Final Project Report

Smart Lender - Applicant Credibility Prediction for Loan Approval using Machine Learning

1. Introduction

1.1 Project overviews

Smart Lender - Applicant Credibility Prediction for Loan Approval is an innovative system developed to assist financial institutions in evaluating the creditworthiness of loan applicants using machine learning algorithms. Traditional loan approval processes often rely on fixed criteria and manual reviews, which can be time-consuming and subject to human bias.

This project aims to overcome these limitations by leveraging advanced machine learning techniques to analyze a wide range of applicant data—such as income, employment status, credit history, loan amount, and more. By identifying complex patterns in these variables, the model can provide accurate predictions regarding the likelihood of an applicant successfully repaying a loan.

This data-driven approach enables lenders to make faster, more informed decisions, reduce the risk of defaults, and optimize overall lending efficiency. Ultimately, Smart Lender not only enhances the decision-making process but also contributes to more inclusive and responsible lending practices.

1.2 Objectives

The primary objective of the Smart Lender project is to develop a machine learning model that can accurately predict the creditworthiness of loan applicants based on various financial and personal attributes. Key goals include enhancing prediction accuracy compared to traditional credit scoring models, identifying important factors that influence loan approval decisions, and creating a reliable tool for lending institutions to streamline their approval process.

The ultimate aim is to help financial institutions make well-informed decisions that minimize the risk of defaults, optimize loan management processes, and promote financial inclusion by extending credit to a broader range of applicants.

Additionally, the project seeks to contribute to the field of financial technology by demonstrating the effectiveness of machine learning in handling complex lending scenarios and providing data-driven insights for risk management.

2. Project Initialization and Planning Phase

2.1 Define Problem Statements (Customer Problem Statement Template):

The current loan application process challenges customers, impacting their journey and overall satisfaction. Applicants, particularly those seeking urban property loans, encounter hurdles such as limited co-applicant income and a cumbersome application process. These challenges lead to a less-than-optimal customer experience, potentially affecting trust and satisfaction. To enhance our services and improve customer perceptions, we aim to address these pain points. By understanding customers' specific frustrations during the application journey and implementing solutions, we can create an efficient, user-friendly experience that aligns with our customer's expectations and fosters a positive relationship with our brand.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A male applicant seeking a loan.	Secure a loan for the property.	Married with no co-applicant income.	Self-employed with a good credit history.	Optimistic about loan approval.

I am
Applicant seeking a loan
I'm trying to
Secure a loan for property
But
Married with no co-applicant income
Because
Self-employed with a good credit history
Which makes me feel
Optimistic about loan approval

2.2 Project Proposal (Proposed Solution) template:

The proposal report aims to transform loan approval using machine learning, boosting efficiencyand 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. -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	Jupyter Notebook, pycharm
Data		
Data	Source, size, format	Kaggle dataset, 614, csv UCI dataset, 690, csv

2.3 Initial Project Planning Template:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection and Preprocessing	USN-1	Understanding and loading data	2	High	Gopi	23/09/2024	26/09/2024
Sprint-1	Data Collection and Preprocessing	USN-2	Data cleaning	1	High	Gopi	23/09/2024	26/09/2024
Sprint-1	Data Collection and Preprocessing	USN-3	EDA	2	Low	Gopi	23/09/2024	26/09/2024
Sprint-2	Model Development	USN-4	Training the model	2	Medium	Mohan	27/09/2024	30/09/2024
Sprint-2	Model tuning and testing	USN-5	Evaluating the model	1	High	Mohan	27/09/2024	30/09/2024
Sprint-2	Model tuning and testing	USN-6	Model tuning	2	High	Mohan	27/09/2024	30/09/2024
Sprint-2	Model tuning and testing	USN-7	Model testing	1	Medium	Sairam	27/09/2024	30/09/2024
Sprint-3	Web integration and Deployment	USN-8	Building HTML templates	2	Medium	Venkata Sai	01/10/2024	05/10/2024
Sprint-3	Web integration and Deployment	USN-9	Local deployment	2	High	Venkata Sai	01/10/2024	05/10/2024
Sprint-4	Project Report	USN-10	Report	2	Medium	Vishnu, Sairam	06/10/2024	10/10/2024

3 Data Collection and Preprocessing Phase

3.1 Data Collection Plan & Raw Data Sources Identification Template:

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 analysisand decision-making endeavor.

Data Collection Plan:

Section	Description
Project Overview	The machine learning project aims to predict loan approval based on applicant information. Using a dataset with features such as gender, marital status, income, and credit history, the objective is to build a model that accurately classifies loan status (approved or denied), facilitating efficient and informed decision-making in the lending 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 & UCI, the popular platforms for data science competitions and repositories. The provided sample data represents a subset of the collected information, encompassing variables such as gender,
	marital status, income, and loan-related details for machine learning analysis.

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/rishikeshkonapure/home-loan-approval?select=loan_sanction_train.csv	CSV	15 kB	Public
UCI	This data concerns credit card applications; a good mix of attributes	https://archive.ics.uci.edu/dataset/27/credit+approval	CSV	13.6 kB	Public

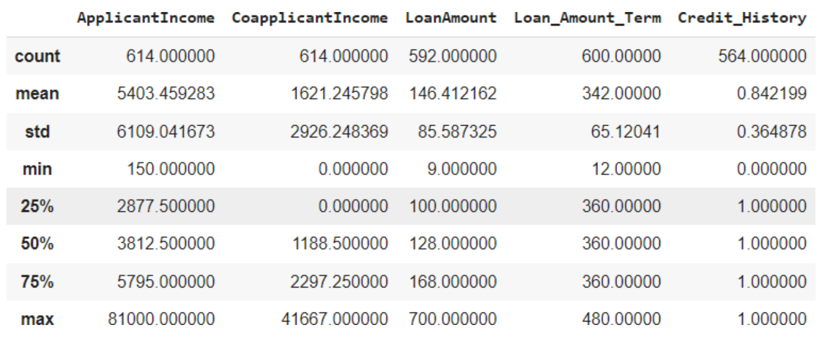
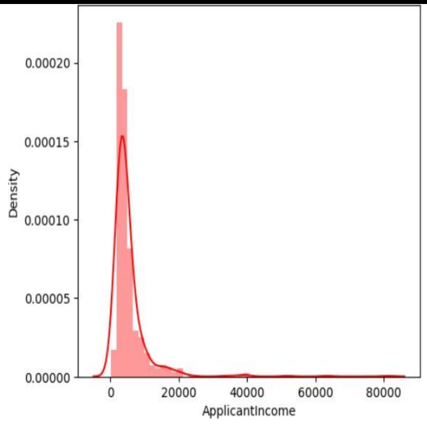
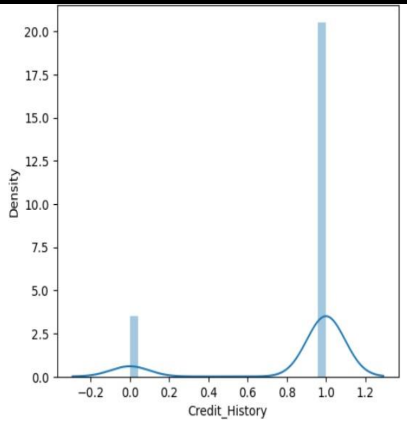
3.2 Data Quality Report Template:

Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset	Missing values in the 'Gender', ‘Married’, ‘Dependents’, ‘Self_Employed’, ‘LoanAmount’, ‘Loan_Amount_Term’, and ‘Credit_History’ columns.	Moderate	Use mean/median imputation.
Kaggle Dataset	Categorical data in the dataset	Moderate	Encoding has to be done in the data.

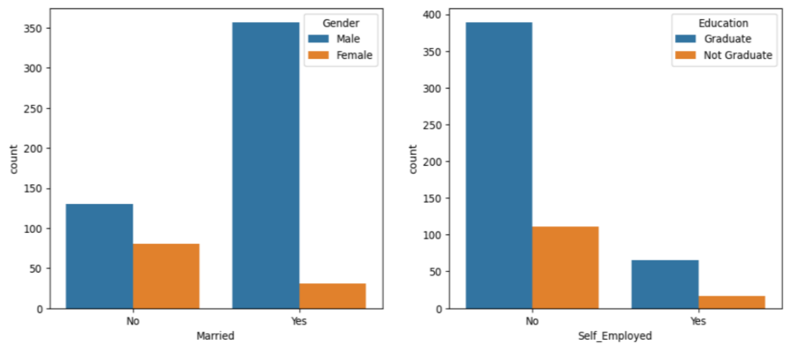
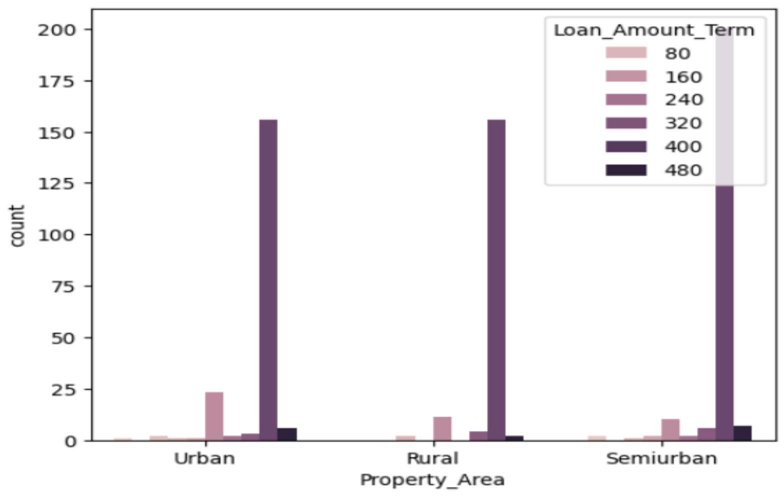
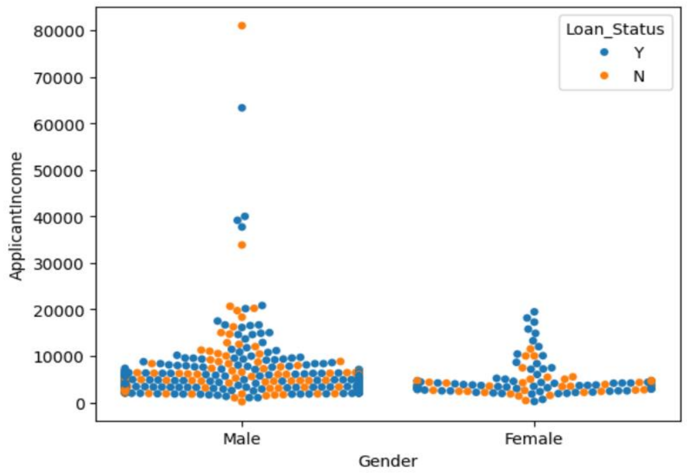
The Data Quality Report will summarize data quality issues from the selected source, includingseverity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

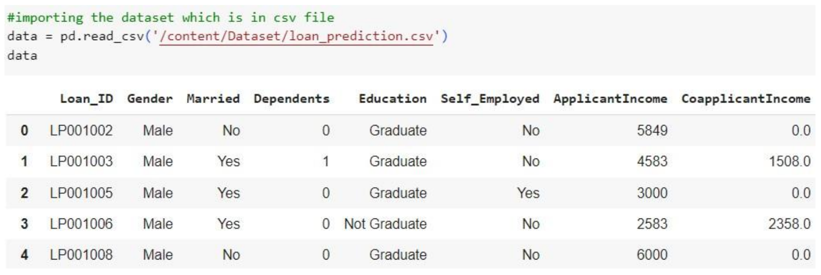
3.3 Data Exploration and Preprocessing Report:

Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning willaddress missing values and outliers, ensuring quality for subsequent analysis and modeling, andforming a strong foundation for insights and predictions.

Section	Description
Data Overview	Dimension: 614 rows × 13 columns Descriptive statistics: 
Univariate Analysis	 

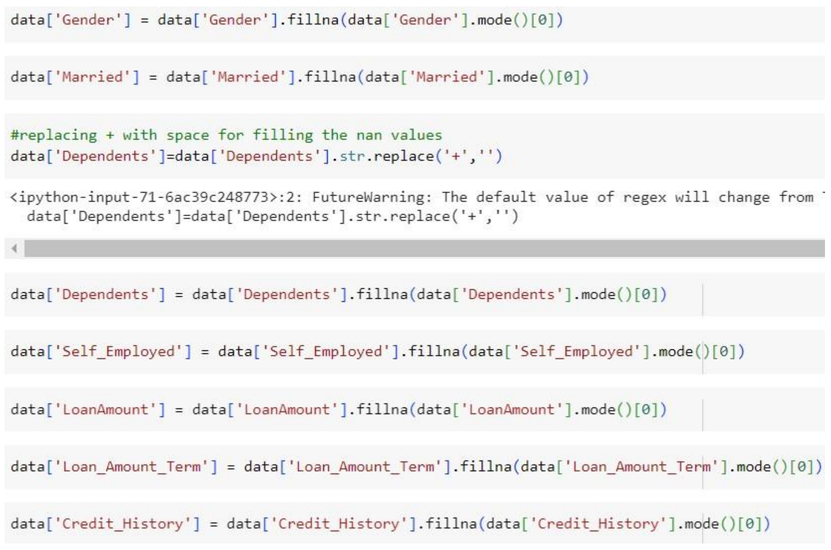
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Bivariate Analysis	 
Multivariate Analysis	

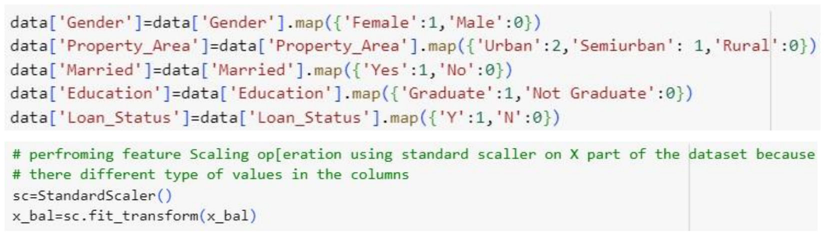
Outliers and Anomalies	-
Data Preprocessing Code Screenshots	
Loading Data	

```
#importing the dataset which is in csv file
data = pd.read_csv('/content/Dataset/loan_prediction.csv')
data
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849	0.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0

Handling Missing Data	

```
data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])

data['Married'] = data['Married'].fillna(data['Married'].mode()[0])

#replacing + with space for filling the nan values
data['Dependents']=data['Dependents'].str.replace('+','')

<ipython-input-71-6ac39c248773>:2: FutureWarning: The default value of regex will change from
  data['Dependents']=data['Dependents'].str.replace('+','')

data['Dependents'] = data['Dependents'].fillna(data['Dependents'].mode()[0])

data['Self_Employed'] = data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])

data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount'].mode()[0])

data['Loan_Amount_Term'] = data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0])

data['Credit_History'] = data['Credit_History'].fillna(data['Credit_History'].mode()[0])
```

Data Transformation	

```
data['Gender']=data['Gender'].map({'Female':1,'Male':0})
data['Property_Area']=data['Property_Area'].map({'Urban':2,'Semiurban': 1,'Rural':0})
data['Married']=data['Married'].map({'Yes':1,'No':0})
data['Education']=data['Education'].map({'Graduate':1,'Not Graduate':0})
data['Loan_Status']=data['Loan_Status'].map({'Y':1,'N':0})

# perfroming feature Scaling op[eration using standard scaller on X part of the dataset because
# there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
```

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

4 Model Development Phase Template

4.1 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.
Gender	Applicant's gender	Yes	Relevant for assessing diversity and potential bias in loan approval.
Married	Marital status of the applicant	Yes	Marital status can impact financial stability and loan eligibility.
Dependents	Number of dependents	Yes	Indicates financial responsibilities and influences loan capacity.

Self_Employed	Self-employment status	Yes	Self-employed individuals may have different financial profiles.
Applicant Income	Income of the applicant	Yes	It is crucial in determining the applicant's financial capacity.
Co-applicant Income	Income of the co-applicant	Yes	Combined income provides a more accurate picture of financial stability.
Loan Amount	Amount of loan applied	Yes	Fundamental for assessing the financial magnitude of the loan.
Loan Amount Term	Term of the loan (in months)	Yes	The loan term influences monthly repayments and impacts eligibility.
Credit_History	Credit history of the applicant	Yes	A major factor in loan approval is reflecting the applicant's creditworthiness.
Loan_Status	Loan approval outcome	Yes	The target variable for predictive modeling – is essential for the project's goal.

4.2 Model Selection Report:

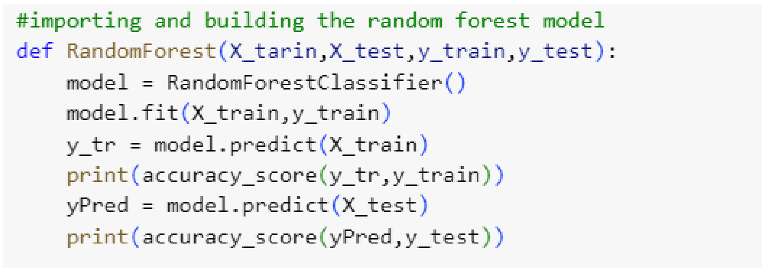
In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

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 = 82%
Decision Tree	Simple tree structure; interpretable, captures non-linear relationships, suitable for initial insights into loan approval patterns.	-	Accuracy score = 77%
KNN	Classifies based on nearest neighbors; adapts well to data patterns, effective for local variations in loan approval criteria.	-	Accuracy score = 75%
Gradient Boostin g	Gradient boosting with trees; optimizes predictive performance, handles complex relationships, and is suitable for accurate loan approval predictions.	-	Accuracy score = 75%

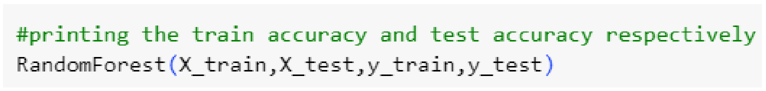
4.3 Initial Model Training Code, Model Validation and Evaluation Report:

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.

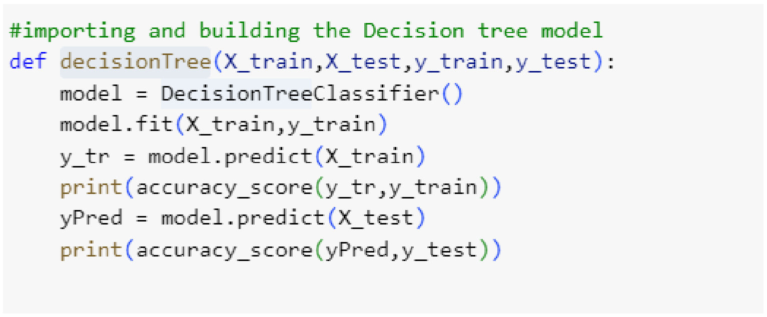
Initial Model Training Code:



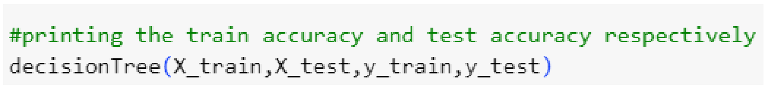
```
#importing and building the random forest model
def RandomForest(X_tarin,X_test,y_train,y_test):
    model = RandomForestClassifier()
    model.fit(X_train,y_train)
    y_tr = model.predict(X_train)
    print(accuracy_score(y_tr,y_train))
    yPred = model.predict(X_test)
    print(accuracy_score(yPred,y_test))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



```
#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)
```

```python
#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))
```

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

```python
#importing and building the Xg boost model
def XGB(X_train,X_test,y_train,y_test):
    model = GradientBoostingClassifier()
    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))
```

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

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix

Random Forest	
	```
	Classification report:
	              precision    recall  f1-score   support

	           0       0.92      0.73      0.81        74
	           1       0.76      0.93      0.84        70

	    accuracy                           0.83       144
	   macro avg       0.84      0.83      0.83       144
	weighted avg       0.84      0.83      0.82       144
	```
		82%	
			```
			Confusion matrix:
			[[54 20]
			 [ 5 65]]
			```

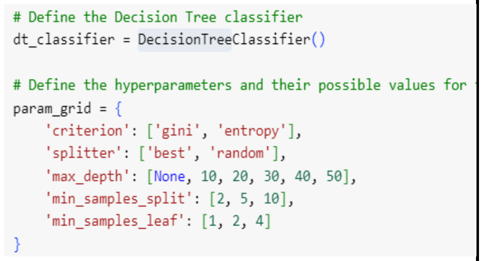
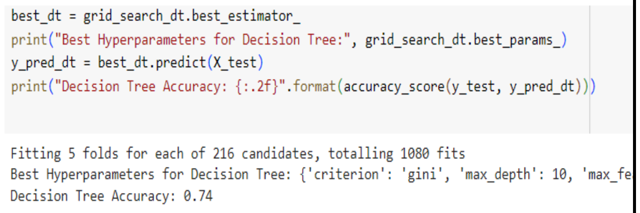
Decision Tree	Classification report precision recall f1-score support 0 0.81 0.73 0.77 74 1 0.74 0.81 0.78 70 accuracy 0.77 144 macro avg 0.77 0.77 0.77 144 weighted avg 0.77 0.77 0.77 144	77%	Confusion matrix [[54 20] [13 57]]
KNN	Classification report: precision recall f1-score support 0 0.87 0.61 0.71 74 1 0.68 0.90 0.78 70 accuracy 0.75 144 macro avg 0.78 0.75 0.75 144 weighted avg 0.78 0.75 0.75 144	75%	Confusion matrix: [[45 29] [7 63]]
Gradient Boosting	Classification report: precision recall f1-score support 0 0.90 0.59 0.72 74 1 0.68 0.93 0.79 70 accuracy 0.76 144 macro avg 0.79 0.76 0.75 144 weighted avg 0.79 0.76 0.75 144	75%	Confusion matrix: [[44 30] [5 65]]

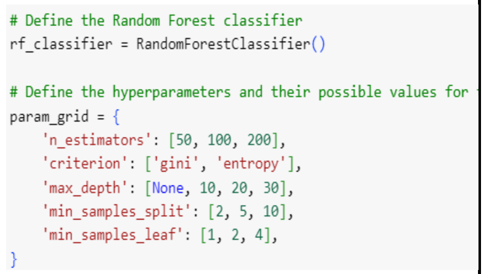
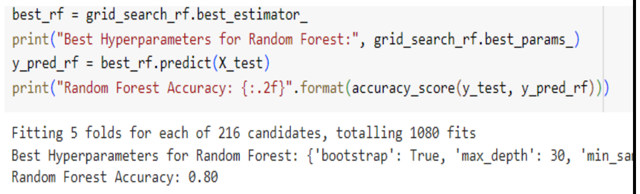
5 Model Optimization and Tuning Phase Template

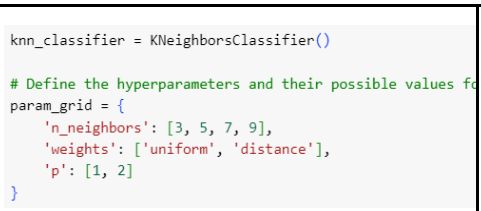
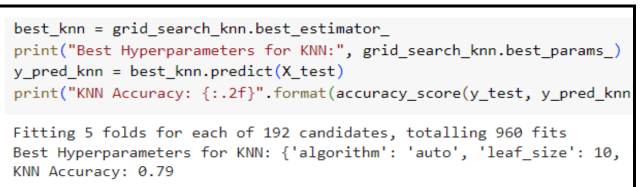
The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

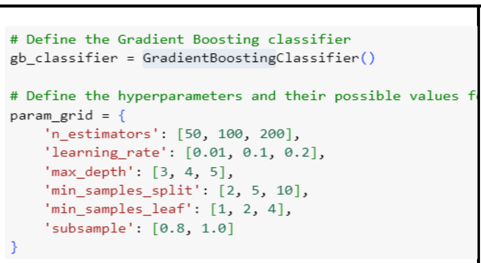
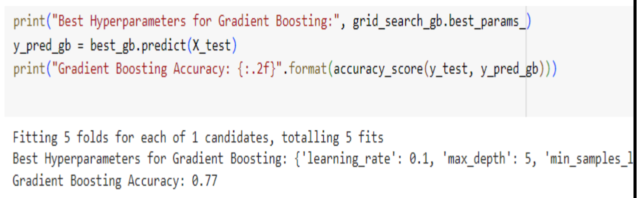
5.1 Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values

Decision Tree	
	`# Define the Decision Tree classifier`
	`dt_classifier = DecisionTreeClassifier()`
	
	`# Define the hyperparameters and their possible values for`
	`param_grid = {`
	`    'criterion': ['gini', 'entropy'],`
	`    'splitter': ['best', 'random'],`
	`    'max_depth': [None, 10, 20, 30, 40, 50],`
	`    'min_samples_split': [2, 5, 10],`
	`    'min_samples_leaf': [1, 2, 4]`
	`}`
		
		`best_dt = grid_search_dt.best_estimator_`
		`print("Best Hyperparameters for Decision Tree:", grid_search_dt.best_params_)`
		`y_pred_dt = best_dt.predict(X_test)`
		`print("Decision Tree Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_dt)))`
		
		`Fitting 5 folds for each of 216 candidates, totalling 1080 fits`
		`Best Hyperparameters for Decision Tree: {'criterion': 'gini', 'max_depth': 10, 'max_fe`
		`Decision Tree Accuracy: 0.74`

Random Forest	
	`# Define the Random Forest classifier`
	`rf_classifier = RandomForestClassifier()`
	
	`# Define the hyperparameters and their possible values for`
	`param_grid = {`
	`    'n_estimators': [50, 100, 200],`
	`    'criterion': ['gini', 'entropy'],`
	`    'max_depth': [None, 10, 20, 30],`
	`    'min_samples_split': [2, 5, 10],`
	`    'min_samples_leaf': [1, 2, 4],`
	`}`
		
		`best_rf = grid_search_rf.best_estimator_`
		`print("Best Hyperparameters for Random Forest:", grid_search_rf.best_params_)`
		`y_pred_rf = best_rf.predict(X_test)`
		`print("Random Forest Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_rf)))`
		
		`Fitting 5 folds for each of 216 candidates, totalling 1080 fits`
		`Best Hyperparameters for Random Forest: {'bootstrap': True, 'max_depth': 30, 'min_sa`
		`Random Forest Accuracy: 0.80`

KNN	
	`knn_classifier = KNeighborsClassifier()`
	
	`# Define the hyperparameters and their possible values fo`
	`param_grid = {`
	`    'n_neighbors': [3, 5, 7, 9],`
	`    'weights': ['uniform', 'distance'],`
	`    'p': [1, 2]`
	`}`
		
		`best_knn = grid_search_knn.best_estimator_`
		`print("Best Hyperparameters for KNN:", grid_search_knn.best_params_)`
		`y_pred_knn = best_knn.predict(X_test)`
		`print("KNN Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_knn`
		
		`Fitting 5 folds for each of 192 candidates, totalling 960 fits`
		`Best Hyperparameters for KNN: {'algorithm': 'auto', 'leaf_size': 10,`
		`KNN Accuracy: 0.79`

Gradient Boosting		

```
# Define the Gradient Boosting classifier
gb_classifier = GradientBoostingClassifier()

# Define the hyperparameters and their possible values fo
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'subsample': [0.8, 1.0]
}
```

```
print("Best Hyperparameters for Gradient Boosting:", grid_search_gb.best_params_)
y_pred_gb = best_gb.predict(X_test)
print("Gradient Boosting Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_gb)))
```

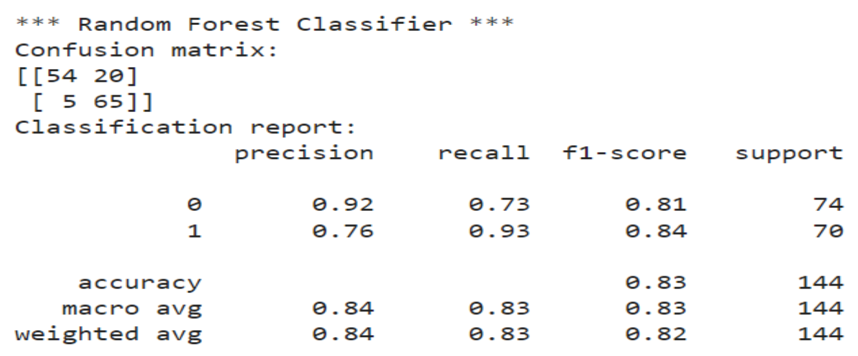
```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
Best Hyperparameters for Gradient Boosting: {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_l
Gradient Boosting Accuracy: 0.77
```

5.2 Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Decision Tree	See output below

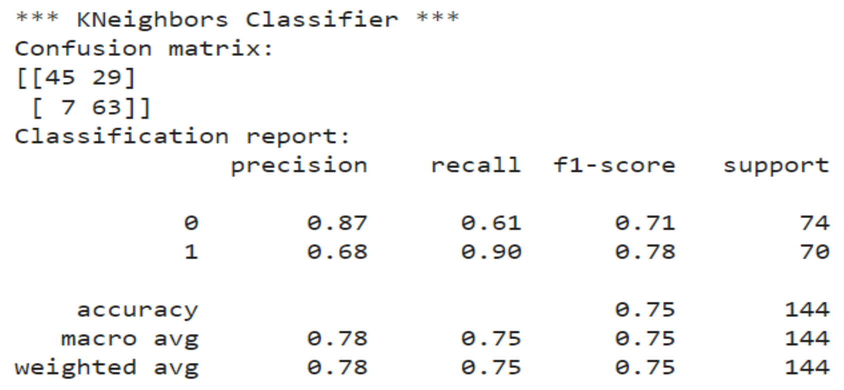
```
**Decision Tree Classifier**
Confusion matrix
[[54 20]
 [13 57]]
Classification report
              precision    recall  f1-score   support

           0       0.81      0.73      0.77        74
           1       0.74      0.81      0.78        70

    accuracy                           0.77       144
   macro avg       0.77      0.77      0.77       144
weighted avg       0.77      0.77      0.77       144
```

Random Forest



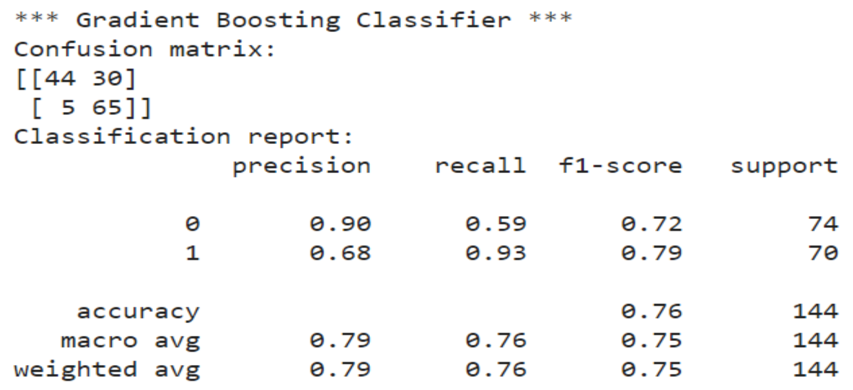
```
*** Random Forest Classifier ***
Confusion matrix:
[[54 20]
 [ 5 65]]
Classification report:
              precision    recall  f1-score   support

           0       0.92      0.73      0.81        74
           1       0.76      0.93      0.84        70

    accuracy                           0.83       144
   macro avg       0.84      0.83      0.83       144
weighted avg       0.84      0.83      0.82       144
```

KNN



```
*** KNeighbors Classifier ***
Confusion matrix:
[[45 29]
 [ 7 63]]
Classification report:
              precision    recall  f1-score   support

           0       0.87      0.61      0.71        74
           1       0.68      0.90      0.78        70

    accuracy                           0.75       144
   macro avg       0.78      0.75      0.75       144
weighted avg       0.78      0.75      0.75       144
```

Gradient Boosting



```
*** Gradient Boosting Classifier ***
Confusion matrix:
[[44 30]
 [ 5 65]]
Classification report:
              precision    recall  f1-score   support

           0       0.90      0.59      0.72        74
           1       0.68      0.93      0.79        70

    accuracy                           0.76       144
   macro avg       0.79      0.76      0.75       144
weighted avg       0.79      0.76      0.75       144
```

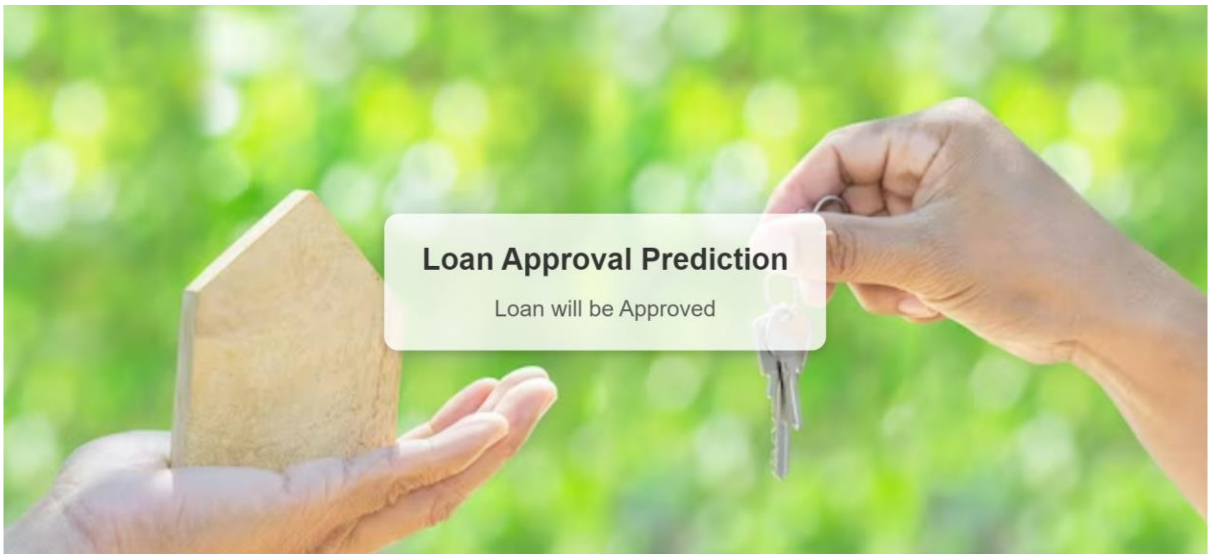
5.3 Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	The Random Forest model was selected for its robust performance and strong generalization capabilities. Its ability to handle a large number of features, mitigate overfitting through bagging, and provide feature importance metrics made it a top contender. Random Forest demonstrated competitive accuracy during hyperparameter tuning, and its ensemble nature ensures better stability and resilience to noise in the dataset. These factors align with the project's objectives, making it an excellent choice as the final model.

6 Results

6.1 Outputs screenshots:

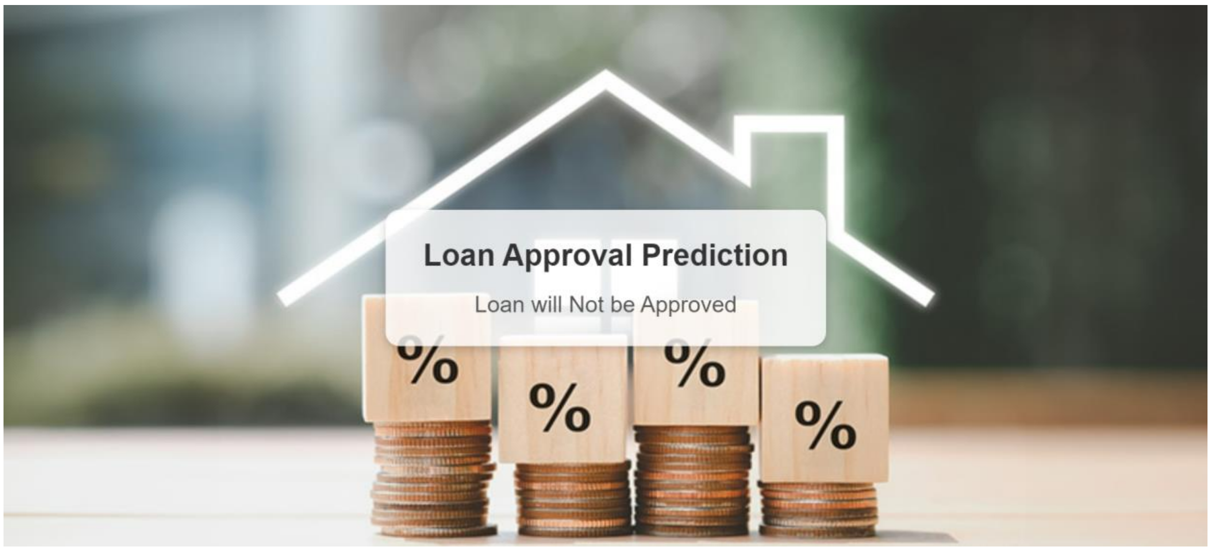
Output for the loan will be approved



Loan Approval Prediction

Loan will be Approved

Output for the loan will not be approved



Loan Approval Prediction

Loan will Not be Approved

7 Advantages & Disadvantages

Advantages:

1. **Accuracy:** Machine learning models can provide highly accurate predictions for loan approvals by analyzing a wide range of applicant attributes and financial data.
2. **Automation:** The loan approval process can be automated, allowing for quicker decision-making and reducing manual intervention, improving the efficiency of lending operations.
3. **Scalability:** ML models can easily handle large volumes of loan applications, processing them efficiently and scaling as the number of applicants grows.
4. **Risk Management:** By identifying high-risk applicants, ML models help financial institutions mitigate risks, adjust loan terms, and reduce the probability of defaults.
5. **Customization:** Models can be fine-tuned for different types of loans or customer segments, making predictions more relevant and tailored to specific lending needs.
6. **Improved Financial Inclusion:** ML models can evaluate applicants with limited credit history or unconventional financial backgrounds, allowing lenders to extend credit to underserved populations.

Disadvantages:

1. **Data Dependency:** The model's performance heavily depends on the quality and quantity of data. Incomplete or inaccurate applicant data may negatively impact prediction accuracy.
2. **Complexity:** Developing, training, and deploying machine learning models for loan approval requires expertise in both finance and data science, making it a complex endeavor.
3. **Resource Intensive:** Training and maintaining ML models can require substantial computational resources and time, especially for large datasets.
4. **Model Maintenance:** The model must be updated frequently with new financial data and trends to maintain its accuracy and relevance over time, requiring ongoing maintenance efforts.
5. **Interpretability:** ML models, especially more advanced ones, can be opaque, making it difficult for lenders to understand the specific reasons behind an applicant's approval or rejection.
6. **Bias:** If the training data is biased (e.g., based on historical lending practices), the model might reinforce such biases, leading to unfair or skewed loan approval decisions.

8 Conclusion:

The application of Random Forest in our Loan Approval Prediction project has highlighted the strength and versatility of machine learning techniques in the financial sector. Random Forest, a robust ensemble method, combines multiple decision trees to create a strong predictive model, significantly improving accuracy and reducing overfitting in our loan approval predictions.

In this project, we carefully preprocessed the dataset, ensuring that high-quality data inputs were provided, which is critical for any machine learning model's performance. The Random Forest algorithm was employed to capture the complex relationships between various applicant attributes, such as income, credit history, and loan amount. This method excels at handling both categorical and numerical data, making it well-suited for our application. By constructing multiple decision trees and averaging their predictions, Random Forest was able to enhance model accuracy and stability, resulting in reliable loan approval predictions.

Our evaluation metrics, including accuracy, precision, and recall, demonstrated the effectiveness of the Random Forest model. Accuracy reflected the model’s ability to correctly predict loan approvals, while precision and recall offered insights into the model’s capability to correctly identify applicants who should be approved or rejected. These metrics are essential in financial contexts where both false positives (approving high-risk applicants) and false negatives (rejecting creditworthy applicants) carry significant consequences.

One of the major strengths of Random Forest is its flexibility and ability to handle imbalanced data—a common issue in loan datasets, where approved and rejected applications may not be equally represented. Moreover, the model's ability to handle missing values and noisy data makes it highly applicable in real-world scenarios, where applicant information may not always be complete or perfectly structured.

Despite these advantages, it is important to recognize the limitations of Random Forest. While the algorithm is less prone to overfitting compared to other models, it can still become computationally expensive when dealing with very large datasets or many trees. Additionally, while Random Forest offers good accuracy, it can sometimes be less interpretable than simpler models, making it harder to understand the specific factors driving each individual loan decision.

In conclusion, the use of Random Forest for loan approval prediction has proven to be an effective and valuable tool for enhancing decision-making processes in lending institutions. The insights derived from this project can help improve risk management and operational efficiency, leading to better financial outcomes for both lenders and applicants. By continually refining the model, incorporating additional applicant features, and ensuring up-

9 Future Scope

1. **Model Enhancement**: Continuously refine and optimize the model by incorporating more diverse datasets, such as alternative financial data, social media profiles, or transaction history, to improve prediction accuracy and capture a broader view of an applicant's creditworthiness.
2. **Real-time Data Integration**: Integrate real-time financial data, credit score updates, and employment verification systems to ensure that the model can provide up-to-date predictions based on the most current applicant information.
3. **User-Friendly Interface**: Develop a more intuitive web or mobile application, allowing both lenders and applicants to easily access loan approval predictions and detailed insights into their creditworthiness.
4. **Geographic Expansion**: Adapt the model to accommodate applicants from different regions or countries by including localized financial behaviors, regulations, and lending practices, making the system scalable for global use.
5. **Inclusion of Alternative Credit Data**: Incorporate alternative credit data sources, such as payment histories for rent, utilities, or mobile services, to enhance the model's ability to assess applicants with limited traditional credit histories, thereby improving financial inclusion.
6. **Collaboration with Financial Institutions**: Partner with banks, microfinance institutions, and credit unions to implement the model in real-world loan approval processes, continuously refining the model based on real-world feedback and performance.
7. **Risk Assessment Optimization**: Expand the model to offer more granular risk assessments, helping lenders tailor loan terms, such as interest rates and loan amounts, based on the applicant's predicted credibility, optimizing risk management for both lenders and borrowers.
8. **Regulatory Compliance and Explainability**: Enhance the model to ensure compliance with evolving financial regulations and develop features that provide explainable AI (XAI), allowing lenders and applicants to understand how decisions are made, increasing transparency and trust in the system.
9. **Credit Behavior Prediction**: Extend the model to not only predict loan approval but also to assess future financial behavior, such as loan repayment tendencies or risk of default, enabling proactive credit management and customized financial products for applicants.
10. **AI-Powered Fraud Detection**: Integrate fraud detection mechanisms that leverage AI and machine learning to identify fraudulent applications or inconsistencies in applicant data, further improving the accuracy and reliability of the loan approval process.

10 Appendix

10.1 Source Code:

```
#IMPORTING NECESSARY LIBRARIES

"""



import pandas as pd

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

from sklearn.neighbors import KNeighborsClassifier

import xgboost

from sklearn.metrics import accuracy_score, classification_report

from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import confusion_matrix, classification_report

from sklearn.metrics import roc_curve, auc, RocCurveDisplay

from sklearn.model_selection import GridSearchCV

import pickle

from imblearn.over_sampling import SMOTE


"""#Importing the Dataset"""


df=pd.read_csv('/content/loan_prediction.csv')

df

"""#Analysing the data"""

df.head()


df.describe()


df.info()

#Uni-variate analysis

# Plotting the figure

plt.figure(figsize=(12, 5))


plt.subplot(121)

sns.distplot(data['ApplicantIncome'], color='r')


plt.subplot(122)
```

```
sns.distplot(data['Credit_History'])


plt.show()

#Bivariate analysis

# Plotting the count plot

plt.figure(figsize=(18, 4))


plt.subplot(1, 4, 1)

sns.countplot(x=data['Gender'])


plt.subplot(1, 4, 2)

sns.countplot(x=data['Education'])


plt.show()

# Plotting count plots with subplots

plt.figure(figsize=(20, 5))


plt.subplot(131)

sns.countplot(x=data['Married'], hue=data['Gender'])


plt.subplot(132)

sns.countplot(x=data['Self_Employed'], hue=data['Education'])


plt.subplot(133)
```

```
sns.countplot(x=data['Property_Area'], hue=data['Loan_Amount_Term'])


plt.show()

#Multivariate analysis

sns.swarmplot(x='Gender',y='ApplicantIncome',hue='Loan_Status',data=data)

#Descriptive analysis

data.describe()


"""#Handling Missing Values"""

data.info()

df.isnull().sum()

# Filling missing values

data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])

data['Married'] = data['Married'].fillna(data['Married'].mode()[0])

data['Dependents'] = data['Dependents'].str.replace(' ', '')

data['Dependents'] = data['Dependents'].fillna(data['Dependents'].mode()[0])

data['Self_Employed'] = data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])

data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount'].median())

data['Loan_Amount_Term'] =
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].median())

data['Credit_History'] = data['Credit_History'].fillna(data['Credit_History'].median())


# Handling Categorical Values

data['Gender']=data['Gender'].map({'Male':1,'Female':0})

data['Property_Area']=data['Property_Area'].map({'Urban':2,'Rural':0,'Semiurban':1})
```

```
  data['Married']=data['Married'].map({'Yes':1,'No':0})

  data['Education']=data['Education'].map({'Graduate':1,'Not Graduate':0})

  data['Loan_Status']=data['Loan_Status'].map({'Y':1,'N':0})

  data['Self_Employed']=data['Self_Employed'].map({'Yes':1,'No':0})

  data['Dependents']=data['Dependents'].map({'0':0,'1':1,'2':2,'3+':3})


data['Gender']=data['Gender'].astype('int64')
data['Married']=data['Married'].astype('int64')
data['Dependents']=data['Dependents'].astype('int64')
data['Self_Employed']=data['Self_Employed'].astype('int64')
data['CoapplicantIncome']=data['CoapplicantIncome'].astype('int64')
data['LoanAmount']=data['LoanAmount'].astype('int64')
data['Loan_Amount_Term']=data['Loan_Amount_Term'].astype('int64')
data['Credit_History']=data['Credit_History'].astype('int64')

#Balancing the dataset
from enum import auto
from imblearn.combine import SMOTETomek
smote = SMOTETomek(sampling_strategy='auto')

X = data.drop(columns=['Loan_Status','Loan_ID'], axis=1)
y = data['Loan_Status']

X_bal, y_bal = smote.fit_resample(X, y)

print("Before balancing:")
print(y.value_counts())

print("\nAfter balancing:")
print(y_bal.value_counts())
# Scaling the Data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_bal = sc.fit_transform(X_bal)
X_bal = pd.DataFrame(X_bal, columns=X.columns)
X_bal.head()
y_bal.head()
# Calculate correlation matrix
z = data.drop(columns=['Loan_ID'],axis=1)
correlation_matrix = z.corr()
print(correlation_matrix['Loan_Status'].sort_values(ascending=False))
# Splitting data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_bal, y_bal, test_size=0.2,
random_state=42)


# Model Building
##Decision tree model
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score


def decisionTree(X_train, X_test, y_train, y_test):
    dt = DecisionTreeClassifier()
    dt.fit(X_train, y_train)
    y_pred = dt.predict(X_test)
    print("**Decision Tree Classifier**")
    print("Confusion matrix")
    print(confusion_matrix(y_test, y_pred))
    print("Classification report")
    print(classification_report(y_test, y_pred))
    print("Accuracy Score: {}".format(accuracy_score(y_test, y_pred)))
decisionTree(X_train, X_test, y_train, y_test)

# Random forest model
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

def randomforest(X_train, X_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    print("*** Random Forest Classifier ***")
    print("Confusion matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("Classification report:")
    print(classification_report(y_test, y_pred))
    print("Accuracy Score: {}".format(accuracy_score(y_test, y_pred)))
randomforest(X_train, X_test, y_train, y_test)

#KNN model

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

def gradientboost(X_train, X_test, y_train, y_test):
    gb = GradientBoostingClassifier()
    gb.fit(X_train, y_train)
    y_pred = gb.predict(X_test)
    print("*** Gradient Boosting Classifier ***")
    print("Confusion matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("Classification report:")
    print(classification_report(y_test, y_pred))
    print("Accuracy Score: {}".format(accuracy_score(y_test, y_pred)))
gradientboost(X_train, X_test, y_train, y_test)

# Comparing the models
df = DecisionTreeClassifier()
rf = RandomForestClassifier()
knn = KNeighborsClassifier()
gb = GradientBoostingClassifier()

df.fit(X_train,y_train)
rf.fit(X_train,y_train)
knn.fit(X_train,y_train)
gb.fit(X_train,y_train)
pred1=df.predict(X_train)
pred2=rf.predict(X_train)
pred3=knn.predict(X_train)
pred4=gb.predict(X_train)

print('Decision Tree:',accuracy_score(y_train,pred1))
print('Random Forest:',accuracy_score(y_train,pred2))
print('KNN:',accuracy_score(y_train,pred3))
print('XGBoost:',accuracy_score(y_train,pred4))

y_pred1=df.predict(X_test)
y_pred2=rf.predict(X_test)
y_pred3=knn.predict(X_test)
y_pred4=gb.predict(X_test)
```

```
print('Decision Tree:',accuracy_score(y_test,y_pred1))
print('Random Forest:',accuracy_score(y_test,y_pred2))
print('KNN:',accuracy_score(y_test,y_pred3))
print('XGBoost:',accuracy_score(y_test,y_pred4))

#Checking Accuracy by excluding negative correlated features
from sklearn.feature_selection import RFE
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

dt = DecisionTreeClassifier()
rfe = RFE(estimator=dt, n_features_to_select=7)

rfe.fit(X_train, y_train)

X_train_rfe = X_train[X_train.columns[rfe.support_]]
X_test_rfe = X_test[X_test.columns[rfe.support_]]

dt.fit(X_train_rfe, y_train)

y_pred = dt.predict(X_test_rfe)

print("Accuracy Score: {:.2f}".format(accuracy_score(y_test, y_pred)))

#Hyperparameter tuning of models

#Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
```

```
# Define the parameter grid for Decision Tree
param_grid_dt = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': [None, 'sqrt', 'log2']
}

dt = DecisionTreeClassifier()
grid_search_dt = GridSearchCV(estimator=dt, param_grid=param_grid_dt,
                              cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
grid_search_dt.fit(X_train, y_train)

best_dt = grid_search_dt.best_estimator_
print("Best Hyperparameters for Decision Tree:", grid_search_dt.best_params_)
y_pred_dt = best_dt.predict(X_test)
print("Decision Tree Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_dt)))

#Random forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for Random Forest
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

rf = RandomForestClassifier()
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf,
                              cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
grid_search_rf.fit(X_train, y_train)

best_rf = grid_search_rf.best_estimator_
print("Best Hyperparameters for Random Forest:", grid_search_rf.best_params_)
y_pred_rf = best_rf.predict(X_test)
print("Random Forest Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_rf)))

#KNN
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for KNN
param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size': [10, 30, 50],
    'p': [1, 2]  # p=1: Manhattan, p=2: Euclidean
}

knn = KNeighborsClassifier()
grid_search_knn = GridSearchCV(estimator=knn, param_grid=param_grid_knn,
                               cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
grid_search_knn.fit(X_train, y_train)
```

```
best_knn = grid_search_knn.best_estimator_
print("Best Hyperparameters for KNN:", grid_search_knn.best_params_)
y_pred_knn = best_knn.predict(X_test)
print("KNN Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_knn)))

#Gradient boosting
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

# Extended parameter grid for Gradient Boosting
param_grid_gb = {
    'n_estimators': [100],
    'learning_rate': [0.1],
    'max_depth': [5],
    'subsample': [0.8],
    'min_samples_split': [5],
    'min_samples_leaf': [2]
}

gb = GradientBoostingClassifier()
grid_search_gb = GridSearchCV(estimator=gb, param_grid=param_grid_gb,
                              cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
grid_search_gb.fit(X_train, y_train)

best_gb = grid_search_gb.best_estimator_
print("Best Hyperparameters for Gradient Boosting:", grid_search_gb.best_params_)
y_pred_gb = best_gb.predict(X_test)
print("Gradient Boosting Accuracy: {:.2f}".format(accuracy_score(y_test, y_pred_gb)))
```

```
#Model Selection

"""After checking the performace of all models after Evaluation RandomForestClassifier is giving good results so choosing RandomForestClassifier as our final models"""

model = RandomForestClassifier(verbose=2,n_estimators=120,max_features='log2',max_depth=10,criterion='entropy')

model.fit(X_train, y_train)


#pickle files

pickle.dump(model,open('loan_prediction_approve.pkl','wb'))

pickle.dump(sc,open('scaling.pkl','wb'))


#Checking user input values for prediction

#scaled input values

input=[[-1.809367,-1.157049,-0.697003,-1.550111,-0.315501,-0.428375,-0.532097,-0.825588,0.268167,-1.581910,0.087535]]

input=sc.transform(input)

prediction = model.predict(input)

prediction
```

app.py

```
# app.py

from flask import Flask, render_template, request
import numpy as np
import pandas as pd
import os
import pickle

app = Flask(__name__)
model = pickle.load(open(r'C:\Users\vamsi\OneDrive\Desktop\Loan Approval\loan_prediction.pkl', 'rb'))
scale = pickle.load(open(r'C:\Users\vamsi\OneDrive\Desktop\Loan Approval\scaling (2).pkl','rb'))
@app.route('/')
def home():
    return render_template('home.html')

@app.route('/predict', methods=['POST', 'GET'])
def predict():
    return render_template('input.html')

@app.route('/submit', methods=['POST', 'GET'])
def submit():
    input_feature = [int(x) for x in request.form.values()]
    input_feature = [np.array(input_feature)]
    print(input_feature)
    names = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History', 'Property_Area']
    data = pd.DataFrame(input_feature, columns=names)
    print(data)
    scaled_data = scale.transform(data)
    print(scaled_data)
    prediction = model.predict(scaled_data)
    print(prediction)
    prediction = int(prediction)
    print(type(prediction))

    if prediction == 0:
        return render_template('output1.html', result='Loan will Not be Approved')
    else:
        return render_template('output.html', result='Loan will be Approved')

if __name__ == '__main__':
    port = int(os.environ.get('PORT', 5000))
    app.run(debug=False)
```

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Loan Prediction</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      font-family: 'Arial', sans-serif;
      background: url('../static/banking-technology.jpg') no-repeat center center fixed;
      background-size: cover;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
    }

    .box {
      background-color: rgba(255, 255, 255, 0.85); /* Semi-transparent box */
      padding: 40px;
      border-radius: 15px;
      text-align: center;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Soft shadow for depth */
      max-width: 600px;
      width: 100%;
    }
```

```
        h1 {
          font-size: 36px;
          color: #000; /* Dark color for the heading */
          margin-bottom: 20px;
        }

        p {
          font-size: 18px;
          color: #333; /* Slightly lighter text color */
          line-height: 1.6;
        }

        a {
          display: inline-block;
          padding: 12px 24px;
          background-color: #28a745; /* Green button color */
          color: white;
          text-decoration: none;
          font-size: 18px;
          border-radius: 8px;
          margin-top: 20px;
          transition: background-color 0.3s ease;
        }

        a:hover {
          background-color: #218838; /* Darker green on hover */
        }
      </style>
    </head>
```

```
<body>
   <div class="box">
      <h1>Welcome to Loan Prediction</h1>
      <p>Loan Approval is based on a lot of things. Rather than going to a bank and getting
rejected, we made it simple. You can get your loan approval prediction by our machine
learning model. To predict, we need some of your information.</p>
      <a href="/predict">Predict</a>
   </div>
</body>
</html>
```

input.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Loan Prediction</title>
   <style>
      /* Style inputs with type="text", select elements and textareas */
      input[type=number], select, textarea {
         width: 100%; /* Full width */
         padding: 12px; /* Some padding */
         border: 1px solid #ccc; /* Gray border */
         border-radius: 4px; /* Rounded borders */
         box-sizing: border-box; /* Make sure that padding and width stays in place */
         margin-top: 6px; /* Add a top margin */
         margin-bottom: 16px; /* Bottom margin */
         resize: vertical; /* Allow the user to vertically resize the textarea (not horizontally) */
      }
```

```
        /* Style the submit button with a specific background color etc */
        input[type=submit] {
          background-color: #04AA6D;
          color: white;
          padding: 12px 20px;
          border: none;
          border-radius: 4px;
          cursor: pointer;
        }

        /* When moving the mouse over the submit button, add a darker green color */
        input[type=submit]:hover {
          background-color: #45a049;
        }

        /* Add a background color and some padding around the form */
        .container {
          border-radius: 5px;
          background-color: #f2f2f2;
          padding: 20px;
        }

        body {
          background-size: cover;
          background-image: url('https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcR70TDTAUcRk7Q7W2nK-
aIqsmoTN6VruMi0mA&usqp=CAU');
        }
    </style>
</head>
<body>
```

```
<h3>Enter your Details for Loan Approval Prediction</h3>


<div class="container">
    <form action='/submit' method='POST'>
        <label for="Gender">Gender</label>
        <select id="Gender" name="Gender">
            <option value=0>Male</option>
            <option value=1>Female</option>
        </select>


        <label for="Married">Married</label>
        <select id="Married" name="Married">
            <option value=1>Yes</option>
            <option value=0>No</option>
        </select>


        <label for="Dependents">Dependents</label>
        <input type="number" id="Dependents" min=0 max=10 name="Dependents"
placeholder="No of Dependents on you...">


        <label for="Education">Education</label>
        <select id="Education" name="Education">
            <option value=1>Graduate</option>
            <option value=0>Not Graduate</option>
        </select>


        <label for="Self_Employed">Self Employed</label>
        <select id="Self_Employed" name="Self_Employed">
            <option value=1>Yes</option>
            <option value=0>No</option>
        </select>


        <label for="ApplicantIncome">Applicant Income</label>

        <input type="number" min=1000 id="ApplicantIncome" name="ApplicantIncome"
placeholder="Your Income...">


        <label for="CoapplicantIncome">CO Applicant Income</label>

        <input type="number" min=0 id="CoapplicantIncome" name="CoapplicantIncome"
placeholder="Your Co Applicant Income...">


        <label for="LoanAmount">Loan Amount</label>

        <input type="number" min=0 id="LoanAmount" name="LoanAmount"
placeholder="Enter the Loan Amount...">


        <label for="Loan_Amount_Term">Loan Amount Term</label>

        <input type="number" min=30 max=15000 id="Loan_Amount_Term"
name="Loan_Amount_Term" placeholder="Enter the Term Loan Amount in days...">


        <label for="Credit_History">Credit History</label>

        <input type="number" min=0 max=5 id="Credit_History" name="Credit_History"
placeholder="Enter Your Previous Credit History...">


        <label for="Property_Area">Property Area</label>

        <select id="Property_Area" name="Property_Area">

          <option value=2>Urban</option>

          <option value=0>Rural</option>

          <option value=1>Semi Urban</option>

        </select>


        <input type='submit' value='Submit'>

    </form>

</div>
```

```
</body>
</html>
```

output.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Loan Approval Prediction</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: 'Arial', sans-serif;
            background: url('../static/close-up-holding.jpg') no-repeat center center fixed;
            background-size: cover;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
        }

        .content {
            display: flex;
            justify-content: center;
            align-items: center;
            width: 100%;
            height: 100%;
        }

        .title-box {
            background-color: rgba(255, 255, 255, 0.8); /* Semi-transparent white background */
            padding: 30px 40px;
            border-radius: 15px;
            text-align: center;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2); /* Soft shadow for depth */
            max-width: 600px;
        }

        .title {
            font-size: 32px;
            font-weight: bold;
            color: #333;
            margin-bottom: 20px;
        }

        .result {
            font-size: 24px;
            color: #555;
        }
    </style>
</head>
<body>
    <div class="content">
        <div class="title-box">
            <div class="title">Loan Approval Prediction</div>
            <div class="result">{{ result }}</div>
        </div>
    </div>
</body>
</html>
```

output1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Loan Approval Prediction</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: 'Arial', sans-serif;
            background: url('../static/variable-home-loans.jpg') no-repeat center center fixed;
            background-size: cover;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
        }

        .content {
            display: flex;
            justify-content: center;
            align-items: center;
            width: 100%;
            height: 100%;
        }

        .title-box {
            background-color: rgba(255, 255, 255, 0.8); /* Semi-transparent white background */
            padding: 30px 40px;
            border-radius: 15px;
            text-align: center;
            box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2); /* Soft shadow for depth */
            max-width: 600px;
        }

        .title {
            font-size: 32px;
            font-weight: bold;
            color: #333;
            margin-bottom: 20px;
        }

        .result {
            font-size: 24px;
            color: #555;
        }
    </style>
</head>
<body>
    <div class="content">
        <div class="title-box">
```

```
            <div class="title">Loan Approval Prediction</div>

            <div class="result">{{ result }}</div>

        </div>

    </div>

</body>

</html>
```

10.2 GitHub & Project Demo Link:

https://github.com/Adrangi-Mohan-Vamsi-4501/Smart-Lender---Applicant-Credibility-Prediction-for-Loan-Approval-using-Machine-Learning

10.3 Demonstration Video link:

https://drive.google.com/file/d/1Hh310pSgi2cni25vpxa3M7fP_W_PpteD/view?usp=sharing