Sentiment Analysis of Movie Reviews Using Support Vector Machines

A Comparative Study of Linear, RBF, and Polynomial SVM Kernels

Atanu Roy

1. Introduction:

Online reviews contain valuable opinions that influence consumer decisions. However, manually analyzing thousands of reviews is time-consuming. A machine learning model is needed to automatically classify text-based reviews as positive or negative based on sentiment.

In this project, I have addressed sentiment analysis as a binary text classification problem using Support Vector Machine (SVM)–based approaches. The study not only focuses on building an effective sentiment classifier but also on systematically comparing different SVM kernels, including Linear SVM, SVM with Radial Basis Function (RBF) kernel, and SVM with Polynomial kernel, to understand their behavior on high-dimensional text data.

For this project I have used Stanford Large Movie Review Dataset (IMDB). This dataset contains a total of 50,000 reviews split evenly into train and test sets.

My objective in this project is not only to build an accurate sentiment classifier, but also to empirically compare different SVM kernels and understand their behavior on high-dimensional text data.

2. Methodology:

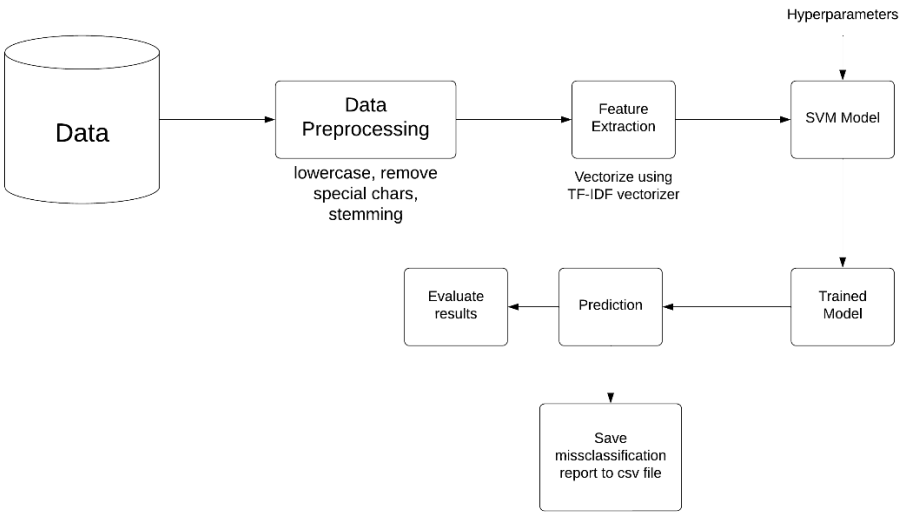
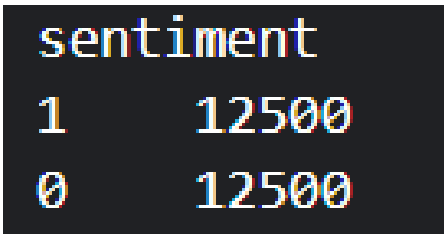


Fig 2.1: Methodology flowchart

I. Dataset Description:

I have used the Stanford Large Movie Review Dataset (IMDB) for this project. It consists of 50,000 English movie reviews labeled as positive or negative. The dataset is evenly balanced, with 25,000 positive and 25,000 negative reviews, and is pre-split into 25,000 training and 25,000 testing samples. This dataset is widely used as a benchmark for sentiment analysis tasks.



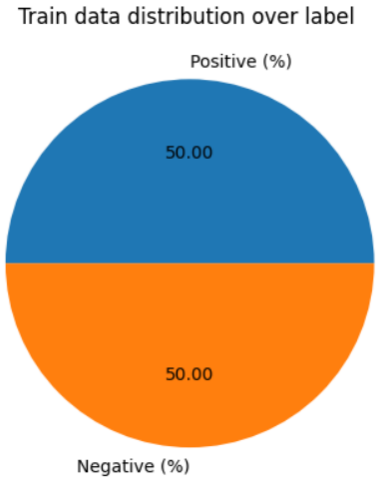


Fig 2.2: Train data distribution count

I have assigned positive reviews as numerical label 1 and negative reviews as numerical label 0. I have used the full training data for training of Linear SVM. But for SVM with RBF and Polynomial kernels I have only used 5000 samples of training data from the original training data.

II. Text Processing:

Before feature extraction and training the raw reviews are preprocessed to reduce noise and standardize text. This included lowercasing, removal of unnecessary characters, perform stemming. Preprocessing helps improve model generalization by ensuring that semantically identical words are treated consistently. Fig 2.2 and 2.3 shows the effect of test processing for a random review in dataset.

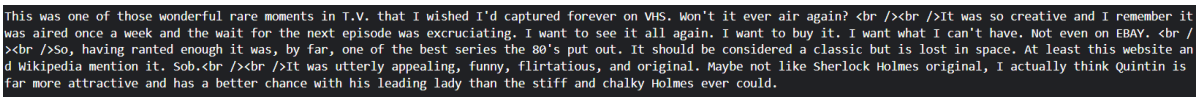


Fig 2.3: A sample review before text processing

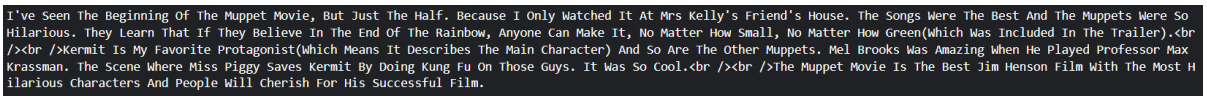


Fig 2.4: A sample review after text processing

III. Feature Extraction:

Purpose: Since machine learning models operate on numerical data, textual reviews are transformed into numerical feature vectors

Method Used: TF-IDF vectorization is used for this purpose. Term Frequency–Inverse Document Frequency (TF-IDF) converts text data to numerical data by assigning higher weights to words that are frequent within a review but rare across the corpus, making it effective for capturing sentiment-bearing terms. The raw text data is thus converted to very higher dimensional numeric data using TF-IDF vectorization.

Both unigrams and bigrams were used to capture local context such as negations. Sublinear term frequency scaling was applied to reduce the influence of very frequent words in long reviews.

For Linear SVM I have used a TF-IDF vectorizer with larger vocab (25000) and for SVM with RBF and Polynomial Kernel I have used smaller vocab (3500). This is done as due to extremely large time complexity.

The following configurations were used for defining the TF-IDF vectorizer:

i. For Linear SVM:
   a. Maximum vocal (features) is limited to 25000. This balances training speed and most variety of words are covered with in this limit
   b. Unigram and bigram both used. This is because unigram alone can destroy meaning. For example, for words like "not good", "very bad" will be broke to "not", "very bad", "good", "very", "bad" and meaning will be lost. Also, trigram is not used as it will be very rare.
   c. Min document frequency is set to 5: This will remove extremely rare words (that appears in ≥0.02% of reviews). And will remove noise without discarding useful sentiment words.
   d. Max document frequency is set to 0.92: This will remove too common words (appearing in >92% of reviews). Very common words carry zero sentiment information.

e. Log scaled term frequency used. This will prevent long reviews from dominating. $TF_{log} = 1 + \log(TF)$

ii. For SVM with RBF and Polynomial kernels:
   a. Maximum vocabulary (features) is limited to 3500. This is done to reduce the feature space of data and reduce computation time.
   b. Unigram and bigram both used.
   c. Min document frequency is set to 5.
   d. Max document frequency is set to 0.92.
   e. Log scaled term frequency used.

```
For Linear SVM:
For train: Shape of X: (25000, 25000), Shape of y: (25000,)
For test: Shape of X: (25000, 25000), Shape of y: (25000,)
```

Fig 2.5: Shape of feature extracted train and test data used for Linear SVM

```
For SVM with RBF and Polynomial kernels:
For train: Shape of X: (5000, 3500), Shape of y: (5000,)
For test: Shape of X: (25000, 3500), Shape of y: (25000,)
```

Fig 2.6: Shape of feature extracted train and test data used for SVM with RBF and Polynomial Kernel

IV. Model Selection:

Support Vector Machines (SVMs) is chosen due to their strong performance in high-dimensional spaces and their effectiveness in text classification tasks. Three variants are evaluated: Linear SVM, SVM with Radial Basis Function (RBF) kernel, and SVM with Polynomial kernel. These kernels differ in how they model decision boundaries.

I have compared between SVMs with different kernel and using different hyperparameters for each.

V. Hyperparameter Tuning and Evaluation:

Hyperparameter tuning is performed to study the effect of model complexity on performance. For Linear SVM, different values of the regularization parameter C were evaluated. For RBF and Polynomial kernels, a limited set of C, gamma, and degree values was tested due to computational constraints. Model performance was evaluated using accuracy, precision, recall, F1-score, confusion matrices, and ROC–AUC curves to provide both threshold-dependent and threshold-independent analysis.

The following hyperparameters are tuned:

i. C: Regularization parameter. Controls the tradeoff between margin size and classification errors.
ii. Gamma: Kernel coefficient (RBF & Poly kernels). Controls how far the influence of a single data point reaches.
iii. Degree: For polynomial kernel only. Controls the degree of the polynomial used to separate data.

For Linear SVM, 4 models were train with following configurations:

i. C = 0.01

ii. C = 0.1
iii. C = 1
iv. C = 10

For SVM with RBF kernel, 4 models were train with following configurations:

i. C = 0.1 & gamma = 0.001
ii. C = 0.1 & gamma = 0.01
iii. C = 1 & gamma = 0.001
iv. C = 1 & gamma = 0.01

For SVM with Polynomial kernel, 4 models were train with following configurations:

i. C = 0.1 & degree = 2
ii. C = 1 & degree = 2
iii. C = 0.1 & degree = 3
iv. C = 1 & degree = 3

3. Results:

I. Linear SVM

4 Linear SVM models were trained with 4 different values of C: C=0.01, 0.1, 1, 10.
The experimental results of Linear SVM models are shown in Fig 3.1, Fig 3.2 and Fig 3.3. Linear SVM achieved the best overall performance among all models. The optimal value of the regularization parameter was C = 0.1, which resulted in the highest F1-score (0.887) and ROC–AUC (0.956). At this value of C, Precision and Recall were also highly and balanced. Performance improved as C increased from 0.01 to 0.1 and then degraded after that for larger values, indicating a tradeoff between bias and variance.
The table 3.1 below shows the various evaluation scores for the models.

	Model	C	Gamma	Degree	Accuracy (%)	Precision	Recall	F1 Score	AUC
0	Linear SVM C=0.01	0.01	None	None	86.520	0.851912	0.88408	0.867698	0.940581
1	Linear SVM C=0.1	0.10	None	None	88.720	0.881824	0.89424	0.887989	0.956317
2	Linear SVM C=1	1.00	None	None	87.672	0.879391	0.87320	0.876285	0.949139
3	Linear SVM C=10	10.00	None	None	85.444	0.859589	0.84728	0.853390	0.932740

Table 3.1: Accuracy, Precision, Recall, F1 Score, AUC for 4 Linear SVM models with different C values

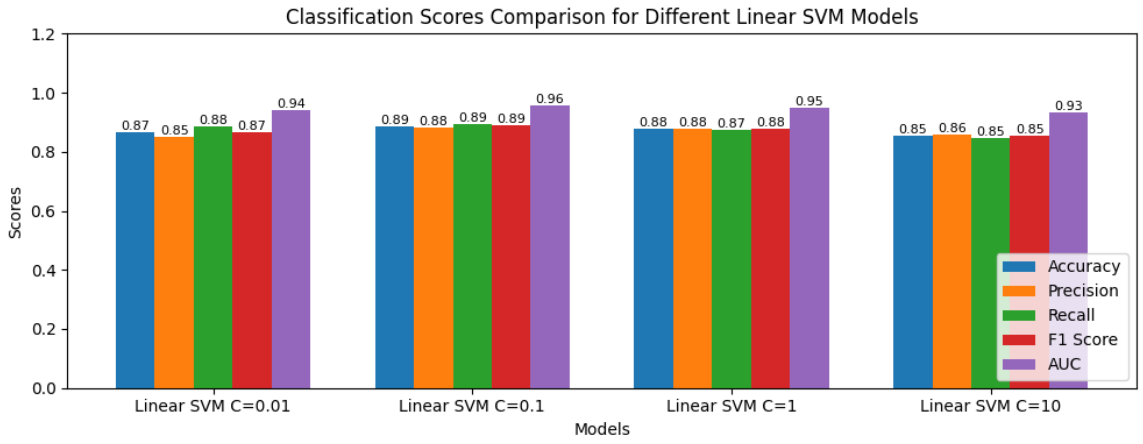


Fig 3.1: Comparison between 4 Linear SVM models trained with configurations mentioned above

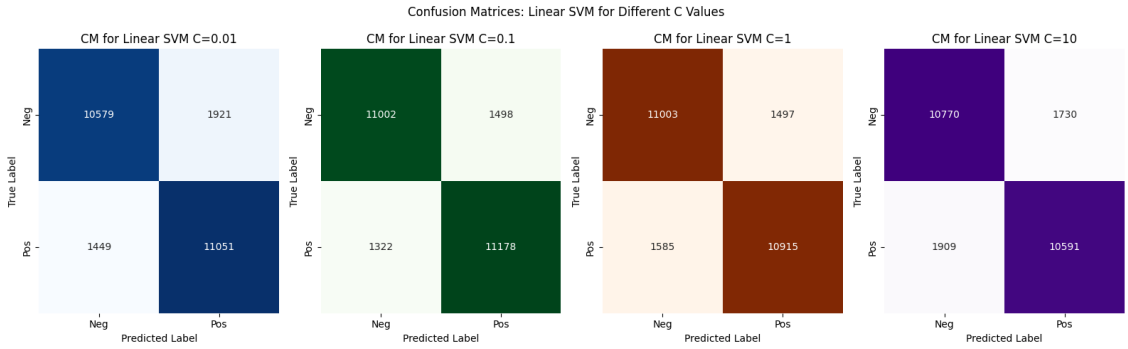


Fig 3.2: Heat Map Confusion Matrix of 4 Linear SVM models trained with configurations mentioned above

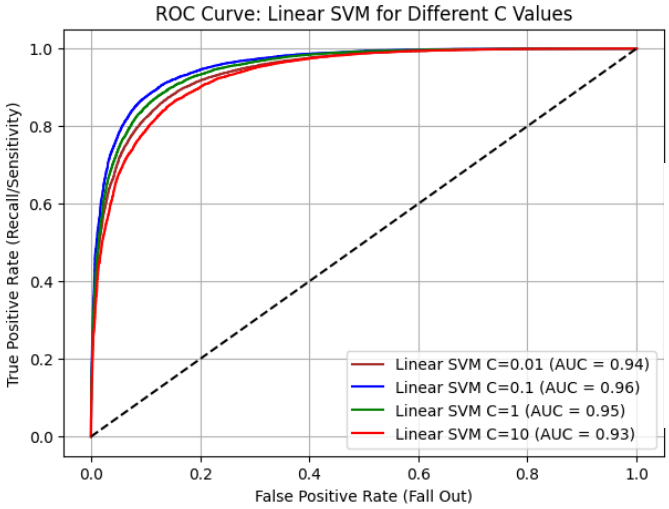


Fig 3.3: ROC curve of 4 Linear SVM models trained with configurations mentioned above

II. SVM with RBF kernel

4 Linear SVM models with RBF kernels were trained with different values of C and gamma: C=0.1 & gamma=0.001, C=1 & gamma=0.01, C=0.1 & gamma=0.01, C=1 & gamma=0.001. The experimental results of RBF kernel SVM models are shown in Fig 3.4, Fig 3.5 and Fig 3.6. RBF SVM models were trained on a reduced subset of the training data due to high computational cost.

All models gave similar results for all tested values of C and gamma. The model achieved accuracy close to random guessing (~51%), with very high recall (0.99) but very low precision (0.50). ROC–AUC values remained around 0.69, indicating weak class separation. Very high recall and low precision suggests that the model is predicting almost everything as positive label, indicating unstable decision behavior and this can also be verified by observing the confusion matrix heat map on Fig 3.5.

The table 3.2 shows the various evaluation scores for the models.

	Model	C	Gamma	Degree	Accuracy (%)	Precision	Recall	F1 Score	AUC
0	RBF SVM (C=0.1, g=0.001)	0.1	0.001	None	50.648	0.503273	0.99632	0.668743	0.581635
1	RBF SVM (C=0.1, g=0.01)	0.1	0.010	None	50.648	0.503273	0.99632	0.668743	0.581634
2	RBF SVM (C=1, g=0.001)	1.0	0.001	None	50.648	0.503273	0.99632	0.668743	0.581635
3	RBF SVM (C=1, g=0.01)	1.0	0.010	None	50.648	0.503273	0.99632	0.668743	0.581634

Table 3.2: Accuracy, Precision, Recall, F1 Score, AUC for 4 SVM models with RBF kernel with different C and gamma values

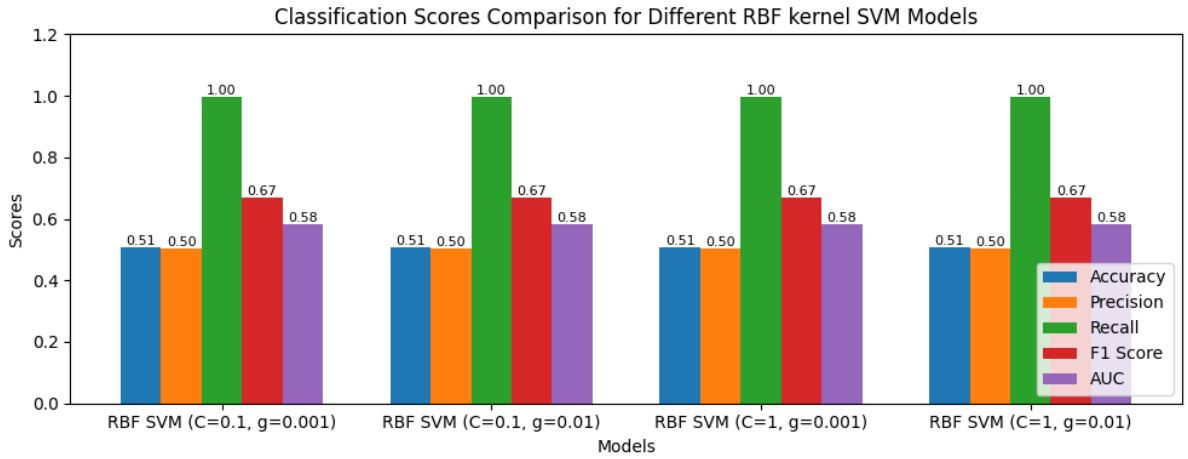


Fig 3.4: Comparison between 4 SVM models with RBF kernel trained with configurations mentioned above

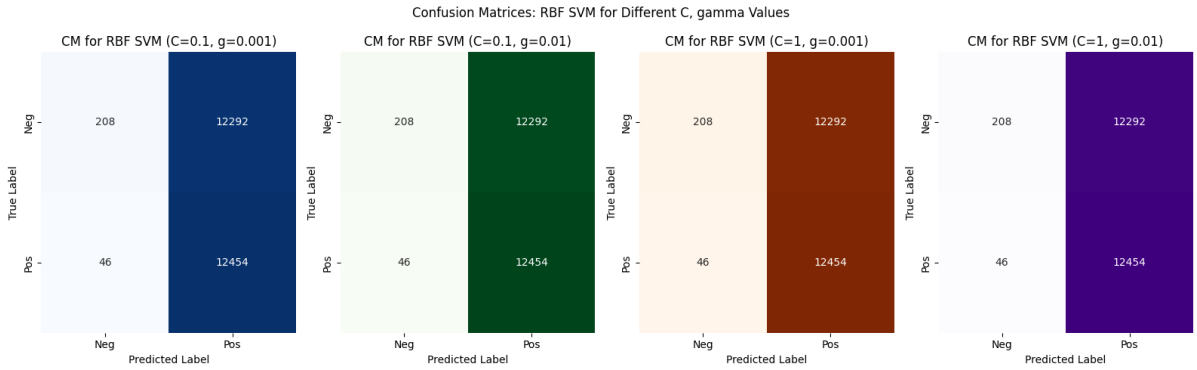


Fig 3.5: Heat Map Confusion Matrix of 4 SVM models with RBF kernel trained with configurations mentioned above

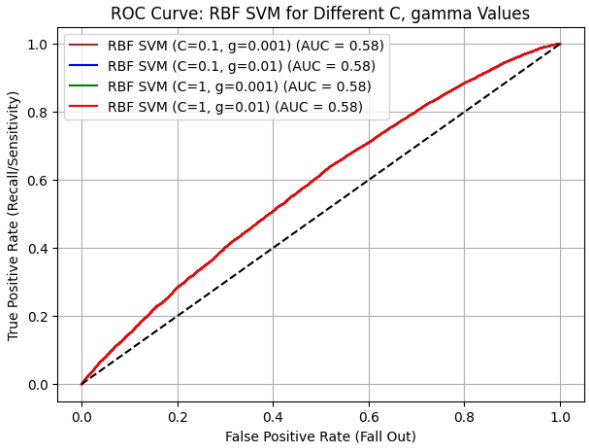


Fig 3.6: ROC curve of 4 SVM models with RBF Kernel trained with configurations mentioned above

III. SVM with Polynomial kernel

4 Linear SVM models with RBF kernels were trained with different values of C and degree: C=0.1 & degree=2, C=1 & degree=2, C=0.1 & degree=3, C=1 & degree=3.

The experimental results of Polynomial kernel SVM models are shown in Fig 3.7, Fig 3.8 and Fig 3.9. Polynomial SVM models were trained on a reduced subset of the training data due to high computational cost.

All the Polynomial SVM models tested across all values of C and degree showed near-random performance across all tested configurations. Accuracy remained close to 50%, and ROC–AUC values were approximately 0.56. High recall combined with low precision suggests that the model is predicting almost everything as positive label, indicating unstable decision behavior and this can also be verified by observing the confusion matrix heat map on Fig 3.8. The table 3.3 below shows the various evaluation scores for the models.

	Model	C	Gamma	Degree	Accuracy (%)	Precision	Recall	F1 Score	AUC
0	Poly SVM (C=0.1, d=2)	0.1	None	2.0	50.000	0.500000	1.00000	0.666667	0.575655
1	Poly SVM (C=0.1, d=3)	0.1	None	3.0	50.000	0.500000	1.00000	0.666667	0.569871
2	Poly SVM (C=1, d=2)	1.0	None	2.0	53.596	0.523697	0.79472	0.631351	0.558531
3	Poly SVM (C=1, d=3)	1.0	None	3.0	50.784	0.503984	0.99184	0.668356	0.564819

Table 3.3: Accuracy, Precision, Recall, F1 Score, AUC for 4 SVM models with RBF kernel with different C and degree values

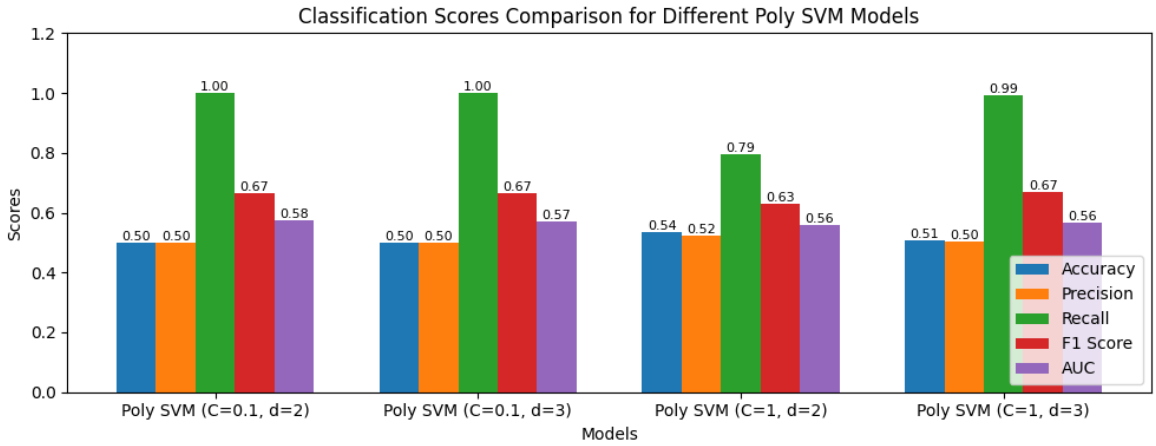


Fig 3.7: Comparison between 4 SVM models with Polynomial kernel trained with configurations mentioned above

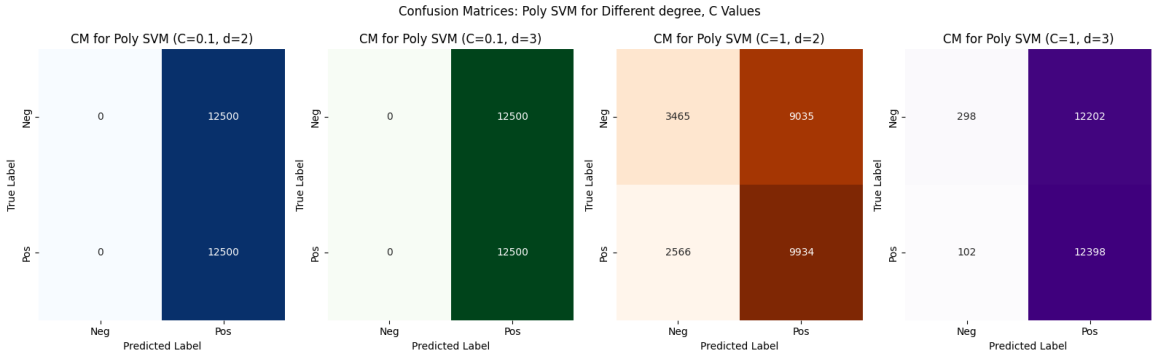


Fig 3.8: Heat Map Confusion Matrix of 4 SVM models with Polynomial kernel trained with configurations mentioned above

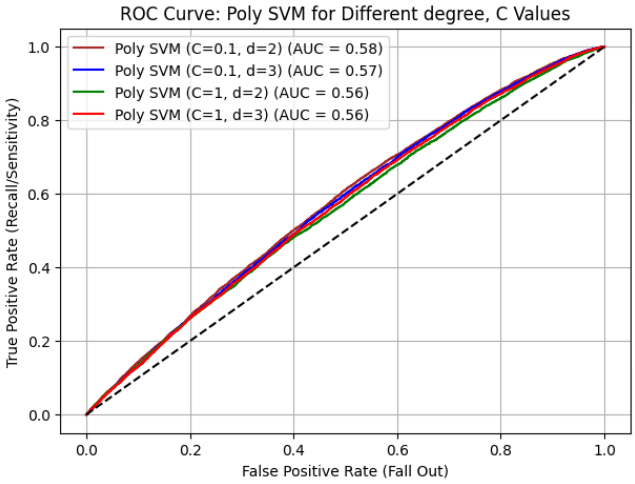


Fig 3.9: ROC curve of 4 SVM models with Polynomial Kernel trained with configurations mentioned above

IV. Final Comparison

From Table 3.4 and Fig: 3.10 Linear SVM significantly outperforms both RBF and Polynomial SVMs across all evaluation metrics.

Also, the optimal value of hyperparameter C for Linear SVM is C=0.1, using which accuracy of 88.72%, F1 score of 0.887 ad AUC of 0.956 is achieved with a good balance of Precision and Recall.

	Model	C	Gamma	Degree	Accuracy (%)	Precision	Recall	F1 Score	AUC	Kernel
0	Linear SVM C=0.01	0.01	N/A	N/A	86.520	0.851912	0.88408	0.867698	0.940581	Linear
1	Linear SVM C=0.1	0.10	N/A	N/A	88.720	0.881824	0.89424	0.887989	0.956317	Linear
2	Linear SVM C=1	1.00	N/A	N/A	87.672	0.879391	0.87320	0.876285	0.949139	Linear
3	Linear SVM C=10	10.00	N/A	N/A	85.444	0.859589	0.84728	0.853390	0.932740	Linear
4	RBF SVM (C=0.1, g=0.001)	0.10	0.001	N/A	50.648	0.503273	0.99632	0.668743	0.581635	RBF
5	RBF SVM (C=0.1, g=0.01)	0.10	0.01	N/A	50.648	0.503273	0.99632	0.668743	0.581634	RBF
6	RBF SVM (C=1, g=0.001)	1.00	0.001	N/A	50.648	0.503273	0.99632	0.668743	0.581635	RBF
7	RBF SVM (C=1, g=0.01)	1.00	0.01	N/A	50.648	0.503273	0.99632	0.668743	0.581634	RBF
8	Poly SVM (C=0.1, d=2)	0.10	N/A	2.0	50.000	0.500000	1.00000	0.666667	0.575655	Polynomial
9	Poly SVM (C=0.1, d=3)	0.10	N/A	3.0	50.000	0.500000	1.00000	0.666667	0.569871	Polynomial
10	Poly SVM (C=1, d=2)	1.00	N/A	2.0	53.596	0.523697	0.79472	0.631351	0.558531	Polynomial
11	Poly SVM (C=1, d=3)	1.00	N/A	3.0	50.784	0.503984	0.99184	0.668356	0.564819	Polynomial

Table 3.4: Accuracy, Precision, Recall, F1 Score, AUC for all 12 SVM models 4 with Linear, 4 with RBF and 4 with Polynomial kernel with different hyperparameters

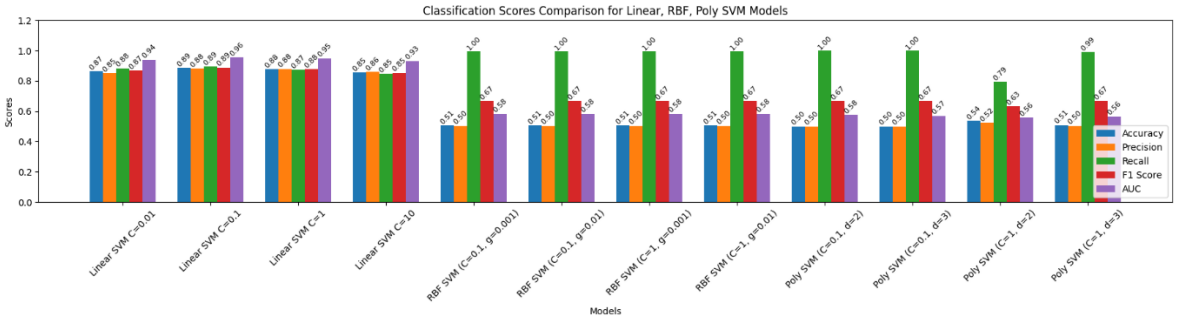


Fig 3.10: Comparison between all 12 SVM models 4 with Linear, 4 with RBF and 4 with Polynomial kernel with different hyperparameters

4. Discussion:

I. Why Linear SVM Performed Best?

The Linear SVM achieved the best performance among all evaluated models.

This is because TF-IDF representations of text data are high-dimensional and sparse, and in these cases linear classifiers are known to perform well. In such feature spaces, positive and negative sentiment reviews often become linearly separable. Linear SVM effectively uses this property by learning a stable decision boundary that generalizes well to unseen data while remaining computationally efficient.

II. Bias–Variance Tradeoff (Linear SVM)

The effect of the regularization parameter C in Linear SVM clearly demonstrated the bias–variance tradeoff.

When C is very small, the model gets more constrained and is unable to capture all sentiment-related patterns, leading to underfitting. As C increases, model performance improved and reached an optimal point (C=0.1 in my case), where both precision and recall were well balanced. Beyond this point, further increasing C caused a decline in performance, indicating overfitting.

This shows that moderate regularization provides the best balance between model simplicity and flexibility.

III. Why RBF and Polynomial Kernels Failed?

SVMs with RBF and Polynomial kernels performed significantly poor than Linear SVM in this project. These kernels rely on distance-based similarity measures or complex feature interactions, which are not well suited for sparse TF-IDF vectors that is created for reviews here. In high-dimensional text spaces, distance metrics become unreliable and noise is easily amplified. As a result, both RBF and Polynomial SVMs struggled to create a meaningful decision boundary and often collapsed into predicting a single class (positive in my case). Due to this the model is just good as random guesses, which can also be verified from looking at the ROC curves plotted in Fig 3.6 and Fig 3.9.

IV. Limitations and Future Work

In this project I have focused on classical machine learning methods using TF-IDF features and Support Vector Machines. These approaches are effective, but they do not capture contextual or semantic information beyond word frequency.

Future work could be on using modern deep learning methods, such as transformer-based models (e.g., BERT), which can model contextual relationships between words. Additionally, experimenting with word embeddings or hybrid models may further improve sentiment classification performance.

V. Misclassification Analysis

To better understand the limitations of the best-performing Linear SVM model (C = 0.1), I have conducted an error analysis using the misclassified samples from the test set. A separate report was generated containing the raw and processed reviews along with basic text statistics such as number of characters, words, and sentences.

Appendix A contains the misclassified review samples produced by the best Linear SVM model.

After analyzing the misclassified reviews, I founded the following things:

1. Long reviews with mixed sentiments:
   Many misclassified reviews are: long and contains both positive and negative expressions.
   Eg: "I didn't expect much… some parts were boring… but overall it was surprisingly enjoyable."
   This mixed sentiment creates conflicting signals.
2. Negation and contrast structures:
   Phrases involving: "not", "but", "however" are misclassified as TF-IDF represents text as independent weighted tokens, it cannot fully capture the scope of negation or contrast, leading to incorrect sentiment interpretation.
   Eg: "didn't expect it to be good, but…". Here the model counts "good" positively even when it is negated.
3. Ironic Sentiments:
   Several misclassified reviews are mildly positive or mildly negative.
   Eg: "It's not the worst thing you'll ever see.", "I suppose it has its moments."
   These are lacking presence of strong sentiment words.
   In such cases, conflicting sentiment cues result in ambiguous feature representations, making it difficult for a linear classifier to assign a correct label.
4. Uncommon Vocab:
   Some reviews contain rare words. Since I have set the minimum document frequency to 5, so rare phrases may not survive TF-IDF filtering. These words are dropped as they are present in very less numbers.
5. Very short reviews:
   Some reviews are extremely short. Eg: "Disappointing. Expected more."
   With very few words, model has very low evidence to decide.
6. Misclassifications occurred across wide range of reviews across all length and types. This indicates that misclassification errors are not strongly correlated with review size but rather with linguistic complexity and contextual structure.

Overall, these findings highlight the inherent limitations of TF-IDF features and linear classifiers in handling language phenomena such as negation, mixed sentiment, and implicit opinions. Minor improvements may be achieved through enhanced preprocessing and feature engineering. Fundamentally this requires models capable of capturing semantic context.

5. References:

i. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

ii. Yaoyong Li, Kalina Bontcheva, Hamish Cunningham Department of Computer Science. Adapting SVM for Natural Language Learning: A case Study Involving Information Extraction. The University of Sheffield, UK

iii. Chetna Khaparde. Sentiment Analysis Using SVM Classifier in Data Mining: A Machine Learning Approach. Volume 1 Issue 1 January-March 2025. International Journal of Computer Science & Information Technology.

iv. Pang, B., Lee, L., then Vaithyanathan. (2002). thumbs up Machine learning methods of sentiment classification. 10, 79–86. ACL-02 Conference on Empirical Methods in Natural Language Processing proceedings

v. Support Vector Machines - scikit-learn 1.8.0 documentation

vi. NumPy documentation - NumPy v2.4 Manual

vii. User Guide - pandas 2.3.3 documentation

Appendix A: Supplementary Materials

The following supplementary materials are provided to support the experiments and analysis presented in this report:

- **Misclassification Report (CSV)**
  Contains detailed information about misclassified samples from the best-performing Linear SVM model (with C=0.1), including actual and predicted labels, raw and processed review text, and basic text statistics.
- **Implementation Notebook (Jupyter Notebook)**
  Includes the complete implementation of data preprocessing, feature extraction, model training, hyperparameter tuning, evaluation, comparison and visualization steps used in this project.

The supplementary files are available at:

- Misclassified Report (CSV)
- Implementation Notebook (Jupyter Notebook)