Lifestyle and Demographic Predictors of Diabetes: An Analysis of NHIS Survey Data

Tyler Franck

https://github.com/senaisle/DATA-5322

Introduction

We explore how predictive basic demographics, health metrics, and lifestyle are of diabetes in (Western U.S.) adults using data collected by the National Health Interview Survey and accessed through IPUMS Health Survey. The dataset and detailed codebook can be found here. As an overview, the dataset contains 48 variables, which can be generally be organized into one of four categories: **(1) survey information**, **(2) demographic information**, **(3) disease indicators**, and **(4) basic health metrics and habits**. The survey information variables are ID numbers and sample weights from the survey methodology, which we will largely ignore.

It is important to note that while this dataset may be useful in predicting whether someone *has* a disease, but it is unclear whether it could also be used to help predict whether someone will *develop* a disease. For example, frequent consumption of sugary drinks might be correlated with developing diabetes, but here we might be more likely to see low consumption of sugary drinks correlated with diabetes because people with diabetes have to be a lot more careful about their blood sugar.

As mentioned before, we are interested in predicting diabetes in adults in the Western region of the United States using basic demographics, health metrics, and lifestyle. More specifically, we will try to predict **DIABETICEV** (whether they have diabetes) using **support vector machines** with the following predictors:

- **AGE**: age
- **SEX**: sex
- **BMICALC**: body mass index
- **HRSLEEP**: usual number of hours of sleep per day
- **MOD10DMIN**, **VIG10DMIN**: duration of moderate/vigorous activity in minutes
- **JUICEMNO**: frequency of drinking fruit juice in past month

Theoretical Background

The Maximal Margin Classifier

When classes are linearly separable, there are generally an *infinite* number of separating hyperplanes that can do the job, which means that we must have a reasonable way to choose which of the candidate hyperplanes to use. Since the **margin** between an observation and the decision boundary is an indicator of classification confidence, then a natural choice is the **maximal margin hyperplane**, which is the hyperplane that has the largest "gap" between itself and the data it is separating. The **maximal margin classifier** simply classifies observations based on which side of the maximal margin hyperplane they're on.

Construction of the Maximal Margin Classifier

The maximal margin hyperplane is given by the following optimization problem:

$$\begin{aligned} &\underset{\beta_0,\ldots,\beta_p,M}{\text{maximize}} \; M \\ &\text{subject to } \sum_{i=1}^{p} \beta_i^2 = 1 \\ &\qquad y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \end{aligned}$$

The maximal margin classifier predicts $y^* = \text{sign}(\beta_0 + \vec{\beta}^\top \vec{x}^*)$.

Theoretical Background (Cont.)

The problem with maximal margin classifiers is that they only work if the data can be *perfectly* linearly separated: you just can't use a maximal margin classifier if there is no separating hyperplane. In such cases, you must either choose to (1) ignore some observations, or (2) use a non-linear decision boundary.

Support Vector Classifiers: Ignore Some Observations

Even when a separating hyperplane *does* exist, we might prefer a classifier that *doesn't* perfectly separate the classes in exchange for (1) more robustness to individual observations and (2) better classification of *most* training observations. The **support vector classifier** does this by incorporating a **soft margin**: rather than *enforce* that all observations be on the correct side of the margin, we allow some observations to be on the wrong side of the margin and even the wrong side of the hyperplane but *penalize* such violations (up to some *budget* C).

Construction of the Support Vector Classifier

The support vector classifier is given by the following optimization problem:

$$\begin{aligned} &\underset{\beta_0,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{maximize}} \; M \\ &\text{subject to } \sum_{i=1}^{p} \beta_i^2 = 1 \\ &\qquad y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M \\ &\qquad \epsilon_i \geq 0 \\ &\qquad \sum_{i=1}^{n} \epsilon_i \leq C \end{aligned}$$

The support vector classifier predicts $y^* = \text{sign}(\beta_0 + \vec{\beta}^\top \vec{x}^*)$.

Support Vector Machines: Don't Use a Line

Support vector classifiers can be extended to support non-linearity in much the same way that linear regression can: through the inclusion of non-linear features. The idea is to lift the data into a higher-dimensional space where the data *is* linearly separable. The **support vector machine** is an extension of the support vector classifier that results from enlarging the feature space using **kernels**.

It turns out that the support vector classifier problem can be solved using only the *inner products* of the observations and, furthermore, only those of the **support vectors** (the observations that bound their classes). A kernel is a generalization of the inner product that quantifies the similarity/distance between two observations. The general equation for the support vector machine is: $f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(\vec{x}, \vec{x}_i)$.

Kernels *implicitly* map data into a higher-dimensional space without *explicitly* computing the coordinates, which is useful when the enlarged feature space is very large or even infinite. Choosing the kernel is essentially equivalent to choosing the higher-dimensional space:

- **Linear Kernel:** $K(\vec{x}, \vec{x}_i) = \vec{x}_i^\top \vec{x}$
- **Polynomial Kernel:** $K(\vec{x}, \vec{x}_i) = \left(1 + \vec{x}_i^\top \vec{x}\right)^d$
- **Radial Kernel:** $K(\vec{x}, \vec{x}_i) = e^{-\gamma \|\vec{x} - \vec{x}_i\|_2^2}$

Using the linear kernel is equivalent to using a support vector classifier. Using a polynomial kernel of degree d amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d (increasing d leads to more inflection points in the decision boundary). The radial kernel exaggerates the Euclidean distance between points so that only very *local* points matter; using a radial kernel allows for more "circular" decision boundaries (γ tunes the "spheres of influence" by scaling distance fall-off). All the kernels must consider the tuning parameter C as well.

Methodology

We first filtered the dataset to only include the parts we're actually interested in. In particular, we only included rows pertaining to *adults* in the *Western* region of the United States, and we only included the columns discussed in the introduction:

- **DIABETICEV**: indicator for diabetes
- **AGE**: age
- **SEX**: sex
- **BMICALC**: body mass index
- **HRSLEEP**: usual number of hours of sleep per day
- **MOD10DMIN**, **VIG10DMIN**: duration of moderate/vigorous activity in minutes
- **JUICEMNO**: frequency of drinking fruit juice in past month

Then we randomly split the dataset into a training and test set using a 50%-50% split.

Once we loaded, cleaned, and split the data, we then turned towards data analysis. For each of the three kernels—linear, polynomial, and radial—we use cross-validation to tune the parameters and then evaluate model performance using training and test accuracy and ROC curves. The details of the hyperparameter search are as follows:

- For the linear kernel, the only tuning parameter is C. We cross-validated the values $C \in \{0.01, 0.1, 1, 5, 10\}$.
- For the polynomial kernel, the tuning parameters are C and d. We cross-validated the grid $(C, d) \in \{0.1, 1, 10, 100, 1000\} \times \{2, 3, 4\}$
- For the radial kernel, the tuning parameters are C and γ. We cross-validated the grid $(C, \gamma) \in 10^{\{-2,\ldots,5\} \times \{-4,\ldots,1\}}$

Results

Linear

Grid search failed to find a setting of C that performed better than the rest (they all performed *exactly* the same). I ended up choosing the largest value $C = 10$, and the resulting model just predicted the most common response on the training and test set (no diabetes). That being said, SVMs aren't decision trees—they can't just *choose* to predict the most common response. Indeed, the AUC of the ROC curve being 0.65 and 0.64 for the training and test set, respectively, suggests that the model *is* capturing some signal in the data, just not enough to actual identify positive cases in the dataset.

Polynomial

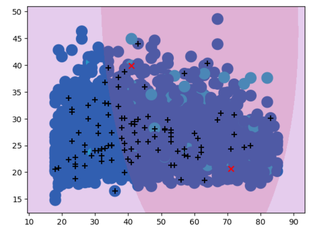
Unlike with the linear kernel, some of the settings actually did worse than others. Grid search found $C = 1000$, $d = 2$ as *one of* the best models. Unfortunately, the constructed model just predicted the most common response like the linear model, and the AUC of its ROC curve was actually worse than that of the linear model (0.7 on the training set, 0.61 on the test set). The AUC values suggest that the polynomial model is fitting the data much tighter than the linear model, though there ends up being no difference in the predictions.

Radial

The radial kernel also had some settings better than others, but one of the best settings $C = 0.1$, $\gamma = 0.0001$ still led to a model that only predicts the most common response. If you actually look at the confusion tables of some of the "worse" settings, you'll see that the reason they perform worse is because they're actually trying to identify diabetes. The problem is that they overfit the training set and don't do a good job on the test set. Regardless, the fact that the radial kernel *can* overfit and the AUC of the best model's ROC curve was better than both the polynomial and linear models, suggest that it is the best suited model.

Conclusion

All in all, our models didn't *predict* anything useful or interesting. We can try to look some plots, but the multi-dimensionality and density of the data makes it hard to extract insights from them. For example, the following plot shows the "decision boundary" of the radial model the dimensions of age and BMI, which *might* suggest that higher age and BMI correlate with diabetes, but it's hard to tell.



The radial kernel was the most promising, yielding some models that overfit the training data, but this predictive power didn't generalize to the test set. Perhaps we could test a larger grid of hyperparameters, or perhaps we need more data (more rows or more columns). Another option, if we're interested in *inference*, would be to use decision trees or a logistical model rather than an SVM.

References

1. Lynn A. Blewett, Julia A. Rivera Drew, Miriam L. King, Kari C.W. Williams, Daniel Backman, Annie Chen, and Stephanie Richards. IPUMS Health Surveys: National Health Interview Survey, Version 7.4 [dataset]. Minneapolis, MN: IPUMS, 2024. https://doi.org/10.18128/D070.V7.4.