Machine Learning Applications and Insights
Final Paper

Alexandria Toothman
CUNY Graduate Center
DATA 71200: Advanced Data Analysis Methods
Dr. Devaney
July 12, 2024

Introduction

The Advanced Data Analysis course included three projects. Project 1 involved curating and cleaning a labeled dataset. Project 2 applied two supervised learning techniques, and Project 3 used two unsupervised learning techniques on the same dataset. This paper explores the technical implementation, performance evaluation, and key insights gained from these machine learning approaches.

Dataset Selection

For these projects, I chose the Glass Identification dataset from the UC Irvine Machine Learning Repository. This dataset consists of 214 instances and includes 9 continuous features: refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron. The target variable, type of glass, is categorical with seven distinct class labels: building windows (float processed), building windows (non-float processed), vehicle windows (float processed), containers, tableware, and headlamps. It is important to note, there are no instances for vehicle windows (non-float processed) in this dataset. The projects aimed to utilize machine learning algorithms to predict the category of glass.

The Glass Identification dataset was chosen for several key reasons, but primarily due to its manageable size, with 214 instances and 9 features. While small datasets can introduce challenges like overfitting and limited generalization, issues that were observed during these projects, they are easier to manage and require less preparation. This simplicity allowed me to focus on grasping and applying machine learning techniques. The dataset was also selected for its categorical target variable, which is ideal for classification tasks.

Data Cleaning and Preparation

Project 1 focused on the critical steps of cleaning, preparing, and exploring the dataset. Selecting the Glass Identification dataset simplified this process, as it was already well-cleaned. This advantage became apparent after several previous attempts with other datasets proved challenging. Once the dataset was selected, it was loaded into a data frame for examination. During this review, it was noted that the dataset included an ID number column. Since this column duplicated the index information, it was redundant and dropped. As part of the data

preparation phase, I ensured that there were no missing values, which is important before proceeding to data splitting and further analysis.

Preparing the target variable was straightforward. The target variable, "Class," which indicates the type of glass, was already represented as categorical integers (1 to 7). This meant there was no need for conversion like one-hot encoding. Since these classes represent distinct types of glass, like table glass (Class 6), and are not ranked or ordered, converting them to ordinal was also unnecessary.

Another important data preparation step was splitting the dataset into a training and testing sets. A stratified split is useful for classification tasks and aids in provided more accurate results (Müller & Guido, 2018). To create a balanced distribution of classes in the set, StratifiedShuffleSplit method from scikit-learn was used with a 80% training and 20% testing split.

In Project 2 and 3, feature scaling was applied as part of data preparation. This was important because several machine learning algorithms are sensitive to the scale of features, for example, K-Nearest Neighbors (KNN) used in Project 2. KNN calculates distances between data points to measure similarity (Müller & Guido, 2018). If the features are on different scales, it can introduce bias in calculations and lead to incorrect neighbor selection. By setting the scale from 0 to 1 for the training (X_train) and testing (X_test) sets, all the features are standardized and adjusted to fit within this range.

Insights from Data Visualizations

Throughout the three projects, data visualization proved invaluable for understanding both the dataset and the algorithms applied. It facilitated the detection of outliers and the identification of key features, crucial steps in model selection and evaluation. Visualization helped uncover critical insights that would have been challenging to identify. Additionally, it provided a powerful means to effectively communicate findings, making complex patterns and models more interpretable.

One of the most impactful visualizations was the histogram of the target attribute distribution in the training set. This histogram displayed the frequency of each glass type on the y-axis and the class labels on the x-axis. However, the histogram revealed a notable class imbalance, which was not Gaussian, uniform, or logarithmic in nature. This uneven distribution

presented challenges for model selection and performance. This imbalance needed careful consideration when evaluating a model's performance, as the overrepresentation, particularly the dominance of Class 2, could skew the results, leading to potential misclassification.

Visualizing the data in Project 1, and well as in Project 2 and 3, revealed critical insights the deepened the understanding of the Glass Identification dataset. The visualizations not only illuminated key aspects of the data, but also highlighted the potential impact on algorithm selection and suitability. It is evident that using data visualization is not only essential for exploratory data analysis, but also vital in effectively understanding and communicating the insights found.

Supervised Machine Learning Application

In Project 2, supervised learning techniques were applied. Supervised learning involves using an algorithm to learn a function to map input data to the desired output (Müller & Guido, 2018). Essentially, given input data, the algorithm can predict what the output will be. This is demonstrated with the Glass Identification dataset, where the algorithms predict the type of glass based on the feature inputs. Two supervised algorithms, decision tree and K-Nearest Neighbors, were applied to the dataset.

The first type of algorithm applied was a decision tree, which uses a hierarchical framework of if/else statements to arrive at a final classification (Müller & Guido, 2018). Each node in the decision tree represents a question about the data, and each branch represents a possible answer that leads to another question or a final decision, known as a leaf (Müller & Guido, 2018). In this project, the decision tree was constructed to classify different types of glass into classes 1-7. A decision tree can be thought of as playing the game "21 Questions", where one person thinks of an item and the other asks yes/no questions until they guess the item correctly. The decision tree navigates through the data's features to classify the type of glass based on questions about the features in the dataset. This iterative process continues until a final decision is made at the leaf nodes, determining the type of glass.

Before adjusting parameters, the accuracy report indicated a training set accuracy of 100%, signifying that the model did not make any errors. Achieving 100% accuracy on the training set signals overfitting and is common issue seen with decision trees (Géron, 2017). The accuracy on the test set was 65%, suggesting that the model does not generalize well to unseen

data. Adjusting the decision tree's parameters is necessary to mitigate overfitting and improve model performance.

Since the decision tree showed signs of overfitting on the training set, reducing the max_depth parameter can help regularize the model by reducing its freedom during training (Géron, 2017). Max_depth controls the depth of the tree, limiting complexity and overfitting through restricting the number of questions that can be asked (Müller & Guido, 2018). The choice of max_depth was guided by the dataset's 9 features, including None as an option. A grid search was conducted to find the optimal max_depth, resulting in 3 with an accuracy of 67.9%. However, evaluating the model's overall performance revealed that setting max_depth to 3 achieved 72% accuracy on the training set and 69% on the test set. It is typical when applying max_depth parameters to see a lower accuracy on the training set but an improvement on the testing set (Müller & Guido, 2018). This parameter application did indeed help to address the problem over overfitting seen in the model.

Another parameter that can control model complexity and mitigate overfitting is max_leaf_nodes which restricts the total number of leaf nodes in the decision tree (Müller & Guido, 2018). A grid search for max_leaf_nodes was conducted with parameters including None, 5, 10, 15, and 20. The optimal setting identified for max_leaf_nodes was 5, achieving an accuracy of 69.8%. Both parameters contributed to a modest improvement in the model's performance and aided in reducing overfitting.

The second supervised machine learning algorithm used for classification was K-Nearest Neighbor (KNN). This algorithm makes predictions on new data points based on which data points from the training set are closest. For example, if a new data point is close to a data point labeled as tableware, KNN predicts that the new point is likely tableware as well, hence the name neighbor. The algorithm predicts the new point's class based on its closest neighbor. The number of neighbors can be adjusted, and this was one of the parameter adjustments made to improve the model's performance.

Before making parameter adjustments, the baseline performance of the algorithm was measured. The accuracy of the model was 72% on the training set and 63% on the testing set, indicating room for improvement. However, when running the classification report to examine detailed metrics for each class, the overall accuracy significantly fell to 14% for the testing set and 13% for the training set. This discrepancy could be due to the class imbalance. But upon

further examination, it appeared there was an error in the code, it was not accounting for the scaled data. Proper scaling is crucial for KNN, as previously discussed. After correcting the code to use scaled data, the accuracy scores increased to 63% for the testing set and 72% for the training set.

To improve the model's performance, the first parameter adjustment involved determining the optimal number of neighbors for classification. A grid search indicated that k=1 was the best choice, yielding an accuracy score of 67%. However, the training set accuracy was 100%, suggesting overfitting. The next, parameter adjustment was the distance metric, which revealed that the Manhattan distance was the most effective. The Manhattan distance measures the distance between two points by moving up, down, left or right on the Cartesian plane, like walking along the grid of New York City streets (Devaney, 2024). In contrast, the default Euclidean distance calculates the straight-line distance between two points (Devaney, 2024). The Euclidean distance can be thought of with the saying "as the crow flies." The grid search for the best metric showed a 70% accuracy using the Manhattan distance.

When adjusting the parameters to k = 1 and metric = Manhattan, the training set accuracy remained at 100%, while the test set accuracy dropped to 53%, indicating overfitting and a decline in test accuracy. A visualization plotting accuracy against the number of neighbors for Manhattan distance showed that k = 1 was not optimal. Instead, k = 3 appeared to be a better option, underscoring the importance of visualizations for interpreting data insights.

With parameters set to k = 3 and metric = Manhattan, the training set accuracy improved to 83%, and the test set accuracy slightly improved to 67%. Adjusting only the parameter to k = 3, without specifying the metric as Manhattan, resulted in the best test set accuracy of 69%. Without specifying a metric, the model defaulted to using Euclidean distance. However, the grid search had not identified Euclidean as the best option. When examining the plot for Euclidean distance, like Manhattan, k = 3 emerged as the best number of neighbors.

Despite the grid search suggesting k = 1 as the best parameter, k = 3 was found to improve test accuracy more effectively. This demonstrates that additional data visualization and further validation are crucial in understanding model behavior and selecting the best parameters, beyond relying solely on grid search results.

PCA application on Supervised Machine Learning Algorithms

Initially Project 3 focused on exploring the impact of Principal Component Analysis (PCA) on decision trees. However, upon reevaluation, it became evident that KNN may be a more suitable model. Therefore, this write up will examine how applying PCA influenced the performance of the KNN model.

PCA is a pre-processing method that aids in dimensionality reduction and reduces overfitting by rotating the data so that the features are uncorrelated and then selecting a subset of features based on their importance (Müller & Guido, 2018). When PCA was applied to the KNN model with a goal of explaining 95% of the variance, seven components were identified as sufficient to capture the dataset's essential information.

Comparing the model's performance before and after applying PCA revealed a slight drop in accuracy. The training set accuracy decreased from 84% to 82%, and the test set accuracy fell from 79% to 77%. This reduction may stem from the simplification of the data through dimensionality reduction, which could lead to the model lacking the necessary details to identify certain patterns. Despite the minor decline in accuracy, PCA offers significant benefits by helping to mitigate overfitting. By reducing the model's complexity through dimensionality reduction, PCA helps the model generalize better to new data.

Additionally, PCA aids in feature extraction (Müller & Guido, 2018). A heatmap analysis of the PCA data showed that to capture 95% of the variance, seven components were necessary, making the interpretation challenging. However, when the PCA was limited to the two components that explained the most variance, the heatmap revealed that the features 'calcium' and 'refractive index' had the highest impact. This insight highlights the critical features driving the variance in the dataset and can guide further model refinement.

Unsupervised Machine Learning Applications and PCA as a Pre-Processing Step

Project 3 centered on the application of three unsupervised machine learning techniques and examined the impact of using PCA as a pre-processing step for these algorithms. The techniques explored were k-Means clustering, agglomerative clustering, and DBSCAN.

K-means is a clustering algorithm used in unsupervised machine learning that identifies cluster centers representing specific regions within the data (Müller & Guido, 2018). To grasp this concept, imagine trying to group cars in a parking lot by color without seeing the cars beforehand. The k-means algorithm randomly selects parking spaces as centroids and assigns

each car to the nearest centroid, forming clusters. Once cars are grouped, k-means recalculates each centroid based on the positions of all cars in their respective groups. When PCA is applied the resulting clusters become more distinct with reduced overlap. This preprocessing step tightens the clusters around their centroids and improves accuracy.

Agglomerative clustering is another machine learning algorithm used which treats each data point as its own cluster and gradually merges the most similar clusters to form larger ones. A dendrogram serves as a visual representation of this process, resembling a map that illustrates how individual clusters merge to form a larger group. The dendrogram is read from bottom to top and has branch which indicates where clusters were merged based on their similarities (Müller & Guido, 2018). The height of the branches indicates to how similar the groups were when merged. Higher branches show groups were less alike, while shorter ones show more similarities in a group. Using PCA enhanced the accuracy by reducing cluster overlap. When PCA was applied to the dendrogram, similar groups are identified and merged with less branches, revealing a more concise representation of the data. When applying PCA, the reduced data allows the algorithm to identify and merge cluster more effectively.

The third type of unsupervised machine learning algorithm applied in Project 3 was DBSCAN which creates clusters by identifying the density of data points and how close the points are to each other. Points further away from central clusters are classified as outliers. This algorithm identifies groups of data points that are closely packed together. Applying PCA as a preprocessing step for DBSCAN showed tighter clusters of data points with a clearer identification of outliers. PCA enhances DBSCAN by effectively filtering out noise in the dataset and emphasizing more meaningful data points.

Project 3 demonstrated how combining PCA with unsupervised machine learning techniques like k-Means, agglomerative clustering, and DBSCAN can significantly boost their performance and creates clearer, more precise clusters. Each algorithm, when used with PCA, performed better, with more distinct groupings and less overlap, making the data analysis much more effective.

Conclusion

I worked on different stages of machine learning across the three projects, from data preparation to the application and analysis of supervised and unsupervised machine learning

techniques. This journey deepened my understanding of machine learning concepts. In Project 1, I learned that data quality is critical. Well-prepared data is vital and sets the foundation for algorithm application. Additionally, preforming exploratory data analysis was instrumental in gaining insights and familiarity with the dataset before modeling. Through this process I was able to understand how preparing, cleaning and exploratory analysis can help with understanding which algorithms may be applied and what to be aware of in subsequent steps.

Supervised learning techniques applied in Project 2 provided insights on the importance of selecting appropriate algorithms tailored to specific data and tasks. While working with these algorithms, the importance of fine-tuning a model's parameters and reassessing the model's performance post-tuning was made clear. A takeaway from my review was the need for an additional evaluation of parameter options through data visualization and to check the accuracy of a combination of parameter adjustments.

In Project 3, I became familiar with unsupervised learning techniques using clustering algorithms. Applying PCA demonstrated how complex datasets could be simplified, enhancing data visualization and interpretation. Visualizing the data helped to better understand theoretical concepts, particularly through dendrograms and cluster plots, which highlighted the significant impact of PCA.

Overall, these projects provided a comprehensive experience where I gained foundational knowledge on applying machine learning models. This work helped to transform abstract concepts into practical application and made it clear how machine learning can be used in data analysis.

References

Devaney, J. (2024). *Class 5: k-Nearest Neighbors and Linear Models*. Lecture.

Géron, A. (2017). *Hands-on machine learning with scikit-learn and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. O'Reilly Media.

Müller, A. C., & Guido, S. (2018). *Introduction to machine learning with python: A guide for data scientists*. O'Reilly Media, Inc.