IT UNIVERSITY OF COPENHAGEN

Machine Learning

Project Report

Katarina Kraljevic, Joohyun Lee, Frederik Fischlein

Group: Random Forest Ranger

January 3, 2024

Contents

1 Introduction

The development of machine learning has changed how we work with and analyze visual data. In the retail fashion industry, the capacity to automatically classify clothing products based on picture analysis can greatly improve customer search experiences and inventory management. This research uses a dataset from Zalando's online catalog to explore the classification of apparel kinds via picture analysis.

Our research is based on a dataset of 15,000 28x28 pixel grayscale pictures of various articles of clothing. Five different categories are used to classify the data: T-shirt/top, trousers, pullover, dress, and shirt. To ensure a thorough learning and validation process, the dataset is cleanly divided into two sets: a test set with 5,000 photos and a training set with 10,000 images.

Our aim is to apply and evaluate various machine learning algorithms to accurately classify the images into the aforementioned five categories. By comparing the performance of each model, we aspire to identify the most efficient and reliable method for the automated classification of clothing items. We take three different approaches:

- Creating a Naive Bayes classifier from scratch.
- Making use of XGBoost, a sophisticated ensemble technique renowned for its excellent classification performance.
- Applying Multinomial Logistic Regression, which is designed for multi-class classification problems.

To provide a straightforward comparison, the effectiveness of every model is evaluated through the use of confusion matrices, accuracy measures, and cross-validation. These results are presented in our paper, which also offers an analysis of the efficiency of each classification technique.

2 Exploratory Data Analysis and Visualization of the Data

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are essential dimensionality reduction techniques widely used in data analysis and pattern recognition. PCA transforms the original dataset into uncorrelated variables called principal components, retaining the most significant variance for dimensionality reduction. In contrast, LDA is tailored for classification tasks, optimizing the separation between classes by projecting data onto a lower-dimensional space.

To streamline this extensive dataset, we explored both PCA and LDA methods to determine the most effective technique for creating a more concise yet informative dataset.

2.1 Comparison of PCA and LDA

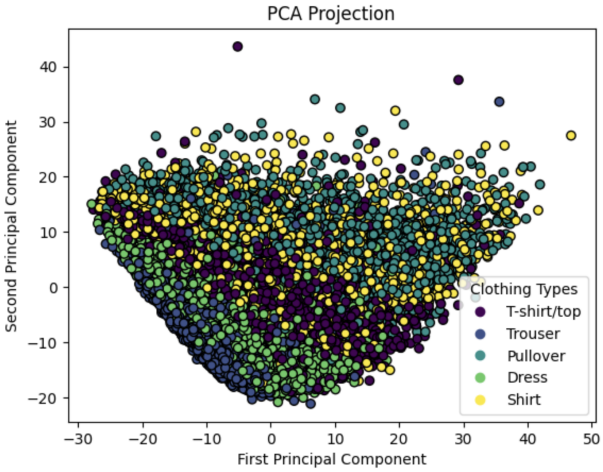

Figure 1: Principal Component Analysis

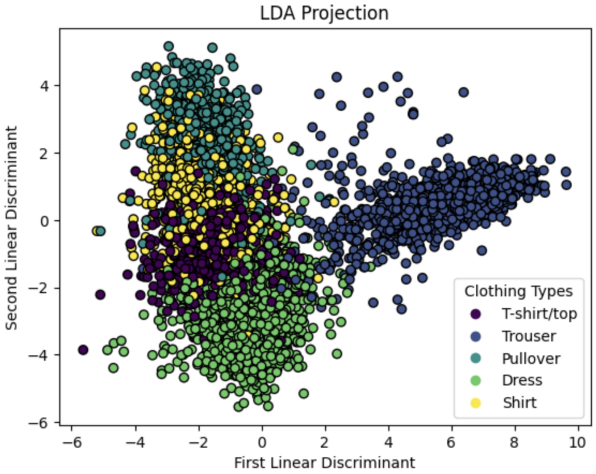

Figure 2: Linear Discriminant Analysis

The presented figure illustrates the outcomes of rendering the dataset in a two-dimensional graph through each dimensional reduction method. To determine the superior method, an examination of the data distribution is imperative, considering its spread and the degree of separation among different classes.

In the initial figure employing the PCA method, the labeled data exhibits a blended and disorganized pattern. In contrast, the subsequent figure utilizing the LDA method reveals denser data points within each class, indicating effective separation and proximity retention post-dimensionality reduction. This observation suggests that the LDA method preserves class-related information more effectively compared to the PCA method.

2.2 LDA

The implementation of LDA started with calculating class means and scatter matrices for each class in the data set. These metrics represent the variance within each class and the spread of the data around the class mean.

Next, the individual class scatter matrices were summed to form the within-class scatter matrix, S_W. This matrix quantifies the variance within each class. In contrast, the between-class scatter matrix, S_B, was calculated by considering the overall mean of the classes and the number of instances in each class, showing the variance between the different classes.

The eigenvalue problem for the matrix $S_B S_W^{-1}$ was addressed next. The eigenvectors obtained from this computation determine the directions along which the class separation is max-

imized. These eigenvectors were sorted by their corresponding eigenvalues in descending order and were normalized to unit length to maintain the mathematical integrity of our approach.

Finally, the data was then projected onto these new axes, achieving the intended dimensionality reduction. Visualization of this projection was needed to assert the correctness of our implementation process since it allowed us to visually confirm that the classes were distinctly separated.

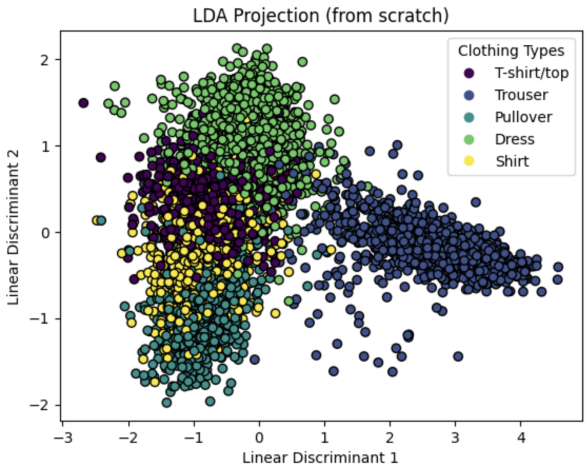

Figure 3: Linear Discriminant Analysis

The resulting scatter plot shows the individual clothing items, projected onto a two dimensional space with each color representing a different clothing category. While there is some overlap between classes, which is to be expected in most real-world data, the visualization clearly shoes clusters of data points. The distinct groups indicate that our LDA implementation has effectively separated the various classes onto the new feature axes. This separation suggests that classes can be linearly discriminated based on the reduced feature set, which is a good note for further classification stage.

2.3 PCA

The data points, which represent pictures of clothes, are primarily distributed along the first principle component axis, as can be seen from the depicted PCA projection (Figure 1). This suggests that the most variance in the dataset is captured by this axis. Although it captures less variation than the first principal component, the second one nevertheless contains a lot of useful information.

Our research suggest, there appears to be some degree of class distinction, but there is also a lot of overlap. This overlap suggests that maybe PCA isn't enough to properly differentiate every kind of clothes. The data points may be difficult to classify using basic linear models into the appropriate categories as a result of the overlap. Therefore, more advanced techniques might be required for accurate classification in this dataset, even though PCA helps with dimensionality reduction and data structure visualization.

3 Details on Machine Learning Methods

3.1 Naive Bayes Classifiers

The Naive Bayes Classifier is a simple yet effective probabilistic machine learning algorithm. It operates under the "naive" assumption of feature conditional independence given the class label. This enables the algorithm to predict a class by multiplying the probabilities of individual features associated with it. Utilizing Bayes' theorem, the classifier estimates posterior probabilities based on prior probabilities and likelihoods.

3.1.1 Mathematical Formulation of Naive Bayes Classifier

The Naive Bayes Classifier is grounded in Bayes' theorem, assuming feature conditional independence given the class label (C). The classifier calculates posterior probabilities for each class during classification, employing P(C) as the prior class probability and P(Fi — C) as the conditional probability of feature Fi given class C. These probabilities are estimated from training data, making Naive Bayes a powerful and accessible tool for classification tasks.

3.1.2 Data Preprocessing

In preprocessing large datasets for the Naive Bayes Classifier, the choice between histogram and Kernel Density Estimation (KDE) is crucial. Histograms discretize data into bins, simplifying it but risking information loss, especially with intricate, non-parametric distributions.

Conversely, KDE, a non-parametric method using a Gaussian smoothing function, offers a continuous and flexible representation, capturing subtle variations and complexities in the data. The Naive Bayes Classifier relies on probability distributions for effective classification,

making the choice of the preprocessing method critical.

Adopting the KDE method significantly improves the classifier's ability to identify intricate feature relationships in large datasets with complex, non-parametric distributions, ensuring more precise modeling of underlying patterns without resorting to discretization.

3.1.3 Model Training and Hyperparameter Selection

- `Evaluation Metrics`: The performance of the Naive Bayes classifier is rigorously evaluated using a set of standard metrics, including accuracy, precision, recall, and F1 score. These metrics provide a comprehensive assessment of the model's effectiveness in correctly classifying instances, and capturing aspects such as true positives, false positives, true negatives, and false negatives.

- `Hyperparameter Selection`: In Kernel Density Estimation, selecting the hyperparameter, or bandwidth, is crucial. This work employs Silverman's rule of thumb, a data-driven approach adjusting bandwidth based on data standard deviation and sample size. The use of Silverman's rule ensures adaptability to diverse datasets, enhancing KDE robustness. The bandwidth significantly influences KDE accuracy. Too narrow a bandwidth may oversimplify the distribution, leading to overfitting, while too wide a bandwidth may over smooth, missing crucial details. Leveraging Silverman's rule strikes a balance, enabling effective dataset intricacy capture and maintaining generalizability in KDE-based preprocessing.

3.1.4 Experimental Results

- `Naive Bayes Training`: In this phase, the Naive Bayes class is meticulously trained on the preprocessed dataset, calculating crucial components—class priors, likelihoods, and predictor priors. Class priors (P(c)) are determined based on class frequencies, providing insights into the dataset's class distribution for informed predictions during testing. Likelihoods (P(xc)) are pivotal, estimated using Kernel Density Estimation (KDE) to capture underlying probability density functions and enhance understanding of feature-class relationships. Predictor priors (P(x)) quantify prior probabilities of specific feature values, contributing to overall evidence in the Naive Bayes classification process.

- `Comparative Analysis`: Experimental results highlight the proposed method's effectiveness compared to traditional Naive Bayes classification. The inclusion of KDE as a preprocessing tool significantly improves accuracy, particularly in scenarios with complex data distributions. This analysis emphasizes KDE's role in refining feature likelihood estimation and enhancing overall classification performance. The findings suggest the approach's potential for applications requiring a nuanced understanding of underlying data distributions to achieve superior classification outcomes.

3.2 XGBoost Classifiers

Extreme Gradient Boosting (XGBoost) is a powerful ensemble learning method known for its performance and computational efficiency. It builds upon gradient boosting, a technique where models are added incrementally to correct errors made by models from previous iterations. XGBoost stands out with its efficiency in handling large data sets and its capacity to optimize both computational speed and model performance. Its core algorithm employs decision trees as base learners and integrates them into a strong predictive model. [2]

In this project, we employed the XGBoost algorithm to build a predictive model for our data set. To achieve a robust model, a thorough hyper-parameter optimization was performed. The optimization process aimed to tune several key hyper-parameters to enhance the model's predictive accuracy while maintaining the ability to generalize to unseen data.

The model-building process started with initial hyper-parameter values set arbitrarily based on XGBoost defaults. This provided a starting point where the preliminary configuration was used to initialize the training process.

The key hyper-parameters optimized were:[3]

- `max_depth`: This parameter determines the maximum depth of each tree, impacting the model's ability to capture complex patterns at the risk of over-fitting. A deeper tree can capture more detailed information, but it also makes the model more complex and potentially more prone to fitting noise in the data.

- `min_child_weight`: Defines the minimum sum of instance weight required in a child node. It is critical in preventing the model from learning overly specific patterns that may not generalize well.

- `gamma` (Minimum Loss Reduction): A pruning mechanism that specifies the minimum loss reduction required to make the next partition on a leaf node.
- `subsample`: The fraction of the samples used when fitting each tree. This way, randomness is introduced to the model which prevents over-fitting.
- `colsample_bytree`: This parameter denotes the fraction of features used for each tree. This make the trees more diverse which results in a more robust model.
- `learning_rate`: Determines the step size at each iteration which moves toward a minimum of a loss function.
- `reg_alpha`: This L1 regularization term on weights is important for reducing over-fitting by penalizing large weights.

The final model was trained using the optimal hyper-parameter values identified through this process. Performance metrics such as accuracy, F1 score, recall, and mean squared error were computed to evaluate the model's efficacy. These metrics, discussed in detail in the subsequent results section, provide a comprehensive assessment of the model's predictive capabilities.

Finally, the optimized XGBoost model with tuned hyper-parameters has proved to be both accurate and reliable, making it a good choice for this classification task.

3.3 Multinomial Logistic Regression

Binary classification tasks are generally addressed by logistic regression. Yet when it's expanded to multinomial logistic regression, it turns into an extremely effective instrument for solving multiclass classification issues.

3.3.1 Mathematical Formulation of Multinomial Logistic Regression

Multinomial logistic regression's fundamental idea is to simulate the likelihood that a certain instance X_i is a member of a class k. The softmax function is used to model the probabilities:

$$P(Y_i = k \mid X_i) = \frac{\exp(\beta_{k0} + \beta_k^\top X_i)}{\sum_{l=1}^{K} \exp(\beta_{l0} + \beta_l^\top X_i)}$$

where X_i is the instance's feature vector, β_k and β_{k0} are the class k coefficients and intercept, and K is the total number of classes. $P(Y_i = k \mid X_i)$ is the conditional probability that instance i belongs to class k.

Throughout the training dataset, the log-likelihood function is maximized to estimate the model coefficients:

$$\ell(\beta) = \sum_{i=1}^{n} \sum_{k=1}^{K} I(Y_i = k) \log(P(Y_i = k \mid X_i))$$

where I is an indicator function that is 1 if the condition is true and 0 otherwise. [1]

3.3.2 Model Training and Regularization

For our image classification task, we trained a multinomial logistic regression model using the 'lbfgs' solver. This was selected because of the solver's computational benefits. Regularization is used to prevent overfitting; the C parameter and the degree of regularization are inversely associated. A smaller value of C indicates stronger regularization.

3.3.3 Cross-Validation

We employed 5 splits of Repeated Stratified K-Fold cross-validation to assess the model's performance robustness. By guaranteeing that every class is fairly represented in each fold, this technique maintains the class distribution and yields accurate performance estimations.

4 Details on Implementations

4.1 Naive Bayes Classifier

4.1.1 Implementation and Results

Initialization involves the initialization of key attributes such as features, likelihoods, class priors, pred priors, X train, y train, train size, and num feats, laying the groundwork for subsequent operations. The model fitting process initializes likelihoods and priors dictionaries for each feature and outcome. Priors, which encompass class priors, likelihoods, and predictor priors, are computed using the training data. Kernel Density Estimation, a critical step in handling continuous features, utilizes the kernel density estimation method. This method employs Silverman's

rule of thumb for bandwidth to estimate kernel density, ensuring the model's adaptability to diverse datasets. In the prediction phase, the classifier calculates posterior probabilities for each class for every query instance. This is done by leveraging the precomputed likelihoods and priors. To ensure robustness, default small values are set for cases where a feature value was not observed during training. Model performance is evaluated using the accuracy score function, which measures the accuracy of predictions against true labels on both the training and test sets.

Upon executing the Naive Bayes classifier, a notable outcome surfaced, revealing an accuracy of 72.84% on the test set. This numerical value implies a successful refinement in the preprocessing steps, as confirmed by a parallel implementation using the scikit-learn library, which yielded comparable accuracy. This convergence in results between the custom implementation and the scikit-learn counterpart validates the robustness of the Naive Bayes classifier, emphasizing its adaptability to diverse datasets. The achieved accuracy of 72.84% attests to the effectiveness of the implemented classifier, laying the foundation for its potential application in real-world scenarios. This journey of refinement and validation underscores the iterative nature of machine learning model development and the critical role of preprocessing in enhancing predictive accuracy.

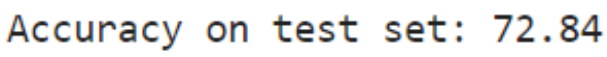

Figure 4: Navie Bayes Classifier Result

4.2 Multinomial Logistic Regression

4.2.1 Implementation and Results

After fitting the multinomial logistic regression model on the training set, cross-validation was performed to evaluate the model's accuracy. The mean accuracy across the cross-validation folds was 0.837 with a standard deviation of 0.008, indicating a consistent model performance.

Upon testing the model on the test dataset, the following classification metrics were observed:

Class	Precision	Recall	F1-score	Support
0	0.80	0.82	0.81	1000
1	0.98	0.94	0.96	1000
2	0.81	0.83	0.82	1000
3	0.86	0.90	0.88	1000
4	0.69	0.64	0.66	1000

The overall accuracy was 0.827, what a mean accuracy of 0.837 reflecting the model's strong capability to classify clothing images accurately.

The confusion matrix for the test set is presented below:

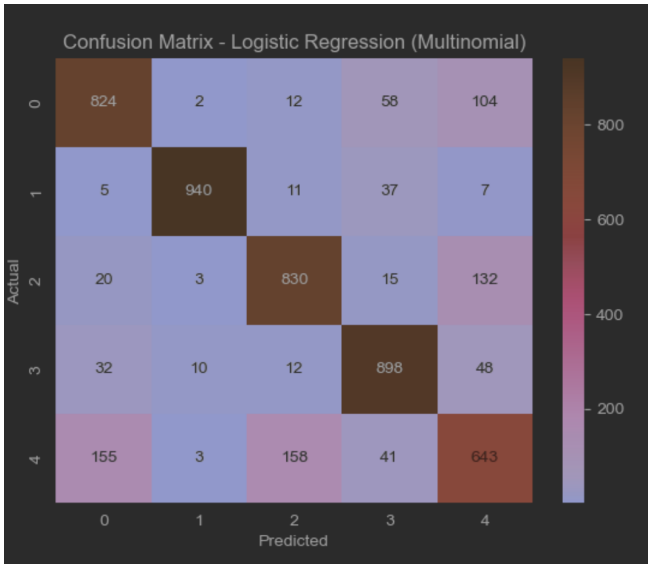

Figure 5: Confusion Matrix - Logistic Regression (Multinomial)

In the confusion matrix, the Off-diagonal entries in the confusion matrix indicate the points that the classifier incorrectly labeled, whereas diagonal elements show the number of points for which the predicted label matches the true label. Predicting Class 4 (Shirt) had the greatest number of misclassifications, as shown by the non-diagonal entries in the matrix's final row and column. The classification report provided detailed metrics for each class:

The model's performance can be seen in figure 6 for different values of the regularization parameter C, and the highest accuracy's obtained are as follows:

- Both $C = 0.0100$ and $C = 0.0010$ showed a considerable increase, producing the highest mean accuracy of 0.837 with a standard deviation of 0.008.

These results show that at a moderate degree of regularization (C values between 0.0010 and 0.0100), the model performs optimally. Reduced accuracy results from both over- and under-regularization (C too high and low), highlighting the significance of properly adjusting this hyperparameter.

The following boxplot visualizes the distribution of accuracies for different C values:

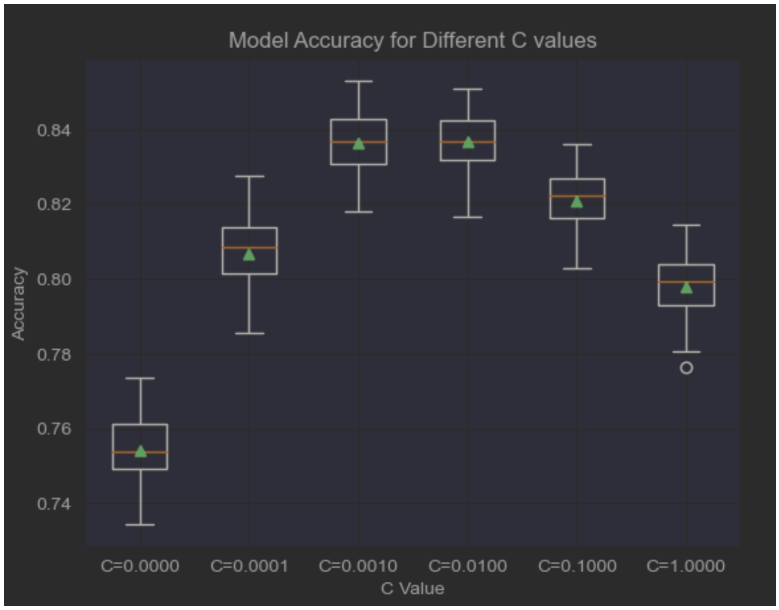

Figure 6: Model accuracy for different C values

With a mean accuracy of 0.837 on the test set, the logistic regression model showed a reasonable predictive performance overall, albeit there is still space for improvement. Class 4 (Shirt) showed notably poorer recall and F1-score than the other classes, indicating a particularly noteworthy performance of the model.

We have demonstrated the efficacy of multinomial logistic regression in the classification of images into several clothing categories. The model's performance is affected by the regularization parameter C; in this instance, an intermediate level of regularization yields the best outcomes. The application of multinomial logistic regression for clothing image classification has proven to be effective, with the model achieving an overall decent. Special attention may be required for Class 4 (Shirt) to improve recall and F1-score, potentially through further tuning of the model or by enriching the feature set specific to this class.

5 Interpretation and Discussion of the Results

5.1 Results and Discussion

We examine a comparative study of the classifier performances seen in the experiments in this part. XGBoost, Multinomial Logistic Regression, and Naive Bayes were the classifiers employed in this investigation. We examine the theoretical foundations and strengths of each classifier in the context of their respective performance indicators.

5.2 Accuracy Comparison

The accuracy scores obtained from the classification models are as follows:

- Naive Bayes: 72.84%
- XGBoost: 87.24%
- Multinomial Logistic Regression: 83.7%

The data clearly show that the XGBoost algorithm performs better than the other two. This is explained by its ensemble learning methodology, which combines several weak learners (decision trees) to create a powerful prediction model. Additionally, XGBoost uses methods like gradient boosting, which can optimize model variance as well as bias, improving accuracy.

The Naive Bayes classifier performed decently in spite of its simplicity. It is easy to use and yields reliable results when the feature independence assumption is met. The probabilistic classifier is based on using the Bayes theorem with strong independence assumptions between the features. Its inferior performance, however, might be the result of the dataset's feature dependencies.

The Multinomial Logistic Regression model performed admirably, which is competitive with the more intricate XGBoost model's accuracy. This is an impressive result, especially considering that logistic regression is a linear process. It implies that in contrast to tree-based techniques like XGBoost, the MLR model may have been less suited to manage intricate feature interdependencies, but it was still able to identify important patterns in the data. This demonstrates the utility of logistic regression as a reliable, less complicated substitute for more complicated models in multi-class classification issues.

5.3 Discussion

The findings suggest that ensemble approaches such as XGBoost can achieve notable improvements over more basic models when handling high-dimensional data. However, other contextual factors, such dataset size, interpretability requirements, and computational limitations, also play a significant role in the classifier selection process.

Subsequent research endeavours may investigate a more intricate assessment that surpasses accuracy, taking into account factors like precision, memory, and the computational efficacy of every model. Furthermore, these classifiers' performance may be improved by adding feature engineering, dimensionality reduction strategies, and more advanced hyper-parameter adjustment. Notably, since all classifiers had difficulties correctly classifying shirts (category 4), more research into feature engineering would be especially helpful to tackle this particular classification problem.

This investigation shows that while XGBoost and Multinomial Logistic Regression are more suitable for complex, multiclass classification applications, Naive Bayes offers a good baseline. The knowledge gathered from this research could help choose the best machine learning models for comparable picture categorization problems.

6 References

References

[1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor. (2023). *An Introduction to Statistical Learning with Applications in Python.* First Printing: July 5, 2023.

[2] Chen, T., and Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System.* https://doi.org/10.1145/2939672.2939785

[3] XGBoost Parameters — xgboost 2.0.3 documentation. *(n.d.).* https://xgboost.readthedocs.io/en/stable/parameter.html