Vasiliki Zafeiropoulou

Student Number: 20204984

Machine Learning 1: Exam Project

Title: Predicting Happiness with sklearn

Experiment Report

1. Introduction

1.1. General

The aim of this report is to outline and discuss the steps that I followed while doing a machine learning experiment in the context of the ”Machine Learning I” course, part of the Master of Digital Text Analysis offered by the University of Antwerp.

The task I had to perform, and thus the goal of this experiment, was to predict how happy the citizens of a country were, given an array of features such as economy, health, freedom etc. By analyzing this information, and if supposedly this experiment manages to actually predict happiness based on these factors ,then that could prove to be useful in eliminating issues that affect the global happiness, by first understanding each factor's importance and effect on the happiness scores. Besides that, predicting happiness is alone a very interesting subject for obvious reasons.

1.2. Dataset

This experiment is possible thanks to the data collected and published at World Happiness Report (https://worldhappiness.report/), which is a publication of the Sustainable Development Solutions Network, powered by data from the Gallup World Poll. This report and the associated data are published every year and contain happiness scores for many countries of the world. Of course, it's not possible to include data from every country: this results in data for approximately 150 countries and sometimes these ~150 countries aren't the same for each year.

The data is collected in the form of CSV files (so in a table format) where each row corresponds to each country and each column corresponds to one feature, one factor used to calculate the overall happiness score, which of course is also included as a column. The column names are the following : 'Country', 'Region', 'Happiness Rank', 'Happiness Score', 'Standard Error', 'Economy (GDP per Capita)', 'Family', 'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)', 'Generosity', 'Dystopia Residual'. The data is collected by asking people of each country to participate in a poll (“Gallup World Poll”) and to describe their life quality on a scale from 0 to 10. What is most important, all scores are calculated in comparison to the scores of an imaginary Dystopian country which scores the lowest in every factor and thus is the unhappiest of all. Each country's score on every category can be read for example as “How much better is my Economy compared to this Dystopian economy?”. Then, through an equation the happiness score of each country is estimated not directly, but by calculating how much of the happiness can be explained by this factor's score.

For a machine learning experiment, columns and data such as “Dystopia Residual” or “Standard Error” are not very useful. So, I opted for a smaller version of this dataset containing only “Happiness Score”, “Economy”, “Family” , ”Health”, “Freedom”, “Trust” and “Generosity”.

2. Experiment

2.1. Preprocessing

The dataset I originally downloaded contained 5 tables, one for each year from 2015 to 2019. I became aware that the tables and data of the last 3 years were collected and structured in a different format. Since this task has already been done for the data of 2015, I decided to reconstruct the topic of my experiment by including in my dataset the data of 2015 and then adding a column with the happiness scores of 2016 for the same countries. So, now, the task is rephrased as: "How well can you predict the happiness scores of the following year based on the data from the previous one?".

To do so, I merged the two datasets (of 2015 and 2016), excluding data for countries that weren't part of both. Following that, I also removed the columns 'Country', 'Region', 'Standard Error', 'Happiness Rank' and 'Dystopia Residual' which were clearly not going to be useful in the context of a machine learning experiment. That left me with a dataset of 151 instances (countries).

Consequently, I decided to go for a classification problem instead of a regression one. By using the quartiles of the values of the happiness score columns, I created two classes; 0 for unhappy (<= 25% of data), 1 for mediocre (25% - 75%) and 2 for happy (>= 75%). In the following table, we can see that most of our data ended up being less or equal to 4, so most of our data belonged to the class 0 (unhappy) and the same proportions are observed in both 2015 and 2016:

CLASS	2015	2016
0	60	56
1	48	49
2	43	46

Lastly, I assigned the happiness scores of 2016 to the y variable (so the class labels to be predicted) and the rest of the features to X (so the features based on which the class labels are to be predicted).

Machine learning algorithms might be very powerful, but this strength of theirs can easily be undermined by introducing too many features; it can lead to overfitting. To avoid that I waned to make sure that I am using only the most important features of all. I proceeded to feature selection and I did so by using two popular methods for it: first, applying a X^2 test and then validating my results by also running the built in importance test of ExtraTreesClassifier. Both showed the same effect; the most important feature was as expected the previous year's happiness score and the least important was generosity. I chose to keep as predictors the first 4 most important features and those are 'Happiness_Score_2015', 'Economy (GDP per Capita)', 'Family', 'Health (Life Expectancy)'.

We shouldn't be too surprised about this; when deciding to use the happiness score of 2015 as a predictor for the happiness score of 2016, it was expected that such a move would have a major effect on the prediction. One country's happiness changing over a year is possible, but of course not very common. Also, as expected economy, family values and health are understandably more important than government trust, freedom and generosity, which although are undoubtedly important, they are more abstract and vague ideas, and do not infer with everyday life as much.

The final step of our preprocessing is splitting our data into train and test sets, using sklearn's train_test_split. Our data are now ready to be used for training the algorithms.

2.2. Algorithm Implementation

In this part of the paper, I'm going to discuss my results from the different machine learning models I tried. For each of the experiments I followed the same method; creating a pipeline of a standard scaler and each model's algorithm. The algorithms that I tried are; Perceptron, Logistic Regression, Decision Tree Classifier, Support Vector Machine, Stochastic Gradient Descent classifier and K-Nearest Neighbors. Although, standardization isn't necessary for algorithm such as KNN and Decision Trees, I chose to keep the standardized version of my data, in order to make the results as comparable as possible. I run a grid search (and the built-in 10-fold cross validation) on each algorithm's parameters to select the exact model that fits my task the most. The accuracy of each model is tested following once again the same steps: a cross validation accuracy score and a boxplot showing the average accuracy of each fold, a classification report and a confusion matrix.

All the models I implemented managed to achieve a relatively high accuracy. More detailed:

- **Perceptron**

The perceptron performed worse than all the rest but also relatively well, since it got 87% accuracy. For the unhappy class it scored a lower recall, meaning that it didn't find all the true positives, but for the rest of the classes it did very well, scoring 93% recall. However, when it comes to correctly classifying them, all of the data of class 0 (unhappy) were correctly classified, something that is not true when it comes to the rest of the classes. The best parameters returned from the grid search were a learning rate of 1 (so quite a big one) and the smallest amount of iterations provided (50) which means that the perceptron was quite fast and converged relatively easily.

- **Logistic Regression**

Logistic regression seemed to work very well with my task. It scored a total of 91% accuracy. I chose to run a grid search providing two different solvers a 'liblinear' and a 'lbfgs'. If 'liblinear' was chosen, it would mean that the task is handled with a one vs rest scheme where as 'lbfgs' is used, according to documentation for multiclass problems supporting multinomial loss. The best parameters chosen by grid search were the mo-del of 'lbfgs' solver with a C parameter set at 10, indicating that not a very strong regularization, and a tolerance of 0.0001. The model had better recall and precision scores than the previous ones; it reached a perfect precision for classes 0-unhappy and 2-happy (so everything was correctly classified for those two) and a very good recall for classes 0-unhappy and 1-happy. As we can see from the confusion matrix, 4 data instances were misclassified.

- **Decision Tree Classifier**

This model's version with a max node depth of 4 and entropy as a criterion to split the nodes on (as always provided by grid search) also performed very well, with an accuracy of 96% and only misclassifying two data instances. The average recall and precision (f1-score) is very good as well, for all three of the classes (0: 97%, 1:97%, 2:96%). The tree plot nicely describes how difficult it is according to the model to be happy (out of all the nodes, only results in the happy class!). The only thing we can comment, is that in the cross validation accuracy score, the boxplot seems to have quite a large confidence interval compared to the other algorithms meaning that the accuracy score fluctuated quite a lot depending the section of the data used.

- **K-Nearest Neighbors Classifier**

The best KNN model according to the grid search was the one taking into account 5 of the closest datapoints. It reaches a very good accuracy of 93% with very good recall and precision values for all the classes. It misclassifies only 3 datapoints.

- **SVM Classifier**

Grid search chose as best an SVM classifier with a curvature of 0.1 (gamma) and a C of 1.0 meaning that our regularization strength is not very large. The kernel chosen is the linear, which means that our problem is linearly separable, and thus explains a why perceptron managed such a high accuracy score. The accuracy of this model is 93% and the scores it gets, are very similar to the KNN classifier. The cross validation accuracy boxplot though shows a higher accuracy range compared to the KNN, although the mean of both seems to be about the same.

- **SGD Classifier**

Finally, the SGD model chosen with L2 penalty and a 'log' loss, which makes it closer to a logistic regression. It achieves equally good accuracy, reaching up to 96%. However, I have to report that it is not very stable; every time I run it, it returns different results, not in terms of accuracy as it is always above 90% but in terms of what parameters are deemed the best by grid search. That's why I won't go into further analysis of this model.

3. Conclusion

Upon considering all the models and their results, it makes sense to consider this an easy task for machine learning algorithms. They all seemed to perform almost perfectly, of course in the light of using Grid Search, which is a powerful tool in the hands of anyone designing a machine learning algorithm especially when making their first steps in the field. Speaking strictly from an accuracy perspective, the best performing models for this task were the Decision Tree algorithm and the SGD Classifier, with an accuracy of 96% and a total of misclassified instances of 2, but again, the difference is so small that it is almost not worth mentioning.

The happiness score of the previous year seems to play a very important role in predicting the happiness score of the next year, as expected. Future implementations of this experiment could also include the happiness scores of other years and inspect the extent to which one country's happiness could be predicted based on features and happiness scores of 2,3,5 etc. years before. This could lead to major improvement of many recurrent social problems.

Sources:

Sklearn documentation: https://scikit-learn.org/stable/

Theobald, O. (2017). *Machine Learning for Absolute Beginners* (2nd ed.).

Raschka, S. & Mirjalili, V. (2015). *Python Machine Learning* (3rd ed.).Birmingham, UK: Packt Publishing Ltd.