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

The purpose of this report is to evaluate the performance of a Support Vector Machine classifier (using 3 different kernels) and a Boosting based classifier, with regards to their ability to infer the sentiment of a movie review (i.e. is the sentiment of the movie review positive or negative).
The code used is in the provided notebook `Coursework2_final.ipynb`, which can also be found along with the report source files at the URL provided in Appendix A.

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. This correlation is derived from the training data alone, correlating each feature with the sentiment value.
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 5, which shows the same heatmap, but using the embeddings without performing chunking and overlapping, i.e. where sentences where truncated to 128 word items.

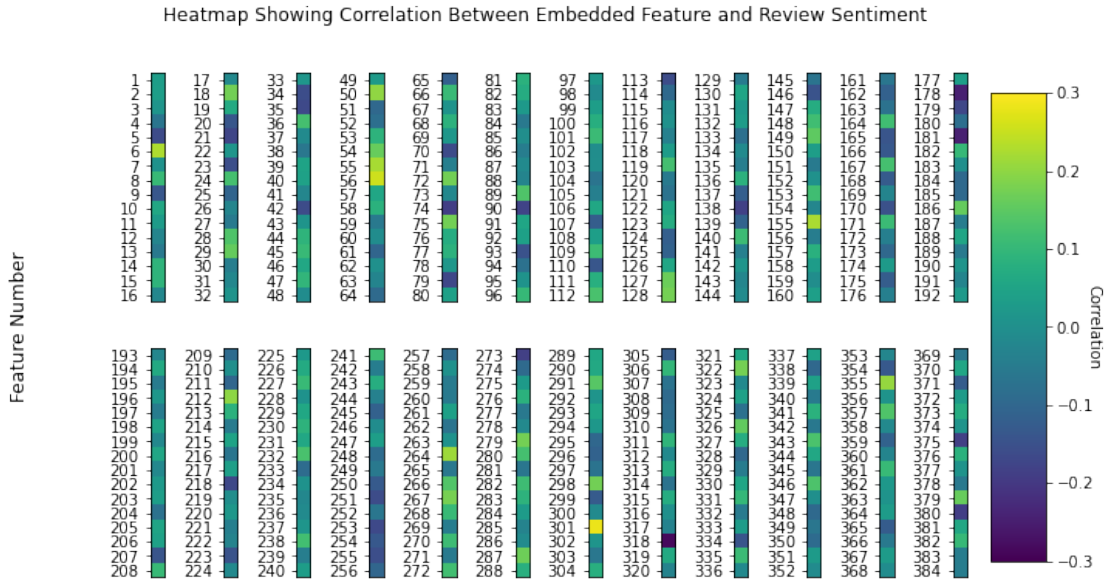

Figure 1: Heat-map showing correlation of embedding features to the sentence sentiment

2 Support Vector Machine (SVM)

2.1 Radial Basis Function (RBF)

The RBF kernel used was the built-in implementation provided within `sklearn.svm.SVC`.
The hyper-parameters for the RBF kernel model where `C`: the regularisation parameter (L2), `gamma`: a coefficient used by the kernel.
The hyper-parameters were optimised through 5-fold cross-validation, within a grid search of C (within the range of 0.1 - 1, with 9 total steps) and gamma (within the range of 0.5 - 3, with 8 total steps).
The hyper-parameter tuning produced the following values: C = 0.97, Gamma = 2.2857142857142856.
The tuning has been visualised within Figure 2, with the optimal hyper-parameters shown by the red cross.

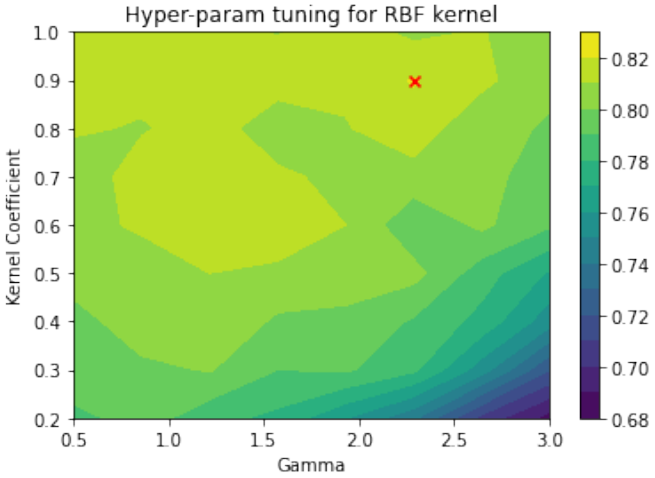

Figure 2: Contour of RBF kernel model hyper-parameter tuning

Once the hyper-parameters were tuned, the model was then trained on the training set, and evaluated against the testing set. This yielded the following metrics:
Accuracy: 0.9753333333333334, F1 Score: 0.9757377049180328, Sensitivity: 0.9674902470741222, False Negative rate: 0.032509752925877766, False Positive rate: 0.016415868673050615, Specificity: 0.9835841313269493.

The high accuracy indicates very little over-fitting.

2.2 Polynomial

The Polynomial kernel used was also a built-in implementation provided within `sklearn.svm.SVC`.
The hyper-parameters for the Polynomial kernel model where `C`: the regularisation parameter (L2), `gamma`: a coefficient used by the kernel, `degree`: the maximum exponent for the polynomial used.
The hyper-parameters were optimised through 5-fold cross-validation, within a grid search of C (within the range of 0.1 - 1, with 9 total steps), gamma (within the range of 0.5 - 3,

with 8 total steps), and degree (within the range of 1 - 3, with 3 total steps).
The hyper-parameter tuning produced the following values: C = 0.71, Gamma = 2.2857142857142856, Degree: 2.
Due to there being more than 2 parameters to tune, it isn't possible to visualise a contour for all dimensions. However you can perform a pair-wise contours, where you fix one parameter (using the optimal value) and plot the contour of the other parameters.
Contours can be found under subsection C.1, in Figure 6, Figure 7, and Figure 8.

Once the hyper-parameters were tuned, the model was then trained on the training set, and evaluated against the testing set. This yielded the following metrics:
Accuracy: 0.982, F1 Score: 0.9822485207100592, Sensitivity: 0.9713914174252276, False Negative rate: 0.02860858257477243, False Positive rate: 0.006839945280437756, Specificity: 0.9931600547195623.

2.3 Linear

The Linear kernel used was created by ourselves. It should work by applying a weight to 6 specific features identified from the embedding that are most correlated (3 positively, 3 negatively) to the review sentiment.
The features were: 300 (correlation: 0.282470), 55 (0.253391), 5 (0.225303), 317 (-0.285555), 180 (-0.248269), and 177 (-0.247295)

The hyper-parameters for the Linear kernel model where `C`: the regularisation parameter (L2), and 6 weights for each feature.
The hyper-parameters were optimised through 5-fold cross-validation, within a grid search of C (within the range of 0.1 - 1, with 9 total steps) and each of the 6 weights (within the range of 1 - 5, with 3 total steps for each feature).
The hyper-parameter tuning produced the following values: C = 0.7, weight for 300 = 3, weight for 55 = 3, weight for 5 = 1, weight for 317 = 3, weight for 180 = 5, and weight for 177 = 1.
As with the polynomial kernel model, a contour for all features isn't possible, however you can find contours for each weight paired with C under subsection C.2 in Figure 9, Figure 10, Figure 11, Figure 12, Figure 13, and Figure 14.

Once the hyper-parameters were tuned, the model was then trained on the training set, and evaluated against the testing set. This yielded the following metrics:
Accuracy: 0.8666666666666667, F1 Score: 0.8689384010484927, Sensitivity: 0.8621586475942783, False Negative rate: 0.1378413524057217, False Positive rate: 0.12859097127222982, Specificity: 0.8714090287277702.

The above metrics show that this kernel is both over-fitting and under-fitting, with a greater than 10% False Positive and Negative Rate.

2.4 Kernel Comparison

Under Appendix D you can find the confusion matrices for each model.
Along with the metrics above, and the confusion matrices, you can the RBF and Polynomial kernels significantly outperforming the Linear kernel.

This could possibly be due to the linear kernel being impacted by sentence length. The tables shown in Table 6, Table 4, and Table 2 show the RBF and Polynomial kernels more often correctly classifying shorter sentences. However the Linear kernel doesn't seem to have any correlation between sentence length and classification accuracy.

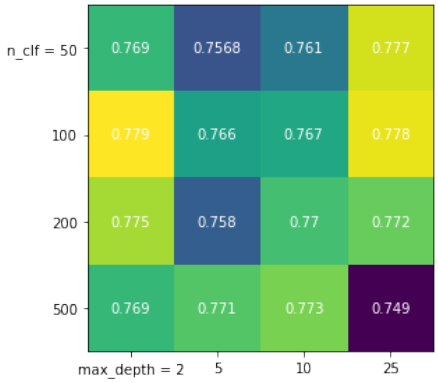

Figure 3: Hyper-parameter grid search for AdaBoost classifier

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 3 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 4 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

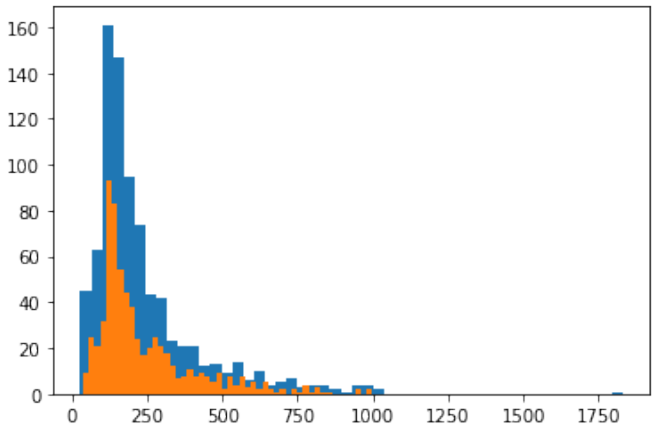

Figure 4: Lengths of correctly classified (blue) and misclassified (red) reviews.

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

C SVM Hyper-Parameter Tuning

C.1 Polynomial Kernel Hyper-Parameter Grid Search Heatmaps

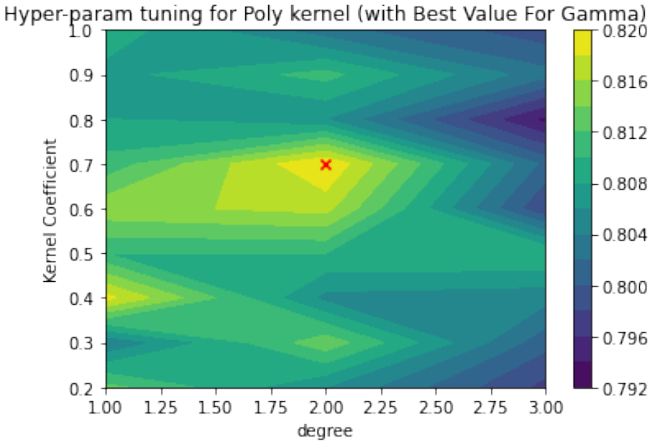

Figure 6: Contour of Polynomial kernel model hyper-parameter tuning (for C and degree)

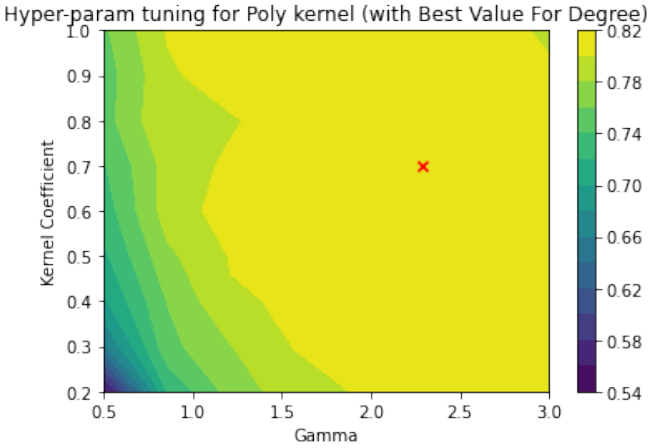

Figure 7: Heatmap of optimal hyper-parameters C and Gamma for Polynomial Kernel

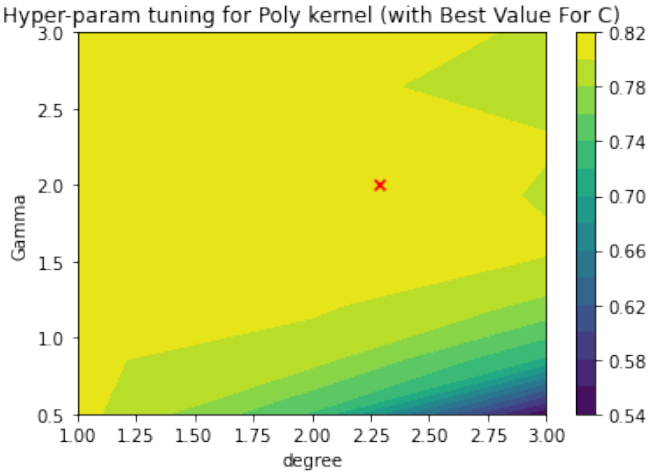

Figure 8: Heatmap of optimal hyper-parameters Gamma and Degrees for Polynomial Kernel

C.2 Linear Kernel Hyper-Parameter Grid Search Heat-Maps

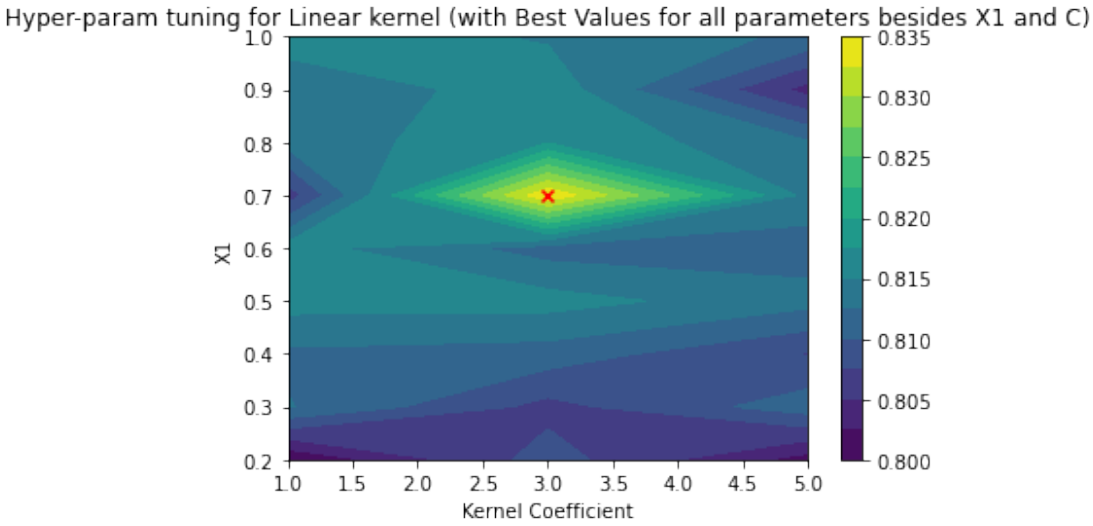

Figure 9: Heatmap of optimal hyper-parameters C and Weight for feature 300 for Linear Kernel

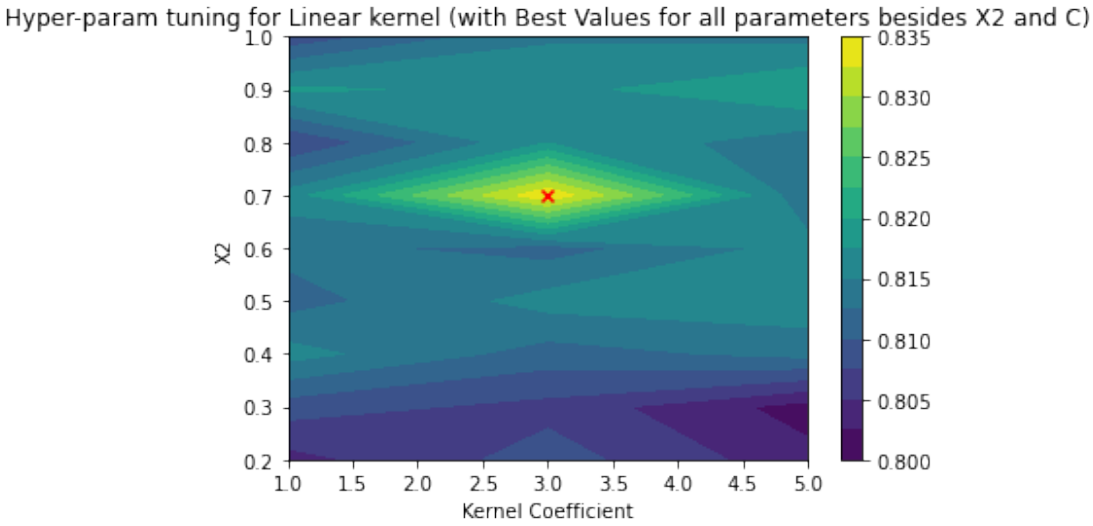

Figure 10: Heatmap of optimal hyper-parameters C and Weight for feature 55 for Linear Kernel

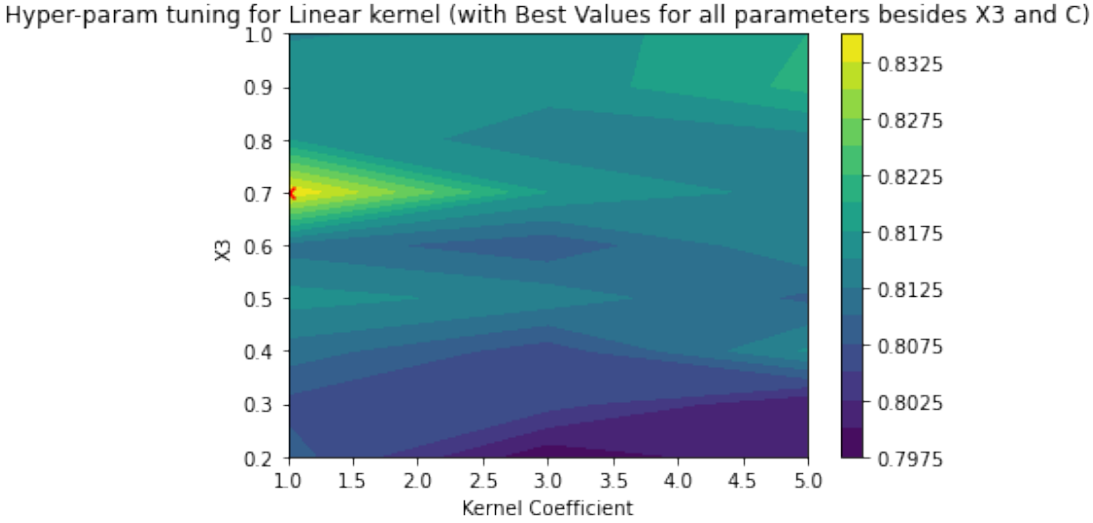

Figure 11: Heatmap of optimal hyper-parameters C and Weight for feature 5 for Linear Kernel

D SVM Evaluation

D.1 RBF Kernel Model Metrics

D.2 Polynomial Kernel Model Metrics

D.3 Linear Kernel Model Metrics

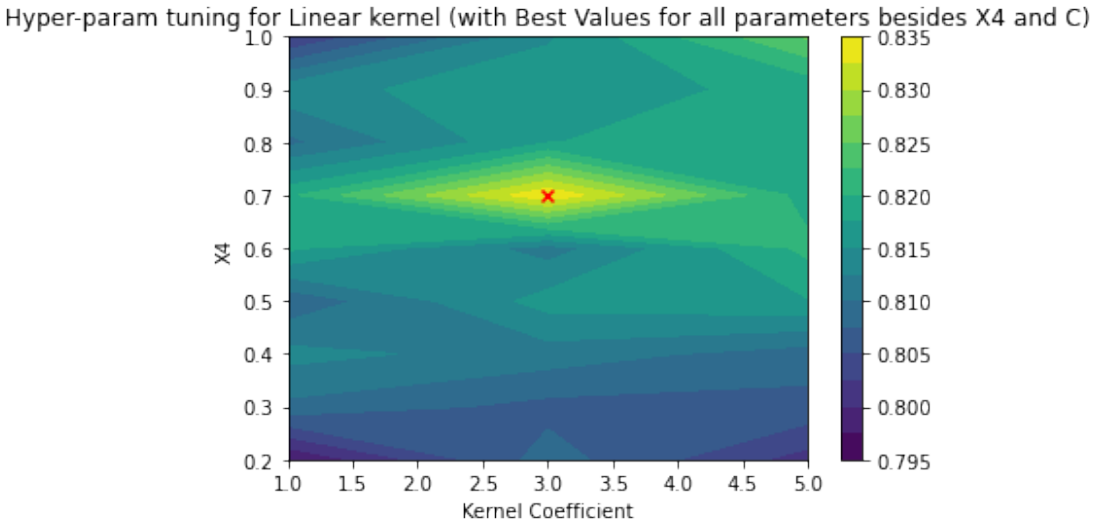

Figure 12: Heatmap of optimal hyper-parameters C and Weight for feature 317 for Linear Kernel

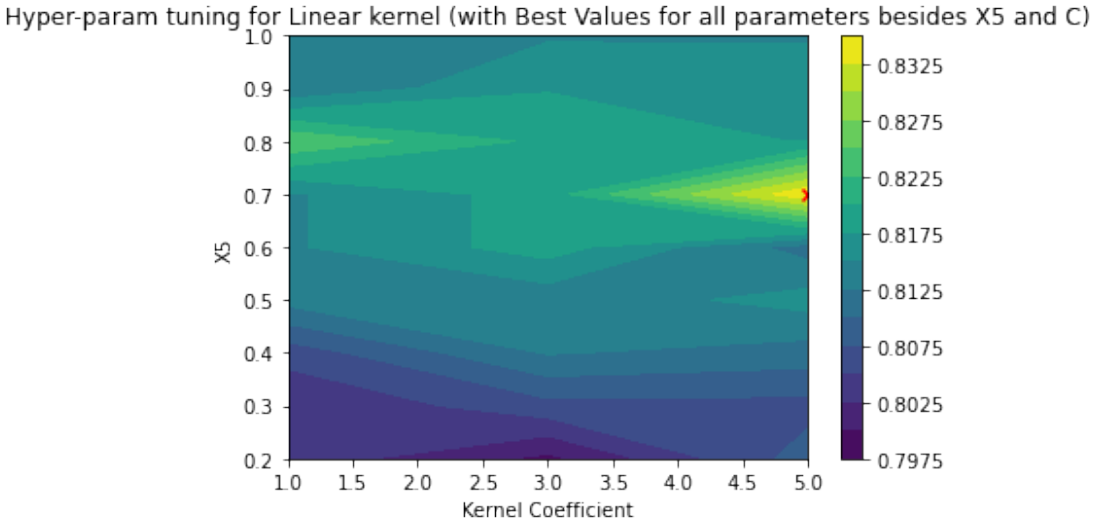

Figure 13: Heatmap of optimal hyper-parameters C and Weight for feature 180 for Linear Kernel

		Predicted	
		+	-
Actual	+	744	25
	-	12	25

Table 1: RBF Kernel Model Confusion Matrix

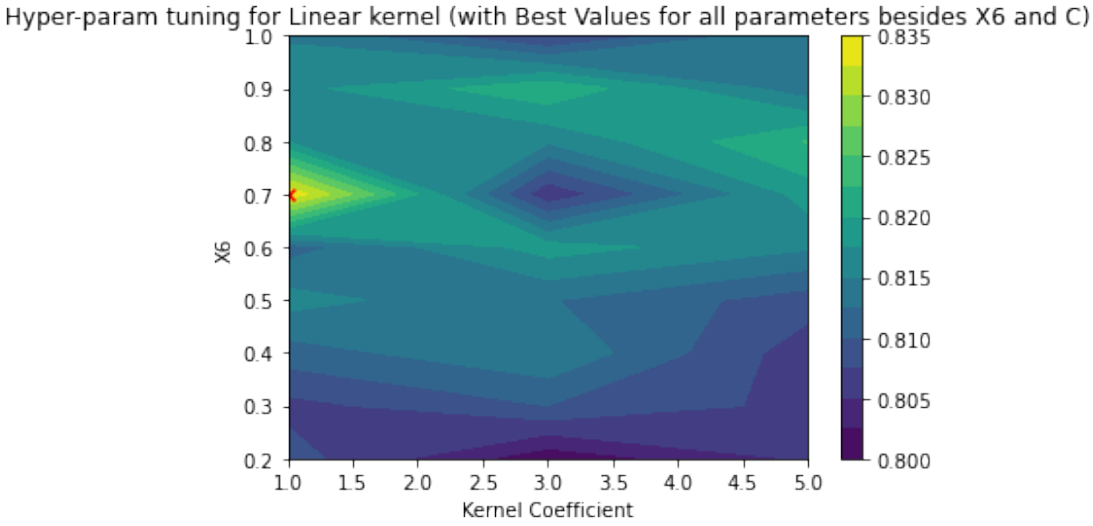

Figure 14: Heatmap of optimal hyper-parameters C and Weight for feature 177 for Linear Kernel

Classification	Sentence Count	Total Word Count	Average Sentence Length
False Negative	25	7945	317.8
False Positive	12	3831	319.25
True Negative	719	167818	233.4047288
True Positive	744	177046	237.9650538

Table 2: RBF Kernel Model impact of sentence length

		Predicted	
		+	-
Actual	+	747	22
	-	5	726

Table 3: Polynomial Kernel Model Confusion Matrix

Classification	Sentence Count	Total Word Count	Average Sentence Length
False Negative	22	7192	326.9090909
False Positive	5	1635	327
True Negative	726	170014	234.1790634
True Positive	747	177799	238.0174029

Table 4: Polynomial Kernel Model impact of sentence length

		Predicted	
		+	-
Actual	+	663	106
	-	94	637

Table 5: Linear Kernel Model Confusion Matrix

Classification	Sentence Count	Total Word Count	Average Sentence Length
False Negative	106	27228	256.8679245
False Positive	94	20946	222.8297872
True Negative	637	150703	236.5824176
True Positive	663	157763	237.9532428

Table 6: Linear Kernel Model impact of sentence length