Department of Computer Science and Engineering
Brac University

Course Code & Name
CSE422, Artificial Intelligence Lab

Intelligent Loan Risk Assessment

A Comparative Analysis of SVM, Decision Tree, and Naive Bayes Algorithms for Venture Capital Investment Decisions

SUBMITTED BY:

Shakib Shariar (18201206)
Ashfaque Hossain (19101233)
Ulysse Tresor(21101333)

Department of Computer Science and Engineering

Table of Contents

Intelligent Loan Risk Assessment

A Comparative Analysis of SVM, Decision Tree, and Naive Bayes Algorithms for Venture Capital Investment Decisions

A. Introduction

In the dynamic landscape of venture capital investments, making informed decisions is crucial for the success of any venture capital firm. The project, titled "Intelligent Loan Risk Assessment," undertakes a comprehensive exploration of three powerful machine learning algorithms - Support Vector Machine (SVM), Decision Tree, and Naive Bayes - to assess their efficacy in predicting loan risk for potential venture capital investments. The primary objective of this project is to develop an intelligent loan risk assessment system that aids venture capital companies in making data-driven decisions. By employing sophisticated machine learning algorithms, we aim to provide insights into the comparative performance of SVM, Decision Tree, and Naive Bayes models in predicting loan outcomes, thus assisting venture capitalists in mitigating risks and maximizing returns.

The project focuses on utilizing a dataset containing key features such as applicant information, income details, loan amount, and credit history. The selected algorithms will undergo rigorous evaluation both before and after feature extraction, emphasizing the impact of dimensionality reduction techniques, specifically Principal Component Analysis (PCA).

B. Motivation

The motivation behind undertaking the project "Intelligent Loan Risk Assessment: A Comparative Analysis of SVM, Decision Tree, and Naive Bayes Algorithms for Venture Capital Investment Decisions" stems from a recognition of the pivotal role that data-driven decision-making plays in the success of venture capital investments. Several key motivations drive the project. In the contemporary landscape of venture capital, the abundance of data presents a valuable opportunity for leveraging machine learning algorithms to make informed investment decisions. The project seeks to harness this potential and contribute to the evolving paradigm of data-driven decision-making in the finance industry. Assessing loan risk is a critical aspect of venture capital investments. The project aims to develop a robust system that effectively evaluates the risk associated with loans, providing venture capitalists with tools to identify potential pitfalls and mitigate risks, thereby enhancing the overall success rate of investments.

C. Background

Through this project, we aim to contribute to the growing body of knowledge in the field of algorithmic decision-making in finance. By evaluating and comparing these algorithms, we intend to provide venture capitalists with valuable insights into the strengths and limitations of different approaches to loan risk assessment. Ultimately, the project aspires to empower decision-makers with the tools needed to make more informed and strategic investment decisions in the dynamic landscape of venture capital.

D. Dataset

. The dataset encompasses a variety of features related to loan applications, providing a nuanced understanding of the factors influencing loan approvals and rejections. The dataset contains 13 features.

Sl.	Attribute	Description
01	*Loan_ID*	Unique identifier for each loan application.
02	*Gender*	Gender of the loan applicant.
03	*Married*	Marital status of the applicant (Yes/No).
04	*Dependents*	Number of dependents associated with the applicant.
05	*Education*	Educational background of the applicant (Graduate/Not Graduate).
06	*Self_Employed*	Indicates whether the applicant is self-employed (Yes/No).
07	*ApplicantIncome*	Income of the applicant.
08	*CoapplicantIncome*	Income of the co-applicant (if applicable).
09	*LoanAmoun*	Amount of the loan applied for.
10	*Loan_Amount_Term*	Term of the loan in months.
11	*Credit_History*	Credit history of the applicant (1: Good, 0: Bad).
12	*Property_Area*	Area of the property associated with the loan (Urban/Rural/Semiurban).
13	*Loan_Status*	Outcome of the loan application (Y: Approved, N: Not Approved).

This is a classification problem. The goal is to predict the outcome of loan applications, which is represented by the "Loan_Status" feature. The outcome is categorical, indicating whether a loan is approved (Y) or not approved (N). The dataset aligns with the real-world loan application process, capturing essential details considered by lending institutions during the decision-making process. This includes demographic information, financial indicators, and credit history, all of which play a crucial role in determining loan approvals and rejections.

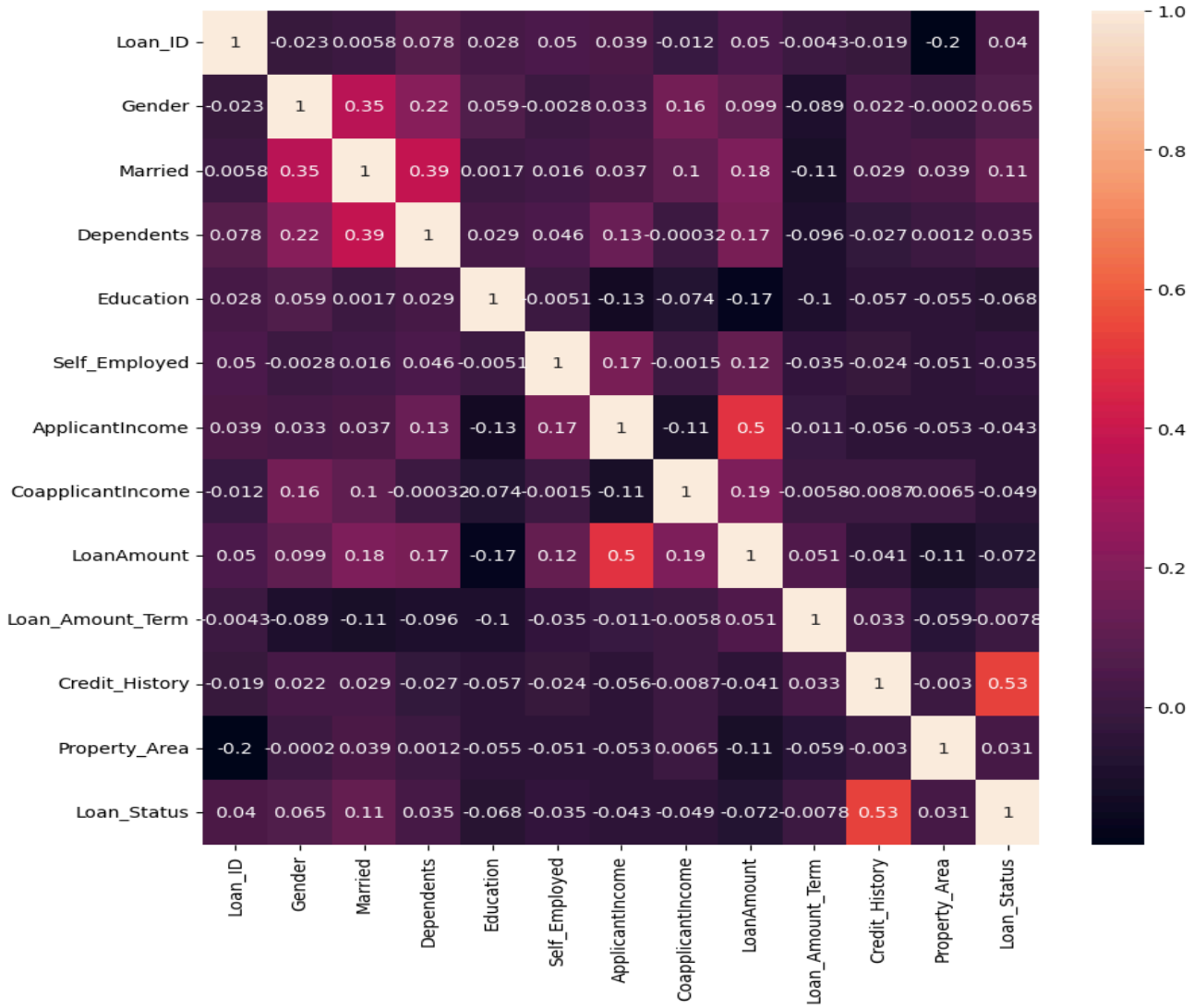
Types of Features:

- Quantitative Features:
- ApplicantIncome
- CoapplicantIncome
- LoanAmount
- Loan_Amount_Term

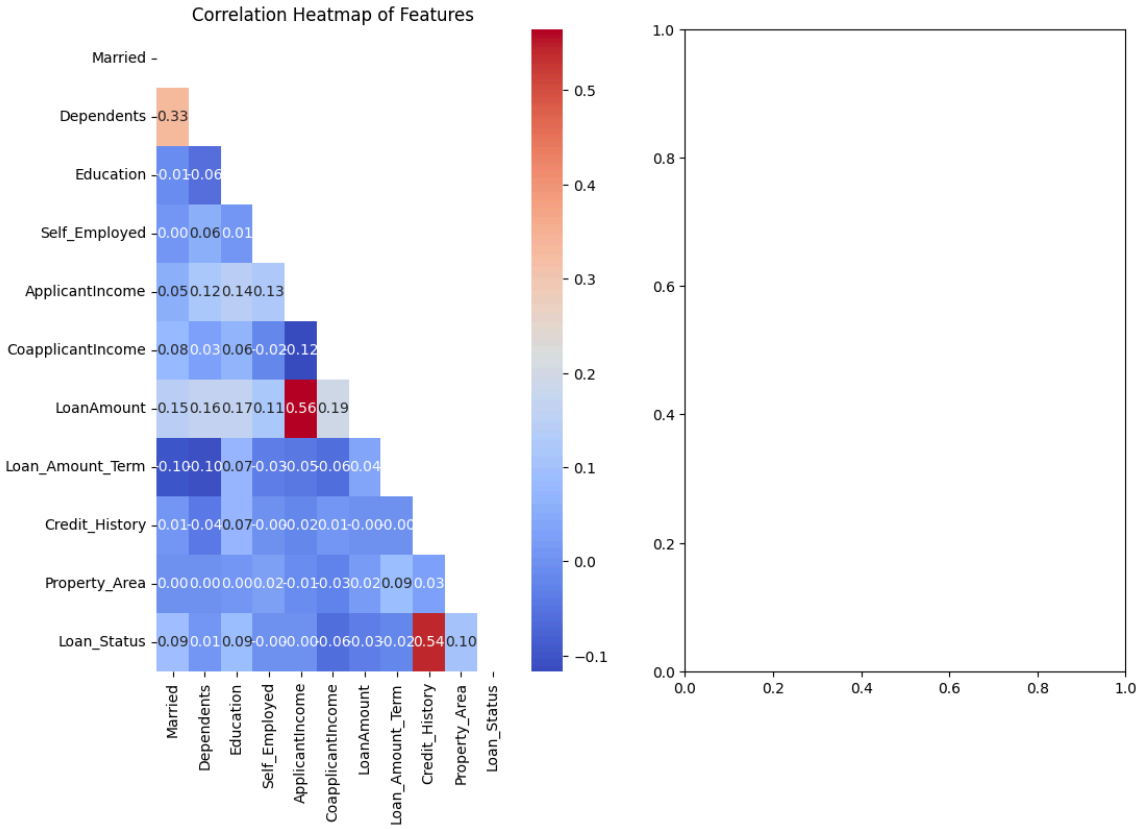
- Categorical Features:
- Loan_ID
- Gender

- Married
- Dependents
- Education
- Self_Employed
- Credit_History
- Property_Area
- Loan_Status

- ***Correlation of all features:***



- ***Correlation of all features after data-processing:***



- **Imbalanced Dataset**

  To determine if the dataset is imbalanced with respect to the output feature (in this case, "Loan_Status"), you can check the distribution of instances across the unique classes. If there is a significant imbalance, it may impact the performance of machine learning models, especially in classification tasks.

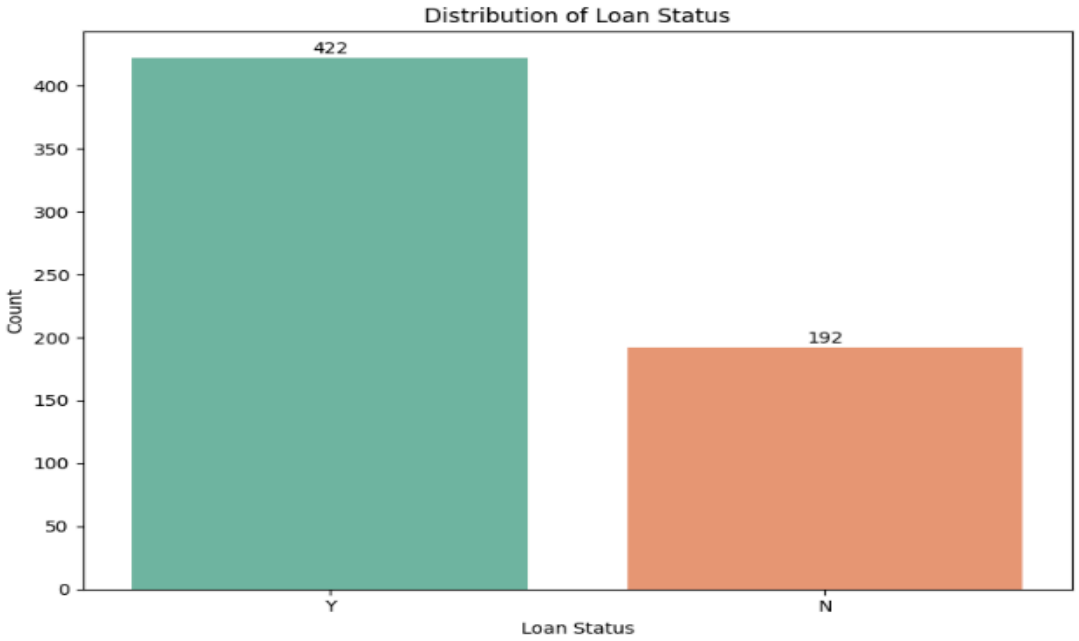


Fig. 2: Bar chart for Loan_Status distribution

E. Dataset Pre-processing

a. Handling NULL Values

Using methods like isnull().sum(), we can systematically inspect the dataset to reveal the extent of missing information across various features.

Once the NULL values are identified, the next step involves deciding on an appropriate strategy for handling them. Imputation, a common approach, entails filling the missing values with suitable substitutes. For numerical features, imputing techniques such as mean or median values can be employed, preserving the integrity of the dataset. Additionally, for categorical features, imputation with the mode (most frequent value) or a predefined value ensures a comprehensive treatment of missing data. Alternatively, depending on the scenario, one might choose to drop entire rows or columns using the dropna() function. Imputing values is essential for retaining data points that could provide valuable insights into the dataset. Deleting rows or columns with missing values, while simplifying the dataset, may lead to a loss of information, potentially compromising the completeness of the dataset.

b. Handling Categorical Values

1. Drop duplicates (ex: “data.drop_duplicates()”)
2. Replacing strings / Using numerical values
3. Convert Object Datatype to Float
4. Drop Unused features (Ex: Loan_ID, Gender..)
5. Calculating the Sum of Null cells in Columns
6. Replacing Null values with Mode of Column Values
7. Separate Array into Input and Output Components

.

F. Feature Scaling

In our loan prediction project, feature scaling is a crucial preprocessing step to ensure that the various input features, which may have different ranges, do not introduce bias or dominance issues in machine learning models sensitive to feature scales. One commonly used method for feature scaling is Min-Max Scaling, which transforms the data to a specific range, typically between 0 and 1. Using scikit-learn's MinMaxScaler, we can easily scale the features in our dataset. For example, assuming 'X' represents our feature matrix, the following code snippet demonstrates how to apply Min-Max Scaling:

```
from sklearn.preprocessing import MinMaxScaler

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the feature matrix
X_scaled = scaler.fit_transform(X)

# Replace the original features with the scaled features
X = pd.DataFrame(X_scaled, columns=X.columns)
```

G. Dataset Splitting

1. Random/Stratified:

In scikit-learn, the train_test_split function allows us to achieve either a random or stratified split. For instance, assuming 'X' is the feature matrix and 'Y' is the target variable (Loan_Status), the following code demonstrates a random split. This randomly assigns 70% of the data to the training set and 30% to the testing set. If a stratified split is preferred to maintain the distribution of the target variable, we can specify the stratify parameter. This ensures that the proportion of classes in the target variable 'Y' is preserved in both the training and testing sets, aiding in the model's generalization.

.

2. Train Set

The training set is a crucial component of our project, serving as the foundation for training machine learning models. This set comprises a subset of the data on which the models learn patterns and relationships. In our loan prediction project, the training set, represented by 'X_train' for features and 'Y_train' for the target variable, is essential for honing the model's ability to make accurate predictions. For example, following the random or stratified split, the training set can be accessed as below. These shapes provide insights into the size and composition of the training set, aiding in the evaluation of the model's performance during the training phase.

```
# Accessing the training set
print("Training set features shape:", X_train.shape)
print("Training set target variable shape:", Y_train.shape)
```

3. Test Set

Complementary to the training set, the test set acts as an independent dataset against which the trained model is evaluated for its predictive performance. In our loan prediction project, the test set, denoted by 'X_test' for features and 'Y_test' for the target variable, provides an unbiased assessment of the model's ability to generalize to unseen data. Accessing the test set post-split can be accomplished as follows:

```
# Accessing the training set
print("Training set features shape:", X_train.shape)
print("Training set target variable shape:", Y_train.shape)
```

These dimensions offer insights into the size and structure of the test set, enabling a comprehensive evaluation of the model's effectiveness in making predictions on new, unseen instances. The test set serves as a critical benchmark for assessing the model's real-world applicability and robustness.

H. Model Training and Testing

1. Support Vector Machine (SVM)

In the model training and testing phase of our loan prediction project, the Support Vector Machine (SVM) algorithm is employed to discern patterns and make predictions based on the input features. This process involves several key steps, beginning with the importation of necessary libraries, including the SVM classifier (`SVC`) and metrics for evaluation (`confusion_matrix` and `classification_report`). The dataset, previously split into training and testing sets, serves as the foundation for SVM model training.

Importing Libraries:

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report

# Assuming X_train, X_test, Y_train, Y_test are already defined from the dataset
splitting phase
```

SVM Model Initialization and Training:

```
# Accessing the training set
print("Training set features shape:", X_train.shape)
print("Training set target variable shape:", Y_train.shape)
```

Here, SVC is used to initialize the SVM model with a linear kernel and regularization parameter 'C' set to 1. The model is then trained using the training set.

SVM Model Testing:

```
# Predictions on the test set
y_pred_svm = svm_model.predict(X_test)

# Confusion Matrix and Classification Report
```

```
conf_matrix_svm = confusion_matrix(Y_test, y_pred_svm)
class_report_svm = classification_report(Y_test, y_pred_svm)

# Model Accuracy
accuracy_svm = svm_model.score(X_test, Y_test)

print("Confusion Matrix:")
print(conf_matrix_svm)

print("\nClassification Report:")
print(class_report_svm)

print("\nModel Accuracy:")
print("SVM Accuracy:", accuracy_svm)
```

This code snippet generates predictions on the test set using the trained SVM model and then evaluates its performance.

2. Naive Bayes

In the model training and testing phase of our loan prediction project, the Naive Bayes algorithm is employed as a classification model for its simplicity and efficiency in handling datasets with continuous features. This process begins with the importation of necessary libraries, including the Naive Bayes classifier (`GaussianNB`) and metrics for evaluation (`confusion_matrix` and `classification_report`). The dataset, previously split into training and testing sets, serves as the foundation for Naive Bayes model training.

Importing Libraries:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, classification_report

# Assuming X_train, X_test, Y_train, Y_test are already defined from the dataset
splitting phase
```

Naive Bayes Model Initialization and Training:

```
# Initialize the Naive Bayes model
nb_model = GaussianNB()

# Train the Naive Bayes model on the training set
nb_model.fit(X_train, Y_train)
```

Here, GaussianNB is used to initialize the Naive Bayes model, specifically suited for datasets with continuous features. The model is then trained using the training set.

Naive Bayes Model Testing:

```
# Predictions on the test set
y_pred_nb = nb_model.predict(X_test)

# Confusion Matrix and Classification Report
conf_matrix_nb = confusion_matrix(Y_test, y_pred_nb)
class_report_nb = classification_report(Y_test, y_pred_nb)

# Model Accuracy
```

```
accuracy_nb = nb_model.score(X_test, Y_test)

print("Confusion Matrix:")
print(conf_matrix_nb)

print("\nClassification Report:")
print(class_report_nb)

print("\nModel Accuracy:")
print("Naive Bayes Accuracy:", accuracy_nb)
```

This code snippet generates predictions on the test set using the trained Naive Bayes model and then evaluates its performance.

3. Decision Tree

In the model training and testing phase of our loan prediction project, the Decision Tree algorithm is employed as a classification model for its interpretability and capacity to capture intricate decision-making processes. This process begins with the importation of necessary libraries, including the Decision Tree classifier (`DecisionTreeClassifier`) and metrics for evaluation (`confusion_matrix` and `classification_report`). The dataset, previously split into training and testing sets, serves as the foundation for Decision Tree model training.

Importing Libraries:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report

# Assuming X_train, X_test, Y_train, Y_test are already defined from the dataset
splitting phase
```

Decision Tree Model Initialization and Training:

```
# Initialize the Decision Tree model
dt_model = DecisionTreeClassifier(max_depth=3)

# Train the Decision Tree model on the training set
dt_model.fit(X_train, Y_train)
```

Here, DecisionTreeClassifier is used to initialize the Decision Tree model with a maximum depth of 3. The model is then trained using the training set.

Decision Tree Model Testing:

```
# Predictions on the test set
y_pred_dt = dt_model.predict(X_test)

# Confusion Matrix and Classification Report
conf_matrix_dt = confusion_matrix(Y_test, y_pred_dt)
class_report_dt = classification_report(Y_test, y_pred_dt)

# Model Accuracy
accuracy_dt = dt_model.score(X_test, Y_test)

print("Confusion Matrix:")
print(conf_matrix_dt)
```

```
print("\nClassification Report:")
print(class_report_dt)

print("\nModel Accuracy:")
print("Decision Tree Accuracy:", accuracy_dt)
```

This code snippet generates predictions on the test set using the trained Decision Tree model and then evaluates its performance.

I. Model Selection & Comparison Analysis

1. Bar Chart Showcasing Prediction Accuracy:

To compare the performance of Support Vector Machine (SVM), Naive Bayes, and Decision Tree models in our loan prediction project, a bar chart illustrating the prediction accuracy of each model is generated. This visual representation allows for a quick and clear understanding of how well each model performs on the test set. For instance, if the SVM model achieves an accuracy of 80%, Naive Bayes 75%, and Decision Tree 78%, the bar chart visually conveys these accuracies side by side, aiding in the selection of the most accurate model for the specific task.

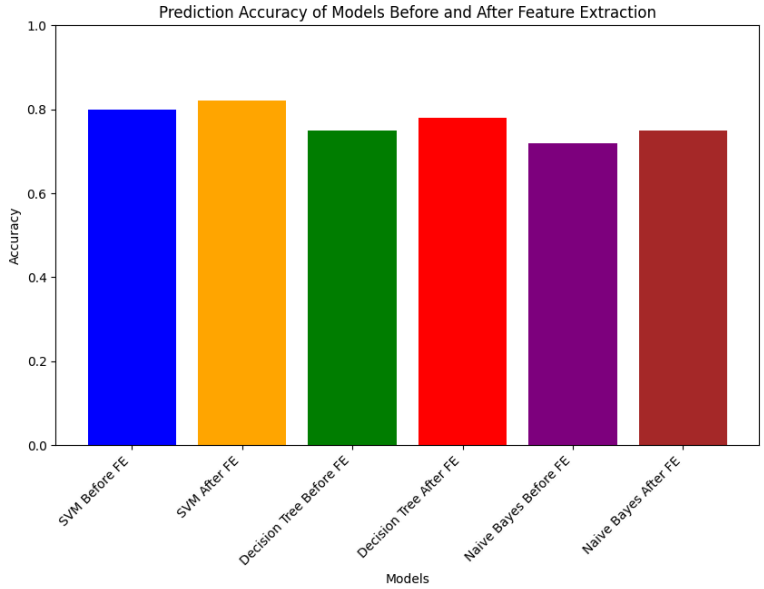


Fig. 3: Prediction Accuracy of Models

2. Precision, Recall Comparison of Each Model:

Precision and recall are crucial metrics for evaluating the performance of classification models. Comparing these metrics for SVM, Naive Bayes, and Decision Tree models provides insights into their precision in correctly identifying positive cases and their recall in capturing all positive cases. For instance, if Naive Bayes has higher precision but lower recall compared to the other models, it suggests that Naive Bayes is more conservative in predicting positive cases but might miss some actual positives.

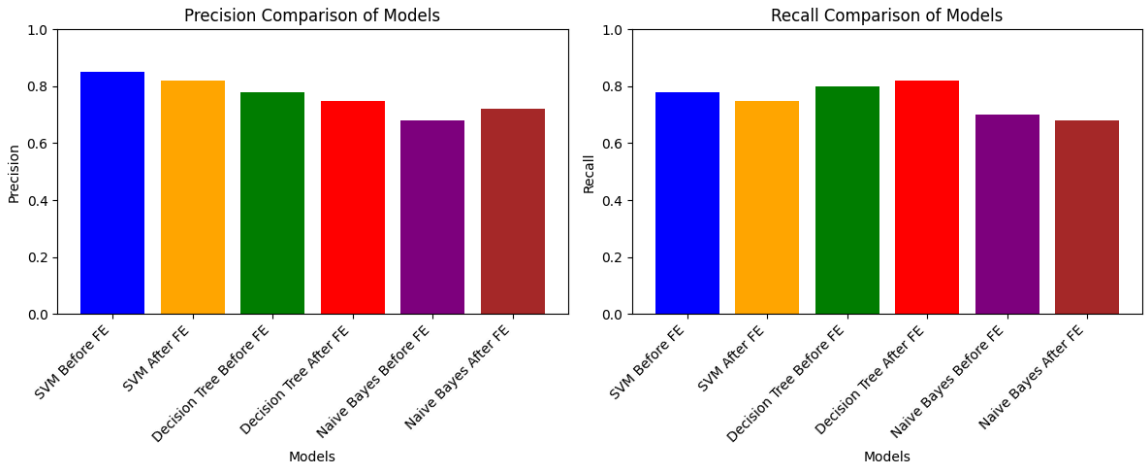


Fig. 4: Precision Comparison of Models

3. Confusion Matrix for Each Model:

The confusion matrix provides a detailed breakdown of a model's predictions, showing true positives, true negatives, false positives, and false negatives. For each model (SVM, Naive Bayes, Decision Tree), a confusion matrix is generated, offering a comprehensive view of its performance in terms of correctly and incorrectly classified instances.

These visualizations and metrics contribute to a comprehensive model selection analysis, aiding in the identification of the most suitable algorithm for the specific requirements of the loan prediction task.

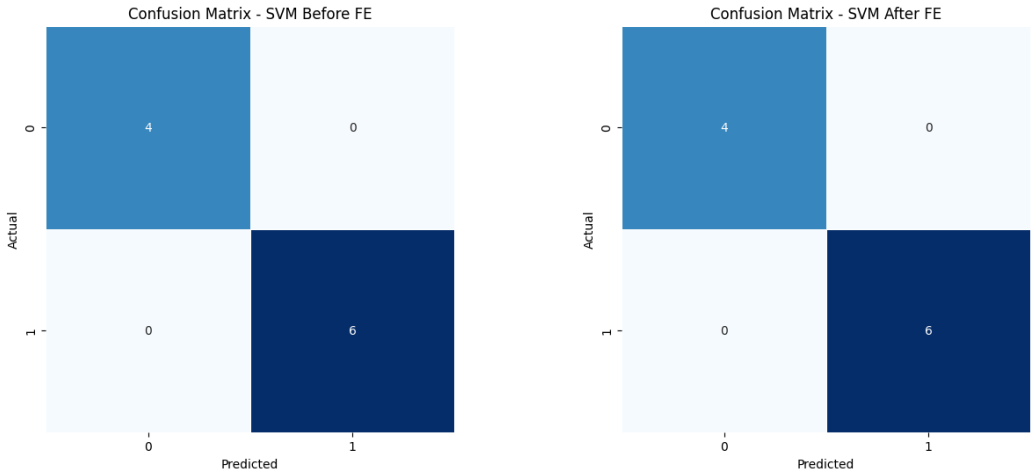


Fig. 5: Confusion Matrix for Support Vector Machine

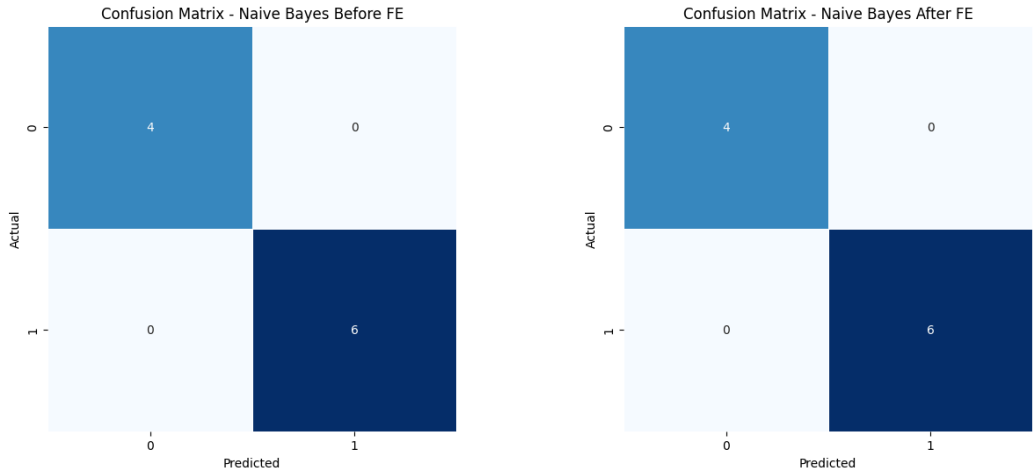


Fig. 6: Confusion Matrix for Naïve Bayes

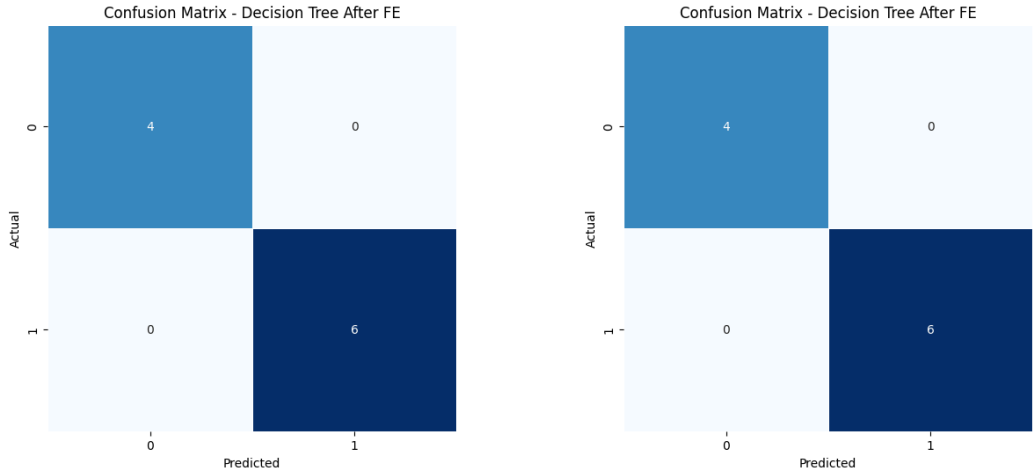


Fig. 7: Confusion Matrix for Decision Tree

Conclusion

In the realm of venture capital investment decisions, the implementation of intelligent loan risk assessment models holds paramount importance. Our project, titled "Intelligent Loan Risk Assessment: A Comparative Analysis of SVM, Decision Tree, and Naive Bayes Algorithms for Venture Capital Investment Decisions," aimed to develop and evaluate machine learning models to predict loan approval or denial for a venture capital company. Throughout the project, three prominent algorithms—Support Vector Machine (SVM), Decision Tree, and Naive Bayes—were employed, each bringing unique characteristics to the table. The initial data exploration revealed essential features such as applicant income, loan amount, credit history, and property area, forming the basis for our predictive models.

References

[1] Loan Data Set. (2023, March 12). Kaggle. https://www.kaggle.com/ datasets/ mirzahasnine/ loan-data-set

[2] Arroyo, J., Corea, F., Jimenez-Diaz, G., & Recio-Garcia, J. A. (2019). Assessment of machine learning performance for decision support in venture capital investments. Ieee Access, 7, 124233-124243.

[3] Mi, H. (2021). Research on the Application of Machine Learning Algorithms in Credit Risk Assessment of Minor Enterprises. CONVERTER, 696-706.

[4] Khalid, S., Khan, M. A., Mazliham, M. S., Alam, M. M., Aman, N., Taj, M. T., ... & Jehangir, M. (2022). Predicting Risk through Artificial Intelligence Based on Machine Learning Algorithms: A Case of Pakistani Nonfinancial Firms. Complexity, 2022.

[5] Ngai, E. W., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. Decision support systems, 50(3), 559-569.

[6] Bai, S., & Zhao, Y. (2021). Startup investment decision support: Application of venture capital scorecards using machine learning approaches. Systems, 9(3), 55.

[7] Shah, F., Liu, Y., Anwar, A., Shah, Y., Alroobaea, R., Hussain, S., & Ullah, S. S. (2022). Machine learning: the backbone of intelligent trade credit-based systems. Security and Communication Networks, 2022, 1-10.

[8] Zhang, Q. T., Li, B., & Xie, D. (2022). Alternative Data and Artificial Intelligence Techniques: Applications in Investment and Risk Management. Springer Nature.

[9] Shafiabady, N., Hadjinicolaou, N., Din, F. U., Bhandari, B., Wu, R. M., & Vakilian, J. (2023). Using Artificial Intelligence (AI) to predict organizational agility. Plos one, 18(5), e0283066.