# **Predictive Modeling for Loan Default Risk Assessment: Leveraging Logistic Regression and Data-Driven Insights**

### Name: Siddhant Sarnobat

### Mail ID: siddhantsarnobat20@gmail.com

# **Table of Contents**

1. Introduction  
   Overview of the project's objective and context.  
   Importance of predicting loan default for financial institutions.
2. Data Exploration and Preprocessing  
   Dataset

Initial exploration of the dataset.  
Handling missing values.  
Dropping unnecessary columns.  
Converting categorical variables to numerical using label encoding.  
Checking for relationships between variables using correlation coefficient.

Checking for Multicollinearity using Variance Inflation Factor (VIF)  
Balancing the dataset using SMOTE.  
Removing outliers using Isolation Forest.  
Normalizing numerical features to ensure consistent scales.

1. Feature Engineering  
   Dividing the dataset into training and testing sets.

Importance of Dataset Splitting

1. Model Training  
   Implementation of logistic regression with regularization.  
   Training the model on the training dataset.  
   Conducting hypothesis testing to evaluate the significance of coefficients.
2. Model Evaluation  
   Testing the trained model on the testing dataset.  
   Calculating evaluation metrics such as accuracy, precision, recall, and F1-score.  
   Generating a classification report for detailed performance analysis.
3. Interpretation and Insights  
   Exploring top reasons for predictions.  
   Analyzing feature importance and their impact on loan default prediction.  
   Discussing implications and actionable insights for financial decision-making.
4. Conclusion  
   Summary of key findings and outcomes.  
   Reflection on the effectiveness of the model.  
   Suggestions for future improvements and research directions.

## Appendix

# **Introduction**

Overview of the Project's Objective and Context  
The objective of this project is to develop a predictive model that can accurately forecast loan defaults for financial institutions. In today's financial landscape, lending institutions face the constant challenge of managing risks associated with loan portfolios. One of the critical risks they encounter is the possibility of borrowers defaulting on their loans, which can lead to significant financial losses and disrupt the stability of the institution.  
  
This project aims to leverage machine learning techniques to build a robust model that can effectively identify the likelihood of loan default based on various borrower attributes and financial indicators. By predicting which borrowers are more likely to default on their loans, financial institutions can proactively take measures to mitigate risks, such as adjusting lending criteria, implementing targeted interventions, or allocating resources more efficiently.  
  
Importance of Predicting Loan Default for Financial Institutions  
Predicting loan default is of paramount importance for financial institutions due to several key reasons:  
  
**Risk Management:** Loan default poses a substantial risk to the financial health and stability of lending institutions. By accurately predicting loan defaults, institutions can proactively manage risks and minimize potential losses.

**Resource Allocation:** Identifying high-risk borrowers allows institutions to allocate resources more effectively. By focusing efforts on monitoring and supporting these borrowers, institutions can improve the overall quality of their loan portfolios.

**Regulatory Compliance:** Regulatory bodies often require financial institutions to maintain certain standards regarding risk management and loan quality. Predictive models for loan default help institutions comply with regulatory requirements and ensure responsible lending practices.

**Customer Relationships:** Loan default can have negative consequences not only for the lender but also for the borrower. By identifying borrowers at risk of default early on, institutions can offer support and guidance to help them avoid financial distress, thereby preserving customer relationships and loyalty.

**Business Sustainability:** Effective risk management practices, including accurate prediction of loan default, contribute to the long-term sustainability and profitability of financial institutions. By reducing losses and maintaining a healthy loan portfolio, institutions can sustain their operations and continue to serve their customers effectively.

In summary, predicting loan default is a critical task for financial institutions to safeguard their interests, maintain regulatory compliance, and ensure the financial well-being of both lenders and borrowers. This project aims to address this challenge by developing a predictive model that enhances risk management practices and supports informed decision-making in the lending process.

# **Data Exploration and Preprocessing**

Dataset:  
Columns: Loan ID, Age, Income, Loan Amount, Credit Score, Months Employed, Num Credit Lines, Interest Rate, Loan Term, DTI Ratio, Education, Employment Type, Marital Status, Has Mortgage, Has Dependents, Loan Purpose, Has CoSigner, Default

Dataset Shape: 255347 rows and 18 columns

Link to dataset: <https://www.kaggle.com/datasets/nikhil1e9/loan-default>

Initial Exploration of the Dataset  
The first step in our analysis involves gaining a comprehensive understanding of the dataset. We begin by loading the dataset and conducting an initial exploration to familiarize ourselves with its structure, features, and values.  
  
Handling Missing Values  
Missing values are a common occurrence in real-world datasets and can significantly impact the performance of machine learning models if not handled appropriately. In this step, we identify and address any missing values in the dataset through imputation or removal, ensuring that our data is complete and ready for analysis.

We have found that there are no null values in any of the column from dataset.  
  
Dropping Unnecessary Columns  
Not all features in the dataset may be relevant or useful for our predictive modeling task. We carefully evaluate each column and decide which ones to retain based on their significance and contribution to our target variable, loan default prediction. Any unnecessary columns are dropped from the dataset to streamline our analysis and reduce noise.

From this dataset have dropped ‘LoanID’ column as it offers no information for our decision making.  
  
Converting Categorical Variables to Numerical using Label Encoding  
Machine learning algorithms require numerical input data, which means categorical variables need to be transformed into a numerical format. We employ label encoding to convert categorical variables into numerical representations while preserving the ordinal relationship between categories, where applicable.  
  
Checking for Relationships Between Variables using Correlation   
Understanding the relationships between variables is crucial for building an effective predictive model. We compute both correlation coefficients to assess the strength and direction of relationships between variables. Correlation analysis helps identify linear relationships.

Checking for Multicollinearity using Variance Inflation Factor (VIF)  
Multicollinearity occurs when independent variables in a regression model are highly correlated with each other, leading to unstable coefficient estimates. We use the Variance Inflation Factor (VIF) to detect multicollinearity among the features in our dataset. High VIF values indicate high multicollinearity, prompting us to take corrective actions such as removing correlated features or employing regularization techniques.

Balancing the Dataset using SMOTE  
  
Imbalanced datasets, where one class significantly outweighs the other, can pose challenges for predictive modeling, particularly in binary classification tasks.

The output ‘df['Default'].value\_counts()’ is counting the occurrences of unique values in the 'Default' column of a data frame.  
So, the output tells us that:  
  
Class 1- 225694 instances  
Class 0- 29653 instances

It is seen that our dataset is unbalanced which can affect our model performance. We address class imbalance by employing Synthetic Minority Over-sampling Technique (SMOTE), which generates synthetic samples for the minority class to achieve a balanced distribution of classes.

We have preferred SMOTE over other resampling techniques because it creates synthetic samples for the minority class, preserving information, effectively handling class imbalance, reducing overfitting, being compatible with various algorithms, and being easy to use.

The state of column ‘Default’ after using resampling technique:  
  
Class 1- 220277 instances  
Class 0- 185972 instances

So, our dataset is balanced.   
  
Removing Outliers using Isolation Forest  
Outliers are data points that deviate significantly from the rest of the dataset and can distort the performance of machine learning models.   
We have utilized Isolation Forest over other outlier removal techniques because it's efficient in high-dimensional datasets, robust to outliers, requires minimal parameter tuning, and is capable of identifying outliers without relying on distance or density measures, making it suitable for various types of data distributions.  
  
Normalizing Numerical Features to Ensure Consistent Scales  
Normalization of numerical features is essential for ensuring that all features contribute equally to the model training process. We standardize the numerical features to have a mean of 0 and a standard deviation of 1, ensuring consistent scales and facilitating convergence during model training.

We do not normalize the categorical columns which have been turned into numerical values. As, label encoded variables might not have a meaningful magnitude or distance relationship, making normalization potentially misleading.  
  
Through these preprocessing steps, we ensure that our dataset is clean, structured, and suitable for training robust predictive models for loan default prediction.

# **Feature Engineering**

Dividing the Dataset into Training and Testing Sets  
Before proceeding with model training and evaluation, it is essential to partition the dataset into separate training and testing sets. This division ensures that the model is trained on a subset of the data and evaluated on an independent subset, enabling unbiased assessment of its performance.

We have chosen the train-test split for its simplicity, clear separation of training and testing data, control over data sizes, faster evaluation, suitability for large datasets, and common industry practice, making it efficient for initial model development and evaluation compared to more complex techniques like k-fold cross-validation.  
  
We randomly split the dataset into training and testing sets, typically allocating a larger portion of the data to training (e.g., 80%) and the remaining portion to testing (e.g., 20%). The training set is used to train the predictive model, while the testing set is used to evaluate its performance on unseen data.  
  
By dividing the dataset into training and testing sets, we facilitate the development and evaluation of our predictive model in a systematic and robust manner. This approach allows us to assess the model's generalization ability and its ability to accurately predict loan default on unseen data, thereby ensuring the reliability and effectiveness of our predictive solution.

Importance of Dataset Splitting  
**Avoiding Overfitting:** Dividing the dataset into training and testing sets helps prevent overfitting, where the model learns to memorize the training data rather than generalizing well to unseen data. By evaluating the model on a separate testing set, we can assess its performance on data it hasn't seen during training.

**Evaluating Generalization:** The testing set serves as a proxy for real-world data. It allows us to evaluate how well our model generalizes to new, unseen instances. A model that performs well on the testing set is more likely to perform well in production or when deployed in real-world scenarios.

**Assessing Model Performance:** Splitting the dataset enables us to quantify the performance of our predictive model using various evaluation metrics. These metrics provide insights into the model's accuracy, precision, recall, F1-score, etc., helping us understand its strengths and weaknesses.

# **Model Training**

Implementation of Logistic Regression with Regularization  
Logistic Regression is a popular machine learning algorithm used for binary classification tasks, such as predicting loan default. We chose Logistic Regression for our binary classification problem due to its simplicity, interpretability, and effectiveness in modeling the probability of a binary outcome. Unlike other complex algorithms, Logistic Regression provides clear insights into how each feature contributes to the classification decision, making it suitable for tasks where understanding the underlying factors is crucial, such as predicting loan default. Additionally, Logistic Regression performs well even with limited data and is less prone to overfitting compared to more complex models, making it a robust choice for real-world applications where data may be noisy or limited.

In this step, we implement Logistic Regression with regularization to train our predictive model. Regularization technique, L2 (Ridge) regularization is used to, help prevent overfitting by penalizing large coefficients and promoting simpler models.  
  
Training the Model on the Training Dataset  
Once the logistic regression model with regularization is implemented, we proceed to train it using the training dataset. During the training process, the model learns the optimal parameters (coefficients) by minimizing a loss function, typically the logistic loss or cross-entropy loss. The training algorithm iteratively updates the model parameters using optimization techniques such as gradient descent until convergence.  
  
By training the model on the training dataset, we aim to capture the underlying patterns and relationships in the data, enabling the model to make accurate predictions on new, unseen data.  
  
Conducting Hypothesis Testing to Evaluate the Significance of Coefficients  
After training the logistic regression model, we conduct hypothesis testing to evaluate the significance of the coefficients associated with each feature. Hypothesis testing helps determine whether the estimated coefficients are statistically different from zero, indicating whether a feature has a significant impact on the prediction of loan default.  
  
We employ statistical tests such as the Wald test to assess the significance of coefficients. These tests provide p-values that indicate the probability of observing the estimated coefficient values under the null hypothesis that the coefficient is zero. Lower p-values suggest stronger evidence against the null hypothesis, indicating greater significance of the coefficient.  
  
By conducting hypothesis testing, we gain insights into the importance of each feature in predicting loan default and identify the most influential factors that contribute to the model's predictive performance. This information guides feature selection, model interpretation, and decision-making in the lending process, enabling financial institutions to make informed decisions and effectively manage risks associated with loan portfolios.

In the output, all features have p-values of 0.0(<0.05), indicating that they are all statistically significant at any reasonable significance level. This suggests strong evidence against the null hypothesis that the coefficients are equal to zero. Therefore, each feature appears to have a significant impact on the target variable in the regression model.

# **Model Evaluation**

Testing the Trained Model on the Testing Dataset  
Once the logistic regression model is trained on the training dataset, the next step is to evaluate its performance on the testing dataset. Testing the model on unseen data provides a robust assessment of its generalization ability and predictive accuracy in real-world scenarios.  
  
We apply the trained model to the testing dataset and generate predictions for loan default. These predictions are compared with the actual labels (true values) in the testing dataset to assess the model's performance across various evaluation metrics.  
  
Calculating Evaluation Metrics such as Accuracy, Precision, Recall, and F1-Score  
To quantify the performance of the logistic regression model, we calculate a range of evaluation metrics that provide insights into different aspects of its predictive performance.

We have chosen precision, recall, F1-score, and accuracy as evaluation metrics for binary classification using logistic regression, as they allow for a comprehensive assessment of the model's performance, considering both false positives and false negatives, as well as the overall correctness of predictions. These metrics are particularly relevant in scenarios such as loan default prediction, where the consequences of misclassification can vary and need to be carefully balanced.  
  
**Accuracy:** The proportion of correctly classified instances out of the total instances in the testing dataset. It provides an overall measure of the model's correctness.

**Precision:** The ratio of true positive predictions to the total number of positive predictions made by the model. It indicates the model's ability to correctly identify positive instances without falsely labeling negative instances as positive.

**Recall (Sensitivity):** The ratio of true positive predictions to the total number of actual positive instances in the testing dataset. It measures the model's ability to capture all positive instances without missing any.

**F1-Score:** The harmonic mean of precision and recall. It provides a balanced measure of the model's precision and recall, taking into account both false positives and false negatives.

We have got the following values for the above evaluation parameters:  
Accuracy: 0.7355323076923077  
Precision: 0.7625525716031109  
Recall: 0.7403563200036499  
F1 Score: 0.7512905391328504

Generating a Classification Report for Detailed Performance Analysis  
In addition to individual evaluation metrics, we generate a comprehensive classification report that summarizes the model's performance across multiple metrics for each class (default and non-default). The classification report typically includes precision, recall, F1-score, and support (number of instances) for each class, along with the overall accuracy and macro/micro-averaged metrics.  
  
The classification report provides detailed insights into the model's performance for different classes and helps identify any class-specific issues, such as imbalances or biases in predictions. It serves as a valuable tool for stakeholders to assess the model's effectiveness, understand its strengths and weaknesses, and make informed decisions regarding its deployment in real-world applications.

The classification report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Recall |
| 0 | 0.71 | 0.73 | 0.72 | 37413 |
| 1 | 0.76 | 0.74 | 0.75 | 43837 |
|  |  |  |  |  |
| Accuracy |  |  | 0.74 | 81250 |
| Macro Avg | 0.73 | 0.74 | 0.73 | 81250 |
| Weighted Avg | 0.74 | 0.74 | 0.74 | 81250 |

Interpretation:  
  
The precision, recall, and F1-score values for both classes are relatively balanced, indicating that the model performs reasonably well for both defaulters and non-defaulters.

The accuracy of 0.74 suggests that the model is able to correctly classify about 74% of all instances, which can be considered acceptable depending on the context and the specific requirements of the application.

By evaluating the logistic regression model using a combination of evaluation metrics and generating a detailed classification report, we gain a comprehensive understanding of its performance in predicting loan default. This information is essential for assessing the model's reliability, identifying areas for improvement, and informing decision-making in the context of lending and risk management.

# **Interpretation and Insights**

Exploring Top Reasons for Predictions  
After evaluating the logistic regression model's performance, we delve deeper into understanding the top reasons behind its predictions. We identify the key features or factors that contribute most significantly to predicting loan default for individual instances in the dataset. By examining the coefficients associated with each feature, we uncover the primary drivers influencing the model's predictions.  
  
Analyzing Feature Importance and their Impact on Loan Default Prediction  
Feature importance analysis provides valuable insights into the relative importance of different features in predicting loan default. We examine the magnitude and direction of coefficients assigned to each feature in the logistic regression model. Features with larger coefficients have a more significant impact on the model's predictions, indicating their importance in distinguishing between default and non-default instances.  
  
Furthermore, we explore the directionality of coefficients to understand how changes in each feature affect the likelihood of loan default. Positive coefficients indicate that higher values of the feature increase the probability of default, while negative coefficients suggest the opposite effect. This analysis helps identify risk factors associated with loan default and informs decision-making in risk assessment and lending practices.

We have identified ‘Age’, ‘InterestRate’ and ‘HasCoSigner’ features as the most influential features in our decision making of the model.  
  
Discussing Implications and Actionable Insights for Financial Decision-Making  
The insights derived from the logistic regression model have significant implications for financial decision-making in lending institutions. By understanding the factors driving loan default predictions, institutions can:  
  
**Refine Lending Criteria:** Incorporate predictive features identified by the model into their lending criteria to better assess borrower risk profiles. Adjusting lending criteria based on data-driven insights enhances the institution's ability to identify and mitigate risks associated with loan default.

**Optimize Risk Management Strategies:** Develop targeted risk management strategies to address specific risk factors identified by the model. Implementing proactive measures such as early intervention programs, tailored financial education, or alternative repayment options can help minimize default rates and improve portfolio performance.

**Enhance Customer Relationships:** Leverage insights from the model to engage with borrowers more effectively and provide personalized support based on their risk profiles. Building trust and fostering positive relationships with borrowers can improve repayment behavior and reduce default rates over time.

**Compliance and Regulatory Considerations:** Ensure compliance with regulatory requirements and industry standards by integrating data-driven risk assessment practices into lending operations. Transparent and responsible lending practices contribute to regulatory compliance and foster trust among stakeholders.

By translating model insights into actionable strategies, lending institutions can enhance their risk management practices, optimize lending decisions, and ultimately achieve better outcomes in managing loan portfolios. The integration of data-driven approaches into financial decision-making processes enables institutions to adapt to evolving market conditions, mitigate risks, and sustain long-term growth and profitability.

# **7. Conclusion**

Summary of Key Findings and Outcomes  
In conclusion, this project aimed to develop a predictive model for loan default prediction using logistic regression with regularization. Through extensive data exploration, preprocessing, model training, and evaluation, several key findings and outcomes were achieved:  
  
**Data Preparation:** The dataset was carefully prepared through handling missing values, dropping unnecessary columns, encoding categorical variables, and addressing multicollinearity, imbalance, and outliers.

**Model Training and Evaluation:** The logistic regression model was trained and evaluated on a testing dataset, achieving satisfactory performance in predicting loan default. Evaluation metrics such as accuracy, precision, recall, and F1-score were calculated to assess the model's effectiveness.

**Interpretation and Insights:** Feature importance analysis revealed significant predictors of loan default, providing valuable insights into the drivers of default risk. Actionable insights were derived for financial decision-making, including refinements to lending criteria, optimization of risk management strategies, and enhancement of customer relationships.

Reflection on the Effectiveness of the Model  
Overall, the logistic regression model demonstrated effectiveness in predicting loan default and provided valuable insights into the underlying factors influencing default risk. By leveraging machine learning techniques and data-driven approaches, we gained a deeper understanding of borrower behavior and risk dynamics in lending.  
  
The model's performance was satisfactory, achieving competitive accuracy and precision rates in predicting loan default. However, there is room for improvement in certain areas, such as enhancing model robustness to outliers and addressing potential biases in the data.  
  
Suggestions for Future Improvements and Research Directions  
Moving forward, several avenues for future improvements and research directions can be explored:  
  
**Model Refinement:** Further optimization of the logistic regression model with advanced regularization techniques and feature engineering methods to improve predictive accuracy and generalization.

**Ensemble Methods:** Exploration of ensemble learning techniques, such as random forests or gradient boosting, to ensemble multiple models for enhanced predictive performance.

**Advanced Analytics:** Integration of advanced analytics techniques, such as deep learning or natural language processing, to leverage unstructured data sources and extract additional insights for loan default prediction.

**Real-Time Monitoring:** Development of real-time monitoring systems to continuously assess borrower risk profiles and adapt lending strategies dynamically in response to changing market conditions.

**Ethical Considerations:** Consideration of ethical and regulatory implications in model development and deployment, including fairness, transparency, and accountability in lending practices.

By embracing these future improvements and research directions, we can further enhance the effectiveness and reliability of predictive models for loan default prediction, ultimately supporting more informed decision-making and risk management in the financial industry.

# **Appendix**

Code:

# Importing Libraries  
  
import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Reading the Data  
  
df= pd.read\_csv('/content/Loan\_default.csv')  
  
# Data Exploration and Preprocessing  
  
df.head()  
  
## Dropping the column - 'LoanID'  
  
df = df.drop(['LoanID'], axis=1)  
  
## Checking Data Types  
  
df.info()  
  
## Checking for null values in dataset  
  
df.isnull().sum()  
  
df.describe()  
  
## Checking values in categorical columns  
  
df['Education'].value\_counts()  
  
df['EmploymentType'].value\_counts()  
  
df['MaritalStatus'].value\_counts()  
  
df['HasMortgage'].value\_counts()  
  
df['HasDependents'].value\_counts()  
  
df['LoanPurpose'].value\_counts()  
  
df['HasCoSigner'].value\_counts()  
  
df['Default'].value\_counts()  
  
## Converting categorical columns into numerical data  
  
from sklearn.preprocessing import LabelEncoder  
  
# List of categorical columns to be label encoded  
categorical\_columns = ['Education', 'EmploymentType', 'MaritalStatus', 'HasMortgage', 'HasDependents', 'LoanPurpose', 'HasCoSigner']  
  
# Initialize LabelEncoder  
label\_encoder = LabelEncoder()  
  
# Iterate through each categorical column and apply label encoding  
for col in categorical\_columns:  
df[col] = label\_encoder.fit\_transform(df[col])  
  
  
# Drop the original categorical columns if needed  
# df.drop(columns=categorical\_columns, inplace=True)  
  
## Checking relation between columns in dataset  
  
correlation\_with\_default = df.corr()['Default'].sort\_values(ascending=False)  
  
print(correlation\_with\_default)  
  
## Checking for collinearity in data  
  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  
from [statsmodels.tools.tools](http://statsmodels.tools.tools/) import add\_constant  
  
# Create a DataFrame containing only the independent variables (features)  
X = df.drop(columns=['Default'])  
  
# Add a constant to the independent variables matrix for intercept calculation  
X = add\_constant(X)  
  
# Calculate VIF for each independent variable  
vif\_data = pd.DataFrame()  
vif\_data["feature"] = X.columns  
vif\_data["VIF"] = [variance\_inflation\_factor(X.values, i) for i in range(X.shape[1])]  
  
# Print VIF values  
print(vif\_data)  
  
  
## Correcting imbalance in dataset and removing outliers from data  
  
from imblearn.over\_sampling import SMOTE  
from sklearn.ensemble import IsolationForest  
  
# Assuming your dataset is stored in a DataFrame called 'df'  
  
# Step 1: Balance the dataset using SMOTE  
X = df.drop('Default', axis=1) # Features  
y = df['Default'] # Target  
  
# Instantiate SMOTE  
smote = SMOTE(random\_state=42)  
X\_resampled, y\_resampled = smote.fit\_resample(X, y)  
  
# Convert back to DataFrame  
df\_resampled = pd.concat([pd.DataFrame(X\_resampled), pd.DataFrame(y\_resampled, columns=['Default'])], axis=1)  
  
# Step 2: Remove outliers using Isolation Forest  
# Assuming your dataset is already scaled appropriately  
  
# Instantiate Isolation Forest  
isolation\_forest = IsolationForest(contamination=0.1, random\_state=42)  
  
# Fit Isolation Forest  
outlier\_preds = isolation\_forest.fit\_predict(df\_resampled.drop('Default', axis=1))  
  
# Filter outliers  
df\_no\_outliers = df\_resampled[outlier\_preds != -1]  
  
# Separate target variable from features  
X\_no\_outliers = df\_no\_outliers.drop('Default', axis=1)  
y\_no\_outliers = df\_no\_outliers['Default']  
  
# Concatenate features and target variable  
df\_final = pd.concat([X\_no\_outliers, y\_no\_outliers], axis=1)  
  
# Now, df\_final contains your balanced dataset without outliers, including the target variable  
  
  
df\_final['Default'].value\_counts()  
  
## Normalizing the dataset  
  
from sklearn.preprocessing import StandardScaler  
  
# Assuming df\_final is your final dataset  
  
# Columns to be standardized  
columns\_to\_standardize = ['Age', 'Income', 'LoanAmount', 'CreditScore', 'MonthsEmployed', 'NumCreditLines', 'InterestRate',  
'LoanTerm', 'DTIRatio']  
  
# Instantiate StandardScaler  
scaler = StandardScaler()  
  
# Standardize selected columns  
df\_final[columns\_to\_standardize] = scaler.fit\_transform(df\_final[columns\_to\_standardize])  
  
# Now, df\_final contains standardized numerical columns  
  
  
## Dividing the data  
  
from sklearn.model\_selection import train\_test\_split  
  
# Assuming df\_final is your final dataset  
  
# Splitting into features (X) and target variable (y)  
X = df\_final.drop('Default', axis=1)  
y = df\_final['Default']  
  
# Splitting the dataset into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Now, you have X\_train (features for training), X\_test (features for testing), y\_train (target variable for training), and y\_test (target variable for testing)  
  
  
# Training the model  
  
import numpy as np  
  
class LogisticRegressionWithRegularization:  
def \_\_init\_\_(self, learning\_rate=0.01, num\_iterations=1000, lambda\_val=0.01):  
self.learning\_rate = learning\_rate  
self.num\_iterations = num\_iterations  
self.lambda\_val = lambda\_val  
self.weights = None  
self.bias = None  
  
def sigmoid(self, z):  
return 1 / (1 + np.exp(-z))  
  
def fit(self, X, y):  
num\_samples, num\_features = X.shape  
self.weights = np.zeros(num\_features)  
self.bias = 0  
  
# gradient descent  
for \_ in range(self.num\_iterations):  
# linear model  
linear\_model = np.dot(X, self.weights) + self.bias  
# sigmoid function  
y\_predicted = self.sigmoid(linear\_model)  
  
# compute gradients with regularization  
dw = (1 / num\_samples) \* (np.dot(X.T, (y\_predicted - y)) + 2 \* self.lambda\_val \* self.weights)  
db = (1 / num\_samples) \* np.sum(y\_predicted - y)  
  
# update parameters  
self.weights -= self.learning\_rate \* dw  
self.bias -= self.learning\_rate \* db  
  
def predict(self, X):  
linear\_model = np.dot(X, self.weights) + self.bias  
y\_predicted = self.sigmoid(linear\_model)  
y\_predicted\_cls = [1 if i > 0.5 else 0 for i in y\_predicted]  
return y\_predicted\_cls  
  
  
log\_reg = LogisticRegressionWithRegularization()  
log\_reg.fit(X\_train, y\_train)  
  
# Hypothesis Testing  
  
# Define hypothesis testing function  
def wald\_test(model, X, y):  
# Get coefficient estimates and their standard errors  
coef = model.weights  
num\_samples, num\_features = X.shape  
y\_predicted = model.predict(X)  
residuals = y\_predicted - y  
sigma\_squared = np.dot(residuals, residuals) / (num\_samples - num\_features - 1)  
cov\_matrix = np.linalg.inv(np.dot(X.T, X)) \* sigma\_squared  
  
# Compute z-statistics  
z\_stat = coef / np.sqrt(np.diag(cov\_matrix))  
  
# Compute Wald statistic  
wald\_stat = z\_stat \*\* 2  
  
# Compute p-values  
p\_values = 1 - chi2.cdf(wald\_stat, df=1)  
  
return {'Coefficient': coef.flatten(), 'Standard Error': np.sqrt(np.diag(cov\_matrix)), 'Z-Statistic': z\_stat, 'Wald Statistic': wald\_stat, 'P-Value': p\_values}  
  
  
# Perform hypothesis testing  
results = wald\_test(log\_reg, X\_train, y\_train)  
  
# Print results  
print("Hypothesis Testing Results:")  
print("{:<20} {:<20} {:<20} {:<20} {:<20}".format('Feature', 'Coefficient', 'Standard Error', 'Z-Statistic', 'P-Value'))  
for i in range(len(log\_reg.weights)):  
print("{:<20} {:<20} {:<20} {:<20} {:<20}".format(f'Feature {i}', results['Coefficient'][i], results['Standard Error'][i], results['Z-Statistic'][i], results['P-Value'][i]))  
  
# Testing effectiveness of the model  
  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score  
from sklearn.metrics import classification\_report  
  
# Predict on test data  
y\_pred = log\_reg.predict(X\_test)  
  
# Calculate accuracy  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("Accuracy:", accuracy)  
  
# Calculate precision  
precision = precision\_score(y\_test, y\_pred)  
print("Precision:", precision)  
  
# Calculate recall  
recall = recall\_score(y\_test, y\_pred)  
print("Recall:", recall)  
  
# Calculate F1 score  
f1 = f1\_score(y\_test, y\_pred)  
print("F1 Score:", f1)  
  
  
# Generate classification report  
class\_report = classification\_report(y\_test, y\_pred)  
  
# Print the classification report  
print("Classification Report:")  
print(class\_report)  
  
# Exploring top 3 resons for prediction along with its values  
  
# Assuming df\_final is your final dataset and y\_pred contains your predictions  
  
# Define feature names (replace these with your actual feature names)  
feature\_names = ['Age', 'Income', 'LoanAmount', 'CreditScore', 'MonthsEmployed', 'NumCreditLines', 'InterestRate', 'LoanTerm', 'DTIRatio', 'Education', 'EmploymentType', 'MaritalStatus', 'HasMortgage', 'HasDependents', 'LoanPurpose', 'HasCoSigner']  
  
# Initialize an empty DataFrame to store top reasons  
top\_reasons\_df = pd.DataFrame(columns=['Prediction', 'Top Reason 1', 'Value 1', 'Top Reason 2', 'Value 2', 'Top Reason 3', 'Value 3'])  
  
# Iterate through the last 10 predictions and extract top three reasons for each  
for i, prediction in enumerate(y\_pred[-10:], start=len(y\_pred)-10):  
# Get coefficients from the logistic regression model for this prediction  
coefficients = log\_reg.weights  
  
# Create a dictionary to map feature names to coefficients  
feature\_coefficients = dict(zip(feature\_names, coefficients))  
  
# Sort the features based on their coefficients  
sorted\_features = sorted(feature\_coefficients.items(), key=lambda x: abs(x[1]), reverse=True)  
  
# Extract top three reasons  
top\_three\_reasons = sorted\_features[:3]  
  
# Extract top three reasons and their corresponding values  
top\_three\_reasons\_values = [(feature, coefficient, df\_final.iloc[i][feature]) for feature, coefficient in top\_three\_reasons]  
  
# Store top three reasons in DataFrame  
row\_values = [prediction]  
for j in range(3):  
if j < len(top\_three\_reasons\_values):  
reason, coefficient, value = top\_three\_reasons\_values[j]  
row\_values.extend([reason, value])  
else:  
row\_values.extend(['', ''])  
top\_reasons\_df.loc[i] = row\_values  
  
# Print or use top\_reasons\_df as needed  
print(top\_reasons\_df)