original-field-experiment

March 5, 2021

[2]:
```
import pandas as pd
from datetime import datetime
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
```

[3]:
```
from pymatch.Matcher import Matcher
```

[4]:
```
from sklearn.linear_model import LogisticRegression
```

[5]:
```
import seaborn as sns
```

[155]:
```
T_e2e_comp = pd.read_csv('order_e2e_post.csv')
T_other_comp = pd.read_csv('order_control_post.csv')
```

[741]:
```
print('SKU num', T_e2e_comp['item_sku_id'].nunique())
print('DC num', T_e2e_comp['delv_center_num'].nunique())
```

```
SKU num 1052
DC num 12
```

[656]:
```
len(T_e2e_comp)
```

[656]: 6097

[726]:
```
skus = np.concatenate((T_other_comp['item_sku_id'].values,␣
↪T_e2e_comp['item_sku_id'].values))
```

[732]:
```
print('SKU num', len(np.unique(skus)))
```

```
SKU num 9308
```

[713]:
```
len(T_e2e_comp)+len(T_other_comp)
```

[713]: 61430

1 Propensity score matching

[660]:
```
N_e2e = len(T_e2e_comp)
vlt_e2e = []
for i in range(N_e2e):
    if type(T_e2e_comp.vlt[i]) == int:
        vlt_e2e.append(T_e2e_comp.vlt[i])
    else:
        vlt_e2e.append(T_e2e_comp.vlt[i].days)
```

[661]:
```
N_other = len(T_other_comp0)
vlt_other = []
for i in range(N_other):
    if type(T_other_comp0.vlt[i]) == int:
        vlt_other.append(T_other_comp0.vlt[i])
    else:
        vlt_other.append(T_other_comp0.vlt[i].days)
```

[662]:
```
T_e2e_comp['vlt_num'] = vlt_e2e
T_other_comp0['vlt_num'] = vlt_other
```

[663]:
```
test = T_e2e_comp[['ave_demand', 'vlt_num']]
test['e2e'] = np.ones(N_e2e)
control = T_other_comp0[['ave_demand', 'vlt_num']]
control['e2e'] = np.zeros(N_other)
```

```
/Users/meng/opt/anaconda3/envs/python3/lib/python3.7/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/meng/opt/anaconda3/envs/python3/lib/python3.7/site-
packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

1.0.1 Compare T_e2e_comp & T_other_comp

[664]:
```
control0 = control
```

[667]:
```
N_e2e = len(T_e2e_comp)
print('E2E Sample Number:', N_e2e)
index = np.arange(len(control0))
T_other_comp = T_other_comp0.iloc[index]
```

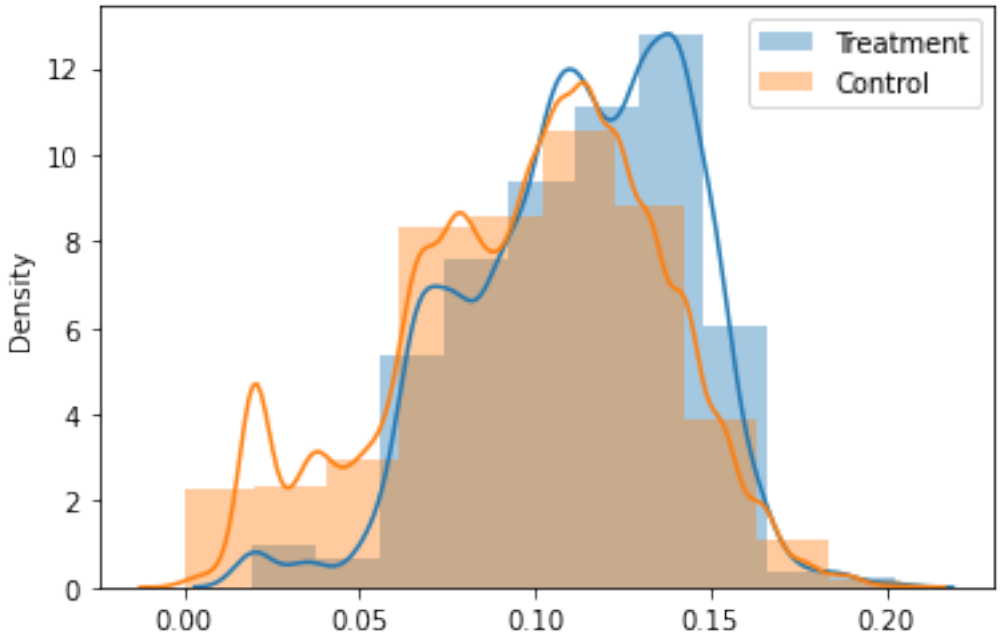
```
E2E Sample Number: 6097
```

[669]:
```
origin_data = pd.concat([test, control0], ignore_index=True)
```

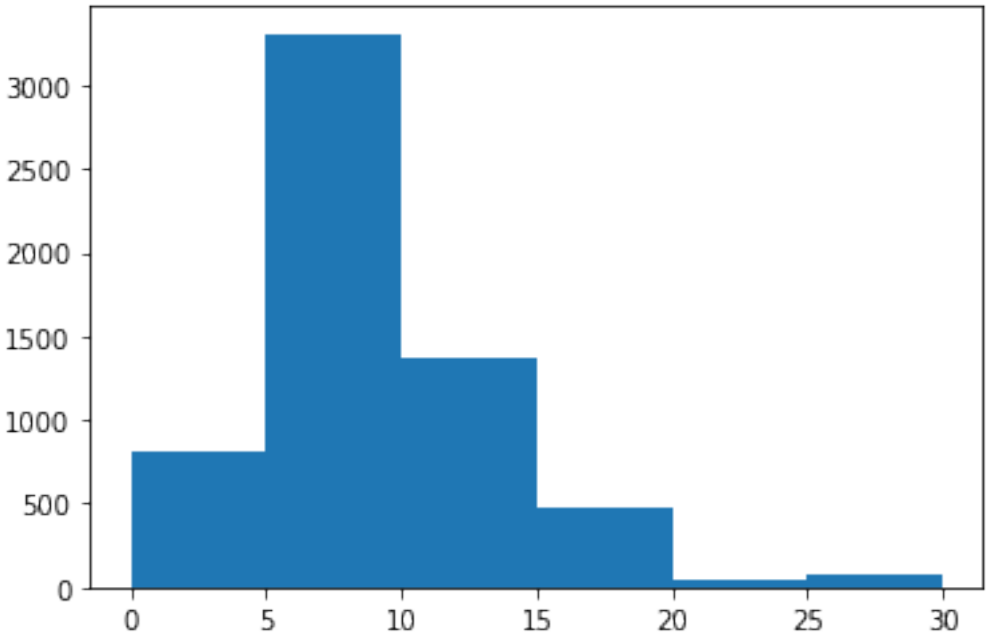
[751]:
```
#calculate original propensity score:
propensity = LogisticRegression()
propensity = propensity.fit(origin_data[['ave_demand', 'vlt_num']], origin_data.
 ↪e2e)
pscore = propensity.predict_proba(origin_data[['ave_demand', 'vlt_num']])[:,1]␣
 ↪# The predicted propensities by the model
```

[756]:
```
sns.distplot(pscore[:N_e2e], hist = True, bins = 10, label = 'Treatment')
sns.distplot(pscore[N_e2e:], hist = True, bins = 10, label = 'Control')
plt.legend()
# plt.show()
plt.savefig('prop_score_prep.pdf')
```

```
/Users/meng/opt/anaconda3/envs/python3/lib/python3.7/site-
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/meng/opt/anaconda3/envs/python3/lib/python3.7/site-
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



[673]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors

def get_matching_pairs(treated_df, non_treated_df):

    treated_x = treated_df.values
    non_treated_x = non_treated_df.values

    nbrs = NearestNeighbors(n_neighbors=1, algorithm='auto').fit(non_treated_x)
    distances, indices = nbrs.kneighbors(treated_x)
#     print(indices.reshape(-1,3)[:,2])
    indices = indices.reshape(indices.shape[0])
    return indices

matched_idx = get_matching_pairs(test, control0)

T_matched_comp = T_other_comp.iloc[matched_idx]

plt.hist(T_matched_comp.vlt_num, bins = 6)
plt.show()
```



[674]:
```
control = control0.iloc[matched_idx]
N_other = len(T_other_comp0)
N_match = N_e2e
print('JD method Sample Number:', N_match)
h = 1
b = 9
```

```
JD method Sample Number: 6097
```

[675]:
```
prop_data = pd.concat([test, control], ignore_index=True)
```

[676]:
```
N_e2e
```

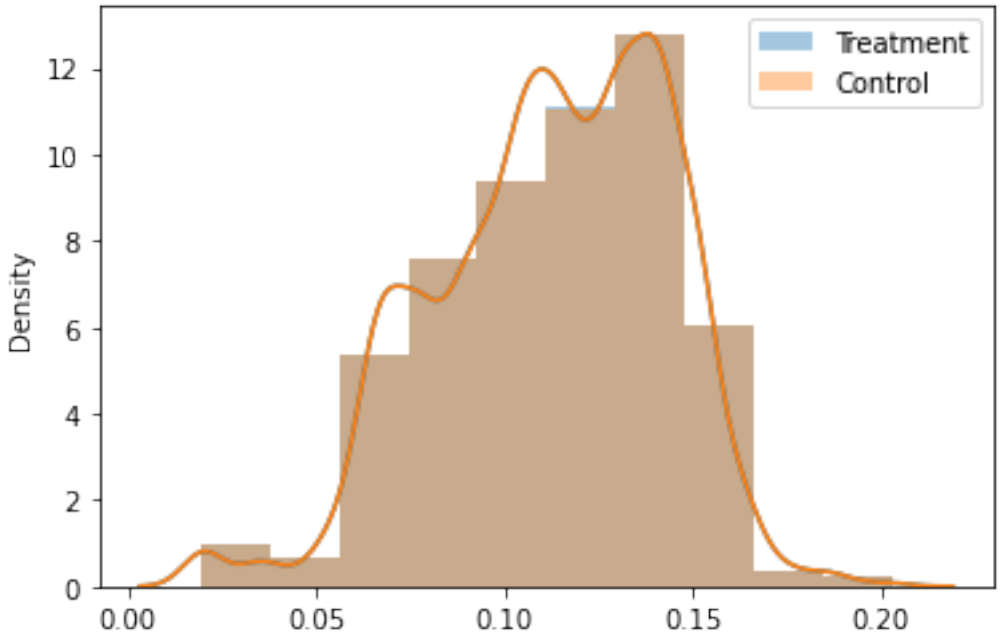
[676]: 6097

[757]:
```
#calculate propensity score:
pscore = propensity.predict_proba(prop_data[['ave_demand', 'vlt_num']])[:,1] #␣
 ↪The predicted propensities by the model
```

[760]:
```
sns.distplot(pscore[:N_e2e], hist = True, bins = 10, label = 'Treatment')
sns.distplot(pscore[N_e2e:], hist = True, bins = 10, label = 'Control')
plt.legend()
plt.savefig('prop_score2.pdf')
plt.show()
```

```
/Users/meng/opt/anaconda3/envs/python3/lib/python3.7/site-
```

```
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/meng/opt/anaconda3/envs/python3/lib/python3.7/site-
packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



[682]:
```
# calculate inventory metric
e2e_holding_cost =  []
e2e_stockout_cost = []
e2e_total_cost = []
e2e_turnover = []
e2e_stockout_ratio = []
for n in range(N_e2e):
    inv = T_e2e_comp.iloc[n].test_inv
    holding_cost = 0
    stockout_cost = 0
    stockout_day = 0
    T = len(T_e2e_comp.iloc[n].test_demand)
    for t in range(T):
        cur_inv = inv[t]
```

```
        if(cur_inv>=0):
            holding_cost += h*cur_inv
        else:
            stockout_cost += -b*cur_inv
#             if(T_e2e_comp.iloc[n].test_demand[t]<=0):
            stockout_day+=1
    turnover = T_e2e_comp.iloc[n].ave_inv/T_e2e_comp.iloc[n].ave_demand
#     turnover = np.maximum(0, T_e2e_comp.iloc[n].ave_inv/T_e2e_comp.iloc[n].
↪ave_demand)
    stockout_ratio = stockout_day/T

    e2e_holding_cost.append(holding_cost)
    e2e_stockout_cost.append(stockout_cost)
    e2e_total_cost.append(holding_cost+stockout_cost)
    e2e_turnover.append(turnover)
    e2e_stockout_ratio.append(stockout_ratio)
```

[683]:
```
print('Algorithm A Average Holding cost: ', np.mean(e2e_holding_cost))
print('Algorithm A Average Stockout cost: ', np.mean(e2e_stockout_cost))
print('Algorithm A Average Total cost: ', np.mean(e2e_total_cost))
# e2e_turnover1 = [max(0, item) for item in e2e_turnover]
print('Algorithm A Average Turnover rate: ', np.mean(e2e_turnover))
print('Algorithm A Average Stockout rate: ', np.mean(e2e_stockout_ratio))
```

```
Algorithm A Average Holding cost:  598.2009184845006
Algorithm A Average Stockout cost:  496.344759717894
Algorithm A Average Total cost:  1094.5456782023946
Algorithm A Average Turnover rate:  18.592980603044012
Algorithm A Average Stockout rate:  0.16808800001671162
```

[684]:
```
T_other_comp0.to_csv('order_other_useinpaper_0927.csv')
T_matched_comp.to_csv('order_matche_useinpaper_0927.csv')
```

[685]:
```
T_e2e_comp.to_csv('order_e2e_useinpaper_0927.csv')
```

[686]:
```
# calculate inventory metric
jd_holding_cost =  []
jd_stockout_cost = []
jd_total_cost = []
jd_turnover = []
jd_stockout_ratio = []
for n in range(N_match):
    inv = T_matched_comp.iloc[n].test_inv
    holding_cost = 0
    stockout_cost = 0
    stockout_day = 0
    T =  len(T_matched_comp.iloc[n].test_demand)
```

```
    for t in range(T):
        cur_inv = inv[t]
        if(cur_inv>=0):
            holding_cost += h*cur_inv
        else:
            stockout_cost += -b*cur_inv
#             if(T_matched_comp.iloc[n].test_demand[t]<=0):
#             if(cur_inv<=0):
            stockout_day+=1
    turnover = T_matched_comp.iloc[n].ave_inv/T_matched_comp.iloc[n].ave_demand
    stockout_ratio = stockout_day/T

    jd_holding_cost.append(holding_cost)
    jd_stockout_cost.append(stockout_cost)
    jd_total_cost.append(holding_cost+stockout_cost)
    jd_turnover.append(turnover)
    jd_stockout_ratio.append(stockout_ratio)
```

[687]:
```
print('Algorithm B Average Holding cost: ', np.mean(jd_holding_cost))
print('Algorithm B Average Holding Stockout cost: ', np.mean(jd_stockout_cost))
print('Algorithm B Average Total cost: ', np.mean(jd_total_cost))
jd_turnover1 = [max(0, item) for item in jd_turnover]
print('Algorithm B Average Turnover rate: ', np.mean(jd_turnover1))
print('Algorithm B Average Stockout rate: ', np.mean(jd_stockout_ratio))
```

```
Algorithm B Average Holding cost:  809.5422338855175
Algorithm B Average Holding Stockout cost:  1027.9293094964737
Algorithm B Average Total cost:  1837.471543381991
Algorithm B Average Turnover rate:  20.393857811615447
Algorithm B Average Stockout rate:  0.2561209373197818
```

[688]:
```
x_demand = np.concatenate((T_e2e_comp.ave_demand, T_matched_comp.ave_demand),␣
↪axis = 0)
```

[689]:
```
vlt_e2e = []
for i in range(N_e2e):
    if type(T_e2e_comp.vlt[i]) == int:
        vlt_e2e.append(T_e2e_comp.vlt[i])
    else:
        vlt_e2e.append(T_e2e_comp.vlt[i].days)
```

[690]:
```
vlt_match = T_matched_comp.vlt.dt.days.values
# for i in range(N_other):
# #     if type(T_other_comp.vlt[i]) == int:
# #         vlt_other.append(T_other_comp.vlt[i])
# #     else:
```

```
#     print(i)
#     vlt_other.append(T_other_comp.vlt[i].days)
```

[691]:
```
x_vlt = np.concatenate((vlt_e2e,vlt_match), axis = 0)

x_e2e = np.concatenate((np.ones(N_e2e), np.zeros(N_e2e)), axis = 0)
```

[693]:
```
ot_e2e = []
for i in range(N_e2e):
    ot_e2e.append(datetime.strptime(T_e2e_comp.create_tm[i], '%Y-%m-%d').
 ↪toordinal())
```

[694]:
```
ot_other = []
for i in range(N_other):
    ot_other.append(datetime.strptime(T_other_comp.create_tm[i], '%Y-%m-%d').
 ↪toordinal())
```

[695]:
```
ot_match = []
for i in range(N_e2e):
#     print(matched_idx[i])
    ot_match.append(ot_other[matched_idx[i]])
```

[696]:
```
x_ot = np.concatenate((ot_e2e, ot_match), axis = 0)
```

[697]:
```
x_oe = np.multiply(x_ot,x_e2e)
```

[698]:
```
import statsmodels.api as sm

from sklearn import preprocessing
```

[700]:
```
# t-test holding cost:
a = np.array(e2e_holding_cost)
b = np.array(jd_holding_cost)
t, p = stats.ttest_ind(a,b)
u,p2 = stats.mannwhitneyu(a,b)
print("t = " , str(t))
print("p = " , str(p))
print("u = " , str(u))
print("p2 = " , str(p2))
```

```
t =  -14.75484521783182
p =  7.54820122414664e-49
u =  17856434.5
p2 =  8.584230533160078e-05
```

[701]:
```
#linear regression:
y = np.concatenate((a,b), axis = 0)
df_lr = pd.DataFrame({'y': y, 'e2e':x_e2e, 'vlt':x_vlt, 'd':x_demand, 'ot':␣
 ↪x_ot})
X = df_lr.drop('y',1)    #Feature Matrix
y = df_lr['y']           #Target Variable
```

[703]:
```
model = sm.OLS(y, X)
lr =model.fit()
print(lr.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                        y   R-squared (uncentered):
0.458
Model:                              OLS   Adj. R-squared (uncentered):
0.458
Method:                   Least Squares   F-statistic:
2579.
Date:                  Sun, 04 Oct 2020   Prob (F-statistic):
0.00
Time:                          17:30:34   Log-Likelihood:
-98553.
No. Observations:                12194   AIC:
1.971e+05
Df Residuals:                    12190   BIC:
1.971e+05
Df Model:                            4
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
e2e         -211.3302     14.184    -14.900      0.000    -239.132    -183.528
vlt           -4.3977      1.621     -2.713      0.007      -7.575      -1.221
d             92.2846      6.195     14.895      0.000      80.140     104.429
ot             0.0009   2.97e-05     30.627      0.000       0.001       0.001
==============================================================================
Omnibus:                     3233.352   Durbin-Watson:                   1.598
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7142.429
Skew:                           1.536   Prob(JB):                         0.00
Kurtosis:                       5.150   Cond. No.                     1.48e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
```

[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 1.48e+06. This might indicate that there are strong multicollinearity or other numerical problems.

[704]:
```
# t-test stockout cost:
a = np.array(e2e_stockout_cost)
b = np.array(jd_stockout_cost)
t, p = stats.ttest_ind(a,b)
u,p2 = stats.mannwhitneyu(a,b)
print("t = " , str(t))
print("p = " , str(p))
print("u = " , str(u))
print("p2 = " , str(p2))
```

```
t =  -19.787436442736016
p =  8.425421360691402e-86
u =  14959338.0
p2 =  5.90363517221594e-89
```

[705]:
```
y = np.concatenate((a,b), axis = 0)
model = sm.OLS(y, X)
lr =model.fit()
print(lr.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                      y   R-squared (uncentered):
0.293
Model:                            OLS   Adj. R-squared (uncentered):
0.293
Method:                 Least Squares   F-statistic:
1265.
Date:                Sun, 04 Oct 2020   Prob (F-statistic):
0.00
Time:                        17:30:34   Log-Likelihood:
-1.0581e+05
No. Observations:               12194   AIC:
2.116e+05
Df Residuals:                   12190   BIC:
2.117e+05
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
e2e         -531.5254     25.714    -20.671      0.000    -581.929    -481.122
```

```
vlt           40.9508      2.939     13.936      0.000      35.191      46.711
d            358.8603     11.232     31.949      0.000     336.844     380.877
ot         -1.683e-05   5.38e-05     -0.313      0.754      -0.000    8.86e-05
==============================================================================
Omnibus:                     5483.805   Durbin-Watson:                   1.616
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            25386.552
Skew:                           2.219   Prob(JB):                         0.00
Kurtosis:                       8.502   Cond. No.                     1.48e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 1.48e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[706]:
```
# t-test total cost:
a = np.array(e2e_total_cost)
b = np.array(jd_total_cost)
t, p = stats.ttest_ind(a,b)
u,p2 = stats.mannwhitneyu(a,b)
print("t = " , str(t))
print("p = " , str(p))
print("u = " , str(u))
print("p2 = " , str(p2))
```

```
t =  -28.541731875091184
p =  1.6759264191764458e-173
u =  12850273.5
p2 =  9.569863668036659e-192
```

[707]:
```
y = np.concatenate((a,b), axis = 0)
model = sm.OLS(y, X)
lr =model.fit()
print(lr.summary())
```

```
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:                      y   R-squared (uncentered):
0.589
Model:                            OLS   Adj. R-squared (uncentered):
0.589
Method:                 Least Squares   F-statistic:
4363.
Date:                Sun, 04 Oct 2020   Prob (F-statistic):
```

```
0.00
Time:                        17:30:38   Log-Likelihood:
-1.0508e+05
No. Observations:               12194   AIC:
2.102e+05
Df Residuals:                   12190   BIC:
2.102e+05
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
e2e          -742.8556     24.234    -30.653      0.000    -790.359    -695.353
vlt            36.5530      2.769     13.199      0.000      31.125      41.982
d             451.1449     10.586     42.618      0.000     430.395     471.895
ot              0.0009   5.07e-05     17.593      0.000       0.001       0.001
==============================================================================
Omnibus:                     3512.605   Durbin-Watson:                   1.720
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10278.098
Skew:                           1.508   Prob(JB):                         0.00
Kurtosis:                       6.337   Cond. No.                     1.48e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 1.48e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[708]:
```
# t-test turnover:
a = np.array(e2e_turnover1)
b = np.array(jd_turnover1)
t, p = stats.ttest_ind(a,b)
u,p2 = stats.mannwhitneyu(a,b)
print("t = " , str(t))
print("p = " , str(p))
print("u = " , str(u))
print("p2 = " , str(p2))
```

```
t =  -2.5707025758472155
p =  0.010161000379500061
u =  16874670.0
p2 =  4.657049264673571e-19
```

[709]:
```
y = np.concatenate((a,b), axis = 0)
model = sm.OLS(y, X)
lr =model.fit()
print(lr.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                          y   R-squared (uncentered):
0.451
Model:                                OLS   Adj. R-squared (uncentered):
0.451
Method:                     Least Squares   F-statistic:
2508.
Date:                   Sun, 04 Oct 2020   Prob (F-statistic):
0.00
Time:                           17:30:50   Log-Likelihood:
-55805.
No. Observations:                  12194   AIC:
1.116e+05
Df Residuals:                      12190   BIC:
1.116e+05
Df Model:                              4
Covariance Type:               nonrobust
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
e2e           -1.1559      0.426     -2.714      0.007      -1.991      -0.321
vlt           -0.4960      0.049    -10.190      0.000      -0.591      -0.401
d             -6.7993      0.186    -36.547      0.000      -7.164      -6.435
ot          5.112e-05   8.91e-07     57.394      0.000    4.94e-05    5.29e-05
==============================================================================
Omnibus:                     6008.358   Durbin-Watson:                   1.592
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            48627.378
Skew:                           2.221   Prob(JB):                         0.00
Kurtosis:                      11.716   Cond. No.                     1.48e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 1.48e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[710]:
```
# t-test stockout:
a = np.array(e2e_stockout_ratio)
b = np.array(jd_stockout_ratio)
t, p = stats.ttest_ind(a,b)
u,p2 = stats.mannwhitneyu(a,b)
print("t = " , str(t))
print("p = " , str(p))
print("u = " , str(u))
print("p2 = " , str(p2))
```

```
t =  -16.775774867045808
p =  1.8378082008489155e-62
u =  15449754.5
p2 =  4.345803398624435e-67
```

[711]:
```
y = np.concatenate((a,b), axis = 0)
model = sm.OLS(y, X, hasconst=True)
lr =model.fit()
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.054
Model:                            OLS   Adj. R-squared:                  0.054
Method:                 Least Squares   F-statistic:                     234.1
Date:                Sun, 04 Oct 2020   Prob (F-statistic):          1.05e-147
Time:                        17:30:55   Log-Likelihood:                -1993.4
No. Observations:               12194   AIC:                             3995.
Df Residuals:                   12190   BIC:                             4025.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
e2e           -0.0880      0.005    -17.054      0.000      -0.098      -0.078
vlt            0.0101      0.001     17.115      0.000       0.009       0.011
d              0.0294      0.002     13.037      0.000       0.025       0.034
ot          1.533e-07   1.08e-08     14.205      0.000    1.32e-07    1.75e-07
==============================================================================
Omnibus:                     2227.134   Durbin-Watson:                   1.377
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3642.060
Skew:                           1.286   Prob(JB):                         0.00
Kurtosis:                       3.741   Cond. No.                     1.48e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

```
[2] The condition number is large, 1.48e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[828]:
```
# dump e2e
import pickle
outcome_e2e = [e2e_holding_cost, e2e_stockout_cost, e2e_total_cost,␣
 ↪e2e_turnover1, e2e_stockout_ratio]
pickle.dump(outcome_e2e, open("e2e_postexp.pkl", 'wb'))
# dump jd
outcome_jd = [jd_holding_cost, jd_stockout_cost, jd_total_cost, jd_turnover,␣
 ↪jd_stockout_ratio]
pickle.dump(outcome_jd, open("jd_postexp.pkl", 'wb'))
```

[ ]:
```

```