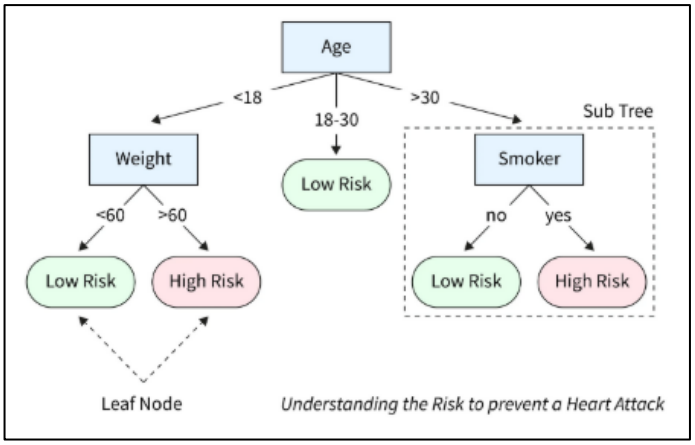
DMBI Experiment 4 – Classification Algorithms using Rapid Miner Tool

Name: Atharva Lotankar
Class: D15C; **Rnum:** 27
Batch B

Aim: To implement Classification Algorithms (Decision Tree & Naïve Bayes' Algorithms) using Rapid Miner Tool.

Theory:

Classification algorithms are a core part of supervised machine learning, where the goal is to predict categorical outcomes based on given input features. They are widely used in applications such as medical diagnosis, spam detection, fraud detection, and customer behaviour prediction. The significance of classification lies in its ability to transform raw data into meaningful knowledge for decision-making.



Among the most popular classification algorithms are the **Decision Tree** and **Naïve Bayes**. A Decision Tree is a tree-structured model that splits data based on attribute values, leading to leaf nodes representing class labels. It is simple to understand, interpretable, and works well with both categorical and numerical data. Naïve Bayes, on the other hand, is a probabilistic algorithm based on Bayes' theorem, assuming independence among features. Despite its "naïve" assumption, it is highly efficient, scalable, and particularly effective for text classification and spam filtering tasks.

The **RapidMiner Tool** is a powerful data science platform that allows users to design, train, and evaluate machine learning models through a simple drag-and-drop interface. It is especially useful for students and professionals who want to focus on concepts without worrying about complex programming. Its intuitive workflow design, rich algorithm library, and visualization features make it ideal for implementing classification techniques like Decision Trees and Naïve Bayes in practical experiments.

Rapid Miner Design Implementation:

1. Load Data: Use the "Read CSV" operator to load your luxury_cosmetics_fraud dataset.

2. Attribute Preparation (if needed): If "Transaction_Time" is numerical, convert it to a polynomial/nominal type using "Numerical to Polynomial" or "Numerical to Nominal" so the Decision Tree can use it for splitting.

3. Set Label and Roles: Use "Set Role" operator to designate "Fraud_Flag" as the label, and all other columns, especially "Transaction_Time," as regular attributes. Exclude or set other columns as "id" as needed, since you're focusing only on "Transaction_Time".

4. Split Data: Use "Split Data" to divide your dataset into training and testing sets (usually an 80/20 or 70/30 split).

5. Train Decision Tree

- Input the training data into the "Decision Tree" operator. (later “Naives Bayes” operator)
- Set the criterion to "gain_ratio" for classification.
- Optionally, adjust parameters like maximal depth, minimal leaf size, and pruning for better model behavior.

6. Apply Model: Use "Apply Model" operator to predict on the holdout (test) set.

7. Evaluate Performance

- Connect "Apply Model" and the true labels to the "Performance" operator.
- Select metrics like accuracy, classification error, recall, and precision.

8. Interpret Results

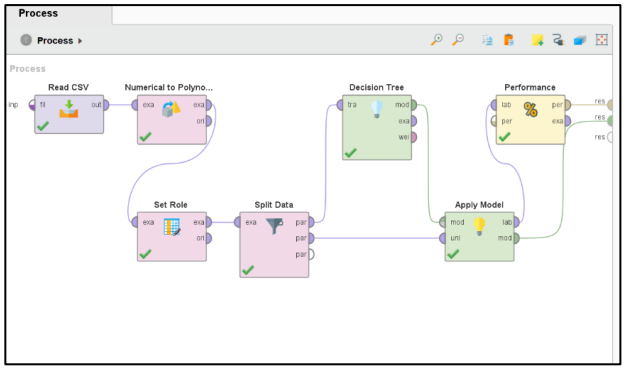
- View the Decision Tree visualization: confirms that "Transaction_Time" is being used for splits and shows the structure of leaf nodes (fraud/not fraud).
- Check the confusion matrix and accuracy metrics for model evaluation (accuracy 96.88% in your run).
- Inspect the 3D and 2D confusion matrix plots for understanding prediction quality.

Inference:

- Both **Decision Tree** and **Naive Bayes** models were implemented using "Transaction_Time" as the primary predictor. The Decision Tree achieved higher accuracy (96.88%) with all predictions as 'not fraud', while Naive Bayes showed lower accuracy (64.38%) but provided some positive class predictions, revealing imbalanced classification performance.
- The workflow demonstrates that "Transaction_Time" is a strong predictor for the majority class, but both models struggle to correctly identify the minority (fraud) class, highlighting the challenge of class imbalance in the dataset.
- This process validates the modeling pipeline and provides a foundation for future enhancements, such as balancing methods or introducing more features to improve fraud detection on imbalanced datasets.

Results:

Decision Tree



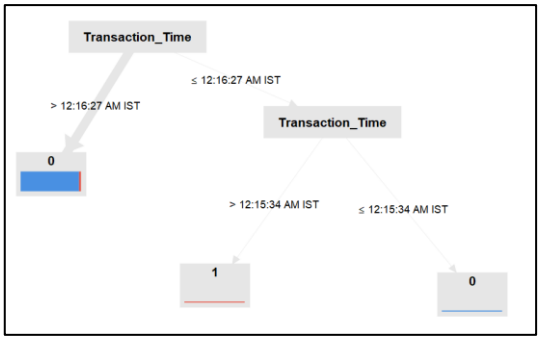
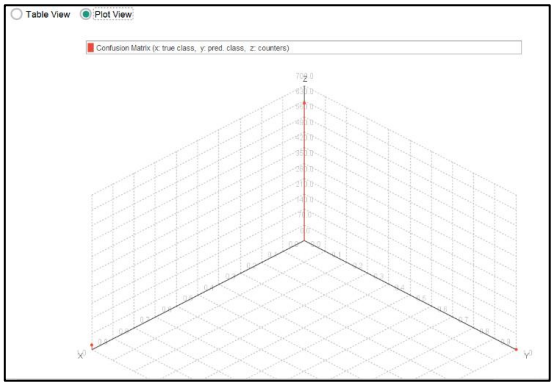


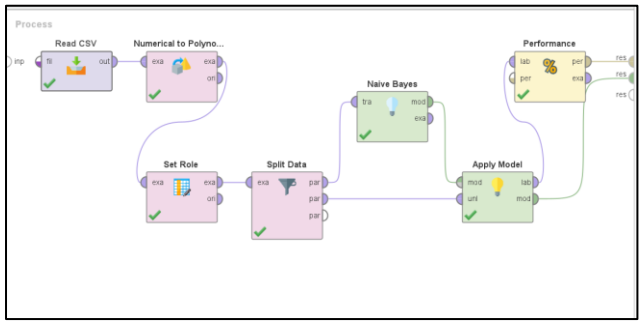
Table View | Plot View

accuracy: 96.88%

	true 0	true 1	class precision
pred. 0	620	20	96.88%
pred. 1	0	0	0.00%
class recall	100.00%	0.00%	



Naïve Bayes' Algorithm



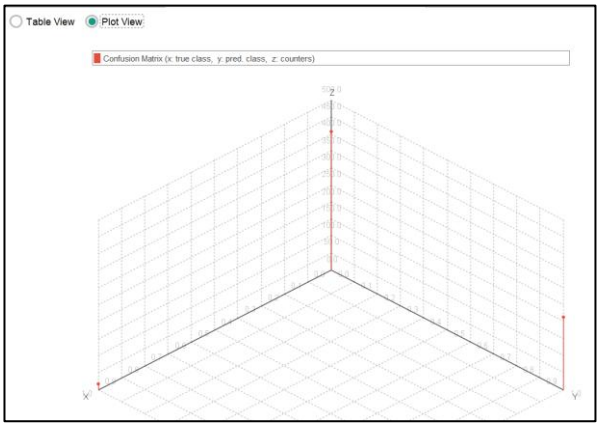


Table View | Plot View

accuracy: 64.38%

	true 0	true 1	class precision
pred. 0	407	15	96.45%
pred. 1	213	5	2.29%
class recall	65.65%	25.00%	

Conclusion: The entire experiment successfully demonstrated the use of Decision Tree and Naive Bayes algorithms for fraud detection using "Transaction_Time." The Decision Tree showed high accuracy but predicted only the majority class, while Naive Bayes captured some fraud cases with lower accuracy. This workflow is robust and can be extended with feature engineering and balancing techniques for better fraud detection.