Oscar Ko

Data 71200

Prof. Devaney

The Data

I selected a dataset for classifying a diabetes outcome. I chose this data because I thought it would be good for classification and unsupervised learning techniques since the sample size (n = 614) was sufficiently large with plenty of features. I found this dataset on *Kaggle*, which was originally from the *National Institute of Diabetes and Digestive and Kidney Diseases*. The cases were limited to females at least 21 years of age and Pima Indian heritage. The features include the number of times pregnant, glucose concentration, blood pressure, skin thickness, insulin concentration, BMI, and age. “DiabetesPedigreeFunction” was also a feature in this dataset, which includes scores derived from calculating the likelihood of diabetes based on family history.

Data Cleaning

When I first used the “isnull” function to search for missing values, none were detected, but after visualizing the data with histograms, I found that some patients had values of 0 for glucose, blood pressure, and BMI. This is unrealistic because to have measurements of 0 in any of these categories would mean the patient is dead. When checking the number of these cases, there were only 38 within the training set, so I removed all of these rows from the training and test set. There were also 182 “zero values” for the “SkinThickness" feature in the training set alone. Case removal would have greatly decreased the sample size. For this reason, I decided to remove the “SkinThickness” column instead.

Visualizing the Data

Aside from discovering the 0 values, I learned a few more things with visualization. With a correlation matrix, I discovered all the features in the cleaned dataset had some positive correlation with the diabetes outcome. Most of them were fairly weak correlations with the two strongest correlations being Glucose (r =0.48) and BMI (r = 0.28).

I gained a few insights using histograms as well. The number of pregnancies that most subjects had were much closer to zero, and as pregnancies increased, the frequency of cases decreased. Most cases in this dataset were closer to the minimum acceptable age of 21, and as the ages increased, the frequency of cases decreased. The histogram for blood pressure matched the shape of a Gaussian distribution.

When visualizing the outcome variable with a bar chart, it showed that there were twice as many negative outcomes as positive outcomes.

Supervised Learning Algorithms

In experimenting with supervised learning, I chose K-Nearest Neighbors and Support Vector Machines.

K-Nearest Neighbors

For K-Nearest Neighbors, the algorithm classifies new data points using a voting system. The neighboring cases with the closest values to the new data point are treated as the voters. Each neighboring case will contribute a vote for the class that it is in. The class with the most votes will be assigned to the new data point. The number of neighboring data points considered is based on a parameter “k” that the user decides.

The parameters I optimized using grid search are n_neighbors, metric, algorithm, and weights. “N_neighbors” is the number of neighbors considered for voting. “Metric” is the type of distance used when measuring which data points count as the nearest neighbors. “Algorithm” selects how KNN will compute the nearest neighbors. “Weight” decides the voting power each neighbor will have. If the selected weight parameter is “uniform,” all neighbors will have the same voting power. If the selected weight parameter is “distance,” the closer the neighbor is to the new data point, the more its vote counts.

Support Vector Machines

In binary classification, linear SVM finds a line between the two classes that creates the maximum margins between the two groups. With the kernel trick, SVM approaches the data from higher dimensions and creates a hyperplane that can separate the groups with maximum margins. When the results are projected back onto a 2D plane, the decision boundary will not be linear.

The parameters I optimized with grid search are C, gamma, and kernel. “C” is the regularization parameter. It is inversely proportional to the strength of how much the model regularizes the coefficients on the features. “Gamma” is inversely proportional to the width of the RBF kernel. The “kernel” parameter selects the type of kernel that the algorithm will use such as linear or RBF.

Comparing Models

Optimizing the parameters for KNN with grid search helped improve the model. The mean cross-validation score on the training set increased from 0.73 to 0.74, the accuracy score on the test set increased from 0.71 to 0.76, the macro f1 score increased from 0.65 to 0.72, and the weighted recall score increased from 0.65 to 0.71.

My focus is on recall instead of precision because, in a scenario where the goal is to predict a patient's risk for diabetes, it could be detrimental if diabetes were undiagnosed and went untreated. When we want to minimize false negatives, it is best to focus on recall which is the proportion of all positive cases the model managed to capture.

Optimizing the parameters for SVM with grid search did not improve all the evaluation scores. It only improved the mean cross-validation score on the training set from 0.78 to 0.79. Everything else slightly decreased: the accuracy score on the test set decreased from 0.72 to 0.70, the macro f1 score decreased from 0.68 to 0.65, and the macro recall score decreased from 0.68 to 0.65.

It seems SVM did worse with the optimized parameters. Maybe by optimizing the parameters the model was also slightly overfitted. Since SVM is already a complex algorithm, it already performed okay at the beginning, so optimizing the parameters might have caused the model to become slightly overfitted. The default kernel for SVM is RBF, and the optimized model had a linear kernel. That may have made a big difference. Also, by default, "C" is 1, and the optimized SVM had a C of 100. Using a higher C value results in less regularized coefficients, and the algorithm attempts to correctly classify as many individual points as possible. Higher C's can also lead to overfitting.

The optimized KNN model performed the best of all four models. KNN focuses on the closest neighbors for an individual incoming data point. Maybe this is why KNN performed slightly better after optimizing. While people with diabetes can share similar characteristics, they might have some differences between them like Type I vs Type II. Speculatively, an incoming data point that has Type II diabetes might land close to the values of other data points that have Type II diabetes but further away from Type I cases and non-diabetic cases. KNN focuses on the cases that are relatively closest to the new data point, which can help deal with the uniqueness of individual diabetes types. In contrast, linear SVM tries to create a decision boundary dividing the most possible data points where it treats all positive diabetes cases as generally the same.

PCA for Feature Selection

PCA for feature selection did not improve the performance of the optimized K-Nearest Neighbors model. KNN with optimized parameters and PCA feature selection increased the mean cross-validation score from 0.74 to 0.76, but for all the other metrics it declined in performance: accuracy in the test set decreased from 0.76 to 0.74, the macro f1 score decreased from 0.72 to 0.69, and the macro recall score decreased from 0.71 to 0.69. This decrease in performance might be due to the KNN parameters being optimized for the individual features as opposed to the principal components. Doing the same comparison with default parameters might yield different results.

Unsupervised Algorithms with PCA

Algorithm Descriptions

In a brief description, the K-Means algorithm starts with a “K” number of clusters, where K is a number selected by the user. It will go through a series of iterations where it starts with K cluster center points at random locations. Each data point will be grouped into the nearest cluster. In the next iteration, new cluster centers are created from the center of all the grouped data points. Each data point will be reassigned to the nearest new cluster centers. The algorithm will keep iterating through these steps until there are no more changes in the clustering of the data points.

The agglomerative clustering algorithm starts with every data point in its own cluster, and then it goes through a series of iterations. In each iteration, it will group two clusters based on the linkage type specified. It will keep iterating until the number of clusters has gone down to the number of clusters that the user specified.

For DBSCAN, the user specifies “eps,” the distance that the algorithm should look for data points, and, “min_samples,” the minimum number of data points that it needs to find within that specified range to declare a cluster. With these two parameters, this algorithm finds clusters based on how densely packed data points are. Data points that are farther out or in sparse areas will not be clustered and will be considered “noise.”

Results Comparison

K-means

Before using K-means on the data, I had to decide how many clusters to use and which features to visualize. Using the elbow method, I decided on four clusters. I chose to use Glucose and BMI for visualization as they are the two highest correlated features with the diabetes outcome.

Without PCA, K-means divided the data into two overlapping clusters on the right side of the plot and two overlapping clusters on the left side of the plot. As the clusters on the right side have higher values of glucose, they have higher chances of diabetes due to the positive correlation. In addition, two of the clusters have slightly higher BMI values. This could mean those two clusters also have higher chances of diabetes than their counterparts as BMI is positively correlated with diabetes.

With PCA, K-means separated the clusters into four directions. There was a top cluster, right cluster, left cluster, and bottom cluster. The first component accounts for the most variability in PCA, so it is the most valuable for interpretation. Examining the heatmap of PCA components that I created shows that all the features in the first component have a positive sign, so there is some general correlation with all of them. All the features will generally increase as we go to the right side of the first component's axis. The closer a cluster is to the right side, the higher the chances of diabetes they might have. For the second component, the heatmap shows that Glucose, Insulin, BMI, and the DiabetesPedigreeFunction have an opposite sign to Pregnancies and Age, so if Glucose, Insulin, and DiabetesPedigreeFunction increased, then Pregnancies and Age would decrease. Speculatively, the second component might be a great way to distinguish cases of younger individuals with genetic type II diabetes because they would likely be cases with higher values of Glucose, Insulin, and DiabetesPedigreeFunction while having lower values of Pregnancies and Age. The top cluster might be these cases of younger type II diabetics.

Agglomerative Clustering

The results for agglomerative clustering without PCA were that of K-Means. There were two overlapping clusters on the right and two overlapping clusters on the left. With higher glucose levels, the clusters to the right are likely to have higher chances of diabetes as they are positively correlated. Some clusters had slightly higher BMI and possibly higher chances of diabetes as they are positively correlated.

With PCA, the agglomerative clustering's results were, again, similar to the results of K-means. There was one cluster in each of the four directions: top, right, left, and bottom. Since the first component has all the features with some general correlation, I speculate that clusters closer to the right side have a higher likelihood of diabetes.

The similar results with K-Means may be due to their algorithms both creating clusters based on distance.

DBSCAN

Whether it was with or without PCA, DBSCAN had a lot of issues with this dataset. No matter how the parameters “min_samples” and “eps” were adjusted, there were always these issues involved: one cluster was too large and hogged most of the data points, a few clusters were too small, and way too many data points were considered noise.

These problems might be due to DBSCAN assigning clusters based on density. It assumes that all clusters will have the same density in their data points, but for this dataset, the cases with higher glucose and higher BMI were sparse, so DBSCAN considered them as noise. The lower density of the high glucose cases may be caused by there being twice as many negative cases compared to the positive cases. (That also results in the higher density of the low glucose cases.)

ARI and Silhouette Coefficient

When evaluating the three algorithms with the Adjusted Rand Index, DBSCAN (ARI = 0.19) and KMeans (ARI = 0.18) performed the best. It was surprising that DBSCAN had the highest score when it classified a large chunk of the data as noise. I wonder if the Adjusted Rand Index considers “noise” as its own cluster, which might help explain the higher ARI score. Overall, the ARI scores for all the algorithms were pretty low.

For the Silhouette Coefficient, KMeans (SC = 0.21) and AgglomerativeClustering (SC = 0.17) performed the best. It seems KMeans performed the best overall when judging with both evaluation methods. In general, PCA made no difference in any of the scores.

Summary

From project 1, I learned how to perform a stratified split on the data for balanced training and testing sets. Visualizing with histograms helped me spot cases of 0 values that I would have missed if I only relied on the “isnull” method. Visualizing a correlation matrix on a heatmap helped me see which features had the highest correlation with diabetes.

From project 2, I compared K-Nearest Neighbors and SVM supervised learning algorithms. Grid search worked wonderfully with K-Nearest Neighbors, and there was a score improvement, which made its performance surpass the SVM model. Using a combination of cross-validation scores, confusion matrix, and evaluation metrics helped me understand how the models were performing.

From project 3, I got to practice with unsupervised algorithms. Visualizing PCA on a heatmap helped me understand how I might interpret the components. When using K-Means and agglomerative clustering on the PCA components, it produced neat and distinct clusters. I also got a better sense of how the density clustering methods of DBSCAN work and what kinds of datasets might be better for the algorithm.

If I were to do these projects again, I would like to use ARI earlier on to evaluate if I had chosen the best number of clusters with the elbow method. While I chose four clusters with the elbow method, when I experimented with the ARI and silhouette coefficients, I got slightly higher scores with three clusters.