COMPARISON BETWEEN CLASSIFICATION METHODS BASED ON TITANIC DATASET

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Introduction: Predictive Modelling uses supervised learning data mining methods to extract valuable insights from data. It is used for decision-making which depends on the outcome of the analysis and its accuracy.

Aim and Methodology: This report will conduct data mining using three binary classification algorithms- Random Forest, Decision Trees and Naïve Bayes classification. It will explore various factors that help to improve the model accuracy and will evaluate the results based on performance metrics.

Results: The Random Forest classification proved to be the best model with highest accuracy, while the Naïve Bayes exhibited the lowest accuracy among the three algorithms.

Conclusion: The analysis of correlation, partitioning method, feature selection and the performance metrics provided insights into the various algorithm performances and relevance on the Titanic dataset.

1. INTRODUCTION

Every business-oriented decision-making process is unique, each requiring a suitable method for analysing large datasets involving multiple goals, constraints and possibilities. For extracting meaningful and valuable insights from the available data, data mining algorithms are used, which assist in predicting certain phenomena that eventually lead to a decision in the business strategies. In several business problems, one aims to find correlations between the variables in the problem to predict the value of a specific factor that helps in the further actions to be taken in the business.

According to Provost & Fawcett (2013), classification is a supervised learning method that creates a model from the initial data and, given a new data value, anticipates which class that data belongs to. It utilises a training dataset with inputs and the correct outputs so that the model learns the data patterns. Accordingly, it classifies the new inputs, and a class label is assigned to them (Provost & Fawcett, 2013). Sometimes, it uses labelled datasets to train algorithms to predict what label a new data value will acquire as accurately as possible. It categorises data into groups based on specific features of the data.

The Titanic dataset used in this report is split across two files, one with the passengers' personal information and the other containing data about their tickets, fare and cabins allotted. There are 1,204 entries, i.e., information about 1,204 passengers. The study aims to predict whether a passenger will survive the sinking of the Titanic. This requires the predictive model to perform binary classification.

In the Titanic dataset, the two class labels are "the passengers who will survive" and the "the passengers who will not survive". Hence the target prediction variable will be ‘Survived’ and its two class label values will be 1 and 0.

2. DATA MINING THEORY

Binary Classification, as the name suggests, categorises the data into two distinct classes based on the characteristics of the data. Some common examples of binary classification problems are – 1) Churn Prediction to estimate which consumers are likely to stop using a product or service, based on their behaviour. This has two class labels, 'will stop using the product' and 'will not stop using the product'. Similarly, another example can be 2) Medical Testing to diagnose whether a patient has a disease or not, based on specific symptoms. From the above instances, one can infer that one class represents the normal condition and the other represents the aberrant condition. The algorithm predicts if a data value belongs to class 1 or the abnormal state. (*What is binary classification,* 2022).

Binary Classification is performed on the Titanic dataset using the following supervised data mining methods. True to its name, a Decision Tree is a tree-like structure in which each internal node represents a characteristic, the branches represent rules, and the leaf nodes provide the algorithm's output that represents the class label (*Decision tree*, 2023). The decision tree is the most commonly used algorithm due to its ease and speed of implementation. Decision trees are able to handle both numerical and categorical values, like 'Age' and Embarked' in the current prediction problem. Based on the features in the passenger personal and ticket information provided in the dataset, the decision tree is used to predict the survival of a passenger. According to Donges (2013), Random Forest algorithm builds a combination of the output of multiple decision trees to create a more accurate result. It uses a slightly different subset of the training data set and thus each new model learns the error created by the prior model. While splitting a node, it looks for the best feature among a random subset of features (Donges, 2023). Due to this random selection of features, it also reduces overfitting. Random forests demonstrate significant improvements in the prediction accuracy as compared to decision trees where a single tree is involved; while random forest algorithms use 'n' number of trees sampled randomly from the training dataset. (Kirasich et al., 2018). The Naïve Bayes algorithm is based on the Baysian theory. It assumes that the features are independent of each other and equally contribute to the analysis. It is relatively simple to understand, and is fast to predict classes. A small dataset can easily train the Naïve bayes algorithm (Kaviani & Dhotre, 2017). In order to grasp the association of the input parameters with a class, it is essential for the classifying algorithm to properly familiarize itself with the data using the training dataset so that it determines the category of a new variable as accurately as possible. This makes it imperative for the training dataset to be adequately representative of the issue and contain a sufficient number of samples for each class label. (Brownlee, 2020).

A confusion matrix is considered as the evaluation of an optimal solution for binary classification problems

	Actual Positive Class	Actual Negative Class
Predicted Positive Class	True Positive (tp)	False Negative (fn)
Predicted Negative Class	False Positive (fp)	True Negative (tn)

The rows of the matrix denote the predicted class whereas the columns represent the actual class. It compares the predictions made by the model and the actual class labels.

Here, tp and tn denote the number of positive and negative instances that are correctly classified. Meanwhile, fp and fn denote the number of misclassified positive and negative instances, respectively.

Several metrics like accuracy, precision, sensitivity (recall), specificity and F1 score can be calculated from the confusion matrix. These metrics are an indication of the model's performance in terms of correctly and wrongly classified data points.

Table No.1: Confusion Matrix

3. DATA EXPLORATION AND PREPARATION

The Titanic dataset is divided into two separate files. One is the titanic_ticket_data.csv which consists of all the information of the travellers' tickets like Ticket number, Fare, Cabin number and the port of embarkation. There are three places where the passengers are embarked- Cherbourg, Queenstown and Southampton. This dataset also contains information on whether a particular passenger survived the sinking of the Titanic or not. It shows binary values, 1 and 0, 1 representing they survived and 0 that they did not. The second file is titanic_personal_data.csv which contains all the passengers' personal data. It includes the passenger's name, sex, age, number of siblings or spouses, and number of parents or children. It also contains information about their jobs and salaries. The value to be predicted in the classification problem is whether a passenger will endure the sinking of the Titanic or not. This is a binary classification problem; the two labelled classes are "Will survive" and "Will not survive".

Data pre-processing and cleaning are essential to any analysis which deals with missing values, redundancies, data transformations, data aggregations and feature selection. The Titanic dataset is provided in two separate files; hence, it must be merged into one file to create a unified dataset for comprehensive analysis. Once the datasets are joined, the decimal values in the age variable are rounded off to whole numbers. The 'Survived' column containing binary values 1 and 0 is converted o string so that they are considered as class labels. The statistics of the dataset are calculated to explore the dataset more. They display the minimum and maximum values of each column, their mean, median, standard deviation, skewness, kurtosis, histograms and number of missing values. The Fare and Embarked columns had 1 and 2 missing values respectively, whereas the Age column had 242 missing values. The Cabin column had 934 missing values. The correlation matrix is also plotted for the dataset to find out the correlation among the variables. As per the correlation matrix, the

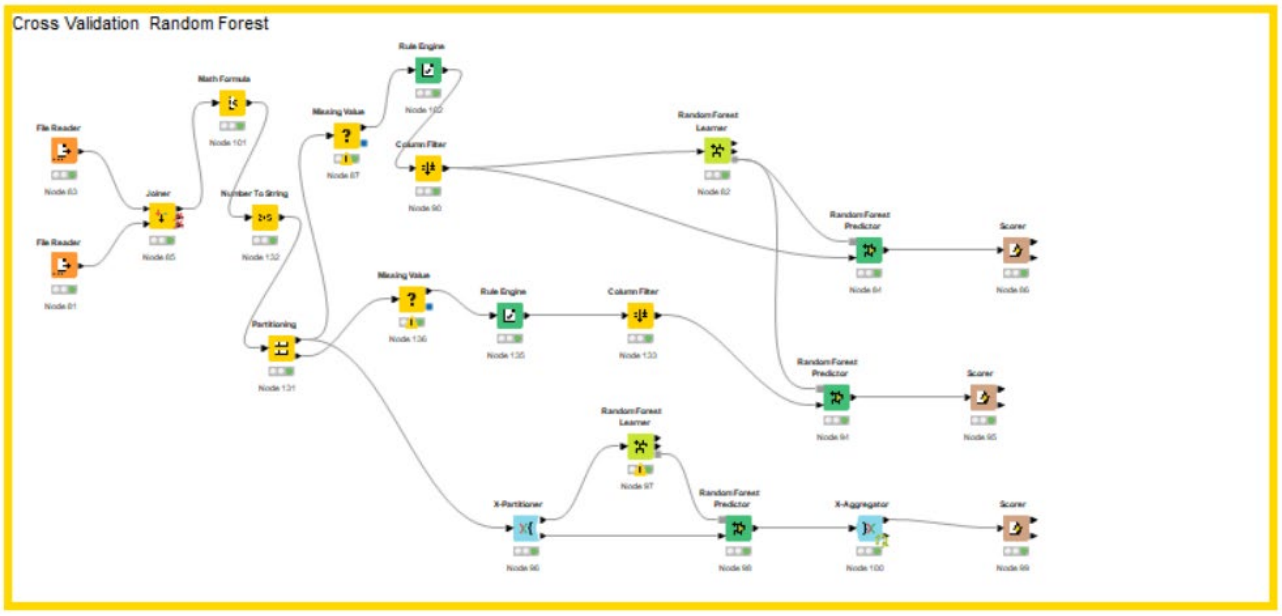
variables ‘Parch’ and ‘Sibsp’ were negatively correlated with Age. Hence, they were filtered out.

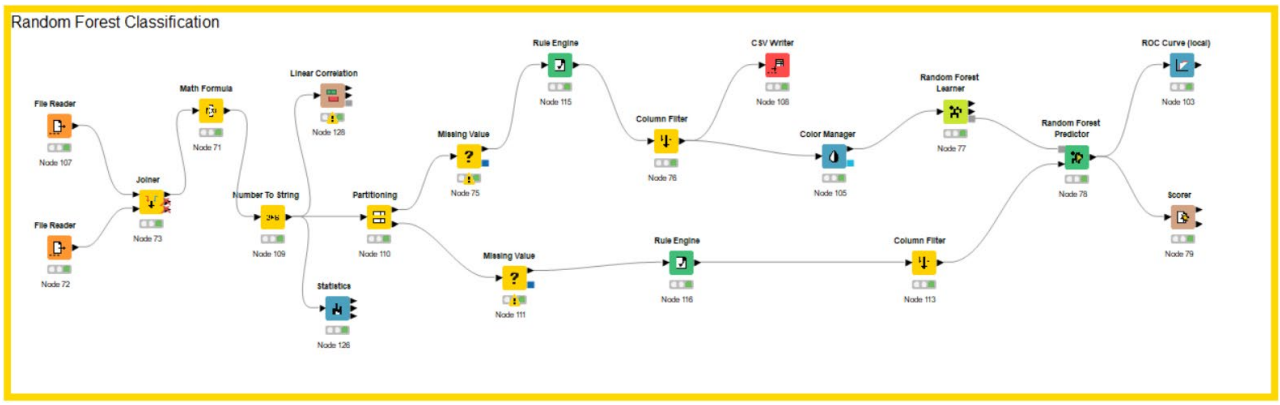
The dataset is then partitioned into training and testing subsets using the 80-20 split ratio. The dataset obtained after partitioning contains missing values in the columns- Age, Embarked, Fare and Cabin values. Age and Fare being numerical values, their Median value is used to replace the missing values in the dataset. The Next Observation Carried Backward (NOCB) method is used for the Embarked column values that are categorical, where each missing value is replaced with the next value. The Cabin column is discarded entirely from the analysis as there were too many missing values, which would not be enough for the model to "learn" from, and if tried to replace, would create a bias in the analysis. This entire pre-processing is performed on the training and testing data separately to avoid data leakage.

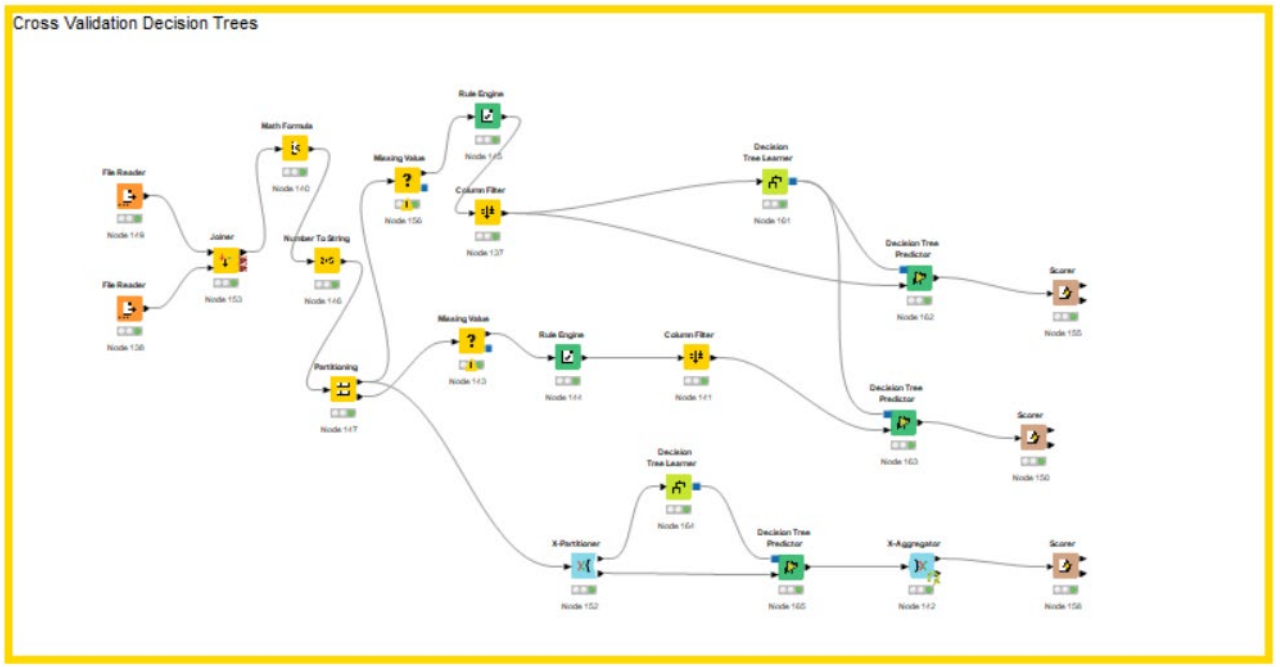
4. EXPERIMENTAL SETUP

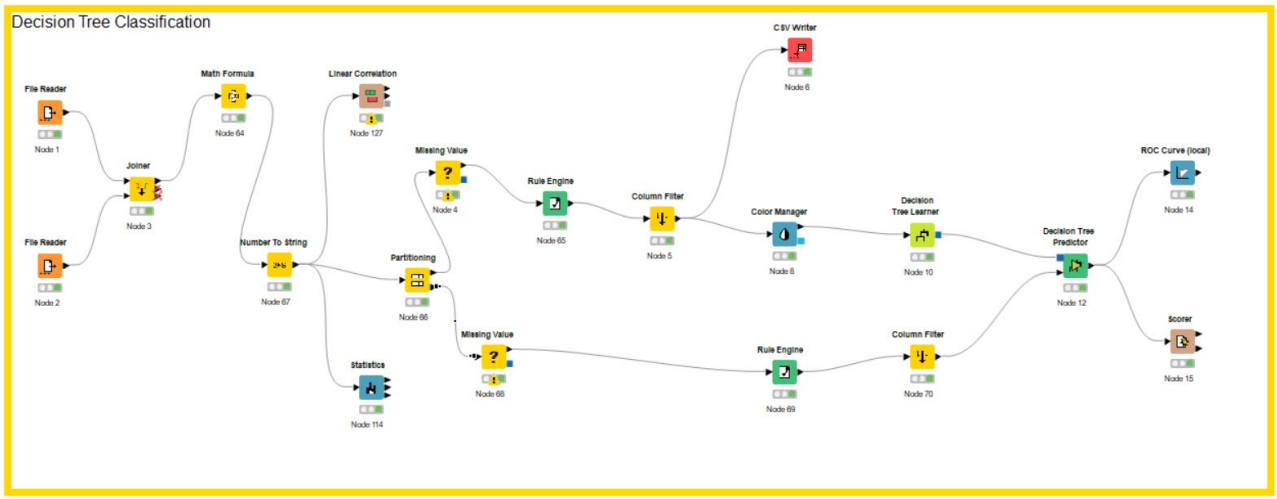
KNIME is an open-source Analytics Platform that provides a visual interface for users to develop and monitor analytic models with multiple complexity levels. It allows users to access, blend, analyse and visualise data using simple functional nodes without any need for complicated coding.

KNIME does not require any type of coding by the users since it already has pre-defined operations fed into its functional nodes, making it easier for the user to configure according to the problem requirement.




Random Forest Classification
File Reader
Node 107
File Reader
Node 72
Joiner
Node 73
Math Formula
Node 71
Number To String
Node 109
Linear Correlation
Node 128
Statistics
Node 126
Partitioning
Node 110
Missing Value
Node 75
Rule Engine
Node 115
Missing Value
Node 111
Rule Engine
Node 116
Column Filter
Node 76
CSV Writer
Node 108
Color Manager
Node 105
Column Filter
Node 113
Random Forest Learner
Node 77
Random Forest Predictor
Node 78
ROC Curve (local)
Node 103
Scorer
Node 79


Cross Validation Decision Trees
File Reader
Node 149
File Reader
Node 138
Joiner
Node 153
Math Formula
Node 140
Number To String
Node 146
Partitioning
Node 147
Missing Value
Node 156
Rule Engine
Node 145
Column Filter
Node 137
Missing Value
Node 143
Rule Engine
Node 144
Column Filter
Node 141
Decision Tree Learner
Node 161
Decision Tree Predictor
Node 162
Scorer
Node 155
Decision Tree Predictor
Node 163
Scorer
Node 150
Decision Tree Learner
Node 164
X-Partitioner
Node 152
Decision Tree Predictor
Node 165
X-Aggregator
Node 142
Scorer
Node 158


Decision Tree Classification
File Reader
Node 1
File Reader
Node 2
Joiner
Node 3
Math Formula
Node 64
Number To String
Node 67
Linear Correlation
Node 127
Statistics
Node 114
Partitioning
Node 66
Missing Value
Node 4
Rule Engine
Node 65
Missing Value
Node 68
Rule Engine
Node 69
Column Filter
Node 5
CSV Writer
Node 6
Color Manager
Node 8
Column Filter
Node 70
Decision Tree Learner
Node 10
Decision Tree Predictor
Node 12
ROC Curve (local)
Node 14
Scorer
Node 15

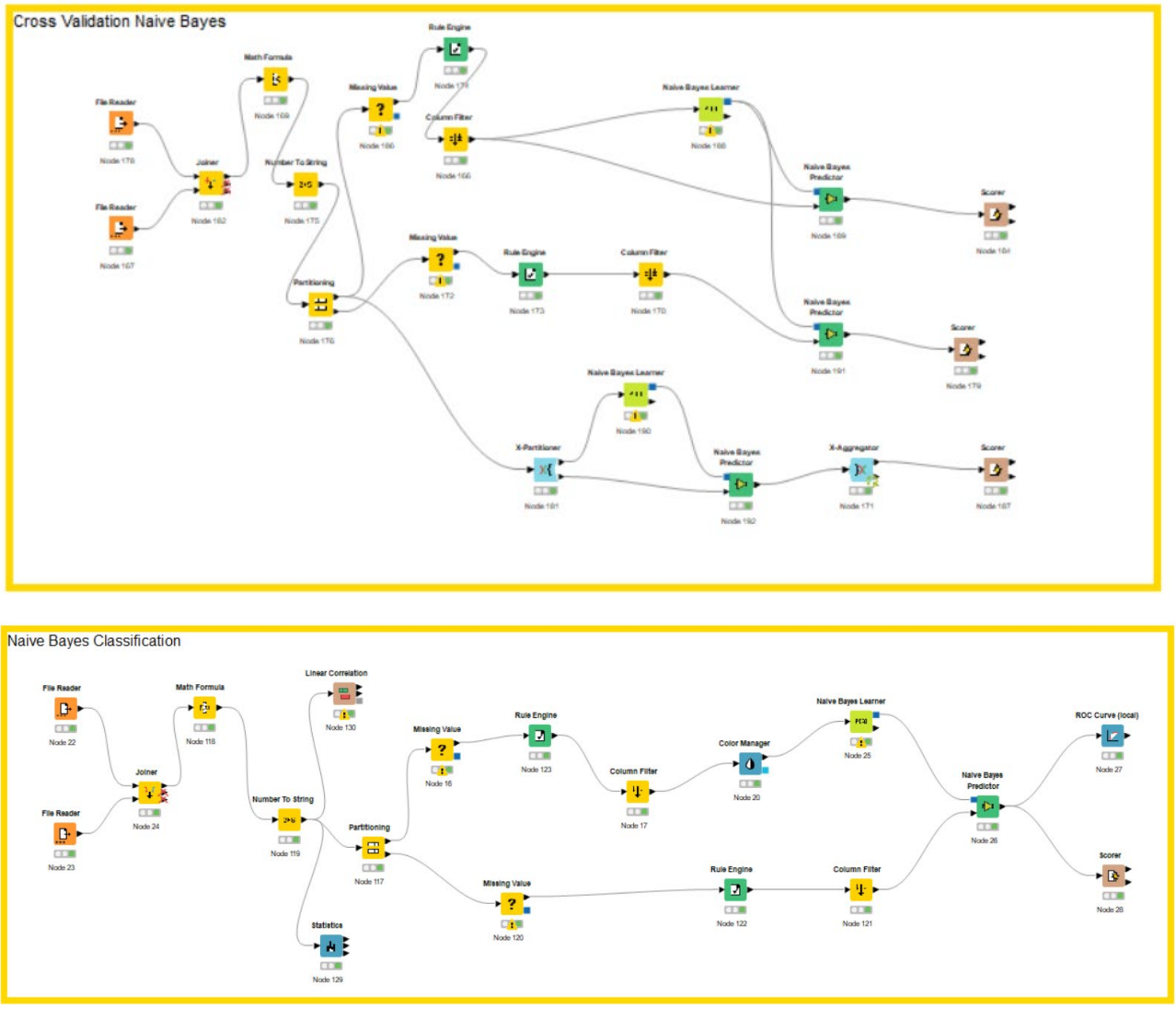


Fig.No.1: KNIME Workflow

Initially a cross validation was performed on the training dataset for all three algorithms to evaluate their performance. The nodes used for loading the dataset, data pre-processing and the actual analysis are listed below: -

File Reader: The File Reader node in KNIME is used to upload the dataset in the KNIME environment. Since the dataset is divided into two separate files, two file readers are used to read both the Titanic dataset files. These two files are then fed to the joiner node.
Joiner: The Joiner node combines two different dataset files; hence it is used to perform an inner join on the two data files – the Titanic passenger personal information and their ticket information, using Passenger ID as the Join column. The output of joiner node is fed into the math formula node.

Math Formula: The Math formula node is used to convert decimal values of the variable age into whole numbers. The decimal values are rounded off to the closest whole number. The round off function formula is configured in the node. The output of this node is fed into the Number to String node.

Number to String: The Number to String node is used whenever numerical values have to be treated as categorical. The "Survived" column is the target column having the two class labels "Will survive" and "Will not survive" that have numerical binary values 1 and 0. Hence, these values are converted to string format to classify the data into labels. This output is passed on to the partitioning node.

Statistics: The Statistics node is connected to the Number to String node to view the statistical summary of the dataset.

Linear Correlation: The linear correlation node is used to display the correlation matrix for the dataset.

Partitioning: The output from the math formula node is connected to the partitioning node. Partitioning is one of the most critical steps in any classification algorithm. The partitioning node is used for dividing the dataset into two subsets- training and testing. The node is configured to specify the split ratio of the dataset. In this workflow, the data is split as 80% for training and 20% for testing. After partition, the training dataset contains 963 rows and testing set contains 241 rows. The 70-30 split ratio is also used to test if it gives different results. Stratified sampling is selected in the partitioning node to ensure that both the class labels of the target variable are distributed equally in the training and testing dataset. Further, both training and testing outputs are fed into different missing value nodes to pre-process them separately.

Missing Value: The Missing Value node assists in handling the missing values in the input data. It allows the user to select the available columns with missing values and choose a method to handle them. In the selected dataset, there are missing values in the columns Age, Embarked, Fare and Cabin. The missing values in Age and Fare are replaced by their median values. The values in Embarked column are replaced by the next value in the column. Further, this outcome is passed on to the Rule Engine node for both the training and testing subsets.

Rule Engine: The rule engine node is used for converting the age variable into categories. The age categories are as follows –

1) Age less than and equal to 20
2) Age greater than 20 but less than and equal to 40
3) Age greater than 40 but less than and equal to 60
4) Age greater than 60

The output of this node contains the age groups and not individual age values. This is fed into the column filter node.

Column Filter: As the name suggests, this node filters out the selected column from the dataset. The Cabin column is chosen to be filtered out of the dataset since it had too many missing values to be considered. This produced the final clean dataset without missing values or redundancies.

CSV Reader: This node is used in the workflow to view the clean dataset as a CSV file.
Color Manager: The Color Manager node is used to assign colours to the binary categories of the target variable.

Table No. 2: Node Descriptions

This completes the data exploration and pre-processing task and the data is ready for analysis and modelling. The above is the initial setup for the Binary Classification task and is the same for all the classification algorithms mentioned below. The output of the Color Manager is input to the Learner nodes of the respective algorithms for the training dataset. For the output of the testing dataset, the output of the column filter is directly input to the Predictor node of that algorithm.

Decision Tree	Random Forest	Naïve Bayes
Decision tree Learner: The decision tree learner node takes the training dataset as the input from the partitioning node. The target column is set to "Survived". The Gini index is selected as the quality measure, and the number of threads is set to 8.	Random Forest Learner: Similar to the decision tree learner, the target column is selected as "survived" in the configuration. It takes the input from the training dataset.	Naïve Bayes Learner: In the Naïve Bayes Learner node, the target column is set as 'Survived' and it takes the training dataset as the input.
Decision tree Predictor: There are two inputs in the decision tree predictor node. One is for the decision tree learner output, i.e., the algorithm, and the other one is for the testing dataset. The outcome of this node is then fed into the scorer node to evaluate its performance.	Random Forest Predictor: This node is similar to the decision tree node and takes the two inputs, the learner algorithm and the testing dataset. It is then connected to the scorer node for evaluation.	Naïve Bayes Predictor: The predictor node takes the learner node and testing dataset as the input and is further connected to the scorer node for evaluation.

Table No.3: Configurations of Classification Algorithms

Scorer: The output of the predictor node is fed into the scorer node. The Scorer node is used to display the performance metrics of the model. It displays all the metrics that indicate how well the model predicts the target variable. It is essentially used to estimate the accuracy of the model's predictive power.

5. RESULTS AND DISCUSSION

The classification algorithms are executed in the data mining tool KNIME. The models are evaluated based on their accuracy and other performance metrics. The accuracy values for the classification algorithms performed are as follows:

Classification Algorithm	Accuracy
Random Forest	89.627%
Decision Tree	85.062%
Naïve Bayes	81.065%

Table No.4: Accuracy Comparisons

Out of all the classification algorithms, the highest accuracy (89.627%) was achieved by the Random Forest algorithm. It was able to correctly classify survival outcomes of 216 rows out of the 241 rows in the testing dataset. It was followed by the decision tree which achieved an accuracy of 85.062%. Even though it was slightly lower than the random forest algorithm accuracy, it still performed quite well in predicting the survival classes. A major difference between Decision trees and Random Forest, or rather what makes the Random Forest algorithm produce more accurate results is how it focuses on choosing data samples at random and aggregates multiple decision trees to produce the output. Where the characteristics and labels are chosen in a decision tree just once, the random forest algorithm chooses them at random and constructs numerous decision trees to eventually average the results. The Naïve Bayes algorithm produced the lowest accuracy (81.065%) among all the three classification algorithms. Naïve Bayes algorithm assumes that the individual features or variables are independent of each other. It can be said that it performed comparatively poorly than the other two algorithms due to this assumption not fulfilled by the Titanic dataset.

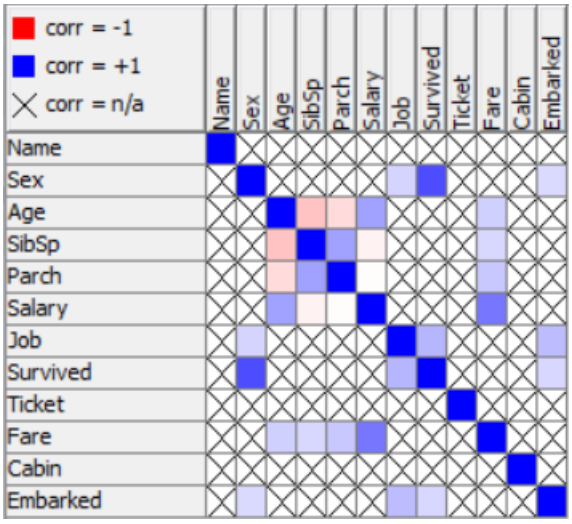


Fig.no.2: Correlation Matrix

The correlation matrix derived from the dataset indicated that ‘sex’ and ‘survived’ columns are highly correlated. It implies that the survival outcome greatly depends on the sex of the passenger. Furthermore, the ‘fare’ and ‘salary’ variables also turned out to be somewhat correlated to each other. It is natural that a person with higher salary

is able to afford higher fare. The ‘age’ and ‘salary’ also seem to be correlated, which can be explained by the fact that usually with age the salary of a person increases.

Furthermore, the ‘Sibsp’ and ‘Parch’ variables are observed to be negatively correlated with age. It suggests that passengers with more siblings or spouses and parents or children tend to be younger. These two variables seem to be less relevant to the analysis compared to other variables. Hence, they are excluded from the analysis. It can be said that eliminating them might simplify the model. This is considered as feature selection which implies choosing only the most relevant features for analysis. Selecting the most relevant features helps to identify exactly which factors affect the target variable and what makes it distinct from the other classes.

The partitioning between the training and testing datasets was done using two split ratios- 70-30 split and 80-20 split. The training set is used to train the model where it captures the patterns in the data and learns which class a data value belongs to. It is thus necessary that there are sufficient examples in the training data for the model to learn from. It was also observed that larger the training dataset, better the model performance. Almost all the models performed well when the split ratio between the training and testing dataset was 80-20 rather than when it was 70-30. Hence, in the final analysis, the 80-20 split ratio was chosen. This is because more observations and possibilities were fed into the model when the training dataset was large, eventually helping the model to learn better.

Further, the age variable that was numerical was converted to categorical by dividing it into groups. Although decision trees and random forest can handle numerical variables well, this approach allows the algorithms to capture potential non-linear relationships between the age groups and the survival variable. Decision trees can hence make splits based on the age groups rather than the actual values. Similarly, each decision tree in random forest algorithm can create binary splits based on the categories. As for naïve bayes, it can calculate the conditional probabilities of each age group separately by considering them as a separate feature.

The confusion matrix for the three algorithms is as follows-

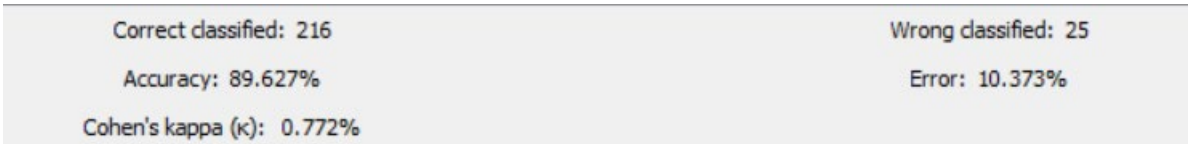


Fig.no.3: Accuracy for Random Forest

Survived\ Prediction (survived)	**0**	**1**
0	145	6
1	19	71
The random forest algorithm has classified 71 positive instances and 145 negative instances correctly. It has classified 19 positive instances and 6 negative ones incorrectly. The Cohen’s kappa coefficient is 0.772% indicating substantial agreement between the actual classes and the predictions.		

Table No.5: Confusion Matrix for Random Forest

Fig.no.4: Accuracy for Decision Trees

Survived\ Prediction (survived)	**0**	**1**
0	137	14
1	22	68
The decision tree algorithm has classified 68 positive instances and 137 negative instances correctly. It has classified 22 positive instances and 14 negative ones incorrectly. The Cohen's kappa coefficient is 0.675% which also denotes substantial agreement between the actual classes and the predictions.		

Table No.6: Confusion Matrix for Decision Trees

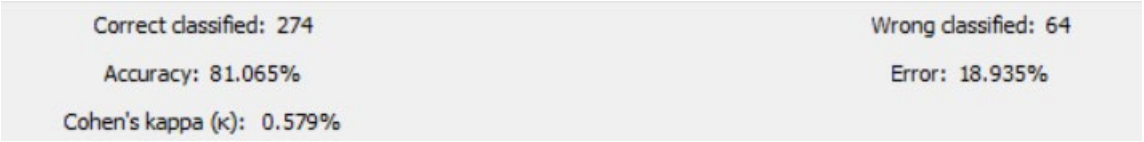


Fig.no.5: Accuracy for Naïve Bayes

Survived\ Prediction (survived)	**0**	**1**
0	192	19
1	45	82
The Naïve Bayes algorithm has classified 82 positive instances and 192 negative instances correctly. It has classified 45 positive instances and 19 negative ones incorrectly. The Cohen's kappa coefficient is 0.579% which represents moderate agreement between the actual classes and the predictions.		

Table No.7: Confusion Matrix for Naïve Bayes

Overall, higher values of Cohen's kappa coefficient indicate a stronger level of agreement between the actual class labels and the predicted values of class labels. Thus, it is observed that the random forest algorithm exhibits the strongest agreement, which implies that the algorithm is reliable to accurately classify the data into correct labels.

Based on the confusion matrix values, the algorithms calculated the performance metrics as follows-

Algorithm	Recall/ Sensitivity		Precision		Specificity		F-measure	
	0	1	0	1	0	1	0	1
Random Forest	0.96	0.789	0.884	0.922	0.789	0.96	0.921	0.85
Decision Trees	0.907	0.756	0.862	0.829	0.756	0.907	0.884	0.791
Naïve Bayes	0.91	0.646	0.81	0.812	0.646	0.91	0.857	0.719
The recall/sensitivity values indicate the true positive rate, and it can be observed that the values for all the algorithms in the negative class are almost close to 1, indicating that it has a lower rate of false negative predictions. Precision represents the proportion of correctly predicted positive instances, and all the algorithms exhibit a higher value denoting lower rate of false positive predictions. The F1 score is the mean of precision and recall, and it displays a balanced measure of the model performance. It is observed that random forest exhibits a good balance.								

Table No.8: Accuracy Statistics Comparison

6. CONCLUSION AND REFLECTIONS

In overview, the random forest algorithm performed the best, exhibiting highest accuracy among the three algorithms. In addition, random forest reduces overfitting by combining multiple decision trees to average the results. It handles numerical and categorical features effectively. Decision tree captures non-linear relationships between the features well. It performed well in terms of accuracy, precision and F1 score. The naïve bayes algorithm performed comparatively less and exhibited lowest accuracy. It assumes that all features are independent of each other, which is not true for some features in the titanic dataset.

The correlation analysis identified some relationships among variables. The sex of the passenger was highly correlated with the target variable meaning it was the feature that contributed the most in the prediction in all three models. Another variable ‘salary’ was moderately correlated to the ‘fare’ and ‘age’ variables, indicating that higher the value of one, higher is the ‘salary’ value. In contrast to the ‘sex’ variable, the ‘Parch’ and ‘Sibsp’ variables contributed the least in the analysis as they were negatively correlated to ‘age’ variable.

Finally, the partitions that were executed indicated that larger a training dataset produces higher accuracy. The random forest exhibited substantial agreement with the actual class labels, while the other two represented a moderate one.

In conclusion, the Random Forest classification performed the best in terms of accuracy. The Decision Tree and Naïve Bayes models also performed well, but exhibited a slightly lower accuracy. The analysis of correlation, partitioning method, feature selection and the performance metrics provided insights into the various algorithm performances and relevance on the Titanic dataset.

REFERENCES

Brownlee, J. (2020, August 19). *4 types of classification tasks in machine learning*. MachineLearningMastery.com. https://machinelearningmastery.com/types-of-classification-in-machine-learning/

Donges, N. (2023). *Random Forest: A complete guide for machine learning*. Built In. https://builtin.com/data-science/random-forest-algorithm#real

GeeksforGeeks. (2023, May 8). *Decision tree*. GeeksforGeeks. https://www.geeksforgeeks.org/decision-tree/

Kaviani, P., & Dhotre, S. (2017). Short Survey on Naive Bayes Algorithm. *International Journal of Advance Research in Computer Science and Management*. https://www.researchgate.net/publication/323946641

Kirasich, K., Smith, T., & Sadler, B. (2018). Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets . *SMU Dta Science Review*, *1*. https://scholar.smu.edu/datasciencereview/vol1/iss3/9/

Provost, F., & Fawcett, T. (2013). Business Problems and Data Science Solutions. In *Data Science for Business: What you need to know about data mining and data-analytic thinking* (pp. 20–31). essay, O'Reilly.

What is binary classification. Deepchecks. (2022, December 22). https://deepchecks.com/glossary/binary-classification/