IEOR 142A: Introduction to Machine Learning and Data Analytics I, Fall 2024

Homework Assignment #1

Jay Zhang

Due September 25th, 2024 at 11:59pm Pacific time

IMPORTANT NOTE: As indicated in the slides of Lecture 1 (pgs. 36-38) and the syllabus (pgs. 6-8), please list any resources outside of the course materials that you find helpful in completing the assignment (e.g. peers you discuss with, materials from different classes, blog posts, AI Tools, etc.). Please also be mindful of all policies in the syllabus concerning academic integrity and the use of AI Tools, including that you need to write your own solutions individually.

Problem 1: (30 points)

Consider the simple linear regression model without an intercept:

$$Y = \beta X + \epsilon \tag{1}$$

Here there is a single feature $X \in \mathbb{R}$ and a corresponding coefficient $\beta \in \mathbb{R}$. Note that the above model does not include an intercept term, i.e., it assumes that the intercept is zero. Suppose that we have collected data $(x_1, y_1), \ldots, (x_n, y_n)$, where each x_i is an observed scalar value of the feature X and each y_i is a corresponding observed scalar value of the dependent variable Y.

Please answer the following:

a) (5 points) Following the principle of minimizing the RSS error (i.e., ordinary least squares), derive a closed form solution for $\hat{\beta}$, the estimator of β, in terms of the data that we have collected.

coefficient

model $\rightarrow$ $\underline{Y} = \beta \underline{X} + \varepsilon$

error term

dependent var

independent var

Residual sum of squares (RSS) :

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\hat{y}_i = \beta x_i \longrightarrow \text{so} \quad RSS = \sum_{i=1}^{n} (y_i - \beta x_i)^2$$

minimizing RSS:

steps: ① take derivative of RSS with respect to β

② set eq = 0

③ solve for β

① $$\frac{d}{d\beta} RSS = \frac{d}{d\beta} \sum_{i=1}^{n} (y_i - \beta x_i)^2$$

$$= 2 \sum_{i=1}^{n} (y_i - \beta x_i)(-x_i) = -2 \sum_{i=1}^{n} x_i (y_i - \beta x_i)$$

② set = 0 $\rightarrow$ $-2 \sum_{i=1}^{n} x_i (y_i - \beta x_i) = 0$

③ solve for β $\sum_{i=1}^{n} x_i y_i - \beta \sum_{i=1}^{n} x_i^2 = 0$

$$\sum_{i=1}^{n} x_i y_i = \beta \sum_{i=1}^{n} x_i^2$$

$$\boxed{\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}}$$

estimator of β

Assume now that the collected data $(x_1, y_1), \ldots, (x_n, y_n)$ satisfies:

$$y_i = \beta x_i + \epsilon_i \quad \text{for all } i = 1, \ldots, n \ ,$$

where $\beta \in \mathbb{R}$ is the true coefficient value and $\epsilon_1, \ldots, \epsilon_n$ are i.i.d. *normally distributed* random variables with mean zero and variance σ^2. Here we assume that the x_i values (as well as the parameters β and σ) are fixed and deterministic.

Under the above assumptions, please answer the following:

b) (2 points) Explain why $\hat{\beta}$ is a random variable.

c) (6 points) Derive formulas for the expected value and variance of $\hat{\beta}$.

d) (2 points) Argue that $\hat{\beta}$ is normally distributed.

b) given $y_i = \beta X_i + \varepsilon_i$

$$\hat{\beta} = \frac{\sum_{i=1}^{n} X_i y_i}{\sum_{i=1}^{n} X_i^2} \quad ; \text{ plug in } y_i$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n} X_i (\beta X_i + \varepsilon_i)}{\sum_{i=1}^{n} X_i^2}$$

b/c $\hat{\beta}$ involves random variable $\underline{\underline{\varepsilon_i}}$, which was said to be iid normal random vars

$\Rightarrow$ $\hat{\beta}$ <u>depends</u> on ε_i (random var), making it also a random var! ✓

c) $$\hat{\beta} = \frac{\sum_{i=1}^{n} X_i (\beta X_i + \varepsilon_i)}{\sum_{i=1}^{n} X_i^2} = \frac{\sum_{i=1}^{n} \beta X_i^2 + \sum_{i=1}^{n} X_i \varepsilon_i}{\sum_{i=1}^{n} X_i^2}$$

$$= \beta + \frac{\sum_{i=1}^{n} X_i \varepsilon_i}{\sum_{i=1}^{n} X_i^2}$$

expected val of $\hat{\beta}$

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}\left[\beta + \frac{\sum_{i=1}^{n} X_i \varepsilon_i}{\sum_{i=1}^{n} X_i^2}\right]$$

linearity of expectation

$\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$

constant

$$\longrightarrow \quad \mathbb{E}[\hat{\beta}] = \beta + \frac{1}{\sum_{i=1}^{n} X_i^2} \mathbb{E}\left[\sum_{i=1}^{n} X_i \varepsilon_i\right]$$

given : mean of $\varepsilon_i = 0 \rightarrow \mathbb{E}[\varepsilon_i] = 0$ for all i

$$\mathbb{E}\left[\sum_{i=1}^{n} X_i \varepsilon_i\right] = 0$$

thus $\mathbb{E}[\hat{\beta}] = \beta + 0 = \beta$

variance of $\hat{\beta}$:

b/c constant, can be ignored since var $= 0$

$var(cZ) = c^2\, var(Z)$

$$var(\hat{\beta}) = var\left(\beta + \frac{\sum_{i=1}^{n} X_i \varepsilon_i}{\sum_{i=1}^{n} X_i^2}\right) = \frac{1}{\left(\sum_{i=1}^{n} X_i^2\right)^2} \cdot \underbrace{var\left(\sum_{i=1}^{n} X_i \varepsilon_i\right)}$$

$\varepsilon_i \Rightarrow$ iid , var σ^2

$$var\left(\sum_{i=1}^{n} X_i \varepsilon_i\right) = \sum_{i=1}^{n} X_i^2\, var(\varepsilon_i)$$

$$= \left(\sum_{i=1}^{n} X_i^2\right) \cdot \sigma^2$$

thus $$var(\hat{\beta}) = \frac{\sigma^2}{\sum_{i=1}^{n} X_i^2}$$

d) argue that $\hat{\beta}$ is normally distributed.

mean $= 0$, var σ^2

normal random var, iid

⇓

b/c this holds, then the linear combo of it also is normally distr

$$\hat{\beta} = \beta + \frac{\sum_{i=1}^{n} X_i \epsilon_i}{\sum_{i=1}^{n} X_i^2}$$

$\sum_{i=1}^{n} X_i \epsilon_i \rightarrow$ linear combo of ϵ_i

then $\sum_{i=1}^{n} X_i \epsilon_i \Rightarrow$ normally distr with mean $= 0$

var $= \sigma^2 \sum_{i=1}^{n} X_i^2$

thus $\dfrac{\sum_{i=1}^{n} X_i \epsilon_i}{\sum_{i=1}^{n} X_i^2} \Rightarrow$ normally distr with mean $= 0$

var $= \dfrac{\sigma^2}{\sum_{i=1}^{n} X_i^2}$ ✓

Let x_{n+1} be a newly observed feature value, with associated value of the dependent variable y_{n+1}, satisfying:

$$y_{n+1} = \beta x_{n+1} + \epsilon_{n+1} \ ,$$

where ϵ_{n+1} is independent of $\epsilon_1, \ldots, \epsilon_n$ and follows the same distribution. Currently, we have observed x_{n+1} but have not yet observed y_{n+1}.

Please answer the following:

e) (5 points) Assuming that σ is known but β is not, describe how to construct a 95% *confidence interval* for βx_{n+1}. This is a random interval, constructed from the observed data and the value of σ, such that the probability that the interval contains βx_{n+1} is 95%. This probability is calculated over the randomness associated with the observed data $(x_1, y_1), \ldots, (x_n, y_n)$.

f) (5 points) Assuming that both σ and β are known, describe how to construct a 95% *prediction interval* for y_{n+1}. This is an interval such that the probability that y_{n+1} lies in the interval is 95%.

g) (5 points) Assume now that σ is known but β is not. Describe how to construct a 95% *prediction interval* for y_{n+1}. This is a random interval, constructed from the observed data and the value of σ, such that the probability that y_{n+1} lies in the interval is 95%. This probability is calculated over all randomness associated with $(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1})$.

NOTE: if you choose to use linear algebra techniques to address any part of this problem (which is not required), please simplify all answers in terms of summations instead of matrix/vector notation.

e) $\hat{\beta} = \dfrac{\sum_{i=1}^{n} X_i y_i}{\sum_{i=1}^{n} X_i^2}$ $\quad SE(\hat{\beta}) = \dfrac{\sigma}{\sqrt{\sum_{i=1}^{n} X_i^2}}$

σ known

β unknown

95% confidence interval for $\hat{\beta}_{n+1} \Rightarrow \left(\hat{\beta}_{n+1} - 1.96 \cdot \dfrac{\sigma X_{n+1}}{\sqrt{\sum_{i=1}^{n} X_i^2}} , \ \hat{\beta}_{n+1} + 1.96 \cdot \dfrac{\sigma X_{n+1}}{\sqrt{\sum_{i=1}^{n} X_i^2}} \right)$

f) σ known β known

from $\beta = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} \Rightarrow \hat{y}_{n+1} = \beta x_{n+1}$

sub in for βx_{n+1}

given $y_{n+1} = \beta x_{n+1} + \varepsilon_{n+1} \Rightarrow$ know also $= \hat{y}_{n+1} + \varepsilon_{n+1}$

prediction interval $\Rightarrow y_{n+1} = \hat{y}_{n+1} + \varepsilon_{n+1}$ where $\varepsilon_{n+1} \sim N(0, \sigma^2)$

for a 95% prediction interval : $\hat{y}_{n+1} \pm z_{\alpha/2} \cdot \sigma$

$= (\beta x_{n+1} - 1.96\,\sigma,\ \beta x_{n+1} + 1.96 \cdot \sigma)$

g) σ known β unknown

estimate $\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$ $\qquad \hat{y}_{n+1} = \hat{\beta} x_{n+1}$

when constructing a prediction interval for y_{n+1}, need to consider both ① uncertainty + fitted model and ② randomness of new data pt

↳ overall uncertainty = sum + 2 vars

$\Rightarrow Var(y_{n+1}) = Var(\hat{y}_{n+1}) + \sigma^2 = \frac{\sigma^2 x_{n+1}^2}{\sum_{i=1}^{n} x_i^2} + \sigma^2$

prediction interval = range of vals that contain y_{n+1} with 95% confidence

$$\hat{\beta} x_{n+1} \pm z_{\alpha/2} \cdot \sqrt{\frac{\sigma^2 x_{n+1}^2}{\sum_{i=1}^{n} x_i^2} + \sigma^2}$$

$$\left[\hat{\beta} x_{n+1} - 1.96 \sqrt{\frac{\sigma^2 x_{n+1}^2}{\sum_{i=1}^{n} x_i^2} + \sigma^2},\ \hat{\beta}_{n+1} + 1.96 \sqrt{\frac{\sigma^2 x_{n+1}^2}{\sum_{i=1}^{n} x_i^2} + \sigma^2} \right]$$

142AHW1

September 26, 2024

0.1 IEOR 142A: Introduction to Machine Learning and Data Analytics I, Fall 2024

0.2 Homework Assignment #1

0.2.1 Problem 2: Forecasting Honda Civic Sales

[103]:
```python
from pandas.plotting import scatter_matrix
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import math
import random
import statsmodels.api as sm
random.seed(30)
```

[104]:
```python
#load csv data
civic = pd.read_csv("civiccar.csv")
civic
```

[104]:
```
     MonthNumeric MonthFactor  Year  CivicSales  Unemployment  CivicQueries  \
0               1     January  2014       21824           6.6            66
1               2    February  2014       21575           6.7            69
2               3       March  2014       27697           6.7            72
3               4       April  2014       27611           6.2            69
4               5         May  2014       36089           6.3            69
..            …           …   …          …            …           …
122             3       March  2024        5664           3.8            87
123             4       April  2024        5348           3.9            83
124             5         May  2024        6700           4.0            88
125             6        June  2024        5935           4.1            92
126             7        July  2024        5755           4.3            92

      CPIAll  CPIEnergy  MilesTraveled  interest
0    235.288    250.340         246531       NaN
1    235.547    249.925         249499       NaN
2    236.028    249.961         251120       NaN
3    236.468    249.864         251959       NaN
```

```
4    236.918    249.213         252289        NaN
..       …         …             …        …
122  312.230    287.399         273352   8.877857
123  313.207    290.631         273430  17.200000
124  313.225    284.742         274175   8.877857
125  313.049    278.938         274160   8.877857
126  313.534    279.012         274273  17.080000

[127 rows x 10 columns]
```

a) (25 points) Start by splitting the data into a training set and a testing set. The training set should contain all observations from 2014 through 2019. The testing set should have all observations from January 2020 through July 2024

[105]:
```python
#splitting data into training and test set- filtering years
civic_train = civic[civic['Year'] <= 2019]
civic_test = civic[civic['Year'] >= 2020]
```

[106]:
```python
civic.info() #looking at different features
```

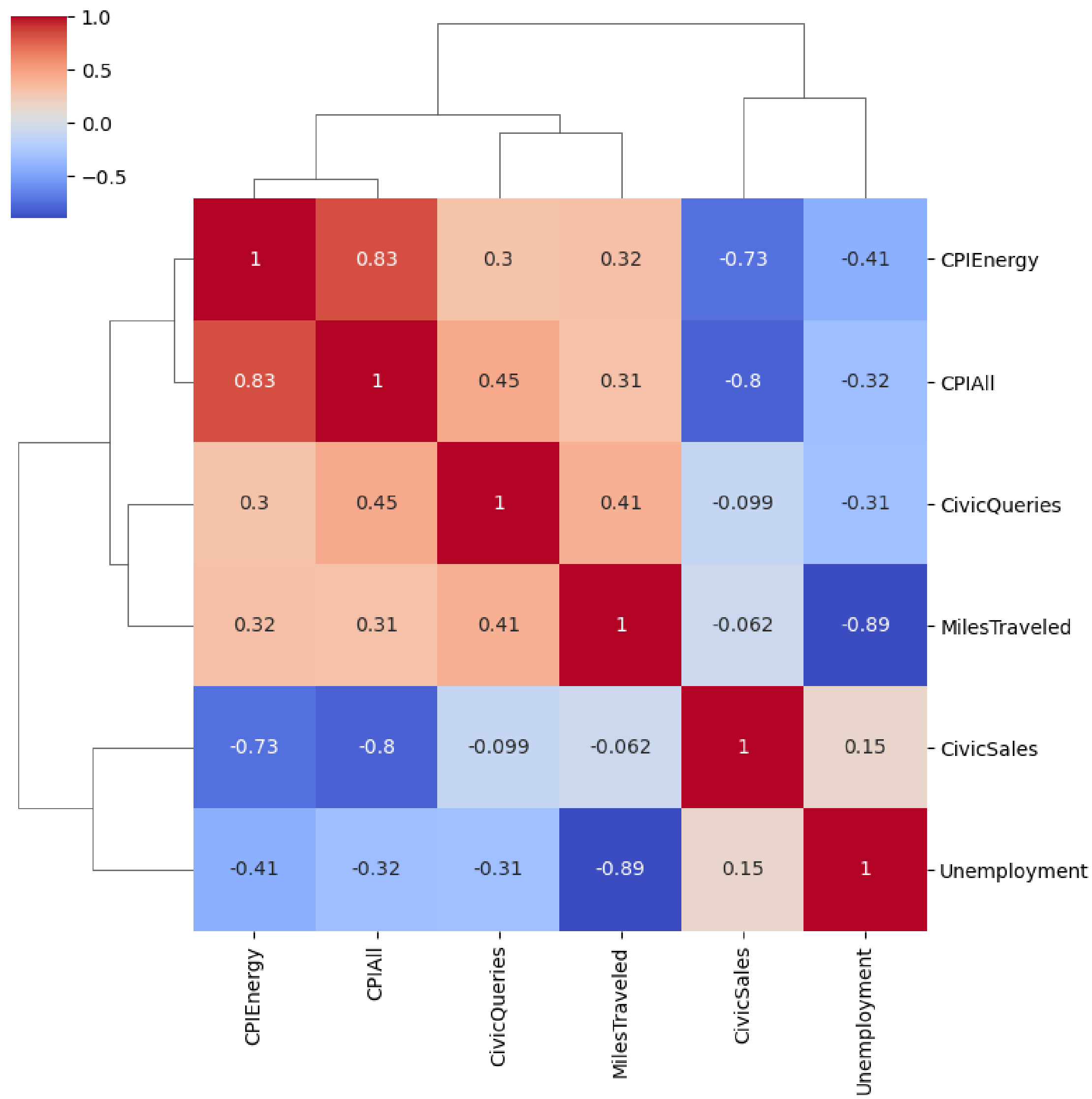
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 127 entries, 0 to 126
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MonthNumeric   127 non-null    int64
 1   MonthFactor    127 non-null    object
 2   Year           127 non-null    int64
 3   CivicSales     127 non-null    int64
 4   Unemployment   127 non-null    float64
 5   CivicQueries   127 non-null    int64
 6   CPIAll         127 non-null    float64
 7   CPIEnergy      127 non-null    float64
 8   MilesTraveled  127 non-null    int64
 9   interest       116 non-null    float64
dtypes: float64(4), int64(5), object(1)
memory usage: 10.1+ KB
```

[107]:
```python
features = ['CivicSales',
            'Unemployment',
            'CivicQueries',
            'CPIEnergy',
            'CPIAll',
            'MilesTraveled']
```

[108]:
```python
corr = civic[features].corr()
```

```python
# Plot clustermap to showcase correlation
sns.clustermap(corr, annot=True, cmap='coolwarm', figsize=(8, 8))
```

[108]: <seaborn.matrix.ClusterGrid at 0x7f420ec4a690>



Model 1: Regular OLS

i) Feature Selections: Through the displayed diagram, we can finalize our selection of features correlating best to Civic Sales:

- CPIAll: CPI-All measures inflation by tracking price changes for a basket of goods and services commonly purchased by urban consumers, including car sales It reflects cost-of-living changes and is a key indicator used to assess inflation in urban areas.

- CivicQueries: CivicQueries is included because search frequency and behavior are often reasonable indicators in gauging the interest of the population in specific products/services, which can be beneficial for our case in understanding Car Sales.
- MilesTraveled: MilesTraveled displays a potential increase in wants and desires. If more people are driving/traveling, this could lead to overuse of their cars, which could lead to them needing to buy a new one because of tears or malfunction issues.
- Unemployment: Unemployment was selected because economic fluctuations are strong indicators of consumer demand. With overburdening economic stress and without a surplus of money, this could lead to shifts in consuming behaviors/patterns– the overall population may not choose to spend their money on purchasing vehicles OR choosing cheaper alternatives.
- CPIEnergy was excluded from my final model because it is a subset of CPIAll and I did not believe it was relevant to include both. This variable could even be capable of creating collinearity issues with CPIAll, which could negatively impact our model. While comparing the accuracy of the model, it also just seemed overall better after we excluded CPIEnergy and kept the rest of the variables that seemed more linearly independent of one another.

ii) Our Model: Using OLS with these Civic Sales as our dependent variable and the 4 independent variables, we have the following equation:

$$Y = -33,180 - 149.39 \cdot X_{CPIA} + 366.19 \cdot X_{CQ} + 0.220 \cdot X_{MT} + 2535.01 \cdot X_u$$

iii) Coefficient Interpretation:

- Intercept -33,180: When all the other variables are 0, this would be the value we obtain in sales. Because the intercept is negative, it does not make logical/statistical sense on its own, as you cannot really incur negative sales. One reason for this could be due to the relative collinearity in the chosen features.
- CPIAll -149.39: A 1-unit increase in the overall CPIAll is associated with a decrease in Civic sales by 149 units. A rise in the CPIAII can lead to a decrease in Honda Civic sales by impacting consumer confidence, increasing the overall cost of vehicle ownership, and making financing less accessible.
- CivicQueries 366.19: A 1-unit increase in Civic-related online searches is associated with a 366 Civic sale increase, indicating a pretty positive/strong relationship between car sales and internet search queries since people likely do a lot of research on cars before they decide to purchase. This is logically reasonable, as we do see the skyrocket of online traffic correlating to real-world interactions and surges in purchases.
- MilesTraveled 0.22: For every 1-unit increase in miles traveled, Civic Sales are predicted to increase by 0.22 units. Compared to the other variables, this change is statistically insignificant, indicating that it likely has a minimal impact on Civic Sales overall.
- Unemployment 2,535.01: For every 1% increase in unemployment, Civic Sales are predicted to increase by 2,535 units. Surprised that the coefficient was actually positive, this could imply that during times when there is an increase in unemployment, people might opt to buy cheaper alternatives for cars, like the Honda Civic compared to expensive/luxury cars. If the

coefficient was negative, that would follow with my original thought, where people in general would be less inclined to buy cars in general, leading to a decrease in sales.

[109]:
```
model1_features = ['CPIAll','CivicQueries',
                   'MilesTraveled','Unemployment']

X_train_1 = civic_train[model1_features]
X_train_1 = sm.add_constant(X_train_1)
y_train_1 = civic_train['CivicSales']
```

[110]:
```
model1 = sm.OLS(y_train_1,X_train_1).fit()
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             CivicSales   R-squared:                       0.412
Model:                            OLS   Adj. R-squared:                  0.377
Method:                 Least Squares   F-statistic:                     11.75
Date:                Thu, 26 Sep 2024   Prob (F-statistic):           2.75e-07
Time:                        01:37:04   Log-Likelihood:                -688.83
No. Observations:                  72   AIC:                             1388.
Df Residuals:                      67   BIC:                             1399.
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
=
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const        -3.318e+04   6.61e+04     -0.502      0.617   -1.65e+05
9.87e+04
CPIAll        -149.3855    141.679     -1.054      0.295    -432.178
133.407
CivicQueries   366.1866     63.883      5.732      0.000     238.675
493.698
MilesTraveled    0.2202      0.187      1.178      0.243      -0.153
0.594
Unemployment  2535.0126   1844.274      1.375      0.174   -1146.173
6216.198
==============================================================================
Omnibus:                        1.861   Durbin-Watson:                   1.576
Prob(Omnibus):                  0.394   Jarque-Bera (JB):                1.427
Skew:                           0.142   Prob(JB):                        0.490
Kurtosis:                       2.372   Cond. No.                     4.13e+07
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.13e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

iv) Results and Observations: Our model's **R-squared** value is **0.412**, indicating that the model explains approximately **41.2%** of the variance seen within our Civic Sales. Thus we were able to capture around half of the factors influencing sales.

- The **F-statistic** of **11.75** with a p-value of **2.75e-07** shows that the model is statistically significant overall, meaning a statistically meaningful relationship was created with at least one of the independent variable features. The statistic overall tells us whether the predictors in our model are useful for predicting the outcome of Civic Sales.
- The **Adjusted R-squared** of **0.377** tells us how well our model fits the data and accounts for the number of predictors we have included.

Final Conclusion: While our model is pretty reasonable in its predictions, we could have performed better if we were not limited to the select features given to us that could be able to encompass more of what influences Civic Sales.

Model 2: Seasonality

i) Our Model: $CivicSales = *0 +* 1 X + 2 X*{CQ} +* 3 X*{MT} +* 4 X*{CPIE} +* 5 X*{CPIA} +* {i=1}^{11} *{i+5} X*{MFi} $

Coefficient Interpretation: No Variable Selection Occurred in this problem.

- Intercept
- Unemployment
- CivicQueries
- MilesTraveled
- CPIEnergy
- CPIAll
- Month Factor: **new** independent variable in addition to all five of the variables we used at the start of Part (a). Seasonality is important in predicting demand and sales since demand for many products tends to be periodic in time. In regards to Civic Sales, people may buy cars more during winter since there are more discounts that occur during Black Friday or Christmas to incentivize consumers. Also, people may just be in a more spending habit/spirit due to festivities. Here, each dummy variable of Month Factor accounts for every month (Jan to Dec) in a year.

ii) iii) Results and Observations

[111]:
```
model2_features =␣
↪['Unemployment','CivicQueries','CPIEnergy','CPIAll','MilesTraveled']
```

```python
# One-hot encode 'MonthFactor' variable
X_train_months = pd.get_dummies(civic_train['MonthFactor'], drop_first=True)
X_train_2 = pd.concat([civic_train[model2_features], X_train_months], axis=1)

#boolean to integer
X_train_2 = X_train_2.astype(int)

#merge with original features
X_train_2 = sm.add_constant(X_train_2)
y_train_2 = civic_train['CivicSales']
```

[112]:
```python
model2 = sm.OLS(y_train_2, X_train_2).fit()
print(model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             CivicSales   R-squared:                       0.806
Model:                            OLS   Adj. R-squared:                  0.749
Method:                 Least Squares   F-statistic:                     14.25
Date:                Thu, 26 Sep 2024   Prob (F-statistic):           3.86e-14
Time:                        01:37:04   Log-Likelihood:                -649.00
No. Observations:                  72   AIC:                             1332.
Df Residuals:                      55   BIC:                             1371.
Df Model:                          16
Covariance Type:            nonrobust
================================================================================
=
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const          -5.08e+04   3.11e+04     -1.632      0.108   -1.13e+05
1.16e+04
Unemployment   2942.7866    871.126      3.378      0.001    1197.010
4688.563
CivicQueries    258.0527     65.154      3.961      0.000     127.480
388.625
CPIEnergy        -8.0006     30.288     -0.264      0.793     -68.700
52.699
CPIAll         -117.6125    152.561     -0.771      0.444    -423.352
188.127
MilesTraveled     0.2922      0.148      1.977      0.053      -0.004
0.589
August         3417.9077   1377.606      2.481      0.016     657.124
6178.692
December       2446.2169   1574.916      1.553      0.126    -709.985
```

```
5602.418
February      -3508.0232   1393.483     -2.517      0.015   -6300.626
-715.420
January       -5124.7768   1374.966     -3.727      0.000   -7880.270
-2369.283
July           1502.8602   1387.625      1.083      0.284   -1278.002
4283.723
June           1167.4982   1400.212      0.834      0.408   -1638.589
3973.585
March          1267.1779   1383.511      0.916      0.364   -1505.441
4039.797
May            5269.6578   1395.259      3.777      0.000    2473.495
8065.820
November       -940.8309   1464.543     -0.642      0.523   -3875.841
1994.179
October       -1412.3330   1382.125     -1.022      0.311   -4182.174
1357.508
September      -689.1123   1368.468     -0.504      0.617   -3431.584
2053.359
==============================================================================
Omnibus:                       10.476   Durbin-Watson:                   1.500
Prob(Omnibus):                  0.005   Jarque-Bera (JB):               12.013
Skew:                           0.680   Prob(JB):                      0.00246
Kurtosis:                       4.468   Cond. No.                     3.07e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.07e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Our model's **R-squared** value is **0.806**, indicating that the model explains approximately **80.6%** of the variance seen within our Civic Sales. Comparing this to our first simple model, we were able to capture a lot more variance around the factors influencing our Civic Car Sales.

Due to the big jump in improvement from part A, we can attribute a lot of this to our new variable **Month Factor**. It seems like seasonality truly has an impact on the performance of our Car Sales. Based on the chart we have generated, we see an impact of increases in Car Sales during the months of August, December, and May, which could be explained as starting school/new job, during holidays/festivities/gift giving/discount season, and end-of-school-year celebrations. Looking back to our previous variables, we see that Unemployment, followed by CivicQueries, and CPIALL has stronger coefficients that impact our model, thereby impacting Civic Sales.

Final Conclusion: Month Factor was a good choice in adding to our model since it seemed to encompass/capture our variance a lot better than our first model was able to do with just the 4 features. Our higher R-squared value shows an overall increase in assessing the explanatory power of our regression model.

iv) Another Way to Model Seasonality We could potentially look into other models that are known to capture seasonality well. One example that is commonly used is **the Seasonal Autoregressive Integrated Moving Average (SARIMA)**, a powerful and versatile model for time series forecasting that effectively incorporates seasonal patterns while providing a solid framework for understanding and predicting data trends. If the data exhibits strong seasonal patterns, a model specifically designed to capture seasonality like SARIMA might outperform a more general model like the one we had. From what we observed in the previous model implementing Month Factor, it does seem like seasonality is strongly correlated to our model, thus I do believe using a model like SARIMA would be able to improve our performance.

Model 3: Mixed Model

[113]:
```
model3_features = ['Unemployment','CivicQueries','CPIAll','MilesTraveled']



# One-hot encode 'MonthFactor' and merge
X_train_with_month = pd.get_dummies(civic_train['MonthFactor'], drop_first=True)
X_train_3 = pd.concat([civic_train[model3_features], X_train_with_month],␣
↪axis=1)

X_train_3 = X_train_3.astype(int) #bool to int

X_train_3 = sm.add_constant(X_train_3)
y_train_3 = civic_train['CivicSales']
```

[114]:
```
model3 = sm.OLS(y_train_3, X_train_3).fit()
print(model3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             CivicSales   R-squared:                       0.805
Model:                            OLS   Adj. R-squared:                  0.753
Method:                 Least Squares   F-statistic:                     15.45
Date:                Thu, 26 Sep 2024   Prob (F-statistic):           9.89e-15
Time:                        01:37:04   Log-Likelihood:                -649.04
No. Observations:                  72   AIC:                             1330.
Df Residuals:                      56   BIC:                             1367.
Df Model:                          15
Covariance Type:            nonrobust
================================================================================
=
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const         -5.024e+04   3.08e+04     -1.631      0.109   -1.12e+05
1.15e+04
Unemployment   2845.4641    782.791      3.635      0.001    1277.345
```

```
4413.583
CivicQueries    260.5953     63.902      4.078      0.000     132.584
388.607
CPIAll         -150.2400     88.795     -1.692      0.096    -328.117
27.637
MilesTraveled     0.3145      0.120      2.611      0.012       0.073
0.556
August         3436.6776   1364.298      2.519      0.015     703.661
6169.694
December       2485.6162   1554.760      1.599      0.116    -628.943
5600.175
February      -3450.5977   1364.940     -2.528      0.014   -6184.902
-716.293
January       -5070.8322   1348.374     -3.761      0.000   -7771.951
-2369.714
July           1517.3142   1374.981      1.104      0.275   -1237.104
4271.733
June           1207.2495   1380.490      0.875      0.386   -1558.205
3972.704
March          1321.3146   1356.835      0.974      0.334   -1396.753
4039.382
May            5279.3923   1383.140      3.817      0.000    2508.630
8050.154
November       -863.6963   1423.166     -0.607      0.546   -3714.641
1987.248
October       -1384.5645   1366.627     -1.013      0.315   -4122.248
1353.119
September      -638.6332   1343.757     -0.475      0.636   -3330.503
2053.236
==============================================================================
Omnibus:                       10.047   Durbin-Watson:                   1.495
Prob(Omnibus):                  0.007   Jarque-Bera (JB):               11.366
Skew:                           0.659   Prob(JB):                      0.00340
Kurtosis:                       4.432   Cond. No.                     3.06e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.06e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Results and Observations:

Our model's **R-squared** value is **0.805**, indicating that the model explains approximately **80.5%** of the variance seen within our Civic Sales. Comparing to the first 2 models we created, there was a very slight decrease in the value compared to our 2nd model.

Comparing to the **OSR Squared** value below (0.5105) to our R-squared value (0.805), we can see that the OSR Squared value is smaller. When R-squared is greater than OSR Squared, that usually implies overfitting (aka the model fits the training data better than it generalizes to new data). We could infer that our model might be a little too complex and capture noise rather than the underlying pattern. To mitigate this, we might consider simplifying the model or using regularization techniques to avoid overfitting.

[115]:
```
#Obtained function from AI as I could not find the one from class that works␣
↪with code
def calculate_osr2_from_ols(r_squared, n, df_model, df_residual):

    # Calculate total sum of squares (SS_total)
    ss_total = (n - 1)  # Using a normalized approach (this assumes the␣
↪variance of Y is 1)

    # Calculate SS_regression and SS_residual
    ss_regression = r_squared * ss_total
    ss_residual = ss_total - ss_regression

    # Calculate OSR²
    osr2 = (ss_regression - ss_residual) / (ss_total + ss_residual)

    return osr2

#from our regression results
df_model = 15
df_residual = 56
r_squared = 0.805
n = 72

osr2_value = calculate_osr2_from_ols(r_squared, n, df_model, df_residual)
print(f"OSR² Value: {osr2_value:.4f}")
```

```
OSR² Value: 0.5105
```

Additional Variable to Implement Another variable that could be related to Honda sales is **monthly interest rates on auto loans**. When interest rates are low, consumers are more likely to finance car purchases, which could lead to higher sales of Honda vehicles. Auto loan rates are a key factor in consumer decision-making when purchasing cars, so adding this variable to our regression model could provide insights into whether lower rates are driving higher Honda sales.

[116]:
```
#load csv data
new = pd.read_csv("newfeature.csv")
new
```

[116]:

	DATE	TERMCBAUTO48NS
0	1972-02-01	10.20
1	1972-03-01	.
2	1972-04-01	.

```
3    1972-05-01          9.96
4    1972-06-01             .
..          …           …
623  2024-01-01             .
624  2024-02-01          8.57
625  2024-03-01             .
626  2024-04-01             .
627  2024-05-01          8.65

[628 rows x 2 columns]
```

[117]:

```
#splitting data into training and test set- filtering years

# Convert the 'date' column to datetime
new['DATE'] = pd.to_datetime(new['DATE'])
# Extract the year and create a new column for it
new['year'] = new['DATE'].dt.year

new.replace('.', pd.NA, inplace=True)

#Convert the column to a numeric type (this will handle non-numeric values like␣
↪'.')
new['TERMCBAUTO48NS'] = pd.to_numeric(new['TERMCBAUTO48NS'], errors='coerce')

#Replace NaN values with the mean of the column
new['TERMCBAUTO48NS'].fillna(new['TERMCBAUTO48NS'].mean(), inplace=True)

new_filter= new[new['year'] > 1972]
```

```
/tmp/ipykernel_357/2309341815.py:14: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  new['TERMCBAUTO48NS'].fillna(new['TERMCBAUTO48NS'].mean(), inplace=True)
```

[118]:

```
# Step 3: Select the specific column from the first DataFrame
# Replace 'column_name' with the actual name of the column you want to import
interest = new['TERMCBAUTO48NS']
```

```
# Step 4: Add the selected column to the second DataFrame
# Ensure the lengths match, or consider how to handle differing lengths
civic['interest'] = interest  # Replace 'new_column_name' with your desired␣
 ↪column name

# Step 5: Save the updated DataFrame back to a CSV file
civic.to_csv('civiccar.csv', index=False)

print(civic)
```

```
     MonthNumeric MonthFactor  Year  CivicSales  Unemployment  CivicQueries  \
0               1     January  2014       21824           6.6            66
1               2    February  2014       21575           6.7            69
2               3       March  2014       27697           6.7            72
3               4       April  2014       27611           6.2            69
4               5         May  2014       36089           6.3            69
..            …         …   …         …           …           …
122             3       March  2024        5664           3.8            87
123             4       April  2024        5348           3.9            83
124             5         May  2024        6700           4.0            88
125             6        June  2024        5935           4.1            92
126             7        July  2024        5755           4.3            92

      CPIAll  CPIEnergy  MilesTraveled   interest
0    235.288    250.340         246531  10.200000
1    235.547    249.925         249499   8.877857
2    236.028    249.961         251120   8.877857
3    236.468    249.864         251959   9.960000
4    236.918    249.213         252289   8.877857
..       …        …            …        …
122  312.230    287.399         273352   8.877857
123  313.207    290.631         273430  17.200000
124  313.225    284.742         274175   8.877857
125  313.049    278.938         274160   8.877857
126  313.534    279.012         274273  17.080000

[127 rows x 10 columns]
```

[119]:

```
#splitting data into training and test set- filtering years

civic['interest'].fillna(civic['interest'].mean(), inplace=True)
civic_train = civic[civic['Year'] <= 2019]
civic_test = civic[civic['Year'] >= 2020]
model4_features =␣
 ↪['Unemployment','CivicQueries','CPIAll','MilesTraveled','interest']

X_train_with_month = pd.get_dummies(civic_train['MonthFactor'], drop_first=True)
```

```python
X_train_4 = pd.concat([civic_train[model4_features], X_train_with_month],␣
 ↪axis=1)

X_train_4 = X_train_4.astype(int) #bool to int


X_train_4 = sm.add_constant(X_train_4)
y_train_4 = civic_train['CivicSales']


model4 = sm.OLS(y_train_4, X_train_4).fit()
print(model4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             CivicSales   R-squared:                       0.812
Model:                            OLS   Adj. R-squared:                  0.757
Method:                 Least Squares   F-statistic:                     14.83
Date:                Thu, 26 Sep 2024   Prob (F-statistic):           1.68e-14
Time:                        01:37:04   Log-Likelihood:                -647.84
No. Observations:                  72   AIC:                             1330.
Df Residuals:                      55   BIC:                             1368.
Df Model:                          16
Covariance Type:            nonrobust
===============================================================================
=
                   coef    std err          t      P>|t|      [0.025
0.975]
-------------------------------------------------------------------------------
-
const         -5.012e+04   3.06e+04     -1.639      0.107   -1.11e+05
1.11e+04
Unemployment   3011.4380    786.184      3.830      0.000    1435.889
4586.987
CivicQueries    253.1797     63.642      3.978      0.000     125.639
380.720
CPIAll         -179.1112     90.603     -1.977      0.053    -360.685
2.462
MilesTraveled     0.3915      0.132      2.963      0.004       0.127
0.656
interest      -1346.4870    984.179     -1.368      0.177   -3318.827
625.853
August          680.7725   2427.017      0.280      0.780   -4183.078
5544.622
December       -531.7662   2691.534     -0.198      0.844   -5925.720
4862.187
February      -6139.6753   2386.993     -2.572      0.013   -1.09e+04
```

```
-1356.034
January       -4482.1482   1405.485     -3.189      0.002   -7298.803
-1665.494
July           2143.4046   1439.102      1.489      0.142    -740.619
5027.429
June          -1495.5478   2404.016     -0.622      0.536   -6313.304
3322.209
March         -1390.2979   2396.046     -0.580      0.564   -6192.081
3411.485
May            2579.6164   2403.702      1.073      0.288   -2237.510
7396.743
November      -3629.1572   2465.801     -1.472      0.147   -8570.733
1312.419
October        -822.9414   1416.882     -0.581      0.564   -3662.436
2016.553
September     -3427.1101   2435.594     -1.407      0.165   -8308.150
1453.929
==============================================================================
Omnibus:                       11.113   Durbin-Watson:                   1.527
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               12.869
Skew:                           0.716   Prob(JB):                      0.00161
Kurtosis:                       4.496   Cond. No.                     3.06e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.06e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

/tmp/ipykernel_357/1113378029.py:3: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  civic['interest'].fillna(civic['interest'].mean(), inplace=True)
```

Results and Observations Based on the table above, our R-squared value is **0.807**, which is actually slightly better than all our above models from the previous questions (with model 2 being 0.806 and model 3 being 0.805). From the new variable we added named 'interest' (standing for the monthly interest rates on auto loans), we can observe that a 1% increase leads to a -327.098

decrease in Civic Sales. This is in line with the original thought I had when choosing this variable. With interest rates increasing, people who cannot afford cars are less likely to borrow money to buy cars since interest rates are higher.

Just an elaboration, but the dataset I had imported had some missing data for some months, but it did basically reflect all the years our original dataset had. To mitigate this, I took the mean of the interest rates and instead replaced the null values to get a better overall estimation of what the data would look like and for it to be integrable within our previous data's CSV.

sourcing: https://fred.stlouisfed.org/categories/33058 https://fred.stlouisfed.org/series/TERMCBAUTO48NS

[ ]:
```

```