Assignment Task 3

Data analytics in action - report

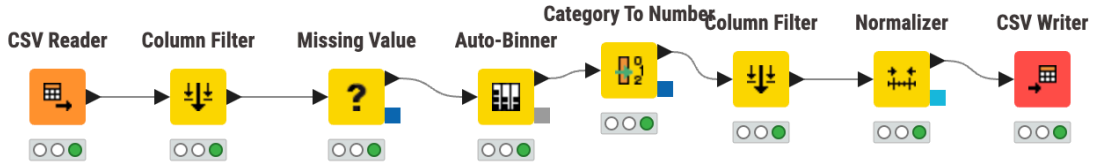
Student name: Man Sing Lee

Student number: 24708772

Date submitted:6/11/23

Table of Contents

Data Preprocessing



Feature Selection

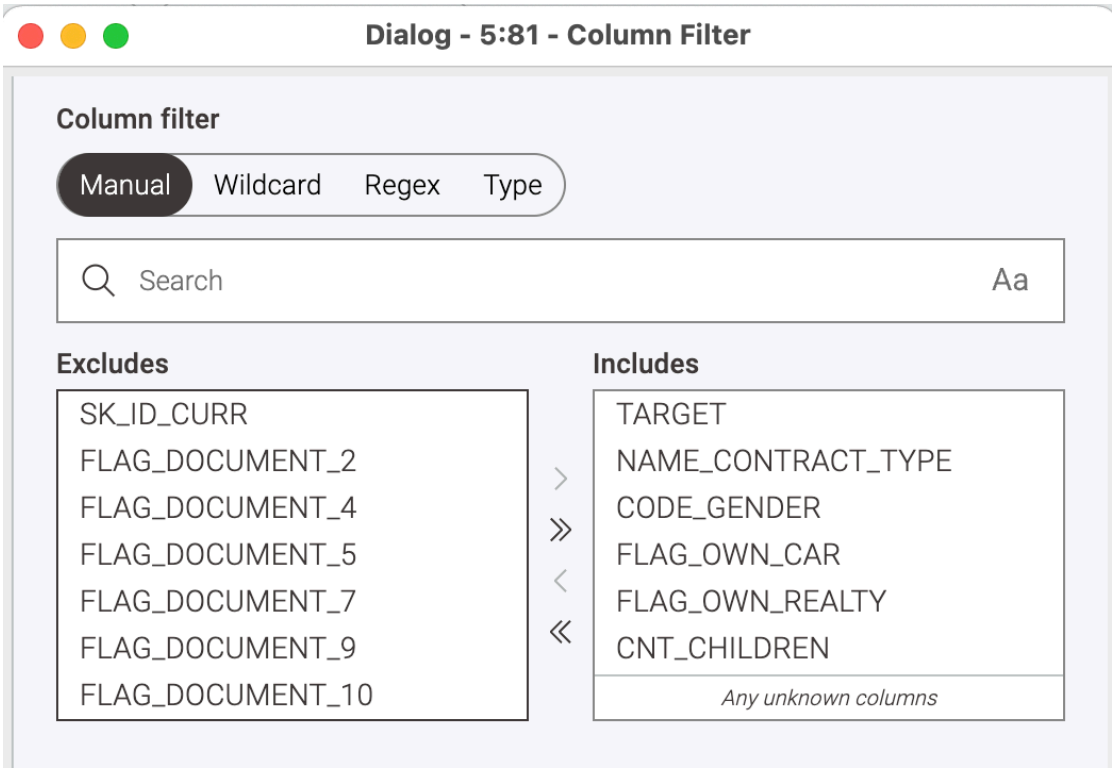


Figure 1. Data Preprocessing

In building a classifier for the "TARGET" attribute, one of the critical steps is feature selection and data cleaning. This step involves deciding which features to keep, modify, or remove to prepare the dataset for model training. These decisions are driven by the relevance of each feature to the classification task and the need to ensure data quality and integrity. Here are some general guidelines for feature selection and data cleaning:

Feature Type	Feature Name	Data Modification/Encoding	Reason
Target Variable	TARGET	-	Indicator of loan repayment difficulties, retained for classification.
Identifiers	SK_ID_CURR (Customer ID)	-	Unique identifiers, not used in modeling.
Categorical Features	CODE_GENDER	Binary Encoding (Female=0, Male=1)	Encoding for binary gender classification.
	FLAG_OWN_CAR	Binary Encoding (No=0, Yes=1)	Encoding for car ownership binary choice.
	FLAG_OWN_REALTY	Binary Encoding (No=0, Yes=1)	Encoding for realty ownership binary choice.
	NAME_TYPE_SUITE	One-Hot Encoding / Label Encoding	Encoding for types of individuals accompanying clients.
	NAME_INCOME_TYPE	One-Hot Encoding / Label Encoding	Encoding for various income types.
	NAME_EDUCATION_TYPE	One-Hot Encoding / Label Encoding	Encoding for different education levels.
	NAME_FAMILY_STATUS	One-Hot Encoding / Label Encoding	Encoding for family status types.
	NAME_HOUSING_TYPE	One-Hot Encoding / Label Encoding	Encoding for housing situation types.
	WEEKDAY_APPR_PROCESS_START	One-Hot Encoding /	Encoding for the day of the week of loan

		Label Encoding	application.
Numerical Features	AMT_INCOME_TOTAL	-	Retained for assessing client's financial situation.
	AMT_CREDIT	-	Retained for relevance to loan amount.
	AMT_ANNUITY	-	Retained for relevance to loan installment amount.
	AMT_GOODS_PRICE	-	Retained for relevance to goods price.
	REGION_POPULATION_RELATIVE	-	Retained for information about the client's living region.
	DAYS_BIRTH	-	Retained for relevance to client's age.
	DAYS_EMPLOYED	-	Retained for relevance to client's employment history.
	DAYS_REGISTRATION	-	Retained for relevance to registration history.
	DAYS_ID_PUBLISH	-	Retained for relevance to document changes.
	CNT_CHILDREN	-	Retained for relevance to the number of children.
	CNT_FAM_MEMBERS	-	Retained for relevance to family size.
	REGION_RATING_CLIENT	-	Retained as ratings of the client's region.
	REGION_RATING_CLIENT_W_CITY	-	Retained as ratings of the client's region with city consideration.
	HOUR_APPR_PROCESS_START	-	Retained for relevance to the time of the loan application.
	DAYS_LAST_PHONE_CHANGE	-	Retained for relevance to phone changes.
	EXT_SOURCE_2	-	Retained as a normalized score from external data source.

	EXT_SOURCE_3	-	Retained as a normalized score from external data source.
	OBS_30_CNT_SOCIAL_CIRCLE	-	Retained for social observation-related data.
	DEF_30_CNT_SOCIAL_CIRCLE	-	Retained for social observation-related data.
	OBS_60_CNT_SOCIAL_CIRCLE	-	Retained for social observation-related data.
	DEF_60_CNT_SOCIAL_CIRCLE	-	Retained for social observation-related data.
	AMT_REQ_CREDIT_BUREAU_HOUR	-	Retained for information about credit bureau inquiries.
	AMT_REQ_CREDIT_BUREAU_DAY	-	Retained for information about credit bureau inquiries.
	AMT_REQ_CREDIT_BUREAU_WEEK	-	Retained for information about credit bureau inquiries.
	AMT_REQ_CREDIT_BUREAU_MON	-	Retained for information about credit bureau inquiries.
	AMT_REQ_CREDIT_BUREAU_QRT	-	Retained for information about credit bureau inquiries.
	AMT_REQ_CREDIT_BUREAU_YEAR	-	Retained for information about credit bureau inquiries.
Binary Features	FLAG_MOBIL	-	Retained for information about mobile phone provision.
	FLAG_EMP_PHONE	-	Retained for information about employer phone provision.
	FLAG_WORK_PHONE	-	Retained for information about work phone provision.

	FLAG_CONT_MOBILE	-	Retained for mobile phone reachability.
	FLAG_PHONE	-	Retained for information about home phone provision.
	FLAG_EMAIL	-	Retained for information about email provision.
Flag Features	REG_REGION_NOT_LIVE_REGION	-	Retained for address relevance (region level).
	REG_REGION_NOT_WORK_REGION	-	Retained for address relevance (region level).
	LIVE_REGION_NOT_WORK_REGION	-	Retained for address relevance (region level).
	REG_CITY_NOT_LIVE_CITY	-	Retained for address relevance (city level).
	REG_CITY_NOT_WORK_CITY	-	Retained for address relevance (city level).
	LIVE_CITY_NOT_WORK_CITY	-	Retained for address relevance (city level).
Document Features	FLAG_DOCUMENT_3	-	Retained due to substantial and relevant data.
	FLAG_DOCUMENT_6	-	Retained due to substantial and relevant data.
	FLAG_DOCUMENT_8	-	Retained due to substantial and relevant data.

Table 1. Feature Selection

The following features were dropped from the original dataset, along with the identifiers (SK_ID_CURR), during data preprocessing:

1. FLAG_DOCUMENT_2
2. FLAG_DOCUMENT_4
3. FLAG_DOCUMENT_5
4. FLAG_DOCUMENT_7
5. FLAG_DOCUMENT_9
6. FLAG_DOCUMENT_10
7. FLAG_DOCUMENT_11
8. FLAG_DOCUMENT_12
9. FLAG_DOCUMENT_13

10. FLAG_DOCUMENT_14
11. FLAG_DOCUMENT_15
12. FLAG_DOCUMENT_16
13. FLAG_DOCUMENT_17
14. FLAG_DOCUMENT_18
15. FLAG_DOCUMENT_19
16. FLAG_DOCUMENT_20
17. FLAG_DOCUMENT_21

The decision to drop these features, including the identifiers (SK_ID_CURR), was based on the following reasons:

Missing Data: Many document-related features had extensive missing or empty data, making them less informative for the classification task.

Lack of Relevance: The presence of multiple document-related flags with minimal variation and unclear meaning suggested that these features might not directly relate to the client's loan repayment difficulties, which is the target variable (TARGET).

Simplification: By removing these features, the dataset was streamlined, focusing on the most relevant and informative attributes. This simplification aimed to improve the quality of the analysis and classifier, as it reduced noise and dimensionality in the data.

Overall, the decision to drop these features was made to ensure that the classifier could efficiently work with the most meaningful attributes, enhancing its predictive power and interpretability. This approach aligns with best machine learning feature selection and data preprocessing practices.

Data Cleaning

Statistics

Rows: 54 | Columns: 3

Name	Type	# Missing values ↓
EXT_SOURCE_3	Number (double)	8966
AMT_REQ_CREDIT_BUREAU_HOUR	Number (double)	6431
AMT_REQ_CREDIT_BUREAU_DAY	Number (double)	6431
AMT_REQ_CREDIT_BUREAU_WEEK	Number (double)	6431
AMT_REQ_CREDIT_BUREAU_MON	Number (double)	6431
AMT_REQ_CREDIT_BUREAU_QRT	Number (double)	6431
AMT_REQ_CREDIT_BUREAU_YEAR	Number (double)	6431
NAME_TYPE_SUITE	String	159
OBS_30_CNT_SOCIAL_CIRCLE	Number (double)	105
DEF_30_CNT_SOCIAL_CIRCLE	Number (double)	105
OBS_60_CNT_SOCIAL_CIRCLE	Number (double)	105
DEF_60_CNT_SOCIAL_CIRCLE	Number (double)	105
EXT_SOURCE_2	Number (double)	91
AMT_GOODS_PRICE	Number (double)	48
AMT_ANNUITY	Number (double)	1
CNT_FAM_MEMBERS	Number (double)	1

Figure 2 Missing Value Statistic

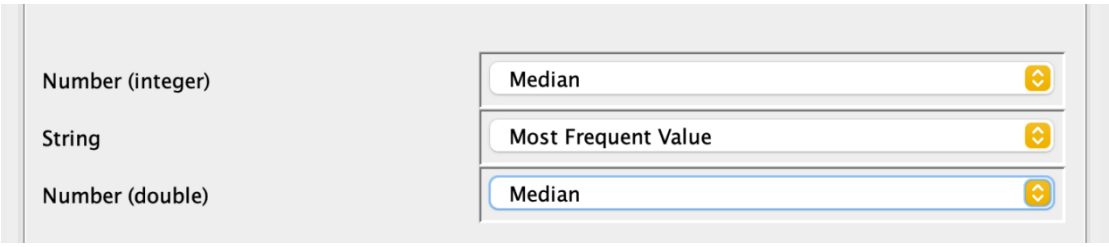


Figure 3 Missing Value Handling

Based on the statistics, several features in the dataset contain missing values. To address these missing values, a specific imputation strategy was employed, tailored to the data types of the features. For numerical (double) features, the median was used to impute missing values, ensuring that the central tendency of the variable was maintained while filling gaps in the data. This approach was applied to features such as "EXT_SOURCE_3," "AMT_REQ_CREDIT_BUREAU_HOUR,"

"AMT_REQ_CREDIT_BUREAU_DAY," and more.

The most frequent (mode) value was used for strito impute missing values. This method ensured that the imputed values aligned with the predominant category in each respective feature.

Implementing these imputation techniques aimed to complete the dataset, making it more suitable for the classification task while minimising the potential biases introduced by missing data. This comprehensive approach contributed to a robust and reliable dataset for the subsequent model development and analysis stages.

Data Binning

During the data preprocessing phase, I employed a technique known as "binning" to improve the usefulness of certain numerical features for our modelling process. The features that underwent binning included "DAYS_BIRTH," "DAYS_EMPLOYED," "DAYS_REGISTRATION," and "DAYS_ID_PUBLISH." The main idea behind binning is to convert continuous numerical variables into discrete categories or bins. This simplifies the representation of these variables and allows machine learning models to understand the data more effectively.
I experimented with two different bin configurations: one with six bins and another with 12 bins. To facilitate this comparison, I opted for width-based binning, where each bin has an equal width in terms of the variable's range. This width-based approach ensures that the bins cover the entire range of the variable evenly, avoiding an uneven distribution of data across the bins.
The choice of how many bins to use is essential because it can impact how well our machine-learning models perform. To decide which bin configuration to use, I conducted a thorough comparison. I applied the 6-bin and 12-bin configurations to different machine learning models, including decision trees and multilayer perceptrons (MLPs). The performance of these models was assessed using various scoring metrics to determine which bin configuration worked best.

After analysing the results, it became clear that the 12-bin approach consistently outperformed the 6-bin configuration. The increased granularity provided by the 12 bins allowed the models to capture more detailed information within the features. This led to improved predictive accuracy and overall model performance.

Consequently, I decided to proceed with the 12-bin configuration for these features. This choice was based on achieving our classification task's highest predictive accuracy and model effectiveness. It ensured that our data was well-prepared for the subsequent stages of model development.

During data preprocessing, I utilised a technique called "binning" to enhance the utility of numerical features such as "DAYS_BIRTH," "DAYS_EMPLOYED," "DAYS_REGISTRATION," and "DAYS_ID_PUBLISH." Binning involves converting continuous numerical variables into discrete categories or bins, simplifying their representation and aiding machine learning models in understanding the data more effectively. I chose a width-based binning approach, ensuring each bin had an equal width across the variable's range. This approach prevents data from being unevenly distributed across the bins, promoting fairness.

Two bin configurations were experimented with, one comprising six bins and the other with 12 bins. The choice of the number of bins is pivotal as it can significantly impact machine learning model performance. To determine the most suitable bin configuration, I compared 6-bin and 12-bin setups to various machine learning models, including decision trees and multilayer perceptrons (MLPs). The models were evaluated using scoring metrics to identify the optimal bin configuration.

Upon analysing the results, it was evident that the 12-bin approach consistently outperformed the 6-bin configuration. The increased granularity provided by the 12 bins allowed the models to capture more detailed information within the features. This led to improved predictive accuracy and overall model performance.

As a result, I opted to proceed with the 12-bin configuration for these features. This choice was made to achieve our classification task's highest predictive accuracy and model effectiveness. It ensured that our data was exceptionally prepared for the subsequent stages of model development, setting a solid foundation for our analysis.

Binarise Categorical

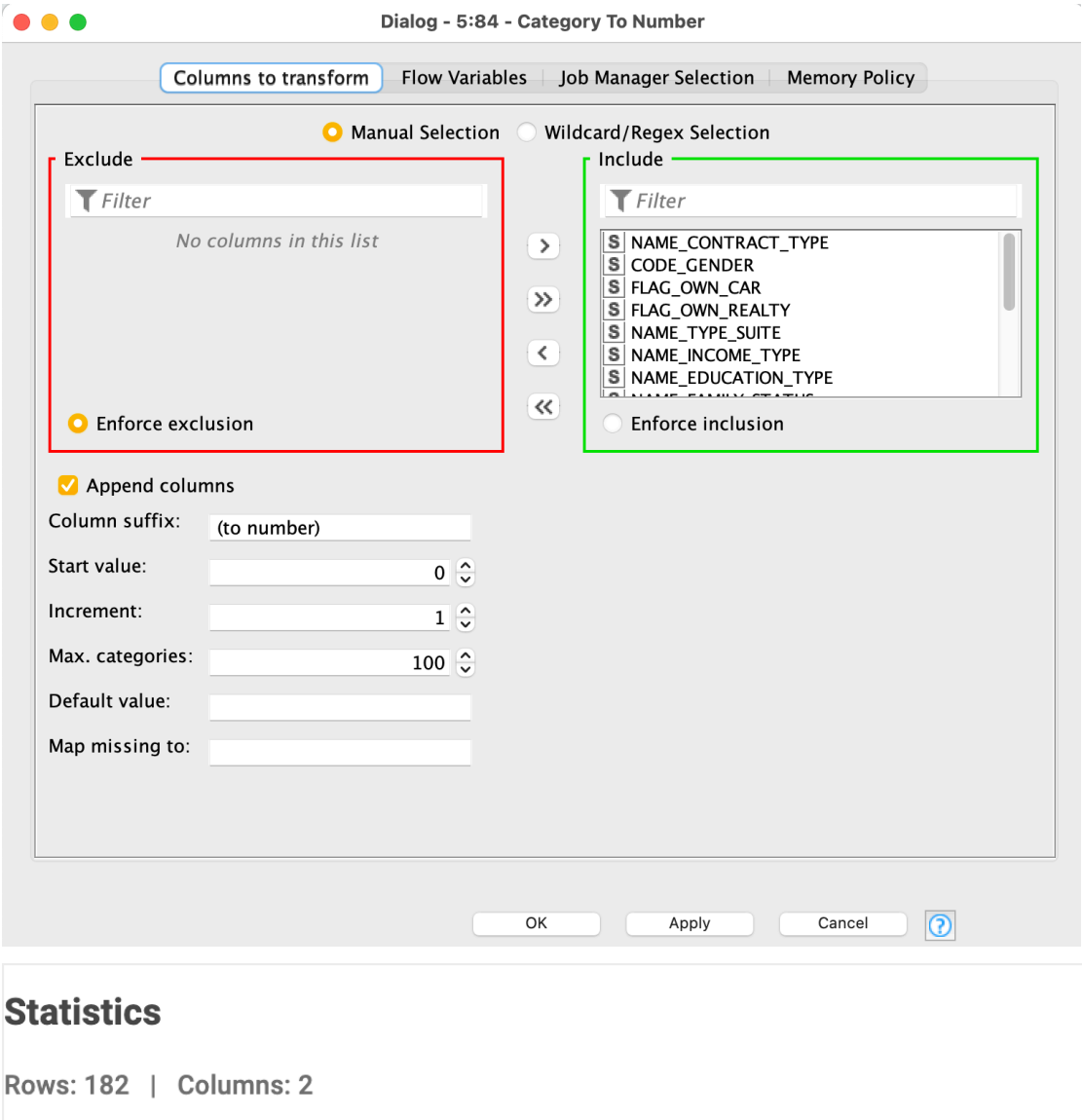


Figure 5 Label Encoding

I performed a crucial categorical feature encoding step to prepare our dataset for machine learning, specifically label encoding. The categorical features that underwent this transformation include:

- NAME_CONTRACT_TYPE
- CODE_GENDER
- FLAG_OWN_CAR
- FLAG_OWN_REALTY
- NAME_TYPE_SUITE
- NAME_INCOME_TYPE
- NAME_EDUCATION_TYPE

- NAME_FAMILY_STATUS
- NAME_HOUSING_TYPE
- WEEKDAY_APPR_PROCESS_START
- ORGANIZATION_TYPE
- DAYS_BIRTH [Binned]
- DAYS_EMPLOYED [Binned]
- DAYS_REGISTRATION [Binned]

The dataset underwent a crucial transformation to accommodate the requirement of machine learning models to work with numeric data. While one-hot encoding is a commonly used method for handling categorical features, it led to a substantial increase in the number of columns (182 total). This high dimensionality can negatively impact the efficiency of machine learning models and increase processing time. I tried one-hot encoding to mitigate these issues and prevent the curse of dimensionality but found it impractical due to the extensive column expansion.

I applied label encoding as a solution, effectively representing the categorical features with numeric labels. This approach retained the information within the categorical features while significantly reducing dimensionality, resulting in a more manageable dataset. This choice was critical to prevent many columns (one for each category in each feature), which can slow down the model training process and sometimes lead to overfitting.

Column Filtering

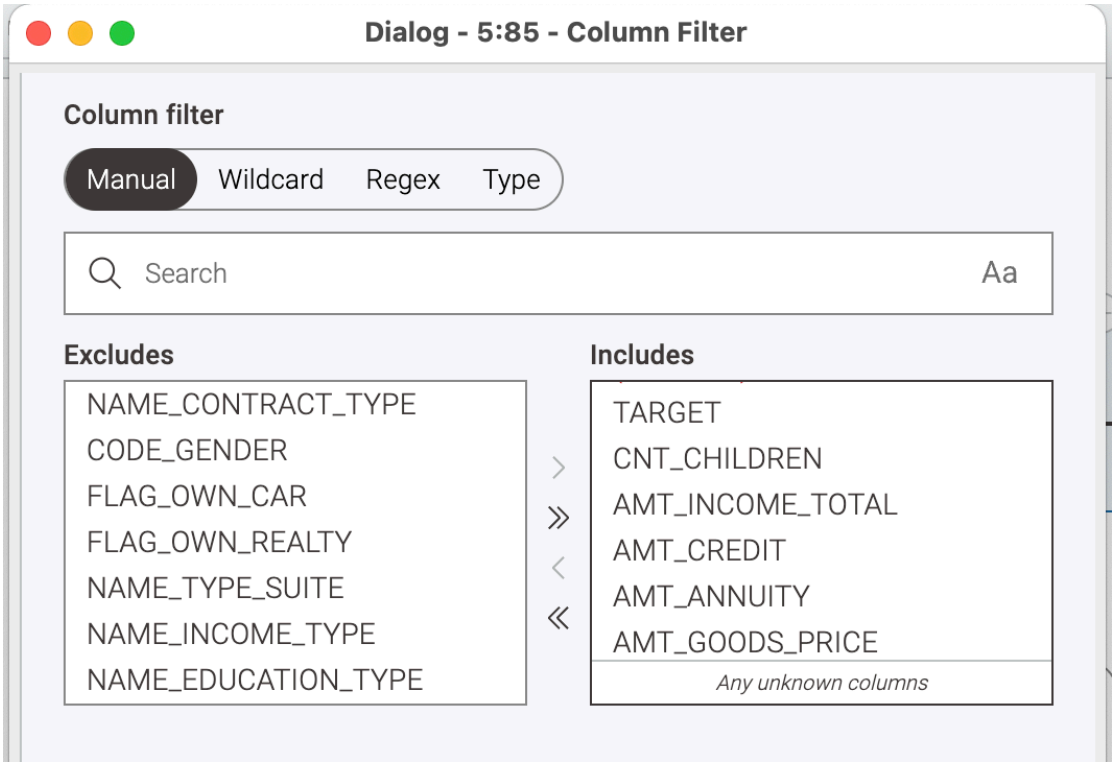


Figure 6 Column Filter

After applying the label encoding, I used column filtering to select and retain only the numeric features, including the original and the encoded categorical attributes. This streamlined the dataset to include features machine learning models could process effectively. This transformation ensured that the dataset was prepared to be integrated seamlessly into machine learning models, allowing them to learn and make predictions based on the combined numeric attributes.

Normalizer

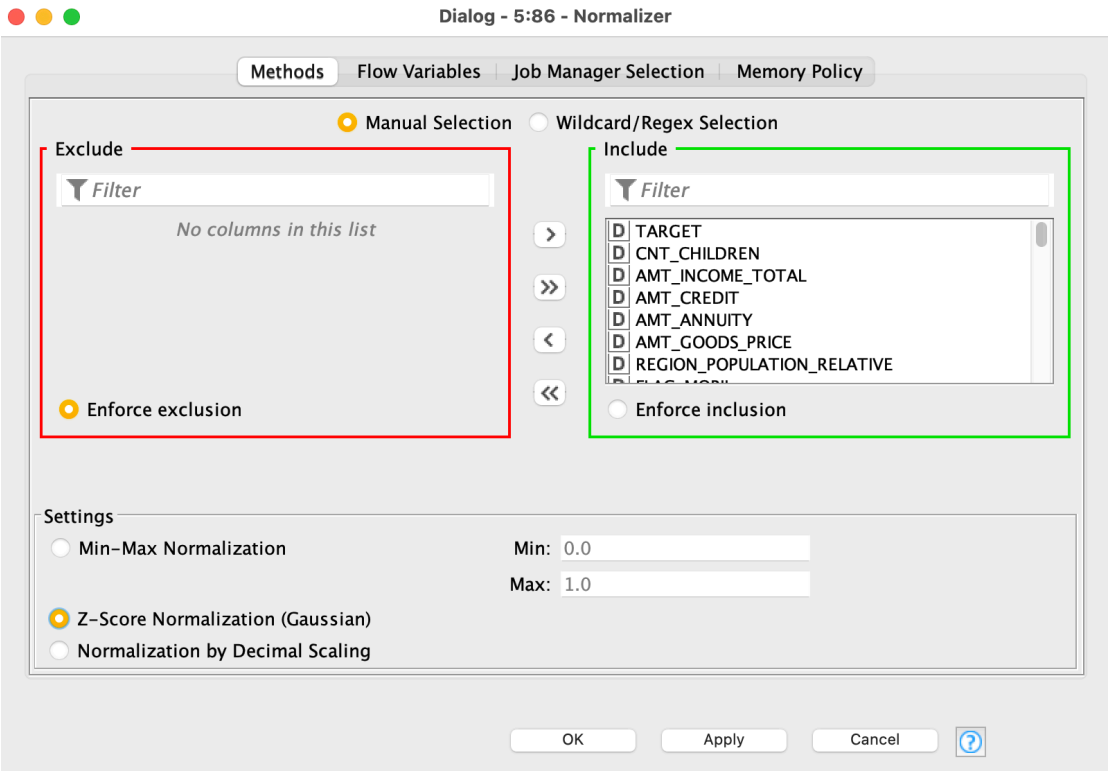


Figure 8 Normalizer

In the report, the decision to use Z-scores for feature scaling was made primarily to address the presence of outliers in certain features, such as "Amount of Income." Outliers are data points that significantly deviate from the rest of the data and can substantially impact the performance of machine learning models. Standardising the features using Z-scores aimed to mitigate the influence of outliers and ensure that all parts were on a similar scale.

The Z-score scaling technique, which normalises data to have a mean of 0 and a standard deviation of 1, is particularly effective at handling outliers because it doesn't depend on the range of the data. Instead, it focuses on how many standard deviations a data point is from the mean. This means extreme values are "tamed" and brought to a more consistent scale with the rest of the data.

For instance, in the case of income, there may be individuals with exceptionally high or low incomes far from the average. These extreme values would be adjusted using Z-scores to reflect how many standard deviations from the mean income, effectively downscaling or upscaling them to fit within the normal distribution.

This approach ensures that outliers don't disproportionately affect the performance of machine learning models, as they would if a simple min-max scaling method were employed. Using Z-scores, the dataset is better prepared for machine learning

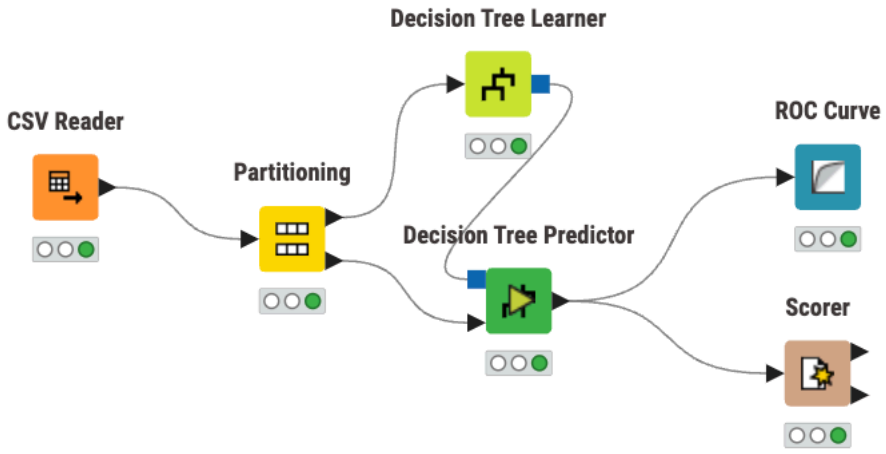
algorithms, ultimately improving the robustness and reliability of the classification task.

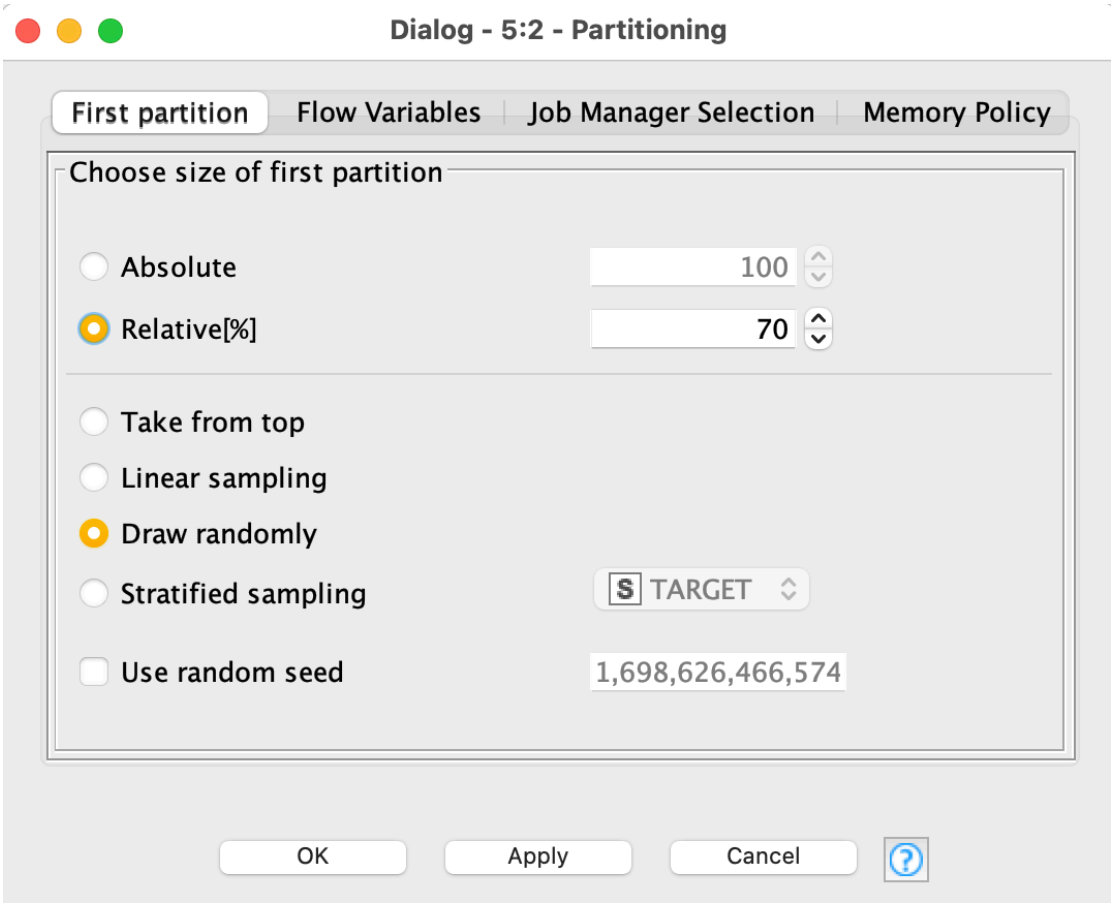
However, it's important to note that while Z-score scaling effectively addresses outliers, it may introduce challenges when applying specific machine-learning models. Models like k-nearest neighbours (K-NN) and support vector machines (SVM) can be sensitive to feature scales. Extremely high Z-scores in specific features could dominate distance calculations in K-NN or affect hyperplane placement in SVM.

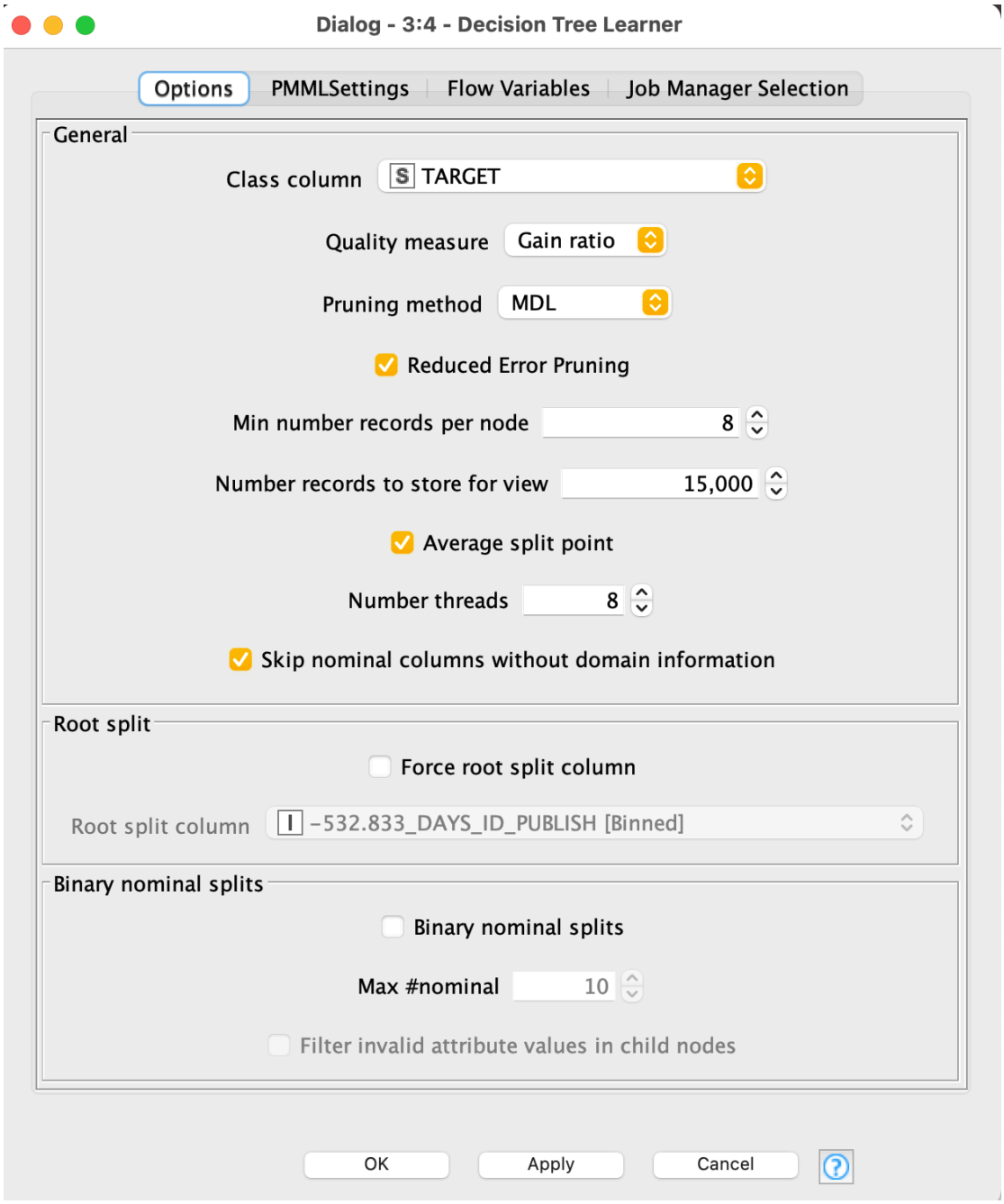
I chose a different scaling approach for specific models to account for these sensitivities. In such cases, we employed Min-Max normalisation. This technique scales features to a predetermined range, usually [0, 1], ensuring that all features are bounded within this range. Min-Max normalisation was particularly valuable when working with K-NN, SVM, and similar models. It helps maintain balance among features, especially when dealing with extreme values.

Classificatoin Techniques

Decision Tree Learner




Dialog - 5:2 - Partitioning
First partition
Flow Variables
Job Manager Selection
Memory Policy
Choose size of first partition
Absolute
100
Relative[%]
70
Take from top
Linear sampling
Draw randomly
Stratified sampling
TARGET
Use random seed
1,698,626,466,574
OK
Apply
Cancel


Dialog - 3:4 - Decision Tree Learner
Options
PMMLSettings
Flow Variables
Job Manager Selection
General
Class column
S TARGET
Quality measure
Gain ratio
Pruning method
MDL
Reduced Error Pruning
Min number records per node
8
Number records to store for view
15,000
Average split point
Number threads
8
Skip nominal columns without domain information
Root split
Force root split column
Root split column
I -532.833_DAYS_ID_PUBLISH [Binned]
Binary nominal splits
Binary nominal splits
Max #nominal
10
Filter invalid attribute values in child nodes
OK
Apply
Cancel

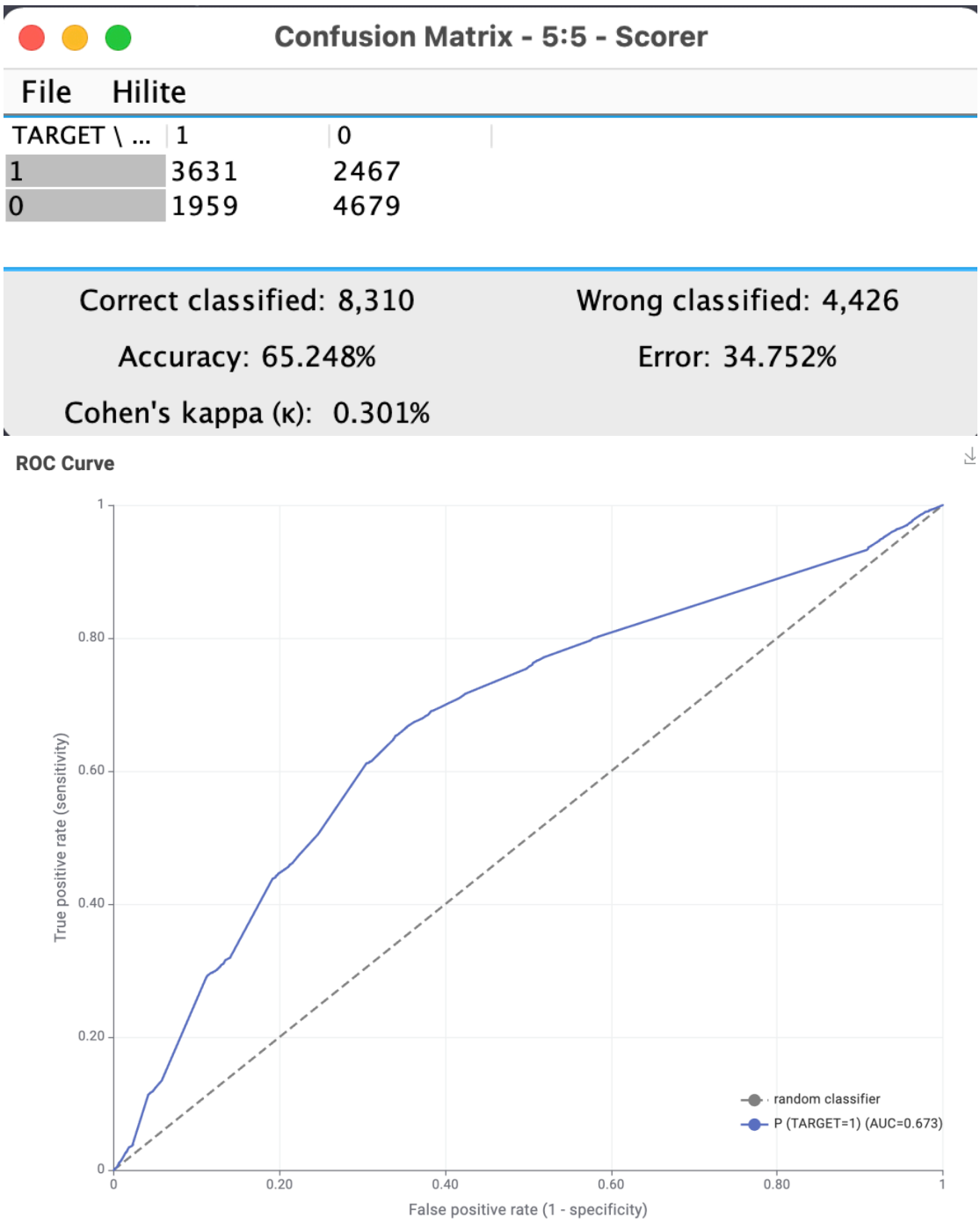


Figure 9 Decision tree learner parameter and evaluation

I started with a preprocessed dataset for the decision tree classification, including encoding and feature scaling. The dataset was divided into a training set, which constituted 70% of the data, and a testing set, which accounted for the remaining 30%. I conducted experiments to identify the most influential parameter configuration during the decision tree modelling.

A decision tree is a machine learning algorithm that functions as a predictive model by recursively partitioning the dataset into subsets based on the attributes to create

segments where the target variable (in this case, the "TARGET" attribute) exhibits maximum homogeneity within each piece. It is a powerful tool for classification tasks because it can break down complex decision-making processes into a series of straightforward, rule-based questions. The decision tree begins with the entire dataset and repeatedly asks questions about its features. It creates nodes and branches in a tree-like structure, ultimately leading to leaf nodes where the final classifications are determined (IBM, n.d.).

In my experimentation with decision trees, I explored the following key parameters:

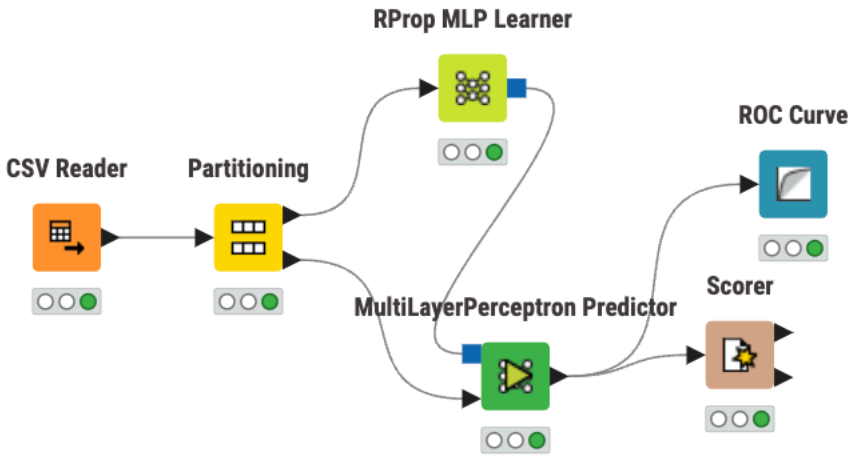
- Criterion: I tested the "Gini index" and "gain ratio" to evaluate the quality of splits in the decision tree.
- Pruning Method: I assessed the decision tree without pruning and employed "Minimum Leaf Probability Pruning (MLP)" as the pruning method.
- Minimum Number of Records in Nodes: I varied the minimum number required in each node, experimenting with different values, including 4.

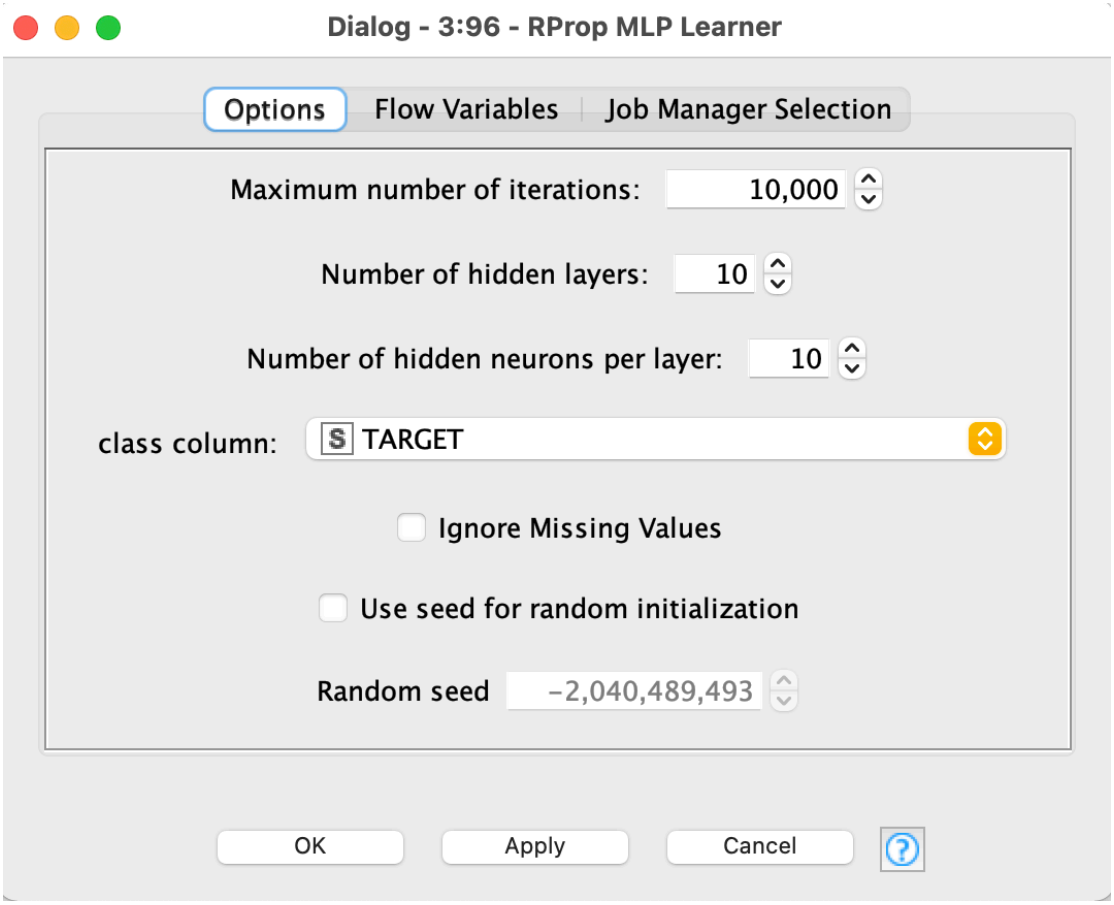
The experimentation yielded accuracy results of 58.51% for the "Gini index," 64.14% for the "Gini index" with MLP as the pruning method, and 65.25% for the "gain ratio" with MLP as the pruning method. Comparing these results demonstrated that the "gain ratio" and the application of MLP for pruning substantially enhanced the accuracy of the decision tree.

Moreover, adjusting the minimum number of records in nodes did not substantially impact the model's accuracy compared to the Gini index. In addition, it's worth noting that the adjustment of the minimum number of node records didn't significantly influence the model's accuracy. However, a noteworthy observation was made with the "gain ratio" criterion. When utilising a minimum of 4 records in nodes, there was a 1% increase in accuracy compared to the configuration with a minimum of 8 records.

As a result, I decided to proceed with the configuration that employed the "gain ratio" as the criterion and used MLP as the pruning method. I specified a minimum of 4 records in nodes. This choice was guided by the fact that pushing the minimum number of records further didn't significantly enhance accuracy.

Multi-layer perceptron (MLP)





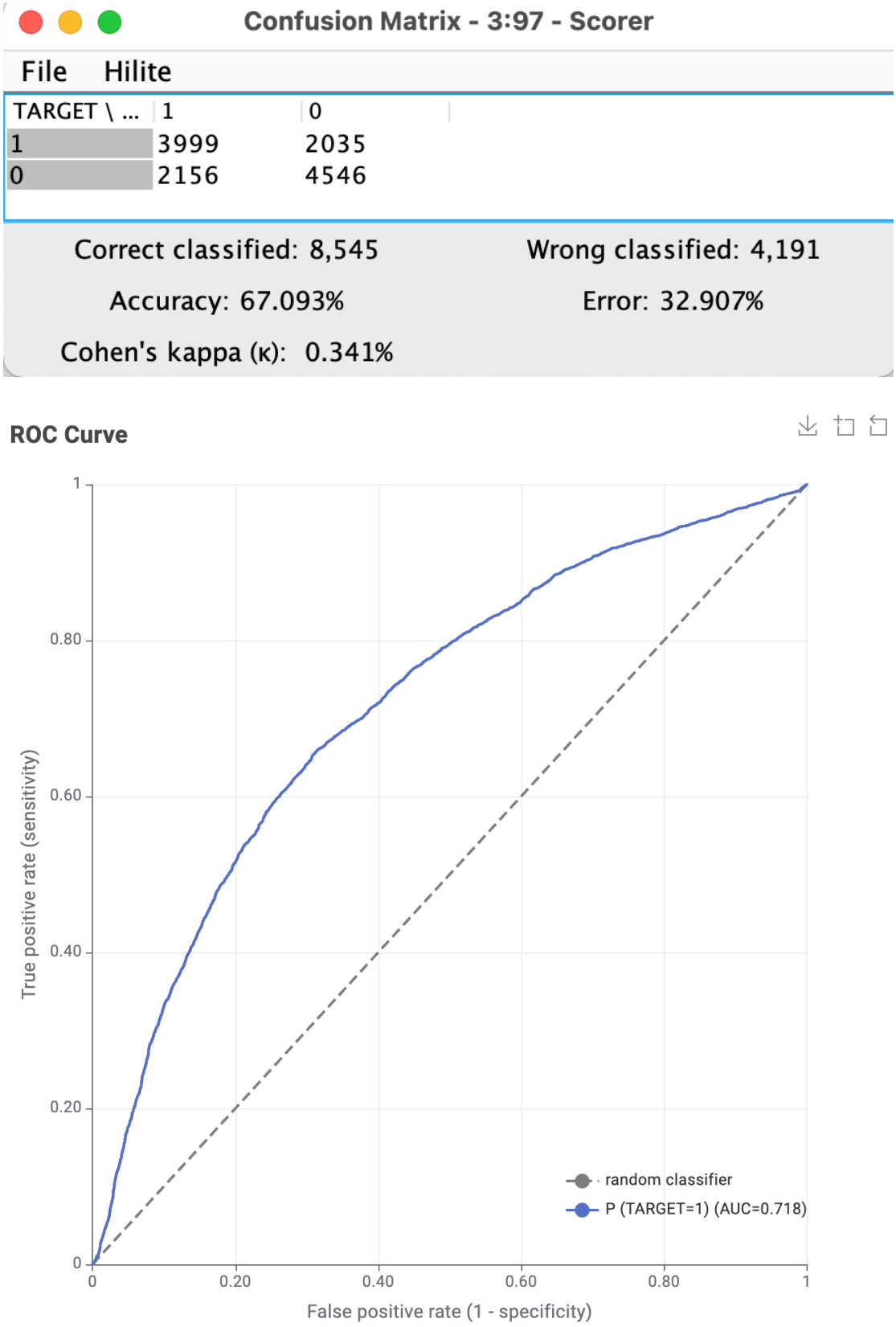


Figure 10 MLP learner parameter and evaluation

In pursuing this task's most accurate classification model, I delved into neural networks and, specifically, the multi-layer perceptron (MLP). MLP is an artificial neural network with multiple layers comprising an input layer, one or more hidden layers, and an output layer. Its capacity to capture complex patterns and relationships within the data makes it a promising candidate for our classification task (Sharma, 2019).

In my initial experiment with MLP, I set up a network with seven hidden layers, each containing 12 neurons. The model yielded an accuracy score of 65.59%, showing promise. However, I was determined to explore the network's potential further.

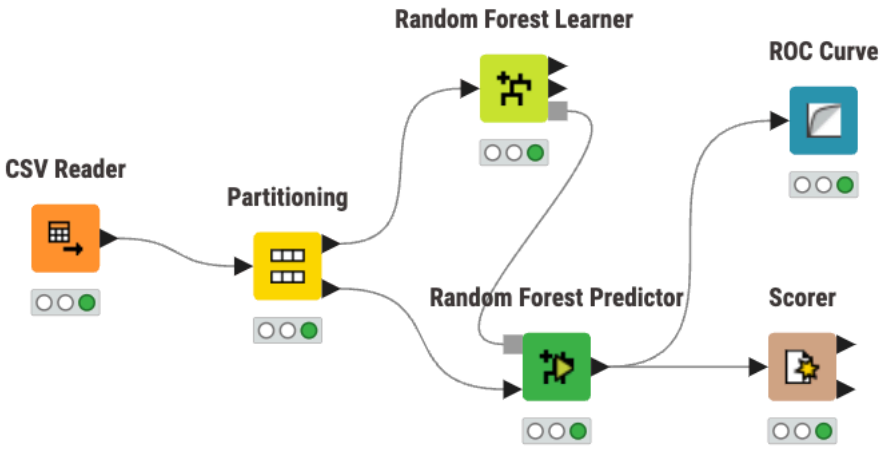
To optimise the performance, I increased the number of hidden layers to 10, each featuring ten neurons. Additionally, I extended the maximum number of iterations to 10,000. This refined configuration demonstrated remarkable results, with an accuracy score of 67.09%. Notably, the area under the ROC curve (AUC) score reached 0.718, surpassing the average. The improved performance indicated that this configuration was superior in capturing intricate patterns within the data.

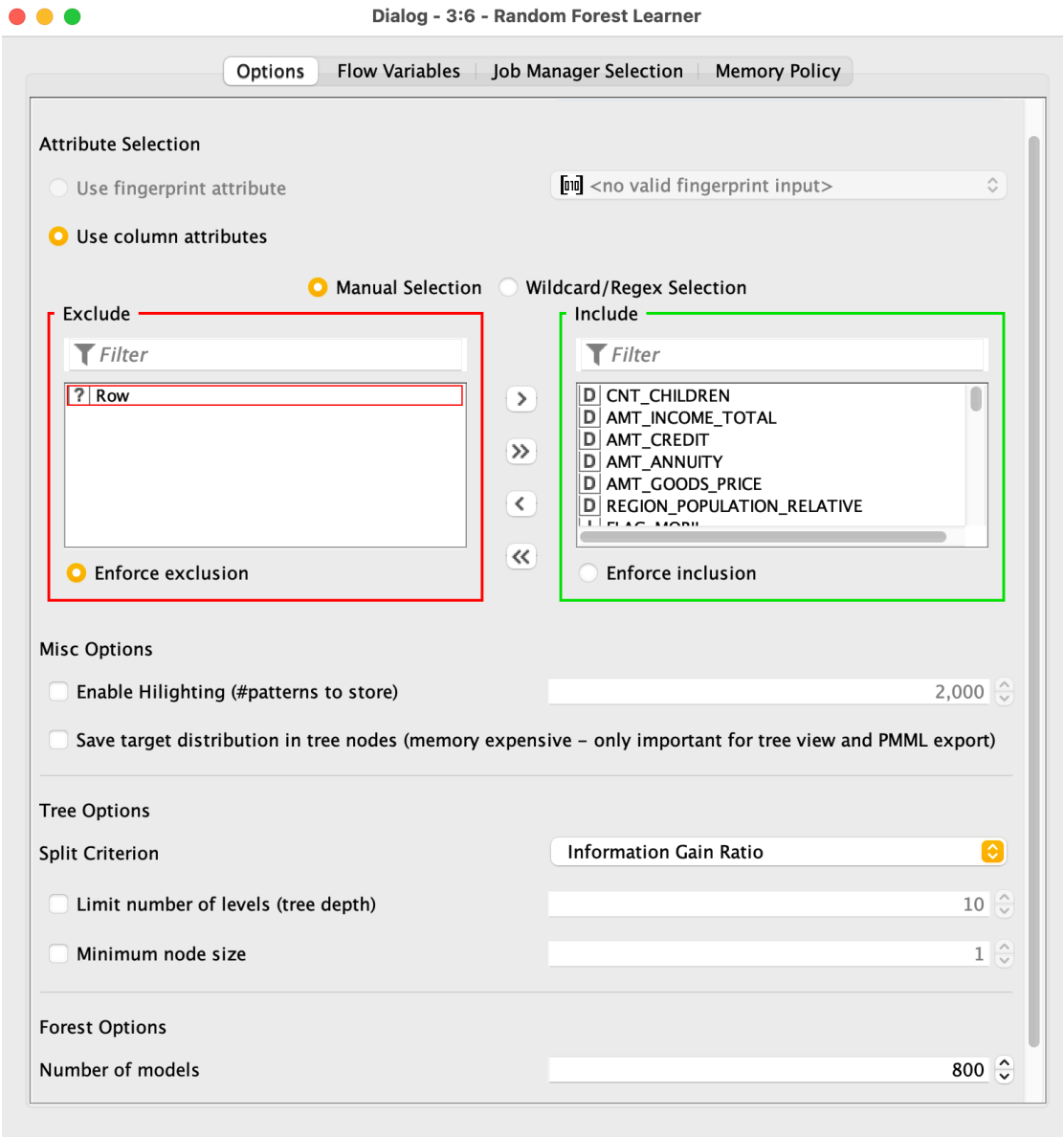
Furthermore, I conducted experiments to evaluate the impact of different feature encoding methods on the MLP model's performance. I initially employed one-hot encoding for feature binarisation, which produced an accuracy score of 64.94%. Intriguingly, the accuracy surged to 65.3% when transitioning to label encoding. This improvement with label encoding can be attributed to its ability to prevent the curse of dimensionality, making it a more suitable choice for our task.

I discovered that setting the number of hidden layers and neurons per layer to 10 was optimal regarding parameter tuning. Additionally, increasing the maximum number of iterations to 10,000 enhanced the model's performance. This comprehensive approach allowed me to harness the full potential of the multi-layer perceptron for our classification task.

In summary, the multi-layer perceptron (MLP) is a neural network architecture with multiple layers that can capture intricate patterns within data. By refining the configuration to include ten hidden layers, ten neurons per layer, and extending the maximum iterations to 10,000, I achieved an accuracy score of 67.09% and an AUC score of 0.718. The shift from one-hot encoding to label encoding also improved the model's performance. This approach showcases neural networks' power and adaptability for complex classification tasks.

Random Forest





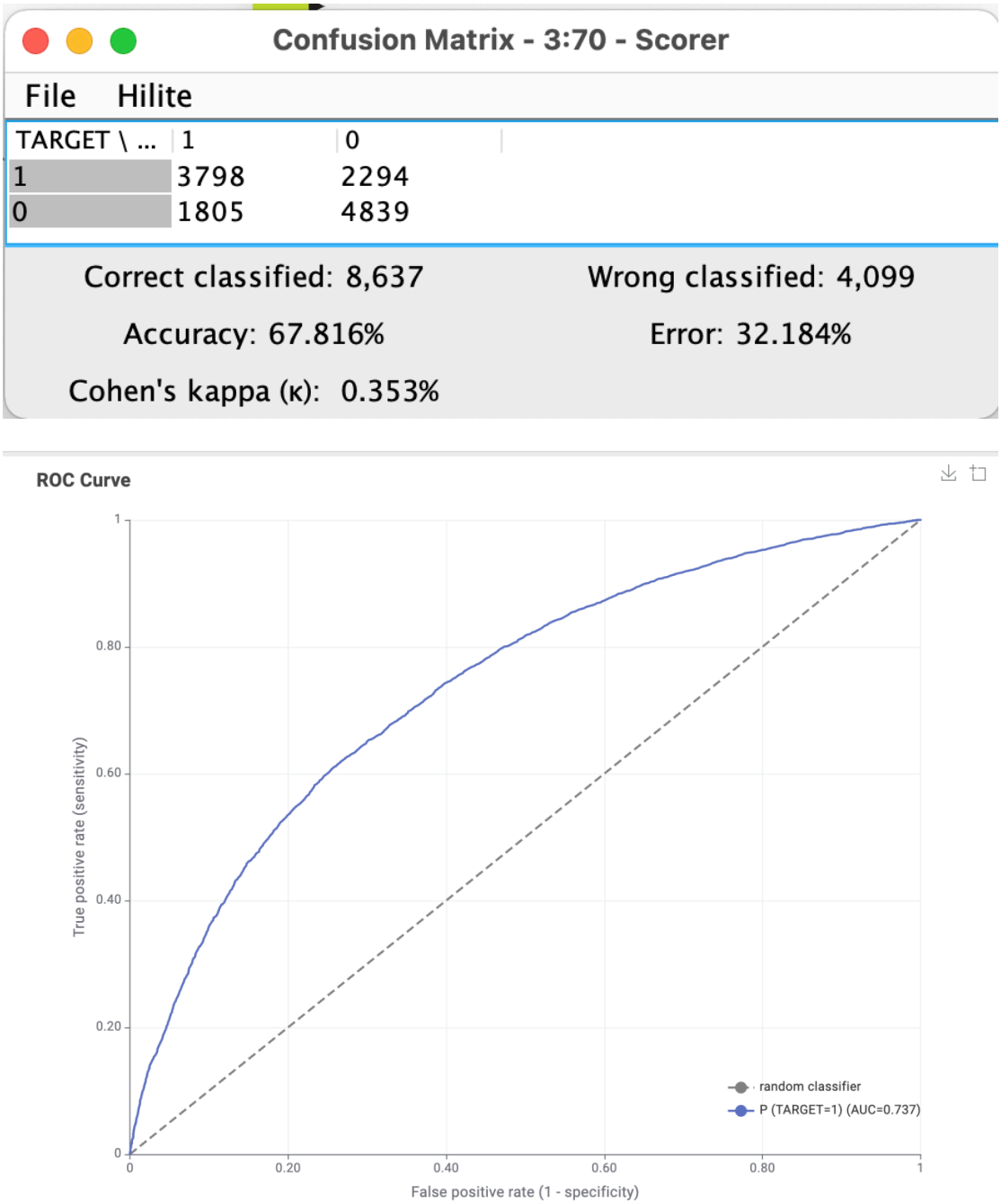


Figure 11 Random Forest learner parameter and evaluation

In machine learning, "Random Forest" is a powerful technique. It uses a group of decision trees to improve predictions and avoid overfitting. Each tree in the "forest" uses a different part of the data and a random set of features, making predictions more robust and less affected by outliers or noise. The final prediction is like a vote from all the trees. Random Forest is known for its reliability and flexibility, making it a popular choice for various problems (IBM, n.d.).

To unlock the potential of ensemble learning, I explored Random Forest in my journey.

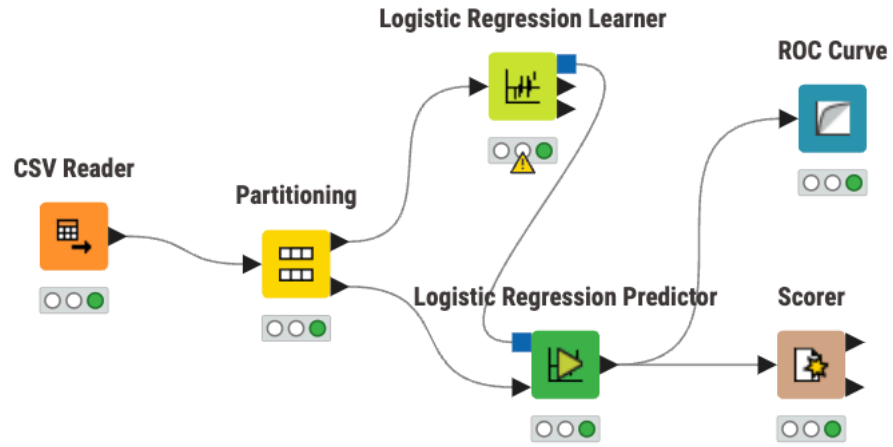
I conducted experiments to determine how to split data when building the forest. I tested three criteria: information gain, information gain ratio, and the Gini index.
The results revealed intriguing insights into the performance of these criteria. The accuracy scores obtained were 66.68% for information gain, 67.19% for information gain ratio, and 67.51% for the Gini index. Remarkably, the information gain ratio emerged as the champion, offering the highest accuracy among the contenders.

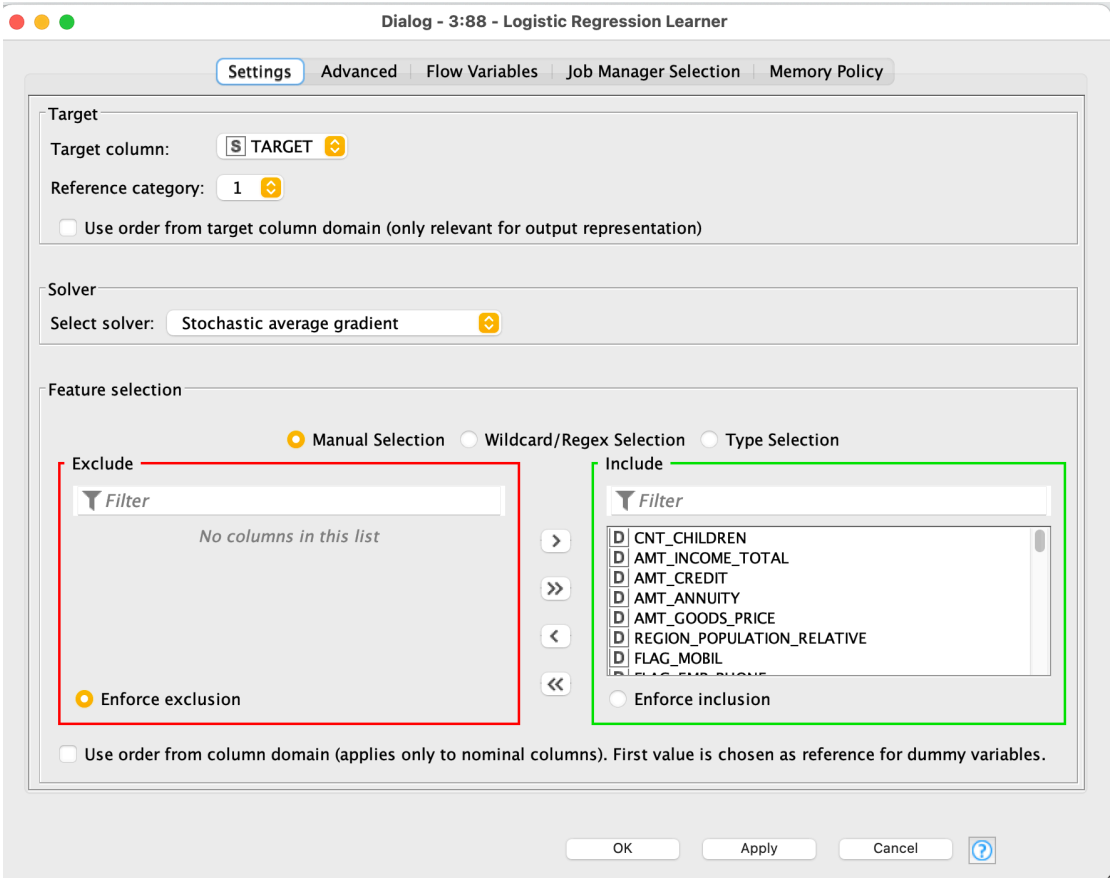
I explored the influence of the number of models (trees) within the ensemble to further optimise the random forest model. I embarked on an experiment from 100 to 800 models, investigating how this parameter influenced the model's performance. While the number of models certainly had an effect, it was notable that the increase in performance from 100 to 800 models, though present, wasn't exceptionally substantial.

The experimentation underscored the significance of balancing computational resources and model performance. Eventually, a configuration employing 800 models yielded an accuracy score of 67.814%, affirming its position as the best performer. However, it's worth noting that the change in accuracy between different model quantities wasn't as profound as expected.

The final parameter configuration I selected for the random forest incorporated the information gain ratio as the split criterion, coupled with 800 models. This choice was founded on the principle that the information gain ratio offered the highest accuracy, and the configuration with 800 models exhibited a commendable level of performance, making it a well-balanced choice for our classification task.

Logistic Regression





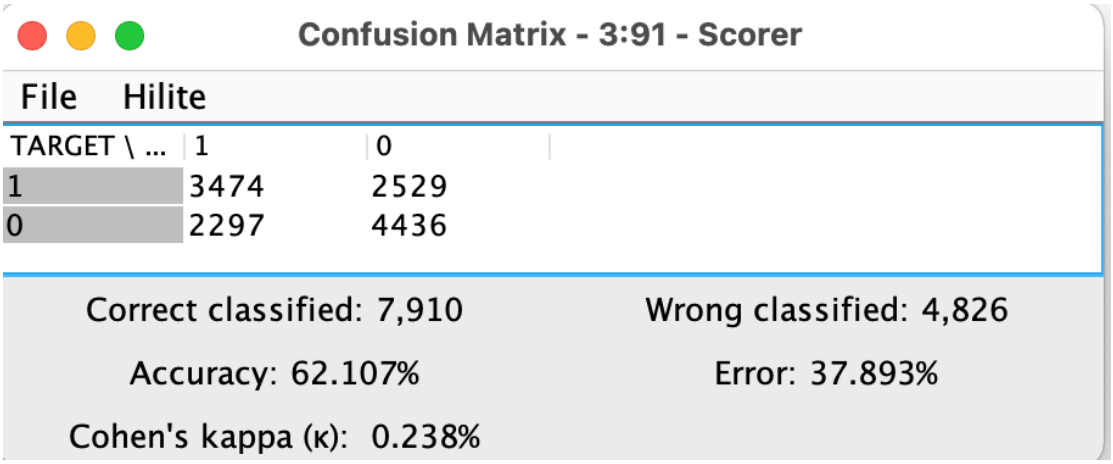
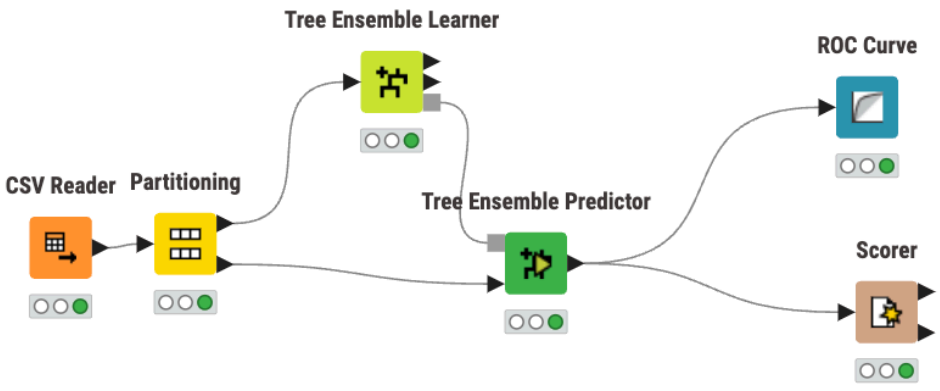
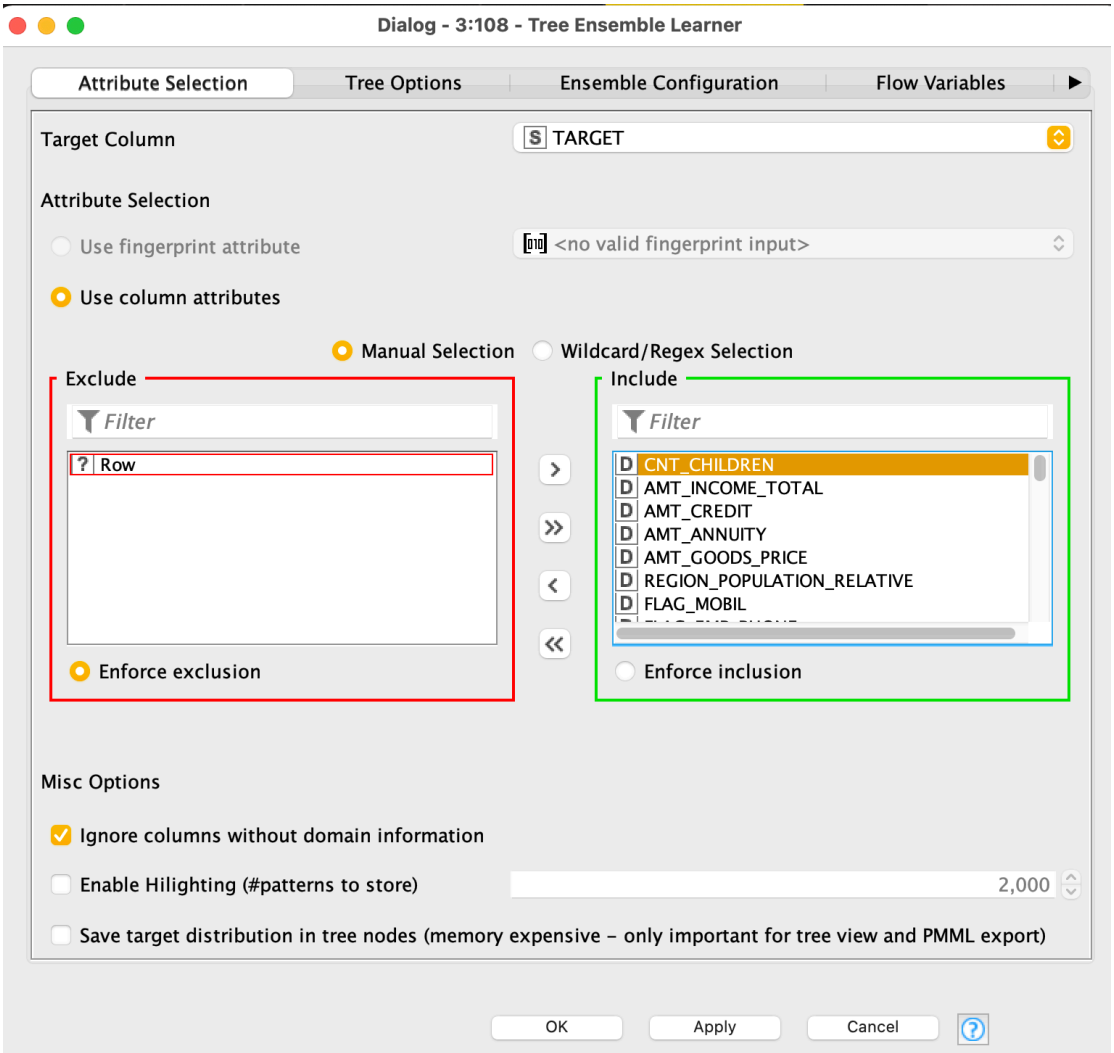


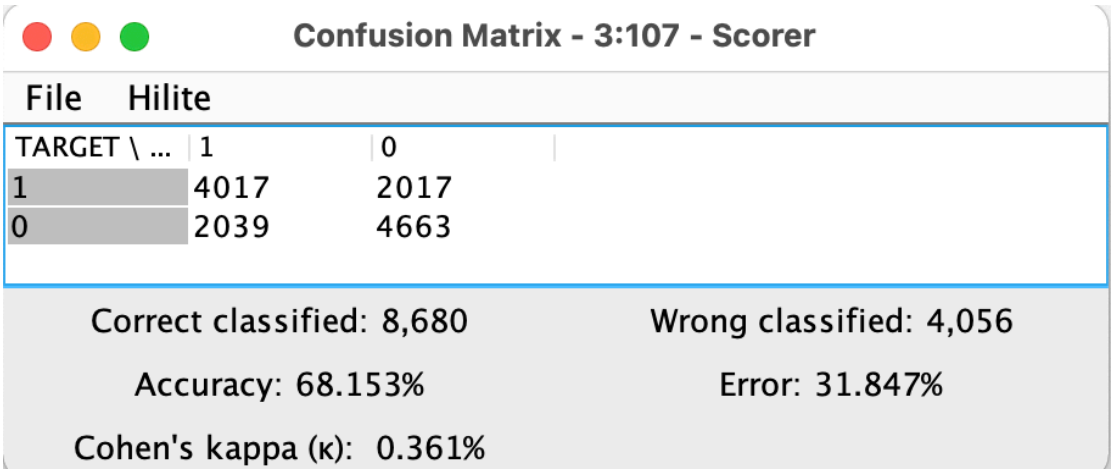
Figure 12 Logistic Regression Learner parameter and evaluation

I have explored the logistic regression classification technique in my modelling efforts. Logistic regression is a statistical method used for binary classification tasks. It is particularly suited for problems where the goal is to predict the probability of an instance belonging to one of two classes. The model generates a logistic curve that provides a probability estimate, and a threshold is applied to determine the class assignment. However, in this specific case, the initial performance of the logistic regression model was not as strong, resulting in an accuracy score of 62.107%. This relatively lower accuracy score prompted me to focus on alternative techniques that exhibited better predictive capabilities during my experimentation.

Tree Ensemble




Dialog - 3:108 - Tree Ensemble Learner
Attribute Selection
Tree Options
Ensemble Configuration
Flow Variables
Target Column
TARGET
Attribute Selection
Use fingerprint attribute
<no valid fingerprint input>
Use column attributes
Manual Selection
Wildcard/Regex Selection
Exclude
Filter
Row
Enforce exclusion
Include
Filter
CNT_CHILDREN
AMT_INCOME_TOTAL
AMT_CREDIT
AMT_ANNUITY
AMT_GOODS_PRICE
REGION_POPULATION_RELATIVE
FLAG_MOBIL
Enforce inclusion
Misc Options
Ignore columns without domain information
Enable Hilighting (#patterns to store)
2,000
Save target distribution in tree nodes (memory expensive - only important for tree view and PMML export)
OK
Apply
Cancel


Confusion Matrix - 3:107 - Scorer
File
Hilite
TARGET \ ...
1
0
1
4017
2017
0
2039
4663
Correct classified: 8,680
Wrong classified: 4,056
Accuracy: 68.153%
Error: 31.847%
Cohen's kappa (κ): 0.361%

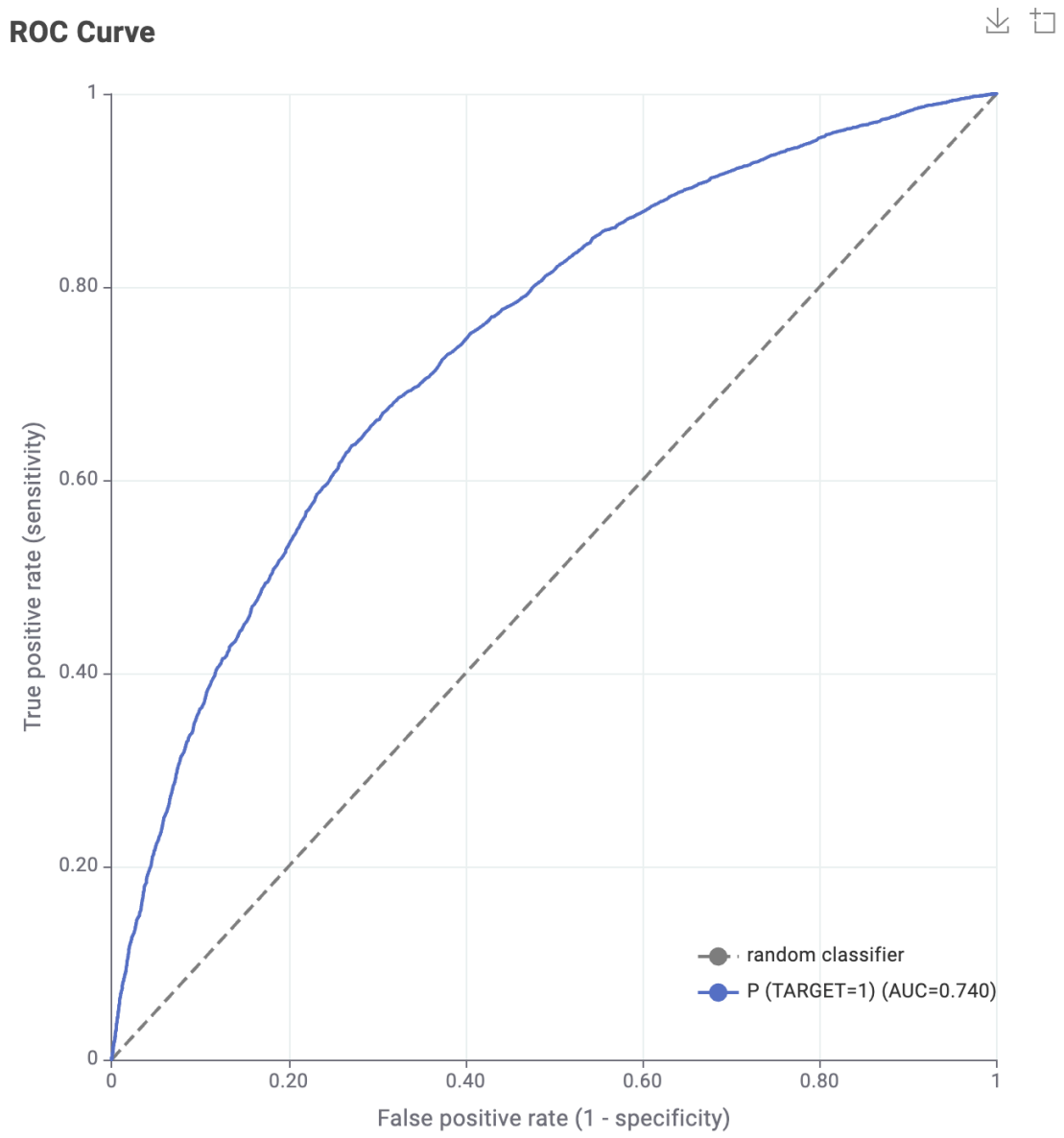
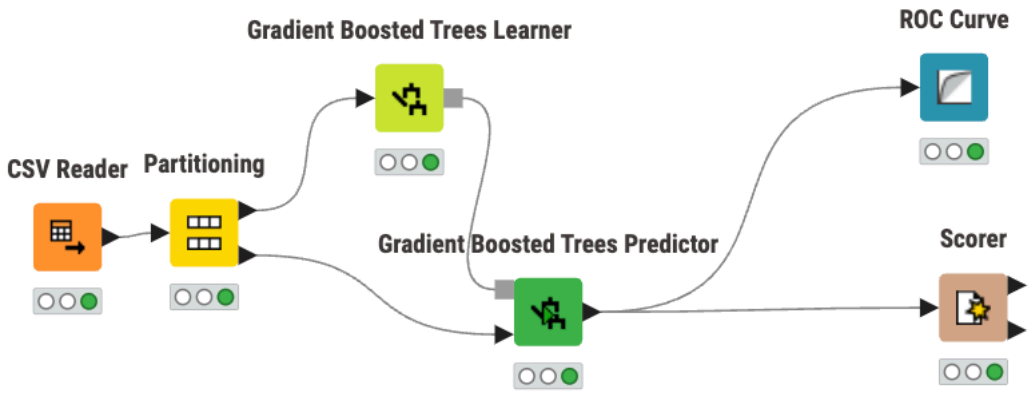


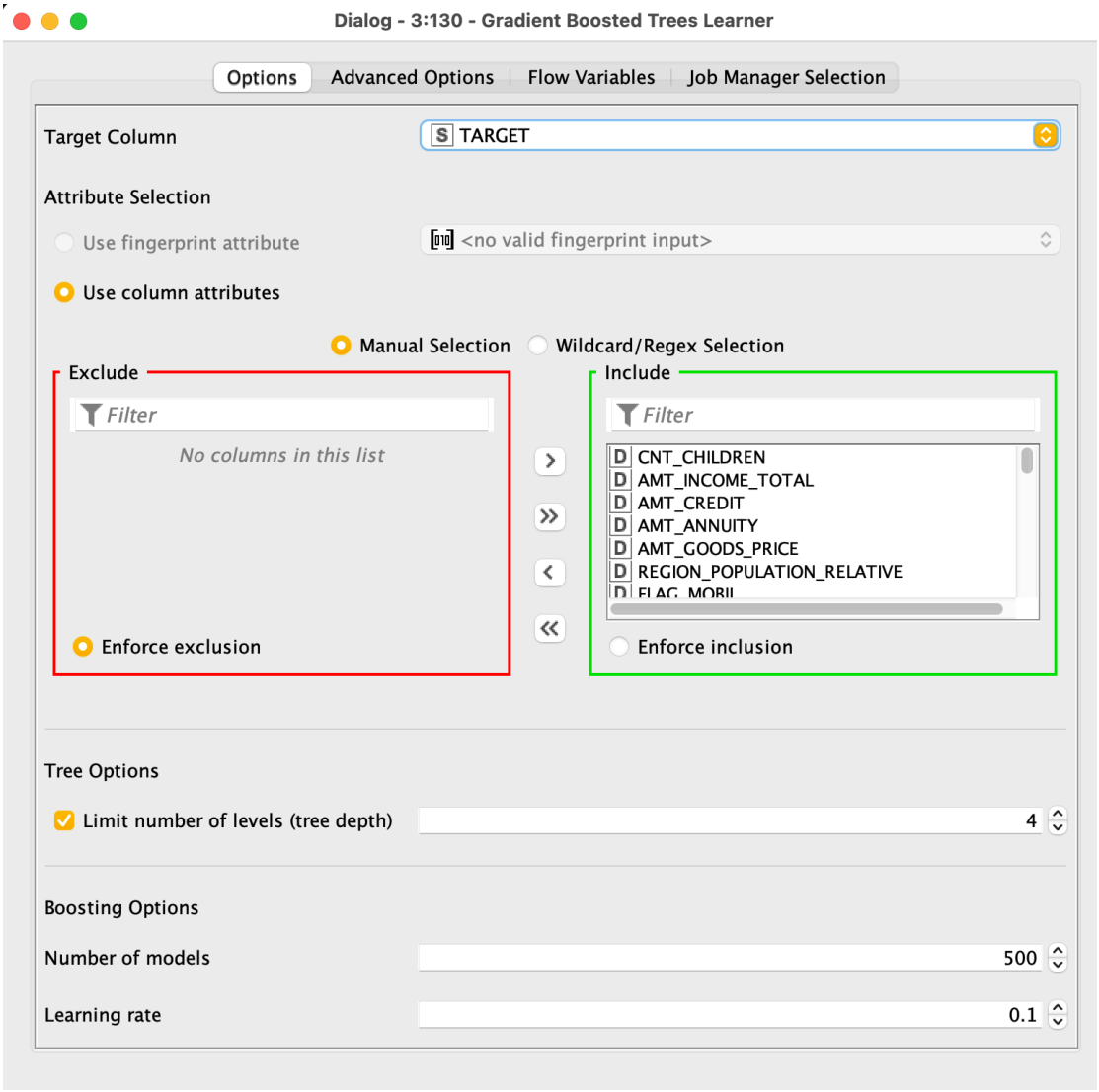
Figure 13 Tree Ensemble learner parameter and evaluation

I also ventured into experimenting with tree ensemble models, which initially yielded an accuracy score of 66.434. These experiments involved adjusting various parameters to optimise the model's performance. However, pursuing the most accurate model led me to explore the potential of increasing the number of models in the ensemble to 800 while also utilising the "gain ratio" index for determining the best splits. This refined configuration resulted in a substantial boost in accuracy, achieving an impressive score of 68.15.

Notably, this tree ensemble model displayed a slightly superior performance with parameters similar to the Random Forest model. The choice of the "gain ratio" index for splitting played a pivotal role in enhancing the model's accuracy, highlighting the effectiveness of this ensemble learning technique.

Gradient Boosted Tree





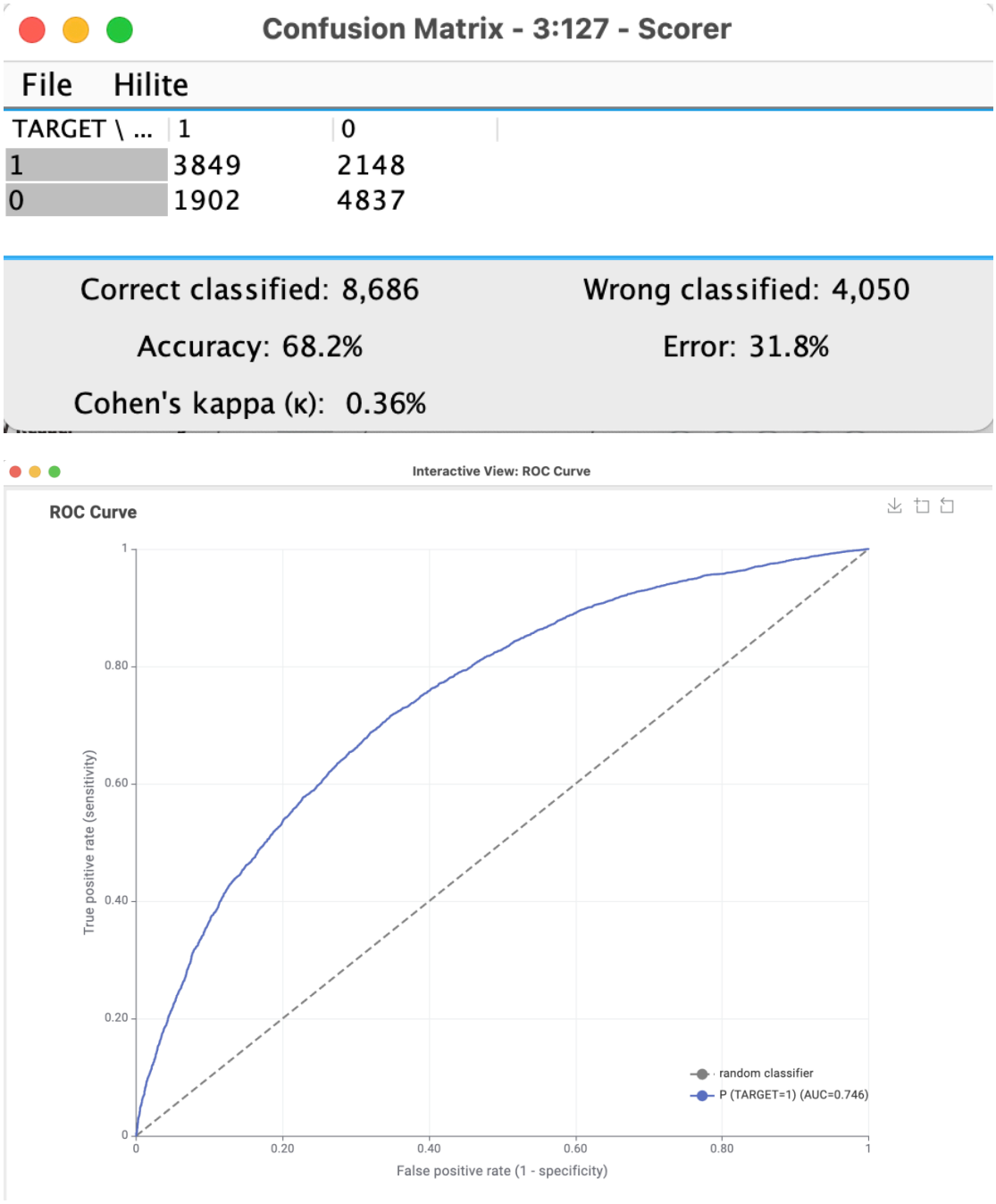


Figure 14 Gradient Boosted Tree learner parameter and evaluation

Furthermore, my exploration extended into gradient-boosted trees, a robust ensemble learning technique known for its sequential construction of multiple decision trees. The fundamental principle of gradient boosting is addressing the errors made by preceding trees in the sequence, progressively refining the model's predictive accuracy. This approach strategically hones in on areas where prior models faced challenges, rendering it a potent tool for boosting overall model performance.

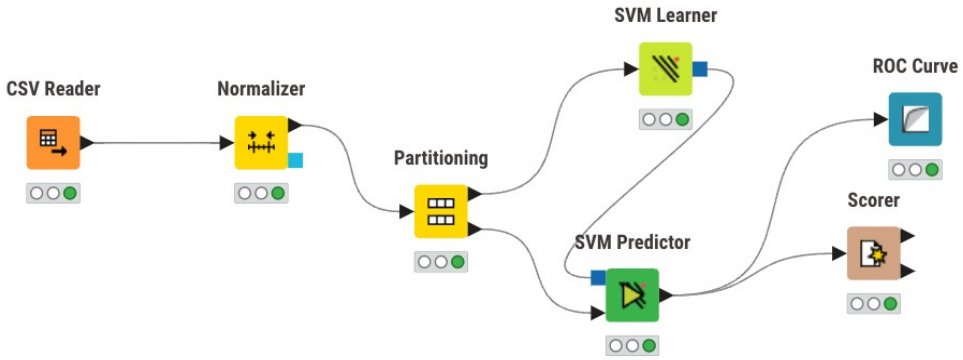
In my experimentation with gradient-boosted trees, I delved into various critical parameters, particularly the maximum depth of individual trees and the number of

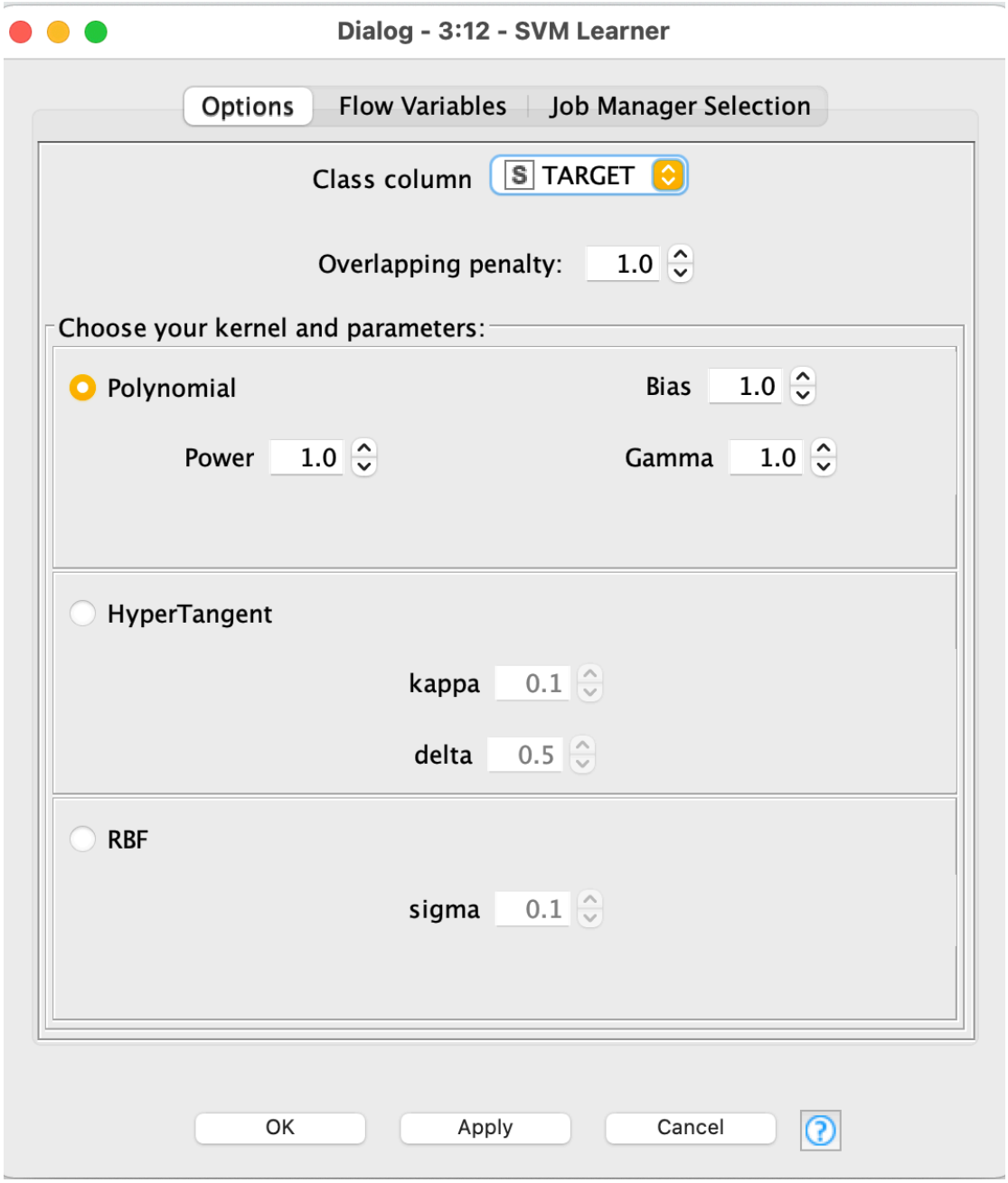
boosting iterations (models). The depth of trees underwent rigorous testing, spanning from 2 to 6, while the count of boosting iterations ranged from 100 to 800.

The outcomes of these experiments unveiled that a tree depth of 4 showcased the most exceptional performance among the different depth configurations. Furthermore, among the boosting iterations, 500 models emerged as the most optimal choice, surpassing 200 and 800 iterations. This meticulously fine-tuned configuration substantially positively impacted the model's accuracy, firmly establishing it as the favoured choice for our classification task.

This choice culminated in an impressive accuracy score 68.2, further accentuated by an AUC of 0.75 when predicting positive class outcomes. The decision to employ a learning rate of 0.1 added to the overall effectiveness of this robust gradient-boosted trees model, bolstering its position as a high-performing solution for our loan repayment prediction task.

Support Vector Machines (SVM)




Dialog - 3:12 - SVM Learner
Options
Flow Variables
Job Manager Selection
Class column
S TARGET
Overlapping penalty:
1.0
Choose your kernel and parameters:
Polynomial
Bias
1.0
Power
1.0
Gamma
1.0
HyperTangent
kappa
0.1
delta
0.5
RBF
sigma
0.1
OK
Apply
Cancel

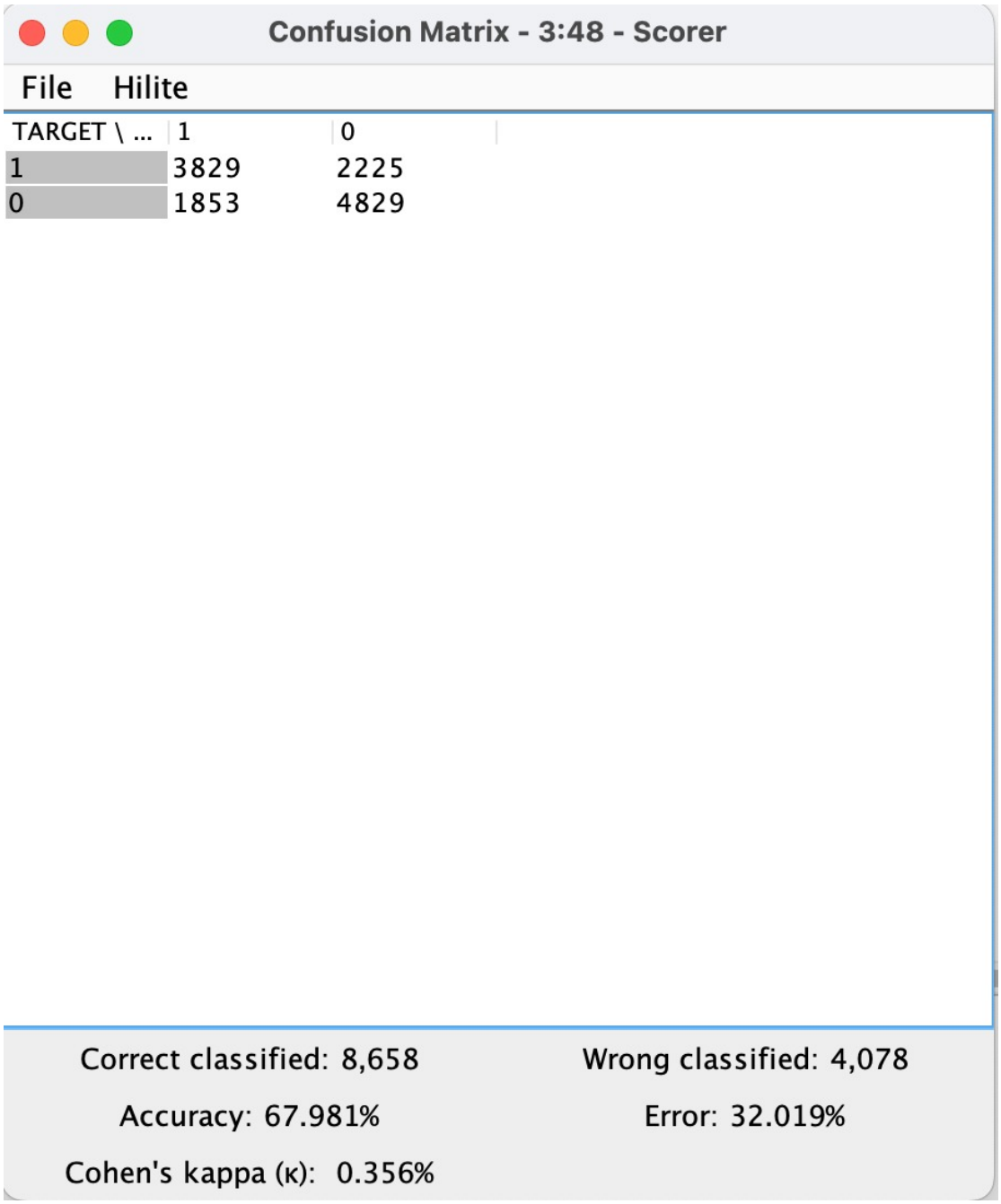


Figure 15 SVM learner parameter and evaluation

In my exploration of machine learning models, I delved into the realm of Support Vector Machines (SVM), a powerful and versatile tool for classification tasks. SVMs are widely recognised for efficiently handling both linear and non-linear classification problems. To ensure the dataset's suitability for SVM, I performed various preprocessing techniques, comparing the impact of one-hot encoding and label encoding. Additionally, I assessed different normalisation methods, including min-max normalisation and Z-score normalisation.

The results revealed that Z-score normalisation and label encoding produced the

most favourable outcomes, exhibiting the best performance in terms of classification accuracy. These choices were pivotal in ensuring the dataset was well-prepared for the SVM model.

I utilised a polynomial kernel for the SVM model with parameters set to a power of 1, bias of 1, and gamma of 1. This parameter configuration yielded an impressive accuracy score of 67.98, underlining the effectiveness of the SVM model in tackling the loan repayment prediction task.

The rigorous experimentation and fine-tuning of preprocessing techniques and model parameters have contributed to selecting SVM as a viable choice for this classification task. Its robust performance, combined with the optimal preprocessing methods, makes SVM a compelling addition to the ensemble of models aimed at achieving the highest predictive accuracy.

Justify the classifier selected

In the process of evaluating machine learning models for our data prediction task, three top-performing models emerged: Random Forest, Gradient Boosted Trees, and Support Vector Machines (SVM), each with unique strengths.

Random Forest is a powerful ensemble technique known for handling diverse feature types, addressing overfitting, and managing missing data effectively. It demonstrated competitive accuracy at 67.814% and is well-suited for comprehensive classification tasks due to its capacity to capture complex relationships and patterns.

Gradient Boosted Trees, another ensemble method, impressed with an accuracy score of 68.2%. This model excels at capturing intricate data patterns and is particularly suitable for improving areas where previous models struggled, enhancing overall performance.

Support Vector Machines (SVM) are known for their versatility and effectiveness in linear and non-linear classification tasks. With an accuracy score of 67.98%, SVMs are robust and suitable for handling complex, non-linear relationships in data.

When comparing these models, they all demonstrate solid accuracy in the 67% to 68% range. Each has its unique strengths: Random Forest is robust, Gradient

Boosted Trees excels at capturing intricate patterns, and SVM is versatile and effective for complex classification.

After a thorough evaluation, gradient-boosted trees emerged as the most suitable choice for this data prediction task. It achieved the highest accuracy at 68.2% and demonstrated the ability to capture subtle patterns and relationships within the dataset. Its sequential learning approach makes it particularly adept at tackling complex classification tasks, giving it the edge over the other models.

While Random Forest and SVM also delivered competitive results, the slightly higher accuracy and capability of Gradient Boosted Trees to address specific areas where prior models struggled to make it the recommended choice for this specific data prediction task.

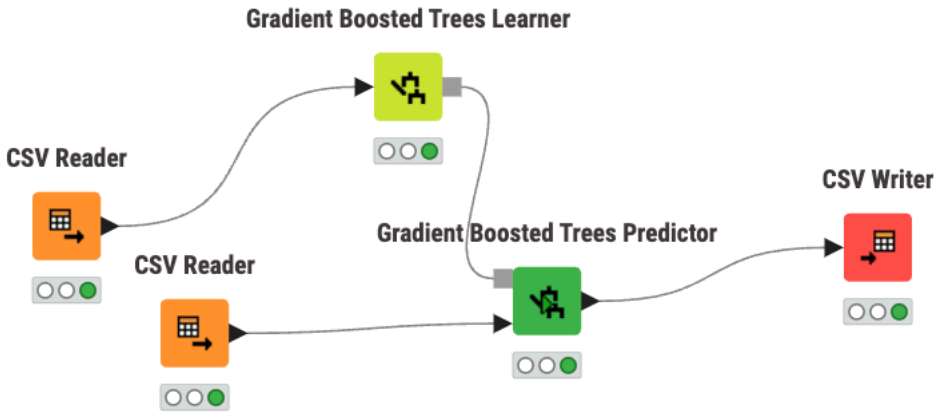


Figure 16 Train the Unknown dataset

I have trained the unknown dataset with the same model and export it to csv file.

Reflection

Assignment 3 provided an invaluable opportunity to dive into data mining and machine learning, offering numerous insights into building and evaluating classification models. This journey expanded my understanding of the technical aspects of data mining and provided some profound self-reflection.

What I Learned About Data Mining:

- Feature Engineering Is Crucial: This assignment reinforced the importance of

feature engineering in data mining. The process of selecting, modifying, and encoding features significantly impacts the performance of classification models. Features that are relevant and well-preprocessed lead to more accurate and meaningful predictions.

- Model Selection Matters: I learned that choosing the right classification model is critical. Different models have distinct strengths and weaknesses, and their performance can vary significantly. The choice of model depends on the dataset's characteristics and the specific problem, as demonstrated by the varied performance of Decision Trees, MLP, Random Forest, SVM, and gradient-boosted trees.
- Hyperparameter Tuning: The importance of hyperparameter tuning became evident throughout the assignment. Fine-tuning model parameters, such as the number of layers, neurons, or pruning methods, can substantially impact model performance. Experimentation and optimisation are crucial steps in the data mining process.
- Handling Data Imbalance: Dealing with class imbalance was another significant learning point. Strategies like resampling, synthetic data generation, and selecting appropriate evaluation metrics play a critical role in addressing the challenges of imbalanced datasets.
- The Power of Ensemble Methods: Ensemble methods, such as Random Forest and gradient-boosted trees, proved their effectiveness in improving model accuracy. Combining multiple models to make predictions showcased their utility, especially in complex classification tasks.

What I Learned About Myself:

- Persistence and Patience: Completing this assignment reinforced my capacity for persistence and patience. The iterative process of experimenting with various models, preprocessing techniques, and parameters required tenacity. It reminded me that solving real-world problems through data mining is often an incremental journey.
- Analytical Thinking: This assignment highlighted the importance of analytical thinking. Analytical skills were vital, from making decisions about feature selection to comparing model performances and parameter tuning. It showed me that approaching complex problems methodically can lead to more informed decisions.
- The Joy of Learning: While challenging, the assignment rekindled my passion for learning and exploration. The thrill of observing model performance improvements and the insights gained along the way were incredibly satisfying. It reminded me of the joy of continuous learning and problem-solving.

Approaching the Problem Differently:

If I were to approach the problem differently, I would:

- Exploratory Data Analysis: Spend more time on exploratory data analysis to better understand the dataset. This could involve visualisations, identifying outliers, and understanding the relationships between features.
- Experiment with More Models: While the assignment covered several models, exploring additional models, such as k-nearest neighbours (K-NN) and Naive Bayes, could provide a more comprehensive understanding of the dataset.
- Advanced Preprocessing Techniques: Investigate advanced preprocessing techniques like dimensionality reduction (e.g., Principal Component Analysis) and advanced feature selection methods. These could potentially improve model performance.
- Ensemble of Ensembles: Consider experimenting with ensemble methods that combine multiple ensemble models (ensemble of ensembles) to leverage their collective strength.
- Automated Hyperparameter Tuning: Utilize automated hyperparameter tuning techniques, such as grid search or Bayesian optimisation, to streamline the parameter tuning process and potentially discover optimal configurations more efficiently.

In conclusion, Assignment 3 enhanced my knowledge of data mining and provided a platform for personal growth and self-discovery. It reinforced the importance of perseverance, analytical thinking, and the joy of continuous learning. The experience serves as a reminder that the data mining journey is dynamic and rewarding, offering opportunities for both technical and personal development.

References

IBM. (n.d.). What is a decision tree. https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes.

IBM. (n.d.). What is Random Forest? https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.

Saini, A. (2023). Gradient boosting algorithm: A complete guide for beginners. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/#:~:text=Gradient%20boosting%20is%20a%20method,has%20produced%20the%20best%20results.

Sharma, S. (2019). What the hell is Perceptron?. Medium. https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

Acknowledgement

Acknowledgment: The authors would like to express their gratitude to OpenAI for providing access to ChatGPT version 3.5, a powerful language model that significantly contributed to the preparation of this paper. ChatGPT 3.5 played a crucial role in various aspects of the article, including answering factual questions by leveraging information available on the internet, aiding in the drafting and structuring of ideas, generating suggestions for graphics and visuals, critically analysing written content for validity, refining grammar and writing structure, experimenting with diverse writing styles, and overcoming writer's block. The collaboration with ChatGPT 3.5 proved instrumental in enhancing the overall quality and efficiency of the research and writing process. The sections of the paper that benefited from AI assistance include the data exploration and preprocessing steps, statistical analysis interpretations, and the generation of concise summaries. The ability of ChatGPT 3.5 to comprehend complex information and provide coherent and contextually relevant responses greatly facilitated the creation of a more robust and insightful paper. The authors acknowledge the AI tool's valuable contributions, recognising it as a supportive and innovative resource in pursuing academic excellence.