Econ 434 Project

July 20, 2024

1 Econ 434 Final Project

1.1 Team Members: Mihnea Tatu-Chitoiu, Anulika Nwude-Jacobs

The scope of this project is to analyze whether Uber is a substitute or a complement for public transit. To analyze this matter, we will be running a number of regressions whose dependent variable will be the log of the number of rides for a public transit agency in a given year-month. Each of the regressions that we will be running will also include at least one independent variable related to the presence of UBER in the corresponding Metropolitan Statistical Area of the public transit agency. By interpreting the coefficients of such variables throughout multiple regressions, we will obtain a better understanding of the relation between public transit activity and UBER presence.

We begin the project by importing some of the libraries we will be using.

[1]:
```
import pandas as pd
from sklearn.impute import SimpleImputer
import statsmodels.api as sm
import numpy as np
from sklearn.linear_model import LassoCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
import warnings
warnings.filterwarnings('ignore')
```

Below we upload the dataframe we will use for this project. The columns of the dataframe are:

-UPTTotal – the number of rides for the public transit agency;
-treatUberX - a dummy for Uber presence in the corresponding MSA;
-treatGTNotStd - a variable measuring google search intensity for Uber in the corresponding MSA;
-popestimate - population in the corresponding MSA;
-employment - employment in the corresponding MSA;
-aveFareTotal - average fare for the public transit agency;
-VRHTTotal - vehicle hours for the public transit agency;
-VOMSTotal - number of vehicles employed by the public transit agency;
-VRMTotal - vehicle miles for the public transit agency;
-gasPrice - gas price in the corresponding MSA
-agency - the name of the public transit agency -city - the city where the public transit agency is located

-state - the state where the public transit agency is located
-dateSurvey- the date when the observation was registered

[2]:
```
df = pd.read_csv('/Users/sandinatatu/Desktop/uber_dataset.csv', index_col = 0)
df.head()
```

[2]:
```
    UPTTotal  treatUberX  treatGTNotStd  popestimate  employment  aveFareTotal  \
0   8296756         0.0           0.00      3163703     1572859      0.778015
1   7847113         0.0           1.40      3163703     1581307      0.778015
2   9011399         0.0           3.00      3163703     1592152      0.778015
3   8656389         0.0           2.25      3163703     1598167      0.778015
4   8378406         0.0           2.60      3163703     1593356      0.778015

   VRHTotal  VOMSTotal    VRMTotal  gasPrice  \
0  333329.0     2626.0  4740396.0     1.701
1  310535.0     2626.0  4398939.0     1.862
2  356761.0     2626.0  5176183.0     2.063
3  341191.0     2626.0  4889387.0     2.121
4  333418.0     2626.0  4747018.0     2.266

                                              agency     city state  \
0  King County Department of Transportation - Met…  Seattle    WA
1  King County Department of Transportation - Met…  Seattle    WA
2  King County Department of Transportation - Met…  Seattle    WA
3  King County Department of Transportation - Met…  Seattle    WA
4  King County Department of Transportation - Met…  Seattle    WA

   dateSurvey
0  2004-01-01
1  2004-02-01
2  2004-03-01
3  2004-04-01
4  2004-05-01
```

[3]:
```
df.shape
```

[3]:
```
(76213, 14)
```

[4]:
```
df.describe()
```

[4]:
```
           UPTTotal    treatUberX  treatGTNotStd   popestimate    employment  \
count  7.621300e+04  76213.000000   61824.000000  7.621300e+04  7.621300e+04
mean   1.557973e+06      0.125017       2.711728  3.287213e+06  1.544130e+06
std    1.247141e+07      0.329335       5.013406  5.090858e+06  2.363277e+06
min    2.100000e+01      0.000000       0.000000  6.944200e+04  3.215000e+04
25%    3.866000e+04      0.000000       0.000000  2.825200e+05  1.314350e+05
50%    1.214730e+05      0.000000       1.000000  8.370360e+05  3.903810e+05
75%    4.169980e+05      0.000000       2.250000  4.260236e+06  1.891851e+06
```

```
max    3.227260e+08       1.000000       56.024097  1.944570e+07  9.357873e+06

       aveFareTotal       VRHTotal      VOMSTotal      VRMTotal      gasPrice
count  72016.000000  7.602000e+04  76066.000000  7.603200e+04  76213.000000
mean       1.766518  4.040557e+04    185.220690  6.165114e+05      2.980399
std        4.134002  1.589262e+05    590.133284  2.326701e+06      0.653412
min        0.000026  4.000000e+01      1.000000  2.740000e+02      1.541000
25%        0.651672  4.076750e+03     22.000000  6.293075e+04      2.471000
50%        0.914369  8.701000e+03     48.000000  1.351800e+05      2.970000
75%        1.427222  2.191775e+04    125.000000  3.456725e+05      3.563000
max      135.849040  3.370515e+06  11260.000000  4.548309e+07      4.423000
```

Then, we check for the presence of mising values.

[5]:
```
df.isna().sum()
```

[5]:
```
UPTTotal             0
treatUberX           0
treatGTNotStd    14389
popestimate          0
employment           0
aveFareTotal      4197
VRHTotal           193
VOMSTotal          147
VRMTotal           181
gasPrice             0
agency               0
city                 0
state                0
dateSurvey           0
dtype: int64
```

Since some of our variables contain missing value, we will imputation via their mean to ensure that the dataset we will use for our regressions does not contain any missing values.

[6]:
```
imputer = SimpleImputer(strategy = 'mean')

df[['treatGTNotStd', 'aveFareTotal', 'VRHTotal', 'VOMSTotal', 'VRMTotal']] = \
imputer.fit_transform(df[['treatGTNotStd', 'aveFareTotal', 'VRHTotal',␣
 ↪'VOMSTotal', 'VRMTotal']])
```

We check to make sure that we do not have missing values anymore in our dataset.

[7]:
```
df.isna().sum()
```

[7]:
```
UPTTotal         0
treatUberX       0
treatGTNotStd    0
popestimate      0
```

```
employment        0
aveFareTotal      0
VRHTotal          0
VOMSTotal         0
VRMTotal          0
gasPrice          0
agency            0
city              0
state             0
dateSurvey        0
dtype: int64
```

Prior to running our regressions, we create two dataframes whose contentes will be used as independent variables:

-agency_dummies (a dataframe containing dummy values for the public transit agency of an observation)

-date_dummies (a dataframe containing dummy values for the year-month of an observation)

[8]:
```
agency_dummies = pd.get_dummies(df['agency']).astype('int')
date_dummies = pd.get_dummies(df['dateSurvey']).astype('int')
```

1.2 Regression 1

Our first regression is:

$$\log Y_{it} = \alpha + \beta D_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-alpha is a constant
-Y is UPTTotal
-D is treatUberX
-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHTotal, VOMSTotal, VRMTotal, gasPrice

[9]:
```
df['log_UPTTotal'] = np.log(df['UPTTotal'])

y = df['log_UPTTotal'].copy()

d_uberx = df['treatUberX'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()
w = sm.add_constant(w)

x_uberx = pd.concat([d_uberx, w], axis=1)
```

```
model1 = sm.OLS(y, x_uberx).fit()

summary = model1.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'treatUberX' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
                    coef    std err          t      P>|t|      [0.025      0.975]
0   treatUberX    0.1694      0.018      9.364      0.000       0.134       0.205
```

The coefficeint of treatUberX in this regression is 0.1694 with a standard error of 0.018. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to increase the number of rides for the public transit agency in that area by 16.94%, keeping everything else the same.

1.3 Regression 2

Our second regression is:

$$\log Y_{it} = \eta_i + \delta_t + \beta D_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHTTotal, VOMSTotal, VRMTotal, gasPrice

```
[10]: y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()

x_uberx = pd.concat([d_uberx, w, agency_dummies, date_dummies], axis=1)

model2 = sm.OLS(y, x_uberx).fit()

summary = model2.summary()
coef_table = summary.tables[1]
```

```
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'treatUberX' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
                    coef    std err          t    P>|t|      [0.025      0.975]
0  treatUberX    -0.0606      0.006     -9.829    0.000      -0.073      -0.048
```

The coefficeint of treatUberX in this regression is -0.0606 with a standard error of 0.006. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to decrease the number of of rides for the public transit agency in that area by 6.06%, keeping everything else the same.

1.4 Regression 3

Below we create P, a a dummy that takes value 1 if the corresponding MSA has population larger than the median population in the dataset and 0 otherwise. We also create a column we call 'D * P', which is the product between our newly-created P column and the 'treatUberX' column.

[11]:
```
df['P'] = np.where(df['popestimate'] > df['popestimate'].median(),1,0)
df['D*P'] = df['P'] * df['treatUberX']
```

Our third regression is:

$$\log Y_{it} = \eta_i + \delta_t + \beta_1 D_{it} + \beta_2 D_{it} P_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * P is the product between our newly-created P column and the 'treatUberX' column.
-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHTTotal, VOMSTotal, VRMTotal, gasPrice

[12]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()
```

```
x_uberx = pd.concat([d_uberx, dp_uberx, w, agency_dummies, date_dummies],␣
↪axis=1)

model3 = sm.OLS(y, x_uberx).fit()

summary = model3.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_two_rows = []
for row in coef_table.data[1:3]:
    first_two_rows.append(row)

df_summary = pd.DataFrame(first_two_rows, columns=titles)
print(df_summary)
```

```
                  coef    std err          t    P>|t|     [0.025     0.975]
0  treatUberX   -0.0303      0.009     -3.218    0.001     -0.049     -0.012
1         D*P   -0.0397      0.009     -4.249    0.000     -0.058     -0.021
```

The coefficeint of treatUberX in this regression is -0.0303 with a standard error of 0.009. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to decrease the number of rides for the public transit agency in that area by 3.03%, keeping everything else the same.

The coefficeint of D * P in this regression is -0.0397 with a standard error of 0.009. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a larger MSA is expected to decrease the number of rides for the public transit agency in that area by a further 3.97%, keeping everything else the same.

1.5 Regression 4

Below we create F, a a dummy that takes value 1 iif the number of rides of the public travel agency is larger than the median number of rides among all public transit agencies in the dataset. We also create a column we call 'D * F', which is the product between our newly-created F column and the 'treatUberX' column.

[13]:
```
df['F'] = np.where(df['UPTTotal'] > df['UPTTotal'].median(),1,0)
df['D*F'] = df['F'] * df['treatUberX']
```

Our fourth regression is:

$$\log Y_{it} = \eta_i + \delta_t + \beta_1 D_{it} + \beta_2 D_{it} F_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * F is the product between our newly-created F column and the 'treatUberX' column.

-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHT-Total, VOMSTotal, VRMTotal, gasPrice

[14]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

x_uberx = pd.concat([d_uberx,df_uberx, w, date_dummies, agency_dummies], axis=1)

model4 = sm.OLS(y, x_uberx).fit()

summary = model4.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_two_rows = []
for row in coef_table.data[1:3]:
    first_two_rows.append(row)

df_summary = pd.DataFrame(first_two_rows, columns=titles)
print(df_summary)
```

```
                   coef    std err          t    P>|t|     [0.025      0.975]
0  treatUberX   -0.0537      0.008     -6.742    0.000     -0.069      -0.038
1         D*F   -0.0110      0.008     -1.370    0.171     -0.027       0.005
```

The coefficeint of treatUberX in this regression is -0.0537 with a standard error of 0.008. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to decrease the number of rides for the public transit agency in that area by 5.37%, keeping everything else the same.

The coefficeint of D * F in this regression is -0.0110 with a standard error of 0.008. This result is not statistically significant at the 95% confidence level. This result indicates that the presence of UBER in an area with a high number of public transit agency rides is expected to decrease the number of rides for the public transit agency in that area by a further 1.11%, keeping everything else the same.

1.6 Regression 5

Our fifth regression is:

$$\log Y_{it} = \eta_i + \delta_t + \beta_1 D_{it} + \beta_2 D_{it} P_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created

-D is treatUberX
-D * P is the product between our newly-created P column and the 'treatUberX' column.
-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHT-Total, VOMSTotal, VRMTotal, gasPrice

It is identical to our third regression. However, we will estimate this regression this time through LASSO. Prior to running LASSO we will scale our variables using sklearn's StandardScaler. We will select the LASSO penalty parameter in this case through 5-fold cross-validation

[15]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

x_uberx = pd.concat([d_uberx, dp_uberx, w, agency_dummies, date_dummies],␣
↪axis=1)

scaler5 = StandardScaler()
x_uberx = scaler5.fit_transform(x_uberx)

lasso5 = LassoCV(cv = 5, fit_intercept = False)
model5 = lasso5.fit(x_uberx, y)

coefficients = model5.coef_
feature_names = ['treatUberX', 'D*P'] + w.columns.tolist() + agency_dummies.
↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

print(coef_df.head(2))
```

```
      Feature  Coefficient
0  treatUberX          0.0
1         D*P          0.0
```

The coefficeint of treatUberX in this regression is 0, indicating that the presence of UBER in a MSA is not expected to impact the number of rides for the public transit agency in that area.

The coefficeint of D * P is 0, indicating that presence of UBER in a larger MSA is not expected to further impact the number of rides for the public transit agency in that area.

1.7 Regression 6

Our sixth regression is:

$$\log Y_{it} = \eta_i + \delta_t + \beta_1 D_{it} + \beta_2 D_{it} F_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * F is the product between our newly-created F column and the 'treatUberX' column.
-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHTTotal, VOMSTotal, VRMTotal, gasPrice

It is identical to our fourth regression. However, we will estimate this regression this time through LASSO. Prior to running LASSO we will scale our idnependent variables using sklearn's StandardScaler. We will select the LASSO penalty parameter in this case through 5-fold cross-validation

[16]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

x_uberx = pd.concat([d_uberx,df_uberx, w, date_dummies, agency_dummies], axis=1)

scaler6 = StandardScaler()
x_uberx = scaler6.fit_transform(x_uberx)

lasso6 = LassoCV(cv = 5, fit_intercept = False)
model6 = lasso6.fit(x_uberx, y)

coefficients = model6.coef_
feature_names = ['treatUberX', 'D*F'] + w.columns.tolist() + agency_dummies.
 ↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

print(coef_df.head(2))
```

```
      Feature  Coefficient
0  treatUberX          0.0
1         D*F          0.0
```

The coefficeint of treatUberX in this regression is 0, indicating that the presence of UBER in a MSA is not expected to impact the number of rides for the public transit agency in that area.

The coefficeint of D * F is 0, indicating the presence of UBER in an area with a high number of public transit agency rides is not expected to further impact the number of rides for the public transit agency in that area.

1.8 Regression 7

Our seventh regression is:

$$\log Y_{it} = \eta_i + \delta_t + \beta_1 D_{it} + \beta_2 D_{it} P_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * P is the product between our newly-created P column and the 'treatUberX' column.
-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHTTotal, VOMSTotal, VRMTotal, gasPrice

This regression is identical to our third and fifth regressions. However, this time we will obtain estimates for beta1 and beta2 through Double Lasso. We first focus on obtaining a coefficient for beta 1 through Double Lasso. To do so, we will first run Lasso of Y on the variables from above. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from LASSO are different from 0 in a list i call 'nz1'.

[17]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

x_uberx = pd.concat([d_uberx, dp_uberx, w, agency_dummies, date_dummies],␣
↪axis=1)

scaler7 = StandardScaler()
x_uberx = scaler7.fit_transform(x_uberx)

lasso7 = LassoCV(cv = 5, fit_intercept = False)
model7 = lasso7.fit(x_uberx, y)

coefficients = model7.coef_

feature_names = ['treatUberX', 'D*P'] + w.columns.tolist() + agency_dummies.
↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

nz1 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

Then, I run LASSO of D (treatUberX) on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list i call 'nz2'.

[18]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()
```

```
d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

temp = pd.concat([dp_uberx, w, agency_dummies, date_dummies], axis = 1)

scaler72 = StandardScaler()
temp = scaler72.fit_transform(temp)

lasso72 = LassoCV(cv = 5)
model72 = lasso72.fit(temp, d_uberx)

coefficients72 = model72.coef_

feature_names72 = ['D*P'] + w.columns.tolist() + agency_dummies.columns.
↪tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names72, 'Coefficient':␣
↪coefficients72})

nz2 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

I also run LASSO of P * D on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list i call ‘nz3’.

[19]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

temp2 = pd.concat([d_uberx, w, agency_dummies, date_dummies], axis = 1)

scaler73 = StandardScaler()
temp2 = scaler73.fit_transform(temp2)

lasso73 = LassoCV(cv = 5)
model73 = lasso73.fit(temp2, dp_uberx)

coefficients73 = model73.coef_

feature_names73 = ['treatUberX'] + w.columns.tolist() + agency_dummies.columns.
↪tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names73, 'Coefficient':␣
↪coefficients73})

nz3 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

To obtain an estimate and standard errors for ‘treatUberX’, we now run OLS on the unscaled ‘treatUberX’ and the independent variables that yielded non-zero coefficients in at least one of the first two LASSO regressions.

[20]:
```
all_nz = list(set(['treatUberX'] + nz1 + nz2))

y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

x_uberx = pd.concat([d_uberx, dp_uberx, w, agency_dummies, date_dummies],␣
↪axis=1)

lhs7= x_uberx[all_nz].copy()
lhs7 = sm.add_constant(lhs7)

model7final = sm.OLS(y, lhs7).fit()

summary = model7final.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'treatUberX' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
                    coef    std err          t    P>|t|     [0.025     0.975]
0  treatUberX    -0.0864      0.032     -2.733    0.006     -0.148     -0.024
```

The ceofficient of treatUberX in this regression is -0.0864 with a standard error of 0.032. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to decrease the number of rides for the public transit agency in that area by 8.64%, keeping everything else the same.

To obtain an estimate and standard errors for ‘D * P’, we now run OLS on the unscaled ‘treatUberX’ and the unscaled independent variables that yielded non-zero coefficients in at least one of the first and third LASSO regressions.

[21]:
```
all_nz3 = list(set(['D*P'] + nz1 + nz3))

lhs72= x_uberx[all_nz3].copy()
```

```
lhs72 = sm.add_constant(lhs72)

model72final = sm.OLS(y, lhs72).fit()

summary = model72final.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'D*P' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
                coef    std err          t    P>|t|      [0.025      0.975]
0  D*P       -0.2231      0.042     -5.346    0.000      -0.305      -0.141
```

The ceofficient of D * P in this regression is -0.2231 with a standard error of 0.042. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a large MSA is expected to decrease the number of rides for the public transit agency in that area by a further 22.31%, keeping everything else the same.

1.9 Regression 8

Our eighth regression is:

$$\log Y_{it} = \eta_i + \delta_t + \beta_1 D_{it} + \beta_2 D_{it} F_{it} + \gamma W_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * F is the product between our newly-created F column and the 'treatUberX' column.
-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHTotal, VOMSTotal, VRMTotal, gasPrice

This regression is identical to our fourth and sixth regressions. However, this time we will obtain estimates for beta1 and beta2 through Double Lasso. We first focus on obtaining a coefficient for beta 1 through Double Lasso. To do so, we will first run Lasso of Y on the variables from above. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from LASSO are different from 0 in a list I call 'nz4'.

[22]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()
```

```
d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

x_uberx = pd.concat([d_uberx, df_uberx, w, agency_dummies, date_dummies],␣
 ↪axis=1)

scaler8 = StandardScaler()
x_uberx = scaler8.fit_transform(x_uberx)

lasso8 = LassoCV(cv = 5, fit_intercept = False)
model8 = lasso8.fit(x_uberx, y)

coefficients2 = model8.coef_

feature_names2 = ['treatUberX', 'D*F'] + w.columns.tolist() + agency_dummies.
 ↪columns.tolist() + date_dummies.columns.tolist()
coef_df8 = pd.DataFrame({'Feature': feature_names2, 'Coefficient':␣
 ↪coefficients2})

nz4 = coef_df8[coef_df8['Coefficient'] != 0]['Feature'].tolist()
```

Then, I run LASSO of D (treatUberX) on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list I call 'nz5'.

[23]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

temp3 = pd.concat([df_uberx, w, agency_dummies, date_dummies], axis = 1)

scaler82 = StandardScaler()
temp3 = scaler82.fit_transform(temp3)

lasso82 = LassoCV(cv = 5)
model82 = lasso82.fit(temp3, d_uberx)

coefficients82 = model82.coef_

feature_names82 = ['D*F'] + w.columns.tolist() + agency_dummies.columns.
 ↪tolist() + date_dummies.columns.tolist()
coef_df8 = pd.DataFrame({'Feature': feature_names82, 'Coefficient':␣
 ↪coefficients82})
```

```
nz5 = coef_df8[coef_df8['Coefficient'] != 0]['Feature'].tolist()
```

I also run LASSO of F * D on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list I call ‘nz6’.

[24]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

temp4 = pd.concat([d_uberx, w, agency_dummies, date_dummies], axis = 1)

scaler83 = StandardScaler()
temp4 = scaler83.fit_transform(temp4)

lasso83 = LassoCV(cv = 5)
model83 = lasso83.fit(temp4, df_uberx)

coefficients83 = model83.coef_

feature_names83 = ['treatUberX'] + w.columns.tolist() + agency_dummies.columns.
 ↪tolist() + date_dummies.columns.tolist()
coef_df8 = pd.DataFrame({'Feature': feature_names83, 'Coefficient':␣
 ↪coefficients83})

nz6 = coef_df8[coef_df8['Coefficient'] != 0]['Feature'].tolist()
```

To obtain an estimate and standard errors for ‘treatUberX’, we now run OLS on the unscaled ‘treatUberX’ and the independent variables that yielded non-zero coefficients in at least one of the first two LASSO regressions (that we ran after the “Regression 8” tite)

[25]:
```
all_nz2 = list(set(['treatUberX'] + nz4 + nz5))

y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

x_uberx = pd.concat([d_uberx, df_uberx, w, agency_dummies, date_dummies],␣
 ↪axis=1)

lhs8= x_uberx[all_nz2].copy()
lhs8 = sm.add_constant(lhs8)
```

```
model8final = sm.OLS(y, lhs8).fit()

summary2 = model8final.summary()
coef_table2 = summary2.tables[1]
titles = coef_table2.data[0]
first_variable_row = []
for row in coef_table2.data:
    if 'treatUberX' in row:
        first_variable_row = row
        break

df_summary2 = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary2)
```

```
                     coef    std err          t    P>|t|     [0.025     0.975]
0  treatUberX     -0.0305      0.015     -2.044    0.041     -0.060     -0.001
```

The coefficient of treatUberX in this regression is -0.0305 with a standard error of 0.015. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to decrease the number of rides for the public transit agency in that area by 3.05%, keeping everything else the same.

[26]:
```
y = df['log_UPTTotal'].copy()

all_nz6 = list(set(['D*F'] + nz4 + nz6))

lhs82= x_uberx[all_nz6].copy()
lhs82 = sm.add_constant(lhs82)

model82final = sm.OLS(y, lhs82).fit()

summary2 = model82final.summary()
coef_table2 = summary2.tables[1]
titles = coef_table2.data[0]
first_variable_row = []
for row in coef_table2.data:
    if 'D*F' in row:
        first_variable_row = row
        break

df_summary2 = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary2)
```

```
              coef    std err          t    P>|t|     [0.025     0.975]
0  D*F      2.5126      0.034     74.436    0.000      2.446      2.579
```

The coefficient of D * F in this regression is 2.5126 with a standard error of 0.034. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in an area with a high number of public transit agency rides is expected to increase the number of rides for the public transit agency in that area by a further 251.26%, keeping everything else the same. This result is an exaggeration which occured because of the absence of several variables from this last regression, such as 'gasPrice'. Furthermore, this last regression yielded more exaggerated results, as the coefficient of "treatUberX" is -1.1798. This would mean that the presence of UBER in a MSA is expected to decrease the number of rides for the public transit agency in that area by more than 100%, keeping everything else constant.

1.10 Regression 9

Our ninth regression is:

$$\log Y_{it} = \eta_i + \delta_t + D_{it}\beta_1 + D_{it}P_{it}\beta_2 + \gamma\tilde{W}'_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * P is the product between our newly-created P column and the 'treatUberX' column.
-W~, which includes all interactions of order 5 of variables in the vector W

We will estimate this regression this through LASSO. Prior to running LASSO we will scale our variables using sklearn's StandardScaler. We will select the LASSO penalty parameter in this case through 5-fold cross-validation.

[27]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

x_uberx = pd.concat([d_uberx, dp_uberx, wnew, agency_dummies, date_dummies],␣
 ↪axis=1)
x_uberx.columns = x_uberx.columns.astype('str')

scaler9 = StandardScaler()
x_uberx = scaler9.fit_transform(x_uberx)

lasso9 = LassoCV(cv = 5, fit_intercept = False)
model9 = lasso9.fit(x_uberx, y)

coefficients = model9.coef_
```

```
feature_names = ['treatUberX', 'D*P'] + wnew.columns.tolist() + agency_dummies.
↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

print(coef_df.head(2))
```

```
      Feature  Coefficient
0  treatUberX          0.0
1         D*P          0.0
```

The coefficeint of treatUberX in this regression is 0, indicating that the presence of UBER in a MSA is not expected to impact the number of rides for the public transit agency in that area.

The coefficeint of D * P is 0, indicating that presence of UBER in a larger MSA is not expected to further impact the number of rides for the public transit agency in that area.

1.11 Regression 10

Our tenth regression is:

$$\log Y_{it} = \eta_i + \delta_t + D_{it}\beta_1 + D_{it}F_{it}\beta_2 + \gamma \tilde{W}'_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * F is the product between our newly-created F column and the 'treatUberX' column.
-W~, which includes all interactions of order 5 of variables in the vector W

We will estimate this regression through LASSO. Prior to running LASSO we will scale our variables using sklearn's StandardScaler. We will select the LASSO penalty parameter in this case through 5-fold cross-validation.

[28]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()

polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

x_uberx = pd.concat([d_uberx, df_uberx, wnew, agency_dummies, date_dummies],␣
↪axis=1)
x_uberx.columns = x_uberx.columns.astype('str')
```

```
scaler10 = StandardScaler()
x_uberx = scaler10.fit_transform(x_uberx)

lasso10 = LassoCV(cv = 5, fit_intercept = False)
model10 = lasso10.fit(x_uberx, y)

coefficients = model10.coef_
feature_names = ['treatUberX', 'D*F'] + wnew.columns.tolist() + agency_dummies.
 ↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

print(coef_df.head(2))
```

```
      Feature  Coefficient
0  treatUberX          0.0
1         D*F          0.0
```

The coefficient of treatUberX in this regression is 0, indicating that the presence of UBER in a MSA is not expected to impact the number of rides for the public transit agency in that area.

The coefficeint of D * F is 0, indicating that presence of UBER in an area with a high number of public transit agency rides is not expected to impact the number of rides for the public transit agency in that area.

1.12 Regression 11

Our eleventh regression is:

$$\log Y_{it} = \eta_i + \delta_t + D_{it}\beta_1 + D_{it}P_{it}\beta_2 + \gamma\tilde{W}'_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * P is the product between our newly-created P column and the 'treatUberX' column.
-W~ includes all interactions of order 5 of variables in the vector W

This regression is identical to our ninth regression. However, this time we will obtain estimates for beta1 and beta2 through Double Lasso. We first focus on obtaining a coefficient for beta 1 through Double Lasso. To do so, we will first run Lasso of Y on the variables from above. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from LASSO are different from 0 in a list I call 'nz111'.

[29]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()
```

```
polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

x_uberx = pd.concat([d_uberx, dp_uberx, wnew, agency_dummies, date_dummies],␣
 ↪axis=1)
x_uberx.columns = x_uberx.columns.astype('str')

scaler111 = StandardScaler()
x_uberx = scaler111.fit_transform(x_uberx)

lasso111 = LassoCV(cv = 5, fit_intercept = False)
model111 = lasso111.fit(x_uberx, y)

coefficients = model111.coef_

feature_names = ['treatUberX', 'D*P'] + wnew.columns.tolist() + agency_dummies.
 ↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

nz111 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

Then, I run LASSO of D (treatUberX) on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list i call 'nz112'.

[30]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

temp = pd.concat([dp_uberx, wnew, agency_dummies, date_dummies], axis = 1)
temp.columns = temp.columns.astype('str')

scaler112 = StandardScaler()
temp = scaler112.fit_transform(temp)

lasso112 = LassoCV(cv = 5)
model112 = lasso112.fit(temp, d_uberx)

coefficients112 = model112.coef_
```

```
feature_names112 = ['D*P'] + wnew.columns.tolist() + agency_dummies.columns.
 ↪tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names112, 'Coefficient':␣
 ↪coefficients112})

nz112 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

I also run LASSO of P * D on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list I call 'nz113'.

[31]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

temp2 = pd.concat([d_uberx, wnew, agency_dummies, date_dummies], axis = 1)
temp2.columns = temp2.columns.astype('str')

scaler113 = StandardScaler()
temp2 = scaler113.fit_transform(temp2)

lasso113 = LassoCV(cv = 5)
model113 = lasso113.fit(temp2, dp_uberx)

coefficients113 = model113.coef_

feature_names113 = ['treatUberX'] + wnew.columns.tolist() + agency_dummies.
 ↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names113, 'Coefficient':␣
 ↪coefficients113})

nz113 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

To obtain an estimate and standard errors for 'treatUberX', we now run OLS on the unscaled 'treatUberX' and the independent variables that yielded non-zero coefficients in at least one of the first two LASSO regressions.

[32]:
```
all_nz = list(set(['treatUberX'] + nz111 + nz112))

y = df['log_UPTTotal'].copy()
```

```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()
polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
dp_uberx = df['D*P'].copy()

x_uberx = pd.concat([d_uberx, dp_uberx, wnew, agency_dummies, date_dummies],␣
 ↪axis=1)

x_uberx.columns = x_uberx.columns.map(str)

all_nz = list(map(str, all_nz))

all_nz_filtered = [col for col in all_nz if col in x_uberx.columns]

lhs111 = x_uberx[all_nz_filtered]
lhs111 = sm.add_constant(lhs111)

model111final = sm.OLS(y, lhs111).fit()

summary = model111final.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'treatUberX' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
                    coef    std err          t   P>|t|     [0.025     0.975]
0  treatUberX  1.899e-07   3.19e-09     59.574   0.000   1.84e-07   1.96e-07
```

The coefficient of treatUberX in this regression is 1.958e-07 with a standard error of 3.29e-09. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to increase the number of rides for the public transit agency in that area by a very small amount, keeping everything else the same.

To obtain an estimate and standard errors for ‘D * P’, we now run OLS on the unscaled ‘treatUberX’ and the unscaled independent variables that yielded non-zero coefficients in at least one of the first and third LASSO regressions.

[33]:
```
all_nz112 = list(set(['D*P'] + nz111 + nz113))
```

```
all_nz112 = list(map(str, all_nz112))

all_nz112_filtered = [col for col in all_nz112 if col in x_uberx.columns]

lhs112= x_uberx[all_nz112_filtered].copy()
lhs112 = sm.add_constant(lhs112)

model112final = sm.OLS(y, lhs112).fit()

summary = model112final.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'D*P' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
              coef    std err          t    P>|t|     [0.025     0.975]
0  D*P     -0.2213      0.042     -5.308    0.000     -0.303     -0.140
```

The ceofficient of D * P in this regression is -0.2213 with a standard error of 0.042. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a large MSA is expected to decrease the number of rides for the public transit agency in that area by a further 22.13%, keeping everything else the same.

1.13 Regression 12

Our twelth regression is:

$$\log Y_{it} = \eta_i + \delta_t + D_{it}\beta_1 + D_{it}F_{it}\beta_2 + \gamma\tilde{W}'_{it} + \epsilon_{it}$$

Where:

-eta is the agency dummy that we previously created
-tau is the year-month dummy we previously created
-D is treatUberX
-D * F is the product between our newly-created F column and the 'treatUberX' column
-W~ includes all interactions of order 5 of variables in the vector W

This regression is identical to our tenth regression. However, this time we will obtain estimates for beta1 and beta2 through Double Lasso. We first focus on obtaining a coefficient for beta 1 through Double Lasso. To do so, we will first run Lasso of Y on the variables from above. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from LASSO are different from 0 in a list I call 'nz121'

[34]:
```
y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

x_uberx = pd.concat([d_uberx, df_uberx, wnew, agency_dummies, date_dummies],␣
 ↪axis=1)
x_uberx.columns = x_uberx.columns.astype('str')

scaler121 = StandardScaler()
x_uberx = scaler121.fit_transform(x_uberx)

lasso121 = LassoCV(cv = 5, fit_intercept = False)
model121 = lasso121.fit(x_uberx, y)

coefficients = model121.coef_

feature_names = ['treatUberX', 'D*F'] + wnew.columns.tolist() + agency_dummies.
 ↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})

nz121 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

Then, I run LASSO of D (treatUberX) on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list I call 'nz122'.

[35]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
 ↪'VRMTotal', 'gasPrice']].copy()

polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

temp = pd.concat([df_uberx, wnew, agency_dummies, date_dummies], axis = 1)
temp.columns = temp.columns.astype('str')

scaler122 = StandardScaler()
temp = scaler122.fit_transform(temp)
```

```
lasso122 = LassoCV(cv = 5)
model122 = lasso122.fit(temp, d_uberx)

coefficients122 = model122.coef_

feature_names122 = ['D*F'] + wnew.columns.tolist() + agency_dummies.columns.
    ↪tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names122, 'Coefficient':␣
    ↪coefficients122})

nz122 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

I also run LASSO of F * D on all of the other independent variables. Again, we obtain the regularization parameter for LASSO through cross-validation. Then, I store the name of variables whose coefficients from this LASSO are different from 0 in a list I call 'nz123'

[36]:
```
w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
    ↪'VRMTotal', 'gasPrice']].copy()

polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

temp2 = pd.concat([d_uberx, wnew, agency_dummies, date_dummies], axis = 1)
temp2.columns = temp2.columns.astype('str')

scaler123 = StandardScaler()
temp2 = scaler123.fit_transform(temp2)

lasso123 = LassoCV(cv = 5)
model123 = lasso123.fit(temp2, df_uberx)

coefficients123 = model123.coef_

feature_names123 = ['treatUberX'] + wnew.columns.tolist() + agency_dummies.
    ↪columns.tolist() + date_dummies.columns.tolist()
coef_df = pd.DataFrame({'Feature': feature_names123, 'Coefficient':␣
    ↪coefficients123})

nz123 = coef_df[coef_df['Coefficient'] != 0]['Feature'].tolist()
```

To obtain an estimate and standard errors for 'treatUberX', we now run OLS on the unscaled 'treatUberX' and the independent variables that yielded non-zero coefficients in at least one of the first two LASSO regressions.

[37]:
```python
all_nz = list(set(['treatUberX'] + nz121 + nz122))

y = df['log_UPTTotal'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()
polint = PolynomialFeatures(degree=5, include_bias=False)
wnew = pd.DataFrame(polint.fit_transform(w))

d_uberx = df['treatUberX'].copy()
df_uberx = df['D*F'].copy()

x_uberx = pd.concat([d_uberx, df_uberx, wnew, agency_dummies, date_dummies],␣
↪axis=1)

x_uberx.columns = x_uberx.columns.map(str)


all_nz_filtered = [col for col in all_nz if col in x_uberx.columns]

lhs121 = x_uberx[all_nz_filtered]
lhs121 = sm.add_constant(lhs121)

model121final = sm.OLS(y, lhs121).fit()

summary = model121final.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'treatUberX' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
                      coef    std err          t    P>|t|     [0.025     0.975]
0  treatUberX      -0.0725      0.029     -2.498    0.013     -0.129     -0.016
```

The coefficient of treatUberX in this regression is -0.0725 with a standard error of 0.029. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to decrease the number of rides for the public transit agency in that area by 7.25%, keeping everything else the same.

To obtain an estimate and standard errors for 'D * F', we now run OLS on the unscaled 'treatUberX' and the unscaled independent variables that yielded non-zero coefficients in at least one of the first and third LASSO regressions.

[38]:
```
all_nz122 = list(set(['D*F'] + nz121 + nz123))

all_nz122 = list(map(str, all_nz122))

all_nz122_filtered = [col for col in all_nz122 if col in x_uberx.columns]

lhs122= x_uberx[all_nz122_filtered].copy()
lhs122 = sm.add_constant(lhs122)

model122final = sm.OLS(y, lhs122).fit()

summary = model122final.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'D*F' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
              coef    std err          t    P>|t|     [0.025     0.975]
0  D*F      2.4669      0.034     72.994    0.000      2.401      2.533
```

The coefficient of D * F in this regression is 2.4669 with a standard error of 0.034. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in an area with a high number of public transit agency rides is expected to increase the number of rides for the public transit agency in that area by a further 246.69%, keeping everything else the same. This result is an exaggeration which occured because of the absence of several variables from this last regression, such as 'gasPrice'. Furthermore, this last regression yielded more exaggerated results, as the coefficient of "treatUberX" is -1.1699. This is equivalent to saying that the presence of UBER in a MSA is expected to decrease the number of rides for the public transit agency in that area by more than 100%, keeping everything else constant.

1.14 Bonus Regression

Our bonus regression is:

$$\log Y_{it} = \alpha + \eta_i + D_{it}\beta_1 + \gamma W'_{it} + \epsilon_{it}$$

Where:

Where:

-eta is the agency dummy that we previously created
-D is treatUberX

-W is the vector including remaining variables: popestimate, employment, aveFareTotal, VRHTTotal, VOMSTotal, VRMTotal, gasPrice

This regression does not include our year-month dummy and will be estimated through OLS. The penalty parameter in this case will be chosen through cross-validation.

[39]:
```
df['log_UPTTotal'] = np.log(df['UPTTotal'])

y = df['log_UPTTotal'].copy()

d_uberx = df['treatUberX'].copy()

w = df[['popestimate', 'employment', 'aveFareTotal', 'VRHTotal', 'VOMSTotal',␣
↪'VRMTotal', 'gasPrice']].copy()
w = sm.add_constant(w)

x_uberx = pd.concat([d_uberx, w, agency_dummies], axis=1)

modelb = sm.OLS(y, x_uberx).fit()

summary = modelb.summary()
coef_table = summary.tables[1]
titles = coef_table.data[0]
first_variable_row = []
for row in coef_table.data:
    if 'treatUberX' in row:
        first_variable_row = row
        break

df_summary = pd.DataFrame([first_variable_row], columns=titles)
print(df_summary)
```

```
                    coef    std err          t    P>|t|     [0.025     0.975]
0  treatUberX     0.0270      0.004      6.326    0.000      0.019      0.035
```

The coefficient of treatUberX in this regression is 0.0270 with a standard error of 0.004. This result is statistically significant at the 95% confidence level. This result indicates that the presence of UBER in a MSA is expected to increase the number of rides for the public transit agency in that area by 2.7%, keeping everything else the same.

1.15 Summary Tables

[40]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import font_manager

# Data for treatUberX
```

```
data_treatUberX = {
    'Regression': [
        'Regression 1', 'Regression 2', 'Regression 3', 'Regression 4',
        'Regression 5', 'Regression 6', 'Regression 7', 'Regression 8',
        'Regression 9', 'Regression 10', 'Regression 11', 'Regression 12',
        'Bonus Regression'
    ],
    'Coefficient': [
        0.1694, -0.0606, -0.0303, -0.0537,
        0.0, 0.0, -0.0864, -0.0305,
        0.0, 0.0, 1.012e-05, -0.0725,
        0.0270
    ],
    'Interpretation': [
        'Increase by 16.94%', 'Decrease by 6.06%', 'Decrease by 3.03%',␣
↪'Decrease by 5.37%',
        'No impact', 'No impact', 'Decrease by 8.64%', 'Decrease by 3.05%',
        'No impact', 'No impact', 'Increase by 0.001%', 'Decrease by 7.25%',
        'Increase by 2.7%'
    ]
}

# Convert to DataFrame
df_treatUberX = pd.DataFrame(data_treatUberX)

# Plotting the table for treatUberX
fig, ax = plt.subplots(figsize=(14, 7))
ax.axis('off')
font_properties = font_manager.FontProperties(family='DejaVu Sans',␣
↪weight='bold', size=14)

colors = sns.color_palette("Pastel1", len(df_treatUberX.columns))
tbl = ax.table(
    cellText=df_treatUberX.values,
    colLabels=df_treatUberX.columns,
    cellLoc='center',
    loc='center',
    colColours=colors
)
tbl.auto_set_font_size(False)
tbl.set_fontsize(12)
tbl.scale(1.5, 1.5)
for key, cell in tbl.get_celld().items():
    cell.set_edgecolor('navy')
    cell.set_linewidth(1.5)
    if key[0] == 0:
        cell.get_text().set_fontproperties(font_properties)
```

```
        cell.get_text().set_color('navy')
    cell.get_text().set_fontproperties(font_properties)
    cell.PAD = 0.3
ax.set_title('Summary of Regression Results for treatUberX', fontsize=20, fontweight='bold', color='navy', fontname='DejaVu Sans')
ax.title.set_position([0.5, 1.1])  # Adjust spacing

plt.show()
```

Summary of Regression Results for treatUberX

Regression	Coefficient	Interpretation
Regression 1	0.1694	Increase by 16.94%
Regression 2	-0.0606	Decrease by 6.06%
Regression 3	-0.0303	Decrease by 3.03%
Regression 4	-0.0537	Decrease by 5.37%
Regression 5	0.0	No impact
Regression 6	0.0	No impact
Regression 7	-0.0864	Decrease by 8.64%
Regression 8	-0.0305	Decrease by 3.05%
Regression 9	0.0	No impact
Regression 10	0.0	No impact
Regression 11	1.012e-05	Increase by 0.001%
Regression 12	-0.0725	Decrease by 7.25%
Bonus Regression	0.027	Increase by 2.7%

[41]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import font_manager

# Data for D*P
data_DP = {
    'Regression': [
        'Regression 3', 'Regression 5', 'Regression 7', 'Regression 9', 'Regression 11'
    ],
    'Coefficient': [
        -0.0397, 0.0, -0.2231, 0.0, -0.2213
    ],
    'Interpretation': [
        'Decrease by 3.97%', 'No impact', 'Decrease by 22.31%', 'No impact', 'Decrease by 22.13%'
    ]
}

# Convert to DataFrame
df_DP = pd.DataFrame(data_DP)
```

```python
# Plotting the table for D*P
fig, ax = plt.subplots(figsize=(14, 7))
ax.axis('off')
font_properties = font_manager.FontProperties(family='DejaVu Sans', weight='bold', size=14)

colors = sns.color_palette("Pastel1", len(df_DP.columns))
tbl = ax.table(
    cellText=df_DP.values,
    colLabels=df_DP.columns,
    cellLoc='center',
    loc='center',
    colColours=colors
)
tbl.auto_set_font_size(False)
tbl.set_fontsize(12)
tbl.scale(1.5, 1.5)
for key, cell in tbl.get_celld().items():
    cell.set_edgecolor('navy')
    cell.set_linewidth(1.5)
    if key[0] == 0:
        cell.get_text().set_fontproperties(font_properties)
        cell.get_text().set_color('navy')
    cell.get_text().set_fontproperties(font_properties)
    cell.PAD = 0.3
ax.set_title('Summary of Regression Results for D*P', fontsize=20, fontweight='bold', color='navy', fontname='DejaVu Sans')
ax.title.set_position([0.5, 1.05])  # Decrease the distance between title and table

plt.show()
```

Summary of Regression Results for D*P

Regression	Coefficient	Interpretation
Regression 3	-0.0397	Decrease by 3.97%
Regression 5	0.0	No impact
Regression 7	-0.2231	Decrease by 22.31%
Regression 9	0.0	No impact
Regression 11	-0.2213	Decrease by 22.13%

[42]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import font_manager

# Data for D*F
data_DF = {
    'Regression': [
        'Regression 4', 'Regression 6', 'Regression 8', 'Regression 10',␣
 ↪'Regression 12'
    ],
    'Coefficient': [
        -0.0110, 0.0, 2.5126, 0.0, 2.4669
    ],
    'Interpretation': [
        'Decrease by 1.11%', 'No impact', 'Increase by 251.26%', 'No impact',␣
 ↪'Increase by 246.69%'
    ]
}

# Convert to DataFrame
df_DF = pd.DataFrame(data_DF)

# Plotting the table for D*F
fig, ax = plt.subplots(figsize=(14, 7))
ax.axis('off')
font_properties = font_manager.FontProperties(family='DejaVu Sans',␣
 ↪weight='bold', size=14)

colors = sns.color_palette("Pastel1", len(df_DF.columns))
tbl = ax.table(
    cellText=df_DF.values,
    colLabels=df_DF.columns,
    cellLoc='center',
    loc='center',
    colColours=colors
)
tbl.auto_set_font_size(False)
tbl.set_fontsize(12)
tbl.scale(1.5, 1.5)
for key, cell in tbl.get_celld().items():
    cell.set_edgecolor('navy')
    cell.set_linewidth(1.5)
    if key[0] == 0:
        cell.get_text().set_fontproperties(font_properties)
        cell.get_text().set_color('navy')
```

```
    cell.get_text().set_fontproperties(font_properties)
    cell.PAD = 0.3
ax.set_title('Summary of Regression Results for D*F', fontsize=20,␣
 ↪fontweight='bold', color='navy', fontname='DejaVu Sans')
ax.title.set_position([0.5, 1.05])  # Decrease the distance between title and␣
 ↪table

plt.show()
```

Summary of Regression Results for D*F

Regression	Coefficient	Interpretation
Regression 4	-0.011	Decrease by 1.11%
Regression 6	0.0	No impact
Regression 8	2.5126	Increase by 251.26%
Regression 10	0.0	No impact
Regression 12	2.4669	Increase by 246.69%

1.16 Conclusion

While the effect of Uber on Public Transit is uncertain, most of our regressions where we employoed some type of variable selection (through either LASSO or Double-LASSO) yielded negative statistically significant coefficients for our treatment variable, 'treatUberX'. This indicated that when analyzing the variables with a high predictive power for the number of rides in an MSA, the presence of Uber in that MSA will likely have a negative effect on the number of rides in the MSA.

To reach a more definit conclusion with regard to the effect of Uber presence on the number of rides from an MSA, it would be useful to have additional covariates, such as the efficiency of the public transit in an MSA and the cleanliness of public transit vehicles in an MSA.