extended_case_7

May 29, 2020

1 Does a job training program improve the earnings of disadvantaged workers?

[1]:
```
# Core
import numpy as np
import scipy
import statsmodels.api as sm
import statsmodels.formula.api as smf
import pylab

# Data
import pandas as pd

# Visualisation
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
sns.set_style('darkgrid')
plt.style.use('ggplot')
```

[2]:
```
# Silent warnings
import warnings
warnings.filterwarnings("ignore")
```

1.1 Introduction

Business Context. In the 1980s, Robert Lalonde conducted a study to evaluate the effects of training programs on labour workers. He observed the characteristic background data of the individuals involved. Several of them were selected for enrollment in the job training program (the National Supported Work Demonstration). The training program specifically targeted unemployed women, ex-drug addicts, ex-criminal offenders, and high school dropouts.

The Department of Labor is interested in digging deeper into this data and coming up with some actionable insights in order to raise the earnings of disadvantaged workers. They have contracted you as a data science consultant to assist them with this task.

Business Problem. Your goal is to evaluate **whether the job training program *causes* the earnings of disadvantaged workers to go up**.

Analytical Context. In this case we will continue our discussion of causal inference. We will study the importance of covariate balance and explore how to perform matching to get this balance. We will leverage a lot of the new classification models we have learned for this purpose.

1.2 Examining the data

The Lalonde dataset provides information on annual income for workers who enroll in the training workshop and those who did not enroll in the year subsequent to training. The outcome of interest is `re78` in which we want to know if there was an increase in earnings in 1978:

1. **age**: age in years
2. **educ**: years of schooling
3. **black**: indicator variable for blacks
4. **hisp**: indicator variable for Hispanics
5. **married**: indicator variable for martial status
6. **nodegr**: indicator variable for high school diploma
7. **re74**: real earnings in 1974
8. **re75**: real earnings in 1975
9. **re78**: real earnings in 1978 - this is the outcome of interest
10. **treat**: an indicator variable for treatment status

[3]:
```
lalonde_df = pd.read_csv('lalonde.csv', index_col=0)
lalonde_df.head()
```

[3]:

	treat	age	educ	black	hispan	married	nodegree	re74	re75	\
NSW1	1	37	11	1	0	1	1	0.0	0.0	
NSW2	1	22	9	0	1	0	1	0.0	0.0	
NSW3	1	30	12	1	0	0	0	0.0	0.0	
NSW4	1	27	11	1	0	0	1	0.0	0.0	
NSW5	1	33	8	1	0	0	1	0.0	0.0	

	re78
NSW1	9930.0460
NSW2	3595.8940
NSW3	24909.4500
NSW4	7506.1460
NSW5	289.7899

1.2.1 Exercise 1:

1.1 Load and examine the Lalonde data. Provide summary statistics for all the variables in the dataset.

Answer. One possible solution is shown below:

[4]:
```
lalonde_df = pd.read_csv('lalonde.csv', index_col=0)
lalonde_df.head()
```

[4]:

	treat	age	educ	black	hispan	married	nodegree	re74	re75	re78
NSW1	1	37	11	1	0	1	1	0.0	0.0	9930.0460
NSW2	1	22	9	0	1	0	1	0.0	0.0	3595.8940
NSW3	1	30	12	1	0	0	0	0.0	0.0	24909.4500
NSW4	1	27	11	1	0	0	1	0.0	0.0	7506.1460
NSW5	1	33	8	1	0	0	1	0.0	0.0	289.7899

[5]:
```
lalonde_df.shape
```

[5]: (614, 10)

[6]:
```
# let's have an overview of the data
lalonde_df.describe()
```

[6]:

	treat	age	educ	black	hispan	married	nodegree	re74	re75	re78
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000
mean	0.301303	27.363192	10.268730	0.395765	0.117264	0.415309	0.630293	4557.546569	2184.938207	6792.834483
std	0.459198	9.881187	2.628325	0.489413	0.321997	0.493177	0.483119	6477.964479	3295.679043	7470.730792
min	0.000000	16.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	20.000000	9.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	238.283425
50%	0.000000	25.000000	11.000000	0.000000	0.000000	0.000000	1.000000	1042.330000	601.548400	4759.018500
75%	1.000000	32.000000	12.000000	1.000000	0.000000	1.000000	1.000000	7888.498250	3248.987500	10893.592500
max	1.000000	55.000000	18.000000	1.000000	1.000000	1.000000	1.000000	35040.070000	25142.240000	60307.930000

1.2 Find the difference in means and medians of earnings between the control group and the treatment group. Are these differences statistically significant?

Answer. One possible solution is shown below:

[7]:
```
lalonde_df.groupby('treat')['re78'].agg(['median','mean'])
```

[7]:

treat	median	mean
0	4975.505	6984.169742
1	4232.309	6349.143530

Control group

treatmet group

[8]:
```
6984.169742 - 6349.143530
```

[8]:
```
635.0262119999998
```

[9]:
```
scipy.stats.ttest_ind(lalonde_df.loc[lalonde_df['treat']==1, 're78'],␣
↪lalonde_df.loc[lalonde_df['treat']==0, 're78'], axis=0, equal_var= False)
```

[9]:
```
Ttest_indResult(statistic=-0.9377296979393183, pvalue=0.3490766555566698)
```

It seems that treatment "hurts" workers; we see a decrease of $635 dollars, although this does not seem to be statistically significant.

1.3 Does your previous result mean that the treatment had a negative impact? Why or why not?

Answer. This is a naive conclusion. This measurement does not take into account the background of those who are selected for training program vs. those who don't enroll in the program. In the LaLonde study, those who were selected into training program were chosen specifically due to their low earning potential and therefore systematically differ from those in the control group who were not selected for job training. This source of bias needs to be accounted for when doing subsequent analysis.

1.3 Finding causal relationships

Before we start our exploration on causality, let's perform some simple setup:

1. We seperate the `treat` (treatment) indicator from our dataset. We will want to use this as our objective variable to fit models to later
2. We are interested in knowing the treatment effect on `re78` (revenue/earnings in 1978)

[10]:
```
# create seperate structure for data and target
treatment = lalonde_df['treat']
rev78 = lalonde_df['re78']
cleaned_df = lalonde_df.drop(['treat', 're78'], axis=1)
```

1.4 Assessing balance between the control group and the treatment group

Suppose we want to assess whether balance has been achieved for a particular feature $x = age$. We can look at **Standardized Mean Differences (SMD)**, which is calculated by the difference in

the means of the ages between the two groups divided by the pooled standard deviation.

$$\text{SMD}(x) = \frac{\bar{x}_t - \bar{x}_c}{\sqrt{\frac{s_t^2 + s_c^2}{2}}}.$$

The following explains the notation in the above equation:

1. $(\bar{x}_t,\ \bar{x}_c)$ denotes the mean of that feature (in our example, the age) for the treatment and control group respectively. Note that people often times report the absolute value of this number.
2. $(s_t^2,\ s_c^2)$ denotes the standard deviation of that feature for the treatment and control group respectively. For the denominator we are just pooling standard deviation. We can calculate the standardized mean differences for every feature. If our calculated SMD is 1, then that means there's a 1 standard deviation difference in means. The benefit of having standard deviation in the denominator is that this number becomes insensitive to the scale of the feature.

After computing this measurement for all of our features, there is a rule of thumb that is commonly used to determine whether that feature is balanced or not (similar to the idea of using 0.05 as a threshold for p - values):

1. **SMD $<$ 0.1**: For a **randomized trial**, the SMD between all of the covariates should typically fall into this bucket.
2. **SMD is between 0.1 and 0.2**: Not necessarily balanced, but small enough that people are usually not too worried about them. Sometimes, even after performing matching, there might still be a few covariates whose SMD fall under this range.
3. **SMD $>$ 0.2**: Values that are greater than this threshold are considered seriously imbalanced.

1.4.1 Exercise 2:

2.1 Assess covariate balance between the treatment and control groups. What features (covariates) are imbalanced between the control group and the treatment group in this dataset?

Answer. One possible solution is shown below:

[11]:
```
# Define params and inputs
covariates = ['age', 'educ', 'black', 'hispan', 'married', 'nodegree', 're74',␣
 ↪'re75']
agg_operations = {'treat': 'count'}
agg_operations.update({
    feature: ['mean', 'std'] for feature in covariates
})
```

[12]:
```
def compute_table_one_smd(table_one, covariates, round_digits=4):
    feature_smds = []
    for feature in covariates:
        feature_table_one = table_one[feature].values
        neg_mean = feature_table_one[0, 0]
```

```
        neg_std = feature_table_one[0, 1]
        pos_mean = feature_table_one[1, 0]
        pos_std = feature_table_one[1, 1]

        smd = (pos_mean - neg_mean) / np.sqrt((pos_std ** 2 + neg_std ** 2) / 2)
        smd = round(abs(smd), round_digits)
        feature_smds.append(smd)

    return pd.DataFrame({'covariates': covariates, 'smd': feature_smds})
```

[13]:
```
table_one_matched = lalonde_df.groupby('treat').agg(agg_operations)
table_one_smd_matched = compute_table_one_smd(table_one_matched, covariates)
table_one_smd_matched
```

[13]:

	covariates	smd
0	age	0.2419
1	educ	0.0448
2	black	1.6677
3	hispan	0.2769
4	married	0.7195
5	nodegree	0.2350
6	re74	0.5958
7	re75	0.2870

We see that `age`, `hispan`, `married`, `nodegree`, `re74` and `re75` are considered seriously imbalanced.

2.2 Make plots which show the histograms for both the treatment and control groups of the imbalanced covariates.

Answer. One possible solution is shown below:

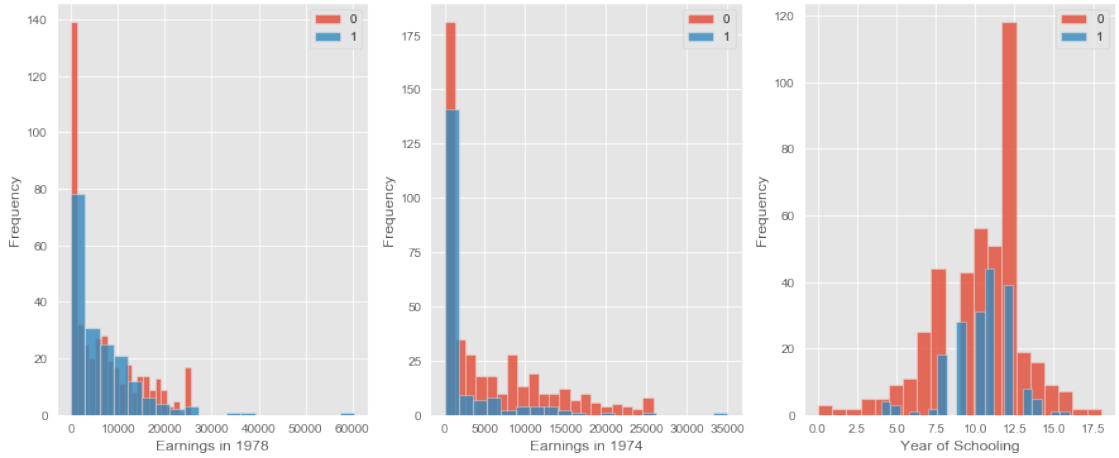
[14]:
```
# we seperate the dataframe based on it's
# corresponding true label value
mask = lalonde_df['treat'] == 1
lalonde_treat = lalonde_df[mask]
lalonde_notreat = lalonde_df[~mask]

print('treatment count:', lalonde_treat.shape)
print('control count:', lalonde_notreat.shape)
```

```
treatment count: (185, 10)
control count: (429, 10)
```

[15]:
```
rev_treated_mean = lalonde_treat['re78'].mean()
rev_notreat_mean = lalonde_notreat['re78'].mean()

print("The raw difference: ", rev_treated_mean - rev_notreat_mean)
```

```
The raw difference:  -635.0262120374209
```

[16]:
```
fig, (ax1, ax2, ax3) = plt.subplots(ncols=3, nrows=1, figsize=(15,6))

# plot income distribution for 1978
ax1, _ = lalonde_df.groupby('treat')['re78'].plot(kind='hist',
                                        bins=20, alpha=0.8, legend=True,
                                        ax=ax1)
ax1.set_xlabel("Earnings in 1978");

# plot income distribution for 1974
ax2, _ = lalonde_df.groupby('treat')['re74'].plot(kind='hist',
                                            bins=20, alpha=0.8, legend=True,
                                            ax=ax2)
ax2.set_xlabel("Earnings in 1974");

# plot education distribution
ax3, _ = lalonde_df.groupby('treat')['educ'].plot(kind='hist',
                                      bins=20, alpha=0.8, legend=True,
                                      ax=ax3)
ax3.set_xlabel("Year of Schooling");
```



1.5 Matching

Matching can be defined as any method that *strategically subsamples* a dataset with the aim of balancing observable covariate distributions between the treatment and control groups. It is one of several statistical techniques that emerged in the 1980s with the aim of estimating causal effects.

Note that matching is not a standalone estimation technique by itself; rather, it is a pre-processing step that prepares the data for further analysis. In this particular situation, the challenge is that

we are trying to perform matching when there are many features present in the data, so it is not clear which ones to match on.

1.5.1 Exercise 3:

Why can we not just match on all of the features, rather than selected ones? Wouldn't this control for differences between the control and treatment groups most effectively?

Answer. Yes, although this is true, such a plan is often not feasible. The number of samples which have similar enough values across all features scales *exponentially* with the number of features. In order to find sufficiently matching samples across all features, you would need an extraordinarily large number of data points, which is simply not possible in most problems.

In this context, a metric called the **propensity score** helps us do the matching on select features in an efficient way. The key idea is that *the features that are correlated with the probability of a worker being selected into the treatment group affecting our causal results.* A propensity score is this probability. The propensity score for subject i, denoted as π_i is defined as:

$\pi_i = \text{P}(\text{Subject } i \text{ is given training given their features } X_i)$

As an example, if a person had a propensity score of 0.3, that would mean that given their particular covariates, there was a 30% chance that they were placed in the treatment group. We can calculate this score by fitting a classification model to our data, where the input features are our covariates, and the output is whether that person was part of the treatment group or not.

1.5.2 Exercise 4:

Estimate the propensity scores using logistic regression; i.e. build a logistic regression model where the outcome variable is whether a subject was in the treatment group or not, and the covariates are all the other variables except `re78`. The predicted probabilities are the propensity scores; call these the **pscores**. We are going to use them later.

Answer. One possible solution is shown below:

[17]:
```
# drop labeling (e.g NSW3) from the targets data
targets = treatment.reset_index().drop('index', axis=1);
X = cleaned_df.reset_index().drop('index', axis=1);
```

[18]:
```
formula = 'treat ~ age + educ + black + hispan + married + nodegree + re74 +␣
 ↪re75'
logit_model = smf.logit(formula=formula, data=lalonde_df).fit()

print(logit_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.397267
         Iterations 7
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                  treat   No. Observations:                  614
Model:                          Logit   Df Residuals:                      605
Method:                           MLE   Df Model:                            8
Date:                Sun, 24 Nov 2019   Pseudo R-squ.:                  0.3508
Time:                        12:01:25   Log-Likelihood:                -243.92
converged:                       True   LL-Null:                       -375.75
Covariance Type:            nonrobust   LLR p-value:                 2.194e-52
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.7286      1.017     -4.649      0.000      -6.722      -2.735
age            0.0158      0.014      1.162      0.245      -0.011       0.042
educ           0.1613      0.065      2.477      0.013       0.034       0.289
black          3.0654      0.287     10.698      0.000       2.504       3.627
hispan         0.9836      0.426      2.311      0.021       0.149       1.818
married       -0.8321      0.290     -2.866      0.004      -1.401      -0.263
nodegree       0.7073      0.338      2.095      0.036       0.045       1.369
re74       -7.178e-05   2.87e-05     -2.497      0.013      -0.000   -1.54e-05
re75        5.345e-05   4.63e-05      1.153      0.249   -3.74e-05       0.000
==============================================================================
```

[19]:
```
pscore = logit_model.predict()
```

1.6 Matching using k-nearest neighbors

We are going to match treated and control subjects using nearest neighbor matching on the estimated propensity score. Here we use another popular clustering algorithm: the **k-nearest neighbors (k-NN) algorithm**.

[20]:
```
# we seperate the pscore based on it's
# corresponding true label value
# remember pscore comes from your answer for question 5.
tags = targets.squeeze() == 1
pos_pscore = pscore[tags]
neg_pscore = pscore[~tags]
```

[21]:
```
# Define params and inputs
mask = lalonde_df['treat'] == 1
covariates = ['age', 'educ', 'black', 'hispan', 'married', 'nodegree', 're74',␣
 ↪'re75']
agg_operations = {'treat': 'count'}
agg_operations.update({
    feature: ['mean', 'std'] for feature in covariates
})
```

[22]:
```
from sklearn.neighbors import NearestNeighbors

# Helper function that uses K-NN algorithm to calculate
# the similarity distance from propensity score
# and the corresponding similarity indices for the purpose of matching later
def get_similar(pos_pscore, neg_pscore, topn=5, n_jobs=1):
    knn = NearestNeighbors(n_neighbors=topn,
                           metric='euclidean',
                           n_jobs=n_jobs)
    knn.fit(neg_pscore.reshape(-1, 1))

    distances, indices = knn.kneighbors(pos_pscore.reshape(-1, 1))
    sim_distances = distances[:, :]
    sim_indices = indices[:, :]

    return sim_distances, sim_indices
```

[23]:
```
sim_distances, sim_indices = get_similar(pos_pscore, neg_pscore, topn=1)
```

[24]:
```
# Display matching table
df_pos = lalonde_df[mask]
df_neg = lalonde_df[~mask].iloc[sim_indices[:, 0]]
df_matched_join = (df_pos.reset_index(drop=True)
              .merge(df_neg.reset_index(drop=True),
                     left_index=True,
                     right_index=True))

num_matched_pairs = df_neg.shape[0]

print('Number of matched pairs: ', num_matched_pairs)
```

```
Number of matched pairs:  185
```

[25]:
```
df_matched_join
```

[25]:

	treat_x	age_x	educ_x	black_x	hispan_x	married_x	nodegree_x	\
0	1	37	11	1	0	1	1	
1	1	22	9	0	1	0	1	
2	1	30	12	1	0	0	0	
3	1	27	11	1	0	0	1	
4	1	33	8	1	0	0	1	
..	…	…	…	…	…	…	…	
180	1	33	12	1	0	1	0	
181	1	25	14	1	0	1	0	
182	1	35	9	1	0	1	1	
183	1	35	8	1	0	1	1	
184	1	33	11	1	0	1	1	

```
        re74_x    re75_x      re78_x  treat_y  age_y  educ_y  black_y  \
0         0.00      0.00   9930.0460        0     30      17        1
1         0.00      0.00   3595.8940        0     51      11        0
2         0.00      0.00  24909.4500        0     16       9        1
3         0.00      0.00   7506.1460        0     39      10        1
4         0.00      0.00    289.7899        0     19       9        1
..         …         …          …       …     …       …        …
180   20279.95  10941.35  15952.6000        0     18      11        0
181   35040.07  11536.57  36646.9500        0     26      12        0
182   13602.43  13830.64  12803.9700        0     34      12        1
183   13732.07  17976.15   3786.6280        0     47       8        1
184   14660.71  25142.24   4181.9420        0     23      13        1

     hispan_y  married_y  nodegree_y       re74_y     re75_y      re78_y
0           0          0           0  17827.37000  5546.4190  14421.1300
1           0          0           1     48.98167  3813.3870   1525.0140
2           0          0           1      0.00000     0.0000   2158.9590
3           0          0           1    844.44400   889.7903    701.9201
4           0          0           1   1079.55600  2873.4680  14344.2900
..        …          …           …           …          …          …
180         1          0           1      0.00000   630.1935      0.0000
181         1          0           0   7968.33800  5109.5810   4181.9660
182         0          1           0      0.00000     0.0000  18716.8800
183         0          1           1   9275.16900  8543.4190      0.0000
184         0          0           0    172.41550   272.1290    582.2243

[185 rows x 20 columns]
```

1.7 Propensity score distribution plot after matching

Let's take a glance at the distribution of propensity score after matching. Both groups have similar distributions on the propensity score. This means we have reach a balance in both groups. This enables us to further analyse the causality effect of the treatment.

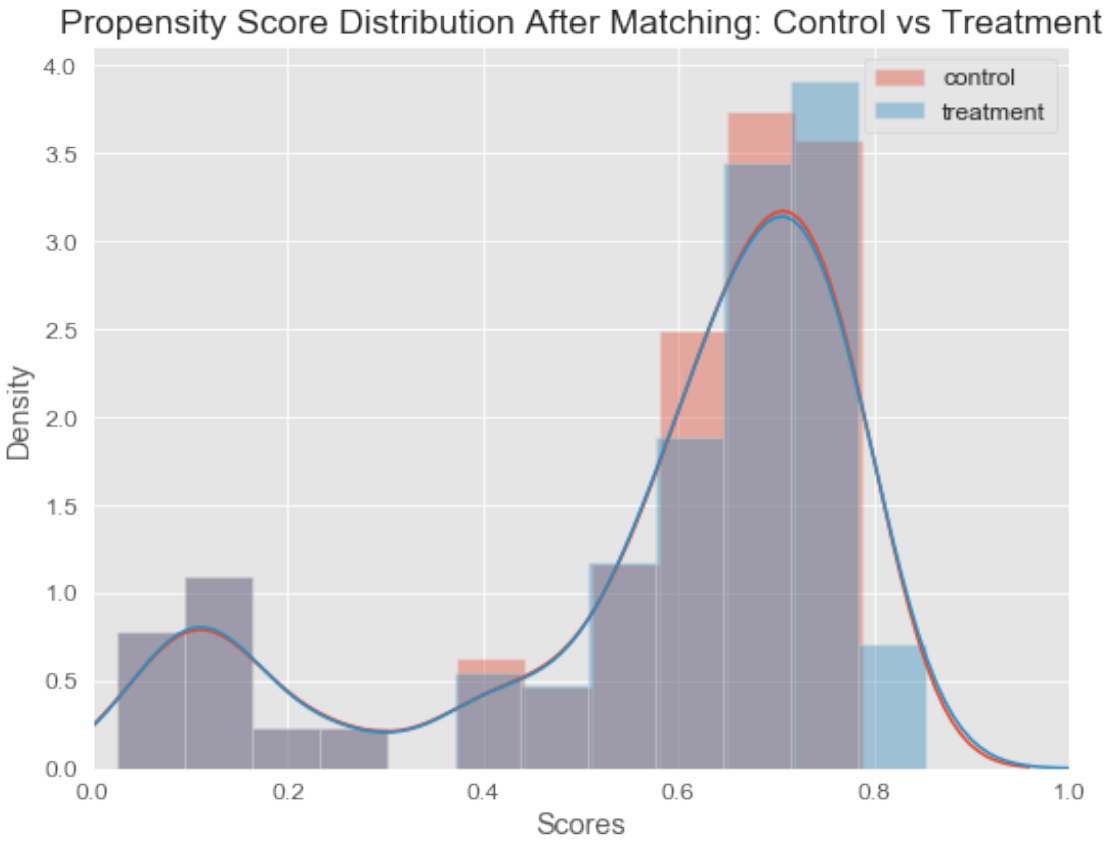
1.7.1 Exercise 5:

Make a plot which shows histograms of propensity scores for the control and treatment groups after matching using the k - NN code given above.

Answer. One possible solution is given below:

[26]:
```
# change default style figure and font size
plt.rcParams['figure.figsize'] = 8, 6
plt.rcParams['font.size'] = 12
```

```
sns.distplot(neg_pscore[sim_indices[:, 0]], label='control')
sns.distplot(pos_pscore, label='treatment')
plt.xlim(0, 1)
plt.title('Propensity Score Distribution After Matching: Control vs Treatment')
plt.ylabel('Density')
plt.xlabel('Scores')
plt.legend()
plt.tight_layout()
plt.show()
```



1.7.2 Exercise 6:

Assess covariate balance after the k-NN matching. For this, compute the absolute standardized differences in means in the covariates after matching,

$$ASMD_a(x) = \frac{\bar{x}_{t,a} - \bar{x}_{c,a}}{\sqrt{\frac{s^2_{t,b}+s^2_{c,b}}{2}}},$$

where $\bar{x}_{t,a}$ and $\bar{x}_{c,a}$ are, respectively, the means of covariate x in the treatment and control groups after matching, and $s^2_{t,b}$ and $s^2_{c,b}$ are, correspondingly, the sample variances treatment and control

groups before matching.

One reason to use the sample variances before matching rather than the sample variances after matching is to free the comparisons of the means after matching from simultaneous changes in the variances. Comment on covariate balance.

Answer. One possible solution is given below:

[27]:
```
# Create matching table
df_pos = lalonde_df[mask]
df_neg = lalonde_df[~mask].iloc[sim_indices[:, 0]]
df_matched = pd.concat([df_pos, df_neg], axis=0) # this time we join by axis-0

# Calculate Rosenbaum ASMD
table_one_matched = df_matched.groupby('treat').agg(agg_operations)
table_one_smd_matched = compute_table_one_smd(table_one_matched, covariates)

table_one_smd_matched
```

[27]:

	covariates	smd
0	age	0.2036
1	educ	0.0139
2	black	0.0147
3	hispan	0.0223
4	married	0.1625
5	nodegree	0.0118
6	re74	0.0499
7	re75	0.0102

[28]:
```
table_one_matched
```

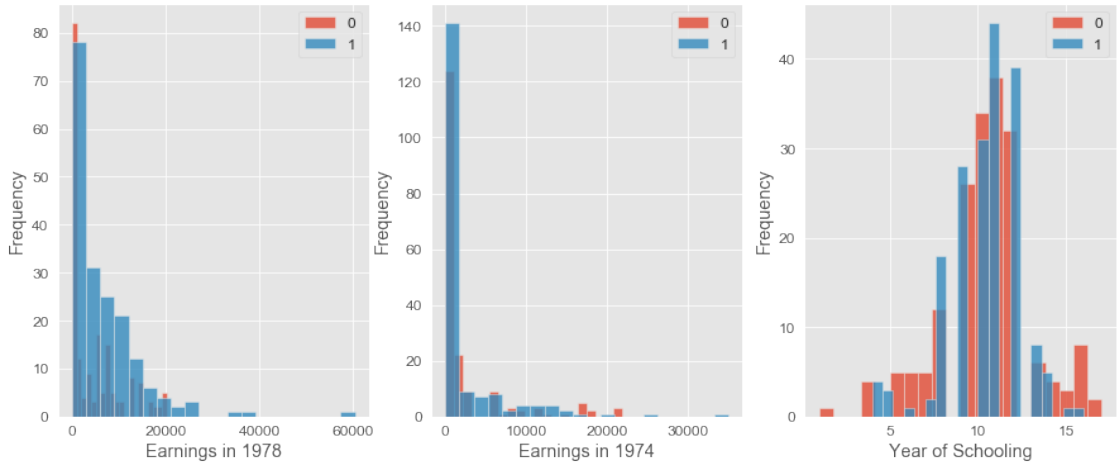
[28]:

	treat	age		educ		black		hispan		married		nodegree		re74		re75	
	count	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
treat																	
0	185	24.102703	9.512877	10.378378	2.624626	0.837838	0.369600	0.064865	0.246956	0.129730	0.336918	0.702703	0.458309	2336.462883	4760.957206	1503.928959	2188.070067
1	185	25.816216	7.155019	10.345946	2.010650	0.843243	0.364558	0.059459	0.237124	0.189189	0.392722	0.708108	0.455867	2095.573689	4886.620353	1532.055314	3219.250870

It seems that covariate balance is much better, as all but two of the variables have SMD less than 0.1, and all but one has SMD less than 0.2 (and the remaining one is right on the cusp).

If we look once again at the histograms of several covariates after we apply matching, we can see a marked improvement in the similarity of the histograms of the control and treatment groups:

[29]:
```
fig, (ax1, ax2, ax3) = plt.subplots(ncols=3, nrows=1, figsize=(15,6))

# plot income distribution for 1978
ax1, _ = df_matched.groupby('treat')['re78'].plot(kind='hist',
                                        bins=20, alpha=0.8, legend=True,
                                        ax=ax1)
ax1.set_xlabel("Earnings in 1978");

# plot income distribution for 1974
ax2, _ = df_matched.groupby('treat')['re74'].plot(kind='hist',
                                            bins=20, alpha=0.8, legend=True,
                                            ax=ax2)
ax2.set_xlabel("Earnings in 1974");        # set title for y axis

# plot education distribution
ax3, _ = df_matched.groupby('treat')['educ'].plot(kind='hist',
                                     bins=20, alpha=0.8, legend=True,
                                     ax=ax3)
ax3.set_xlabel("Year of Schooling");        # set title for y axis
```



1.7.3 Exercise 7:

Estimate the average effect of treatment on the treated after matching. What do you conclude?

Answer. Shown below:

[30]:
```
df_matched_join.head()
```

[30]:

	treat_x	age_x	educ_x	black_x	hispan_x	married_x	nodegree_x	re74_x	\
0	1	37	11	1	0	1	1	0.0	
1	1	22	9	0	1	0	1	0.0	
2	1	30	12	1	0	0	0	0.0	
3	1	27	11	1	0	0	1	0.0	
4	1	33	8	1	0	0	1	0.0	

	re75_x	re78_x	treat_y	age_y	educ_y	black_y	hispan_y	married_y	\
0	0.0	9930.0460	0	30	17	1	0	0	
1	0.0	3595.8940	0	51	11	0	0	0	
2	0.0	24909.4500	0	16	9	1	0	0	
3	0.0	7506.1460	0	39	10	1	0	0	
4	0.0	289.7899	0	19	9	1	0	0	

	nodegree_y	re74_y	re75_y	re78_y
0	0	17827.37000	5546.4190	14421.1300
1	1	48.98167	3813.3870	1525.0140
2	1	0.00000	0.0000	2158.9590
3	1	844.44400	889.7903	701.9201
4	1	1079.55600	2873.4680	14344.2900

[31]:
```
df_matched_join["ATE"] = df_matched_join["re78_x"] - df_matched_join["re78_y"]
```

[32]:
```
df_matched_join["ATE"].mean()
```

[32]:
```
1788.9590464864864
```

It seems like there is indeed is a positive effect on wages after training.

1.7.4 Exercise 8:

8.1 Estimate the propensity scores now by using random forests. What is the AUC for your random forest model?

Answer. One possible solution is shown below:

[33]:
```
# drop labeling (e.g NSW3) from the targets data
treatment = lalonde_df['treat']
rev78 = lalonde_df['re78']
cleaned_df = lalonde_df.drop(['treat', 're78'], axis=1)
targets = treatment.reset_index().drop('index', axis=1);
X = cleaned_df.reset_index().drop('index', axis=1);
```

[34]:
```
def calc_and_plot_auc_curve(y_pred_proba, targets, model_name=""):
    fpr, tpr, _ = metrics.roc_curve(targets,  y_pred_proba)
```

```
    auc = metrics.roc_auc_score(targets, y_pred_proba)

    # Display plot for AUC curve
    plt.figure(figsize=(10, 6))
    plt.plot(fpr,tpr,label="data 1, auc=%s " % str(auc))
    plt.legend(loc=4)
    plt.title("AUC Curve for %s" % model_name)
    plt.show()

    return auc
```

[35]:
```
from sklearn.ensemble import RandomForestClassifier
#
rf_model = RandomForestClassifier(n_estimators=100, max_depth=4,␣
 ↪random_state=42)
rf_model.fit(X, targets);
```

[36]:
```
from sklearn.metrics import roc_curve, auc, accuracy_score
y_pred_proba = rf_model.predict_proba(X)[::, 1]
roc_p = roc_curve(targets, y_pred_proba)
auc_p  = auc( roc_p[0], roc_p[1] )
auc_p
```

[36]:
```
0.9325206325206326
```

8.2 Repeat the matching procedure now with the estimated propensity scores using the random forest classifier.

Answer. One possible solution is shown below:

[37]:
```
tags = targets.squeeze() == 1
pscore = y_pred_proba
pos_pscore = pscore[tags]
neg_pscore = pscore[~tags]
```

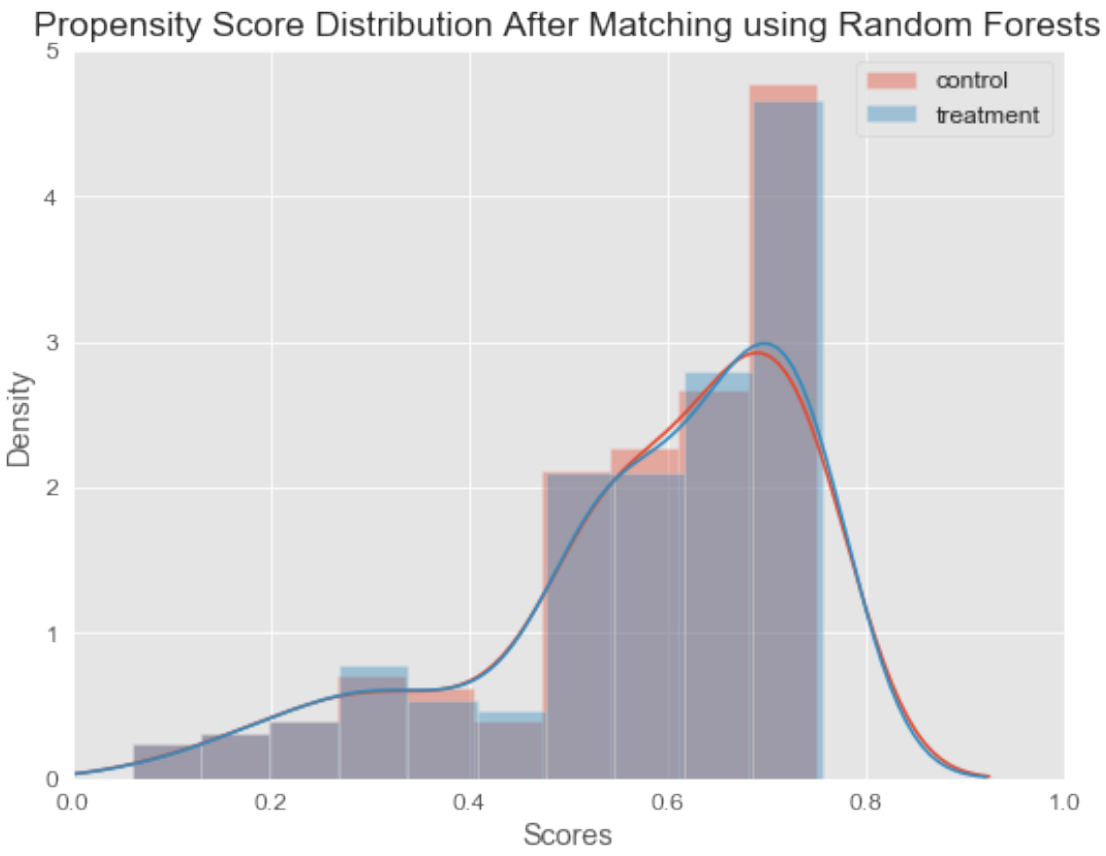
[38]:
```
# Define params and inputs
mask = lalonde_df['treat'] == 1
covariates = ['age', 'educ', 'black', 'hispan', 'married', 'nodegree', 're74',␣
 ↪'re75']
agg_operations = {'treat': 'count'}
agg_operations.update({
    feature: ['mean', 'std'] for feature in covariates
})
```

[39]:
```
sim_distances, sim_indices = get_similar(pos_pscore, neg_pscore, topn=1)
```

[40]:
```
# Display matching table
df_pos = lalonde_df[mask]
df_neg = lalonde_df[~mask].iloc[sim_indices[:, 0]]
df_matched_join = (df_pos.reset_index(drop=True)
               .merge(df_neg.reset_index(drop=True),
                      left_index=True,
                      right_index=True))

num_matched_pairs = df_neg.shape[0]

print('Number of matched pairs: ', num_matched_pairs)
```

```
Number of matched pairs:  185
```

[41]:
```
# change default style figure and font size
plt.rcParams['figure.figsize'] = 8, 6
plt.rcParams['font.size'] = 12

sns.distplot(neg_pscore[sim_indices[:, 0]], label='control')
sns.distplot(pos_pscore, label='treatment')
plt.xlim(0, 1)
plt.title('Propensity Score Distribution After Matching using Random Forests')
plt.ylabel('Density')
plt.xlabel('Scores')
plt.legend()
plt.tight_layout()
plt.show()
```



8.3 Give the table showing the balance of covariates. What do you get the treatment effect to be in this case?

Answer. One possible solution is given below:

[42]:
```
# Create matching table
df_pos = lalonde_df[mask]
df_neg = lalonde_df[~mask].iloc[sim_indices[:, 0]]
df_matched = pd.concat([df_pos, df_neg], axis=0) # this time we join by axis-0

# Calculate ASMD
table_one_matched = df_matched.groupby('treat').agg(agg_operations)
table_one_smd_matched = compute_table_one_smd(table_one_matched, covariates)

table_one_smd_matched
```

[42]:

	covariates	smd
0	age	0.1354
1	educ	0.0605
2	black	0.1788

```
3     hispan  0.1595
4    married  0.1310
5   nodegree  0.1385
6       re74  0.1646
7       re75  0.2511
```

[43]:
```
df_matched_join.head()
```

[43]:

	treat_x	age_x	educ_x	black_x	hispan_x	married_x	nodegree_x	re74_x	\
0	1	37	11	1	0	1	1	0.0	
1	1	22	9	0	1	0	1	0.0	
2	1	30	12	1	0	0	0	0.0	
3	1	27	11	1	0	0	1	0.0	
4	1	33	8	1	0	0	1	0.0	

	re75_x	re78_x	treat_y	age_y	educ_y	black_y	hispan_y	married_y	\
0	0.0	9930.0460	0	40	11	1	0	1	
1	0.0	3595.8940	0	25	12	1	0	1	
2	0.0	24909.4500	0	27	10	1	0	0	
3	0.0	7506.1460	0	27	10	1	0	0	
4	0.0	289.7899	0	34	12	1	0	0	

	nodegree_y	re74_y	re75_y	re78_y
0	1	0.0000	0.000	0.0000
1	0	295.8493	6942.871	461.0507
2	1	0.0000	0.000	7543.7940
3	1	0.0000	0.000	7543.7940
4	0	0.0000	0.000	0.0000

[44]:
```
df_matched_join["ATE"] = df_matched_join["re78_x"] - df_matched_join["re78_y"]
```

[45]:
```
df_matched_join["ATE"].mean()
```

[45]:
```
977.4715777837835
```