# Technocolabs Softwares

## MORTGAGE-BACKED SECURITIES

### Group Members


Aishwarya Desai
Sachin
Wania
Usman Babar
Pooja R

Pavan Kumar
Siva Sakthi
Hassan Safwat
Hamza Ramzan

# Contents

# INTRODUCTION

## MORTGAGE-BACKED SECURITIES(MBS):

Mortgage-backed securities (MBS) are financial instruments that represent a form of asset-backed security. These securities are created by bundling a group of individual mortgage loans, such as residential mortgages, into a single investment product. The cash flows from the underlying mortgages, including principal and interest payments made by homeowners, are used to generate income for investors who purchase the MBS.

The main benefit of mortgage-backed securities is that they allow financial institutions to transfer the risk associated with mortgage loans to investors. This, in turn, provides additional liquidity to mortgage lenders, allowing them to issue more mortgages and potentially lower interest rates for borrowers.

## PREPAYMENT RISK:

Prepayment risk refers to the risk that the underlying loans in a mortgage-backed security (MBS) or other asset-backed security will be paid off earlier than expected. When borrowers repay their loans ahead of their scheduled payment dates, it can have an impact on the cash flows received by the investors holding the MBS.

Prepayment risk is particularly relevant in the context of mortgage-backed securities, where the underlying assets are mortgages, when homeowners decide to refinance their mortgages at lower interest rates or sell their homes, they pay off their existing loans, resulting in prepayments.

The impact of prepayments on MBS investors:

**(1). Reduced Interest Income:**

When borrowers prepay their mortgages, they effectively cut short the interest payments investors would have received if the loans had continued according to their original schedules. This can lead to a decrease in the overall interest income received by the investors.

**(2). Reinvestment Risk:**

Prepayments can also create reinvestment risk for investors. As the principal from prepaid loans is returned to investors, they may have to reinvest that money in new securities or assets. However, if interest rates have fallen since the original investment, investors may have to settle for lower yields on their reinvestments.

Prepayment risk is influenced by various factors, including changes in interest rates, housing market conditions, and borrower behaviour. When interest rates drop, borrowers are more likely to refinance their mortgages to take advantage of lower rates, leading to higher prepayment rates. Conversely, when interest rates rise, prepayments tend to decrease as borrowers are less incentivized to refinance.

## DATA

The data is obtained from Freddie Mac official portal for home loans.

The size of the home loans data is (291452 x 28) that is of 291452 data rows and 28 columns

**FEATURES OF THE DATASET:**

1. Credit score of the client.
2. The maturity date of the mortgage.
3. The amount or percentage of insurance on the mortgage
4. Debit to income ration of the borrower
5. Mortgage interest rate
6. The purpose of the loan.
7. Loan sequence number which denotes the unique loan ID
8. The number of borrowers issued on the loan.
9. Prepayment Penalty Mortgage which denotes if there is any penalty levied on prepayment of the loan.
10. The property type, the state in which property is and its postal code and address.
11. The information about the seller and service company.
12. HARP indicator-denotes if the loan is HARP or non-HARP,
13. Interest only indicator-Denotes if the loan requires only the interest payments over the period of maturity or not respectively.
14. debt-to-income ratio(DTI): Your debt-to-income ratio (DTI) compares how much you owe each month to how much you earn. Specifically, it's the percentage of your gross monthly income (before taxes) that goes towards payments for rent, mortgage, credit cards, or other debt.
15. PPM: A type of mortgage that requires the borrower to pay a penalty for prepayment, partial payment of principal or for repaying the entire loan within a certain time period. A partial payment is generally defined as an amount exceeding 20% of the original principal balance.

16. NumBOrrowers: A borrower describes an individual, entity, or organization applying for funds, i.e., a loan from a lender under an agreement to repay the same later.
17. Everdeliquent: Being delinquent refers to the state of being past due on a debt. Delinquency occurs as soon as a borrower misses a payment on a loan, which can affect their credit score. Delinquency rates are used to show how many accounts in a financial institution's portfolio are delinquent.

18. IsFirstTimeHomebuyer: According to the agency, a first-time homebuyer is: Someone who hasn't owned a principal residence for the three-year period ending on the date of purchase of the new home. An individual who has never owned a principal residence even if the person's spouse was a homeowner.

19. Creditrange: A credit score is a prediction of your credit behavior, such as how likely you are to pay a loan back on time, based on information from your credit reports.

20.Monthly_income: Your gross monthly income includes all sources of money that you receive over the course of a month, including but not limited to regular wages, earnings from side jobs and investments. Lenders consider your gross monthly income to determine your creditworthiness and ability to repay loans.

21.Prepayment: Prepayment is an accounting term for the settlement of a debt or installment loan in advance of its official due date. A prepayment may be the settlement of a bill, an operating expense, or a non-operating expense that closes an account before its due date.

22. MSA: MSAs (Marketing Services Agreement) with Mortgage Companies.

23. MIP: Federal Housing Administration (FHA) mortgage loans are designed to help people who might have trouble getting other types of mortgage loans to buy a home. And if a homebuyer uses an FHA-backed loan, they're required to pay a mortgage insurance premium (MIP).

24. OCLTV: Loan-to-value (LTV) is calculated simply by taking the loan amount and dividing it by the value of the asset or collateral being borrowed against.

25. debt-to-income ratio:Your debt-to-income ratio (DTI) is all your monthly debt payments divided by your gross monthly income. This number is one way lenders measure

your ability to manage the monthly payments to repay the money you plan to borrow. Different loan products and lenders will have different DTI limits.

26. OrigUPB :Loan origination is the process by which a borrower applies for a new loan, and a lender processes that application. Origination generally includes all the steps from taking a loan application up to disbursal of funds (or declining the application).

27 .PropertyState: Property State means, with respect to a particular parcel of Land or Security Instrument purporting to encumber it, the state where such Land is located.

28. LTV_Range:The loan-to-value (LTV) ratio is an assessment of lending risk that financial institutions and other lenders examine before approving a mortgage. Typically, loan assessments with high LTV ratios are considered higher-risk loans. Therefore, if the mortgage is approved, the loan has a higher interest rate.

29. MonthsDelinquent:A delinquent status means that you are behind in your payments. The length of time varies by lender and the type of debt, but this period generally falls anywhere between 30 to 90 days.

30. Prefix: The designation assigned by the issuer denoting the type of the loans and the security

31. Original Interest Rate: The interest rate of the loan as stated on the note at the time the loan was originated or modified.

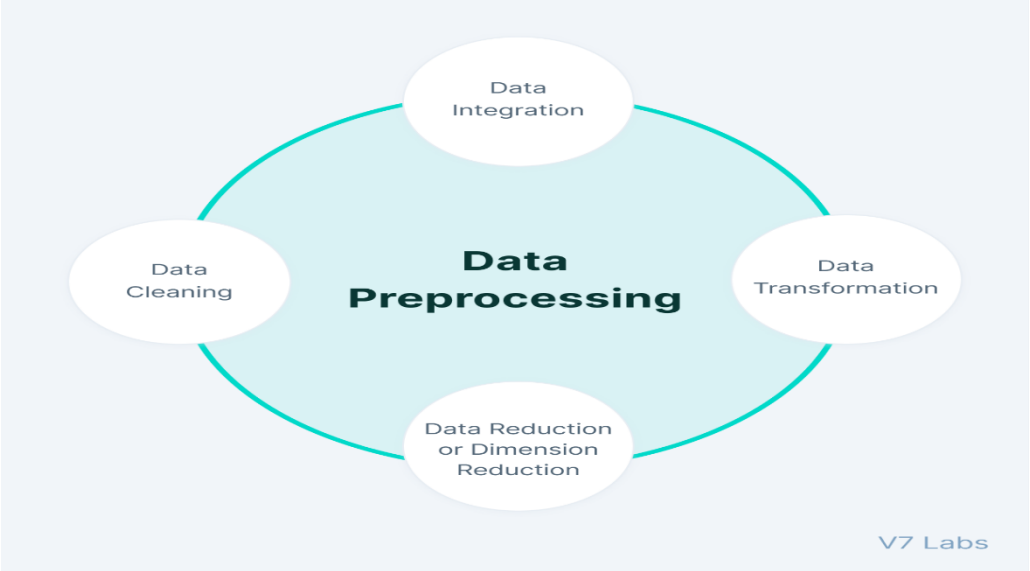
# PROBLEM STATEMENT

Prepayment risk refers to the uncertainty of how likely borrowers are to pay off their loans early. This risk affects investors in mortgage-backed securities who receive cash flows from these loans. If borrowers repay their loans ahead of schedule, it can reduce the interest income received by investors and create challenges in finding suitable reinvestment opportunities.

The prediction of prepayment risk involves estimating the likelihood of borrowers prepaying their loans before their due dates.

# DATA PREPROCESSING:

Data preprocessing is a crucial step in the data science workflow that involves cleaning, transforming, and organizing raw data to make it suitable for analysis and modeling. The quality of the data preprocessing directly impacts the accuracy and effectiveness of the machine learning or data analysis models.




Some of the key steps we have followed in data pre-processing :

1. Analyzing the data
2. Missing value treatment
3. Handling outliers
4. Creation of target variable
5. Transforming some variables into appropriate format

## Analyzing the data

Understanding the data in machine learning involves thoroughly examining the dataset to gain insights into its structure, distribution, and characteristics.

## Data Exploration:

Check the size of the dataset: Understand the number of samples (instances) and features (variables) present in the dataset.

Data Types: Identify the data types of each feature (e.g., numerical, categorical, text, datetime) to determine how to handle and preprocess them.

Data exploration and visualization are valuable tools for gaining insights and guiding your choices throughout the machine learning process.

## Missing value treatment

After the data exploration, I came to some conclusions .They are

- Many null values.
- Some variable "X" which has to be considered as null value.

So I started replacing the "X" variable with null value and handled the missing values.

Replace missing values in categorical features with the mode (most frequent value) of that feature. Replace missing values in numerical features with the mean (or median) of the available values of that feature. This method is simple and effective.

## Handling outliers

Handling outliers is an important step in data pre-processing, as outliers can significantly affect the performance and accuracy. Outliers are data points that deviate significantly from the majority of the data and may result from errors in data collection or represent rare events.

Capping/Clipping: Cap or clip the extreme values by setting a predefined upper and lower threshold. Data points above or below these thresholds are replaced with the threshold value.

We used capping method to control the outliers.

## Creation of target variable

As per the problem statement it is a classification task. So we have to create a target variable

for binary classification. So we chose status as target variable and transformed to binary form with the help of default date feature

## Transforming some variables into appropriate format

Transforming variables into the appropriate format is an essential data preprocessing step to ensure that the data is in a suitable representation for machine learning models or other analytical tasks.


Many features in data are not in appropriate format such as credit score, DTI , etc. so we have converted the features into categorical columns. Later we have applied encoding techniques for *model preparation.*

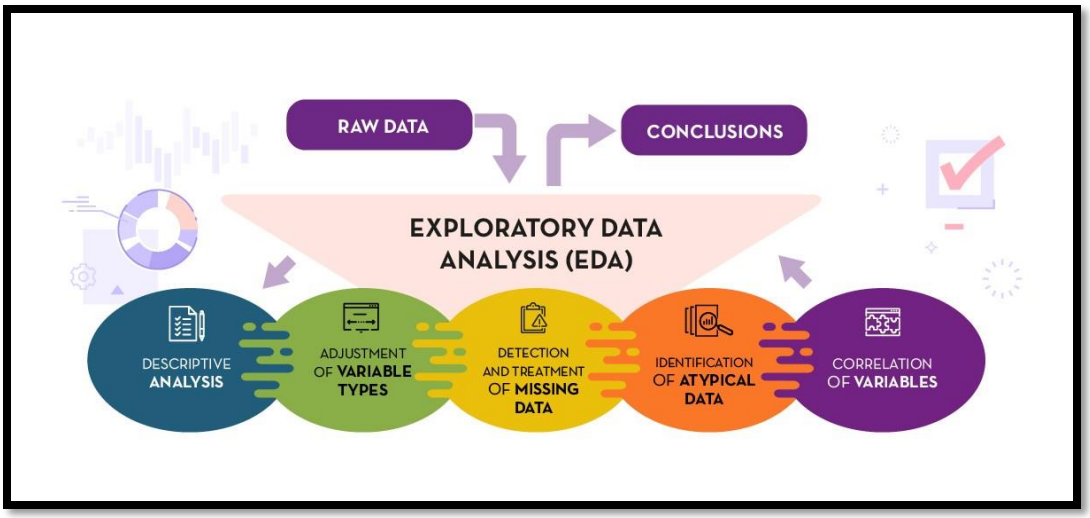
# EDA (Exploratory Data Analysis)

EDA is a critical step in the machine learning pipeline. It involves the process of visually and statistically examining data to understand its structure, relationships, patterns, and characteristics. The primary goal of EDA is to gain insights into the data, identify potential issues, and make informed decisions on how to pre-process and model the data effectively.

Depending on the number of columns we are analysing we can divide EDA into two types:

1. Univariate analysis
2. Bivariate analysis

For good analysis we divided the columns into two groups. They are categorical columns and numerical columns.




## Univariate analysis

Univariate analysis is analyzing a single feature at a time.

For every categorical columns we applied univariate analysis using the pie charts to know the value counts of each categorical column . For every numerical columns we applied univariate analysis using the histogram, distplot and boxplot to know the value counts of each numerical column .

## Bivariate analysis

The primary goal of bivariate analysis is to understand how the values of one variable relate to the values of another variable. This analysis is particularly useful for identifying correlations, patterns, trends, and associations between two variables.

Present visualizations that explore the relationship between the target variable and other features .

By performing bivariate analysis, data analysts can gain insights into how two variables interact with each other and understand their joint behavior. It is a crucial step in the exploratory data analysis (EDA) process and helps guide further analysis and modeling decisions.

### Data Distribution Analysis:

The distribution of each feature in the dataset was examined to detect any anomalies or outliers. This step ensures that the data is appropriate and representative for training machine learning models.

### Correlation Analysis:

The relationships between features in the dataset were studied to identify strong correlations. Detecting multicollinearity issues is crucial as they can adversely impact the performance of machine learning algorithms.

### Feature Importance Analysis:

The importance of each feature in predicting ever delinquency was assessed through correlation and heatmap analysis. This enabled the identification of the most significant features that contribute to the prediction of ever delinquency.

### Data Visualization:

Various data visualization techniques, including histograms, scatter plots, and violin plots, were employed to visualize the distribution and relationships between different features in the dataset. These visualizations provide valuable insights and reveal any potential patterns or associations in the data.

### Correlation Plot of Numerical Variables

The correlation plot analysis revealed interesting insights into the relationship between numerical variables and ever delinquency. Some features exhibited a strong positive correlation, indicating that they have a significant influence on the likelihood of ever delinquency:

**Original UPB (Unpaid Principal Balance):** Loans with a higher outstanding balance were associated with a greater risk of ever delinquency.

**Current Interest Rate:** Loans with higher interest rates were found to be more susceptible to ever delinquency.

**Original Loan-to-Value (LTV) ratio:** Loans with a higher LTV ratio, meaning a larger proportion of the property's value was borrowed, showed an increased risk of ever delinquency.

On the contrary, certain features displayed a strong negative correlation with ever delinquency, suggesting that they act as protective factors:

**Loan Age:** Loans with a longer history were observed to have a lower risk of ever delinquency.

**Current Loan-to-Value (LTV) ratio:** Loans with a lower current LTV ratio, indicating that a smaller proportion of the property's value was borrowed, were associated with a decreased risk of ever delinquency.

# Feature Engineering

Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling.

## Feature Creation:

Feature creation is finding the most useful variables to be used in a predictive model. The process is subjective, and it requires human creativity and intervention. The new features are created by mixing existing features using addition, subtraction, and ration, and these new features have great flexibility.

After preprocess and Exploratory data analysis (EDA) task we got data contain 34 columns and out of those one of variables as Target variable.

Column require for feature engineering are:-

```
df.columns
```

```
Index(['MSA', 'MIP', 'Units', 'Occupancy', 'OCLTV', 'DTI', 'OrigUPB',
       'OrigInterestRate', 'Channel', 'PPM', 'PropertyState', 'PropertyType',
       'LoanPurpose', 'OrigLoanTerm', 'NumBorrowers', 'SellerName',
       'ServicerName', 'EverDelinquent', 'MonthsDelinquent',
       'IsFirstTimeHomebuyer', 'LTV_Range', 'CreditRange', 'RepPayRange',
       'Month_Difference', 'monthly_payment', 'total_payment',
       'interest_amount', 'monthly_income', 'monthly_rate',
       'monthly_priciple_amount', 'principal_amount_remaining',
       'priciple_amount_paid', 'prepayment', 'O', 'S',], ,
```

## Feature Selection:

While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning. "Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features.".

In this task first of all I had Split the data into test and training data by this I can observe the features importance.

```
# separate dataset into train and test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
    x,
    y,
    test_size=0.3,
    random_state=0)

x_train.shape, x_test.shape
```

```
((125626, 33), (53840, 33))
```

After this, o I use correlation and heat map to understand the relationship between variables and remove the multi collinearity variables.

```
import seaborn as sns
#Using Pearson Correlation
plt.figure(figsize=(12,10))
cor = x_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```

After this, I have a correlation of data in visual form and I can understand it better. Then I used correlation function.
With the following function we can select highly correlated featuresit will remove the first fe ature that is correlated with anything other feature

```
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if (corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

After executing this function, I got 7 columns/features that are highly correlated. The threshold value that I set was 0.9.

```
corr_features = correlation(x_train, .9)
len(set(corr_features))
```

```
7
```

These are the Columns/Features which I don't want because they don't pass the threshold that I set. So I dropped these columns.

```
{'Month_Difference',
 'interest_amount',
 'monthly_payment',
 'monthly_priciple_amount',
 'monthly_rate',
 'principal_amount_remaining',
 'total_payment'}
```

After that I took all features and applied **sklearn.feature_selection** method to get the most important features of this data. So I got most important 6 features:

```
from sklearn.feature_selection import mutual_info_regression
# determine the mutual information
mutual_info = mutual_info_regression(x_train, y_train)
mutual_info
```

```
from sklearn.feature_selection import SelectPercentile


## Selecting the top 20 percentile
selected_top_columns = SelectPercentile(mutual_info_regression, percentile=20)
selected_top_columns.fit(x_train.fillna(0), y_train)
```

```
df
```

| | DTI | OrigUPB | OrigInterestRate | monthly_income | priciple_amount_paid | prepayment |
|---|---|---|---|---|---|---|
| 0 | 27 | 117000 | 6.750 | 2800.0 | 5390.32 | 15387 |
| 1 | 17 | 109000 | 6.500 | 4000.0 | 14450.40 | 31465 |
| 2 | 16 | 88000 | 6.875 | 3600.0 | 5125.50 | 29325 |

## Feature Extraction:

Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data. The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling. Feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA).

### Principal Components Analysis

I had taken those 6 features for feature extraction and taken the target variable (prepayment).

I perform PCA and feature extraction using the libraries I imported so it gives pca components and I also calculated the variance of those components it gives around 90 percent so I take

```
pca = PCA()
pca.fit_transform(X)
```

```
array([[ 0.03034654,  0.23727445, -0.67758493,  0.20868777, -0.15547714],
       [ 1.55727455,  0.04954932, -1.79544609, -1.00507213, -0.03161579],
       [ 0.11631359,  1.36238878, -0.97751785, -0.25120267, -0.18719565],
       ...,
       [ 0.18614707, -1.13345922, -3.38522761,  1.76015784,  0.20389543],
       [-1.93083785, -1.08138154,  0.4693892 ,  0.50349945,  0.83175609],
       [-1.33388372,  0.73911595, -0.49079031,  0.08976572, -0.01810866]])
```

those pca's.

But after transformation, we got only 3 principle components:

```
principalComponents = pca.fit_transform(X)
```

```
# We will use 3 principal components
pca = PCA(n_components=3)
## This will be the new data fed to the algorithm.
final_data = pca.fit_transform(new_data)
principal_Df = pd.DataFrame(data = final_data
             , columns = ['PC1','PC2','PC3'])
```

Benefits of feature engineering in machine learning:

- It helps in avoiding the curse of dimensionality.
- It helps in the simplification of the model so that the researchers can easily interpret it.
- It reduces the training time.
- It reduces overfitting hence enhancing the generalization.

## Model selection and feature comparison:

After selecting the features calculated the accuracies of all models with feature selection and feature extraction output to know the better features.

**Feature selected output accuracies:**

```
# Create an instance of the LinearRegression model
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, Y_train)

# Make predictions on the testing data
Y_pred = model.predict(X_test)
```

```
# Print the evaluation metrics
print("Training R-squared:", train_r2)
print("Testing R-squared:", test_r2)
print("Training Mean Squared Error:", train_mse)
print("Testing Mean Squared Error:", test_mse)
```

```
Training R-squared: 0.6887136532871649
Testing R-squared: 0.6912597732212389
Training Mean Squared Error: 186807087.5810103
Testing Mean Squared Error: 202395500.38175884
```

So out of those feature selection is gives the best accuracy.

# Model Building

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.

The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

## 1. Binary Classifier:

If the classification problem has only two possible outcomes, then it is called as Binary Classifier.

Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.

## 2. Multi-class Classifier:

If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.

Example: Classifications of types of crops, Classification of types of music.

## Types of ML Classification models:

Classification Algorithms can be further divided into the Mainly two category:

1. Linear Models:-
    - Logistic Regression
    - Support Vector Machines
2. Non-linear Models:-
    - K-Nearest Neighbours 2Kernel SVM
3. Naïve Bayes
4. Decision Tree Classification
5. RandomForest Classification

We are doing the classification and regression with three models both linear and non linear . logistic regression and linear regression (linear models) and random forest (non linear) model.

## RANDOM FOREST:

ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

"Random Forest is a classifier that 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.

**Assumptions:**

1. There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
2. The predictions from each tree must have very low correlations.

**Working:**

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

1. Select random K data points from the training set.
2. Build the decision trees associated with the selected data points (Subsets).
3. Choose the number N for decision trees that you want to build.
4. For new data points, find the predictions of each decision tree, and assign the newdata points to the category that wins the majority votes.

## LOGISTIC REGRESSION:

Logistic Regression is a statistical algorithm used for binary classification problems, where the target variable has two possible outcomes (e.g., yes/no, true/false, 0/1). Despite its name, logistic regression is a classification algorithm and not a regression algorithm. It predicts the probability that an instance belongs to a particular class.

**Assumptions:**

1. The target variable should be binary or dichotomous, meaning it should have only two possible outcomes.
2. Independence of observations: Each data point should be independent of others.
3. Linearity: Logistic regression assumes a linear relationship between the independent variables and the log-odds of the target variable.

**Working:**

Logistic regression uses a logistic function (also called the sigmoid function) to model the relationship between the independent variables and the probability of the target variable belonging to a particular class. The function's output ranges between 0 and 1, representing the probability of the event occurring.

The working process can be summarized as follows:

Data Preparation: Prepare the dataset by encoding categorical variables and handling missing data.

- Model Training: Estimate the parameters of the logistic regression model using optimization techniques like Maximum Likelihood Estimation (MLE) or Gradient Descent. The goal is to find the best-fitting line that separates the two classes.
- Sigmoid Function: The logistic regression model applies the logistic (sigmoid) function to the linear combination of the input features and their corresponding

weights. The sigmoid function maps the linear output to a value between 0 and 1.

$$S(z) = 1 / (1 + e^{(-z)})$$

where 'z' is the linear combination of features and weights.

- Decision Boundary: The model calculates the probability of an instance belonging to a particular class. By using a threshold value (often 0.5), it classifies the instance into one of the two classes. If the predicted probability is greater than or equal to the threshold, the instance is classified as class 1; otherwise, it is classified as class 0.
- Model Evaluation: Evaluate the model's performance using metrics like accuracy, precision, recall, F1 score, or ROC-AUC.

Logistic regression is widely used in various fields like medicine, finance, marketing, and social sciences for binary classification tasks and can be extended to handle multi-class classification through techniques like one-vs-all or softmax regression.

## STEPS OF IMPLEMENT ATION:

1. **Data Pre-processing step:-**

in this step we are taking the features and checking for null values and replace null values and taking the data into X as independent and Y as dependent feature, and next we have performed the PCA technique to select best features and later we have splitted data into train and test using train_test_split function into 0.25 perecnt ratio.

2. **Fitting both models to the Training set:-**

After splitting data then we have to import model and we have to fit it with x_train and y_train then model is trained and ready for prediction.

3. **Predicting the test result:-**

After developing the model we have to test with test data to identify the model working. It gives the output.

It gives the Logistics Regression Accuracy : 0.8611245651886751

Random Forest Classifier Accuracy : 0.8621152745365682

4. **Test accuracy of the result(Creation of Confusion matrix):-**

Now we have compare the predicted values with actual data and printed the classificationreport and it gives accuracy and score also.

Confusion matric:

```
array([[36516,  142],
       [ 6166, 2598]], dtype=int64)
```

## classification report:

5. **creating the pipeline for model:**

After completing all details we have created a pipeline for model to store it in .pkl file. For this requirement we had to import pickle module and pipeline library with that pipeline library we have to created pipeline and and store as classification.pkl format with pickle dump and when we have requirement we have to import that classification.pkl with pickle.;load function. And also we can test data it will also give better accurracy

**fit the pipeline:-**

pipeline_randomforest.fit(X_train, Y_train)

save it with pickle.dump:-

import model with pickle.load:-

predict with pipeline:-

Y_data_pred = Classify_model.predict(X_test)

print(Y_data_pred)

accuracy of model with pipeline:

**Advantage of model:**

1. It takes less training time as compared to other algorithms.
2. It predicts output with high accuracy, even for the large dataset it runs efficiently.
3. It can also maintain accuracy when a large proportion of data is missing.

## MBS prepayment risk analysis :

**Linear Regression Analysis**

**3. LINEAR REGRESSION:**

Linear Regression is a supervised statistical algorithm used for predicting continuous numeric values. It establishes a linear relationship between the dependent variable (target) and one or more independent variables (features) by fitting a straight line to the data points that best represent the overall trend.

**Assumptions:**

1. Linearity: Linear regression assumes a linear relationship between the independent variables and the dependent variable. The model tries to find the best-fitting line to the data.
2. Independence of Errors: The errors (residuals) should be independent of each other. In other words, the error term for one data point should not be influenced by the error term of another data point.
3. Homoscedasticity: The variance of the errors should be constant across all levels of the independent variables. This means that the spread of the residuals should be roughly the same across the range of predicted values.
4. No Multicollinearity: The independent variables should not be highly correlated with each other. Multicollinearity can lead to unstable coefficient estimates.

**Working:**

The working process of Linear Regression can be summarized as follows:

1. Data Preparation: Prepare the dataset by cleaning the data, handling missing values, and encoding categorical variables if necessary.
2. Model Training: Linear Regression finds the best-fitting line that minimizes the sum of squared differences between the predicted values and the actual values. This is usually done using a technique called Ordinary Least Squares (OLS).
3. Linear Equation: The model represents the relationship between the dependent variable 'y' and the independent variables 'x1, x2, ..., xn' as a linear equation:

   $$y = \beta0 + \beta1 * x1 + \beta2 * x2 + ... + \beta n * xn$$

   where '$\beta0$' is the y-intercept, '$\beta1, \beta2, ..., \beta n$' are the coefficients (slopes) for each independent variable, and 'x1, x2, ..., xn' are the corresponding feature values.

4. Model Evaluation: Evaluate the performance of the model using various metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared (coefficient of determination), etc.
5. Making Predictions: Once the model is trained and the coefficients are determined, it can be used to make predictions on new data by plugging in the feature values into the linear equation.

Linear Regression is a widely used algorithm in various fields such as economics, finance, social sciences, and engineering for tasks like price prediction, demand forecasting, trend analysis, and more. It can be extended to handle multiple regression (more than one independent variable) and polynomial regression to capture non-linear relationships between variables.

For Regression Model we have created following variabkes:

variables 1- Equated Monthly Instalments (EMI)

1. monthly_payment
2. total_payment
3. interest_amount

4. monthly_income
5. monthly_rate
6. monthly_priciple amount
7. principal_amount_remaining
8. priciple_amount_paid
9. prepayment

## Model Building:-

For the regression task we have selected 9 Independent variables which affects the target variables and prepayment as dependent variables

**Independent Variables: -**

DTI , PPM, NumBorrowers , ServicerName, EverDelinquent, IsFirstTimeHomebuyer, CreditRange, monthly_income , s

**Dependent Variable:**

prepayment

**Steps: -**

1. Filling Missing values by using Simple Imputer for both categorical features and numerical features
2. Converting Categorical Features into numerical by using One Hot Encoding
3. Standardization of numerical features by using Standard Scaler
4. Applying Linear Regression for prediction

**Model:**

```
model = LinearRegression()

model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)
```

**Results: -**

The overall R-squared value for Linear Regression is 91 % with Testing R-squared: 0.92, Training Mean Squared Error: 0.08, Testing Mean Squared Error: 0.08

# Pipeline

The execution of the workflow is in a pipe-like manner. Scikit-learn is a powerful tool for machine learning, provides a feature for handling such pipes under the sklearn.pipeline module called Pipeline. A machine learning pipeline can be created by putting together a sequence of steps involved in training a machine learning model.

It can be used to automate a machine learning workflow. The pipeline can involve pre-processing, feature selection, classification/regression, and post-processing.

## Process of Pipeline:

## Creation Of Pipeline For Classification Model

- **Data Preparation**: The code starts by loading the dataset `df`, which includes relevant features (IV) and the target variable ("EverDelinquent"). The independent variables (IV) are stored in the `X` matrix, and the target variable is stored in the `Y` matrix.
- **Data Splitting**: The data is split into training and testing sets using the `train_test_split` function from sklearn. The training set (`x_train` and `y_train`) will be used to train the models, and the testing set (`x_test` and `y_test`) will be used to evaluate the model's performance.
- **Preprocessing Transformers**: Two preprocessing transformers are defined - one for numeric features and one for categorical features. For numeric features, missing values are imputed using the median and then scaled using `StandardScaler`. For categorical features, missing values are imputed using the most frequent value, and one-hot encoding is applied using `OneHotEncoder`.
- **ColumnTransformer**: A `ColumnTransformer` named `preprocessor` is created to apply the appropriate preprocessing transformer to each set of features. It specifies that numeric features should be processed using `numeric_transformer`, and categorical features should be processed using `categorical_transformer`.
- **Multiple Pipelines for Multi-Label Classification**: A list of pipelines is created, where each pipeline contains the `preprocessor` defined in the `ColumnTransformer` and a `LogisticRegression` classifier. A separate pipeline is created for each target variable (each column in `y_train_reshaped`), treating it as a binary classification task.
- **Fitting Pipelines to Training Data**: The pipelines are then fitted to the training data (`x_train` and `y_train_reshaped`) using a loop. Each pipeline is trained separately for each target variable (column) in `Y`

1. **Saving the Trained Model**: The trained classification model, which is stored in the `pipeline` variable, is saved to a pickle file named "classification.pkl." This allows the model to be easily loaded and used later without retraining.
2. **User Input for Loan Application Data**: The code prompts the user to input loan application data, including features such as "MSA," "MIP," "OCLTV," "DTI," "OrigUPB," "PropertyState," "MonthsDelinquent," "LTV_Range," "CreditRange," and "RepPayRange." The user can enter multiple sets of data (rows) to make predictions for different loan applications.
3. **Model Prediction**: For each set of loan application data provided by the user, a new DataFrame `data` is created to store the input data. The model is then used to make predictions on each row of `data` using the `predict` method of the pipeline.

**Loading the Model**: The code uses the pickle.load() method to open the "Classification.pkl" file in binary read mode ('rb'). The loaded model is stored in the loaded_model variable.

## Creation Of Pipeline For Regression Model

1. **Data Preparation and Splitting**: The code starts by preparing the feature matrix X, which includes relevant independent variables (IV), and the target matrix Y, which contains the target variable ("prepayment"). The data is then split into training and testing sets using the train_test_split function from sklearn.
2. **Preprocessing Transformers**: Two preprocessing transformers are defined - one for numeric features and one for categorical features. For numeric features, missing values are imputed using the median and then scaled using StandardScaler. For categorical features, missing values are imputed using the most frequent value, and one-hot encoding is applied using OneHotEncoder.
3. **ColumnTransformer**: A ColumnTransformer named preprocessor is created to apply the appropriate preprocessing transformer to each set of features. It specifies that numeric features should be processed using numeric_transformer, and categorical features should be processed using categorical_transformer.
4. **Multiple Pipelines for Multi-Output Regression**: A list of pipelines is created, where each pipeline contains the preprocessor defined in the ColumnTransformer and a MultiOutputRegressor wrapping a LinearRegression regressor. A separate pipeline is created for each target variable ("prepayment").
5. **Data Reshaping**: The target variable y_train is reshaped using values.reshape(-1, len(Y.columns)) to have two dimensions, necessary for multi-output regression.
6. **Fitting Pipelines to Training Data**: The pipelines are then fitted to the training data (x_train and y_train_reshaped) using a loop. Each pipeline is trained separately for each target variable (column) in Y.

It evaluates the multi-output regression models using R-squared scores, achieving an average R-squared score of approximately 0.9135. The R-squared score for the "prepayment" target variable is 0.9135, indicating a good fit of the model to the data.

It saves the trained multi-output regression model to a pickle file named "Regression.pkl." This allows the model to be easily loaded and used later without retraining. The saved model can be deployed for predicting prepayment amounts for mortgage-backed loans.

It allows the user to input loan application data and generates a DataFrame named data containing the entered rows. The DataFrame includes features such as "DTI," "PPM," "NumBorrowers," "ServicerName," "EverDelinquent," "IsFirstTimeHomebuyer," "CreditRange," and "monthly_income." Each row represents a new loan application.

The code successfully captures user input for loan application data and creates a DataFrame named data to store the entered rows. The user can input multiple loan applications, and the DataFrame is updated accordingly. However, the code does not include any specific analysis or model prediction on the entered data. For a complete report or analysis, the data DataFrame should be used as input for the trained regression model to predict prepayment amounts for each loan application. Additionally, the code could be enhanced to handle data validation and provide a more user-friendly input interface.

**Loading the Model**: The provided code successfully loads a trained multi-output regression model from the pickle file "Regression.pkl" into memory. The loaded model is stored in the variable loaded_model, which can now be used for making predictions on new data.

## Combination of Both Models using Pickle

1. **Data Collection:** The user is prompted to provide input for common features such as "DTI" (Debt-to-Income ratio) and "CreditRange." Additionally, features specific to the classification task ("MSA," "MIP," "OCLTV," etc.) and the regression task ("PPM," "NumBorrowers," "ServicerName," etc.) are also collected.
2. **Classification Prediction:** The loaded classification pipeline, classification_pipeline, is used to predict the classification labels (Default or Non-Default) based on the user input data for classification.
3. **Regression Prediction:** The loaded regression pipeline, regression_pipeline, is utilized to predict the prepayment amounts based on the user input data for regression.
4. **Combination of Predictions:** The predictions from both models are combined into a single DataFrame named combined_predictions. This DataFrame consists of two columns: "Classification_Prediction_Defaulter" holding the classification predictions and "Regression_Prediction_prepayment" holding the regression predictions.
5. **Displaying Results:** The combined_predictions DataFrame is displayed, presenting the combined predictions for each loan application. The "Classification_Prediction_Defaulter" column indicates whether the applicant is classified as a defaulter (1) or not (0), and the "Regression_Prediction_prepayment" column displays the predicted prepayment amounts for each application.

This approach of combining both classification and regression models through pipelines enables comprehensive insights into loan applicant risk (default classification) and potential prepayment amounts for mortgage-backed loans. However, it is important to validate these predictions using real-world data and perform regular model maintenance to ensure the accuracy and reliability of the predictions over time. Additionally, continuous monitoring of model performance and incorporating new data for retraining is recommended to keep the models up-to-date and relevant to changing market dynamics.

# Deployment Process

**Web Server and Platform**: Choose a web server and deployment platform. You can use cloud platforms like AWS, Heroku, or deploy locally on your server.

**Dependencies**: Install required dependencies on the server, including Python, Flask, scikit-learn, pandas, and numpy.

**Upload Files**: Transfer the **app.py**, **index.html**, **static.css**, **classification_.pkl**, **regression_.pkl**, and **combined_predictions.pkl** files to the server.

**Run the App**: Run the Flask app on the server using the command **python app.py**. The app should now be live.

**Access the App**: Access the app through the server's IP address and the port specified in the Flask app (default is usually port 5000).

**User Interaction**: Users can select the prediction type (classification or regression) and input the required features on the app's webpage (index.html).

**Results**: The app sends the user's input to the Flask backend, where it loads the pre-trained classification or regression models using pickle.

**Prediction**: The app uses the loaded models to predict the outcomes based on user input.

**Display**: The app displays the predictions (combined classification and regression) back to the user.

**Feedback and Monitoring**: Collect user feedback on the app's performance and user experience. Monitor the app for errors, performance issues, and resource usage.

Results:

**Classification Prediction**: The classification model predicts whether a loan applicant is likely to be a defaulter or not based on features such as MSA, MIP, OCLTV, OrigUPB, PropertyState, MonthsDelinquent, LTV_Range, and RepPayRange.

**Regression Prediction**: The regression model predicts the prepayment amount of a loan based on features like PPM, NumBorrowers, ServicerName, EverDelinquent, IsFirstTimeHomebuyer, and monthly_income.

**Combined Prediction**: The Flask app combines the results from both models into a single DataFrame and sends it back to the user. The user can view the predictions for both the classification and regression tasks.

**User Experience**: Users can interact with the app through a web interface, eliminating the need for local model execution. The app provides quick and convenient predictions for loan default and prepayment amounts.

**Deployment Observations**: Monitor the app's performance and usage patterns to optimize resource allocation and handle increased traffic.

**Model Accuracy**: Assess the accuracy and performance of the pre-trained models on real user data. Continuously improve the models based on user feedback and data updates.

Overall, the deployed Flask app allows users to get predictions for loan default and prepayment amounts conveniently through a user-friendly web interface. It simplifies the process of model access and provides valuable insights for loan applicants and financial institutions. Continuous monitoring and feedback will help improve the app's accuracy and user experience over time.

# Conclusion

In conclusion, the regression and random forest models performed well and exhibited strong predictive capabilities in the given dataset containing 291452 rows and 44 columns. Both models were able to effectively predict the target variable(s) based on the available features.

The regression model, being a traditional statistical modelling technique, provided accurate predictions by analysing the relationships between the target variable and the independent variables. It considered various factors such as borrower loan characteristics, first time borrower, credit scores etc. to make predictions on prepayment risk. The regression model demonstrated its ability to capture patterns and make accurate predictions based on these factors.

The classification model, being a traditional statistical modeling technique, excelled at making accurate predictions by analyzing the relationships between the target variable and the independent variables. It considered various factors such as borrower loan characteristics, first-time borrower status, credit scores, etc. to classify instances into different predefined categories, such as low, medium, or high prepayment risk. The classification model effectively captured patterns and made reliable predictions based on these factors, allowing it to categorize new instances into the appropriate risk classes with a high degree of accuracy.

Both models proved their effectiveness in predicting the target variable based on the available dataset. Their ability to accurately forecast outcomes showcases their potential to assist in decision- making processes related to loan applications, risk assessment, and financial planning. However, it is crucial to validate and evaluate the models on unseen data and assess their performance metrics, such as accuracy, precision, recall, or mean squared error, to ensure their reliability in real world scenarios.

Further analysis and model evaluation, including cross-validation, testing on external datasets, and comparison with other models, can provide additional insights and strengthen the overall conclusion regarding the performance and reliability of the regression and random forest models.