PROJECT AND DESIGN PHASE
IDEATION

DATE	19 September 2022
TEAM ID	PNT2022TMID28895
PROJECT TITLE	Smart Lender - Applicant Credibility Prediction for Loan Approval
TEAM MEMBERS	KOTHAI S 411719104029 SHARMILA K 411719104046 KALAIVANI L 411719104021 ASHMITHA R 411719104004

Smart Lender - Applicant Credibility Prediction for Loan Approval

ABSTRACTION:

Loan default prediction is one of the most important and critical problems faced by banks and other financial institutions as it has a huge effect on profit. Although many traditional methods exist for mining information about a loan application, most of these methods seem to be underperforming as there have been reported increases in the number of bad loans.
Most of the people depend on bank loans for different purposes like buying home, car, for education purpose. So they apply for loans in banks and provide their details. So, identifying the deserving applicant amongst all the applicantsis very difficult task for banking officials and sometimes banking officials can be biased so there is need of automationin this loan approval system sector.

Let us Loan Prediction model namely,

- Decision Tree
- Random Forest
- K-NN model
- XG-Boost model

1 . DECISION TREE MODEL:

Decision tree is a type of supervised education algorithm (having a pre- defined target variable) that is generally used in category problems. In this approach, we disassociate the population or sample into two or added homogeneous sets (or sub-populations) based on the most significant splitter/ differentiator in input variables.

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

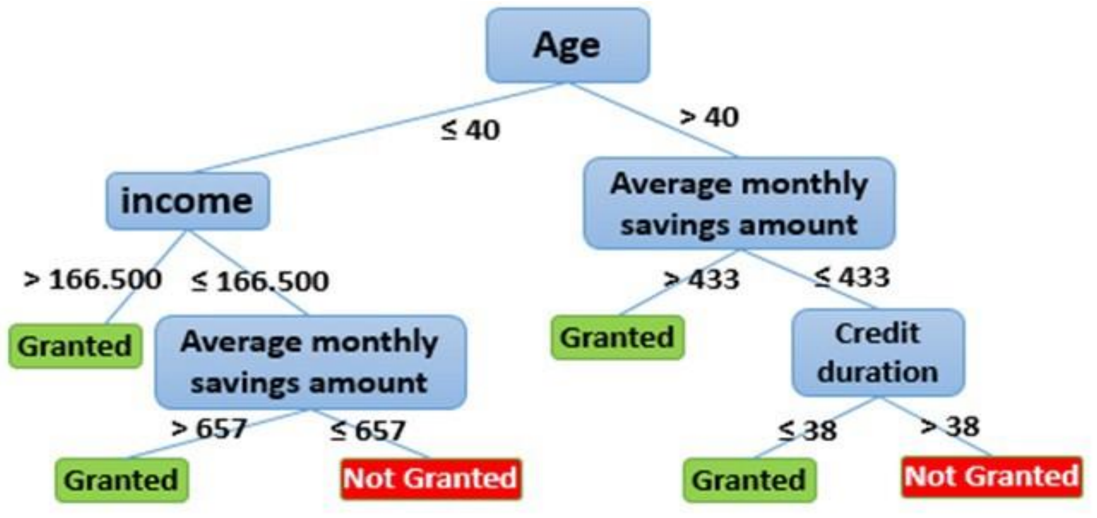
Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but a real so a popular tool in machine learning.

The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Popular selection measures are:

- Information Gain
- Gini Index

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal node, Leaf nodes.

Algorithm:



Decision Tree Algorithm for Loan Prediction

Step 1: use splitting criterion (like Information Gain, Gain Ratio, Gini Index) to select the attribute with the best score that will be chosen to produce the purest node regarding to the target variable (in our case, the attribute that best separates “Granted” from “Not granted”).

Step 2: create the root split node with the consequent’s subsets, then repeat step 1 for each subset by reusing splitting criterion to select the next best attribute to produce the purest sub-nodes regarding to the target variable.

Step 3: repeat step 2 until reaching a stopping Criteria, for instance: Purity of the node > pre-specified limit or Depth of the node > pre-specified limit or simply Predictor values for all records are identical (no more rule could be generated)

Step 4: apply Pruning to avoid overfitting by using a criterion to remove sections of the tree that provide little power to classify and determine the optimum tree size. To do so, we create distinct dataset “training set” and “validation set”, to evaluate the effect of pruning and use statistical test (like Chi-square for CHAID) to estimate whether pruning or expanding a given node produce an improvement. We have two types of Pruning:

- **Pre-pruning** stop growing the tree earlier, before it perfectly classifies the training set.

- **post-pruning** allows the tree to grow and then prune it back

2 . RANDOM FOREST MODEL:

Random forest is a supervised learning algorithm for classification purpose. Random forest is a classifier that contains a number of decision trees on various subsets of given dataset and takes average to improve the classificationaccuracy. Random forest takes the output from each tree in itand based on the maximum votes on predictions, it classifiesdata.

As we are going to classify the large dataset as per the bank’scriteria we need classifying algorithms. In those algorithms, random forest algorithm provides higher efficiency for classification of applicants. It can classify data on the basis ofbank’s criteria and applicant’s provided information.

Main steps of classification algorithm:

- Finalize the dataset that need to be classified.
- Pre-process data and train the training dataset usingrandom forest algorithm
- Apply that model on testing dataset

1. **Input Data:** Download train and test dataset consisting the loan applicant’s information such as application id, name,loan amount, income, co-applicant income, gender, service years. This data is stored in csv files.

2. **Pre-Processing**: The csv file contains some null values and irrelevantinformation which needs to be cleansed. The iloc() and shape() functions can be used to remove the null values. Noise cancellation and data cleansing is done in this step.

3. **Finalize Classifier**: Compare all the classification algorithms and after comparing, finalize the appropriate algorithm on basis of efficiency. So, here we finalize the random forest algorithm.

4. **Train Model** :Train the dataset using finalized random forest classifier.

5. **Apply Model**: Apply this built model on test dataset.

6. **Output**: Classify the applicants based on the applicant's information and bank's criteria using random forest classifier.

Algorithm:

Random forest works in two phase. First phase is to create forest containing N decision trees and Second is to make predictions for each tree and take the average of all outputs.

Step 1: Select random 'k' data points from training set.

Step 2: Build decision tree associated with selected data points.
Step 3: Choose the number 'N' for decision trees that you want to build.
Step 4: Repeat step 1 and 2.

Step 5: For new data point, find the predictions of each decision tree, and assign new data point to the category which was predicted by majority trees.

3 . K-NEAREST NEIGHBOR MODEL:

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data. 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.

We need various libraries to be imported for designing a code that performs training and testing tasks, Predictions, array operations etc.

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

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 certain 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.

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.

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 scikitlearn. 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.

<u>Steps in KNN algorithm</u>:

1. The k-nearest neighbor algorithm is imported from the scikit-learn package.
2. Create feature and target variables.
3. Split data into training and test data.
4. Generate a k-NN model using neighbors value.
5. Train or fit the data into the model.
6. Predict the future.

<u>4 . XGBOOST MODEL :</u>

the Extreme Gradient Boosting (XG-Boost) for bank loan default prediction. The task was to predict if a loan applicant will default in loan payment or not. The analysis was implemented in the python programming language, and performance metrics like accuracy, recall, precision, f1-score were calculated. From the analysis, we found out that the most important features used by our model for predicting if a customer would default in payment or not depends heavily on the location and age of the customer. This paper provides an effective basis for loan credit approval in order to identify risky customers from a large number of loan applicants using predictive modelling.

XG-Boost-Extreme Gradient Boosting is a scalable and highly efficient boosting system. It has been shown to achieve state-of-the-art results on many machine learning tasks. In XG-Boost algorithm unlike the traditional gradient boosting, the process of adding weak learners does not happen sequentially; it approaches this phase in parallel using a multithreaded pattern, thereby resulting in proper utilization of hardware resources leading to greater speed and efficiency. Some important features that make

XG-Boost more efficient than traditional boosting algorithms are:

1. Sparse aware implementation.

2. Weighted quantile sketch for approximate tree learning.
3. Cache-aware access.
4. Blocks for out-of-core computation.

MAIN STEPS FOR MODEL:

1. **Data collection**
   Download train and test dataset consisting the loan applicant's information This data is stored in csv files. This dataset will be further used for preprocessing.

2. **Data cleaning**
   - Firstly import the the DataSet and explore it
   - Then discover the unnecessary and missing values
   - Date are not in DateTime data-type convert this first
   - Dropping unnecessary columns in a Dataset
   - Renaming columns to a more recognizable set of labels
   - Skipping unnecessary rows in a CSV file

3. **Data exploratory analysis**
   - Use Date Column as an Index for analysis based on Date
   - Then select the required columns for visualization.
   - Completely remove the dates we are not certain about and replace them with NumPy's NaN
   - Now design the aesthetic graph using Seaborn.
   - Display the required graph using Matplotlib.

4. **Feature construction and feature selection**
   - Firstly, using the feature standardization method, the feature is scaled by calling the StandardScaler method of the preprocessing sub-module of the scikit-learn module, and the feature value is converted into a standard normal distribution value.
   - Secondly, perform numerical mapping for non-continuous variables, such as mapping the grade (loan grade) variable from A-G to 1-7.
   - Finally, choose the variables that have practical significance and influence on the model as derived variables from the original variables

5. **One-hot-encoding:**
   Unlike other three algorithm to use the XG-Boost algorithm on the data, one-hot-encoding is necessary.
   Step 1: Create the Data.
   Step 2: Perform One-Hot Encoding.
   Syntax:
   class sklearn.preprocessing.**OneHotEncoder**(**, categories='auto', drop=None, sparse=True, dtype=<class 'numpy.float64'>, handle_unknown='error', min_frequency=None, max_categories=None*)
   Step 3: Drop the Original Categorical Variable.

6. **XG-Boost modelling:**

   Inputs:

   - Input data (x, y) N i=1
   - Number of iterations M
   - Choice of the loss-function (y, f)
   - Choice of the base-learner model h(x, θ)

   Algorithm:

   ```
   1. initialize f0 with a constant
   2. for t = 1 to M do
   3. compute the negative gradient gt (x)
   4. fit a new base-learner function h(x, θt )
   5. find the best gradient descent step-size
      ρt : ρt=arg minρ ∑ N i=1 φ [yi, fi−1 (xi) + ρh (xi, θt)]
   6. update the function estimate:
      ft ← ft−1 + ρth(x, t θ )
   7. end for
   ```

7. **Train and test the model.**

   Then we split the train and test data for the give model and perform the required testing and training for the data in order to improve the accuracy of the model

<u>CONCLUSION:</u>

These are the models we use for the loan prediction in Smart Lender project. Now let see the accuracy of the model used:

Model	Decision Tree	Random forest	KNN	XG-Boost
Accuracy score	91%	87.87%	90%	91.67%