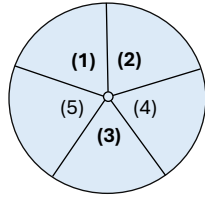


## Part One: Classification, Regression, and Clustering



For the following problems, we will consider the challenges and complexities of real-world big data problems through the “3V/5V’s In Big Data”: **(1) Volume**; size, **(2) Variety**; types, forms, sources, **(3) Velocity**; speed of data generation, movement, and analysis requirements, **(4) Value**; potential usefulness, and **(5) Veracity**; trustworthiness and availability (Nguyen, 2025d).

### (i) Real-world Big Data Classification Example

One example of a real-world big data classification problem is automated disease diagnosis from multi-modal medical data. Data generated by hospitals for a single patient are a point-in-time reference and can include a range of multi-modal sources, such as human fluid or waste test results (saliva, blood, urine, etc.), Electrocardiogram (ECG) results, Magnetic Resonance Imaging (MRI) scans, Electronic Health Records (EHRs), and so on. One potential use for this data is to automate disease diagnosis to enhance professional diagnostic services – for example, by classifying patients into disease categories or predicting their likelihood of developing a specific condition – and to provide model results to a licensed GP who is responsible for making the patient’s final diagnosis. This can often be a challenging task for many reasons, such as large or voluminous data (e.g., images, videos, genetic sequences, etc.), missing data, disjointed or distributed patient records, class imbalances for rare diseases, and privacy, ethical, or cultural considerations surrounding data availability. There are further complexities introduced by multi-modal data fusion tasks and real-time decision-making for earlier diagnosis and treatment. Real examples like Google’s DeepMind AI for Diabetic Retinopathy Screening and IBM Watson for Oncology serve as ‘real-world’ ‘big’ multi-modal model examples, while simplified datasets like the Cleveland database from the UCI Machine Learning Repository can be used to help illustrate automated disease diagnosis using heterogeneous data in lower-dimensional space (Digital, 2025; Farahat et al., 2024; Janosi et al., 1989; Liu et al., 2018). In so far as the 5V’s of Big Data, we refer to the following layers of complexity:

Problem Type	Example	Description	1-Volume	2-Variety	3-Velocity	4-Value	5-Veracity
Classification	Disease Diagnosis from Multi-Modal Medical Data	High-dimensional, multi-modal data fusion	Terabytes (1000 <sup>4</sup> ) (hospitals, genomics, medical imaging, etc.)	ECGs, MRI scans, genomic data, urinalysis, blood tests, EHRs, clinical notes, etc.	Periodic data updates (per patient visit); near real-time analysis needed for emergency cases	Extremely High: Early diagnosis saves lives and informs treatments	Low: Privacy and ethical concerns, missing or distributed records, unstructured clinical notes, etc.

### (ii) Real-world Big Data Regression Example

One example of a real-world big data regression problem is weather forecasting. While some tasks involving weather data may involve classification or clustering, such as categorising weather patterns (e.g., cyclone or anticyclone) and grouping weather conditions based on similarity (without labels), generic weather prediction is primarily concerned with estimating continuous meteorological variables and hence serves as an example of a complex **spatiotemporal regression** problem requiring big data. Forecasting refers to predicting continuous values (e.g., temperature, wind speed, moisture) at future time steps based on past and current observations. As weather forecasting is a multi-variate problem, the model must be able to predict multiple continuous variables at different times and spatial locations. More specifically, all forecasts are examples of initial-value problems in which the solution is to obtain the best estimate for 3D multi-variate conditions on a

grid given time to initialise (Benjamin et al., 2019). Weather problems are traditionally solved using Numerical Weather Prediction (NWP) techniques represented as a series of complex minimisation and data assimilation problems, or Machine Learning Weather Prediction (MLWP), where models are trained to learn from historical weather data to produce or improve weather predictions (Bonavita, 2024; Lam et al., 2023). Examples of datasets used in weather predictions include satellite imagery, radar data, atmospheric pressure measurements, temperature and humidity readings from ground stations, ocean buoy observations, wind speed and direction data, climate reanalysis datasets (e.g., ERA5, MERRA-2), numerical weather model outputs (e.g., ECMWF, GFS), and real-time data from weather balloons and remote sensing instruments (Modeling & (GMAO), 2015; Owens & Hewson, 2018; Service, 2023). In so far as the 5V's of Big Data, we refer to the following layers of complexity:

Problem Type	Example	Description	1-Volume	2-Variety	3-Velocity	4-Value	5-Veracity
Regression	Weather Forecasting	Multi-variate, continuous spatiotemporal predictions	Petabytes (1000 <sup>5</sup> ) (global climate models)	Satellite imagery, IoT sensor data, atmospheric simulations	Real-time updates, forecasts needed every few hours - minutes	High: impacts agriculture, aviation, disaster preparedness, etc.	Moderate: Open Data sources; Data gaps, sensor malfunctions, numerical model biases

### (iii) Real-world Big Data Clustering Example

One example of a real-world big data clustering problem is anomaly detection in cybersecurity networks. Many advanced cybersecurity systems are designed to monitor large streams of network traffic logs and detect unknown attack patterns using machine learning, including unsupervised clustering techniques. One such example would be an ELK Stack (Elasticsearch, Logstash, Kibana) overlaid with unsupervised clustering algorithms to group similar network events or signal anomalies that deviate from learned clusters (Amazon Web Services, 2025). This is an example of a real-world big data clustering problem because of the high volumes and velocities of network data, as well as the contextual requirement for real-time analysis to group similar traffic patterns and identify anomalies. Further, data fusion can present challenges in this setting, particularly if there is multi-modal cross-referencing, such as matching firewall logs against user behaviour analytics and device logs. Like weather forecasting, the system must process spatiotemporal data at scale, and (arguably) with greater urgency. However, unlike classification, ground labels are often missing, which can make evaluation more challenging. In so far as the 5V's of Big Data, we refer to the following layers of complexity:

Problem Type	Example	Description	1-Volume	2-Variety	3-Velocity	4-Value	5-Veracity
Clustering	Threat Detection	No ground truth, evolving attack patterns, streaming data	Petabytes (1000 <sup>5</sup> ) per day (large networks)	Firewall logs, API calls, user behaviour analytics, packet traces	Real-time threat detection requires near-instantaneous processing	Very High: Ensures the safety of individuals and assets, etc.	Moderate-to-High: Noise and spoof attacks can occur, as well as adversarial evasion

## Part Two: Feature Selection and Construction Methods

**(i) Feature Ranking/Weighting:** A filter method that assigns a score (weight) to each feature based on its relevance to a target variable; it is an approach best suited to situations with high-dimensional data where interpretability is essential. Feature ranking weights features individually (i.e., without considering their interactions). It is often quantified via Entropy, Mutual Information (MI), or Information Gain (IG), where rankings favour features that yield higher class purities (Cover & Thomas, 2005; Nguyen, 2025b). Another effective method to assess feature and target variable linear relationships is the Pearson Correlation Coefficient (PCC). Through PCC, positive and negative (inverse) correlative relationships are represented as values ranging from  $[-1, 1]$  (ii below has more details). Weights also help reduce noisy contributions from irrelevant and redundant features. However, redundancy elimination is more directly addressed using subset selection methods. Of the two feature manipulation methods discussed (ranking vs subset selection), feature ranking is lower complexity, and therefore less costly.

**Feature Subset Selection:** Selects a subset of features (i.e., a group of features jointly) which optimise some criterion. This approach considers feature interactions when selecting a subset by evaluating their classifier performance. Feature subset selection can be achieved through wrapper methods, such as various sequential search techniques, including (1) Sequential Forward Selection (SFS) – Heuristic, greedy search, (2) Sequential Backwards Selection (SBS) – Heuristic search, (3) Bidirectional Search (BDS) – SFS + SBS simultaneously, (4) Plus-L, Minus-R Selection (LRS) – Generalisation of SFS/SBS, and (5) Sequential Floating Forward/Backward Selection (SFFS/SFBS) – An extension of LRS (Kohavi & John, 1997; Nguyen, 2025b). Feature subset selection is most commonly achieved using wrapper or embedded methods and is best for improving model generalisation or reducing overfitting. It is also possible to have a filter-based subset selector, like CFS.

**(ii) Filter methods** like feature ranking are used to evaluate the relevance of features based on statistical measures without the use of a specific predictive model. Filtering is computationally efficient and works well with high-dimensional data while retaining interpretability. Examples include (1) Mutual Information (MI) and the (2) Pearson Correlation Coefficient (PCC):

$$(1) MI = I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = \sum_{x \in X, y \in Y} p(x, y) \log_2 \left( \frac{p(x, y)}{p(x)p(y)} \right)$$

$$(2) PCC = r_{xy} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}$$

The MI of two random variables is a measure of the mutual dependence between the two variables (Nguyen, 2025c). That is, MI is a general measure for (linear or non-linear) relationships and useful for when your data has both continuous and categorical variables. The PCC measures the linear correlation between a feature  $X$  and the target  $Y$ , where  $1 \rightarrow$  Perfect positive relationship (positive association),  $-1 \rightarrow$  Perfect inverse relationship (negative association),  $0 \rightarrow$  No association between two variables (correlation). PCC is best suited for continuous data where linear relationships are likely.

**Wrapper methods** use a predictive model to evaluate feature subsets iteratively. Wrappers typically use search algorithms like SFS, SFBS, etc., and are useful for feature selection approaches where “the main goal is to select a subset of  $d$  features from the given set of  $D$  measurements,  $d < D$ , without significantly degrading the performance of the recognition system” (Pudil et al., 1994). As previously described, this approach assumes that a suitable criterion is used to evaluate the performance of feature subsets and selects features accordingly. Two examples of wrapper methods are Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS). SFS begins with the empty set and selects the best feature based on some criterion. Then, pairs of features are formed using the selected feature and each remaining feature until the best pair is selected. This continues until a predefined number of features are selected or the criterion is not improved (Nguyen, 2025c). SFS performs best when the optimal subset is small. SBS is the opposite to SFS; it begins with all features selected, then iteratively removes the worst feature from each subset until a

predefined number of features are selected or the criterion is not improved (Nguyen, 2025c). SBS performs best when the optimal subset is large.

**Embedded methods** integrate feature selection into the learning process and use regularisation techniques to penalise irrelevant features during model training. Embedded methods are more efficient than wrapper methods, but they tend to be model-specific and may not generalise well. Two examples of embedded methods are Decision Tree-Based Feature Selection and L1-norm. **Decision Tree-Based Feature Selection** creates a tree of Boolean decisions operating under the principle of Occam's Razor (e.g., ID3), whereby the simplest solution (smallest possible tree) is the preferred one (Nguyen, 2025a; Quinlan, 1986). Algorithmically, this is implemented by: (1) Choosing the best attribute(s) to split remaining instances, making that attribute a decision node, (2) Repeating this process recursively for each child, then (3) Stopping when either all instances have the same target attribute value, there are no further attributes, or there are no further instances. The best attributes to use as decision nodes can be selected using Ross Quinlan's Iterative Dichotomiser 3 (ID3) algorithm, or its expanded variants (C4.5, C5.0). Other methods to reduce the number of attributes are prepruning, which is unideal, and postpruning, the preferred approach accomplished via subtree replacement or subtree raising (Nguyen, 2025a). **L1-norm** implicitly selects features by eliminating less important ones during training. A neural network baseline objective function can be expressed as:  $J(W; x, y)$ , where vector  $W$  represents all network weights. An L1-regularised objective function is therefore  $J(W; x, y) + \lambda \|W\|_1$ , where  $\|W\|_1$  is the sum of the absolute values of the weights and  $\lambda$  is selected by trial and error. By making use of sparse weights, the model implicitly selects more important features. More specifically, the L1-norm is considered an embedded feature selection method because its geometry drives some feature weights exactly to zero. Tangentially, while the L2-norm is also embedded in the training process, it is considered a feature regularisation method (i.e., not selection) because all features remain in the model with non-zero weights (Appendix: Figure 1; Kleijn, 2024).

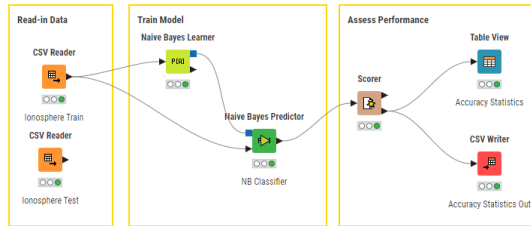
### (iii) Compare & Contrast: Filter, Wrapper, Embedding

Method	Acc.	Discussion #1	Cost	Discussion #2	Gen.	Discussion #3	More Details
Filter	Low	Evaluation is independ. of classifier; can miss feature interactions	Low	Simple stats/ info. theory metrics without model involvement; best cost	High	Independ. of classifier; FS is based on intrinsic feature properties, so generalises well	ii. Filter Methods
Wrapper	High	Specific to classifier; captures feature interactions; highest acc.	High	Ea. eval. requires train/validat-ing classifier multiple times (e.g., K-CV for FS/FC)	Low	Selected features optimised for specific model; less general to other classification algorithms	ii. Wrapper Methods
Embedding	Med	FS in model train; captures dependencies with moderate acc.	Med	More efficient than wrappers, as FS part of train; cost more than filter	Med	Partly depends on model (e.g., L1, DT); may not generalise well to different classifiers	ii. Embedded Methods

**(iv) Correlation-based Feature Selection (CFS)** assumes that good feature subsets contain features that are highly correlated with the class label, and uncorrelated with each other (Hall, 2000). CFS is considered a **filter**, as it performs **feature subset selection** without repeatedly invoking a model. CFS uses heuristic search algorithms like **greedy stepwise** and **best-first search (BFS)** to explore feature subsets and find local maxima. Unlike most single-feature filter methods which treat each feature independently, CFS considers interdependencies and can identify relevant features in the presence of moderate feature dependencies. CFS may fail to select *all* relevant features in situations where features depend strongly on others. In summary, CFS evaluates subsets of features as a whole using a multivariate merit function, unlike feature ranking methods that evaluate each feature independently. The premise of CFS is: *If the correlations between each individual component of a test and an external criterion are known, and the inter-correlations among the components are given, then it is possible to analytically determine the expected correlation between the composite test score (formed by summing the components) and the external variable independently* (Appendix: Equation 1; Hall, 2000).

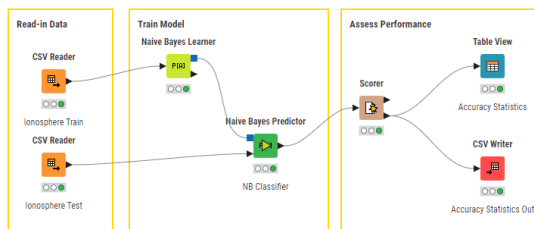
## Part Three (i.a, i.b, i.c)

### (a) NB-Classifier Performance on Training Data



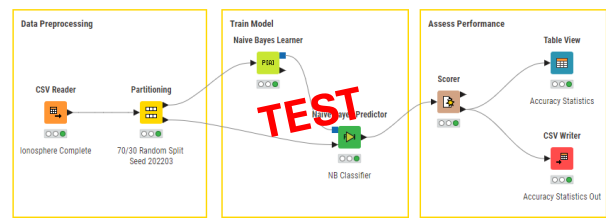
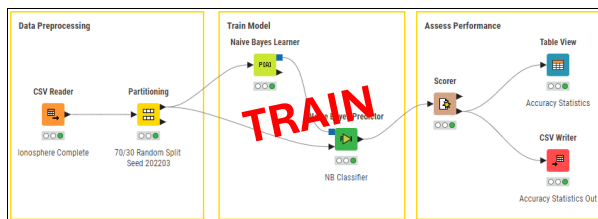
TP		FP		TN	FN
128		13		75	29
75		29		128	13
Accuracy = 0.828571 (82.9%)					
Recall	Precision	Sensitivity	Specificity	F-measure	
0.81529	0.907801	0.815287	0.852273	0.85906	
0.85227	0.721154	0.852273	0.815287	0.78125	
Cohen's kappa = 0.641913					

### NB-Classifier Performance on Test Data



TP	FP	TN	FN	
51	4	34	17	
34	17	51	4	
Accuracy = 0.8018868 (80.2%)				
Recall	Precision	Sensitivity	Specificity	F-measure
0.75	0.927272	0.75	0.894736	0.8292682
0.8947368	0.6666667	0.894736	0.75	0.7640449
Cohen's kappa = 0.5994962				

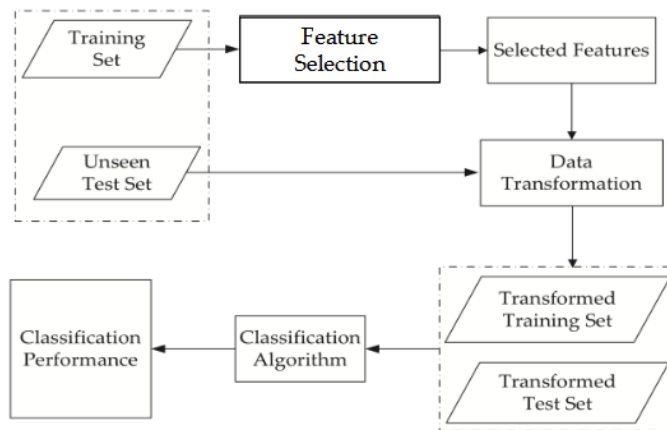
### (b) NB-Classifier Performance on Train-Test (70/30) Split Data



TRAIN DATA								
TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
138	12	73	22	0.8625	0.92	0.8625	0.858823	0.89032
73	22	138	12	0.858823	0.768421	0.858823	0.8625	0.81111
Accuracy = 0.85849056 (85.9%)				Cohen's kappa = 0.7082568				

TEST DATA								
TP	FP	TN	FN	Recall	Precision	Sensitivity	Specificity	F-measure
55	5	36	10	0.846	0.91666	0.846153	0.878048	0.88
36	10	55	5	0.878	0.782608	0.878048	0.846153	0.8275862
Accuracy = 0.86122448 (86.1%)				Cohen's kappa = 0.7019678				

**(c) Results Discussion:** The random split approach in (b) leads to higher overall accuracy for both training and testing data. This suggests that (a), the pre-defined train-test split contained within *IonosphereTrain.csv* and *IonosphereTest.csv*, did not present an optimal distribution of features and class labels, despite also presenting the model with a 70/30 train-test split *Ionosphere* dataset. In conclusion, random splitting (b) produces a more reliable NB model, likely as a result of better feature distribution and class representation, and improves the model's generalisability to new and unseen (test) data.

**Part Three (ii)**

In this description of Sam's approach, there was no mention of a proper train-test or train-test-validation split to prevent data leakage during feature selection (FS). If the complete dataset is used during the feature selection process, the experiments (or evaluations) are said to have **feature selection bias**. This may lead to better performance on the immediate data, but the model itself may not generalise or perform as well on new and unseen data. The figure on the left depicts the *correct* workflow for feature selection that Sam should use, which acts only on the training data, thus mitigating FS-bias (Nguyen, 2025b). After feature selection, data transformation is

applied to the train and test sets [as well as validation, if applicable], based on the features selected in the previous step. We then use our classification algorithm of choice and evaluate the classifier's performance on the transformed data. Comparing the two workflows, one in which FS-bias is present and another in which it is not, we observe the following results:

Method	Acc.	$n$	Feature Subset (FFS)	
FS-bias	93.6%	10	[F4, F3, F13, F15, F16, F1, F18, F26, F2, F23]	
No FS-bias	93.1%	10	[F4, F3, F27, F11, F17, F1, F18, F24, F13, F23]	

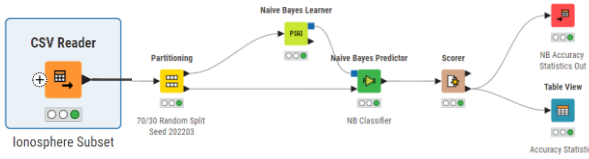
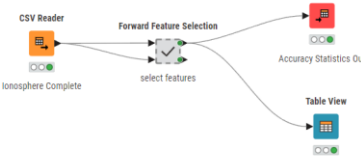
In Sam's approach, the complete Ionosphere dataset is used during feature selection, causing the test set to no longer be truly unseen. Specifically, Sam should first randomly split the Ionosphere dataset into 70/30 train-test datasets, then perform feature selection only on the training set. Sam may also choose to use k-fold cross-validation on the FS/FC system and take the average test accuracy as the final performance, but this is only beneficial if the dataset is small. Finally, Sam can transform the test set, applying the same selected features to the transformation, and re-evaluate the NB-classifier's overall accuracy (performance). Without these steps, Sam introduces FS-bias and risks overfitting to the immediate data.

**Part Three (iii, iv.a, iv.b, iv.c)**

The below table offers a comparison of NB performance on **i.a** no subsets, **iii** manual subsets, and **iv.a** FFS:

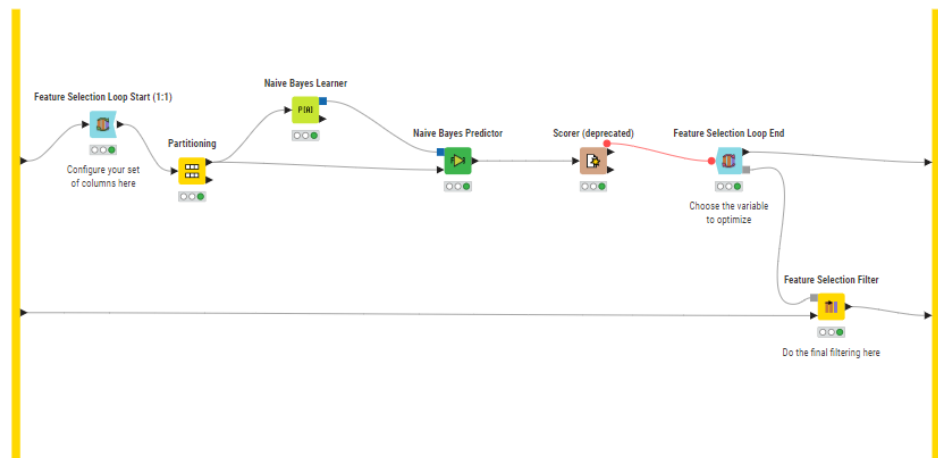
Method	Acc.	$n_F$	Features	Diagram
<b>(i.a) ORIGINAL</b> No subset → NB	80.2%	33	[F1-F33]	



<b>(iii) FILTER</b> Manual Subset → NB (no FS-bias)	<b>92.5%</b>	<b>10</b>	[F4, F3, F27, F11, F17, F1, F18, F24, F13, F23]	
<b>(iv.a) WRAPPER</b> FFS → NB (no FS-bias)	<b>93.1%</b>	<b>10</b>	[F4, F3, F27, F11, F17, F1, F18, F24, F13, F23]	

(iii) For this part of the experiment, the complete Ionosphere dataset was transformed by manually selecting a subset of 10 features, as identified through forward selection (ii) on the entire dataset. Manual selection was applied to both the training and test sets, and an NB classifier was trained and evaluated within KNIME. The resulting classification accuracy was 92.5%, which demonstrated a significant improvement over the baseline performance of 80.2% that was obtained using all 33 features with no feature selection (i.a). This result can be attributed to the removal of irrelevant and redundant features, which likely reduced noise and allowed NB to generalise better. Since feature selection was applied before the data split, this process remains methodologically sound and avoids data leakage.

(iv.a) The next phase of the experiment implemented a wrapper-based FFS method using only the training set and an NB classifier as the wrapped learner. The node-level implementation of FFS can be seen in the diagram on the right.



The FFS threshold was set to 10, as performance peaked and stabilised around this size (10-15), and worsened thereafter. The final subset

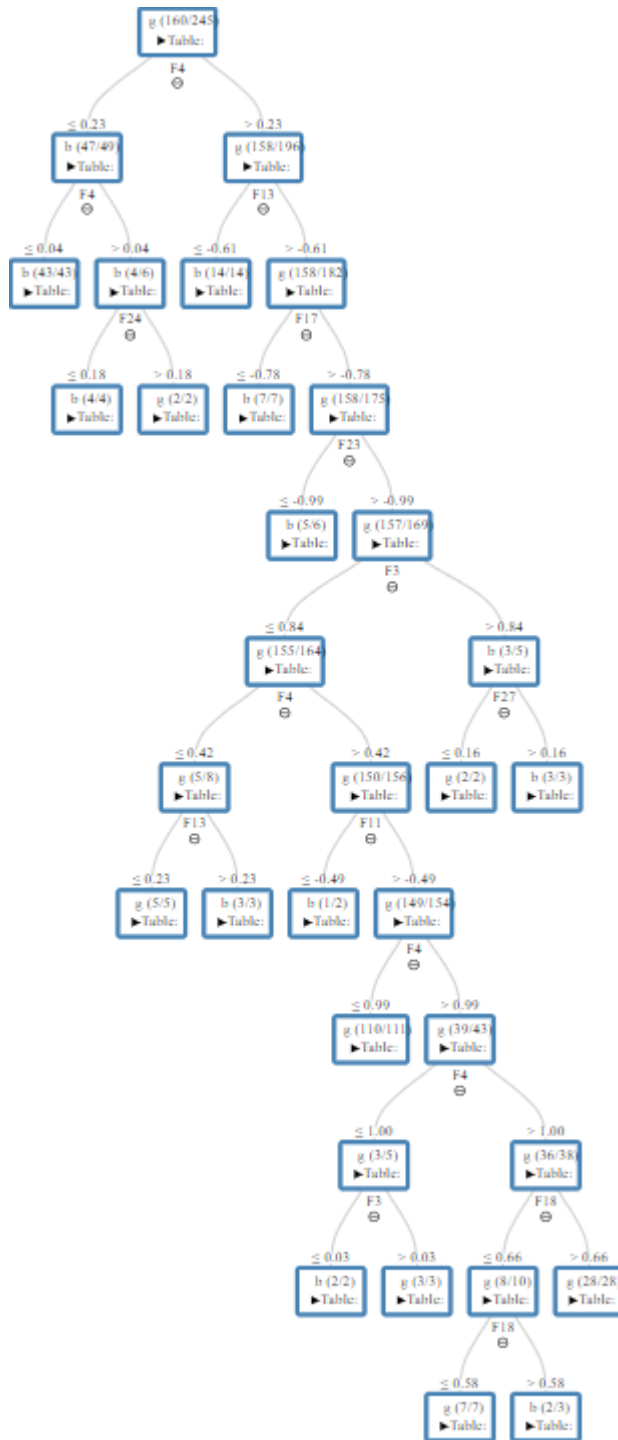
can be seen in the third row of the above table. After applying the same subset to the test data, the NB classifier achieved a testing accuracy of 93.1%, slightly outperforming the manual-filter method from (iii). This confirms that wrapper methods, while computationally more expensive, can more effectively optimise feature subsets for specific classifiers and datasets.

(iv.b) Compared to the baseline result (i.a), the wrapper-selected model achieved a 12.9% improvement in accuracy, thus confirming that a wrapper could effectively identify a compact and informative subset of features, allowing the classifier to generalise better. This outcome was expected, as a wrapper evaluates features based on their direct impact to model accuracy and, in doing so, captures both feature relevance and interactions. However, wrapper methods also raise the level of computational complexity (Part 2.iii).

(iv.c) Although the feature subsets in (iii) and (iv.a) were identical, their performance differed slightly (92.5% and 93.1%, respectively). Though small, this discrepancy highlights the benefit of data-driven selection methods like wrappers. While the manually chosen filter subset (used before partitioning) was effective, the wrapper approach tuned the feature subset specifically for NB on the actual training set, potentially leading to slightly better generalisation on unseen data. Further, the wrapper method is more robust in practice because it does not rely on prior knowledge or external heuristics for FS.

## Part Three (v)

Below is the Decision Tree View (simple) output for the specified DT in KNIME. Please note that Ionosphere subset from [Part3.iii](#) was used for simplicity.



The tree structure illustrates how only a subset of features were used to make class decisions. This indicates that the tree inherently performed feature selection by splitting only on informative features.

Dataset	Acc.
Train – C4.5	0.979591837 (98.0%)
Test – C4.5	0.91509434 (91.5%)

The C4.5 DT performs embedded feature selection as part of its tree-building process. At each internal node, the algorithm uses some criterion to determine the next best feature to use as a splitting point. As discussed in [Part 2.ii](#), embedded methods can use different criterion for splitting, but C4.5 typically selects the feature that yields the highest information gain, or gain ratio, (i.e., the feature that best splits the data to reduce entropy). Features that do not contribute meaningfully to classification are never invoked for splitting and so are effectively excluded from the model. In this way, C4.5 handles irrelevant or redundant features implicitly, making it robust to high-dimensional data without explicit feature preprocessing (Quinlan, 1993).

Thus, C4.5 is considered an embedded method because the process of selecting features is integrated into the model training phase. Also, as discussed in [Part 2.ii](#), a model may overfit if the tree becomes too deep or uses noisy features. Pruning methods can help mitigate this (but were not used in this default run).



### Part Three (vi.a, vi.b, vi.c)

(vi.a) Principal Component Analysis (PCA) was conducted in KNIME to perform dimensionality reduction on the designated Ionosphere training data. Using PCA on the training dataset, the top five principal components were retained for dimensionality reduction. According to the PCA Apply node configuration, these five components were predicted to preserve ~65.3% of the total variance in the original feature space. However, the true spectral decomposition revealed that the variance preserved by PC1-PC5 was in fact 59.79%, and the total variance had an eigenvalue of 9.9571. Figure 2 in the Appendix illustrates the new C4.5 tree structure using principal components 1-5 as decision nodes.

Component	Eigenvalue	Variance Explained (%)
PC1	2.915	$\frac{2.915}{9.9571} \approx 29.29\%$
PC2	1.231	$\approx 12.36\%$
PC3	0.694	$\approx 6.97\%$
PC4	0.626	$\approx 6.29\%$
PC5	0.486	$\approx 4.88\%$
Total Variance (%)		$\sum_{i=0}^{33} \lambda_i \approx 59.79\%$

(vi.b) The training and testing accuracy of the above PCA KNIME model were as follows:

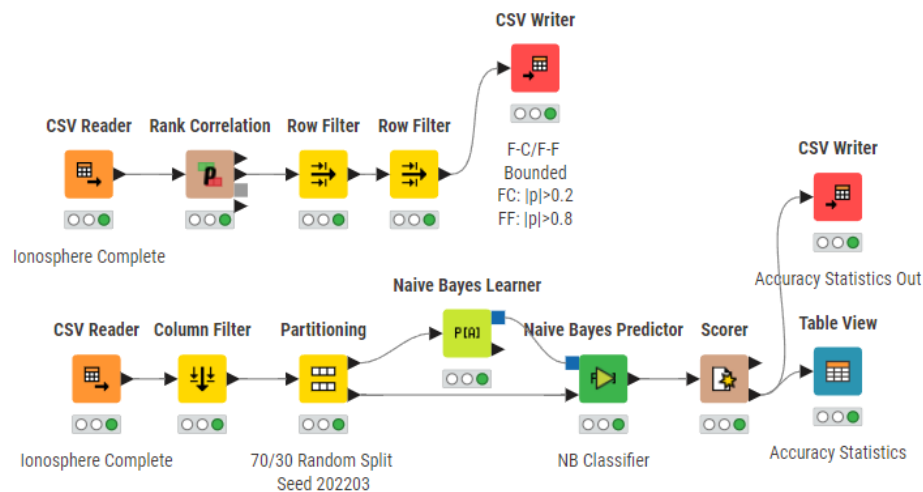
Dataset	Acc.
Train – PCA C4.5	0.959183673 (96.0%)
Test – PCA C4.5	0.905660377 (90.6%)

(vi.c) The results from (v) and (vi.b) show a clear performance distinction between C4.5 DT using original features versus principal components. While the difference in test accuracy is relatively small (91.5% vs 90.6%), there is a noticeable drop in training accuracy (98.0% vs 96.0%) when using PCA-

reduced inputs. This is because C4.5 builds decision trees by splitting along individual feature axes. When PCA is applied, the original features are replaced with linear combinations of features (i.e., principal components). The slightly lower training accuracy when using PCA suggests less overfitting, as PCA acts as a form of regularisation by discarding lower-variance directions (i.e., noise). However, the similar test accuracy indicates that generalisation remained strong even after PCA. Further, over 40% of the original data's variation was discarded through PCA, thus explaining the small performance degradation compared to the full-feature model (v). In summary, PCA reduced the input space from 33, or even 10, features to just 5 orthogonal components. Thus, PCA reduced model complexity and input redundancy, which can help maintain model generalisability despite some information loss.

### Part Three (vii.a, vii.b, vii.c)

(vii.a) As per Part 2.iv, we know that Correlation-based Feature Selection (CFS) assumes that good feature subsets contain features that are highly correlated with the class label, and uncorrelated with each other. The task is to use a Rank Correlation node with Spearman's rank correlation to measure the feature-class and feature-feature correlations for CFS on the training set. In KNIME's Rank Correlation node, the relevant outputs are "correlation matrix", indicating feature-feature correlations (used to detect redundant features), and "rank table", indicating feature-class correlation scores (used to select relevant features). In this example, we used common absolute values for Spearman's rho to bound the feature-class correlation  $>0.2$  (slightly correlated) and the feature-feature correlation  $>0.8$  (highly correlated), then trained an NB learner.



Using these thresholds (0.2 and 0.8, respectively), we select the following features:

Feature	F-C Correlation	F-F Correlation (MAX)	F-C Correlation (MIN)
F2	0.3433	0.5879	-0.1592
F4	0.3001	0.6169	-0.1494
F6	0.3113	0.6950	-0.1690
F8	0.2079	0.7171	-0.3382
F26	-0.2992	0.5578	-0.2947
F30	0.2251	0.6838	-0.2476
F32	0.2304	0.6838	-0.2220

(vii.b) NB Train & Test Accuracy:

Dataset	Acc.
Train – CFS	0.83265306 (83.3%)
Test – CFS	0.830188679 (83.0%)

## (vii.c) Final Results Comparison:

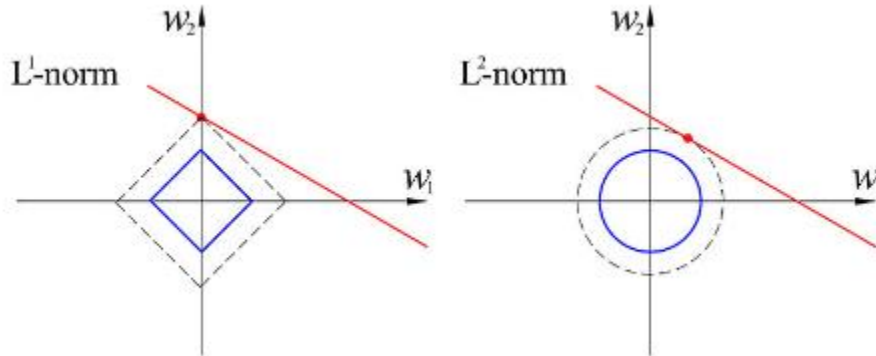
Method	Acc.	$n_F$	Features	Diagram
<b>(i.a) ORIGINAL</b> No subset $\rightarrow$ NB	80.2%	33	[F1-F33]	
<b>(iii) FFS FILTER</b> Manual Subset $\rightarrow$ NB (no FS-bias)	92.5%	10	[F4, F3, F27, F11, F17, F1, F18, F24, F13, F23]	
<b>(iv.a) WRAPPER</b> FFS $\rightarrow$ NB (no FS-bias)	93.1%	10	[F4, F3, F27, F11, F17, F1, F18, F24, F13, F23]	
<b>(vii.a) CFS FILTER</b> CFS $\rightarrow$ NB	83.0%	7	[F2, F4, F6, F8, F26, F30, F32]	

The CFS filter method selected 7 features based on their individual relevance to the class ( $F-C |p| > 0.2$ ) and low inter-feature redundancy ( $F-F |p| \leq 0.8$ ), achieving an 83.0% test accuracy using an NB learner. In contrast, the wrapper-based forward feature selection method (iv.a) optimised feature subsets by using classifier performance directly. The wrapper approach selected 10 features that yielded a much higher test accuracy of 93.1%, aligning with our expectations outlined in Part 2.iii:

Method	Acc.	Discussion #1	Cost	Discussion #2	Gen.	Discussion #3	More Details
Filter	Low	Evaluation is independ. of classifier; can miss feature interactions	Low	Simple stats/ info. theory metrics without model involvement; best cost	High	Independ. of classifier; FS is based on intrinsic feature properties, so generalises well	ii. Filter Methods
Wrapper	High	Specific to classifier; captures feature interactions; highest acc.	High	Ea. eval. requires train/validat-ing classifier multiple times (e.g., K-CV for FS/FC)	Low	Selected features optimised for specific model; less general to other classification algorithms	ii. Wrapper Methods

This ~10.1% performance gap reflects the key distinction between the two methods: CFS is a model-agnostic filter that is efficient but limited to pairwise correlations. The FFS wrapper method is more expensive, but it also captures higher-order interactions tailored to the classifier, which improves prediction accuracy. We also observed the impact of our selection criteria in the way that only one feature (F4) was common to both subsets. Finally, although not explicitly required, comparing the (vii) CFS filter with the (iii) FFS-based manual subset reveals that hybrid strategies may offer performance gains while controlling methodological complexity.

## Appendix



**Figure 1: Geometries of the L1- and L2-norm** (Kleijn, 2024).

$$(3) \quad CFS = r_{zc} = \frac{k\bar{r}_{zi}}{\sqrt{k+k(k-1)\bar{r}_{li}}}$$

where...  $r_{zc}$  is the correlation between the summed components and the outside variable;  
 $k$  is the number of components;  
 $\bar{r}_{zi}$  is the avg. of the correlations between the components and the outside variable;  
 $\bar{r}_{li}$  is the average inter-correlation between components.

*Interpretation:*

- Higher correlations b/w components and outside variable = Higher the correlation b/w composite and outside variable;
- Lower inter-correlations among components = Higher correlation b/w composite and outside variable;
- As number of components in composite increases, correlation b/w composite and outside variable also increases (assuming additional components are the same as the original components in terms of their average inter-correlation with other components and with outside variable).

**Equation 1: Definition of Correlation-based Feature Selection (CFS)** (Hall, 2000).



**References**

- Amazon Web Services. (2025). *What is ELK Stack?* <https://aws.amazon.com/what-is/elk-stack/>
- Benjamin, S. G., Brown, J. M., Brunet, G., Lynch, P., Saito, K., & Schlatter, T. W. (2019). 100 Years of Progress in Forecasting and NWP Applications. In *Meteorological Monographs*. American Meteorological Society.  
<https://doi.org/10.1175/AMSMONOGRAPHS-D-18-0020.1>
- Bonavita, M. (2024). On Some Limitations of Current Machine Learning Weather Prediction Models. *Geophysical Research Letters*, 51, e2023GL107377.  
<https://doi.org/10.1029/2023GL107377>
- Cover, T. M., & Thomas, J. A. (2005). Entropy, Relative Entropy, and Mutual Information. In *Elements of Information Theory, Chapter 2: Entropy, Relative Entropy, and Mutual Information* (pp. 13–55). <https://doi.org/https://doi.org/10.1002/047174882X.ch2>
- Digital, H. (2025). *IBM's Watson was once heralded as the future of healthcare – what went wrong?* <https://www.healthcare.digital/single-post/ibm-s-watson-was-once-heralded-as-the-future-of-healthcare-what-went-wrong>
- Farahat, Z., Zrira, N., Souissi, N., Bennani, Y., Bencherif, S., Benamar, S., Belmekki, M., Ngote, M. N., & Megdiche, K. (2024). Diabetic retinopathy screening through artificial intelligence algorithms: A systematic review. *Survey of Ophthalmology*, 69(5), 707–721. <https://doi.org/https://doi.org/10.1016/j.survophthal.2024.05.008>
- Hall, M. (2000). Correlation-Based Feature Selection for Machine Learning. *Department of Computer Science*, 19.
- Janosi, A., Steinbrunn, W., Pfisterer, M., & Detrano, R. (1989). *Heart Disease [Dataset]*. UCI Machine Learning Repository. <https://doi.org/10.24432/C52P4X>
- Kleijn, B. (2024). Objective Functions and Regularisation: AIML425 [Lecture 4 Notes]. *Master of Artificial Intelligence, Victoria University of Wellington*, 4–11.  
[https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML425\\_2024T2/LectureSchedule/lect4ObjF.pdf](https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML425_2024T2/LectureSchedule/lect4ObjF.pdf)
- Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1), 273–324. [https://doi.org/https://doi.org/10.1016/S0004-3702\(97\)00043-X](https://doi.org/https://doi.org/10.1016/S0004-3702(97)00043-X)
- Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Meroze, A., Hoyer, S., Holland, G., Vinyals, O.,



- Stott, J., Pritzel, A., Mohamed, S., & Battaglia, P. (2023). Learning skillful medium-range global weather forecasting. *Science*, 382(6677), 1416–1421.  
<https://doi.org/10.1126/science.adi2336>
- Liu, C., Liu, X., Wu, F., Xie, M., Feng, Y., & Hu, C. (2018). Using Artificial Intelligence (Watson for Oncology) for Treatment Recommendations Amongst Chinese Patients with Lung Cancer: Feasibility Study. *Journal of Medical Internet Research*, 20(9), e11087.  
<https://doi.org/10.2196/11087>
- Modeling, G., & (GMAO), A. O. (2015). *MERRA-2 3D IAU State, Meteorology Instantaneous 3-hourly (p-coord, 0.625x0.5L42), version 5.12.4*. Goddard Space Flight Center Distributed Active Archive Center (GSFC DAAC). <https://doi.org/10.5067/VJAFPLI1CSIV>
- Nguyen, B. H. (2025a). Embedded Feature Selection [AIML427 Lecture Notes]. *Master of Artificial Intelligence, Victoria University of Wellington*.
- Nguyen, B. H. (2025b). Feature Manipulation [AIML427 Lecture Notes]. *Master of Artificial Intelligence, Victoria University of Wellington*.  
[https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML427\\_2025T1/LectureSchedule/Week2\\_FeatureManipulation.pdf](https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML427_2025T1/LectureSchedule/Week2_FeatureManipulation.pdf)
- Nguyen, B. H. (2025c). Filter Feature Selection [AIML427 Lecture Notes]. *Master of Artificial Intelligence, Victoria University of Wellington*.  
[https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML427\\_2025T1/LectureSchedule/Week3\\_FS\\_Embedded.pdf](https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML427_2025T1/LectureSchedule/Week3_FS_Embedded.pdf)
- Nguyen, B. H. (2025d). Introduction to Big Data [AIML427 Lecture Notes]. *Master of Artificial Intelligence, Victoria University of Wellington*.  
[https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML427\\_2025T1/LectureSchedule/Week1\\_WhatIsBigData.pdf](https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML427_2025T1/LectureSchedule/Week1_WhatIsBigData.pdf)
- Owens, R., & Hewson, T. (2018). *ECMWF Forecast User Guide*. ECMWF.  
<https://doi.org/10.21957/m1cs7h>
- Pudil, P., Novovičová, J., & Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11), 1119–1125.  
[https://doi.org/https://doi.org/10.1016/0167-8655\(94\)90127-9](https://doi.org/https://doi.org/10.1016/0167-8655(94)90127-9)
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81–106.  
<https://doi.org/10.1007/BF00116251>
- Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. *Morgan Kaufmann*, 3.  
<https://doi.org/10.1007/BF00993309>

Service, C. C. C. (2023). *ERA5 hourly data on single levels from 1940 to present*.  
Copernicus Climate Change Service (C3S) Climate Data Store (CDS).  
<https://doi.org/10.24381/cds.adbb2d47>