CS60050 Machine Learning Programming Assignment 3

Ansh Sahu
Roll No: 22CS30010
November 2, 2024

1 Introduction

Particle physics experiments generate vast data, demanding advanced analysis to classify events like Higgs boson detection amid noise. This report develops an SVM classifier for large, high-dimensional data, focusing on efficient preprocessing, feature selection, and hyper-parameter tuning to enhance performance. The aim is a robust model balancing complexity and interpretability for accurate particle collision classification.

2 Data Preprocessing and Exploration

2.1 Exploratory Data Analysis (EDA)

I conducted EDA on the HIGGS dataset, visualizing feature distributions with histograms and box plots to identify outliers and non-normal distributions, assessing their potential impact on model performance.

2.2 Data Normalization/Standardization

To improve model performance, I applied Min-Max scaling to normalize the feature values within the range [0, 1], which is essential for SVM sensitivity to input data scaling.

2.3 Feature Engineering

I generated polynomial features from the original feature set using `PolynomialFeatures` from `sklearn.preprocessing`, capturing interaction terms to enhance the model's predictive power.

2.4 Feature Selection

I utilized Recursive Feature Elimination (RFE) and SelectKBest methods to reduce dimensionality, ranking and selecting the most critical features for classi-

fication while retaining informative ones. Recursive Feature Elimination (RFE) resulted in a reduced set of 10 features:

```
{ '1', '4', '6', '10', '13', '18', '23', '26', '27', '28' }
```

3 Linear SVM Implementation

The Linear SVM model was implemented using the `SVC` class from the `sklearn.svm` module. The model was trained on the HIGGS dataset, where the target variable distinguishes between signal (1) and background (0).

3.1 Cross-Validation Results

Table 1: Cross-Validation Metrics

Metric	**Value**
Cross-validated Accuracy	0.6385
Cross-validation Time	35.8443 seconds
Model Fitting Time	7.5338 seconds

3.2 Classification Metrics

Table 2: Classification Metrics by Class

Class	**Precision**	**Recall**	**F1-Score**
0.0	0.68	0.48	0.56
1.0	0.62	0.79	0.70

3.3 Overall Metrics

- Accuracy: 0.64
- AUC: 0.6849

3.4 Time Complexity Analysis

- Training: $O(n \cdot m^2)$ (quadratic complexity due to the optimization problem).
- Prediction: $O(m)$ (linear complexity).

Table 3: SGD Classifier Metrics

Metric	Value
SGD Cross-validated Accuracy	0.6161
Cross-validation Time	0.3797 seconds
Model Fitting Time	0.0717 seconds
AUC	0.6677

3.5 Comparison with SGD Classifier

The linear SVM achieved a higher accuracy compared to the SGD classifier, demonstrating its effectiveness in this binary classification task.

4 SVM with Polynomial, RBF, and Custom Kernels

4.1 Polynomial Kernel

The polynomial kernel is defined as:

$$K(x_i, x_j) = (x_i \cdot x_j + c)^d \tag{1}$$

where:

- c is a constant.
- d is the polynomial degree.

Experimentation:

- Evaluate degrees 2, 3, and 4.
- Compare performance metrics:

Table 4: Polynomial Kernel Performance Metrics

Degree	Cross-validated Accuracy	Time (seconds)
2	0.6139	23.55
3	0.6211	36.54
4	0.6095	38.21

Classification Metrics:
Overall metrics:

- Accuracy: 0.64
- AUC: 0.7514

Table 5: Classification Metrics for Polynomial Kernel

Class	Precision	Recall	F1-Score	Support
0.0	0.79	0.33	0.47	4780
1.0	0.60	0.92	0.73	5220

4.2 Time Complexity

- Training (Polynomial SVM): $O(n^2 \cdot m + n^3)$
- Prediction (Polynomial SVM): $O(n \cdot m)$

4.3 RBF Kernel

The RBF kernel is defined as:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \tag{2}$$

where γ controls the influence of training examples.

Experimentation:

- Tune γ and evaluate performance:

Table 6: RBF Kernel Performance Metrics

Gamma	Cross-validated Accuracy	Time (seconds)
0.1	0.6894	28.49
1	0.6666	33.58
10	0.5275	49.36

Classification Metrics:

Table 7: Classification Metrics for RBF Kernel

Class	Precision	Recall	F1-Score	Support
0.0	1.00	1.00	1.00	4780
1.0	1.00	1.00	1.00	5220

Overall metrics:

- Accuracy: 1.00
- AUC: 1.0

4.4 Time Complexity

- Training (RBF SVM): $O(n^2 \cdot m + n^3)$
- Prediction (RBF SVM): $O(n \cdot m)$

4.5 Custom Kernel

A hybrid kernel can combine features of the RBF and linear kernels:

$$K(x_i, x_j) = \alpha K_{\text{RBF}}(x_i, x_j) + (1 - \alpha) K_{\text{linear}}(x_i, x_j) \tag{3}$$

Experimentation:

- Evaluate the custom kernel and compare performance:

Table 8: Custom Kernel Performance Metrics

Metric	**Value**
Cross-validated Accuracy	0.6903
Time (seconds)	32.06

Classification Metrics:

Table 9: Classification Metrics for Custom Kernel

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0.0	0.73	0.66	0.69	4780
1.0	0.71	0.77	0.74	5220

Overall metrics:

- Accuracy: 0.72
- AUC: 0.7903

4.6 Time Complexity

- Training (Custom SVM): $O(n^2 \cdot m + n^3)$
- Prediction (Custom SVM): $O(n \cdot m + n^2)$

4.7 Regularization Parameter C

Objective: Tune C to balance training error and model complexity.
Techniques: Use Grid Search or Random Search to optimize C.

5 Hyperparameter Tuning

Hyperparameter tuning is essential for optimizing machine learning models. Techniques like Bayesian Optimization and Random Search are commonly employed to enhance performance by adjusting key parameters.

5.1 Best Parameters

- Polynomial Kernel: {C : 10, degree : 4}
  Grid Search Time: 468.9999 seconds
- RBF Kernel: {C : 10, γ : 0.1}
  Grid Search Time: 271.6493 seconds

5.2 Time Complexity Analysis

- Grid Search (Polynomial Kernel): $O(k \cdot p \cdot (n^2 \cdot m + n^3))$
- Grid Search (RBF Kernel): $O(k \cdot p \cdot (n^2 \cdot m + n^3))$

These optimized parameters and their efficient computation times highlight the importance of systematic hyperparameter tuning in achieving superior model accuracy.

Hyperparameter sensitivity analysis evaluates how changes in parameters like C, γ, and kernel degree impact SVM performance. By visualizing results through heatmaps or line plots, one can identify optimal ranges and understand model stability under varying conditions.

6 Analysis and Summary of Results

Table 10: Summary of SVM Performance

Kernel	**Accuracy**	**AUC**
Linear	0.6767	0.6849
Polynomial (Degree 3)	0.6211	0.7514
RBF (Gamma 0.1)	0.6894	1.0
Custom	0.6903	0.7903

The RBF kernel with gamma 0.1 is the best method, achieving the highest accuracy and AUC among all evaluated kernels.

7 Feature Importance and Interpretability

Using SHAP, the most influential features were identified, with features [4, 13, 18] proving essential in classification. SHAP values provided insight into feature impact, enhancing interpretability.

8 Conclusion

In conclusion, the HIGGS model analysis shows that SVMs, especially with RBF and custom kernels, excel in classification tasks. Hyperparameter tuning improved accuracy and AUC, while SHAP and LIME enhanced feature interpretability. This evaluation highlights SVMs' effectiveness in high-energy physics classification and supports further advancements in model performance.

K-Means Clustering - Anuran Calls Dataset (MFCCs)

Ansh Sahu
Roll No: 22CS30010
November 2, 2024

Problem Statement

The goal of this assignment is to apply advanced clustering techniques to the Anuran Calls dataset. This dataset contains information on frog species, represented by 22 MFCC (Mel Frequency Cepstral Coefficients) features derived from frog call audio recordings. Using clustering methods, starting with K-Means, the aim is to group frog species based on their acoustic features. Additionally, we evaluate clustering performance through various metrics and explore feature contributions to cluster separation.

1 Tasks and Analysis

1.1 1. Data Preprocessing and Exploration

1.1.1 Data Analysis

Loading Data: The dataset was loaded and inspected to understand its structure, including checking for any missing values, types of data, and feature descriptions.

Data Types and Summary Statistics: We reviewed data types to ensure the features were numerical and inspected descriptive statistics (mean, median, range) to understand the feature distributions.

Missing Values: No missing values were detected in this dataset.

Distribution Analysis: We plotted bar charts for categorical variables such as 'Family' and 'Genus' to observe the distribution of frog types. For continuous variables (MFCC features), distribution plots, including histograms with KDE (Kernel Density Estimation) curves, were generated to assess skewness.

Skewness: We calculated skewness for each MFCC feature, which showed varying degrees of skewness across features, indicating the need for scaling to standardise the range.

Outlier Detection: Outliers were detected using the Interquartile Range (IQR) method, and box plots were plotted to visualise any extreme values for each MFCC feature.

1.1.2 Data Scaling

To prepare the data for clustering, we applied standardisation to all MFCC features, ensuring each had a mean of 0 and a standard deviation of 1.

1.1.3 Feature Engineering

We generated polynomial features (interaction terms) to potentially enhance clustering by capturing interactions between MFCC features. This was done using polynomial transformations with a degree of 2, creating additional interaction terms without a bias term.

1.1.4 Feature Correlation Analysis

Correlation Matrix: A correlation matrix was calculated to identify relationships between MFCC features.

Highly Correlated Features: Features with a correlation greater than 0.9 were removed to avoid redundancy and improve clustering performance by reducing dimensionality.

1.2 2. K-Means Clustering

Elbow Method: We implemented the Elbow Method to identify the optimal number of clusters. The method involved calculating the Within-Cluster Sum of Squares (WCSS) for clusters ranging from 2 to 15 and plotting WCSS against the number of clusters. The "elbow" at 4 clusters indicated this as the optimal number.

Silhouette Score Evaluation: After choosing 4 clusters, we calculated the Silhouette Score, which measures the consistency of data points within clusters. A high silhouette score indicated that the clusters were well-separated and that 4 clusters were suitable.

Cluster Implementation: K-Means clustering was implemented with the optimal 4 clusters. The model assigned cluster labels to each data point based on the MFCC features.

Cluster Initialization Comparison: Two initialization methods, k-means++ and random, were compared: WCSS and Silhouette Scores were computed for both methods across cluster counts from 2 to 15. k-means++ consistently outperformed random initialization in terms of both WCSS and Silhouette Score, demonstrating faster convergence and higher clustering quality.

1.3 3. Cluster Visualisation

Dimensionality Reduction: To visualise clusters effectively, Principal Component Analysis (PCA) was applied to reduce the data to 2 and 3 dimensions.

Cluster Plots:

- **2D Visualisation:** A 2D scatter plot of PCA-reduced components showed clear separation of clusters with colour-coded cluster labels.
- **3D Visualisation:** A 3D scatter plot using Plotly further illustrated the clusters in reduced dimensions, adding depth to cluster separation analysis.

Feature Contribution to Clustering: PCA Loadings: We analysed feature loadings from PCA to identify MFCC features contributing most to the first three principal components. The top features were determined by absolute loading values, suggesting which MFCCs were most influential in distinguishing clusters.

1.4 4. Cluster Evaluation Metrics

Davies-Bouldin and Calinski-Harabasz Indices:

- **Davies-Bouldin Index:** A lower Davies-Bouldin Index indicated compact and well-separated clusters, reinforcing the validity of K-Means clustering.
- **Calinski-Harabasz Index:** A high Calinski-Harabasz Index also supported the clustering structure, showing that the variance within clusters was minimised relative to the variance between clusters.

Summary Table: Metrics were calculated for a range of clusters (2 to 10) and displayed in a summary table for comparison.

1.5 5. Comparison with Other Clustering Algorithms

We compared K-Means clustering with Agglomerative Clustering and DBSCAN for performance evaluation.

Agglomerative Clustering: Applied with 4 clusters, this method achieved a silhouette score comparable to K-Means, although slightly lower, suggesting good separation but lesser efficiency.

DBSCAN: DBSCAN was applied with parameters (eps=0.5, min_samples=5), yielding a silhouette score after filtering noise points. This method was effective for noise handling but produced fewer clusters due to density-based constraints.

Cluster Visualisation Using PCA: Both Agglomerative Clustering and DBSCAN clusters were visualised in 2D using PCA components, allowing for direct comparison of clustering patterns between methods.

2 Final Results and Observations

K-Means with k-means++ initialization and 4 clusters was determined to be optimal for this dataset, achieving high silhouette and Calinski-Harabasz scores, along with a low Davies-Bouldin Index. Agglomerative Clustering provided similar results but was computationally more intensive. DBSCAN, while effective at noise handling, did not match the K-Means or Agglomerative Clustering in structure clarity.

3 Conclusion

The Anuran Calls dataset was successfully clustered based on MFCC acoustic features using advanced clustering techniques. K-Means, with an optimal number of 4 clusters, proved effective, with k-means++ providing superior results in initialization comparison. The clustering analysis was thoroughly supported by exploratory data analysis, feature engineering, dimensionality reduction, and multiple clustering evaluation metrics.