Practical Data Science with Python COSC 2670/2738 Answers to Assignment 1 Part 2

Student ID: s3726377

Student Name: Ji Hei Kim

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

The goal is to predict people's demographic based on their attitudes or opinion about Star Wars movies. Different classifier is built to find the most predictive modelling.

Feature Engineering and Model Selection

Assuming that prior data preparation and cleaning data has been completed, one of the first things required in feature engineering is to reduce the dimensionality of data set. Not all features or values are essential to predicting people's demographic. It was needed to select the relevant ones that contribute to the prediction of results. Features like Respondent ID can be considered as feature that does not help in predicting people's demographic. There was a feature where the information was not directly related to attitude toward Star Wars movies. It is my assumption that column like 'which character shot first' is unlikely to be useful for learning purposes and, in fact, can be detrimental due to the large number of distinct values they contain.

I will create a new feature to help the prediction, categorizing content variables in a meaningful way. From the *Which of the following Star Wars films have you seen?* we can categorise respondents to 'Seen all episodes?' with Yes/No answer. Ranking movies can be grouped as Most Favourable Ep, Least Favourable Ep. Where 1 being the most favourable, episode that has been ranked 1 will be present and ranked 6 in Least Favourable. For example, if respondent selected Ep1(1) as Most favourable and Ep 2(6) as least favourable, those two will be noted. Ranking characters can be dealt in a same way: Most favourable character and least favourable character.

Now with selected features, the data type of each features should ideally be converted to numeric variables. Yes/No can be converted to 0s and 1s. Episode chosen will be mapped as episode numbers 1-6 and characters will also be mapped from 1-14 (Han Solo – Yoda). Where features are on different metrics scaling data is required otherwise much larger scale in one feature will be problematic.

Without further calculations it was impossible to found out which features were best. As all features represent respondent's attitudes/opinions all features selected can be used to classify/predict each people's demographics. However, to avoid outfitting I have chosen Yes/No questions to predict people's gender and age. To classify and predict people's income and education I have chosen ranking features and to predict people's location I will use all relevant features.

I have chosen Decision Tree as a possible modelling techniques to classify gender and age. The reason is that Decision Tree will allow to segregate respondents based on each features and helps to identify which variable creates the best homogeneous sets of respondent. As the features only contain binary values, I will be using Gini index which will split based on the most homogenous features.

As the size of the data is not too big, I will use KNN to classify and predict respondent's income, education and location. Just like Decision Tree, KNN commonly used for classification as it is very simple, easy to implement for multi-class problem and works with numerical values.

Training the Model

Decision Tree – Gender & Age

Before training the model data will be split into two parts, training set and test set. Training set consist of records where target variables, Gender & Age is known. Python has its 'industry-ready implementations' which help of few code lines allows to split data between test set and training set. By default, Python splits 75% as training set and 25% as test set which I think is reasonable for Star Wars survey data.

As Decision Tree accuracy is heavily reliant on reasonable splitting, Gini index will be used to found out which features in order to split for best homogeneous set. The order of which features to split may be differ in Gender and Age. To avoid overfitting I will need to set some parameters such as maximum number of features, minimum number of splits and minimum number of samples required. Setting these parameters requires for me to repeat steps to tune for right numbers. This will also vary between predicting Age and Gender.

KNN – Income, Education & Location

As mentioned above, training model for KNN will require to split data into two, training set and test set. Just like how I set the training set with Python, same line of coding is required to set the train set and test set. Default values (75% vs 25%) will be used again for Income, Education and Location.

For KNN the key point is to train who is the neighbours. The neighbours are taken from a set of Star Wars survey data for which the class is known. This can be thought of as the training set for the KNN. Star Wars survey data set has been already converted to feature vectors, the training examples are vectors in a different feature each with a class label (Most Favourable, Least Favourable etc.). As previously mentioned, to predict people's location all relevant features will be used. Which means that the values should be scaled equally to train the data in KNN. The training phase consists storing the feature vectors and class labels of the training samples in memory.

Model Validation and selection

KNN & Decision Tree

Trained classification model must be validated to see if the right (best) modelling technique has been used. For all classifiers created I have built, validation will be made using k-fold cross validation so that all given data is used. K-fold cross validation is used to found out classification accuracy with minimised bias comparison. It allows to take the validation to a different level by splitting the survey data into smaller sets 5 folds (anywhere between 5-10 should be reasonable). Of the 5 sets, a single set is retained as the validation data for testing the model, and the remaining 4 subset are used as training data. This process will be repeated until every K-fold is served as the test set (5 rounds). Once the 5 scores are noted I can get an average scores which will be performance metric for my chosen classifiers.

Based on the result, if the score is too low this means that classifier built has weak predictive powers. Which than I will need to create another classifier using different modelling techniques to until I can get highest score and choose that model. For example, I've used KNN to predict people's income,

education and location. I can build Decision Tree to predict the same demographic and see which one returns higher score. Model will be selected based on the highest validation score.

Applying the trained model to unseen future data

Decision Tree & KNN

As the models are ready, it is time to apply the selected model to unseen future data. To apply new data into the model, I am required to prepare an input data set that matches with feature values defined by the classifiers. Which means with a new data set, I will need to re-do the data preparation to change the new input values to numerical variables that are equivalent to what have been used to build the model. Scaling is also required for classifier of predicting Location.

For KNN, the output will be a class membership. Where the object being assigned to the class most common among its k nearest neighbour, result will be the assigned class of that nearest neighbour, i.e. respondents' certain income group, education level or specific location.

For Decision Tree the output will be predicted variable, the last terminal node, which will tell whether respondent is Female or Male or certain age group.