Final Project Report Template

1. INTRODUCTION

1.1 Project Overview

The 'Smart Lender: Applicant Credibility Prediction for Loan Approval' project aims to assist banks and financial institutions in predicting whether a loan applicant is likely to be eligible for loan approval based on their financial profile. The project leverages machine learning models to analyze key factors, such as income, credit history, loan amount, and more, to determine the likelihood of an applicant's loan approval.

This solution helps banks reduce risks associated with non-performing loans and make more informed lending decisions. By predicting the credibility of applicants, the model provides a faster and more efficient process for loan approval, benefiting both banks and customers. The system can be scaled to work with multiple loan types, making it adaptable to different banking needs.

1.2 Objectives

The main objectives of this project are:

- Accurate Loan Eligibility Prediction: Build a machine learning model capable of predicting whether an applicant will likely default or be eligible for a loan.

- Minimize Credit Risk: Reduce financial losses for banks by identifying potential defaulters, thereby improving the loan approval process and ensuring only credible applicants receive loans.

- Improve Decision-making Process: Provide a system that assists in the fast, accurate, and reliable decision-making process for loan approval.

- Increase Efficiency: Automate the loan approval process, reducing manual intervention, speeding up decision times, and improving customer satisfaction.

2. Project Initialization and Planning Phase

2.1 Define Problem Statement

Date	01 October 2024
Team ID	LTVIP2024TMID24947
Project Name	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	3 Marks

Define Problem Statements (Customer Problem Statement Template):

Financial institutions and **lending companies** struggle with making quick, reliable, and unbiased decisions when evaluating loan applicants. With an increasing volume of loan applications, manual review processes often lead to delays, errors, and inconsistent approvals. Lenders need to identify credible applicants with a high likelihood of timely repayment to minimize risk and defaults.

Applicants are also frustrated with the slow and opaque loan approval process. They need quicker decisions and more transparency about how their eligibility is evaluated.

SmartLender aims to solve this problem by providing an automated system that evaluates applicants based on multiple factors like their number of dependents, education level, employment status, loan amount, loan term, CIBIL score, and assets. This model will ensure fast, fair, and accurate loan approval decisions, helping both the lender and applicant experience a smoother loan approval process.

Problem Statement (PS)

PS	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-2	Applicant seeking a loan.	Get approved for a personal loan.	I have dependents and an uncertain employment history.	I am self-employed but have a good credit score (CIBIL score) and valuable assets.	Hopeful for loan approval.

2.2 Project Proposal (Proposed Solution)

Date	01 October 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	3 Marks

Project Proposal (Proposed Solution) report

The proposal report aims to transform loan approval using machine learning, boosting efficiency and accuracy. It tackles system inefficiencies, promising better operations, reduced risks, and happier customers. Key features include a machine learning-based credit model and real-time decision-making.

Project Overview	
Objective	The primary objective is to revolutionize the loan approval process by implementing advanced machine learning techniques, ensuring faster and more accurate assessments.
Scope	The project comprehensively assesses and enhances the loan approval process, incorporating machine learning for a more robust and efficient system.
Problem Statement	
Description	Addressing inaccuracies and inefficiencies in the current loan approval system adversely affects operational efficiency and customer satisfaction.
Impact	Solving these issues will result in improved operational efficiency, reduced risks, and an overall enhancement in the lending process, contributing to customer satisfaction and organizational success.
Proposed Solution	
Approach	Employing machine learning techniques to analyze and predict creditworthiness, creating a dynamic and adaptable loan approval system.
Key Features	- Implementation of a machine learning-based credit assessment model.
Key Features	- Real-time decision-making for quicker loan approvals. - Continuous learning to adapt to evolving financial landscapes.

Resource Requirements

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	T4 GPU
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, numpy, matplotlib, seaborn
Development Environment	IDE	Google colab Notebook, vscode
Data		
Data	Source, size, format	Kaggle dataset, 4269, csv

2.3. Initial Project Planning

Date	28-09-2024
Team ID	LTVIP2024TMID24947
Project Name	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	4 Marks

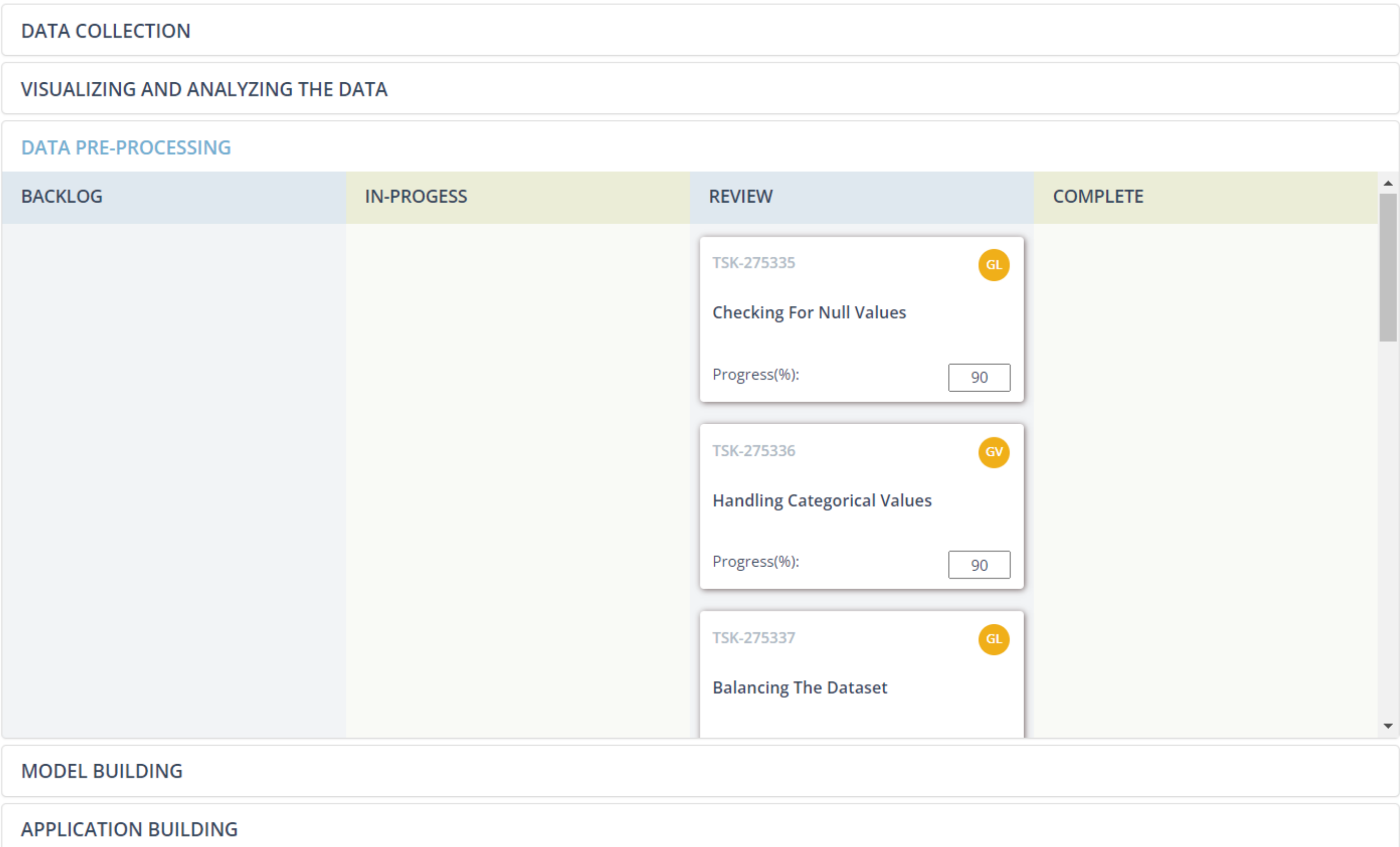
Product Backlog, Task Schedule, and Estimation

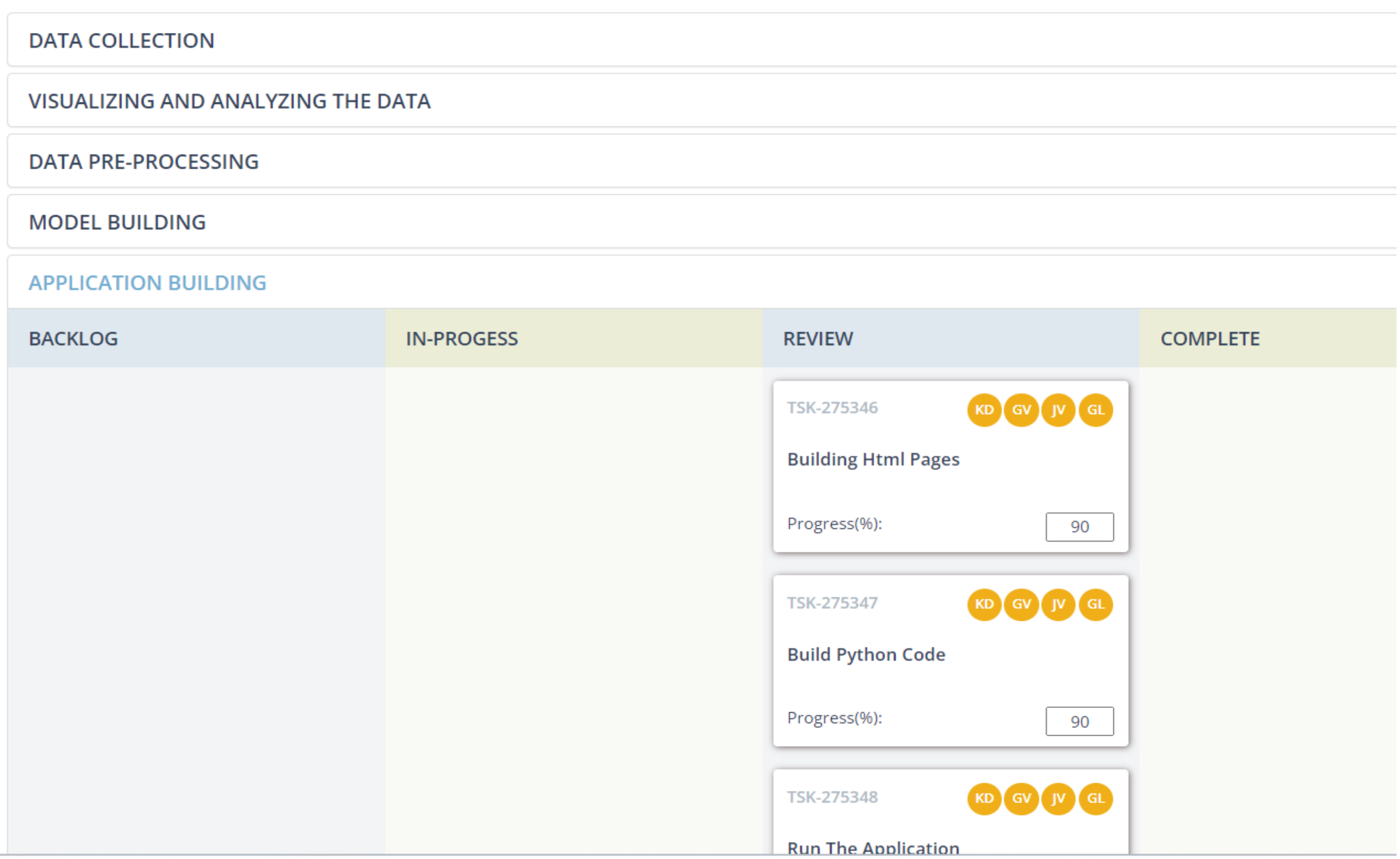
Use the below template to create a product backlog and Task schedule

TASKS	**Functional Requirement (Epic)**	**User Story Number/ Task no**	**User Story / Task**	**Priority**	**Team Members**	**Task Start Date**	**Task End Date (Planned)**
Task-1	Data Collection	TSk-275328	Download the dataset	Low	nivas	2024/09/20	2024/09/22
Task-2	Visualization and analyzing the data	TSK-275329	Importing the libraries	Low	nivas	2024/09/20	2024/09/22
Task-2	Visualization and analyzing the data	TSK-275330	Read the dataset	Medium	nivas	2024/09/24	2024/09/29
Task-2	Visualization and analyzing the data	TSK-275331	Univariant analysis	Medium	surya	2024/09/24	2024/09/29
Task-2	Visualization and analyzing the data	TSK-275332	Bi variant analysis	Medium	surya	2024/09/24	2024/09/29
Task-2	Visualization and analyzing the data	TSK-275333	Multi variant analysis	Medium	surya	2024/09/24	2024/09/29
Task-2	Visualization and analyzing the data	TSK-275334	Descriptive analysis	Low	venkatesh	2024/09/29	2024/10/01
Task-3	Data Pre - Processing	TSK-275335	Check null values	High	nivas	2024/09/29	2024/10/02
Task-3	Data Pre-Processing	TSK-275336	Handling Categorial Values	High	venkatesh	2024/10/01	2024/10/03
Task-3	Data Pre - Processing	TSK-275337	Balancing the data	High	nivas	2024/10/02	2024/10/04
Task-3	Data Pre - Processing	TSK-275338	Scaling the data	Medium	naik	2024/10/03	2024/10/05
Task-3	Data Pre - Processing	TSK-275339	Splitting Data into Train and Test	Medium	surya	2024/10/04	2024/10/05

Tasks	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members	Task Start Date	Task End Date (Planned)
Task-4	Model Building	TSK-275340	Decision tree model	High	surya	2024/10/05	2024/10/07
Task-4	Model Building	TSK-275341	Random forest model	High	nivas	2024/10/06	2024/10/07
Task-4	Model Building	TSK-275342	KNN model	High	naik	2024/10/07	2024/10/08
Task-4	Model Building	TSK-275343	Xgboost Model	High	venkatesh	2024/10/08	2024/10/08
Task-4	Model Building	TSK-275344	Compare the model	low	surya	2024/10/09	2024/10/08
Task-4	Model Building	TSK-275345	Evaluating performance of the model and saving the model	low	surya	2024/10/09	2024/10/08
Task-5	Application building	TSK-275346	Building the html pages	high	nivas	2024/10/09	2024/10/13
Task-5	Application building	TSK-275347	Build python code	high	nivas	2024/10/10	2024/10/15
Task-5	Application building	TSK-275348	Run the application	low	nivas	2024/10/10	2024/10/15

Screenshots:





3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

Date	3 october 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	2 Marks

Data Collection Plan & Raw Data Sources Identification Report:

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan:

Section	**Description**
Project Overview	The machine learning project **SmartLender** aims to solve this problem by providing an automated system that evaluates applicants based on multiple factors like their number of dependents, education level, employment status, loan amount, loan term, CIBIL score, and assets. This model will ensure fast, fair, and accurate loan approval decisions, helping both the lender and applicant experience a smoother loan approval process.

Data Collection Plan	• Search for datasets related to loan approvals, financial information, and applicant details. • Prioritize datasets with diverse demographic information.
Raw Data Sources Identified	The raw data sources for this project include datasets obtained from Kaggle the popular platforms for data science competitions and repositories. The provided sample data represents a subset of the collected information, encompassing variables such as their number of dependents, education level, employment status, loan amount, loan term, CIBIL score, and assets.

Raw Data Sources Report:

Source Name	Description	Location/URL	Format	Size	Access Permissions
Kaggle Dataset	The dataset comprises applicant details (gender, marital status), financial metrics (income, loan amount), and loan approval outcomes.	https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset/data	CSV	384 KB	Public

3.2. Data Quality Report

Date	03 October 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	2 Marks

Data Quality Report:

The Data Quality Report will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

Data Quality Report:

Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset	Categorical data in the dataset	Moderate	encoding has to be done in the data.

3.3. Data Exploration and Preprocessing

Date	03 October 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	6 Marks

Data Exploration and Preprocessing Report

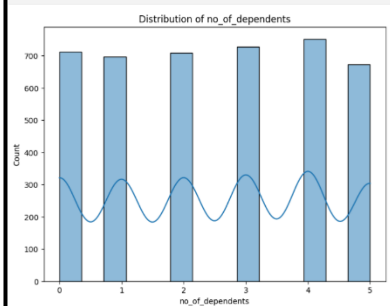
Dataset variables will be statistically analyzed to identify patterns and outliers, with Python

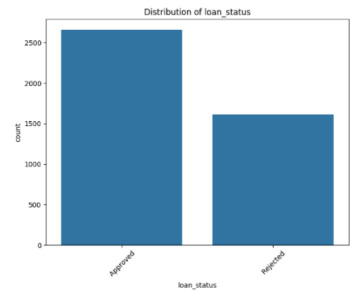
employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for insights and predictions.

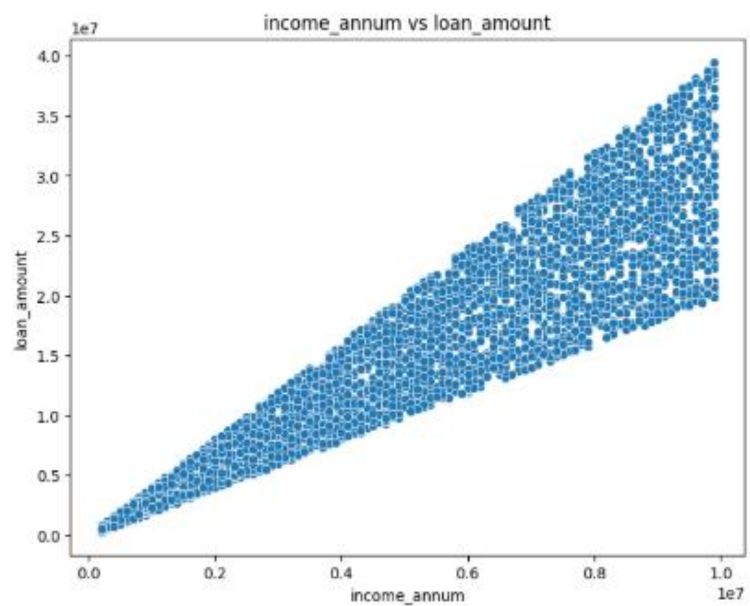
<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Data Overview</td><td>
<u>Dimension: </u>4269 rows × 12 columns
<u>Descriptive statistics:</u>

```python
data.describe()
```

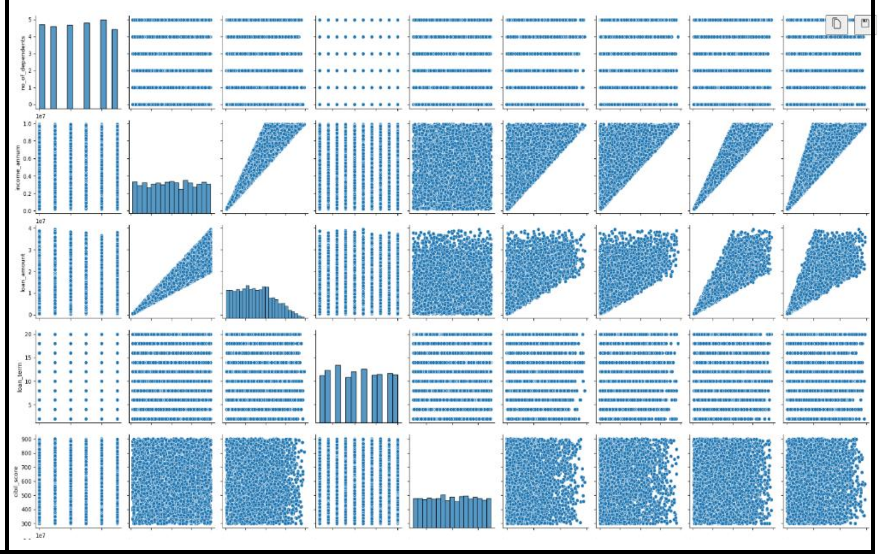
<table>
<tr><th></th><th>no_of_dependents</th><th>income_annum</th><th>loan_amount</th><th>loan_term</th><th>cibil_score</th><th>residential_assets_value</th><th>commercial_assets_value</th><th>luxury_assets_value</th><th>bank_asset_value</th></tr>
<tr><td>count</td><td>4269.000000</td><td>4.269000e+03</td><td>4.269000e+03</td><td>4269.000000</td><td>4269.000000</td><td>4.269000e+03</td><td>4.269000e+03</td><td>4.269000e+03</td><td>4.269000e+03</td></tr>
<tr><td>mean</td><td>2.498712</td><td>5.059124e+06</td><td>1.513345e+07</td><td>10.900445</td><td>599.936051</td><td>7.472617e+06</td><td>4.973155e+06</td><td>1.512631e+07</td><td>4.976692e+06</td></tr>
<tr><td>std</td><td>1.695910</td><td>2.806840e+06</td><td>9.043363e+06</td><td>5.709187</td><td>172.430401</td><td>6.503637e+06</td><td>4.388966e+06</td><td>9.103754e+06</td><td>3.250185e+06</td></tr>
<tr><td>min</td><td>0.000000</td><td>2.000000e+05</td><td>3.000000e+05</td><td>2.000000</td><td>300.000000</td><td>-1.000000e+05</td><td>0.000000e+00</td><td>3.000000e+05</td><td>0.000000e+00</td></tr>
<tr><td>25%</td><td>1.000000</td><td>2.700000e+06</td><td>7.700000e+06</td><td>6.000000</td><td>453.000000</td><td>2.200000e+06</td><td>1.300000e+06</td><td>7.500000e+06</td><td>2.300000e+06</td></tr>
<tr><td>50%</td><td>3.000000</td><td>5.100000e+06</td><td>1.450000e+07</td><td>10.000000</td><td>600.000000</td><td>5.600000e+06</td><td>3.700000e+06</td><td>1.460000e+07</td><td>4.600000e+06</td></tr>
<tr><td>75%</td><td>4.000000</td><td>7.500000e+06</td><td>2.150000e+07</td><td>16.000000</td><td>748.000000</td><td>1.130000e+07</td><td>7.600000e+06</td><td>2.170000e+07</td><td>7.100000e+06</td></tr>
<tr><td>max</td><td>5.000000</td><td>9.900000e+06</td><td>3.950000e+07</td><td>20.000000</td><td>900.000000</td><td>2.910000e+07</td><td>1.940000e+07</td><td>3.920000e+07</td><td>1.470000e+07</td></tr>
</table>
</td></tr>
<tr><td>Univariate Analysis</td><td></td></tr>
</table>

<table>
<tr>
<td></td>
<td>




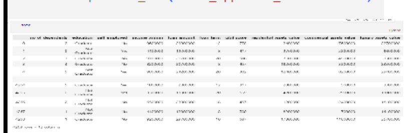
</td>
</tr>
<tr>
<td>Bivariate Analysis</td>
<td>


</td>
</tr>
</table>

<table>
<tr><td>Multivariate Analysis</td><td></td></tr>
</table>

<table>
<tr><td>Outliers and Anomalies</td><td>-</td></tr>
<tr><td colspan="2">Data Preprocessing Code Screenshots</td></tr>
<tr><td>Loading Data</td><td>

```
data = pd.read_csv('loan_approval_dataset.csv')
```


</td></tr>
<tr><td>Data Transformation</td><td>

```
data.loan_status.unique()
```

```
array([' Approved', ' Rejected'], dtype=object)
```

```
data.loan_status = data.loan_status.apply(clean_data)
```

```
data.loan_status.unique()
```

```
array(['Approved', 'Rejected'], dtype=object)
```

```
data.loan_status = data.loan_status.replace(['Approved', 'Rejected'],[1,0])
```

```
data.self_employed = data.self_employed.apply(clean_data)
```

```
data.self_employed.unique()
```

```
array(['No', 'Yes'], dtype=object)
```

```
data.self_employed = data.self_employed.replace(['No', 'Yes'],[0,1])
```

```
thon-input-30-a5f9be13468d>:1: FutureWarning: Downcasting behavior in `replace` is deprecatec
```
</td></tr>
</table>

Balancing the data	
Feature Engineering	Attached the codes in final submission.
Save Processed Data	-

4.Model Development Phase

4.1. Feature Selection Report

Date	03 October 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	5 Marks

Feature Selection Report Template

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
Loan_ID	Unique identifier for each loan applicant	No	For predicting the loan, a Loan ID is not required.
Dependents	Number of dependents	Yes	Indicates financial responsibilities and influences loan capacity.
Self_Employed	Self-employme nt status	Yes	Self-employed individuals may have different financial profiles.
Income in annum	Income of the applicant in a year	Yes	It is crucial in determining the applicant's financial capacity.
Loan Amount	Amount of loan applied	Yes	Fundamental for assessing the financial magnitude of the loan.
Loan Term	Term of the loan (in years)	Yes	The loan term influences monthly repayments and impacts eligibility.
Cibil score	Cibil score of the applicant	Yes	A major factor in loan approval is reflecting the applicant's creditworthiness.
Assets	Assets of applicant	Yes	It is crucial in determining the applicant's financial capacity.
Loan_Status	Loan approval outcome	Yes	The target variable for predictive modeling – is essential for the project's goal.

4.2. Model Selection Report

Date	03 October 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	6 Marks

Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model	**Description**	**Hyperparameters**	**Performance Metric (e.g., Accuracy, F1 Score)**
Random Forest	Ensemble of decision trees; robust, handles complex relationships, reduces overfitting, and provides feature importance for loan approval prediction.	-	Accuracy score = 97%
Decision Tree	Simple tree structure; interpretable, captures non-linear relationships, suitable for initial insights into loan approval patterns.	-	Accuracy score = 91%
KNN	Classifies based on nearest neighbors; adapts well to data patterns, effective	-	Accuracy score = 96%

	for local variations in loan approval criteria.		
Xg boost Gradient Boosting	Gradient boosting with trees; optimizes predictive performance, handles complex relationships, and is suitable for accurate loan approval predictions.	-	Accuracy score = 97%

4.3. Initial Model Training Code, Model Validation and Evaluation Report

Date	03 October 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot.

The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
#importing and building the random forest model
def RandomForest(X_tarin,X_test,y_train,y_test):
    model = RandomForestClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```

```
#printing the train accuracy and test accuracy respectively
RandomForest(X_train,X_test,y_train,y_test)
```

```
#importing and building the Decision tree model
def decisionTree(X_train,X_test,y_train,y_test):
    model = DecisionTreeClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```

```
#printing the train accuracy and test accuracy respectively
decisionTree(X_train,X_test,y_train,y_test)
```

```
#importing and building the KNN model
def KNN(X_train,X_test,y_train,y_test):
    model = KNeighborsClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```

```
#printing the train accuracy and test accuracy respectively
KNN(X_train,X_test,y_train,y_test)
```

```
#importing and building the KNN model
def KNN(X_train,X_test,y_train,y_test):
    model = KNeighborsClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))
```

```
#printing the train accuracy and test accuracy respectively
KNN(X_train,X_test,y_train,y_test)
```

Model Validation and Evaluaion Report:

Model	Classification Report	F1 Score	Confusion Matrix
Random Forest	Classification Report: precision recall f1-score support 0 0.97 0.95 0.96 319 1 0.97 0.98 0.98 535 accuracy 0.97 854 macro avg 0.97 0.97 0.97 854 weighted avg 0.97 0.97 0.97 854	98%	Confusion Matrix: [[304 15] [11 524]]
Decision Tree	Decision Tree Classification Report: precision recall f1-score support 0 0.98 0.99 0.99 313 1 0.99 0.99 0.99 541 accuracy 0.99 854 macro avg 0.99 0.99 0.99 854 weighted avg 0.99 0.99 0.99 854	98%	Decision Tree Confusion Matrix: [[310 3] [5 536]]
KNN	KNN Classification Report: precision recall f1-score support 0 0.87 0.93 0.90 313 1 0.96 0.92 0.94 541 accuracy 0.92 854 macro avg 0.91 0.92 0.92 854 weighted avg 0.93 0.92 0.92 854	92%	KNN Confusion Matrix: [[290 23] [42 499]]
Gradient Boosting	XGBoost Classification Report: precision recall f1-score support 0 0.98 0.99 0.98 313 1 0.99 0.99 0.99 541 accuracy 0.99 854 macro avg 0.99 0.99 0.99 854 weighted avg 0.99 0.99 0.99 854	98%	XGBoost Confusion Matrix: [[309 4] [7 534]]

5.Model Optimization and Tuning Phase

5.1. Hyperparameter Tuning Documentation

Date	03 October 2024
Team ID	LTVIP2024TMID24947
Project Title	SmartLender - Applicant Credibility Prediction for Loan Approval
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values

Decision Tree	# Hyperparameter tuning for Decision Tree
	param_grid_dt = {
	'max_depth': [None, 5, 10], # Limit max depth
	'min_samples_split': [2, 5] # Limit minimum samples to split
	}
	
	dt_grid = GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=param_grid_dt, cv=3, scoring='accuracy', n_jobs=-1)
	dt_grid.fit(x_train_scaled, y_train)
	
	print("Best Decision Tree Parameters:", dt_grid.best_params_)
	print("Best Decision Tree Accuracy:", dt_grid.best_score_)
		Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}
		Best Random Forest Accuracy: 0.9795041642814564
		Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}
		Best KNN Accuracy: 0.918009713656467
		Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}
		Best Decision Tree Accuracy: 0.981259833367022
		Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
		Best XGBoost Accuracy: 0.9830165311147149

Random Forest	# Hyperparameter tuning for Random Forest
	param_grid_rf = {
	'n_estimators': [50, 100], # Reduced number of estimators
	'max_depth': [None, 10], # Limit max depth
	'min_samples_split': [2, 5] # Limit minimum samples to split
	}
	
	rf_grid = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid_rf, cv=3, scoring='accuracy', n_jobs=-1)
	rf_grid.fit(x_train_scaled, y_train)
	
	print("Best Random Forest Parameters:", rf_grid.best_params_)
	print("Best Random Forest Accuracy:", rf_grid.best_score_)
		Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}
		Best Random Forest Accuracy: 0.9795041642814564
		Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}
		Best KNN Accuracy: 0.918009713656467
		Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}
		Best Decision Tree Accuracy: 0.981259833367022
		Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
		Best XGBoost Accuracy: 0.9830165311147149

KNN	# Hyperparameter tuning for KNN
	param_grid_knn = {
	    'n_neighbors': [3, 5, 7], # Reduced number of neighbors
	    'weights': ['uniform', 'distance'] # Explore different weighting schemes
	}
	
	knn_grid = GridSearchCV(estimator=KNeighborsClassifier(), param_grid=param_grid_knn, cv=3, scoring='accuracy', n_jobs=-1)
	knn_grid.fit(x_train_scaled, y_train)
	
	print("Best KNN Parameters:", knn_grid.best_params_)
	print("Best KNN Accuracy:", knn_grid.best_score_)
		Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}
		Best Random Forest Accuracy: 0.9795041642814564
		Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}
		Best KNN Accuracy: 0.918009713656467
		Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}
		Best Decision Tree Accuracy: 0.981259833367022
		Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
		Best XGBoost Accuracy: 0.9830165311147149

XG Boost	# Hyperparameter tuning for XGBoost
	param_grid_xgb = {
	    'n_estimators': [50, 100], # Reduced number of estimators
	    'max_depth': [3, 5], # Limit max depth
	    'learning_rate': [0.1, 0.01] # Explore different learning rates
	}
	
	xgb_grid = GridSearchCV(estimator=xgb.XGBClassifier(), param_grid=param_grid_xgb, cv=3, scoring='accuracy', n_jobs=-1)
	xgb_grid.fit(x_train_scaled, y_train)
	
	print("Best XGBoost Parameters:", xgb_grid.best_params_)
	print("Best XGBoost Accuracy:", xgb_grid.best_score_)
		Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}
		Best Random Forest Accuracy: 0.9795041642814564
		Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}
		Best KNN Accuracy: 0.918009713656467
		Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}
		Best Decision Tree Accuracy: 0.981259833367022
		Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
		Best XGBoost Accuracy: 0.9830165311147149

5.2 Performance Metrics Comparison Report:

Model	**Optimized Metric**
Decision Tree	Best Decision Tree Confusion Matrix: [[310 6] [8 530]] Best Decision Tree Classification Report: precision recall f1-score support 0 0.97 0.98 0.98 316 1 0.99 0.99 0.99 538 accuracy 0.98 854 macro avg 0.98 0.98 0.98 854 weighted avg 0.98 0.98 0.98 854

<table>
<tr>
<td>Random Forest</td>
<td><pre>
Best Random Forest Confusion Matrix:
[[308   8]
 [14 524]]
Best Random Forest Classification Report:
              precision   recall f1-score   support

           0   0.96   0.97   0.97   316
           1   0.98   0.97   0.98   538

    accuracy   0.97   854
   macro avg   0.97   0.97   0.97   854
weighted avg   0.97   0.97   0.97   854
</pre></td>
</tr>
<tr>
<td>KNN</td>
<td><pre>
Best KNN Confusion Matrix:
[[288 28]
 [43 495]]
Best KNN Classification Report:
              precision   recall f1-score   support

           0   0.87   0.91   0.89   316
           1   0.95   0.92   0.93   538

    accuracy   0.92   854
   macro avg   0.91   0.92   0.91   854
weighted avg   0.92   0.92   0.92   854
</pre></td>
</tr>
<tr>
<td>XG Boost</td>
<td><pre>
Best XGBoost Confusion Matrix:
[[311   5]
 [10 528]]
Best XGBoost Classification Report:
              precision   recall f1-score   support

           0   0.97   0.98   0.98   316
           1   0.99   0.98   0.99   538

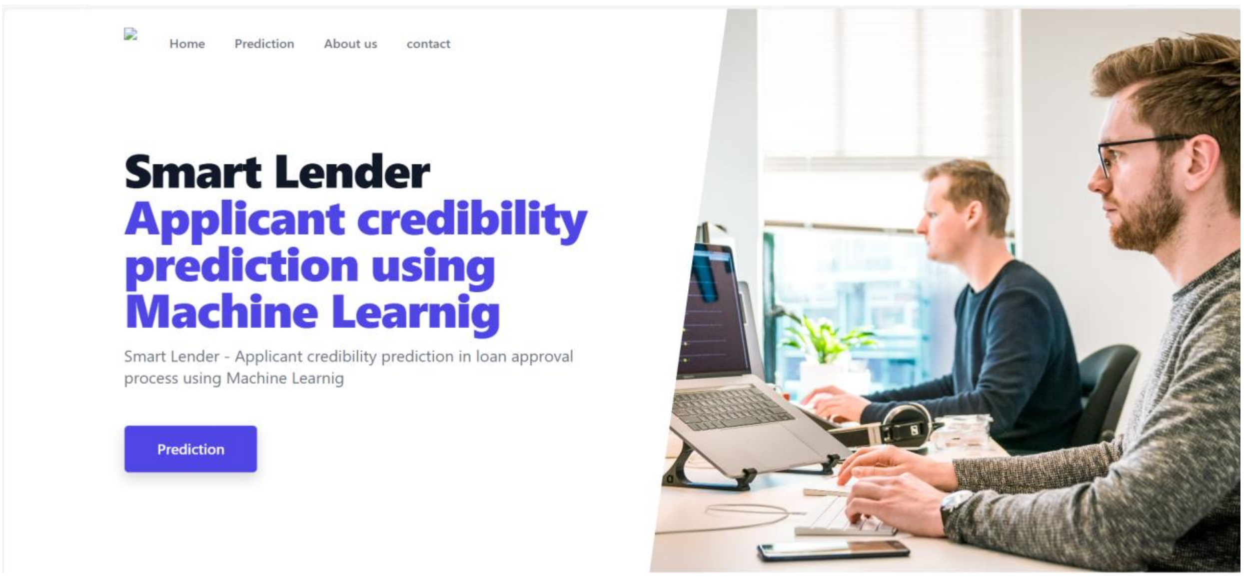
    accuracy   0.98   854
   macro avg   0.98   0.98   0.98   854
weighted avg   0.98   0.98   0.98   854
</pre></td>
</tr>
</table>

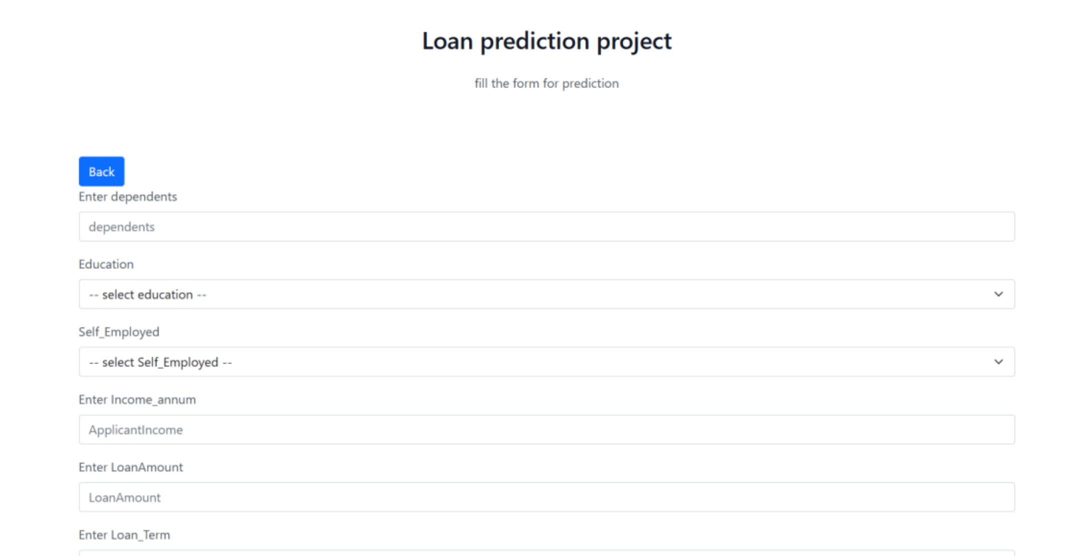
5.3 Final Model Selection Justification:

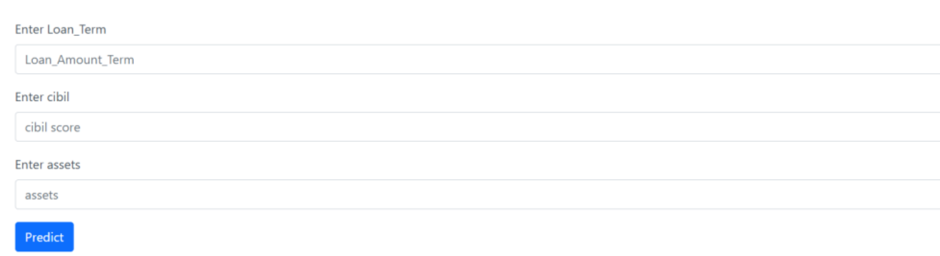
Final Model	Reasoning
Random forest	After evaluating the models based on several metrics such as **accuracy**, **precision**, **recall**, and **F1-score**, all models demonstrated good performance. However, **Random Forest (RF)** was selected as the final model due to its combination of accuracy, robustness, and interpretability. While XGBoost showed competitive performance, RF was easier to interpret and required less computational overhead for deployment, making it more suitable for this application. **Final Choice**: **Random Forest** was chosen as the model for predicting loan eligibility because of its high performance and the ability to generalize well to new, unseen data.

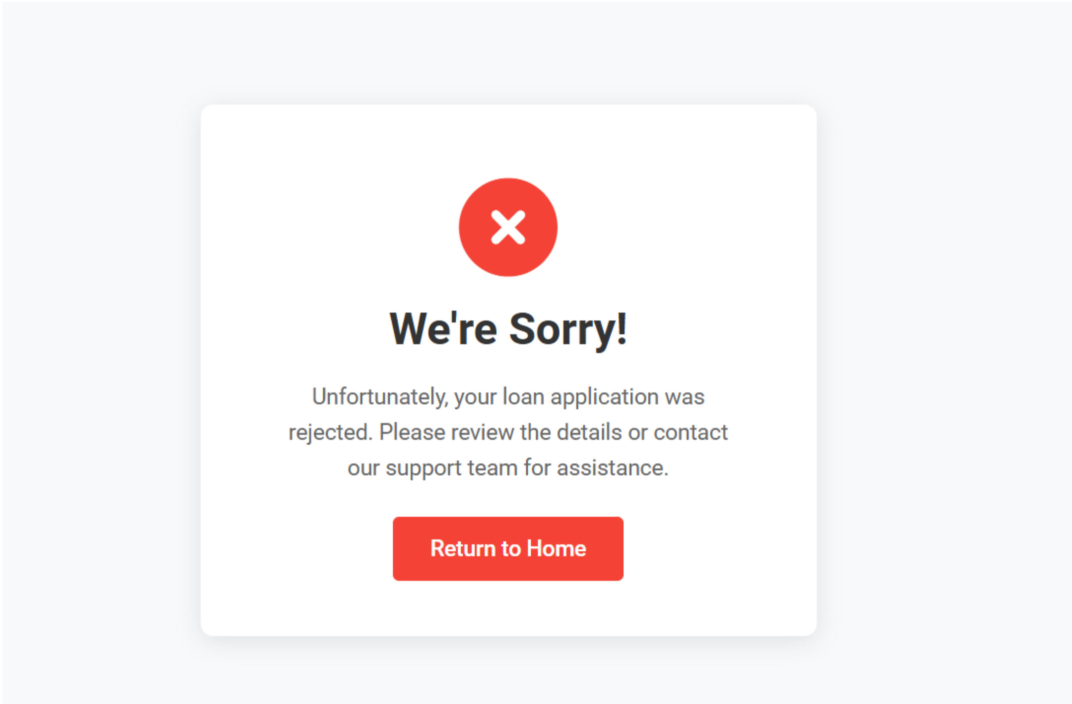
6 . Results:

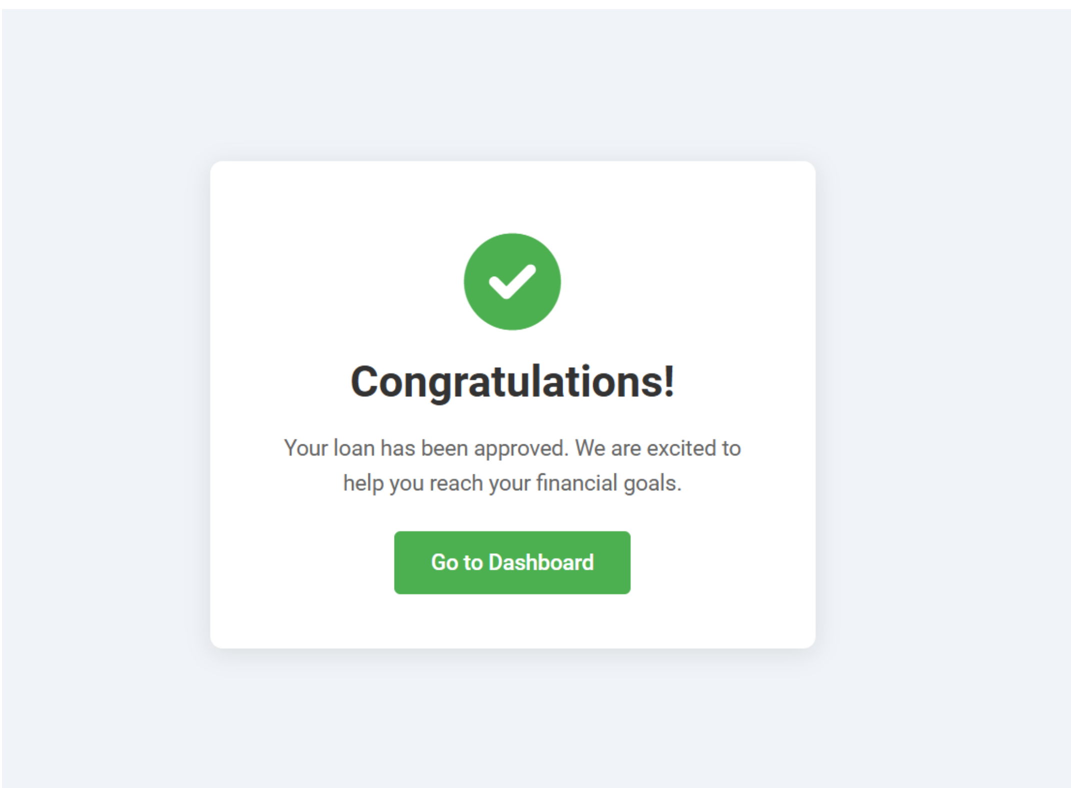
Output screens:








We're Sorry!
Unfortunately, your loan application was rejected. Please review the details or contact our support team for assistance.
Return to Home


Congratulations!
Your loan has been approved. We are excited to help you reach your financial goals.
Go to Dashboard

pyNotebook output screens:

import all required libraries

```
[ ] # Dataframe manipulation
    import pandas as pd

    # Linear algebra
    import numpy as np

    # Data visualization with plotnine
    from plotnine import *
    import plotnine

    # Data visualization with matplotlib
    import matplotlib.pyplot as plt

    # to serialize and deserialize the dataframes
    import pickle

    # Data partitioning
    from sklearn.model_selection import train_test_split
    from sklearn.model_selection import KFold


    # for quick statistical representation
    import seaborn as sns

    # to analyze the library
    import sklearn
    # Machine learning models
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import  RandomForestClassifier
    from sklearn.neighbors import KNeighborsClassifier
```

load the data

```
[ ] data = pd.read_csv('loan_approval_dataset.csv')
```

```
[ ] data.drop(columns = ['loan_id'], inplace=True)
```

```
[ ] data.columns
```

```
Index([' no_of_dependents', ' education', ' self_employed', ' income_annum',
       ' loan_amount', ' loan_term', ' cibil_score',
       ' residential_assets_value', ' commercial_assets_value',
       ' luxury_assets_value', ' bank_asset_value', ' loan_status'],
      dtype='object')
```

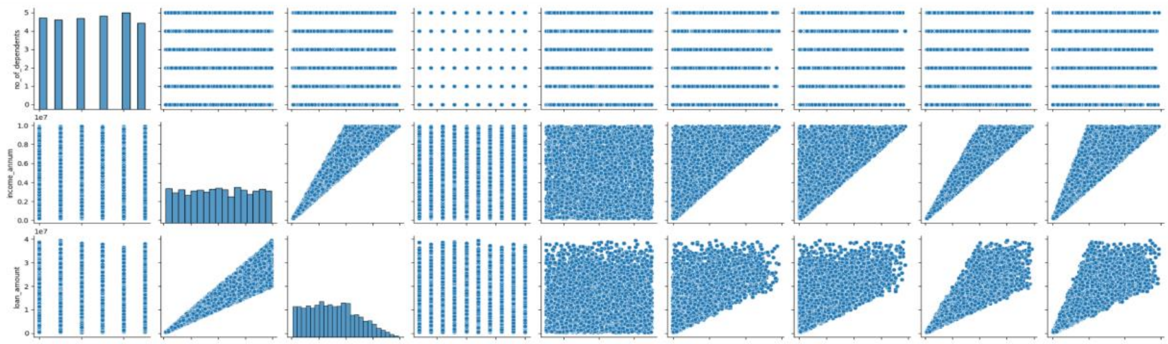
read the data

```
data
```

	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets_value	commercial_assets_value	luxury_assets_value	bank_asset_value	loan_status
0	2	Graduate	No	9600000	29900000	12	778	2400000	17600000	22700000	8000000	Approved
1	0	Not Graduate	Yes	4100000	12200000	8	417	2700000	2200000	8800000	3300000	Rejected
2	3	Graduate	No	9100000	29700000	20	506	7100000	4500000	33300000	12800000	Rejected
3	3	Graduate	No	8200000	30700000	8	467	18200000	3300000	23300000	7900000	Rejected
4	5	Not Graduate	Yes	9800000	24200000	20	382	12400000	8200000	29400000	5000000	Rejected
...	...	...	...	...	...	...	...	...	...	...	...	...
4264	5	Graduate	Yes	1000000	2300000	12	317	2800000	500000	3300000	800000	Rejected
4265	0	Not Graduate	Yes	3300000	11300000	20	559	4200000	2900000	11000000	1900000	Approved
4266	2	Not Graduate	No	6500000	23900000	18	457	1200000	12400000	18100000	7300000	Rejected
4267	1	Not Graduate	No	4100000	12800000	8	780	8200000	700000	14100000	5800000	Approved
4268	1	Graduate	No	9200000	29700000	10	607	17800000	11800000	35700000	12000000	Approved

4269 rows × 12 columns

```python
# Pairplot for numerical features
sns.pairplot(data.select_dtypes(include=np.number))
plt.show()
```



data information

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4269 entries, 0 to 4268
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   no_of_dependents   4269 non-null   int64 
 1   education          4269 non-null   object
 2   self_employed      4269 non-null   object
 3   income_annum       4269 non-null   int64 
 4   loan_amount        4269 non-null   int64 
 5   loan_term          4269 non-null   int64 
 6   cibil_score        4269 non-null   int64 
 7   loan_status        4269 non-null   object
 8   Assets             4269 non-null   int64 
dtypes: int64(6), object(3)
memory usage: 300.3+ KB
```

checking null values and treatment

```python
data.isnull().sum()
```

	0
no_of_dependents	0
education	0
self_employed	0
income_annum	0
loan_amount	0
loan_term	0
cibil_score	0
loan_status	0
Assets	0

dtype: int64

handling the categorial values

```
[ ] data.education.unique()
```

```
array([' Graduate', ' Not Graduate'], dtype=object)
```

```
[ ] def clean_data(st):
        st = st.strip()
        return st
```

```
clean_data(' Graduate')
```

```
'Graduate'
```

```
[ ] data.education = data.education.apply(clean_data)
```

```
[ ] data.education.unique()
```

```
array(['Graduate', 'Not Graduate'], dtype=object)
```

+ Code + Text

```
[ ] data['education'] = data['education'].replace(['Graduate', 'Not Graduate'],[1,0])
```

```
<ipython-input-24-7c9149ef7d84>:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a
```

```
[ ] input_data = data.drop(columns=['loan_status'])
    output_data = data['loan_status']
```

balancing the data

```
[ ] # prompt: balance the data and print it

    from imblearn.over_sampling import RandomOverSampler

    ros = RandomOverSampler(random_state=42)
    X_resampled, y_resampled = ros.fit_resample(input_data, output_data)

    print(y_resampled.value_counts())
```

```
loan_status
1    2656
0    2656
Name: count, dtype: int64
```

spliting the data

```
[ ] from sklearn.model_selection import train_test_split
```

+ Code + Text

```
[ ] x_train,x_test,y_train,y_test = train_test_split(input_data,output_data, test_size=0.2)
```

```
[ ] x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((3415, 8), (854, 8), (3415,), (854,))
```

scaling the data

```
[ ] from sklearn.preprocessing import StandardScaler
```

```
[ ] scaler = StandardScaler()
```

```
[ ] x_train_scaled = scaler.fit_transform(x_train)
```

```
[ ] x_test_scaled = scaler.transform(x_test)
```

```
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.89      0.88       316
           1       0.93      0.93      0.93       538

    accuracy                           0.91       854
   macro avg       0.90      0.91      0.91       854
weighted avg       0.91      0.91      0.91       854

KNN Confusion Matrix:
[[281  35]
 [ 40 498]]
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       316
           1       0.99      0.99      0.99       538

    accuracy                           0.98       854
   macro avg       0.98      0.98      0.98       854
weighted avg       0.98      0.98      0.98       854

Decision Tree Confusion Matrix:
[[310   6]
 [  8 530]]
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       316
           1       0.99      0.98      0.99       538

    accuracy                           0.98       854
   macro avg       0.98      0.98      0.98       854
weighted avg       0.98      0.98      0.98       854
```

- random forest

```
# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(x_train_scaled, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

```
y_pred_rf = rf_model.predict(x_test_scaled)

# Evaluate Random Forest
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
f1_rf = f1_score(y_test, y_pred_rf)

print("Random Forest:")
print("Accuracy:", accuracy_rf)
print("Precision:", precision_rf)
print("Recall:", recall_rf)
print("F1-Score:", f1_rf)
print("\n")
```

```
Random Forest:
Accuracy: 0.9777517564402810
Precision: 0.9850467289719627
Recall: 0.9795539033457249
F1-Score: 0.9822926374650512
```

	Model	Accuracy	Precision	Recall	F1-Score
0	Random Forest	0.977752	0.985047	0.979554	0.982293
1	KNN	0.912178	0.934334	0.925651	0.929972
2	Decision Tree	0.983607	0.988806	0.985130	0.986965
3	XGBoost	0.982436	0.988785	0.983271	0.986021

```
print("Best XGBoost Accuracy:", xgb_grid.best_score_)
```

```
Best Random Forest Parameters: {'max_depth': None, 'min_samples_split': 5, 'n_estimators': 100}
Best Random Forest Accuracy: 0.9795041642814564
Best KNN Parameters: {'n_neighbors': 7, 'weights': 'distance'}
Best KNN Accuracy: 0.918009713656467
Best Decision Tree Parameters: {'max_depth': None, 'min_samples_split': 2}
Best Decision Tree Accuracy: 0.981259833367022
Best XGBoost Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
Best XGBoost Accuracy: 0.9830165311147149
```

saving the model

```
[ ] import pickle as pk
```

```
[ ] pk.dump(model, open('rfmodel.pkl','wb'))
```

```
[ ] pk.dump(scaler,open('scaler.pkl','wb'))
```

```
[ ] Start coding or generate with AI.
```

7. Advantages & Disadvantages

- **Advantages:**
- **Efficiency**: The automated loan approval system streamlines decision-making, reducing the time required for manual reviews.
- **Accuracy**: Machine learning models provide data-driven decisions, minimizing human errors and biases.
- **Scalability**: The system can easily handle a large volume of loan applications without the need for proportional resource expansion.
- **Fairness**: Automated decision-making reduces the risk of biased outcomes, ensuring more fair assessments based on objective criteria.
- **Adaptability**: Continuous learning from new data allows the system to adapt to changing financial patterns and applicant profiles.

Disadvantages:

- **Data Dependency**: The model's accuracy is heavily dependent on the quality and quantity of data, which may introduce biases if the data is unbalanced.
- **Model Interpretability**: Some machine learning models, especially complex ones like XGBoost, may lack transparency, making it difficult for stakeholders to understand decisions.
- **Costs**: Implementing and maintaining such a system, especially with hardware and computational needs, can incur significant costs.
- **Overfitting Risk**: There's a possibility of models overfitting to training data, which can affect the generalization to unseen loan applicants.

8. Conclusion

- ✓ The SmartLender project demonstrates the potential of integrating machine learning into the loan approval process. By addressing the inefficiencies and inaccuracies of traditional manual reviews, SmartLender offers a solution that not only speeds up decision-making but also ensures more accurate and fair outcomes for applicants. The automated evaluation process will benefit both lenders and applicants, fostering a more streamlined and reliable loan approval process. As the project progresses, the continuous improvement of models and data quality will further enhance the system’s performance and adaptability in real-world scenarios.

9. Future Scope

SmartLender has a promising future with potential enhancements that can improve its effectiveness and applicability:

- **Incorporation of Additional Financial Metrics**: Future iterations of the model could include more detailed financial indicators like debt-to-income ratios, credit card histories, or previous loan repayments.
- **Expansion to Other Financial Services**: The same machine learning techniques could be adapted for use in insurance underwriting, credit card approvals, or mortgage lending.
- **Real-Time Data Integration**: The inclusion of real-time financial data feeds could allow for more dynamic loan approval decisions that account for market fluctuations.
- **Advanced Model Techniques**: Incorporating deep learning techniques or reinforcement learning could further boost the accuracy of creditworthiness predictions.
- **Ethical AI Practices**: Ongoing work should focus on ensuring the model adheres to ethical standards, addressing potential biases in the data and decision

10 . APPENDIX

10.1 source code:

App.py

```
from markupsafe import escape

from flask import Flask, request, render_template,redirect,url_for

import pickle

import numpy as np



app = Flask(__name__)

model = pickle.load(open(r'rfmodel.pkl', 'rb'))

scaler = pickle.load(open(r'scaler.pkl', 'rb'))

# Define mapping dictionaries

education_mapping = {"Graduate": 0, "Not Graduate": 1}

employed_mapping = {"Yes": 1, "No": 0}
```

```
@app.route('/')

def home():

    return render_template("index.html")



@app.route('/predict', methods=['GET', 'POST'])

def predict():

    if request.method == 'POST':

        try:

            # Fetch input values from the form

            dependents = int(request.form['dependents'])

            education = request.form['education']

            employed = request.form['employed']

            income_annum = int(request.form['income_annum'])

            LoanAmount = int(request.form['LoanAmount'])

            Loan_Term = int(request.form['Loan_Term'])

            cibil = int(request.form['cibil'])
```

```
        assets = int(request.form['assets'])

        print(f"Dependents: {dependents}, Education: {education},
Employed: {employed}, "
              f"Income Annum: {income_annum}, Loan Amount:
{LoanAmount}, Loan Term: {Loan_Term}, "
              f"CIBIL: {cibil}, Assets: {assets}")

        # Use predefined mappings for education and employment
status
        grad_s = education_mapping.get(education, 1)
        emp_s = employed_mapping.get(employed, 0)

        # Prepare input data for the model
        data = [[dependents, grad_s, emp_s, income_annum,
LoanAmount, Loan_Term, cibil, assets]]
        data = scaler.transform(data)  # Apply scaling
```

```
        # Make prediction
        prediction = model.predict(data)

        if prediction[0] == 1:
            return redirect(url_for('loan_approved'))
        else:
            return redirect(url_for('loan_rejected'))

    except KeyError as e:
        return f"KeyError: {str(e)}. Please check your form data.", 400
    except ValueError as e:
        return f"ValueError: {str(e)}. Please check your input values.",
400
    except Exception as e:
        return str(e), 400
```

```python
    return render_template('prediction.html')


@app.route('/loan_approved')
def loan_approved():
    return render_template("approved.html")  # Create this template for approved loans


@app.route('/loan_rejected')
def loan_rejected():
    return render_template('rejected.html')  # Create this template for rejected loans


if __name__ == "__main__":
    app.run(debug=True)
```

index.html

```
hh<!doctype html>

<html lang="en">

  <head>

    <!-- Required meta tags -->

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">



    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxl
pbzKgwra6" crossorigin="anonymous">

    <link
href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css"
rel="stylesheet">



    <title>loan Prediction</title>
```

```
  </head>

  <body>


<!-- This example requires Tailwind CSS v2.0+ -->

<div class="relative bg-white overflow-hidden">

  <div class="max-w-7xl mx-auto">

    <div class="relative z-10 pb-8 bg-white sm:pb-16 md:pb-20 lg:max-
w-2xl lg:w-full lg:pb-28 xl:pb-32">

      <svg class="hidden lg:block absolute right-0 inset-y-0 h-full w-48
text-white transform translate-x-1/2" fill="currentColor" viewBox="0
0 100 100" preserveAspectRatio="none" aria-hidden="true">

        <polygon points="50,0 100,0 50,100 0,100" />

      </svg>


      <div class="relative pt-6 px-4 sm:px-6 lg:px-8">

        <nav class="relative flex items-center justify-between sm:h-10
lg:justify-start" aria-label="Global">
```

```
    <div class="flex items-center flex-grow flex-shrink-0 lg:flex-grow-0">
      <div class="flex items-center justify-between w-full md:w-auto">
        <a href="#">
          <span class="sr-only">Workflow</span>
          <img class="h-8 w-auto sm:h-10" src="https://tailwindui.com/img/logos/workflow-mark-indigo-600.svg">
        </a>
        <div class="-mr-2 flex items-center md:hidden">
          <button type="button" class="bg-white rounded-md p-2 inline-flex items-center justify-center text-gray-400 hover:text-gray-500 hover:bg-gray-100 focus:outline-none focus:ring-2 focus:ring-inset focus:ring-indigo-500" aria-expanded="false">
            <span class="sr-only">Open main menu</span>
            <!-- Heroicon name: outline/menu -->
```

```
                  <svg class="h-6 w-6"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24
24" stroke="currentColor" aria-hidden="true">
                    <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />
                  </svg>
                </button>
              </div>
            </div>
          </div>
          <div class="hidden md:block md:ml-10 md:pr-4 md:space-x-
8">
            <a href="#" class="font-medium text-gray-500 hover:text-
gray-900">Home</a>

            <a href="#" class="font-medium text-gray-500 hover:text-
gray-900">Prediction</a>
```

```
            <a href="#" class="font-medium text-gray-500 hover:text-
gray-900">About us</a>

            <a href="#" class="font-medium text-gray-500 hover:text-
gray-900">contact</a>

          </div>
        </nav>
      </div>

      <!--
        Mobile menu, show/hide based on menu open state.

        Entering: "duration-150 ease-out"
          From: "opacity-0 scale-95"
          To: "opacity-100 scale-100"
        Leaving: "duration-100 ease-in"
```

```
        From: "opacity-100 scale-100"

        To: "opacity-0 scale-95"

    -->

      <div class="absolute top-0 inset-x-0 p-2 transition transform
origin-top-right md:hidden">

        <div class="rounded-lg shadow-md bg-white ring-1 ring-black
ring-opacity-5 overflow-hidden">

          <div class="px-5 pt-4 flex items-center justify-between">

            <div>

              <img class="h-8 w-auto"
src="https://tailwindui.com/img/logos/workflow-mark-indigo-
600.svg" alt="">

            </div>

            <div class="-mr-2">

              <button type="button" class="bg-white rounded-md p-2
inline-flex items-center justify-center text-gray-400 hover:text-gray-
500 hover:bg-gray-100 focus:outline-none focus:ring-2 focus:ring-
inset focus:ring-indigo-500">
```

```
                <span class="sr-only">Close main menu</span>
                <!-- Heroicon name: outline/x -->
                <svg class="h-6 w-6"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24
24" stroke="currentColor" aria-hidden="true">
                  <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M6 18L18 6M6 6l12 12" />
                </svg>
              </button>
            </div>
          </div>
        <div class="px-2 pt-2 pb-3 space-y-1">
          <a href="#" class="block px-3 py-2 rounded-md text-base
font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-
50">Home</a>
```

```
<a href="#" class="block px-3 py-2 rounded-md text-base
font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-
50">prediction</a>

<a href="#" class="block px-3 py-2 rounded-md text-base
font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-
50">about us</a>

<a href="#" class="block px-3 py-2 rounded-md text-base
font-medium text-gray-700 hover:text-gray-900 hover:bg-gray-
50">contact</a>
</div>

</div>
</div>

<main class="mt-10 mx-auto max-w-7xl px-4 sm:mt-12 sm:px-6
md:mt-16 lg:mt-20 lg:px-8 xl:mt-28">
```

```
    <div class="sm:text-center lg:text-left">
     <h1 class="text-4xl tracking-tight font-extrabold text-gray-900
sm:text-5xl md:text-6xl">
      <span class="block xl:inline">Smart Lender</span>
      <span class="block text-indigo-600 xl:inline"> Applicant
credibility prediction using Machine Learnig </span>
     </h1>
     <p class="mt-3 text-base text-gray-500 sm:mt-5 sm:text-lg
sm:max-w-xl sm:mx-auto md:mt-5 md:text-xl lg:mx-0">
      Smart Lender - Applicant credibility prediction in loan
approval process using Machine Learnig
     </p>
     <div class="mt-5 sm:mt-8 sm:flex sm:justify-center lg:justify-
start">
      <div class="rounded-md shadow">
       <a href="./predict" class="w-full flex items-center justify-
center px-8 py-3 border border-transparent text-base font-medium
```

```
rounded-md text-white bg-indigo-600 hover:bg-indigo-700 md:py-4
md:text-lg md:px-10">
                Prediction
              </a>
            </div>


          </div>
        </div>
      </main>
    </div>
  </div>
  <div class="lg:absolute lg:inset-y-0 lg:right-0 lg:w-1/2">
    <img class="h-56 w-full object-cover sm:h-72 md:h-96 lg:w-full
lg:h-full" src="https://images.unsplash.com/photo-1551434678-
e076c223a692?ixlib=rb-
1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=2
850&q=80" alt="">
```

```
  </div>

 </div>




    <!-- Option 1: Bootstrap Bundle with Popper -->

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
 beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
 JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG
 2vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>


  </body>

 </html>

 #Prediction.html

hhh<!doctype html>

<html lang="en">

 <head>

  <!-- Required meta tags -->
```

```
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">


    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlp
bzKgwra6" crossorigin="anonymous">

    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css"
rel="stylesheet">


    <title>loan Prediction</title>

  </head>

  <body>


<!-- This example requires Tailwind CSS v2.0+ -->
```

```
<div class="relative bg-white overflow-hidden">
  <div class="max-w-7xl mx-auto">
    <div class="relative z-10 pb-8 bg-white sm:pb-16 md:pb-20 lg:max-w-2xl lg:w-full lg:pb-28 xl:pb-32">
      <svg class="hidden lg:block absolute right-0 inset-y-0 h-full w-48 text-white transform translate-x-1/2" fill="currentColor" viewBox="0 0 100 100" preserveAspectRatio="none" aria-hidden="true">
        <polygon points="50,0 100,0 50,100 0,100" />
      </svg>

      <div class="relative pt-6 px-4 sm:px-6 lg:px-8">
        <nav class="relative flex items-center justify-between sm:h-10 lg:justify-start" aria-label="Global">
          <div class="flex items-center flex-grow flex-shrink-0 lg:flex-grow-0">
            <div class="flex items-center justify-between w-full md:w-auto">
```

```
<a href="#">

  <span class="sr-only">Workflow</span>

  <img class="h-8 w-auto sm:h-10"
src="https://tailwindui.com/img/logos/workflow-mark-indigo-
600.svg">

</a>

<div class="-mr-2 flex items-center md:hidden">

  <button type="button" class="bg-white rounded-md p-2
inline-flex items-center justify-center text-gray-400 hover:text-gray-500
hover:bg-gray-100 focus:outline-none focus:ring-2 focus:ring-inset
focus:ring-indigo-500" aria-expanded="false">

    <span class="sr-only">Open main menu</span>

    <!-- Heroicon name: outline/menu -->

    <svg class="h-6 w-6" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor" aria-
hidden="true">

      <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />
```

```
          </svg>
        </button>
      </div>
    </div>
  </div>
  <div class="hidden md:block md:ml-10 md:pr-4 md:space-x-8">
    <a href="#" class="font-medium text-gray-500 hover:text-gray-
900">Home</a>


    <a href="#" class="font-medium text-gray-500 hover:text-gray-
900">Prediction</a>


    <a href="#" class="font-medium text-gray-500 hover:text-gray-
900">About us</a>


    <a href="#" class="font-medium text-gray-500 hover:text-gray-
900">contact</a>
```

```
    </div>

  </nav>

</div>



<!--

  Mobile menu, show/hide based on menu open state.



  Entering: "duration-150 ease-out"

    From: "opacity-0 scale-95"

    To: "opacity-100 scale-100"

  Leaving: "duration-100 ease-in"

    From: "opacity-100 scale-100"

    To: "opacity-0 scale-95"

-->
```

```
    <div class="absolute top-0 inset-x-0 p-2 transition transform
origin-top-right md:hidden">
     <div class="rounded-lg shadow-md bg-white ring-1 ring-black
ring-opacity-5 overflow-hidden">
      <div class="px-5 pt-4 flex items-center justify-between">
       <div>
        <img class="h-8 w-auto"
src="https://tailwindui.com/img/logos/workflow-mark-indigo-600.svg"
alt="">
       </div>
       <div class="-mr-2">
        <button type="button" class="bg-white rounded-md p-2
inline-flex items-center justify-center text-gray-400 hover:text-gray-500
hover:bg-gray-100 focus:outline-none focus:ring-2 focus:ring-inset
focus:ring-indigo-500">
         <span class="sr-only">Close main menu</span>
         <!-- Heroicon name: outline/x -->
```

```
                <svg class="h-6 w-6" xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke="currentColor" aria-
hidden="true">
                  <path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M6 18L18 6M6 6l12 12" />
                </svg>
              </button>
            </div>
          </div>
        <div class="px-2 pt-2 pb-3 space-y-1">
          <a href="#" class="block px-3 py-2 rounded-md text-base font-
medium text-gray-700 hover:text-gray-900 hover:bg-gray-
50">Home</a>


          <a href="#" class="block px-3 py-2 rounded-md text-base font-
medium text-gray-700 hover:text-gray-900 hover:bg-gray-
50">prediction</a>
```

```
            <a href="#" class="block px-3 py-2 rounded-md text-base font-
medium text-gray-700 hover:text-gray-900 hover:bg-gray-50">about
us</a>


            <a href="#" class="block px-3 py-2 rounded-md text-base font-
medium text-gray-700 hover:text-gray-900 hover:bg-gray-
50">contact</a>
          </div>


        </div>
      </div>


      <main class="mt-10 mx-auto max-w-7xl px-4 sm:mt-12 sm:px-6
md:mt-16 lg:mt-20 lg:px-8 xl:mt-28">
        <div class="sm:text-center lg:text-left">
          <h1 class="text-4xl tracking-tight font-extrabold text-gray-900
sm:text-5xl md:text-6xl">
```

```
<span class="block xl:inline">Smart Lender</span>

<span class="block text-indigo-600 xl:inline"> Applicant
credibility prediction using Machine Learnig </span>

</h1>

<p class="mt-3 text-base text-gray-500 sm:mt-5 sm:text-lg
sm:max-w-xl sm:mx-auto md:mt-5 md:text-xl lg:mx-0">

Smart Lender - Applicant credibility prediction in loan approval
process using Machine Learnig

</p>

<div class="mt-5 sm:mt-8 sm:flex sm:justify-center lg:justify-
start">

<div class="rounded-md shadow">

<a href="./predict" class="w-full flex items-center justify-
center px-8 py-3 border border-transparent text-base font-medium
rounded-md text-white bg-indigo-600 hover:bg-indigo-700 md:py-4
md:text-lg md:px-10">

Prediction

</a>
```

```
          </div>

        </div>

      </div>

    </main>

  </div>

</div>

<div class="lg:absolute lg:inset-y-0 lg:right-0 lg:w-1/2">

  <img class="h-56 w-full object-cover sm:h-72 md:h-96 lg:w-full
lg:h-full" src="https://images.unsplash.com/photo-1551434678-
e076c223a692?ixlib=rb-
1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=28
50&q=80" alt="">

</div>

</div>
```

```
    <!-- Option 1: Bootstrap Bundle with Popper -->

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2
vI5DkLtS3qm9Ekf" crossorigin="anonymous"></script>


  </body>

</html>

#rejected.html
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Loan Rejected</title>
```

```
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;50
0;700&display=swap" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
    <style>
        /* Inline CSS */

        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Roboto', sans-serif;
```

```
    background-color: #f8f9fa;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

}



.container {

    background-color: white;

    box-shadow: 0px 4px 20px rgba(0, 0, 0, 0.1);

    border-radius: 10px;

    padding: 40px;

    text-align: center;

    width: 90%;

    max-width: 500px;

}
```

```css
.content {
    padding: 20px;
}

.icon {
    font-size: 80px;
    color: #f44336;
    margin-bottom: 20px;
}

h1 {
    font-size: 36px;
    font-weight: 700;
    color: #333;
    margin-bottom: 20px;
```

```
}

p {
    font-size: 18px;
    font-weight: 400;
    color: #666;
    margin-bottom: 40px;
    line-height: 1.6;
}

.button {
    background-color: #f44336;
    color: white;
    text-decoration: none;
    padding: 15px 30px;
    border-radius: 5px;
```

```css
    font-size: 18px;
    font-weight: 500;
    transition: background-color 0.3s ease;
}

.button:hover {
    background-color: #e53935;
}

@media (max-width: 600px) {
    .icon {
        font-size: 60px;
    }

    h1 {
        font-size: 28px;
```

```
            }

            p {
                font-size: 16px;
            }

            .button {
                font-size: 16px;
                padding: 12px 25px;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="content">
```

```
            <i class="fas fa-times-circle icon"></i>

            <h1>We're Sorry!</h1>

            <p>Unfortunately, your loan application was rejected. Please
review the details or contact our support team for assistance.</p>

            <a href="/" class="button">Return to Home</a>

        </div>

    </div>

</body>

</html>
```

10.2 github link:

[GitHub - NIVAS523/-Smart-Lender-Applicant-Credibility-Prediction-for-Loan-Approval-]