HANOI UNIVERSITY

Faculty of Information Technology

BDM Midterm Project Report

(Link to Github: https://github.com/hvuminh02/BDM_MidtermProject)

Student: Hoang Vu Minh - 2001040131

Luong Dinh Thai - 2001040184

Lecturer: Bui Quoc Khanh

Class: Tut 03

Course: Big Data Mining

I. Overview

In exploring the intricate world of algorithms through our recent project, we unearthed a myriad of findings that not only deepened our understanding but also reshaped our perspective on the computational solutions that govern much of our digital existence. This report distills those discoveries, presenting key insights into our comprehension of algorithms and personal reflections that add a layer of authenticity and depth to our exploration.

II. Analysis & Discussion

A. Logistic Regression

This project explores the implementation and application of Logistic Regression on the Wine Quality dataset from the UCI Machine Learning Repository. This dataset provides an interesting opportunity to predict the quality of red wine based on various physicochemical properties.

Key Findings and Takeaways

Normalization and Standardization: The project implements two crucial preprocessing steps - Min-Max normalization and Z-score standardization. Normalization adjusts the data dimensions so that they are of approximately the same scale, while Z-score standardization ensures the data follows a standard distribution, which is critical for logistic regression to perform optimally.

Logistic Regression from Scratch: By building the logistic regression model from the ground up, the project illuminates the underlying mechanics of this popular classification algorithm, particularly how it uses the sigmoid function to model the probability that a given input belongs to a particular category.

Gradient Descent Optimization: The project employs gradient descent, a core algorithm for optimizing the logistic regression model parameters. This process iteratively adjusts the parameters to minimize the cost function, which is a measure of how well the model predicts the wine quality.

Prediction Accuracy: The final part of the project assesses the model's prediction accuracy on the dataset. Through iteration and optimization, the model learns to distinguish between different quality ratings (in this case, quality 5 vs. 6) with a calculated accuracy. This demonstrates the practical applicability of logistic regression in binary classification problems.

Understanding of Algorithms

Through this project, I've deepened my understanding of logistic regression and the importance of data preprocessing steps like normalization and standardization. The hands-on experience has made it clear how gradient descent serves as a powerful tool for optimizing the model's parameters, illustrating the direct impact of learning rate and iteration count on model performance.

Personal Perspectives
Working on this project has been both challenging and rewarding. Implementing logistic regression from scratch provided me with a profound appreciation for the elegance and simplicity of this algorithm, despite its powerful capabilities. The process of manually coding the functions that are often taken for granted in high-level libraries like Scikit-learn was particularly enlightening. It not only cemented my understanding of the theoretical aspects but also improved my programming skills, especially in Numpy.

Moreover, applying the model to the Wine Quality dataset and achieving tangible results in predicting wine quality based on its physicochemical properties was immensely satisfying. It was a concrete demonstration of how data science can bridge the gap between data and practical insights, even in fields as nuanced as enology.

B. Decision Tree & Random Forest

**** Decision Tree**

Key Findings and Takeaways
This project led to several critical insights into the design, implementation, and efficacy of a custom Decision Tree classifier, especially in the context of predicting wine quality. The ability to manipulate parameters such as max_depth, size_allowed, n_features, and n_split provided a nuanced understanding of how decision trees function and their optimization. The model's commendable performance on the wine quality dataset underscored the value of decision trees in multi-class classification challenges, highlighting the importance of feature selection and the intricate balance between model complexity and the risk of overfitting.

Understanding of Algorithms
Building the Decision Tree from scratch provided practical insights into its workings, including entropy calculations and the significance of information gain for splits. This hands-on experience clarified the theoretical aspects of decision trees and introduced the complexities of algorithm optimization.

Personal Perspectives
This project was not just a technical challenge but a rewarding learning experience. It improved my problem-solving skills and deepened my understanding of machine learning. Working on a real-world problem like wine quality prediction made the learning process engaging and highlighted the practical value of machine learning skills.

**** Random Forest**

Key Findings and Takeaways

The project centered on building a RandomForest classifier, resulting in a model configured with 100 trees and a maximum depth of 1000. This configuration was pivotal in demonstrating the power of ensemble methods, showcasing significant improvement in prediction accuracy over single decision trees. The model's success underscored the value of aggregating predictions to enhance performance and reliability.

Understanding of Algorithms

Through the implementation process, there was a deep dive into the mechanics of the RandomForest algorithm. This included an exploration of how it builds multiple decision trees and uses their collective decisions to improve accuracy. Key algorithmic insights gained were the role of random feature selection and the use of majority voting for final prediction, which collectively contribute to the model's robustness against overfitting. The customizable parameters such as the number of trees (n_trees), maximum depth (max_depth), and the minimum size allowed for splits (size_allowed) were instrumental in fine-tuning the model's performance.

Personal Perspectives

On a personal note, this project was highly enriching, offering a hands-on experience in developing a complex machine-learning model from the ground up. It was particularly illuminating to see how theoretical concepts apply in practice, especially the nuances of balancing bias and variance through parameter adjustments. The project not only enhanced technical skills in coding and algorithm implementation but also fostered a deeper appreciation for the craftsmanship involved in machine learning. The challenges encountered, especially in optimizing the model and debugging, were valuable learning experiences, emphasizing the iterative nature of developing an effective machine-learning model.

C. KMeans

Key Findings and Takeaways:

K-Means Clustering:

Detailed insights into algorithmic behavior and optimization techniques were obtained by investigating K-means parameters such as the number of clusters (k), maximum iterations, and convergence criteria. Comprehending how centroid initialization techniques (e.g., random selection, K-Means++) affect clustering results improved understanding of K-Means's convergence speed and sensitivity to initial conditions

Performance Evaluation:

K-means clustering performed well in dividing data into discrete clusters, as demonstrated by the Iris dataset evaluation. This was especially true for lower-dimensional feature spaces such as sepal length and width. By analyzing within-cluster sum of squares (WCSS) metrics for a range of k values, it was possible to discern the trade-off between variance reduction and model complexity, which helped with cluster selection decision-making.

Understanding of Algorithms:
Practical insights into the Expectation-Maximization nature of K-means clustering were obtained through deeper investigation of the centroid update mechanism and cluster assignment. Iterative optimization processes in unsupervised learning algorithms are better understood through the examination of convergence criteria and algorithmic convergence behavior.

Personal Perspectives:
A rewarding learning experience was provided by building K-Means from scratch, which enabled practical experimentation with clustering techniques and a deeper comprehension of algorithmic principles. Working with real-world datasets such as Iris helped people understand how K-Means can be applied practically in a variety of contexts, which emphasized the importance of machine learning abilities in data-driven problem-solving.

D. Gaussian Mixture

Key Findings and Takeaways:

Practical Application:

Creating an initial Gaussian Mixture Model allowed for practical clustering approach exploration. Understanding the effects of different convergence criteria and initialization procedures on model performance and convergence speed was made possible by investigating them. The capacity to tackle clustering problems in many areas and apply unsupervised learning techniques to real-world datasets was improved by this hands-on experience.

Algorithm Evaluation:

The significance of thorough evaluation and validation was brought to light by contrasting the outcomes of the custom implementation with those achieved using well-known libraries like sci-kit-learn. The means and scores generated by the two implementations were analyzed, providing a useful baseline for evaluating the performance of the custom model. This underlined how important it is to evaluate custom algorithms against proven implementations to guarantee their correctness and dependability.

Understanding of Algorithms:

The EM (Expectation-Maximization) algorithm is essential to the Gaussian Mixture Model (GMM). Iteratively, it calculates parameters of Gaussian distributions representing the mixture's clusters, including weights, covariances, and means. Expectation (E-step) and Maximisation (M-step) are the two primary steps of the EM algorithm. The current parameter estimates are used to calculate the probabilities of data points belonging to each cluster during the E-step. These probabilities are used to update the parameters in the next M-step to maximize the likelihood of the observed data. The model is guaranteed to converge to an ideal solution by this iterative refining procedure, which is carried out until convergence.

Personal Perspectives:

The Gaussian Mixture Model proved to be a great tool for understanding the complexities of unsupervised learning techniques. A deeper comprehension of the EM algorithm's inner workings and the difficulties associated with parameter estimation was made possible by starting from scratch when implementing it. Furthermore, testing various convergence criteria and initialization strategies brought to light how crucial algorithm design and parameter tuning are. All things considered, this experience cultivated a stronger respect for the theoretical underpinnings of machine learning algorithms in addition to improving technical skills.

E. Naive Bayes Classifier

Key Findings and Takeaways

Practical Application:
Developing the Naive Bayes classifier from the ground up provided practical experience in creating a fundamental machine learning algorithm. Calculating prior probabilities and likelihoods based on discrete feature values was a necessary step in fitting the model to training data, underscoring the significance of data preprocessing and comprehending the characteristics of input features. Additionally, the classifier's practical utility in real-world classification problems was showcased by its ability to make probabilistic predictions based on learned parameters when predicting class labels for test instances.

Algorithm Evaluation:
Analyzing the Naive Bayes classifier's performance on an actual dataset revealed both its advantages and disadvantages. The impact of the Naive Bayes assumption of

feature independence and the accuracy of the classification was evaluated by comparing the predicted class labels with the ground truth labels. A better understanding of the model's behavior in various settings and the variables affecting its predicted performance was made possible by the analysis of the classification findings.

Algorithmic Understanding:

The Naive Bayes classifier's fundamental conditional probability model was revealed through investigation, highlighting the method's ease of use and efficiency. The classifier makes the computation of the posterior probability simpler by assuming feature independence given the class label. This makes the classifier scalable and computationally efficient for large datasets. Knowing how the Naive Bayes model was built helped to clarify its underlying presumptions and how broadly it might be used to classify jobs.

Personal Perspectives:

The fundamental ideas and mechanisms of Naive Bayes were revealed through exploration. It uses conditional probability, assuming feature independence, to predict class labels from features. It computes prior probabilities and feature likelihoods to find the most likely class. Despite its simplicity, Naive Bayes offers insights into performance-complexity trade-offs and performs well in a variety of scenarios.