**Practical Machine Learning**

Executive Summary

Machine learning has potentially large implications for the world of media analysis, as for every industry. The ability to classify, tune and refine a model that will produce high quality analysis of large datasets, is something that will change the way many tasks are performed in future. Onclusive could leverage the power of machine learning to improve the quality of both human coded data, and the automated coded data.

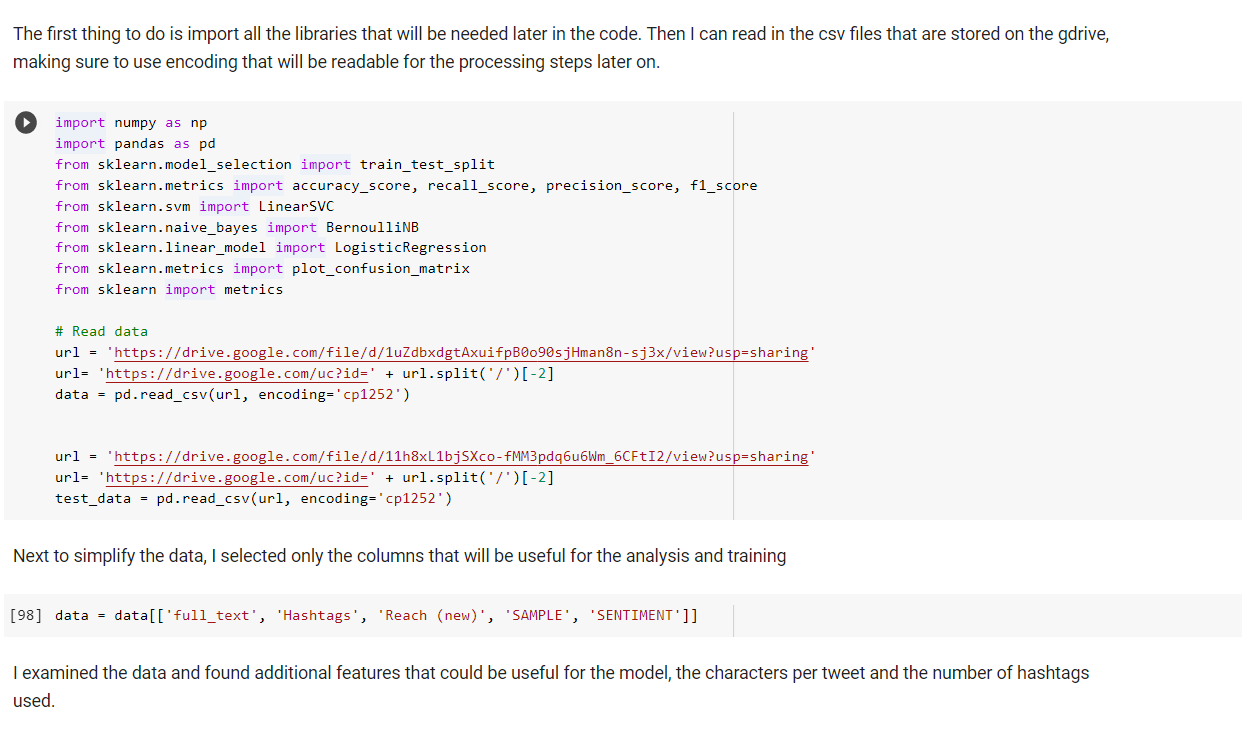
So, I have set out to create a predictive model for the sentiment of tweets that has been trained on human coded data, with the potential to fine tune it to deal with specific client datasets. The training consists of tokenizing tweets, defining the features to target in the training and setting the parameters and hyperparameters to achieve the best result. Setting it up in this way will maximise the chance of a model that has a high accuracy and that will prove to be a valuable resource for producing deliverables for clients. The current machine coding of tweets is very basic, with a large volume of neutral items coded and the polarity of tweets mostly missed.

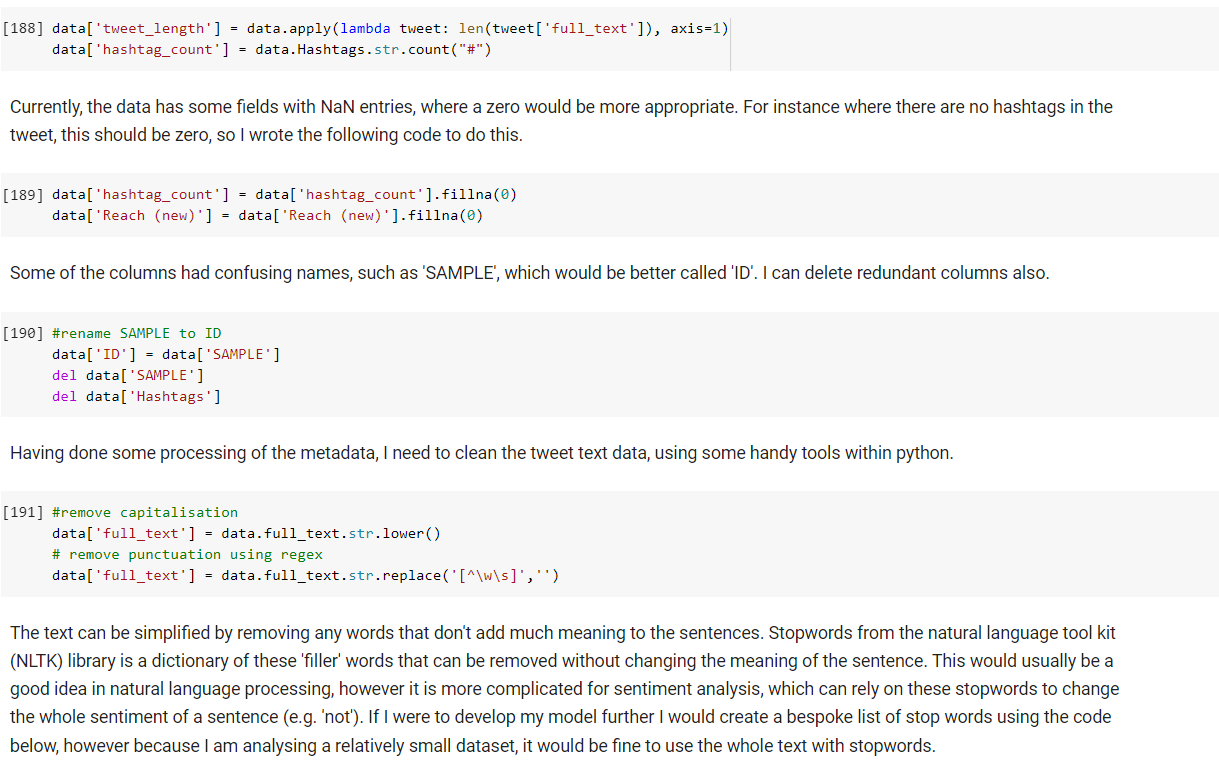
Although being able to offer clients a reliable automated sentiment service for social media will be useful, in order for it to be feasible for the company, it cannot require too much in the way of capital resources, such as high computational capacity or lots of new staff. I believe, using algorithms such as logistic regression, Bernoulli Naïve Bayes and a support vector machine would be ideal for this application, as they do not require deep learning techniques and can be scaled cheaply to the millions of tweets that may need analysis.

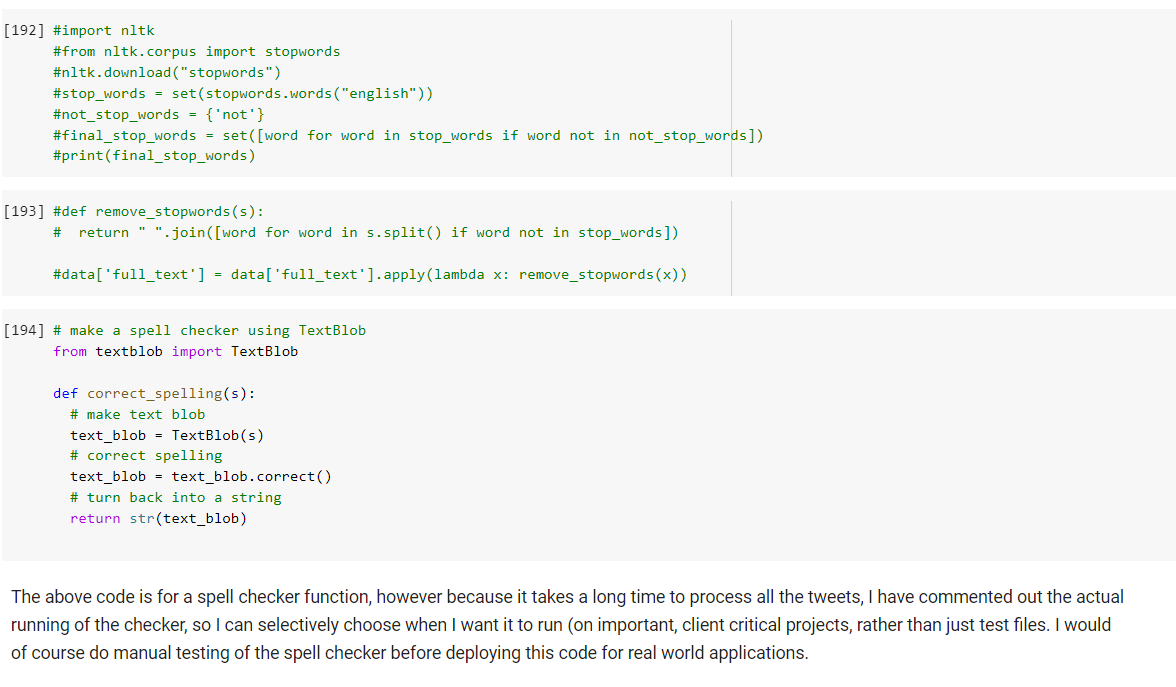
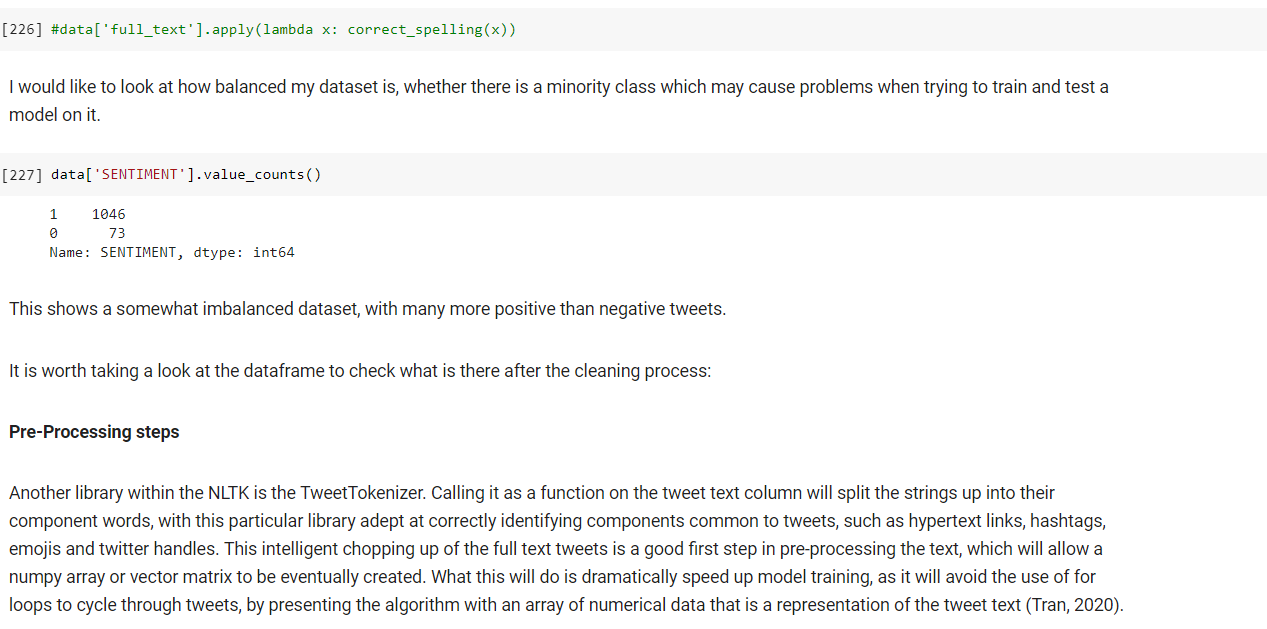
However, I am interested in how deep learning can be used in the future at the company, because this type of machine learning will offer the highest levels of sentiment analysis capability, with neural networks that will take not just single words and their nearby neighbours, but whole phrases and context specific language into account when determining a sentiment prediction. Therefore, I have investigated using a neural network pre-trained model (the Bert base uncased model), which is an industry leading natural language processing classifier, to determine the sentiment of tweets. It uses a 16Gb corpus of text data with which to assess the context of a body of words and determine the sentiment of them. I will fine-tune this Bert model incorporating the unique weights and balances that will be determined from training it on a dataset of human coded tweets.

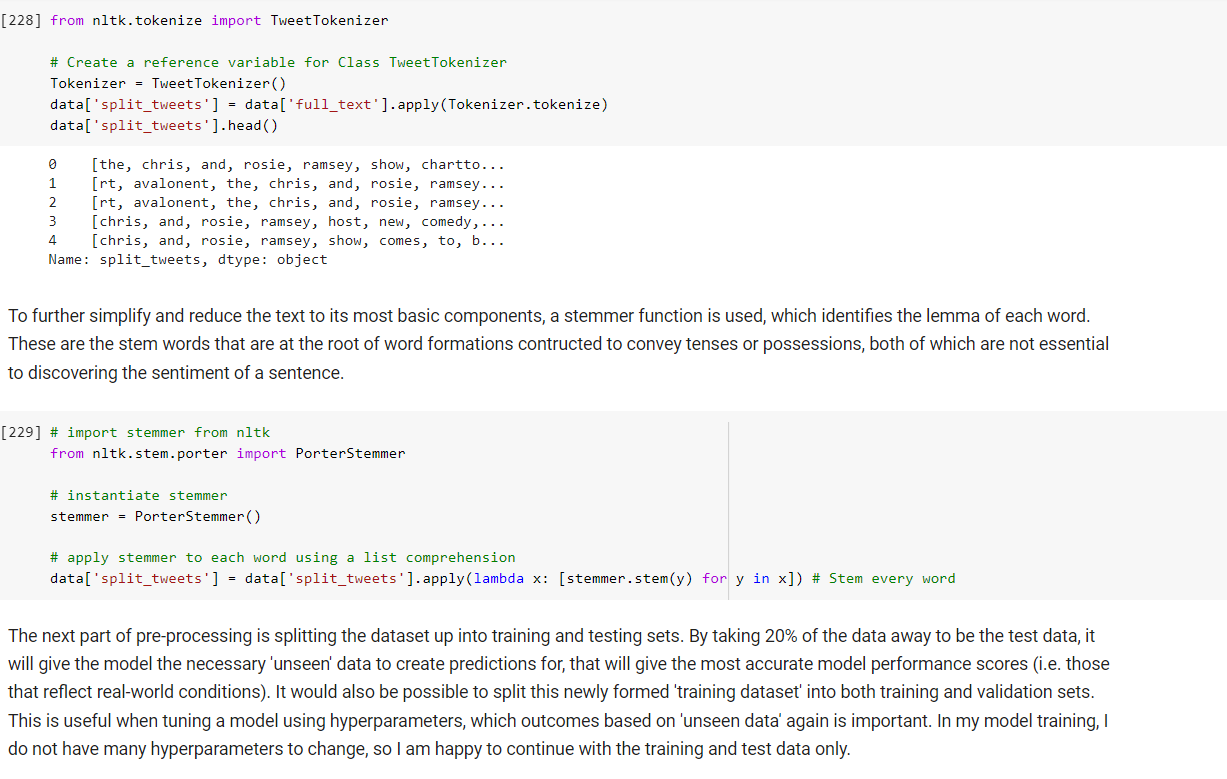
I want to carry out testing of all the models and compare the f1 score and accuracy of each to determine if there is a significant difference and what their overall performance is. I will come back to this alternative approach to creating a machine learning model for sentiment analysis after the conclusion of the first approach.

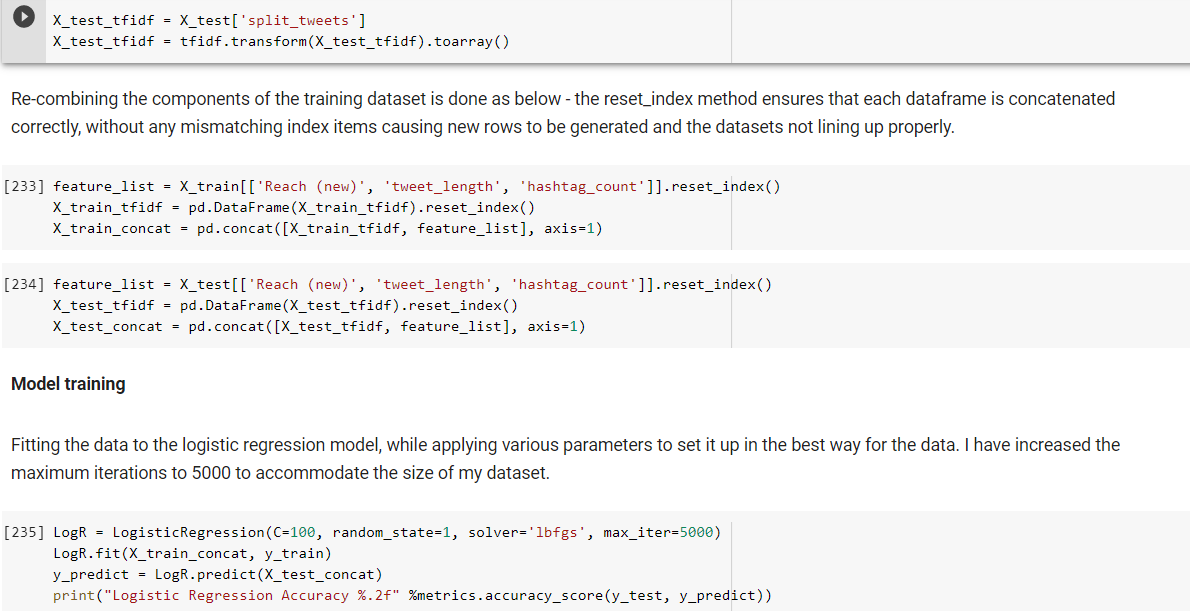
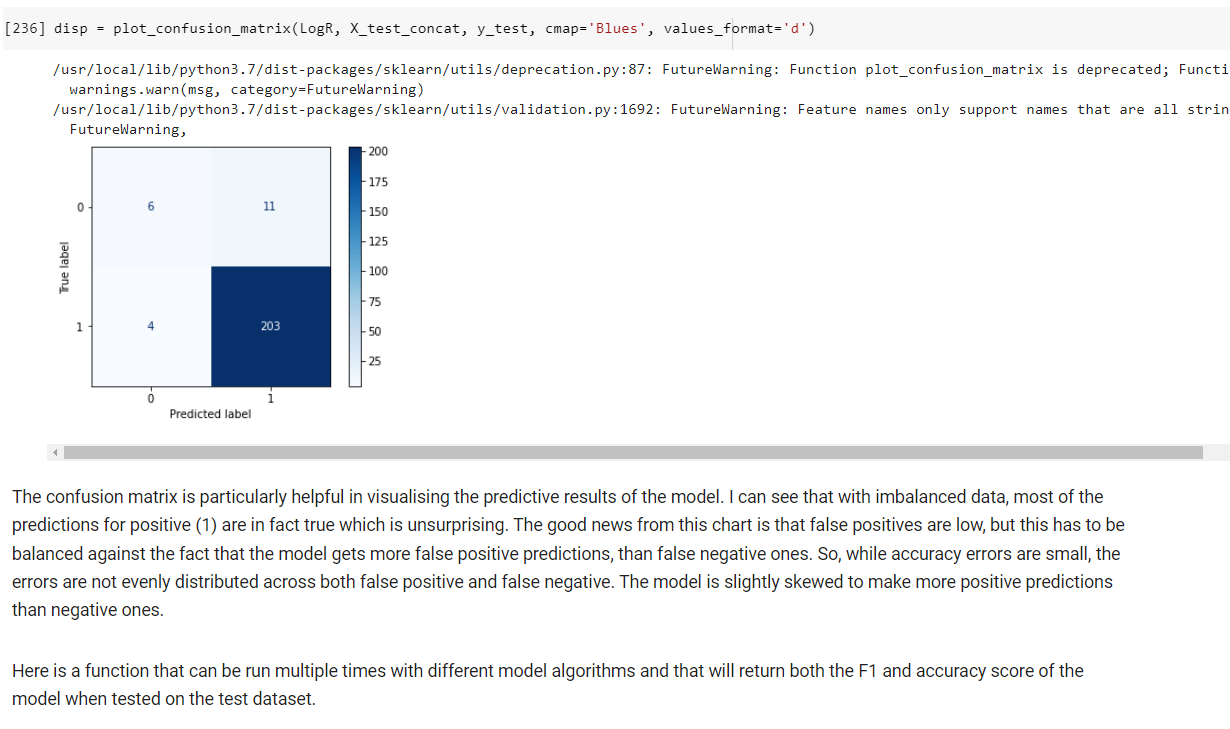
Technical documentation of machine learning model:

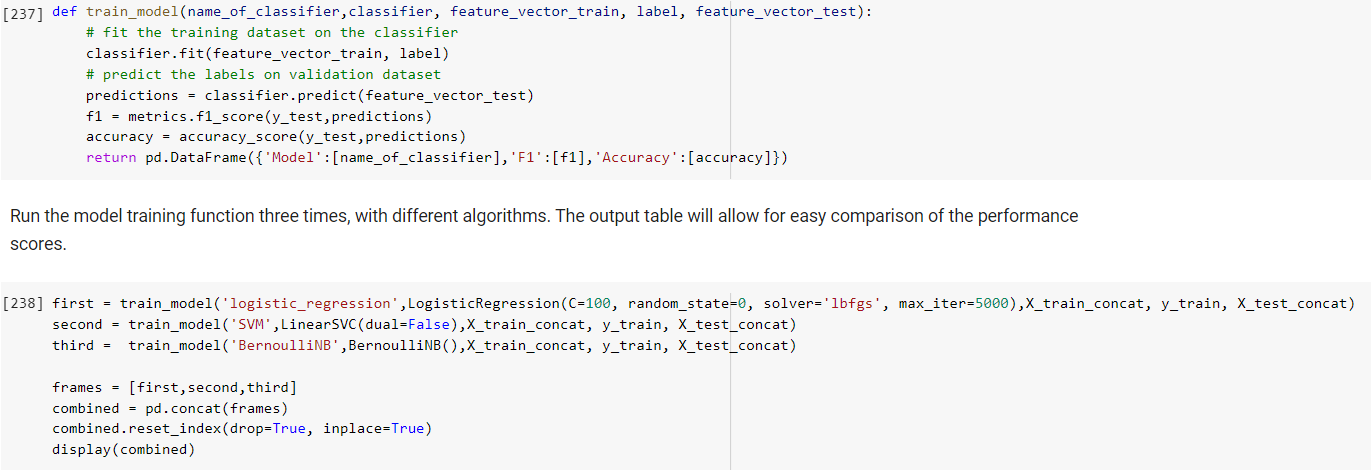


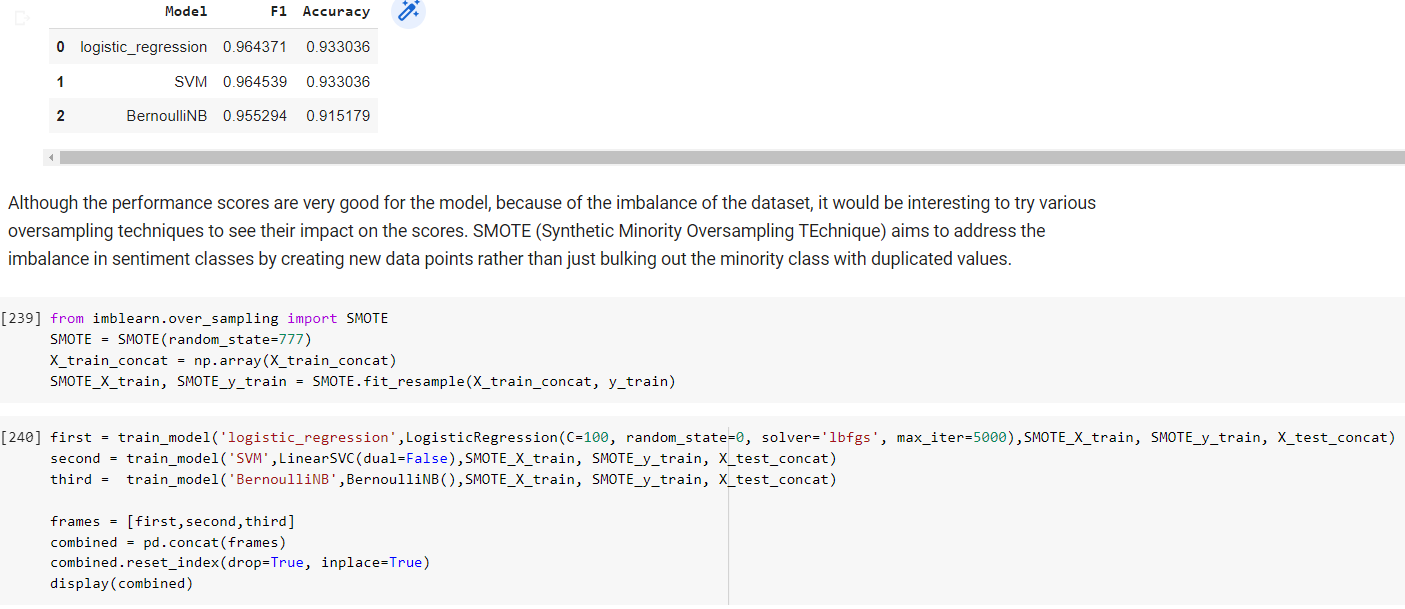


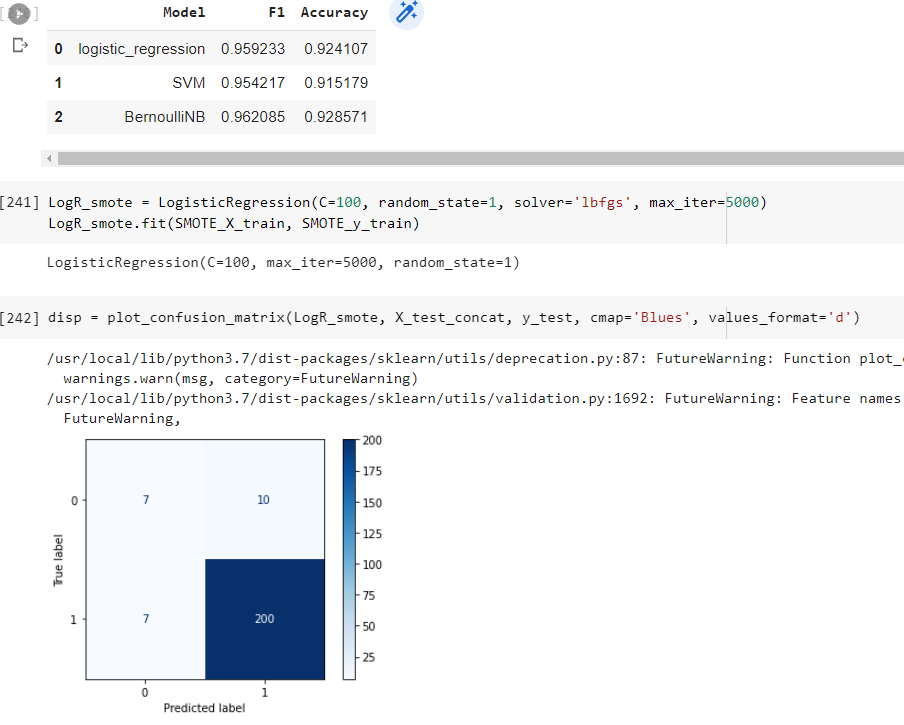
 

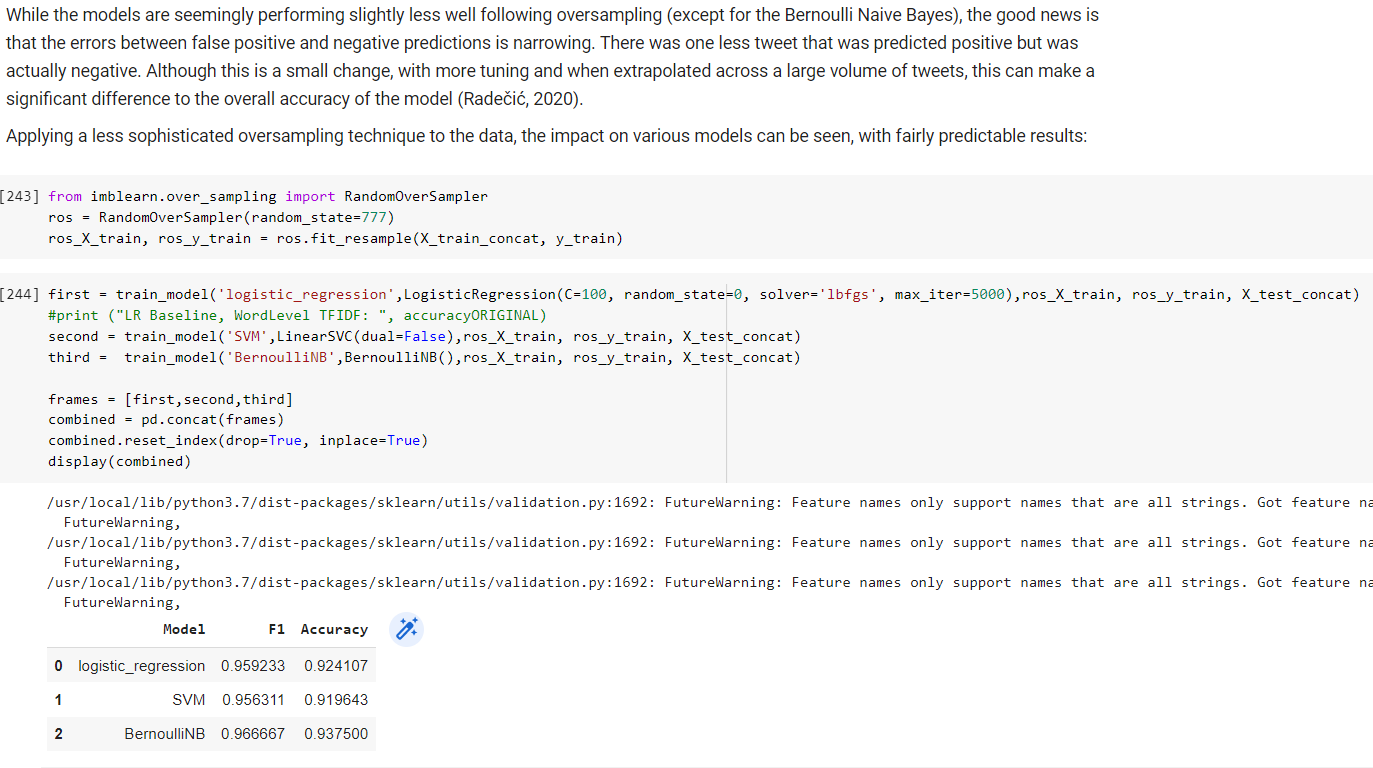
 

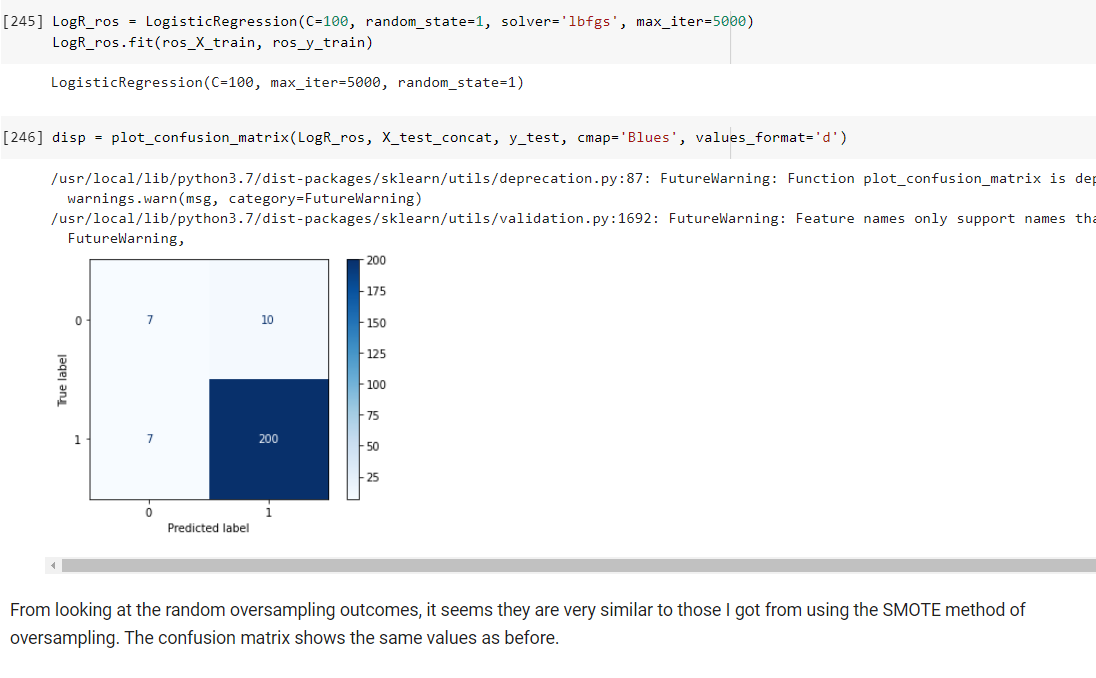
 











Rationale of my machine learning model approach

The first step is analysing and understanding the data I am working with to create the machine learning (ML) model to solve the business problem. I have a corpus of labelled data, i.e. data that is already structured to a degree and that has already some classification applied to it (the human coded sentiment analysis). Therefore, my approach will be supervised learning. I am wanting the model to replicate the human coded classification of tweets, so logistic regression, Bernoúlli Naïve Bayes and support vector machines which are ideal for binary classification will work well. Logistic regression is one of the most popular machine learning models, probably because it mimics the strengths of neural networks to achieve a well-fitted model (Raschka, 2019). This is in part because it makes no assumptions about the distribution of classes. The Bernoulli Naïve Bayes works in a different way to logistic regression, with generally better performance on smaller datasets and where all features are independent (no collinearity). This was, in practice, a slightly more successful algorithm, which had higher scores for f1 and accuracy when used in combination with the oversampling techniques which I employed to try and balance the sentiment classes within the data. Finally, the support vector machine (SVM) algorithm is mostly used for classification problems where each item is in an n-dimensional space (n being the number of features) and the value of each feature is used to generate the coordinates of the data point. A hyper-plane is then found that will separate the two classes and the best version will be selected for the model (Ray, 2017).

One of the first tasks of pre-processing is to clean the data, so that only relevant data remains, and the model is not diluted by noise or extraneous data. After this, with the focus on relevant features, it is useful to convert text values into numbers, which are most easily handled by any machine learning model. Although unsupervised learning can develop pattern recognition on text fields or even images, this requires large neural networks and processing power, which is unnecessary for this application.

There is an imbalanced class problem with the data. The greater proportion of positive over negative tweets will present various problems to creating a reliable prediction model. This can be because it will tend to predict that the minority class never occurs, and unfortunately will be right on most occasions so score well in testing. To address this, it is useful to change the data used for the training, or the loss function that dictates parameter optimization or the evaluation techniques used on the results. This could be by oversampling – I have used a couple of techniques for this with my data, the only disadvantage to this can be that it can skew the weighting of the minority class and make it more important than it is. Undersampling is another method, which will decrease the majority class and necessarily have the downside of reducing the total data I can use to train a model (Mueller, 2021).

The logistic regression classifier is from the sklearn library and has various parameters that I can alter to affect how the classification process works. These include the solver used, the random state and the value of C. This latter parameter is a hyperparameter, in that it controls the training process, rather than the internal workings of the model. To explain further, the process of regularization needs to be understood. It applies a penalty to extreme parameters and leads to shrinking of the coefficients which can help reduce complexity of the model if some of the weights are reduced to zero. There is a default regularization included in the logistic regression function within the sklearn library. The value of C is used to oppose the tendency of regularization to minimize overfitting and over-reliance on the training data. If C is small, regularization will have a larger effect, constraining the size of the model coefficients, which will stop overfitting and how closely the model will try and fit to the data. I experimented using a smaller value for C (=1) which led to a much high volume of false negatives. The random state parameter controls whether each time the classifier is run the same number set is used within the model processing, and therefore is reproducible over time, or whether random number sets are used, and model outcomes will vary each time.

The logistic regression algorithm takes continuous or discrete dependent variables and outputs a binary independent variable. It does this by assigning probabilities to the two outcomes using a sigmoid function on the input variables. The binary outcome will be determined by whether the overall probability is closer to 0 or 1.

The accuracy score from sklearn is produced by dividing the total number of accurate predictions by the total number of predictions. This, in principle, is a good evaluating measure for my business problem, because what clients will be most interested in is how reliably a positive or negative tweet is classified as such. False negatives (which are picked up by the recall score) and false positives (picked up by precision score) are not as important, because there is likely to be little cost to the presence of these in the results, compared to the overall ability of the model to detect the tone of tweets. The scenario that would have to occur for this to be problematic would be the presence of a specifically negative tweet with a large reach being incorrectly labelled positive with no other follow-up tweets, retweets or reinforcing tweets by other users on the same topic. This is highly unlikely and so the value of measuring false positives is small.

However, the caveat to the above, is that because of the imbalance in my dataset, with very few negatives and lots of positives, the accuracy score loses some of its potency as an evaluating measure. This is because any inaccuracy in detecting negatives will be drowned out by the magnitude of the true positives that are found. This is where the f1 score can be of more value, as it combines the scores of recall and precision into a single metric, and in this way brings to attention more of the overall ability of the model to classify successfully, which will be much closer to the true performance of the model.

Conclusion

I have had reasonable success in creating a simple machine learning model for analysing the sentiment of tweets. I have employed various techniques to achieve this, testing various algorithms, hyperparameters and parameters to tune the model. The limitations of what I have created for Onclusive include the specificity of the model, i.e. whether the model success is because the data is imbalanced toward positive sentiment, and the model is just good at identifying that class without also being sensitized to the negative sentiment class. I would recommend further work to improve the model, exploring other ways of scaling, tuning hyperparameters (such as the ‘penalty’ and ‘class weight’ for instance), possibly using Grid Search method to find optimal values for these hyperparameters (Bhor, 2021). However, I would say that despite the potential for a falsely high accuracy rate that my dataset could produce, the confusion matrix has shown a minimal level of incorrect predictions, with a narrowing of the gap between false positives and false negatives.

I have used the f1 and accuracy scores to evaluate the performance of the model and while these are very helpful, I would recommend the use of the ROC AUC evaluation method for future use. This is the Area Under the Receiver Operating Characteristic Curve and tells us about the relationship between the true positive rate versus the false positive rate. It does this by plotting the ratio between these two rates at difference classification thresholds. It has benefits over the accuracy score when used on data that is highly imbalanced (Google, n.d.).

Some of the issues that will be encountered in using a machine learning model in Onclusive will be in relation to the data cleansing and preparation phases. The need for data that is in precisely the right format to feed into the cleaning pipeline will be resource intensive. There is also the problem of reliability given the ‘black box’ nature of machine learning models. There is no way to see exactly how the model is performing and could potentially produce inaccurate data on future datasets without warning. There is also the need for trained employees in understanding machine learning code, who can work to improve, monitor and maintain the code for the model.

# Bibliography

Bansal, S. (2018). *analyticsvidhya.com*. Retrieved from https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/

Bhor, Y. (2021). Retrieved from analyticsvidhya.com: https://www.analyticsvidhya.com/blog/2021/09/guide-for-building-an-end-to-end-logistic-regression-model/

Bowne-Anderson, H. (2016). Retrieved from Datacamp: https://www.datacamp.com/tutorial/preprocessing-in-data-science-part-2-centering-scaling-and-logistic-regression

Google. (n.d.). *Classification: ROC Curve and AUC*. Retrieved from Google Developers: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Mueller, J. P. (2021). Retrieved from Machine Learning For Dummies. 2nd edn: https://www.perlego.com/book/2770772/machine-learning-for-dummies-pdf

Radečić, D. (2020, December). Retrieved from Toward Data Science: https://towardsdatascience.com/how-to-effortlessly-handle-class-imbalance-with-python-and-smote-9b715ca8e5a7

Raschka, S. a. (2019). *Python Machine Learning. 3rd edn.* Retrieved from https://www.perlego.com/book/1323528/python-machine-learning-pdf

Ray, S. (2017). Retrieved from analyticsvidhya.com: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Tran, K. (2020). Retrieved from Towards Data Science: https://towardsdatascience.com/an-introduction-to-tweettokenizer-for-processing-tweets-9879389f8fe7

(Radečić, 2020)

(Bansal, 2018)

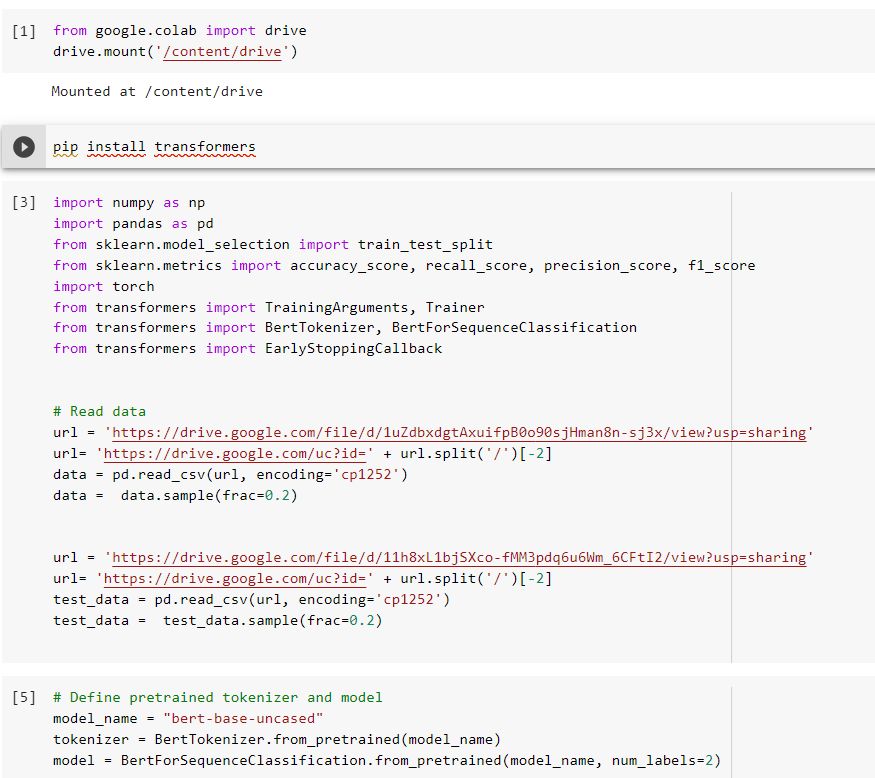
(Ray, 2017)

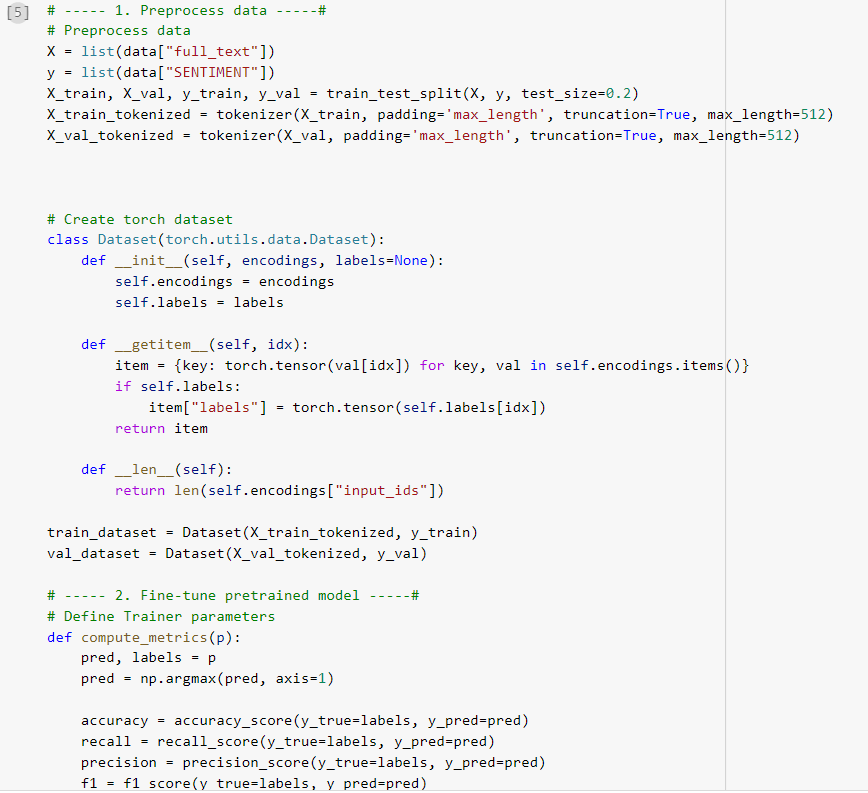
(Tran, 2020) (Bowne-Anderson, 2016)

**Appendix**

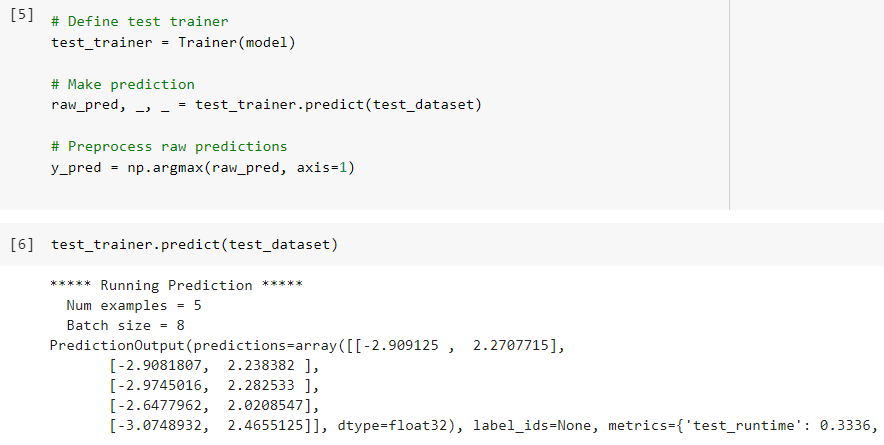
Note on pre-trained deep learning model

I have included the code for the deep learning model in an appendix here:









I carried this out using Google’s colab service, which provides the machine learning computational capabilities for free with only a limitation on computing resources and a 12-hour time limit (12Gb RAM, 110Gb Hard Disk and 1 Tesla GPU). Using this resource I reduced the total machine learning time from just over 4 hours for my dataset (1,119 items) to around 4 minutes, with a dataset of 19 items for testing. I would propose using this trained model that is saved to analyse tweets and compare the difference between this and the Bernoulli Naïve Bayes model I created earlier.