Analysis of SVM and Boosting Implementations For Text Sentiment Classification

CM50265 - Coursework 2 - Group 4

Davis, Daniel
Department of Computer Science
University of Bath
Bath, BA2 7AY
dd684@bath.ac.uk

Whiffing, James
Department of Computer Science
University of Bath
Bath, BA2 7AY
jw2304@bath.ac.uk

April 29, 2022

Contents

List of Figures

List of Tables

1 Introduction

1.1 Data Preprocessing

Training data was processed by running each sample through a pre-trained deep sentence embedding model to produce a 384 dimensional dense vector. The embedding model used was `all-MiniLM-L12-v2` [Reimers(2021)] from the `sentence-transformers` library provided by HuggingFace , this was chosen for its strong listed transfer performance and its relatively small size. However, this model is optimised for sentences of 128 words or less and will truncate sentences that are longer than 512 words. Because many of the reviews in the dataset are longer than 512 words this poses an issue. Experiments were performed using tokenisation and stop word removal to reduce sentence length but this severely worsened the performance of the model. Intuitively this makes sense because the embedding model is trained on natural language and assumes input sentences are grammatical. Instead, any reviews longer than 128 words are split into 128 word long chunks which are embedded individually and averaged together. In order to preserve semantic continuity between chunks they overlap by 15 words. Chunks are weighted by length when averaged to account for reviews which are not a multiple of 128 long. This provided a significant improvement in accuracy, approximately 5%, over allowing long reviews to be truncated.

Shown in Figure 1 you can see the correlation between the features provided by the embedding method and the sentence sentiment.
It is clear that although there does not appear to be any single feature that is positively or negatively correlated to the sentiment, there are several features which are all over 20% positively or negatively correlated (indicated by the bright yellows and dark blues). You can also find within the appendix Figure 4, which shows the same heatmap, but using the embeddings without performing chunking and overlapping, i.e. where sentences where truncated to 128 word items.

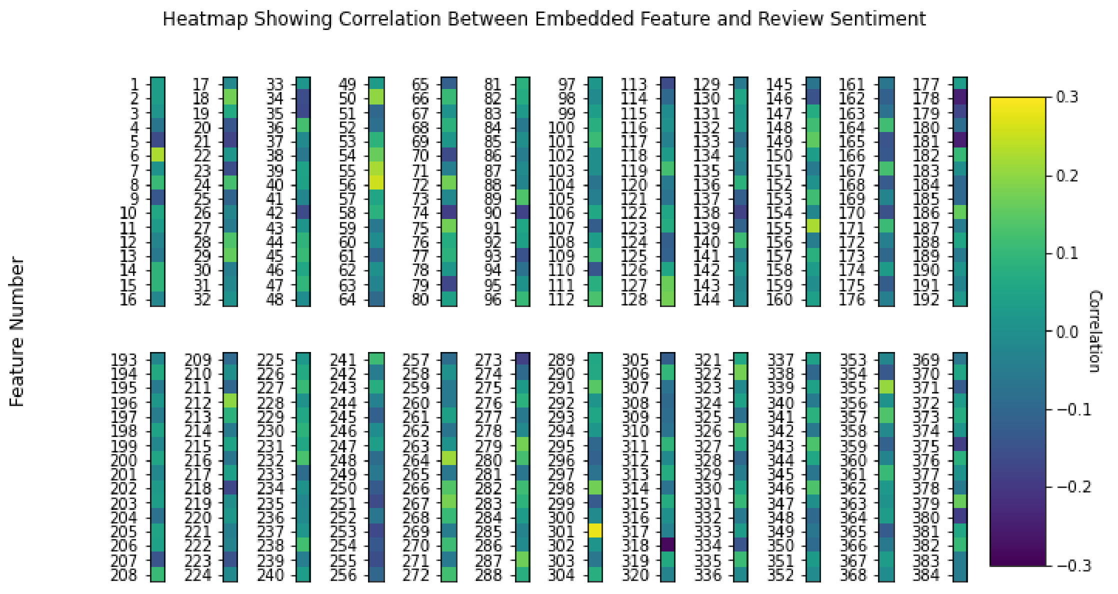

Figure 1: Heat-map showing correlation of embedding features to the sentence sentiment

2 Support Vector Machine (SVM)

2.1 Radial Basis Function (RBF)

		Predicted	
		+	-
Actual	+	744	25
	-	12	25

Table 1: RBF Kernel Model Confusion Matrix

2.2 Polynomial

		Predicted	
		+	-
Actual	+	747	22
	-	5	726

Table 2: Polynomial Kernel Model Confusion Matrix

2.3 Linear

		Predicted	
		+	-
Actual	+	663	106
	-	94	637

Table 3: Linear Kernel Model Confusion Matrix

2.4 Kernel Comparison

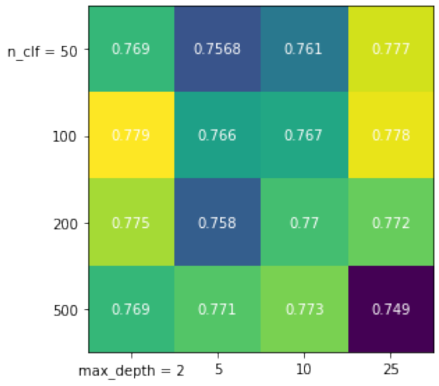

Figure 2: Hyper-parameter grid search for AdaBoost classifier

3 Boosting

3.1 Hyper-parameter Tuning

AdaBoost was chosen as our boosting algorithm. The weak classifiers used were decision trees. Hyper-parameter tuning was done using grid search over the parameters `max_depth`, which controls the maximum depth of the trees used as weak classifiers, and `num_classifiers` which controls the total number of trees used. The range searched for `num_classifiers` was $[50, 100, 200, 500]$ and the range of `max_depth` was $[2, 5, 10, 25]$. It was only possible to search a small range of parameters because grid search scales in $O(n * m)$ time where n and m are the numbers of candidates for each hyper-parameter, and because training the model can take a long time, especially when using many deep trees. The performance of each model in the search space was assessed using 5-fold cross validation and the model with the best average accuracy was chosen as our final model. Figure 2 shows the results of the grid search. There seems to be only a small different in model accuracy between models with different hyper-parameters. The final hyper-parameters chosen were `num_classifiers = 100` and `max_depth = 2`. This is because they achieved the best cross-validated accuracy.

3.2 Analysis of misclassification

The model achieved approximately 77% accuracy on the test data. It had a false positive rate of 0.17 and a false negative rate of 0.29 suggesting that negative reviews are more likely to be misclassified. Figure 3 shows a histogram of the lengths of misclassified and correctly classified reviews. There seems to be no significant effect of review length on classification quality. This is perhaps surprising given the embedding model is optimised for shorter sentences and suggests that the 'folding' strategy employed in data processing is sufficient to retain the majority of the sentiment information present in the longer reviews.

Qualitatively, the reviews that were misclassified often contained many negative individual words but conveyed a positive point overall, for example this excerpt from a review which was misclassified as negative: "I blamed this very unusual voting pattern (a sudden surge

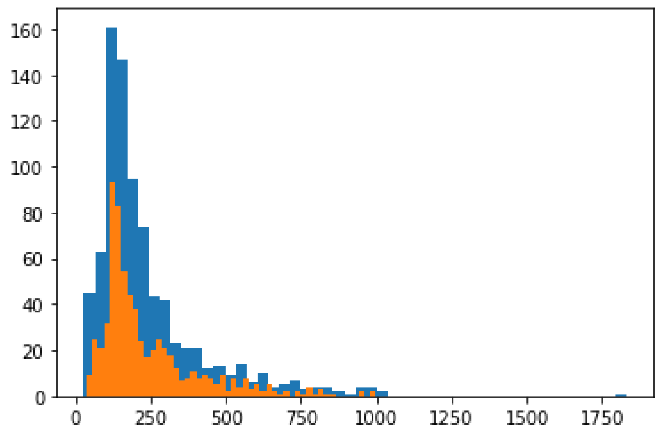

Figure 3: Lengths of correctly classified (blue) and misclassified (red) reviews.

in 1 ratings, with a high 10 rating, dropping only gradually and then suddenly reversing course and jumping at the 1 rating level) on only one thing: hatred for Leonardo DiCaprio. Believe me, Iv́e tuned into enough chat rooms to see the banter by young people (young men, mostly), who defame him left and right. They absolutely hate the man,..." This sentence clearly has a negative sentiment but the sentiment is not about the film, rather about a tangential issue which the author is using to illustrate their point in defence of the film.

Similarly this excerpt from another review which was misclassified as negative: " What results is a relentlessly chilling experience that feels very real and very disturbing, despite the fact that the story itself is fake." This is a negative sentiment but in the context of the film - in the horror genre - it is a good thing. It is easy to see why the model would misclassify these examples without knowledge of the context surrounding them.

References

[Reimers(2021)] Reimers, N., 2021. *sentence-transformers* [Online]. HuggingFace. Available from: https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2 [Accessed 2022-04-28].

Appendix

A Code And Report Source

Please find our code and report source here: https://github.com/whattheforkbomb/CM50265/tree/trunk/cw2

B Embedding Approach

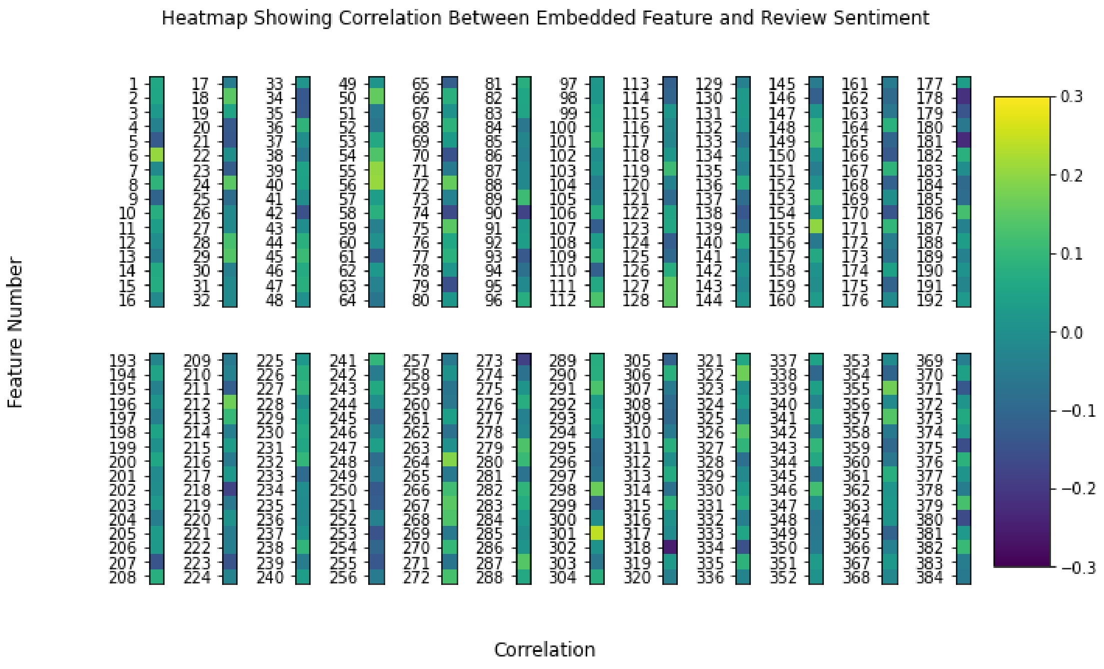

Figure 4: Heat-map showing correlation of embedding features to the sentence sentiment with sentence truncation

C SVM Hyper-Parameter Tuning

C.1 Tuning With Truncated Sentences

C.2 Polynomial Kernel Hyper-Parameter Grid Search Heatmaps

C.3 Linear Kernel Hyper-Parameter Grid Search Heat-Maps

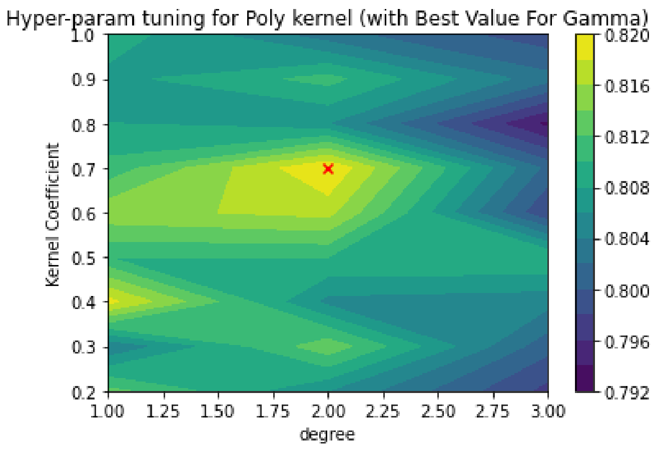

Figure 5: Heatmap of optimal hyper-parameters C and Degrees for Polynomial Kernel

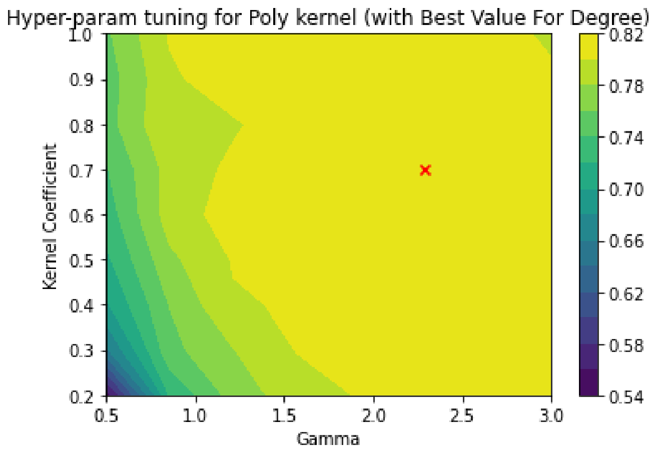

Figure 6: Heatmap of optimal hyper-parameters C and Gamma for Polynomial Kernel

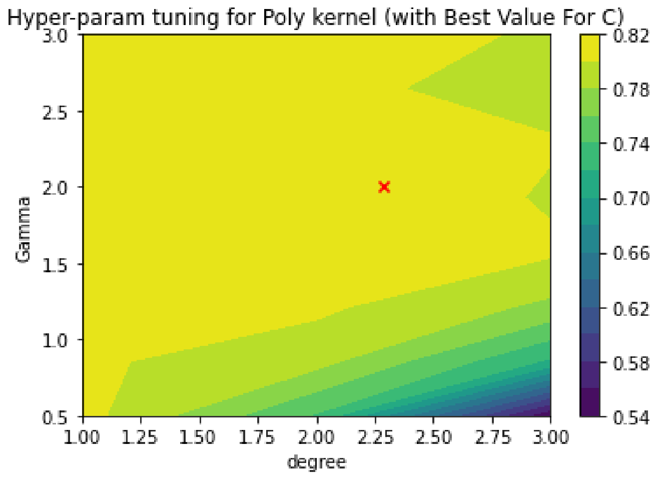

Figure 7: Heatmap of optimal hyper-parameters Gamma and Degrees for Polynomial Kernel

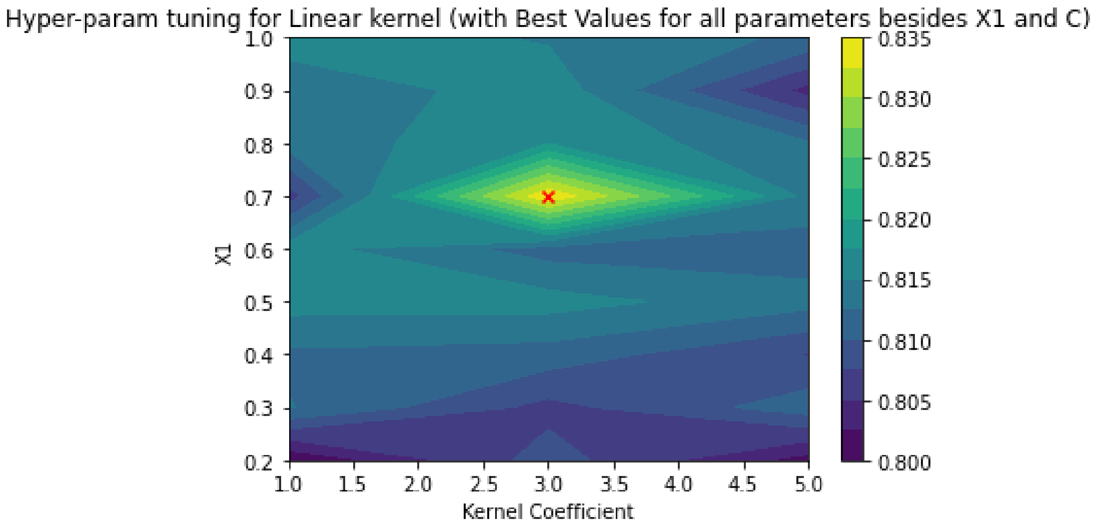

Figure 8: Heatmap of optimal hyper-parameters C and Weight for feature 300 for Linear Kernel

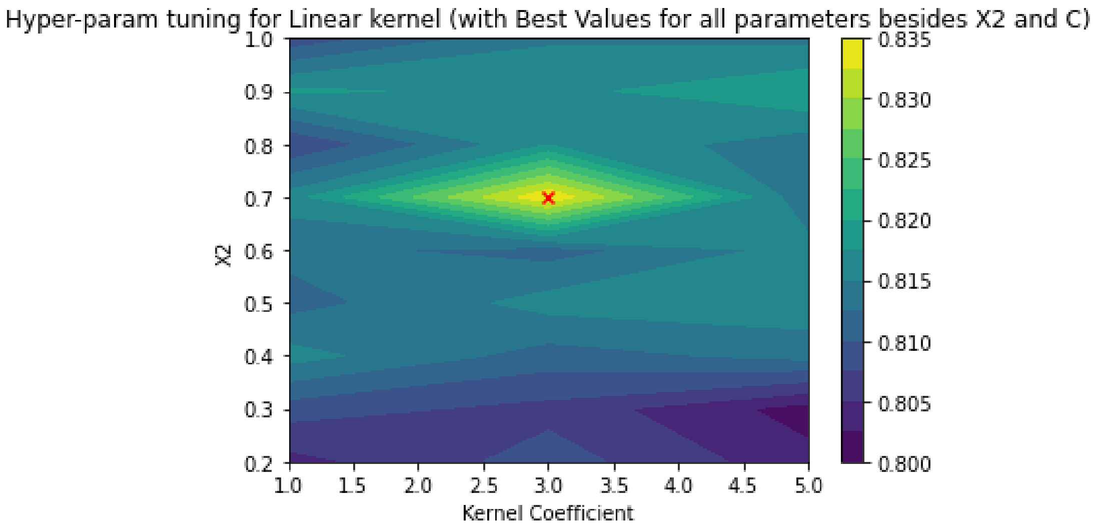

Figure 9: Heatmap of optimal hyper-parameters C and Weight for feature 55 for Linear Kernel

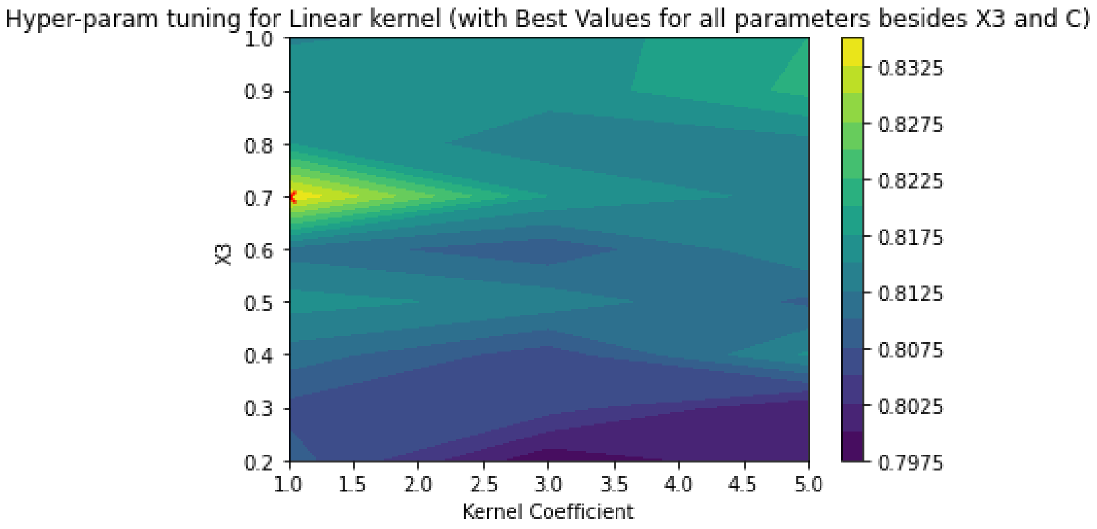

Figure 10: Heatmap of optimal hyper-parameters C and Weight for feature 5 for Linear Kernel

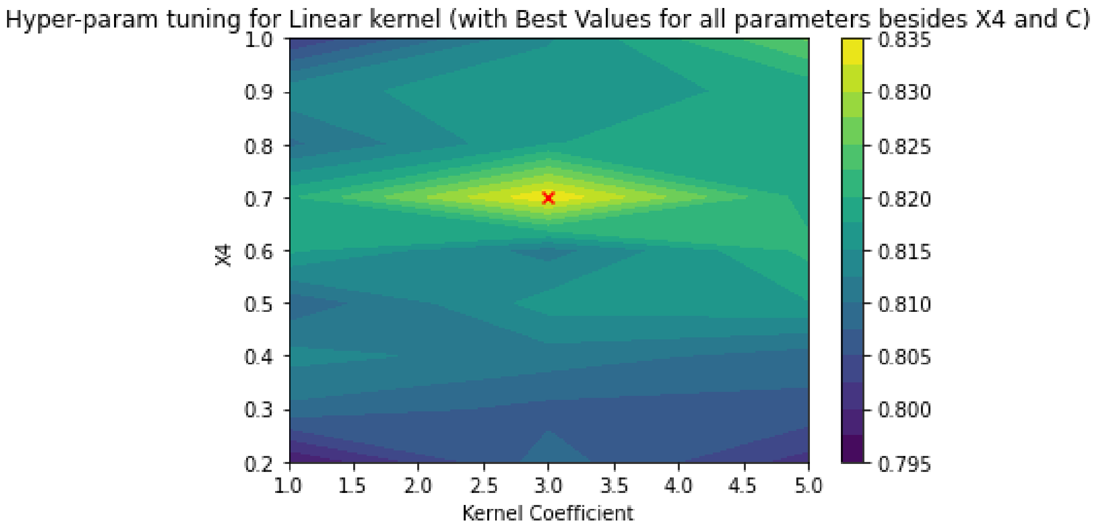

Figure 11: Heatmap of optimal hyper-parameters C and Weight for feature 317 for Linear Kernel

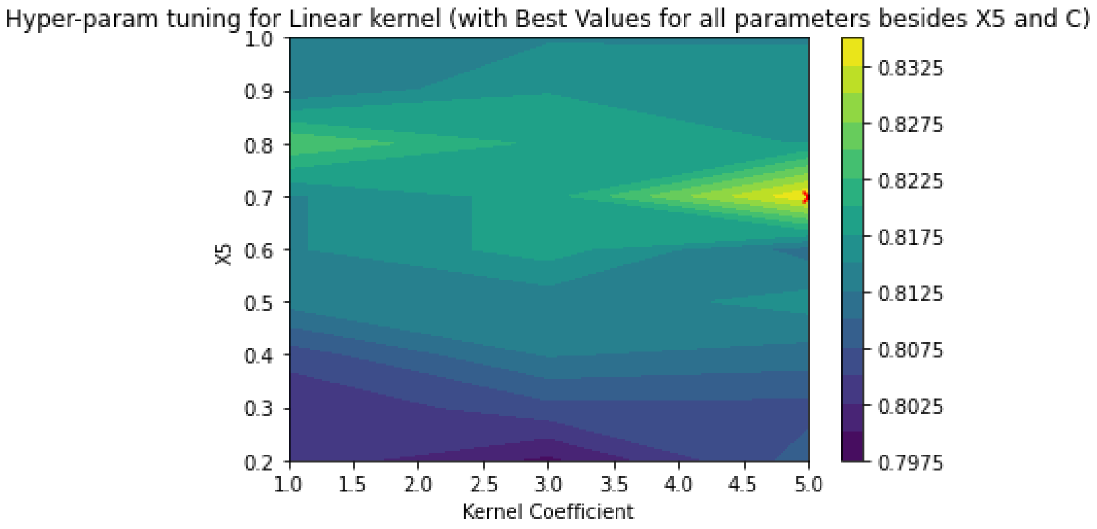

Figure 12: Heatmap of optimal hyper-parameters C and Weight for feature 180 for Linear Kernel

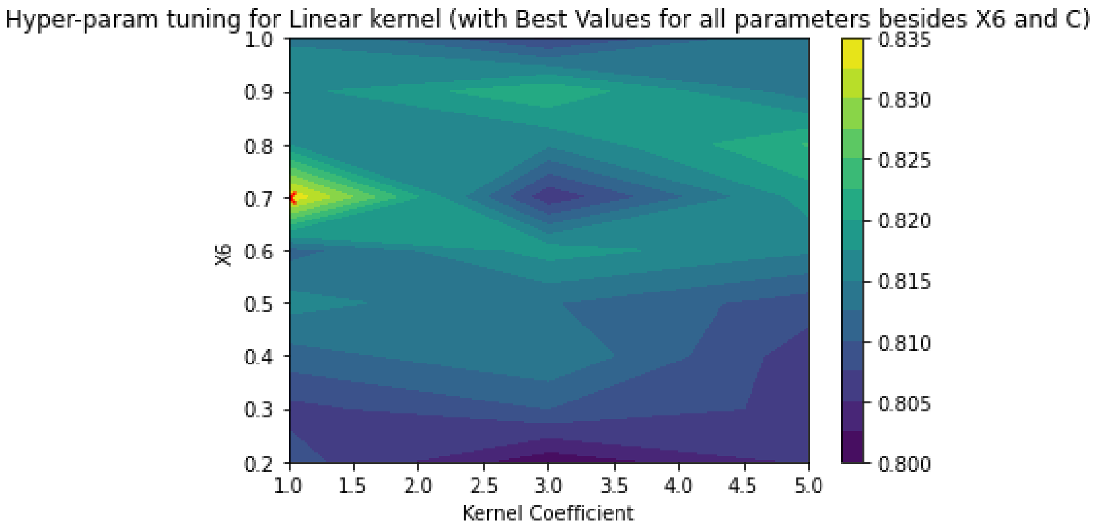

Figure 13: Heatmap of optimal hyper-parameters C and Weight for feature 177 for Linear Kernel