Machine Learning: Predicting Gender from Humor Styles

Module Name: Artificial Intelligence – Machine Learning

Date: 17/01/2022. Name: Owen Shaw

Submitted as part of the degree of MSci Natural Sciences to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract — Machine Learning is the study of algorithms that improve over task T with respect to performance measure P based on experience E. There are a variety of different Machine Learning algorithms created for different types of data that can produce different results. We trained 3 different types of algorithms to predict gender from a set of data about a participant's humor style. The algorithms we trained were all classification types. The K Nearest Neighbours and the Ensemble Bagging Classifier with a base learner of Radial Basis Function Support Vector Classification both performed the best, resulting in average precision scores, a model's predictive power, of 0.57, and average recall scores of 0.58, when evaluating using the test set. The results indicate a weak correlation between gender and humor styles.

Index Terms—Machine Learning, Humor Styles Questionnaire, Classification, K Nearest Neighbour, Support Vector Classification, Sci-Kit Learn.

1 Introduction

The project was to build a Machine Learning (ML) model that can make predictions based upon answers to a Humor Styles Questionnaire (HSQ) [1]. The problem we are solving is to see if there is a correlation between a participant's humor styles and their gender and if the correlation exists, to make a predictive *ML* model from it. This problem is interesting as it can show a disparity between humor styles in men and women. The problem is a classification task: The inputs to the model are the 4 attribute scores from 1 to 5 (affiliative, self-enhancing, aggressive, and self-defeating) calculated from the 32 questions from the *HSQ*; the output is the gender that the model predicts the participant is. The questions were answered on a scale from 1 to 5, where 1 and 5 represented "Never or very rarely true" and "Very often or always true" respectively.

2 Methodology

Exploratory Data Analysis

The data obtained for each participant: the 1 to 5 answers from the 32 questions; the scale scores which were generated from the questions; the age; the gender; the accuracy, which was a score the participant gave indicating how accurate they thought their answers were from 1 to 100.

To analyse the data, we first attempted to determine which columns of data were useful and which were redundant. Since the answers to the 32 questions are summarized in the 4 attribute scores also included in the data, we concluded that they were not useful inputs for the model. It is beneficial to reduce the number of features where appropriate as the time and memory used for the *ML* algorithm used to build the model is correlated with the number of features that are being inputted.

Next, we inspected the remaining variables: age, gender, and accuracy. The distribution of age was heavily positively skewed, meaning that whilst there is lots of data for participants between 14 and 34, there is very few for above that. We inspected this by producing a histogram, after having removed the outliers [Fig. 1]. This may mean that a *ML* model may not be able to accurately predict values outside of where most of the training data lies due to a lack of data.

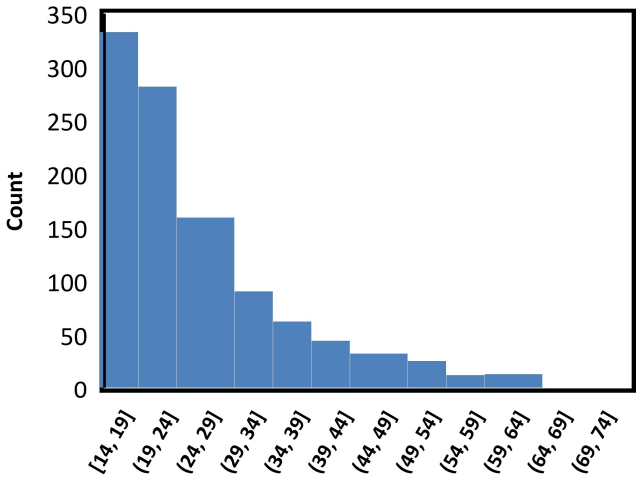

Fig. 1: The distribution of age across the whole dataset. It is heavily postively skewed. The mean is 26.3 and the standard deviation is 11.1.

For accuracy, we, again, plotted the data as a histogram [Fig. 2], which revealed that the data was heavily negatively skewed, like the distribution of age. This, for the same reasons as for age, would not be able to produce a good *ML* model. The data is also not significant, important, or interesting; in the *HSQ,* accuracy represents how accurate the participant thought their answers were on a scale from 1 to 100. Being able to predict this from a participant's humor style isn't a very interesting result, when compared to gender or age.

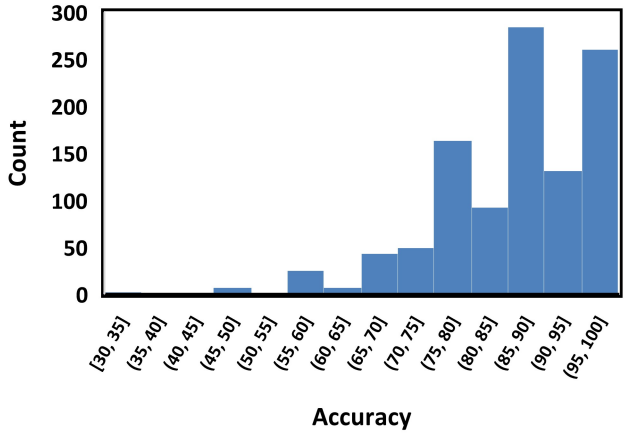

Fig. 2: The distribution of the participant's accuracy from the whole dataset, where accuracy is how accurate a participant thought their answers were. The distribution is very negatively skewed. The mean is 87.9 and the standard deviation is 10.7.

For gender, since there were only 3 options (1 = male, 2 = female, 3 = other), we looked at the count of each result. There were 581 males, 477 females and 8 in the "other" category. Since the "other" category was <1% of the data, there was an extreme imbalance of data. The "other" category only contained 8 data points, which is not enough to be able to make a robust *ML* model from. In addition to this, most classification algorithms are for binary classification. Because of these 3 factors, we decided to only consider male and females when building the model. The ratio of males to females is approximately 55:45, so the data was sufficiently balanced there (since the minority class is >40% of the data).

Data Preparation

To prepare the data we first had to convert it from .csv format to a pandas dataframe. To clean the data, we removed any examples which contained unwanted information; the participant had the options to submit "-1" or "0" to some questions to indicate they didn't know an answer to a question, or they didn't want their results to be included in any analysis, these examples were removed along with any other outliers.

Next, we extracted the necessary features and target columns from the dataframe. After this we split the data into training, validation, and test sets. To do this, we used 2 train_test_split (TTS) functions from the model_selection package from sklearn. The ratio of train: val: test is 0.64: 0.16: 0.2, by specifying a training size parameter of each *TTS* function of 0.8. The other parameters for each *TTS* function were: Random_state = 2, as this randomizes the order of the examples the same way each time the algorithm is ran, as the order is an irrelevant piece of information; stratify = "y" for the first split and "y_train" for the second split, which were the targets for each, this forces the splits to contain proportionally the same number of examples from each class, this is advantageous as it means that every sample we use is representative of the original dataset.

The final step of data preparation was to transform the raw data to NumPy feature vectors. To do this, we used the "to_numpy()" method. Since the nominal categorical data was already in integer format, nominal encoding was not necessary. To scale the features into similar ranges we used Standardisation (Z-Score Normalisation), which rescales the feature values to have zero-mean and unit-variance. We used this type of scaling as it is commonly used for the *ML* algortihms that we implemented, which will be explained in further detail in the following section.

Learning Algorithm Selection

To select the learning algorithms that we would implement we considered the types of data in the dataset. Since we were trying to predict an integer category where the targers were labelled, I needed to use a classification algorithm. Dimensionality reduction was not necessary and not possible as there were only 4 input features, and they were unrelated. As such, we needed a binary classification algorithm. The options were Linear Support Vector Classification (LSVC), K Nearest Neighbours Classifier (KNN), Radial Basis Function Suport Vector Classification (RBF SVC) [2]. When forming the *RBF SVC,* we experimented using this algorithm as a base estimator for a bagging classifier.

3 Main Body

Model Training and Evaluation

When training the models, we used the make_pipeline function from the pipeline package from sklearn [3]. The parameters for this function consist of the steps the pipeline is instructed to do. The pipeline is then fitted against the training data. The benefits of using a pipeline to do this step instead of doing the steps separately is that when evaluating the model, using the validation data or the test data, the steps, such as the type of feature scaling, do not need to be repeated in the code, as they are stored in the pipeline. There were no concerns about the convergenece of the optimization for a dataset of this size, however if the number of examples were orders of magnitude larger then there would be.

For all the pipelines, the first step is the StandardScaler function from the preprocessing package from sklearn. This feature scaled the data using the Standardisation method as mentioned earlier.

To optimize the models, the performance metrics used for each were chosen to maximise the overall accuracy of each model and to maximise the accuracy of each class, as a model which is very accurate for one class but not for the other is useless. The method that was used to obtain the optimum hyperperameters was either using a rule of thumb, which is explained in each case or a search by trial and error (grid search). Unless specified, when searching for a hyperparameter by grid search, we compared the cross-validation (CV) results for that hyperparameter and selected the option that gave the best result. For the cross-validation we split the data 10-fold (K = 10). K = 10 is the value for the majority of uses.

For the *KNN* pipeline, the second step is to input the data into the *KNN* classifier which has the hyperparameter of n_neighbours, known as the value of K, which specifies the number of nearest neighbours that the model should consider when deciding what class an example should be assigned to. This value is important and directly affects whether a model is underfitted/overfitted. The general rule for the optimal value of K is the square root of the number of examples in the training data [4], which in this case is 25, as only 627 examples are in the training set. However, to find the actual best value for K we need to evaluate the possibilities for K and see which value minimises the error, where the error is the percentage of incorrect predictions made by the model for each K. The range of values for K that were being inspected were the integers from 1 to 40, as we should expect K to be around the general optimal value of 25. To evaluate, we plotted a graph of mean error against the K value using the pyplot package from matplotlib [Fig. 3]. This indicated a minimum point at K = 16.

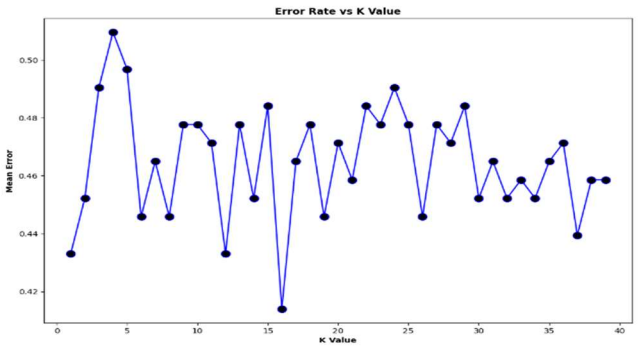

Fig. 3: A line graph of the mean error of the KNN model when K, the number of neighbours, is varied. There is a minimum point at K = 16.

However, when implementing this value, the model was outputting the majority class, males, too much, and the minority class, females, too little. To combat this, we tried a variety of K values around this 'optimal' value to try to minimize the overall error whilst simultaneously minimizing the error for the minority class. From this, we concluded that the best value for K was 25, coincidentally the general rule value. This can be seen in the confusion matrix for the pair [Fig. 4]. This displays the predictions for each category and whether they were correct or not.

K = 16 / K = 25		Actual Class: Male	Actual Class: Female
Predicted Class	Male	**65** 63	**21** 23
	Female	**45** 39	**26** 32

Fig. 4: The confusion matrix for K = 16 and K = 25 for the KNN model when evaluating against the validation set.

For the *LSVC* pipeline, the second step is to input the data into *LSVC* classifier, which has the following hyperparameters: random_state = 0, which triggers the pseudo random number generator to shuffle the data, as the order of the examples should not have an affect on the model; class_weight = "balanced", this uses the values of the target data, y, to automatically adjust weights inversely proportional to class frequencies [5], this is to counter the 55:45 ratio between the majority to minority class and, when comparing this to the default class_weight, this yields far better results; dual = True, as n_samples <= n_features; C = 1, this is the default value for this hyperparameter and is inversely proportional to strength of the regularization, and as such, impacts whether a model is under or overfit to the training data, however when C is increased, there is no effect on the *CV* score or the accuracy on the validation set; when C is decreased, there is a slight decrease in the *CV* score. Therefore C = 1 is optimal.

For the *RBF SVC* classifier, we experimented using it as a base estimator for a bagging classifier, which is a type of ensemble method. A bagging classifier is a combination of Bootstrapping and Aggregating. Bootstrapping generates different subsets of the training set to then train the same amount of different base learners, each with a different subset. Aggregating inserts the same input into the base learners to make a prediction in parallel and then the majority vote is the prediction of the bagging classifier. These were compared on the validation set using the respective classification reports [Fig. 5], which displays an array of data. This includes precision, which represents how likely a model's prediction is to be correct. The recall/selectivity is also shown, which shows the

	Precision	Recall	F_1-Score
1	**0.63** 0.64	**0.63** 0.65	**0.63** 0.65
2	**0.55** 0.57	**0.55** 0.56	**0.55** 0.57
Accuracy			**0.59** 0.61
Macro Average	**0.59** 0.61	**0.59** 0.61	**0.59** 0.61
Weighted Average	**0.59** 0.61	**0.59** 0.61	**0.59** 0.61

Key: RBF SVC = Black, Bagging Classifier = Red

Fig. 5: The classification report for the 3 models when evaluating against the test set.

proportion of predictions for that category that were correct. It displays the F_1 score for each category, which is the harmonic mean of precision and recall. It also displays the overall accuracy across the 2 classes. In addition to this, for each of these performance metrics, a macro average and weighted average (proportional to the number of examples in each category) is displayed. The bagging classifier outperformed the standard *RBF SVC* classifier on every metric by a small but not insignificant amount.

For the hyperparameters of the *RBF SVC* classifier within the bagging classifier we used: random_state = 0 and class_weight = "balanced", both for the same reason as in the *LSVC* classifier.

Model Comparison

To compare the models, we used a series of performance measures that they produce after the test data is inputted. The performance measured involved did not include ROC (receiver operating characteristic) curves or the area under the *ROC*'s curve as only the *KNN* model supports this function. Since there are 4 input features, a 5d plot would be necessary to visualize the data against the classification models. Thus, we used the classification report and confusion matrix to compare.

The information produced indicated that the *KNN* and the ensemble bagging *RBF SVC* classifier produced very similar results [Fig. 6]. The precision and F_1 scores for both classes were identical, there was a slight disparity for the recall in each class, but the average was the same. Comparatively, The *LSVC* results were much worse, with much lower scores across the board, apart from the recall on females, as the model predicted that class much more often than the other 2 models.

	Precision	Recall	F_1-Score
1	0.57 0.60 0.60	0.59 0.67 0.66	0.57 0.63 0.63
2	0.48 0.54 0.54	0.49 0.46 0.47	0.49 0.50 0.50
Accuracy			0.53 0.58 0.58
Macro Average	0.53 0.57 0.57	0.53 0.57 0.57	0.53 0.57 0.57
Weighted Average	0.53 0.57 0.57	0.53 0.58 0.58	0.53 0.57 0.57

Key: LSVC = Blue, KNN = Green, Bagging Classifier = Red

Fig. 6: Comparison of the classification reports for the 3 models on the test set. It is quite clear here that the KNN and bagging classifier produce similar results, and both outperform the LSVC model.

5 Conclusion and Discussion

To conclude, the project was to find the best *ML* model that can use results from a *HSQ* to predict a participant's gender based upon their answers and as such humor style. This was achieved by first framing the problem we are trying to solve. This involved preparing the data by constructing the dataset and transforming the data where necessary. Following this we then chose the appropriate models that we would train. To improve our models, we tuned the hyperparameters to them where necessary, any improvements were measured by passing the validation set through them. Since we trained multiple models, we compared them using the test data to find the best model.

The result of the project was that the *KNN* model and the *RBF SVC* Ensemble model were both the best model for our test data and that the *LSVC* model was weaker than them both. We evaluated this by using a series of performance measures which could be represented in a classification report for each model. These measures included precision, recall, F_1 score and accuracy.

The major limitations of the approach that we took was that it meant that the "other" class could not be predicted and as such the model would always be incorrect whenever a participant was from that category. Secondly, due to the inability to reduce the number of features or represent them in less variables, we were unable to graph the data to provide more guidance when picking hyperparamaters etc.

To improve the procedure, we could've tried more options for the gender = "other" category. Since some of the classifiers we used are designed to work as a binary classifier, as opposed to multi-class, so we could've created multiple binary classifiers within 1 model to create a pseudo multi-class classifier. We also could've gathered more data which would hopefully include examples that had their gender as "other".

We were surprised at the result of what was the best model; following the *ML* cheat sheet for scikit-learn, it suggests trying the *LSVC* algorithm first, seemingly because on average it performs the best, however it was infact the worst classifier we built.

Ultimately, the *KNN* and Ensemble models both produced precision values of 0.60 for males and 0.54 for females. Whilst the models can always have room for more tuning of hyperparameters and the other steps involved, we would infer from this that there is only a weak correlation between gender and humor styles.

References

[1] Martin, R. A., Puhlik-Doris, P., Larsen, G., Gray, J., & Weir, K; Individual differences in uses of humor and their relation to psychological well-being: Development of the Humor Styles Questionnaire; Journal of Research in Personality, 37, 48-75; 2003.

[2] Scikit-Learn, Choosing the right estimator, scikit-learn.org/stable/tutorial/machine_learning_map/index.html, 2021.

[3] Scikit-Learn, make_pipeline(), scikit-learn.org/stable/modules/generated/sklearn.pipeline.make_pipeline.html, 2021.

[4] Amey Band, *How to find the optimal value of K in KNN?*, towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb, 2020.

[5] Scikit-Learn, *sklearn.svm.LinearSVC()*, scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html, 2021.