CORE

Business and Data understanding

The data was likely collected so that people who wished to walk on any of the public tracks operated by DOC would be able to gauge what they will be embarking upon. Because of this, the data may help someone decide whether they are able or want to take on a particular track.

The feature ‘difficulty’ shows us that some tracks have varying difficulties for the same track. This likely means that some track have more than one difficulty in the sense that difficulty may change as the track progresses. For cleaner attribute values and to remove outliers it may be better to look at a tracks difficulty holistically (one value) as opposed to the progression of difficulties (multiple values). This is because classification will struggle to pull out some of the more important patterns if there are many outlier values given by the difficulty feature. When regarding what difficulty the track should be DOC have regard to the information given on this webpage https://www.doc.govt.nz/parks-and-recreation/things-to-do/walking-and-tramping/track-categories/. The fact the track may have multiple difficulties may mean that it a longer track and may comparatively take longer to other tracks.

The ‘X’ and ‘Y’ features give us a understanding of where the track actually is in New Zealand. This may help inform hikers whether they can easily access the track or not. Furthermore these features give understanding as to what the type of terrain the track has. For example if the coordinates correspond to somewhere in the Southern Alps of New Zealand, it is easily deduced that the track will likely be steep and difficult. Whereas if it corresponds to a beach, it will likely be flat and long. The ‘Shape_Length’ feature gives us insight into how long the track actually is in meters. Similarly this can give insight into how long the track will take as the longer a track is the longer it will take.

While having regard to all these features we can see that they will be useful in order to predict their length class. How they are useful will be explained in the dataset exploration section.

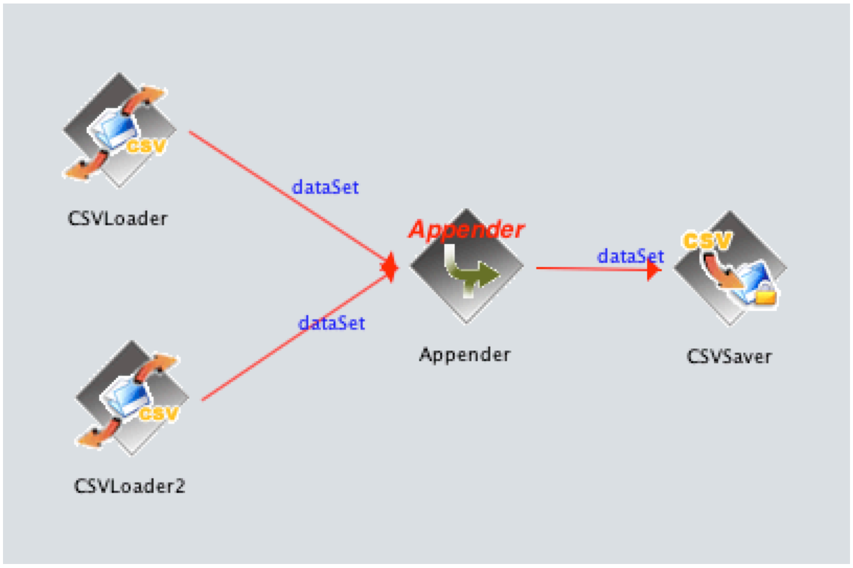
Data preparation (Pre-processing)

Firstly I attempted to classify the data set without any pre-processing just to see what the classification accuracy was like originally. Through this I found that the classification accuracy was sitting around 40%. Because of this I found that the data set was in need of some serious imputation and pre-processing, namely:

- I needed to apply a numeric to nominal filter as the class was originally being displayed as a continuous variable as opposed to being three distinct classes of 0,1,2.
- With regard to the difficulty attribute there are many outlier instances because they are said to have more than one difficulty. Because of this I made inferences from these instances as to what the difficulty should be by looking at them holistically. This would align with the data understanding described above with regard to the categories of difficulty. For example if the attribute value displayed “Easy, Intermediate, Advanced” I would take the middle value. This is because it would give a fair estimate of the difficulty of the track and would align with.
- I removed the ID attribute as it was giving me no actual predictive value and I regarded it as a redundant feature.
- There was one missing data value in the difficulty feature so I replaced it with the most common value for that feature which is ‘Easy’

Before I began to actually predict the test data I ran into an error which described an incompatibility between the trained model and the test data. Because of this I had to append the two datasets using the pipeline described in 1.1 so that I could ensure that they would have the same header files (which became very large after using the filter numeric to nominal). After this I converted the respective CSV files to ARFF files then replaced their header with the appended ARFF file.

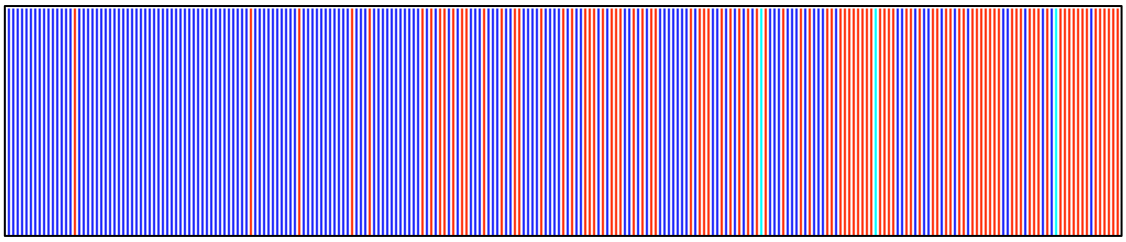
1.1 Pipeline for appending two datasets



Dataset exploration

I observed many patterns as to what attributes and values helped the prediction model make the best classification. One of which was the length of the track corresponding to different time classes dependent on how long the track is. Below in 1.2 we can see that as the length of the track increases/decreases the time class which the track corresponds to changes. In the middle of 1.2 representing tracks with an average length are much more difficult for the prediction model to predict because prima facie there seems to be no pattern. However this feature still provides a useful pattern for the prediction model in the sense that in both extremes of track lengths (short and long) it will likely give us the time class correctly. This is a very intuitive pattern as we would expect that as the length of the track increased the time it would take to complete would increase, hence 1.2. I should think that this pattern should drastically increase the classification accuracy for time class. This is because there is a clear pattern and it gives the model direct insight as to what time class a new instance of track should be given if it gives a length value. For example a track with a small length will be classified as a track with a low time class and visa versa.

1.2 Length of track instances in order from least to most



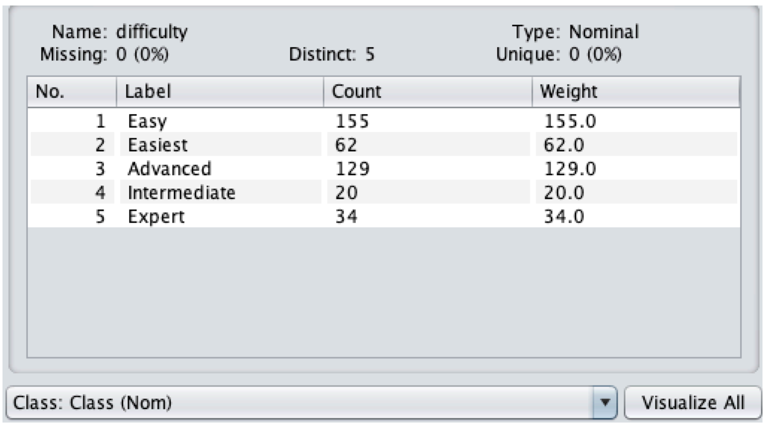
Key for time class

Blue = 0

Red = 1

Teal = 2

1.3 Respective difficulties of track proportions of each time class.

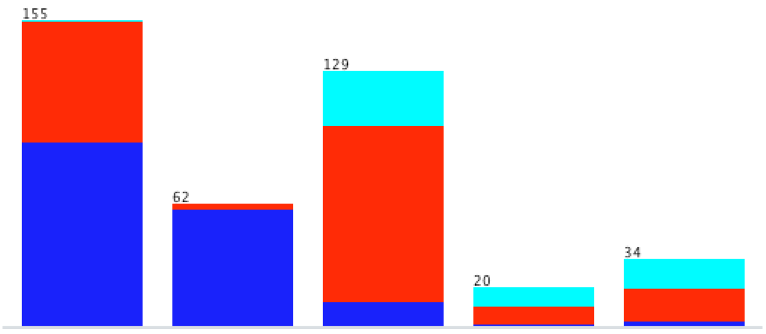


Name: difficulty
Missing: 0 (0%)
Distinct: 5
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	Easy	155	155.0
2	Easiest	62	62.0
3	Advanced	129	129.0
4	Intermediate	20	20.0
5	Expert	34	34.0

Class: Class (Nom)

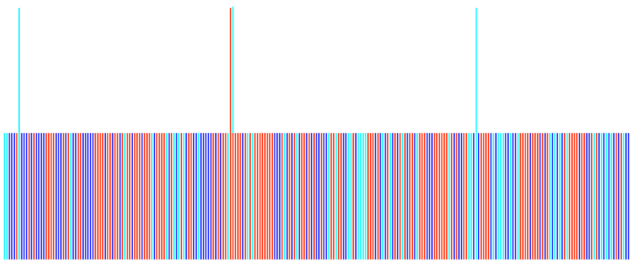
Visualize All



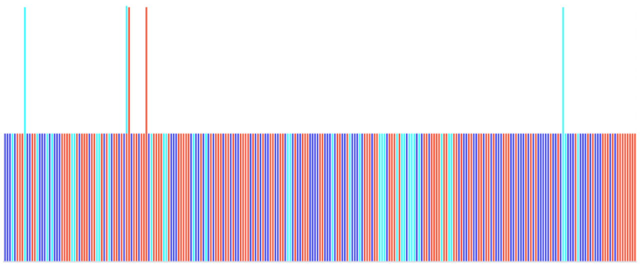
By having regard to 1.3 we can infer some patterns from the respective difficulties as to what time class they correspond to. Although the patterns are not as clear as given above by 1.2 the predictive model will still find it useful to see what type of difficulties correspond to which time class. We can firstly see the pattern that the tracks with difficulty ‘Easiest’ almost always are of the lowest time class (0 - blue). We can also infer the pattern that only tracks with difficulties of ‘Intermediate’, ‘Advanced’ and ‘Expert’ correspond to a high time class (2 - teal). These are intuitive patterns the model will have picked up on as harder tracks usually will take longer than less difficult tracks.

These patterns identified in 1.3 I don’t think are nearly as influential to the model as the patterns deduced by 1.2. However I do think that it will increase the accuracy when a new instance of track gives a length which is somewhere in the middle of what the prediction model has been built on. This is because the model can then look to the difficulty feature and see what type of time class it is likely to be as track instances which have the difficulty value of ‘intermediate’ and above have very few small time classes (0). Thus, it is likely be a higher time class of either 1 or 2 and should be predicted/classified accordingly. Furthermore if a high/low difficulty is paired with a short/long length of track respectively it will likely give the model strong reason to give either a low or high time class. This is an example of how features may interact with one another to achieve the best results on instances which provide less strong patterns.

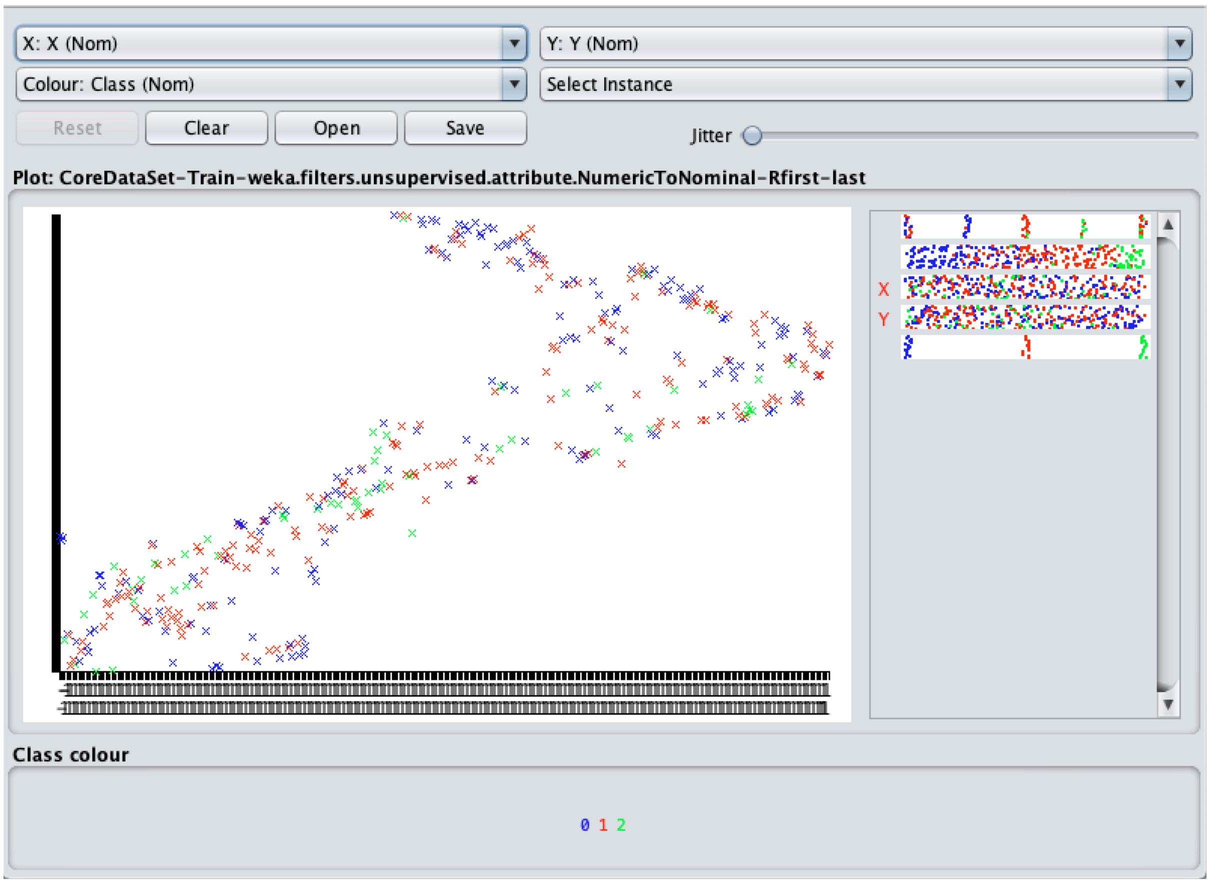
1.4 X coordinate time class in ascending order



1.5 Y coordinate time class in ascending order



<u>1.6 the positions of each track with regard to what time class they are.</u>

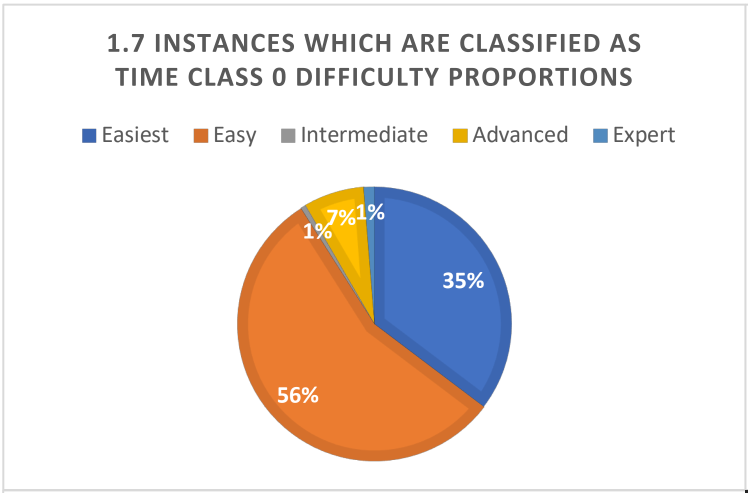


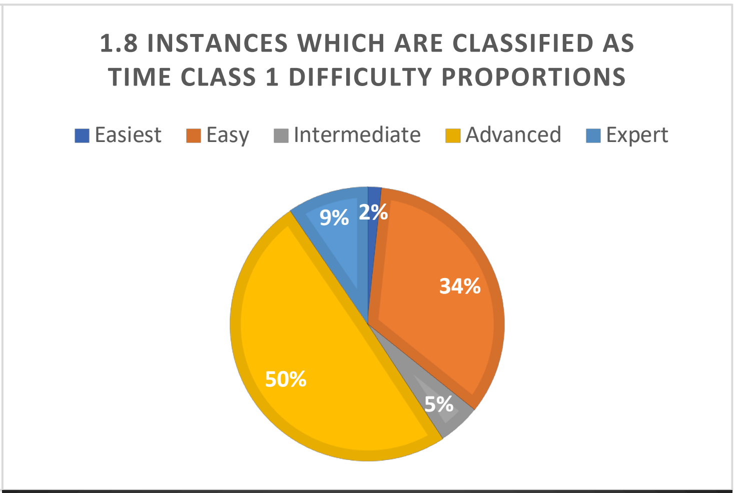
Plot: CoreDataSet-Train-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last

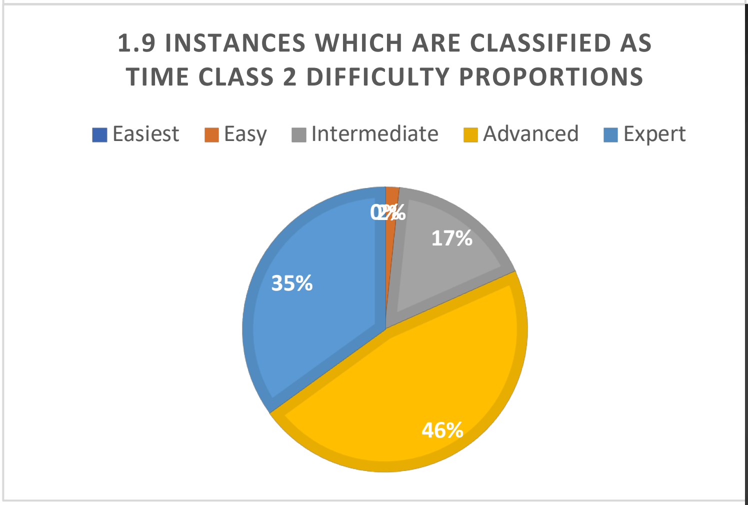
By having regard to 1.4 and 1.5 we can see that we are not able to deduce any strong patterns as the different time classes seem almost random for X and Y coordinates. However when you take the X and Y coordinates together they give you some interesting patterns for the model to use to predict time classes. We can see in 1.6 that essentially the coordinates give us a distorted version of New Zealand, which would make sense as there are many waling tracks in New Zealand. However more interestingly by regarding how these two features interact (X and Y) we can see that the highest time class instances seem to be more central in New Zealand. This is opposed to the lower time class instances being situated on the outside of New Zealand. Now, why is that? This is likely because the track which are in the middle of the country are treks which involve mountain ranges such as the Southern Alps, which correspond to longer and more difficult walks and take a longer time (a time class of 2). Similarly for the tracks on the coasts of New Zealand, they will be far easier than the mountain treks as they are relatively flat and easy to access, which in turn will make the track easier to complete in good time (a time class of 0 and 1). The predictive model will use this information to help predict the time class by looking at where the track is in relation to others. For example instances which have a lower Y coordinate are more likely to have a track with a high time class than a instance with a high Y coordinate. This information is very interesting and would be very influential to correctly classifying time class instances correctly if we could combine the coordinates with its corresponding terrain in relation to where it is in New Zealand. Because at the moment I think the model will struggle to draw the same type of inference we are able to deduce from just the coordinates alone.

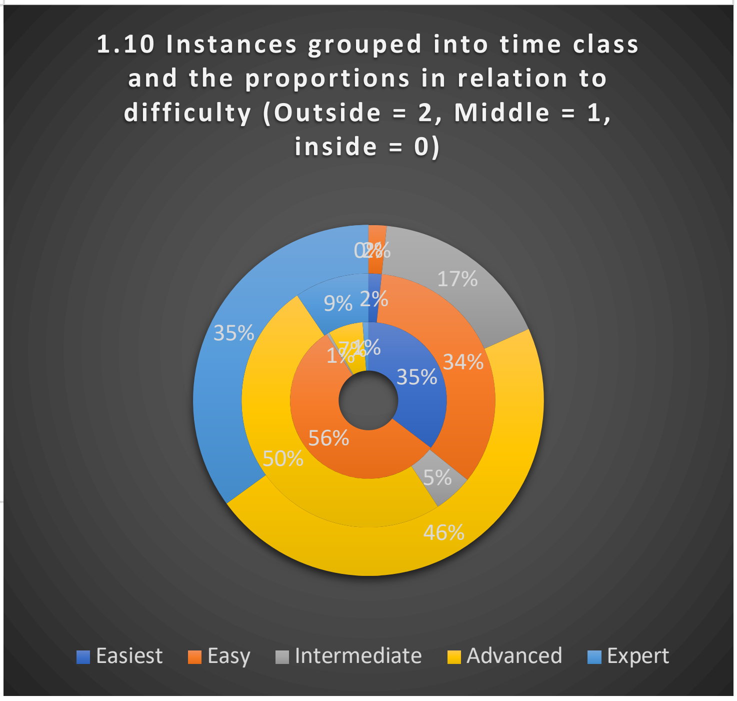
<u>Visualisation</u>

I believe I have visualised and illustrated some of my important findings through 1.2/1.3/1.4/1.5/1.6. However I also believe that some of these findings regarding patterns can be visualised better for the lay person. Because of this I have reimagined them by using excel and perhaps they will provide easier understanding for lay persons who have little knowledge of machine learning. Below are some visualisations and illustrations of the important finding of how difficulty relates to track length (time class) through pie charts (1.7/1.8/1.9). This I believe allows any lay person to observe that there is a pattern in the dataset in the sense that as the time class increases the difficult tracks are more prevalent and the easier tracks diminish. A combination of all the charts is represented by a donut chart in 1.10, which shows the evolution of each grouping of time class instances in relation to difficulty from looking inside out.









Kaggle Competition

I found myself with a slightly lower than average accuracy score with regard to the public leader board, which I can only attribute to my handling of the difficult feature in the data set. I now realise that it may not have been the best idea to holistically rank the difficulty of the specific track instance because the test set also had these outliers. This would have caused my prediction model to fail to understand these types of outliers as it had not been exposed to them and thus the model classified the instances incorrectly. However I believe that because I have removed these outliers my prediction model will have picked up on some of the more genuine patterns given above and hopefully should perform better on the hidden test set.

COMPLETION

Business and Data understanding

The business and data understanding is much the same as given in the CORE component except instead of a time class feature which we will be trying to classify, actual times are given by means of hours etc. This will allows us to be more accurate in our findings. This will consequentially give people more information to allow them to decide whether they wish to embark upon the trek.

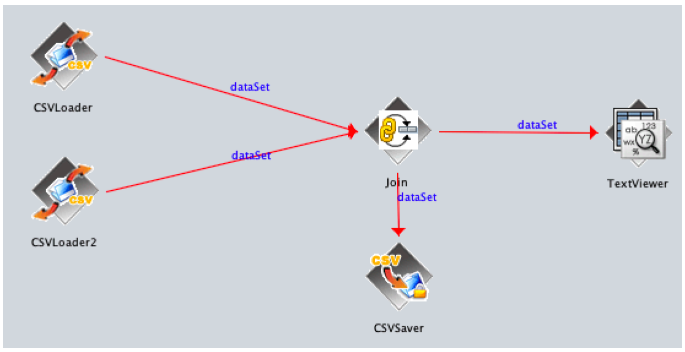
Another reason why we may want to predict the time a track will take is for safety reasons. If a person has embarked on a relatively larger walk and has told someone when they have left, the DOC rangers on the track will be able to refer to the predicted times to see if there is need to worry about that particular person(s).

Data preparation (Pre-processing)

Firstly because there was two data sets I had to find common ground between them, which I found to be the name of the track. However because the csv instances identified through their names were not in the same order and would consequentially not merge correctly. Because of this I sorted the instances from A-Z and the two csv files were then merged using 2.1. However before I was able to merge the two files I had to undertake significant cleaning of the data in order for the pipeline described below to work namely:

- An I/O error was occurring due to the apostrophes in the feature values of 'name', so I had to replace that with a '/'
- I replaced all commas in the feature values of 'difficulty' with '/' because it was causing layout issues with the join process.
- All null values for the meantime have been replaces with "unknown" until I deal with missing values properly.

2.1 Merge pipeline created in knowledge flow



I have also undertaken some more pre-processing regarding the features in the dataset. With regards to the completion time feature it has varying times in terms of how long a track may take. This is because each track may have subsidiary tracks or it might be longer to go one way and not the other. Because of this I have decided to just take the mean time of the values presented by the feature. I believe this is the fairest way to deal with it, without having more data as to what specific subsidiary track etc someone may want to take. One other form of processing which runs parallel with the one given above is that I have converted hours to minutes. This will allow for easier understanding of the continuous values (track times in minutes) for the prediction model and it is a requirement of the specification. There were also some missing values regarding the time in minutes a track would take, usually they would be tracks which were extremely long (60-70 Kilometres). To alleviate this problem I decided to take information from similar track and give them values which I believed to be intuitive. For example a track which was 77 kilometres I would assume to be a trek which would take a couple of days, therefore I would give the instance a value which matches this reasoning.

I have fought with myself over whether I should make the same changes to the 'difficulty' feature values. However I have decided even though it may cause me detriment in the public Kaggle competition I believe it allows my model to better handle some of the stronger patterns as the outliers will not cause the model to vary from the patterns which are present. This may actually help me in the private competition. There were also some missing values which needed dealing with regarding the difficulty and similarly to the time feature I made an informed decision as to what I think they are by having regard to the surrounding feature values.

Some dimensionality reduction was necessary as many of the features present after the merge I believe are redundant and do not give value to my model picking up real patterns from the dataset. I have removed the following features as I don't think they give any value to my model classifying the time a track will take to complete:

- Introduction
- Date loaded
- Has Alerts
- Walking and tramping webpage
- Data loaded to GIS
- Introduction thumbnail
- Object ID
- Name

<u>**Initial design of system**</u>

My initial design used the classifier NaiveBayes. I thought this would be a good first choice for a predictive model as it would use probabilistic inference mechanisms to predict the times for completion of each track. This method also gave me rather good classification accuracy in the CORE competition. However I am sceptical as to how the NaiveBayes method will perform with some more complex features and classification. The fact the probabilistic model only had to choose from three different time classes in the CORE may have worked to its benefit. Whereas now the probabilistic model has to classify many more different time classes and may struggle as a result.

Furthermore, Instead of taking the average of all instances to fill in the missing values, I made intuitive decision as to what the values should be. I believe this will give me better overall instances for the model to evaluate and subsequently use to predict future instances time class.

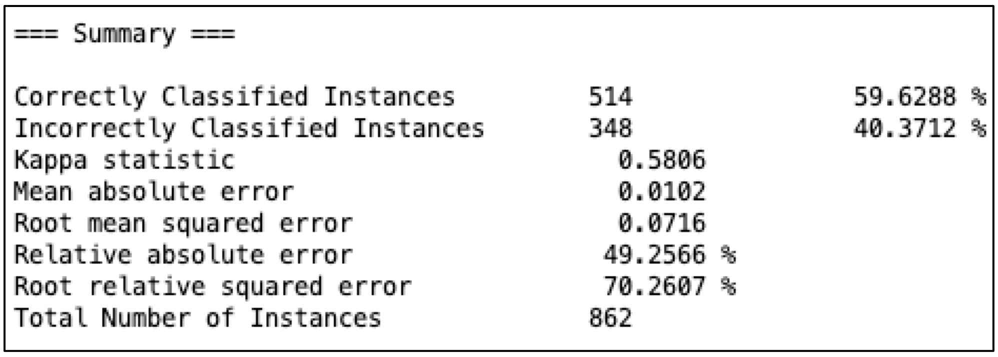
My initial designs submission using the NaiveBayes classification method gave a mean squared error value of **560.88**. This is not very good prima facie looking at the leader board. I believe I was correct in assessing that the probabilistic model would struggle more given the array of classification values there are (many track times as opposed to track time classes).

Discussion of the future designs

J48 design

I decided to use the J48 classification model after the NaiveBayes because I thought it would give stronger classification accuracy by using the divide and conquer methodology. This is because the divide and conquer strategy would be able to give paths to all difference time classifications based on their other feature values. This is unlike the probabilistic model NaiveBayes as that will only classify something if it is likely (in probabilistic terms) and subsequently may miss some of the more intricate patterns in the dataset. I ran the model on the training set and it gave me rather promising classification accuracy. Below in 2.2 we can see that the classification accuracy is 59.6288% which is a significant increase from the NaiveBayes classification accuracy which was sitting around 17%.

2.2 J48 results regarding classification of time to complete a New Zealand track



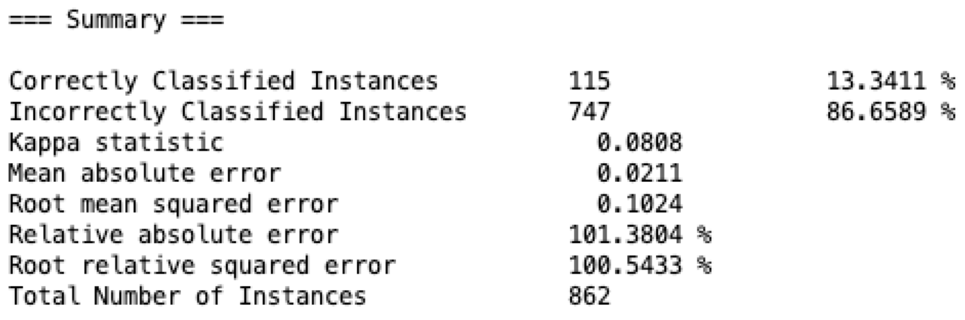
```
=== Summary ===

Correctly Classified Instances         514               59.6288 %
Incorrectly Classified Instances       348               40.3712 %
Kappa statistic                          0.5806
Mean absolute error                      0.0102
Root mean squared error                  0.0716
Relative absolute error                 49.2566 %
Root relative squared error             70.2607 %
Total Number of Instances              862
```

I then derived the predicted values from the test set and submitted them to the Kaggle competition to see how they would do. The mean squared error for this submission was **457.45**. This is a significant improvement on the previous model. However, I suspect that the model has been overfitted to the training data as the result does not relay the great classification accuracy received after training.

SMO Design

I do not wish to have any overfitting as I am only interested in picking up the real underlying patterns. This is because I want my model to be consistently successful on new data rather than just the particular training set. Because of this I have yet again decided to abandon the previous classification technique as divide and conquer did not identify the underlying patterns as well as I would like. This time I have chosen to use the classification algorithm SMO. I have chosen this method because the Analogizer tribe method all use constrained optimisation. Because of this characteristic I believe that this may lead to better pattern recognition by the model, instead of fitting only to the particular training set.

2.3 SMO results regarding classification of time to complete a New Zealand track



```
=== Summary ===

Correctly Classified Instances         115               13.3411 %
Incorrectly Classified Instances       747               86.6589 %
Kappa statistic                          0.0808
Mean absolute error                      0.0211
Root mean squared error                  0.1024
Relative absolute error                101.3804 %
Root relative squared error            100.5433 %
Total Number of Instances              862
```

As we can see the classification accuracy has significantly dropped off compared to the accuracy given by J48 (2.2) after training the test set. However the mean squared error for this submission was **398.14**, which is significantly better than the scores given earlier. I believe this is because the SMO algorithm comparatively to the other methods has picked up the underlying patterns more readily. Even though the model did not perform well on the training set it has correctly identified the patterns which correlate to classifying new instances closer. This submission made me jump in the leader board, which I hope is representative of a great predictive model for new track instances.

MultilayerPerceptron Design

I wanted to see how the dataset would be handled by a classifier from the connectionist tribe of artificial intelligence so I decided I would submit some predictions using the MultilayerPerceptron. This classifier will use a neural network to classify the correct times and by using this the model will creates many connections between features which it believes to be patterns. Because of this it may pick up on many of patterns contained in the dataset. However the mean squared error value of **457.45**, which comparatively to the other classifiers is not very impressive. I think the reason the classifier may have struggled on the test set is because it picked up on a vast amount of intricate and specific patterns for the training set which do not correspond to the test sets instances.

Which design is the best and why?

I believe that the SMO design is the best model for predicting the time completion of any given track. This is because I believe the model picked up on patterns which are relevant to all instances of tracks. This ensured the model was not overfitted and subsequently performed well on the test set. I believe my other designs similarly to the SMO design identified relevant patterns for all track instances, however they also identified patterns which are only relevant to the particular training sets instances. This caused those designs to perform relatively worse as they were overfitted.

SMO as a classifier I think performed better than other designs because the nature of the classifier is to identify the margin (difference) between two classes using a support vector. Rather than just trying to identify patterns between the given features like the Multilayer Perceptron does, SMO finds what is different about each class by looking at what features constitute them. Because of this SMO was the best design because of the reasons above.

Notes about Completion and handling of Labelled Data

I thought about identifying key words in the introduction feature, which is a feature consisting of labelled data. However, I did not believe I had the skills to deal with this type of data (labelled) at the present time. I will still however inform you on how I would have gone about it. From the key words I would have created a dictionary which a word processor would use to read through each track instances introduction to give them a vector of binary values for whether the words appeared (YES=1/NO=0). This would have allowed me to give values to the models which I think would be extremely influential in correctly predicting the time it takes to complete a track. This is because tracks which are described as steep or gentle in the training data would correlate to longer or shorter track times respectively. Consequentially the model will predict the track time more accurately.

Reflecting on your findings

How easy is it to interpret my chosen model?

I believe that my final chose model is rather easy to interpret even for the lay person. This is because all of the features I have selected to use are of a kind which any normal person would be able to associate with corresponding to a particular time. For example if a person sees two instances of track which have the same length, but one is described as 'Advanced' and the other is described as 'Easiest' they will easily be able to deduce that one will take longer that the other. This in itself shows how simple some of the patterns are in this dataset, which are then being identified by my machine learning models to predict the actual time it takes to finish a track. Furthermore the X and Y features are easily understandable for why they are used in predicting track times. This is because any coordinate correlates to specific type of area which may cause track times to increase/decrease.

Ethical consequences resulting from feature selection…etc

There are certain biases within the features presented which I was unable to alleviate. For example the features used to predict the track completion time has no regard to the weather of that particular track/area. Because of this the model assumes that the weather is normal or at least fine walking conditions all the time. This is something the officials should be worried about because the weather of a walking track may significantly change the time it takes to complete it. As a result of this the times which my model predict can swing in both directions (smaller/larger) times if the weather is either great or terrible for walking. This may also give DOC rangers false reasons to worry regarding someone who is walking a track and is taking longer due to weather. Because of this DOC's human resources may not be allocated efficiently, if they rely solely on the models predictions.

This model also presents some bias towards walkers with no physical impairments. The model has no regard to walkers with impairments such as a replaced hip, sprained ankles… etc. Because of this officials should be warned that they ought to know the physical capabilities of the people who walk the tracks. This is so they can identify whether the times predicted by the model are reasonable given the persons circumstances.

Both identified biases identified (Weather/physical ability) are examples of ethical consequences of how the specific features used in my model make predictions.