

DMBI Lab

EXPERIMENT NO. 5

Name : Kaushik Kotian

Roll no. 30

Div : D15B

Batch:B

AIM : To explore Rapid Miner and implement classification models like Decision Tree and Naive Bayes.

1. Preprocess data. Split data into train and test set
2. Build Classification model using Rapid miner on training data (use decision tree and naive Bayes method)
3. Calculate metrics based on test data
4. Compare the models based on metrics and find which model is best suited for the dataset.

Link to the dataset :

<https://www.kaggle.com/datasets/bhavikjikadara/loan-status-prediction?resource=download>

THEORY :

RapidMiner is a comprehensive open-source platform for data science and machine learning, providing a user-friendly interface for building, deploying, and maintaining predictive models. It offers a visual workflow environment where users can design and execute data analysis tasks without needing extensive programming knowledge. RapidMiner supports various data preprocessing techniques, feature engineering, model training using diverse algorithms such as decision trees and neural networks, and model evaluation with metrics like accuracy and F1-score. It also enables model deployment for making predictions on new data or integrating models into production systems. With its intuitive interface and robust functionality, RapidMiner empowers data scientists and analysts to explore data, create powerful machine learning models, and derive valuable insights from complex datasets.

A **classification model** is a type of machine learning model that is used to categorize input data into predefined classes or categories. The goal of a classification model is to

learn the mapping between input features and the corresponding output labels, enabling it to accurately predict the class of new, unseen data points.

Key components of a classification model include:

1. Input Data: The data used to train and test the classification model typically consists of features or attributes that describe the input, along with the corresponding output labels or classes.

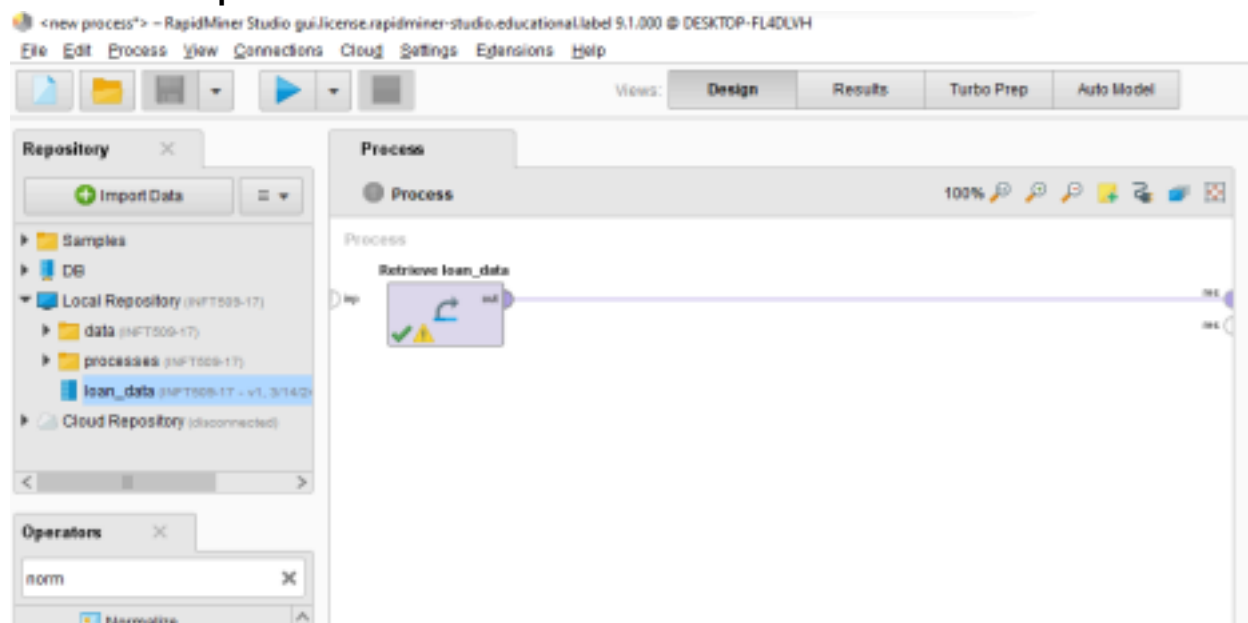
2. Training Phase: During the training phase, the classification model learns from the input data by adjusting its internal parameters based on the provided input-output pairs. Different algorithms, such as decision trees, logistic regression, support vector machines (SVM), k-nearest neighbors (KNN), and neural networks, can be used for training classification models.

3. Prediction: Once the model is trained, it can be used to predict the class labels of new, unseen data points. The model applies the learned mapping to the input features of the new data and outputs the predicted class or probability distribution over classes.

4. Evaluation: The performance of a classification model is evaluated using metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (ROC-AUC). These metrics assess how well the model generalizes to unseen data and its ability to correctly classify instances into their respective classes.

Steps :

1. Data Preparation



The screenshot shows the 'Results' view of RapidMiner Studio. The top toolbar has icons for opening files, saving, and running processes. The 'Views' tab at the top right shows 'Design', 'Results', 'Turbo Prep', and 'Auto Model'. The 'ExampleSet (Retrieve loan_data)' is selected. The 'Filter (381 / 381 examples)' dropdown is set to 'all'. The table below shows the data for the 'ExampleSet (Retrieve loan_data)'.

son	Self_Employ...	ApplicantInc...	CoapplicantL...	LoanAmount	Loan_Amos...	Credit_Histo...	Property_Ar...	Loan_Status
ste	No	1853	2840	114	360	1	Rural	N
ste	No	1299	1086	47	120	1	Urban	Y
ste	No	4950	0	125	360	1	Urban	Y
aduate	No	3596	0	160	240	?	Urban	Y
ste	No	3510	0	76	360	0	Urban	N
aduate	No	4887	0	133	360	1	Rural	N
ste	?	2600	3500	115	?	1	Urban	Y
aduate	No	7660	0	104	360	0	Urban	N
aduate	No	2600	1911	116	360	0	Semiurban	N
aduate	No	3365	1917	112	360	0	Rural	N
ste	No	2799	2253	122	360	1	Semiurban	Y
aduate	No	4226	1040	110	360	1	Urban	Y
aduate	No	1442	0	35	360	1	Urban	N
ste	?	3750	2083	120	360	1	Semiurban	Y

ExampleSet (381 examples, 0 special attributes, 13 regular attributes)

2.

Filter the data to eliminate missing values

Process

Process

100%

Retrieve loan_data

Filter Examples

Create Filters: filters

Defines the list of filters to apply.

Loan_Status

is not missing

is not in

contains

does not contain

starts with

ends with

matches

is missing

is not missing

Add Entry

OK

Cancel

Activate Wisdom of Crowds

Views: Design Results Turbo Prep Auto Model

ExampleSet (Filter Examples)

ExampleSet (\\Local Repository\\loan_data)

Open in Turbo Prep Auto Model

Filter (360 / 360 examples): all

tion	Self_Employ...	Applicantinc...	CoapplicantL...	LoanAmount	Loan_Amou...	Credit_Histo...	Property_Ar...	Loan_Status
ste	No	4583	1508	128	360	1	Rural	N
ste	Yes	3000	0	66	360	1	Urban	Y
aduate	No	2583	2358	120	360	1	Urban	Y
ste	No	6000	0	141	360	1	Urban	Y
aduate	No	2333	1516	95	360	1	Urban	Y
ste	No	3200	700	70	360	1	Urban	Y
ste	No	1853	2840	114	360	1	Rural	N
ste	No	1299	1086	17	120	1	Urban	Y
ste	No	4950	0	125	360	1	Urban	Y
aduate	No	3596	0	100	240	?	Urban	Y
ste	No	3510	0	76	360	0	Urban	N
aduate	No	4887	0	133	360	1	Rural	N
aduate	No	7660	0	104	360	0	Urban	N
aduate	No	2500	1911	116	360	0	Semiurban	N

3.

Check the statistics tab in results, and eliminate missing values from entire dataset

File Edit Process View Connections Cloud Settings Extensions Help

Views: Design Results Turbo Prep Auto Model

Result History ExampleSet (Filter Examples) ExampleSet (/Local Repository/loan_data)

	Name	Type	Missing	St...	Filter (13 / 13 attributes)
✓	Loan_ID	Polynomial	0	Least LP002753 (0)	Most LP001003 (1)
✓	Gender	Polynomial	5	Least Female (78)	Most Male (277)
✓	Married	Polynomial	0	Least No (143)	Most Yes (217)
✓	Dependents	Polynomial	8	Least 3+ (27)	Most 0 (222)
✓	Education	Polynomial	0	Least Not Graduate (98)	Most Graduate (262)
✓	Self_Employed	Polynomial	0	Least Yes (35)	Most No (325)
✓	ApplicantIncome	Integer	0	Min 150	Max 9703

After removing all missing values:

Result History ExampleSet (Filter Examples) ExampleSet (/Local Repository/loan_data)

	Name	Type	Missing	St...	Filter (13 / 13 attributes)
✓	Loan_ID	Polynomial	0	Least LP002943 (0)	
✓	Gender	Polynomial	0	Least Female (63)	
✓	Married	Polynomial	0	Least No (123)	
✓	Dependents	Polynomial	0	Least 3+ (24)	
✓	Education	Polynomial	0	Least Not Graduate (79)	
✓	Self_Employed	Polynomial	0	Least Yes (28)	
✓	ApplicantIncome	Integer	0	Min 150	

4. Building decision tree : Set label attribute

The screenshot shows the Orange3 interface with a workflow consisting of four nodes: 'Retrieve loan_data', 'Filter Examples', 'Set Role', and 'Decision Tree'. The 'Set Role' node is highlighted with an orange border. The 'Parameters' panel on the right is open for the 'Set Role' node, showing the following settings:

- attribute name: Loan_Status
- target role: label
- set additional rol...: Edit List (0)...

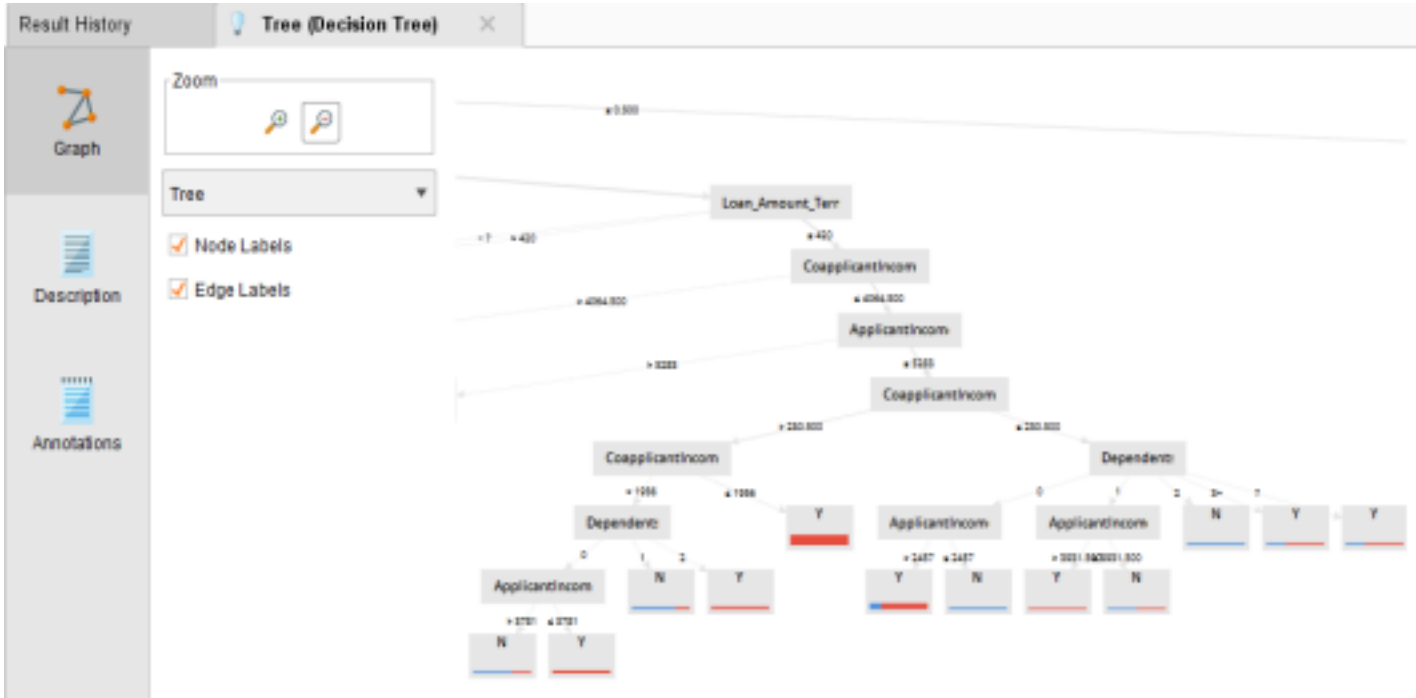
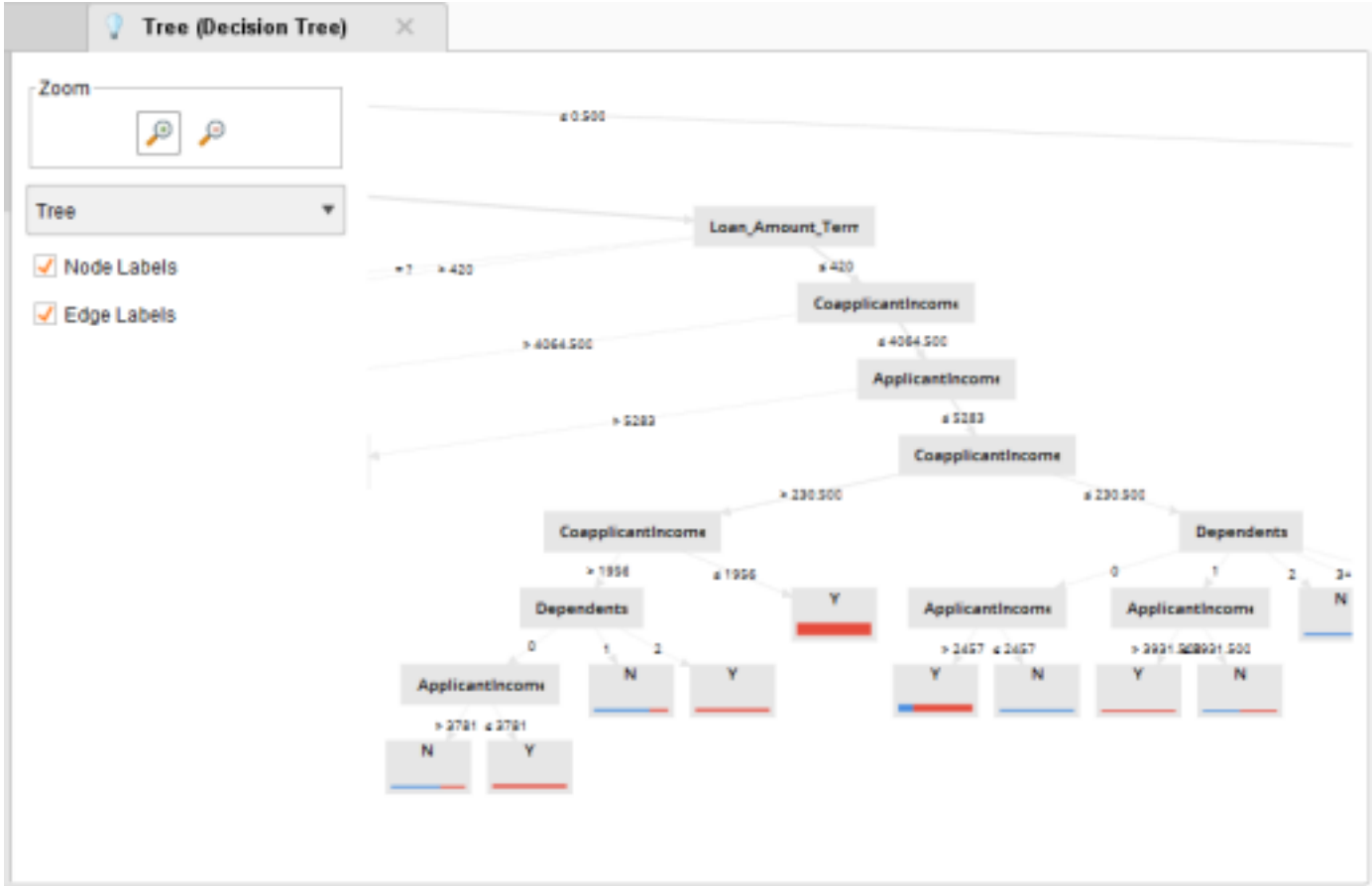
At the bottom of the panel, there are links for 'Show advanced parameters' and 'Change compatibility (9.1.000)'.

Set criterion (information gain/ gini index) :

The screenshot shows the same workflow as the previous image, but now the 'Decision Tree' node is highlighted with an orange border. The 'Parameters' panel on the right is open for the 'Decision Tree' node, showing the following settings:

- criterion: information...
- maximal depth: 10
- apply pruning: ☒
- confidence: 0.1
- apply prepruning: ☒
- minimal gain: 0.01


At the bottom of the panel, there is a link for 'Show advanced parameters'.





Numeric description of the decision tree

Result History

Tree (Decision Tree) X


Graph


Description


Annotations

Tree

```
Credit_History = ?
| ApplicantIncome > 1937
| | Dependents = 0: Y (N=0, Y=17)
| | Dependents = 1: Y (N=1, Y=3)
| | Dependents = 2: Y (N=0, Y=5)
| | Dependents = 3+: N (N=1, Y=1)
| ApplicantIncome ≤ 1937: N (N=2, Y=0)
Credit_History > 0.500
| Property_Area = Rural
| | CoapplicantIncome > 1563.500
| | | LoanAmount > 113.500
| | | | ApplicantIncome > 1905.500
| | | | | LoanAmount > 133
| | | | | | LoanAmount > 137: Y (N=0, Y=5)
| | | | | | LoanAmount ≤ 137: N (N=1, Y=1)
| | | | | LoanAmount ≤ 133: Y (N=0, Y=12)
| | | | ApplicantIncome ≤ 1905.500: N (N=2, Y=0)
| | | LoanAmount ≤ 113.500: Y (N=0, Y=14)
| | CoapplicantIncome ≤ 1563.500
| | | LoanAmount > 135.500: Y (N=0, Y=5)
| | | LoanAmount ≤ 135.500
| | | | LoanAmount > 132.500: N (N=3, Y=0)
| | | | LoanAmount ≤ 132.500
| | | | | ApplicantIncome > 4788.500: Y (N=1, Y=7)
| | | | | ApplicantIncome ≤ 4788.500
| | | | | | CoapplicantIncome > 1248.500: N (N=4, Y=0)
```

Conclusion : Hence we have understood and implemented a classification model using Rapid Miner software and performed data pre processing , data cleaning, and built the decision tree.