Task 1

**1.1.** Implement and train a method for automatically labelling texts in the Financial Phrasebank with their sentiment labels. Refer to the labs, lecture materials and textbook to identify a suitable method. Include the following in your report:

* Briefly explain how your chosen sentiment analysis method works and its main strengths and limitations;
* Describe the features you have chosen and why you chose them, and hypothesise how they will affect your results;
* Explain the preprocessing steps your method requires.

(7 marks)

**1.2.** Implement, train, and test your method. Briefly document this process in the report. (6 marks) **1.3.** Evaluate, interpret and discuss your results, making sure to include the following points:

* Define your performance metrics and state their limitations;
* Show your results using suitable plots, tables and/or a confusion matrix;
* How could you improve the method or experimental process? Consider the errors that your

method makes. (9 marks)

High performance figures are less important for getting high marks than motivating your method well and implementing and evaluating it correctly.

**1.1**

This sentiment analysis method works by starting with pre-processing the input sentences as to limit the number of extra features generated in the vectorising stage. Pre-processing starts with setting all words to lowercase and removing any punctuation or numeric characters so only letters remain. We also noticed that the abbreviation ‘mn’ was commonly used in place of ‘million’, so we also made that substitution to make the results more readable. We also use a WordNet Lemmatiser, to eradicate any additional columns created when vectorising, with a part of speech tag encoded in the lemmatiser to give it more information on the context of the word. We finish by removing any stopwords from the sentences so that our model does not get attracted to with low-level information.

The strengths in our text pre-processing stage really lie in using a lemmatiser as opposed to a stemmer to perform a reduction on our words. The provision of part of speech tags to give context to the mapping, allows for better, more accurate results since we have real, dictionary words to build our model on. This pre-processing is limited by some of the text normalisation such at setting all words to lowercase since this can remove some sentence structure. For example, one could consider a word in capitals to provide a greater emphasis towards the meaning of a sentence. Further, the time it takes to process the statements is another limitation since lemmatising is more rigorous, it takes longer to perform these substitutions.

To generate features to feed into our model, we use different techniques to generate types of features: TF-IDF Vectorisation features, to reflect the importance of words in a sentence, and calculating the number of positive and negative words in a sentence. Using TF-IDF vectorisation as opposed to a simple Count Vectoriser was chosen because it incorporated a frequency element into the counts to give our model a better chance of detecting which words are most important in describing the sentiment of a sentence. Further, we decided to set the hyperparameter ngram\_range to (1, 2) so that it created features for both unigrams and bi-grams. This meant we were able to capture the influence of a negative positive word, such as *not good*, on the sentiment of our sentence – a big strength in this approach.

However, the choice to encode negatives at this point does have its limitations. If we consider the phrase ‘this is not going to have a good effect’, the ‘not’ extends beyond the word ‘going’ to affect adjective ‘good’. In its current state our pre-processing, would not this into account which is the major limitation in this stage of our pre-processing.

We chose to utilise two more features: to provide the model with the number of positive and negative words in the sentence. Using Vader’s sentiment scoring to deduce which words were positive and which were negative, we felt this was a useful feature to use as it was a direct metric on the sentiment of the sentence’s words ignoring and underlying understanding. This is where both the approach’s strength and limitations lie. Since it is a direct word scoring, we believe it will give our model a good understand of the sentence’s contents; however, limitations are that Vader’s word dictionary is not exhaustive, so we cannot produce a score for each and every word, and it is just a count so will not provide any deeper weighting to more negative or positive words.

**1.2**

To train and test our model effectively, we split our data using 60/20/20 percentage split into three datasets, train, test and validation, so that we could train the model, tune the hyperparameters and validate our results without data leakage between stages to avoid bias is our final results. We then pre-processed each dataset as discussed above and, during the vectorisation stage, ensured that we used the fit\_transform method on the train dataset and only used the transform method on the test and validation sets.

Once our features have been created, we will be using sklearn’s Logistic Regression Classifier to label our texts. An algorithm chosen due to its robust nature, since we will be dealing with a lot of uncommon observations, and its low prediction and training time when dealing with such a large set of features.

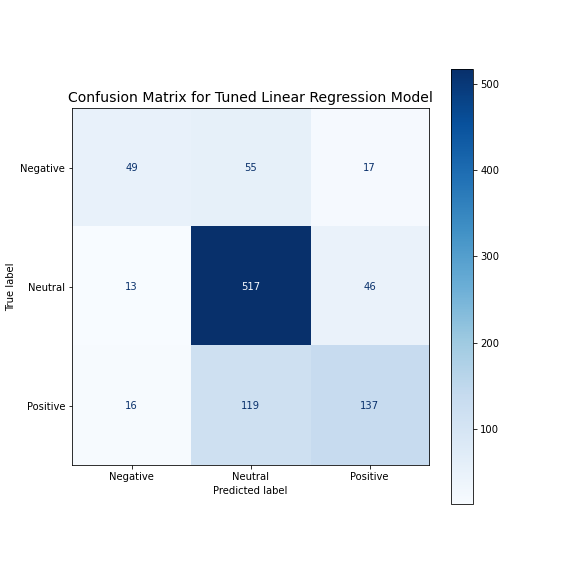
Using the ROC AUC Score as our central performance metric, we created a baseline Logistic Regression model and found severe overfitting with train and test ROC AUC Scores of 0.993246 and 0.860424 respectively. A good start, but a model that needed some hyperparameter tuning. Utilising Optuna’s SearchCV framework we focussed on tuning ‘C’ between the values of 0.0001 and 1 as this was the key regularisation parameter to reduce to model’s overfitting. We used ten cross validation folds in this hyperparameter search to confirm our tuning was performing effectively. This search suggested a value of 0.999904 ( to 6 decimal places) for our ‘C’ hyperparameter, which after careful evaluation resulted in train and test ROC AUC Scores of 0.866426 and 0.808279 respectively. These are much better results since there is a lower difference between our train and test scores, telling us that we had reduced the amount of overfitting in our model whilst simultaneously keeping our performance high.

**1.3**

To effectively evaluate our model, we used the validation set to do this. Since this was unseen data, it would give us a good understanding of how our model will truly perform.

To evaluate our final model, we used four performance metrics in conjunction: ROC AUC Score, Accuracy, Precision and Recall. ROC AUC Score refers to the area under the ROC Curve which measures the level of separability between classes and is useful in giving an overall representation of how well a model performs. Something to be careful of when quoting ROC AUC Scores in evaluation is that it depends entirely on the order of the probabilities, not the values themselves. The implication is that it is not useful for model comparison, just intrinsic model performance. We use Accuracy to give us another view on an aggregate level and can be defined as the proportion of predictions that were actually true – quite an intuitive metric. However, we need to be careful of reading too deeply into this value since it can easily misrepresent of what is actually going on. With a model that simply predicted the most common class, one can get a very high accuracy score if there is a class imbalance – something we do have in our dataset as Neutral sentences are significantly the most common. Finally, Precision and Recall are closely linked and can be looked at class-by-class or on an overall level; Precision is the proportion of predicted classes that are actually true, and Recall is the proportion of actual classes that were correctly predicted. However, due to how these values are defined, there is a trade-off to be made between which type of error ones wants to prioritise. Further, used together, these metrics completely ignores the impact of true negative predictions for each class. Now this completely depends on the situation, but the implication of this is that these metrics do not use all of the information available and so does not give the evaluator the complete picture.

To give us an initial view on how our model was classifying our sentences, we used a confusion matrix. As we can see below, Neutral sentences are being classified the most successfully, Negative sentences least successfully and Positive sentences in between. Now



**1.3.** Evaluate, interpret and discuss your results, making sure to include the following points:

* Define your performance metrics and state their limitations;
* Show your results using suitable plots, tables and/or a confusion matrix;
* How could you improve the method or experimental process? Consider the errors that your

method makes. (9 marks)

Task 2

**2.1**

To create our named entity recognition tagger, we pre-processed our sentences to get word features to put into sklearn-crfsuite’s CRF algorithm.

**2.1.** Design a method for tagging named entities in the SEC-Filings dataset. Refer to the labs, lecture materials and textbook to identify a suitable method. Include the following in your report:

* Briefly explain how your chosen named entity recognition method works and its main strengths and limitations;
* Describe the features you have chosen and why you chose them, and hypothesise how they will affect your results;
* Explain the tagging scheme for labelling entities in this dataset. (7 marks)

**2.2.** Implement, train, and test your method. Briefly document this process in the report. (6 marks) **2.3.** Evaluate, interpret, and discuss your results, making sure to include the following points:

* Explain your choice of performance metrics and their limitations;
* Show your results using suitable plots and/or tables;
* How could you improve the method or experimental process? Consider the errors your method makes. (8 marks)

**2.4.** Apply your trained NER tagger to the Financial Phrasebank dataset.

* Compute a sentiment score for each entity that you detect. Briefly explain your method. One way you could compute a score for an organisation is to count the number of positive texts it occurs in and subtract the number of negative documents it occurs in;
* Show your results, for example by listing the five most positive and five most negative organisations, along with their scores.

Task 3

**3.1.** Justify the design chosen in terms of key information visualisation principles. (5 marks)

**3.2.** Define and explain the visual queries that the user carries out when viewing your presentation of results. (3 marks)