جامعة طيبة
TAIBAH UNIVERSITY

AI331 Machine Learning Project

Name	ID number	section
Shatha Ali Alharbi	4453086	F3
Shahad al rshidi	4453301	F3
Layal Naif Alrehaili	4453584	F3
Mayar Ayiedh Almatrafi	4453328	F3
Riman MohammedAlsaedi	4453048	F3

Task	student
Feature selection using ANOVA F-test	Shahad
Handling Zero Variance	Mayar -Layal
SVM implementation	Shahad -Riman
SVM theory	Riman
Logistic Regression implementation	Mayar -Layal
Logistic Regression theory	Mayar -Layal
Hierarchical Clustering implementation	Shatha
Hierarchical Clustering theory	Shatha

Abstract

Handwritten digit recognition is the ability for computers to recognize human handwritten digits from different sources such as images and papers. Handwritten digit recognition is a core problem in pattern recognition and machine learning, with significant application in document preprocessing, automated data entry and postal services.

This project focuses on binary classification task using Gisette dataset, which consist of 5000 numerical features representing the digits 4 and 9. The objective is to distinguish between these two digits, which are visually similar and pose a challenge for classification algorithms.

The motivation of this project is to apply supervised machine learning classifiers such as Linear SVM and Logistic Regression, assess their performance on complex, high dimensional data and to evaluate the impact of feature selection on the predictive performance and model interpretability. Alongside this we also employ hierarchical clustering for unsupervised exploratory analysis of the intrinsic structure of the data.

The results indicated that incorporating feature selection Improved classification accuracy by 1% which illustrates its effectiveness in reducing the dimensionality while enhancing model performance.

Dataset and Domain Description

In this project, we used the GISETTE dataset from the UCI Machine Learning Repository. It is a handwritten digit dataset where the goal is to classify between the digits 4 and 9, which look very similar. So, this is a binary classification problem.

The full dataset has 13,500 samples and 5,000 numeric features. These features are not raw pixel values, but engineered values extracted from 28×28 digit images. Some features are useful, but many were intentionally added as noisy features. In our project, we used the training split only, which contains 6000 samples, and the classes are balanced (around 3000 for each digit).

The data consists of two text files gisette_train.data which are the features and gisette_train.labels which are the targets. While loading the feature file, we noticed an extra last column that has all missing values because of a space at the end of each row. Since this column was completely NaN, we decided to remove it.

After that, we encoded the labels from -1, +1 into 0, 1, where 1 represents the digit 4 and 0 represents the digit 9. Then checked for missing values and duplicate rows, and no problems were found other than the removed NaN column.

We also checked for constant features which are features that have the same value for all samples. Leading us to find 45 constant features, then we removed them because they don't help the model distinguish between classes. Which helped us reduce the features from 5000 to 4955.

Finally, we applied StandardScaler (z-score standardization) because the models we used are sensitive to differences in feature scale. This step transforms each feature to have approximately zero mean and unit variance, which makes training more stable, helps the optimizer converge faster, and can improve with the model performance. After preprocessing, the data shape is (6000, 4955) and it is ready for feature selection and training.

Therefore, why is this dataset could be useful because Handwritten digit classification is a common machine learning task with real applications such as recognizing postal codes and processing bank checks and reading digitized forms. The GISETTE dataset is useful because it focus on distinguishing between two visually similar digits (4 and 9) which makes the classification harder, the data it intentionally includes a lot of noisy and irrelevant features, which makes it a good dataset to learn the importance of preprocessing and feature selection with improving model performance.

Theoretical Background

- ## Support Machines Vectors (SVM)

Support vectors machines (svm) are supervised learning algorithms, and its designed to construct a discriminant decision boundary (hyperplane) to separate different classes in a feature space with the maximum possible margin, the margin is defined as the distance between support vectors (nearest data points to the hyperplane) and the hyperplane itself, that is represented by the following equation, f(x)= wx+b , hence the goal is the maximize the margin ensuring all datapoints are correctly classified, its shown the optimization function

$$\text{Minimize } \frac{1}{2} ||w||^2$$

but since real-world data are can rarely be perfectly separable, SVM allow misclassification to some extent, which introduces the idea of a soft margin by incorporating slack variables modifying the optimization formula to be as following

$$\text{Minimize } \frac{1}{2} ||w||^2 + C \sum \xi_i.$$

- ## Logistic Regression

Logistic Regression is a binary classification model. It's called "regression" because it computes a numeric value first, but the goal is classification predicting whether the sample belongs to class 1 or class 0. The main idea is to build a linear score, convert it to probability then learn the weights by minimizing a loss

First, the model computes a linear score by combining the input features into one value:

$$h(x) = \theta^T x^{(i)}$$
$$z = h(x)$$

z is simply a weighted sum of the features. If z is large and positive, the model tends to lean toward class 1, and if z is negative, it leans toward class 0

Second, since z can take any real value, Logistic Regression converts it into a probability using the sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}}$$

This makes the prediction interpretable as a probability $g(\theta^T x) = P(y = 1 \mid x)$,

Which is the core idea of Logistic Regression instead of outputting an unrestricted score, it outputs a probability between 0 and 1.

Third, the model measures how wrong the predicted probabilities are using the logistic (cross-entropy) cost function:

$$j(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(g(z^{(i)})) + (1-y^{(i)})\log(1-g(z^{(i)}))]$$

This loss strongly penalizes confident incorrect predictions, which pushes the model to adjust its probabilities toward the true labels.

To minimize this cost, Logistic Regression computes the gradient with respect to each parameter:

$$\frac{\partial}{\partial\theta_j}j(\theta) = \frac{1}{m}\sum_{i=1}^{m}(g(z^{(i)}) - y^{(i)})x_j^{(i)}$$

this term (g(z) − y) represents the prediction error, and multiplying by x shows how many features j contributed to that error. This is what makes learning "data-driven": features that contribute more to the error receive larger corrections.

Finally, the parameters are updated using a gradient descent step:

$$\theta_j := \theta_j - \gamma\frac{1}{m}\sum_{i=1}^{m}(g(z^{(i)}) - y^{(i)})x_j^{(i)}$$

Here y is the learning rate that controls how big each update is. Repeating this update gradually reduces the loss, meaning the predicted probabilities become closer to the true labels.

The decision rule in Logistic Regression comes from the probability threshold.

Since $g(z) \geq 0.5$ happens exactly when $z \geq 0$ the decision boundary is the linear surface This means Logistic Regression creates a linear separating boundary in feature space, while still producing probabilistic outputs, which makes it useful for binary classification problems.

- Hierarchical Clustering

Hierarchical Clustering is an unsupervised learning technique used to group similar data points into clusters based on a predefined distance or similarity measure. Hierarchical Clustering does not require specifying the number of clusters in advance. Instead, it constructs a tree-like structure called a dendrogram, which represents the hierarchical relationships between data The clustering process .points and clusters at different levels of similarity relies on a distance metric defined between instances. In this work, Euclidean distance was used to measure the similarity between data points.

$$d(p,q) = \sqrt{\sum_{i=1}(q_i - p_i)^2}$$

Hierarchical Clustering can be divided into two main type: Agglomerative Clustering (Bottom-up approach) and Divisive Clustering (Top-down approach).In this project, Agglomerative Clustering was applied. This approach starts with N clusters, where each cluster initially contains a single data instance. At each step, the two most similar clusters are merged together based on the chosen distance metric. This process continues iteratively until all data points are merged into a single cluster.

$$\Delta(A,B) = \frac{n_A n_B}{n_A + n_B} ||\vec{m}_A - \vec{m}_B||^2$$

Hierarchical Clustering is well-suited for clustering problems, as it does not rely on labeled data and provides an interpretable hierarchical structure that helps analyze relationships between data point.

Methodology

The dataset required minimal cleaning as the inspection of the data showed no missing values and no duplicates hence no imputations or data cleaning interventions were necessary. A simple encoding step was applied to the label where the original classes {+1, -1} were mapped to 0 and 1 respectively, this was done for simplicity following the standard in binary classification problems.

Before splitting the data, we removed zero variance (constant features), These features contained only a single value across all samples and therefore contributed no discriminative information. Eliminating such features help reduce the dimensionality and prevents them from diluting the effect of informative features.

We employ supervised methods including Linear (SVM) and Logistic regression, as well as unsupervised hierarchical clustering. Since these methods rely on distance-based optimization, they are highly sensitive to the difference in feature magnitude, taking this into account we applied Standardization using the StandardScaler(), transforming each feature to have zero mean and unit variance.

Given that the data is high dimensional and contains many irrelevant and noisy features we applied supervised dimensionality reduction. we selected a supervised technique because the main task of this project is classification, and supervised methods exploit the relationship between features and class labels, allowing them to retain the most discriminative information. In contrast unsupervised approaches ignore the labels and may preserve variance that is not relevant for classification.

Thus, we applied SelectKBest() with an ANOVA F-test which is a supervised dimensionality reduction method that evaluates each feature independently by measuring how well it separates the two classes, selecting the features with the strongest discriminative power. And since the optimal number of the features to keep is not known beforehand, we experimented with multiple values of K and monitored the validation error and selected the value of K that yielded the best validation performance.

A linear support vector classifier (LinearSVC) from scikit-learn was implemented due to its suitability for high dimensional data and computational efficiency, the linearSVC was used with its default configuration from scikit learn. the regularization parameter C was treated as the primary hyperparameter to tune .GridsearchCV was used to search over a range of C values, with cross -validation automatically performed inside the grid search. GridsearchCV fits the model during this process, therefore the returned best_estimitor_ is already trained on the full training set using the optimal C value and no additional manual training is required. This fitted estimator serves as the final model for evaluation.

Logistic Regression was implemented as another supervised classification model as part of the overall methodology, due to its simplicity and strong performance on high-dimensional binary classification problems. A baseline Logistic Regression model was first trained to obtain an initial reference performance. followed by Logistic Regression that was added into the same preprocessing and feature selection pipeline used for the other supervised models. The main hyperparameters of the model, which are the number of selected features k and the regularization parameter C, were optimized using GridSearchCV with Stratified K-Fold cross-validation to ensure balanced class distributions across folds. L2 regularization with the lbfgs solver was used throughout. The resulting best estimator was then used as the final Logistic Regression model and evaluated on the testing set using accuracy, F1-score, and the confusion matrix.

For the unsupervised models, the Agglomerative Hierarchical Clustering with ward linkage was used to estimate the appropriate number of clusters, as Ward linkage minimizes within-cluster variance and is compatible with the Euclidean distance used after feature scaling. Different cluster configurations were evaluated, and the Silhouette score was employed as an internal metric to select the most suitable number of clusters based on cluster cohesion and separation. Additionally, Adjusted Rand Score (ARS) was used as an external evaluation metric to assess the agreement between the resulting clusters and the true class labels, which help to validate the clustering quality.

For evaluating the supervised models, the data was split into training and testing sets using 80/20 ratio. And the classifiers were evaluated on the testing set using accuracy since the classes distribution is balanced and a confusion matrix also was used to analyze the distribution of true positive, true negative, false positive and false negative.

Also, a custom evaluation function was implemented to compute additional SVM-specific statistics including margin width, weight vector magnitude, mean distance from hyperplane, and standard deviation of distances. These values were compared before and after feature selection to measure the impact on model performance and separation strength.

Results and Evaluation

To balance generalization with accuracy in our implementation of linear support vector machines (svm), we precisely tuned its key parameter c -regularization parameter to penalize misclassification, over the range [0.001,0.01, 0.1,1], using cross-validation, the model performed its best when c =0.001, relying on the modified optimization formula shown above.

After training our Linear support vector classifier on the gisette dataset, the model produced a weight vector and a bias term b= - 0.0026 forming the hyperplane and with 98% test accuracy.

In addition to standard performance metrics, a custom evaluation function was implemented to analyze the geometric behavior of the linearSVM before and after feature selection.

This function computed key statistics of the model including the weight, geometric margin, and the distribution of sample distances from the hyperplane.

The results indicated that after feature selection, the weight norm increased while the geometric margin decreased slightly, as expected given their opposite relationship. Feature selection selects the features that capture the most information and the strongest separation between classes thus, more weight is assigned to the selected features which explains why the weight increased. More importantly, the mean distance from the hyperplane moved closer to zero and the standard deviation decreased indicating reduced variance in how the samples are spread around the hyperplane. these results show that the feature selection improved feature relevance, reduced noise, and led to a better generalization where the test accuracy increased from by 1%

for Logistic Regression model we first trained a baseline Logistic Regression model with StandardScaler (L2 penalty, LBFGS solver). It achieved 100% training accuracy, with 97.75% test accuracy and an F1-score of 0.9776, which shows a classic overfitting problem since the training score is perfect. To improve generalization, we built a pipeline that applies VarianceThreshold to remove constant features inside each CV fold, then SelectKBest using ANOVA F-test, followed by scaling and Logistic Regression. then We tuned C which is the inverse strength of L2 regularization over (0.01, 0.1, 1) using cross validation, and the model performed best at C = 0.1 with mean validation score ≈ 0.974792 , which reflects a good trade off

between keeping weights controlled while still fitting the patterns in the data. Using the best estimator from GridSearchCV, the final model achieved 98.33% test accuracy and an F1-score of 0.9835, improving over the baseline, and it also achieved the best overall performance compared to the other algorithms we tested.

And to understand performance more than just accuracy, we generated a confusion matrix. The final model produced 585 true negatives and 595 true positives, with only 15 false positives and 5 false negatives. This indicates that the classifier performs consistently across both classes and is not biased toward one label, which matches our expectations since the dataset is balanced.

To estimate the appropriate clustering structure, the Silhouette Score was computed for different clustering configurations. The results showed relatively low Silhouette values across all tested configurations, indicating limited separation between clusters. The highest Silhouette value was obtained for the smallest clustering configuration, while the score gradually decreased and became negative as the number of clusters increased. Negative Silhouette values suggest overlapping clusters and weak cluster cohesion.

```
k=2, Silhouette=0.005980483642947196
k=3, Silhouette=0.005652015819429669
k=4, Silhouette=0.003727634497623908
k=5, Silhouette=-0.00873869214280132
k=6, Silhouette=-0.010381338734938022
k=7, Silhouette=-0.0099111157099618
k=8, Silhouette=-0.020826526929805078
k=9, Silhouette=-0.019903093512540986
```

The Adjusted Rand Score (ARS) was used as an external evaluation metric to compare the clustering results with the true class labels. The obtained ARS value of 0.056 indicates a low level of agreement between the predicted clusters and the actual class labels, confirming that the clustering structure does not closely correspond to the true class distribution.

```
Adjusted Rand Score: 0.0557079979934911 66
```

Overall, the results suggest that while Hierarchical Clustering is capable of identifying some underlying structure in the data, the clustering quality is limited due

to overlapping feature distributions. Furthermore, Hierarchical Clustering is computationally expensive for large datasets, as it requires storing and processing the full distance matrix . We applied several clustering techniques, including hierarchical clustering and other methods, on the handwritten digits dataset. However, all approaches failed to produce a clear and meaningful grouping of the data.
The most likely reason is that the dataset does not exhibit a strong natural clustering structure, making it inherently difficult for any clustering algorithm to separate the classes accurately. Moreover, some clustering techniques rely on specific assumptions about cluster shapes or distances between points, which may not hold for handwritten digits with high feature overlap.
In other words, the failure of these techniques reflects the intrinsic nature of the data, rather than a limitation of the algorithms themselves.

Conclusion

This project aimed to investigate the application of linear SVM, Logistic Regression, and hierarchical clustering, within a Consolidated pipeline on the gisette high denominational dataset. Through incorporated techniques like feature selection, such as ANOVA F-test.

We demonstrated how the SVM classifier achieved a strong generalization score, while logistic regression provides more statistical interpretations.
And the structural groupings in the data were shown by the implementation of hierarchical clustering.

While each method offers distinct strengths and limitations, but it sets a solid groundwork for future explorations like extending the pipeline to include kernel-based methods to increase robustness, or integrate ensemble methods

Beyond the technical implementations and outcomes, this project has enhanced our knowledge with the importance of designing modular pipelines that enable fair comparisons across models, we learned how challenging it often is to balance generalization and interpretability with a models accuracy and performance and how easy it's to fall into the risk of overfitting a relatively simple problem, but at broader level the project has strengthened our ability to effectively communicate findings, collaborate, and evaluate our choices.