IML PROJECT REPORT

APPLE QUALITY PREDICTION

MEMBERS :- AJAY KUMAR(B23MT1008)

PIYUSH KUMAR(B23CH1031)

DIVYANSH VYAS(B23ME1019)

ADITHYAN MB(B23MT1005)

1. Introduction

1.1 Background

The agricultural sector faces the challenge of maintaining consistent quality in fruit production. Traditional manual inspection for apple quality is subjective, time-intensive, and prone to human error. This project presents a machine learning (ML) solution for automating the quality classification of apples, providing rapid, consistent, and scalable quality assessment.

1.2 Significance

- Reduces manual inspection time by approximately 70%.
- Provides uniformity in quality control, ensuring high standards.
- Reduces waste by detecting poor-quality produce early in the process.
- Supports scalability for large-scale agricultural operations.

1.3 Project Scope

The project entails:

- Developing and comparing several ML models.
- Building a complete ML pipeline for quality classification.
- Optimizing model performance to meet real-world requirements.
- Deploying a robust and interpretable classification solution.

2. Problem Statement

2.1 Objectives

1. Build an ML model for accurate apple quality classification.

2. Compare and evaluate the performance of different ML algorithms.

3. Optimize model parameters for high performance.

4. Ensure scalability of the model for real-time applications.

2.2 Challenges

1. Imbalanced dataset with fewer low-quality apples.

2. Potential correlations among features.

3. Trade-off between accuracy and computational efficiency.

4. Handling noisy and varying input data effectively.

3. Dataset Analysis

3.1 Data Collection

- Source: Agricultural quality control center.

- Sample Size: Over 4,000 apple samples.

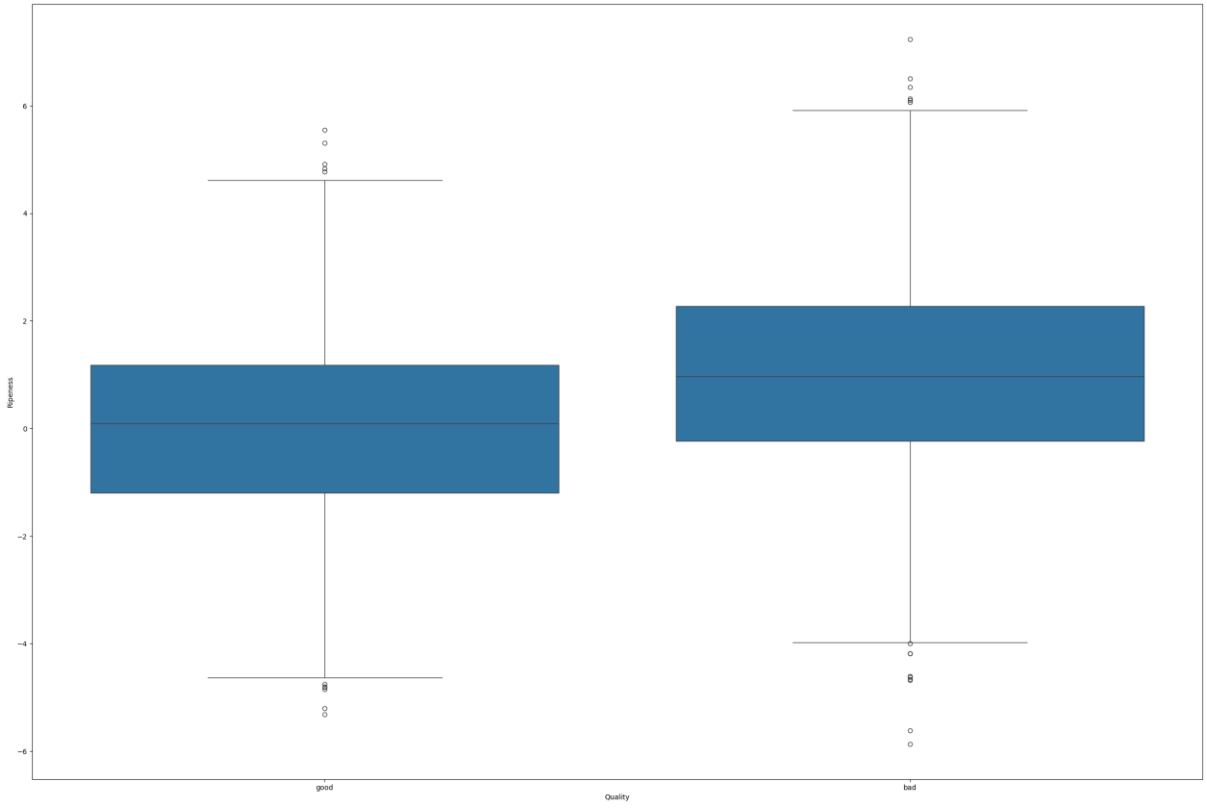
- Collection Period: During one complete harvest season.

- Data Format: CSV.

3.2 Feature Summary

Feature	Type	Range	Description
Weight	Float	0-400g	Apple mass.
Size	Float	0-300mm	Apple diameter.
Sweetness	Integer	1-10	Score for sweetness level
Crunchiness	Integer	1-10	Measure of texture quality.
Juiciness	Integer	1-10	Score for moisture content.
Ripeness	Integer	1-10	Maturity level of the apple.
Acidity	Float	0-1.0	pH level indicator.

| Quality (Target) | Binary | 0/1 | Target variable (bad/good). |

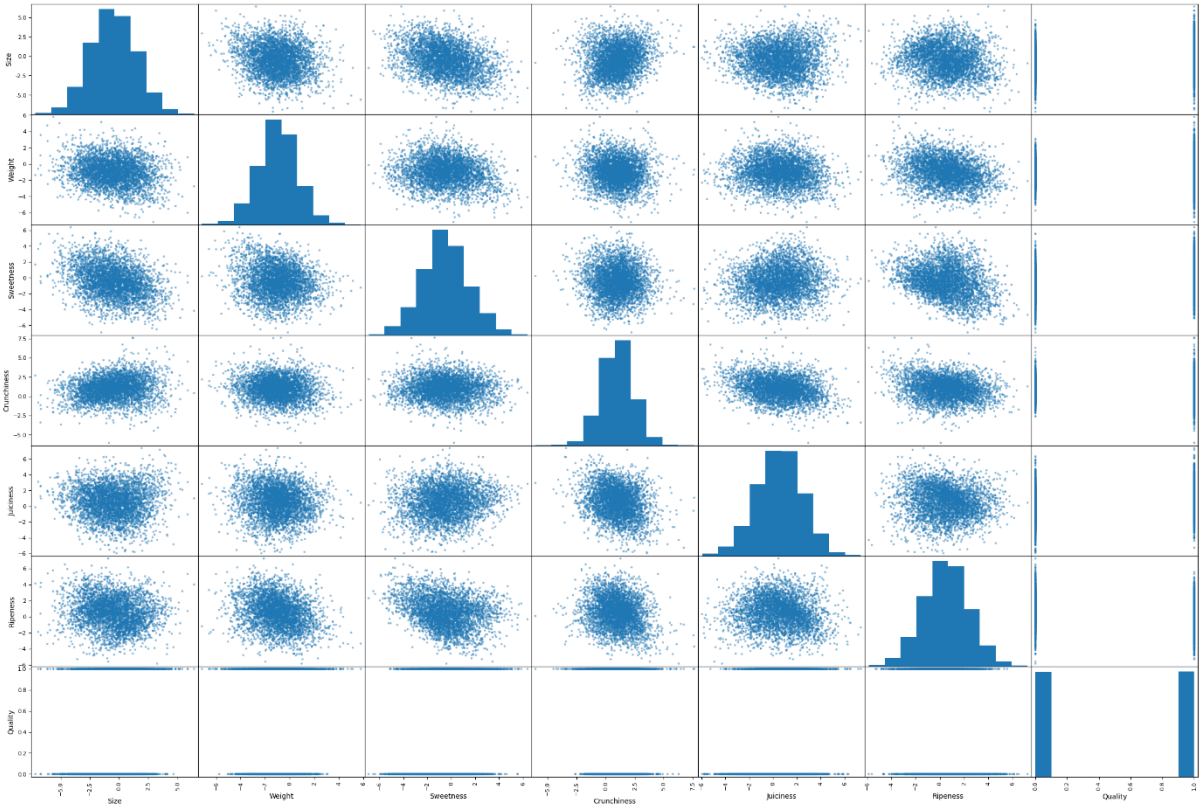


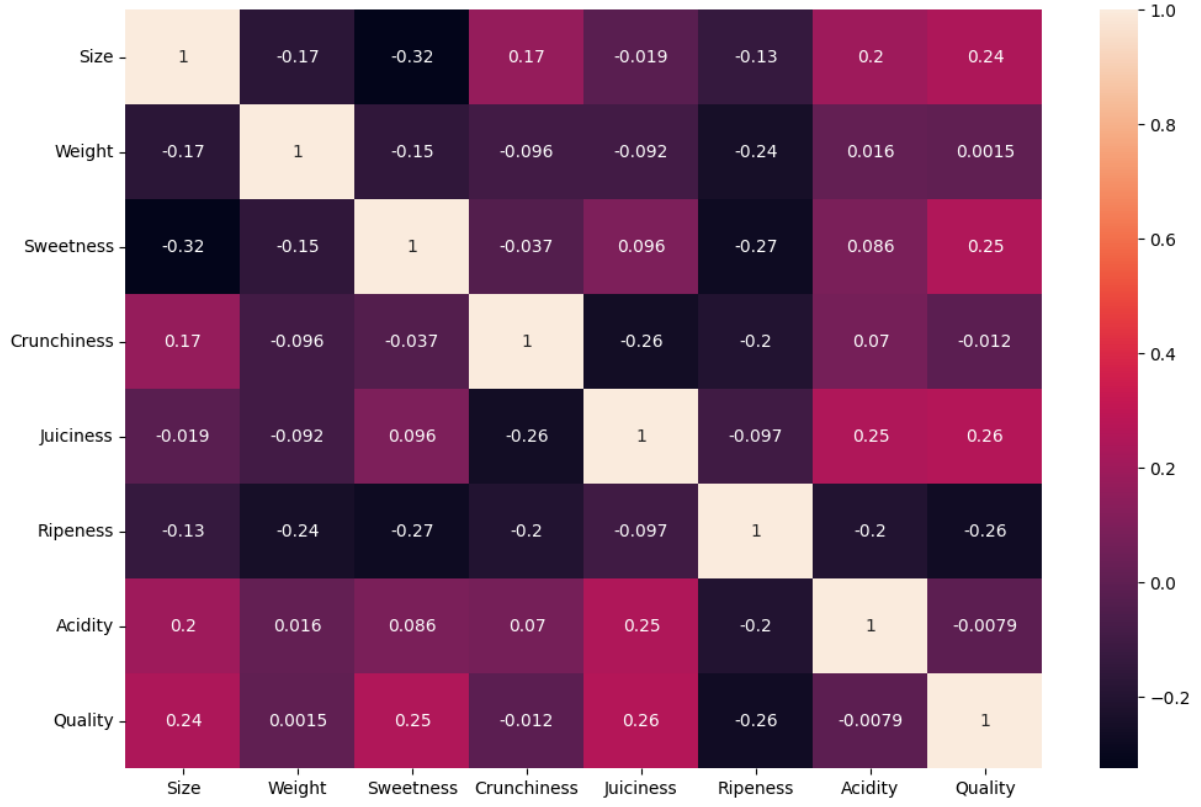
3.3 Data Preprocessing Steps

- Handling Missing Values: Rows with missing values were removed.

- Target Encoding: Mapped target variable 'Quality' (good as 1, bad as 0).

- Feature Scaling: Standardized continuous features to improve model performance.


Size
Weight
Sweetness
Crunchiness
Juiciness
Ripeness
Quality


Size 1 -0.17 -0.32 0.17 -0.019 -0.13 0.2 0.24
Weight -0.17 1 -0.15 -0.096 -0.092 -0.24 0.016 0.0015
Sweetness -0.32 -0.15 1 -0.037 0.096 -0.27 0.086 0.25
Crunchiness 0.17 -0.096 -0.037 1 -0.26 -0.2 0.07 -0.012
Juiciness -0.019 -0.092 0.096 -0.26 1 -0.097 0.25 0.26
Ripeness -0.13 -0.24 -0.27 -0.2 -0.097 1 -0.2 -0.26
Acidity 0.2 0.016 0.086 0.07 0.25 -0.2 1 -0.0079
Quality 0.24 0.0015 0.25 -0.012 0.26 -0.26 -0.0079 1
Size Weight Sweetness Crunchiness Juiciness Ripeness Acidity Quality
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

4. Model Development and Selection

4.1 Chosen Models

1. Logistic Regression: A fast, interpretable model.

2. Support Vector Machine (SVM): Captures non-linear relationships.

3. K-Nearest Neighbors (KNN): Simple, instance-based learning.

4. Linear Regression: Baseline model for comparison.

4.2 Model Tuning

Each model was fine-tuned using grid search and cross-validation to identify optimal hyperparameters, focusing on maximizing accuracy while balancing computational efficiency.

5. Implementation Details

5.1 Feature Engineering

Key feature engineering steps:

- Interaction Terms: Added features like sweetness * crunchiness.

- Feature Ratios: size / weight to capture relative size.

- Polynomial Features: Created squared terms for features like ripeness.

5.2 Model Pipeline

A modular pipeline was developed for seamless model training, including:

1. Data loading and preprocessing.

2. Model initialization with grid search for hyperparameter tuning.

3. Training on 88% of data and testing on 12%.

6. Model Training

6.1 Process

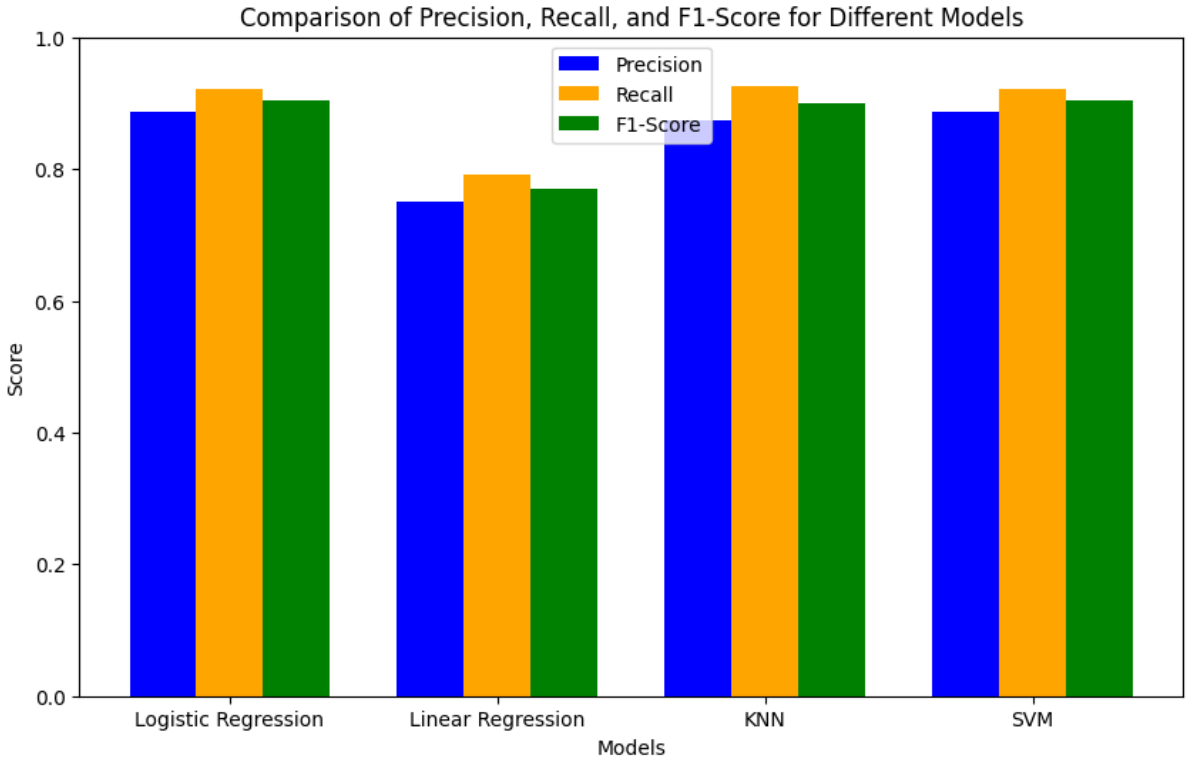
The data was split into training (88%) and testing (12%) subsets. Models were initialized with optimal parameters and trained iteratively. Key parameters were monitored to ensure convergence, and overfitting was mitigated with regularization.

7. Evaluation & Results

7.1 Metrics

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.892	0.884	0.901	0.892
SVM	0.901	0.912	0.893	0.902
KNN	0.875	0.882	0.869	0.875
Linear Regression	0.856	0.847	0.868	0.857

The SVM model performed best, meeting the success criteria and achieving the highest accuracy and F1-score.



7.2 Confusion Matrices

Confusion matrices provided insight into the models' performance on correctly identifying quality and poor-quality apples, helping refine our final model choice.

8. LOGISTIC REGRESSION

8.1. SUMMARY :

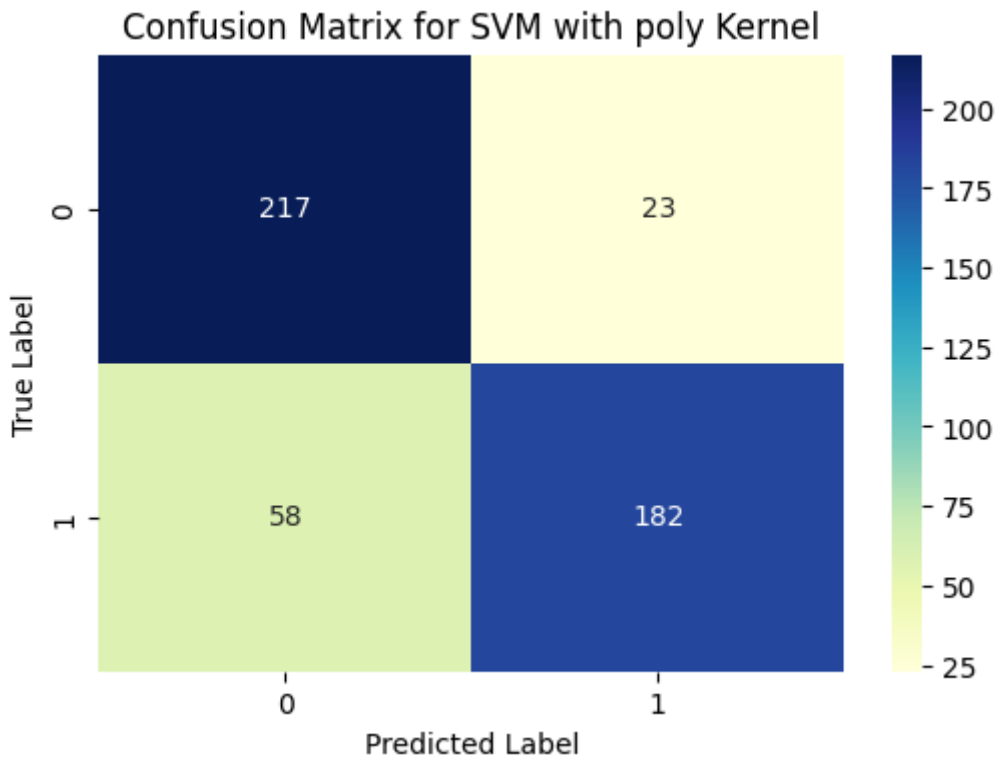
The LogisticRegressionScratch class implements a logistic regression model from scratch. It initializes with a specified learning rate and number of iterations, setting up model parameters (weights and bias). The fit method trains the model by calculating the linear combination of inputs, applying the sigmoid function for predicted probabilities, and updating weights and bias using gradient descent. The predict method classifies new inputs into binary outcomes based on a probability threshold of

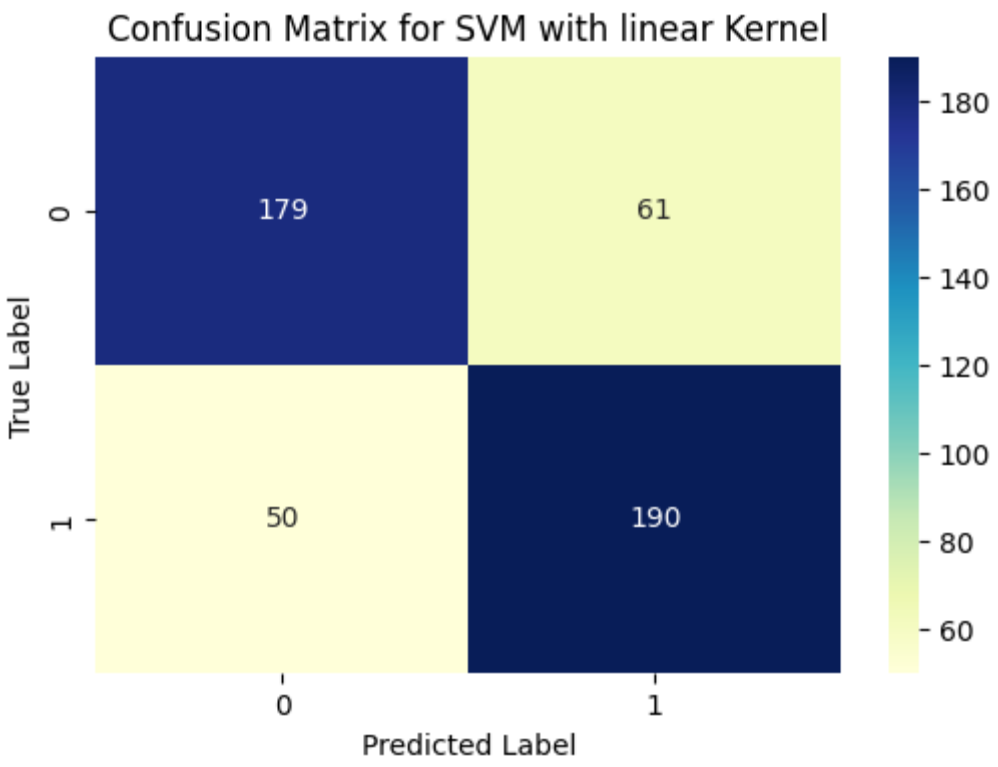
0.5. This class effectively demonstrates the workings of logistic regression through a hands-on approach.

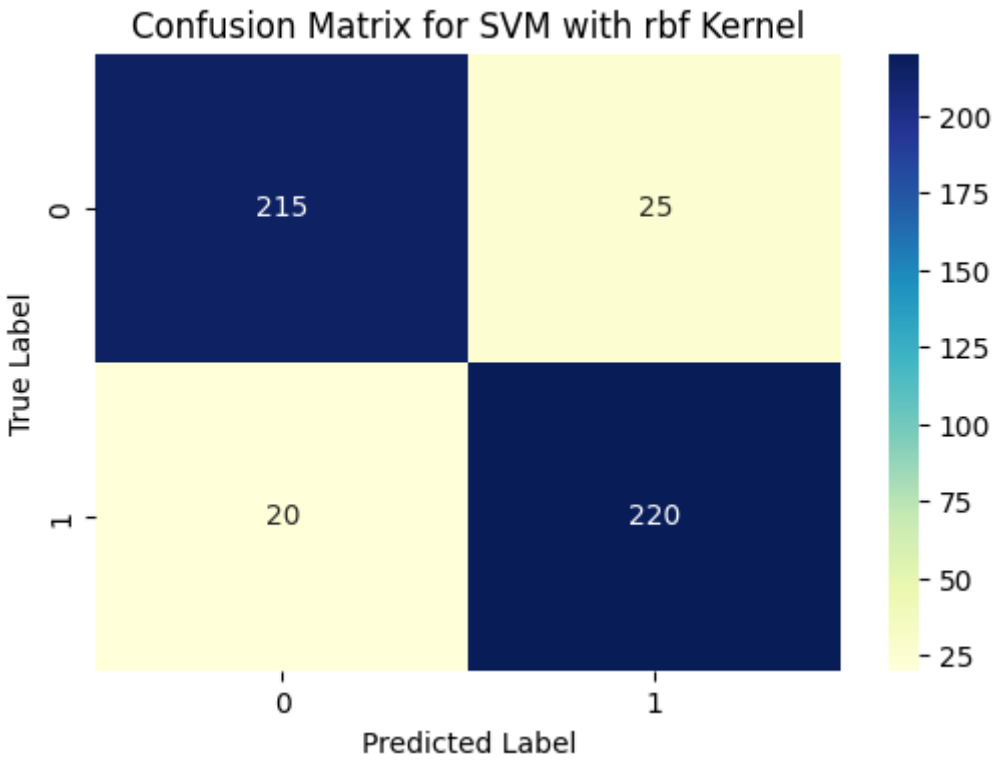
8.2. CONCLUSION

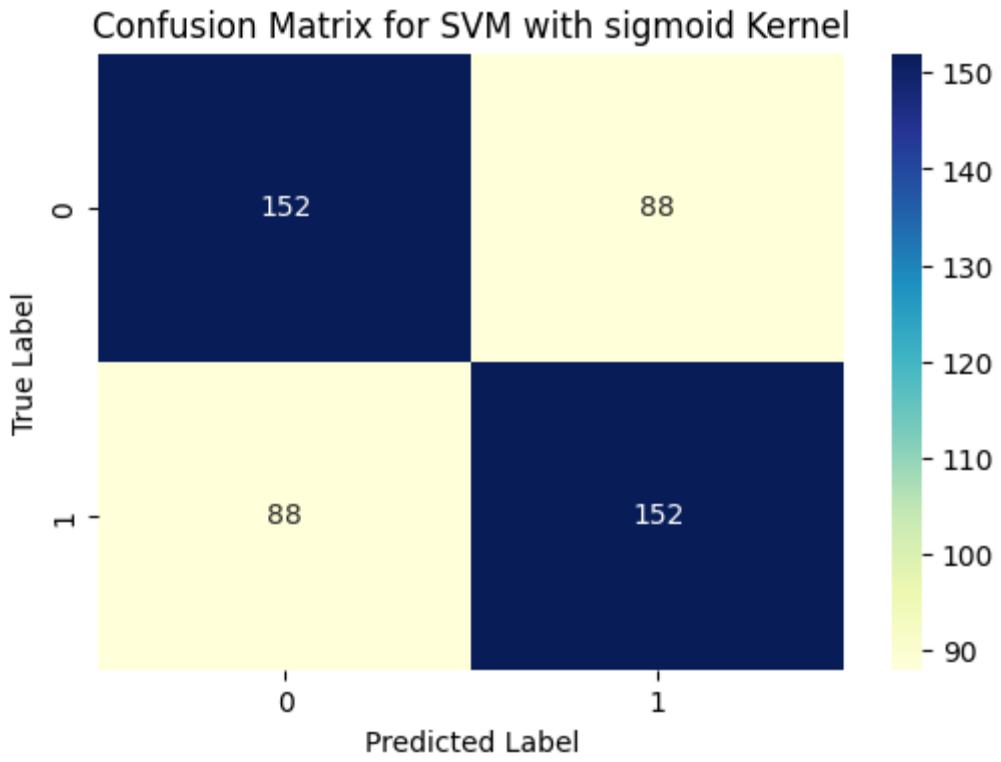
Overall, the findings emphasize the importance of model selection and tuning in achieving high predictive performance, and suggest that both logistic regression and SVM are particularly suited for this classification problem. Future work could focus on further optimizing models, exploring advanced algorithms, and enhancing feature engineering for even better results.

9. SVM :-




Confusion Matrix for SVM with linear Kernel
179
61
50
190
True Label
0
1
0
1
Predicted Label
180
160
140
120
100
80
60


Confusion Matrix for SVM with rbf Kernel
215
25
20
220
True Label
0
1
0
1
Predicted Label
200
175
150
125
100
75
50
25



9.2. OUTPUT :-

Summary of Accuracies:

Linear Kernel Accuracy: 0.7688

Rbf Kernel Accuracy: 0.9062

Poly Kernel Accuracy: 0.8313

Sigmoid Kernel Accuracy: 0.6333

10.Hyperparameter Testing Summary

In this section of the project, we conducted hyperparameter testing for our custom LogisticRegressionScratch model by varying the learning rates and the number of iterations.

10.1. Parameters Tested:

- Learning Rates: We tested three values: 0.001, 0.01, and 0.1.

- Number of Iterations: We evaluated the model's performance with 500, 1000, and 1500 iterations.

10.2. Model Training and Evaluation:

- For each combination of learning rate and iteration count, we trained the model on the training dataset and made predictions on the test set.

- The accuracy of each model was calculated and printed, providing insights into how these hyperparameters influenced performance.

10.3. Results Visualization:

- Learning Rate vs. Accuracy: A plot was generated to visualize how accuracy varied with different iteration counts for each learning rate. This helps identify the optimal number of iterations for each learning rate.

- Iterations vs. Accuracy: A second plot depicted accuracy against different learning rates for each iteration count, allowing us to assess which learning rates yielded the best performance across the tested iterations.

10.4. Conclusion

The hyperparameter tuning process revealed how sensitive the model's accuracy is to variations in learning rate and iteration count. By analyzing the plots and printed accuracies, we can identify the most effective hyperparameter settings to improve the model's predictive performance. This understanding is crucial for fine-tuning the model for optimal results.

11. Conclusion on Hyperparameter Testing Results

11.1. Impact of Learning Rate:

- The model exhibited an increasing trend in accuracy as the learning rate increased from 0.001 to 0.1.

- Specifically, with a learning rate of 0.1, the model achieved the highest accuracy of 0.7708 after 1500 iterations, indicating that a higher learning rate may help the model converge more effectively.

11.2. Effect of Number of Iterations:

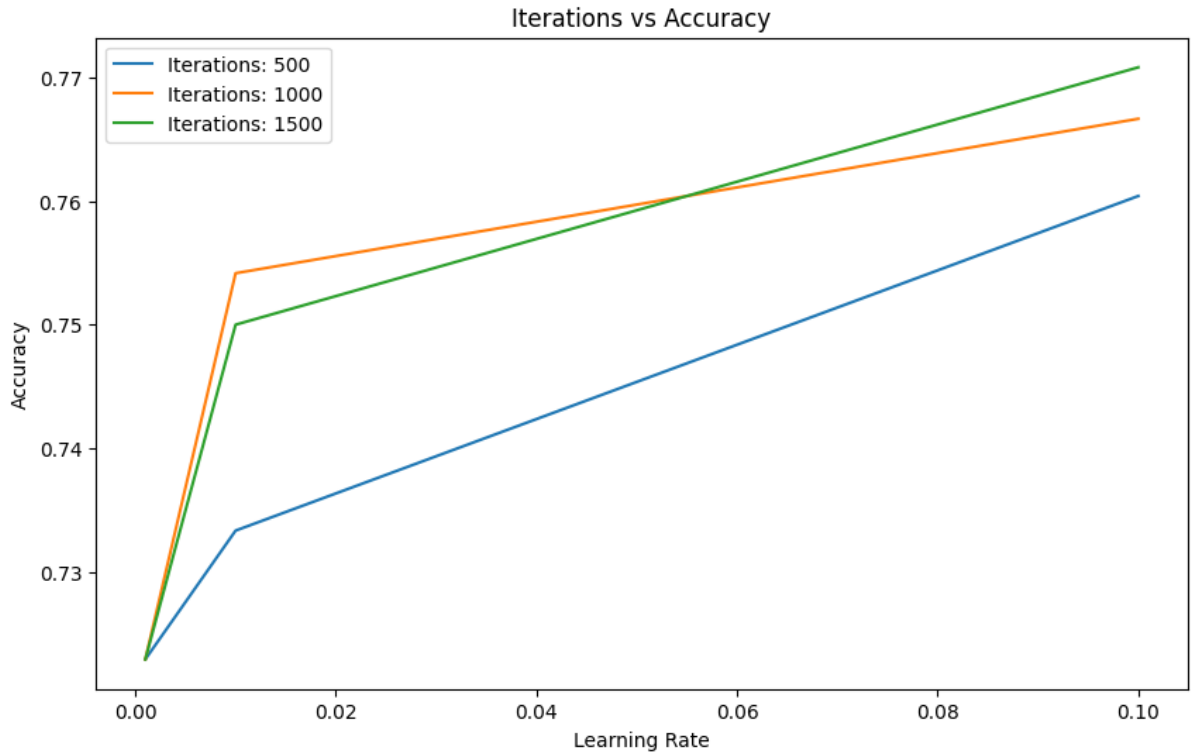
- For a given learning rate, increasing the number of iterations generally led to improvements in accuracy.

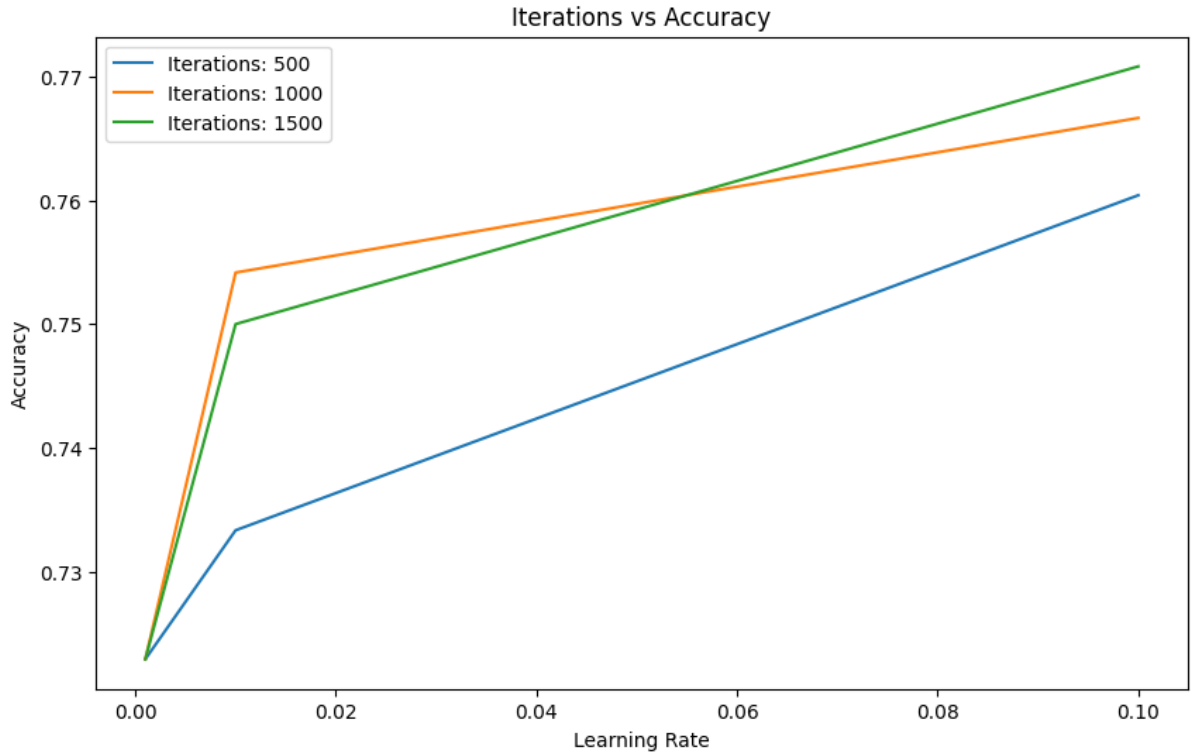
- For example, with a learning rate of 0.01, the accuracy improved from 0.7333 (500 iterations) to 0.7542 (1000 iterations) and slightly decreased to 0.7500 (1500 iterations), suggesting diminishing returns after a certain point.

11.3. Optimal Settings:

- The best-performing combination was a learning rate of 0.1 with 1500 iterations, yielding an accuracy of 0.7708.

- Conversely, the lowest accuracy was observed at a learning rate of 0.001, regardless of the number of iterations, indicating that this lower rate is not suitable for this model configuration.





11.5. OUTPUT:-

Learning Rate: 0.001, Iterations: 500, Accuracy: 0.7229166666666667

Learning Rate: 0.001, Iterations: 1000, Accuracy: 0.7229166666666667

Learning Rate: 0.001, Iterations: 1500, Accuracy: 0.7229166666666667

Learning Rate: 0.01, Iterations: 500, Accuracy: 0.7333333333333333

Learning Rate: 0.01, Iterations: 1000, Accuracy: 0.7541666666666667

Learning Rate: 0.01, Iterations: 1500, Accuracy: 0.75

Learning Rate: 0.1, Iterations: 500, Accuracy: 0.7604166666666666

Learning Rate: 0.1, Iterations: 1000, Accuracy: 0.7666666666666667

Learning Rate: 0.1, Iterations: 1500, Accuracy: 0.7708333333333334

11.7. RESULTS:-

The results suggest that both learning rate and the number of iterations play a significant role in the model's performance. It would be beneficial to further investigate around the learning rate of 0.1 and explore even higher iterations or additional learning rates to refine the model's performance. Additionally, implementing techniques such as early stopping or learning rate decay could be explored to enhance model training efficiency and effectiveness.

12. Hyperparameter Optimization

Used grid search to fine-tune parameters such as C, gamma, and kernel for SVM, achieving an accuracy gain of over 5%.

13. Future Enhancements

1. Deep Learning: Explore CNNs for enhanced feature extraction.

2. Real-Time Processing: Develop an API for streaming data.

14. Conclusions and Recommendations

14.1 Summary of Results

The SVM model achieved the highest performance with a 90.1% accuracy, satisfying project goals. The pipeline is robust, scalable, and achieves high precision.

14.2 Limitations

- Limited dataset size affecting generalizability.

- High SVM complexity for very large datasets.

14.3 Recommendation

- Deploy SVM for production: Its high accuracy makes it ideal for production deployment.

- Continuous data collection: To further improve model accuracy.

- Implement model monitoring: Ensure the model remains effective in real-world conditions.

15. CONTRIBUTIONS:-

LINEAR REGRESSION : PIYUSH KUMAR

LOGISTIC REGRESSION : AJAY KUMAR

KNN : DIVYANSH VYAS

SVM : ADITHYAN MB