Breast Cancer Diagnoses Using Random Forest and SVM Models with Cross-Validation

Prepared by: Nor El Islam Messedad

Section 1: Data Understanding and Preprocessing

Since data understanding and pre-processing steps are integral parts of the data analytics/data science process, we will be dedicating this section to highlighting the most key information relating to our dataset prior to constructing any classification model for the purpose of the project.

To better understand our data, the first step is to assess our data to have a clear grasp of the available data and how it corresponds to the problem to best fit our envisaged model.

1.1. Data Assessment:

According to the project package, the available data consists of two main files both labeled *breast-cancer-wisconsin* whereas one file bears an extension of *.data* (also referred to as "**the data file**" or "**dataset**" in this report) and the other file bears an extension of *.names* (also referred to as "**the names file**" or "**description**" in this report) both accessible via notepad for basic reading, but also via KNIME for data pre-processing and classification purposes.

The data file represents tabular data encompassing columns and rows, whereas the names file is a description of the data file that provides comprehensive background information about the content of the data file to understand the source of the data, the collection process and timeline, and data usage. Hence, as per the names file, we understand that the dataset is a breast cancer database obtained from the University of Wisconsin Hospitals, Madison in the United States from Dr. William H. Wolberg. The data was collected, compiled, and modified between January 1989 and July 1992.

The dataset size, according to the description, has instances of 699 which is the number of records/rows and 11 attributes/columns including the class label. The data range sits within a ballpark of 1-10 across all the existing attributes excluding the same code number column.

More precisely, the attributes of the dataset consist of the following fields:

Table 1. Attributes of the dataset

Attribute	Domain
1. Sample code number	1 - 10
2. Clump Thickness	1 - 10
3. Uniformity of Cell Size	1 - 10
4. Uniformity of Cell Shape	1 - 10
5. Marginal Adhesion	1 - 10
6. Single Epithelial Cell Size	1 - 10
7. Bare Nuclei	1 - 10
8. Bland Chromatin	1 - 10
9. Normal Nucleoli	1 - 10
10. Mitoses	1 - 10
11. Class	2 = benign, 4 = malignant

As shown in Table 1, the description also highlights that the class label is binary where 2 is assigned for benign and 4 for malignant. Hence, we confirm that accessibility to the ground truth is provided in the dataset. This implies that our problem is a binary classification problem to predict the outcome of a patient according to attributes of whether this patient has a (i) benign tumor (non-cancerous) or a (ii) malignant tumor (cancerous). The distribution of the class is 458 records (65.5%) as benign and 241 records (34.5%) as malignant.

Assessing the attributes underlined, we can see that the fields from 2 to 10 are medical-related attributes relevant to the diagnosis process, while in contrast field 1 concerns the sample code number, which does not reflect the medical status of the patient nor provides information impacting the diagnosis. In that sense, the sample code number attribute will be filtered out in the pre-processing phase and before feeding the dataset to our machine-learning model.

In the description, we recognized that missing values exist in the data up to 16 values for instances in Groups 1 to 6 denoted by an exclamation mark "?".

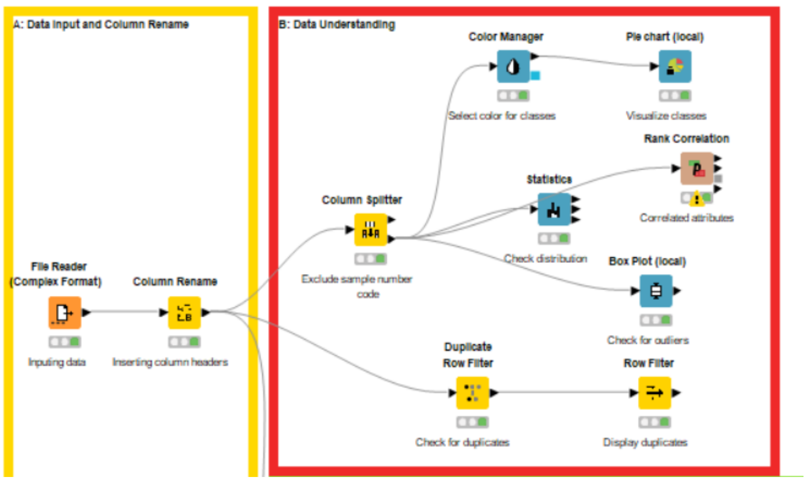
To confirm what has already been identified in the description and extract more insight from the available dataset, we will explore the dataset via KNIME in the Data Exploration subsection:

1.2. Data Exploration:

In order to understand our data, we have inputted the dataset into a ***File Reader (Complex Format)*** in KNIME which provides better visibility over missing values compared to the simple one. The file reader shows the count record encompassing 699 rows and 11 attributes that correspond with the description.

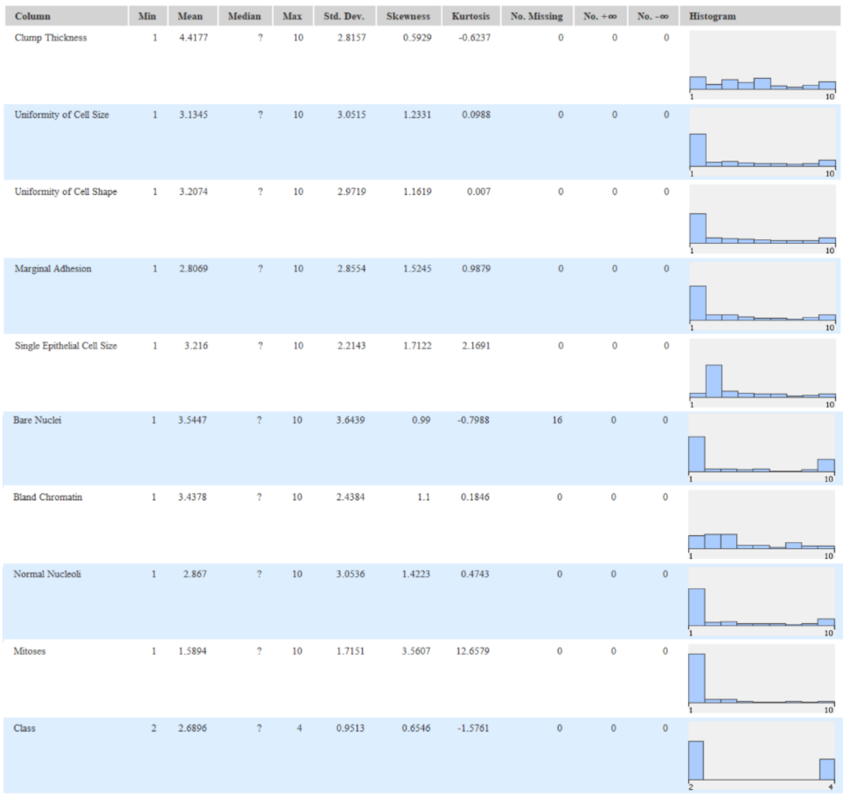
The dataset does not have column headers and contains a sample code number column. To obtain a better perception of data, we are assigning the column headers to the data using the ***Column Rename*** by inputting the column header manually; and filtering out the sample code number using ***Column Splitter*** since it does not represent relevant information as shown in Figure 1.

Figure 1. Data Understanding workflow



By connecting the Column Splitter to the Statistics node, we can obtain an overview of the data size, missing values, and the distribution of each attribute to determine whether the data is balanced and determine the data pre-processing steps onwards.

Figure 2. Statistics



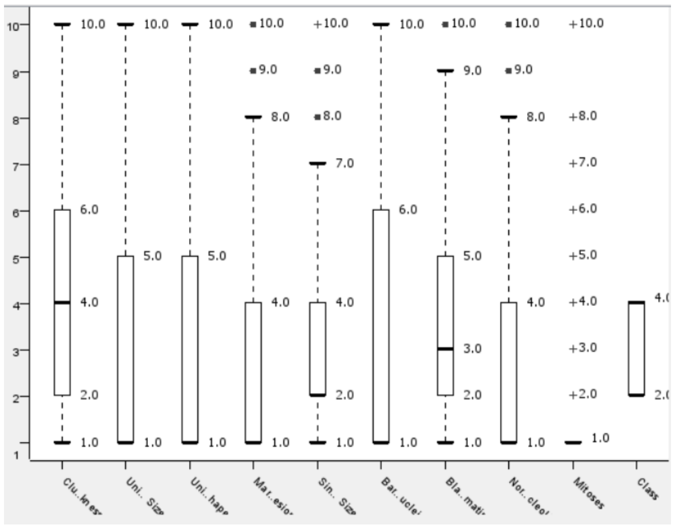
Column	Min	Mean	Median	Max	Std. Dev.	Skewness	Kurtosis	No. Missing	No. +∞	No. -∞	Histogram
Clump Thickness	1	4.4177	?	10	2.8157	0.5929	-0.6237	0	0	0	1 10
Uniformity of Cell Size	1	3.1345	?	10	3.0515	1.2331	0.0988	0	0	0	1 10
Uniformity of Cell Shape	1	3.2074	?	10	2.9719	1.1619	0.007	0	0	0	1 10
Marginal Adhesion	1	2.8069	?	10	2.8554	1.5245	0.9879	0	0	0	1 10
Single Epithelial Cell Size	1	3.216	?	10	2.2143	1.7122	2.1691	0	0	0	1 10
Bare Nuclei	1	3.5447	?	10	3.6439	0.99	-0.7988	16	0	0	1 10
Bland Chromatin	1	3.4378	?	10	2.4384	1.1	0.1846	0	0	0	1 10
Normal Nucleoli	1	2.867	?	10	3.0536	1.4223	0.4743	0	0	0	1 10
Mitoses	1	1.5894	?	10	1.7151	3.5607	12.6579	0	0	0	1 10
Class	2	2.6896	?	4	0.9513	0.6546	-1.5761	0	0	0	2 4

According to the Statistics View option of Figure 2 (right click, View, Numeric tab), we can identify that column **Bare Nuclei** contains 16 missing values which is the total count of missing values in the dataset. The data size is between 1-10. This corresponds with the description of the data.

As apparent in Figure 2, the distribution of most of the attributes is highly skewed, particularly having a right-skew distribution since the skewness in the statistics table is indicating a skewness rate greater than 1. This implies that outliers could lay on the right side of the distribution.

To identify outliers, we use the ***Box Plot (Local)*** node. Upon execution, it indicates the following:

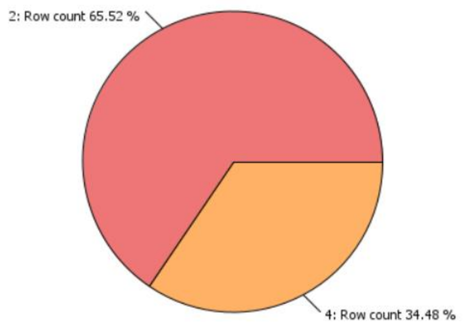
Figure 3. Box Plot graph



We can observe that data points ranging 1.5 above the interquartile range represent outliers in our data concerning Marginal Adhesion, Single Epithelial Cell Size, Bland Chromatin, Normal Nucleoli, and Mitoses. Having this in mind, we should consider either handling the outliers in the pre-processing step or selecting a model that is robust to outliers and asymmetric distribution.

We can establish that the class label between 2 and 4 is unequal. To provide the percentage of the portion of each class, we use the ***Pie Chart*** node connected to ***Color Manager*** to specify the color of each class as shown in Figure 1 and Figure 4:

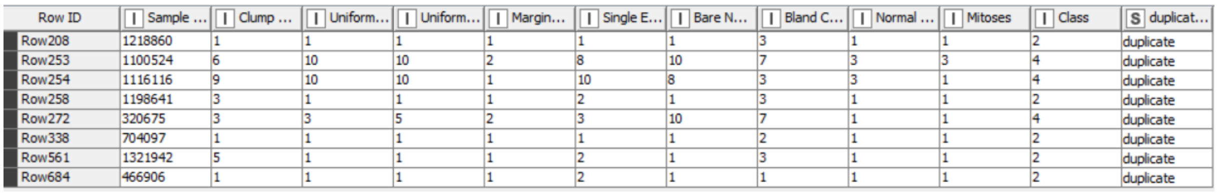
Figure 4. Pie Chart of Label Class



We can recognize that the benign class ("2") is greater than the malignant class ("4") which means that our class label is slightly imbalanced, but not as extreme as to undermine our model. Knowing that imbalances could bias the model, it is worth attempting a test to either oversample the minority class or undersample the majority class during the pre-processing phase.

We can also check for duplicates using the ***Duplicate Row Filter*** node. In the configuration, we select the advance tab and tick "Keep duplicate rows" and "Add column showing duplicates". When we connect the node to the ***Row Filter*** node and filter only the duplicate, we can showcase the number of duplicates consisting of 8 rows as follows:

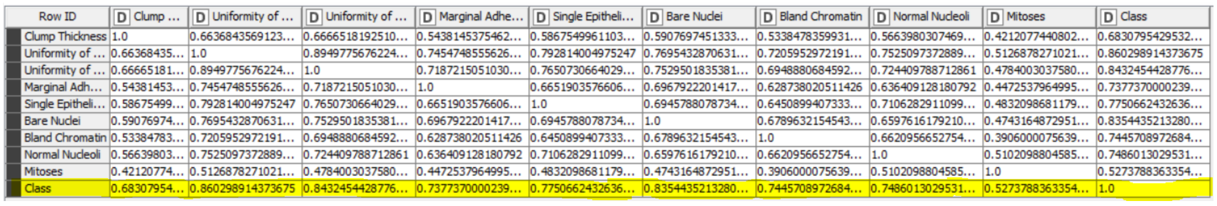
Figure 5. Duplicate rows



Row ID	I Sample ...	I Clump ...	I Uniform...	I Uniform...	I Margin...	I Single E...	I Bare N...	I Bland C...	I Normal ...	I Mitoses	I Class	S duplicat...
Row208	1218860	1	1	1	1	1	1	3	1	1	2	duplicate
Row253	1100524	6	10	10	2	8	10	7	3	3	4	duplicate
Row254	1116116	9	10	10	1	10	8	3	3	1	4	duplicate
Row258	1198641	3	1	1	1	2	1	3	1	1	2	duplicate
Row272	320675	3	3	5	2	3	10	7	1	1	4	duplicate
Row338	704097	1	1	1	1	1	1	2	1	1	2	duplicate
Row561	1321942	5	1	1	1	2	1	3	1	1	2	duplicate
Row684	466906	1	1	1	1	2	1	1	1	1	2	duplicate

The last node is the ***Rank Correlation*** node, which provides us with the correlation rate between the attributes of the dataset:

Figure 5. Correlation matrix



Row ID	D Clump ...	D Uniformity of ...	D Uniformity of ...	D Marginal Adhe...	D Single Epitheli...	D Bare Nuclei	D Bland Chromatin	D Normal Nucleoli	D Mitoses	D Class
Clump Thickness	1.0	0.6636843569123...	0.6666518192510...	0.5438145375462...	0.5867549961103...	0.5907697451333...	0.5338478359931...	0.5663980307469...	0.4212077440802...	0.6830795429532...
Uniformity of ...	0.66368435...	1.0	0.8949775676224...	0.7454748555626...	0.792814004975247	0.7695432870631...	0.7205952972191...	0.7525097372889...	0.5126878271021...	0.860298914373675
Uniformity of ...	0.66665181...	0.8949775676224...	1.0	0.7187215051030...	0.7650730664029...	0.7529501835381...	0.6948880684592...	0.724409788712861	0.4784003037580...	0.8432454428776...
Marginal Adh...	0.54381453...	0.7454748555626...	0.7187215051030...	1.0	0.6651903576606...	0.6967922201417...	0.628738020511426	0.636409128180792	0.4472537964995...	0.7377370000239...
Single Epitheli...	0.58675499...	0.792814004975247	0.7650730664029...	0.6651903576606...	1.0	0.6945788078734...	0.6450899407333...	0.7106282911099...	0.4832098681179...	0.7750662432636...
Bare Nuclei	0.59076974...	0.7695432870631...	0.7529501835381...	0.6967922201417...	0.6945788078734...	1.0	0.6789632154543...	0.6597616179210...	0.4743164872951...	0.8354435213280...
Bland Chromatin	0.53384783...	0.7205952972191...	0.6948880684592...	0.628738020511426	0.6450899407333...	0.6789632154543...	1.0	0.6620956652754...	0.3906000075639...	0.7445708972684...
Normal Nucleoli	0.56639803...	0.7525097372889...	0.724409788712861	0.636409128180792	0.7106282911099...	0.6597616179210...	0.6620956652754...	1.0	0.5102098804585...	0.7486013029531...
Mitoses	0.42120774...	0.5126878271021...	0.4784003037580...	0.4472537964995...	0.4832098681179...	0.4743164872951...	0.3906000075639...	0.5102098804585...	1.0	0.5273788363354...
Class	0.68307954...	0.860298914373675	0.8432454428776...	0.7377370000239...	0.7750662432636...	0.8354435213280...	0.7445708972684...	0.7486013029531...	0.5273788363354...	1.0

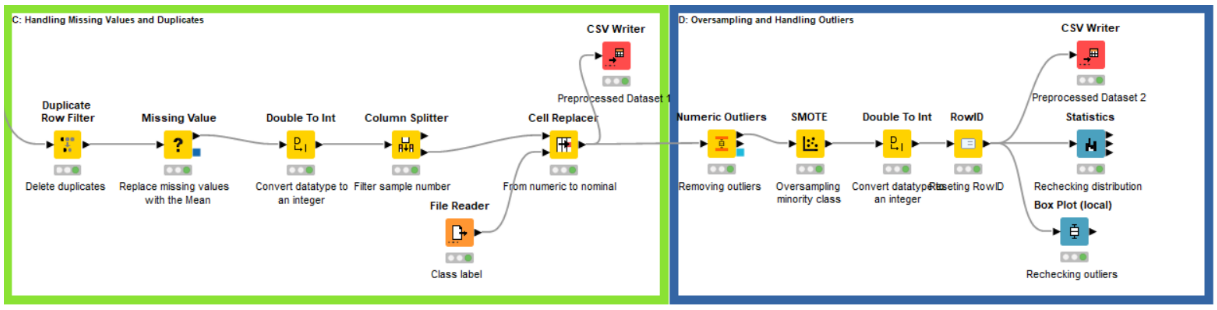
According to the correlation matrix, we notice that the attributes are highly correlated with the class label the highest being Uniformity of Cell Size and the lowest being Mitoses. Given the material correlation, we will not be performing any feature selection in the pre-processing phase as we would like to include as many relevant attributes as possible to train our model.

1.3. Data Preprocessing

In this subsection, we will use the insights obtained throughout the Data Assessment steps to perform data cleansing and pre-processing prior to feeding the dataset to the classification algorithm. As we will be using two algorithms Support Vector Machine (SVM) and Decision Tree (DT). As per the lecture, we establish that DT is not sensitive to outliers, however, SVM will not be able to tolerate a large number of outliers.

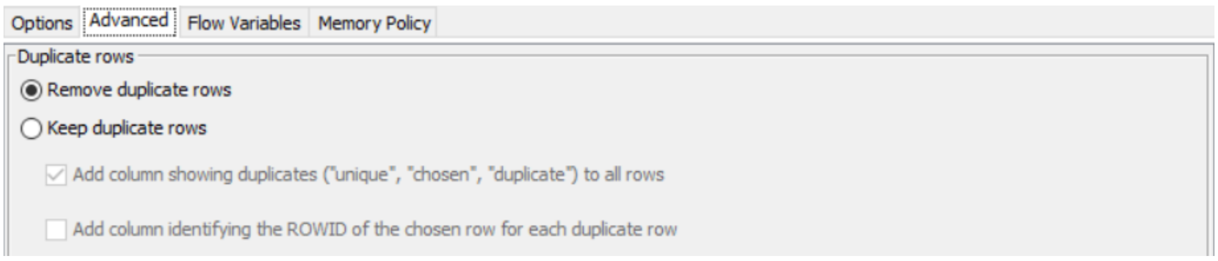
To address this problem, we will output two datasets in which (i) is exclusive of outliers and imbalances and (ii) is inclusive of outliers and retain the original proportion of the class labels. To do so, we construct the following workflow in KNIME:

Figure 6. Data pre-processing workflow



We need to remove the duplicates to ensure a better split of the train and test sets. Again, we use the ***Duplicate Row Filter*** this time to remove the duplicates by ticking the "Remove duplicate rows". As we observe 8 duplicates, this reduces the data size from 699 to 691 records.

Figure 7. Duplicate removal configuration



Duplicate rows

[✓ Remove duplicate rows]

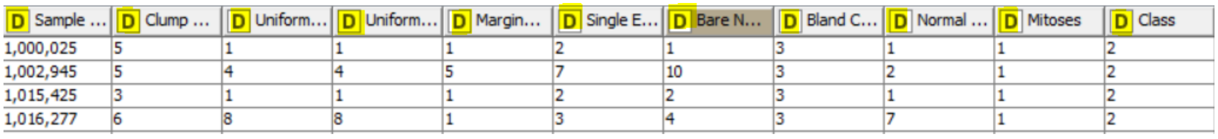
Keep duplicate rows

☑ Add column showing duplicates ("unique", "chosen", "duplicate") to all rows

☐ Add column identifying the ROWID of the chosen row for each duplicate row

Since the missing values only represent 16 records in **Bare Nuclei** among 691 records**,** we will not delete the column but rather replace the missing values with the Mean of the attribute by employing the ***Missing Value*** node and selecting the Mean in the configuration. This will render our dataset complete to avoid biased results and support the machine learning algorithms in the classification step.

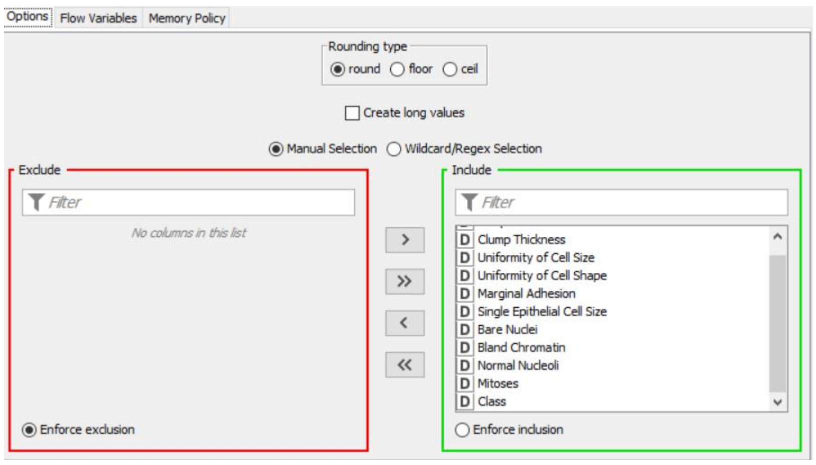
Figure 8. Datatype after handling missing values



D Sample ...	D Clump ...	D Uniform...	D Uniform...	D Margin...	D Single E...	D Bare N...	D Bland C...	D Normal ...	D Mitoses	D Class
1,000,025	5	1	1	1	2	1	3	1	1	2
1,002,945	5	4	4	5	7	10	3	2	1	2
1,015,425	3	1	1	1	2	2	3	1	1	2
1,016,277	6	8	8	1	3	4	3	7	1	2

After executing the node, the datatype of the attributes shifts to double since the Mean has a decimal value (3.538). As we need the attributes to be of an integer type to fit the machine learning algorithm, we convert the datatype by connecting the ***Missing Value*** node to the ***Double to Int*** node by ticking "round" in the rounding type box as shown below:

Figure 10. Conversion from double to integer configuration



Rounding type

[✓ round] floor ceil

☐ Create long values

[✓ Manual Selection] Wildcard/Regex Selection

Exclude

No columns in this list

[✓ Enforce exclusion]

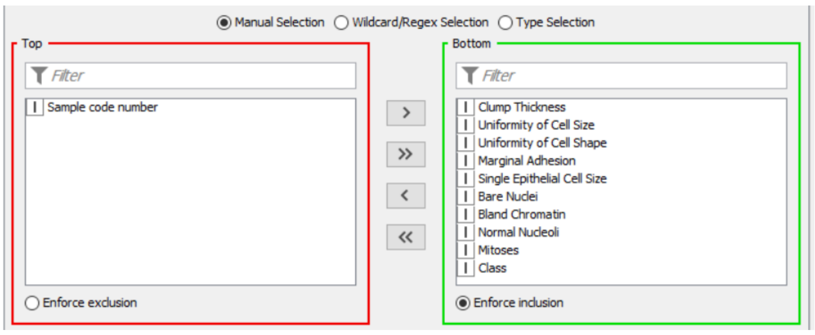
Include

D Clump Thickness
D Uniformity of Cell Size
D Uniformity of Cell Shape
D Marginal Adhesion
D Single Epithelial Cell Size
D Bare Nuclei
D Bland Chromatin
D Normal Nucleoli
D Mitoses
D Class

Enforce inclusion

Afterward, we also need to exclude the **sample code number** attribute from the dataset using the ***Column Splitter*** node given that this attribute does not determine/impact the class label.

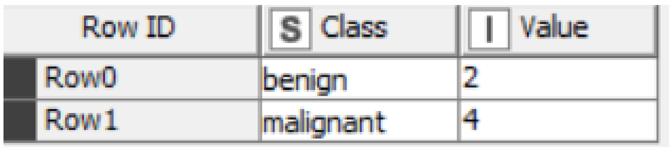
Figure 11. Filtering the sample code number


Manual Selection [✓] | Wildcard/Regex Selection | Type Selection

Top	Bottom
Sample code number	Clump Thickness
	Uniformity of Cell Size
	Uniformity of Cell Shape
	Marginal Adhesion
	Single Epithelial Cell Size
	Bare Nuclei
	Bland Chromatin
	Normal Nucleoli
	Mitoses
	Class
Enforce exclusion	[✓ Enforce inclusion]

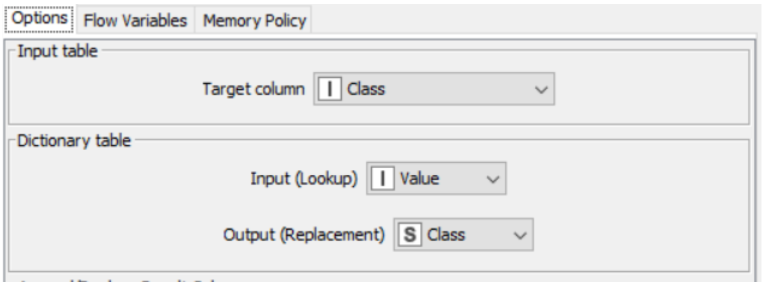
Following the previous step and understanding that the class label is numeric, KNIME does not execute with nominal class labels when using the SVM learner and DT learners. This requires converting the class label from numeric to nominal values (benign, and malignant). In doing so, we employ the ***Call Replacer*** node as shown in Figure 6 which we use to input a manually created CSV file labeled "category label" through ***File Reader*** entailing the Value (2, 4) and the Class (benign, malignant). Executing the node will result in replacing the numeric values with nominal values.

Figure 12. inputted category label



Row ID	S Class	I Value
Row0	benign	2
Row1	malignant	4

Figure 13. Configuration of Cell Replacer

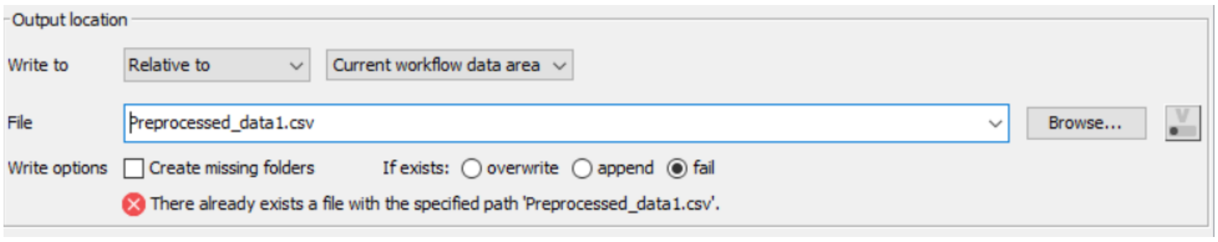

Options | Flow Variables | Memory Policy

Input table
Target column: I Class

Dictionary table
Input (Lookup): I Value
Output (Replacement): S Class

At this stage, we want to export the first CSV file with the current parameters which will be our output dataset used for the Decision Tree model only by skipping the outlier removal and class label balance steps. We drag the ***CSV Writer*** node and save the file in CSV format in the directory labeled as "Preprocessed_data1".

Figure 14. Output location of preprocessed data 1



Output location

Write to: Relative to — Current workflow data area

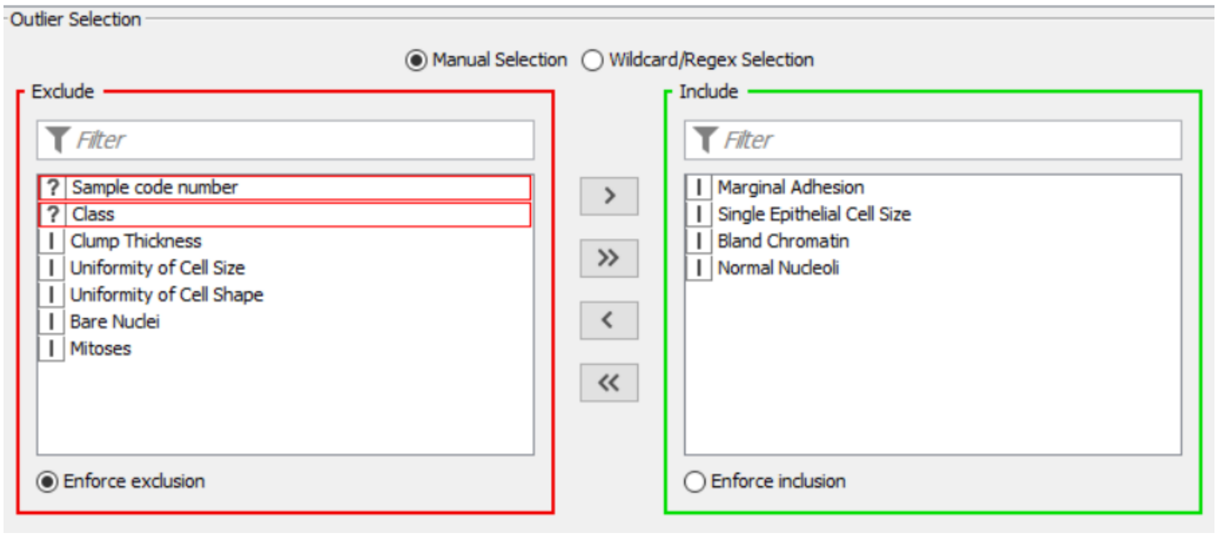
File: Preprocessed_data1.csv

Write options: [ ] Create missing folders — If exists: ( ) overwrite ( ) append [✓ fail]

There already exists a file with the specified path 'Preprocessed_data1.csv'.

For the second dataset fed to the SVM model, we will conduct outlier removal and oversampling of minority class by using the ***Numeric Outliers*** node for outliers and the ***SMOTE*** node for oversampling.

Figure 15. Outlier Removal using Numeric Outliers



Outlier Selection

[✓ Manual Selection] ( ) Wildcard/Regex Selection

Exclude:
- ? Sample code number
- ? Class
- I Clump Thickness
- I Uniformity of Cell Size
- I Uniformity of Cell Shape
- I Bare Nuclei
- I Mitoses

[✓ Enforce exclusion]

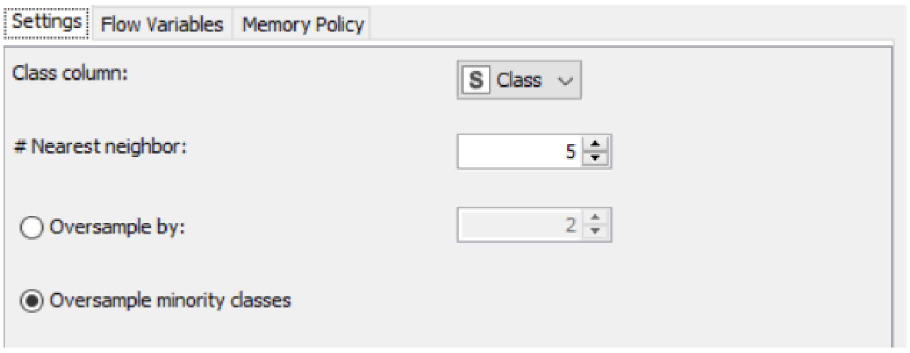
Include:
- I Marginal Adhesion
- I Single Epithelial Cell Size
- I Bland Chromatin
- I Normal Nucleoli

( ) Enforce inclusion

We only include the attributes that recognize outliers 1.5 above the IQR as demonstration through the ***Box Plot*** previously ***(***Marginal Adhesion, Single Epithelial Cell Size, Bland Chromatin, Normal Nucleoli, and Mitoses). However, we exclude "Mitoses" providing that the data in this attribute is centralized in one data point (e.i., value of 1) at which point the ***Numeric Outliers*** would remove any opposite variations to that value that is not necessarily an outlier [1]. Executing the node with the configuration in Figure 14 reduces the data size from 691 to 547 rows as it deletes the rows containing an outlier making a total of 144 rows of outlier data points.

Subsequently, we use the Synthetic Minority over-sampling technique (***SMOTE***) node using Nearest neighbour configuration for the class column [2]. Oversampling the minority class of malignant will proportionate to benign as to balance the class label. Balancing the class label will enhance the model to learn from the dataset to produce better results especially knowing that we are interested in the malignant class as it represents the cancerous outcome (positive diagnosis). On the other hand, balancing the data will help fitting the dataset into the SVM algorithm. This will also circumvent biasing the model to the majority class. The new class label records accounts for 447 for each class and 894 in total.

Figure 16. Configuration of SMOTE



Class column: S Class

Nearest neighbor: 5

○ Oversample by: 2

● Oversample minority classes

Afer oversampling, the new samples will be in decimal point values and different Row ID. To remedy this, we use again the ***Double to Int*** node to round the values, and the ***RowID*** node to reset row ID numbers to default which renders the data cleaner for the model.

Using ***Statistics*** and ***Box Plot*** nodes, we can recheck for skewness, distribution of the attributes and outliers as well as the balance of the class labels. Our data post-processing is less skewed and exclusive of outliers with a balanced class label.

Similar to preprocessed data 1, we drag the ***CSV Writer*** node and save the file in CSV format in the directory labeled as "Preprocessed_data2".

Section 2: Classification

In this task, we will be using two models featuring two preprocessed datasets. The first preprocessed data will be dedicated to the DT model whereas the second preprocessed data will be fed to the SVM model as follows:

2.1 Random Forest Classification Model:

As simple introduction to Random Forest, it is a supervised learning algorithm which is an expansion of the Decision Tree as it aggregates several decision trees used for both classification and regression problems where it mimics the human-level thinking by splitting according to the features and attributes of the dataset to classify records. One of the features of the Random Forest is that it does not require substantial data preparation which is the reason for which we are using the dataset inclusive of outliers and without oversampling. In addition, it provides better accuracy than the decision trees in general.

We drag the ***File Reader*** to import preprocessed_data1, then assign colors to the label cases via the ***Color Manager.*** To implement our model, we will be using the cross-valuation method of leave-one-out through the ***X-Partitioner*** node.

Figure 17. Random forest model

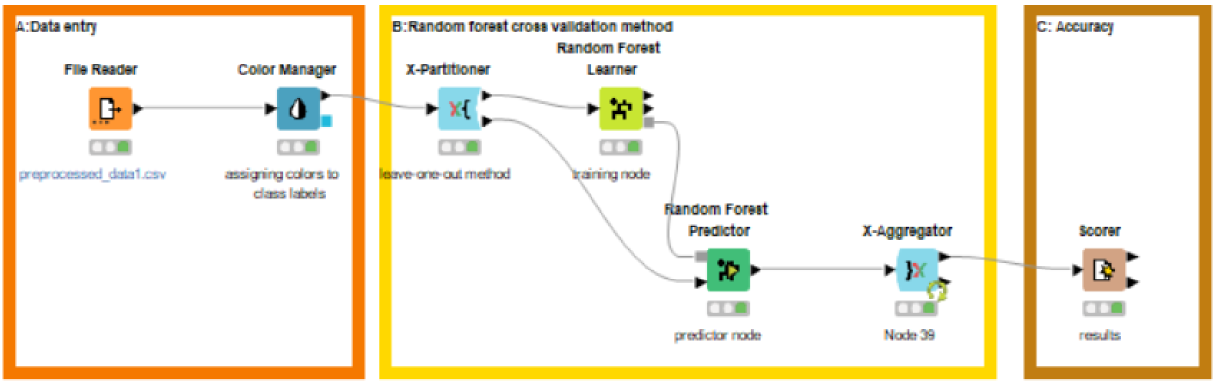
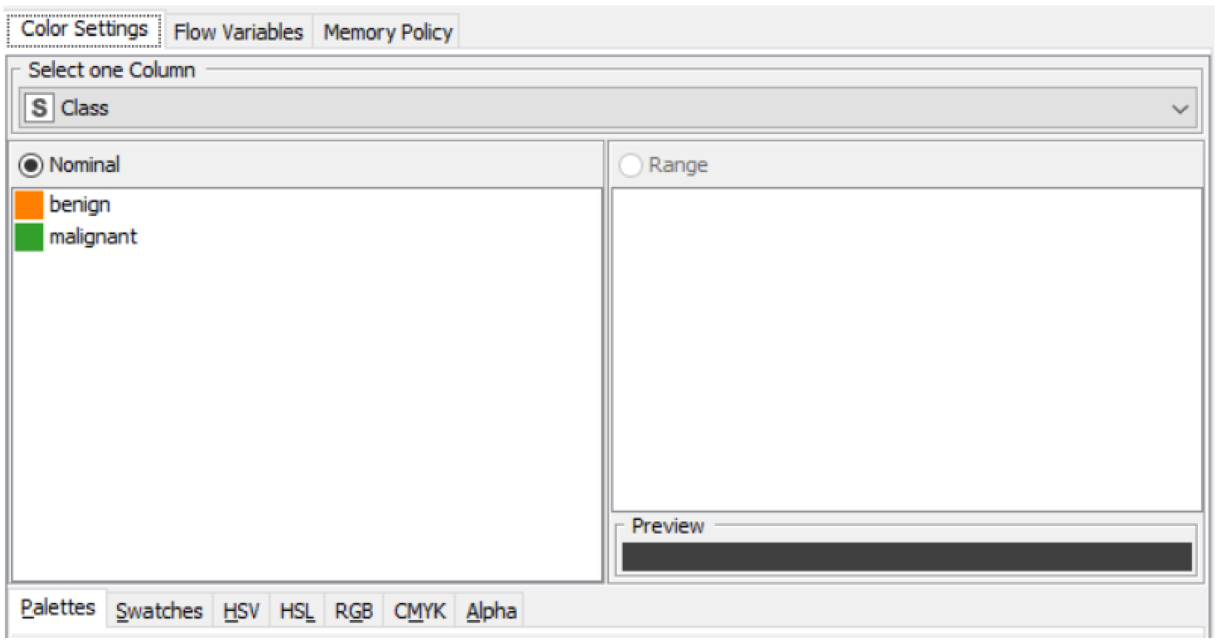


Figure 18. Color Manager configuration


Color Settings | Flow Variables | Memory Policy

Select one Column
S Class

(● Nominal) ○ Range

🟧 benign
🟩 malignant

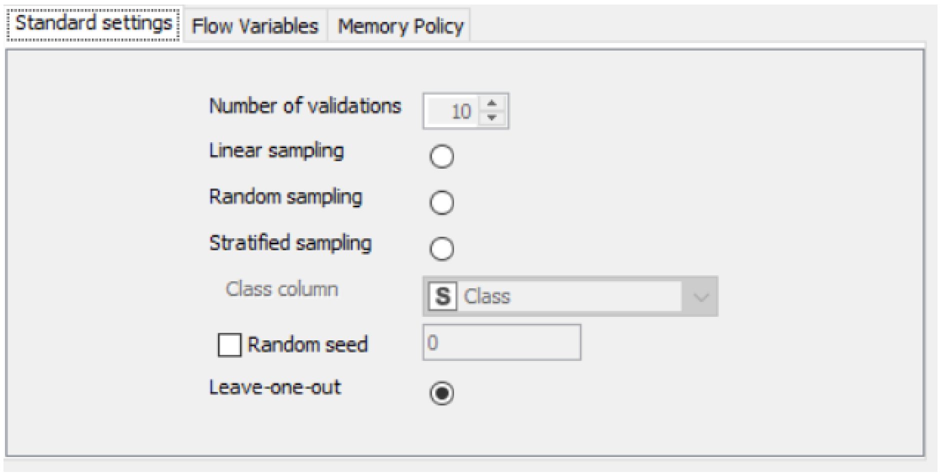
Preview

Palettes | Swatches | HSV | HSL | RGB | CMYK | Alpha

Using leave-one-out implies that for each iteration the node holds one record for evaluation and the rest for testing by preforming iterations equating to the number of records in the data set, in order words, 691 iterations. For this reason, the leave-one-out cross validation method provides reliable outcomes for which it was selected for this model.

This is configured through the ***X-Partitioner*** by ticking the leave-one-out option as follows:

Figure 19. X-Partitioner leave-one-out configuration

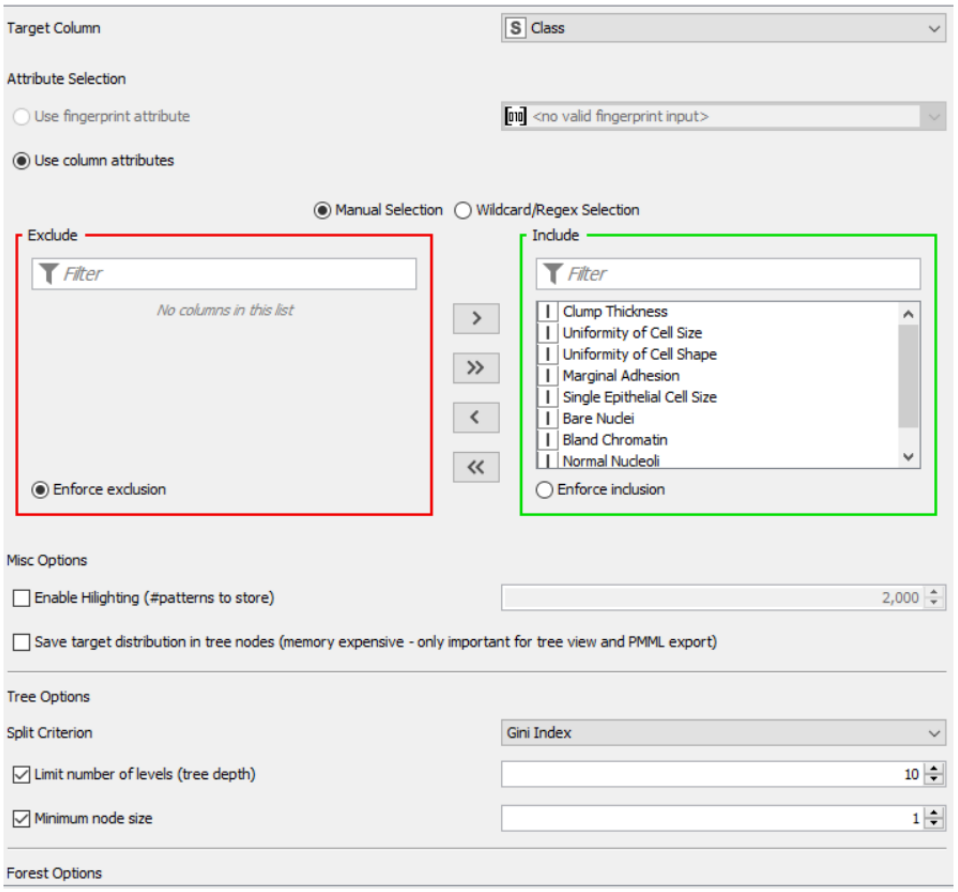

Standard settings | Flow Variables | Memory Policy

Number of validations: 10
Linear sampling ○
Random sampling ○
Stratified sampling ○
Class column: S Class
☐ Random seed: 0
Leave-one-out ●

Through the node, the upper port connects to ***Random Forest Learner*** node, which represents the training data, whereas the lower port connects to the ***Random Forest Predictor*** node, which is the test data.

In the configuration of the Random Forest Learner, we select the splitting parameter as "Gini Index", while also ticking the box to limit the number of tree depth to 10. This will help the model to not overfit as it runs and, hence, generalize well for unseen data.

Figure 20. Configuration of Random Forest Learner


Target Column: S Class

Attribute Selection
○ Use fingerprint attribute: <no valid fingerprint input>
● Use column attributes

● Manual Selection ○ Wildcard/Regex Selection

Exclude
Filter
No columns in this list
● Enforce exclusion

Include
Filter
Clump Thickness
Uniformity of Cell Size
Uniformity of Cell Shape
Marginal Adhesion
Single Epithelial Cell Size
Bare Nuclei
Bland Chromatin
Normal Nucleoli
○ Enforce inclusion

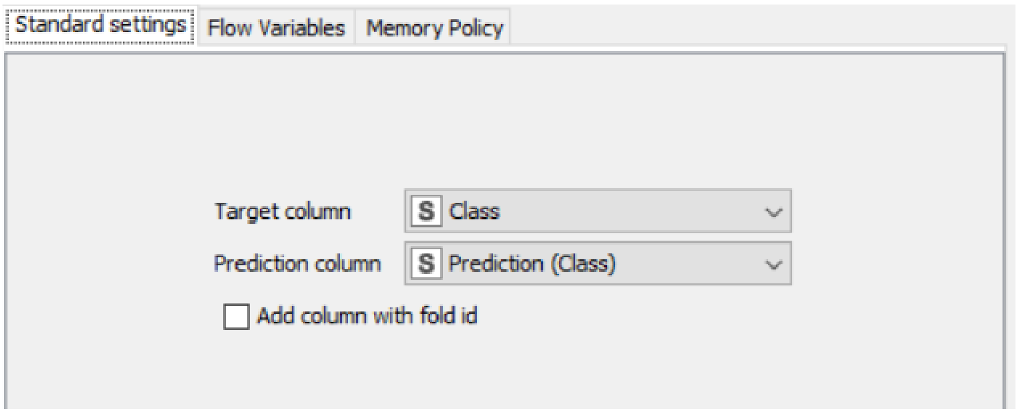
Misc Options
☐ Enable Hilighting (#patterns to store): 2,000
☐ Save target distribution in tree nodes (memory expensive - only important for tree view and PMML export)

Tree Options
Split Criterion: Gini Index
☑ Limit number of levels (tree depth): 10
☑ Minimum node size: 1

Forest Options

We can also preform feature selection through including or excluding attributes from the model through the manual selection, however, for the purpose of this model, we will include all of the attributes to predict the class. As for the ***Random Forest Predictor*** and ***X-Aggregator*** nodes, we keep the configuration set to default.

Figure 21. X-Aggregator configuration



Target column: S Class
Prediction column: S Prediction (Class)
☐ Add column with fold id

In the ***X-Aggregator,*** we set the Target Column to be the Class. ***X-Aggregator*** can also provide the Prediction Table (class and prediction) and Error Rates per each prediction. Running the model takes computation time to preform due to the iteration of the leave-one-out method.

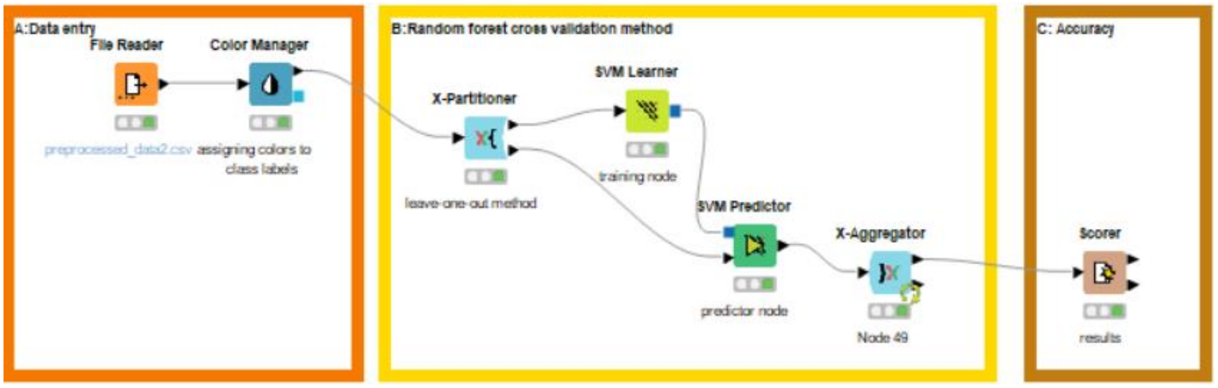
Connected to the ***Scorer*** node, we can obtain insight by extracting the confusion matrix and accuracy rates of the model performance.

2.2 Support Vector Machine Classification Model:

The Support Vector Machine (SVM) is also a supervised learning algorithm used for binary classification problems but also for regressions. SVM classifies records by identifying the hyper-plane which is a straight line that separates the classes linearly. SVM work well with small datasets and provides good accuracy metrics for binary classification which fits the properties of our dataset. Although SVM requires data preparation given its relative sensitivity to outliers, we were able to preform data balancing and outlier removal to fit the SVM model.

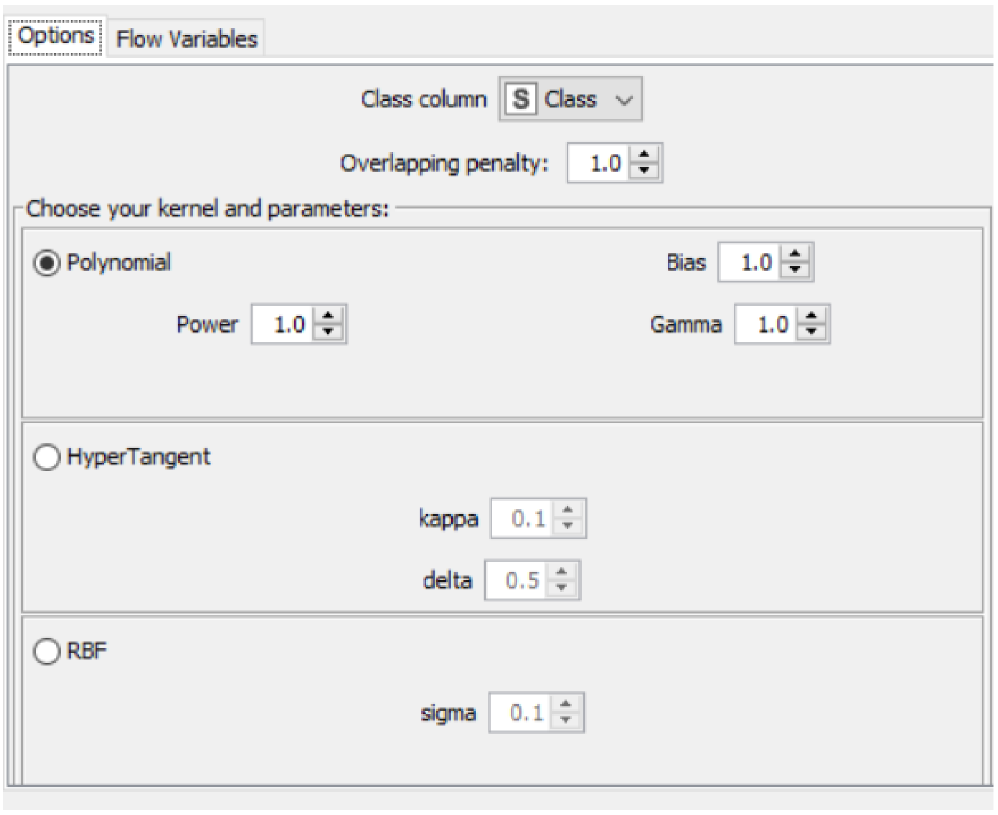
Given the reliability of the leave-one-out cross validation method, we will use the same method that of the previous model with a different pre-processed data that fits the model. Following the same steps in the previous model, we drag the ***File Reader*** to import preprocessed_data2 (i.e., the data set after balancing and outlier removal), then assign colors to the label cases via the ***Color Manager.*** Similarly, to implement our model, we will be implementing the ***X-Partitioner*** node.

Figure 22. SVM model



We keep the configuration of the X-Partitioner the same (by retaining the leave-one-out option). The only two nodes that change in comparison to the previous model is the ***SVM Learner*** and the ***SVM Predictor.*** In the configuration of the SVM Learner, we utilize the following:

Figure 23. Configuration of SVM Learner



Options | Flow Variables

Class column: S Class

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

The target column is the class attribute, whereas the Overlapping penalty is set to 1.0 which is the default setting. This determines the extent to which an overlapping penalty is allocated to misclassified records. Similarly, kernel and parameter options remain within a default setting to find the separating hyperplane and derive reliable outcomes.

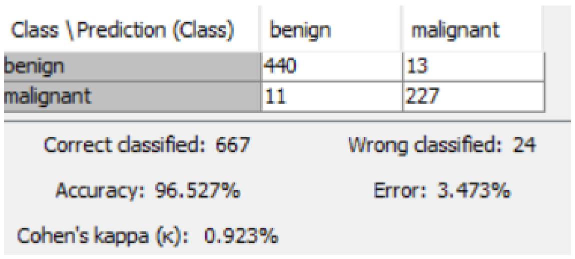
Like the previous model, running the workflow takes computation time to preform due to the iteration of the leave-one-out method.

Connected to the ***Scorer*** node, we can obtain insight by extracting the confusion matrix and accuracy rates of the model performance that will be discussed in the Model Evaluation section.

Section 3: Model Evaluation

Through this section, we will evaluate the accuracy of the Random Forest model and the SVM model using relevant performance metrics. By running the ***Scorer*** node, we output the following confusion matrices for both models:

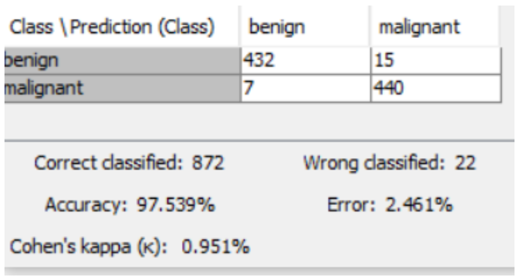
Figure 24. Radom Forest confusion matrix



Class \ Prediction (Class)	benign	malignant
benign	440	13
malignant	11	227

Correct classified: 667 Wrong classified: 24
Accuracy: 96.527% Error: 3.473%
Cohen's kappa (κ): 0.923%

Figure 25. SVM confusion matrix



Class \ Prediction (Class)	benign	malignant
benign	432	15
malignant	7	440

Correct classified: 872 Wrong classified: 22
Accuracy: 97.539% Error: 2.461%
Cohen's kappa (κ): 0.951%

Measuring the accuracy as a standalone metric, it is apparent that the SVM model yields better accuracy reaching **97.539%** verses **96.527%** for Random Forest model. Since our pre-processed data differs in size, we cannot compare the number of FP, FN and TP, TN. Also, the accuracy metric only accounts for the total number of correct predictions and incorrect prediction for both positive and negative classes which does not serve the actual need for performance purposes in this case.

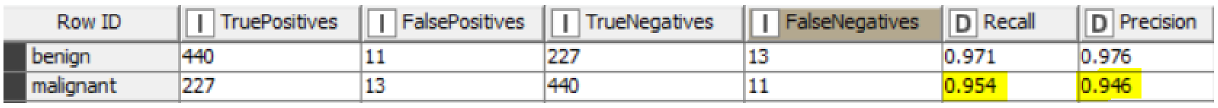
As the most undesirable outcome in cancer diagnostic is the False Negative – the patient has been diagnosed Negative whereas the correct diagnosis is Positive – this results in the patient not taking the necessary treatment at the appropriate time, which undermines life.

The other undesirable outcome is the False Positive – the patient has been diagnosed Positive whereas the correct diagnosis is Negative – this results in the patient undertaking costly unnecessary treatment while the diagnosis is non-cancerous.

Hence, a more reliable performance metric to use in this case is the recall and precision metrics calculated on malignant classification which measure the extent of the error caused by total False Negatives (recall) as well as False Negatives (precision) as standalone metric to assess the proportion of correctly predicted positive class across all of the true positive class and the predicted positives among all real positive cases [3].

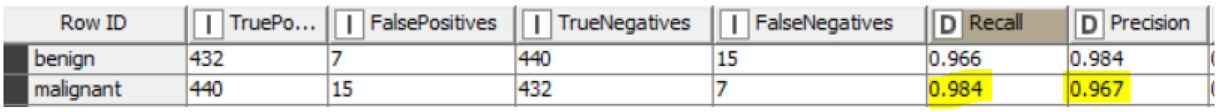
We can extract this information from the ***Scorer*** node by selecting "Accuracy Statistics" for both models as follows:

Figure 26. Precision and recall metrics for Random Forest model



Row ID	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision
benign	440	11	227	13	0.971	0.976
malignant	227	13	440	11	0.954	0.946

Figure 27. Precision and recall metrics for SVM model



Row ID	TruePo...	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision
benign	432	7	440	15	0.966	0.984
malignant	440	15	432	7	0.984	0.967

Interpreting the precision and recall metrics on the malignant class, we observe that the SVM model preforms slightly better than the Random Forest model by **98.4%** vs **97.1%** for recall**,** which means SVM has less False Negative rate. For precision, likewise, SVM obtains higher performance by achieving **96.7%** vs **94.6%** with Random Forest.

For benign, recall is slightly better on Random Forest with **97.1%** vs **96.6%** but not the case for Precision where SVM outperforms with **98.4%** vs **97.6%.**

Conclusion:

As per the above, we deduce that SVM model outperforms Random Forest by setting Recall and Precision as our performance metrics. Throughout the process it was vital to undertake data understanding and pre-processing steps to fit the dataset to the two models in varying pre-processing requirements.

Preforming a mix of tailored-based preprocessing techniques and hyperparameters, such as the leave-one-out method yields elevated outcomes for our model. For future testing, we can also use a validation set (unseen data) to perform more robust evaluation on the model in terms of overfitting testing and generalization ability.

References:

[1] KNIME Hub, “Numeric Outliers”, [Online]. Available: https://hub.knime.com/knime/extensions/org.knime.features.stats/latest/org.knime.base.node.stats.outlier.handler.NumericOutliersNodeFactory. [Accessed: 16- Nov- 2022].

[2] KNIME Hub, “SMOTE (Synthetic Minority Over-sampling Technique”, [Online]. Available: https://hub.knime.com/knime/extensions/org.knime.features.base/latest/org.knime.base.node.mine.smote.SmoteNodeFactory. [Accessed: 17- Nov- 2022].

[3] F. Shamout, T. Zhu and D. A. Clifton, "Machine Learning for Clinical Outcome Prediction," in IEEE Reviews in Biomedical Engineering, vol. 14, pp. 116-126, 2021, doi: 10.1109/RBME.2020.3007816.