Assignment 1 Part 2

Student ID: S3755729

Student Name: Jialun Darren Huang

I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my submission. I will show I agree to this honor code by typing "Yes": *Yes*.

Task 2

Note: Data modelling will be different for whatever dataset is given. For my dataset, it will be based on Assignment 1 Part 1, where all errors have been fixed and some rows of data are removed.

Feature Engineering & Model Selection
Feature Engineering

To build our classifier, firstly, we must firstly choose the right model and features. As our survey does not take in any further data if they have not seen Star Wars, we must remove these data from our dataset.

Instead of creating a classifier for those removed data, we can simply create a prediction of the number of people who have seen Star Wars, for example, based on each location. This will give us some analysis on which location is the most popular with Star Wars.

Moving back, we have to next, declare our features and target features. The survey questions on the attitude and opinions on Star Wars movies are the features, and we want to guess what the people's demographics are. These will be our target features. We will have one training, validating and testing set per target feature so that we can safely determine the relationship between the survey questions and people's demographics. For this case, we would have 5 different target features, and each target feature will have its own specific model.

However, not all features are useful and redundant features may alter the model negatively, resulting in poor modelling prediction. We have to remove features that are not useful to our model.

There are 2 ways to choose the right features for our model. One of which is by asking an expert in this domain to tell us what the more important factors are, or questions in this case. Since I am poor and have no money to hire an expert, I will determine the features based on my own judgement.

Do you consider yourself to be a fan of the Star Wars film franchise?

This question is seeming relevant but based on the data, the proportion or percentage of respondents who are and are not fans may be similar across all choices in each demographic. For example, looking at location, it is impossible to determine which location a person can be if they answer No as the percentages for Yes and No are quite similar within all regions. Gender, Age, Household Income and Education have some choices that have a higher percentage in difference compared to other choices and this question may help identify the respondent's demographics.

Therefore, this question is relevant, although it does not clearly distinguish people demographics easily. We will still take this question as one of the features for our model.

Which of the following Star Wars films have you seen? Please select all that apply.

This question is no doubt important as it affects the other questions on their ranking of the Star Wars films and their liking for each character. It is impossible to rank something or like a character highly without watching or knowing them.

However, as the data has it in Strings, it may be difficult for the model to understand labels. One way to fix that is by using One Hot Encoding, which changes the labels into numbers and for this data, all movies that have been seen before will be marked as 1, while movies not seen before are marked as 0.

Please rank the Star Wars films in order of preference with 1 being your favourite film in the franchise and 6 being your least favourite film.

Just like the previous question on "which films have you seen?", this question is important for the same reason. However, the problem is that it is tough for the model to understand rankings of films. We can use one hot encoding for the data, for all 6 columns. In the end, we should have 36 columns, 6 columns for each film, from the first to the last (6th) rank. Episode1_Rank1, Episode1_Rank2, …, Episode1_Rank6, Episode2_Rank1, …

Please state whether you view the following characters favorably, unfavorably, or are unfamiliar with him/her.

This question is useful as there are many different opinions on all characters. However, it may get messy if the data has too many different results for each demographic. We should take note of this. Similarly, by using one hot encoding, we can generate how the respondents feel for every character in numbers.

Which character shot first?

There are a lot of different percentages for each choice for each demographic. This would be a good question to use as a feature. Just like the above, we will use one hot encoding.

Are you familiar with the Expanded Universe?

Location may not benefit from the column as much as the other demographics as the choices have about similar percentages for those who are familiar or not familiar. Other than that, the rest of the demographics can benefit from this feature.

Do you consider yourself to be a fan of the Expanded Universe?

The problem with this question is that only respondents who are familiar with the Expanded Universe (the previous question) can answer this question. Therefore, this feature should not be included in the model.

However, just like the first survey question, we can use this information to see how, in general, respondents feel about the Expanded Universe, and as we can see from each demographic, most of the respondents do not like the Expanded Universe. There is a small percentage of respondents who are fans, but there aren't many.

Do you consider yourself to be a fan of the Star Trek franchise?

Overall, all choices in each demographic have a different percentage and this feature with the rest of the above will make it easier for the model to determine the respondent’s demographic.

With these theoretical concepts is not enough as there could be hidden or unseen patterns that are not obvious. We can sklearn feature selection libraries and by using the feature selection scores, we can see which feature is more important than others.

Sklearn feature selection library has many methods that can be used. SelectKBest() is an example, which takes a number of features based on k, a parameter.

Another method we can use is hill climbing. The idea is to brute force each feature into the model with the same dataset split, and determine which features are good for data modelling if the metric score is better.

For this example, we train a model based by splitting the whole dataset into 2 parts, training and testing data. However, we have to make sure that we have random_state = 1 as we want the same exact training and testing dataset and only one feature is selected. Next, we pick our main model to be used for hill climbing. Lets say knn model. We train the knn model with our training data and use our trained model to predict the testing data. We can use Mean Square Errors (MSE) as our metric. Next, with the same exact training and testing data (based on random_state = 1), we create our model but this time, we add another feature. We check the MSE and if the MSE is lesser, this means that having this feature improves our prediction, and we should have this feature. However, if MSE increases, then we should not add this feature. We repeat this step for all features in the Star Wars survey, until all features have been tested.

With what we know and what sklearn tells us, as well as the concept of hill climbing, we can determine the best features to use for our model.

Model Selection

The 2 models we will be focusing on will be K Nearest Neighbour (KNN) and Decision Trees (DT). There are 2 ways we can determine which model is best suited for the data. One way would be to see which model is more suitable based on theory by comparing the advantages and disadvantages of both models. We choose a model based on our own knowledge to the data we have. However, this may not be the best option as there may be unseen circumstances based on the data and this may result in poor evaluation while selecting a model.

Another way would be to calculate the predicted success rate on both models and compare them. However, you must run training and validating on both models, you have to only check for every parameter that can be tweaked. This method would take a lot of time and computing power.

Training and validating each models

Firstly, we split the data into 2 sets, training and testing sets, having 80% and 20% data from the dataset, respectively. Make sure that we set random_state to a value(e.g. random_state = 0) so that we would not have multiple predictions, which can affect our

analysis since we do not know the exact data in each dataset which is causing such problems.

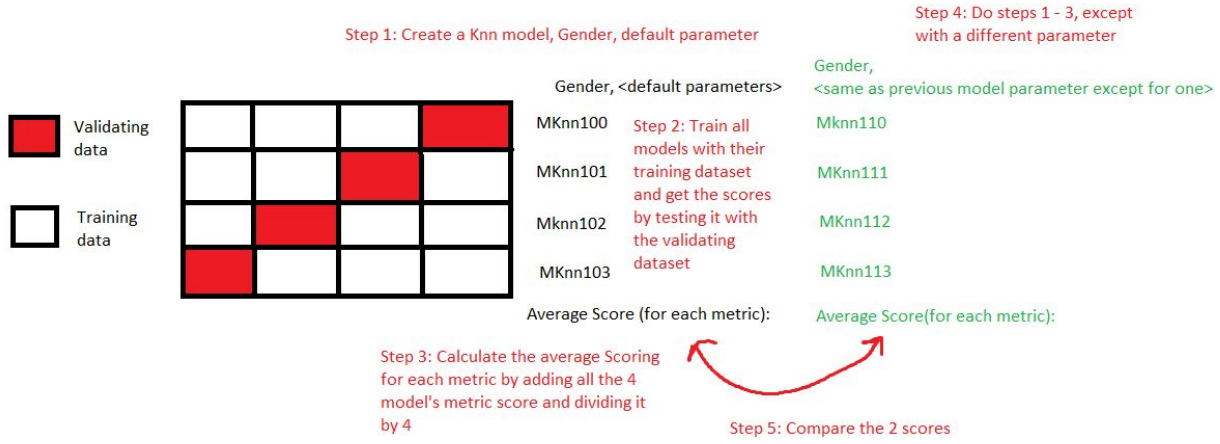
Instead, for our training set, we can split it into training and validating sets, using k-folds cross validation. We have to split the folds such that each fold has the same number of data. Since we have 668 data in the training data, we should split into 4 folds, each fold having 167 data (Factors of 668 are 1, 2, 4, 167, 334, 668). We will use 3 folds (training folds) to train the model and the last fold (validation fold) to validate how well the model can predict the validation data. Since there are 4 folds, we need to run 4 different experiments, with a different validation fold in each test.

For each model, we will train the model and check how well our model can predict based on the validation data. We will use a confusion matrix and classification report to get our results. The confusion matrix shows the number of instances the model has not predicted well and the classification report shows us the classification score for the model. It uses precision, recall, f1-score and support as a scoring system for the model.

Determining model parameters (Using a loop)

To begin, we will have all models begin with default parameters. After getting the confusion matrix and classification report, we will begin tweaking each parameter in both models, one at a time. Ultimately, we want to tweak our parameters until we get the best result in both models. By using k-folds cross validation, we will take the average scoring and take note of the number of instances the model has not predicted well from the confusion matrix for each model and its tweaked parameters.

Below is an example of how to create a knn model with training and validating data, with the demographic, Gender, as our target feature. This same example will also be used in DT modelling.



Comparing models

To make it simpler, if by increasing a specific parameter further reduces the score, it can be assumed that increasing it further would reduce the score even further. We can safely say that the parameter is best at its current value and we should move on to the next parameter. We may also cross check with what we know (theory) about each model. For example, for KNN, we know that the model may predict better if the parameter “weights” is set to distance. However, we must still have evidence (calculations or/and graphs) to support our theory.

As we have many scores to focus on, I would propose that the F1-score be made as priority to determine which model is better, since the F1-score has both the combinations of precision and recall. However, if both models that are being compared have the same F1-score, then the winner would be the simpler model (i.e. lower parameters), as less complex models are always preferred. The current parameter must still be checked for its next value, in case there is a better value for this parameter.

We should have 10 models in the end, 5 from KNN and the other 5 from DT, for each demographic.

Applying testing data to each trained model

After this, we would have the best models for both KNN and DT, for each demographic. Even though one model may be better than the other model based on the current score we have, we should base the score from the testing dataset (168 data) instead. Since the testing data is unseen data, it has not been seen by the model before and this gives a fairer judgement as to which model is better. We will pick the best model based on the confusion matrix and classification scores from the testing data. Likewise, we will priortise F1 score as our main score to determine which score is the best.

An important note to consider is, there is a possibility where KNN may be better than DT in some demographics, and vice versa.