# UMC 203 Assignment 2

K. Sai Sandesh Reddy
SR Number: 23627

28 th March 2025

## Question 1: Support Vector Machine and Perceptron

### Deliverable 1: Misclassification Rate vs. Iterations for Initial Perceptron

A plot of the misclassification rate versus the number of iterations for the initial perceptron algorithm is shown in Figure 1. The perceptron did not converge, indicating non-separability in the original dataset.

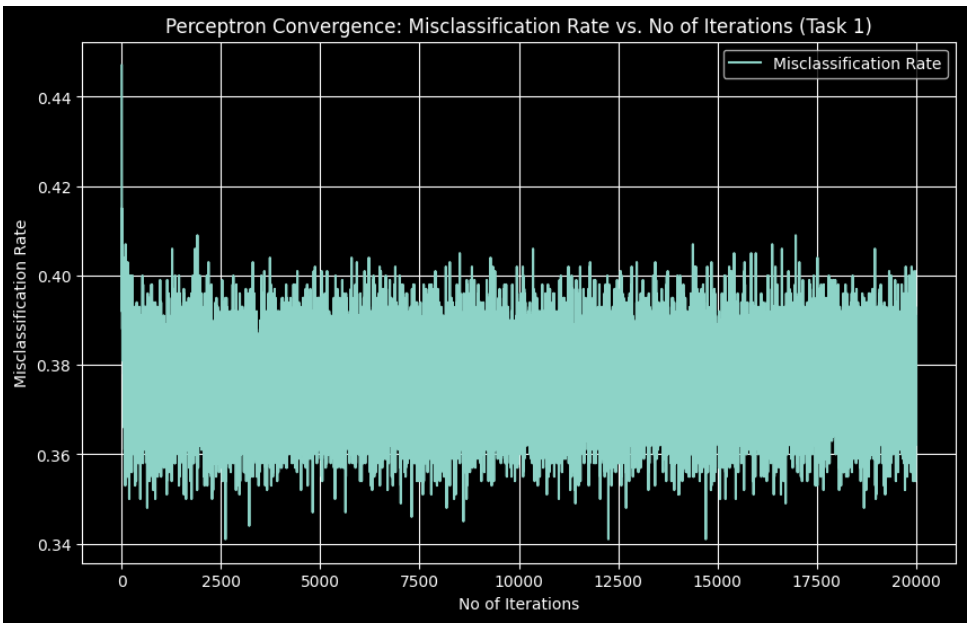


Figure 1: Misclassification Rate vs. Iterations for Initial Perceptron (Task 1)

### Deliverable 2: Primal vs. Dual SVM Solution Times

1. **Solution Times and Justification**
   The primal SVM took 1.0282 seconds to solve, while the dual SVM took 0.9675 seconds, making the dual slightly faster by approximately 0.06 seconds.
   **Justification**: The dual SVM outperforms the primal because it optimizes fewer variables in the quadratic programming problem. The primal formulation optimizes

$d + 1 + n$ variables, where $d$ is the number of features ( 27 in this case), 1 accounts for the bias $b$, and $n$ is the number of slack variables $\xi_i$ (one per sample). In contrast, the dual formulation optimizes only $n$ variables—the Lagrange multipliers $\alpha_i$ (1000). For a dataset with $n = 1000$ and $d = 27$, the primal has 1028 variables $(27 + 1 + 1000)$, while the dual has 1000. This reduction in variables (28 fewer) leads to a slightly lower computation requirement in the dual, resulting in faster solution times.

2. **Consistency of Solutions**
   The weight vectors and biases obtained from both the primal and dual optimization problems coincide, with a difference in the L2 norm of the weight vectors of 0.000 and a bias difference of 0.000. This identical outcome indicates that both optimization approaches solve the same underlying SVM problem, differing only in their mathematical formulations (primal variables $w$, $b$, $\xi$ vs. dual variables $\alpha$), yet converging to equivalent solutions within numerical precision.

## Deliverable 3: Images Causing Non-Separability

The points causing inseparability in the linear SVM coincide for both primal and dual formulations, identified through distinct yet equivalent conditions. In the primal case, a point $\mathbf{x}_i$ is deemed non-separable if its slack variable satisfies $\xi_i > 1$, which implies $y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0$, indicating misclassification. In the dual case, a point is non-separable if it is misclassified, i.e., $y_i(\mathbf{w}^T\mathbf{x}_i + b) < 0$, typically among those with $\alpha_i = C$, meaning it lies on the wrong side of the decision boundary and incurs the maximum penalty. In both formulations, this corresponds to the condition where the predicted label times the true label is negative (prediction $\cdot\, y_i < 0$), confirming that the same set of points violates linear separability.

The indices of these non-separable datapoints, as identified by the linear SVM, are saved in `inseparable_23627.csv`. There were 281 such points, corresponding to images in the CIFAR-100 dataset given to me.

## Deliverable 4: Final Misclassification Rate for Kernelized SVM

Task 3 required implementing a kernelized SVM with a Gaussian kernel, replacing the linear kernel from previous tasks, to address non-separable data. The question, "The final misclassification rate for the kernelized SVM for Task 3," asks for the percentage of training points misclassified after training, with hyperparameters chosen to eliminate non-separability and ensure the decision boundary perfectly aligns with the training labels (i.e., zero training error). The Gaussian kernel enables this by mapping the data into a higher-dimensional space where a linear boundary can separate all points.

The final misclassification rate for the kernelized SVM with the Gaussian kernel was 0%, achieved with hyperparameters $C = 1$ and $\gamma = 999$. This indicates complete separability of the training data, as the high $\gamma$ value allows the Gaussian kernel to tightly fit the decision boundary to the training points, ensuring perfect classification.

### Deliverable 5: Misclassification Rate vs. Iterations for Retrained Perceptron

A plot of the misclassification rate versus iterations for the perceptron trained by removing the points which caused the inseperability (Task 4) is shown in Figure 2. After removing non-separable points(i.e. 281 points obtained from Deliverable 3), the dataset became linearly seperable, and thus perceptron converged in 296 iterations.

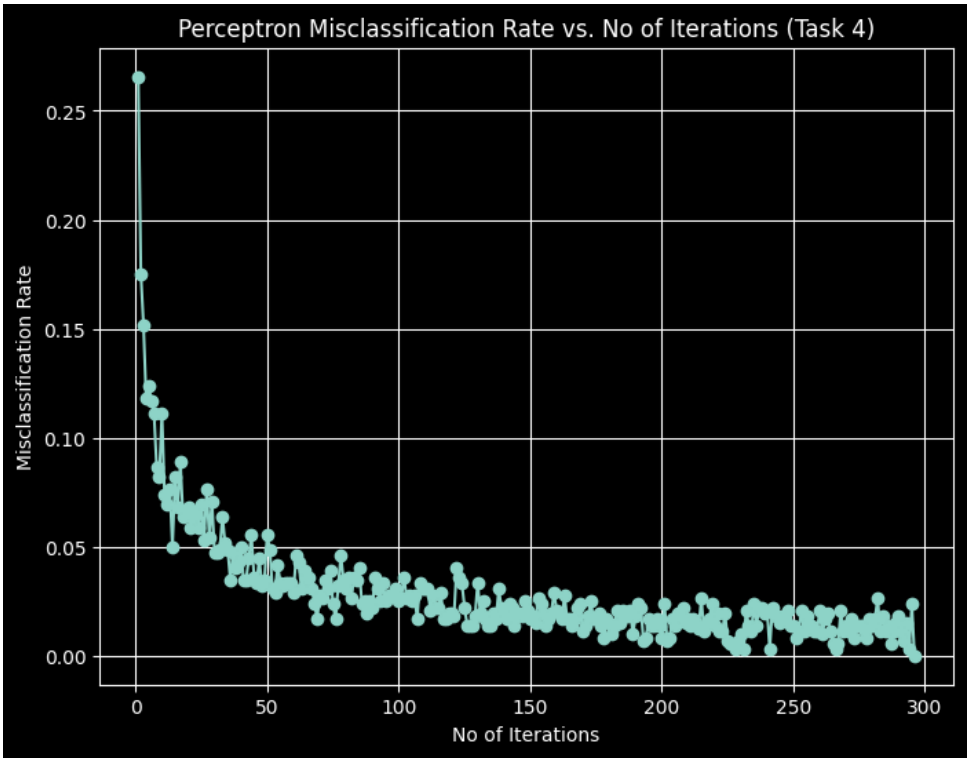


Figure 2: Misclassification Rate vs. Iterations for Retrained Perceptron (Task 4)

## Question 2: Logistic Regression, MLP, CNN & PCA

### Deliverable 1: Reconstructed Image Using PCA

The reconstruction of an image using principal components is shown in Figure 3. Observations:

- Using just 1 component, the image is very blurry and unrecognizable, indicating that a single component captures insufficient variance.
- At $k = 50$, the digit's outline (a "0") emerges but remains noisy and unclear.
- By $k = 200$, the digit becomes recognizable, though some noise persists.
- At $k = 400$, the reconstruction is clear, with most details of the "0" preserved.
- At $k = 600$, the image is nearly identical to the original, and at $k = 784$, it matches exactly.

Conclusion: Approximately 400 components are sufficient for a good reconstruction of the MNIST digit, balancing quality and dimensionality reduction.

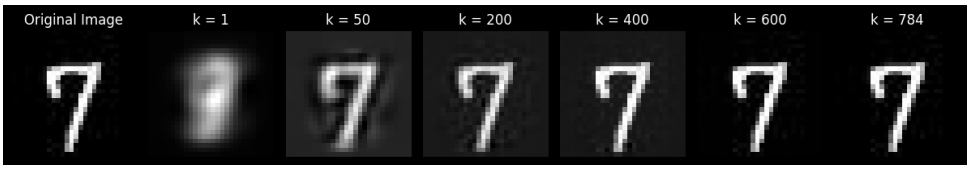


Figure 3: Reconstruction of an MNIST Image Using PCA Components

## Deliverable 2: Confusion Matrix and Metrics Comparison

Confusion matrices for MLP, CNN, MLP with PCA, and Logistic Regression with PCA are shown in Figures 4 to 7. Performance metrics (accuracy, precision, recall, F1-score) are summarized below.

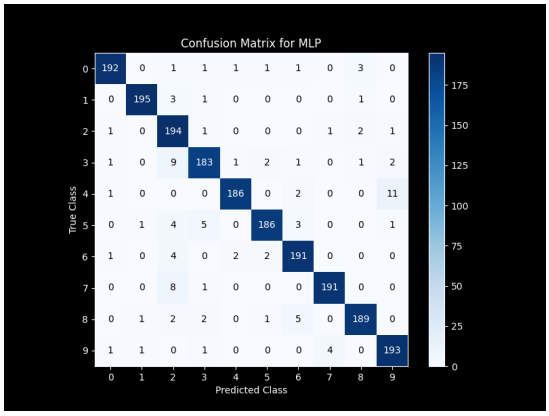


Figure 4: Confusion Matrix for MLP

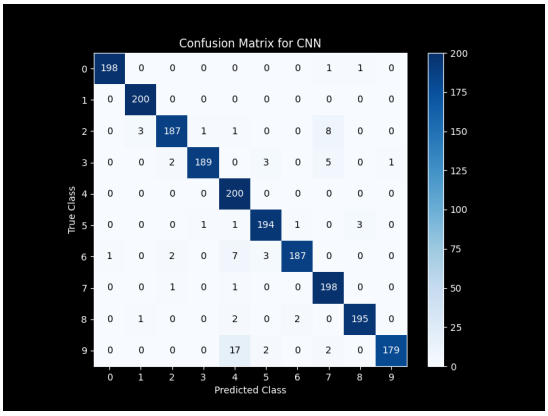


Figure 5: Confusion Matrix for CNN

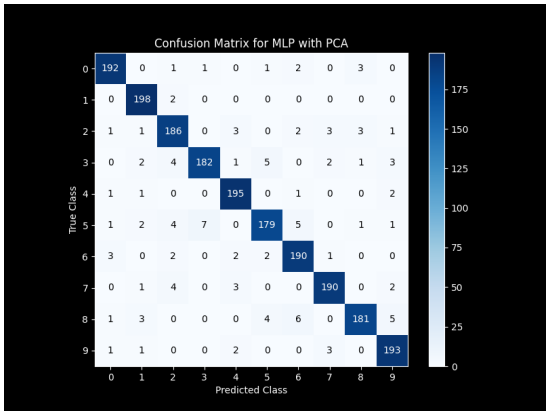


Figure 6: Confusion Matrix for MLP with PCA

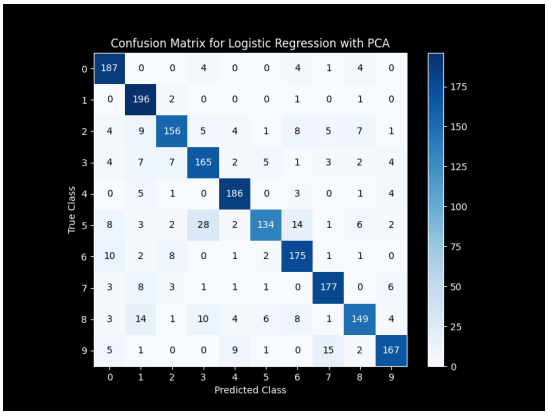


Figure 7: Confusion Matrix for Logistic Regression with PCA

**Overall Accuracy:**

Table 1: Overall Accuracy Across Models

| **Model** | **Accuracy** |
|---|---|
| MLP | 95.00% |
| CNN | 96.35% |
| MLP with PCA | 94.30% |
| Logistic Regression with PCA | 84.60% |

**Class-Wise Metrics:**

Table 2: Metrics for Class 0

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.975 | 0.960 | 0.967 | 0.960 |
| CNN | 0.995 | 0.985 | 0.990 | 0.995 |
| MLP with PCA | 0.960 | 0.960 | 0.960 | 0.960 |
| Logistic Regression with PCA | 0.835 | 0.935 | 0.882 | 0.935 |

Table 3: Metrics for Class 1

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.985 | 0.975 | 0.980 | 0.975 |
| CNN | 0.971 | 1.000 | 0.985 | 1.000 |
| MLP with PCA | 0.947 | 0.990 | 0.968 | 0.990 |
| Logistic Regression with PCA | 0.800 | 0.980 | 0.881 | 0.980 |

Table 4: Metrics for Class 2

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.862 | 0.970 | 0.913 | 0.970 |
| CNN | 0.989 | 0.890 | 0.937 | 0.940 |
| MLP with PCA | 0.916 | 0.930 | 0.923 | 0.930 |
| Logistic Regression with PCA | 0.867 | 0.780 | 0.821 | 0.780 |

Table 5: Metrics for Class 3

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.938 | 0.915 | 0.927 | 0.915 |
| CNN | 0.975 | 0.980 | 0.978 | 0.965 |
| MLP with PCA | 0.958 | 0.910 | 0.933 | 0.910 |
| Logistic Regression with PCA | 0.775 | 0.825 | 0.799 | 0.825 |

Table 6: Metrics for Class 4

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.979 | 0.930 | 0.954 | 0.930 |
| CNN | 0.966 | 0.980 | 0.973 | 1.000 |
| MLP with PCA | 0.947 | 0.975 | 0.961 | 0.975 |
| Logistic Regression with PCA | 0.890 | 0.930 | 0.910 | 0.930 |

Table 7: Metrics for Class 5

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.969 | 0.930 | 0.949 | 0.930 |
| CNN | 0.985 | 0.970 | 0.977 | 0.965 |
| MLP with PCA | 0.937 | 0.895 | 0.916 | 0.895 |
| Logistic Regression with PCA | 0.893 | 0.670 | 0.766 | 0.670 |

Table 8: Metrics for Class 6

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.941 | 0.955 | 0.948 | 0.955 |
| CNN | 0.965 | 0.960 | 0.962 | 0.945 |
| MLP with PCA | 0.922 | 0.950 | 0.936 | 0.950 |
| Logistic Regression with PCA | 0.818 | 0.875 | 0.845 | 0.875 |

Table 9: Metrics for Class 7

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.974 | 0.955 | 0.965 | 0.955 |
| CNN | 0.930 | 1.000 | 0.964 | 0.990 |
| MLP with PCA | 0.955 | 0.950 | 0.952 | 0.950 |
| Logistic Regression with PCA | 0.868 | 0.885 | 0.876 | 0.885 |

Table 10: Metrics for Class 8

| **Model** | **Precision** | **Recall** | **F1 Score** | **Accuracy** |
|---|---|---|---|---|
| MLP | 0.964 | 0.945 | 0.955 | 0.945 |
| CNN | 0.989 | 0.940 | 0.964 | 0.915 |
| MLP with PCA | 0.958 | 0.905 | 0.931 | 0.905 |
| Logistic Regression with PCA | 0.861 | 0.745 | 0.799 | 0.745 |

Table 11: Metrics for Class 9

| Model | Precision | Recall | F1 Score | Accuracy |
|---|---|---|---|---|
| MLP | 0.928 | 0.965 | 0.946 | 0.965 |
| CNN | 0.943 | 0.995 | 0.968 | 0.915 |
| MLP with PCA | 0.932 | 0.965 | 0.948 | 0.965 |
| Logistic Regression with PCA | 0.888 | 0.835 | 0.861 | 0.835 |

**Comparison and Analysis:**

- *Overall Accuracy:* As shown in Table 1, CNN achieves the highest accuracy at 97.00%, followed by MLP at 95.00%, MLP with PCA at 94.30%, and Logistic Regression with PCA at 84.60%. We can observe that when using the model MLP with PCA, we are removing some features and focusing only on the important features, which slightly decreases the accuracy compared to just using MLP. Logistic Regression with PCA has the lowest accuracy, which makes sense because, being a linear model, it has far fewer parameters to learn compared to other neural network models, which have many more parameters, leading to better performance.

- *Class-Wise Observations:* The class-wise metrics (Tables 2 to 11) reveal that CNN consistently achieves the highest precision, recall, F1-score, and accuracy across most classes. For example, in class 0, CNN attains an F1-score of 0.990 and accuracy of 0.995, while in class 3, it reaches an F1-score of 0.978 and accuracy of 0.965. MLP with PCA performs similarly to MLP but exhibits lower metrics in classes such as 5 (F1=0.916, accuracy=0.895). Logistic Regression with PCA shows the weakest performance, particularly in classes 5 (F1=0.766, accuracy=0.670) and 8 (F1=0.799, accuracy=0.745), due to its linear nature.

- *Key Differences:* CNN excels due to its ability to extract spatial features, making it best suited for image data. This is reflected in its high accuracy and F1-scores across all classes. MLP with PCA and Logistic Regression with PCA struggle with certain classes (e.g., 5, 8) due to information loss from PCA, as seen in their lower accuracy and F1-scores. Common misclassifications, such as class 5 being predicted as class 3, are evident in the confusion matrices, especially for simpler models.

- *Insights:* CNN's architecture is ideal for MNIST image data, consistently achieving high accuracy and F1-scores. PCA-based models, while reducing dimensionality, lose some discriminative power, resulting in lower metrics, especially in challenging classes. Logistic Regression with PCA, being a linear model, underperforms compared to neural networks, as evidenced by its lower accuracy and F1-scores.

## Deliverable 3: Average AUC Score for Binary Classifiers

The average AUC score across the 10 binary logistic regression classifiers (one-vs-rest) using PCA features is 0.969. The ROC curves for each classifier are shown in Figure 8, and detailed AUC scores along with comparisons to class-wise accuracies are provided below.

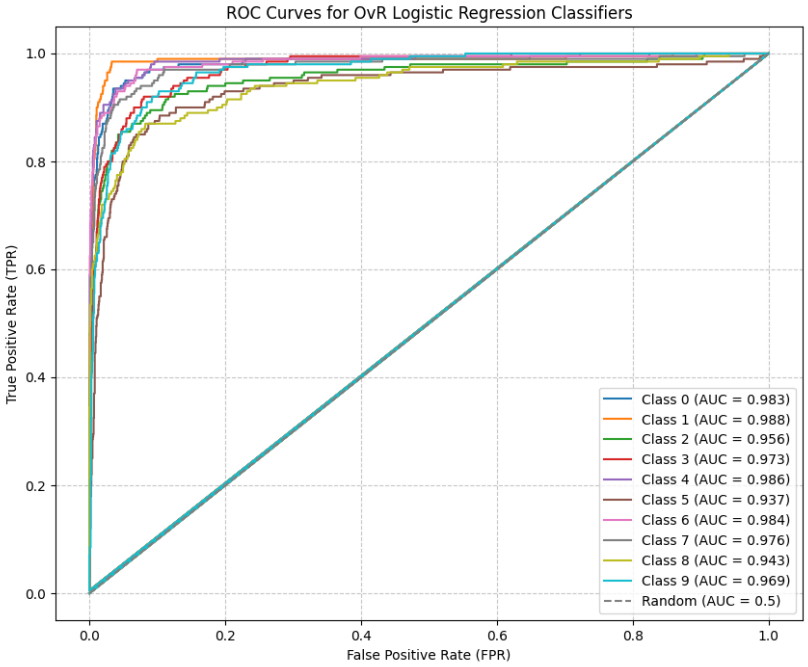


Figure 8: ROC Curves for OvR Logistic Regression Classifiers

**AUC Scores:**

Table 12: AUC Scores for Binary Classifiers

| **Class** | **AUC** |
|---|---|
| Class 0 | 0.983 |
| Class 1 | 0.988 |
| Class 2 | 0.956 |
| Class 3 | 0.973 |
| Class 4 | 0.986 |
| Class 5 | 0.937 |
| Class 6 | 0.984 |
| Class 7 | 0.976 |
| Class 8 | 0.943 |
| Class 9 | 0.969 |
| **Average** | **0.969** |

**Comparison with Class-Wise Accuracies:**

Table 13: Comparison of Class-Wise Accuracies and AUC Scores

| **Class** | **Accuracy** | **AUC** |
|---|---|---|
| Class 0 | 94.50% | 0.983 |
| Class 1 | 97.50% | 0.988 |
| Class 2 | 78.50% | 0.956 |
| Class 3 | 83.50% | 0.973 |
| Class 4 | 90.00% | 0.986 |
| Class 5 | 63.00% | 0.937 |
| Class 6 | 91.50% | 0.984 |
| Class 7 | 90.00% | 0.976 |
| Class 8 | 75.00% | 0.943 |
| Class 9 | 79.00% | 0.969 |

**Analysis:**

- *ROC Curves:* Figure 8 shows the ROC curves for all 10 OvR classifiers. All curves are well above the random guessing line (AUC = 0.5), indicating strong discriminative ability across classes.
- *AUC Scores:* As shown in Table 12, the highest AUC is for Class 1 (0.988), and the lowest is for Class 5 (0.937). This aligns with the class-wise accuracies from Deliverable 2, where Class 1 has the highest accuracy (97.50%) and Class 5 the lowest (63.00%).
- *Average AUC:* The average AUC of 0.969 is notably high, suggesting that the OvR logistic regression classifiers perform well overall, even with PCA-reduced features. This value exceeds the typical range of 0.85–0.90 for logistic regression on MNIST, reflecting effective feature retention by PCA.

- *Correlation with Accuracy:* Table 13 shows a strong correlation between AUC scores and class-wise accuracies. Classes with high accuracies (e.g., Class 1: 97.50%, Class 0: 94.50%) have high AUCs (0.988, 0.983), while classes with lower accuracies (e.g., Class 5: 63.00%, Class 8: 75.00%) have lower AUCs (0.937, 0.943).

- *Insights:* The high average AUC indicates that the OvR logistic regression classifiers are effective at distinguishing each class from the rest, despite the dimensionality reduction from PCA. However, classes like 5 and 8, which have lower AUCs and accuracies, may suffer from feature loss during PCA, making them harder to separate. Compared to neural networks (e.g., CNN with 96.35% accuracy), logistic regression with PCA (84.60% accuracy) underperforms, highlighting the limitations of linear models for complex image data.

**Final Deliverable 3 Output:** The average AUC score across all 10 binary classifiers is **0.969**.

# 1 Question 3: Linear Regression and Support Vector Regression

## Question 3.1: Linear Regression

In this section, we apply Ordinary Least Squares (OLS) and Ridge Regression (RR) to two datasets, $\mathcal{D}_1$ and $\mathcal{D}_2$, provided by the oracle functions `q3_linear_1(23627)` and `q3_linear_2(23627)`. The datasets have the following characteristics:

- $\mathcal{D}_1$: 100 samples, 10 features.

- $\mathcal{D}_2$: 100 samples, 100 features.

Each dataset is split into training and test sets. We compute the weights $\mathbf{w}_1^{\text{ols}}, \mathbf{w}_1^{\text{rr}}$ for $\mathcal{D}_1$, and $\mathbf{w}_2^{\text{ols}}, \mathbf{w}_2^{\text{rr}}$ for $\mathcal{D}_2$, with $\lambda = 1.0$ for Ridge Regression.

### Deliverable 1: Can you do OLS if X does not have full column rank?

No, standard OLS cannot be performed directly if $\mathbf{X}$ does not have full column rank. In OLS, we aim to find the weight vector $\mathbf{w}$ for the linear model $\mathbf{y} = \mathbf{X}\mathbf{w}$, where $\mathbf{X} \in \mathbb{R}^{m \times d}$ is the data matrix (with $m$ training examples and $d$ features) and $\mathbf{y} \in \mathbb{R}^m$ is the target vector, The standard OLS solution is $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$. However, if $\mathbf{X}$ does not have full column rank (i.e., the rank of $\mathbf{X}$ is less than $d$), then $\mathbf{X}^\top \mathbf{X}$ is singular and not invertible, making the standard solution undefined.

In this case, we can use the pseudoinverse to compute a solution. The pseudoinverse $\mathbf{X}^+$ provides a generalized inverse, and the OLS solution becomes:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^+ \mathbf{X}^\top \mathbf{y}$$

This solution minimizes the squared error $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$ and provides a valid weight vector even when $\mathbf{X}$ does not have full column rank.

### Deliverable 2: MSE on $\mathcal{D}_1^{\text{train}}$ and $\mathcal{D}_2^{\text{train}}$

We compute the Mean Squared Error (MSE) for both OLS and Ridge Regression on the training sets of $\mathcal{D}_1$ and $\mathcal{D}_2$.

For $\mathcal{D}_1^{\text{train}}$:

- MSE for $\mathbf{w}_1^{\text{ols}}$: 0.03340458
- MSE for $\mathbf{w}_1^{\text{rr}}$: 0.11333904

For $\mathcal{D}_2^{\text{train}}$:

- MSE for $\mathbf{w}_2^{\text{ols}}$: 7115.79566132
- MSE for $\mathbf{w}_2^{\text{rr}}$: 2.90381334

Ridge Regression typically results in a higher MSE on the training set due to the regularization term, which helps prevent overfitting.

### Deliverable 3: MSE on $\mathcal{D}_2^{\text{train}}$ and Weight Files

I reported the MSE on $\mathcal{D}_2^{\text{train}}$ (already provided above) and attached the weight vectors $\mathbf{w}_2^{\text{ols}}$ and $\mathbf{w}_2^{\text{rr}}$ as CSV files named `w_ols_23627.csv` and `w_rr_23627.csv`, respectively.

## Question 3.2: Support Vector Regression (SVR)

In this section, we apply Support Vector Regression (SVR) to predict stock prices for the stock `MRK`, as determined by the oracle function `q3_stocknet(23627)`.

### Deliverable : Plots of Predicted vs. Actual Prices

For each SVR model ($t \in \{7, 30, 90\}$, linear and RBF with $\gamma \in \{1, 0.1, 0.01, 0.001\}$), we plot the following on the test set:

1. Predicted closing price.
2. Actual closing price.
3. Average price of the previous $t$ days.

### Linear SVR Plots for Different $t$ Values

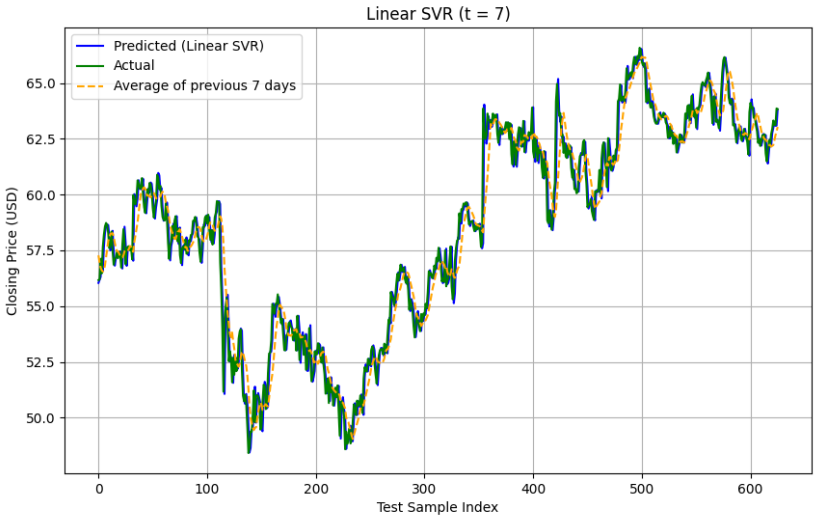


Figure 9: Linear SVR predictions for $t = 7$.

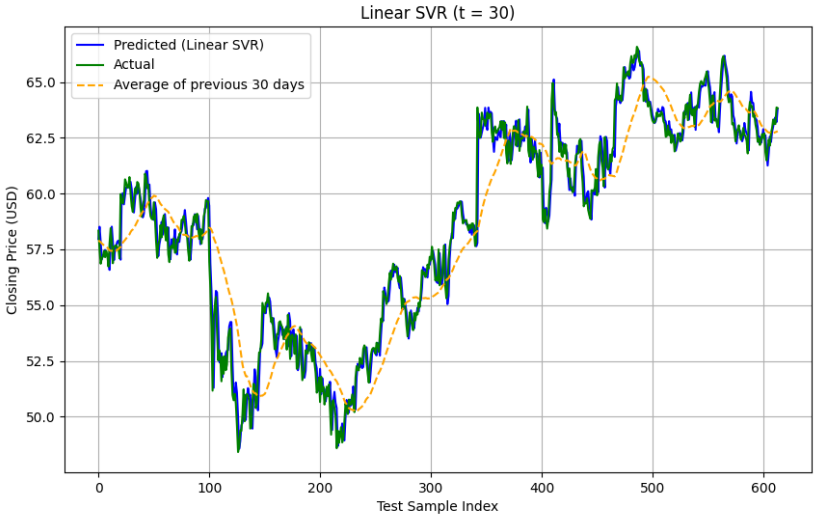


Figure 10: Linear SVR predictions for $t = 30$.

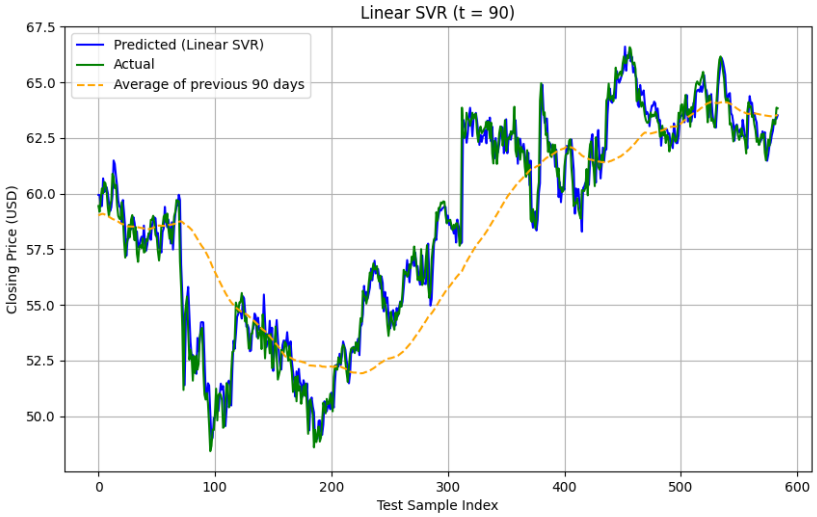


Figure 11: Linear SVR predictions for $t = 90$.

**RBF SVR Plots for $t = 7$ with Varying $\gamma$**

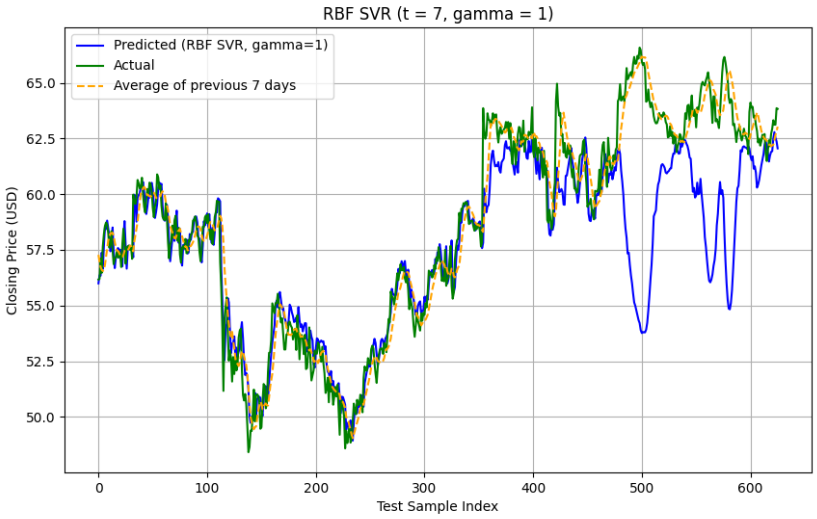


Figure 12: RBF SVR predictions for $t = 7$, $\gamma = 1$.

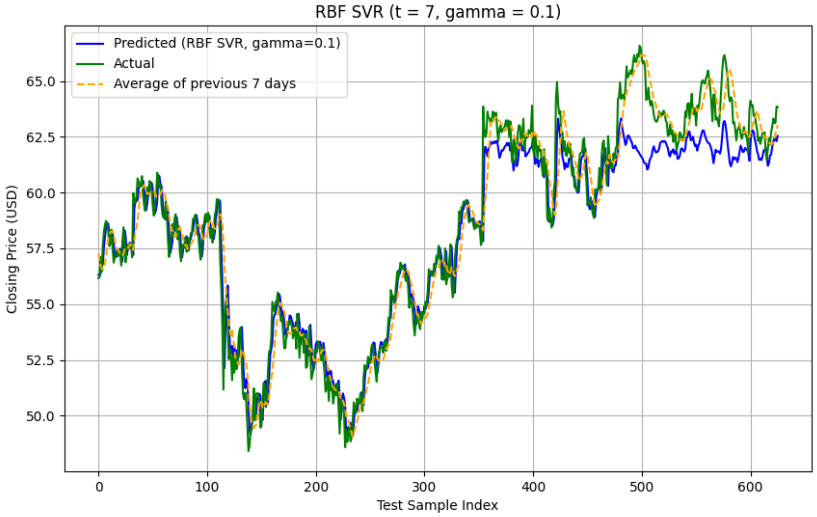


Figure 13: RBF SVR predictions for $t = 7$, $\gamma = 0.1$.

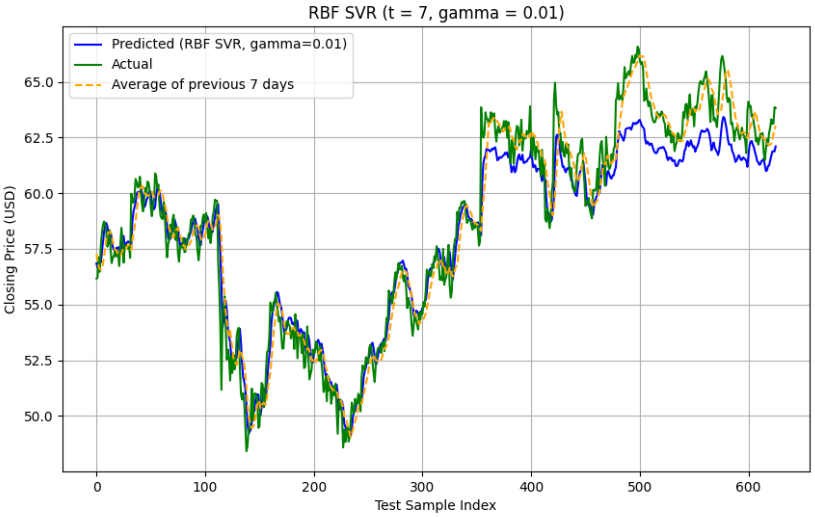


Figure 14: RBF SVR predictions for $t = 7$, $\gamma = 0.01$.

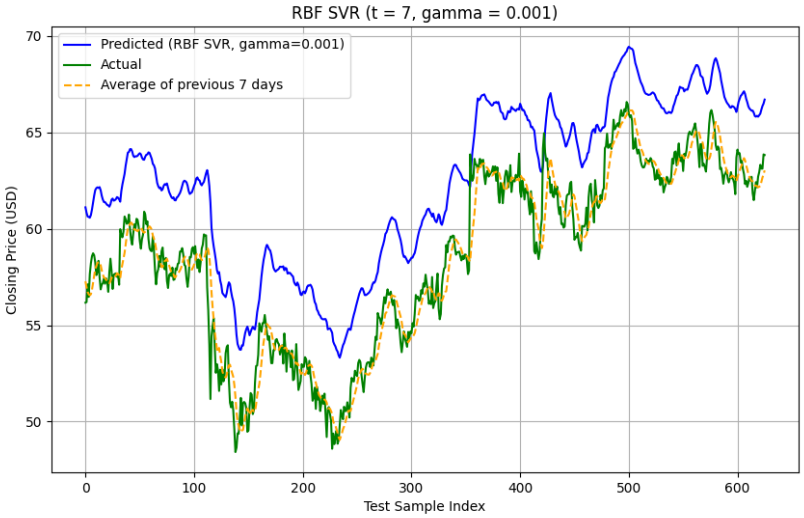


Figure 15: RBF SVR predictions for $t = 7$, $\gamma = 0.001$.

### RBF SVR Plots for $t = 30$ with Varying $\gamma$

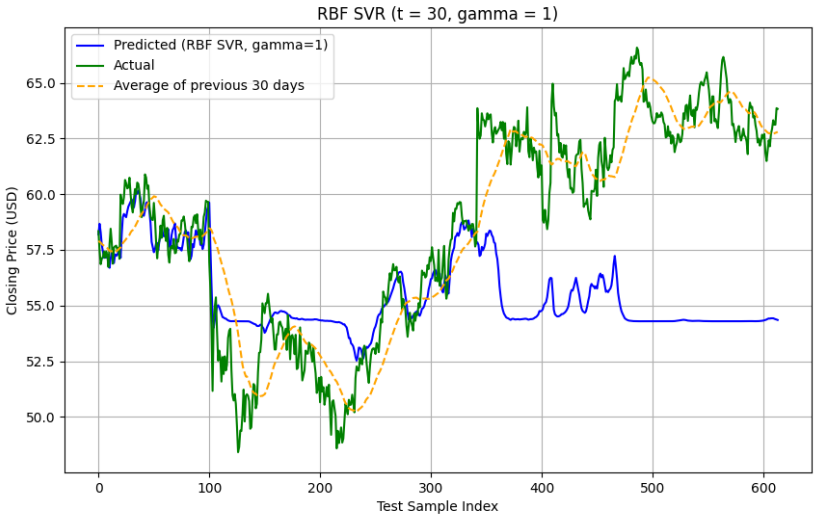


Figure 16: RBF SVR predictions for $t = 30$, $\gamma = 1$.

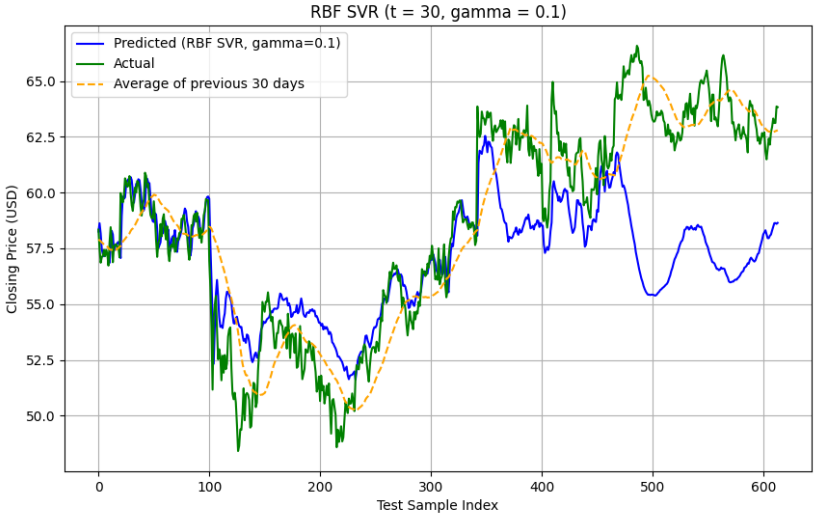


Figure 17: RBF SVR predictions for $t = 30$, $\gamma = 0.1$.

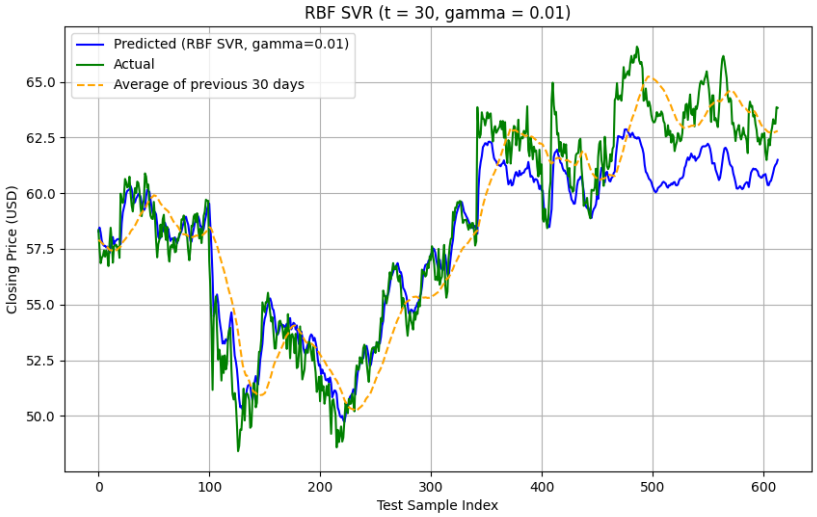


Figure 18: RBF SVR predictions for $t = 30$, $\gamma = 0.01$.

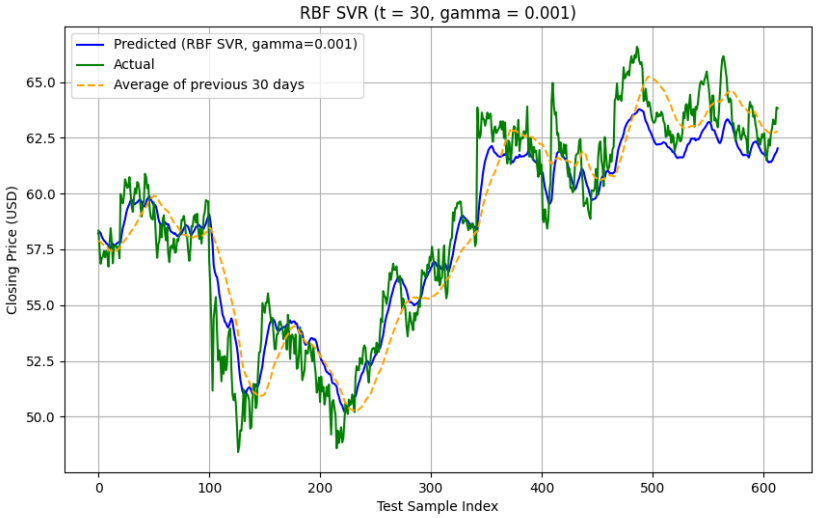


Figure 19: RBF SVR predictions for $t = 30$, $\gamma = 0.001$.

**RBF SVR Plots for $t = 90$ with Varying $\gamma$**

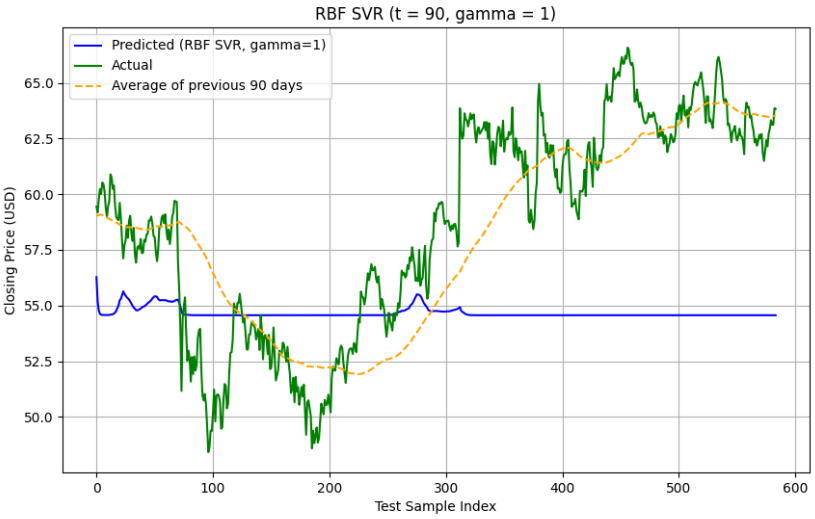


Figure 20: RBF SVR predictions for $t = 90$, $\gamma = 1$.

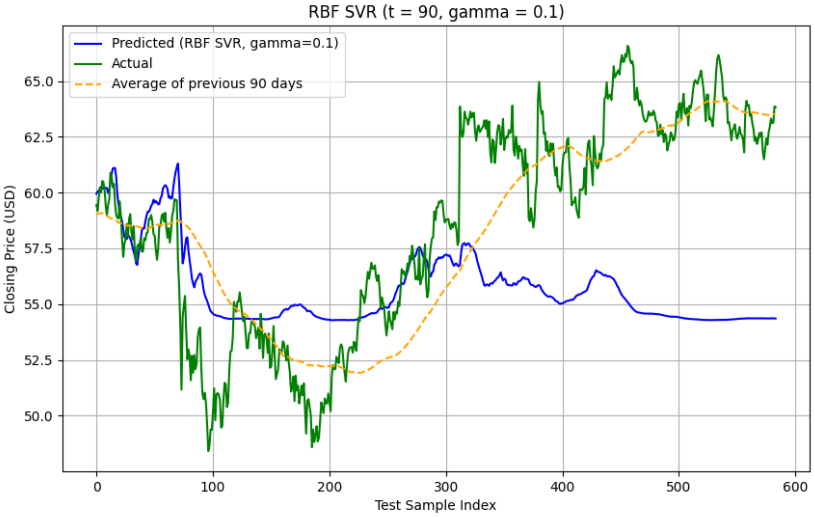


Figure 21: RBF SVR predictions for $t = 90$, $\gamma = 0.1$.

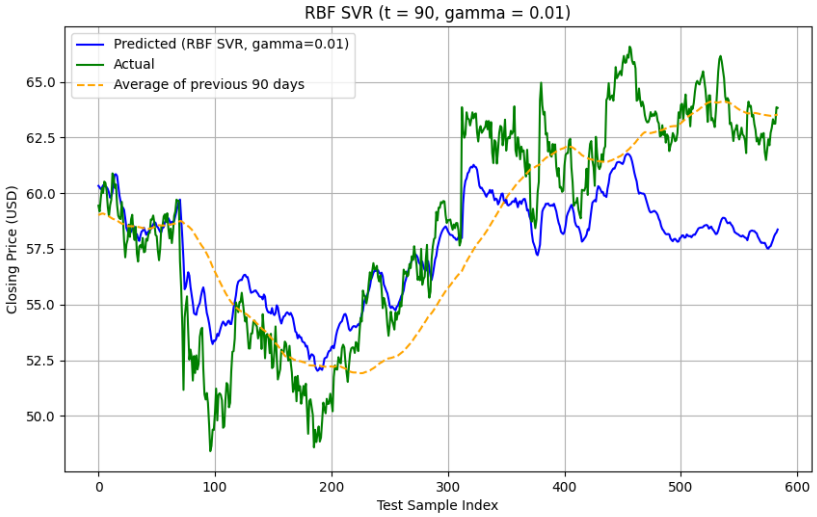


Figure 22: RBF SVR predictions for $t = 90$, $\gamma = 0.01$.

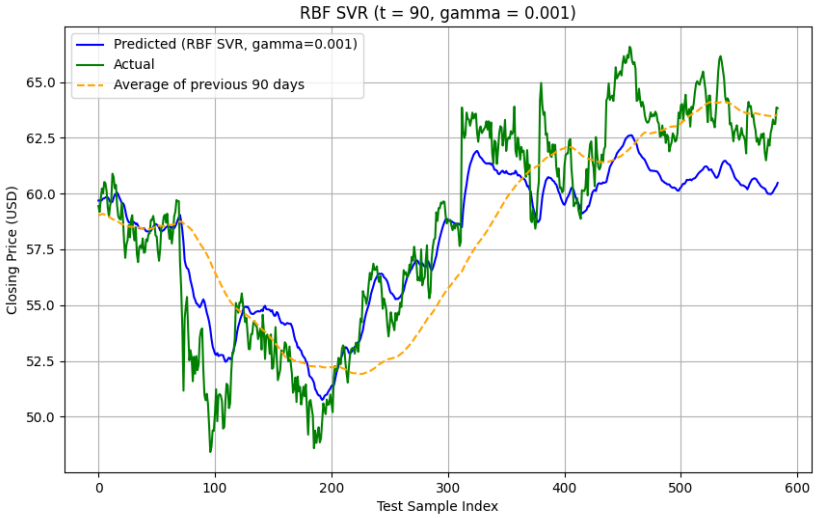


Figure 23: RBF SVR predictions for $t = 90$, $\gamma = 0.001$.