Programming Assignment 3 Part A

LANKA RAMA KRISHNA 22EE30037

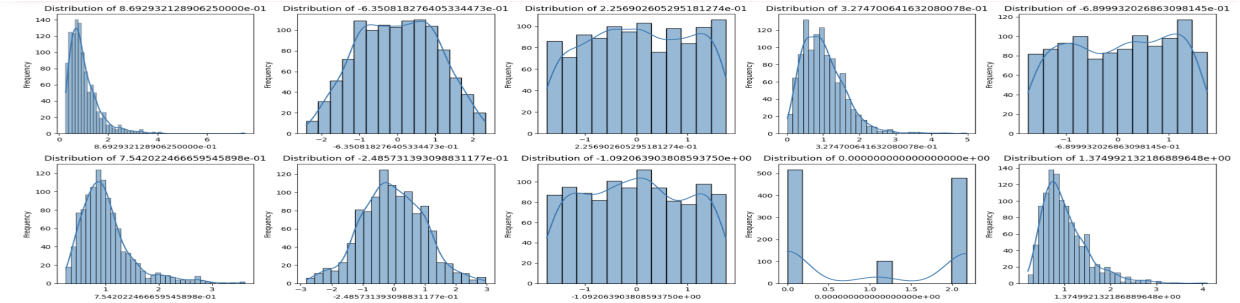
November 5, 2024

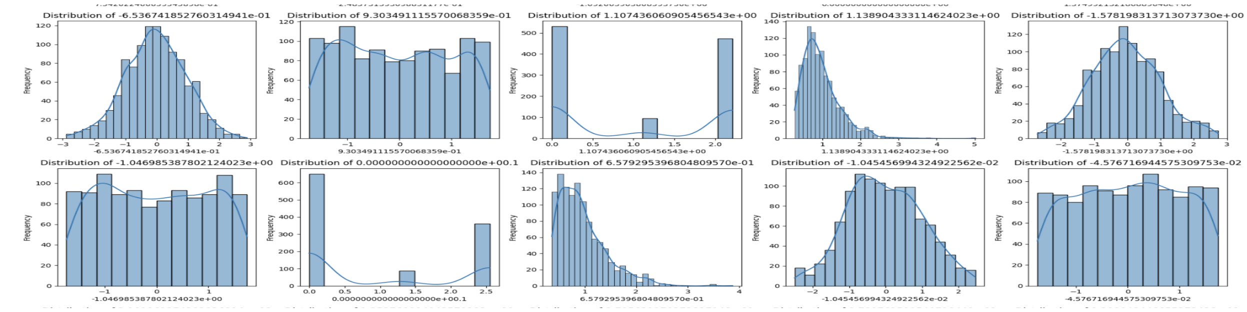
Support Vector Machines (SVMs) and Kernel Methods

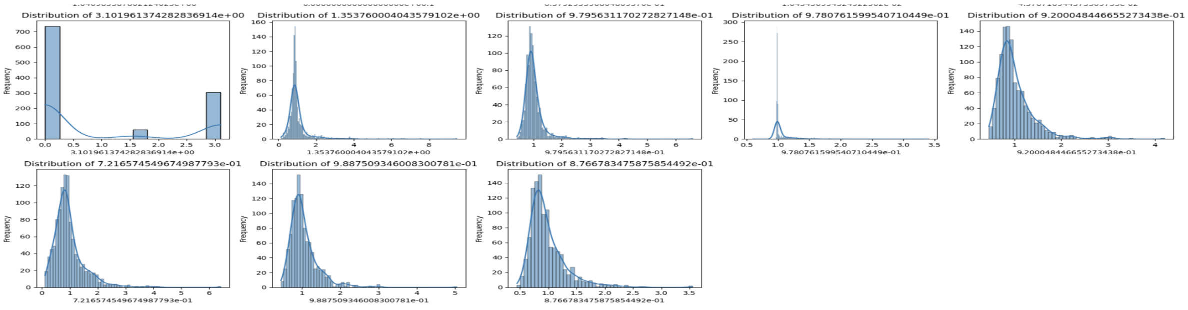
1 Data Preprocessing and Exploration

1.1 Exploratory Data Analysis (EDA)

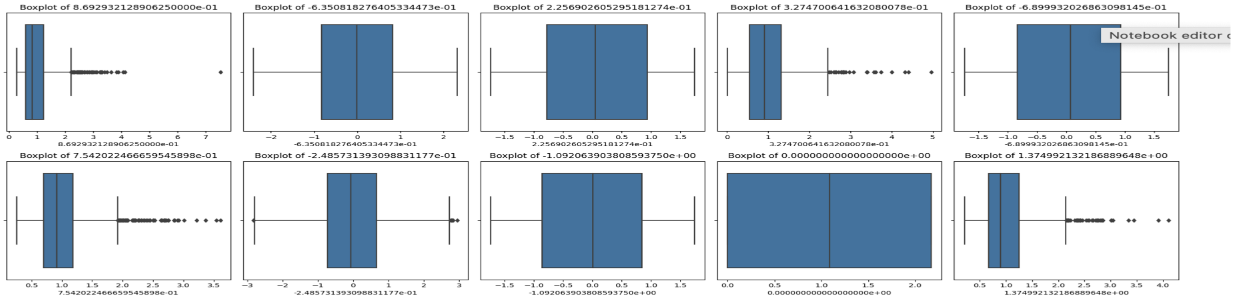
Feature Distribution plots

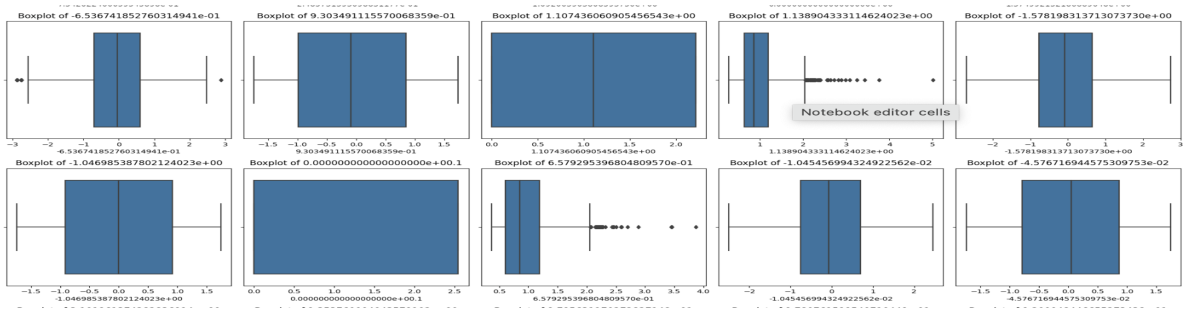


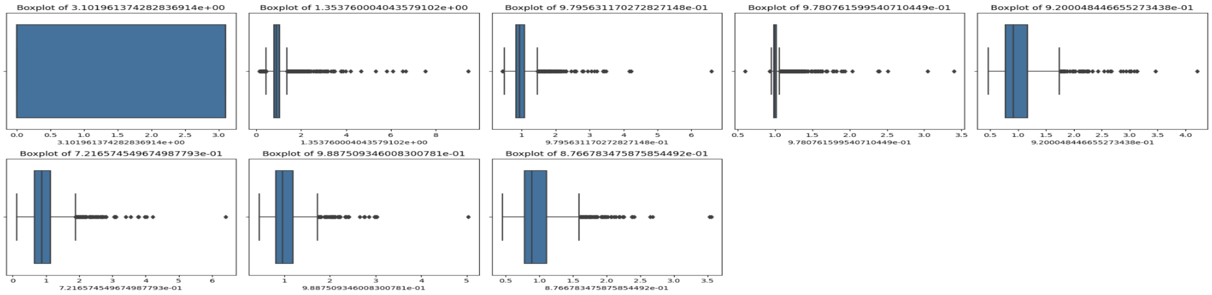




Box plots for Detecting outliers





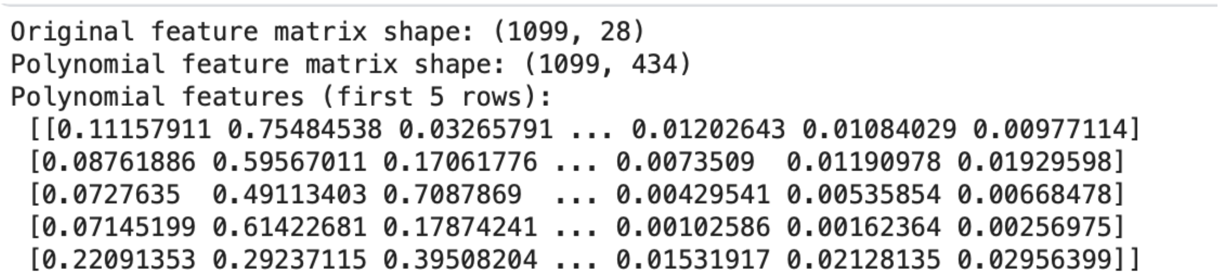


1.2 Data Normalization/Standardization

We use MinMaxScaler to normalize stratifiedsample, bringing all features into a [0, 1] range, and convert the result back to a DataFrame for easier analysis. This makes feature values comparable across columns.

1.3 Feature Engineering

We added new features with Polynomial function of degree 2 using the existing features .

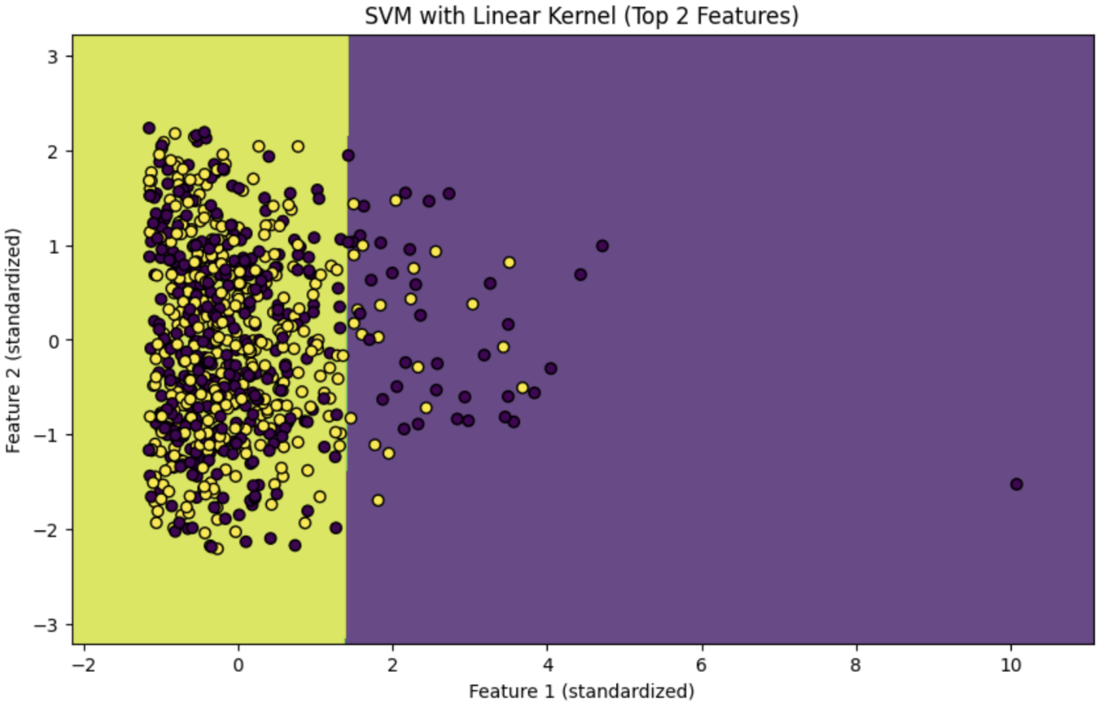


```
Original feature matrix shape: (1099, 28)
Polynomial feature matrix shape: (1099, 434)
Polynomial features (first 5 rows):
 [[0.11157911 0.75484538 0.03265791 ... 0.01202643 0.01084029 0.00977114]
 [0.08761886 0.59567011 0.17061776 ... 0.0073509  0.01190978 0.01929598]
 [0.0727635  0.49113403 0.7087869  ... 0.00429541 0.00535854 0.00668478]
 [0.07145199 0.61422681 0.17874241 ... 0.00102586 0.00162364 0.00256975]
 [0.22091353 0.29237115 0.39508204 ... 0.01531917 0.02128135 0.02956399]]
```

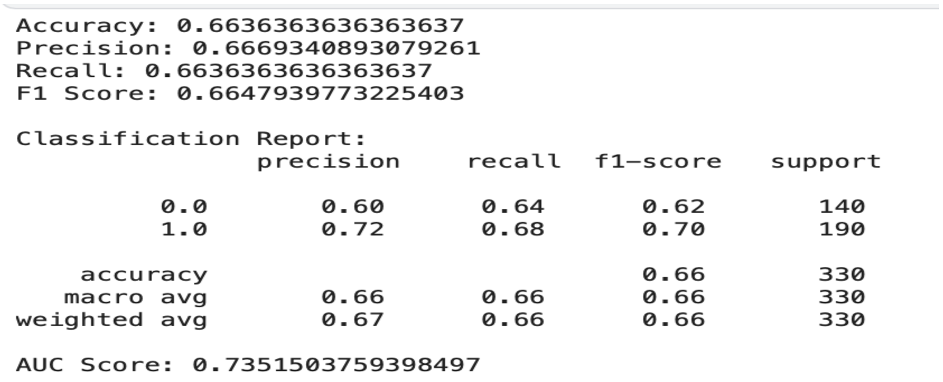
we selected top 20 features from all these features for further analysis

2 Linear SVM implementation

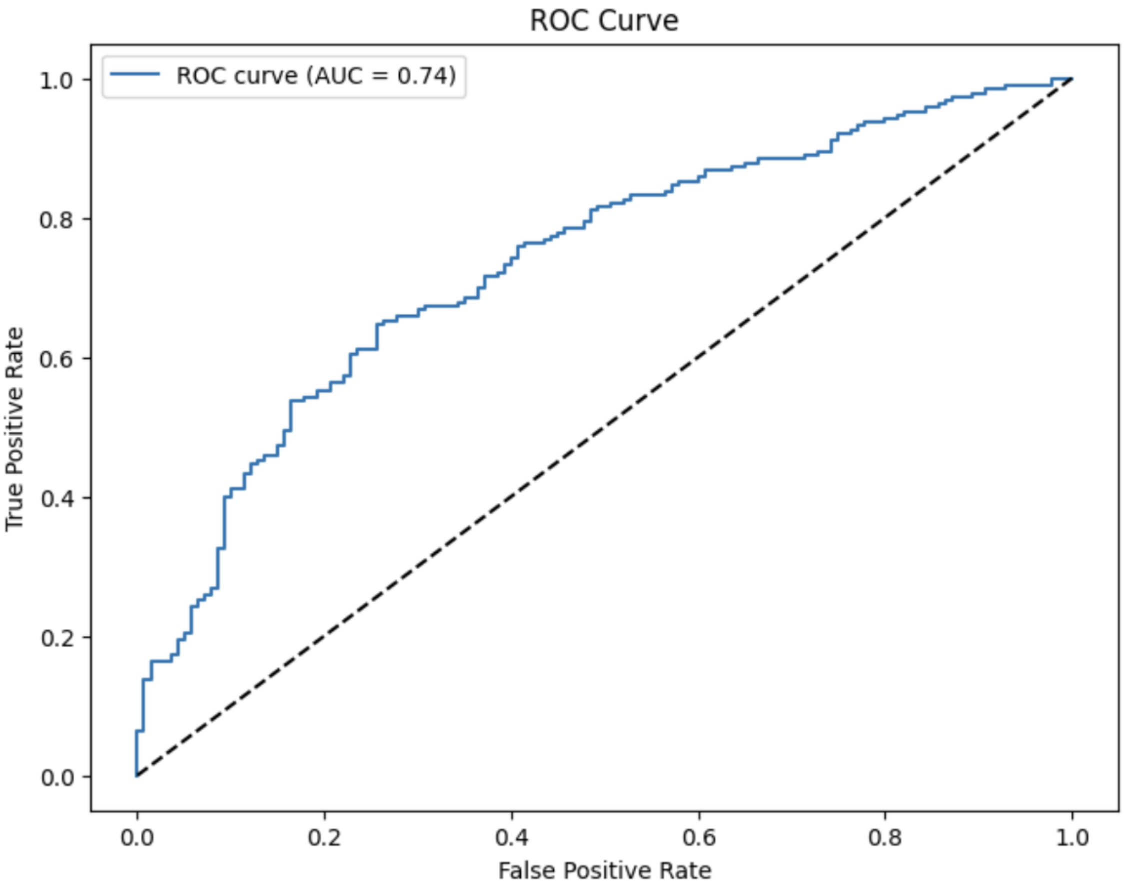
We first checked if we had exactly two features selected in Xselected so that we could visualize the decision boundary. Since we did, we trained an SVM with a linear kernel on these features and plotted the decision boundary with a contour plot, showing class separation. Then, we applied 5-fold cross-validation on the full dataset using a pipeline for standard scaling and a linear SVM model. Finally, we displayed both the cross-validation scores and their mean.

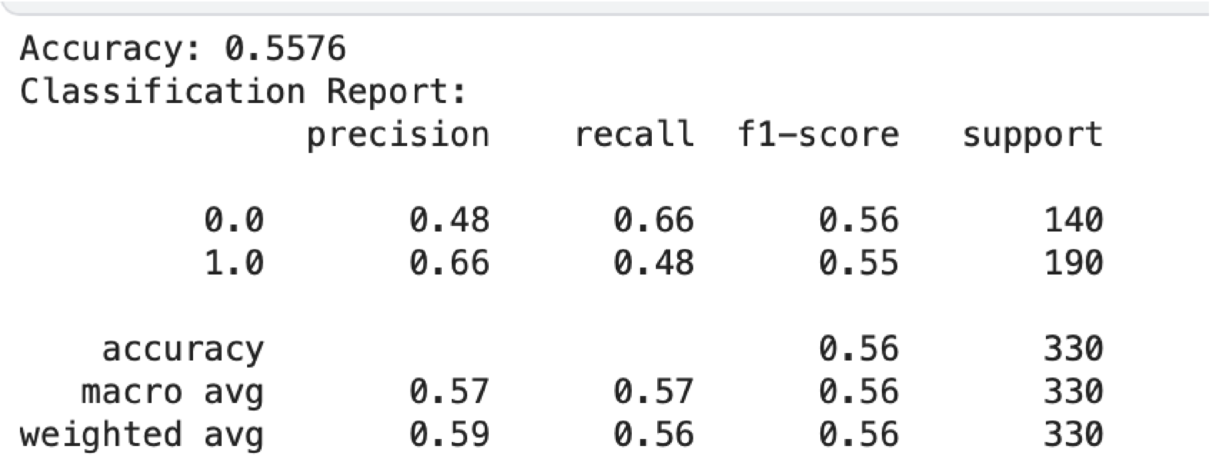


```
Cross-validation scores: [0.54545455 0.59090909 0.51948052 0.62987013 0.55555556]
Mean cross-validation score: 0.5682539682539682
```



```
Accuracy: 0.6636363636363637
Precision: 0.6669340893079261
Recall: 0.6636363636363637
F1 Score: 0.6647939773225403

Classification Report:
               precision    recall  f1-score   support

         0.0       0.60      0.64      0.62       140
         1.0       0.72      0.68      0.70       190

    accuracy                           0.66       330
   macro avg       0.66      0.66      0.66       330
weighted avg       0.67      0.66      0.66       330

AUC Score: 0.7351503759398497
```





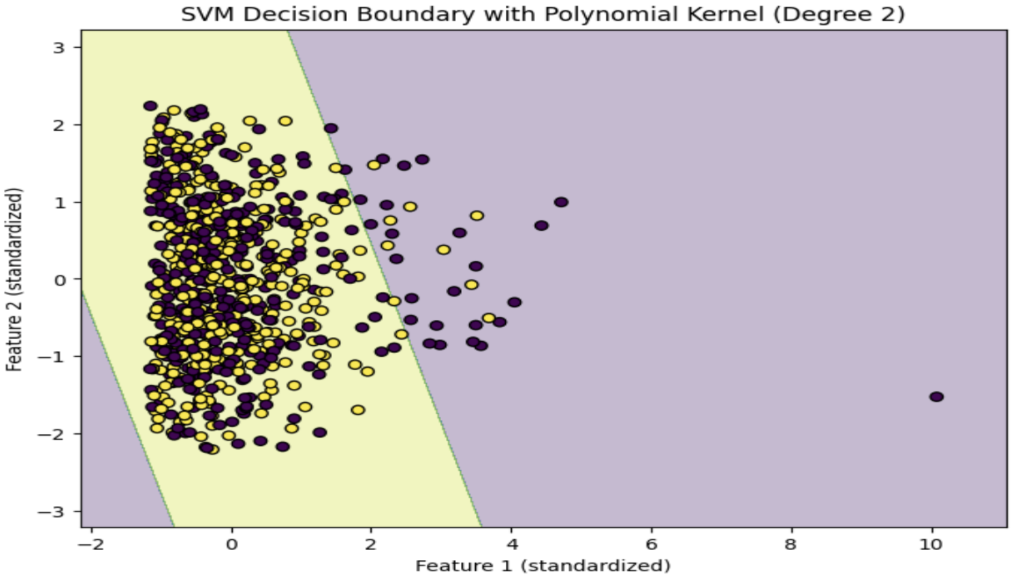
```
Accuracy: 0.5576
Classification Report:
              precision    recall  f1-score   support

         0.0       0.48      0.66      0.56       140
         1.0       0.66      0.48      0.55       190

    accuracy                           0.56       330
   macro avg       0.57      0.57      0.56       330
weighted avg       0.59      0.56      0.56       330
```

accuracy metrics in stochastic gradient descent for linear svm

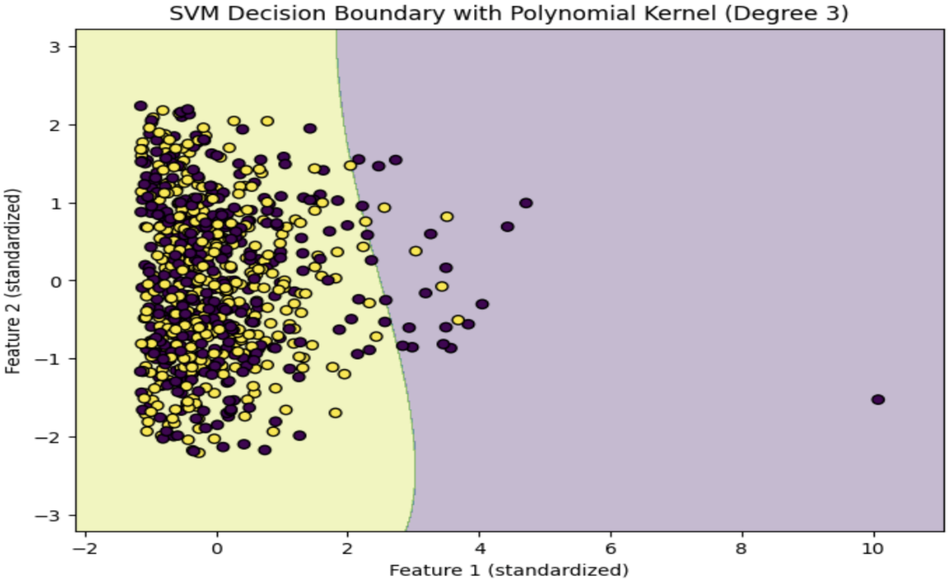
3 SVM with Polynomial, RBF, and Custom Kernels

3.1 Polynomial Kernel

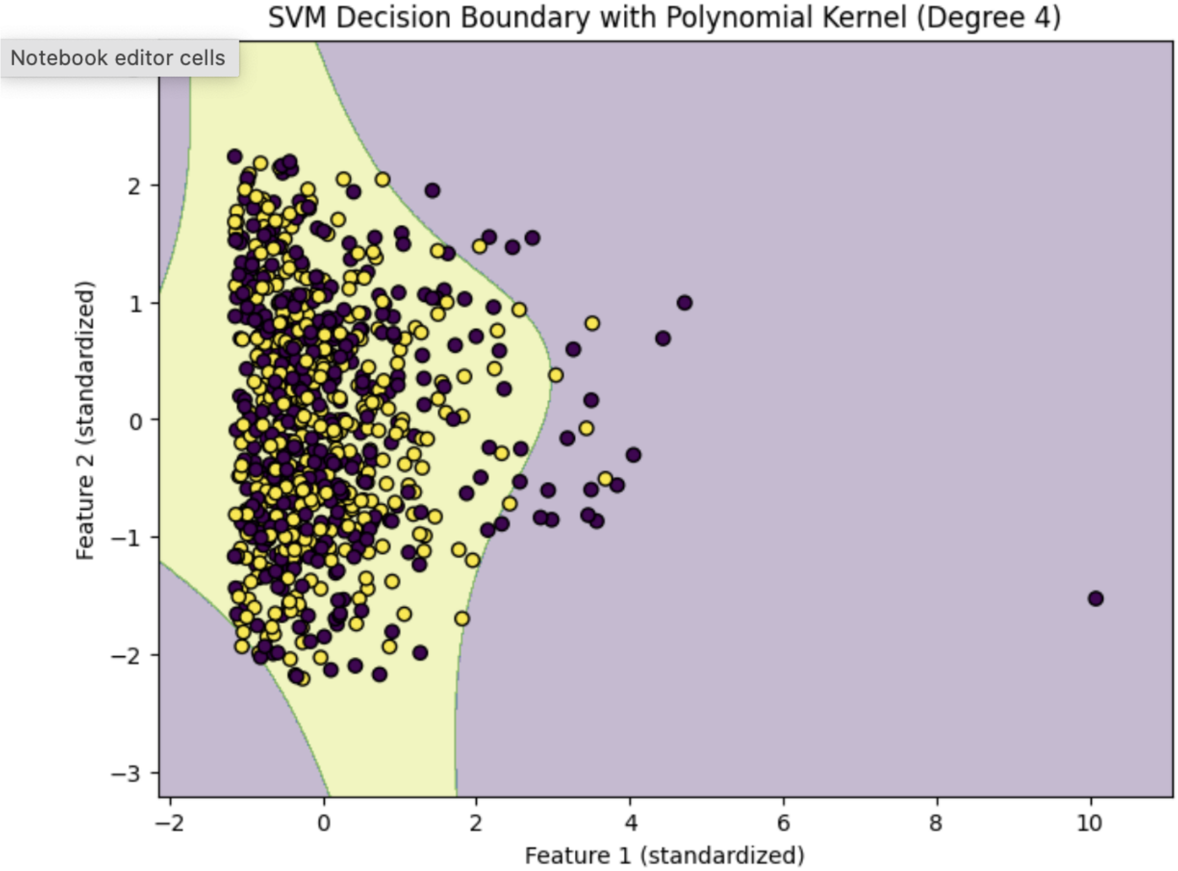
We first checked if we had exactly two features selected in ‘X_selected‘ to allow visualization of decision boundaries. Since this condition was met, we trained an SVM with polynomial kernels of degrees 2, 3, and 4 on these features, visualizing the decision boundaries for each degree with contour plots to show class separation. We then conducted 5-fold cross-validation on the full dataset using a pipeline with standard scaling and a polynomial kernel SVM, displaying the cross-validation scores and their mean.



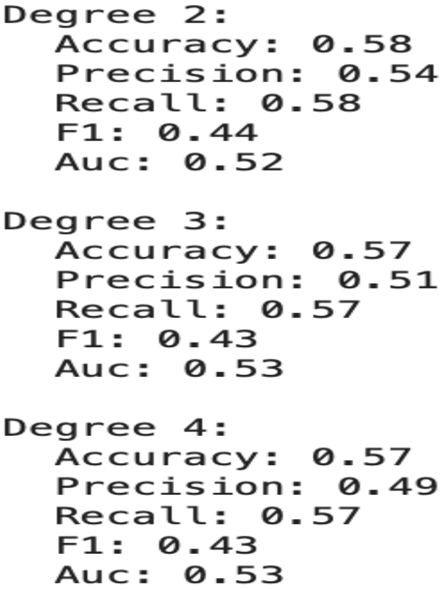
SVM Decision Boundary with Polynomial Kernel (Degree 2)



SVM Decision Boundary with Polynomial Kernel (Degree 3)



SVM Decision Boundary with Polynomial Kernel (Degree 4)

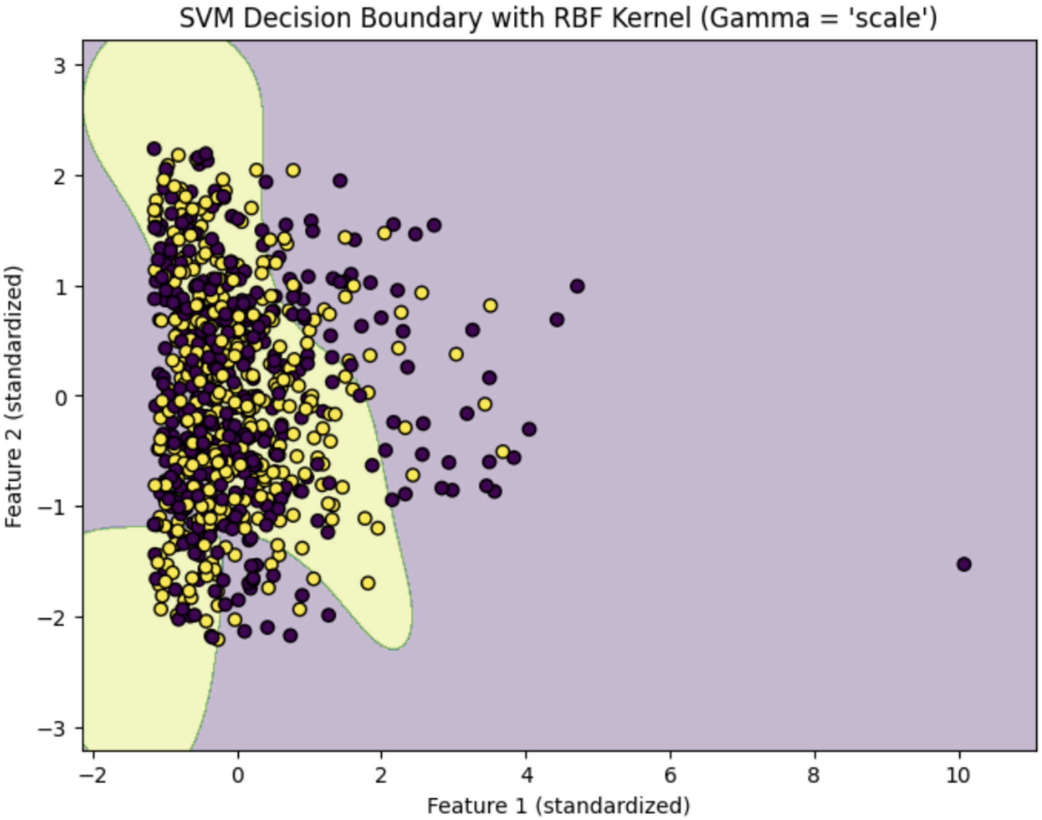


```
Degree 2:
  Accuracy: 0.58
  Precision: 0.54
  Recall: 0.58
  F1: 0.44
  Auc: 0.52

Degree 3:
  Accuracy: 0.57
  Precision: 0.51
  Recall: 0.57
  F1: 0.43
  Auc: 0.53

Degree 4:
  Accuracy: 0.57
  Precision: 0.49
  Recall: 0.57
  F1: 0.43
  Auc: 0.53
```

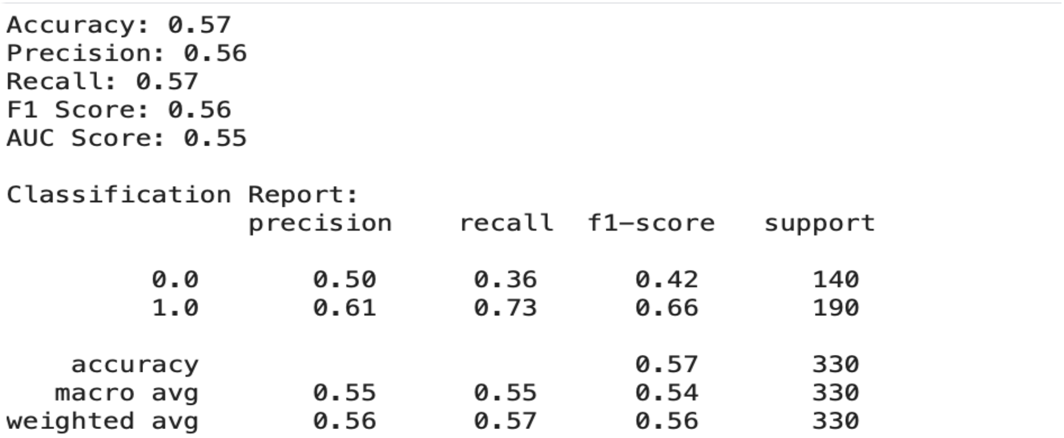
Accuracy Metrics for Polynomial SVM

3.2 RBF Kernel

We first checked if we had exactly two features selected in ‘X_selected‘ to enable visualization of decision boundaries. Since this was the case, we trained an SVM with an RBF kernel on these features, visualizing the decision boundary to illustrate class separation. Then, we performed 5-fold cross-validation on the full dataset using a pipeline with standard scaling and an RBF kernel SVM, and displayed both the cross-validation scores and their mean.



SVM Decision Boundary with RBF Kernel

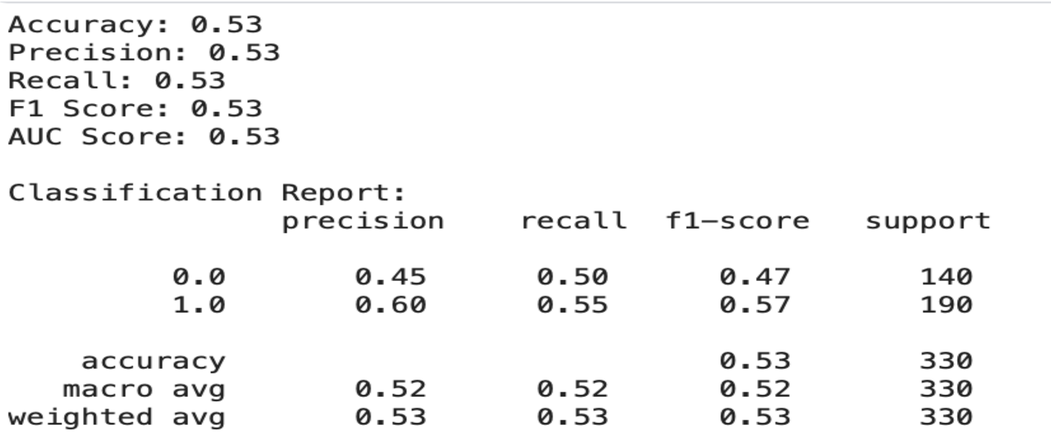


```
Accuracy: 0.57
Precision: 0.56
Recall: 0.57
F1 Score: 0.56
AUC Score: 0.55

Classification Report:
              precision    recall  f1-score   support

         0.0       0.50      0.36      0.42       140
         1.0       0.61      0.73      0.66       190

    accuracy                           0.57       330
   macro avg       0.55      0.55      0.54       330
weighted avg       0.56      0.57      0.56       330
```

Accuracy Metrics in for RBF SVM

3.3 Custom Kernel(sigmoid)

We first checked if we had exactly two features selected in 'X_selected' to allow visualization of decision boundaries. Since this condition was met, we trained an SVM with a Sigmoid kernel on these features, visualizing the decision boundary with a contour plot to show class separation. We then conducted 5-fold cross-validation on the full dataset using a pipeline with standard scaling and a Sigmoid kernel SVM, displaying both the cross-validation scores and their mean.



```
Accuracy: 0.53
Precision: 0.53
Recall: 0.53
F1 Score: 0.53
AUC Score: 0.53

Classification Report:
              precision    recall  f1-score   support

         0.0       0.45      0.50      0.47       140
         1.0       0.60      0.55      0.57       190

    accuracy                           0.53       330
   macro avg       0.52      0.52      0.52       330
weighted avg       0.53      0.53      0.53       330
```

Accuracy Metrics for Sigmoid SVM

3.4 Time Complexity Analysis

The time complexity of Support Vector Machines (SVM) varies depending on the choice of kernel. Below is a breakdown of the time complexity for each kernel used in our experiments:

- **Linear Kernel:** The time complexity of an SVM with a linear kernel is generally $O(n \times d)$, where n is the number of samples and d is the dimensionality of the feature space. Linear SVMs are typically faster than other kernels, making them suitable for high-dimensional data. However, this complexity can increase to $O(n^2)$ for datasets that are not linearly separable, requiring additional computation to identify support vectors.

- **Polynomial Kernel:** The polynomial kernel has a time complexity of $O(n^2 \times d)$ for training, as each pairwise comparison in the feature space must be computed and raised to the specified degree p. The time complexity increases with the degree of the polynomial, which can significantly slow down training for higher-degree polynomials. Prediction complexity is similarly $O(n \times d)$, as each test instance requires evaluation against all support vectors.

- **RBF Kernel:** The RBF kernel is computationally more expensive, with a training time complexity of $O(n^2 \times d)$, as it involves calculating the Gaussian similarity between each pair of data points in n-dimensional space. For very large datasets, this complexity can increase further, especially if cross-validation is required to tune the kernel parameter γ. Prediction time complexity remains $O(n \times d)$, as each test instance is compared with support vectors.

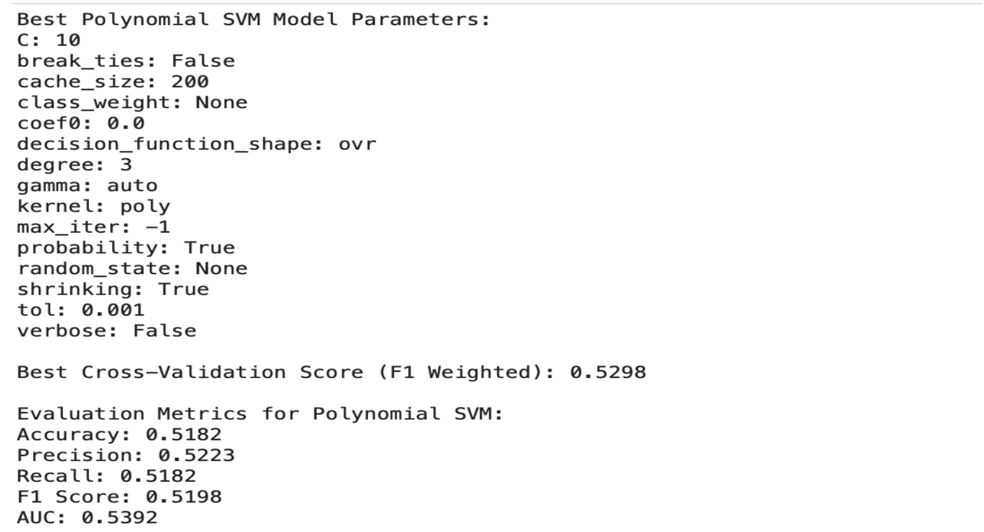
- **Sigmoid Kernel:** The sigmoid kernel has a time complexity similar to the RBF kernel, typically $O(n^2 \times d)$ for training. This kernel calculates the hyperbolic tangent of the inner product, introducing non-linearity while maintaining a complexity that scales quadratically with the number of samples. Prediction complexity remains $O(n \times d)$, but the sigmoid kernel may be more sensitive to parameter settings, requiring careful tuning.

Overall, the time complexity for each kernel is summarized as follows:

Kernel Type	Time Complexity (Training)
Linear	$O(n \times d)$
Polynomial	$O(n^2 \times d)$
RBF	$O(n^2 \times d)$
Sigmoid	$O(n^2 \times d)$

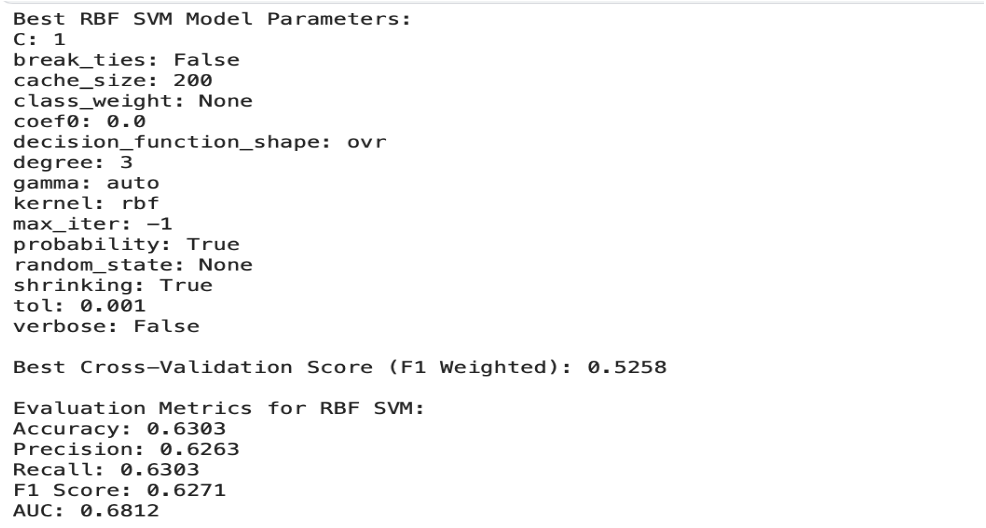
For prediction, all kernels exhibit a time complexity of $O(n \times d)$, though higher-dimensional feature spaces and larger training sets can increase computational costs across the board.

4 Hyperparameter tuning



```
Best Polynomial SVM Model Parameters:
C: 10
break_ties: False
cache_size: 200
class_weight: None
coef0: 0.0
decision_function_shape: ovr
degree: 3
gamma: auto
kernel: poly
max_iter: -1
probability: True
random_state: None
shrinking: True
tol: 0.001
verbose: False

Best Cross-Validation Score (F1 Weighted): 0.5298

Evaluation Metrics for Polynomial SVM:
Accuracy: 0.5182
Precision: 0.5223
Recall: 0.5182
F1 Score: 0.5198
AUC: 0.5392
```

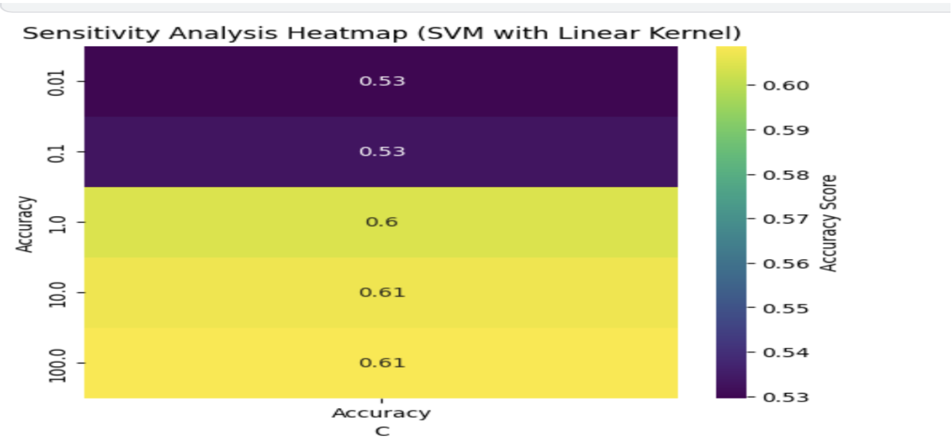
Best parameters for polynomial kernel



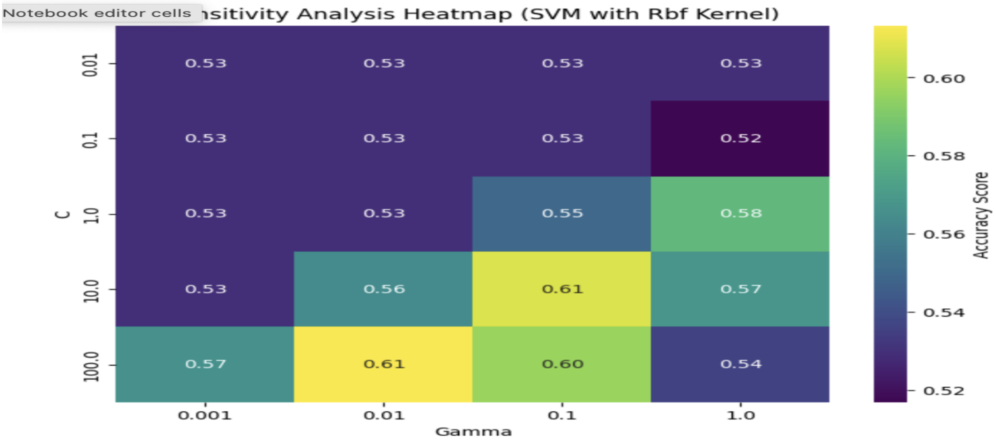
```
Best RBF SVM Model Parameters:
C: 1
break_ties: False
cache_size: 200
class_weight: None
coef0: 0.0
decision_function_shape: ovr
degree: 3
gamma: auto
kernel: rbf
max_iter: -1
probability: True
random_state: None
shrinking: True
tol: 0.001
verbose: False

Best Cross-Validation Score (F1 Weighted): 0.5258

Evaluation Metrics for RBF SVM:
Accuracy: 0.6303
Precision: 0.6263
Recall: 0.6303
F1 Score: 0.6271
AUC: 0.6812
```

Best parameters for RBF kernel

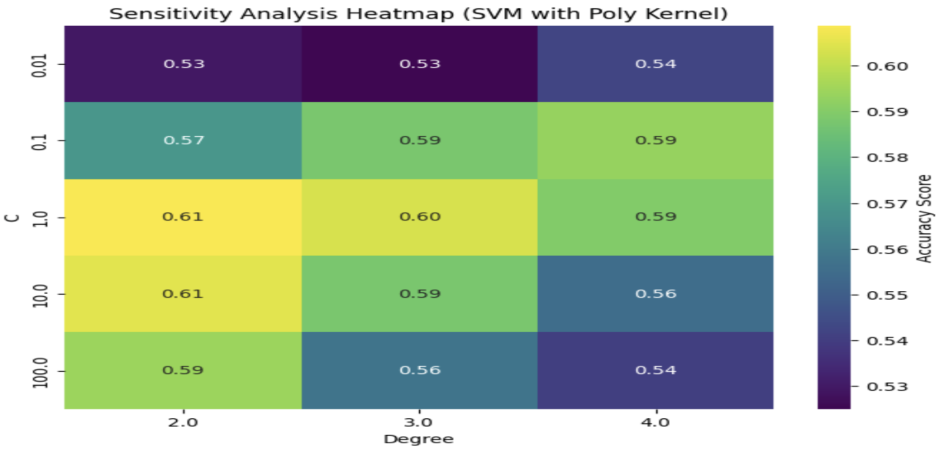
4.1 Hyperparameter Sensitivity Analysis



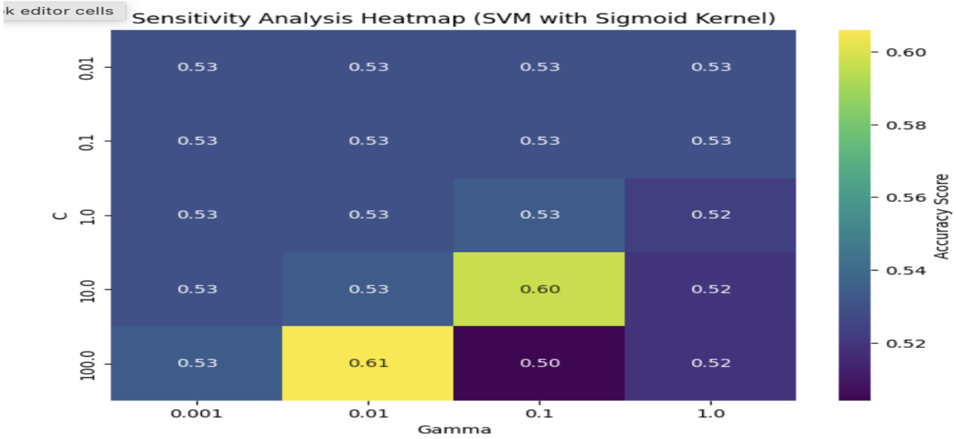
Heat map for Linear kernel



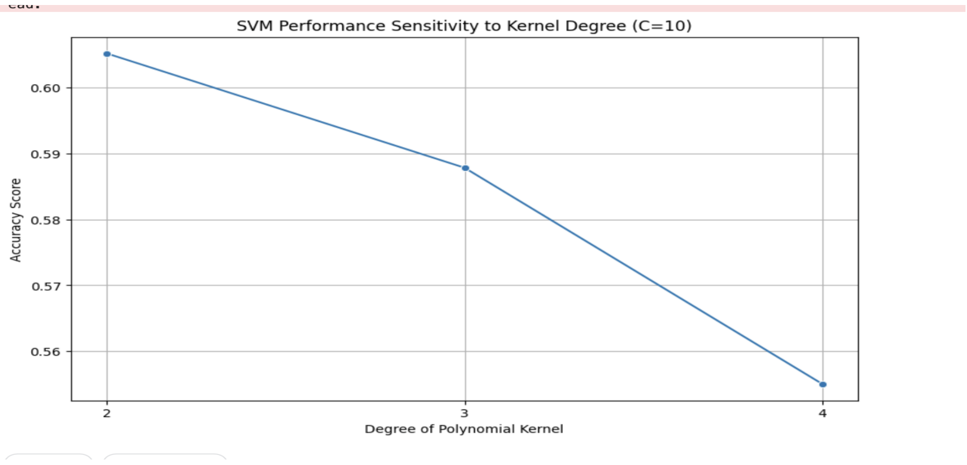
Heat map for RBF kernel



Heat map for Polynomial kernel



Heat map for Sigmoid kernel



Line plot showing performance vs degree for Polynomial kernel