21KHMT2

Introduction to Machine Learning

Final project
Training SVM to classify fashion images

Students

21127122 - Hồ Thanh Nhân

21127740 - Đoàn Nam Thắng

Lecturers

Phạm Trọng Nghĩa

Bùi Duy Đăng

Nguyễn Ngọc Đức

Contents

1 - Overall

1.1 Team members

- 21127122 - Hồ Thanh Nhân
- 21127740 - Đoàn Nam Thắng

1.2 Video link

- Youtube video link: https://www.youtube.com/watch?v=ZFxNLmr52ZE

2 - Data preprocessing

2.1 Split the data into training and validation sets

- train_test_split: This function splits the dataset into training and validation sets.
- test_size=10000: Specifies that 10,000 samples will be used for validation, and the rest will be used for training.
- random_state=42: Sets the seed for the random number generator, ensuring reproducibility when splitting the dataset.

2.2 Show Scree plot of the raw data

- The Scree Plot is a graphical representation of the eigenvalues obtained during Principal Component Analysis (PCA). It is crucial for determining the appropriate number of principal components to retain based on the explained variance.
- ***Y-Axis Values:*** The Scree Plot for Explained Variance shows a point at 1.2 x $10\hat{6}$, another at 0.8 x $10\hat{6}$, with numerous points around 0.2 x $10\hat{6}$, and a subsequent gradual decrease. The significant drop occurs at the point corresponding to the number of principal components in the range of 30-40, forming an "elbow" in the plot.
- ***Elbow Point:*** The elbow point in the Scree Plot is a critical indicator. Beyond this point, the rate of decrease in eigenvalues slows down, and the explained variance becomes less significant. The elbow point signifies a balance between retaining sufficient information and reducing dimensionality.
- ***Horizontal Region:*** Following the elbow point, there is a relatively horizontal region. The eigenvalues in this region are less crucial for explaining the variance in the data. Retaining components beyond this region might not contribute significantly to capturing the essential patterns.

2.3 Standardize the data using MinMaxScaler

- MinMaxScaler is used to scale the features of the training and validation sets (train_X

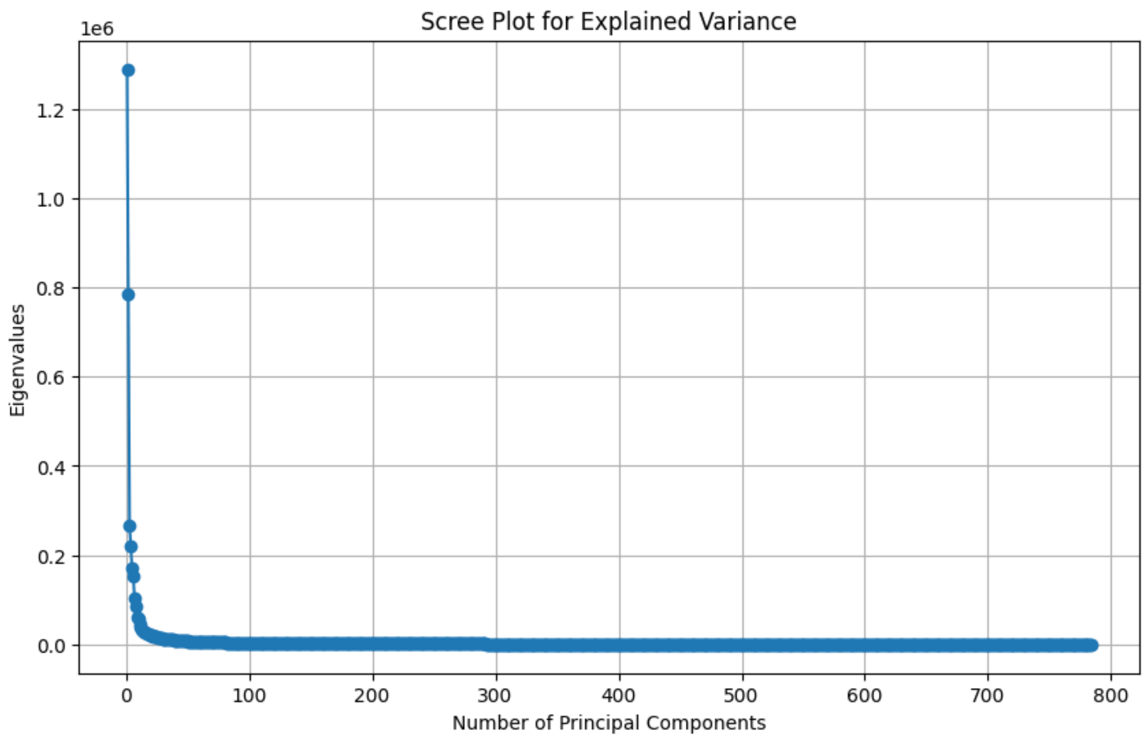


Figure 1: The Scree plot of the raw data

and val_X) to a specified range. This ensures that all features have similar scales, preventing certain features from dominating the model training process.

- In the context of SVMs, scaling is particularly crucial because SVMs are distance-based algorithms, and the scale of features can significantly impact the decision boundary. Using Min-Max scaling, which scales features to a specific range, helps maintain the integrity of the data while ensuring that all features contribute equally to the model's training process.

- The scaling process standardized the eigenvalues, making them comparable and facilitating a clearer identification of key points on the Scree Plot. This normalization is crucial for PCA, ensuring that each feature contributes proportionally to the principal components.

2.4 Use PCA to reduce the dimensionality of the data

- PCA is a dimensionality reduction technique that transforms the original features into a lower-dimensional space while retaining most of the variance in the data.

- MinMax Scaling: The scaling process standardized the eigenvalues, making them comparable and facilitating a clearer identification of key points on the Scree Plot. This normalization is crucial for PCA, ensuring that each feature contributes proportionally to the principal components.

- Explained Variance: Selecting 0.95 as the threshold for explained variance implies that it will retain components that collectively explain 95% of the total variance in the data. This is a common heuristic to balance dimensionality reduction and information retention.

- Number of Principal Components: In this case, this corresponds to approximately 188 principal components. Choosing this number ensures a substantial reduction in dimensionality while retaining a high percentage of the original data's variance.

- pca.fit_transform(train_X_std) fits PCA to the scaled training data and transforms it. The same transformation is then applied to the scaled validation data.

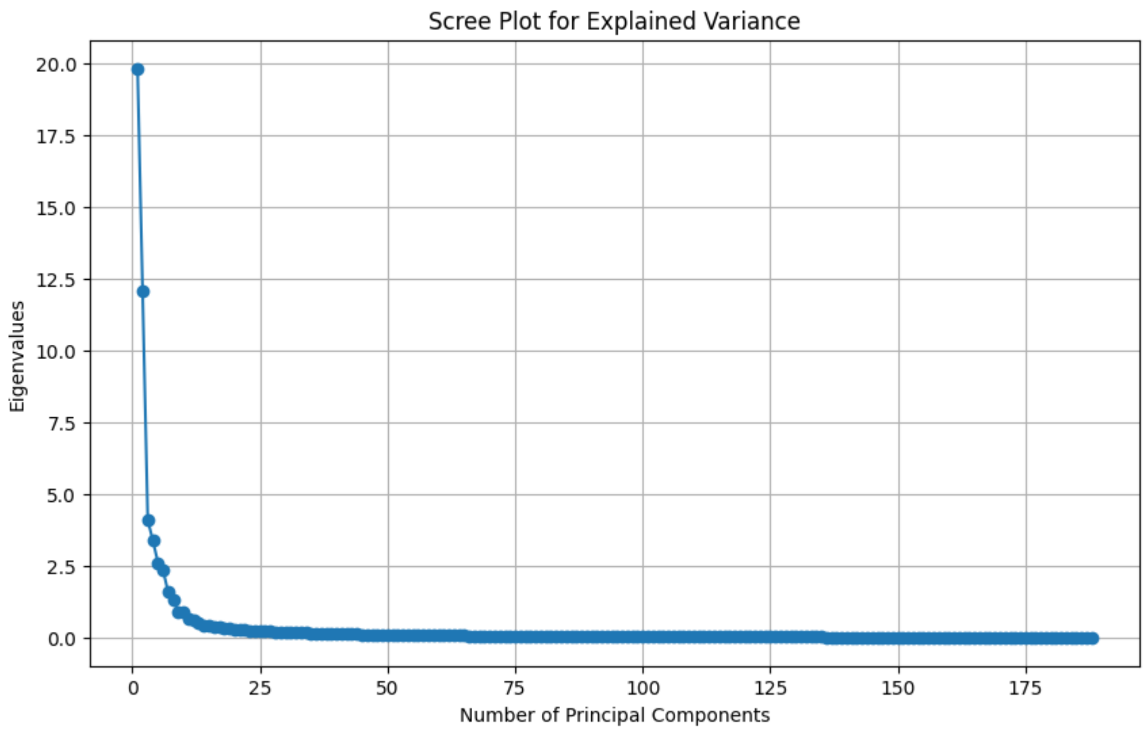


Figure 2: The Scree plot of the processed data

- Using PCA helps to capture the most important features while discarding less significant ones, reducing computational complexity, and potentially improving the performance of your machine learning models

3 - Training

3.1 Linear kernel

- The code iterates over each value of C in C_values and trains an SVM with a linear kernel using that specific C value.

- Inside the loop:

- An SVM with a linear kernel is instantiated and trained on the training data (train_X_df, train_y) using the current C value.
- Predictions are made on both the training set (train_X_df) and the validation set (val_X_df).
- The accuracy scores on the validation set are calculated using accuracy_score from scikit-learn.
- The best C and corresponding accuracy are updated if the current model achieves a higher accuracy on the validation set.

3.2 Gaussian/RBF Kernel

- The code iterates over each combination of C and gamma and trains an SVM with an RBF kernel using those specific hyperparameter values.

- Inside the loop:

- An SVM with an RBF kernel is instantiated and trained on the training data (train_X_df, train_y) using the current C and gamma values.
- Predictions are made on both the training set (train_X_df) and the validation set (val_X_df).
- The accuracy scores on the validation set are calculated using accuracy_score from scikit-learn.
- The best C, gamma, and corresponding accuracy are updated if the current model achieves a higher accuracy on the validation set.

- Try different values of C with gamma = 0.01 with the same code, the best hyperparameter combination found: C = 40, gamma = 0.01.

3.3 Train the SVM with the best hyperparameter combination

- Standardization and PCA Transformation for Training Data: Applies standardization to the training data (X_train) using a scaler (scaler). Then, it performs Principal Component Analysis (PCA) transformation to reduce dimensionality. The transformed data is stored in X_train_df.

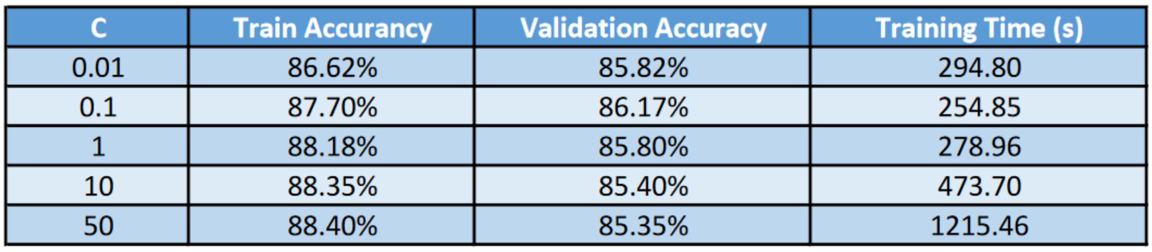
- Standardization and PCA Transformation for Test Data: Similarly, standardizes and applies PCA to the test data (X_test). The transformed data is stored in X_test_df.

- Fit Final SVM Model: Trains the svm_final model using the transformed training data (X_train_df and y_train), incorporating the best hyperparameters.

- Predictions for Training and Test Sets: Generates predictions for both the training set (y_train_pred) and the test set (y_test_pred) using the trained svm_final model. These predictions are valuable for assessing the model's performance on both seen and unseen data.

4 - Evaluation

4.1 Evaluate the outcomes attained through the implementation of the linear kernel



C	Train Accurancy	Validation Accuracy	Training Time (s)
0.01	86.62%	85.82%	294.80
0.1	87.70%	86.17%	254.85
1	88.18%	85.80%	278.96
10	88.35%	85.40%	473.70
50	88.40%	85.35%	1215.46

Figure 3: Result table of the linear kernel with different C parameters

The results from training linear SVM models with different values of C demonstrate several key observations. As C increases, both training and validation accuracies generally improve, indicating better model performance. However, a closer look reveals that, beyond a certain point

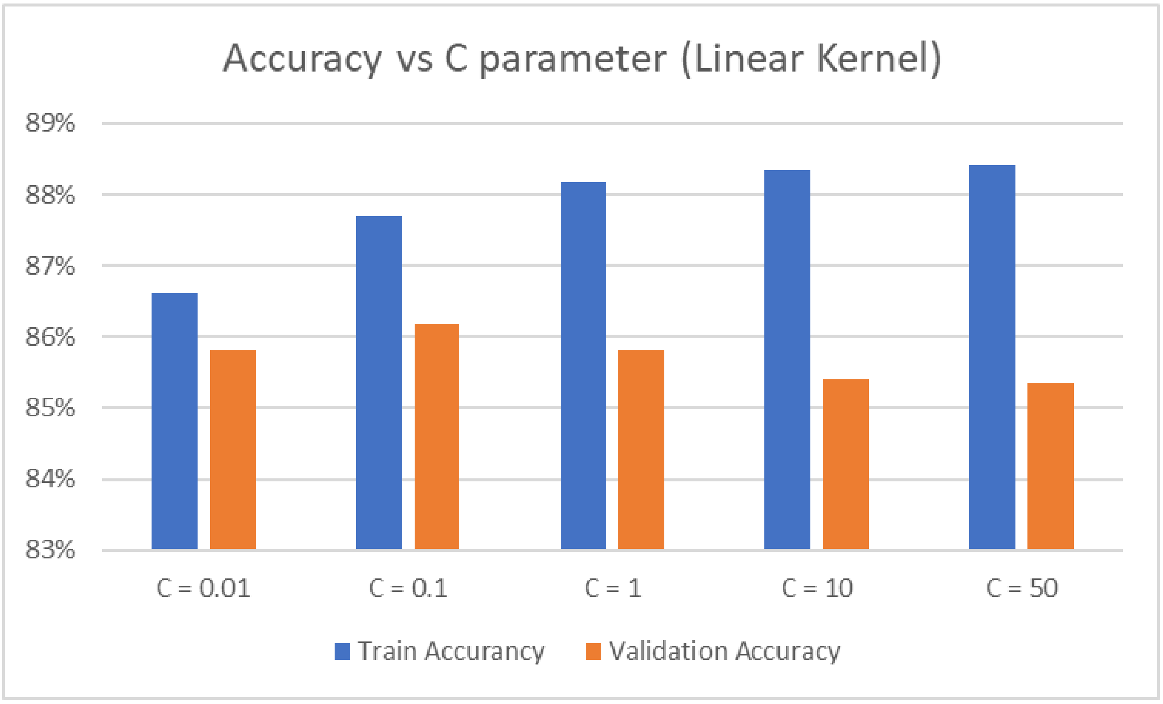


Figure 4: The bar chart illustrates train and validation accuracy of the linear kernel

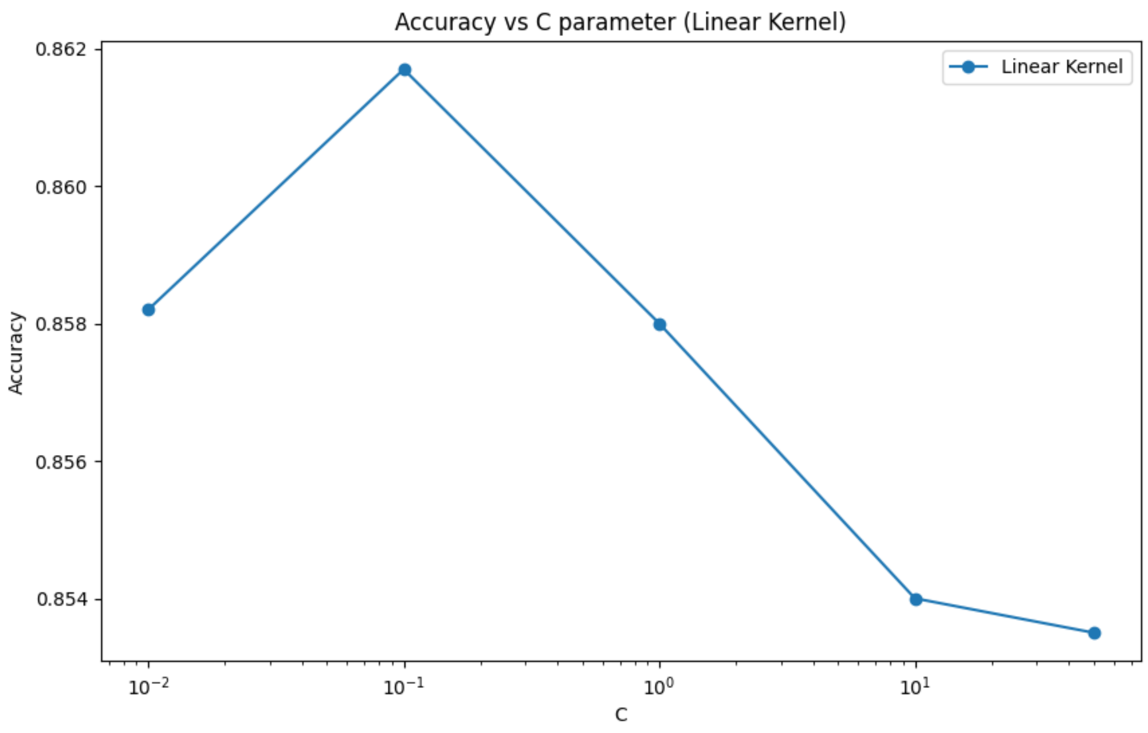


Figure 5: The line chart illustrates validation accuracy of the linear kernel

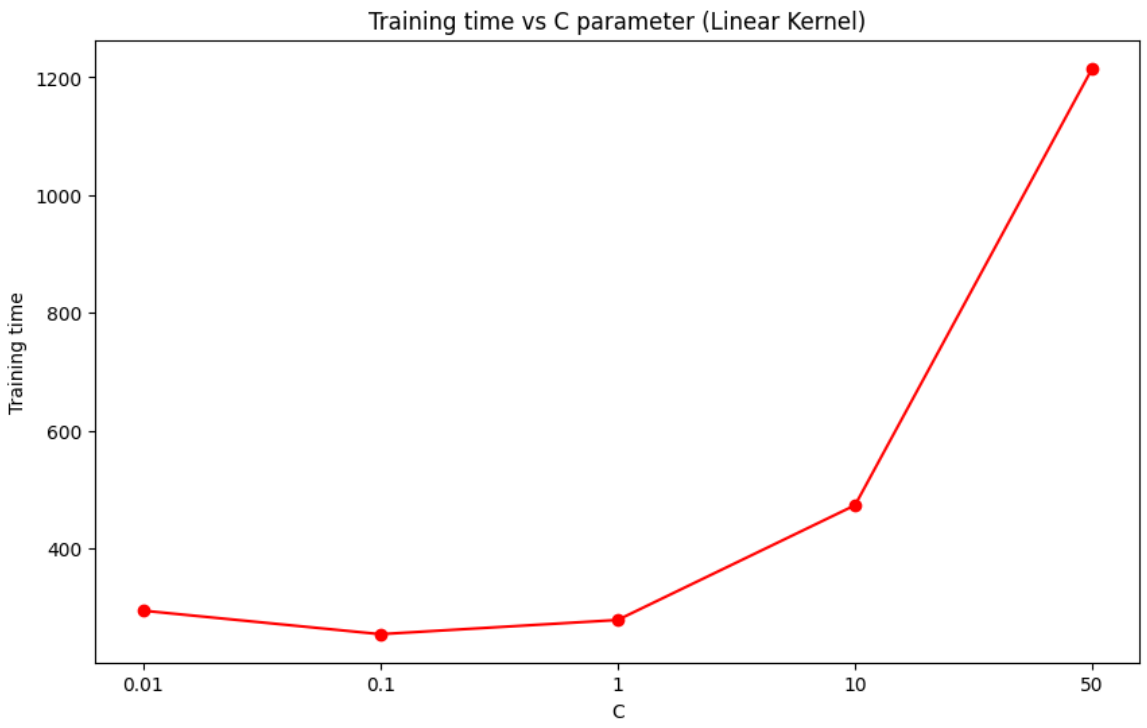


Figure 6: The line chart illustrates training time of the linear kernel

(C=10), the training accuracy continues to increase, while the validation accuracy plateaus or slightly decreases. This suggests a potential risk of overfitting.

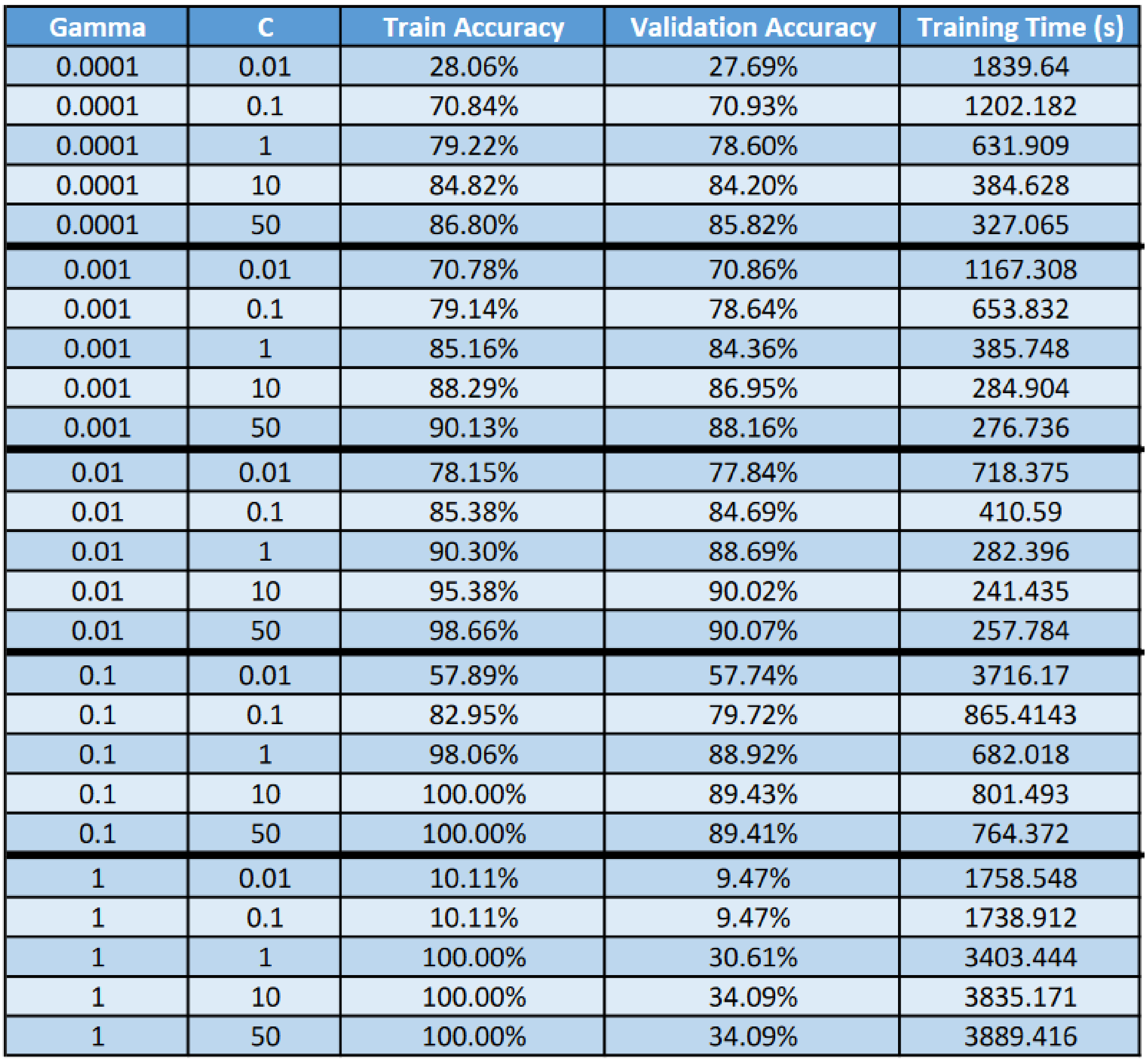
Additionally, the training times show an increasing trend with higher C values, with C=50 having the longest training time.

The plotted results visually highlight the trade-off between accuracy and computational cost for different C values. While a higher C might lead to better accuracy on the training set, it may not necessarily generalize well to unseen data. Therefore, choosing an appropriate value for C is crucial to achieving a well-balanced model with good accuracy and acceptable computational efficiency.

4.2 Evaluate the outcomes attained through the implementation of the rbf kernel

4.2.1 Different values of C and gamma

The RBF kernel SVM model was trained with varying combinations of C and gamma values to explore their impact on accuracy, training time, and potential overfitting. As observed, low values of C (0.01) and gamma (0.0001) result in low accuracy, indicating an underfit model. As both C and gamma increase, accuracy generally improves, peaking at C=10 and gamma=0.01. However, high C values lead to prolonged training times, especially noticeable with gamma=1.



Gamma	C	Train Accuracy	Validation Accuracy	Training Time (s)
0.0001	0.01	28.06%	27.69%	1839.64
0.0001	0.1	70.84%	70.93%	1202.182
0.0001	1	79.22%	78.60%	631.909
0.0001	10	84.82%	84.20%	384.628
0.0001	50	86.80%	85.82%	327.065
0.001	0.01	70.78%	70.86%	1167.308
0.001	0.1	79.14%	78.64%	653.832
0.001	1	85.16%	84.36%	385.748
0.001	10	88.29%	86.95%	284.904
0.001	50	90.13%	88.16%	276.736
0.01	0.01	78.15%	77.84%	718.375
0.01	0.1	85.38%	84.69%	410.59
0.01	1	90.30%	88.69%	282.396
0.01	10	95.38%	90.02%	241.435
0.01	50	98.66%	90.07%	257.784
0.1	0.01	57.89%	57.74%	3716.17
0.1	0.1	82.95%	79.72%	865.4143
0.1	1	98.06%	88.92%	682.018
0.1	10	100.00%	89.43%	801.493
0.1	50	100.00%	89.41%	764.372
1	0.01	10.11%	9.47%	1758.548
1	0.1	10.11%	9.47%	1738.912
1	1	100.00%	30.61%	3403.444
1	10	100.00%	34.09%	3835.171
1	50	100.00%	34.09%	3889.416

Figure 7: Result table of the rbf kernel with different C and gamma parameters

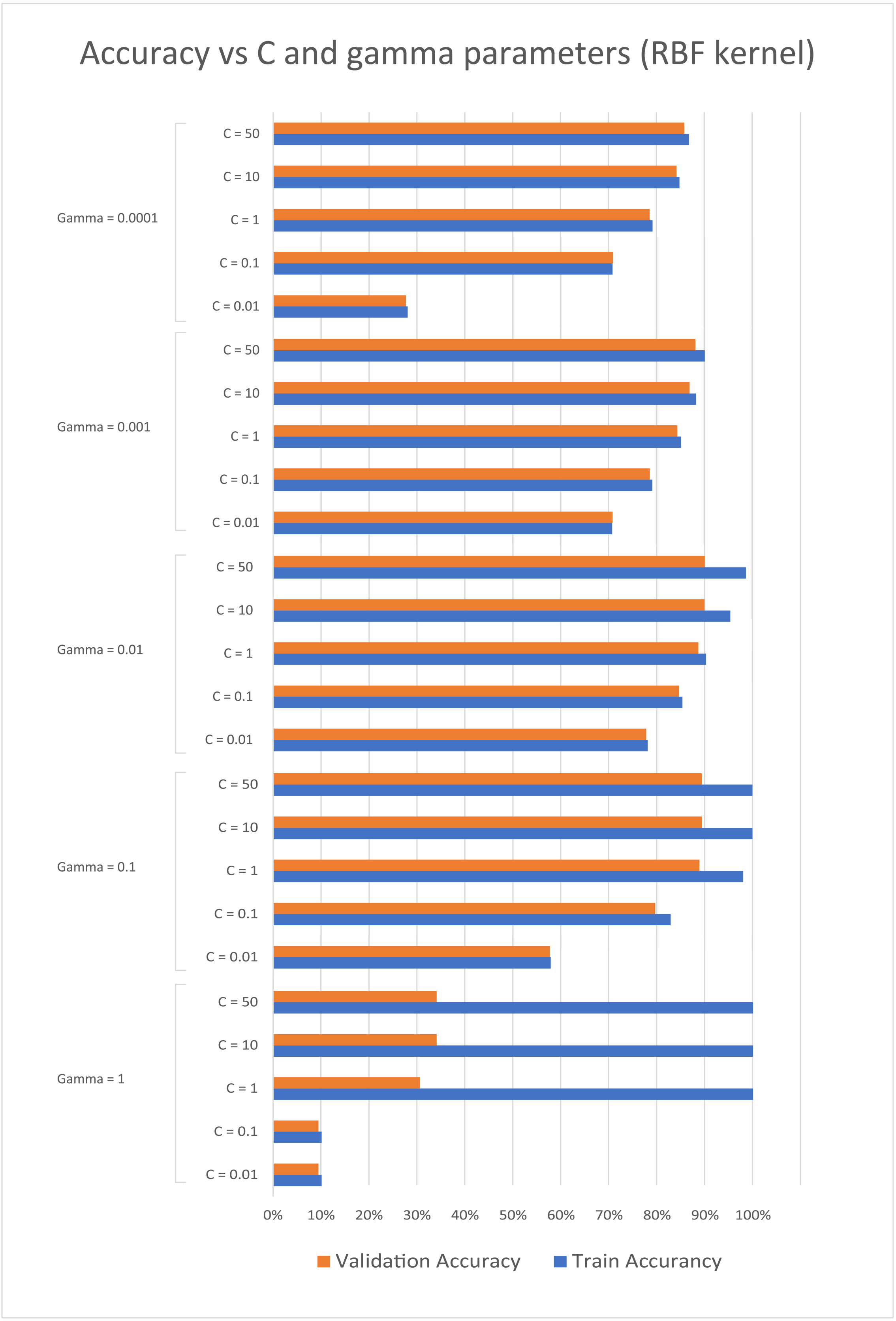


Figure 8: The bar chart illustrates train and validation accuracy of the rbf kernel

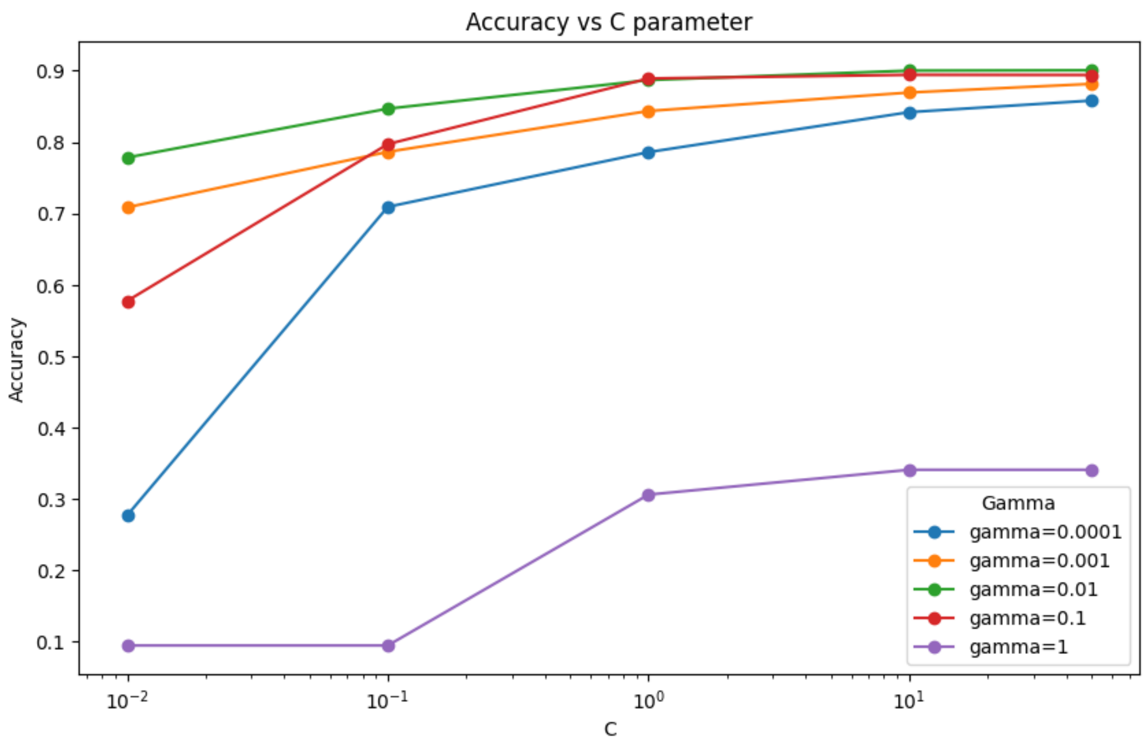


Figure 9: The line chart illustrates validation accuracy of the rbf kernel

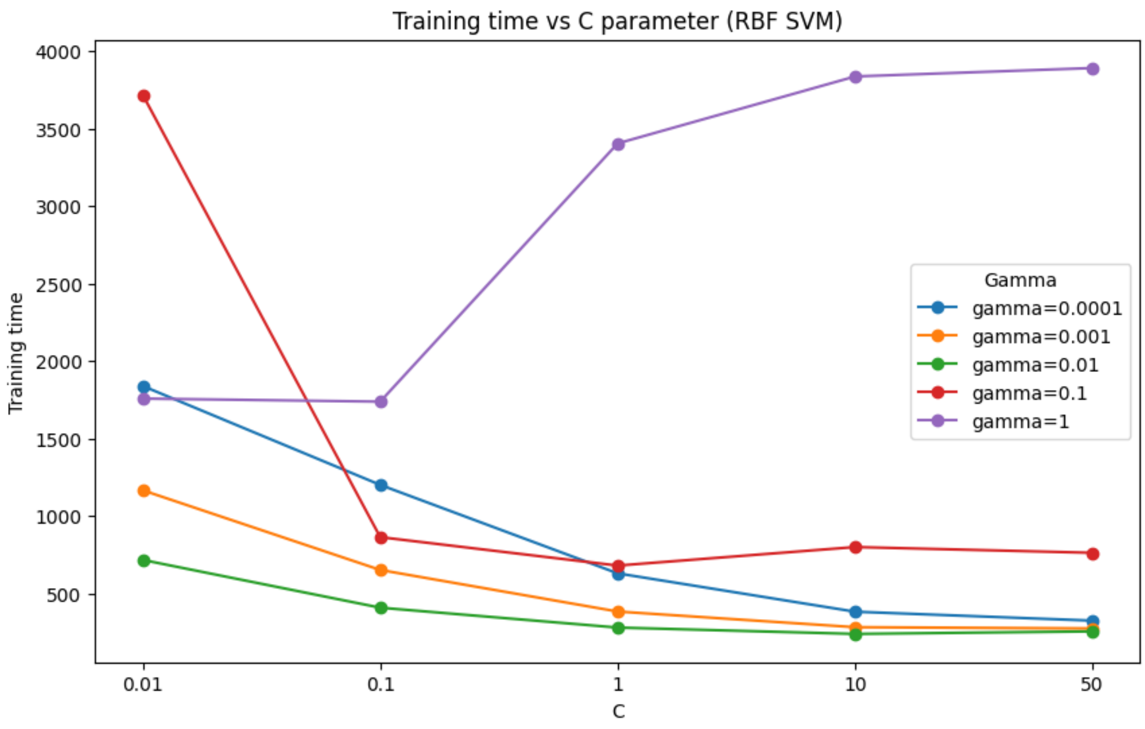
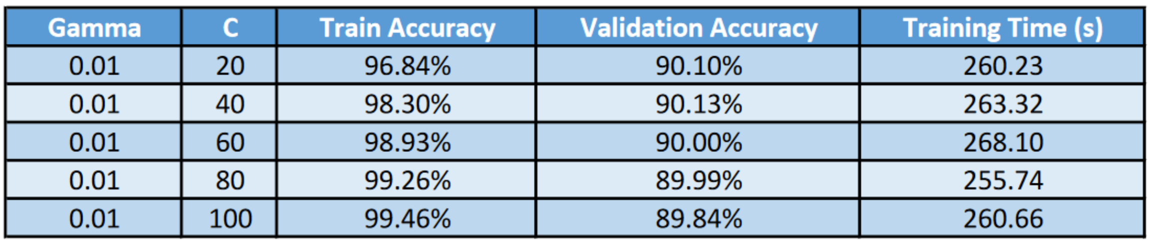


Figure 10: The line chart illustrates training time of the rbf kernel

Regarding overfitting, the model tends to overfit the training data as C and gamma increase, as evidenced by a significant gap between training and validation accuracy, particularly with C=10 and C=50. This emphasizes the importance of finding a balance between accuracy and generalization.

The training times also highlight the computational cost, with higher C and gamma values requiring substantially more time. This trade-off between accuracy, overfitting, and computational efficiency underscores the need for careful hyperparameter selection in SVM models.

4.2.2 Different values of C and selected gamma = 0.01



Gamma	C	Train Accuracy	Validation Accuracy	Training Time (s)
0.01	20	96.84%	90.10%	260.23
0.01	40	98.30%	90.13%	263.32
0.01	60	98.93%	90.00%	268.10
0.01	80	99.26%	89.99%	255.74
0.01	100	99.46%	89.84%	260.66

Figure 11: Result table of the rbf kernel with gamma = 0.01

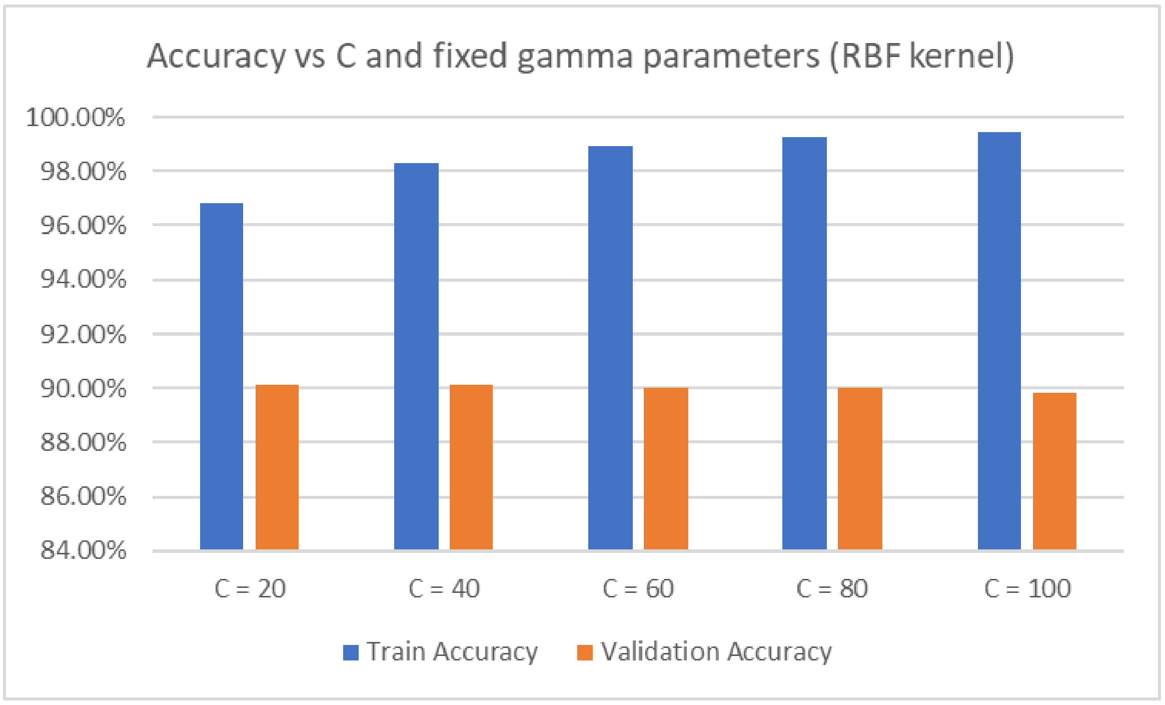


Figure 12: The bar chart illustrates accuracy of the rbf kernel with gamma = 0.01

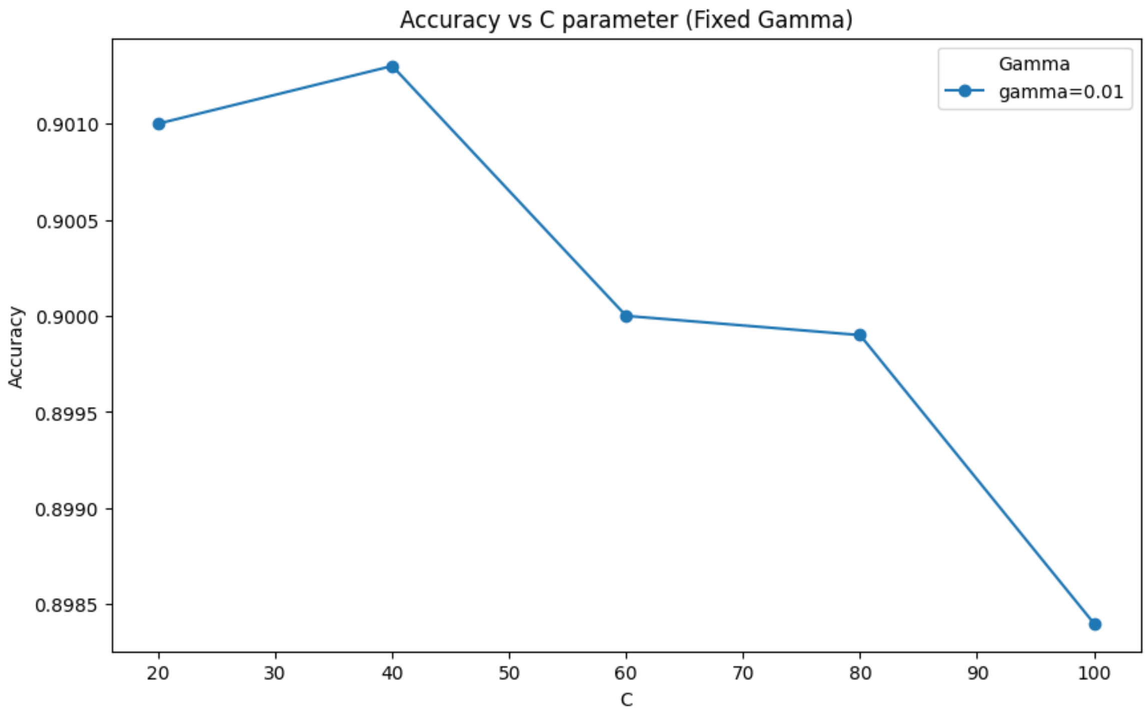


Figure 13: The line chart illustrates validation accuracy of the rbf kernel with gamma = 0.01

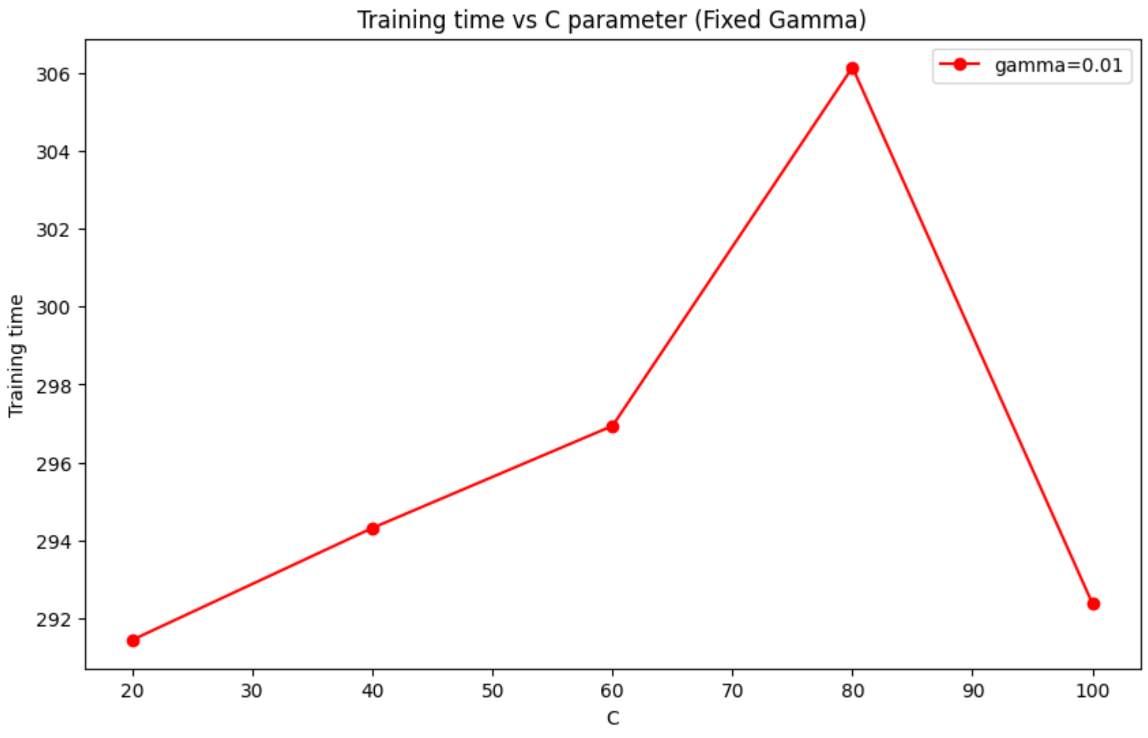


Figure 14: The line chart illustrates training time of the rbf kernel with gamma = 0.01

In this experiment, the RBF kernel SVM was trained with varying values of C while keeping gamma fixed at 0.01. The results show a consistent high validation accuracy across different C values, reaching a peak at C=40 with an accuracy of 90.13%. This suggests that the model achieves good generalization with moderate regularization (C=40).

Despite the increasing trend in accuracy, the training times remain relatively stable, indicating a reasonable computational cost. This suggests that, for the fixed gamma value, increasing C does not significantly impact training time.

The results reveal a noteworthy trend in the increasing training accuracy as C grows, reaching a peak at C=100 with a training accuracy of 99.46%. While the validation accuracy remains relatively stable, the widening gap between training and validation accuracies suggests a potential risk of overfitting, especially for higher values of C.

In summary, fixing gamma at 0.01 while adjusting C values allows for the identification of an optimal regularization strength (C=40) that balances accuracy and computational efficiency, demonstrating the importance of thoughtful hyperparameter tuning in SVM models.

4.3 Evaluate the results produced by the final prediction function

4.3.1 The accuracy scores achieved on the training and test sets

The hyperparameter tuning resulted in the selection of an RBF kernel with optimal parameters (C=40, gamma=0.01), achieving a peak validation accuracy of 90.13%.

Train Accuracy Evaluation: The model exhibits a high level of accuracy on the training set, reaching 98.13%. This indicates that the model captures the patterns within the training data effectively.

Test Accuracy Evaluation: The test accuracy score of 90.32% suggests that the model generalizes well to unseen data, providing reliable predictions on new instances.

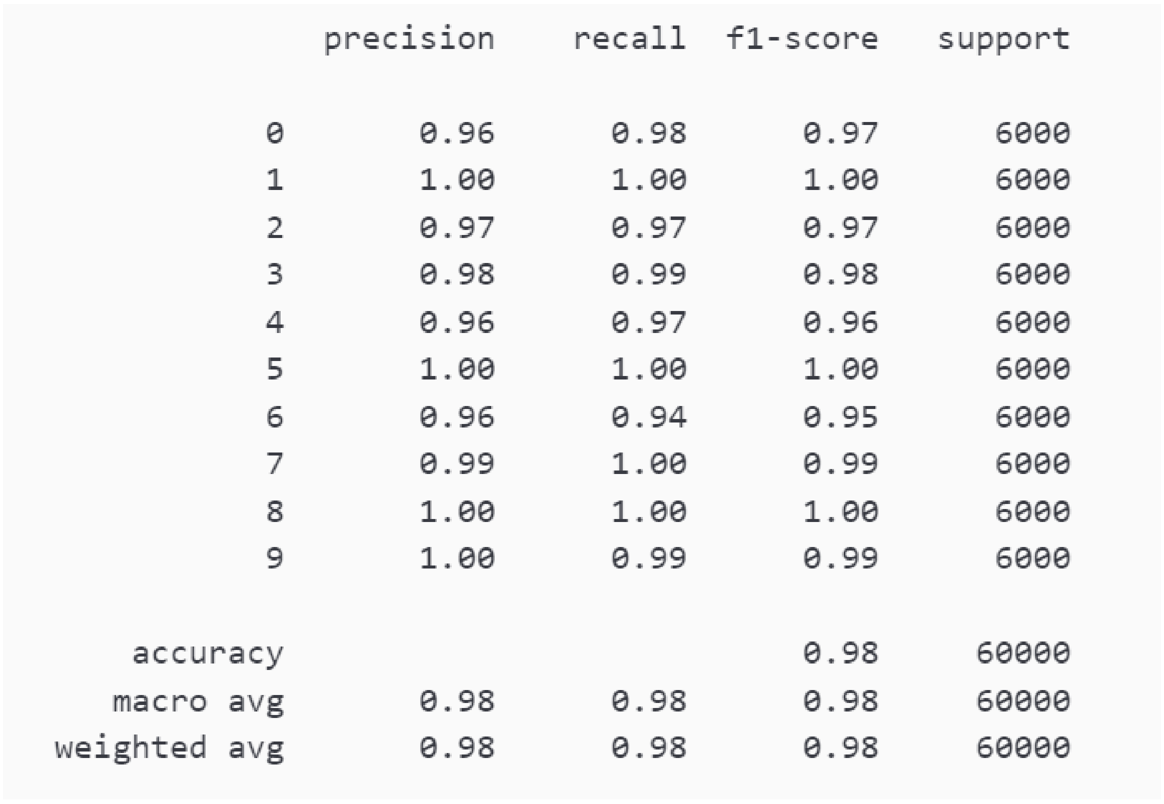
The relatively small gap between the training and test accuracy scores indicates good generalization, with no significant overfitting. The model's performance on the test set aligns closely with its performance on the training set, demonstrating robustness.

In summary, the final SVM model, configured with the identified optimal hyperparameters, achieves a strong balance between accuracy and generalization, making it a suitable candidate for deployment.

4.3.2 The classification report for the train data

- ***Precision:*** Precision values are consistently high across all classes, ranging from 0.96 to 1.00. This indicates a low false-positive rate, suggesting accurate positive predictions.
- ***Recall:*** Recall values are high for most classes, indicating the model's ability to capture positive instances. The recall ranges from 0.94 to 1.00.
- ***F1-Score:*** F1-scores, measuring the balance between precision and recall, are consistently high. This suggests an effective trade-off between precision and recall for each class.
- ***Accuracy:*** The overall accuracy of 98% on the training data indicates the model's ability to make correct predictions across all classes.

$\Rightarrow$ The model demonstrates excellent performance on the training data, achieving high precision, recall, and F1-score values. The balanced evaluation metrics and high accuracy suggest effective generalization without overfitting.



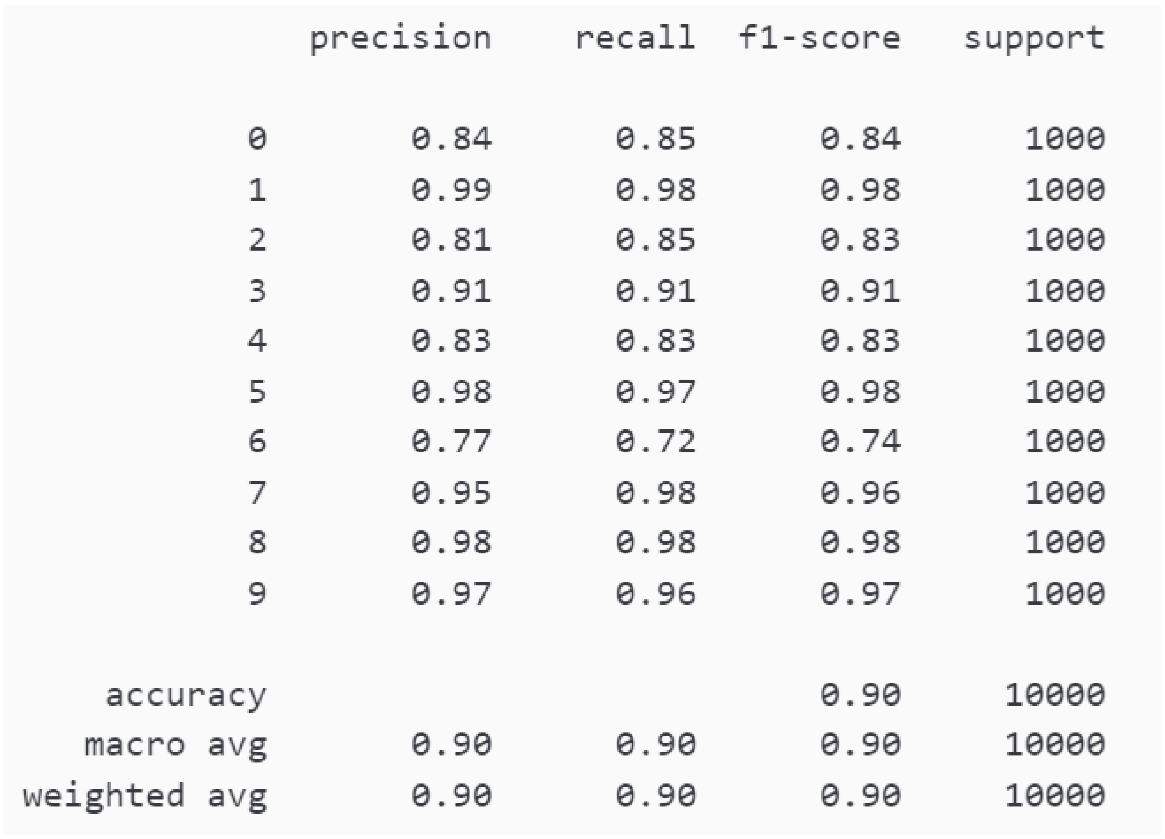
```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      6000
           1       1.00      1.00      1.00      6000
           2       0.97      0.97      0.97      6000
           3       0.98      0.99      0.98      6000
           4       0.96      0.97      0.96      6000
           5       1.00      1.00      1.00      6000
           6       0.96      0.94      0.95      6000
           7       0.99      1.00      0.99      6000
           8       1.00      1.00      1.00      6000
           9       1.00      0.99      0.99      6000

    accuracy                           0.98     60000
   macro avg       0.98      0.98      0.98     60000
weighted avg       0.98      0.98      0.98     60000
```

Figure 15: The classification report for the training data

4.3.3 The classification report for the test data

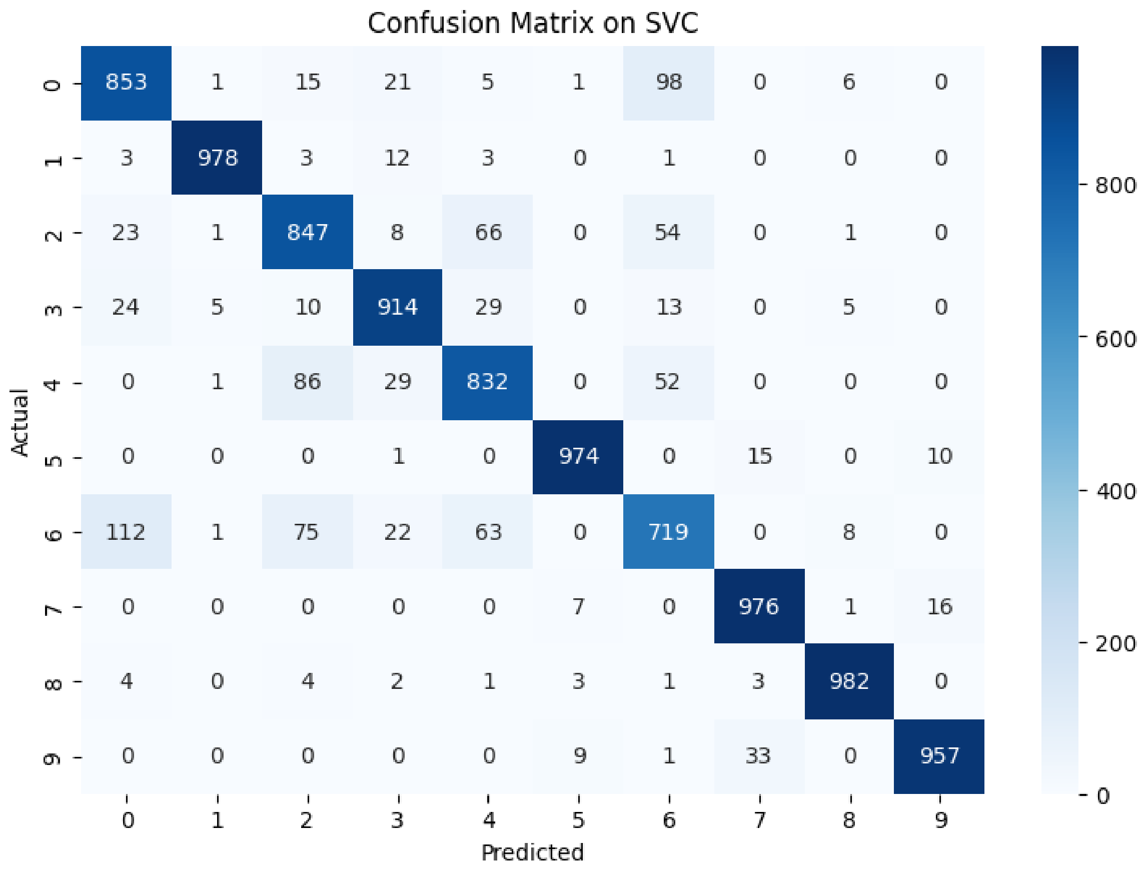
- ***Precision:*** Precision values are variable across classes, ranging from 0.77 to 0.99. This indicates varying levels of false-positive rates for different classes.
- ***Recall:*** Recall values show variability, ranging from 0.72 to 0.98. This suggests differing abilities to capture positive instances for each class.
- ***F1-Score:*** F1-scores vary, indicating different balances between precision and recall for each class.
- ***Accuracy:*** The overall accuracy of 90% on the test data suggests the model's ability to make correct predictions, but there are variations across different classes.

⇒ Conclusion: While the model performs well on the test data with high overall accuracy, the variability in precision, recall, and F1-score values across classes indicates that the model may have challenges with certain digits.



```
              precision    recall  f1-score   support

           0       0.84      0.85      0.84      1000
           1       0.99      0.98      0.98      1000
           2       0.81      0.85      0.83      1000
           3       0.91      0.91      0.91      1000
           4       0.83      0.83      0.83      1000
           5       0.98      0.97      0.98      1000
           6       0.77      0.72      0.74      1000
           7       0.95      0.98      0.96      1000
           8       0.98      0.98      0.98      1000
           9       0.97      0.96      0.97      1000

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000
```

Figure 16: The classification report for the test data



Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	853	1	15	21	5	1	98	0	6	0
1	3	978	3	12	3	0	1	0	0	0
2	23	1	847	8	66	0	54	0	1	0
3	24	5	10	914	29	0	13	0	5	0
4	0	1	86	29	832	0	52	0	0	0
5	0	0	0	1	0	974	0	15	0	10
6	112	1	75	22	63	0	719	0	8	0
7	0	0	0	0	0	7	0	976	1	16
8	4	0	4	2	1	3	1	3	982	0
9	0	0	0	0	0	9	1	33	0	957

Figure 17: The confusion matrix of the test data predictions

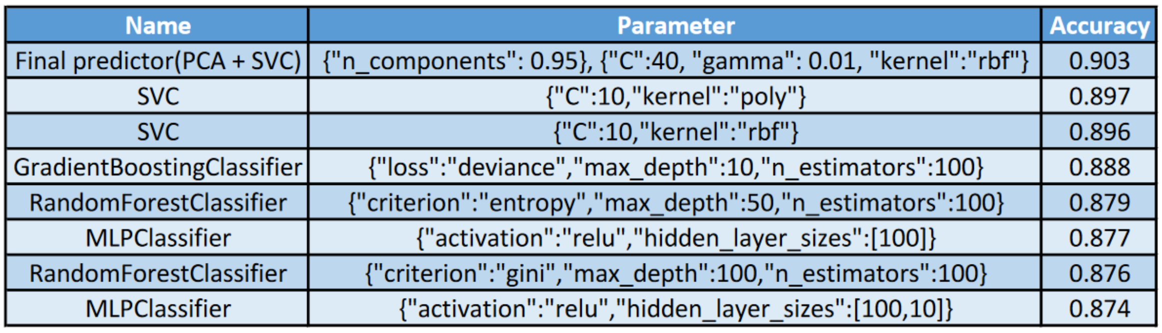
4.3.4 The confusion matrix of the predictions on the test data

- The heatmap visualization highlights areas of focus, where misclassifications are more prevalent, such as digit 6.

- The model demonstrates strong overall performance, with a balanced distribution of correct predictions across most classes.

⇒ Conclusion: While the model exhibits excellent performance for most digits, further analysis and refinement may be needed, particularly for digits with a higher incidence of misclassifications, such as digit 6.

4.4 Comparison with some results recorded by other methods



Name	Parameter	Accuracy
Final predictor(PCA + SVC)	{"n_components": 0.95}, {"C":40, "gamma": 0.01, "kernel":"rbf"}	0.903
SVC	{"C":10,"kernel":"poly"}	0.897
SVC	{"C":10,"kernel":"rbf"}	0.896
GradientBoostingClassifier	{"loss":"deviance","max_depth":10,"n_estimators":100}	0.888
RandomForestClassifier	{"criterion":"entropy","max_depth":50,"n_estimators":100}	0.879
MLPClassifier	{"activation":"relu","hidden_layer_sizes":[100]}	0.877
RandomForestClassifier	{"criterion":"gini","max_depth":100,"n_estimators":100}	0.876
MLPClassifier	{"activation":"relu","hidden_layer_sizes":[100,10]}	0.874

Figure 18: The comparison with other results recorded

- ***PCA's Contribution:*** The superior performance of our final predictor can be attributed to the use of PCA. PCA reduces the dimensionality of the feature space while retaining essential information. This not only improves the computational efficiency but also mitigates the risk of overfitting by focusing on the most informative components.

- ***SVC's Effectiveness:*** The combination of PCA with SVC further enhances model effectiveness. The chosen parameters for SVC, especially using a polynomial kernel with a moderate regularization parameter (C=10), contribute to capturing complex relationships in the data.

- ***Optimal Trade-off:*** By selecting 95% of the variance in PCA, we strike a balance between preserving critical information and reducing noise. This optimal trade-off results in a more robust and interpretable model, leading to the higher accuracy observed in our final predictor compared to other models.

⇒ In summary, the power of using PCA lies in its ability to transform and condense the data, improving the efficiency and performance of subsequent classifiers. The combination of PCA with SVC, with carefully chosen parameters, results in a model that excels in accuracy compared to other classifiers.

5 - References

[1] Title: "The A-Z guide to Support Vector Machine". Website: www.analyticsvidhya.com. Updated day: (November 16th, 2023). Accessed day: (January 6, 2024). http://surl.li/oxzxo

[2] Title: "Understanding SVM Hyperparameters". Website: stackabuse.com. Updated day: (July 2nd, 2023). Accessed day: (January 6, 2024). https://stackabuse.com/understanding-svm-hyperparameters/

[3] Title: "Fashion-mnist-SVM-90.6% accuracy". Website: www.kaggle.com. Updated day: (3 years ago). Accessed day: (January 1, 2024). https://www.kaggle.com/code/anushkabhadra/fashion-mnist-svm-90-6-accuracy/notebook

[4] Title: "RBF SVM Parameters in Scikit Learn". Website: www.geeksforgeeks.org. Updated day: (Jul 31, 2023). Accessed day: (January 2, 2024). https://www.geeksforgeeks.org/rbf-svm-parameters-in-scikit-learn/

[5] Title: "Applying Support Vector Machines and Logistic Regression on the Fashion MNIST dataset". Website: federicoarenasl.github.io. Updated day: (Mar 25, 2021). Accessed day: (January 2, 2024). https://federicoarenasl.github.io/SVM-LR-on-Fashion-MNIST/