Can Humour Styles be used to predict Gender?

AI: <COMP2261>
Date: <24/01/2021 >
Submitted as part of the degree of [e.g. MEng Computer Science] to the
Board of Examiners in the Department of Computer Sciences, Durham University

Abstract—The aim of this study is to see if the Humor Styles Questionnaire can be used as a predictor of gender. This investigation tries to figure out this phenomenon by using the Humour Styles Questionnaire(made by Martin et al 2003) which distinguishes positive and negative humour styles. We will use a data set made from the responses of the questionnaire which consisted of 1071 entries and 39 categories one of which was gender. To see if our prediction is true, we examine and implement several machine learning algorithms, such as KNN, Logistic Regression and SVM. When analysing the accuracy scores to measure the models' performances, the results proved the hypothesis that humour styles could be used to predict gender to an extent.

Index Terms—data, model, predict, algorithm, train, humour styles, gender

1 Introduction

The debate of how different minds are influenced by gender is something that is still studied to this day. In the following report we will assess whether humour styles can be used to predict gender using a data set from the results of the Humor Styles Questionnaire made by Martin et al in 2003. The answer to this could help psychologists and researchers understand if gender differentiates our personalities and mind which can help assess more predictions in the future. The main objective of the questionnaire was to determine if we can differentiate positive(self-enhancing and affiliative) and negative(self-defeating and aggressive) humour styles and how this impacts relationships, mental health, and physical health. Therefore, we will be formulating our question into a machine learning task by inputting the four humour styles using a publicly available dataset of the participant's responses, to see if they are effective for predicting gender. Then we will use exploratory data analysis to construct and analyse visual data graphs to clean our data and then input this data into Python to train the model. We then use predictive analysis algorithms to see if we can predict gender and how much of an impact each of the humour styles have on gender. To evaluate the models created and see how we can improve further, we will compare the different accuracy scores obtained from predictive algorithms used on our dataset.

Data Set

The Humor Styles Questionnaire consists of 32 questions that are scaled from 1 to 5 where 1 is rarely true, 5 is always true and -1 is inputted to the dataset when a question is left unanswered. The statements are then categorised into four humour styles, 8 for each subscale: affiliative (making others feel better), self-enhancing(making oneself feel better), aggressive humour(harming others) and self-defeating (harming oneself) [1]. The participants were then prompted to add their age, gender and how accurate they believe their answers were. This data was then stored in a CVS (comma separated value) file, and we will use this when analysing our data. For data analysis, we will need to assess what types of data we are working with, as they need to be passed into the algorithm in the correct format. There are three types of features: Categorical, Continuous and Ordinal features. Categorical features can be identified as those that have more than one category. Therefore, gender can be identified as a category. Ordinal categories are very similar to categorical however the categories can be ordered, which perfectly describes our Qs columns. A continuous category is where you can take any numeric value. In this case accuracy, age, and the four humour styles are continuous. First, we will show a table (figure 1) to present statistical measures of the data such as count and mean.

	Q1	...	Q32	affiliative	selfenhancing	agressive	selfdefeating	age	gender	accuracy
count	1071.000000	...	1071.000000	1071.000000	1071.000000	1071.000000	1071.000000	1071.000000	1071.000000	1071.000000
mean	2.025210	...	2.838469	4.010644	3.375537	2.956583	2.762745	70.966387	1.455649	87.542484
std	1.075782	...	1.233889	0.708479	0.661533	0.410870	0.645982	1371.989249	0.522076	12.038483
min	-1.000000	...	-1.000000	1.300000	0.000000	0.000000	0.000000	14.000000	0.000000	2.000000
25%	1.000000	...	2.000000	3.600000	2.900000	2.800000	2.300000	18.500000	1.000000	80.000000
50%	2.000000	...	3.000000	4.100000	3.400000	3.000000	2.800000	23.000000	1.000000	90.000000
75%	3.000000	...	4.000000	4.500000	3.800000	3.300000	3.100000	31.000000	2.000000	95.000000
max	5.000000	...	5.000000	5.100000	5.000000	5.000000	5.000000	44849.000000	3.000000	100.000000

Fig. 1. Table of statistics of our dataset

2 Methodology

2.1 Exploratory Data Analysis

Exploratory data analysis is when we analyse our data and produce summarised statistics to see what it can tell us about the data, to check for anomalies and any assumptions made. From the table above we can assess a few things. Firstly, we can see that we have 39 columns, and our count shows that there are 1071 entries or rows. Though we would like to use as much data as possible, meaningless, or wrong data can be harmful to the accuracy of our model hence we will be cleaning the data.

2.2 Data Cleaning

When given a data set, many features and values may not be useful to input into our algorithm. If we have outliers, anomalies, or data that is redundant or irrelevant to us then there is a risk of bad results, overfitting and underfitting. These can be caused by a wrong entry or human error; therefore, we will be exploring the statistics from the table above so we can decide how we will clean our data to maintain quality. If we start looking at the age column, we can see that the max is 44849 which is greatly unlikely for age and would be the reason why the mean is so high at 70.97. As a result, we will remove any value above 100 as it would be unrealistic otherwise. We can also see that the minimum accuracy we have is 2% however only 25% of people put the accuracy as

below 80% and because of this we will remove data with any accuracy lower than 80% which also means that we can trust our data more. When looking at the question columns, we can see the minimum is '-1' and could benefit from replacing it with 0. While also looking at gender we can see that the mean is 1.46, which is between 1 and 2 and that 75% of people are male or female. Therefore, we will assess how valuable those who put "Other" are to our data by using a bar chart. By looking at our bar chart above we can see that 55% of our participants were male, 44% were female and only 1% were other. As only 7 people answered this category, we can state that this sample size is not large enough to explore and so we will need to remove any numbers that are not 1 or 2 from our data. Our final stage of data cleaning would be to remove any data columns that are not one of our inputs or outputs. In this case, we would drop age, accuracy, and all questions. Now we have suitable data for predictive modelling that we can see summarised in figure 2.

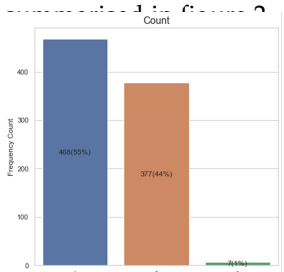


Figure 2: bar chart of gender count

	affiliative	selfenhancing	agressive	selfdefeating	gender
count	905.000000	905.000000	905.000000	905.000000	905.000000
mean	4.052597	3.398564	2.964309	2.759006	1.447514
std	0.702946	0.660524	0.392007	0.651826	0.497513
min	1.300000	1.000000	1.000000	0.900000	1.000000
25%	3.600000	3.000000	2.800000	2.300000	1.000000
50%	4.100000	3.500000	3.000000	2.800000	1.000000
75%	4.600000	3.900000	3.100000	3.100000	2.000000
max	5.100000	5.000000	5.000000	5.000000	2.000000

Figure 3: Table of our cleaned data set statistics

Now by looking at figure 3 we can assess our humour styles a bit more. From the first column we can see that affiliative had the highest mean score with a low standard deviation which shows the majority scored high in this category whereas self-defeating had the lowest mean showing us more people have positive humour styles than negative. This can be explained by the type of questions; "I laugh and joke a lot with my closest friends" would rarely be answered negatively.

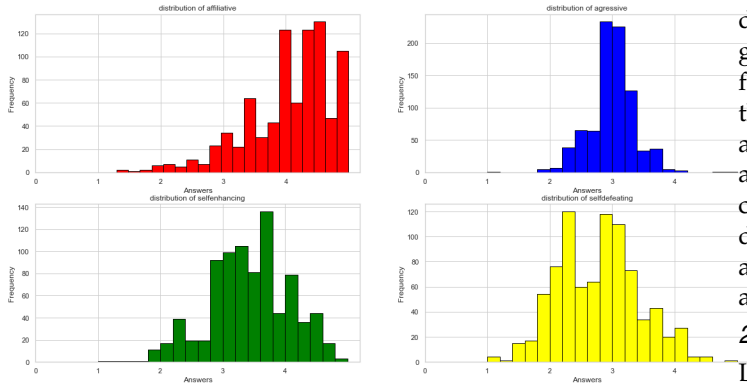


Figure 4: Distributions of our 4 humour styles

As we can see from the distributions in figure 4, though the negative humour styles tend towards a normal distribution, the positive humour styles cannot be seen as a normal distribution. Does this mean gender can have an influence on humour styles? We might be able to investigate this further with the bar chart and correlation matrix below.

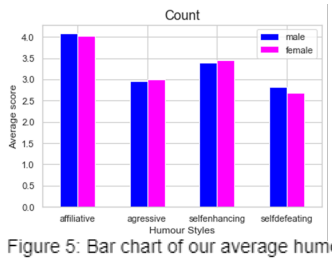


Figure 5: Bar chart of our average humour style scores split by gender

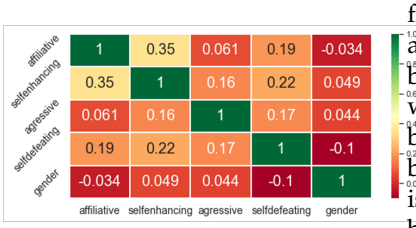


Figure 6:Correlation matrix of our inputs and outputs

From figure 5, we can see that male and female characteristics do not differ very much and that both seem to have the same humour pattern. Next we use a correlation map to be able to compare features. When two features have a positive correlation as one feature increases so does the other. This ranges from 0 to 1 where 1 is the best correlation. A negative correlation is when an increase in one feature leads to a decrease in another. This ranges from 0 to -1 where -1 means a perfect negative correlation. As there are no perfect correlations we do not need to worry about redundant data. From figure 6, we see that self-defeating has a higher correlation with the other 3 humour styles and self-enhancing has the highest correlation with affiliative. Our best correlation for gender is with self-defeating humour styles however, we do not see a relationship between gender and any humour styles but would need to confirm this by modelling and training our data.

2.3 Data Preparation

The first thing we need to do before using predictive algorithms is to split and train our data. We need to separate our data into x and y where x is our independent variables (our four humour styles) and y is our dependent variable (gender). We will then split our data into train and test data where the training set will be 80% of the data and we will use the last 20% as our testing set. From this it is split into 4 parts: x_{train},y_{train},x_{test}, y_{test}. Our x_{train} and y_{train} are our inputs and outputs that are used to train the algorithm, our x_{test} uses the trained algorithm to predict the result and our y_{test} is used to compare our predicted result. When splitting our data, we will also set the parameter randomstate to an integer as this ensures the same set of trained data is used every time.

2.4 Algorithms

Next, we would like to consider three algorithms for classifying our model. All these algorithms use predictive analysis, which uses computational methods to find patterns in large datasets and this will be implemented to validate and predict gender. To choose our algorithms we need to evaluate different factors such as the type of problem we are trying to solve, the data types involved and how much data we have. Firstly, as we have manually trained our data, we need to use an algorithm with supervised learning. Secondly, as we are using continuous features to predict a binary feature, we can narrow down our options to a few classification models. Therefore, all the algorithms below follow the two requirements stated above.

2.4.1 Logistic Regression

Logistic regression is a popular choice as it is a simple algorithm that is used when we need to predict a categorical and binary variable. When predicting the outcome, it gives the probability that the value belongs to the positive class hence when the result from the algorithm is larger than 0.5 it returns 1 or true.

2.4.2 SVM

Support Vector Machine is an algorithm that solves hard classification and regression tasks. The algorithm starts by defining an n -dimensional space with support vectors, where n would be the number of distinct features and our support vectors would be our trained data. It then constructs a hyperplane by dividing our two trained classes by the largest margin between them. When new values are fed into the algorithm, it is assigned to one of the classes. Our algorithm is also defined by kernels and in our problem, we will assign it to a Gaussian

Radial Basis function which works on non-linearly separable data points.

2.4.3 KNN

This algorithm is a non-parametric algorithm that is used when you want to assign values based on similarity. It sets all data as points and stores the most similar points close to each other. When new data is received it uses a distance metric to find the k nearest neighbours. It will then be predicted and assigned based on the most common vote of its k nearest neighbours. Ie if k =1, it is classified by the category of its first nearest neighbour. [2]

2.5 Feature Scaling and Fitting

Next, we will need to work on feature scaling and model fitting. If we do not perform feature scaling before training and testing it can cause data leakage. For example, if we can see that one feature has a higher magnitude than others this will affect the model as these features will have a higher weight than others. We will use the standard scalar function for our model as it is the preferred scalar for our chosen algorithms. This scalar, brings all the features to a mean of 0 with a standard deviation of 1. We will apply it to x_{train} and x_{test} separately instead of on all the data. After applying our algorithm, we will need to fit it to our model for the training process. To do this our code finds parameters for the algorithm from the training data set. For standard scalar these parameters are mean, standard deviation. It then classifies values using a prediction function. We fit and transform our training data to scale it and to find the scaling parameters of the data. The testing data is then transformed only, using the mean and variance found. [3]We will be comparing models using accuracy scores. Accuracy is the measure of how many times the model makes correct predictions by considering both the output of our algorithm and the actual data with equal weight. Figure 7 describes the features of our models and the accuracy of our outcomes:

Model	Parameters	Accuracy: default values	Candidate Values for Hyperparameter tuning
Logistic Regression	**C:** the inverse regularisation strength **max_iter:** maximum number of iterations taken for solvers to converge	0.574	**C:** np.logspace(-4, 4, 20), max_iter: [100, 1000,2500, 5000]
Radial SVM	**C:** regularisation parameter that controls the penalty strength of misclassification. As C increases the model overfits **Gamma:** how much a single training example weighs. Small mean a lot large mean not very much. As gamma increases the model overfits	0.568	**C:** range from 0.1 to 15 in 0.1 increments **Gamma:** ['scale','auto']
KNN	**n_neighbours:** number of neighbours **Leaf_size:** speed of construction and query as well as memory required to store the tree **P:** power parameter for metric **Weights:** distribution of weights **Metric:** distance metric	0.556	**n_neighbors:** integers range(1,31) **leaf_size** : integers range (20,40) p: (1,2) **weights:** ('uniform', 'distance') **metric:** ('minkowski', 'chebyshev','euclidean')

Figure 7: Table showing the parameters, accuracy and candidate values of parameters for each algorithm

From our table above there could be many reasons as to why the accuracy is so low and this can be down to overfitting and underfitting. Underfitting is when the model has a low accuracy on both training and test data due to the algorithm not being able to learn the complex patterns from the training data. This generally happens with linear algorithms or when we have too little data to build an accurate model. We can tackle underfitting by increasing the amount of data, reducing noise and the duration of training. Overfitting is when the model has a high accuracy on the training data but low accuracy on the test data due to trying to cover more than the required data points in the data set. This is usually because nonlinear algorithms try to take in features that reduce the efficiency and accuracy of the model such as caching noise or using inaccurate values. [4] Overfitting has a low bias and high variance, whereas underfitting has a high bias and high variance. For example, from figure 8 we can see that the KNN model is currently underfitted. The lines on this graph demonstrates that there may not be sufficient information to build a model:

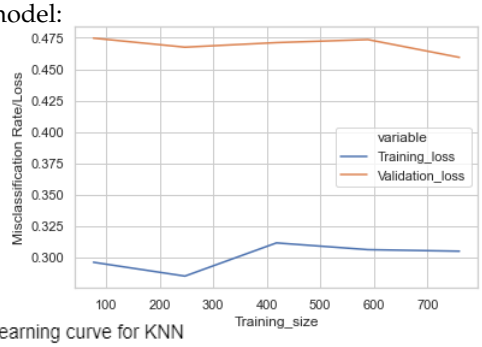


Figure 8: Learning curve for KNN

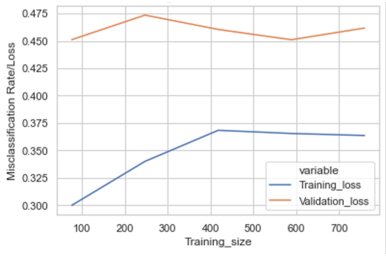
We can solve this by hyperparameter tuning, cross validation and ensembling.

2.6 Hyperparameter tuning

When running our algorithms, they use default parameters, but we can manually change these to get a better model. These optimal values are known as hyper-parameters. To find these we will be using hyperparameter tuning which uses several different trials on the same training data. We then run through different candidate values for parameters of a model to find the ones that give us the best average performance. For this we use a search algorithm called grid search to go through the candidate parameters declared in figure 7 to find the best score. We evaluate these by comparing cross validation scores. [6]

2.7 Cross Validation

Though our accuracy scores give us a good indication of how good our model is we can't assume that the same training data will be used each time, it is likely to change. This is known as model variance. To combat this, we can use cross validation to achieve a generalised model. It helps reduce variance, expends models' predictability by training on multiple train-test splits and can give us a better indication of how well the model will perform on unseen data. We will use k-fold cross validation. This method randomly divides the split data set into k folds or sections of equal size. Most commonly, k is taken as 10. One-fold is taken and the other 9 are used to fit the model. We will then repeat this k/10 times and calculate the average. This is also known as the cross-validation score. Figure 10 shows us the cross-validation scores when applying hyperparameter tuning. Now that we have fixed better parameters, we can see that the new learning curve graph in figure 9, demonstrates the effect of hyperparameter tuning and cross validation. This shows that our KNN model has improved to lean more towards overfitting, and we can tell this as the optimisation convergence no longer changes. This is known as the convergence of optimisation and can be a concern as when optimisation algorithms end, we cannot investigate anymore improvements for our algorithm.



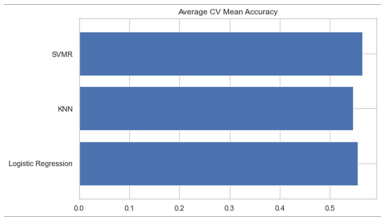


Figure 10:Cross Validation comparisons

Figure 9: Learning curve for KNN after hyperparameter tuning

3 Evaluation

Evaluation is very important as it is a core part of building an effective machine learning algorithm. To see how well our chosen algorithms perform we will compare our results from hyperparameter tuning using a classification report and confusion matrix as evaluation metrics. We use a confusion matrix on our test data to test the performance of our classification models. It gives us the number of correct (left diagonal) and incorrect predictions (right diagonal) made by our model. We can then use a classification report to give us several performance metrics such as accuracy, recall, precision and f1 score, to create a more detailed confusion matrix in the form shown in figure 11: [7]

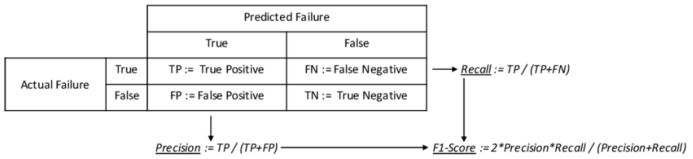


Figure 11: Correlation Matrix Template

Logistic Regression Confusion Matrix	Target		Hyperparameters: {'C': 4.281332398719396, 'max_iter': 100, 'penalty':	
	1	0		
1	12	61	Positive Predictive Value	0.84
0	15	81	Negative Predictive Value	0.16
	Sensitivity	Specificity	Accuracy = 0.55029586	
	0.57	0.44		

Figure 12: Correlation Matrix for Logistic Regression

KNN Confusion Matrix	Target		Hyperparameters: {'leaf_size': 40, 'metric': 'chebyshev', 'n_neighbors': 'p': 1, 'weights': 'uniform'}	
	1	0		
1	25	48	Positive Predictive Value	0.76
0	23	73	Negative Predictive Value	0.34
	Sensitivity	Specificity	Accuracy = 0.57988166	
	0.60	0.52		

Figure 13: Correlation Matrix for KNN

Support Vector Machine Confusion Matrix	Target		Hyperparameters: {'C': 15.0, 'degree': 5, 'gamma': 'scale'}	
	1	0		
1	26	47	Positive Predictive Value	0.74
0	25	71	Negative Predictive Value	0.36
	Sensitivity	Specificity	Accuracy = 0.5739645	
	0.60	0.51		

Figure 14: Correlation Matrix for Support Vector Machine

Though it is not a great increase we can see that our accuracy scores are better and that our best score is from the KNN algorithm though it has a lower cross validation score than the rest. We would now like to see if we can ensemble this model using the bagging classifier.

3.1 Bagging

Bagging works best for models that have high variance. It reduces overfitting by repeating the sample training data to create base estimators. The bagging classifier uses bootstrapped data set which runs algorithms repeatedly on the sample training data. These are then all combined to produce an average and the new data uses this to predict the output. Below shows the outcome:

Accuracy for bagged KNN is: 0.7512792899408284
Cross Validation for bagged KNN is: 0.5598739495798319

As we can see, bagging has increased our accuracy a lot and this is because it reduces the variance of the classifier and combines the outputs from various classifiers which are being bagged. [8] Now we can also plot a graph of the most important features our model used for predicting gender.

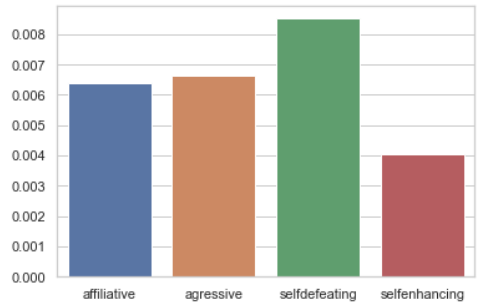


Figure 15: Feature of importance for bagged KNN

Figure 15 , we can see that selfdefeating contributed the most and self-enhancing contributed the least. However, we can see from our scales that they do not differ by very much, hence they are all considered important to our model

4 Conclusion

We wanted to investigate if we can predict gender based of people's humour styles and did this using machine learning. Our first difficulty was assessing our categories for data. As we have outlined, only 1% of people put "other" which was not a large enough sample for our model. As this was taken in 2003, there might be a possibility that 2 decades later less social pressure would increase this category enough to assess it again. Another major limitation of the approach used is that the accuracy was quite low. We could improve this in the future by looking into deep learning or neural networks to solve our problem. When completing the project, we learned how important it was to properly process, train and test our data to get the best results from our model. As we are better at understanding visual data, we produced graphs to analyse what data we will need to process. We have also learned how important parameters are and how changes in our parameter values can affect the accuracy of our model. We used grid search to help us find the optimal parameters for our models. When comparing the results from 3 different predictive algorithms we learned that cross validation gives us a better insight into which algorithm to use. From our confusion matrix, we have seen that KNN is the model that gives us the highest accuracy and to improve this score we used bagging and gained an overall accuracy score of 75.1%. This means that we can conclude that when using predictive algorithms, we may be able to predict gender to a certain extent.

References

[1] http://humorstudies.org/ 2021
[2] https://www.datasklr.com/select-classification-methods/k-nearest-neighbors 2021
[3] https://towardsdatascience.com/what-and-why-behind-fit-transform-vs-transform-in-scikit-learn-78f915cf96fe 2021
[4] https://medium.com/@itbodhi/overfitting-and-underfitting-in-machine-learning-models-76cb60dbdaf6 2021
[5] https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac 2021
[6] https://www.analyticssteps.com/blogs/what-are-model-parameters-and-evaluation-metrics-used-machine-learning 2021
[7] https://www.researchgate.net/figure/Confusion-matrix-illustrating-the-calculation-of-precision-recall-and-F1-score 2021
[8] https://www.educba.com/bagging-and-boosting/ 2021