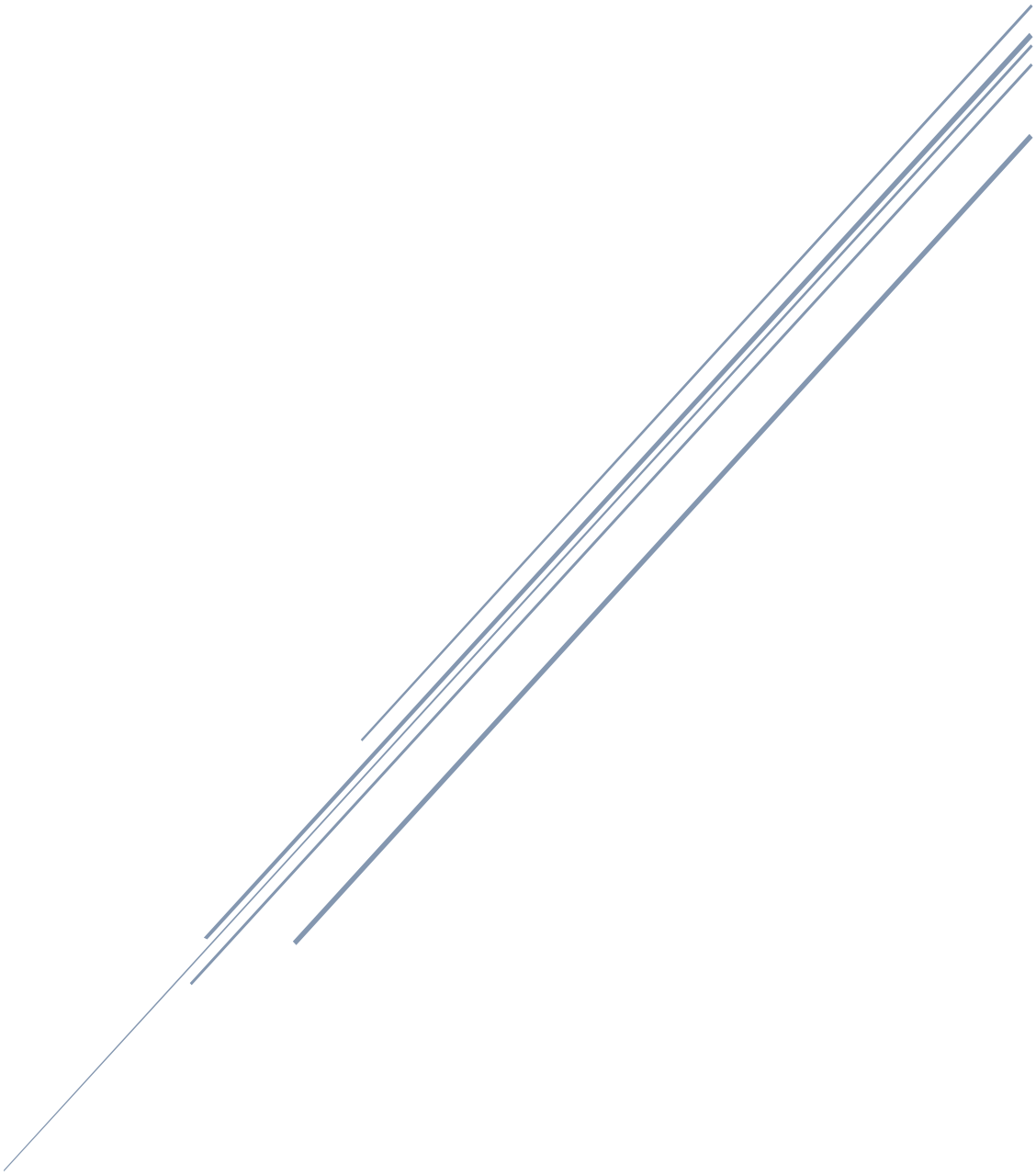
FUNDAMENTALS OF DATA ANALYTICS - 32130

Assignment 3: Data Mining in Action



Abhinav Chaudhary
13313162

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. Introduction

The assigned classifier is K Nearest Neighbour. Apart from KNN two more classifiers named, Random Forest Learner and Decision Tree Learner are used to analyse the datasets provided to predict the class attribute, i.e. “QUALIFIED”. The data mining problem, data pre-processing and transformations, modelling and solution is shown below.

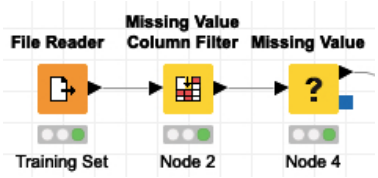
2. Problem Statement and Data Understanding

Two data sets are provided, namely, ‘TrainingSet.csv’ and ‘TestingSet.csv’ to be worked upon. ‘TrainingSet’ is the dataset that will be fed to the classifier to build a predictive model and ‘TestingSet’ is the dataset for which the class attribute value (QUALIFIED) is predicted. The ‘TrainingSet’ contains 75008 rows and 39 attributes. The ‘TestingSet’ contains 32148 rows and 38 attributes. The additional attribute that the ‘TrainingSet’ has is the class attribute, which is utilized to build the classifier model and estimate the classifier’s accuracy.

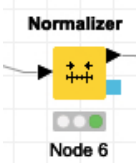
3. Data Pre-processing and Transformations

3.1 Missing Values

After evaluating the dataset it is found that most of the attributes have missing values. To fix this problem, the Missing Value Column Filter and Missing Value nodes are used in KNIME. In missing value column filter node, the missing value threshold is kept at 70% and for the missing values node, different parameter settings are tested such as mean, fix value, rounded mean, etc. and based on the accuracy results, it is decided to keep the parameter settings for the numeric values (number and double) as median and the string values as most frequent value.

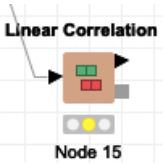


3.2 Normalisation

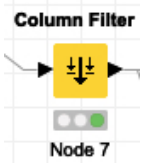


The attribute price contains very large values with the range of 0 to 25,100,000 and the mean of 357,483. These large values decrease the performance of the classifier and thus the normalizer node is used to perform min-max normalisation on the attribute price.

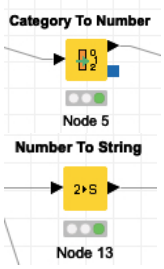
3.3 Irrelevant Data



The large amount of data in the training Set, some of which might not useful at all, would have decreased the performance of the classifiers. Thus, correlation measure between the attribute 'QUALIFIED' and all other attributes is calculated using the liner correlation node. To increase the performance and efficiency of the classifier top eight attributes with highest correlation with the 'QUALIFIED' attribute are chosen and all the other attributes are filtered out using the column filter node.



3.4 Value Transformation



Classifiers such as K Nearest Neighbour and Random Forest need numeric values to work. Therefore category to number node is used to transform values of the attributes such as HEAT_D, AC, STYLE_D, CONDTN_D, etc. Also, number to string node is used in KNIME to transform the value of the class attribute (QUALIFIED) from numeric value to string value because classifiers such as K Nearest Neighbour, Decision Tree and Random Forest need nominal value to work.

4. How the Problem is Solved?

Different classifiers are used to make the models that could predict the class attribute, QUALIFIED. KNIME is used to solve the problem with the steps as mentioned below:

- First the data is cleaned by performing data transformations to the Training Set, missing values are filled, large values are normalised and value transformations are performed to convert numeric values to string and string values to numeric values
- There are 39 attributes in the Training Set. To filter out the most efficient attributes for building the classifier that could predict the class attribute is important
- Linear correlation node is used to measure the correlation between all the attributes with each other
- Top eight attributes with highest correlation with the class attribute 'QUALIFIED' are used to build the classifiers
- Three classifiers named K Nearest Neighbour, Decision Tree and Random Forest are used to build predictive model
- Accuracy test is conducted by inputting different parameter settings on each of the three classifiers build
- After measuring and recording the value of accuracies, the best parameter settings for each of the classifier is selected
- The complete workflow of the work is shown in the Figure 1

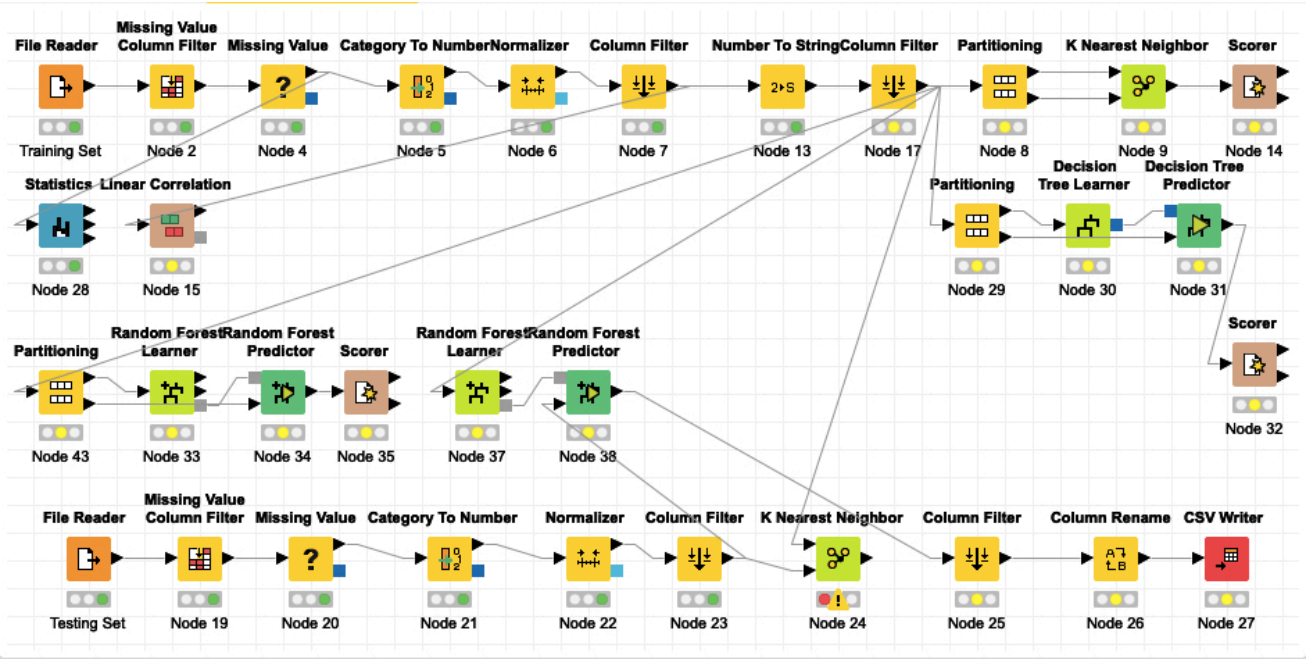


Figure 1

5. Classification Techniques Used and Summary of the Results and Parameter Settings

Three classifiers are used to predict the class attribute. The classifiers' details are as following:

5.1 K Nearest Neighbour

Classifies a group of testing set data with the use of K Nearest Neighbour algorithmic program. An affordable performance is exhibited as the underlying algorithmic program used a KD tree. One limitation of KNN is that it is limited or solely suited for few thousand coaching instances. Another property of KNN classifier is that it only uses numeric values to train the model.

Partitioning node is used to split the data into two parts, 80% of the data is used for training and rest 20% for prediction. The data is drawn randomly with the option Use Random Seed ticked. Figure 2 shows the parameter settings for the partitioning node.

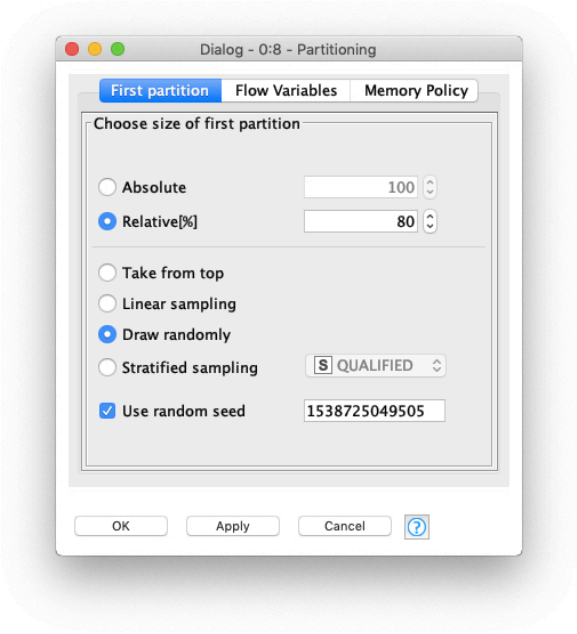


Figure 2

Then the data is fed to the K Nearest Neighbour node with three different values of K and Weight Neighbours by Distance option ticked for the first three observations and then unticked for another three observations. Following results as shown in Table 1 are observed with different parameter settings:

Column with class label	**Number of neighbours (K)**	**Weight neighbours by distance**	**Accuracy**
Qualified	7	Yes	81.636%
Qualified	9	Yes	**81.849%**
Qualified	11	Yes	81.689%
Qualified	7	No	**82.169%**
Qualified	9	No	82.002%
Qualified	11	No	81.609%

Table 1

- It is observed that with Weight neighbours by distance option ticked, the accuracy is highest when the K is 9
- When the Weight neighbours by distance option is unticked, the accuracy is highest when K is 7

5.2 Decision Tree Learner

The main memory of the node for Decision Tree Learner contains a classification decision tree. The decision making attributes used in this classifier can be numeric or nominal but the target attribute should always be nominal. Numeric parts are

constantly twofold (two results), separating the area in two parcels at a given split point. Nominal parts can be either double (two results) or they can have the same number of results as nominal qualities. On account of a parallel split the nominal qualities are partitioned into two subsets. The calculation gives two quality measures to part computation; the GAIN RATIO and the GINI INDEX. Further, there exist a post pruning strategy to decrease the tree size and increment forecast exactness. The pruning technique depends on the base portrayal length standard.

Again, the Partitioning node is used to split the data into two parts, 80% of the data is used for training and rest 20% for prediction. The data is drawn randomly with the option use random seed ticked.
Figure 3 shows the dialog for parameter settings of the decision tree learner.

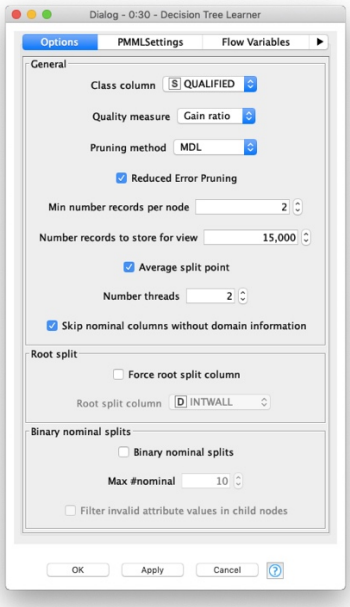


Figure 3

The following results as shown in Table 2 are observed with different parameter settings of the decision tree learner classifier:

Quality Measure	**Pruning Method**	**Accuracy**
Gain Ratio	MDL	89.708%
Gain Ratio	No Pruning	86.608%
Gini Index	MDL	**89.768%**
Gini Index	No Pruning	86.755%

Table 2

- It is observed that the maximum accuracy is achieved when the quality measure is selected as Gini Index and the Pruning method is selected as MDL

5.3 Random Forest Learner

Random Forest Learner takes in an irregular forest, which comprises a picked number of decision trees. Every one of the decision tree models is found out on an alternate arrangement of lines (records) and an alternate arrangement of sections (depicting characteristics), whereby the last can likewise be a bit-vector or byte-vector descriptor (e.g. atomic unique finger impression). The column sets for every decision tree are made by bootstrapping and have an indistinguishable size from the first information table. For every hub of a decision tree, another arrangement of traits is dictated by taking an irregular example of size sqrt(m), where m is the aggregate number of characteristics.

The partitioning node is used to split the data into two parts. 80% of the data is used for training and rest 20% for prediction. The data is drawn randomly and the option 'Use random seed' is kept unticked because by using the random seed, accuracy of the classifier is decreased.
Figure 4 gives the general overview of the parameter settings available in the random forest learner.

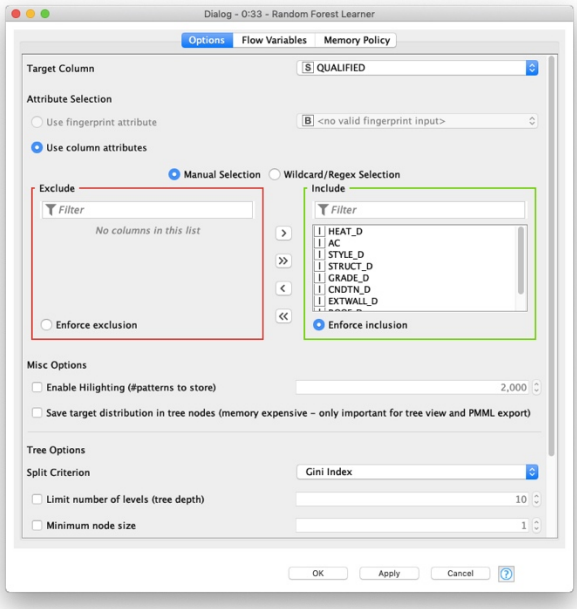


Figure 4

Different parameter settings are tested to find out the accuracy for each of the settings. The result is recorded in the Table 3 below.

Split Criterion	**Tree Depth**	**Number of Models**	**Accuracy**
Information Gain	0	300	89.868%
Information Gain	15	300	89.976%
Information Gain	15	200	89.455%
Gini Index	0	300	**91.541%**

Gini Index	10	300	87.282%
Gini Index	15	200	89.535%
Information Gain Ratio	0	300	89.548%
Information Gain Ratio	15	300	89.535%
Information Gain Ratio	20	200	89.561%

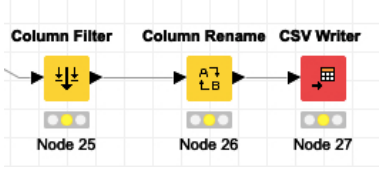
Table 3

- It is observed that the highest accuracy is achieved with Gini Index as split criteria, tree depth as 0 and the number of models as 300
- The accuracy achieved is 91.541%

6. Classifier Selected

Out of the three classifiers that are build, **Random Forest Learner** is selected to be used for the final output. Random Forest depends on directed learning algorithm - which implies that yield datasets are utilized to maximum of their efficiency to prepare the model in order to achieve the preferred output. It fabricates various decision trees and combines them for acquiring exact forecasts. The benefit of utilizing this model is high exact outcomes, more adaptable model, can deal with huge amounts of information and can be utilized for grouping. Moreover, the random forest learner provided the highest accuracy among the three classifiers build. The final result is calculated based on the random forest learner with parameter settings as:

- **Target Column:** QUALIFIED
- **Attribute Selection:** Use column attributes
- **Enforce Inclusion:** Top eight attributes with highest correlation with QUALIFIED are selected
- **Split Criterion:** Gini Index
- **Tree Depth:** 0
- **Number of Models:** 300
- **Use Static Random Seed:** Yes



The final result is written into a CSV file using the node configuration in KNIME as shown. Column Filter is used to filter just the column with predicted results, Column Rename is used to rename that column to QUALIFIED and finally CSV Writer is used to write the results into a CSV file which is uploaded to Kaggle and received a score of 0.89385

7. Reflection

The assignment provided me with an opportunity to work on a data mining problem. I realised that data mining is just a part of the bigger workflow. One needs to perform various tasks for the data mining to be successful. Understanding the data, how different attributes relate to the problem, which of them can be helpful to solve the problem and which of them needs to be eliminated. Data pre-processing also plays a crucial role in shaping the accuracy of the predictive model. By doing this assignment, I now have clearer understanding of the importance of data mining and data analytics in the business environment. I can now picture how data mining and data analytics help companies make insightful decisions. I also learned the importance of data cleaning and accurate data sets.

The assignment helped me learn new nodes in KNIME and how to implement them. For example, when I first built my KNN classifier I used all the attributes for training and the accuracy was 59%, I did some research and learned about the node 'linear correlation'. I implemented that node to calculate the highest correlation of different attributes with the attribute QUALIFIED and the second time I ran my model, I used only top eight attributes with highest correlation with the attribute QUALIFIED and the accuracy increased by almost 30%. I also developed a clearer understanding of how and what data transformations should be performed based on the given problem statement. Deciding on which method to use for performing data transformations such as filling missing value was a challenging task and also time consuming, but as I progressed through my assignment I got a better understanding of how different node's parameter settings for the missing values node affect the classifier's accuracy and how each of the settings such as mean, fix value, rounded mean, etc. fill the missing values.

During the second task of the assignment that was to build different classifiers, I studied and learned about various classifiers and how each of them work. Something essential that I learned was that none of the classifiers are ideal for a wide range of datasets and that attribute determination, parameter settings majorly affect the precision of the forecast. After testing three classifiers, I inferred that Random Forest Learner worked best for the given data set.

If I were to do this assignment again, I would approach the problem differently. The changes I would make are as following:

- I would focus more on data transformations, as it can cause a huge impact on the accuracy of the model. I would use different parameter settings more vividly this time and test the accuracy of the classifiers with more changes made to the parameter settings of the data transformation nodes

- I would also try out new classifiers like Naive Bayes, Gradient Boosted Trees, Logistic Regression Learner and Neural Networks
- Use other tools such as Orange to solve the problem
- I would also try to solve the problem with programming languages such as Python rather than using user interface based tools. This would also improve my programming

References

NodePit, Viewed on 6th October 2018, <https://nodepit.com/node/org.knime.base.node.mine.knn.KnnNodeFactory>

NodePit, Viewed on 6th October 2018, <https://nodepit.com/node/org.knime.base.node.mine.treeensemble2.node.randomforest.learner.classification.RandomForestClassificationLearnerNodeFactory2>

NodePit, Viewed on 6th October 2018, **<**https://nodepit.com/node/org.knime.base.node.mine.decisiontree2.learner2.DecisionTreeLearnerNodeFactory3**>**