Econ 434 Final Project: Is Uber a complement or substitute to public transit?

Maxwell Speil and Yiran Sun
5/28/2021

In a simple OLS, adding year and agency fixed effects increases the coefficient on both the uber dummy and level of uber penetration variables to a significant degree.

	D=dummy_Coef	D=dummy_SE	D=dummy_95%CI	D=pen_Coef	D=pen_SE	D=pen_95%CI
OLS 1	0.025221	0.011457	(0.0028,0.0477)	0.004252	0.000850	(0.0026,0.0059)
OLS 2	2.159200	2.443877	(-2.6308,6.9492)	3.369757	2.018807	(-0.5871,7.3266)

	D=dummy	D=dummy*Above_med	D=pen	D=pen*Above_med
OLS3_pop	72.225(3.0774)	-89.3685(2.8805)	22.371(2.3085)	-31.1851(1.6913)
OLS4_rides	1259.1969(59.7948)	-54.1618(47.1068)	11.1466(0.9964)	-1.3073(1.2685)

In regressions 5 and 6, we replace the OLS form used by Hall, Palsson, and Price, with LASSO, and in regressions 7 and 8 we upgrade further to double lasso. In the single lasso regression, with see a positive coefficient on the Uber dummy but a negative coefficient on the Uber penetration variable, which makes a limited amount of sense. Once we improve the estimate with double lasso, the results are significantly more consistent. All are positive, implying that Uber has a complementary effect on public transit use. We see from the interaction term dummy*above_med_rides that the complementary effect of Uber is significantly stronger in agencies that provide a high number of rides (and slightly so for agencies in an above median population). One reason our numbers may be different from those in the paper is that they included both above median interaction terms in the same model, whereas we have them split into two (see bonus for inclusion of both).

	D=dummy	D=dummy*Above_med	D=pen	D=pen*Above_med
Lasso_pop	0.8759	-0	-3.2253	1.111
Lasso_rides	3.3848	5.4495	-0.9837	0.6804
DbLasso_pop	[0.131]	[0.1127]	[0.1392]	[0.1311]
DbLasso_rides	[0.9625]	[3.6022]	[0.1175]	[0.1004]

For the next four regressions we created polynomial interactions of order 5 on the control variables, to potentially improve the fit of our model and thus the accuracy of the coefficients on the variables of interest. With these added controls the coefficients on the variables of interest become larger than before, and again are more consistent once a double lasso has been run. When D is the dummy variable, previously the coefficient on the Uber dummy was significantly higher in the model that included the above_median_rides dummy than in the model that included the above_median_population dummy, whereas now the effects are quite similar. Not only is the effect of Uber still larger on agencies with above median number of rides, but now it also appears larger (to a greater extent than before) for agencies that serve a population above the median size. When D is the search intensity, we interestingly get the same coefficients when using both above_median_rides and above_median_population. This is likely testament to the power of a lasso, and more specifically double lasso, regression.

	D=dummy	D=dummy*Above_med	D=pen	D=pen*Above_med
Lasso_poly_pop	5.9108	0	1.1269	-0
Lasso_poly_rides	3.0877	16.1912	1.1269	0
DbLasso_poly_pop	[2.4535]	[3.7713]	[0.2154]	[0.9668]
DbLasso_poly_rides	[2.4473]	[5.2432]	[0.2154]	[1.0677]

In the Bonus section, we multiplied the dummy variable by the level of penetration variable. The resulting variable is now equal to 0 when the dummy is equal to zero (before Uber has entered the market) and equal to the level of penetration afterwards. This more robustly combines the effects of Uber's existence with the quantitative level of its search intensity.

When using OLS, we see a large negative coefficient. When we improve our model to lasso and then double lasso, the coefficient becomes positive and smaller on an absolute scale. A weaker effect is seen on agencies in above median population areas or above median rides provided without including the polynomial controls, but with the polynomials this coefficient is not selected by the lasso (effectively equal to 0). Once again, the lasso coefficients (with polynomials) are the same regardless of which interaction variable is used.

	D_new	D_new*Above_med
Bonus1_pop_OLS	-236.554(14.4703)	240.6444(13.4704)
Bonus2_rides_OLS	-9.3358(1.6415)	9.8248(0.9066)
Bonus3_pop_Lasso	0.7431	-0.237
Bonus4_rides_Lasso	0.5311	-0.0113
Bonus5_pop_LassoPoly	0.7237	0
Bonus6_rides_LassoPoly	0.7237	0

For the next bonus regressions, we return to the original two options for D (dummy and penetration). The added change is that we now include both above median interaction variables (population and rides) in the same model. We do this to create the Lasso version of the OLS that was run in the original paper.

	D	D*Above_med_pop	D*Above_med_rides
Bonus7_dummy_OLS	-679.0154(28.693)	948.6319(20.7447)	-26.7858(25.7983)
Bonus7_pen_OLS	-20.7003(1.9708)	17.8559(1.3842)	7.7784(1.1524)

A single stage lasso produces zero coefficients for all terms (barely above zero for D*above_med_rides). However, with the double lasso implementation, we see much more consistent coefficients, and all are positive, except for the interaction between the search intensity coefficient and the above_median_pop dummy in the single lasso with polynomial order 5 controls.

	D	**D*Above_med_pop**	**D*Above_med_rides**
Bonus8_dummy_Lasso	-0	-0	0
Bonus8_pen_Lasso	-0	-0	0.015
Bonus9_dummy_LassoPoly	2.983	0.4952	5.4612
Bonus9_pen_LassoPoly	1.1786	-0.4017	0.0241
Bonus10_dummy_DbLasso	[2.1153]	[1.8114]	[12.6196]
Bonus10_pen_DbLasso	[24.9286]	[0.3039]	[0.8207]

In conclusion, we believe we can say that uber is in fact a complement to public transit. The double lassos are the most powerful of the regression models we utilized, and in all cases our double lassos produce positive coefficients for both specifications of D, the effect of uber. It is also likely that this effect is stronger for agencies that provide above the median number of rides or exist in an area with above median population, as all double lasso coefficients (and many specifications of single lasso as well) produce positive coefficients on these terms.

```python
# In[30]:

#Maxwell Speil and Yiran Sun

#Load libraries
import pandas as pd
import numpy as np
from sklearn.linear_model import LassoCV
#import os


# In[1]:



#Read data
#os.chdir("C:\\Users\\mmspe\\OneDrive\\Documents\\Python Scripts\\434")
data = pd.read_csv("Uber_dataset.csv")

#Add logs
data["log_popestimate"] = np.log(data["popestimate"])
data["log_employment"] = np.log(data["employment"])
data["log_aveFareTotal"] = np.log(data["aveFareTotal"])
data["log_VRHTotal"] = np.log(data["VRHTotal"])
data["log_VOMSTotal"] = np.log(data["VOMSTotal"])
data["log_VRMTotal"] = np.log(data["VRMTotal"])
data["log_gasPrice"] = np.log(data["gasPrice"])

#Drop nas
data = data.dropna()

#Dependant variable
Y = np.log(data["UPTTotal"])
#First of two variations of variable of interest
uber_dummy = np.array(data["treatUberX"], ndmin = 2).T
#Second of two variations of variable of interest
uber_pen = np.array(data["treatGTNotStd"], ndmin = 2).T

#Vectorize controls for matrix multiplication
lnpop = np.array(data["log_popestimate"], ndmin = 2).T
lnemp = np.array(data["log_employment"], ndmin = 2).T
lnfare = np.array(data["log_aveFareTotal"], ndmin = 2).T
lnvhours = np.array(data["log_VRHTotal"], ndmin = 2).T
lnnumv = np.array(data["log_VOMSTotal"], ndmin = 2).T
lnmiles = np.array(data["log_VRMTotal"], ndmin = 2).T
lngas = np.array(data["log_gasPrice"], ndmin = 2).T

controls =  np.concatenate((lnpop, lnemp, lnfare, lnvhours, lnnumv,
                                  lnmiles, lngas), axis = 1)
#Number of samples
n = len(data) #58354 samples
#Constant
cons = np.ones([n ,1])
```

```python
#### Regressions ####
#   Regression 1

#       a) D = dummy

X1a = np.concatenate((cons, uber_dummy, controls), axis = 1)

betahat_1a = np.linalg.inv(X1a.T @ X1a) @ (X1a.T @ Y)
ehat_1a = Y - X1a @ betahat_1a
ehat_1a = np.array(ehat_1a, ndmin = 2).T

Sigmahat_1a = (X1a * ehat_1a).T @ (X1a * ehat_1a) / n

Qhat_1a = np.linalg.inv(X1a.T @ X1a / n)
Vhat_1a = Qhat_1a @ Sigmahat_1a @ Qhat_1a
sdhat_1a = np.sqrt(Vhat_1a[1 ,1]) / np.sqrt(n)
cil_1a = betahat_1a[1] - 1.96 * sdhat_1a; cir_1a = betahat_1a[1] + 1.96 * sdhat_1a

#       b) D = search intensity

X1b = np.concatenate((cons, uber_pen, controls), axis = 1)

betahat_1b = np.linalg.inv(X1b.T @ X1b) @ (X1b.T @ Y)
ehat_1b = Y - X1b @ betahat_1b
ehat_1b = np.array(ehat_1b, ndmin = 2).T

Sigmahat_1b = (X1b * ehat_1b).T @ (X1b * ehat_1b) / n

Qhat_1b = np.linalg.inv(X1b.T @ X1b / n)
Vhat_1b = Qhat_1b @ Sigmahat_1b @ Qhat_1b
sdhat_1b = np.sqrt(Vhat_1b[1 ,1]) / np.sqrt(n)
cil_1b = betahat_1b[1] - 1.96 * sdhat_1b; cir_1b = betahat_1b[1] + 1.96 * sdhat_1b



print('In OLS model 1:\n')
print("The coefficient for 'treatUberX' is ", betahat_1a[1])
print("Standard Error is ", sdhat_1a)
print("95% confidence interval is ["+str(cil_1a)+","+str(cir_1a)+"]")

print("\nThe coefficient for 'treatGTNotStd' is ", betahat_1b[1])
print("Standard Error is ", sdhat_1b)
print("95% confidence interval is ["+str(cil_1b)+","+str(cir_1b)+"]")
```

```
#   Regression 2
#γi is a transit agency specific fixed effect; δt is a year-month specific fixed effect

#       a) D = dummy

#Create dummies for transit agency fixed effects
agency_dummies = pd.get_dummies(data["agency"])
yrmon_dummies  = pd.get_dummies(data["dateSurvey"])


# In[23]:



X2a = np.concatenate((uber_dummy, agency_dummies, yrmon_dummies, controls), axis = 1)

betahat_2a = np.linalg.inv(X2a.T @ X2a) @ (X2a.T @ Y)
ehat_2a = Y - X2a @ betahat_2a
ehat_2a = np.array(ehat_2a, ndmin = 2).T

Sigmahat_2a = (X2a * ehat_2a).T @ (X2a * ehat_2a) / n

Qhat_2a = np.linalg.inv(X2a.T @ X2a / n)
Vhat_2a = Qhat_2a @ Sigmahat_2a @ Qhat_2a
sdhat_2a = np.sqrt(Vhat_2a[0 ,0]) / np.sqrt(n)
cil_2a = betahat_2a[0] - 1.96 * sdhat_2a; cir_2a = betahat_2a[0] + 1.96 * sdhat_2a

#       b) D = search intensity

X2b = np.concatenate((uber_pen, agency_dummies,
                      yrmon_dummies, controls), axis = 1)

betahat_2b = np.linalg.inv(X2b.T @ X2b) @ (X2b.T @ Y)
ehat_2b = Y - X2b @ betahat_2b
ehat_2b = np.array(ehat_2b, ndmin = 2).T

Sigmahat_2b = (X2b * ehat_2b).T @ (X2b * ehat_2b) / n

Qhat_2b = np.linalg.inv(X2b.T @ X2b / n)
Vhat_2b = Qhat_2b @ Sigmahat_2b @ Qhat_2b
sdhat_2b = np.sqrt(Vhat_2b[0 ,0]) / np.sqrt(n)
cil_2b = betahat_2b[0] - 1.96 * sdhat_2b; cir_2b = betahat_2b[0] + 1.96 * sdhat_2b



print('In OLS model 2(including time and location effect):\n')
print("The coefficient for 'treatUberX' is ", betahat_2a[0])
print("Standard Error is ", sdhat_2a)
print("95% confidence interval is ["+str(cil_2a)+","+str(cir_2a)+"]")

print("\nThe coefficient for 'treatGTNotStd' is ", betahat_2b[0])
print("Standard Error is ", sdhat_2b)
print("95% confidence interval is ["+str(cil_2b)+","+str(cir_2b)+"]")
```

```
#   Regression 3-----IMPORTANT

#   a) D = dummy

#Calculate median population: 1304926
median_pop = np.median(data["popestimate"])
#Create dummy
data["pop_med_dummy"] = (data["popestimate"] > median_pop).astype(int)
#Create interaction


# In[35]:


data["pop_med_int"] = data["pop_med_dummy"] * data["treatUberX"]
#pop_med_dum = np.array(data["pop_med_dummy"], ndmin = 2).T
pop_med_int = np.array(data["pop_med_int"], ndmin = 2).T


# In[ ]:


X3a = np.concatenate((uber_dummy, pop_med_int, agency_dummies,
                      yrmon_dummies, controls), axis = 1)

betahat_3a = np.linalg.inv(X3a.T @ X3a) @ (X3a.T @ Y)
ehat_3a = Y - X3a @ betahat_3a
ehat_3a = np.array(ehat_3a, ndmin = 2).T

Sigmahat_3a = (X3a * ehat_3a).T @ (X3a * ehat_3a) / n

Qhat_3a = np.linalg.inv(X3a.T @ X3a / n)
Vhat_3a = Qhat_3a @ Sigmahat_3a @ Qhat_3a
sdhat_3a = np.sqrt(Vhat_3a[0 ,0]) / np.sqrt(n)
cil_3a = betahat_3a[0] - 1.96 * sdhat_3a; cir_3a = betahat_3a[0] + 1.96 * sdhat_3a

sdhat_3a_pop = np.sqrt(Vhat_3a[1 ,1]) / np.sqrt(n)

cil_3a_pop = betahat_3a[1] - 1.96 * sdhat_3a_pop
cir_3a_pop = betahat_3a[1] + 1.96 * sdhat_3a_pop


# In[36]:


#   b) D = search intensity

#Create interaction
data["pop_med_int_pen"] = data["pop_med_dummy"] * data["treatGTNotStd"]
pop_med_int_pen = np.array(data["pop_med_int_pen"], ndmin = 2).T
```

```python
X3b = np.concatenate((uber_pen, pop_med_int_pen, agency_dummies,
                      yrmon_dummies, controls), axis = 1)

betahat_3b = np.linalg.inv(X3b.T @ X3b) @ (X3b.T @ Y)
ehat_3b = Y - X3b @ betahat_3b
ehat_3b = np.array(ehat_3b, ndmin = 2).T

Sigmahat_3b = (X3b * ehat_3b).T @ (X3b * ehat_3b) / n

Qhat_3b = np.linalg.inv(X3b.T @ X3b / n)
Vhat_3b = Qhat_3b @ Sigmahat_3b @ Qhat_3b
sdhat_3b = np.sqrt(Vhat_3b[0 ,0]) / np.sqrt(n)
cil_3b = betahat_3b[0] - 1.96 * sdhat_3b; cir_3b = betahat_3b[0] + 1.96 * sdhat_3b

sdhat_3b_pop = np.sqrt(Vhat_3b[1 ,1]) / np.sqrt(n)

cil_3b_pop = betahat_3b[1] - 1.96 * sdhat_3b_pop
cir_3b_pop = betahat_3b[1] + 1.96 * sdhat_3b_pop




pop_OLS = pd.DataFrame([[betahat_3a[0],betahat_3a[1],betahat_3b[0],betahat_3b[1]],
                        [sdhat_3a,sdhat_3a_pop, sdhat_3b,sdhat_3b_pop]],
                        columns=['Uber_dummy','Above_median_pop*Uber_dummy',
                                 'Uber_pen', 'Above_median_pop*Uber_pen'],
                       index=['coef','SE'])

pop_OLS
```

```python
#   Regression 4----IMPORTANT

#Calculate median rides:
median_rides = np.median(data["UPTTotal"])
#Create dummy
data["rides_med_dummy"] = (data["UPTTotal"] > median_rides).astype(int)
#Create interaction
data["rides_med_int"] = data["rides_med_dummy"] * data["treatUberX"]
#rides_med_dum = np.array(data["rides_med_dummy"], ndmin = 2).T
rides_med_int = np.array(data["rides_med_dummy"], ndmin = 2).T


# In[ ]:


X4a = np.concatenate((uber_dummy, rides_med_int, agency_dummies,
                      yrmon_dummies, controls), axis = 1)

betahat_4a = np.linalg.inv(X4a.T @ X4a) @ (X4a.T @ Y)
ehat_4a = Y - X4a @ betahat_4a
ehat_4a = np.array(ehat_4a, ndmin = 2).T

Sigmahat_4a = (X4a * ehat_4a).T @ (X4a * ehat_4a) / n

Qhat_4a = np.linalg.inv(X4a.T @ X4a / n)
Vhat_4a = Qhat_4a @ Sigmahat_4a @ Qhat_4a
sdhat_4a = np.sqrt(Vhat_4a[0 ,0]) / np.sqrt(n)
cil_4a = betahat_4a[0] - 1.96 * sdhat_4a; cir_4a = betahat_4a[0] + 1.96 * sdhat_4a

sdhat_4a_rides = np.sqrt(Vhat_4a[1 ,1]) / np.sqrt(n)

cil_4a_rides = betahat_4a[1] - 1.96 * sdhat_4a_rides
cir_4a_rides = betahat_4a[1] + 1.96 * sdhat_4a_rides


# In[38]:


#   b) D = search intensity

data["rides_med_int_pen"] = data["rides_med_dummy"] * data["treatGTNotStd"]
rides_med_int_pen = np.array(data["rides_med_int_pen"], ndmin = 2).T


# In[27]:


X4b = np.concatenate((uber_pen, rides_med_int_pen, agency_dummies,
                      yrmon_dummies, controls), axis = 1)

betahat_4b = np.linalg.inv(X4b.T @ X4b) @ (X4b.T @ Y)
ehat_4b = Y - X3b @ betahat_4b
ehat_4b = np.array(ehat_4b, ndmin = 2).T

Sigmahat_4b = (X4b * ehat_4b).T @ (X4b * ehat_4b) / n

Qhat_4b = np.linalg.inv(X4b.T @ X4b / n)
Vhat_4b = Qhat_4b @ Sigmahat_4b @ Qhat_4b
sdhat_4b = np.sqrt(Vhat_4b[0 ,0]) / np.sqrt(n)
cil_4b = betahat_4b[0] - 1.96 * sdhat_4b; cir_4b = betahat_4b[0] + 1.96 * sdhat_4b
```

```python
rides_OLS = pd.DataFrame([[betahat_4a[0],betahat_4a[1],betahat_4b[0],betahat_4b[1]],
                          [sdhat_4a,sdhat_4a_rides, sdhat_4b,sdhat_4b_rides]],
                          columns=['Uber_dummy','Above_median_rides*Uber_dummy',
                                   'Uber_pen', 'Above_median_rides*Uber_pen'],
                         index=['coef','SE'])

rides_OLS
```

```python
#   Regression 5
from sklearn.linear_model import LassoCV

#   a) D = dummy

#Rescale controls
muhat_scale = np.mean(controls,axis = 0)
stdhat_scale = np.std(controls,axis = 0)
controls_scaled = (controls - muhat_scale )/ stdhat_scale


# In[ ]:


X5a = np.concatenate((uber_dummy, pop_med_int, agency_dummies,
                      yrmon_dummies, controls_scaled), axis = 1)


#run lasso
lasso5a = LassoCV(cv = 5, fit_intercept=False,  random_state=0)
lasso5a.fit(X5a ,Y)
coef5a = lasso5a.coef_
sel5a = (coef5a != 0)


# In[50]:


#   b) D = search intensity

#Rescale
muhat_scale_pen = np.mean(uber_pen)
stdhat_scale_pen = np.std(uber_pen)
uber_pen_scaled = (uber_pen - muhat_scale_pen) / stdhat_scale_pen


# In[7]:


X5b = np.concatenate((uber_pen_scaled, pop_med_int_pen, agency_dummies,
                      yrmon_dummies, controls_scaled), axis = 1)

lasso5b = LassoCV(cv = 5, fit_intercept=False,  random_state=0)
lasso5b.fit(X5b, Y)
coef5b = lasso5b.coef_
sel5b = (coef5b != 0)




pop_lasso = pd.DataFrame([[coef5a[0],coef5a[1],coef5b[0],coef5b[1]]],
                         columns=['lasso_Uber_dummy','lasso_Above_med_pop*Uber_dummy',
                                  'lasso_Uber_pen', 'lasso_Above_med_pop*Uber_pen'],
                          index=['coef'])

pop_lasso
```

```python
#   Regression 6

#   a) D = dummy

X6a = np.concatenate((uber_dummy, rides_med_int, agency_dummies,
                      yrmon_dummies, controls_scaled), axis = 1)

#run lasso
lasso6a = LassoCV(cv = 5, fit_intercept=False,  random_state=0)
lasso6a.fit(X6a, Y)
coef6a = lasso6a.coef_
sel6a = (coef6a != 0)


#   b) D = search intensity

X6b = np.concatenate((uber_pen_scaled, rides_med_int_pen, agency_dummies,
                      yrmon_dummies, controls_scaled), axis = 1)

lasso6b = LassoCV(cv = 5, fit_intercept=False,  random_state=0)
lasso6b.fit(X6b, Y)
coef6b = lasso6b.coef_




rides_lasso = pd.DataFrame([[coef6a[0],coef6a[1],coef6b[0],coef6b[1]]],
                           columns=['lasso_Uber_dummy','lasso_Above_med_pop*Uber_dummy',
                                    'lasso_Uber_pen', 'lasso_Above_med_pop*Uber_pen'],
                          index=['coef'])

rides_lasso
```

```python
#   Regression 7---DOUBLE LASSO

#  Regresion 7.1 --- For population(corresponding to regression 5)
#from sklearn.linear_model import MultiTaskLassoCV

#       a) D= dummy
#1st stage--same as regression 5
coef7ia_1 = lasso5a.coef_.copy()

#2nd stage
X7ia = np.concatenate((agency_dummies, yrmon_dummies, controls_scaled), axis = 1)

#fit on uber_dummy
lasso7ia_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=10000).fit(X7ia, uber_dummy)
coef7ia_21 = lasso7ia_21.coef_
ehat7ia_21 = uber_dummy.T- coef7ia_21.T @ X7ia.T

#fit on pop_med_int
lasso7ia_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=10000).fit(X7ia, pop_med_int)
coef7ia_22 = lasso7ia_22.coef_
ehat7ia_22 = pop_med_int.T- coef7ia_22.T @ X7ia.T

#Calculate alpha
alpha7ia_B1 = (np.array(Y - X7ia @ coef7ia_1[2:])
              @ ehat7ia_21.T) @ np.linalg.inv(uber_dummy.T @ (ehat7ia_21).T)

alpha7ia_B2 = (np.array(Y - X7ia @ coef7ia_1[2:])
              @ ehat7ia_22.T) @ np.linalg.inv(pop_med_int.T @ (ehat7ia_22).T)


#       b) D = search intensity

#1st stage--same as regression 5
coef7ib_1 = lasso5b.coef_.copy()

#2nd stage
X7ib = np.concatenate((agency_dummies,yrmon_dummies, controls_scaled), axis = 1)

#fit on uber_pen
lasso7ib_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=10000).fit(X7ib, uber_pen)
coef7ib_21 = lasso7ib_21.coef_
ehat7ib_21 = uber_pen.T- coef7ib_21.T @ X7ib.T

#fit on pop_med_int_pen
lasso7ib_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=10000).fit(X7ib, pop_med_int_pen)
coef7ib_22 = lasso7ib_22.coef_
ehat7ib_22 = pop_med_int_pen.T- coef7ib_22.T @ X7ib.T

#Calculate alpha
alpha7ib_B1 = (np.array(Y - X7ib @ coef7ib_1[2:])
              @ ehat7ib_21.T) @ np.linalg.inv(uber_pen.T @ (ehat7ib_21).T)

alpha7ib_B2 = (np.array(Y - X7ib @ coef7ib_1[2:])
              @ ehat7ib_22.T) @ np.linalg.inv(pop_med_int_pen.T @ (ehat7ib_22).T)
```

```python
pop_dblasso = pd.DataFrame([[alpha7ia_B1,alpha7ia_B2,alpha7ib_B1,alpha7ib_B2]],
                           columns=['dblasso_Uber_dummy','dblasso_Above_med*Uber_dummy',
                                    'dblasso_Uber_pen', 'dblasso_Above_med*Uber_pen'],
                            index=['coef'])

pop_dblasso
```

```python
#  Regresion 7.2 --- For rides(corresponding to regression 6)

#       a) D= dummy
#1st stage--same as regression 6
coef7iia_1 = lasso6a.coef_.copy()

#2nd stage
X7iia = np.concatenate((agency_dummies,yrmon_dummies, controls_scaled), axis = 1)

#fit on uber_dummy
lasso7iia_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                       max_iter=10000).fit(X7iia,uber_dummy)
coef7iia_21 = lasso7iia_21.coef_
ehat7iia_21 = uber_dummy.T- coef7iia_21.T @ X7iia.T

#fit on rides_med_int
lasso7iia_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                       max_iter=10000).fit(X7iia,rides_med_int)
coef7iia_22 = lasso7iia_22.coef_
ehat7iia_22 = rides_med_int.T - coef7iia_22.T @ X7iia.T

#Calculate alpha
alpha7iia_B1 = (np.array(Y - X7iia @ coef7iia_1[2:])
              @ ehat7iia_21.T) @ np.linalg.inv(uber_dummy.T @ (ehat7iia_21).T)

alpha7iia_B2 = (np.array(Y - X7iia @ coef7iia_1[2:])
              @ ehat7iia_22.T) @ np.linalg.inv(rides_med_int.T @ (ehat7iia_22).T)


#       b) D = search intensity

#1st stage--same as regression 6
coef7iib_1 = lasso6b.coef_.copy()

#2nd stage
X7iib = np.concatenate((agency_dummies,yrmon_dummies, controls_scaled), axis = 1)

#fit on uber_pen
lasso7iib_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                       max_iter=10000).fit(X7iib, uber_pen)
coef7iib_21 = lasso7iib_21.coef_
ehat7iib_21 = uber_pen.T - coef7iib_21.T @ X7iib.T

#fit on rides_med_int_pen
lasso7iib_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                       max_iter=10000).fit(X7iib, rides_med_int_pen)
coef7iib_22 = lasso7iib_22.coef_
ehat7iib_22 = rides_med_int_pen.T - coef7iib_22.T @ X7iib.T

#Calculate alpha
alpha7iib_B1 = (np.array(Y - X7iib @ coef7iib_1[2:])
              @ ehat7iib_21.T) @ np.linalg.inv(uber_pen.T @ (ehat7iib_21).T)

alpha7iib_B2 = (np.array(Y - X7iib @ coef7iib_1[2:])
              @ ehat7iib_22.T) @ np.linalg.inv(rides_med_int_pen.T @ (ehat7iib_22).T)
```

```
rides_dblasso = pd.DataFrame([[alpha7iia_B1,alpha7iia_B2,alpha7iib_B1,alpha7iib_B2]],
                          columns=['dblasso_Uber_dummy','dblasso_Above_med*Uber_dummy',
                                   'dblasso_Uber_pen', 'dblasso_Above_med*Uber_pen'],
                          index=['coef'])

rides_dblasso
```

```
#   Regression 8
from sklearn.preprocessing import PolynomialFeatures

#Create interactions
pol_int = PolynomialFeatures(degree=5, include_bias=False)
int_controls = pol_int.fit_transform(controls)

muhat_scale_int = np.mean(int_controls,axis = 0)
stdhat_scale_int = np.std(int_controls,axis = 0)
int_controls_scaled = (int_controls - muhat_scale_int )/ stdhat_scale_int


# In[11]:


#   a) D = dummy

X8a = np.concatenate((uber_dummy, pop_med_int, agency_dummies,
                      yrmon_dummies, int_controls_scaled), axis = 1)

lasso8a = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lasso8a.fit(X8a ,Y)
coef8a = lasso8a.coef_
sel8a = (coef8a != 0)

#   b) D = search intensity

X8b = np.concatenate((uber_pen, pop_med_int_pen, agency_dummies,
                      yrmon_dummies, int_controls_scaled), axis = 1)

lasso8b = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lasso8b.fit(X8b, Y)
coef8b = lasso8b.coef_
sel8b = (coef8b != 0)



pop_poly_lasso = pd.DataFrame([[coef8a[0],coef8a[1],coef8b[0],coef8b[1]]],
                          columns=['lasso_Uber_dummy','lasso_Above_med_pop*Uber_dummy',
                                   'lasso_Uber_pen', 'lasso_Above_med_pop*Uber_pen'],
                          index=['coef'])

pop_poly_lasso
```

```
#   Regression 9

#   a) D = dummy

X9a = np.concatenate((uber_dummy, rides_med_int, agency_dummies,
                      yrmon_dummies, int_controls_scaled), axis = 1)


lasso9a = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lasso9a.fit(X9a ,Y)
coef9a = lasso9a.coef_
sel9a = (coef9a != 0)

#   b) D = search intensity

X9b = np.concatenate((uber_pen, rides_med_int_pen, agency_dummies,
                      yrmon_dummies, int_controls_scaled), axis = 1)

lasso9b = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lasso9b.fit(X9b ,Y)
coef9b = lasso9b.coef_
sel9b = (coef9b != 0)



rides_poly_lasso = pd.DataFrame([[coef9a[0],coef9a[1],coef9b[0],coef9b[1]]],
                           columns=['lasso_Uber_dummy','lasso_Above_med_pop*Uber_dummy',
                                    'lasso_Uber_pen', 'lasso_Above_med_pop*Uber_pen'],
                          index=['coef'])

rides_poly_lasso
```

```python
#   Regression 10 ---DOUBLE LASSO

#  Regresion 10.1 --- For population(corresponding to regression 8)

#        a) D= dummy
#1st stage--same as regression 8
coef10ia_1 = lasso8a.coef_.copy()

#2nd stage
X10ia = np.concatenate((agency_dummies, yrmon_dummies, int_controls_scaled), axis = 1)

#fit on uber_dummy
lasso10ia_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=100000).fit(X10ia, uber_dummy)
coef10ia_21 = lasso10ia_21.coef_
ehat10ia_21 = uber_dummy.T- coef10ia_21.T @ X10ia.T

#fit on pop_med_int
lasso10ia_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=100000).fit(X10ia, pop_med_int)
coef10ia_22 = lasso10ia_22.coef_
ehat10ia_22 = pop_med_int.T- coef10ia_22.T @ X10ia.T

#Calculate alpha
alpha10ia_B1 = (np.array(Y - X10ia @ coef10ia_1[2:])
              @ ehat10ia_21.T) @ np.linalg.inv(uber_dummy.T @ (ehat10ia_21).T)

alpha10ia_B2 = (np.array(Y - X10ia @ coef10ia_1[2:])
              @ ehat10ia_22.T) @ np.linalg.inv(pop_med_int.T @ (ehat10ia_22).T)


#        b) D = search intensity

#1st stage--same as regression 8
coef10ib_1 = lasso8b.coef_.copy()

#2nd stage
X10ib = np.concatenate((agency_dummies,yrmon_dummies, int_controls_scaled), axis = 1)

#fit on uber_pen
lasso10ib_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=100000).fit(X10ib, uber_pen)
coef10ib_21 = lasso10ib_21.coef_
ehat10ib_21 = uber_pen.T- coef10ib_21.T @ X10ib.T

#fit on pop_med_int_pen
lasso10ib_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=100000).fit(X10ib, pop_med_int_pen)
coef10ib_22 = lasso10ib_22.coef_
ehat10ib_22 = pop_med_int_pen.T- coef10ib_22.T @ X10ib.T

#Calculate alpha
alpha10ib_B1 = (np.array(Y - X10ib @ coef10ib_1[2:])
              @ ehat10ib_21.T) @ np.linalg.inv(uber_pen.T @ (ehat10ib_21).T)

alpha10ib_B2 = (np.array(Y - X10ib @ coef10ib_1[2:])
              @ ehat10ib_22.T) @ np.linalg.inv(pop_med_int_pen.T @ (ehat10ib_22).T)
```

```python
pop_poly_dblasso = pd.DataFrame([[alpha10ia_B1, alpha10ia_B2, alpha10ib_B1, alpha10ib_B2]],
                         columns=['dblasso_Uber_dummy','dblasso_Above_med*Uber_dummy',
                                  'dblasso_Uber_pen', 'dblasso_Above_med*Uber_pen'],
                          index=['coef'])

pop_poly_dblasso
```

```python
#  Regresion 10.2 --- For rides (corresponding to regression 9)

#       a) D= dummy
#1st stage--same as regression 9
coef10iia_1 = lasso9a.coef_.copy()

#2nd stage
X10iia = np.concatenate((agency_dummies,yrmon_dummies,int_controls_scaled), axis = 1)

#fit on uber_dummy
lasso10iia_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=500000).fit(X10iia, uber_dummy)
coef10iia_21 = lasso10iia_21.coef_
ehat10iia_21 = uber_dummy.T- coef10iia_21.T @ X10iia.T

#fit on rides_med_int
lasso10iia_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=500000).fit(X10iia, rides_med_int)
coef10iia_22 = lasso10iia_22.coef_
ehat10iia_22 = pop_med_int.T- coef10iia_22.T @ X10iia.T

#Calculate alpha
alpha10iia_B1 = (np.array(Y - X10iia @ coef10iia_1[2:])
              @ ehat10iia_21.T) @ np.linalg.inv(uber_dummy.T @ (ehat10iia_21).T)

alpha10iia_B2 = (np.array(Y - X10iia @ coef10iia_1[2:])
              @ ehat10iia_22.T) @ np.linalg.inv(rides_med_int.T @ (ehat10iia_22).T)


#       b) D = search intensity

#1st stage--same as regression 8
coef10iib_1 = lasso9b.coef_.copy()

#2nd stage
X10iib = np.concatenate((agency_dummies,yrmon_dummies, int_controls_scaled), axis = 1)

#fit on uber_pen
lasso10iib_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=500000).fit(X10iib, uber_pen)
coef10iib_21 = lasso10iib_21.coef_
ehat10iib_21 = uber_pen.T- coef10iib_21.T @ X10iib.T

#fit on rides_med_int_pen
lasso10iib_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=500000).fit(X10iib, rides_med_int_pen)
coef10iib_22 = lasso10iib_22.coef_
ehat10iib_22 = rides_med_int_pen.T- coef10iib_22.T @ X10iib.T

#Calculate alpha
alpha10iib_B1 = (np.array(Y - X10iib @ coef10iib_1[2:])
              @ ehat10iib_21.T) @ np.linalg.inv(uber_pen.T @ (ehat10iib_21).T)

alpha10iib_B2 = (np.array(Y - X10iib @ coef10iib_1[2:])
              @ ehat10iib_22.T) @ np.linalg.inv(rides_med_int_pen.T @ (ehat10iib_22).T)
```

```python
rides_poly_dblasso = pd.DataFrame([[alpha10iia_B1, alpha10iia_B2, alpha10iib_B1, alpha10iib_B2]],
                          columns=['dblasso_Uber_dummy','dblasso_Above_med*Uber_dummy',
                                   'dblasso_Uber_pen', 'dblasso_Above_med*Uber_pen'],
                           index=['coef'])

rides_poly_dblasso
```

```
#Output results:

# Regression 1&2 (OLS)
ols_model = pd.DataFrame([[betahat_1a[1],sdhat_1a,
                           "("+str(round(cil_1a,4))+","+str(round(cir_1a,4))+")",
                           betahat_1b[1],sdhat_1b,
                           "("+str(round(cil_1b,4))+","+str(round(cir_1b,4))+")"],
                          [betahat_2a[0],sdhat_2a,
                           "("+str(round(cil_2a,4))+","+str(round(cir_2a,4))+")",
                           betahat_2b[0],sdhat_2b,
                           "("+str(round(cil_2b,4))+","+str(round(cir_2b,4))+")"]],
                          columns=['D=dummy_Coef','D=dummy_SE','D=dummy_95%CI',
                                   'D=pen_Coef','D=pen_SE','D=pen_95%CI'],
                         index=['OLS 1','OLS 2'])

ols_model


# In[28]:


#Regression 3-10
# 3-4: OLS, adding pop & rides
# 5-6: Lasso, same as 3-4
# 7: Double Lasso on 5&6
# 8-9: Lasso, add poly
# 10: Double Lasso on 8&9

model_group = pd.DataFrame([
    [str(round(betahat_3a[0],4))+"("+str(round(sdhat_3a,4))+")",
     str(round(betahat_3a[1],4))+"("+str(round(sdhat_3a_pop,4))+")",
     str(round(betahat_3b[0],4))+"("+str(round(sdhat_3b,4))+")",
     str(round(betahat_3b[1],4))+"("+str(round(sdhat_3b_pop,4))+")"],
    [str(round(betahat_4a[0],4))+"("+str(round(sdhat_4a,4))+")",
     str(round(betahat_4a[1],4))+"("+str(round(sdhat_4a_rides,4))+")",
     str(round(betahat_4b[0],4))+"("+str(round(sdhat_4b,4))+")",
     str(round(betahat_4b[1],4))+"("+str(round(sdhat_4b_rides,4))+")"],
    [round(coef5a[0],4),round(coef5a[1],4),round(coef5b[0],4),round(coef5b[1],4)],
    [round(coef6a[0],4),round(coef6a[1],4),round(coef6b[0],4),round(coef6b[1],4)],
    [np.round(alpha7ia_B1,4),np.round(alpha7ia_B2,4),np.round(alpha7ib_B1,4),
     np.round(alpha7ib_B2,4)],
    [np.round(alpha7iia_B1,4),np.round(alpha7iia_B2,4),np.round(alpha7iib_B1,4),
     np.round(alpha7iib_B2,4)],
    [round(coef8a[0],4),round(coef8a[1],4),round(coef8b[0],4),round(coef8b[1],4)],
    [round(coef9a[0],4),round(coef9a[1],4),round(coef9b[0],4),round(coef9b[1],4)],
    [np.round(alpha10ia_B1,4),np.round(alpha10ia_B2,4),np.round(alpha10ib_B1,4),
     np.round(alpha10ib_B2,4)],
    [np.round(alpha10iia_B1,4),np.round(alpha10iia_B2,4),np.round(alpha10iib_B1,4),
     np.round(alpha10iib_B2,4)]],
    columns=['D=dummy','D=dummy*Above_med','D=pen','D=pen*Above_med'],
    index=['OLS3_pop','OLS4_rides','Lasso_pop','Lasso_rides','DbLasso_pop',
           'DbLasso_rides','Lasso_poly_pop','Lasso_poly_rides',
           'DbLasso_poly_pop','DbLasso_poly_rides'])

model_group
```

```python
####################### BONUS REGRESSION 1-6 #####################

#New D = dummy * search intensity

#This will be equal to zero before Uber enters (when dummy=0), and equal to
#the search intensity volume after. This combines the effects of Ubers presence
#from the dummy variable with the size/populariy of uber from the search intensity

#Bonus 1 - Reg 3 but with new D

data["D_new"] = data["treatUberX"] * data["treatGTNotStd"]
D_new = np.array(data["D_new"], ndmin = 2).T

data["pop_int_new"] = data["pop_med_dummy"] * data["D_new"]
pop_int_new = np.array(data["pop_int_new"], ndmin = 2).T

#And we will fit into different regressions to see if it improve the result.


Xbonus1_1 = np.concatenate((D_new, pop_int_new, agency_dummies,
                      yrmon_dummies, controls), axis = 1)

betahat_b1 = np.linalg.inv(Xbonus1_1.T @ Xbonus1_1) @ (Xbonus1_1.T @ Y)
ehat_b1 = Y - Xbonus1_1 @ betahat_b1
ehat_b1 = np.array(ehat_b1, ndmin = 2).T

Sigmahat_b1 = (Xbonus1_1 * ehat_b1).T @ (Xbonus1_1 * ehat_b1) / n

Qhat_b1 = np.linalg.inv(Xbonus1_1.T @ Xbonus1_1 / n)
Vhat_b1 = Qhat_b1 @ Sigmahat_b1 @ Qhat_b1
sdhat_b1 = np.sqrt(Vhat_b1[0 ,0]) / np.sqrt(n)

sdhat_b1_pop = np.sqrt(Vhat_b1[1 ,1]) / np.sqrt(n)


#Bonus 2 - Reg 4 but with new D

data["rides_int_new"] = data["rides_med_dummy"] * data["D_new"]
rides_int_new = np.array(data["rides_int_new"], ndmin = 2).T

Xbonus_2 = np.concatenate((D_new, rides_int_new, agency_dummies,
                      yrmon_dummies, controls), axis = 1)

betahat_b2 = np.linalg.inv(Xbonus_2.T @ Xbonus_2) @ (Xbonus_2.T @ Y)
ehat_b2 = Y - Xbonus_2 @ betahat_b2
ehat_b2 = np.array(ehat_b2, ndmin = 2).T

Sigmahat_b2 = (Xbonus_2 * ehat_b2).T @ (Xbonus_2 * ehat_b2) / n

Qhat_b2 = np.linalg.inv(Xbonus_2.T @ Xbonus_2 / n)
Vhat_b2 = Qhat_b2 @ Sigmahat_b2 @ Qhat_b2
sdhat_b2 = np.sqrt(Vhat_b2[0 ,0]) / np.sqrt(n)

sdhat_b2_rides = np.sqrt(Vhat_b2[1 ,1]) / np.sqrt(n)
```

```python
#Bonus 3 - Reg 5 with new D

Xbonus_3 = np.concatenate((D_new, pop_int_new, agency_dummies,
                          yrmon_dummies, controls_scaled), axis = 1)


#run lasso
lassob3 = LassoCV(cv = 5, fit_intercept=False,  random_state=0)
lassob3.fit(Xbonus_3 ,Y)
coefb3 = lassob3.coef_


#Bonus 4 - Reg 6 with new D

Xbonus_4 = np.concatenate((D_new, rides_int_new, agency_dummies,
                          yrmon_dummies, controls_scaled), axis = 1)


#run lasso
lassob4 = LassoCV(cv = 5, fit_intercept=False,  random_state=0)
lassob4.fit(Xbonus_4 ,Y)
coefb4 = lassob4.coef_

# In[59]:


#Bonus 5 - Reg 8 with new D

Xbonus_5 = np.concatenate((D_new, pop_int_new, agency_dummies,
                          yrmon_dummies, int_controls_scaled), axis = 1)

lassob5 = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lassob5.fit(Xbonus_5 ,Y)
coefb5 = lassob5.coef_


#Bonus 6 - Reg 9 with new D

Xbonus_6 = np.concatenate((D_new, rides_int_new, agency_dummies,
                          yrmon_dummies, int_controls_scaled), axis = 1)

lassob6 = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lassob6.fit(Xbonus_6 ,Y)
coefb6 = lassob6.coef_
```

```python
#Output

#Bonus Regression1-6:

bonus_group1 = pd.DataFrame([
    [str(round(betahat_b1[0],4))+"("+str(round(sdhat_b1,4))+")",
     str(round(betahat_b1[1],4))+"("+str(round(sdhat_b1_pop,4))+")"],
    [str(round(betahat_b2[0],4))+"("+str(round(sdhat_b2,4))+")",
     str(round(betahat_b2[1],4))+"("+str(round(sdhat_b2_rides,4))+")"],
    [round(coefb3[0],4),round(coefb3[1],4)],[round(coefb4[0],4),round(coefb4[1],4)],
    [round(coefb5[0],4),round(coefb5[1],4)],[round(coefb6[0],4),round(coefb6[1],4)]],
    columns=['D_new','D_new*Above_med'],
    index=['Bonus1_pop_OLS','Bonus2_rides_OLS','Bonus3_pop_Lasso','Bonus4_rides_Lasso',
           'Bonus5_pop_LassoPoly','Bonus6_rides_LassoPoly'])

bonus_group1
```

```
###################### BONUS REGRESSION 7- ####################

#put the interaction of pop*dummy and rides*dummy in one model as the paper does and
#fit OLS, Lasso rgeression

#Bonus 7 - OLS but with both inteactions.

# a) with D=dummy
Xbonus7_1 = np.concatenate((uber_dummy, pop_med_int, rides_med_int, agency_dummies,
                      yrmon_dummies, controls), axis = 1)


# In[ ]:


betahat_b7_1 = np.linalg.inv(Xbonus7_1.T @ Xbonus7_1) @ (Xbonus7_1.T @ Y)
ehat_b7_1 = Y - Xbonus7_1 @ betahat_b7_1
ehat_b7_1 = np.array(ehat_b7_1, ndmin = 2).T

Sigmahat_b7_1 = (Xbonus7_1 * ehat_b7_1).T @ (Xbonus7_1 * ehat_b7_1) / n

Qhat_b7_1 = np.linalg.inv(Xbonus7_1.T @ Xbonus7_1 / n)
Vhat_b7_1 = Qhat_b7_1 @ Sigmahat_b7_1 @ Qhat_b7_1
sdhat_b7_1 = np.sqrt(Vhat_b7_1[0 ,0]) / np.sqrt(n)

sdhat_b7_1pop = np.sqrt(Vhat_b7_1[1 ,1]) / np.sqrt(n)
sdhat_b7_1rides = np.sqrt(Vhat_b7_1[2 ,2]) / np.sqrt(n)


# In[43]:


# b) with D=search intensity
Xbonus7_2 = np.concatenate((uber_pen, pop_med_int_pen, rides_med_int_pen,
                            agency_dummies, yrmon_dummies, controls), axis = 1)


# In[47]:


betahat_b7_2 = np.linalg.inv(Xbonus7_2.T @ Xbonus7_2) @ (Xbonus7_2.T @ Y)
ehat_b7_2 = Y - Xbonus7_2 @ betahat_b7_2
ehat_b7_2 = np.array(ehat_b7_2, ndmin = 2).T

Sigmahat_b7_2 = (Xbonus7_2 * ehat_b7_2).T @ (Xbonus7_2 * ehat_b7_2) / n

Qhat_b7_2 = np.linalg.inv(Xbonus7_2.T @ Xbonus7_2 / n)
Vhat_b7_2 = Qhat_b7_2 @ Sigmahat_b7_2 @ Qhat_b7_2
sdhat_b7_2 = np.sqrt(Vhat_b7_2[0 ,0]) / np.sqrt(n)

sdhat_b7_2pop = np.sqrt(Vhat_b7_2[1 ,1]) / np.sqrt(n)
sdhat_b7_2rides = np.sqrt(Vhat_b7_2[2 ,2]) / np.sqrt(n)
```

```
#Bonus 8 - Lasso but with both inteactions.

# a) with D=dummy
Xbonus8_1 = Xbonus7_1.copy()


# In[46]:


lassob8_1 = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lassob8_1.fit(Xbonus8_1 ,Y)
coefb8_1 = lassob8_1.coef_


# In[47]:


# b) with D=search intensity
Xbonus8_2 = Xbonus7_2.copy()

lassob8_2 = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lassob8_2.fit(Xbonus8_2 ,Y)
coefb8_2 = lassob8_2.coef_


# In[49]:


#Bonus 9 - Lasso of Poly with both inteactions.

# a) with D=dummy
Xbonus9_1 =  np.concatenate((uber_dummy, pop_med_int, rides_med_int, agency_dummies,
                     yrmon_dummies, controls_scaled), axis = 1)


lassob9_1 = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lassob9_1.fit(Xbonus9_1 ,Y)
coefb9_1 = lassob9_1.coef_

# b) with D=search intensity
Xbonus9_2 =  np.concatenate((uber_pen, pop_med_int_pen, rides_med_int_pen,
                            agency_dummies, yrmon_dummies, controls_scaled), axis = 1)

lassob9_2 = LassoCV(cv = 5, fit_intercept=False, max_iter=100000, random_state=0)
lassob9_2.fit(Xbonus9_2 ,Y)
coefb9_2 = lassob9_2.coef_
```

```python
#Bonus 10 - DoubleLasso with both inteactions, corresponding to 8

# a) with D=dummy
#1st stage--same as Bonus 8_1
coefb10_1 = coefb8_1.copy()

#2nd stage
Xbonus10_1 = np.concatenate((agency_dummies, yrmon_dummies, controls_scaled), axis = 1)

#fit on uber_dummy
lassob10_11 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                       max_iter=10000).fit(Xbonus10_1, uber_dummy)
coefb10_11 = lassob10_11.coef_
ehatb10_11 = uber_dummy.T- coefb10_11.T @ Xbonus10_1.T

#fit on pop_med_int
lassob10_12 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                       max_iter=10000).fit(Xbonus10_1, pop_med_int)
coefb10_12 = lassob10_12.coef_
ehatb10_12 = pop_med_int.T- coefb10_12.T @ Xbonus10_1.T

#fit on rides_med_int
lassob10_13 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                       max_iter=10000).fit(Xbonus10_1, rides_med_int)
coefb10_13 = lassob10_13.coef_
ehatb10_13 = pop_med_int.T- coefb10_13.T @ Xbonus10_1.T


#Calculate alpha
alphab10_11 = (np.array(Y - Xbonus10_1 @ coefb10_1[3:])
              @ ehatb10_11.T) @ np.linalg.inv(uber_dummy.T @ (ehatb10_11).T)

alphab10_12 = (np.array(Y - Xbonus10_1 @ coefb10_1[3:])
              @ ehatb10_12.T) @ np.linalg.inv(pop_med_int.T @ (ehatb10_12).T)

alphab10_13 = (np.array(Y - Xbonus10_1 @ coefb10_1[3:])
              @ ehatb10_13.T) @ np.linalg.inv(rides_med_int.T @ (ehatb10_13).T)
```

```python
# b) with D=search intensity
#1st stage--same as Bonus 8_2
coefb10_2 = coefb8_2.copy()

#2nd stage
Xbonus10_2 = np.concatenate((agency_dummies, yrmon_dummies, controls_scaled), axis = 1)

#fit on uber_dummy
lassob10_21 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=10000).fit(Xbonus10_2, uber_pen)
coefb10_21 = lassob10_21.coef_
ehatb10_21 = uber_dummy.T- coefb10_21.T @ Xbonus10_2.T

#fit on pop_med_int
lassob10_22 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=10000).fit(Xbonus10_2, pop_med_int_pen)
coefb10_22 = lassob10_22.coef_
ehatb10_22 = pop_med_int_pen.T- coefb10_22.T @ Xbonus10_2.T

#fit on rides_med_int
lassob10_23 = LassoCV(cv = 5, fit_intercept=False,random_state=0,
                      max_iter=10000).fit(Xbonus10_2, rides_med_int_pen)
coefb10_23 = lassob10_23.coef_
ehatb10_23 = pop_med_int_pen.T- coefb10_23.T @ Xbonus10_2.T


#Calculate alpha
alphab10_21 = (np.array(Y - Xbonus10_2 @ coefb10_2[3:])
              @ ehatb10_21.T) @ np.linalg.inv(uber_dummy.T @ (ehatb10_21).T)

alphab10_22 = (np.array(Y - Xbonus10_2 @ coefb10_2[3:])
              @ ehatb10_22.T) @ np.linalg.inv(pop_med_int_pen.T @ (ehatb10_22).T)

alphab10_23 = (np.array(Y - Xbonus10_2 @ coefb10_2[3:])
              @ ehatb10_23.T) @ np.linalg.inv(rides_med_int_pen.T @ (ehatb10_23).T)
```

```python
#Output

#Bonus Regression:

bonus_group2 = pd.DataFrame([
    [str(round(betahat_b7_1[0],4))+"("+str(round(sdhat_b7_1,4))+")",
     str(round(betahat_b7_1[1],4))+"("+str(round(sdhat_b7_1pop,4))+")",
     str(round(betahat_b7_1[2],4))+"("+str(round(sdhat_b7_1rides,4))+")"],
    [str(round(betahat_b7_2[0],4))+"("+str(round(sdhat_b7_2,4))+")",
     str(round(betahat_b7_2[1],4))+"("+str(round(sdhat_b7_2pop,4))+")",
     str(round(betahat_b7_2[2],4))+"("+str(round(sdhat_b7_2rides,4))+")"],
    [round(coefb8_1[0],4),round(coefb8_1[1],4),round(coefb8_1[2],4)],
    [round(coefb8_2[0],4),round(coefb8_2[1],4),round(coefb8_2[2],4)],
    [round(coefb9_1[0],4),round(coefb9_1[1],4),round(coefb9_1[2],4)],
    [round(coefb9_2[0],4),round(coefb9_2[1],4),round(coefb9_2[2],4)],
    [np.round(alphab10_11,4),np.round(alphab10_12,4),np.round(alphab10_13,4)],
    [np.round(alphab10_21,4),np.round(alphab10_22,4),np.round(alphab10_23,4)]],
    columns=['D','D*Above_med_pop','D*Above_med_rides'],
    index=['Bonus7_dummy_OLS','Bonus7_pen_OLS','Bonus8_dummy_Lasso','Bonus8_pen_Lasso',
          'Bonus9_dummy_LassoPoly','Bonus9_pen_LassoPoly','Bonus10_dummy_DbLasso',
           'Bonus10_pen_DbLasso'])

bonus_group2
```