# *Project Report*

**Name: Adhya Borisa**

**Batch: DST 20823**

**Project name: Loan Predictor**

## Introduction:

### What is Loan prediction?

loan prediction involves the lender looking at various background information about the applicant and deciding whether the bank should grant the loan. Parameters like credit score, loan amount, lifestyle, career, and assets are the deciding factors in getting the loan approved. If, in the past, people with parameters similar to yours have paid their dues timely, it is more likely that your loan would be granted as well.

Machine learning algorithms can exploit this dependency on past experiences and comparisons with other applicants and formulate a data science problem to predict the loan status of a new applicant using similar rules.

Several collections of data from past loan applicants use different features to decide the loan status. A machine learning model can look at this data, which could be static or time-series, and give a probability estimate of whether this loan will be approved. Let's look at some of these datasets.

### Building Loan Predictor:

## Importing libraries:

- Numpy: It provides support for arrays , matrices and mathematical functions to operate on these arrays

- Pandas: It is for data structures like DataFrames and tools for reading and processing data.
- Matplotlib.pyplot: Plotting library to visualize data.
- Seaborn: It provides a high level interface for drawing attractive and informative statistical graphics.
- Warnings: Using to handle warnings in code execution.

Next, we are loading the dataset as dataframe using pandas.

Then,we are knowing about data by '.columns' , '.info', '.describe()' , etc.

Columns of dataset with their datatypes are:

```
Id                 int64
Income             int64
Age                int64
Experience         int64
Married/Single     object
House_Ownership    object
Car_Ownership      object
Profession         object
CITY               object
STATE              object
CURRENT_JOB_YRS    int64
CURRENT_HOUSE_YRS  int64
Risk_Flag          int64
```

Now we are checking if there are any null or duplicate values.

In 'training data' dataset there is no null and duplicate values.

Now , we are removing the 'ID' column because it is the column that we do not need , keeping it might affect the accuracy of the model.

## Categorizing:

We are categorizing columns ‘profession’ and ‘states’ into broader categories because it’ll help to give:

- Simplified Analysis: Reduces complexity in analysis by focusing on trends across broader groups rather than individual values.

- Feature Engineering: Utilizes these categories as features in machine learning models to improve predictions .

-Improved Model Performance: Enhances model performance by capturing correlations between categories and loan repayment behavior.

## Visualization:

After that, we are performing visualization on data to understand the data more clearly.

## Preprocessing:

Importing Label encoder for categorical data.

The Label Encoder technique from the ‘sklearn.preprocessing’ module facilitates the transformation of categorical labels into numerical representations to train the model.

After that, we are importing preprocessing from sklearn to change values to train machine.

```
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
from sklearn import preprocessing
for col in df.select_dtypes(include=['object']).columns:
    label_encoder = preprocessing.LabelEncoder()
    label_encoder.fit(df[col].unique())
    df[col] = label_encoder.transform(df[col])
    print(f"{col}: {df[col].unique()}")
```

Then we are separating the target variable ‘risk_flag’ because it becomes feasible to train predictive models using the remaining features/variables to predict or classify the target variable accurately.

Now, we are importing train test split function from sklearn

Which divides dataset into training and testing sets.

## Logistic Regression:

Logistic learning is the supervised learning algorithm often used for classification and predictive analytics.

Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables.

```
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression()
reg.fit(x_train,y_train)
```

We are training logistic regression model which is learning the relationships between the target variable and other variables from training data.

Then we are calculating the accuracy of model which is 87.75%.

## Decision Tree:

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 nodes and leaf nodes.

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(random_state=0, max_depth=4,
min_samples_leaf=1, min_samples_split=2, class_weight='balanced')
dtree.fit(x_train, y_train)
```

We are training decision tree classifier using the training dataset.

Parameters

- class_weight='balanced': It is adjusting the class weights to handle imbalanced classes & giving fair importance to minority classes.

- max_depth=4: It is limiting the maximum depth of the tree to prevent overfitting.

- random_state=0: It is ensureing the reproducibility by initializing the random number generator .

Then we are calculating the accuracy score which is 71.07%.

After that we are making confusion matrix:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, pred2))
```

Metrics like accuracy, precision, recall, and F1-score are calculated based on these matrix elements.

F1 score, Precision score and recall score is 0.7106646825.

From the output of confusion matrix we can interpert :

- True Positives: [67719] instances were correctly predicted as positive.

- True Negatives: [3916] instances were correctly predicted as negative.

- False Positives: [20739] instances were incorrectly predicted as positive.

- False Negatives: [8426] instances were incorrectly predicted as negative.

**Gaussian Naive Bayes Classifier:**

Naive Bayes is a supervised learning algorithm that is based on the concept of conditional probability and bayes theorem.

Gaussian Naive Bayes is a type of classification algorithm working on continuous normally distributed features that is based on the Naive Bayes algorithm.

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

After training Gaussian naïve bayes, we are evaluating its accuracy score which is 87.76%.

After that we are making confusion matrix:

```
print(confusion_matrix(y_test, pred3))
```

From the output we can interpert :

- True Positives: [88458] instances were correctly predicted as positive.

- True Negatives: [0] instances were correctly predicted as negative.

- False Positives: [0] instances were incorrectly predicted as positive.

- False Negatives: [12342] instances were incorrectly predicted as negative.

- The resulting confusion matrix indicates that all instances are predicted as belonging to a single class

**Random Forest:**

The random forest is also a supervised learning algorithm which is based on ensemble learning, which is process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

It contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion =
'entropy', random_state = 0)
classifier.fit(x_train, y_train)
```

After training the random forest classifier, we are predicting test set.

Then, we are evaluating accuracy score which is 89.91%.

Confusion matrix is giving,

- True Positives: [84107] instances were correctly predicted as positive.

- True Negatives: [6523] instances were correctly predicted as negative.

- False Positives: [4351] instances were incorrectly predicted as positive.

- False Negatives: [5819] instances were incorrectly predicted as negative.

**Conclusion:**

- The accuracy of Logistic Regression is: 87.75 %

- The accuracy of Decision Tree Classifier is: 71.07 %

- The accuracy of Naive Bayes is: 87.76 %

- The accuracy of Random Forest Classification is: 89.91 %

- In conclusion, Among the tested models the Random Forest Classification demonstrated the highest accuracy, indicating its potential for promising model for predicting loan approvals.