Jacqueline Söegaard
15.097 –Prediction: Machine Learning and Statistics
Prof. Cynthia Rudin
Course Project
May 17, 2012

# Improving Tools for Medical Statistics

Comparing the performance of propensity scores and support vector machines at estimating the effect of treatment on outcomes in observational data


## Abstract

Propensity scores have become a popular method for removing bias in the estimation of treatment effect when working with observational data. However, there are many issues and limitations associated with propensity scores. This project compares the performance of pairs of support vector machines (SVMs) as an alternative, and aims to evaluate whether these paired SVMs can perform better than propensity scores when estimating the effect of treatment on outcomes in non-randomized data. Both problems were evaluated using synthetically generated datasets that modeled 20 different models for the degree of the relationships between baseline variables, treatment, and outcomes. Although neither method performed particularly well at recovering the true treatment effect, it was possible to learn when the different methods fail by looking at the data for the different scenarios.

## Table of Contents



## Index of Equations



## Index of Figures



## Index of Tables

## 1. Introduction

Most medical studies seek to determine the effect of a certain intervention or treatment on clinical outcomes. One measure is the average treatment effect (ATE), which is the effect of moving a population from being untreated to being treated. Considering the case where $i$ patients, each with a set of $X_i$ baseline variables, are each assigned to either the treatment group, $Z_i = 1$, or the control group, $Z_i = 0$, the average treatment effect is:

$$ATE = E[Y_i(Z_i = 1) - Y_i(Z_i = 0)] \quad (1)$$

Ideally, to measure the ATE, one conducts a randomized controlled trial (RCT). In the RCT experimental setup, experimenters randomly assign subjects to either the treatment or the control group. This eliminates the confounding effect of measured and unmeasured baseline variables on the treatment status.[1]

Unfortunately, it is often either infeasible or unethical to conduct a RCT. For example, consider the case where the treatment under evaluation is a surgical intervention. Due to the invasive nature of the treatment, one would only assign patients to the treatment group if their medical condition necessitated the surgery. Another example would be a study of the effect of psychotherapy on drug addiction. Again, the treatment group would be composed of people who are already addicted to the drug, because it would be unethical to expose other individuals to addictive, harmful, and often illegal substances. In both these scenarios, an individual's treatment group assignment is now conditionally dependent on their past history and baseline variables (Figure 1).

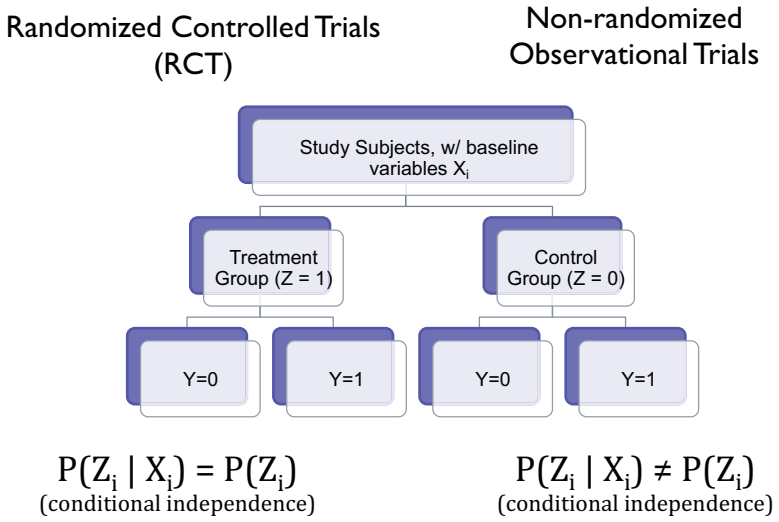


**Figure 1: Comparison of randomized and observational clinical trials.** Note how treatment assignment $Z$ is conditionally independent and dependent of the baseline covariates $X_i$ for randomized and observational studies, respectively.

Thinking back on the surgical study example, one might imagine that only the sickest patients would be given the invasive surgery. In the case where the treatment group has a much lower rate of survival than the control group, it is difficult to determine whether the discrepancy in survival rates occurred because the surgery had an adverse effect on survival. The same discrepancy could have also arisen because the patients in the treatment group were inherently sicker than the patients in the control group. How, then, can the true effect of the

surgery on an outcome like survival be assessed? When working with non-randomized observational data, one must first remove the confounding effect of the baseline covariates $X_i$ on the association between the treatment $Z$ and the outcome $Y$ in order to determine the true treatment effect.

In their 1983 papers, Rosembaum and Rubin proposed using the method of propensity score analysis to reduce bias in observational studies.[2] They defined the propensity scores as the probability that a patient receives a treatment given the distribution baseline covariates. Since being introduced in 1983, propensity scores have become a widely used tool for the evaluation of non-randomized observational studies in medicine and surgery. Despite the method's growing popularity, however, there remain various disadvantages that might limit its usefulness and appropriateness. The inevitability and importance of observational studies makes it imperative that either these shortcomings are addressed, or that a better alternative be proposed. This project proposes using pairs of support vector machines (SVMs) as such an alternative, and aims to evaluate whether these paired SVMs can perform better than propensity score when estimating the effect of treatment on outcomes in non-randomized data.

## 2. Overview of Propensity Scores

### *2.a. Calculation and use as a balancing score*

A propensity score represents the probability of being assigned to the treatment group given a patient's baseline variables. Formally, this is represented as:

$$e_i = P(Z = 1|\boldsymbol{X}_i) \tag{2}$$

In Equation 2 $e_i$ is the propensity score, $Z$ is the binary treatment assignment, and $\boldsymbol{X_i}$ is the vector of baseline variables. The propensity score is most commonly calculated via logistic regression, which uses baseline variables to predict the probability between 0 and 1 that a person is assigned to the treatment group ($Z = 1$) as opposed to the control group ($Z = 0$).[3] These propensity scores can then be used as a balancing score that accounts for the confounding effect of the baseline variables so that the outcomes of the treatment and control groups can be compared. The underlying idea is that among subjects with the same propensity score, the same distribution of observed baseline variables will be the same between treated and untreated subjects. Three methods exist that help achieve this "balance" between the treatment and control groups: matching on the propensity score, stratification on the propensity score, and inverse probability of treatment weighting (IPTW) using the propensity score.[1]

Matching on the propensity score involves paring each treated subject to a control subject such that the both members of the matched pair have identical or highly similar values of the propensity scores. In contrast, stratification on the propensity score assigns subjects to one of various ranges, or strata, of the propensity score. For example, the subjects could be classified into the quintiles of the propensity score. Once the propensity scores have been used either to match or to stratify patients, the effect of treatment on outcome can be estimated as the difference in the fraction of patients experiencing a certain outcome in the treatment group versus in the control groups in within the pairs or strata.[1,4] Inverse probability of treatment weighting (IPTW) differs from either matching or stratification in that IPTW uses the propensity score to weight the treatment and control subjects so that the final distribution of

baseline covariates is independent of treatment assignment.[1,5] The ATE can then be estimated using these weights as described in section 4.b.

### *2.b. Issues and limitations*

Despite the prevalent use and growing popularity of propensity scores, several limitations and issues exist regarding their implementation and calculation. One of the major problems regards the assumption of strongly ignorable treatment assignment described by Rosembaum and Rubin in their original paper. They defined treatment assignment to be "strongly ignorable" if (a) the treatment assignment $Z$ is independent of the outcome Y given the baseline variables, and (b) any subject has a nonzero probability of being assigned to either treatment. For propensity score methods to be admissible for the unbiased estimation of treatment effects, the assumption of strongly ignorable treatment assignment described must be satisfied. [1,2] However, some users overlook this requirement when using propensity scores, and instead "interpret the mathematical proof of Rosembaum and Rubin as a guarantee that, in each strata of the [propensity score], matching treated and untreated subjects somehow eliminates confounding from the data and contributes therefore to overall bias reduction." [6]

Other problems stems from the prevailing use of the logistic regression to calculate propensity scores. The popularity of logistic regression for this purpose stems from its convenient generation of probability values within the range [0,1] and from its accessibility and familiarity to users. However, the requirements of logistic regression create the inconvenience that, for studies involving rare outcomes, there needs to be a high number of $Y = 1$ events per baseline variable to avoid imbalance in the data. When working with high-dimensional data with many baseline variables, the minimum number of $Y = 1$ events could become prohibitively large.[4] More problematic is many users' failure to account for the underlying assumptions of proper logistic regression modeling, including that of the linearity of the risk with respect to the log-odds parametric transformation. It also seems that the common implementation of the logistic regression fails to include higher order and interaction terms that would be necessary for accurate model fit.[3] To eliminate these issues, Westreich et. al. suggested that machine learning methods that make fewer assumptions might be suitable alternatives for logistic regression in the calculation of propensity scores. Of the methods they evaluated, they concluded that boosting, a method whereby many weak classifiers are combined to make a strong classifier, showed the most promise.

## 3. Experimental Methodology

The propensity scores and support vector machines were calculated using synthetic data datasets. Their performances were then evaluated using average treatment effect estimation based on the inverse probability of treatment weighting, and the net reclassification index, respectively.

### *3.a. Synthetic data generation*

Synthetic datasets were used for experimentation so that the true effect of the treatment variable on the outcome variable would be known. This then allow for the adequate assessment of whether the performance of propensity scores or support vector machines had closely estimated the true effect of treatment on outcomes. The approach for synthetic dataset

generation was based from the framework outlined by Setoguchi et. al.[7] To summarize their approach:

- They generated 10 baseline covariates from independent standards normal random variables whose relationships were defined by a correlation matrix. Four of the variables were left as continuous variables, and 6 were binarized. Of the 10 variables, 3 were only associated with the treatment assignment only, 3 were only associated with the outcome only, and four were associated with both the treatment assignment and the outcome.
- They used a fixed set of covariate coefficients to model 7 different scenarios with different degrees of non-linearity and non-additivity in the associations between the covariates and the treatment assignment. These models generated true propensity scores, from which treatment labels $Z$ were derived using Monte-Carlo simulations.
- Their outcome was modeled using logistic regression as a function of the baseline covariates and the treatment labels, where the coefficient for the treatment label was the true effect of treatment $\gamma = -0.4$. Once more, these models generated outcome probabilities, from which outcome labels $Y$ were derived using Monte-Carlo simulations.

Their synthetic generation framework was adapted to the needs of this study. The methodology for generating the covariates was emulated exactly. However, rather than using the 7 variable-treatment relationship scenarios, only Setoguchi's first scenario was used. In this scenario, the true propensity score given the covariates, $P(Z = 1|X_i)$, was modeled by a linear and additive relationship between the covariates:

$$P(Z = 1|X_i) = (1 + \exp\{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7)\})^{-1} \quad (3)$$

After obtaining the treatment labels $Z$ from the Monte-Carlo simulations, the outcome was modeled using the following equation:

$$P(Y = 1|Z, X_i) = (1 + \exp\{-(\alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3 + \alpha_4 X_4 + \alpha_5 X_8 + \alpha_6 X_9 + \beta_7 X_{10} + \gamma_1 Z)\})^{-1} \quad (4)$$

Instead of using Setoguchi et. al's coefficients, I created 20 different scenarios for the covariate coefficients. These scenarios use scenarios use extremes of the coefficients to study the intuitive effect of having low/high effect of covariates on probability of treatment assignment; low/high correlation between the baseline covariates and outcome; and varying degrees of effect of treatment on outcome. This would help to determine whether propensity scores or SVMs would fail to estimate treatment effect in any of these scenarios. The combinations of coefficients are described in Figure 2.

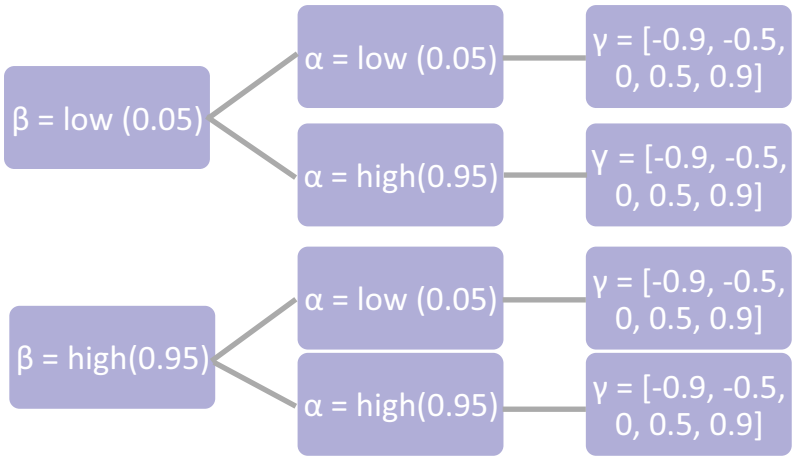


**Figure 2: Coefficient combinations for the generation of 20 datasets.** The betas correspond to the coefficients in true propensity score model (equation 3) that specify the effect of the covariates on the probability of treatment assignment. Similarly, the alphas correspond to the coefficients for the covariates in the outcome model (equation 4) that specify the effect of those covariates on the outcome probabilities. Within each scenario, all of the alpha coefficients will be the same; the same is true for the beta coefficients. The alphas and betas can either take on low or high values. Finally, the gammas correspond to the coefficient for the treatment label Z in the outcome model (equation 3) that specifies effect of treatment on outcome. This coefficient can take on any value within the set {-0.9, -0.5, 0, 0.5, 0.9}.

All of the above steps were followed to create a 20 dataset, each with n = 20,000 data points. The information and composition of each dataset are outlined in Table 1.

**Table 1: Synthetic datasets profiles.**

| Scenario # | Scenario ID | #Patients Receiving Treatment (Z=1) | %Patients Receiving Treatment (Z=1) | True Treatment Effect | #Patients with Outcome Y= 1 | % Patients with Outcome Y =1 |
|---|---|---|---|---|---|---|
| **1** | beta = low/ alpha = low | 10489 | 52% | -0.9 | 317 | 2% |
| **2** | beta = low/ alpha = high | 10489 | 52% | -0.9 | 357 | 2% |
| **3** | beta = high/ alpha = low | 14696 | 73% | -0.9 | 485 | 2% |
| **4** | beta = high/ alpha = high | 14696 | 73% | -0.9 | 634 | 3% |
| **5** | beta = low/ alpha = low | 10489 | 52% | -0.5 | 782 | 4% |
| **6** | beta = low/ alpha = high | 10489 | 52% | -0.5 | 4377 | 22% |
| **7** | beta = high/ alpha = low | 14696 | 73% | -0.5 | 4685 | 23% |
| **8** | beta = high/ alpha = high | 14696 | 73% | -0.5 | 5259 | 26% |
| **9** | beta = low/ alpha = low | 10489 | 52% | 0 | 5904 | 30% |
| **10** | beta = low/ alpha = high | 10489 | 52% | 0 | 6398 | 32% |
| **11** | beta = high/ alpha = low | 14696 | 73% | 0 | 250 | 1% |
| **12** | beta = high/ alpha = high | 14696 | 73% | 0 | 344 | 2% |
| **13** | beta = low/ alpha = low | 10489 | 52% | 0.5 | 486 | 2% |
| **14** | beta = low/ alpha = high | 10489 | 52% | 0.5 | 665 | 3% |
| **15** | beta = high/ alpha = low | 14696 | 73% | 0.5 | 1012 | 5% |
| **16** | beta = high/ alpha = high | 14696 | 73% | 0.5 | 3663 | 18% |
| **17** | beta = low/ alpha = low | 10489 | 52% | 0.9 | 4354 | 22% |
| **18** | beta = low/ alpha = high | 10489 | 52% | 0.9 | 5260 | 26% |

| | | | | | | |
|---|---|---|---|---|---|---|
| **19** | beta = high/ alpha = low | 14696 | 73% | 0.9 | 6199 | 31% |
| **20** | beta = high/ alpha = high | 14696 | 73% | 0.9 | 7079 | 35% |

### *3.b. Propensity scores and average treatment effect*

After considering the limitations of logistic regression for the calculation of propensity scores, the propensity scores were instead calculated using boosting. Specifically, R's AdaBoost function, ada, was applied to each data example to obtain it's probability of treatment class membership, i.e. $P(Y_i = 1|\boldsymbol{X}_i)$. This value is equivalent to the propensity score $e_i$. The propensity scores were then used to estimate the average treatment effect (ATE) using the following formula for inverse probability of treatment weighting [1]:

$$ATE \sim \frac{1}{n}\sum_{i=1}^{n}\frac{Z_i Y_i}{e_i} - \frac{1}{n}\sum_{i=1}^{n}\frac{(1-Z_i)Y_i}{1-e_i} \quad (5)$$

This formula uses the propensity score to give higher weight to the data examples that were assigned to the control group despite having a high propensity score, or vice versa. Once estimated, the ATE was then compared to the known treatment effect to calculate the percent bias of the estimator.

### *3.c. Support vector machines and the net reclassification index*

To determine the effect of treatment on outcome, the performance of pairs of SVMs at predicting outcomes from the data was compared. The framework for the training and testing the SVMs was as follows:

- One of the SVMs in the pair was trained to predict outcomes using solely the baseline covariates as features, whereas the second SVM was trained to predict outcomes using both the baseline covariates and the treatment labels as features. These will be called the "without-SVM" and the "with-SVM," respectively.
- Using the svm() function from the R e1071 package, a pair of SVMs was generated for each of the 21 scenarios described in the dataset generation section.
- After a preliminary evaluation of the performance of the algorithm on the "real" dataset, the SVM's cost parameter was set to 100. The radial basis (Gaussian) kernel function was used with gamma parameter left as the default value of $1/(data\ dimension)$.
- In each of these cases, the SVMs were trained on the first half of the data, and tested on the remaining half.

The general idea is that if the treatment affects outcomes, then the treatment labels contain information that is important for prediction. Therefore, adding the treatment labels to the group of features would be expected to improve the classifier's performance. To compare the two SVMs and learn how much information the treatment contributes to the prediction of outcome, the net reclassification improvement (NRI) metric was used. The NRI summarizes the reclassification table, which itself shows how many data examples would be reclassified into higher or lower risk or outcome categories upon the addition of new information into a prediction model. Once the reclassification table has been computed, the NRI can be

calculated as the difference in the proportions of data examples that are moving up and down between the treatment and control groups[8]:

$$NRI = [P(up|Z = 1) - P(down|Z = 1)] - [P(up|Z = 0) - P(down|Z = 0)] \quad (6)$$

For the case where the outcomes are $Y = \{1,0\}$, this expression can be rewritten as:

$$NRI = [P(Y = 0 \rightarrow 1|Z = 1) - P(Y = 1 \rightarrow 0|Z = 1)] - [P(Y = 0 \rightarrow 1|Z = 0) - P(Y = 1 \rightarrow 0|Z = 0)] \quad (7)$$

where the label on the left hand side of the arrow is that given by the "without-SVM," and the label on the right hand side of the arrow is that given by the "with-SVM."

## 4. Results

### *4.a. Propensity scores*

The results from the propensity score tests described in section 3.b. are summarized in Table 2. Overall, the known treatment effect from the data generation models was not recovered by the propensity score estimate of the ATE. This is not wholly unexpected, since the outcomes in the data could have resulted from many different models of the relationship between the covariates and the treatment assignment. In order to better visualize these results, a plot of the ATE against the true treatment effect gamma was generated (Figure 3).

**Table 2: Average treatment effect, estimated using the propensity score.** For each combination of betas and alphas, and for each value of gamma, the ATE (left) and the percent bias (right) are shown.

| | ATE/%Bias | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | -0.9 | | -0.5 | | 0 | | 0.5 | | 0.9 | |
| **beta=low/alpha=low** | -0.014 | -98% | 0.038 | -104% | 0.059 | 5891% | 0.001 | -100% | 0.012 | -99% |
| **beta=low/alpha=high** | -0.011 | -99% | -0.068 | -93% | 0.127 | 12749% | -0.004 | -101% | 0.013 | -99% |
| **beta=high/alpha=low** | 0.016 | -102% | 0.144 | -114% | -0.007 | -749% | 0.045 | -91% | 0.204 | -77% |
| **beta=high/alpha=high** | 0.006 | -101% | 0.144 | -114% | -0.007 | -728% | 0.058 | -88% | 0.242 | -73% |

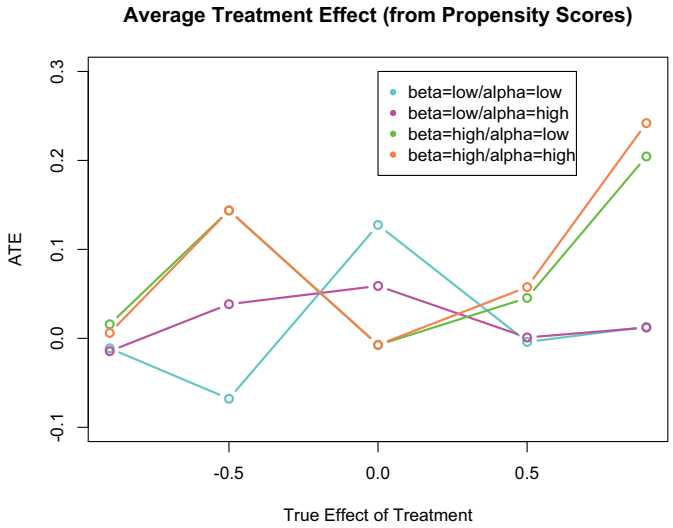


**Figure 3: Plot of the average treatment effect for different true treatment effects under different scenarios.** Note how the beta=low/alpha=high case fails to capture the change in the true treatment effect.

From looking at the plot, it is evident that the ATE estimates did not follow a consistent pattern within the different beta/alpha combination scenarios. However, there was an interesting failure of the low beta/high alpha case, which remained insensitive to the variations in the true treatment effect. This is most likely because the low betas lead to an approximately random chance of being assigned to either treatment. This random chance then confounds the ATE formula captures when there is strong correlation between the treatment and the outcome, i.e. for extremes of the gamma value. Therefore, in the case when the betas are low but the alphas are high, the propensity score will work well when the true treatment effect $\gamma \sim 0$, but will fail when $|\gamma| > 0$ .

### *4.b. Support vector machines*

The results from the SVM tests described in section 3.c. are summarized in Tables 3 and 4. From Table 3, it is evident that in almost every case there was a positive net reclassification improvement. This indicates that the information added by the treatment label variable improve classification. Moreover, sensitivity almost always increased from the "without-SVM" to the "with-SVM." However, the magnitude of the improvement in NRI and in sensitivity was relatively small, especially if you consider that the initial "without-SVM" performance was not impressive. In order to better visualize these results, the NRI was plotted against the true treatment effect (Figure 4).

**Table 3: Net reclassification improvement (NRI) for the SVM pairs.** Shown for each combination of betas and alphas against each value of gamma.

| | NRI | | | | |
|---|---|---|---|---|---|
| | -0.9 | -0.5 | 0 | 0.5 | 0.9 |
| **beta=low/alpha=low** | 2.54% | 2.12% | 0.66% | 1.52% | 0.98% |
| **beta=low/alpha=high** | 1.80% | 1.14% | 0.69% | 1.62% | 0.09% |
| **beta=high/alpha=low** | 0.51% | -0.17% | 1.04% | 0.69% | 0.23% |
| **beta=high/alpha=high** | 0.49% | 0.44% | 0.64% | 0.56% | 0.30% |

**Table 4: Change in sensitivity from the "without-SVM" to the "with-SVM".** Shown for each combination of betas and alphas against each value of gamma.

| | Change in Sensitivity | | | | |
|---|---|---|---|---|---|
| | -0.9 | -0.5 | 0 | 0.5 | 0.9 |
| **beta=low/alpha=low** | 8.8% | 6.7% | 0.4% | 5.3% | 1.6% |
| **beta=low/alpha=high** | 7.2% | 2.8% | 0.6% | 5.7% | 0.2% |
| **beta=high/alpha=low** | -0.2% | -0.2% | -1.3% | 0.4% | 0.2% |
| **beta=high/alpha=high** | 0.0% | 0.6% | 0.1% | 1.6% | 0.1% |

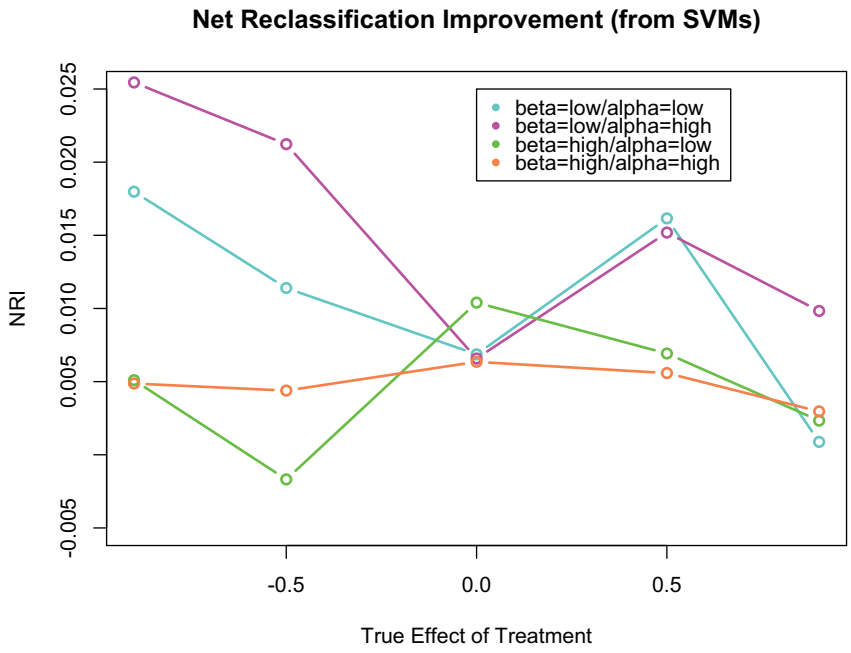


**Figure 4: Plot of the net reclassification improvement for different true treatment effects under different scenarios.** Note how the NRI was higher for the datasets with low betas than it was for the datasets with high betas.

Noticing trends in the data, it is evident that when the betas are low, meaning that the effect of the covariates on treatment assignment is small, the improvement was larger than when the betas were high. This finding is important because it corroborates the idea that when there is a high correlation between the covariates and the treatment assignment, the treatment information is almost redundant with the covariate information. Therefore, for cases such as those with high betas, the net reclassification improvement will not be a useful metric for determining treatment effect on outcome because the improvement might be small despite the treatment effect being large (as is the case when beta is high, alpha is low, and gamma = 0.9).

## 5. Conclusion

After evaluating the results, it seems that neither method performed particularly well at the estimation of treatment effect in these scenarios where the datasets were constructed using coefficients with extreme values. However, even though the magnitude of the improvements was small, the SVMs showed a consistent positive NRI across all cases whereas the patter on the ATE estimations from the propensity score applied to different models was erratic. The extreme value cases were also informative in that they demonstrated how the estimation of the ATE using IPTW with the propensity score fails to capture the variation in treatment effect when the covariates have a low correlation with the treatment assignment but a high correlation with the outcome. Likewise, the results showed that using paired SVMs evaluated with the NRI fails when the baseline covariates are highly correlated with the treatment assignment.

To learn more from these results, further experimentation and validation would be necessary. These future studies could address some of the limitations of this current project. In particular, one of the greatest limitations of this study was insufficient computation power for performing the parameter sweeps necessary for the optimization of the SVMs' performance. Had the SVMs been tuned more meticulously, their performance might have been much better than that which was observed. Another improvement would be to develop a better metric for capturing the treatment effect from the result of the SVM pairs. Moreover, even though other methods for generating synthetic datasets could be explored, it would probably be most useful to perform these tests on real data to validate the results of the experiment.

## 6. References


1. Austin PC. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate Behavioral Research* **46**, 399–424 (2011).
2. Rosembaum PR, Rubin DB. Reducing bias in observational studies using subclassification on the propensity score. *Journal of the American Statistical Association* **79**, 516-524 (1984).
3. Westreich D, Lessler J, Jonsson-Funk M. Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression. *Journal of Clinical Epidemiology* **63**, 826-833 (2010).
4. Adamina M, Guller U, Weber WP, Oertli D. Propensity scores and the surgeon. *British Journal of Surgery* **93**, 389-394 (2006).
5. Lee BK, Lessler J, Stuart EA. Improving propensity score weighting using machine learning. *Statistics in Medicine* **29**, 337-346 (2010).
6. Pearl J. Understanding propensity scores. *Causality: Models, Reasoning, and Inference*, Second Edition. Cambridge University Press, p. 348-352 (2009).
7. Setoguchi S, Schneeweiss S, Brookhart MA, Glynn RJ, Cook EF. Evaluating uses of data mining techniques in propensity score estimation: a simulation study. *Pharmacoepidimemiology and Drug Safety* **17**, 546-555 (2008).
8. Cook, NR. Statistical evaluation of prognostic versus diagnostic models: Beyond the ROC curve. *Clinical Chemistry* **54**, 17-23 (2008).

# Appendix I: Code

## *Synthetic Data Generation (MatLab)*

### *generate_allDatasets.m*

```
%%%%%%%%%%%NOTE: This procedure for synthetic dataset generation modified
%%%%%%%%%%%from Setoguchi et. al., 2008.

ndatasets = 1;
ncohort1 = 20000;

i = 0;
setStore = cell(ndatasets,2);

while i < ndatasets
    [real_set, this_set] = generateSyntheticDatasets(ncohort1);
    setStore{i+1,1} = real_set;
    setStore{i+1,2} = this_set;
    i = i+1;
end

% Save the dataset as a .mat file
save('project097_dataset.mat', 'setStore')

% Concatenate all of the dimensions of the structure into one array
dataset = setStore{1,2};
data_mat = [dataset.W, dataset.TPS, dataset.A, dataset.pY, dataset.Y];

realset = setStore{1,1};
real_mat = [realset.W, realset.TPS, realset.A, realset.pY, realset.Y];

% Save the array as a .csv function for (that can be loaded into R)

fname1 =
['/Users/jacquelinesoegaard/Documents/Spring2012/15.097/CourseProject/SynthData/datas
et_097.csv'];
csvwrite(fname1, data_mat);

fname2 =
['/Users/jacquelinesoegaard/Documents/Spring2012/15.097/CourseProject/SynthData/reals
et_097.csv'];
csvwrite(fname2, real_mat);
```

### *generateSyntheticDatasets.m*

```
function [realdata, dataset] = generateSyntheticDatasets(ncohort)

n_cov = 10; % number of covariates in model

ncohort1 = ncohort;

% A. Create the correlation matrix for the covariates

corr_mat = zeros(n_cov,n_cov);

for i = 1:n_cov
    corr_mat(i,i) = 1;
```

```
end

corr_mat(5,1) = 0.2;
corr_mat(1,5) = 0.2;
corr_mat(6,2) = 0.9;
corr_mat(2,6) = 0.9;
corr_mat(8,3) = 0.2;
corr_mat(3,8) = 0.2;
corr_mat(9,4) = 0.9;
corr_mat(4,9) = 0.9;

% B. Generate the covariates, store in a ncohort x n_covariate matrix

V = zeros(ncohort1,n_cov); % matrix of base covariates
W = NaN(ncohort1, n_cov); % matrix of final covariates

i_base = [1,2,3,4,5,6,8,9];
i_final = [7,10];
i_binary = [1,3,5,6,8,9];

    % 1. Generate 8 base covariates V_i (i = 1...6, 8,9)  and two final
    % covariates W_i (i=7,10) as independent standard normal r.v. ~ N(μ=0,var=1)

        for i = 1:length(i_base)
            V(1:ncohort1, i_base(i)) = random('Normal',0,1,ncohort1,1);
        end
        for i = 1:length(i_final)
            W(1:ncohort1, i_final(i)) = random('Normal',0,1,ncohort1,1);
        end

    % 2. Model the final 8 covariates W_i for i = 1...6,8,9 from linear
    % combinations of the V_i, using the correlations from the correlation
    % matrix.  Also, binarize variables i = 1,3,5,6,8,9

        for i = 1:ncohort1
            for j = 1:length(i_base)
            W(i, i_base(j)) = dot(V(i,:), corr_mat(i_base(j),:), 2);
            end
        end

        var_medians = median(W, 1);

        for i = 1:ncohort1
            for j = 1:length(i_binary)
                var_id = i_binary(j);
                if W(i,var_id) > var_medians(var_id)
                    W(i,i_binary(j)) = 1 ;
                else
                    W(i,i_binary(j)) = 0 ;
                end
            end
        end

% C.  Model the binary exposure for each of seven scenarios

    [truePropScore, exposureA] = calculateTruePropensityScores2(W, ncohort1);

% D. Model the binary outcome Y.

    [tpY_real, outcomeY_real trueProbY, outcomeY] =
calculateTrueOuctomeProb2(W,exposureA, ncohort1);

% Store this dataset's information in a struct object

    realdata.W = W;
    realdata.TPS = truePropScore(:,1);
    realdata.A = exposureA(:,1);
```

```
    realdata.pY = tpY_real;
    realdata.Y = outcomeY_real;

    dataset.W = W;
    dataset.TPS = truePropScore(:,2:3);
    dataset.A = exposureA(:,2:3);
    dataset.pY = trueProbY;
    dataset.Y = outcomeY;
end
```

---

*calculateTruePropensityScores2.m*

---

```
function [TPS, A] = calculateTruePropensityScores2(W, ncohort)

    % Set coefficient values for three different cases:  the Setoguchi
    % et.al. coefficients from real experimental studies, low coefficients,
    % and high coefficients.
    b0 = 0;
    b_real = [0.8, -0.25, 0.6, -0.4, -0.8, -0.5, 0.7];
    b_low = [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05];
    b_high = [0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95];

    TPS = NaN(ncohort,3); % P(A|W_i), true propensity score calculated from model
    A = NaN(ncohort,3);   % binary exposure A

    % Calculate the true propensity scores (TPS) (i.e. P(A|W_i) ) and the dichotomous
    % exposure A for each of the three scenarios.  All scenarios are linear
    % and additive.

    for i = 1:ncohort
        %Scenario A , with the "real" world coefficients.
        TPS(i,1) = (1 + exp( -( b0 + b_real(1)*W(i,1) + b_real(2)*W(i,2) +
b_real(3)*W(i,3) + ...
                 b_real(4)*W(i,4) + b_real(5)*W(i,5) + b_real(6)*W(i,6) +
b_real(7)*W(i,7) )))^-1;
        %Scenario B, with "low" coefficients for all of the covariates.
        TPS(i,2) = (1 + exp( -( b0 + b_low(1)*W(i,1) + b_low(2)*W(i,2) +
b_low(3)*W(i,3) + ...
                 b_low(4)*W(i,4) + b_low(5)*W(i,5) + b_low(6)*W(i,6) +
b_low(7)*W(i,7) )))^-1;
        %Scenario C, with "high" coefficients for all of the covariates.
        TPS(i,3) = (1 + exp( -( b0 + b_high(1)*W(i,1) + b_high(2)*W(i,2) +
b_high(3)*W(i,3) + ...
                 b_high(4)*W(i,4) + b_high(5)*W(i,5) + b_high(6)*W(i,6) +
b_high(7)*W(i,7) )))^-1;

    end

    for i = 1:ncohort
        for j = 1:3
            % generate a random number between [0,1] from the uniform
            % distribution
            rand_num = random('Uniform',0,1);
            if rand_num < TPS(i,j)
                A(i,j) = 1;
            else
                A(i,j) = 0;
            end
        end
    end
end
```

*calculateTrueOutcomeProb2.m*

```
function [pY_real, Y_real, pY, Y] = calculateTrueOuctomeProb2(W, A, ncohort)

    % Here, A has values for real, low, and high beta coeff cases.

    % Number of sets of values for betas, alphas, and gamma.
    num_b = 2;
    num_a = 2;
    num_g = 5;

    % Set coefficient values:
    % The alpha (a0...a7) coefficients are for the outcome model
    a0 = -3.85;
    a_real = [0.3, -0.36, -0.73, -0.2, 0.71, -0.19, 0.26];
    a_low = [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05];
    a_high = [0.95, 0.95, 0.95, 0.95, 0.95, 0.95, 0.95];

    % The treatment labels for the three "beta" cases
    A_real = A(:,1);
    A_low = A(:,2);
    A_high = A(:,3);

    % The gamma (g) coefficient is the effect of exposure on the outcome
    g_real = [-0.4];
    g_range = [-0.9, -0.5, 0, 0.5, 0.95];

    % Instantiate arrays to hold the results of the calculations below

    pY_real = NaN(ncohort,1); % P(Y|W_i, A), probability of the outcome for real
valued coeff scenario
    Y_real = NaN(ncohort, 1); % binary outcome Y for real valued coeff scenario

    pY = NaN(ncohort,num_b*num_a*length(g_range)); % P(Y|W_i, A), probability of the
outcome ...
                                                    % given the covariates and the
exposure
    Y = NaN(ncohort,num_b*num_a*length(g_range));  % binary outcome Y

    % calculate the pY and Y values for the "real" Setoguchi coefficient
    % case
    for i = 1:ncohort
            pY_real(i) = (1 + exp( - (a0 + a_real(1)*W(i,1) + a_real(2)*W(i,2) + ...
                                      a_real(3)*W(i,3) + a_real(4)*W(i,4) +
a_real(5)*W(i,8) + ...
                                      a_real(6)*W(i,9) + a_real(7)*W(i,10) +
g_real*A_real(i) )))^-1 ;
            % generate a random number between [0,1] from the uniform
            % distribution
            rand_num = random('Uniform',0,1);

            if rand_num < pY_real(i)
                Y_real(i) = 1;
            else
                Y_real(i) = 0;
            end
    end

    % Calculate the pY and Y values for the low and high alpha coeff case,
    % with each of the gamma values:
    for i = 1:ncohort
        for gamma = 1:length(g_range)
```

```
            % beta_low and alpha_low
            pY(i,gamma) = (1 + exp( - (a0 + a_low(1)*W(i,1) + a_low(2)*W(i,2) + ...
                    a_low(3)*W(i,3) + a_low(4)*W(i,4) + a_low(5)*W(i,8) + ...
                    a_low(6)*W(i,9) + a_low(7)*W(i,10) +
g_range(gamma)*A_low(i) )))^-1 ;
            % beta_low and alpha_high
            pY(i,(gamma+5)) = (1 + exp( - (a0 + a_high(1)*W(i,1) + a_high(2)*W(i,2)
+ ...
                    a_high(3)*W(i,3) + a_high(4)*W(i,4) + a_high(5)*W(i,8) + ...
                    a_high(6)*W(i,9) + a_high(7)*W(i,10) +
g_range(gamma)*A_low(i) )))^-1 ;
            % beta_high and alpha_low
            pY(i,(gamma+5*2)) = (1 + exp( - (a0 + a_low(1)*W(i,1) + a_low(2)*W(i,2)
+ ...
                    a_low(3)*W(i,3) + a_low(4)*W(i,4) + a_low(5)*W(i,8) + ...
                    a_low(6)*W(i,9) + a_low(7)*W(i,10) +
g_range(gamma)*A_high(i) )))^-1 ;
            % beta_low and alpha high
            pY(i,(gamma+5*3)) = (1 + exp( - (a0 + a_high(1)*W(i,1) + a_high(2)*W(i,2)
+ ...
                    a_high(3)*W(i,3) + a_high(4)*W(i,4) + a_high(5)*W(i,8) + ...
                    a_high(6)*W(i,9) + a_high(7)*W(i,10) +
g_range(gamma)*A_high(i) )))^-1 ;
        end
    end

    for i = 1:ncohort
        for j = 1:(size(pY,2))

            %generate a random number between [0,1] from the uniform
            %distribution
            rand_num = random('Uniform',0,1);

            if rand_num < pY(i,j)
                Y(i,j) = 1;
            else
                Y(i,j) = 0;
            end
        end
    end
end
```

## *Support Vector Machine Calculation (R)*

### *project097_svm.m*

```
# Use SVMs to generate and apply two classifiers:
# minus and plus treatment variables

#Load the necessary packages
library(e1071)

################
# Function for calculating the net reclassification improvement (NRI)

calculateNRI <- function(model1_labels, model2_labels, true_labels){
 # This function calculates the net reclassification improvement, which is a summary
 statistic
 # that describes the reclassification, thus allowing us to compare the clinical impact
```

```
of two models by    # determining how many individuals would be reclassified with new
baseline informaiton.

# NOTE: Here, cases are those examples with outcome Y=1 and noncases are examples with
Y=0,
# as defined by the true (original) labels

num_cases <- sum(true_labels ==1)
num_noncases <- sum(true_labels == 0)

case_ix <- (true_labels ==1)
noncase_ix <- (true_labels ==0)

case_ix <- case_ix - 10000
noncase_ix <- noncase_ix - 10000
# Look at the reclassification movement.
# the "movement" vector will have values in {-1,0,1}
# -1 : "down" movement, reclassified from 1 to 0 with treatment info
#  0 : no reclassification
# +1 : "up" movement, reclassified from 0 to 1 with treatment info

movement <- model2_labels - model1_labels

pUp_cases <- sum(movement[case_ix]==1)/num_cases # = P(up|Y=1)
pDown_cases <- sum(movement[case_ix] == -1)/num_cases # P(down|Y=1)

pUp_noncases <- sum(movement[noncase_ix]==1)/num_noncases # P(up|Y=0)
pDown_noncases <- sum(movement[noncase_ix] == -1)/num_noncases # P(up|Y=0)

netgains_cases <- pUp_cases - pDown_cases
netgains_noncases <- pUp_noncases - pDown_noncases

NRI = netgains_cases - netgains_noncases


sensitivity1 <- sum(model1_labels == test_outcome & test_outcome == 1)/sum(test_outcome
== 1)
specificity1 <- sum(model1_labels == test_outcome & test_outcome == 0)/sum(test_outcome
== 0)

sensitivity2 <- sum(model2_labels == test_outcome & test_outcome == 1)/sum(test_outcome
== 1)
specificity2 <- sum(model2_labels == test_outcome & test_outcome == 0)/sum(test_outcome
== 0)

delta_sens <- sensitivity2-sensitivity1
delta_spec <- specificity2-specificity1

return(c(NRI, netgains_cases, netgains_noncases, delta_sens, delta_spec))
}

################
# Read a practice synthetic dataset
dataset <-
as.data.frame(read.csv("/Users/jacquelinesoegaard/Documents/Spring2012/15.097/CourseProj
```

```
 ect/SynthData/dataset_097.csv", header = FALSE))

n_data <- nrow(dataset)
half_data <- n_data/2 # half of the data will be used for training and half for testing
n_beta <- 2
n_scenario <- 20

# These are the indeces for the different types of data
covariate_ix <- c(1:10)
tps_ix <- c(11:12)
treatmentA_ix <- c(13:14)
pY_ix <- c(15:34)
outcome_ix <- c(35:54)

# Set aside the training data
train_features <- dataset[1:(half_data), covariate_ix]
train_TPS <- dataset[1:(half_data), tps_ix ]
train_treatment <-  dataset[1:(half_data), treatmentA_ix ]
train_pOutcome <-  dataset[1:(half_data), pY_ix ]
train_outcome <-  dataset[1:(half_data), outcome_ix ]

# Set aside the test data
test_features <- dataset[(half_data+1):n_data, covariate_ix]
test_TPS <- dataset[(half_data+1):n_data, tps_ix ]
test_treatment <-  dataset[(half_data+1):n_data, treatmentA_ix ]
test_pOutcome <-  dataset[(half_data+1):n_data, pY_ix ]
test_outcome <-  dataset[(half_data+1):n_data, outcome_ix ]

###############
### Generate SVM models

# We want to train 40 SVMs, with each of the 20 scenarios having two SVMs:
# Model 1 will correspond to the SVMs trained on the baseline covariates (X_is) to
 predict the outcome Y
# Model 2 will correspond to the SVMs trained on BOTH the baseline covariates (X_is) and
 the treatment labels (Z_is) to predict the outcome Y.

# Make lists to hold the svm models
model1_holder <- vector(mode = "list", length = n_scenario)
model2_holder <- vector(mode = "list", length = n_scenario)
performance_stats <- NULL # will fill up later with performance statistics


beta_low_ix <- c(1,2,5,6,9,10,13,14,17,18)
beta_high_ix <- c(3,4,7,8,11,12,15,16,19,20)
C = 100

for (i in 1:n_scenario){
 # train the model that uses only the baseline covariates as features
 X1 <- train_features #features are the n=10 covariates
 Y1 <- train_outcome[1:nrow(train_outcome),i] # outcome
 svm_model_1 <- svm(X1, Y1, scale = FALSE, kernel = "radial", cost = C, decision.values =
 FALSE)
 model1_holder[[i]] <- svm_model_1
```

```
# train the model that uses both the covariates and the treatment labels as features
if (sum(beta_low_ix == i) == 1){
    X2 <- cbind(train_features, train_treatment[1:nrow(train_treatment), 1])
    test_features2 <- cbind(test_features, test_treatment[1:nrow(test_treatment), 1])
    print("im here")
}
if (sum(beta_high_ix == i) == 1){
    X2 <- cbind(train_features, train_treatment[1:nrow(train_treatment), 2])
    test_features2 <- cbind(test_features, test_treatment[1:nrow(test_treatment), 2])
    print("now I'm there")
}
Y2 <- train_outcome[1:nrow(train_outcome),i]
svm_model_2 <- svm(x = X2, y=Y2, scale = FALSE, kernel = "radial", cost = C,
decision.values = FALSE)
model2_holder[[i]] <- svm_model_2

# Apply the SVM classifiers to the test data to obtain predicted labels

model1_labels <- sign(predict(svm_model_1,newdata = test_features, probability = FALSE))
model2_labels <- sign(predict(svm_model_2,newdata = test_features2, probability =
FALSE))

# Convert the +1/-1 labeling convention to +1/0  so that it matches the original labels
model1_labels[model1_labels == 1] <- 0
model2_labels[model2_labels == 1] <- 0

model1_labels[model1_labels == -1] <- 1
model2_labels[model2_labels == -1] <- 1

# Calculate the NRI and other performance statistics
stats <- calculateNRI(model1_labels, model2_labels, test_outcome)
performance_stats <- rbind(performance_stats, stats)
print(stats)

write.table(performance_stats, file = "svm_NRI.csv", sep = ",", col.names = NA, qmethod
= "double")
}
```

## *Propensity Score Calculation (R)*

### *project097_propScore.m*

```
# Perform adaBoost to calculate the propensity scores

library(rpart)
library(ada)

# Read a practice synthetic dataset
dataset <-
 as.data.frame(read.csv("/Users/jacquelinesoegaard/Documents/Spring2012/15.097/CourseProj
 ect/SynthData/dataset_097.csv", header = FALSE))

n_data <- nrow(dataset)
half_data <- n_data/2 # half of the data will be used for training and half for testing
```

```
n_beta <- 2
n_scenario <- 20

# These are the indeces for the different types of data
covariate_ix <- c(1:10)
tps_ix <- c(11:12)
treatmentA_ix <- c(13:14)
pY_ix <- c(15:34)
outcome_ix <- c(35:54)

# Set aside the training data
train_features <- dataset[1:(half_data), covariate_ix]
train_TPS <- dataset[1:(half_data), tps_ix ]
train_treatment <-  dataset[1:(half_data), treatmentA_ix ]
train_pOutcome <-  dataset[1:(half_data), pY_ix ]
train_outcome <-  dataset[1:(half_data), outcome_ix ]

# Set aside the test data
test_features <- dataset[(half_data+1):n_data, covariate_ix]
test_TPS <- dataset[(half_data+1):n_data, tps_ix ]
test_treatment <-  dataset[(half_data+1):n_data, treatmentA_ix ]
test_pOutcome <-  dataset[(half_data+1):n_data, pY_ix ]
test_outcome <-  dataset[(half_data+1):n_data, outcome_ix ]

##############
### Propensity Score calculation using boosting

# We want to train 7 boosting classifiers for calculating the propensity score (i.e.
 probability of receiving treatment), one for each of the scenarios
model_holder <- vector(mode = "list", length = 2)
for (i in 1:n_beta){
 X <- train_features #features are the n=10 covariates
 Y <- train_treatment[1:nrow(train_treatment),i] #the "outcome" is treamtment label
 boost_model_i <- ada(x=X, y=Y)
 model_holder[[i]] <- boost_model_i
}

# Make a matrix for storing the propensity scores for each individual under
# The matrix dimensions (n x m) will be : n = number of instances ; m = number of beta
 scenarios
# Each entry (i,j) will denote the propensity score of instance i in scenario j
prop_scores <- NULL
for (i in 1:n_beta) { # Loop over the two beta coefficient cases
 # the predict function will return a vector with the probability of class membership
 class_prob <- predict(model_holder[[i]], newdata = test_features, type = "prob")
 prop_scores <- cbind(prop_scores,class_prob[1:nrow(class_prob), 2])
}

beta_low_PS <- prop_scores[1:nrow(prop_scores),1]
beta_high_PS <- prop_scores[1:nrow(prop_scores),2]


##############
### Now, use the propensity scores to calculate the ATE for the 20 outcome scenarios
```

```
beta_low_ix <- c(1,2,5,6,9,10,13,14,17,18)
beta_high_ix <- c(3,4,7,8,11,12,15,16,19,20)
holderATE <- c(from = 0, to = 0, length.out = n_scenario )

for (i in 1: n_scenario){ #Loop over the 20 scenarios

 # Select the outcome coefficients for scenario i out of the 20 scenarios.
 outcome <- test_outcome[1:nrow(test_outcome),i]

 # Case where the low beta coefficients were used for the treatment and outcome models
 if (sum(beta_low_ix == i) == 1){
     treatment <- test_treatment[1:nrow(test_treatment),1]
     holderATE[i] <- (1/half_data)*sum((treatment*outcome)/beta_low_PS) -
(1/half_data)*sum(((1-treatment)*outcome)/(1-beta_low_PS))
 }
 # Case where the high beta coefficients were used for the treatment and outcome models
 else{
     treatment <- test_treatment[1:nrow(test_treatment),2]
     holderATE[i] <- (1/half_data)*sum((treatment*outcome)/beta_high_PS) -
(1/half_data)*sum(((1-treatment)*outcome)/(1-beta_high_PS))
 }
}

write.table(holderATE, file = "averageTreatemntEffect.csv", sep = ",", col.names = NA,
 qmethod = "double")
```

MIT OpenCourseWare
http://ocw.mit.edu

15.097 Prediction: Machine Learning and Statistics
Spring 2012

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.