Case Study 1: Classification of Fashion MNIST Data

Objective

The purpose of this assignment is to analyze and compare the performance of various classification algorithms on the Fashion MNIST dataset. The assignment is split into two parts:

- Binary Classification: Distinguish between two specific classes.
- Multi-Class Classification: Classify images into all 10 categories.

We will implement Logistic Regression, k-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Decision Trees for both tasks, experiment with hyperparameters, and compare their performance.

Introduction

The Fashion MNIST dataset is a collection of 70,000 grayscale images of size 28x28 pixels, categorized into 10 different fashion items. Each image represents a piece of clothing, such as T-shirts, trousers, dresses, and more. This dataset serves as a drop-in replacement for the classic MNIST dataset of handwritten digits, providing a more challenging benchmark for machine learning algorithms due to the increased complexity of fashion images.

Part 1: Binary Classification

In this part, we focus on distinguishing between two specific classes from the Fashion MNIST dataset:

- Class 0: T-shirt/top
- Class 1: Trouser

We will implement and evaluate Logistic Regression, k-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Decision Trees. For each algorithm, we'll experiment with at least five different hyperparameter combinations and evaluate their performance using accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix.

1. Dataset Preparation

1.1 Load the Fashion MNIST Dataset

The dataset consists of 60,000 training images and 10,000 testing images. Each image is represented by 784 pixel values (28x28 pixels) and a label indicating the class.

1.2 Visualizing the Data

To better understand the dataset, we visualized one image per class. This helped us to verify the classes and understand the visual differences between them.

1.3 Selecting Two Classes

For binary classification, we selected Class 0 (T-shirt/top) and Class 1 (Trouser). This choice was made because these classes have distinct visual features, making them suitable for binary classification.

- Binary Training Set: 12,000 images (6,000 per class)
- Binary Testing Set: 2,000 images (1,000 per class)

1.4 Feature Scaling

Feature scaling was performed using StandardScaler to standardize the pixel values. This step is crucial for algorithms sensitive to the scale of data, such as KNN and SVM.

2. Algorithms to Use

We implemented the following algorithms:

- Logistic Regression
- k-Nearest Neighbors (KNN)
- Support Vector Machines (SVM)
- Decision Trees

These algorithms were chosen because they are fundamental classification methods with varying approaches, allowing us to compare their performance on image data.

3. Hyperparameter Tuning

For each algorithm, we experimented with at least five different hyperparameter combinations.

3.1 Logistic Regression

3.1.1 Hyperparameter Combinations

We explored different values for the regularization parameter C, solvers, and maximum iterations. The combinations were:

1. C=0.01, solver='liblinear', max_iter=100
2. C=0.1, solver='liblinear', max_iter=100
3. C=1, solver='liblinear', max_iter=100
4. C=10, solver='liblinear', max_iter=100
5. C=1, solver='lbfgs', max_iter=200

3.1.2 Model Training and Evaluation

Each model was trained on the training set and evaluated on the testing set. We collected performance metrics including accuracy, precision, recall, F1-score, ROC-AUC, and the confusion matrix.

3.2 k-Nearest Neighbors

3.2.1 Hyperparameter Combinations
We varied the number of neighbors, weights, and distance metrics:

1. n_neighbors=3, weights='uniform', metric='euclidean'
2. n_neighbors=5, weights='uniform', metric='euclidean'
3. n_neighbors=7, weights='distance', metric='manhattan'
4. n_neighbors=9, weights='distance', metric='manhattan'
5. n_neighbors=11, weights='uniform', metric='minkowski'

3.2.2 Model Training and Evaluation
We trained and evaluated each KNN model, noting that ROC-AUC is not applicable without probability estimates.

3.3 Support Vector Machines

3.3.1 Hyperparameter Combinations
We experimented with different values of the regularization parameter C and maximum iterations:

1. C=0.1, max_iter=1000
2. C=1, max_iter=1000
3. C=10, max_iter=1000
4. C=1, max_iter=2000
5. C=1, max_iter=5000

3.3.2 Model Training and Evaluation
SVM models were trained using SVC with probability=True to enable ROC-AUC calculation.

3.4 Decision Trees

3.4.1 Hyperparameter Combinations
We varied the maximum depth, minimum samples per leaf, and the splitting criterion:

1. max_depth=5, min_samples_leaf=1, criterion='gini'
2. max_depth=10, min_samples_leaf=2, criterion='entropy'
3. max_depth=None, min_samples_leaf=1, criterion='gini'
4. max_depth=15, min_samples_leaf=4, criterion='entropy'
5. max_depth=3, min_samples_leaf=2, criterion='gini'

3.4.2 Model Training and Evaluation
Each Decision Tree model was trained and evaluated, with probability estimates used for ROC-AUC calculation.

4. Performance Evaluation

4.1 Logistic Regression Results

Summary Table: Logistic Regression (Binary Classification)

Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
C=0.01, solver='liblinear', max_iter=100	98.63	99.12	98.12	98.62	99.80
C=0.1, solver='liblinear', max_iter=100	98.73	99.04	98.42	98.73	99.79
C=1, solver='liblinear', max_iter=100	98.71	98.98	98.43	98.70	99.79
C=10, solver='liblinear', max_iter=100	98.64	98.91	98.37	98.64	99.77
C=1, solver='lbfgs', max_iter=200	98.70	98.96	98.43	98.70	99.77

Analysis:

- The highest accuracy achieved was 98.73% with C=0.1.
- Lower values of C (stronger regularization) slightly improved performance.
- The 'liblinear' solver was effective for this binary classification task.
- High precision and recall indicate the model's effectiveness in distinguishing between the two classes.

4.2 KNN Results

Summary Table: KNN (Binary Classification)

Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
n_neighbors=3, weights='uniform', metric='euclidean'	98.25	98.84	97.65	98.24
n_neighbors=5, weights='uniform', metric='euclidean'	98.31	98.99	97.62	98.30
n_neighbors=7, weights='distance', metric='manhattan'	98.48	99.11	97.83	98.47
n_neighbors=9, weights='distance', metric='manhattan'	98.53	99.14	97.92	98.52
n_neighbors=11, weights='uniform', metric='minkowski'	98.15	99.12	97.17	98.13

Analysis:

- The best accuracy achieved was 98.53% with n_neighbors=9, weights='distance', metric='manhattan'.
- Using distance weighting allowed closer neighbors to have more influence, enhancing the model's performance.
- The manhattan distance metric provided better results compared to euclidean.

4.3 SVM Results

Summary Table: SVM (Binary Classification)

Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
C=0.1, max_iter=1000	96.71	99.63	93.77	96.61	99.66
C=1, max_iter=1000	98.35	99.73	96.97	98.33	99.86
C=10, max_iter=1000	98.68	99.66	97.70	98.67	99.92
C=1, max_iter=2000	98.35	99.73	96.97	98.33	99.86
C=1, max_iter=5000	98.35	99.73	96.97	98.33	99.86

Analysis:

- The highest accuracy achieved was 98.68% with C=10.
- Higher C values improved performance, indicating that allowing a smaller margin with more support vectors benefited the model.
- The SVM achieved the highest ROC-AUC score, indicating excellent class separation.

4.4 Decision Tree Results

Summary Table: Decision Tree (Binary Classification)

Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
max_depth=5, min_samples_leaf=1, criterion='gini'	97.91	98.01	97.80	97.91	97.88
max_depth=10, min_samples_leaf=2, criterion='entropy'	97.55	97.74	97.35	97.55	97.73
max_depth=None, min_samples_leaf=1, criterion='gini'	97.12	96.67	97.60	97.13	97.12
max_depth=15, min_samples_leaf=4, criterion='entropy'	97.66	97.78	97.53	97.66	98.14

Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
max_depth=3, min_samples_leaf=2, criterion='gini'	97.58	98.29	96.85	97.57	98.30

Analysis:

- The highest accuracy achieved was 97.91% with max_depth=5.
- Limiting the max_depth to prevent overfitting maintained good performance.
- The gini criterion slightly outperformed entropy in this context.

5. **Comparison**

We compared the performance of all algorithms to determine the best performer for binary classification.

Summary of Best Models: Binary Classification

Model	Best Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
SVM	C=10, max_iter=1000	98.68	99.66	97.70	98.67	99.92
Logistic Regression	C=0.1, solver='liblinear', max_iter=100	98.73	99.04	98.42	98.73	99.79
KNN	n_neighbors=9, weights='distance', metric='manhattan'	98.53	99.14	97.92	98.52	N/A
Decision Tree	max_depth=5, min_samples_leaf=1, criterion='gini'	97.91	98.01	97.80	97.91	97.88

Summary of Results

- **Logistic Regression** achieved high accuracy across all hyperparameter settings, with the best accuracy of 98.73% using C=0.1, solver='liblinear', and max_iter=100. Precision and recall were also high, indicating a strong ability to correctly identify both classes.
- **k-Nearest Neighbors (KNN)** performed well, with the best accuracy of 98.53% achieved using n_neighbors=9, weights='distance', and metric='manhattan'. The use of distance weighting and the Manhattan distance metric improved performance.
- **Support Vector Machines (SVM)** achieved the highest accuracy among all algorithms, with 98.68% using C=10 and max_iter=1000. The high ROC-AUC score of 99.92% indicates excellent class separation.

- **Decision Trees** had slightly lower accuracy compared to the other algorithms, with the best accuracy of 97.91% using max_depth=5, min_samples_leaf=1, and criterion='gini'.

Conclusions

- **Best Performing Model Based on Accuracy**: Logistic Regression was the best-performing model in terms of accuracy, achieving the highest score of 98.73%.
- **SVM's Strength in ROC-AUC: Support Vector Machine** had the highest ROC-AUC score of 99.92%, indicating excellent class separation and the model's strong ability to rank positive instances higher than negative ones.
- **Interpretation of Results:**
    - The slight difference in accuracy between Logistic Regression and SVM is minimal (0.05%). Both models performed exceptionally well.
    - The higher ROC-AUC score of SVM suggests that it might perform better in ranking tasks or when considering the trade-off between true positive and false positive rates.
    - However, since Logistic Regression achieved the highest accuracy, it can be considered the best-performing model for this classification task.
- **Recommendations:**
    - Use Logistic Regression for Highest Accuracy: Logistic Regression is recommended for applications where the highest possible accuracy is desired.
    - Consider SVM for Better Class Separation: If the priority is on maximizing the ROC-AUC score and ensuring excellent class separation, SVM is a strong candidate.
    - Evaluate Based on Application Needs: The choice between Logistic Regression and SVM may depend on specific application requirements, such as the importance of accuracy versus ROC-AUC, computational resources, and model interpretability.

Final Remarks

While Logistic Regression achieved the highest accuracy in this binary classification task, both Logistic Regression and SVM demonstrated excellent performance. The choice between them should be guided by specific needs, such as computational efficiency, model interpretability, and the importance of maximizing either accuracy or ROC-AUC.

Part 2: Multi-Class Classification

In this part, we used the entire Fashion MNIST dataset to classify images into all 10 categories. We implemented and evaluated Logistic Regression, KNN, SVM, and Decision Trees. For each algorithm, we experimented with at least five different hyperparameter combinations and evaluated their performance using accuracy, precision, recall, F1-score, and confusion matrix.

1. Dataset Preparation

1.1 Load Data and Import Necessary Libraries

We used the same data loading procedures as in Part 1.

1.2 Separate Features and Labels

We separated the features and labels for both the training and testing datasets.

- Training Set: 60,000 images
- Testing Set: 10,000 images
-

1.3 Feature Scaling

We applied standardization to the data using StandardScaler.

2. Algorithms to Use

We implemented and evaluated the same algorithms as in the binary classification task:

- Logistic Regression
- k-Nearest Neighbors (KNN)
- Support Vector Machines (SVM)
- Decision Trees

3. Hyperparameter Tuning

3.1 Logistic Regression

3.1.1 Hyperparameter Combinations

We included the multi_class='multinomial' parameter to handle multi-class classification. The combinations were:

1. C=0.01, solver='lbfgs', max_iter=100, multi_class='multinomial'
2. C=0.1, solver='lbfgs', max_iter=200, multi_class='multinomial'
3. C=1, solver='sag', max_iter=100, multi_class='multinomial'
4. C=10, solver='saga', max_iter=200, multi_class='multinomial'
5. C=100, solver='lbfgs', max_iter=300, multi_class='multinomial'

3.1.2 Model Training and Evaluation
Models were trained and evaluated, with convergence warnings addressed by adjusting max_iter.

3.2 k-Nearest Neighbors

3.2.1 Hyperparameter Combinations
We explored different values for n_neighbors, weights, and metric:

1. n_neighbors=3, weights='uniform', metric='euclidean'
2. n_neighbors=5, weights='distance', metric='manhattan'
3. n_neighbors=7, weights='uniform', metric='minkowski'
4. n_neighbors=9, weights='distance', metric='euclidean'
5. n_neighbors=11, weights='uniform', metric='manhattan'

3.2.2 Model Training and Evaluation
Each KNN model was trained and evaluated on the multi-class dataset.

3.3 Support Vector Machines

3.3.1 Hyperparameter Combinations
Due to computational constraints, we used LinearSVC. The combinations were:

1. C=0.01, max_iter=1000
2. C=0.1, max_iter=1000
3. C=1, max_iter=2000
4. C=10, max_iter=3000
5. C=100, max_iter=5000

3.3.2 Model Training and Evaluation
Models were trained, with convergence warnings addressed by adjusting max_iter and C.

3.4 Decision Trees

3.4.1 Hyperparameter Combinations
We experimented with various depths and splitting criteria:

1. max_depth=10, min_samples_leaf=1, criterion='gini'
2. max_depth=20, min_samples_leaf=2, criterion='entropy'
3. max_depth=None, min_samples_leaf=1, criterion='gini'
4. max_depth=15, min_samples_leaf=4, criterion='entropy'
5. max_depth=5, min_samples_leaf=2, criterion='gini'

3.4.2 Model Training and Evaluation

Each Decision Tree model was trained and evaluated on the multi-class dataset.

4. Performance Evaluation

4.1 Logistic Regression Results

Summary Table: Logistic Regression (Multi-Class Classification)

Parameters	Accuracy (%)	Macro Precision (%)	Macro Recall (%)	Macro F1-Score (%)
C=0.01, solver='lbfgs', max_iter=100	84.37	84.32	84.37	84.33
C=0.1, solver='lbfgs', max_iter=200	83.07	83.05	83.07	83.05
C=1, solver='sag', max_iter=100	84.38	84.31	84.38	84.33
C=10, solver='saga', max_iter=200	84.38	84.32	84.38	84.33
C=100, solver='lbfgs', max_iter=300	78.31	78.46	78.31	78.37

Analysis:

- The highest accuracy achieved was 84.38% with C=0.01 and C=1.
- Logistic Regression outperformed other algorithms in multi-class classification.
- Moderate regularization provided the best balance between bias and variance.

4.2 KNN Results

Summary Table: KNN (Multi-Class Classification)

Parameters	Accuracy (%)	Macro Precision (%)	Macro Recall (%)	Macro F1-Score (%)
n_neighbors=3, weights='uniform', metric='euclidean'	81.52	82.23	81.52	81.58
n_neighbors=5, weights='distance', metric='manhattan'	82.86	83.28	82.86	82.92
n_neighbors=7, weights='uniform', metric='minkowski'	81.82	82.52	81.82	81.85
n_neighbors=9, weights='distance', metric='euclidean'	81.83	82.62	81.83	81.92
n_neighbors=11, weights='uniform', metric='manhattan'	82.51	82.86	82.51	82.46

Analysis:

- The highest accuracy achieved was 82.86% with n_neighbors=5, weights='distance', metric='manhattan'.
- KNN performed competitively but slightly below Logistic Regression.
- Distance weighting and the manhattan metric consistently improved performance.

4.3 SVM Results

Summary Table: SVM (Multi-Class Classification)

Parameters	Accuracy (%)	Macro Precision (%)	Macro Recall (%)	Macro F1-Score (%)
C=0.01, max_iter=1000	82.75	82.53	82.75	82.57
C=0.1, max_iter=1000	80.75	80.55	80.75	80.60
C=1, max_iter=2000	78.52	78.27	78.52	78.34
C=10, max_iter=3000	77.27	77.06	77.27	77.11
C=100, max_iter=5000	76.70	76.52	76.70	76.56

Analysis:

- The highest accuracy achieved was 82.75% with C=0.01.
- Stronger regularization (smaller C) was beneficial.
- Performance was slightly lower than Logistic Regression.

4.4 Decision Tree Results

Summary Table: Decision Tree (Multi-Class Classification)

Parameters	Accuracy (%)	Macro Precision (%)	Macro Recall (%)	Macro F1-Score (%)
max_depth=10, min_samples_leaf=1, criterion='gini'	77.50	78.03	77.50	77.67
max_depth=20, min_samples_leaf=2, criterion='entropy'	76.33	76.42	76.33	76.34
max_depth=None, min_samples_leaf=1, criterion='gini'	75.36	75.50	75.36	75.42
max_depth=15, min_samples_leaf=4, criterion='entropy'	76.76	76.96	76.76	76.83
max_depth=5, min_samples_leaf=2, criterion='gini'	69.43	71.86	69.43	68.15

Analysis:

- The highest accuracy achieved was 77.50% with max_depth=10.

- Decision Trees struggled with the complexity of multi-class classification.
- The lower accuracy suggests that the model may benefit from ensemble methods.

5. Comparison

We compared the performance of all algorithms to determine the best performer for multi-class classification.

Summary of Best Models: Multi-Class Classification

Model	Best Parameters	Accuracy (%)	Macro Precision (%)	Macro Recall (%)	Macro F1-Score (%)
Logistic Regression	C=1, solver='sag', max_iter=100, multi_class='multinomial'	84.38	84.31	84.38	84.33
KNN	n_neighbors=5, weights='distance', metric='manhattan'	82.86	83.28	82.86	82.92
SVM	C=0.01, max_iter=1000	82.75	82.53	82.75	82.57
Decision Tree	max_depth=10, min_samples_leaf=1, criterion='gini'	77.50	78.03	77.50	77.67

Key Findings:

- Best Performing Model: Logistic Regression with C=1, achieving an accuracy of 84.38%.
- Logistic Regression outperformed other algorithms, likely due to its ability to handle multi-class classification effectively.
- KNN is a viable alternative, especially with n_neighbors=5, weights='distance'.
- SVM showed decreased performance, possibly due to the linear nature of LinearSVC and the complexity of the dataset.
- Decision Trees had the lowest accuracy, indicating limitations in handling complex multi-class problems.

Conclusion

This study comprehensively evaluated the performance of Logistic Regression, KNN, SVM, and Decision Trees on the Fashion MNIST dataset for both binary and multi-class classification tasks.

- Binary Classification:
    - Best Model: Support Vector Machine with C=10.
    - Key Finding: SVM's capacity to handle high-dimensional data and find optimal separating hyperplanes made it the most effective.
- Multi-Class Classification:
    - Best Model: Logistic Regression with C=1.

- Key Finding: Logistic Regression's multinomial option and efficient optimization algorithms allowed it to outperform other models.

Summary Tables of Final Hyperparameters and Performance Metrics

Binary Classification

Model	Best Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC-AUC (%)
SVM	C=10, max_iter=1000	98.68	99.66	97.70	98.67	99.92
Logistic Regression	C=0.1, solver='liblinear', max_iter=100	98.73	99.04	98.42	98.73	99.79
KNN	n_neighbors=9, weights='distance', metric='manhattan'	98.53	99.14	97.92	98.52	N/A
Decision Tree	max_depth=5, min_samples_leaf=1, criterion='gini'	97.91	98.01	97.80	97.91	97.88

Multi-Class Classification

Model	Best Parameters	Accuracy (%)	Macro Precision (%)	Macro Recall (%)	Macro F1-Score (%)
Logistic Regression	C=1, solver='sag', max_iter=100, multi_class='multinomial'	84.38	84.31	84.38	84.33
KNN	n_neighbors=5, weights='distance', metric='manhattan'	82.86	83.28	82.86	82.92
SVM	C=0.01, max_iter=1000	82.75	82.53	82.75	82.57
Decision Tree	max_depth=10, min_samples_leaf=1, criterion='gini'	77.50	78.03	77.50	77.67