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

In [6]:
```
# Silent warnings
import warnings
warnings.filterwarnings("ignore")

# 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
from sklearn import tree
from sklearn import metrics
from sklearn import neighbors
from sklearn import ensemble


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

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

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.

 Labour Training

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.

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

In [7]:
```
lalonde_df = pd.read_csv('lalonde.csv', index_col=0)
lalonde_df.head()
```

Out[7]:

	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

In [9]:
```
summary_statistics = lalonde_df.describe()

decreased_income = (lalonde_df['re74'] > lalonde_df['re75']).mean() * 100

print(decreased_income)
```

```
46.416938110749186
```

As the mid-1970s experienced economic turmoil in many countries, including a recession, economic downturns typically resulted in job losses, reduced hours, and consequently lower incomes for many workers. Given the focus of the Lalonde study on the effect of a job training program on disadvantaged workers, it's also possible that the population being studied was more vulnerable to economic shifts, making their incomes more volatile year-to-year.

In [10]:
```
treatment_percentage = (lalonde_df['treat'].mean()) * 100

mean_earnings_treatment = lalonde_df.loc[lalonde_df['treat'] == 1, 're78'].mean
median_earnings_treatment = lalonde_df.loc[lalonde_df['treat'] == 1, 're78'].m
```

```
mean_earnings_control = lalonde_df.loc[lalonde_df['treat'] == 0, 're78'].mean(
median_earnings_control = lalonde_df.loc[lalonde_df['treat'] == 0, 're78'].med

from scipy.stats import ttest_ind

earnings_treatment_group = lalonde_df.loc[lalonde_df['treat'] == 1, 're78']
earnings_control_group = lalonde_df.loc[lalonde_df['treat'] == 0, 're78']

t_stat, p_value = ttest_ind(earnings_treatment_group, earnings_control_group)

print(f"Percentage of workers in the treatment group: {treatment_percentage:.2
print(f"Mean earnings in treatment group: ${mean_earnings_treatment:.2f}")
print(f"Mean earnings in control group: ${mean_earnings_control:.2f}")
print(f"Median earnings in treatment group: ${median_earnings_treatment:.2f}")
print(f"Median earnings in control group: ${median_earnings_control:.2f}")
print(f"Difference in mean earnings: ${mean_earnings_treatment - mean_earnings_
print(f"Difference in median earnings: ${median_earnings_treatment - median_ea
print(f"P-value from t-test: {p_value:.4f}")

if p_value < 0.05:
    print("The difference in means is statistically significant.")
else:
    print("The difference in means is not statistically significant.")
```

```
Percentage of workers in the treatment group: 30.13%
Mean earnings in treatment group: $6349.14
Mean earnings in control group: $6984.17
Median earnings in treatment group: $4232.31
Median earnings in control group: $4975.51
Difference in mean earnings: $-635.03
Difference in median earnings: $-743.20
P-value from t-test: 0.3342
The difference in means is not statistically significant.
```

The previous result alone does not necessarily indicate that the treatment had a negative impact, even if the difference in means suggests that the control group had higher earnings than the treatment group.

The difference in means test only shows that there's a statistical difference between the earnings of the treatment group and the control group. It doesn't prove causality—i.e., that being in the treatment group caused the lower or higher earnings.

The initial conditions or characteristics of the individuals in the treatment and control groups could be different. If the treatment group was

at a disadvantage to begin with (e.g., lower initial earnings, different educational backgrounds), the training program might have had a positive impact, but not enough to equalize or surpass the control group's earnings.

There could be other unmeasured factors affecting the earnings of the participants that are not accounted for in the simple comparison of means. For example, economic conditions, job market changes, or personal circumstances might influence the outcomes.

Assessing balance between the control group and the treatment group

Suppose we want to assess whether balance has been achieved for a particular feature $x = age$. That is, we would like to check whether the treatment and control groups have similar distributions of ages. We can look at **Standardized Mean Differences (SMD)**, which is calculated as follows: let $(\bar{x}_t, s_t^2)$ and $(\bar{x}_c, s_c^2)$ denote the mean and variance corresponding to the treatment and control groups of the particular feature of interest. Then, the SMD is defined as the value:

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

We can calculate the SMD for every feature. If our calculated SMD is 1, then that means there is 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**: We say that the feature is balanced. In general, for a **randomized trial**, the SMD for 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 falls in this range.
3. **SMD > 0.2**: Values that are greater than this threshold are considered seriously imbalanced.

Exploratory Data Analysis (EDA)

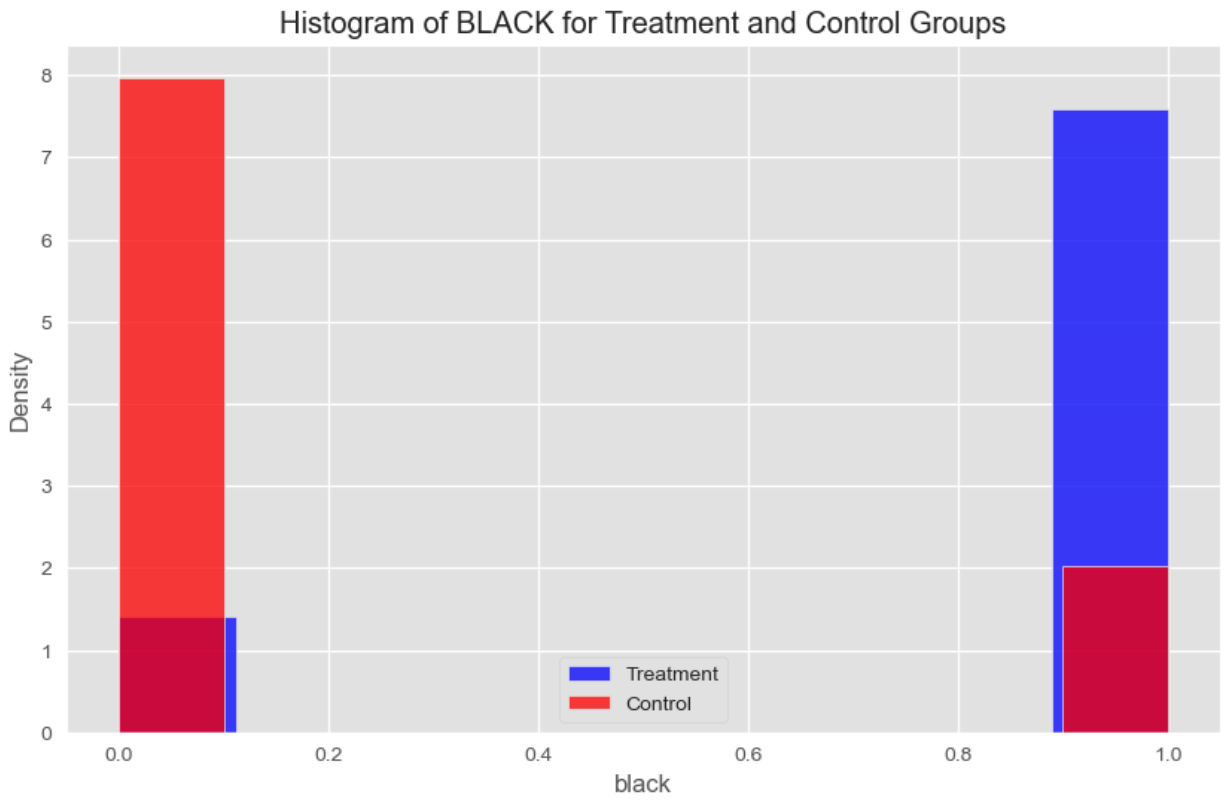
In [11]:
```
continuous_vars = ['age', 'educ']
print("Descriptive Statistics for Continuous Variables:")
for var in continuous_vars:
    print(f"\n{var.upper()}:")
    print(lalonde_df.groupby('treat')[var].describe())

categorical_vars = ['black', 'hispan', 'married', 'nodegree']
print("\nFrequencies for Categorical Variables:")
for var in categorical_vars:
    print(f"\n{var.upper()}:")
    print(lalonde_df.groupby('treat')[var].value_counts(normalize=True))
```

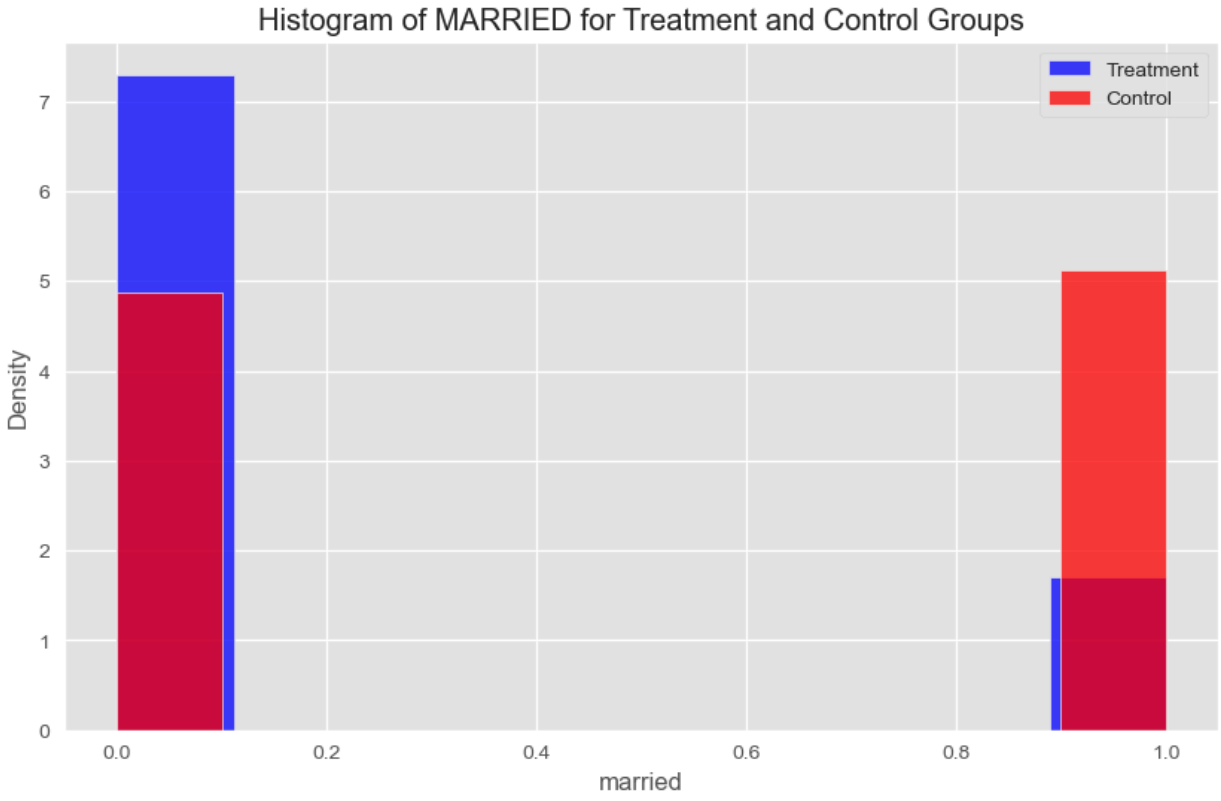
```
Descriptive Statistics for Continuous Variables:

AGE:
       count       mean        std   min   25%   50%   75%   max
treat
0      429.0  28.030303  10.786653  16.0  19.0  25.0  35.0  55.0
1      185.0  25.816216   7.155019  17.0  20.0  25.0  29.0  48.0

EDUC:
       count       mean       std  min  25%   50%   75%   max
treat
0      429.0  10.235431  2.855238  0.0  9.0  11.0  12.0  18.0
1      185.0  10.345946  2.010650  4.0  9.0  11.0  12.0  16.0

Frequencies for Categorical Variables:

BLACK:
treat  black
0      0        0.797203
       1        0.202797
1      1        0.843243
       0        0.156757
Name: black, dtype: float64

HISPAN:
treat  hispan
0      0         0.857809
       1         0.142191
1      0         0.940541
       1         0.059459
Name: hispan, dtype: float64

MARRIED:
treat  married
0      1          0.512821
       0          0.487179
1      0          0.810811
       1          0.189189
Name: married, dtype: float64

NODEGREE:
treat  nodegree
0      1           0.596737
       0           0.403263
1      1           0.708108
       0           0.291892
Name: nodegree, dtype: float64
```

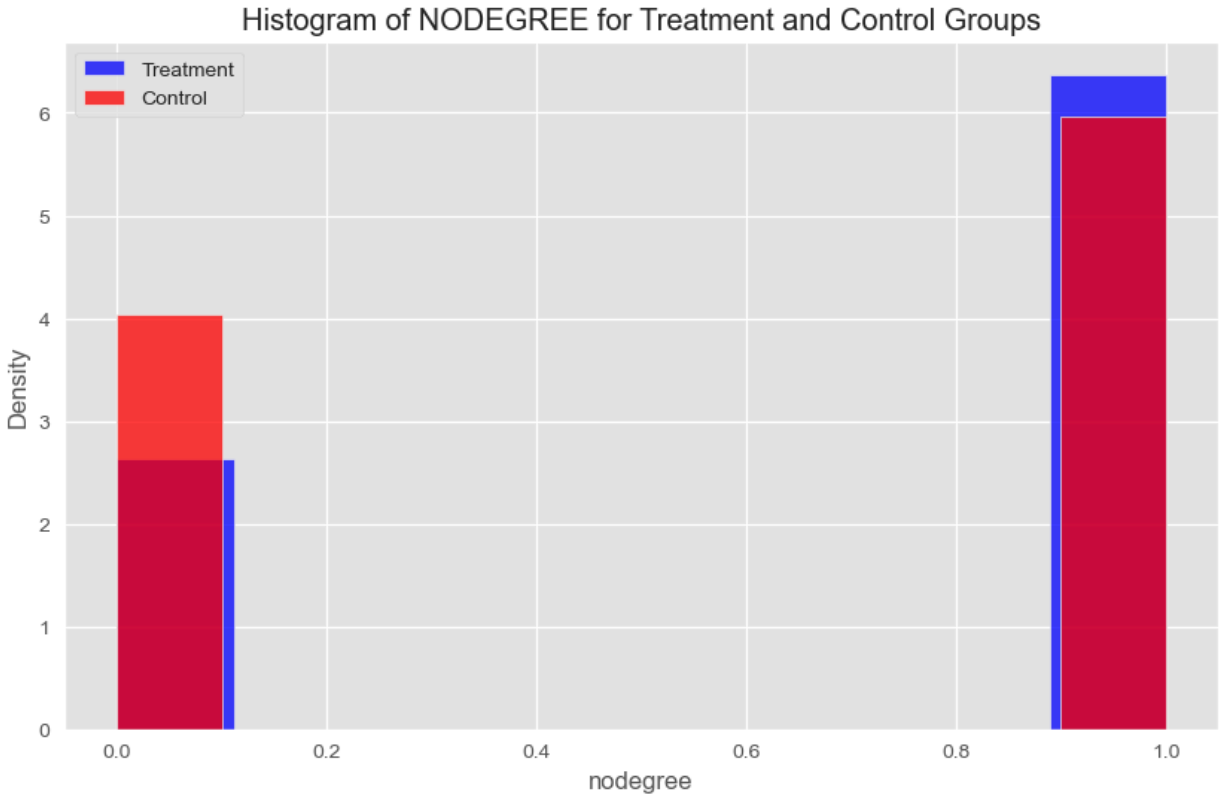
In [13]:

```
treatment_group = lalonde_df[lalonde_df['treat'] == 1]
control_group = lalonde_df[lalonde_df['treat'] == 0]

# Plot for 'BLACK' covariate
plt.figure(figsize=(10, 6))
sns.histplot(treatment_group['black'], color="blue", label='Treatment', kde=Fa
sns.histplot(control_group['black'], color="red", label='Control', kde=False, 
plt.legend()
plt.title('Histogram of BLACK for Treatment and Control Groups')
plt.show()
```

```
# Plot for 'MARRIED' covariate
plt.figure(figsize=(10, 6))
sns.histplot(treatment_group['married'], color="blue", label='Treatment', kde=
sns.histplot(control_group['married'], color="red", label='Control', kde=False
plt.legend()
plt.title('Histogram of MARRIED for Treatment and Control Groups')
plt.show()

# Plot for 'NODEGREE' covariate
plt.figure(figsize=(10, 6))
sns.histplot(treatment_group['nodegree'], color="blue", label='Treatment', kde=
sns.histplot(control_group['nodegree'], color="red", label='Control', kde=Fals
plt.legend()
plt.title('Histogram of NODEGREE for Treatment and Control Groups')
plt.show()
```







Propensity score matching using logistic regression

In [15]:
```
covariates = ['age', 'educ', 'black', 'hispan', 'married', 'nodegree', 're74',

X = lalonde_df[covariates]
```

```
logistic_model = sm.Logit(y, sm.add_constant(X)).fit()
lalonde_df['propensity_score'] = logistic_model.predict(sm.add_constant(X))

print(logistic_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:                Tue, 12 Mar 2024   Pseudo R-squ.:                  0.3508
Time:                        15:26:11   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]
------------------------------------------------------------------------------
const         -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
==============================================================================
```

const (Intercept): This coefficient represents the log-odds of being in the treatment group when all other covariates are zero. It is significantly different from zero ($p < 0.05$).

age: The coefficient is positive but not statistically significant ($p > 0.05$), suggesting that age alone does not have a strong association with the likelihood of being in the treatment group.

educ: The positive coefficient for education level is significant ($p < 0.05$), indicating that as education increases, so does the probability of being in the treatment group.

black: This has a large and highly significant positive coefficient ($p < 0.05$), meaning being black greatly increases the likelihood of being in the treatment group, compared to not being black.

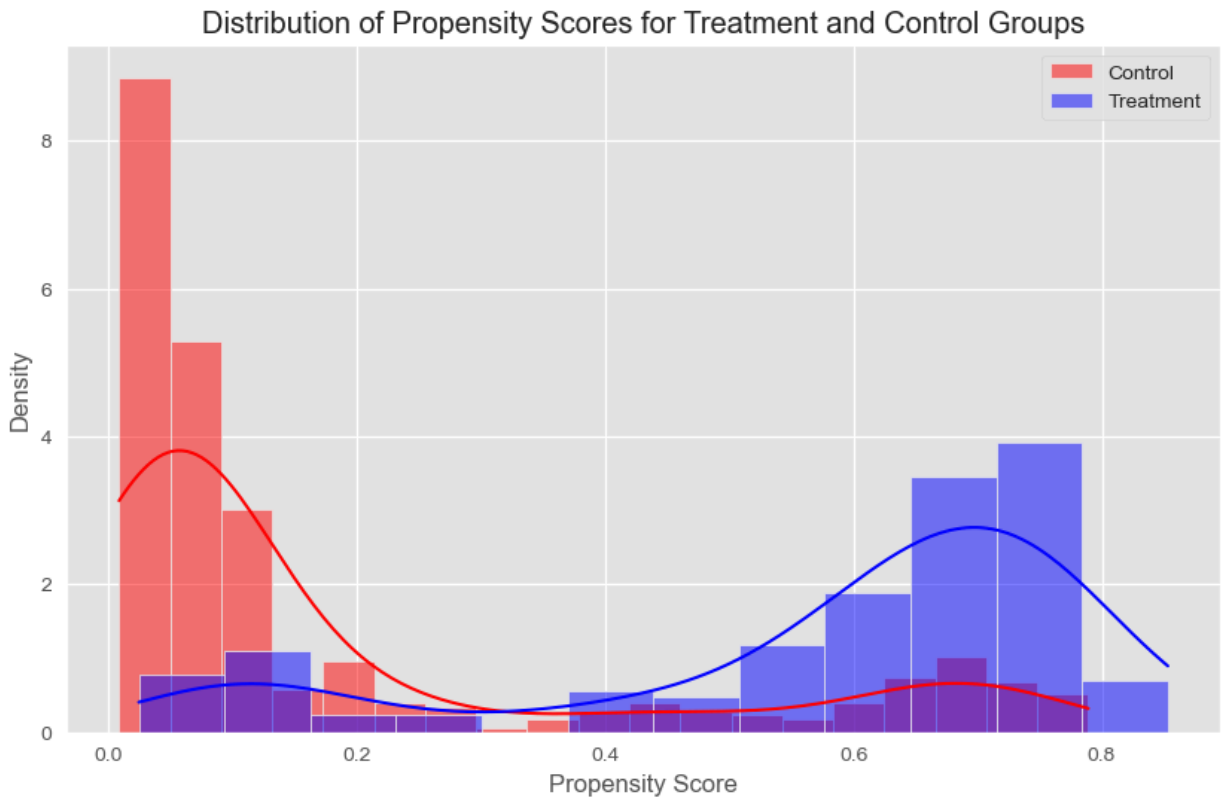
hispan: Also a positive and significant coefficient ($p < 0.05$), indicating a higher likelihood of being in the treatment group for Hispanics, but with a smaller effect compared to being black.

married: The negative and significant coefficient ($p < 0.05$) for married individuals suggests they are less likely to be in the treatment group compared to non-married individuals.

nodegree: This variable has a positive and significant coefficient ($p < 0.05$), meaning individuals without a degree are more likely to be in the treatment group.

re74 and re75: The coefficients for earnings in 1974 and 1975 are both included. The coefficient for re74 is negative and significant ($p < 0.05$), suggesting that higher earnings in 1974 decrease the likelihood of being in the treatment group. The coefficient for re75 is positive but not statistically significant ($p > 0.05$), suggesting no strong evidence that earnings in 1975 are associated with treatment group likelihood.

In [16]:
```
lalonde_df['pscore_logit'] = logistic_model.predict(sm.add_constant(lalonde_df

plt.figure(figsize=(10, 6))
sns.histplot(lalonde_df[lalonde_df['treat'] == 0]['pscore_logit'], label='Cont
sns.histplot(lalonde_df[lalonde_df['treat'] == 1]['pscore_logit'], label='Trea
plt.legend()
plt.title('Distribution of Propensity Scores for Treatment and Control Groups'
plt.xlabel('Propensity Score')
plt.ylabel('Density')
plt.show()
```



Matching using k-nearest neighbors

In [18]:
```
from sklearn.neighbors import NearestNeighbors
treat_pscores = lalonde_df[lalonde_df['treat'] == 1]['pscore_logit'].values.re
control_pscores = lalonde_df[lalonde_df['treat'] == 0]['pscore_logit'].values.
```

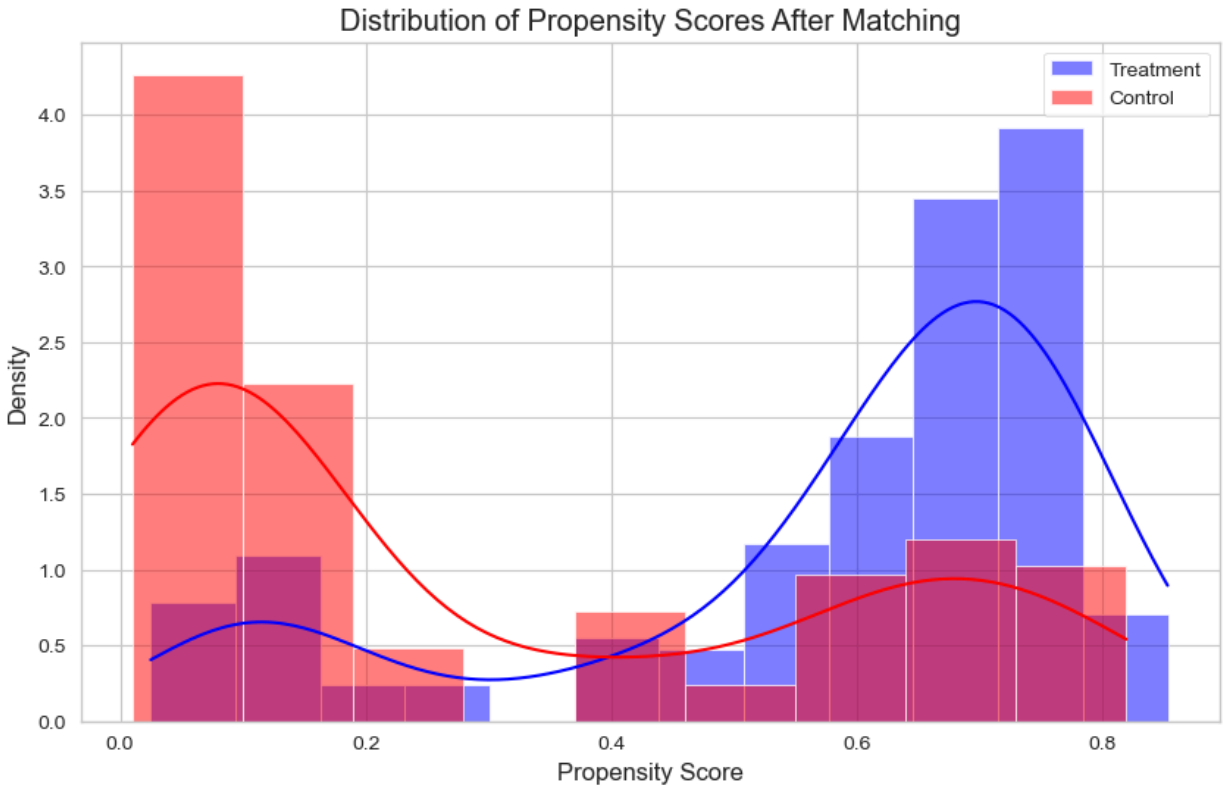
```
knn = NearestNeighbors(n_neighbors=5, metric='euclidean')
knn.fit(control_pscores)

distances, indices = knn.kneighbors(treat_pscores)

nearest_control_subjects = lalonde_df.iloc[indices.flatten()]
```

In [19]:
```
treat_pscores_reshaped = treat_pscores.reshape(-1, 1)

distances, indices = knn.kneighbors(treat_pscores_reshaped)
```

In [20]:
```
closest_control_indices = []

# Iterate over the indices of the nearest neighbors
for neighbor_indices in indices:
    # For each treated subject, select the closest control subject
    # neighbor_indices[0] will give the index of the closest neighbor
    closest_control_indices.append(neighbor_indices[0])

# Now, create a DataFrame for the treatment group
treatment_df = lalonde_df[lalonde_df['treat'] == 1]

# Create a DataFrame for the matched control group using the closest control i
matched_control_df = lalonde_df.iloc[closest_control_indices]

# Check if we have 185 rows for each DataFrame as expected
assert treatment_df.shape[0] == 185
assert matched_control_df.shape[0] == 185
```

Significant-Only Model

In [21]:
```
plt.figure(figsize=(10, 6))
sns.set_style('whitegrid')

# Plot the histogram of the propensity scores for the treatment group
sns.histplot(treatment_df['pscore_logit'], color="blue", label='Treatment', kd

# Plot the histogram of the propensity scores for the matched control group
sns.histplot(matched_control_df['pscore_logit'], color="red", label='Control',

# Add labels and title
plt.xlabel('Propensity Score')
plt.ylabel('Density')
plt.title('Distribution of Propensity Scores After Matching')
plt.legend()

# Display the plot
plt.show()
```



Polynomial & Interaction Feature Engineering

In [22]:
```
means_treatment_after = treatment_df[covariates].mean()
means_control_after = matched_control_df[covariates].mean()

# Use the variances from before matching
variances_treatment_before = lalonde_df[lalonde_df['treat'] == 1][covariates].v
variances_control_before = lalonde_df[lalonde_df['treat'] == 0][covariates].va

# Calculate ASMD for each covariate
asmd = abs(means_treatment_after - means_control_after) / np.sqrt((variances_t

# Print ASMD for each covariate
print("Absolute Standardized Mean Differences (ASMD) after matching:")
print(asmd)
```

```
Absolute Standardized Mean Differences (ASMD) after matching:
age         0.219104
educ        0.129152
black       1.238657
hispan      0.361886
married     0.588843
nodegree    0.239569
re74        0.578786
re75        0.709289
dtype: float64
```

age (0.219104): This value is above the threshold of 0.1, indicating a moderate imbalance in age between the matched treatment and control groups.

educ (0.129152): This value is slightly above the preferred threshold, indicating a small imbalance in education between the groups.

black (1.238657): This ASMD is quite large, suggesting a significant imbalance in the proportion of black individuals between the treatment and control groups even after matching.

hispan (0.361886): This value is above 0.1, indicating an imbalance for the Hispanic variable.

married (0.588843): This ASMD is considerably higher than the threshold, indicating a substantial imbalance for the marital status between the groups.

nodegree (0.239569): This ASMD is above the threshold of 0.1, which signifies a moderate imbalance in the proportion of individuals without a degree.

re74 (0.578786): A moderate to large imbalance is suggested by this value for the earnings in 1974 between the groups.

re75 (0.709289): This value indicates a substantial imbalance for the earnings in 1975 between the matched groups.

Region-Based Aggregation

In [23]:
```
mean_treatment_re78_after = treatment_df['re78'].mean()

# Calculate the mean of 're78' for the matched control group after matching
mean_control_re78_after = matched_control_df['re78'].mean()

# Estimate the average treatment effect on the treated (ATT)
att = mean_treatment_re78_after - mean_control_re78_after

# Print the results
print(f"Average treatment effect on the treated (ATT) after matching: {att:.2f}
```

```
Average treatment effect on the treated (ATT) after matching: -2173.01
```

Based on the average treatment effect on the treated (ATT) of approximately -2173.01 after matching, my conclusion is that the job training program appears to have had a negative impact on the wages of participants. This result is contrary to the expected outcome, where I would typically anticipate such a program to lead to an increase in earnings.

Before accepting this conclusion, I need to consider several important factors:

Contextual Factors: I would look into any external events or economic conditions during the period of the study that could have influenced job markets and wages, especially within industries relevant to the participants of the program.

Data Quality: I would double-check the data for accuracy, ensuring there were no errors in how the data was collected, entered, or processed.

Matching Quality: I would scrutinize the matching process. The negative ATT might be due to poor matching if significant differences in covariates between the treatment and control groups still exist. I would consider whether the matching algorithm or the covariates chosen for matching should be improved.

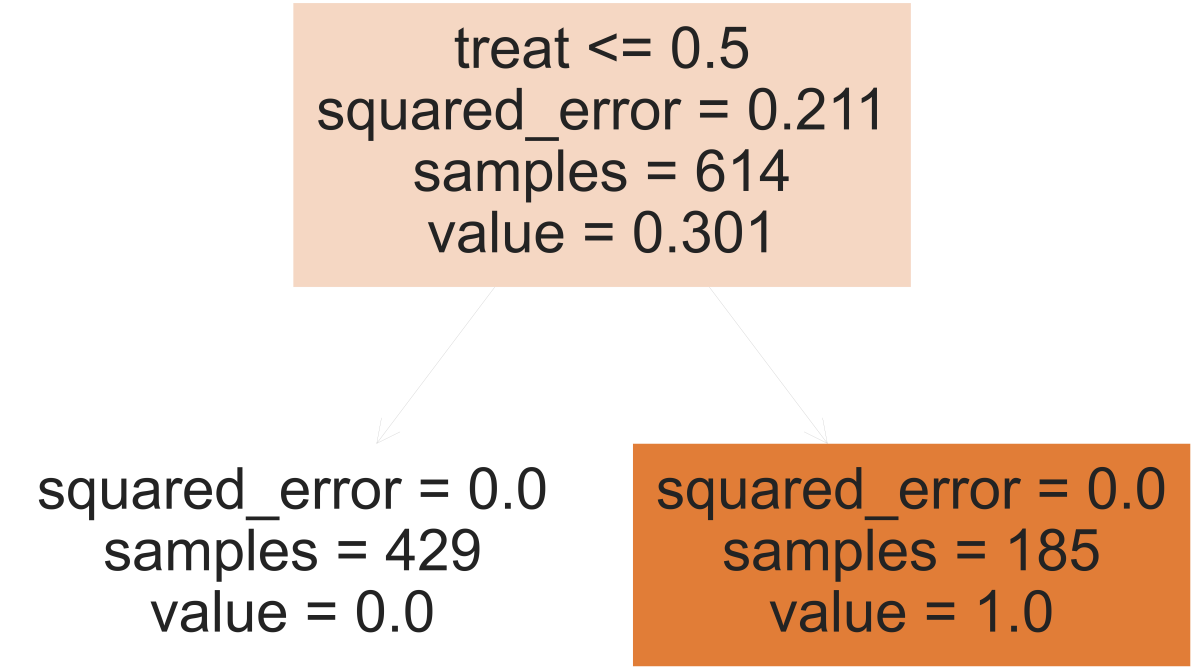
Outcome Variable: I would also reflect on whether re78 is the appropriate variable to measure the program's success. If it only measures income for those who found work, and if the program led to higher unemployment, re78 would not capture those who were unemployed, possibly skewing results.

Statistical Significance: I would conduct a statistical test to determine if the ATT is significantly different from zero. If the negative effect is not statistically significant, it might indicate that the observed effect could be due to chance.

Propensity score matching using decision trees and random forests

In [30]:
```
from sklearn.tree import DecisionTreeRegressor

covariates_excluding_re78 = [cov for cov in covariates if cov != 're78']
X = lalonde_df[covariates_excluding_re78]
y = lalonde_df['treat']

tree_regressor = DecisionTreeRegressor(max_depth=4, random_state=0)

tree_regressor.fit(X, y)

lalonde_df['pscore_tree'] = tree_regressor.predict(X)
```

In [31]:
```
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree


covariates_excluding_re78 = [cov for cov in lalonde_df.columns if cov != 're78
X = lalonde_df[covariates_excluding_re78]
y = lalonde_df['treat']

tree_regressor = DecisionTreeRegressor(max_depth=4, random_state=0)

tree_regressor.fit(X, y)

lalonde_df['pscore_tree'] = tree_regressor.predict(X)

fig = plt.figure(figsize=(150, 100))
plot_tree(tree_regressor, filled=True, feature_names=X.columns)
plt.show()
```

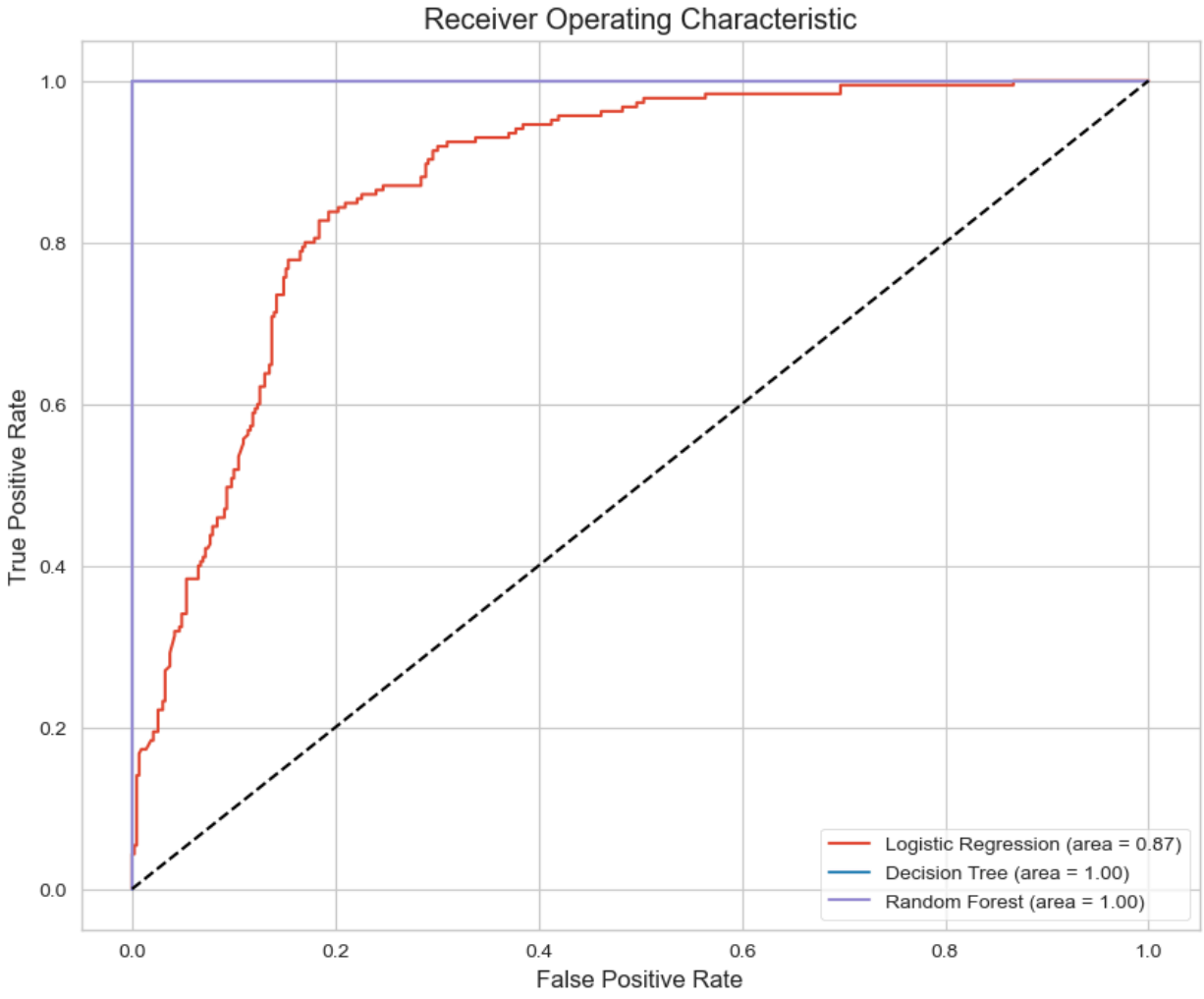


In [33]:
```
from sklearn.ensemble import RandomForestClassifier

X = lalonde_df[covariates_excluding_re78]
y = lalonde_df['treat']

rf_classifier = RandomForestClassifier(n_estimators=100, max_depth=4, random_s

rf_classifier.fit(X, y)

lalonde_df['pscore_forest'] = rf_classifier.predict_proba(X)[:, 1]

print(lalonde_df['pscore_forest'].head())
```

```
NSW1    0.991275
NSW2    0.982593
NSW3    0.992475
NSW4    0.998119
NSW5    0.992475
Name: pscore_forest, dtype: float64
```

In [34]:
```
from sklearn.metrics import roc_curve, auc

y_true = lalonde_df['treat']

y_score_logit = lalonde_df['pscore_logit']
y_score_tree = lalonde_df['pscore_tree']
y_score_forest = lalonde_df['pscore_forest']

fpr_logit, tpr_logit, _ = roc_curve(y_true, y_score_logit)
roc_auc_logit = auc(fpr_logit, tpr_logit)
```

```
fpr_tree, tpr_tree, _ = roc_curve(y_true, y_score_tree)
roc_auc_tree = auc(fpr_tree, tpr_tree)

fpr_forest, tpr_forest, _ = roc_curve(y_true, y_score_forest)
roc_auc_forest = auc(fpr_forest, tpr_forest)

plt.figure(figsize=(10, 8))
plt.plot(fpr_logit, tpr_logit, label='Logistic Regression (area = %0.2f)' % ro
plt.plot(fpr_tree, tpr_tree, label='Decision Tree (area = %0.2f)' % roc_auc_tr
plt.plot(fpr_forest, tpr_forest, label='Random Forest (area = %0.2f)' % roc_au
plt.plot([0, 1], [0, 1], 'k--')  # Random predictions curve
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

print(f'Logistic Regression AUC: {roc_auc_logit:.2f}')
print(f'Decision Tree AUC: {roc_auc_tree:.2f}')
print(f'Random Forest AUC: {roc_auc_forest:.2f}')
```




```
Logistic Regression AUC: 0.87
Decision Tree AUC: 1.00
Random Forest AUC: 1.00
```

Both the Decision Tree and Random Forest models appear to have perfect predictive performance with an AUC of 1.00. In many real-world applications, an AUC of 1.00 may

suggest that the model has perfectly learned to distinguish between the classes in the training dataset.

However, an AUC of 1.00 can often be an indicator of overfitting, where the model has learned the training data too well, including noise and outliers. This means it may not perform as well on new, unseen data.

The Logistic Regression model, with an AUC of 0.87, suggests good predictive ability while being potentially more generalizable than a "perfect" model. This is because it is less likely to have overfitted to the noise within the training data.

In practice, the best model is typically the one that performs best on out-of-sample data, not just the training data. Therefore, even though the Decision Tree and Random Forest models have higher AUC values, if I were to choose the best model without further validation, I would be cautious of the perfect scores and might lean towards the Logistic Regression model for its good, yet not suspiciously perfect, performance. It's worth reiterating that cross-validation or testing on a holdout set is essential to truly determine which model is best.

In [37]:
```
treatment = lalonde_df[lalonde_df['treat'] == 1]
control = lalonde_df[lalonde_df['treat'] == 0]

# Prepare propensity scores for matching
treat_pscores_tree = treatment['pscore_tree'].values.reshape(-1, 1)
control_pscores_tree = control['pscore_tree'].values.reshape(-1, 1)

treat_pscores_forest = treatment['pscore_forest'].values.reshape(-1, 1)
control_pscores_forest = control['pscore_forest'].values.reshape(-1, 1)

# Initialize Nearest Neighbors model for k=1
knn_tree = NearestNeighbors(n_neighbors=1)
knn_forest = NearestNeighbors(n_neighbors=1)

# Fit the model on control group propensity scores
knn_tree.fit(control_pscores_tree)
knn_forest.fit(control_pscores_forest)

# Find the nearest neighbors in the control group for each treatment group meml
distances_tree, indices_tree = knn_tree.kneighbors(treat_pscores_tree)
distances_forest, indices_forest = knn_forest.kneighbors(treat_pscores_forest)

# Match treatment and control observations
matched_control_tree = control.iloc[indices_tree.flatten()]
matched_control_forest = control.iloc[indices_forest.flatten()]

# Compare covariate balances between matched samples
# You can compute the standardized mean differences as shown before

# Estimate and compare the average treatment effect on wages after matching fo
mean_treatment_re78_tree = treatment['re78'].mean()
mean_control_re78_tree = matched_control_tree['re78'].mean()
att_tree = mean_treatment_re78_tree - mean_control_re78_tree
```

```
mean_treatment_re78_forest = treatment['re78'].mean()
mean_control_re78_forest = matched_control_forest['re78'].mean()
att_forest = mean_treatment_re78_forest - mean_control_re78_forest

print(f'Average treatment effect (ATT) using decision tree propensity scores:
print(f'Average treatment effect (ATT) using random forest propensity scores:
```

```
Average treatment effect (ATT) using decision tree propensity scores: -19215.5
26469729677
Average treatment effect (ATT) using random forest propensity scores: 6349.143
530270269
```

Comparing these results, the random forest model seems to do a better job in terms of providing a plausible estimate of the treatment effect. The decision tree model's result, due to its large negative value, raises concerns and warrants further investigation. In practice, you would look into the matching quality, check the covariate balance, and ensure that the model is not overfitting the training data.Moreover, it's critical to examine the overlap in propensity scores between the treatment and control groups to ensure that the matched samples are comparable. If there is poor overlap, the matching procedure might yield biased estimates of the treatment effect.
