we implemented a generalized loan prediction algorithm which can be used as an effective framework for Loan predictions where more attributes and larger dataset are used. We used several machine learning algorithms with the help of different available libraries like pandas, matplotlib, scikit-learn etc. The objective is to examine the performance of the above stated techniques on a real-world data of bank loan record dataset. We trained the models with desired accuracies shown in ROC curve later. Then predictions are made based on the trained models which shows whether one can get the loan or not.

INTRODUCTION

Banks maintain huge amount of data related to their customers. This data is greatly used to create and keep clear-cut relationship and connection with the customers in order to target them individually for different banking offers. Data mining has gained popularity due to its different illustrative and predictive applications in banking processes which are very much efficient and effective in nature. In our project we have applied seven different machine learning algorithms.

A Brief Overview: A loan prediction practice dataset has been collected first. After collecting the dataset, data cleaning is thoroughly done on the dataset in the following ways. Incomplete or missing data in any row that contains numerical values, is filled using the median value (median is better than mean in case of noise avoidance) of all the data of that particular attribute. If the data are categorical, the blank spaces are filled based on the frequency of all the data in that particular attribute (the one having the highest frequency is used to fill). After the data cleaning is done, data transformation is performed. Data transformation is the process of converting data from one format or structure into another format or structure for computational need. Example: - if we have 3 String or character type data to a particular feature. We convert them into 3 integers [0,1,2]. Because computation can only be performed on numerical type data. Attributes that are not important (like row number) are not considered during any calculation (Feature Selection) it reduces the overfitting problem. After feature selection we perform standardization to minimize the data range variations among different features. After standardization is done, the entire dataset was split into two parts Training dataset (70 to 80% of total data) and Testing dataset (remaining 30 to 20% of data) after being shuffled. We did not use another validation dataset as our dataset was lesser in size. Another method is Cross-validation which is a statistical method used to estimate the skill of machine learning models. Then we trained different machine learning models built on MLPNN, ADA, RF, KNN, LR, DT and SVM until we got desired accuracy using threads in python. Those models latter successfully predicted the sanction of loan.

LITERATURE OVERVIEW

The discussions in the above sections makes it clear that the need for loan prediction is a major requirement in the banking sector. Lots of research work regarding this area are being done all over the world. Here we have presented some of the related works in this area done by other researchers. Harish Puvvada and Vamsi Mohan Ramineedi [5] have worked on loan prediction models using svm, K-Nearest Neighbours, MLP, Adaboost

techniques. Vimala S., Sharmili K.C. [6] had presented a paper in the ICACT 2018 in the same area of research where they used Naïve Bayes and SVM techniques. “Bank Loan Default Prediction with Machine Learning” by Hongri Jia [7] on Apr 10,2018. Logistic Regression, Random Forest, XG Boosting was used for loan prediction. “Loan Approval Prediction based on Machine Learning Approach” by Kumar Arun, Garg Ishan, Kaur Sanmeet [8]. Decision Trees, Random Forest, Support Vector Machine, Ada boost, Neural network was used for loan prediction. “An Exploratory Data Analysis for Loan Prediction Based on Nature of the Clients” by X.Francis Jency, V.P.Sumathi, Janani Shiva Sri [9] different data analytics was used.

DATASET DESCRIPTION [10]

Data given consists of training and testing sets. It consists of 13 columns with Loan_Status as our target. (As usual, data contains some missing values). It has 1224 columns.

Serial No.	Attribute Name	Type	Description
1	Loan_ID	Typeless	Unique Loan ID
2	Property_Area	Ordinal	Location of property (Rural/Semi Urban/Urban)
3	Gender	Flag	Male/ Female
4	Married	Flag	Applicant married (Y/N)
5	Self_Employed	Flag	Self-employed (Y/N)
6	Education	Flag	Applicant Education (Graduate/ Not Graduate)
7	Dependents	Ordinal	Number of dependents
8	ApplicantIncome	Cont	Applicant income (not sure unit)
9	CoapplicantIncome	Cont	Coapplicant income (not sure unit)
10	Credit_History	Flag	credit history meets guidelines
11	LoanAmount	Cont	Loan amount in thousands
12	Loan_Amount_Term	Ordinal	Loan amount term in months
13	Loan_Status	Flag	Loan Approved (Y/N)

Fig 1- Dataset Description

The target attribute of the dataset contains of 2 classes 0 and 1, but they are not equal in numbers so the models could get biased to a particular target class. Unfortunately, we got a low sized dataset and could not perform the **data balancing** step. But anyone can perform the step when they got a dataset of proper shape, based on our framework. Data Balancing is a very important step in case of deep neural networks. It is important for this case also. After training all the models we stored them for further use in prediction purpose.

WORKING OVERVIEW

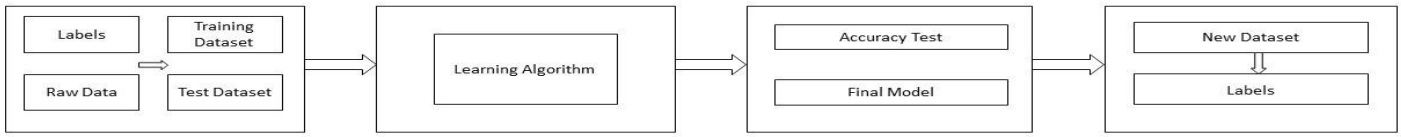


Fig 2- General Workflow

We used six classification models of supervised machine learning stated in the above introduction section. We simultaneously trained all the models using the concept of threading. Detailed explanation is given in methodology section. Decision tree is a weak classifier so we enhanced its classification ability using Adaptive Booster classifier. We don’t have to bother about the detailed algorithm of different classifiers, there are lots of inbuild functions methods classifiers defined in different libraries in Python. So, we did not implement various algorithms from scratch. We used Anaconda3 and Jupyter notebook for better flexibility. Most of the required libraries are available in Anaconda.

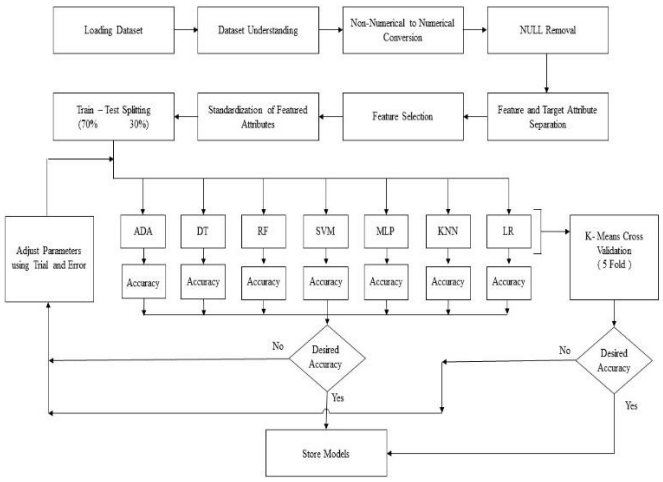


Fig 3- Detailed workflow for storing models

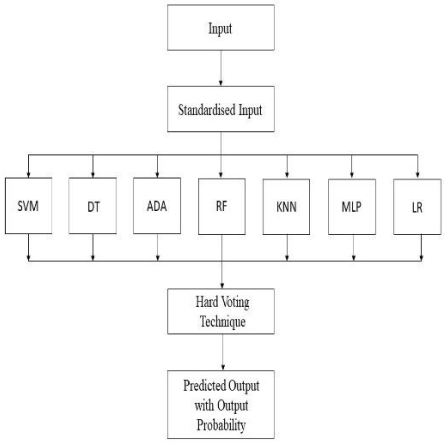


Fig 4- Detailed workflow for prediction

METHODOLOGY [1][2][3][4][13]

We downloaded the raw dataset from internet in .csv format then we load the data form .csv to a dataframe in Python using a method called read_csv ('name of the .csv file including path'). Which is available in pandas library.

Analyzing Dataset: We have 614 rows and 13 attributes, those are: Loan_ID, Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome, CoapplicantIncome ,LoanAmount ,Loan_Amount_Term ,Credit_History ,Property_Area, Loan_Status among them we didn't used the Loan_ID attribute because it is primary key, mixture of characters and numbers, serial in nature. When we randomly draw test train partition, all the data got suffered, so no need of serial primary key, also in some references they didn't choose the primary serial attribute for training the prediction model. Loan_Status was our target attribute, and based on remaining 12 attributes we trained the models.

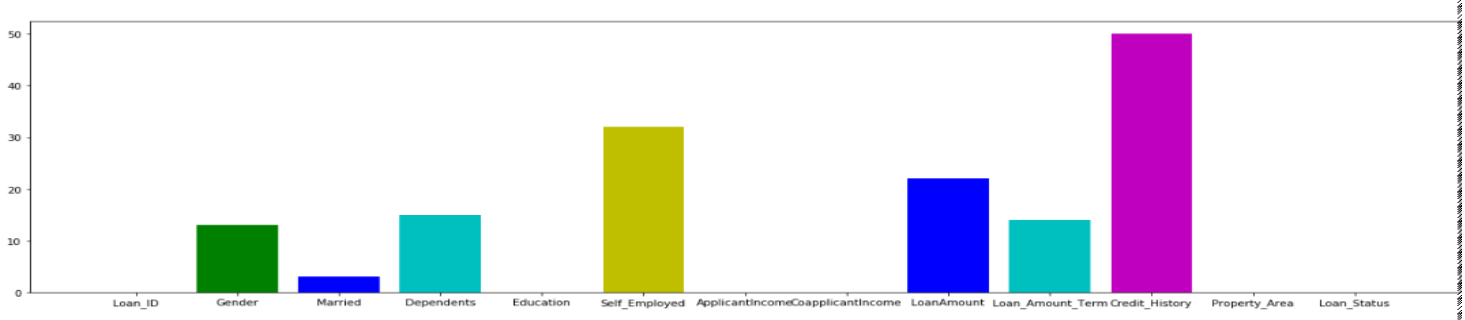
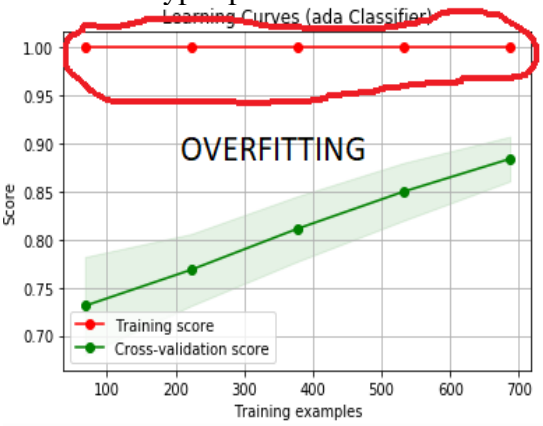


Fig 5- Histogram of null value presence in each attribute

Non-numeric to Numeric Data Conversion and Storing and Cleaning: At the time of model training or prediction the system can't understand or process the nonnumeric data, so this is an obvious process. We applied the replace(['from'] ['to'],'permanent or non-permanent changes') method to some attributes of our data frame. Property_Area had 3 classes Urban, Semi urban and Rural, was replaced by 2, 1 and 0. Married had 2 classes yes and no, was replaced by 1 and 0. Gender had 2 classes Male and Female, was replaced by 1 and 0.Education had 2 classes Graduate and Not Graduate, was replaced by 1 and 0.Self_Employed had 2 classes Yes and No, was replaced by 1 and 0.Dependents had 4 classes '0' '1' '2' '3+', was replaced by 0 1 2 3 . Loan_Status had 2 classes Y and N, was replaced by 1 and 0. Then we checked each attribute and replaced all null values by the median values (not considering the NaN values) of each attribute. Then We used the 1st 11 attributes (without the leftmost loan ID) as Featured attributes. And last one as Target attribute.

Feature Selection for avoiding overfitting [11][12]**:** We used the Feature selection method to avoid Overfitting problem due to too much Features. From sikitlearn.feature_selection we imported REF and from sikitlearn.tree we imported DecisionTreeClassifier . We used the following parameters in Decision Tree Classifier: Hyper parameter: **criterion='entropy'**, **random_state=5**



←Fig 6- Overfitting

and created dt object.We used the following parameters in REF: Classifier=dt, no of top features =8 and created rfe object. Then we fitted the X and **y** into the ref and got the ranks against each feature and we selected the rank one features {Dependents, Self_Employed, ApplicantIncome ,CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History, Property_Area} Then we used the fit_tranform method of rfe and created a new X containing less and effective features using the old **X** and **y.** Entropy method formula**:** $I(T) = -(\frac{P}{(P+E)}\ \log_2\left(\frac{P}{P+E}\right) + \frac{E}{(P+E)}\log_2(\frac{E}{P+E}))$ **I(T)** is the information gain of the target attribute, **P** is the number of zeroes and E is the number of ones in Target attribute. $E(A_C) = I(A_C) \times P(C)$. $E(A_C)$ is entropy of one class of an attribute, $I(A_c)$ is the information gain by the class w.r.t the target attribute classes in the attribute **P(C)** is probability of the class w.r.t the target attribute classes in the attribute. Table of each attribute with respect to the Target attribute is needed $E(A) = \sum E(A_C)$. **E(A)** is the total entropy of an attribute**,** $\sum E(A_C)$ summation of entropies of each class in an attribute**.** $G(A) = I(T) - E(A)$. **G(A)** is the gain of that attribute, this way we found out gains of all the remaining attributes, the more gain the more important the feature is. This way we selected top most 8 gained attributes.

Scaling the data: [14] There are two very popular of scaling methods one standard scaling where the goal is to make the standard deviation 1 and mean 0 of all data of featured attributes, to make the model predict more accurately. Standard_deviation= $\sqrt{(\sum(x - x')^2/n)}$, Mean= $\sum x\ /n$, **X_std=(X-X_Mean)/X_standard_deviation**. The second one is Min Max Scaling. Formula for that=$\frac{X-X_{min}}{X_{max}-X_{min}}$ **X** is an array of values of each attribute, **Xmin** and **Xmax** are the maximum and minimum values of the array. We used the second one (Tried both, gave nearly same results) so that probability of occurrence of each value became random. Before standardization all the features had values of different range some feature might have range of 15,000 to 20,000 and some feature might have range of 0 to 10. This creates a huge biasing problem. The learning algorithm get more biased to those features which have higher data ranges. So, we used the technique for resolving the biasness problem. It converted all the high variant feature ranges into low variant feature range of 0 to 1. **Train Test Split:** We used train_test_split() mothod of model_selection library from sikit learn (most important library for statistical analysis). Sklearn.model_selection.train_test_split() takes X_std as input and split them based on the following parameters **test_size=20% test_size=80% random_state=none shuffle**=true **stratify=None**. The whole dataset was shuffled, and the dataset was split in non-satisfying fashion with no duplicate values in between two splits, All the parameters are trial error based.

DT [13][15][16][17] **and ADA** [13][18][19][32] **and RF** [13][20][21][22]**:** From sklearn.tree library we got DecisionTree Classifier. Hyper parameters: **criterion='gini', splitter='best', random_state=5, max_depth=5** (max_depth reduces overfitting) gini is the function to measure the quality of a split. Supported splitting criterias are "gini" for the Gini impurity and "entropy" for the information gain. We used the Gini impurity. Formula=$\mathbf{1} - \sum_{i'=1}^{n} P^2 i'$ **P** is probability of finding any class. Actual formula is explained with an example: dataset D (X, Y, Z)**.** X [1, 2, 1, …,2], Y [3, 4, 4, …,3], Z [0, 1, 0, …,0] Z Target attribute and X, Y feature attribute**.** Z has 2 classes 0 &1. X has 2 classes 1 &2 N_{Z_1} = Number of 1s in Z. N_{Z_2} = Number of 0s in Z**.** Splits in X= {Φ, {1}, {2}, {1,2}}. $G(X)_{\Phi} = \frac{N_{X_\Phi}}{N}\left(1 - \left(\frac{N_{X_{\Phi Z_1}}}{N_{X_\Phi}}\right)^2 - \left(\frac{N_{X_{\Phi Z_0}}}{N_{X_\Phi}}\right)^2\right) + \frac{N_{X_{1,2}}}{N}\left(1 - \left(\frac{N_{X_{1,2Z_1}}}{N_{X_{1,2}}}\right)^2 - \left(\frac{N_{X_{1,2Z_0}}}{N_{X_{1,2}}}\right)^2\right)$ **&** $G(X)_{1,2} = G(X)_{\Phi}$ **&** N_{X_Φ} = number of Φ in X. $N_{X_{1,2}}$ = number of 1,2 in X. $N_{X_{\Phi Z_1}}$ = number of Φ in X when Z=1 $N_{X_{\Phi Z_0}}$ = number of Φ in X when Z=0. $N_{X_{1,2Z_1}}$ = number of 1,2 in X when Z=1. $N_{X_{1,2Z_0}}$ = number of 1,2 in X

when Z=0. $G(X)_1 = \frac{N_{X_1}}{N}\left(1 - \left(\frac{N_{X_1Z_1}}{N_{X_1}}\right)^2 - \left(\frac{N_{X_1Z_0}}{N_{X_1}}\right)^2\right) + \frac{N_{X_2}}{N}\left(1 - \left(\frac{N_{X_2Z_1}}{N_{X_2}}\right)^2 - \left(\frac{N_{X_2Z_0}}{N_{X_2}}\right)^2\right)$ **&** $\boldsymbol{G(X)_2 = G(X)_1}$ N_{X_1} = number of 1 in X. N_{X_2} = number of 2 in X. $N_{X_1Z_1}$ = number of 1 in X when Z=1. $N_{X_1Z_0}$ = number of 1 in X when Z=0. $N_{X_2Z_1}$ = number of 2 in X when Z=1. $N_{X_2Z_0}$ = number of 2 in X when Z=0. $G(X) = \min(G(X)_{\Phi}, G(X)_{1,2}, G(X)_1, G(X)_2)$. Splits in Y= {Φ, {3}, {4}, {3,4}} $\boldsymbol{G(Y)_{\Phi}} = \frac{N_{Y_{\Phi}}}{N}\left(1 - \left(\frac{N_{Y_{\Phi}Z_1}}{N_{Y_{\Phi}}}\right)^2 - \left(\frac{N_{Y_{\Phi}Z_0}}{N_{Y_{\Phi}}}\right)^2\right) + \frac{N_{Y_{3,4}}}{N}\left(1 - \left(\frac{N_{Y_{3,4}Z_1}}{N_{Y_{3,4}}}\right)^2 - \left(\frac{N_{Y_{3,4}Z_0}}{N_{Y_{3,4}}}\right)^2\right)$ **&** $\boldsymbol{G(Y)_{3,4} = G(Y)_{\Phi}}$ **&** $N_{Y_{\Phi}}$ = number of Φ in Y. $N_{Y_{3,4}}$ = number of 3,4 in Y. $N_{Y_{\Phi Z_1}}$ = number of Φ in Y when Z=1. $N_{Y_{\Phi Z_0}}$ = number of Φ in Y when Z=0. $N_{Y_{3,4Z_1}}$ = number of 3,4 in Y when Z=1. $N_{Y_{3,4Z_0}}$ = number of 3,4 in Y when Z=0. $G(Y)_3 = \frac{N_{Y_3}}{N}\left(1 - \left(\frac{N_{Y_3Z_1}}{N_{Y_3}}\right)^2 - \left(\frac{N_{Y_3Z_0}}{N_{Y_3}}\right)^2\right) + \frac{N_{Y_4}}{N}\left(1 - \left(\frac{N_{Y_4Z_1}}{N_{Y_4}}\right)^2 - \left(\frac{N_{Y_4Z_0}}{N_{Y_4}}\right)^2\right)$ **&** $\boldsymbol{G(Y)_4 = G(Y)_3}$ N_{Y_3} = number of 3 in Y. N_{Y_4} = number of 4 in Y. $N_{Y_3Z_1}$ = number of 3 in Y when Z=1. $N_{Y_3Z_0}$ = number of 3 in Y when Z=0. $N_{Y_4Z_1}$ = number of 4 in Y when Z=1. $N_{Y_4Z_0}$ = number of 4 in Y when Z=0. $\boldsymbol{G(Y) = \min(G(Y)_{\Phi}, G(Y)_{3,4}, G(Y)_3, G(Y)_4) => \Delta G(X) = G(D) - G(X) => \Delta G(Y) = G(D) - G(Y)}$. G is the Gini impurity of an attribute. The more the ΔG(..) the more important the attribute is. The attribute with max ΔG(..) will be the root node of the tree. Maximum value of gini is 0.5: when records are equally distributed among all the classes. Minimum value of gini is 0.0 when all records belongs to one class, implying most interesting information. Random_state is the random number generator with respect to a seed, so the generated random numbers are not completely random in nature as their generation depends on that seed. We got an accuracy of 81.57% on test dataset and got 85.78%, 82.52%, 80.48%, 77.23%, 77.86% accuracies in 5 cross validations. We got adabooster from the ensemble of sikit learn library. It is a Boosting Ensemble method. Adaptive boosting classifier learns from errors of its weak base estimator classifier by running them multiple times. A Weight is applird to every example in the training data, initially these weights are all equal, A weak classifier is first trained on trained data. The errors from the weak classifiers are calculated, repeat the process with the same classifiers, here, **base_estimator=dt,n_estimators=100,learning_rate=0.01,random_state=5,algorithm='SAMME.R'**. It repeats that process 100 times, but the weights of the classifiers are adjusted again and again so that the data can properly be classified. The final output was a sum or collection of all classification boundaries. It repeats the training and weight adjusting iterations until the training error become 0 or when the number of classifiers reaches to a defined value. We got an accuracy of 89.15% on test dataset and got 90.24%, 89.83%, 93.90%, 90.24%, 90.98% accuracies in 5 cross validations. We got RandomForestClassifier from ensemble of sikitlearn library. This is a bagging ensemble method. A sequential ensemble learning. Parameters are: **n_estimators=1000, criterion='gini',random_state=5,max_depth=5**. We got an accuracy of 82.11% on test dataset and got 82.92%, 82.52%, 85.77%, 76.82%, 78.27% accuracies in 5 cross validations.

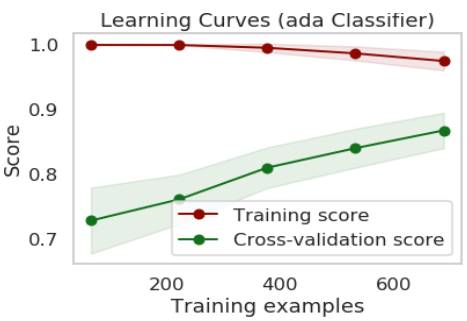


Fig 7- Learning curve of ADA

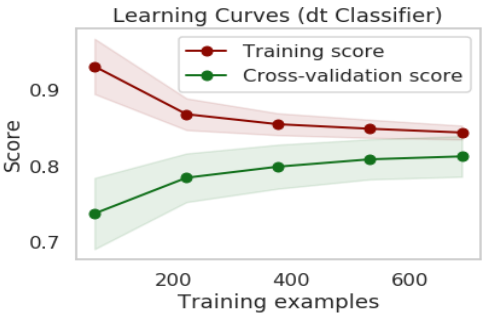


Fig 8- Learning curve of DT

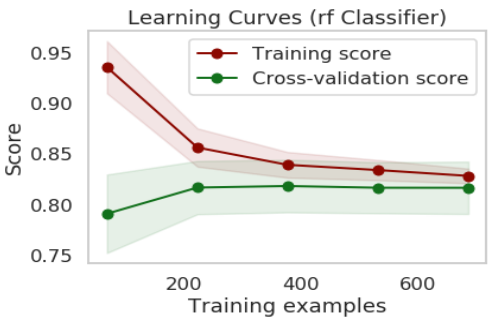


Fig9- Learning curve of RF

MLPNN [13][23][24][25] **& SVM** [13][26][27][28] **& KNN:** [13][29][30][31] From neural_network library of sikit learn we got MLPClassifier Hyper parameters: **hidden_layer_sizes= (5,5,5,5),activation='tanh', max_iter=500, solver='adam', batch_size=100,random_state=5,learning_rate='invscaling'**. Hidden layer size = (number of neurons per layer)**.** basically, there are 2 formulas for estimating number of

neurons per hidden layer. **One:**$\sqrt{(x*y)}$ **Two:**$(x*y)*2/3$, **x**=number of attributes used for finding target value here 11 & y=number of classes in the output here **2.** So, output become nearly 5 based on the one formula**.**We used the number one formula. Activation: logistic. The tanh function **tanh(x)=**$\left(\frac{2}{1+e^{-2x}}\right)-1$ **.** Ranges the values between -1 to 1. We can also use relu activation function. We introduced activation function for implementing nonlinearity in the linear equation of MLPNN otherwise using multiple layers become meaningless as multiple layers using linear equations are equivalent to a single layer of artificial neurons. **'adam'** works on by considering momentum and time instances so it is most effective global minima finder. The 'adam' the default solver on MLPclassifier performs well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score. For small datasets, however, 'lbfgs' can converge faster and perform better than 'adam'. It is a feed forward network. Adam solves the gradient descent problem most accurately with back propagation. The equations (without complete backpropagation equations) for a single layer are given below. **Cost=(Difference)^2, Difference=(Prediction), Prediction=sigmoid(z)**
$\boldsymbol{z=\sum_{i=1}^{n} wipi+b}$ W= weights b=bias (added for more accuracy) p= different training attribute values
$\boldsymbol{Wj(i+1)=Wj(i)-\alpha\Delta cost}$ & $\boldsymbol{bi+1=bi-\alpha\Delta cost}$ α is the learning rate Δcost is the partial differentiation wrt w1, w2, w3..wr and b $\frac{\partial cost}{\partial difference}=2(\boldsymbol{difference})$ & $\frac{\partial difference}{\partial prediction}=1$ & $\frac{\partial prediction}{\partial z}=\boldsymbol{d(sigmoid(z))dz}$ then $\frac{\partial z}{\partial w1}=\boldsymbol{p1}$ $\frac{\partial z}{\partial w2}=\boldsymbol{p2}$.....$\frac{\partial z}{\partial wn}=\boldsymbol{pn}$ & $\frac{\partial z}{\partial b}=1$ then $\frac{\partial cost}{\partial w1}=\frac{\partial cost}{\partial difference}*\frac{\partial difference}{\partial prediction}*\frac{\partial prediction}{\partial z}*\frac{\partial z}{\partial w1}$ then $\frac{\partial cost}{\partial w2}=\frac{\partial cost}{\partial difference}*\frac{\partial difference}{\partial prediction}*\frac{\partial prediction}{\partial z}*\frac{\partial z}{\partial w2}$ ..... $\frac{\partial cost}{\partial wn}=\frac{\partial cost}{\partial difference}*\frac{\partial difference}{\partial prediction}*\frac{\partial prediction}{\partial z}*\frac{\partial z}{\partial wn}$ & $\frac{\partial cost}{\partial b}=\frac{\partial cost}{\partial difference}*\frac{\partial difference}{\partial prediction}*\frac{\partial prediction}{\partial z}*\frac{\partial z}{\partial b}$
Adams main goal is to minimize the cost function effectively in total 500 iterations (max_iter=500 learning_rate='invscaling') Tol=Tolerance for the optimization. When the loss or score is not improving by at least tol for n_iter_no_change consecutive iterations, unless learning_rate is set to 'adaptive', convergence is considered to be reached and training stops. Then we train the classifier by fitting X_train_std and y_train in it. We got an accuracy of 69.91% on test dataset and got 83.33%, 82.52%, 85.77%, 76.42%, 77.86% accuracies in 5 cross validations. As ratio of 1 to 0 is very much high in this dataset the MLPNN became a little biased. From sikit-learn library we imported the svm one of its method is SVC support vector machine classifier. Hyper parameters: **gamma=1.1, C=100, kernel='rbf'.** Gamma is the learning rate of the kernel of SVC here by default rbf (creates overfitting on smaller datasets). C is the penalty parameter of the error term Rbf stands for redial basis function, our trial and error-based observation chooses the rbf kernel among linear, poly, rbf, sigmoid, precomputed kernels, because it is giving maximum accuracy with desired result. Kernel functions takes lowdimensional feature space as input and gives High dimensional feature space as output. All the remaining parameters are trial error based also. We got an accuracy of 90.78% on test dataset and got 84.55%, 84.55%, 86.99%, 78.45%, 79.50% accuracies in 5 cross validations. From sklearn.neigbbors we got KNeighborsClassifier. We need to find the Euclidean distance between each entries of our dataset(featured) and the provided input. Then according to our algorithm, we need to choose nearest K (provided using parameters) values and its corresponding data entries. The target attribute of those nearest entries decides the output of our given input using probability. We choose the max probable ans.Hyper parameters are: **n_neighbors=5, algorithm='auto',leaf_size=30.** Means we considered the 5 nearest entries of our featured dataset. We got an accuracy of 71.00% on test dataset and got 85.36%, 84.55%, 84.95%, 77.23%, 81.14% accuracies in 5 cross validations. As ratio of 1 to 0 is very much high in this dataset the KNN became a little biased.

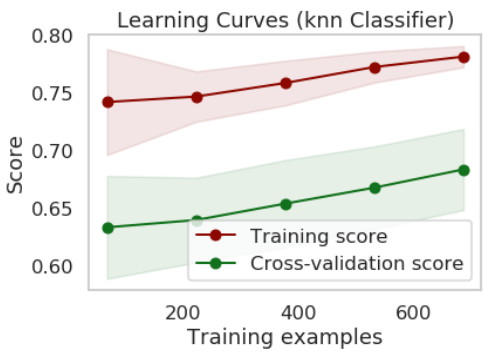


Fig 10- Learning Curve of KNN

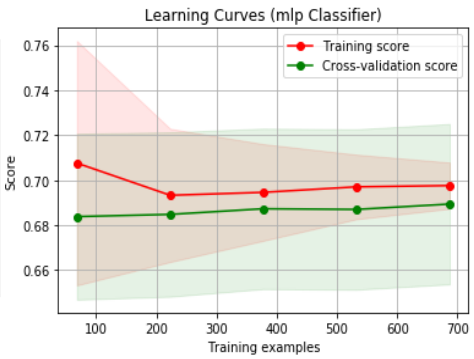


Fig 11- Learning Curve of MLP

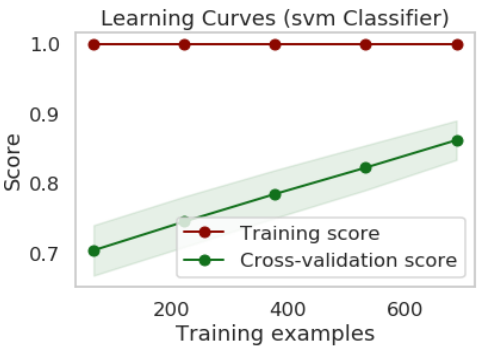
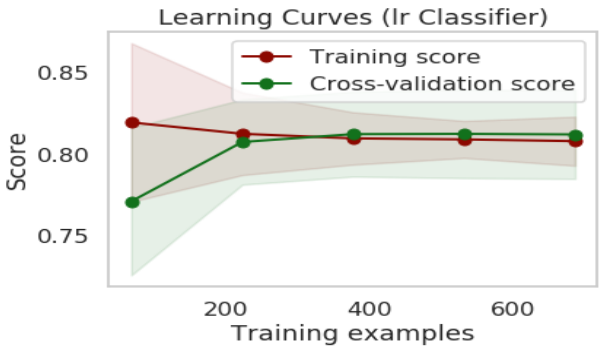


Fig 12- Learning Curve of SVM

LR:[13][33][34] This is a very effective Regression model based on logit function. We use the model when the Dependent variables are binary or discontinuous in nature. We need to find the best fitting model for the Independent and Dependent variable relationship establishment. It deals with probability. A common logit function is sigmoid. Let's explain it with an example. $\boldsymbol{y = a_0 + \sum_{j=1}^{m} \sum_{i=1}^{n} a_i x_j^i}$ .y is the output of the polynomial function of our dataset (Featured). $\boldsymbol{x_1, x_2 \ldots\ldots, x_m}$ are the attributes and $\boldsymbol{a_0, a_1, a_2, a_3, \ldots\ldots, a_n}$ are the coefficients. Also $\boldsymbol{\ln\left(\frac{P}{1-P}\right) = y}$, $\boldsymbol{p = \left(\frac{1}{1+e^{-y}}\right)}$, $\boldsymbol{p = \left(\frac{1}{1+e^{-\left(a_0+\sum_{j=1}^{m}\sum_{i=1}^{n} a_i x_j^i\right)}}\right)}$ .This is the logit function; P is the probability of any event. Hyper parameters are: **C=1.0, solver='lbfgs', penalty='l2', multi_class='ovr', max_iter=500.** C is inverse of the strength of Regularization. Lasso penalty is used here. Then we train the classifier by fitting X_train_std and y_train in it. We got an accuracy of 81.30% on test dataset and got 82.52%, 82.11%, 85.77%, 76.01%, 78.27% accuracies in 5 cross validations.



← Fig 13- Learning curve of LR

Accuracy Measuring: We have the confusion matrix for finding out the accuracy. TN: true negative TP: true positive. FP: false positive. FN: false negative. Among them TN and TP are number of correct answers. Accuracy: what % **percent of predictions are correct** $= \boldsymbol{(TP + TN)/(TP + TN + FP + FN)}$ **. Misclassification rate:** $= \boldsymbol{(FP + FN)/(TP + TN + FP + FN)}$. Precession **(PRE)=** $\boldsymbol{TP/(TP + FP)}$ **. Recall (REC)=** $\boldsymbol{TP/(FN + TP)}$**. F1_score=2** $\boldsymbol{* (PRE * REC)/(PRE + REC)}$. We got confusion matrix and classification report from library metrices of sikit learn library. Moreover, we have score method for all used classifiers. We have another process of estimating the accuracy that is K-fold cross validation, which is available in model_selection library of sikit learn library. We used 5-fold cross validation technique dividing the entire data(standardized) into 5 partitions, considering each partition as a testing dataset each time it iterates through all partitions, and we used all classifiers as estimators in all observations, and got accuracy measures. We can observe here a classification scores without data biasness.

Analysis: SVM provides maximum accuracy on test data (According to ROC) though it had some overfitting problems. MLP provides minimum accuracy on test data (According to ROC). LR provides minimum accuracy on 5-fold cross validation. ADA provides maximum accuracy in 5-fold cross validation. MLP classifier in scikitlearn is not good enough, better to use deep neural networks using Keras or Tensorflow. During training MLP neural network and KNN faced underfitting problem and SVM faced overfitting problem, better to tune those hyperparameters before training the models.

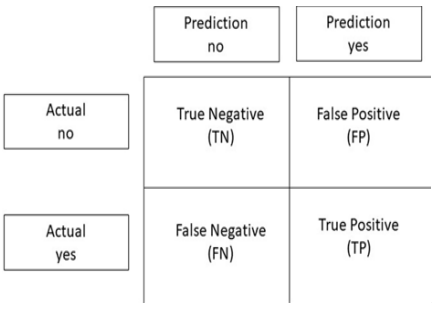


Fig 14- confusion matrix

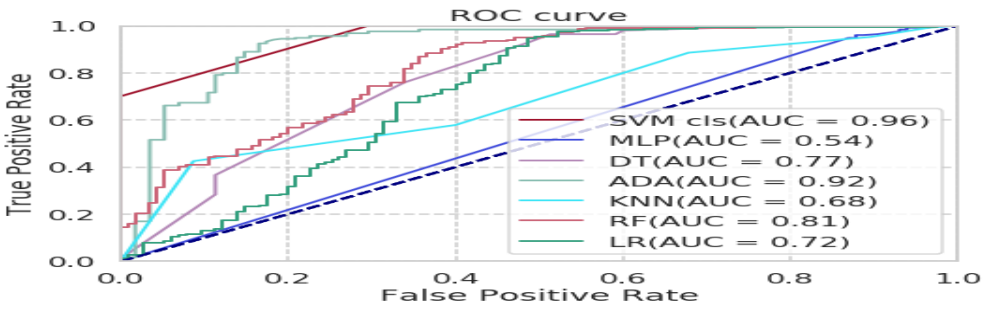


Fig 15-ROC curve on test dataset

Prediction: From scikit learn we go to external and from it we got dumping and loading method. And dumped all the models with. joblib extension. We again load all the. joblib models and used hard voting technique to

predict desired output. We created this function: set_list (Gender, Married, Dependents, Education, Self_Employed, ApplicanIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, Credit_History, Property_Area) =>0/1 . Then we store the predictions in a new .csv file for future use.

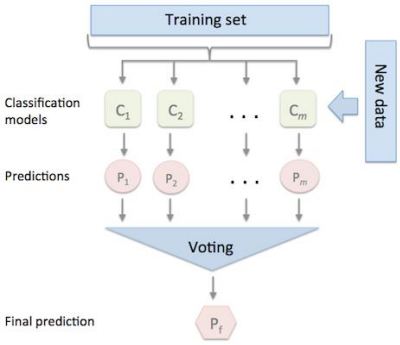


Fig 16-Hard Voting technique [35]

	Loan_ID	Gender	Married	Depender	Education	Self_Empl	Applicant	Coapplica	LoanAmo	Loan_Am	Credit_Hi	Property_	Loan_Status
0	predicted	1	0	0	1	0	5849	0	146.4122	360	1	2	1
1	predicted	1	1	1	1	0	12841	10968	349	360	1	1	0
2	predicted	1	0	0	1	0	5849	0	146.4122	360	1	2	1
3	predicted	1	1	1	1	0	12841	10968	349	360	1	1	0
4	predicted	1	1	0	1	0	1809	1868	90	360	1	2	1
5	predicted	1	1	0	1	0	2083	3150	128	360	1	1	1

Fig 17- Output storing as CSV

FUTURE SCOPE AND IMPROVEMENTS

In this fast-paced world, the number of businesses, small or large, are increasing every day and as a result, the number of loan applicants are growing. Not only the businessmen, but also the common people who aspire to lead a happy life or the youth who wish for a capital boost to be able to implement their revolutionary ideas. In this situation, the banks will constantly be wanting improved means to make their decision of giving loan to a customer even more accurate so that they can make the most profit and stay ahead of the competition. So, our study on bank loan prediction will have a great scope to thrive given more improvements are done to it to make it even better. Our paper right now consists of only the core components for the prediction processive, the trained models that were created using different classifiers. This can be further improved by the implementing it more user-friendly, wherein the models are integrated as part of an application, be it operating system based (Android, IOS, Windows) or Web based. So that users can just go to the bank site and check for themselves whether they are eligible for loan or not. To solve complex Realtime problems. This is basically a frame work, based on this one can further build more and more accurate classifiers by tuning all the hyper parameters. For further details one can go through the Scykit learn documentations or "Hands-On Machine Learning with Scikit-Learn & TensorFlow" written by Aurélien Géron [13], where all classifiers and related hyper parameters are briefly explained. Use larger amount of optimized dataset to improve the model training. One can use complex deep neural networks to create more and more sophisticated and accurate models. Merge the concept of thread with it and speed up the model training.

APPENDIX

A. Abbreviations:

MLPNN: Multilayer Perceptron Neural Network, **CRISP-DM:** Cross Industry Process for Data Mining, **SVM:** Support Vector Machine, **DT:** Decision Tree, **RF:** Random Forest, **ADA:** AdaBoost Classifier, **MLP:** Multilayer Perceptron, **LR:** Logistic Regression, **KNN:** K-Neighbors Classifier, **ROC:** Receiver Operating Characteristic, **ICACT:** International Conference on Advancements in Computing Technologies.

B. System Specification

Laptop: Inspiron 14 3000 Series, **OS:** Windows 10**, RAM:** 4 GB DDR 4**, CPU:** Intel core i3 6th Gen, **GPU:** 0, **Interface:** Jupyter Notebook, **Programming Language:** Python, **Package:** Anaconda.

REFERENCES

1. J. Han and M. Kamber, “Data Mining: Concepts and Techniques,” Morgan Kaufmann, 2000
2. A.K. Pujari, Data Mining Techniques Universities Press (India) Private Limited. 1st Edition, 2001
3. Hu, X., “A data mining approach for retailing bank customer attrition analysis”, Applied Intelligence 22(1):47-60, 2005.

4. Turban, E., Sharda, R. and Delen, D. Decision Support and Business Intelligence Systems – 9th edition, Prentice Hall Press, USA, 2010
5. "The evaluation of consumer loans using support vector machines" by Sheng-Tun Li, Weissor Shiue, Meng-HuahHuang of "Expert Systems with Applications", Volume 30, Issue 4, May 2006, Pages 772-782
6. "Prediction of Loan Risk using Naive Bayes and Support Vector Machine" by Vimala S & Sharmili K.C., International Conference on Advancements in Computing Technologies - ICACT 2018, ISSN: 2454-4248, Volume: 4 Issue: 2,110 – 113
7. "Bank Loan Default Prediction with Machine Learning" by Hongri Jia on Apr 10, 2018
8. "Loan Approval Prediction based on Machine Learning Approach" by Kumar Arun, Garg Ishan & Kaur Sanmeet, IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661, p-ISSN: 2278-8727, PP 18-21
9. "An Exploratory Data Analysis for Loan Prediction Based on Nature of the Clients" by X.Francis Jency, V.P.Sumathi & Janani Shiva Sri, International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-7 Issue-4S, November 2018
10. Kaggle.com->datasets->loan prediction->loan prediction problem uploaded by Amit Prajapat
11. "A Subset Feature Elimination Mechanism for Intrusion Detection System" by Herve Nkiama, Syed Zainudeen Mohd Said and Muhammad Saidu, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 4, 2016
12. "Review of Feature Selection Algorithms for Breast Cancer Ultrasound Image" by Kesari Verma, Bikesh Kumar Singh, Priyanka Tripathi, A. S. Thoke of "New Trends in Intelligent Information and Database Systems", Editors: Dariusz Barbucha, Ngoc Thanh & NguyenJohn Batubara, DOI 10.1007/978-3-319-16211-9
13. "Hands-On Machine Learning with Scikit-Learn and TensorFlow" by Aurélien Géron , Fifth Release on 2018-01-19
14. "About Feature Scaling and Normalization– and the effect of standardization for machine learning algorithms" by Sebastian Raschka on Jul 11, 2014
15. "Decision tree classification of land cover from remotely sensed data" by M.A.Friedl,C.E.Brodley of "Remote Sensing of Environment" Volume 61, Issue 3, September 1997, Pages 399-409
16. "The alternating decision tree learning algorithm" by Yoav Freund and Llew Mason on 1999
17. "A survey of decision tree classifier methodology" by S.R. Safavian, D. Landgrebe, Published in IEEE Transactions on Systems, Man, and Cybernetics (Volume: 21, Issue: 3, May/Jun 1991), Page(s): 660 - 674, INSPEC Accession Number: 4052069, DOI: 10.1109/21.97458
18. "Pruning Adaptive Boosting" by Dragos D.Margineantu and Thomas G. Dietterich, ICML-97 Final Draft
19. "A Short Introduction to Boosting" by Yoav Freund & Robert E. Schapire, Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999.
20. "Classification and Regression by randomForest" by Andy Liaw and Matthew Wiener Vol. 2/3, December 2002, Page no: 18
21. "High density biomass estimation for wetland vegetation using WorldView-2 imagery and random forest regression algorithm" by Onisimo Mutanga, Elhadi Adam and Moses Azong Cho of "International Journal of Applied Earth Observation and Geoinformation", Volume 18, August 2012, Pages 399-406
22. "Random Forests for land cover classification" by Pall Oskar Gislason, Jon Atli Benediktsson and Johannes R. Sveinsson of "Pattern Recognition Letters", Volume 27, Issue 4, March 2006, Pages 294-300
23. "Artificial Neural Networks: A Tutorial", March 1996, pp. 31-44, vol. 29, DOI Bookmark: 10.1109/2.485891, Authors: K.m. Mohiuddin, Jianchang Mao, Anil K. Jain
24. "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences" by M.W Gardner & S.R Dorling of "Atmospheric Environment" Volume 32, Issues 14–15, 1 August 1998, Pages 2627-2636
25. "Neural Networks and Learning Machines" Third Edition by Simon Haykin,McMaster University,Hamilton, Ontario, Canada
26. "Knowledge-based analysis of microarray gene expression data by using support vector machines" by Michael P. S. Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terrence S. Furey, Manuel Ares Jr. and David Haussler on January 4, 2000, 97 (1) 262-267
27. "Least Squares Support Vector Machine Classifiers" by J.A.K. SUYKENS and J. VANDEWALLE, Neural Processing Letters 9: 293–300, 1999

28. "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond" by Bernhard Scholkopf & Alexander J. Smola, ISBN:0262194759
29. "A Branch and Bound Algorithm for Computing k-Nearest Neighbors" by K. Fukunage and P.M. Narendra on July 1975, pp. 750-753, vol. 24, DOI Bookmark: 10.1109/T-C.1975.224297
30. "A fuzzy K-nearest neighbor algorithm" by James M. Keller, Michael R. Gray & James A. Givens, Published in: IEEE Transactions on Systems, Man, and Cybernetics (Volume: SMC-15 , Issue: 4 , July-Aug. 1985), Page(s): 580 - 585, Date of Publication: July-Aug. 1985, INSPEC Accession Number: 2543985, DOI: 10.1109/TSMC.1985.6313426
31. "The Distance-Weighted k-Nearest-Neighbor Rule" by Sahibsingh A. Dudani, Published in: IEEE Transactions on Systems, Man, and Cybernetics (Volume: SMC-6, Issue: 4, April 1976), Page(s): 325 - 327, Date of Publication: April 1976, DOI: 10.1109/TSMC.1976.5408784
32. "Adaptive boosting for SAR automatic target recognition" by Yijun Sun, Zhipeng Liu, Sinisa Todorovic & Jian Li,Published in: IEEE Transactions on Aerospace and Electronic Systems (Volume: 43 , Issue: 1 , January 2007),Page(s): 112 - 125, INSPEC Accession Number: 9457755,DOI: 10.1109/TAES.2007.357120,Date of Publication: 07 May 2007
33. "Applied Logistic Regression Analysis", By Scott Menard, Volume 106; Volume 2002
34. "Logistic Regression in Rare Events Data" by Gary King and Langche Zeng of "Political Analysis" Volume 9, Issue 2, 2001, pp. 137-163, Published online: 04 January 2017
35. http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier_files/majority_voting.png