**Loan Prediction Model**

**Machine Learning**

**(Assessment 4)**

C.P.Narasinghe MAHNDSE232F-022

N.S.K.Gimhan MAHNDSE232F-050

National Institute of Business Management

**Declaration**

We, the undersigned, hereby declare that the project entitled "Loan Prediction Model" is our original work and has not been submitted for any Higher National Diploma or similar qualification at any institution. To the best of our knowledge and belief, this project does not contain any material previously published or written by others, except where due reference is made in the text. We also declare that we have not received any help in writing this project that is not properly acknowledged.

We further consent to the inclusion of this project report in the library and we agree to make it available for photocopying and interlibrary loans, should it be accepted. We also authorize, should it be accepted, that the title and summary of our project be made available to external organizations.

Project Members:

C.P.Narasinghe MAHNDSE232F-022

N.S.K.Gimhan MAHNDSE232F-050

**Acknowledgment**

We want to thank everyone who contributed to the completion of this project. First of all, our deepest thanks go to our supervisor, Mrs. Kaushalya, for guidance, support, and valuable feedback on the development of the loan prediction model.

We would like to extend our gratitude to our classmates and friends who helped us a lot with this project by providing support, encouragement, and assistance. Their insight and suggestions have been very instrumental in reaching the end result.

We also recognize the open-source community that makes available the libraries and tools used in this project. The availability of such resources is very important in implementing the different parts of our model.

Lastly, we would like to extend our sincerest gratitude to our families, who have supported and motivated us relentlessly in this endeavor.

**Contents**

[**Introduction** 1](#_Toc174125678)

[**1.1 Background and Motivation** 1](#_Toc174125679)

[**1.2 Objectives of the Report** 1](#_Toc174125680)

[**1.3 Scope of the Study** 1](#_Toc174125681)

[**Literature Review** 2](#_Toc174125682)

[**2.1 Overview of Machine Learning Approaches** 2](#_Toc174125683)

[**2.2 Previous Work on Predictive Models for Loan Prediction** 2](#_Toc174125684)

[**2.3 Summary of Related Techniques and Tools** 2](#_Toc174125685)

[**Data Exploration** 2](#_Toc174125686)

[**3.1 Dataset Description** 3](#_Toc174125687)

[**3.2 Exploratory Data Analysis (EDA)** 3](#_Toc174125688)

[**Feature Engineering** 4](#_Toc174125689)

[**4.1 Feature Selection** 4](#_Toc174125690)

[**4.2 Feature Extraction** 4](#_Toc174125691)

[**4.3 Feature Scaling and Normalization** 4](#_Toc174125692)

[**4.4 Rationale for Selected Features** 4](#_Toc174125693)

[**Model Selection** 5](#_Toc174125694)

[**5.1 K – Nearest Neighbor Algorithm** 5](#_Toc174125695)

[**5.2 Support Vector Machine (SVM) Algorithm** 8](#_Toc174125696)

[**5.3 Decision Tree Algorithm** 11](#_Toc174125697)

[**5.4 Training the model** 14](#_Toc174125698)

[**Model Evaluation** 15](#_Toc174125699)

[**6.1 Evaluation Metrics** 15](#_Toc174125700)

[**6.2 Best Performing Model Identification** 20](#_Toc174125701)

[**Model Deployment** 21](#_Toc174125702)

[**7.1 Web Application Development** 21](#_Toc174125703)

[**7.2 Real-World Use and Predictions** 21](#_Toc174125704)

[**Results and Discussion** 23](#_Toc174125705)

[**8.1 Model Performance Summary** 23](#_Toc174125706)

[**8.2 Interpretation of Results** 23](#_Toc174125707)

[**8.3 Practical Implications of Findings** 26](#_Toc174125708)

[**Conclusion** 27](#_Toc174125709)

[**9.1 Summary of Findings** 27](#_Toc174125710)

[**9.2 Recommendations for Practical Implementation** 27](#_Toc174125711)

[**9.3 Suggestions for Future Work** 27](#_Toc174125712)

[**References** 28](#_Toc174125713)

**Table Of Figures**

[Figure 1 - Validation Accuracy for Different k Values 7](#_Toc174125483)

[Figure 2 - Validation Accuracy for Different C values 11](#_Toc174125484)

[Figure 3 - Confusion Matrix for Decision Tree 15](#_Toc174125485)

[Figure 4 - Receiver Operating Characteristic for Decision Tree 16](#_Toc174125486)

[Figure 5- ROC Curve for SVM 17](#_Toc174125487)

[Figure 6 - User Interface 1 22](#_Toc174125488)

[Figure 7 - User Interface 2 22](#_Toc174125489)

[Figure 8 - Distribution of SpplicantIncome 24](#_Toc174125490)

[Figure 9 - Distribution of Loan Amount 24](#_Toc174125491)

[Figure 10 - Distribution of Co-Applicant Income 25](#_Toc174125492)

[Figure 11 - Distribution of Loan Status 25](#_Toc174125493)

# **Introduction**

## **1.1 Background and Motivation**

In the financial industries, there is much importance attached to the efficiency of the loan approval process for optimum performance of the operation and satisfaction of customers. Traditional ways of loan approval are manual and time-consuming, most often resulting in delays and inefficiencies. By embracing machine learning, financial institutions will be in a position to develop predictive models that enhance decision-making, smoothen operations, and respond to loan applications quickly. This project, therefore, is set to help resolve these challenges by coming up with a machine learning model that seeks to predict the result of loan approval and hence improve on its accuracy and efficiency.

## **1.2 Objectives of the Report**

This report strives to

• Create a machine learning model for prediction of loan approval based on a variety of features pertaining to the applicant.

• Develop a Streamlit WebApp that interacts with the predictive model for users to receive real-time predictions regarding loan approval.

• Provide practical insights and recommendations on the performance of the predictive model.

## **1.3 Scope of the Study**

All this research encompasses all the studies, designs, and evaluations for the project, from the Kaggle download all the way to the loan prediction model. The project includes data preprocessing, feature engineering, and model training and evaluation, in addition to building a web application that is easy to use and serves the purpose of demoing the model and being able to make real-time predictions.

# **Literature Review**

## **2.1 Overview of Machine Learning Approaches**

Machine learning is a general term that uses techniques generally to help enhance analytic insights and predictive knowledge related to the patterns in datasets. When it comes to predicting loan approval, techniques from the subfield of supervised inference learning come in. For instance, logistic regression and decision trees, among others, are some of the techniques developed nowadays in that field to make prognoses about outcomes based on historical data. These techniques are applied where the goal is to categorize and predict based on input features.

## **2.2 Previous Work on Predictive Models for Loan Prediction**

Several studies have thus been forwarded pertaining to loan prediction using machine learning models, among them logistic regression, decision trees, ensembling in the form of random forest and gradient boosting. All these models range in terms of complexity, performance of the results, accuracy, and interpretability. Literature reviewed proves the effectiveness of machine learning in increasing accuracy and efficiency during a loan approval process.

## 

## **2.3 Summary of Related Techniques and Tools**

Modeling for loan approvals mostly relies on Python libraries and tools. Among the libraries, the most prominent ones are Pandas for data manipulation, scikit-learn for implementing machine-learning algorithms, and visualization tools such as Matplotlib and Seaborn for data analysis and presentation purposes. Such tools are applied in the course of model development from data preprocessing to the last stage of model evaluation; therefore, they form a core part in developing an effective predictive model.

# **Data Exploration**

## **3.1 Dataset Description**

The dataset for the project at hand has been collected from Kaggle and is aimed at giving an insight into the process of loan approval. It contains many features that may be useful in the prediction of the results for loan approval, including gender, marital status, number of dependents, education, self-employment status, income, loan amount, loan term, credit history, and property area. Together, these features provide a good insight into the factors that have a bearing on loan approval decisions

## **3.2 Exploratory Data Analysis (EDA)**

• Loading Data: The dataset shall be first loaded using the pandas package that will help do some preliminary data exploration and view the structure of it.

•  Data Overview: Computation of summary statistics and distributions is done towards inferring the underlying patterns in the data, like distributions of features and their possible correlations.

• Missing Value Handling: Detection of missing data in the dataset, cleaning it with appropriate methods of its imputation or removal to maintain dataset integrity.

• Outlier identification and handling: Identify outliers and treat them in such a way that the effect of this outlier on model performance is reduced so that the prediction can go more accurately and reliably.

• Data Preprocessing: Categorical variables will be encoded to put them into a numerical format so the machine learning algorithms could understand them, and the feature scaling will be done if required for standardizing data to be fed into the model for better performance.

This thorough exploration and preprocessing of the dataset set the foundation for developing a robust loan prediction model.

# **Feature Engineering**

## **4.1 Feature Selection**

In this stage, features will be filtered based on their relevance and impact on the prediction of loan approval. Strong features that include income, loan amount, and credit history are selected since they have direct implications on the probability of a loan approval. At this stage, exploratory data analysis and domain knowledge are used in the identification of features that affect the probability of approval so that the model is trained using the most relevant predictors.

## **4.2 Feature Extraction**

## Additional features can also be added by deriving new ones based on existing features. This can be done through the creation of an interaction term or composite feature that holders have relations between variables, such as combining income with loan amount to come up with a debt-to-income ratio. Feature extraction will then help capture more complex patterns, hence boosting the predictive power of the model.

## **4.3 Feature Scaling and Normalization**

This would involve feature scaling or normalization to make sure the range of values of the different features is standard. This process makes sure that all features are contributing equally towards the model's training process and not dominated by any one feature on the outcome. One may use min-max scaling or standardization in order to attain uniformity across features.

## **4.4 Rationale for Selected Features**

The rationale behind selecting specific features lies in their demonstrated predictive power and relevance to the loan approval decision-making process. Features like income, loan amount, and credit history are directly related to financial stability and repayment capability, making them critical for accurate loan prediction. This careful selection and engineering of features are aimed at building a robust and reliable model.

# **Model Selection**

## **5.1 K – Nearest Neighbor Algorithm**

K-Nearest Neighbors (KNN) is a simple, yet effective, supervised learning algorithm used for classification and regression tasks. The core idea behind KNN is to classify or predict the value of a data point based on the labels or values of its nearest neighbors in the feature space. In classification, KNN assigns a class label to a data point based on the majority class among its

𝑘 closest neighbors. In regression, it predicts the value by averaging the values of its 𝑘

k nearest neighbors. KNN is non-parametric, meaning it does not make assumptions about the underlying data distribution, and is particularly useful for problems with complex decision boundaries.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

from imblearn.over\_sampling import SMOTE

import joblib

from visualization import plot\_feature\_distributions, plot\_class\_distribution, plot\_confusion\_matrix, plot\_roc\_curve

# Load the dataset

df = pd.read\_csv(r"E:\train.csv")

# Handle outliers for 'ApplicantIncome'

max\_threshold\_income = df['ApplicantIncome'].quantile(0.95)

min\_threshold\_income = df['ApplicantIncome'].quantile(0.05)

df = df[(df['ApplicantIncome'] >= min\_threshold\_income) & (df['ApplicantIncome'] <= max\_threshold\_income)]

# Handle outliers for 'LoanAmount'

max\_threshold\_loan = df['LoanAmount'].quantile(0.95)

min\_threshold\_loan = df['LoanAmount'].quantile(0.05)

df = df[(df['LoanAmount'] >= min\_threshold\_loan) & (df['LoanAmount'] <= max\_threshold\_loan)]

# Fill missing values for numeric columns

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

df['CoapplicantIncome'] = df['CoapplicantIncome'].fillna(df['CoapplicantIncome'].median())

# Fill missing values for categorical columns with the mode

categorical\_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area']

for column in categorical\_columns:

    df[column] = df[column].fillna(df[column].mode()[0])

df['Loan\_Amount\_Term'] = df['Loan\_Amount\_Term'].fillna(df['Loan\_Amount\_Term'].median())

df['Credit\_History'] = df['Credit\_History'].fillna(df['Credit\_History'].median())

df['Loan\_Status'] = df['Loan\_Status'].fillna(df['Loan\_Status'].mode()[0])

# Encode categorical columns

label\_encoders = {}

for column in categorical\_columns:

    le = LabelEncoder()

    df[column + '\_T'] = le.fit\_transform(df[column])

    label\_encoders[column] = le

# Convert 'Dependents' column to numeric

df['Dependents'] = df['Dependents'].replace({'3+': 3}).astype(int)

# Drop original categorical columns and other unnecessary columns

x = df.drop(columns=['Loan\_Status'] + categorical\_columns + ['Loan\_ID'])

y = df['Loan\_Status']

# Encode target variable

le\_Loan\_Status = LabelEncoder()

y = le\_Loan\_Status.fit\_transform(y)

# Handle imbalance using SMOTE

sm = SMOTE(random\_state=42)

x\_res, y\_res = sm.fit\_resample(x, y)

# Split the data into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_res, y\_res, test\_size=0.3, random\_state=42)

# Train the model

model = KNeighborsClassifier(n\_neighbors=19)

model.fit(x\_train, y\_train)

# Evaluate the model

accuracy = model.score(x\_test, y\_test)

print("Model Accuracy:", accuracy)

y\_pred = model.predict(x\_test)

y\_scores = model.predict\_proba(x\_test)[:, 1]

# Save the model

joblib.dump(model, 'Loan\_Prediction\_Model.pkl')

print(x\_train.columns.tolist())

# Plot distributions and performance metrics

plot\_feature\_distributions(df, ['ApplicantIncome', 'LoanAmount', 'CoapplicantIncome'])

plot\_class\_distribution(y\_res)

plot\_confusion\_matrix(y\_test, y\_pred, labels=le\_Loan\_Status.classes\_)

plot\_roc\_curve(y\_test, y\_scores)

A graph with numbers and lines

Description automatically generated

Figure 1 - Validation Accuracy for Different k Values

## **5.2 Support Vector Machine (SVM) Algorithm**

Support Vector Machine (SVM) is a powerful supervised learning algorithm used primarily for classification tasks, although it can also be applied to regression. The goal of SVM is to find the optimal hyperplane that separates different classes in the feature space with the maximum margin. The hyperplane is chosen to maximize the distance between the closest points of different classes, known as support vectors. SVM is effective in high-dimensional spaces and can handle non-linear classification through the use of kernel functions, which transform the data into a higher-dimensional space where a linear separation is possible.

import pandas as pd

from sklearn.datasets import load\_digits

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_selection import SelectKBest,chi2,mutual\_info\_classif

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import cross\_val\_score

import matplotlib.pyplot as plt

from sklearn.metrics import classification\_report

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

df = pd.read\_csv(r"E:\train.csv")

df

print("Missing values in each column:")

print(df.isnull().sum())

from sklearn.preprocessing import LabelEncoder

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

df['CoapplicantIncome'] = df['CoapplicantIncome'].fillna(df['CoapplicantIncome'].median())

# Fill missing values for categorical columns with the mode

categorical\_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area']

for column in categorical\_columns:

    df[column] = df[column].fillna(df[column].mode()[0])

# Fill missing values for 'Loan\_Amount\_Term' and 'Credit\_History' with the median

df['Loan\_Amount\_Term'] = df['Loan\_Amount\_Term'].fillna(df['Loan\_Amount\_Term'].median())

df['Credit\_History'] = df['Credit\_History'].fillna(df['Credit\_History'].median())

# Fill missing values for the target column (if any)

df['Loan\_Status'] = df['Loan\_Status'].fillna(df['Loan\_Status'].mode()[0])

# Encode categorical columns

label\_encoders = {}

for column in categorical\_columns:

    le = LabelEncoder()

    df[column + '\_T'] = le.fit\_transform(df[column])

    label\_encoders[column] = le

# Convert 'Dependents' column to numeric after filling NaNs

df['Dependents'] = df['Dependents'].replace({'3+': 3}).astype(float).fillna(0).astype(int)

# Drop original categorical columns if necessary

df\_encoded = df.drop(columns=categorical\_columns)

# Display the processed DataFrame

print(df\_encoded.head())

print("Missing values in each column:")

print(df.isnull().sum())

max\_thresold=df['ApplicantIncome'].quantile(0.95)

max\_thresold

df[df['ApplicantIncome']>=max\_thresold]

min\_thresold=df['ApplicantIncome'].quantile(0.05)

min\_thresold

df[df['ApplicantIncome']<=min\_thresold]

new=df[(df['ApplicantIncome']>=min\_thresold) & (df['ApplicantIncome']<=max\_thresold)]

new

x = new.drop('Loan\_Status',axis='columns')

x

y = new['Loan\_Status']

y

label\_encoders = {

    'Gender': LabelEncoder(),

    'Married': LabelEncoder(),

    'Dependents': LabelEncoder(),

    'Education': LabelEncoder(),

    'Self\_Employed': LabelEncoder(),

    'Property\_Area': LabelEncoder()

}

le\_Loan\_Status = LabelEncoder()

y\_encoded = le\_Loan\_Status.fit\_transform(y)

new.loc[:, 'Loan\_Status\_T'] = y\_encoded

new.head()

x\_new = x.drop (['Gender','Married','Education','Self\_Employed','Property\_Area','Loan\_ID'],axis = 'columns')

x\_new.head()

y\_new = new['Loan\_Status\_T']

y\_new

# Load the digits dataset

dataset = load\_digits()

# split the dataset into features (x) and target labels (y)

x =dataset.data

y=dataset.target

# split the data into training and testing sets

x\_train, x\_test , y\_train , y\_test =train\_test\_split(x\_new,y\_new,test\_size=0.2,random\_state=42)

C\_values = [0.01, 0.1, 1, 10, 100]

scores = []

# Evaluate each C value

for C in C\_values:

    svm = SVC( C=C)

    svm.fit(x\_train, y\_train)

    score = svm.score(x\_test, y\_test)

    scores.append(score)

# Plot the results

plt.figure(figsize=(10, 6))

plt.plot(C\_values, scores, marker='o')

plt.xscale('log')  # Use logarithmic scale for better visualization

plt.xlabel('C value (log scale)')

plt.ylabel('Validation Accuracy')

plt.title('Validation Accuracy for Different C Values')

plt.show()

# Find the best C value

best\_C = C\_values[scores.index(max(scores))]

print(f'The best C value is {best\_C}')

model= SVC(class\_weight='balanced',C=0.01)

model.fit(x\_train,y\_train)

y\_pred = model.predict(x\_test)

cm = confusion\_matrix(y\_test,y\_pred)

print (cm)

accuracy = accuracy\_score (y\_test,y\_pred)

print("Accuracy:" , accuracy)

print (classification\_report(y\_test,y\_pred))

A graph with a line

Description automatically generated

Figure 2 - Validation Accuracy for Different C values

## **5.3 Decision Tree Algorithm**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import LabelEncoder

from imblearn.over\_sampling import SMOTE

import joblib

from visualization import plot\_feature\_distributions, plot\_class\_distribution, plot\_confusion\_matrix, plot\_roc\_curve

# Load the dataset

df = pd.read\_csv(r"E:\train.csv")

# Handle outliers for 'ApplicantIncome'

max\_threshold\_income = df['ApplicantIncome'].quantile(0.95)

min\_threshold\_income = df['ApplicantIncome'].quantile(0.05)

df = df[(df['ApplicantIncome'] >= min\_threshold\_income) & (df['ApplicantIncome'] <= max\_threshold\_income)]

# Handle outliers for 'LoanAmount'

max\_threshold\_loan = df['LoanAmount'].quantile(0.95)

min\_threshold\_loan = df['LoanAmount'].quantile(0.05)

df = df[(df['LoanAmount'] >= min\_threshold\_loan) & (df['LoanAmount'] <= max\_threshold\_loan)]

# Fill missing values for numeric columns

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

df['CoapplicantIncome'] = df['CoapplicantIncome'].fillna(df['CoapplicantIncome'].median())

# Fill missing values for categorical columns with the mode

categorical\_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area']

for column in categorical\_columns:

    df[column] = df[column].fillna(df[column].mode()[0])

df['Loan\_Amount\_Term'] = df['Loan\_Amount\_Term'].fillna(df['Loan\_Amount\_Term'].median())

df['Credit\_History'] = df['Credit\_History'].fillna(df['Credit\_History'].median())

df['Loan\_Status'] = df['Loan\_Status'].fillna(df['Loan\_Status'].mode()[0])

# Encode categorical columns

label\_encoders = {}

for column in categorical\_columns:

    le = LabelEncoder()

    df[column + '\_T'] = le.fit\_transform(df[column])

    label\_encoders[column] = le

# Convert 'Dependents' column to numeric

df['Dependents'] = df['Dependents'].replace({'3+': 3}).astype(int)

# Drop original categorical columns and other unnecessary columns

x = df.drop(columns=['Loan\_Status'] + categorical\_columns + ['Loan\_ID'])

y = df['Loan\_Status']

# Encode target variable

le\_Loan\_Status = LabelEncoder()

y = le\_Loan\_Status.fit\_transform(y)

# Handle imbalance using SMOTE

sm = SMOTE(random\_state=42)

x\_res, y\_res = sm.fit\_resample(x, y)

# Split the data into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_res, y\_res, test\_size=0.3, random\_state=42)

# Train the Decision Tree model

model = DecisionTreeClassifier(random\_state=42)

model.fit(x\_train, y\_train)

# Evaluate the model

accuracy = model.score(x\_test, y\_test)

print("Model Accuracy:", accuracy)

y\_pred = model.predict(x\_test)

y\_scores = model.predict\_proba(x\_test)[:, 1]

# Save the model

joblib.dump(model, 'Loan\_Prediction\_Model\_DecisionTree.pkl')

print(x\_train.columns.tolist())

# Plot distributions and performance metrics

plot\_feature\_distributions(df, ['ApplicantIncome', 'LoanAmount', 'CoapplicantIncome'])

plot\_class\_distribution(y\_res)

plot\_confusion\_matrix(y\_test, y\_pred, labels=le\_Loan\_Status.classes\_)

plot\_roc\_curve(y\_test, y\_scores)

## **5.4 Training the model**

To build and evaluate the predictive models, the dataset is divided into two distinct subsets: training and testing sets. The training set is used to fit the models, allowing them to learn from the data and identify patterns that correlate with loan approval outcomes. This process involves feeding the model with input features and the corresponding target values, enabling it to learn and adjust its parameters.

The testing set, which is separate from the training data, is used to assess the model's performance and generalizability. By evaluating the model on this unseen data, we can measure how well it predicts loan approvals for new, unseen applications. This approach helps ensure that the model is not only accurate on the training data but also performs effectively in real-world scenarios.

x\_train, x\_test , y\_train , y\_test =train\_test\_split(x\_new,y\_new,test\_size=0.2,random\_state=42)

# **Model Evaluation**

## **6.1 Evaluation Metrics**

Model performance is evaluated using the following metrics:

Accuracy: The proportion of correctly predicted instances.

Precision: The proportion of true positives among predicted positives.

Recall: The proportion of true positives among actual positives.

F1-Score: The harmonic mean of precision and recall.

ROC-AUC Curve: The area under the receiver operating characteristic curve.

1. **Decision Tree Algorithm**

Model Accuracy: 0.7860696517412935

Precision: 0.7938144329896907

Recall: 0.77

F1-Score: 0.7817258883248731

ROC-AUC: 0.7859900990099011

A blue squares with white text

Description automatically generated

Figure 3 - Confusion Matrix for Decision Tree

A graph with a line and a blue line

Description automatically generated with medium confidence

Figure 4 - Receiver Operating Characteristic for Decision Tree

Model Accuracy: The model achieved an accuracy of approximately 78.61%, meaning that 78.61% of the predictions made by the model were correct.

Precision: The precision of 79.38% indicates that out of all the positive predictions made by the model, 79.38% were actually correct.

Recall: The recall score of 77% shows that the model was able to correctly identify 77% of the actual positive cases.

F1-Score: The F1-score of 78.17% represents a balance between precision and recall, providing a single metric that accounts for both false positives and false negatives.

ROC-AUC: The ROC-AUC score of 78.60% measures the model's ability to distinguish between positive and negative classes, with a higher score indicating better performance.

1. **Support Vector Machine Algorithm (SVM)**

Accuracy: 0.2072072072072072

Precision: 0.0

Recall: 0.0

F1-Score: 0.0

ROC-AUC: 0.47282608695652173

A graph with a red line

Description automatically generated

Figure 5- ROC Curve for SVM

* Accuracy **(20.72%)**: The accuracy is relatively low, suggesting that the model correctly predicted only about 21% of the instances. This indicates that the model is struggling to differentiate between loan-approved and loan-rejected instances effectively.
* **Precision (0.0)**: Precision being 0.0 indicates that none of the instances predicted as positive (e.g., loan approved) were actually correct. This is problematic, as it shows the model has a high rate of false positives, predicting many loans as approved when they should not be.
* **Recall (0.0)**: Recall being 0.0 suggests that the model did not correctly identify any of the true positive cases (e.g., actual loan approvals). The model fails to capture the actual positive instances, leading to no correct positive predictions.
* **F1-Score (0.0)**: The F1-score, which is the harmonic mean of precision and recall, also being 0.0, confirms that the model's predictions are entirely inaccurate for the positive class. This metric combines both precision and recall, so if either is poor, the F1-score will reflect that.
* **ROC-AUC (0.47)**: The ROC-AUC score being close to 0.5 suggests that the model's ability to distinguish between the classes is no better than random guessing. A score of 0.5 would indicate random performance, so 0.47 shows that the model isn't effectively separating positive and negative classes.

1. **K – Nearest Neighbor Algorithm**

Model Accuracy: 0.5433070866141733

Precision: 0.5071428571428571

Recall: 0.6016949152542372

F1-Score: 0.5503875968992248

ROC-AUC: 0.58714481555334

1. **Model Accuracy: 0.543**

The model correctly predicted the loan approval status for approximately 54.3% of the test instances. This metric gives a general idea of the model’s performance but does not differentiate between the types of errors (false positives vs. false negatives).

**2. Precision: 0.507**

Out of all the instances that the model predicted as positive (i.e., approved loans), about 50.7% were actually positive. Precision indicates how well the model identifies positive cases without making many false positive predictions.

**3. Recall: 0.602**

The model successfully identified 60.2% of all actual positive cases (i.e., actual approved loans). Recall measures how well the model captures all the true positives from the dataset, highlighting its ability to find all relevant instances.

**4. F1-Score: 0.550**

The F1-Score, which is the harmonic mean of precision and recall, is 55.0%. This metric provides a balanced measure of both precision and recall, offering a single score to evaluate the model’s overall performance. A higher F1-Score indicates a better balance between precision and recall.

**5. ROC-AUC: 0.587**

The ROC-AUC score of 0.587 suggests that the model has moderate performance in distinguishing between approved and non-approved loans. ROC-AUC measures the model's ability to discriminate between classes across various threshold settings. A score closer to 1 indicates better performance, while a score closer to 0.5 implies that the model performs similarly to random guessing.

## **6.2 Best Performing Model Identification**

The **Decision Tree algorithm** is the best choice for training your model based on the following reasons:

* It has the highest accuracy (78.6%) and the best balance between precision and recall.
* The F1-Score of 78.2% indicates strong overall performance.
* The ROC-AUC score of 78.6% reflects its effective discrimination ability between loan approval and rejection.

In contrast, the KNN algorithm has lower accuracy and less balanced performance metrics, while the SVM algorithm performs poorly across all metrics, making it unsuitable for this particular task.

# **Model Deployment**

The deployment strategy for the trained model involves leveraging a Streamlit web application to enable real-time predictions. This approach ensures that users can easily interact with the model and obtain loan eligibility predictions based on their inputs.

## **7.1 Web Application Development**

Streamlit Implementation: A Streamlit-based web application is developed to create an intuitive and user-friendly interface. This application allows users to enter details of their loan application, such as income, loan amount, and credit history, and receive immediate predictions on their loan eligibility.

Integration of the Model: The trained model is seamlessly integrated into the Streamlit application. This integration enables the application to utilize the model's predictive capabilities, providing accurate and timely loan approval predictions based on the data entered by users.

## **7.2 Real-World Use and Predictions**

Application Functionality: The functionality of the web application is demonstrated, showcasing how users can interact with the system. The application’s real-time predictions are presented to illustrate the model's practical application and effectiveness in assessing loan applications.

This deployment strategy not only enhances accessibility but also provides a practical tool for users to assess their loan eligibility in real time, leveraging the trained model's predictive power effectively.

A screenshot of a computer

Description automatically generated

Figure 6 - User Interface 1

A screenshot of a computer

Description automatically generated

Figure 7 - User Interface 2

# **Results and Discussion**

## **8.1 Model Performance Summary**

The Decision Tree model demonstrates strong performance in predicting loan approval, with the following metrics:

* Model Accuracy: 0.786, indicating that approximately 78.6% of the predictions made by the model are correct. This reflects the model’s overall ability to correctly classify loan applications as approved or not approved.
* Precision: 0.794, showing that 79.4% of the instances predicted as approved were actually approved. This metric indicates the model’s effectiveness in minimizing false positives.
* Recall: 0.770, meaning that the model correctly identified 77.0% of all actual approved loans. This metric highlights the model’s ability to capture most of the approved loan cases.
* F1-Score: 0.782, representing the harmonic mean of precision and recall. This score balances the trade-off between precision and recall, showing a solid overall performance.
* ROC-AUC: 0.786, which measures the model’s ability to distinguish between the positive and negative classes. A score of 0.786 indicates a high capability in separating approved and non-approved loans.

## **8.2 Interpretation of Results**

The Decision Tree model’s metrics suggest a robust performance in predicting loan approval outcomes. The high accuracy and F1-Score indicate that the model is effective in both identifying approved loans and minimizing classification errors. The precision and recall values demonstrate a good balance between avoiding false positives and capturing actual positives, making the model a reliable tool for predicting loan approvals.

A graph of a distribution of a number of individuals

Description automatically generated with medium confidence

Figure 8 - Distribution of SpplicantIncome

A graph of a distribution of loan

Description automatically generated

Figure 9 - Distribution of Loan Amount

A graph of a number of people

Description automatically generated

Figure 10 - Distribution of Co-Applicant Income

A blue rectangular bars with white text

Description automatically generated

Figure 11 - Distribution of Loan Status

## **8.3 Practical Implications of Findings**

The strong performance of the Decision Tree model has significant practical implications:

* Enhanced Loan Processing: By integrating this model into loan approval systems, financial institutions can streamline their decision-making processes, resulting in more accurate and faster loan approvals.
* Reduced Manual Review: The model’s high accuracy and reliability can reduce the need for extensive manual review of loan applications, leading to cost savings and improved operational efficiency.
* Improved Customer Experience: Faster and more accurate loan approval decisions can enhance the customer experience, as applicants receive timely feedback on their loan status.

In summary, the Decision Tree model offers a valuable tool for automating and improving the loan approval process, with potential benefits for both financial institutions and their customers.

# **Conclusion**

## **9.1 Summary of Findings**

This project involved developing and evaluating machine learning models to predict loan approval. The Decision Tree model emerged as the most effective algorithm, achieving an accuracy of 78.6%, a precision of 79.4%, a recall of 77.0%, an F1-Score of 78.2%, and an ROC-AUC of 78.6%. These metrics highlight the model's strong performance in correctly classifying loan applications. The application of this model was successfully demonstrated through a Streamlit web interface, which allows users to input loan application details and receive real-time predictions.

## **9.2 Recommendations for Practical Implementation**

To implement this model in a real-world setting, financial institutions should consider the following:

* Integration: Deploy the Decision Tree model within existing loan processing systems to automate and enhance the accuracy of loan approvals.
* User Interface: Utilize the Streamlit web application to provide a user-friendly interface for both loan officers and applicants, streamlining the loan application process.
* Continuous Monitoring: Regularly monitor and update the model to ensure it adapts to changing patterns in loan approval criteria and market conditions.

## **9.3 Suggestions for Future Work**

* Feature Expansion: Incorporate additional features such as customer behavior data or external financial indicators to improve model accuracy and robustness.
* Algorithm Exploration: Experiment with other algorithms, such as ensemble methods or deep learning approaches, to compare performance and potentially enhance predictive capabilities.
* Data Enrichment: Utilize larger and more diverse datasets to better generalize the model and address any potential biases.

# **References**

www.kaggle.com. (n.d.). *Loan Prediction Problem Dataset*. [online] Available at: <https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset>.

ProjectPro. (n.d.). How to Train a Machine Learning Model: The Complete Guide. [online] Available at: <https://www.projectpro.io/article/training-a-machine-learning-model/936>.

Kapkar, N. (2020). Training your First Machine Learning Model with sklearn. [online] Analytics Vidhya. Available at: <https://medium.com/analytics-vidhya/training-your-first-machine-learning-model-with-sklearn-e03d5de3bfba>.

‌Python Libraries: pandas, scikit-learn, imblearn, joblib

Frameworks: Streamlit for web application development

Data Analysis Tools: Jupyter Notebook for analysis and experimentation

‌