Nate Rattner
DATA 71200 Advanced Data Analysis Methods – Summer 2024
Final Project

## Introduction

My projects in this course examined a dataset with estimated obesity levels in individuals from Mexico, Peru and Colombia, provided by the UC Irvine Machine Learning Repository. The dataset has 2,111 records, with 17 features including physical attributes and eating and exercise habits.

I chose this dataset to gain experience with classification machine learning tasks, which I believe can help me in my work as a visual and data journalist looking for creative ways to find stories by identifying patterns in data. As an example of applying machine-learning methods to journalism, here is how reporter Peter Aldhous described his 2017 investigation into covert spy planes in American airspace at BuzzFeed:

> *"These discoveries came not from tip-offs from anonymous sources, but by training a computer to recognize known spy planes, then setting it loose on large quantities of flight-tracking data compiled by the website Flightradar24."*

Aldhous used machine learning to identify flight patterns that resembled those operated by the FBI and the Department of Homeland Security. The data prep and feature engineering he describes is much more complex than what I tackled this summer, but the core principles of supervised machine learning are similar. And while the datasets I work with are not always large or structured enough to justify machine-learning methods, I am now armed with more knowledge about how to approach a task like this when the opportunity arises.

Across my three projects, I am working to predict the obesity level of an individual based on the features described above. The target variable, "NObeyesdad," includes seven obesity level categories: Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II, and Obesity Type III.

## Data cleaning and preparation

The UCI “Estimation of Obesity Levels” dataset does not have any missing values, and therefore did not require any cleaning or filtering.

The dataset does, however, contain a handful of categorical variables, such as the individual’s gender, whether they eat food between meals (sometimes, frequently, always, never), how often they drink alcohol (sometimes, frequently, always, never) and which mode of transportation they typically use (public transportation, automobile, walking, motorbike, bike).

To deal with this, I performed one-hot encoding on the dataset to convert the categorical features into a format that machine-learning algorithms can work with. For each categorical feature, the updated dataset contained new columns corresponding to each category in its data; for example, the “MTRANS” categorical column containing information on methods of transportation became these five columns: MTRANS_Automobile, MTRANS_Bike, MTRANS_Motorbike, MTRANS_Public_Transportation and MTRANS_Walking.

For the target variable, which is also categorical, I applied scikit-learn’s LabelEncoder method to encode the target categories as integers instead of strings so that the target variable is in a format suitable for training machine learning models. LabelEncoder transforms labels that were in the target data into values from 0 through the number of categories minus 1; so in my case, the new target values were 0 through 6.

With this preparation done, I was ready to split the dataset into test and training sets. I used stratified sampling on the target variable to place an even distribution of target categories in each dataset which helps to avoid sampling bias.

## Visualizing the data

Across the dataset, the data within various features is distributed differently. Individuals' ages skew younger, for example, with more closer to 20 years old than 50; individuals' heights and weights follow a more normal distribution, however.

Visualizing the target attribute showed that it is fairly evenly distributed. Across the seven obesity types, no category makes up less than 12.9% or more than 16.6% of total categories.

This suggests a few encouraging implications for machine learning. It means there is less risk of overfitting, because there are ample examples across categories for the model to learn from. It also means that most machine learning models should be good fits for this data, and that I did not have to take any particular measures to account for imbalanced data. If one class were much more frequent than others, I might have had to use an evaluation metric that told me whether the model was performing better than just making the "most frequent" precision, but that is not a concern here.

In the real world, it is rare to have such a well-balanced dataset, and imbalanced data may be more of the norm. But for the purposes of an introduction to machine learning, using a balanced dataset is helpful to simplify the process.

## Supervised learning

I used two supervised machine learning models on the data: the k-nearest neighbors algorithm and a random forest.

The k-nearest neighbors algorithm is a simple model that makes predictions for new data points by finding the closest data points, or "nearest neighbors," in the training dataset. The algorithm can consider one nearest neighbor – in which case the prediction is equivalent simply to the closest training point – or any number of "k" neighbors. When using multiple neighbors, the algorithm uses the majority class from the nearest neighbors to assign a prediction value.

Run with the default parameters and k=3 number of neighbors, the KNN model produced an accuracy of 89.6% on the test set.

Using grid search, I iterated through k values of 1 through 24, building and testing a k-nearest neighbors model for each value. The best-performing k value appeared to be 1, which returned accuracy of 90.5% on the test set. This suggests that the points in the “Estimation of Obesity Levels” dataset are well-defined and distinct from one another, making it more accurate to classify using just one nearest neighbor rather than many. The worst-performing value was on the opposite end of the spectrum, at k=24, which returned accuracy of 79.4%.

I also adjusted the KNN method for distance computation, trying the "manhattan" metric to see how it compares to the default distance measurement, which is euclidean distance. A KNN model with k=1 and using the Manhattan distance parameter gave an accuracy of 92.4% on the test data, better than the k=1 model using Euclidean distance (90.5%) but worse than the best-performing KNN model of k=1 with the Manhattan distance parameter (92.4%).

A random forest model is a collection of decision trees, which are algorithms that use a ladder of if/else questions to arrive at a classification decision. Random forests are models that use a collection of decision trees that are each slightly different from one another and average these trees together in an attempt to reduce the amount of overfitting that an individual decision tree might be at risk of.

Using default parameters, with five “n_estimators,” or the number of decision trees, the random forest returned an accuracy of 88.7% on the test set, which was not as good as the best-performing KNN model (92.4%). Adjusting parameters improved the model, however. Using grid search, I tested “n_estimators” values of 100, 200 and 300, and “max_depth,” the maximum depth of the model’s decision trees, of none, 10, 20 and 30. The best-performing parameters were a max_depth of none and an n_estimators of 200, which returned an accuracy of 93.6% on the test set, meaning that the tuned random forest model performed the best of any supervised learning model I ran.

While the simple KNN model performed quite well, this process suggests that the more complex random forest model – with the proper parameter tuning – picked up on nuances in the data that the KNN algorithm did not incorporate. For example, visualizing feature importances from the random forest model showed that weight, a diet consisting of vegetables, age, height and daily number of meals consumed are the most significant in making predictions, and it is possible that leaning on these features to make decisions helped separate the random forest's performance from the simpler KNN model.

## Unsupervised learning

I ran principal component analysis on both unscaled and scaled versions of the obesity levels dataset. Using the unscaled data, only two components were required to explain 95% of variance in the data; with the scaled data, 20 features are required, showing how scaling makes the orders of magnitude more evenly distributed across all features.

Neither PCA experiment, however, improved the results of my best-performing supervised learning algorithm, the random forest with tuned parameters (93.6% accuracy on the test set). Using PCA for feature selection with two principal components returned an accuracy of 80.4% on the test set with the random forest, PCA with 20 principal components returned an accuracy of 82.5%. This improved performance with 20 components suggests that more than a couple of the features in the dataset are predictive of the target class; however, it appears that the random forest algorithm was able to better understand the even more-nuanced original data without PCA analysis.

Using PCA as a pre-processing step for clustering with the obesity levels dataset produced weak results. I used three unsupervised machine learning algorithms for this task: k-Means clustering, a simple approach which splits the data into a number "k" of distinct clusters by iteratively assigning data points to the nearest cluster; agglomerative clustering, which assigns points to clusters by merging similar clusters until the desired number of clusters is reached; and DBSCAN, which identifies points that crowd together in “dense” regions separated by empty space.

With ARI scores of 0.17, 0.10 and 0.08 across the three algorithms, respectively, the clusters formed were only slightly better than if clusters were assigned randomly. Silhouette coefficient scores of 0.24, 0.24 and -0.22 indicate that for the first two algorithms, the clusters were moderately but not very well separated; the negative score using DBSCAN suggests that some points were assigned to the incorrect clusters. Overall, pre-processing the data using PCA improved clustering results, but in general, the clustering results were not excellent.

Clustering results were much stronger using scikit-learn’s breast cancer dataset, with ARI scores as high as 0.67 for agglomerative clustering on PCA pre-processed data and a silhouette coefficient of 0.33 using k-Means clustering. When visualized, clusters appeared more well-defined when using each of the three algorithms.

I do not think my original dataset was inappropriate for clustering; rather, the nature of the data made clustering more challenging. The stronger scores in my second attempts indicate that the breast cancer dataset has more separable data points. With the scaled breast cancer dataset, 10 features explain 95% of variance, half as many as for the obesity dataset, which indicates that the breast cancer dataset is less complex and might contain less noise, both factors that could make clustering more effective.

## Conclusion

This project was a great learning opportunity for my first foray into machine learning. I think the dataset choice was appropriate – it was complex enough to help me learn how to prepare and format a dataset that models could consume, but not so complex that it required serious data cleaning or engineering in order to simply get it in shape to test supervised and unsupervised learning algorithms. I believe my process of one-hot encoding categorical variables was successful in getting the data into shape for fitting models. I wish I had spent more time, however, exploring and understanding the features and potentially calculating new ones. I could have calculated individuals' BMI, for instance, using the height and weight features, and this might have been a valuable input into the models. In general, however, I feel that data preparation went smoothly.

Revisiting the supervised learning models, I was able to successfully iterate on the algorithms' default settings to improve prediction scores. In hindsight, I should have spent more time tuning the random forest model. My best-performing model scored very well, but looking at it again it is clear that tuning parameters using grid search made a significant difference in improving performance; this process helped me understand the value of not only testing different machine learning models, but of the kind of "hyperparameter tuning" that can be done within each model. I spent more time doing this, I might have improved things even further. Grid search on multiple parameters for a random forest was computationally expensive, and it took a long time to iterate through the options. I wish I had explored ways to speed up that process, such as spending time with the most important parameters first (n_estimators and max_depth) before exploring other tuning options, or trying to run multiple parameter settings at once using the n_jobs parameter.

I think it also would have been worth testing other types of decision trees, such as a single decision tree and gradient boosted decision trees. Given the strong performance of the random forest model on this data, it would have been wise to attempt variations of this algorithm to see if there were any accuracy gains to be had.

Running PCA on the data – and the subsequent visualizations of various clustering algorithms – helped me better understand the structure of the dataset and the value of scaling data. For the most part, clusters were more visible using the scaled data that was pre-processed using PCA than when using the original data. Carrying out project three on the breast cancer dataset also helped me understand how clustering algorithms can perform better on some datasets than others. If doing project three over again, though, I would spend more time trying to improve the performance of the k-Means, agglomerative and DBSCAN clustering algorithms. There might have been features I could have removed, or default parameters I could have optimized, that would have improved performance the same way I was able to improve performance of the supervised machine learning models. It is possible that the dataset simply did not lend itself to clustering very well, but I could have done a better job of understanding whether the poor scores were due to the nature of the dataset or due to my unsupervised machine learning approaches.