Deep Learning of Real-time Revenue and Employment in COVID with Data Mining and Machine Learning

Introduction

Covid-19 impact on many businesses has been sudden and severe with forcing to close down. A total of 19.6 total nonfarm U.S jobs have been lost which is twice the total job loss during great recession and an estimation of one quarter in labor force has been disrupted in first three months of lockdown. In the month of May, employment in public sector is increased by 3 million but small business revenue is more affected with high financial risk.

This project uses data from multiple sources and gives mechanisms through which Covid impacts our Economy from changes in consumer activity to business revenue losses to employment.

Throughout the project, it follows real life data science project cycle - Data Analytics life cycle (DALC) with phases like discovering, data preparation, model planning, model building, data visualization, interpreting results, operationalize in iterative manner for better results. Data Preparation includes acquisition, exploratory data analysis with visualization for preprocessing. Selecting models for the data by choose one or more analytical techniques/approaches and then building the model with training and testing data to evaluate the model performance. Interpreting model results using data visualizations to rebuild the model if necessary, to obtain desired results and then interpreting the final data reports or results and then the model/application can be useful for further analytical part or in business operations. Each phase focuses on analytical methods and data mining techniques required to achieve data- driven solutions for better decision-making in an organization as a data Scientist.

Statement of the project problem

This project is studied to understand the total impact of revenue on public sector and employment to mitigate the crisis with deep learning techniques using data mining approaches and machine learning. This study is aimed to,

1. Learn the COVID-19 impacts on private sector revenue in real-time - analyze the total small business revenue from the expenses spent in gradual timely manner across accommodation and food service, Merchant category, arts etc., in several Covid effected areas at national level to understand and make better decisions for new policies across industries.
2. Nonfarm US Employment flow estimation for next month across various industries - estimate the flow of employment in nonfarm public sector with deep insights from job posts across various industries at national level for better employment knowledge helpful for public in recession times.
3. Does lifting up lockdown and public mobility have impact on COVID infection cases? - analyze the nation level google mobility data of people to determine the number of covid infection cases, death records to discover the new patterns from peaks of covid cases helpful for business gains and public welfare.

Review of Literature

The tracker paperwork from Raj, John, Nathaniel, Michael and Opportunity Insights team (2020), the consumer spending from Affinity Solutions, Wombly' business revenues and Burning glass job posts have been explained on limited dataset from these above-mentioned applications. This study includes additional research questions with larger dataset from multiple sources to find accurate answers considering the current and previous occupational Employment

statistics across several industries details from U.S Bureau of Labor Statistics that include nationwide records.

Another excellent study from Sara (2020), provides with detailed reports of Great Recession period and recovery impacts focusing on California's public sector jobs and local education. With available larger dataset, this study provides detailed reports of COVID impacts on Nationwide level public sector jobs and revenue.

From a theory paper called 'Vector Autoregressive models for Multivariate series', has given the introduction to VAR and VECM with in depth explanation of the models for multivariate time series estimations.

Another theory paper, 'Autoregression and vector error-correction models' has provided a clear understanding on when to choose VAR or VECM model with proper implementation of causality matrix and cointegration test.

Excellent theoretical study paper on 'Cointegration' by Bent (2019) has provided a clear understanding of cointegration and its important role in model fitting and predicting for time series analysis.

Also, another IMF working paper on theoretical study of 'Testing for cointegration using johansen methodology when variables are near integrated' by Erik. & Par (2007), has provided deep knowledge on properties and working of of Johansen's max eigen values and traces test for cointegration.

A theoretical study paper on 'Vector error correction model' providing a clear understanding of VECM model works in theory.

An article by Aishwarya (2018), given description in handling non-stationary time series with clear explanation in types of non - stationarities and methods to convert non-stationary times series into stationary

Objectives of the study

This project is studied with objectives to find the solutions to mitigate the COVID crisis in Private sector, the three main objectives are:

1. to analyze the total small business revenue from the expenses spent in gradual timely manner across accommodation and food service, Merchant category, arts etc., across national level along with open business percentage to understand and make better decisions for new policies across industries.
2. to estimate the flow of employment with deep insights from private payroll apps across various industries at national level for better employment knowledge helpful for public in recession times.
3. to analyze the nation level google mobility data of people and percent of covid infection cases, death records to discover the new patterns from peaks of covid cases helpful for business gains and public welfare.

Data Collection

The Datasets of in this study are taken from the sources Kaggle. Dataset taken from Kaggle, is created and updated by Opportunity Insight's Economic Tracker in real-time having least lag possible in current data and available data in dataset. These datasets include separate dataset for customer spending across various businesses like accommodation, food service, arts, entertainment, recreation, general merchandize and etc., Revenue from small business with opening businesses percentage datasets across nationwide, public mobility information datasets

in each geo location, COVID test cases data, infection cases data, deaths data across nation. Employment levels datasets from a private apps in period from January 2020. These datasets are of size 15-32 KB each.

These time series datasets are of multiple .csv files with various attributes which are collected and combined as required for each learning objective using manual feature selection method and then processed to clean the missing values.

Time series dataset with daily granularity from 01-13-2020 to 11-15-2020 to analyze the small business revenues is collected as below

Time series Variable	**Definition**
Spend_all	Seasonally adjusted credit/debit spending in all merchant category codes smoothed to 7 day moving average
Spend_apg	Seasonally adjusted credit/debit spending in general merchandise smoothed to 7 day moving average
Spend_aer	Seasonally adjusted credit/debit spending in arts, entertainement & recreation smoothed to 7 day moving average
Spend_grf	Seasonally adjusted credit/debit spending in grocery and food smoothed to 7 day moving average
Spend_hcs	Seasonally adjusted credit/debit spending in health care and social assistance smoothed to 7 day moving average
Spend_acf	Seasonally adjusted credit/debit spending in accommodation &food services smoothed to 7 day moving average

Spend_tws	Seasonally adjusted credit/debit spending in transport & warehousing smoothed to 7 day moving average
Spend_retail_no_grocery	Seasonally adjusted credit/debit spending in retail with no grocery smoothed to 7 day moving average
Spend_retail_w_grocery	Seasonally adjusted credit/debit spending in retail with grocery smoothed to 7 day moving average
Merchants_all	Seasonally adjusted and smoothed to 7 day moving average of all small business open percent change
Merchants_ss40	Seasonally adjusted and smoothed to 7 day moving average of small business open percent change in transportation
Merchants_ss60	Seasonally adjusted and smoothed to 7 day moving average of small business open percent change in professional and business services
Merchants_ss65	Seasonally adjusted and smoothed to 7 day moving average of small business open percent change in education and health services
Merchants_ss70	Seasonally adjusted and smoothed to 7 day moving average of small business open percent change in leisure and hospitality
Revenue_all	Seasonally adjusted and smoothed to 7 day moving average of all small business net revenue percent change
Revenue_ss40	Seasonally adjusted and smoothed to 7 day moving average of small business net revenue percent change in transportation

Revenue_ss60	Seasonally adjusted and smoothed to 7 day moving average of small business net revenue percent change in professional and business services
Revenue_ss65	Seasonally adjusted and smoothed to 7 day moving average of small business net revenue percent change in education and health services
Revenue_ss70	Seasonally adjusted and smoothed to 7 day moving average of small business net revenue percent change in leisure and hopitality

Time series dataset with daily granularity from 01-27-2020 to 11-5-2020 to analyze the employment levels with covid-19 cases is collected as below

Time series Variable	**Definition**
Emp_combined	Seasonally adjusted credit/debit and smoothed to 7 day moving average of employment levels for all workers
Emp_combined_ss40	Seasonally adjusted credit/debit and smoothed to 7 day moving average of employment levels for workers in trade, transport and utilities
Emp_combined_ss60	Seasonally adjusted credit/debit and smoothed to 7 day moving average of employment levels for workers in professional and business services

Emp_combined_ss65	Seasonally adjusted credit/debit and smoothed to 7 day moving average of employment levels for workers in education and health services
Emp_combined_ss70	Seasonally adjusted credit/debit and smoothed to 7 day moving average of employment levels for workers in leisure and hospilatity
case_rate	Confirmed covid-19 cases per 1,00,000 people with 7 day moving average
New_case_rate	New confirmed covid-19 cases per 1,00,000 people with 7 day moving average

Time series dataset with daily granularity from 01-27-2020 to 11-5-2020 to analyze the google mobility of people across and covid-19 cases is collected as below

Time series Variable	**Definition**
Gps_retail_and_recreation	Seasonally adjusted credit/debit and smoothed to 7 day moving average of time spent in retail and recreation
Gps_grocery_and_pharmacy	Seasonally adjusted credit/debit and smoothed to 7 day moving average of time spent in grocery and pharmacy
Gps_parks	Seasonally adjusted credit/debit and smoothed to 7 day moving average of time spent in parks
Gps_transit_stations	Seasonally adjusted credit/debit and smoothed to 7 day moving average of time spent in transit stations

Gps_workplaces	Seasonally adjusted credit/debit and smoothed to 7 day moving average of time spent in workplaces
Gps_residential	time spent in residential locations
Gps_away_from_home	Time spent outside of residential locations
case_rate	Confirmed covid-19 cases per 1,00,000 people with 7 day moving average
New_case_rate	New confirmed covid-19 cases per 1,00,000 people with 7 day moving average

Data Preparation and Cleaning

As all the data sources taken for any kind of analytics can have potential errors or missing values, data cleaning must be done before processing it to avoid anomalies. Data cleaning addresses many issues like miss specifications of the model, false conclusions from incorrect analysis like including errors in parameter estimation, etc.,.

Time series data is special type of data that includes hidden properties like trend, seasonality, cyclical and irregularity in an chronological order. Cleaning such datasets is slightly more complicated and needs more attention.

Time series data is a list of numbers or sequence of numbers taken at equally spaced consecutive points of time. This time series data consists of four elements - i) Trend (T) : Trend defines a long term pattern in the series which can either be positive or negative depending upon increasing or decreasing long term pattern, ii) Seasonality (S) : In a time series, seasonality occurs when it shows regular fluctuations during period or month or quarter in a year, iii) Cyclical (C) : Any sinusoidal or cos-sinusoidal patterns observed over a trend in the series are defined as cyclical

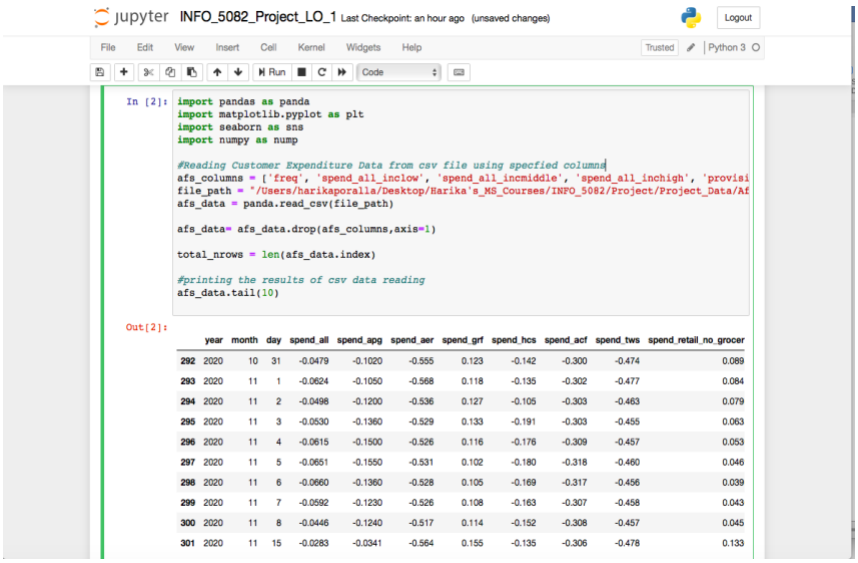
pattern and iv) Irregular (I) : Every time series composes of an unpredictable component called as irregular component which is a random variable and during time series analysis every component except random component is modelled to a point.

In this multivariate time series analysis, three steps are followed for data preparation and processing that included a) understanding the data, b) EDA – Inspection, data profiling, visualizations and c) Data cleaning – missing data, outlier detection and rectifying outliers.

For collecting the required time series variables from multiple datasets for each learning objective as shown in above, first each time series dataset is collected, understood and cleaned the missing values as shown below.

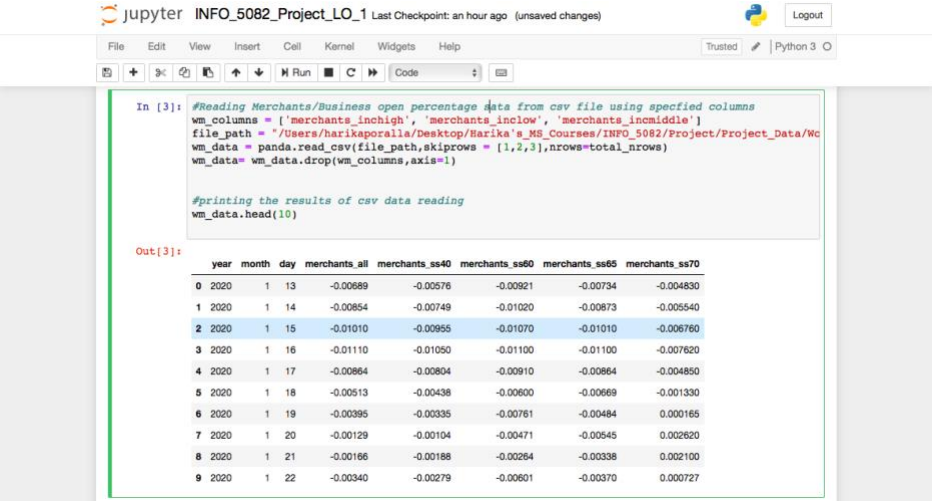
For the first learning objective i.e., to analyze the small business revenues, we used dataset of customer expenditure in various service sectors, dataset of open percentage of small business across various service sectors and also the dataset of revenue generated for small business in various service sectors. Each dataset is uploaded, verified for missing values and rectified them when identified as below.

- Collecting and uploading of customer expenditure dataset,



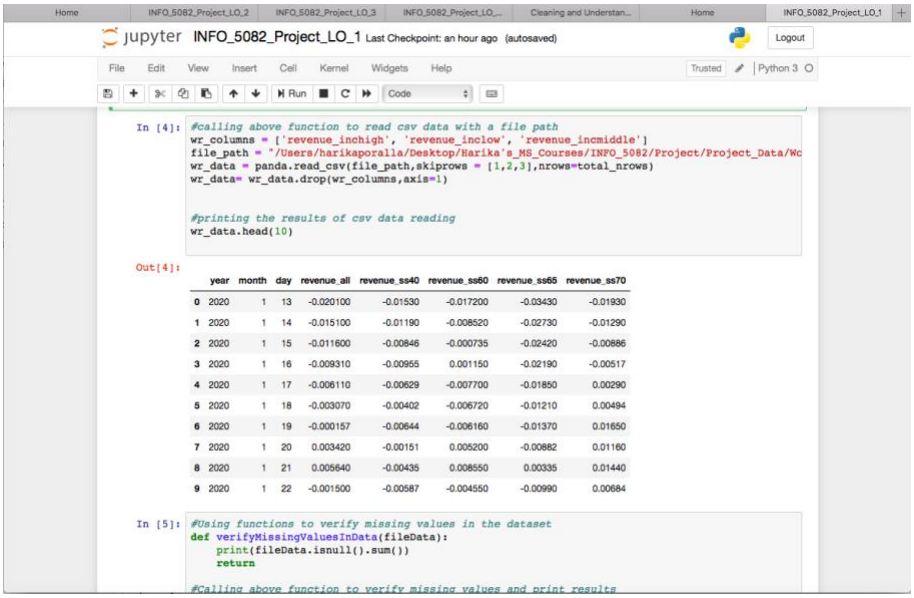
	year	month	day	spend_all	spend_apg	spend_aer	spend_grf	spend_hcs	spend_acf	spend_tws	spend_retail_no_grocer
292	2020	10	31	-0.0479	-0.1020	-0.555	0.123	-0.142	-0.300	-0.474	0.089
293	2020	11	1	-0.0624	-0.1050	-0.568	0.118	-0.135	-0.302	-0.477	0.084
294	2020	11	2	-0.0498	-0.1200	-0.536	0.127	-0.105	-0.303	-0.463	0.079
295	2020	11	3	-0.0530	-0.1360	-0.529	0.133	-0.191	-0.303	-0.455	0.063
296	2020	11	4	-0.0615	-0.1500	-0.526	0.116	-0.176	-0.309	-0.457	0.053
297	2020	11	5	-0.0651	-0.1550	-0.531	0.102	-0.180	-0.318	-0.460	0.046
298	2020	11	6	-0.0660	-0.1360	-0.528	0.105	-0.169	-0.317	-0.456	0.039
299	2020	11	7	-0.0592	-0.1230	-0.526	0.108	-0.163	-0.307	-0.458	0.043
300	2020	11	8	-0.0446	-0.1240	-0.517	0.114	-0.152	-0.308	-0.457	0.045
301	2020	11	15	-0.0283	-0.0341	-0.564	0.155	-0.135	-0.306	-0.478	0.133

❖ Collecting and uploading of small business open percentage dataset,



	year	month	day	merchants_all	merchants_ss40	merchants_ss60	merchants_ss65	merchants_ss70
0	2020	1	13	-0.00689	-0.00576	-0.00921	-0.00734	-0.004830
1	2020	1	14	-0.00854	-0.00749	-0.01020	-0.00873	-0.005540
2	2020	1	15	-0.01010	-0.00955	-0.01070	-0.01010	-0.006760
3	2020	1	16	-0.01110	-0.01050	-0.01100	-0.01100	-0.007620
4	2020	1	17	-0.00864	-0.00804	-0.00910	-0.00864	-0.004850
5	2020	1	18	-0.00513	-0.00438	-0.00600	-0.00669	-0.001330
6	2020	1	19	-0.00395	-0.00335	-0.00761	-0.00484	0.000165
7	2020	1	20	-0.00129	-0.00104	-0.00471	-0.00545	0.002620
8	2020	1	21	-0.00166	-0.00188	-0.00264	-0.00338	0.002100
9	2020	1	22	-0.00340	-0.00279	-0.00601	-0.00370	0.000727

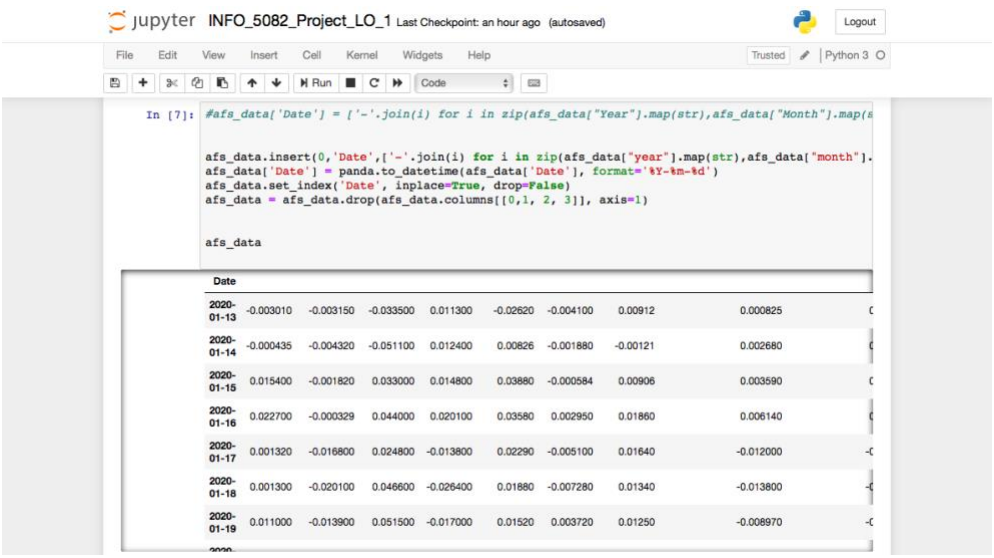
❖ Collecting and uploading of small business's revenue dataset,



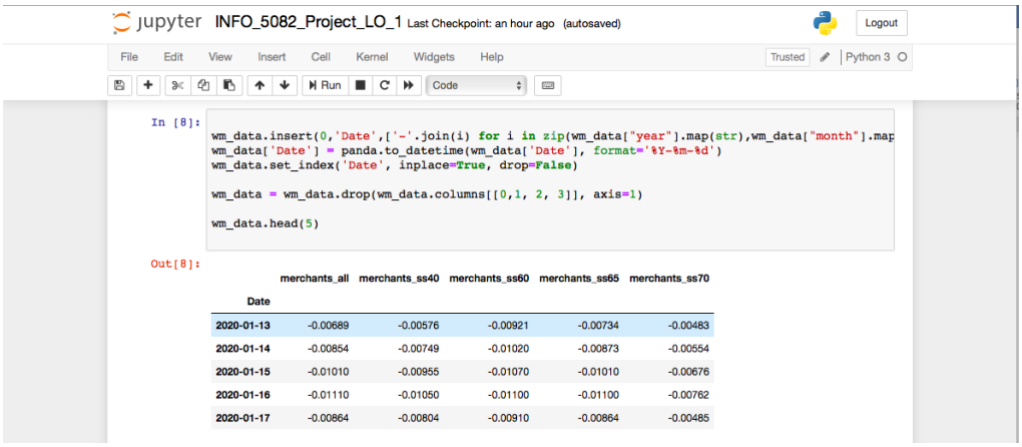
	year	month	day	revenue_all	revenue_ss40	revenue_ss60	revenue_ss65	revenue_ss70
0	2020	1	13	-0.020100	-0.01530	-0.017200	-0.03430	-0.01930
1	2020	1	14	-0.015100	-0.01190	-0.008520	-0.02730	-0.01290
2	2020	1	15	-0.011600	-0.00846	-0.000735	-0.02420	-0.00886
3	2020	1	16	-0.009310	-0.00955	0.001150	-0.02190	-0.00517
4	2020	1	17	-0.006110	-0.00629	-0.007700	-0.01850	0.00290
5	2020	1	18	-0.003070	-0.00402	-0.006720	-0.01210	0.00494
6	2020	1	19	-0.000157	-0.00644	-0.006160	-0.01370	0.01650
7	2020	1	20	0.003420	-0.00151	0.005200	-0.00882	0.01160
8	2020	1	21	0.005640	-0.00435	0.008550	0.00335	0.01440
9	2020	1	22	-0.001500	-0.00587	-0.004550	-0.00990	0.00684

For easy and better data visualizations three separate columns of Days, Month and Year in dataset are converted into an indexed column of Date with Datetime datatype as below.

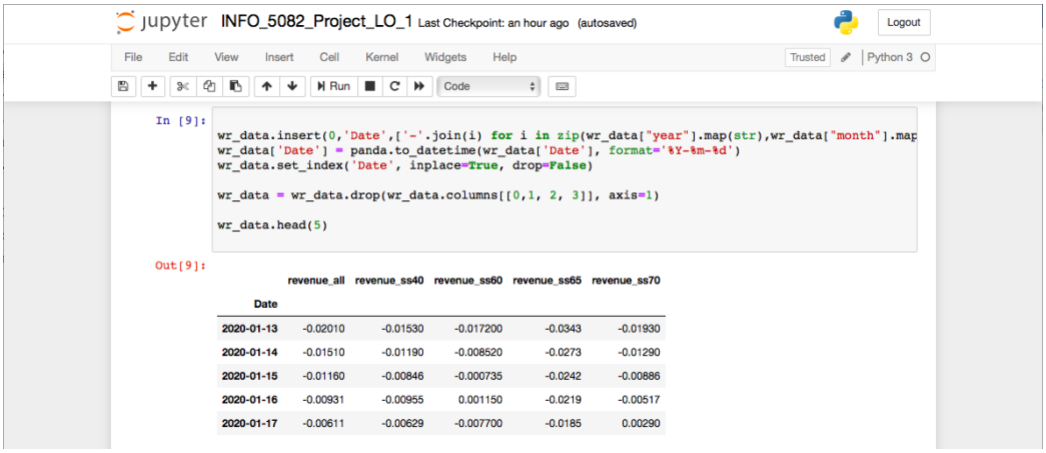
❖ Datetime indexing for customer expenditure dataset,



❖ Datetime indexing for business open percentage dataset,

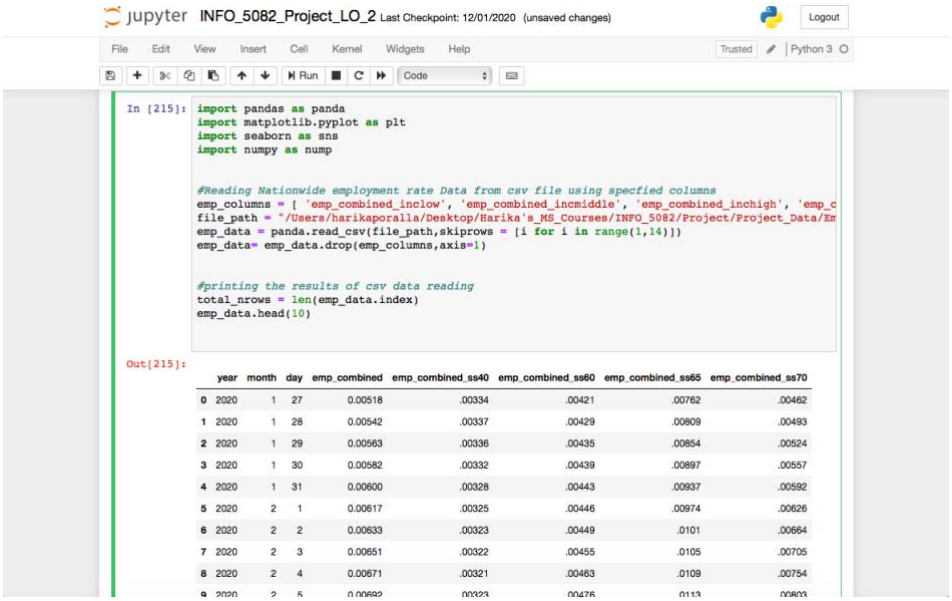


❖ Datetime indexing for business's revenue dataset,



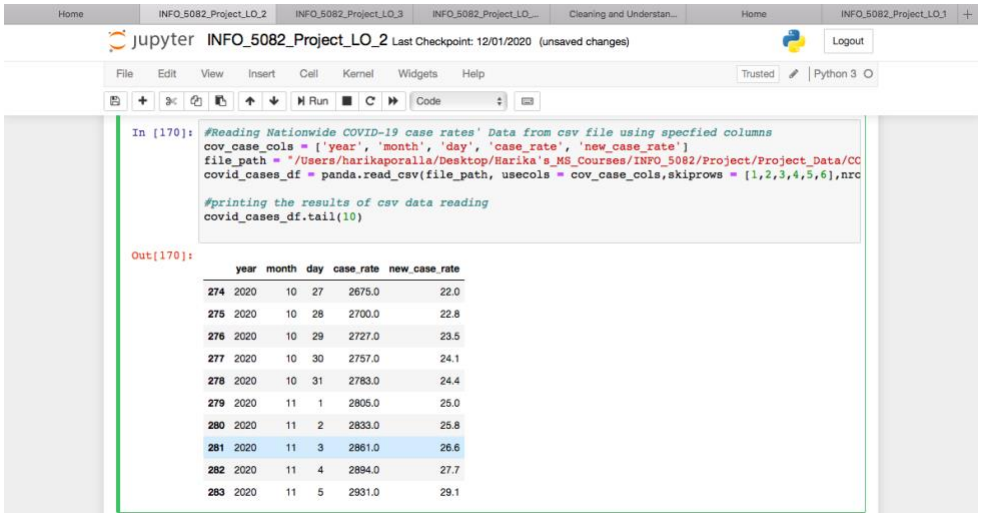
Similarly, for second learning objective i.e., for employment rate estimation, dataset of nationwide employment level rates recorded from 01-27-2020 to 11-05-2020 and the dataset of covid-19 infections rates in same period of time are collected, uploaded and verified for missingvalues to rectify.

- Collecting and uploading of nationwide employment rates' dataset,



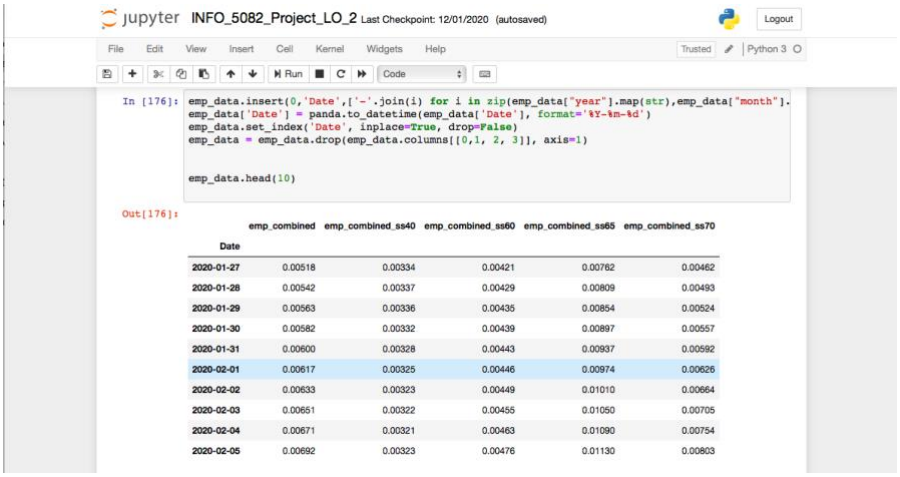
	year	month	day	emp_combined	emp_combined_ss40	emp_combined_ss60	emp_combined_ss65	emp_combined_ss70
0	2020	1	27	0.00518	.00334	.00421	.00762	.00462
1	2020	1	28	0.00542	.00337	.00429	.00809	.00493
2	2020	1	29	0.00563	.00336	.00435	.00854	.00524
3	2020	1	30	0.00582	.00332	.00439	.00897	.00557
4	2020	1	31	0.00600	.00328	.00443	.00937	.00592
5	2020	2	1	0.00617	.00325	.00446	.00974	.00626
6	2020	2	2	0.00633	.00323	.00449	.0101	.00664
7	2020	2	3	0.00651	.00322	.00455	.0105	.00705
8	2020	2	4	0.00671	.00321	.00463	.0109	.00754
9	2020	2	5	0.00692	.00323	.00476	.0113	.00803

- Collecting and uploading of nationwide covid-19 infection rates' dataset,



	year	month	day	case_rate	new_case_rate
274	2020	10	27	2675.0	22.0
275	2020	10	28	2700.0	22.8
276	2020	10	29	2727.0	23.5
277	2020	10	30	2757.0	24.1
278	2020	10	31	2783.0	24.4
279	2020	11	1	2805.0	25.0
280	2020	11	2	2833.0	25.8
281	2020	11	3	2861.0	26.6
282	2020	11	4	2894.0	27.7
283	2020	11	5	2931.0	29.1

❖ Datetime indexing for nationwide employment rates' dataset,



jupyter INFO_5082_Project_LO_2 Last Checkpoint: 12/01/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

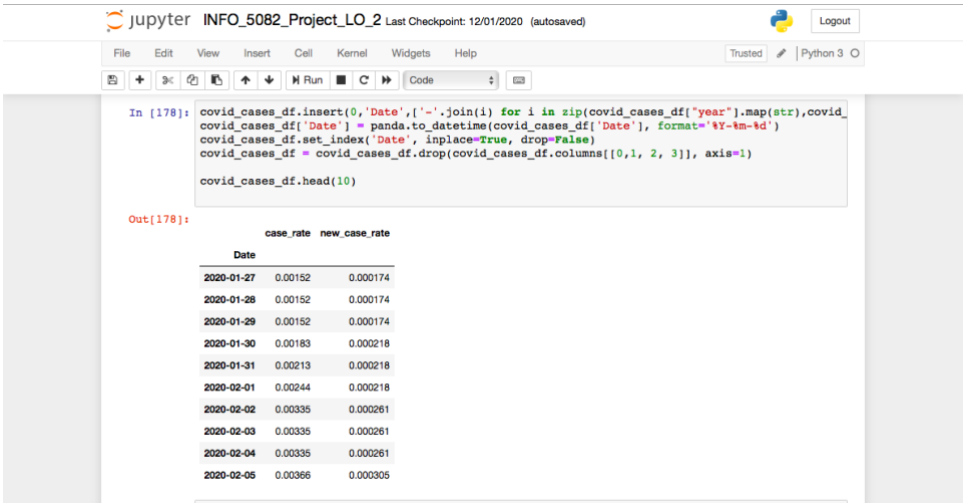
Run Code

In [176]:
```
emp_data.insert(0,'Date',['-'.join(i) for i in zip(emp_data["year"].map(str),emp_data["month"].
emp_data['Date'] = pandas.to_datetime(emp_data['Date'], format='%Y-%m-%d')
emp_data.set_index('Date', inplace=True, drop=False)
emp_data = emp_data.drop(emp_data.columns[[0,1, 2, 3]], axis=1)

emp_data.head(10)
```

Out[176]:

Date	emp_combined	emp_combined_ss40	emp_combined_ss60	emp_combined_ss65	emp_combined_ss70
2020-01-27	0.00518	0.00334	0.00421	0.00762	0.00462
2020-01-28	0.00542	0.00337	0.00429	0.00809	0.00493
2020-01-29	0.00563	0.00336	0.00435	0.00854	0.00524
2020-01-30	0.00582	0.00332	0.00439	0.00897	0.00557
2020-01-31	0.00600	0.00328	0.00443	0.00937	0.00592
2020-02-01	0.00617	0.00325	0.00446	0.00974	0.00626
2020-02-02	0.00633	0.00323	0.00449	0.01010	0.00664
2020-02-03	0.00651	0.00322	0.00455	0.01050	0.00705
2020-02-04	0.00671	0.00321	0.00463	0.01090	0.00754
2020-02-05	0.00692	0.00323	0.00476	0.01130	0.00803

❖ Datetime indexing for nationwide Covid-19 infection rates' dataset,



jupyter INFO_5082_Project_LO_2 Last Checkpoint: 12/01/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

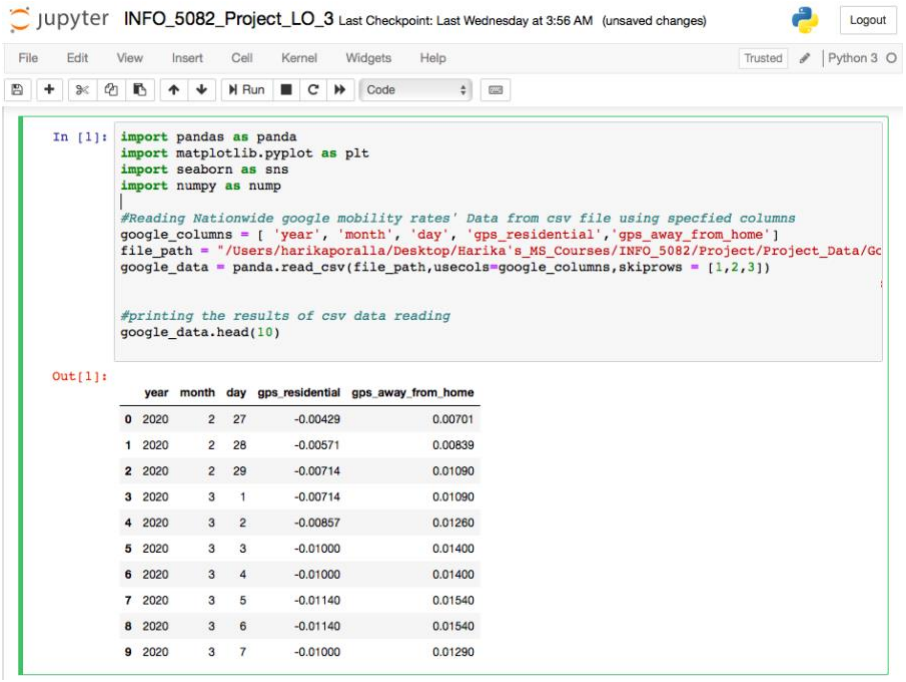
In [178]:
```
covid_cases_df.insert(0,'Date',['-'.join(i) for i in zip(covid_cases_df["year"].map(str),covid_
covid_cases_df['Date'] = pandas.to_datetime(covid_cases_df['Date'], format='%Y-%m-%d')
covid_cases_df.set_index('Date', inplace=True, drop=False)
covid_cases_df = covid_cases_df.drop(covid_cases_df.columns[[0,1, 2, 3]], axis=1)

covid_cases_df.head(10)
```

Out[178]:

Date	case_rate	new_case_rate
2020-01-27	0.00152	0.000174
2020-01-28	0.00152	0.000174
2020-01-29	0.00152	0.000174
2020-01-30	0.00183	0.000218
2020-01-31	0.00213	0.000218
2020-02-01	0.00244	0.000218
2020-02-02	0.00335	0.000261
2020-02-03	0.00335	0.000261
2020-02-04	0.00335	0.000261
2020-02-05	0.00366	0.000305

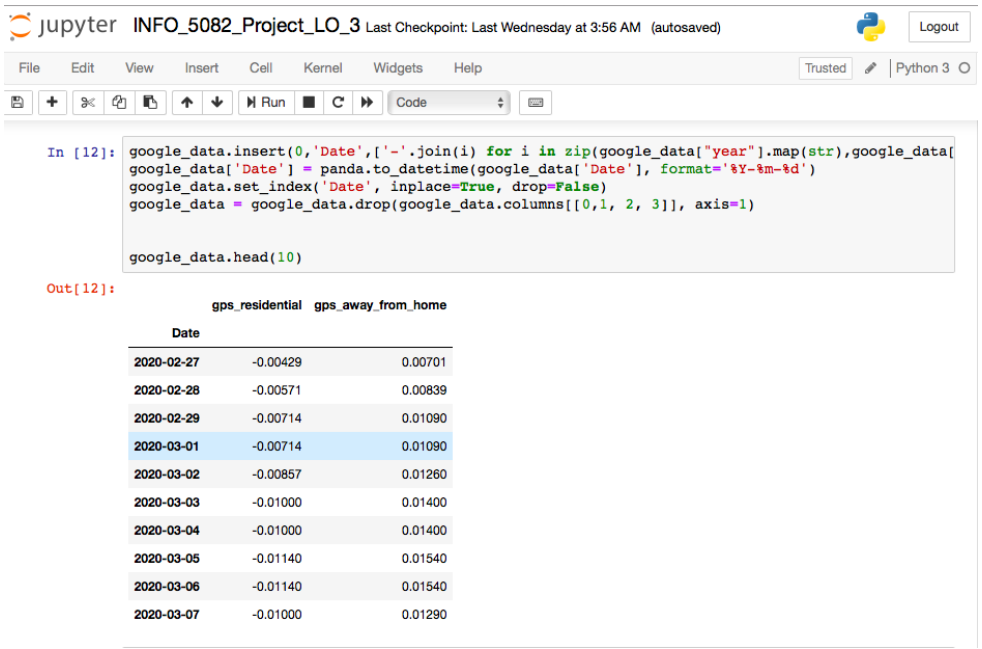
Similarly, for third learning objective i.e., to analyze the people's mobility rate with covid-19 case rate, dataset of nationwide google mobility rate from GPS values recorded from 02-27-2020 to 10-20-2020 and the dataset of covid-19 infections rates in same period of time are collected, uploaded and verified for missing values to rectify.

❖ Collecting and uploading of nationwide google mobility rates' dataset,



	year	month	day	gps_residential	gps_away_from_home
0	2020	2	27	-0.00429	0.00701
1	2020	2	28	-0.00571	0.00839
2	2020	2	29	-0.00714	0.01090
3	2020	3	1	-0.00714	0.01090
4	2020	3	2	-0.00857	0.01260
5	2020	3	3	-0.01000	0.01400
6	2020	3	4	-0.01000	0.01400
7	2020	3	5	-0.01140	0.01540
8	2020	3	6	-0.01140	0.01540
9	2020	3	7	-0.01000	0.01290

❖ Datetime indexing for nationwide google mobility rates' dataset,



Date	gps_residential	gps_away_from_home
2020-02-27	-0.00429	0.00701
2020-02-28	-0.00571	0.00839
2020-02-29	-0.00714	0.01090
2020-03-01	-0.00714	0.01090
2020-03-02	-0.00857	0.01260
2020-03-03	-0.01000	0.01400
2020-03-04	-0.01000	0.01400
2020-03-05	-0.01140	0.01540
2020-03-06	-0.01140	0.01540
2020-03-07	-0.01000	0.01290

Nationwide covid-19 infection rates' dataset is also uploaded and indexed by Date column with Datetime date type.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) for analysis is an approach to gain maximum of insight into data, understand the uncovering structure of data, detect outliers and anomalies in the data and determine optimal factor settings for developing parsimonious models by using few quantitative techniques and mostly graphical structures for visualizing the hidden patterns in the dataset.

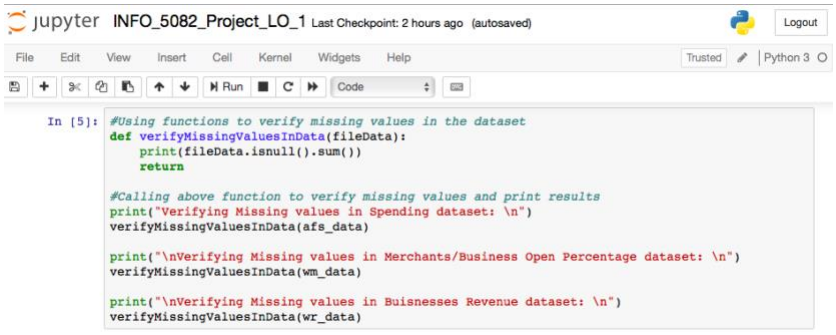
EDA helps in ensuring the results of the data analysis performed are the valid and applicable to given business problems. EDA Techniques include clustering, dimensional reduction in case of high-dimensional data, statistical summary of univariate, bivariate and multivariate visualizations for mapping and better understanding of variables' relationships to provide visual and numeric summaries of the data.

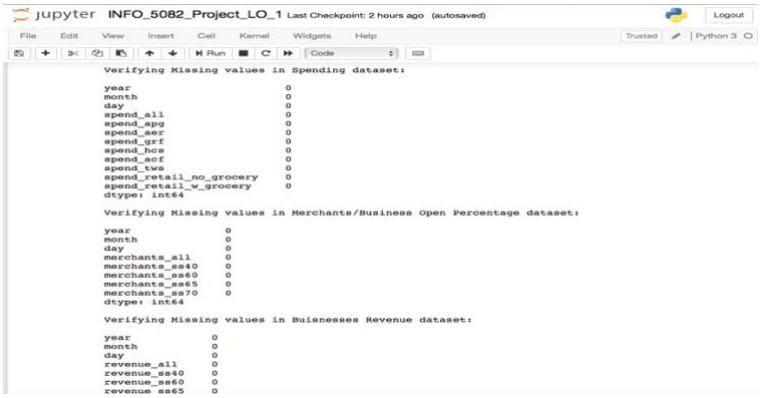
As mentioned in above, in this time series analysis, each dataset collected for all the three objectives are inspected first individually to verify and rectify the missing values and described to get insights for designing optimal model. After visualizing the summary of the dataset cleaning and preprocessing methods are applied as required to each dataset of all the objectives for analysis.

EDA for first Study objective

- Understanding the data:

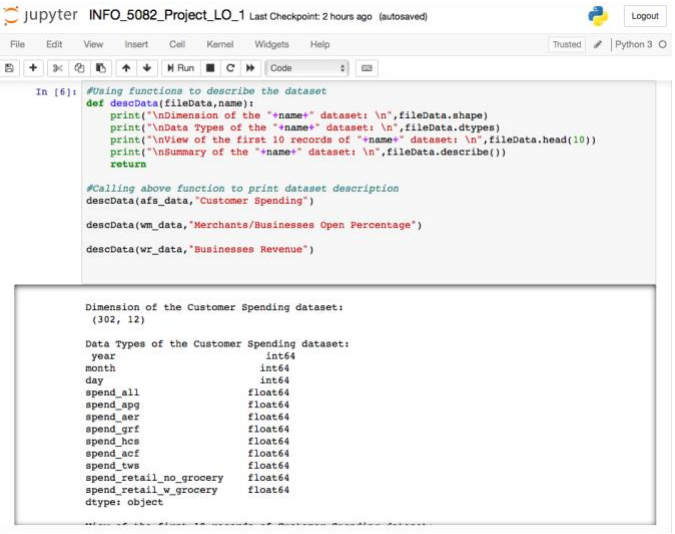
Verification of null values to rectify and the description of summary are coded as below.



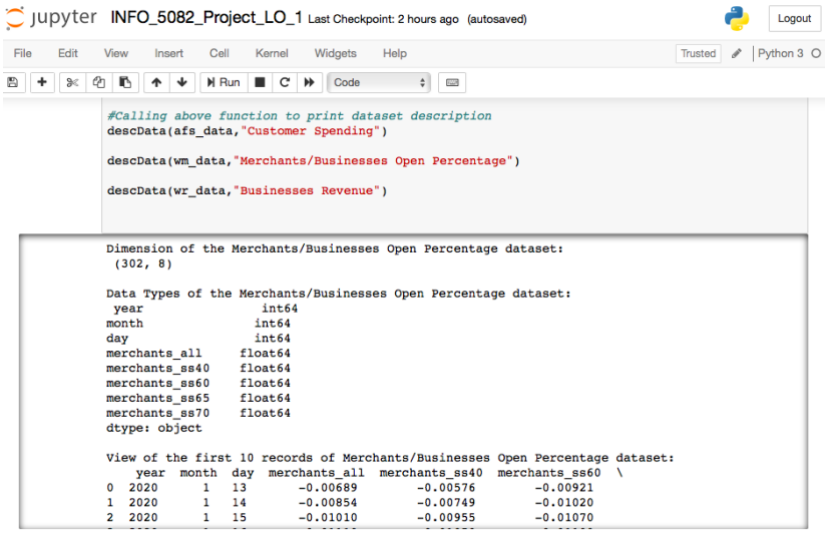


From the output generated there are no null/missing values in the dataset, so further describing the quantitative summaries as below.

- Summary of Customer expenditure dataset:

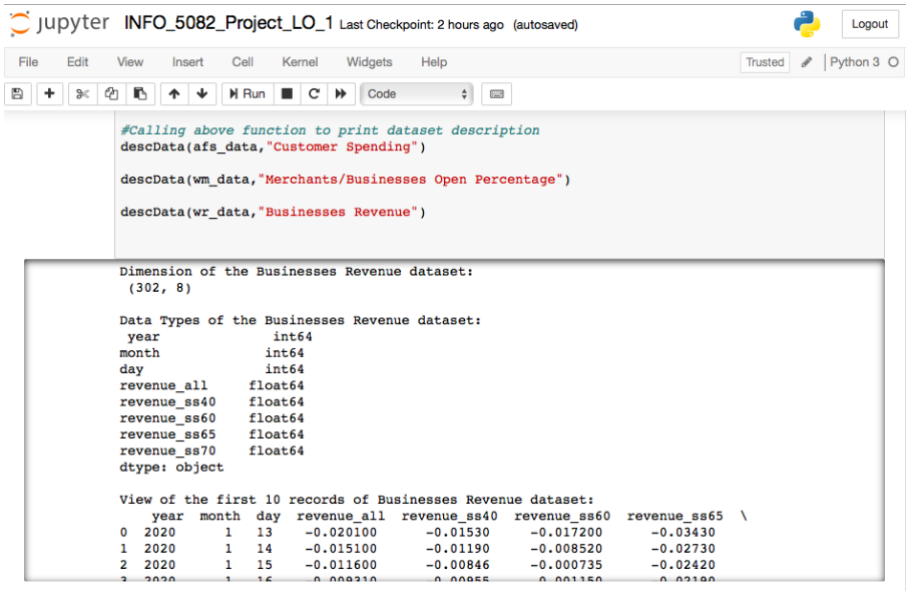


- Summary of Business open percentage dataset :

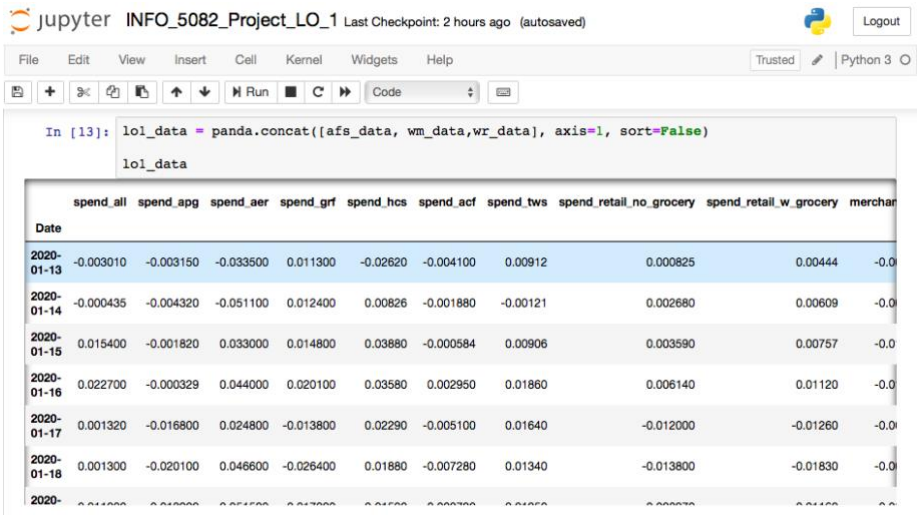


The results generated have shown some outliers in this dataset which are further visualized and rectified by using linear interpolation method.

- Summary of Business open percentage dataset :

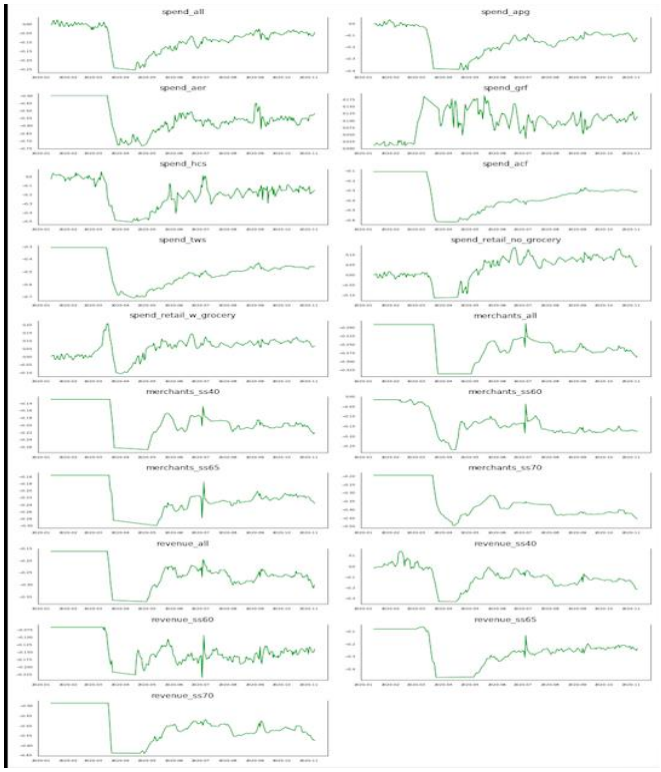


From these datasets required time series variables are combined together for the analysis to first objective,

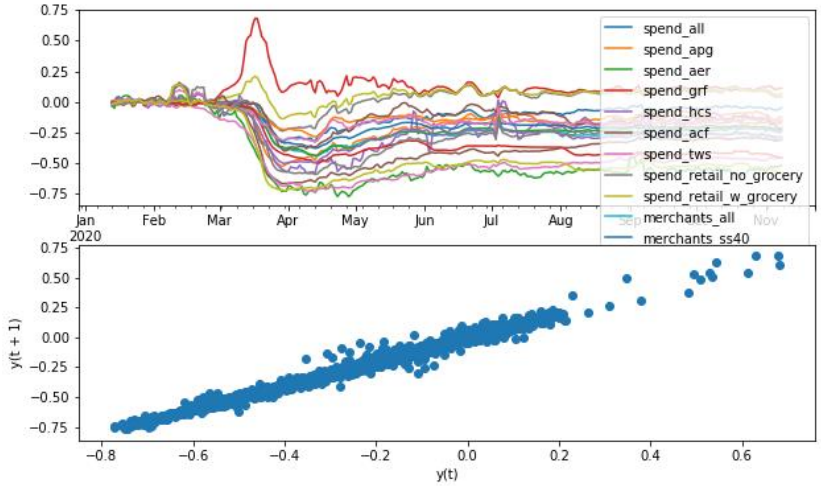


Further dataset is rectified for null values and visualized for better understanding,

- Univariate plots for each time series variable in the dataset:



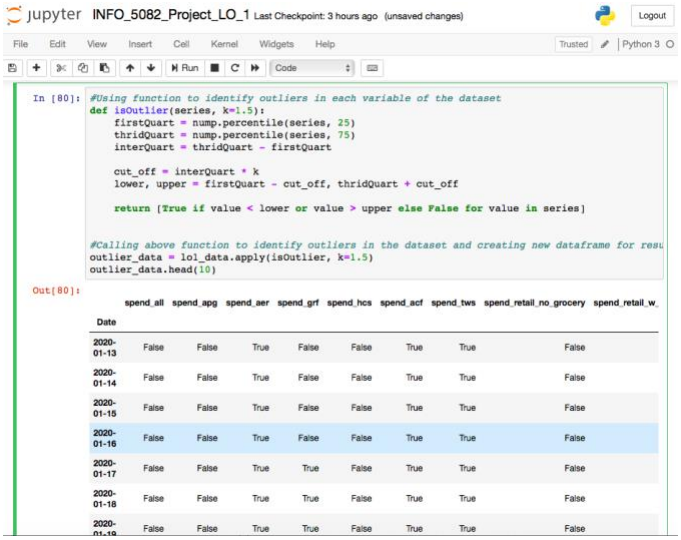
We can see the a slightly positive trend pattern in all variables from the month May, 2020.Further to understand the underlying structure pattern in the dataset, lag plot is drawn as below,



From the above plot of lag variables we can clearly see the outliers in the dataset. Also, we can clearly see the positive trend pattern in the dataset meaning each time series variable has linearity with its lagged variable. So, Vector Autocorrelation (VAR) can be applied if there is no cointegration among the variables in dataset.

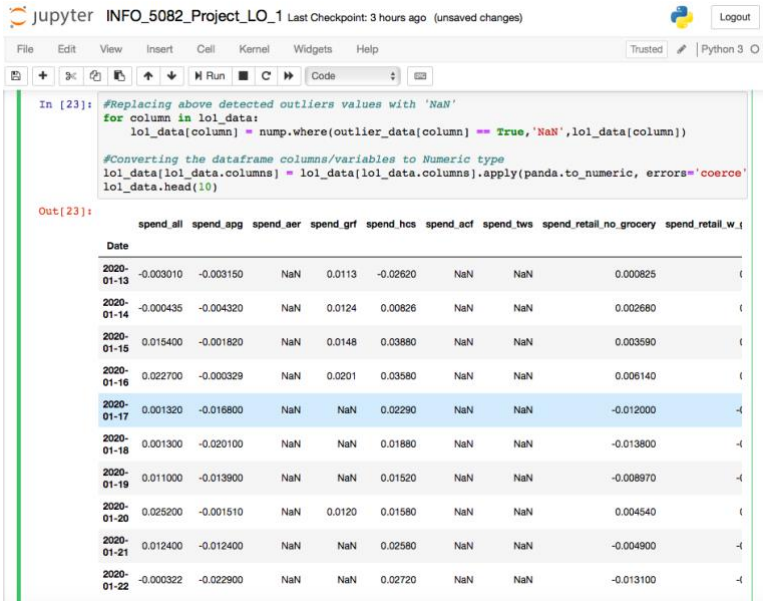
❖ Data Cleaning for rectifying outliers:

As the outliers in each time series variable are identified as below.

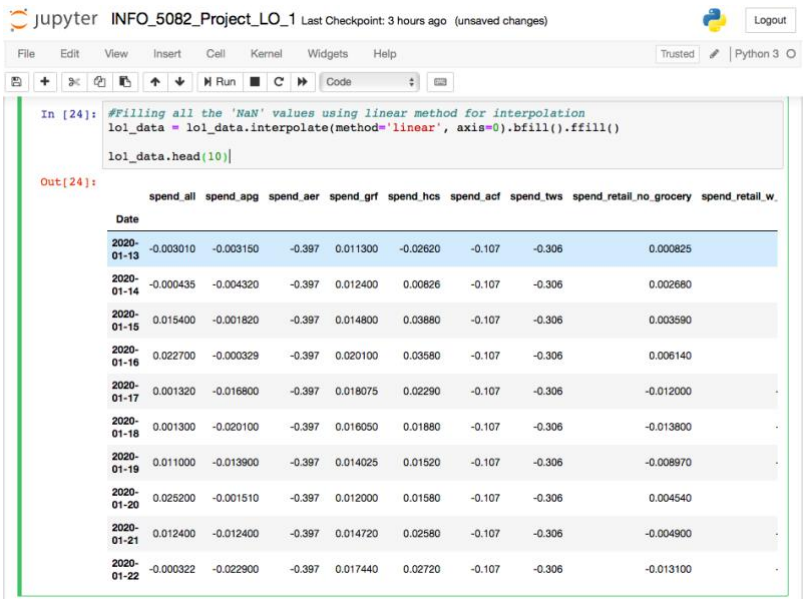


From the results generated, time series variables ‘spend_aer’, ‘spend_acf’, ‘spend_tws’, ‘merchant_all’ and more have shown outliers. So these outlier data points are rectified by using ‘Linear Interpolate’ method as below

❖ Rectifying outliers:



Date	spend_all	spend_apg	spend_aer	spend_grf	spend_hcs	spend_acf	spend_tws	spend_retail_no_grocery	spend_retail_w_
2020-01-13	-0.003010	-0.003150	NaN	0.0113	-0.02620	NaN	NaN	0.000825	
2020-01-14	-0.000435	-0.004320	NaN	0.0124	0.00826	NaN	NaN	0.002680	
2020-01-15	0.015400	-0.001820	NaN	0.0148	0.03880	NaN	NaN	0.003590	
2020-01-16	0.022700	-0.000329	NaN	0.0201	0.03580	NaN	NaN	0.006140	
2020-01-17	0.001320	-0.016800	NaN	NaN	0.02290	NaN	NaN	-0.012000	
2020-01-18	0.001300	-0.020100	NaN	NaN	0.01880	NaN	NaN	-0.013800	
2020-01-19	0.011000	-0.013900	NaN	NaN	0.01520	NaN	NaN	-0.008970	
2020-01-20	0.025200	-0.001510	NaN	0.0120	0.01580	NaN	NaN	0.004540	
2020-01-21	0.012400	-0.012400	NaN	NaN	0.02580	NaN	NaN	-0.004900	
2020-01-22	-0.000322	-0.022900	NaN	NaN	0.02720	NaN	NaN	-0.013100	



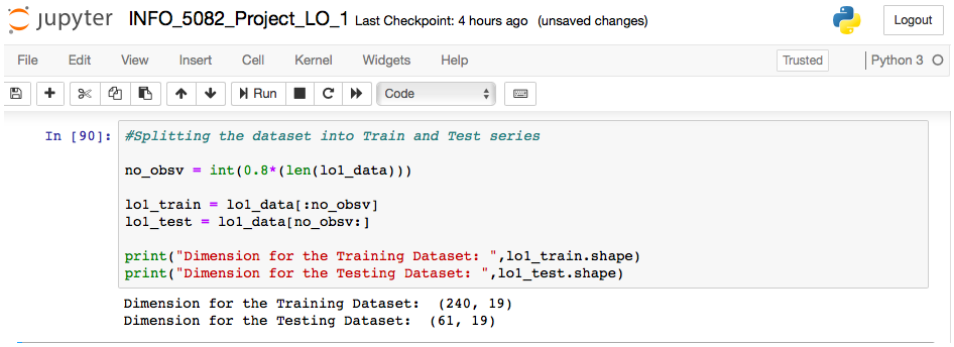
Date	spend_all	spend_apg	spend_aer	spend_grf	spend_hcs	spend_acf	spend_tws	spend_retail_no_grocery	spend_retail_w_
2020-01-13	-0.003010	-0.003150	-0.397	0.011300	-0.02620	-0.107	-0.306	0.000825	
2020-01-14	-0.000435	-0.004320	-0.397	0.012400	0.00826	-0.107	-0.306	0.002680	
2020-01-15	0.015400	-0.001820	-0.397	0.014800	0.03880	-0.107	-0.306	0.003590	
2020-01-16	0.022700	-0.000329	-0.397	0.020100	0.03580	-0.107	-0.306	0.006140	
2020-01-17	0.001320	-0.016800	-0.397	0.018075	0.02290	-0.107	-0.306	-0.012000	
2020-01-18	0.001300	-0.020100	-0.397	0.016050	0.01880	-0.107	-0.306	-0.013800	
2020-01-19	0.011000	-0.013900	-0.397	0.014025	0.01520	-0.107	-0.306	-0.008970	
2020-01-20	0.025200	-0.001510	-0.397	0.012000	0.01580	-0.107	-0.306	0.004540	
2020-01-21	0.012400	-0.012400	-0.397	0.014720	0.02580	-0.107	-0.306	-0.004900	
2020-01-22	-0.000322	-0.022900	-0.397	0.017440	0.02720	-0.107	-0.306	-0.013100	

Now the outliers in dataset is cleaned and rectified, further the dataset is splitted into Training and Test dataset.

❖ Splitting data into Train and Test data:

In this multivariate time series analysis, VAR model is fitted and trained with train dataset to forecast the test dataset and the actual test dataset is used to compare with predicted output for model accuracy.

For first objective's analysis dataset is split into (0.8) 80% for training and (0.2) 20% for testing as below.



❖ Stationarity and making a stationary time series:

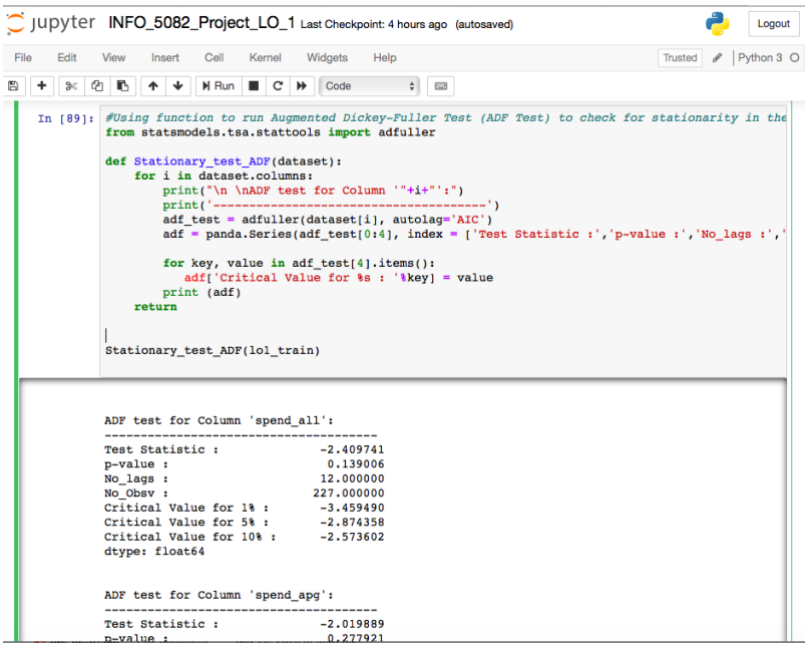
The assumption of summary statistics from observations are consistent require the time series to be stationary. This analysis consistency can be disturbed by the trend, seasonality or other time dependent patterns.

A stationary time series has the mean, variance and covariance that doesn't vary with time. Stationarity is an important element in time series analysis. There are three types stationarity, a) Strict stationary – this strict stationary series has the mean , variance and covariance that are not a function of time. For our analysis, we must convert a non-stationary time series into stationary, b) Trend Stationary – A series that has trend but no unit root is called Trend stationary which can be converted into Strict stationary by removing the trend and c) Difference Stationary – A strict stationary formed by using differencing is called Difference stationary.

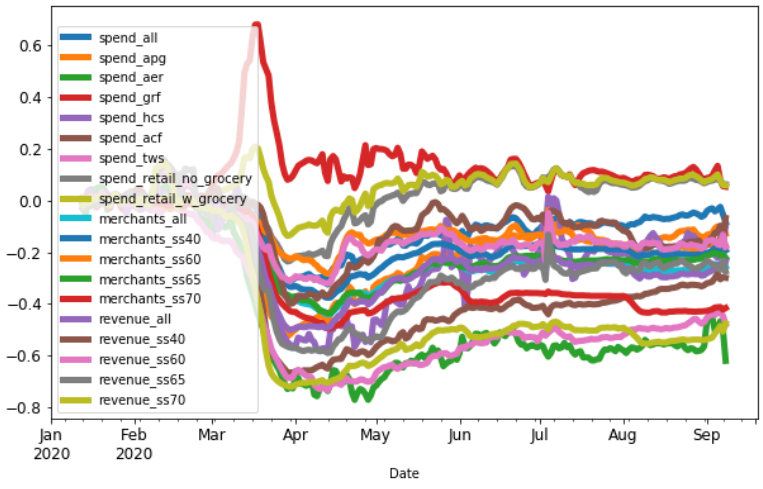
A non-stationary time series can be converted into stationary by i) Differencing – here, to get rid of the varying mean we compute the difference between consecutive terms in series. ii) Seasonal Differencing – here, we compute the difference between two observations in same season over different period like difference between a observation in a month with its previous month. iii) Transformation – here, methods like log transformation, square root and power transform are used to stabilize a series with non-constant variance.

❖ Verifying and converting to stationarity:

To verify time series property of stationarity in the dataset variables, Augmented Dickey Fuller test function is applied as below.



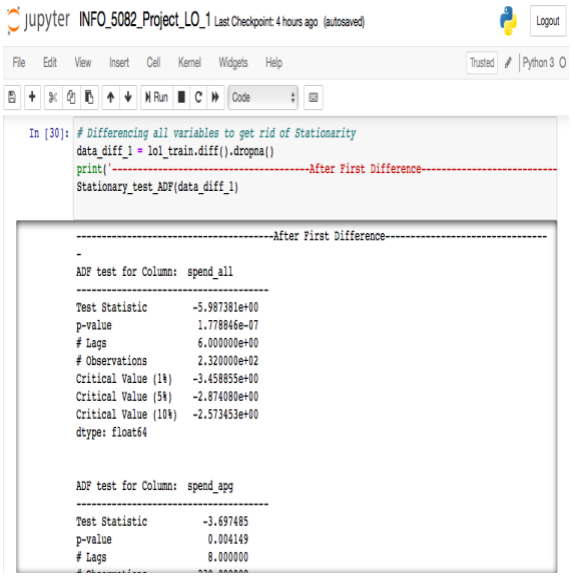
The ADF tests states, Null hypothesis – Data has the unit root meaning it is a non-stationary data when the p-value > 0.05 meaning weak evidence to reject null hypothesis.

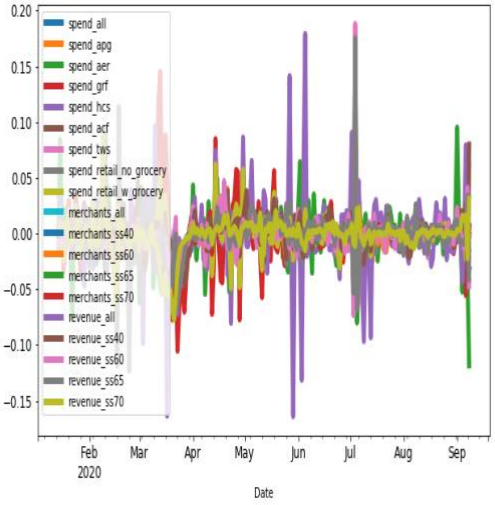


As the results shown that data is not stationary, to convert the time series dataset into stationary Differencing method has been implemented in two orders where the difference of consecutive terms are computed mathematically as,

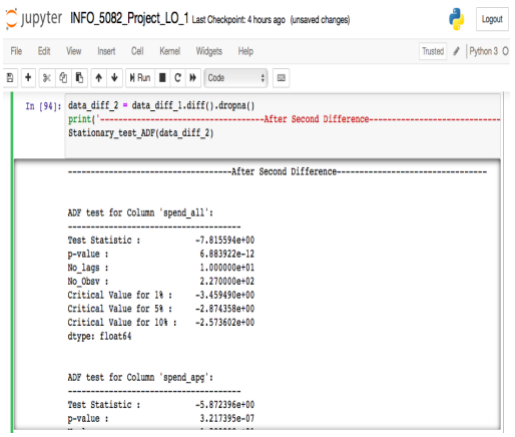
$$y_t = y_t + y_{t-1}, \quad \text{where } y_t \text{ value of variable 'y' at a time 't'.}$$

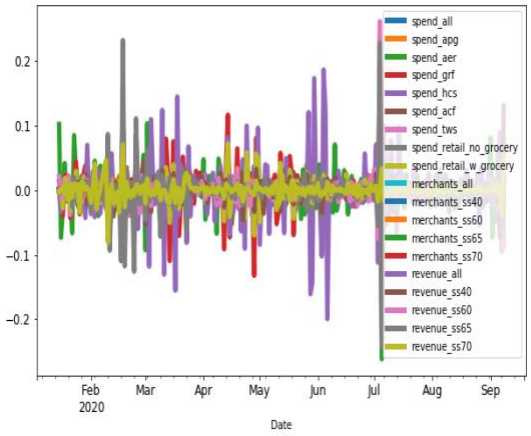
After first differencing, ADF test results shown as non-stationary, so second differencing is applied.





❖ ADF test results from second difference has shown as stationary.





Now the data is in stationary and ready for time series modelling.

❖ Cointegration:

It can be defined in simple as two time series variables x_t and y_t are said to be cointegrated if there exists a parameter 'α' such as $u_t = y_t - \alpha x_t$ is a stationary process.

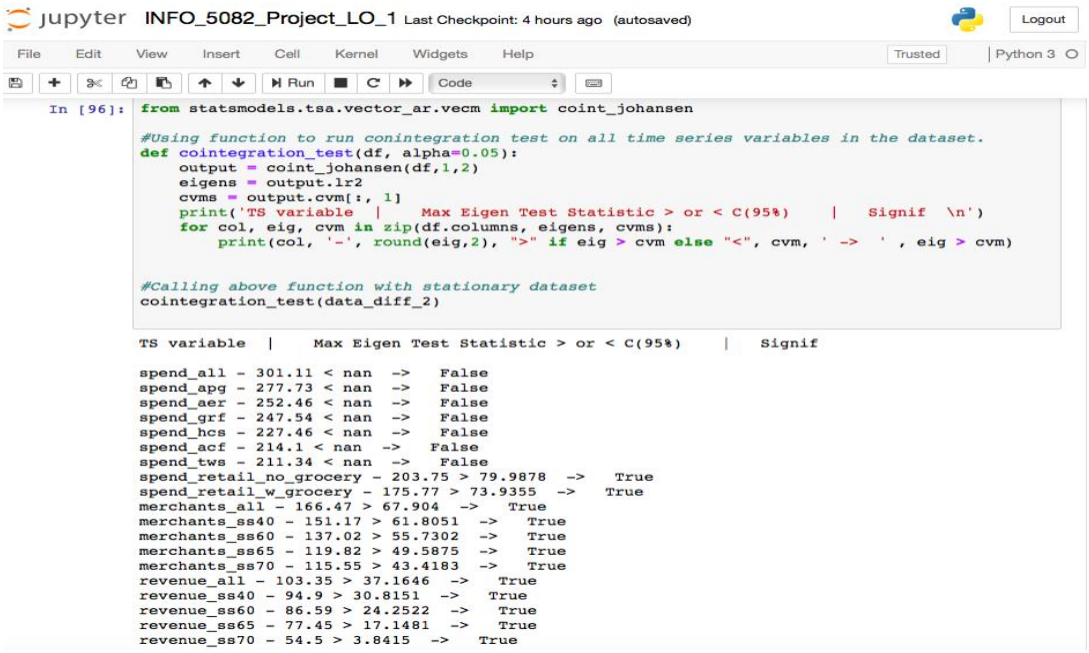
Cointegration for autoregression VAR(k) model in general is given as,

$\Delta y_t = \Pi y_{t-1} + \sum_{j=1}^{k-1} \Gamma_t \Delta y_{t-1} + e_t$

When Π = 0 means there is no cointegration, Π has full cointegrating rank 'r', then y_t must be stationary Π has less then full rank but not equal to zero then there is a conintegration.

❖ Cointegration Testing using Johansen's function:

To select the appropriate model, dataset has been tested for cointegration among the variables to avoid the information loss using Johansen cointegration taking Eigen Test statistic with 95% confidence interval.



Johansen's cointegration test helps us to identify the statistically significant relation among time series variables in the dataset. The null hypothesis of Johansen test implies that the sum of eigen values or trace is equal to zero meaning there is no cointegration or linear trend in

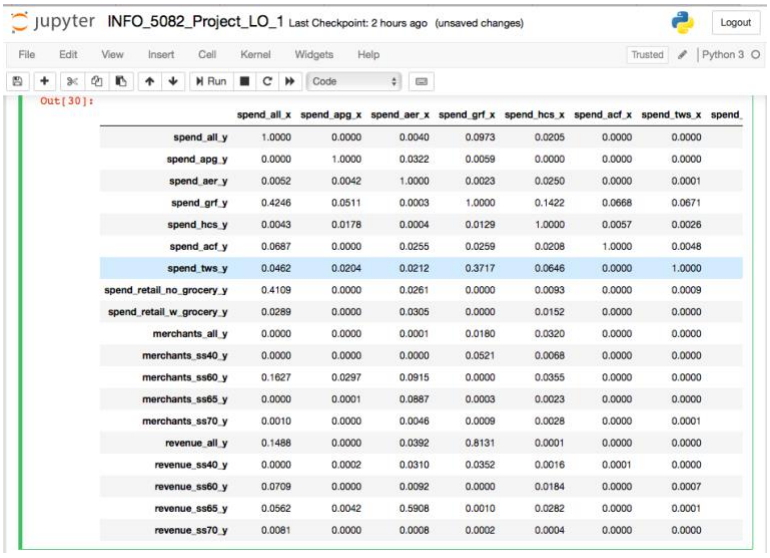
the dataset. It eigen statistic value or trace is greater than critical value then we can reject the null hypothesis with given confidence level.

From the results generated, we can see that some variables are cointegrated that means in long run the time series variables are significant relationship tis dataset when modelled using VECM model can have little loss of information than when modelled with VAR.

❖ Granger Causality Matrix:

This causality matrix reveals if any times series variables in the dataset are causing effect/consequences when forecasting a time series variable. The causation of a time series variable in the dataset by its own lagged variable and other variables is given by the p-value stating as when < 0.05 (significance) then dependent variable(y) is caused by another variable (x) and vice versa.

From the granger causality matrix applied we can see that there is some causality or linearity between some non-cointegrated variables, so this dataset is also modelled with VAR model.



	spend_all_x	spend_apg_x	spend_aer_x	spend_grf_x	spend_hcs_x	spend_acf_x	spend_tws_x	spend_
spend_all_y	1.0000	0.0000	0.0040	0.0973	0.0205	0.0000	0.0000	
spend_apg_y	0.0000	1.0000	0.0322	0.0059	0.0000	0.0000	0.0000	
spend_aer_y	0.0052	0.0042	1.0000	0.0023	0.0250	0.0000	0.0001	
spend_grf_y	0.4246	0.0511	0.0003	1.0000	0.1422	0.0668	0.0671	
spend_hcs_y	0.0043	0.0178	0.0004	0.0129	1.0000	0.0057	0.0026	
spend_acf_y	0.0687	0.0000	0.0255	0.0259	0.0208	1.0000	0.0048	
spend_tws_y	0.0462	0.0204	0.0212	0.3717	0.0646	0.0000	1.0000	
spend_retail_no_grocery_y	0.4109	0.0000	0.0261	0.0000	0.0093	0.0000	0.0009	
spend_retail_w_grocery_y	0.0289	0.0000	0.0305	0.0000	0.0152	0.0000	0.0000	
merchants_all_y	0.0000	0.0000	0.0001	0.0180	0.0320	0.0000	0.0000	
merchants_ss40_y	0.0000	0.0000	0.0000	0.0521	0.0068	0.0000	0.0000	
merchants_ss60_y	0.1627	0.0297	0.0915	0.0000	0.0355	0.0000	0.0000	
merchants_ss65_y	0.0000	0.0001	0.0887	0.0003	0.0023	0.0000	0.0000	
merchants_ss70_y	0.0010	0.0000	0.0046	0.0009	0.0028	0.0000	0.0001	
revenue_all_y	0.1488	0.0000	0.0392	0.8131	0.0001	0.0000	0.0000	
revenue_ss40_y	0.0000	0.0002	0.0310	0.0352	0.0016	0.0001	0.0000	
revenue_ss60_y	0.0709	0.0000	0.0092	0.0000	0.0184	0.0000	0.0007	
revenue_ss65_y	0.0562	0.0042	0.5908	0.0010	0.0282	0.0000	0.0001	
revenue_ss70_y	0.0081	0.0000	0.0008	0.0002	0.0004	0.0000	0.0000	

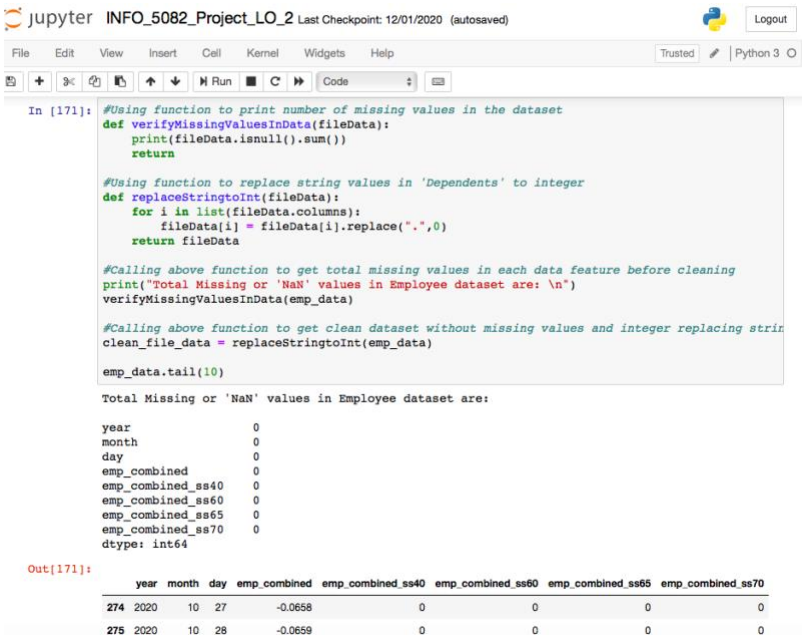
From the accuracy results of VAR and VECM model we can say which model is better for time series analysis of first study objective.

EDA for Second Study objective

- Understanding the data:

  Verification of null values to rectify and the description of summary are coded as below.

- Employment rates' dataset:



jupyter INFO_5082_Project_LO_2 Last Checkpoint: 12/01/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

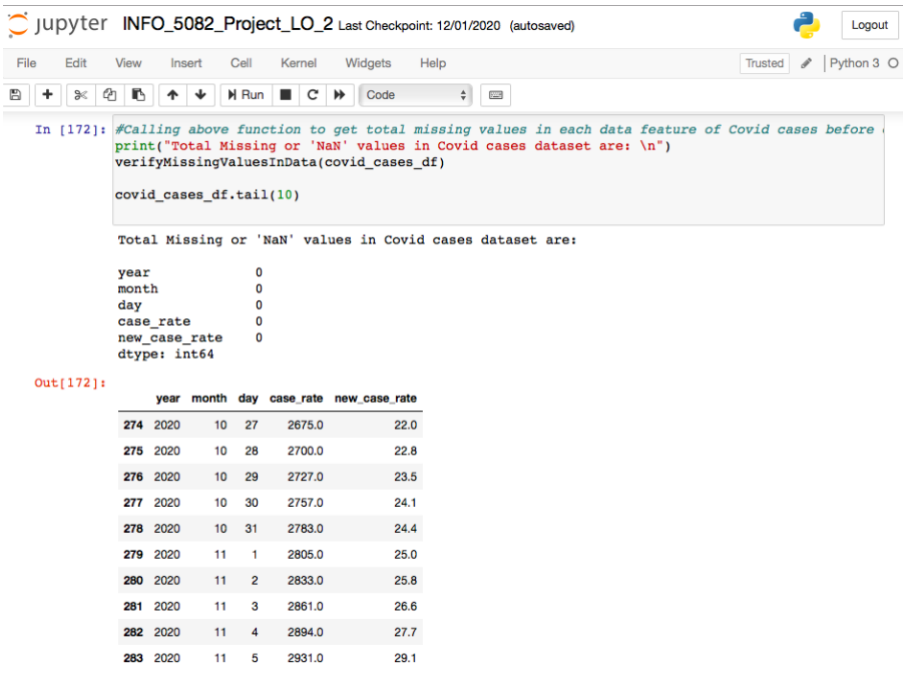
In [171]:
```
#Using function to print number of missing values in the dataset
def verifyMissingValuesInData(fileData):
    print(fileData.isnull().sum())
    return

#Using function to replace string values in 'Dependents' to integer
def replaceStringtoInt(fileData):
    for i in list(fileData.columns):
        fileData[i] = fileData[i].replace(".",0)
    return fileData

#Calling above function to get total missing values in each data feature before cleaning
print("Total Missing or 'NaN' values in Employee dataset are: \n")
verifyMissingValuesInData(emp_data)

#Calling above function to get clean dataset without missing values and integer replacing strin
clean_file_data = replaceStringtoInt(emp_data)

emp_data.tail(10)
```

```
Total Missing or 'NaN' values in Employee dataset are:

year                0
month               0
day                 0
emp_combined        0
emp_combined_ss40   0
emp_combined_ss60   0
emp_combined_ss65   0
emp_combined_ss70   0
dtype: int64
```

Out[171]:

	year	month	day	emp_combined	emp_combined_ss40	emp_combined_ss60	emp_combined_ss65	emp_combined_ss70
274	2020	10	27	-0.0658	0	0	0	0
275	2020	10	28	-0.0659	0	0	0	0

- Covid-19 infection rates' dataset:



jupyter INFO_5082_Project_LO_2 Last Checkpoint: 12/01/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [172]:
```
#Calling above function to get total missing values in each data feature of Covid cases before
print("Total Missing or 'NaN' values in Covid cases dataset are: \n")
verifyMissingValuesInData(covid_cases_df)

covid_cases_df.tail(10)
```

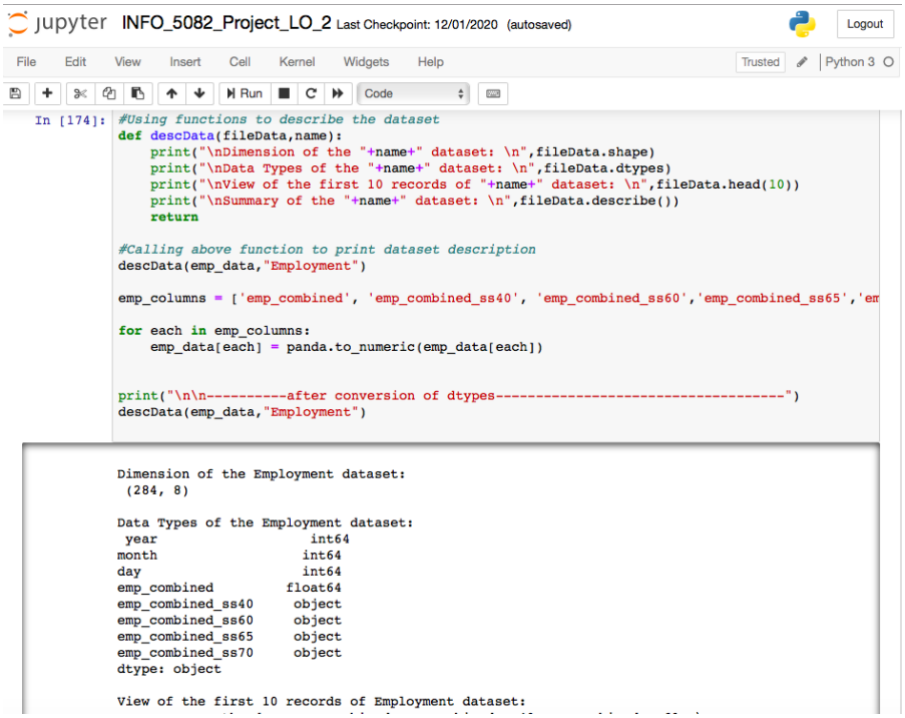
```
Total Missing or 'NaN' values in Covid cases dataset are:

year             0
month            0
day              0
case_rate        0
new_case_rate    0
dtype: int64
```

Out[172]:

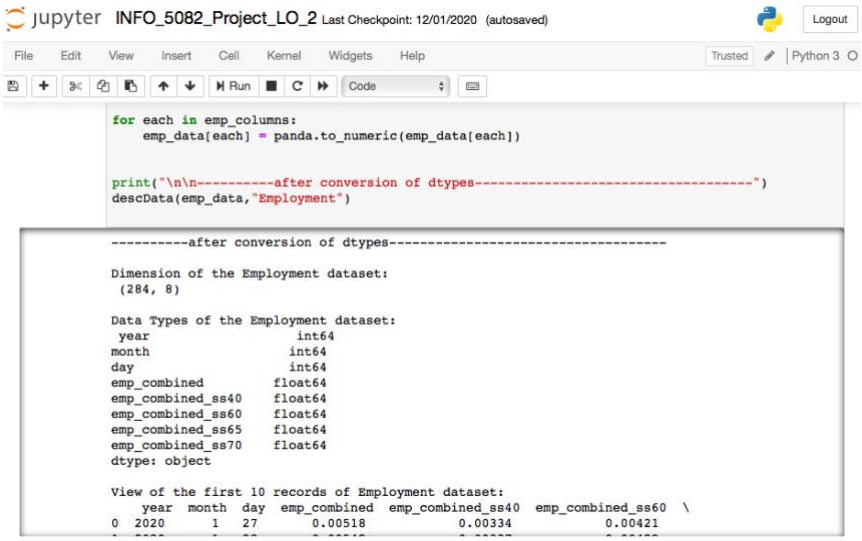
	year	month	day	case_rate	new_case_rate
274	2020	10	27	2675.0	22.0
275	2020	10	28	2700.0	22.8
276	2020	10	29	2727.0	23.5
277	2020	10	30	2757.0	24.1
278	2020	10	31	2783.0	24.4
279	2020	11	1	2805.0	25.0
280	2020	11	2	2833.0	25.8
281	2020	11	3	2861.0	26.6
282	2020	11	4	2894.0	27.7
283	2020	11	5	2931.0	29.1

From the output generated, there are no null/missing values in the dataset, so further describing the quantitative summaries as below.

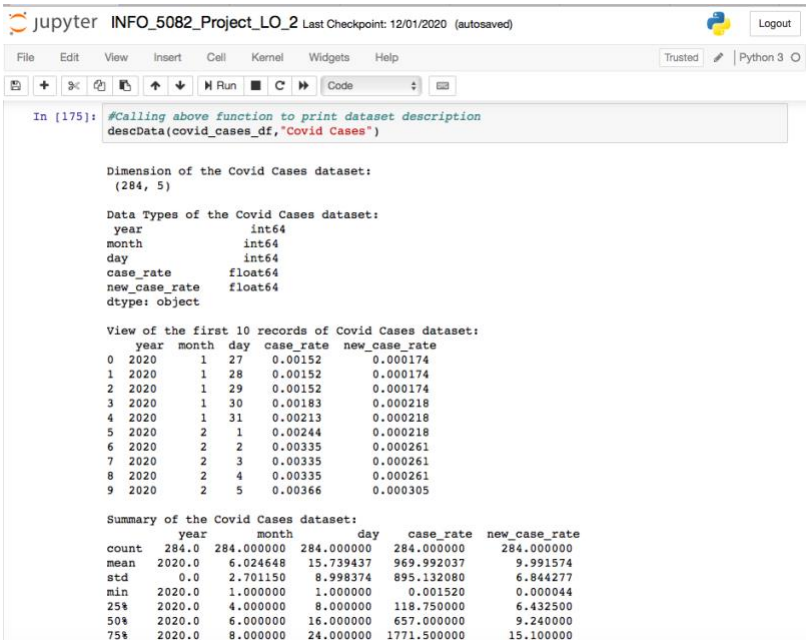
❖ Summary of Employment rates' dataset:



Since, this dataset has variables of object type conversion to numeric type is applied and the results are as below.



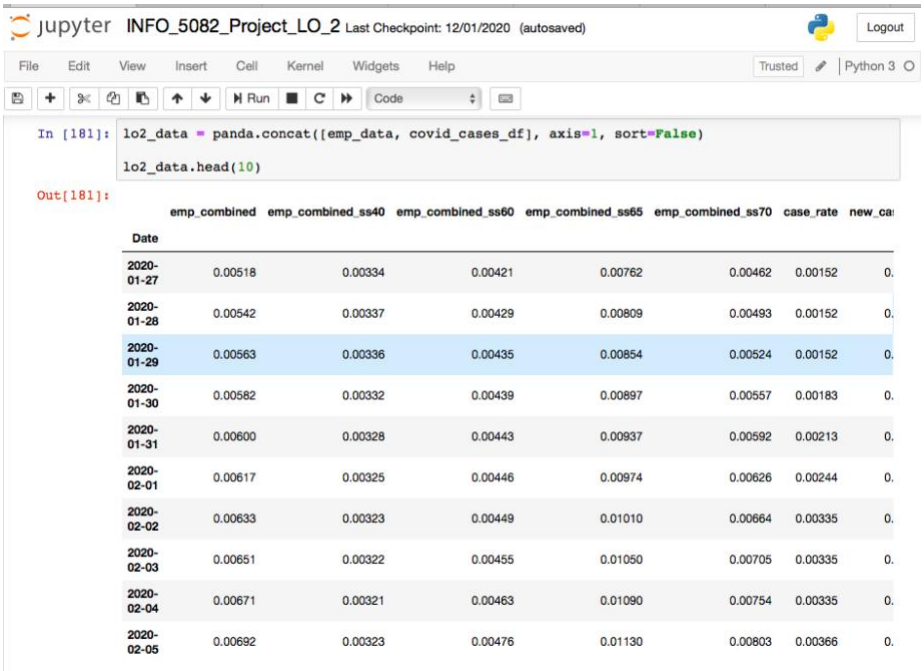
❖ Summary of Covid-19 Infection rates' dataset :



```
In [175]: #Calling above function to print dataset description
          descData(covid_cases_df,"Covid Cases")

Dimension of the Covid Cases dataset:
 (284, 5)

Data Types of the Covid Cases dataset:
 year              int64
month              int64
day                int64
case_rate        float64
new_case_rate    float64
dtype: object

View of the first 10 records of Covid Cases dataset:
    year  month  day  case_rate  new_case_rate
0  2020      1   27    0.00152       0.000174
1  2020      1   28    0.00152       0.000174
2  2020      1   29    0.00152       0.000174
3  2020      1   30    0.00183       0.000218
4  2020      1   31    0.00213       0.000218
5  2020      2    1    0.00244       0.000218
6  2020      2    2    0.00335       0.000261
7  2020      2    3    0.00335       0.000261
8  2020      2    4    0.00335       0.000261
9  2020      2    5    0.00366       0.000305

Summary of the Covid Cases dataset:
             year       month         day    case_rate  new_case_rate
count   284.0  284.000000  284.000000   284.000000     284.000000
mean   2020.0    6.024648   15.739437   969.992037       9.991574
std       0.0    2.701150    8.998374   895.132080       6.844277
min    2020.0    1.000000    1.000000     0.001520       0.000044
25%    2020.0    4.000000    8.000000   118.750000       6.432500
50%    2020.0    6.000000   16.000000   657.000000       9.240000
75%    2020.0    8.000000   24.000000  1771.500000      15.100000
```

The results generated have shown no outliers.

From these datasets required time series variables are combined together for the analysis to second objective,



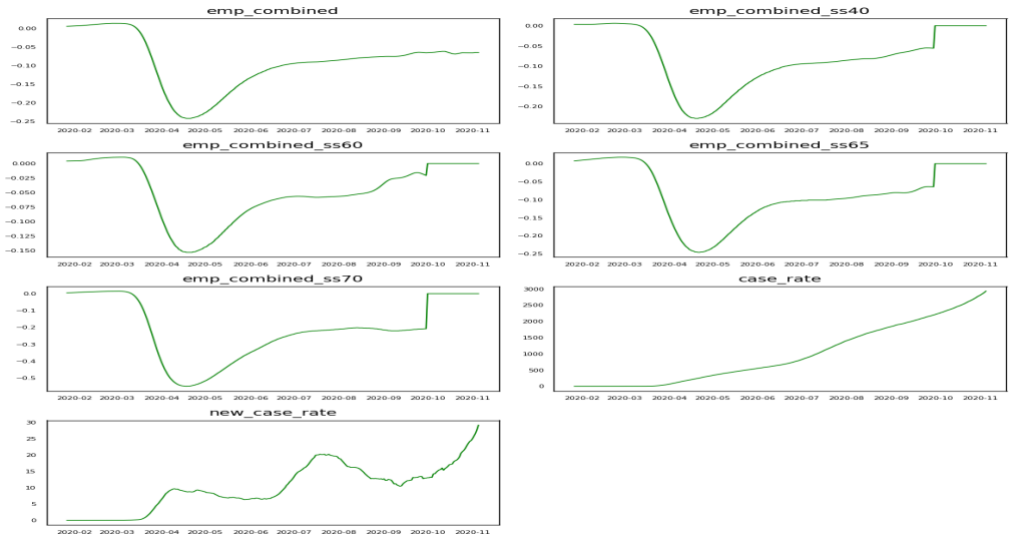
```
In [181]: lo2_data = panda.concat([emp_data, covid_cases_df], axis=1, sort=False)

          lo2_data.head(10)
```

Out[181]:

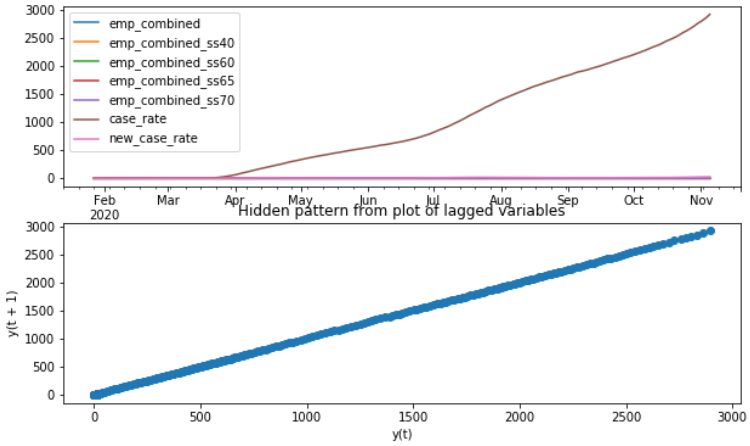
Date	emp_combined	emp_combined_ss40	emp_combined_ss60	emp_combined_ss65	emp_combined_ss70	case_rate	new_ca
2020-01-27	0.00518	0.00334	0.00421	0.00762	0.00462	0.00152	0.
2020-01-28	0.00542	0.00337	0.00429	0.00809	0.00493	0.00152	0.
2020-01-29	0.00563	0.00336	0.00435	0.00854	0.00524	0.00152	0.
2020-01-30	0.00582	0.00332	0.00439	0.00897	0.00557	0.00183	0.
2020-01-31	0.00600	0.00328	0.00443	0.00937	0.00592	0.00213	0.
2020-02-01	0.00617	0.00325	0.00446	0.00974	0.00626	0.00244	0.
2020-02-02	0.00633	0.00323	0.00449	0.01010	0.00664	0.00335	0.
2020-02-03	0.00651	0.00322	0.00455	0.01050	0.00705	0.00335	0.
2020-02-04	0.00671	0.00321	0.00463	0.01090	0.00754	0.00335	0.
2020-02-05	0.00692	0.00323	0.00476	0.01130	0.00803	0.00366	0.

❖ Univariate plots for each time series variable in the dataset:

Further dataset is visualized for better understanding



We can see the a slightly positive trend pattern in all variables from the month May, 2020.Further to understand the underlying structure pattern in the dataset, lag plot is drawn as below,

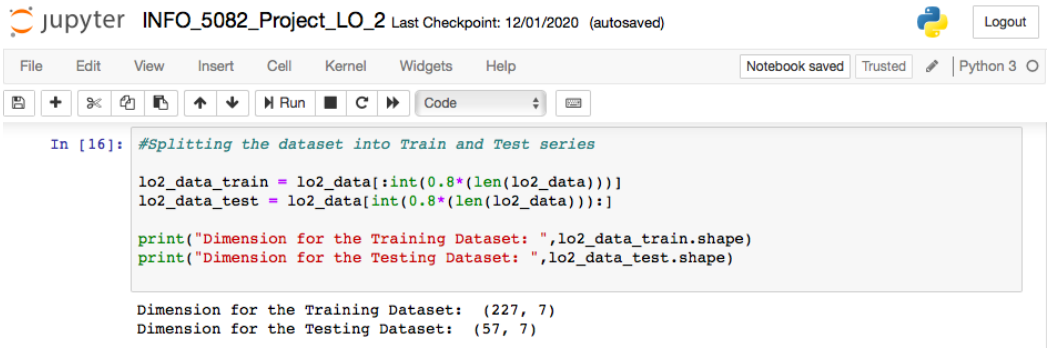


From the above plot of lag variables, there are no outliers in the dataset. Also, we can clearly see the positive trend pattern in the dataset meaning each time series variable has linearity

with its lagged variable. So, Vector Autocorrelation (VAR) can be applied if there is no cointegration among the variables in dataset.

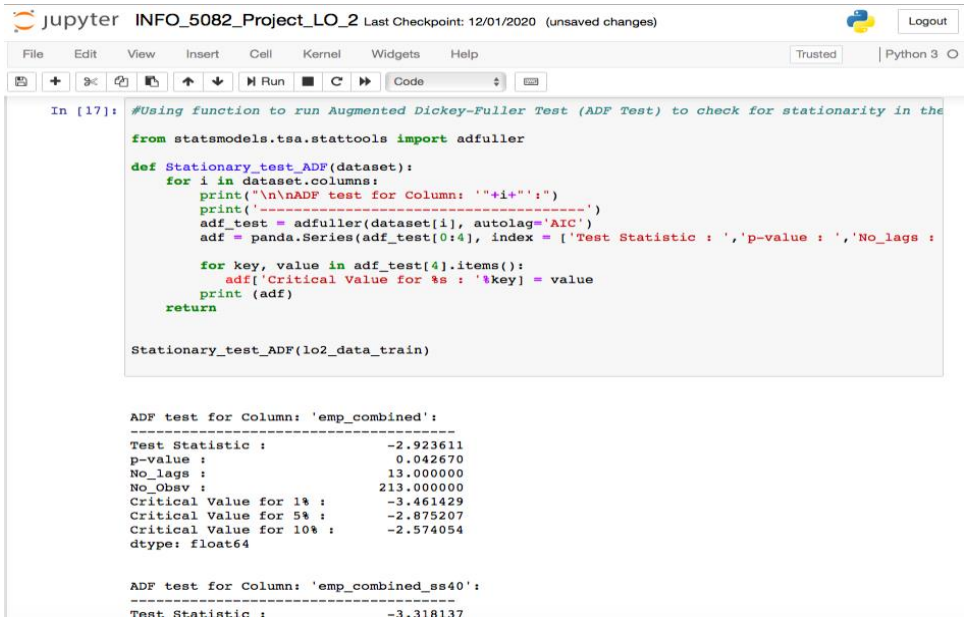
❖ Splitting data into Train and Test data:

In this multivariate time series analysis for second study objective, dataset is split into (0.8) 80% for training and (0.2) 20% for testing as below.



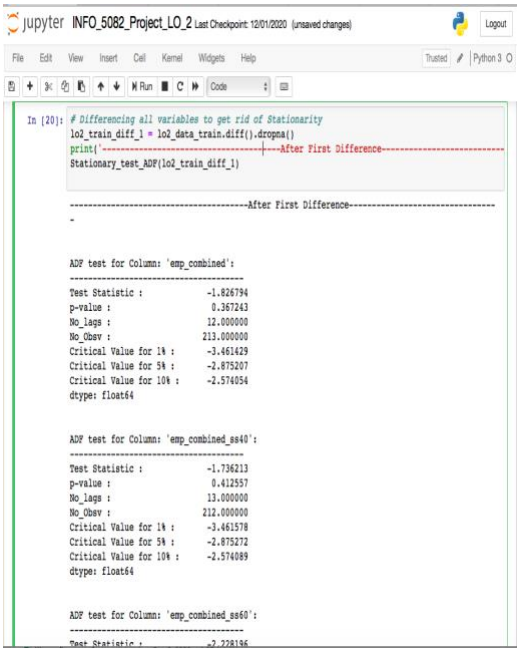
❖ Verifying and converting to stationarity:

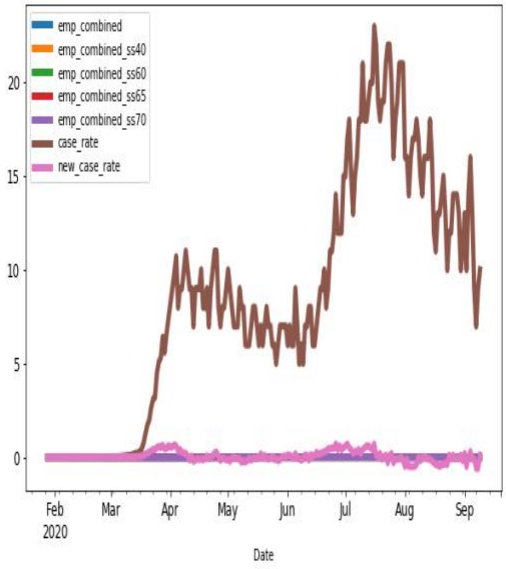
To verify time series property of stationarity in the dataset variables, Augmented Dickey Fuller test function is applied on the train dataset as below.



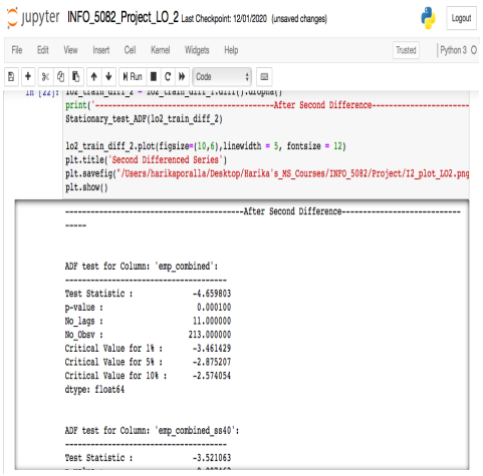
As the results ADF test shown that data is not stationary, to convert the time series dataset into stationary Differencing method has been implemented in three orders.

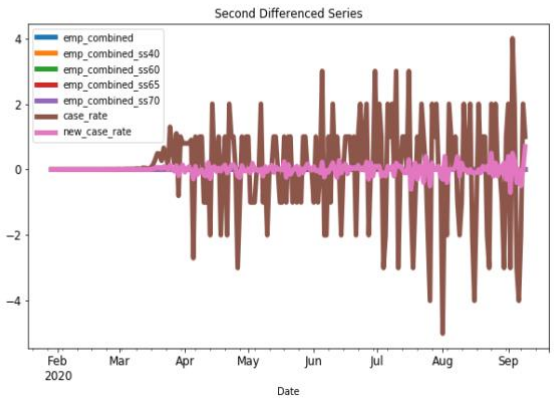
After first differencing, ADF test results shown as non-stationary, so second differencing is applied.



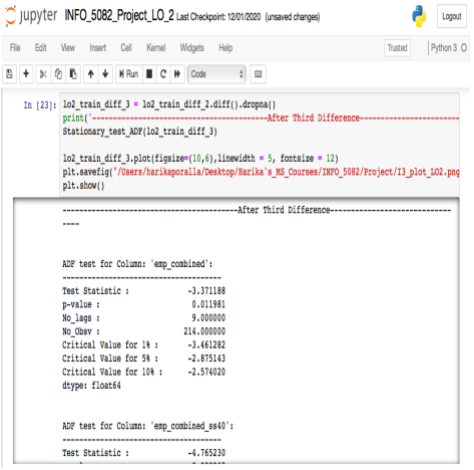


ADF test results from second difference are also not shown as stationary, so third differencing is performed.

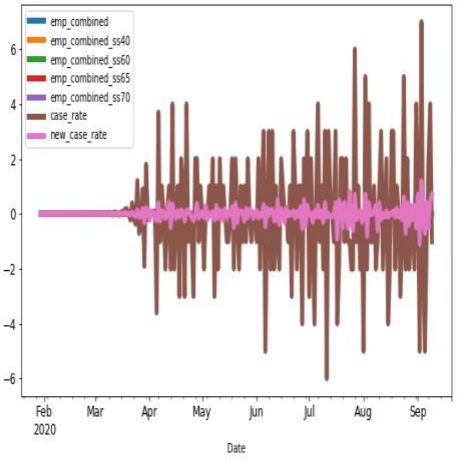




ADF results from third differencing has shown as stationary.



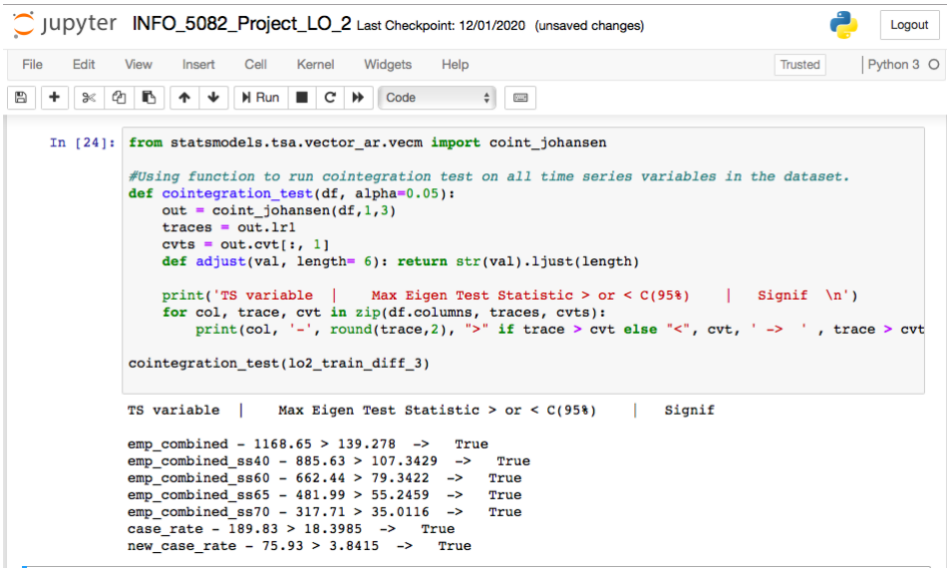
```
In [23]: lo2_train_diff_3 = lo2_train_diff_2.diff().dropna()
         print('----------------------------------------After Third Difference----------------------------------------')
         Stationary_test_ADF(lo2_train_diff_3)

         lo2_train_diff_3.plot(figsize=(10,6),linewidth = 5, fontsize = 12)
         plt.savefig('/Users/harikaporalla/Desktop/Harika's_MS_Courses/INFO_5082/Project/13_plot_LO2.png
         plt.show()

----------------------------------------After Third Difference----------------------------------------
----


ADF test for Column: 'emp_combined':
---------------------------------------
Test Statistic :             -3.371188
p-value :                     0.011981
No_lags :                     9.000000
No_Obsv :                   214.000000
Critical Value for 1% :      -3.461282
Critical Value for 5% :      -2.875143
Critical Value for 10% :     -2.574020
dtype: float64


ADF test for Column: 'emp_combined_ss40':
-----------------------------------------
Test Statistic :             -4.765230
```



Now the data is in stationary and ready for time series modelling.

❖ Cointegration Testing using Johansen's function:

To select the appropriate model, dataset has been tested using Johansen's cointegration among the variables to avoid the information loss taking Trace Test statistic with 95% confidence interval.

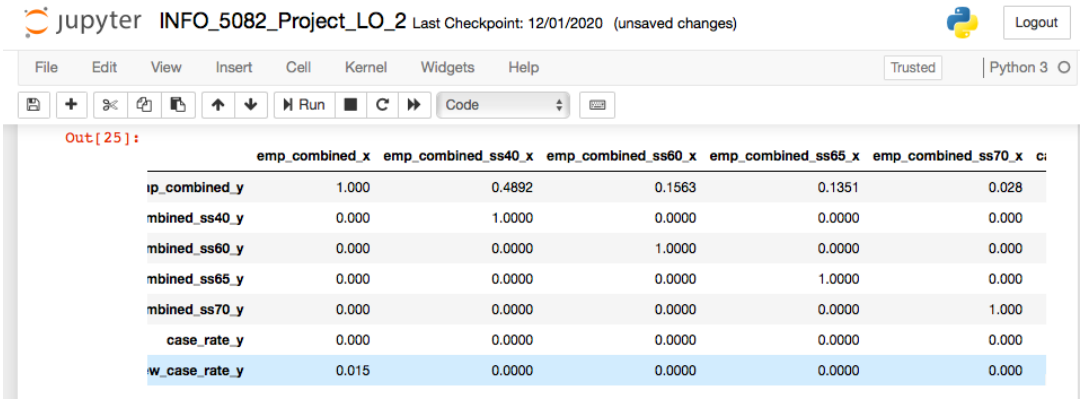


```
In [24]: from statsmodels.tsa.vector_ar.vecm import coint_johansen

         #Using function to run cointegration test on all time series variables in the dataset.
         def cointegration_test(df, alpha=0.05):
             out = coint_johansen(df,1,3)
             traces = out.lr1
             cvts = out.cvt[:, 1]
             def adjust(val, length= 6): return str(val).ljust(length)

             print('TS variable  |    Max Eigen Test Statistic > or < C(95%)    |   Signif  \n')
             for col, trace, cvt in zip(df.columns, traces, cvts):
                 print(col, '-', round(trace,2), ">" if trace > cvt else "<", cvt, ' ->  ' , trace > cvt

         cointegration_test(lo2_train_diff_3)

TS variable  |    Max Eigen Test Statistic > or < C(95%)    |   Signif

emp_combined - 1168.65 > 139.278  ->   True
emp_combined_ss40 - 885.63 > 107.3429  ->   True
emp_combined_ss60 - 662.44 > 79.3422  ->   True
emp_combined_ss65 - 481.99 > 55.2459  ->   True
emp_combined_ss70 - 317.71 > 35.0116  ->   True
case_rate - 189.83 > 18.3985  ->   True
new_case_rate - 75.93 > 3.8415  ->   True
```

From the results generated, we can see that all the time series variables are cointegrated meaning in long run the time series variables have significant correlations, so this dataset is modelled using VECM model to have little loss of information than when modelled with VAR.

❖ Granger Causality Matrix:

From the granger causality matrix applied we can see that there is causality or linearity between all the variables, so this dataset is with VECM model without ignoring any variables.



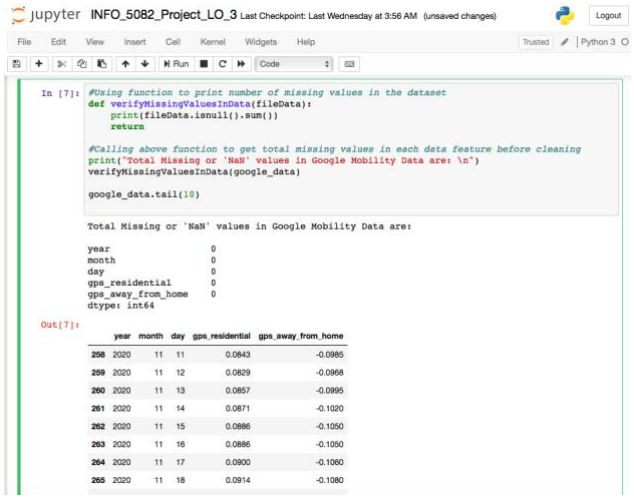
	emp_combined_x	emp_combined_ss40_x	emp_combined_ss60_x	emp_combined_ss65_x	emp_combined_ss70_x	ca
ıp_combined_y	1.000	0.4892	0.1563	0.1351	0.028	
mbined_ss40_y	0.000	1.0000	0.0000	0.0000	0.000	
mbined_ss60_y	0.000	0.0000	1.0000	0.0000	0.000	
mbined_ss65_y	0.000	0.0000	0.0000	1.0000	0.000	
mbined_ss70_y	0.000	0.0000	0.0000	0.0000	1.000	
case_rate_y	0.000	0.0000	0.0000	0.0000	0.000	
w_case_rate_y	0.015	0.0000	0.0000	0.0000	0.000	

EDA for third study objective

- Understanding the data:

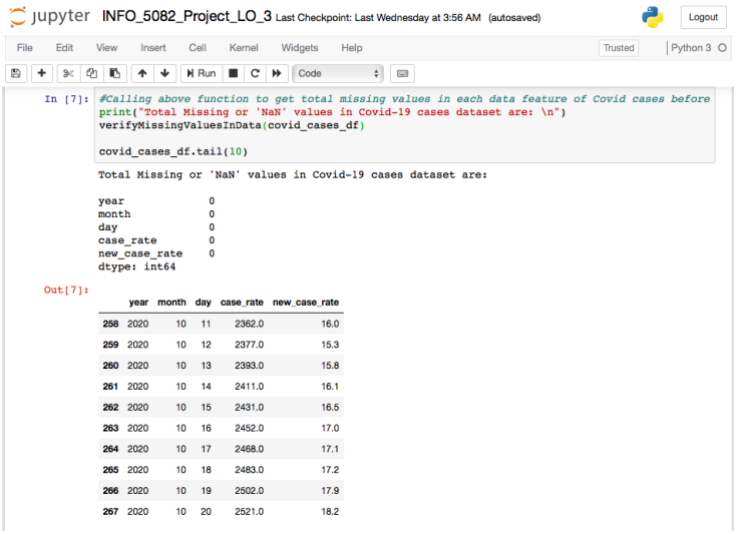
  Verification of null values to rectify and the description of summary are coded as below.

- Google Mobility dataset:



	year	month	day	gps_residential	gps_away_from_home
258	2020	11	11	0.0843	-0.0985
259	2020	11	12	0.0829	-0.0968
260	2020	11	13	0.0857	-0.0995
261	2020	11	14	0.0871	-0.1020
262	2020	11	15	0.0886	-0.1050
263	2020	11	16	0.0886	-0.1050
264	2020	11	17	0.0900	-0.1060
265	2020	11	18	0.0914	-0.1080

- Covid-19 infection rates' dataset:



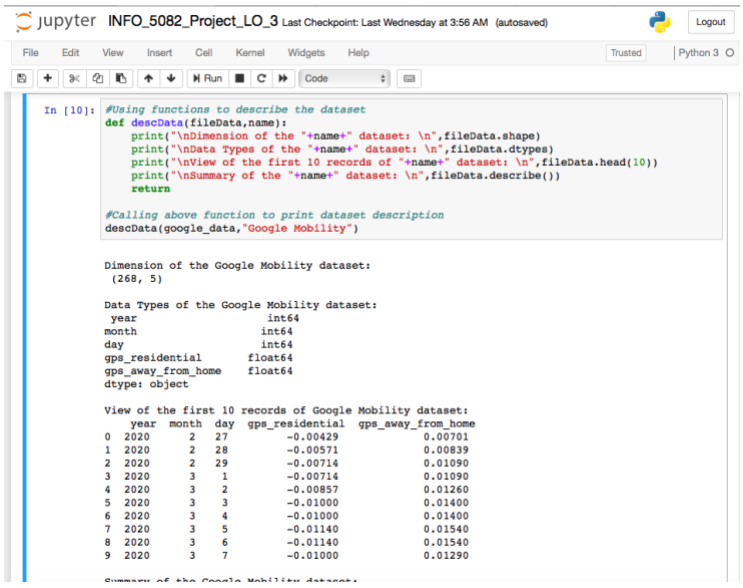
```
In [7]: #Calling above function to get total missing values in each data feature of Covid cases before
        print("Total Missing or 'NaN' values in Covid-19 cases dataset are: \n")
        verifyMissingValuesInData(covid_cases_df)

        covid_cases_df.tail(10)

Total Missing or 'NaN' values in Covid-19 cases dataset are:

year             0
month            0
day              0
case_rate        0
new_case_rate    0
dtype: int64
```

Out[7]:

	year	month	day	case_rate	new_case_rate
258	2020	10	11	2362.0	16.0
259	2020	10	12	2377.0	15.3
260	2020	10	13	2393.0	15.8
261	2020	10	14	2411.0	16.1
262	2020	10	15	2431.0	16.5
263	2020	10	16	2452.0	17.0
264	2020	10	17	2468.0	17.1
265	2020	10	18	2483.0	17.2
266	2020	10	19	2502.0	17.9
267	2020	10	20	2521.0	18.2

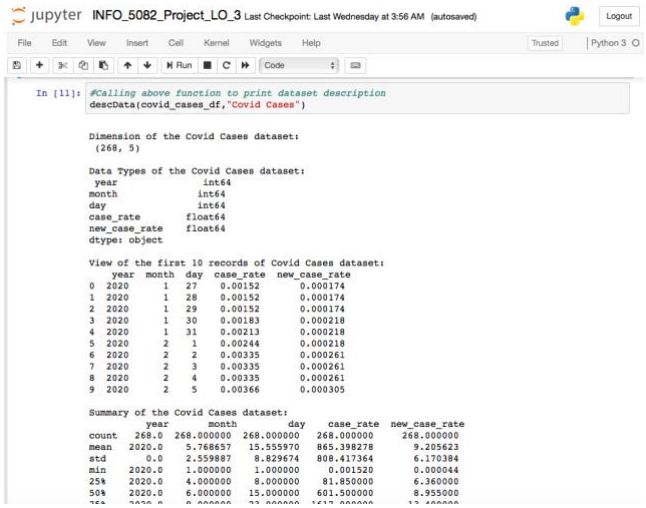
From the output generated, there are no null/missing values in the dataset, so further describing the quantitative summaries as below.

❖ Summary of the Google Mobility dataset:



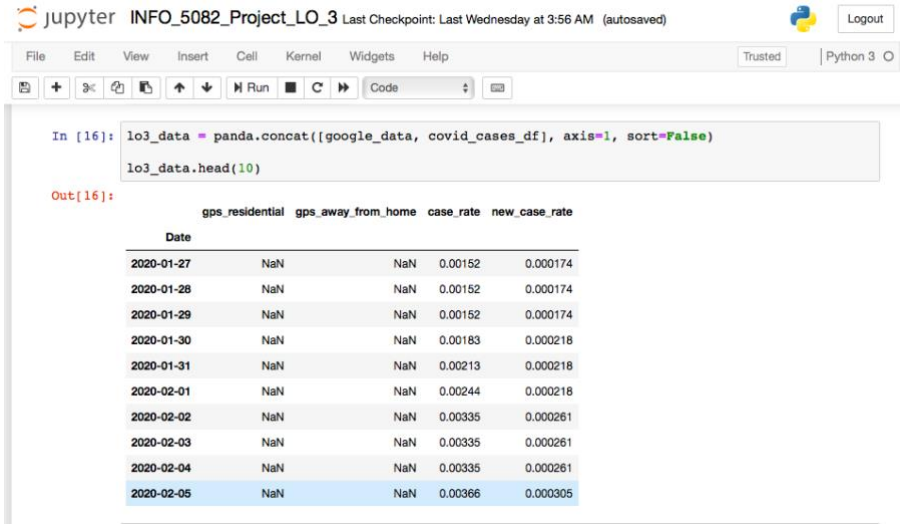
```
In [10]: #Using functions to describe the dataset
         def descData(fileData,name):
             print("\nDimension of the "+name+" dataset: \n",fileData.shape)
             print("\nData Types of the "+name+" dataset: \n",fileData.dtypes)
             print("\nView of the first 10 records of "+name+" dataset: \n",fileData.head(10))
             print("\nSummary of the "+name+" dataset: \n",fileData.describe())
             return

         #Calling above function to print dataset description
         descData(google_data,"Google Mobility")


Dimension of the Google Mobility dataset:
 (268, 5)

Data Types of the Google Mobility dataset:
 year                 int64
month                int64
day                  int64
gps_residential      float64
gps_away_from_home   float64
dtype: object

View of the first 10 records of Google Mobility dataset:
    year  month  day  gps_residential  gps_away_from_home
0  2020      2   27          -0.00429             0.00701
1  2020      2   28          -0.00571             0.00839
2  2020      2   29          -0.00714             0.01090
3  2020      3    1          -0.00714             0.01090
4  2020      3    2          -0.00857             0.01260
5  2020      3    3          -0.01000             0.01400
6  2020      3    4          -0.01000             0.01400
7  2020      3    5          -0.01140             0.01540
8  2020      3    6          -0.01140             0.01540
9  2020      3    7          -0.01000             0.01290
```

❖ Summary of Covid-19 Infection rates' dataset :



```
In [11]: #Calling above function to print dataset description
         descData(covid_cases_df,"Covid Cases")

Dimension of the Covid Cases dataset:
 (268, 5)

Data Types of the Covid Cases dataset:
 year             int64
month             int64
day               int64
case_rate       float64
new_case_rate   float64
dtype: object

View of the first 10 records of Covid Cases dataset:
   year  month  day  case_rate  new_case_rate
0  2020      1   27    0.00152       0.000174
1  2020      1   28    0.00152       0.000174
2  2020      1   29    0.00152       0.000174
3  2020      1   30    0.00183       0.000218
4  2020      1   31    0.00213       0.000218
5  2020      2    1    0.00244       0.000218
6  2020      2    2    0.00335       0.000261
7  2020      2    3    0.00335       0.000261
8  2020      2    4    0.00335       0.000261
9  2020      2    5    0.00366       0.000305

Summary of the Covid Cases dataset:
          year       month         day   case_rate  new_case_rate
count    268.0  268.000000  268.000000  268.000000     268.000000
mean  2020.0    5.768657   15.555970   865.398278       9.205623
std      0.0    2.559887    8.829674   808.417364       6.170384
min   2020.0    1.000000    1.000000     0.001520       0.000044
25%   2020.0    4.000000    8.000000    81.850000       6.360000
50%   2020.0    6.000000   15.000000   601.500000       8.955000
```

The results generated have shown no outliers and these required time series variables from these datasets are combined together for the analysis of third study objective,

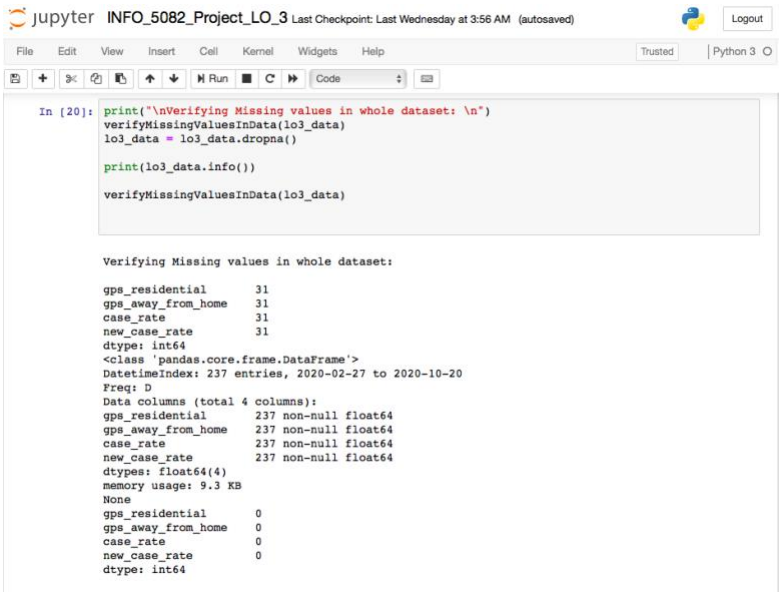


```
In [16]: lo3_data = panda.concat([google_data, covid_cases_df], axis=1, sort=False)
         lo3_data.head(10)
```

Out[16]:

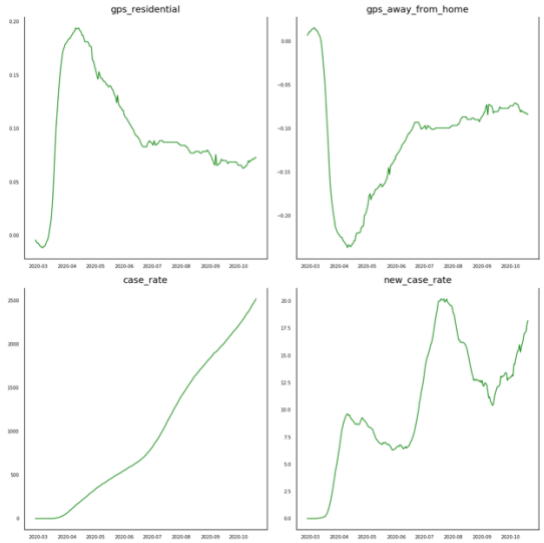
Date	gps_residential	gps_away_from_home	case_rate	new_case_rate
2020-01-27	NaN	NaN	0.00152	0.000174
2020-01-28	NaN	NaN	0.00152	0.000174
2020-01-29	NaN	NaN	0.00152	0.000174
2020-01-30	NaN	NaN	0.00183	0.000218
2020-01-31	NaN	NaN	0.00213	0.000218
2020-02-01	NaN	NaN	0.00244	0.000218
2020-02-02	NaN	NaN	0.00335	0.000261
2020-02-03	NaN	NaN	0.00335	0.000261
2020-02-04	NaN	NaN	0.00335	0.000261
2020-02-05	NaN	NaN	0.00366	0.000305

❖ Null values are rectified as below.



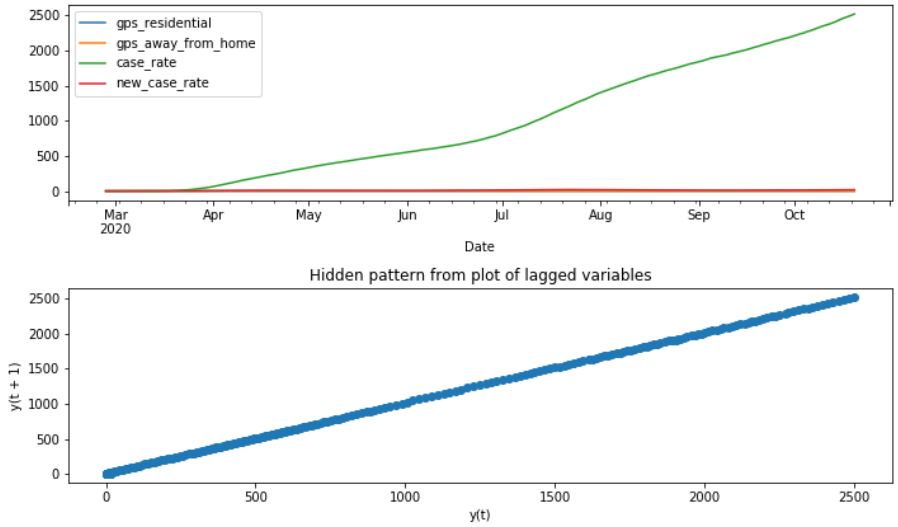
Further dataset is visualized for better understanding,

❖ Univariate plots for each time series variable in the dataset:



We can see the a slightly positive trend pattern in all variables except gps_residential from the month March, 2020.

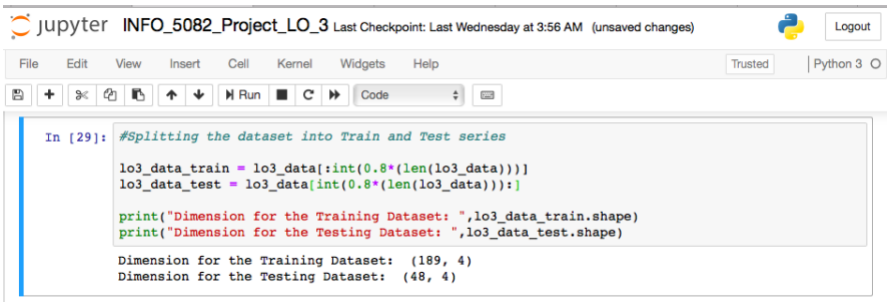
Further to understand the underlying structure pattern in the dataset, lag plot is drawn as below,



From the above plot of lag variables, there are no outliers in the dataset. Also, we can clearly see the positive trend pattern in the dataset meaning each time series variable has linearity with its lagged variable. So, Vector Autocorrelation (VAR) can be applied if there is no cointegration among the variables in dataset.

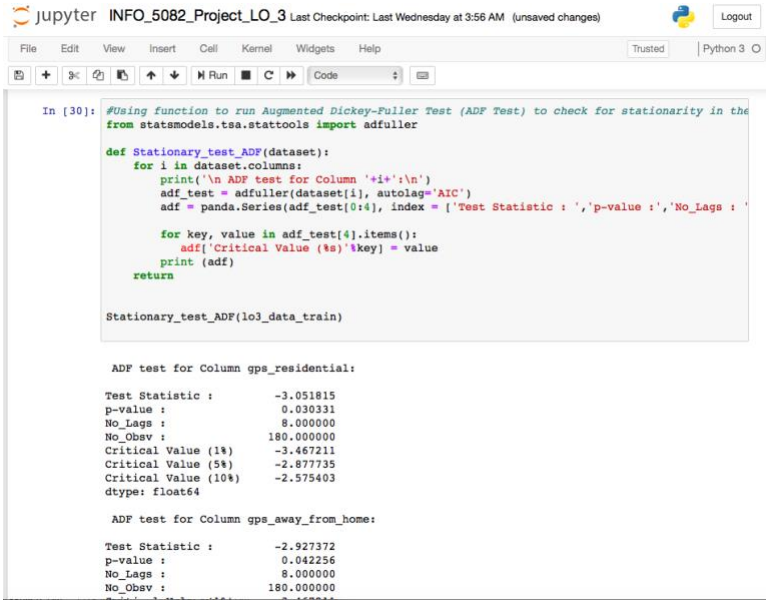
❖ Splitting data into Train and Test data:

In this multivariate time series analysis for third study objective, dataset is split into (0.8) 80% for training and (0.2) 20% for testing as below.



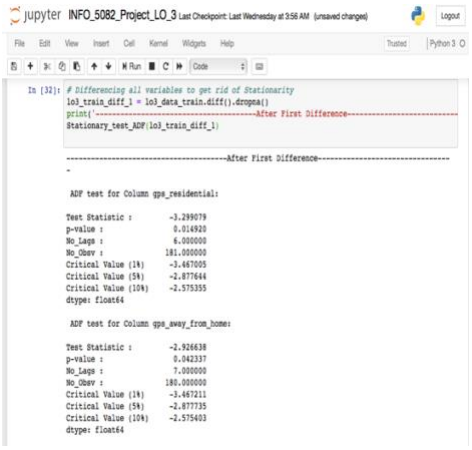
❖ Verifying and converting to stationarity:

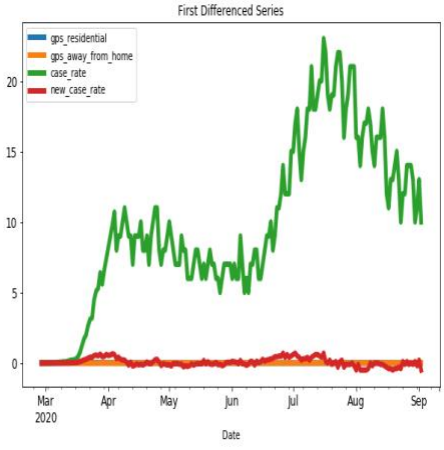
To verify time series property of stationarity in the dataset variables, Augmented Dickey Fuller test function is applied on the train dataset as below.



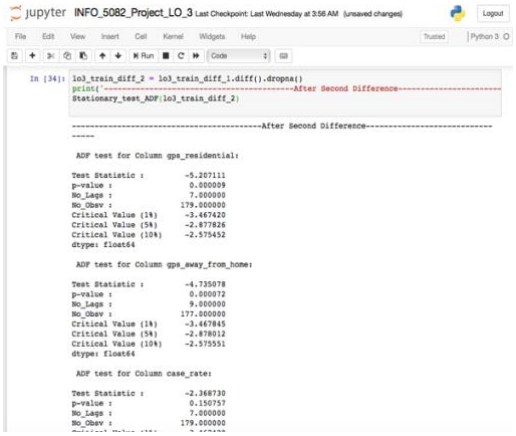
As the results ADF test shown that data is not stationary, to convert the time series dataset into stationary Differencing method has been implemented in three orders.

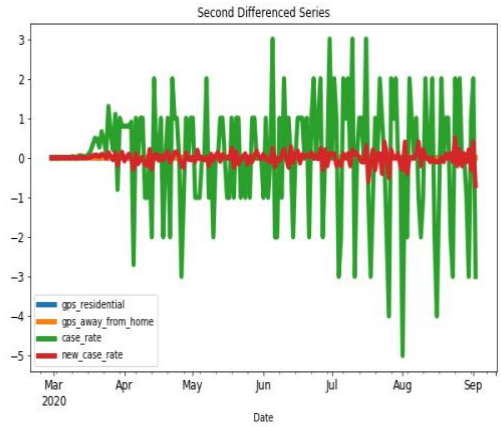
After first differencing, ADF test results shown as non-stationary, so second differencing is applied.



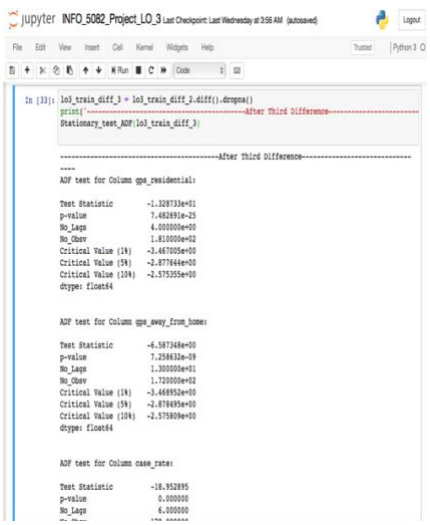


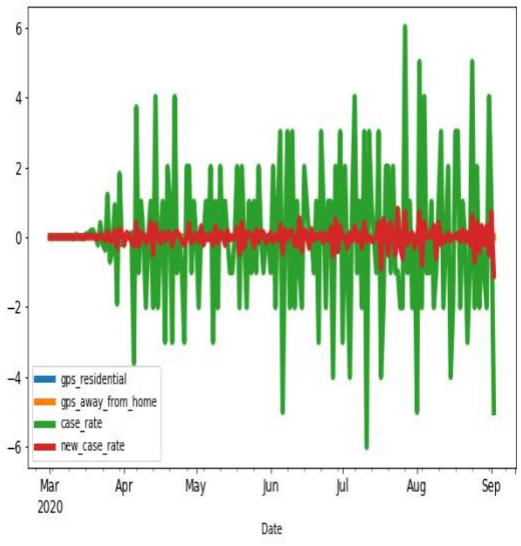
ADF test results from below second difference are also not shown as stationary, so third differencing is performed.





ADF results from third differencing has shown as stationary.

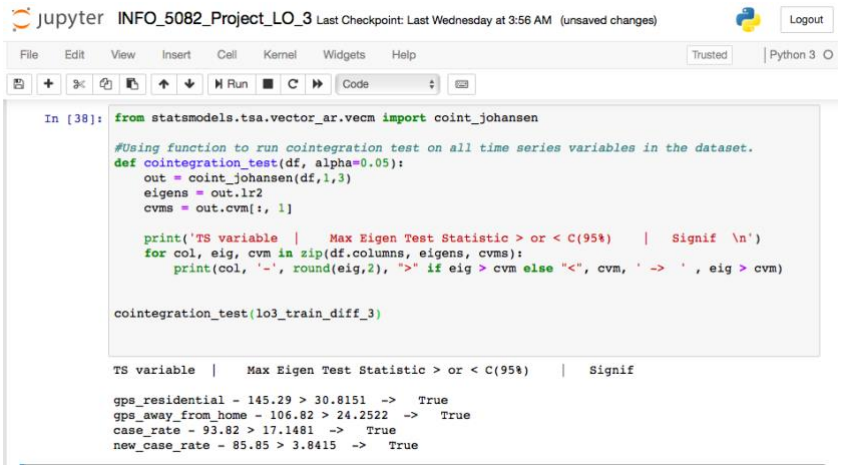




Now the data is in stationary and ready for time series modelling.

❖ Cointegration Testing using Johansen's function:

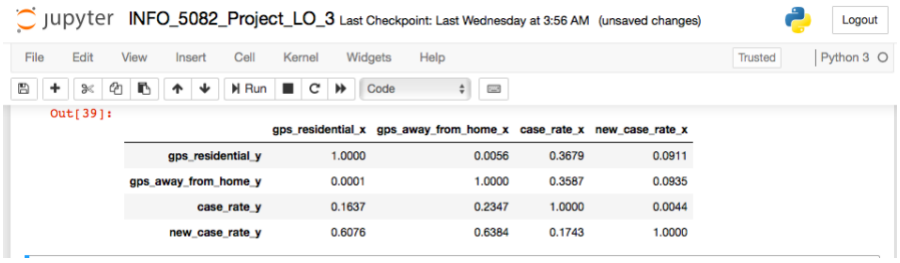
To select the appropriate model, dataset has been tested using Johansen's cointegration among the variables to avoid the information loss taking Eigen Test statistic with 95% confidence interval.



From the results generated, we can see that all the time series variables are cointegrated meaning in long run the time series variables have significant correlations, so this dataset is modelled using VECM model to have little loss of information than when modelled with VAR.

❖ Granger Causality Matrix:

From the granger causality matrix applied we can see that there is no causality or linearity between mobility data columns with covid-19 infection case rates variables, so this dataset is not be useful for analyzing the people's mobility across nationwide with covid-19 case rates.



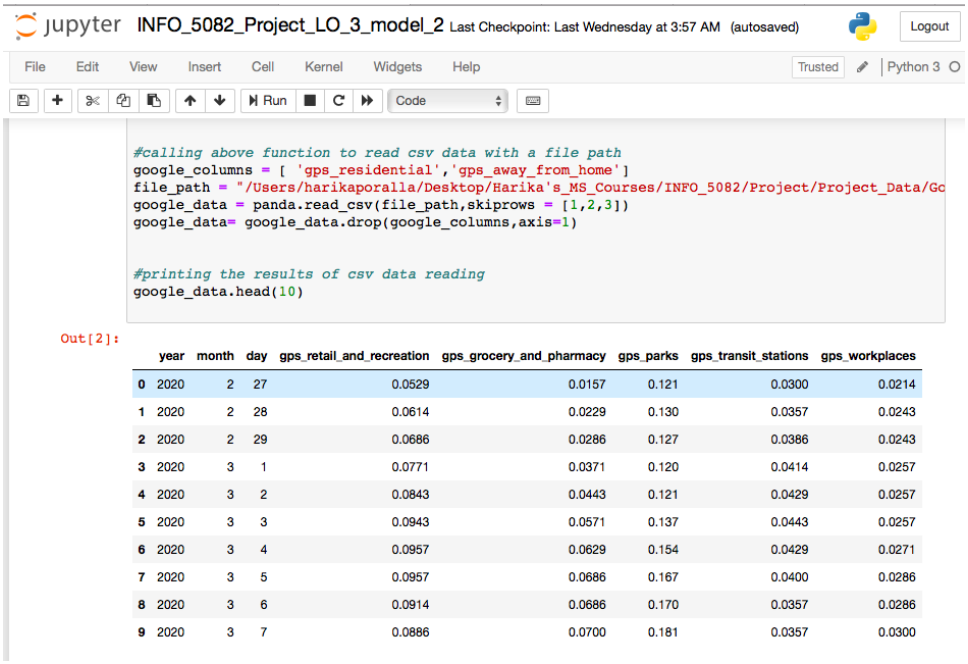
	gps_residential_x	gps_away_from_home_x	case_rate_x	new_case_rate_x
gps_residential_y	1.0000	0.0056	0.3679	0.0911
gps_away_from_home_y	0.0001	1.0000	0.3587	0.0935
case_rate_y	0.1637	0.2347	1.0000	0.0044
new_case_rate_y	0.6076	0.6384	0.1743	1.0000

❖ Model 2:

So, time series variables of people's google mobility in retail and recreation, pharmacy, parks, transit stations and workplaces along with previous covid-19 infection rates' dataset is taken for studying the third objective.

❖ Data preparation and cleaning

- Collection and uploading the google mobility dataset:

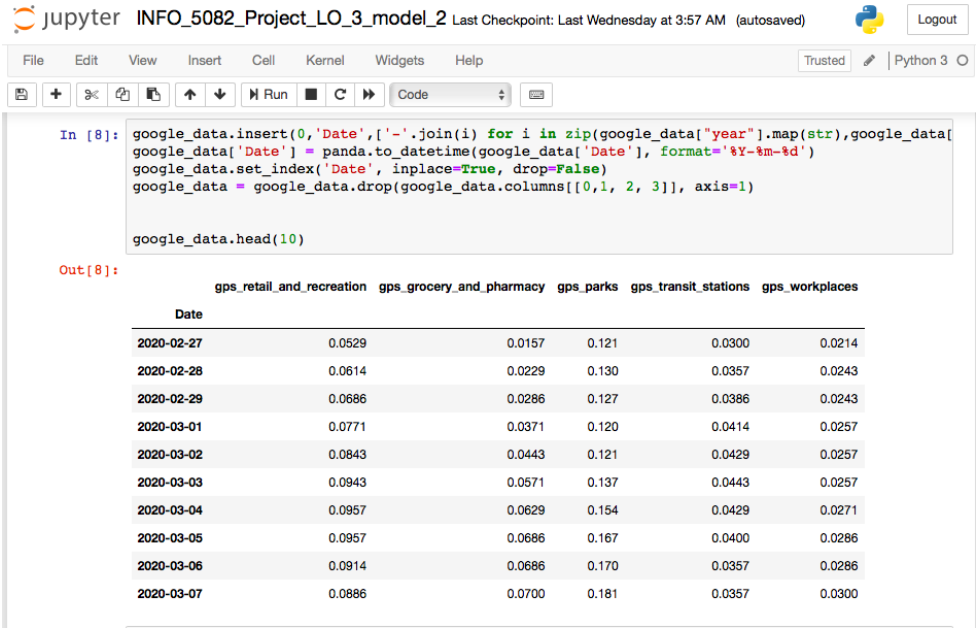



```
#calling above function to read csv data with a file path
google_columns = [ 'gps_residential','gps_away_from_home']
file_path = "/Users/harikaporalla/Desktop/Harika's_MS_Courses/INFO_5082/Project/Project_Data/Go
google_data = panda.read_csv(file_path,skiprows = [1,2,3])
google_data= google_data.drop(google_columns,axis=1)


#printing the results of csv data reading
google_data.head(10)
```

Out[2]:

	year	month	day	gps_retail_and_recreation	gps_grocery_and_pharmacy	gps_parks	gps_transit_stations	gps_workplaces
0	2020	2	27	0.0529	0.0157	0.121	0.0300	0.0214
1	2020	2	28	0.0614	0.0229	0.130	0.0357	0.0243
2	2020	2	29	0.0686	0.0286	0.127	0.0386	0.0243
3	2020	3	1	0.0771	0.0371	0.120	0.0414	0.0257
4	2020	3	2	0.0843	0.0443	0.121	0.0429	0.0257
5	2020	3	3	0.0943	0.0571	0.137	0.0443	0.0257
6	2020	3	4	0.0957	0.0629	0.154	0.0429	0.0271
7	2020	3	5	0.0957	0.0686	0.167	0.0400	0.0286
8	2020	3	6	0.0914	0.0686	0.170	0.0357	0.0286
9	2020	3	7	0.0886	0.0700	0.181	0.0357	0.0300

❖ Datetime indexing for above dataset:



In [8]:
```
google_data.insert(0,'Date',['-'.join(i) for i in zip(google_data["year"].map(str),google_data[
google_data['Date'] = panda.to_datetime(google_data['Date'], format='%Y-%m-%d')
google_data.set_index('Date', inplace=True, drop=False)
google_data = google_data.drop(google_data.columns[[0,1, 2, 3]], axis=1)


google_data.head(10)
```

Out[8]:

Date	gps_retail_and_recreation	gps_grocery_and_pharmacy	gps_parks	gps_transit_stations	gps_workplaces
2020-02-27	0.0529	0.0157	0.121	0.0300	0.0214
2020-02-28	0.0614	0.0229	0.130	0.0357	0.0243
2020-02-29	0.0686	0.0286	0.127	0.0386	0.0243
2020-03-01	0.0771	0.0371	0.120	0.0414	0.0257
2020-03-02	0.0843	0.0443	0.121	0.0429	0.0257
2020-03-03	0.0943	0.0571	0.137	0.0443	0.0257
2020-03-04	0.0957	0.0629	0.154	0.0429	0.0271
2020-03-05	0.0957	0.0686	0.167	0.0400	0.0286
2020-03-06	0.0914	0.0686	0.170	0.0357	0.0286
2020-03-07	0.0886	0.0700	0.181	0.0357	0.0300

EDA

❖ Null values verification:

jupyter INFO_5082_Project_LO_3_model_2 Last Checkpoint: Last Wednesday at 3:57 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [4]:
```
#Using function to print number of missing values in the dataset
def verifyMissingValuesInData(fileData):
    print(fileData.isnull().sum())
    return

#Calling above function to get total missing values in each data feature before cleaning
print("Total Missing or 'NaN' values in Google Mobility Data are: \n")
verifyMissingValuesInData(google_data)

google_data.tail(10)
```

```
Total Missing or 'NaN' values in Google Mobility Data are:

year                          0
month                         0
day                           0
gps_retail_and_recreation     0
gps_grocery_and_pharmacy      0
gps_parks                     0
gps_transit_stations          0
gps_workplaces                0
dtype: int64
```

Out[4]:

	year	month	day	gps_retail_and_recreation	gps_grocery_and_pharmacy	gps_parks	gps_transit_stations	gps_workplaces
258	2020	11	11	-0.173	-0.0814	0.0486	-0.333	-0.261
259	2020	11	12	-0.180	-0.0857	0.0414	-0.333	-0.253
260	2020	11	13	-0.181	-0.0843	0.0171	-0.336	-0.254
261	2020	11	14	-0.184	-0.0843	-0.0129	-0.339	-0.254
262	2020	11	15	-0.187	-0.0843	-0.0429	-0.343	-0.256
263	2020	11	16	-0.189	-0.0829	-0.0529	-0.344	-0.256
264	2020	11	17	-0.191	-0.0829	-0.0671	-0.347	-0.257

- Summary of the above Dataset description:

jupyter INFO_5082_Project_LO_3_model_2 Last Checkpoint: Last Wednesday at 3:57 AM (autosaved) Logout

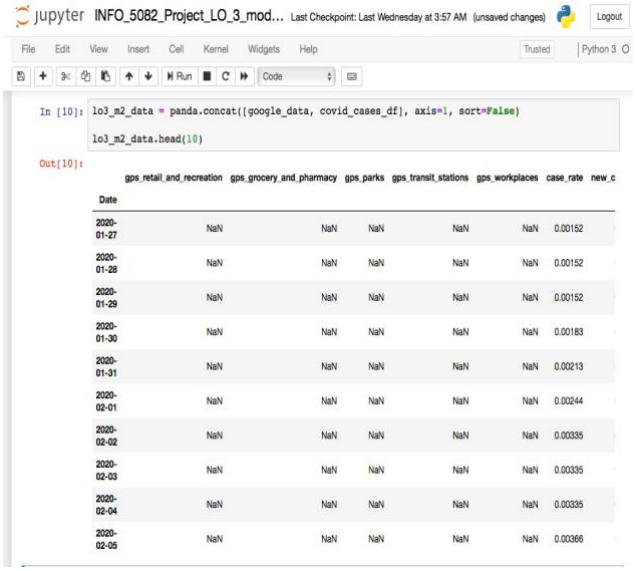
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [6]:
```
#Using functions to describe the dataset
def descData(fileData,name):
    print("\nDimension of the "+name+" dataset: \n",fileData.shape)
    print("\nData Types of the "+name+" dataset: \n",fileData.dtypes)
    print("\nView of the first 10 records of "+name+" dataset: \n",fileData.head(10))
    print("\nSummary of the "+name+" dataset: \n",fileData.describe())
    return

#Calling above function to print dataset description
descData(google_data,"Google Mobility")
```

```
Dimension of the Google Mobility dataset:
 (268, 8)

Data Types of the Google Mobility dataset:
 year                          int64
month                         int64
day                           int64
gps_retail_and_recreation     float64
gps_grocery_and_pharmacy      float64
gps_parks                     float64
gps_transit_stations          float64
gps_workplaces                float64
dtype: object

View of the first 10 records of Google Mobility dataset:
    year  month  day  gps_retail_and_recreation  gps_grocery_and_pharmacy  \
0  2020      2   27                     0.0529                    0.0157
1  2020      2   28                     0.0614                    0.0229
2  2020      2   29                     0.0686                    0.0286
3  2020      3    1                     0.0771                    0.0371
4  2020      3    2                     0.0843                    0.0443
5  2020      3    3                     0.0943                    0.0571
6  2020      3    4                     0.0957                    0.0629
7  2020      3    5                     0.0957                    0.0686
```

Whole combined dataset for analysis and rectifying the null values:



jupyter INFO_5082_Project_LO_3_mod... Last Checkpoint: Last Wednesday at 3:57 AM (unsaved changes) Logout

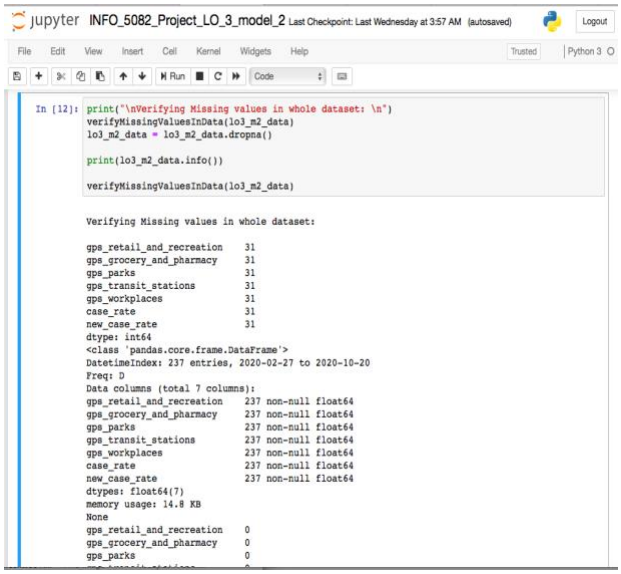
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
In [10]: lo3_m2_data = panda.concat([google_data, covid_cases_df], axis=1, sort=False)

         lo3_m2_data.head(10)
```

Out[10]:

Date	gps_retail_and_recreation	gps_grocery_and_pharmacy	gps_parks	gps_transit_stations	gps_workplaces	case_rate	new_c
2020-01-27	NaN	NaN	NaN	NaN	NaN	0.00152	
2020-01-28	NaN	NaN	NaN	NaN	NaN	0.00152	
2020-01-29	NaN	NaN	NaN	NaN	NaN	0.00152	
2020-01-30	NaN	NaN	NaN	NaN	NaN	0.00183	
2020-01-31	NaN	NaN	NaN	NaN	NaN	0.00213	
2020-02-01	NaN	NaN	NaN	NaN	NaN	0.00244	
2020-02-02	NaN	NaN	NaN	NaN	NaN	0.00335	
2020-02-03	NaN	NaN	NaN	NaN	NaN	0.00335	
2020-02-04	NaN	NaN	NaN	NaN	NaN	0.00335	
2020-02-05	NaN	NaN	NaN	NaN	NaN	0.00366	



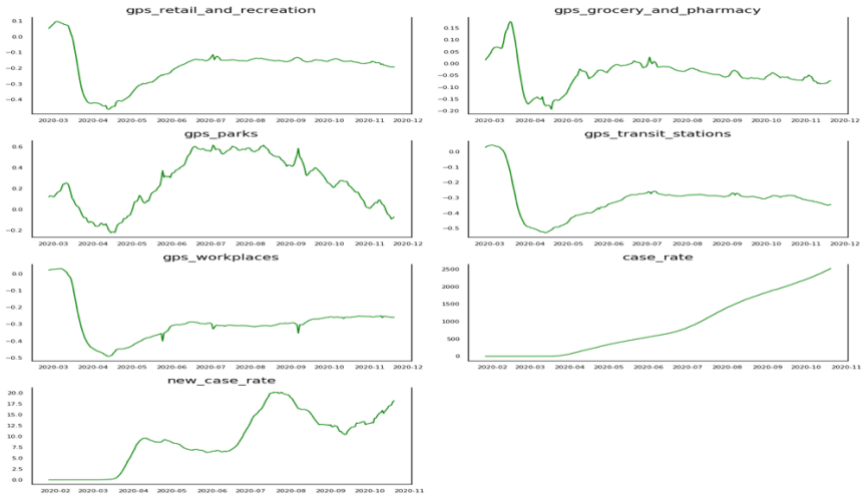
jupyter INFO_5082_Project_LO_3_model_2 Last Checkpoint: Last Wednesday at 3:57 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

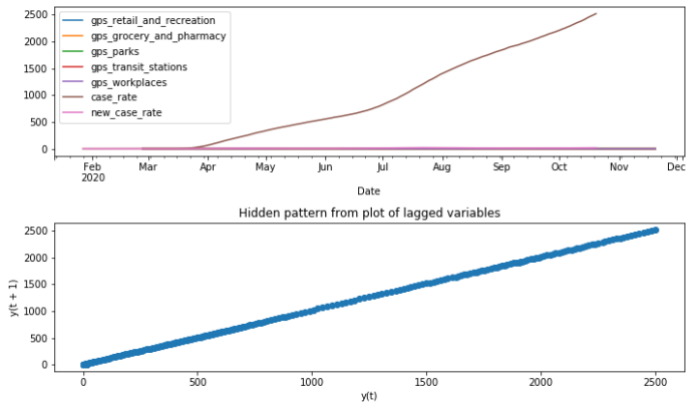
```
In [12]: print("\nVerifying Missing values in whole dataset: \n")
         verifyMissingValuesInData(lo3_m2_data)
         lo3_m2_data = lo3_m2_data.dropna()

         print(lo3_m2_data.info())

         verifyMissingValuesInData(lo3_m2_data)

         Verifying Missing values in whole dataset:

         gps_retail_and_recreation    31
         gps_grocery_and_pharmacy     31
         gps_parks                    31
         gps_transit_stations         31
         gps_workplaces               31
         case_rate                    31
         new_case_rate                31
         dtype: int64
         <class 'pandas.core.frame.DataFrame'>
         DatetimeIndex: 237 entries, 2020-02-27 to 2020-10-20
         Freq: D
         Data columns (total 7 columns):
         gps_retail_and_recreation    237 non-null float64
         gps_grocery_and_pharmacy     237 non-null float64
         gps_parks                    237 non-null float64
         gps_transit_stations         237 non-null float64
         gps_workplaces               237 non-null float64
         case_rate                    237 non-null float64
         new_case_rate                237 non-null float64
         dtypes: float64(7)
         memory usage: 14.8 KB
         None
         gps_retail_and_recreation    0
         gps_grocery_and_pharmacy     0
         gps_parks                    0
```

- Univariate plots of each time series variables:



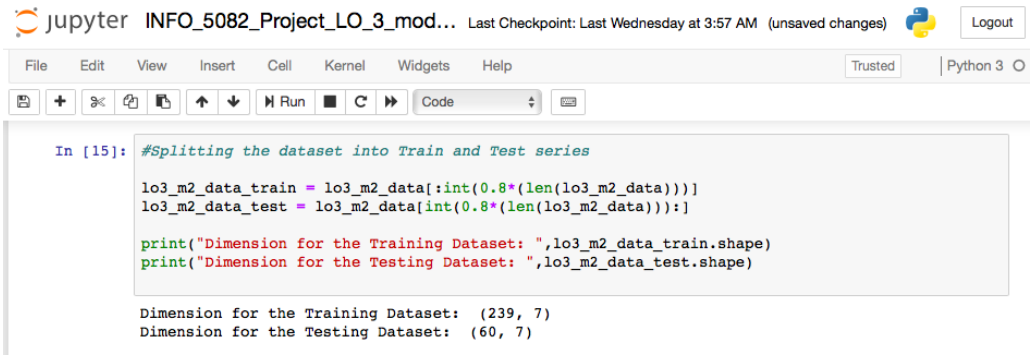
We can see the a slightly positive trend pattern in all variables except gps_residential from the month April, 2020.

- Plotting the lagged variables:



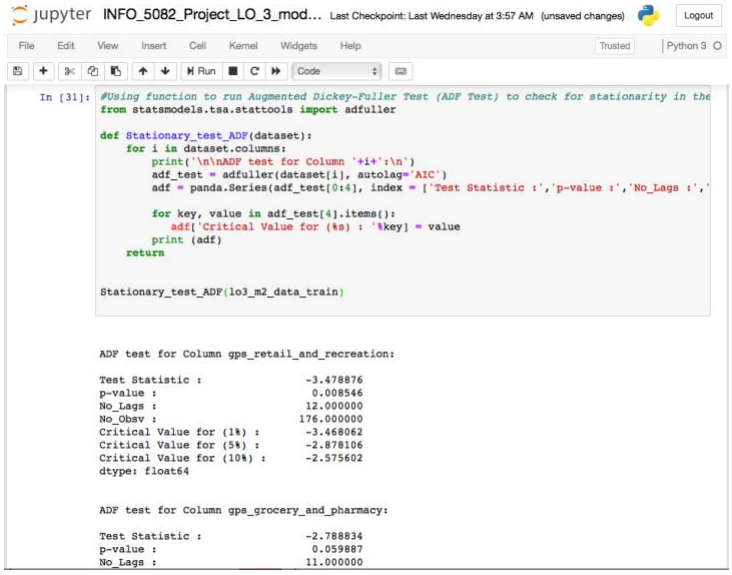
From the above plot of lag variables, there are no outliers in the dataset. Also, he positive trend pattern in the dataset shows the linearity with of time series variables with its lagged variable. Vector Autocorrelation (VAR) will be applied if there is no cointegration among the variables in dataset.

❖ Splitting of the dataset into training and testing



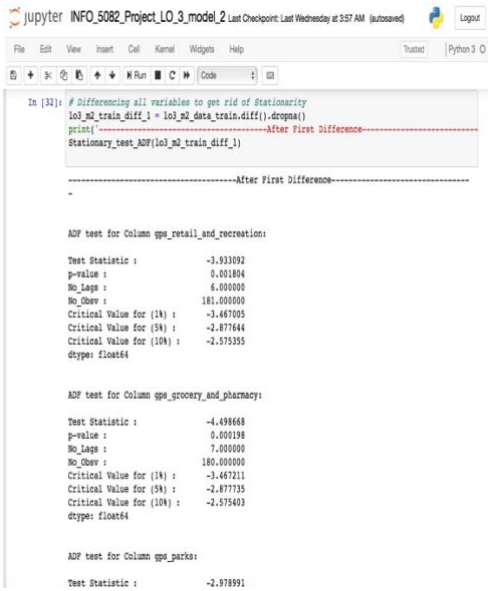
❖ Verifying and converting to stationarity:

Using Augmented Dickey Fuller test function to test the stationarity in the train dataset as below.

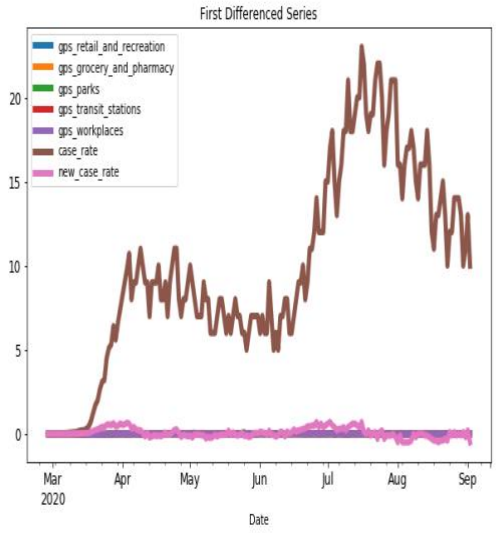


ADF test results have shown that data is not stationary, so dataset is converted into stationary using Differencing method in three orders.

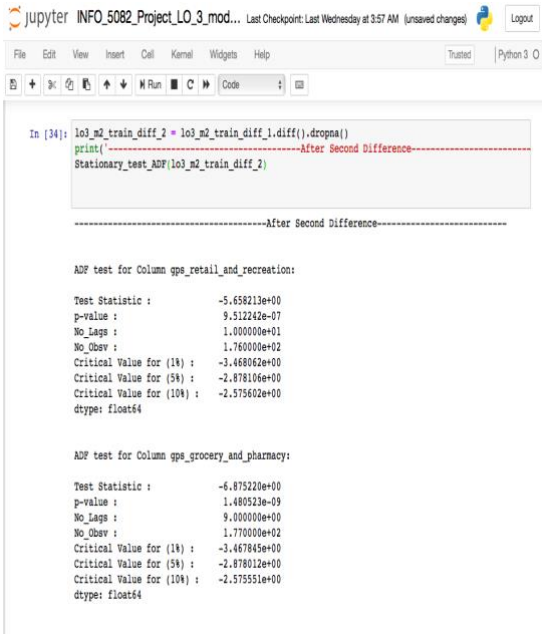
But first differencing ADF test results also have shown as non-stationary, so second differencing is applied.



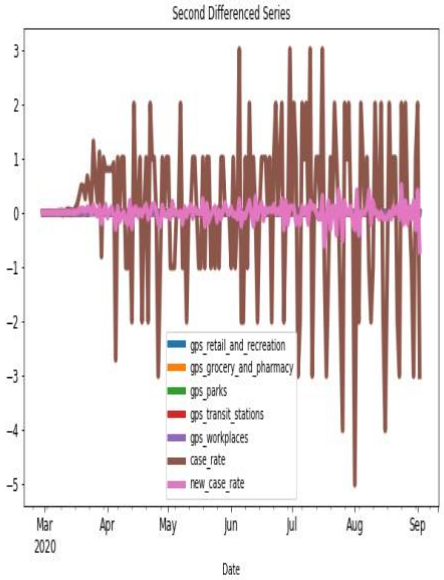
```
In [32]: # Differencing all variables to get rid of Stationarity
         lo3_m2_train_diff_1 = lo3_m2_data_train.diff().dropna()
         print('----------After First Difference----------')
         Stationary_test_ADF(lo3_m2_train_diff_1)

----------After First Difference----------

ADF test for Column gps_retail_and_recreation:

Test Statistic :              -3.933092
p-value :                      0.001804
No_Lags :                      6.000000
No_Obsv :                    181.000000
Critical Value for (1%) :     -3.467005
Critical Value for (5%) :     -2.877644
Critical Value for (10%) :    -2.575355
dtype: float64

ADF test for Column gps_grocery_and_pharmacy:

Test Statistic :              -4.498668
p-value :                      0.000198
No_Lags :                      7.000000
No_Obsv :                    180.000000
Critical Value for (1%) :     -3.467211
Critical Value for (5%) :     -2.877735
Critical Value for (10%) :    -2.575403
dtype: float64

ADF test for Column gps_parks:

Test Statistic :              -2.978991
```



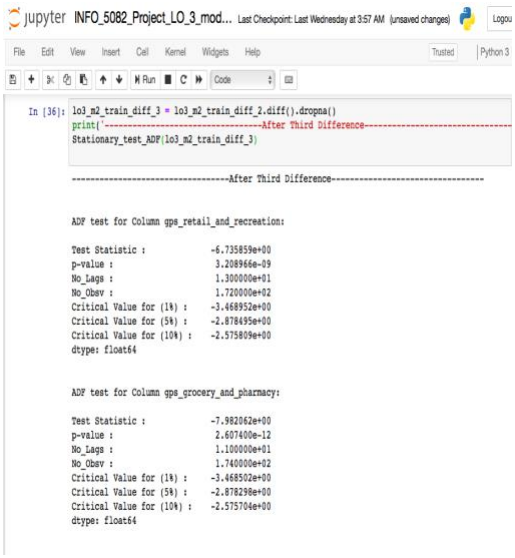
Second differenced ADF test results from below also shown as non-stationary, so third differencing is applied.



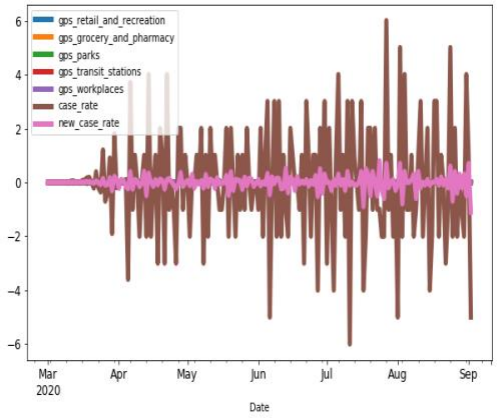
```
In [34]: lo3_m2_train_diff_2 = lo3_m2_train_diff_1.diff().dropna()
         print('----------After Second Difference----------')
         Stationary_test_ADF(lo3_m2_train_diff_2)

----------After Second Difference----------

ADF test for Column gps_retail_and_recreation:

Test Statistic :              -5.658213e+00
p-value :                      9.512242e-07
No_Lags :                      1.000000e+01
No_Obsv :                      1.760000e+02
Critical Value for (1%) :     -3.468062e+00
Critical Value for (5%) :     -2.878106e+00
Critical Value for (10%) :    -2.575602e+00
dtype: float64

ADF test for Column gps_grocery_and_pharmacy:

Test Statistic :              -6.875220e+00
p-value :                      1.480523e-09
No_Lags :                      9.000000e+00
No_Obsv :                      1.770000e+02
Critical Value for (1%) :     -3.467845e+00
Critical Value for (5%) :     -2.878012e+00
Critical Value for (10%) :    -2.575551e+00
dtype: float64
```



And ADF test results from third differencing has shown as stationary.



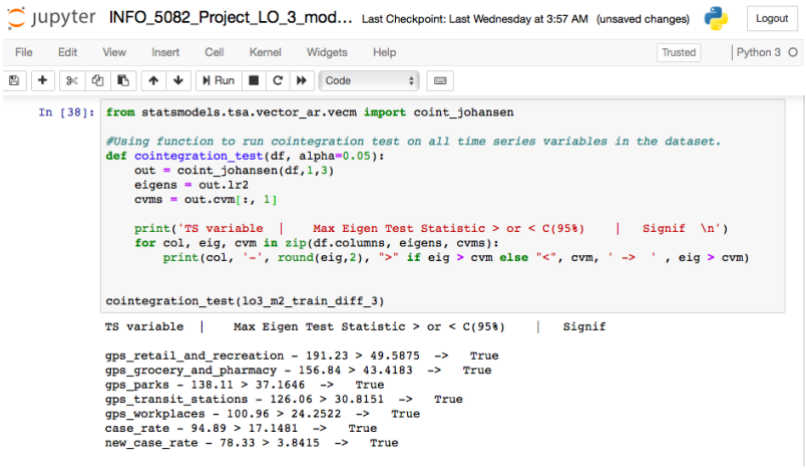
```
In [36]: lo3_m2_train_diff_3 = lo3_m2_train_diff_2.diff().dropna()
         print('---------------------------------After Third Difference---------------------------------')
         Stationary_test_ADF(lo3_m2_train_diff_3)

---------------------------------After Third Difference---------------------------------


ADF test for Column gps_retail_and_recreation:

Test Statistic :            -6.735859e+00
p-value :                    3.208966e-09
No_Lags :                    1.300000e+01
No_Obsv :                    1.720000e+02
Critical Value for (1%) :   -3.468952e+00
Critical Value for (5%) :   -2.878495e+00
Critical Value for (10%) :  -2.575809e+00
dtype: float64


ADF test for Column gps_grocery_and_pharmacy:

Test Statistic :            -7.982062e+00
p-value :                    2.607400e-12
No_Lags :                    1.100000e+01
No_Obsv :                    1.740000e+02
Critical Value for (1%) :   -3.468502e+00
Critical Value for (5%) :   -2.878298e+00
Critical Value for (10%) :  -2.575704e+00
dtype: float64
```



Now the data is in stationary and ready for time series modelling.

❖ Cointegration Testing using Johansen's function:

Dataset has been tested using Johansen's cointegration among the variables to avoid the information loss taking Eigen Test statistic with 95% confidence interval.

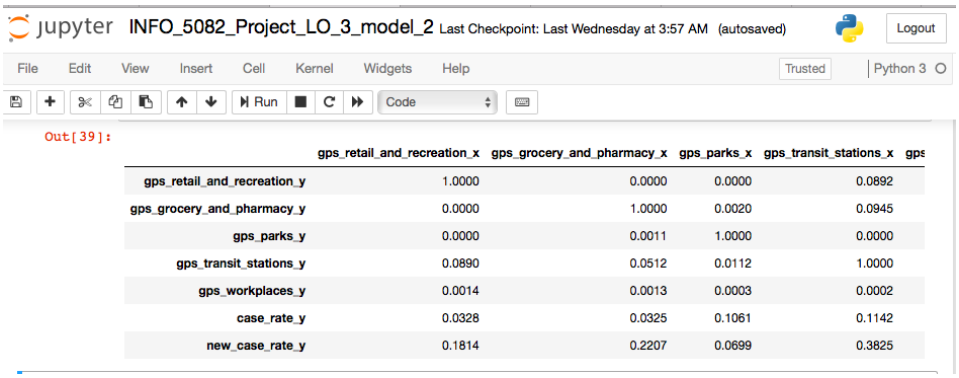


```
In [38]: from statsmodels.tsa.vector_ar.vecm import coint_johansen

         #Using function to run cointegration test on all time series variables in the dataset.
         def cointegration_test(df, alpha=0.05):
             out = coint_johansen(df,1,3)
             eigens = out.lr2
             cvms = out.cvm[:, 1]

             print('TS variable   |    Max Eigen Test Statistic > or < C(95%)    |   Signif  \n')
             for col, eig, cvm in zip(df.columns, eigens, cvms):
                 print(col, '-', round(eig,2), ">" if eig > cvm else "<", cvm, ' ->  ' , eig > cvm)


         cointegration_test(lo3_m2_train_diff_3)

TS variable   |    Max Eigen Test Statistic > or < C(95%)    |   Signif

gps_retail_and_recreation - 191.23 > 49.5875  ->   True
gps_grocery_and_pharmacy - 156.84 > 43.4183  ->   True
gps_parks - 138.11 > 37.1646  ->   True
gps_transit_stations - 126.06 > 30.8151  ->   True
gps_workplaces - 100.96 > 24.2522  ->   True
case_rate - 94.89 > 17.1481  ->   True
new_case_rate - 78.33 > 3.8415  ->   True
```

From above results, we can see that all the time series variables are cointegrated meaning in long run the time series variables have significant correlations, so this dataset is modelled using VECM model to have avoid the loss of information occurred VAR modeling.

❖ Granger Causality Matrix:

Granger's causality matrix is applied and observed some causalities or linearity between mobility data columns with covid-19 infection case rate variables, so this dataset is modelled using VECM for the analysis.



	gps_retail_and_recreation_x	gps_grocery_and_pharmacy_x	gps_parks_x	gps_transit_stations_x	gps
gps_retail_and_recreation_y	1.0000	0.0000	0.0000	0.0892	
gps_grocery_and_pharmacy_y	0.0000	1.0000	0.0020	0.0945	
gps_parks_y	0.0000	0.0011	1.0000	0.0000	
gps_transit_stations_y	0.0890	0.0512	0.0112	1.0000	
gps_workplaces_y	0.0014	0.0013	0.0003	0.0002	
case_rate_y	0.0328	0.0325	0.1061	0.1142	
new_case_rate_y	0.1814	0.2207	0.0699	0.3825	

Data Analytics

Time series is defined as a sequential value of a variable arranged in an orderly fashion spaced at equal time intervals. Mostly applied for Stock Market Forecasting, Workload Projections, Sales Forecasting and Economic Forecasting, etc., to get deep insights from underlying patterns in the data and also for model fitting for forecasting and monitoring.

There are so many techniques available for time series analysis, of which three main techniques to model fit a time series analysis are Box-Jenkins ARIMA Models, Box-Jenkins Multivariate Models and Holt-Winters Exponential Smoothing. Selection of the technique or method appropriate for analysis depends on the user's application and goals.

The AutoRegressive Moving Average Vector or VAR model can be applied for multivariate time series analysis where each times series variable is the vector of number of other times series variables in the dataset. Each variable in this analysis has a linearity with its own past lag values and also with the past lags of other variables.

VAR model with no mean and stationary multivariate time series is given as, $x_t=(x_{1t},x_{2t},x_{3t},\dots\dots,x_{nt})^T$, $-\infty < t < \infty$ (or) $x_t= \phi_1 x_{t-1} + \phi_2 x_{t-2} + \dots\dots\dots + \phi_p x_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \dots\dots - \theta_q x_{t-q}$, where x_t and a_t are n x 1 column vectors and a_t represents the random component called white noise.

Here, $\phi_k = \{\phi_{k.jj}\}$ where k = 1,2, …, p (order of lags) and $\theta_k = \{\theta_{k.jj}\}$ where k = 1,2, …, q are the n x n autoregressive and moving average parameters. In above equation for each variable error or residual are represented by the past lag values of other variables so the these correlations can in general be spurious correlations because of any causal relationship.

When variables in a VAR are cointegrated we apply Vector error correction (VEC) model and VECM for two variables x and y is given as,

$\Delta y_t = \beta_{y0} + \beta_{y1}\Delta y_{t-1} \ldots\ldots\ldots + \beta_{yp}\Delta y_{t-p} + \gamma_{y1}\Delta x_{t-1} + \cdots + \gamma_{yp}\Delta x_{t-p} - \lambda_y(y_{t-1} - \propto_0 - \propto_1 x_{t-1}) + v_t^y$ and

$\Delta x_t = \beta_{x0} + \beta_{x1}\Delta x_{t-1} \ldots\ldots\ldots + \beta_{xp}\Delta x_{t-p} + \gamma_{x1}\Delta y_{t-1} + \cdots + \gamma_{xp}\Delta y_{t-p} - \lambda_x(y_{t-1} - \propto_0 - \propto_1 x_{t-1}) + v_t^x$,

Here, $\lambda_x \; and \; \lambda_y$ are the error correction params that define how the variables and y react in long run and $y_t = \propto_0 + \propto_1 x_t$ is the cointegration relationship .

For identifying the structural shocks and their dynamic additional assumptions are required. VAR can be used for two variables systems without further assumptions and can find the causality direction between x and y using granger's causality analysis. But VAR when applied for more than two variables and one lag the computational can be very complex. Thus, the automated tasks from python functions can make this task easy.

Granger's Causality: Granger's concept that "Correlation does not imply Causality" shows the directions and significance of causal relationships between variables x and y. It's null hypothesis is stated as "x does not cause y" which can be test with p-value significance.

In this project, to analyze each study objective, multivariate time series analysis is performed by selecting appropriate models from VAR and VECM for fitting as per EDA results on taken data.

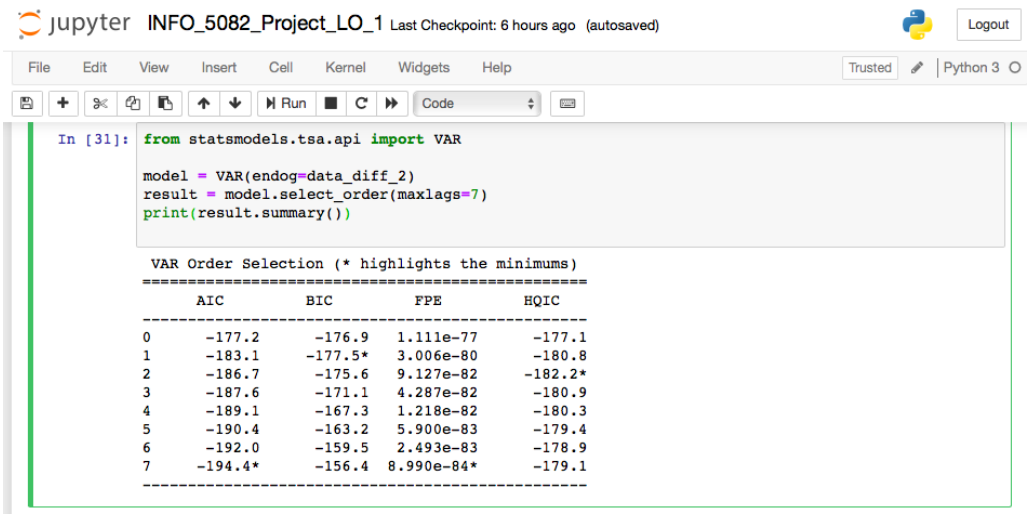
Data analysis for small business revenue

For the first study objective to analyze the small business revenues after the EDA results bothe the multivariate Autoregression model (VAR) and vector error correction models are used for model fitting to get the model with better accuracy.

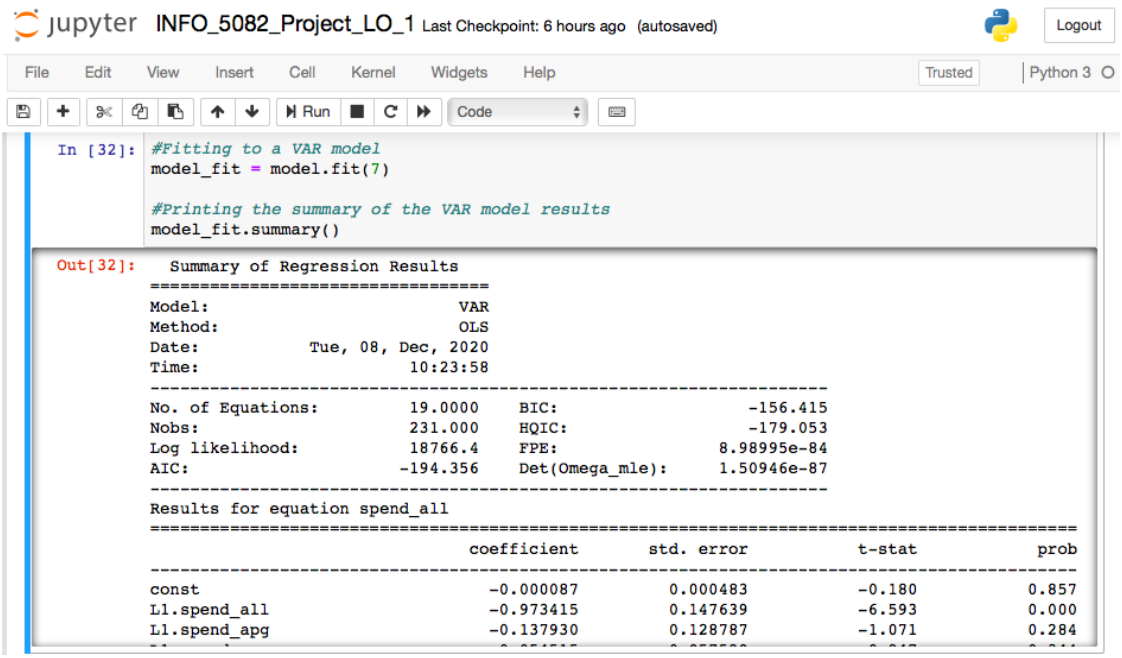
❖ VAR model Analysis:

To gain the minimal information loss, Akaike Information criteria (AIC) is used in selecting the small order of lag for model fitting to forecast.

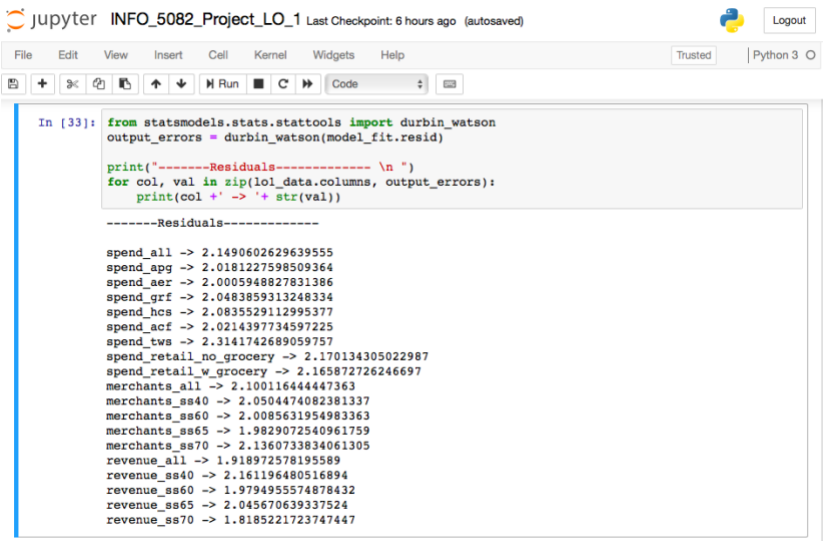
VAR model is iteratively fitted with increasing lag order till maxlag = 7 to pick the order with minimum AIC, as below



From the results, model with lag order = 7 has the minimal AIC values. So, the VAR model is fitted with ‘lag order = 7’ and summary of output can be seen as below.



To check the correlation in the error residuals, durbin_watson function is used and the results are generated as below.

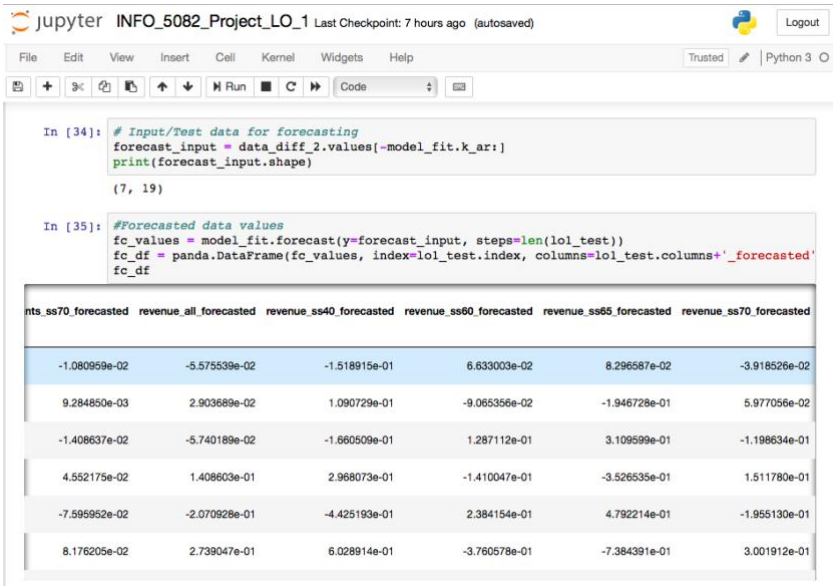


As per the durbin_watson residuals, if the error values are close to 4 then a negative correlation is left in the data, when the error values are close to 0 implies a positive correlation while error values close to 2 then there is no serial correlation in the dataset.

From above, the residuals are observed to close to 2 meaning there is no serial correlation pattern left in the dataset for further analysis.

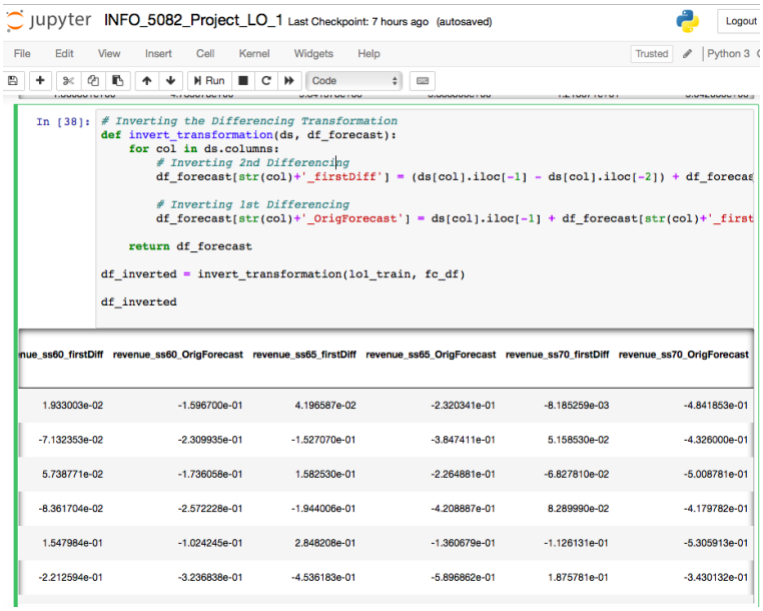
❖ Model Forecasting the test outputs:

VAR model is provided with input data for essential past lag values and the forecasted data values generated below are based on differenced train data.



nts_ss70_forecasted	revenue_all_forecasted	revenue_ss40_forecasted	revenue_ss60_forecasted	revenue_ss65_forecasted	revenue_ss70_forecasted
-1.080959e-02	-5.575539e-02	-1.518915e-01	6.633003e-02	8.296587e-02	-3.918526e-02
9.284850e-03	2.903689e-02	1.090729e-01	-9.065356e-02	-1.946728e-01	5.977056e-02
-1.408637e-02	-5.740189e-02	-1.660509e-01	1.287112e-01	3.109599e-01	-1.198634e-01
4.552175e-02	1.408603e-01	2.968073e-01	-1.410047e-01	-3.526535e-01	1.511780e-01
-7.595952e-02	-2.070928e-01	-4.425193e-01	2.384154e-01	4.792214e-01	-1.955130e-01
8.176205e-02	2.739047e-01	6.028914e-01	-3.760578e-01	-7.384391e-01	3.001912e-01

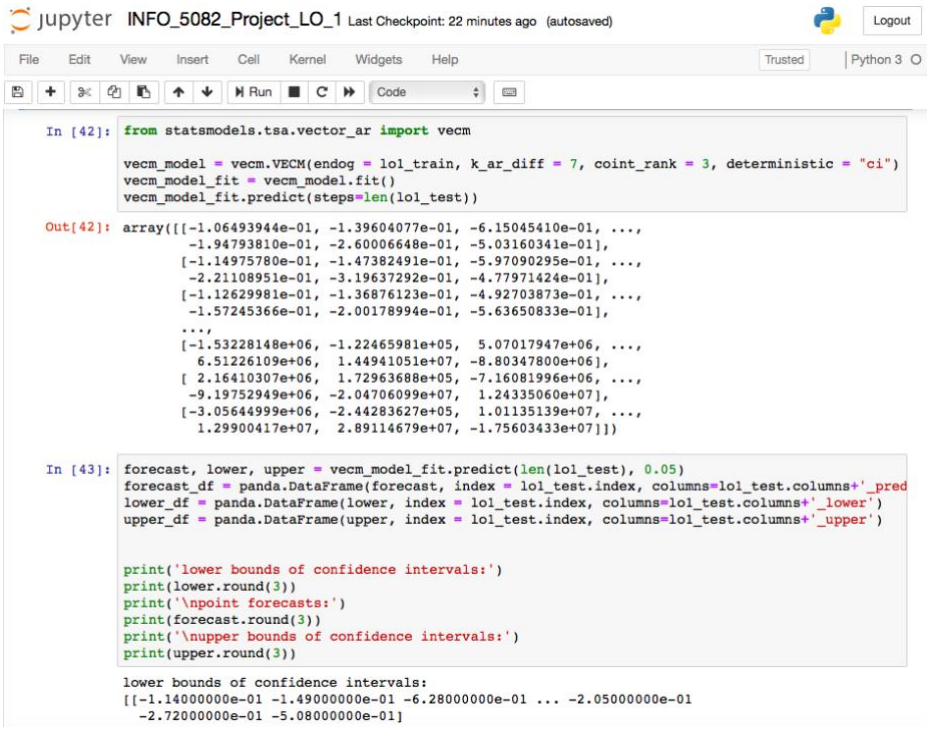
So the values are inverted to remove the applied differencing to get original forecasted values as below



nue_ss60_firstDiff	revenue_ss60_OrigForecast	revenue_ss65_firstDiff	revenue_ss65_OrigForecast	revenue_ss70_firstDiff	revenue_ss70_OrigForecast
1.933003e-02	-1.596700e-01	4.196587e-02	-2.320341e-01	-8.185259e-03	-4.841853e-01
-7.132353e-02	-2.309935e-01	-1.527070e-01	-3.847411e-01	5.158530e-02	-4.326000e-01
5.738771e-02	-1.736058e-01	1.582530e-01	-2.264881e-01	-6.827810e-02	-5.008781e-01
-8.361704e-02	-2.572228e-01	-1.944006e-01	-4.208887e-01	8.289990e-02	-4.179782e-01
1.547984e-01	-1.024245e-01	2.848208e-01	-1.360679e-01	-1.126131e-01	-5.305913e-01
-2.212594e-01	-3.236838e-01	-4.536183e-01	-5.896862e-01	1.875781e-01	-3.430132e-01

❖ VECM Model Analysis:

Vector error correction model is fitted on the train data using lag order = 7 and cointegration rank = 3 with constant deterministic as the data has constant linearity pattern and the lower, upper and forecast values are predicted for test data with 0.05 significance as below,

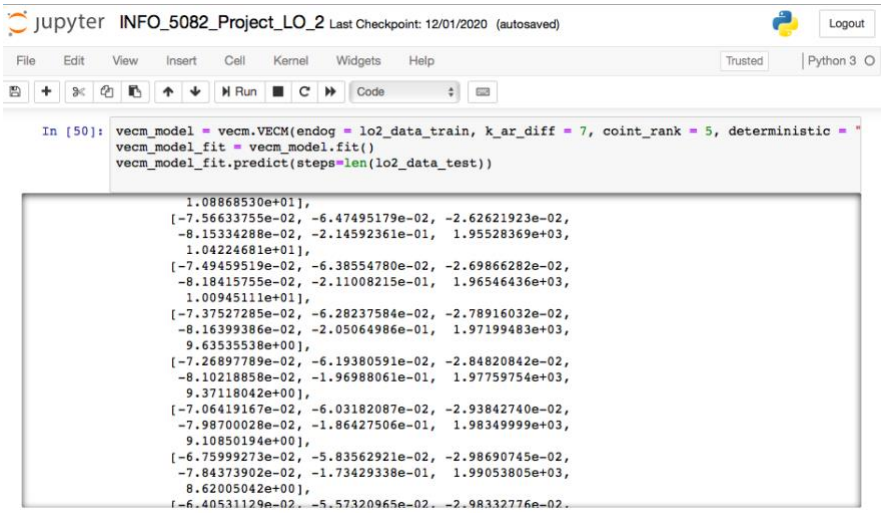


Data Analysis for Employment Rate estimation

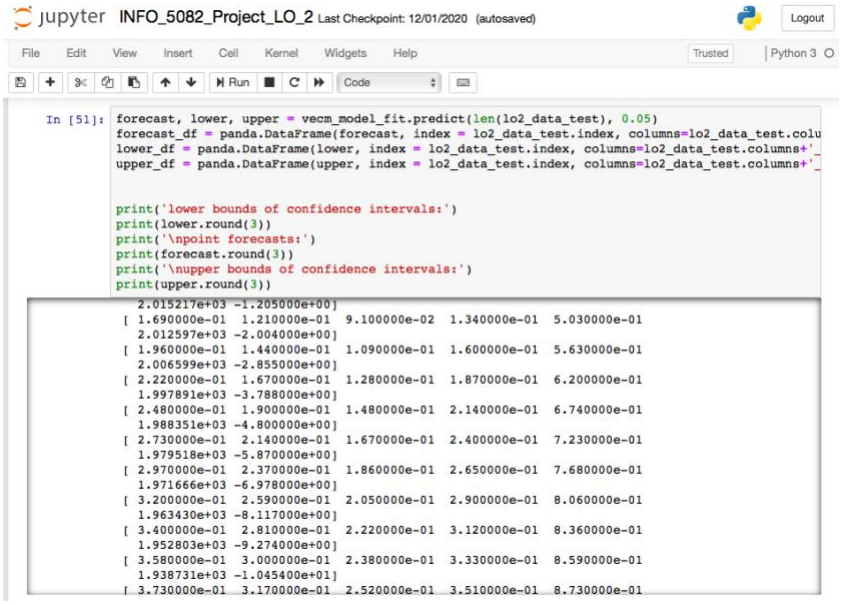
For the second study objective to analyze the estimate employment rates with covid-19 cases since the EDA results has shown linear pattern in the data and presence of cointegration among time series variables vector error correction model (VECM) is used for model fitting to get the model with no information loss and better accuracy.

❖ VECM Model Analysis:

vecm model from ‘statmodels.tsa.vector_ar’ is fitted on the train data using lag order = 7 and cointegration rank = 5 with constant deterministic as the data has constant linearity pattern and forested values as below,



The predicted lower, upper forecasted values for test data with 0.05 significance are given as below,

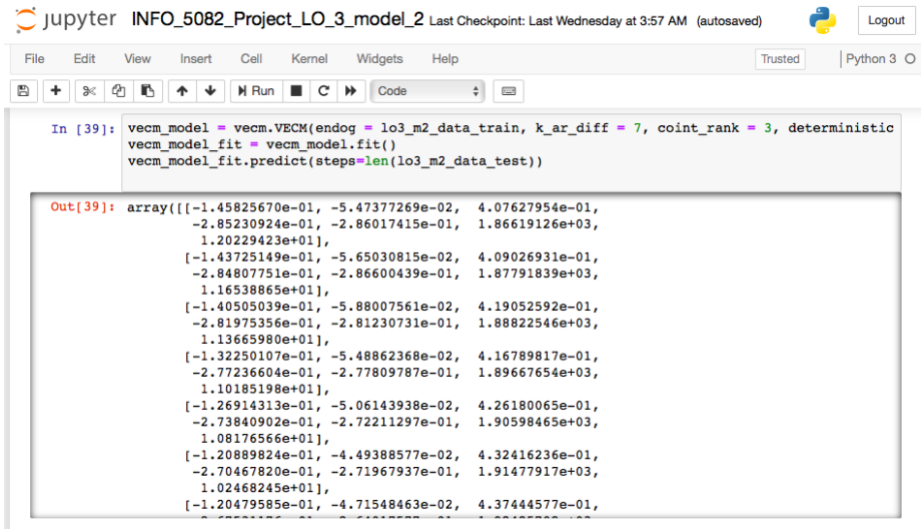


❖ Data Analysis for covid-19 infection rates with google mobility:

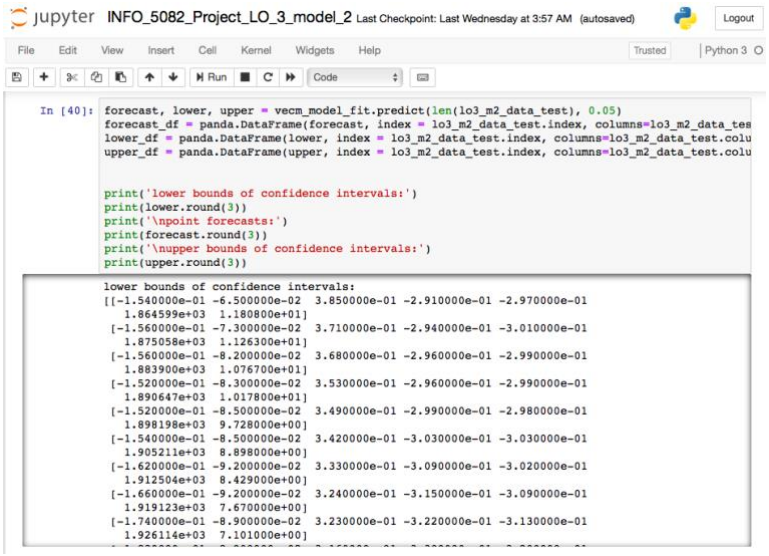
In the third study objective to analyze the covid-19 infection rates with people google mobility data, since the EDA results has shown linear pattern in the data and presence of cointegration among time series variables, again vector error correction model (VECM) is applied for model fitting to get the predictions with no information loss and better accuracy.

❖ VECM Model Analysis:

vecm model from 'statmodels.tsa.vector_ar' is fitted on the train data using lag order = 7 and cointegration rank = 3 with constant deterministic as the constant linearity pattern is observed in dataset and forecasted values as below,



Also predicted lower, upper forecasted values for test data with 0.05 significance are given as below,

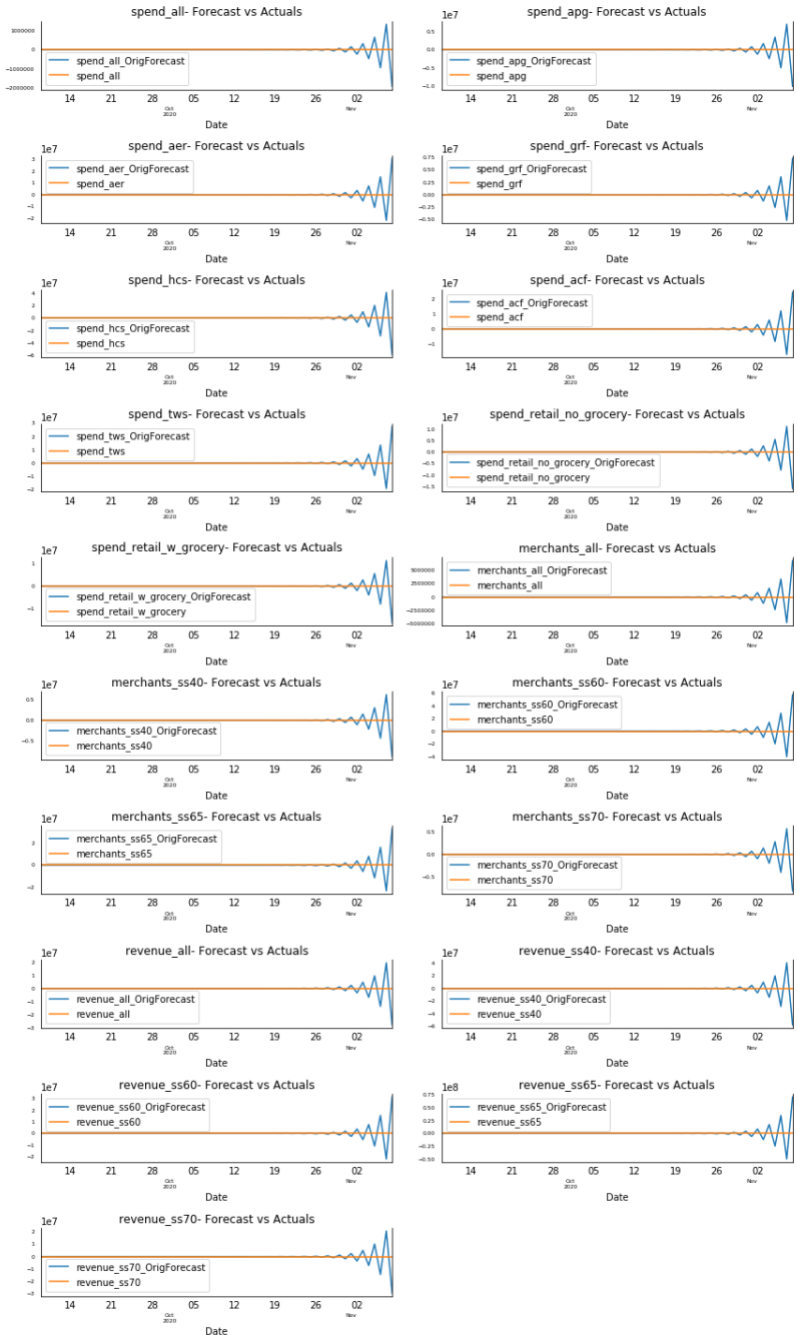


Data Visualization and Results Reports

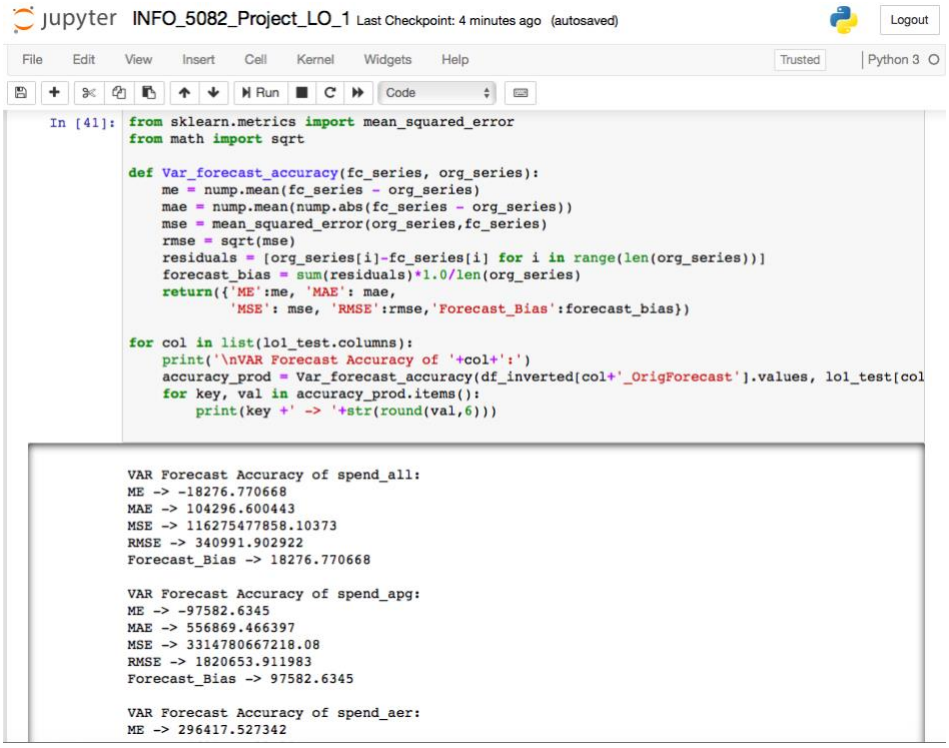
Data visualizations for first study objective – analyzing the small business revenues

❖ VAR model visualizations and evaluations:

- Forecasted values and Actual values are plotted as below,



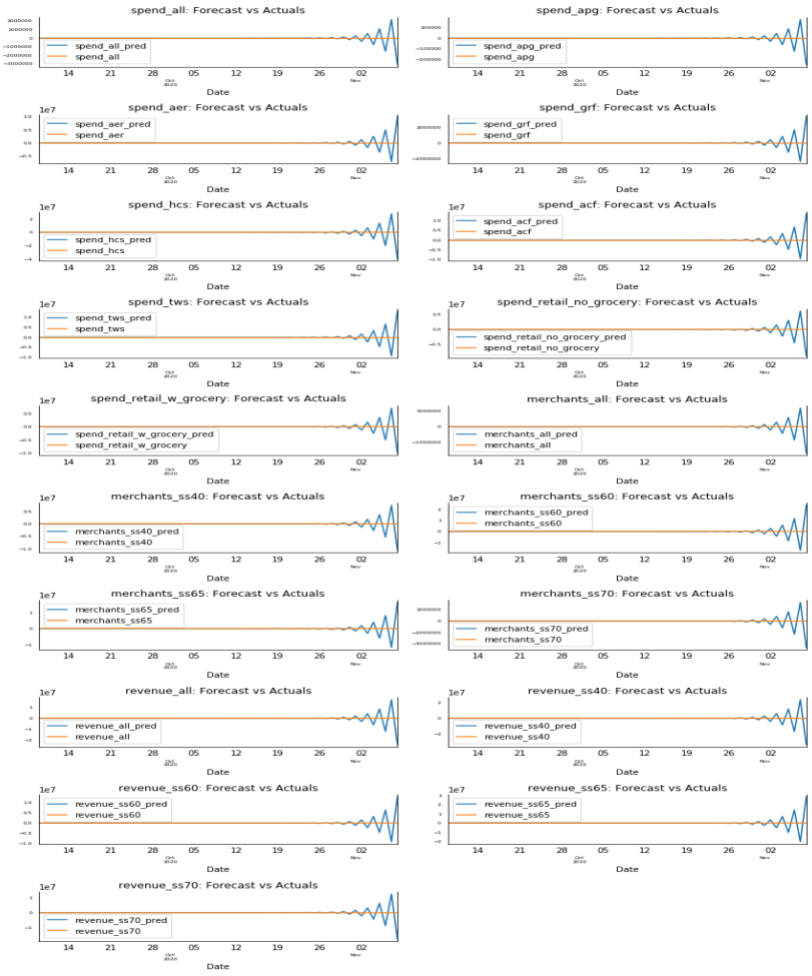
- Evaluation of forecasted values for each time series variable is given as coded as below,



Mean Forecast error (ME), Mean absolute error (MAE), Mean square error (MSE), root mean square error (RMSE) and bias are calculated and for ‘revenue_all’ RMSE = 5000351.65 and bias = 268006.16. So overall, this is not good model and the positive error from bias says that model has been under forecasted.

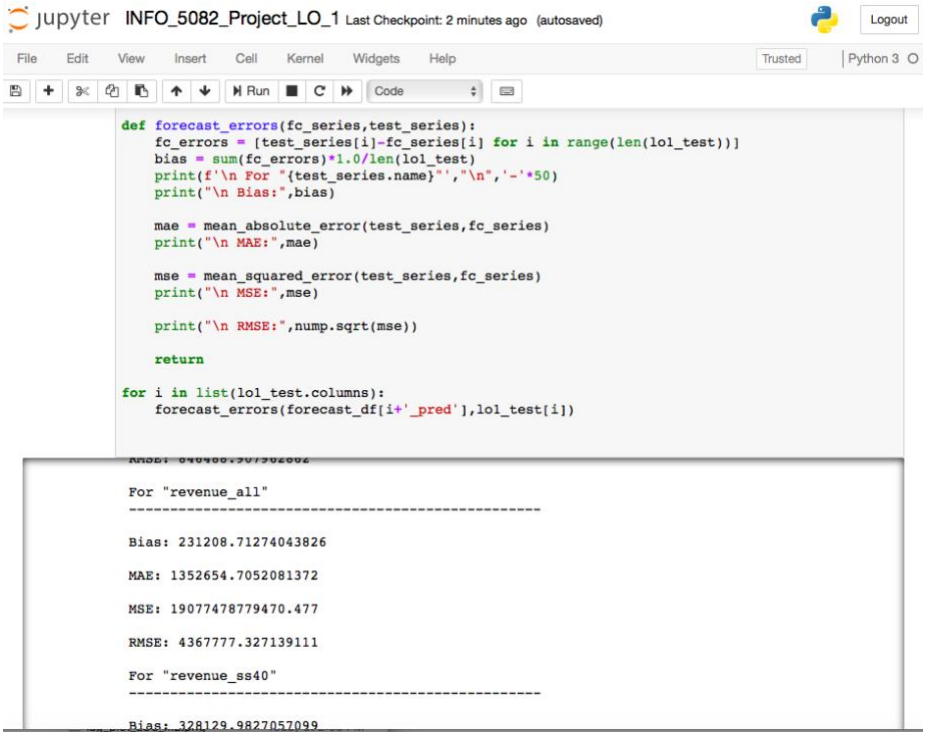
❖ VECM model visualizations and evaluations:

- Forecasted and actual values are plotted as below,



From the above graphs we cannot see much difference in forecasted values from VAR model.

Performing model evaluation using MAE, MSE , RMSE and bias as below.

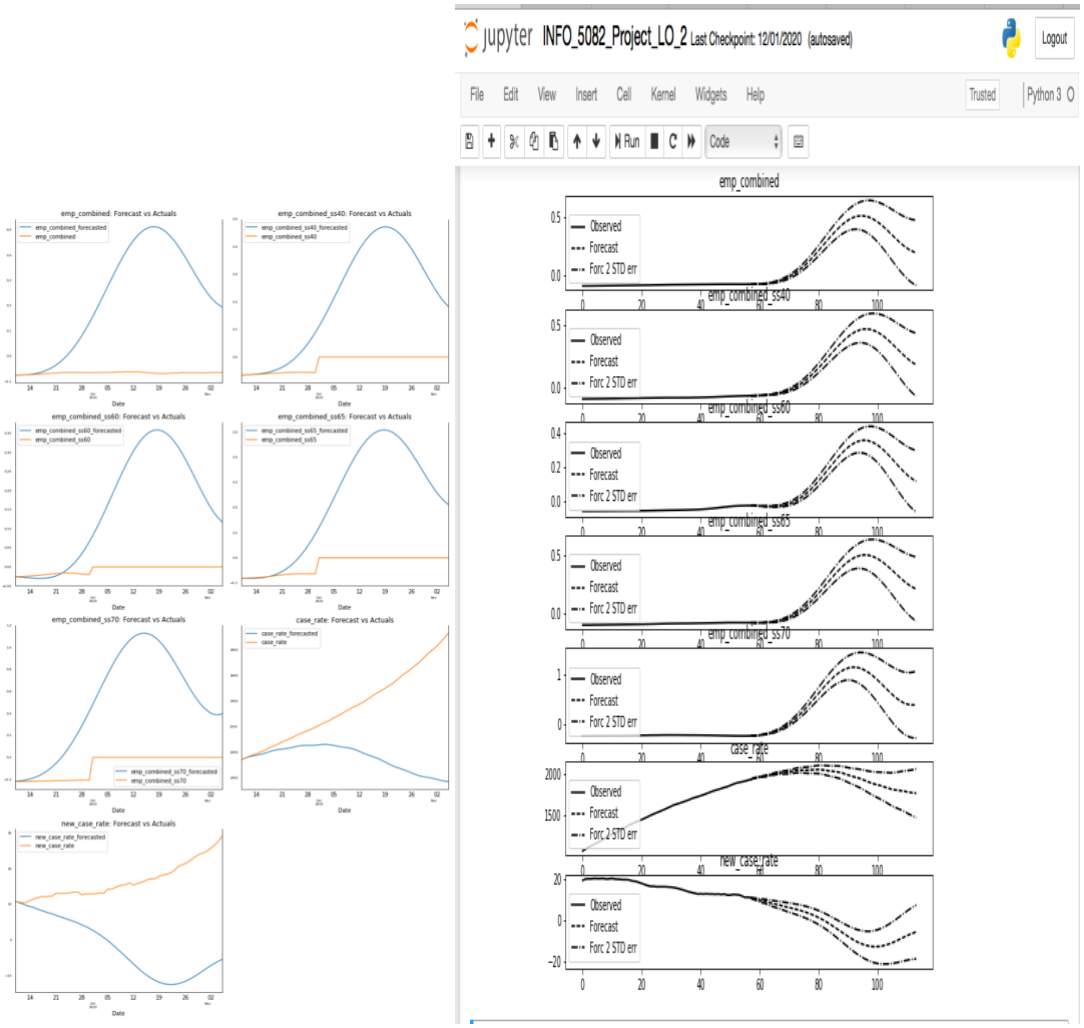


Considering the same ‘revenue_all’ variable, RMSE = 4367777.33 and bias = 231208.71 says the VECM model is relative better for analyzing the overall small business revenue.

Data visualizations for second study objective – estimation of employment rates

❖ VECM model visualizations and evaluation:

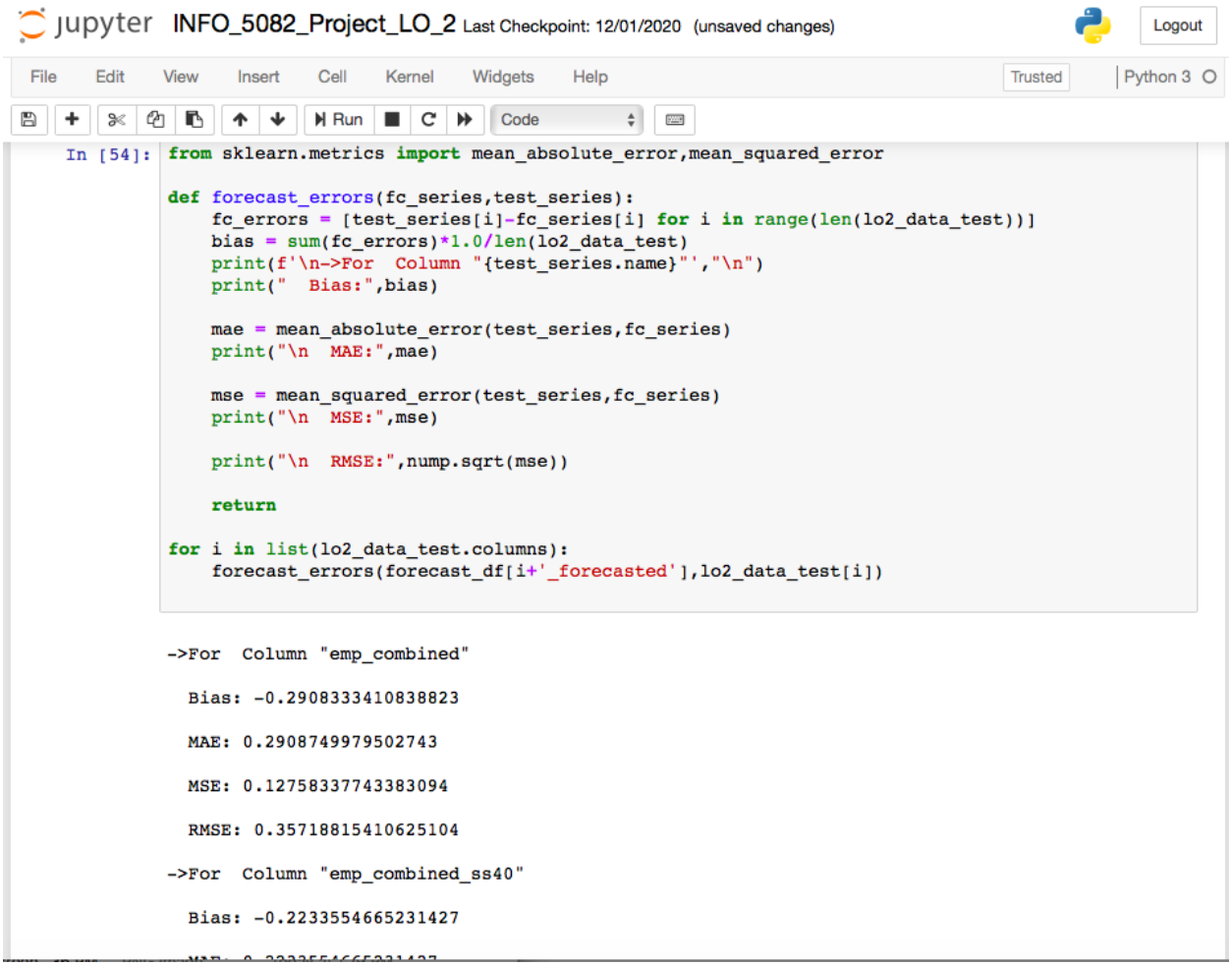
- Forecasted and actual values are plotted and also the upper and lower values of forecasted values are shown as below,



From the above graphs we can see that output error rate is much higher. And upper and lower forecasted values within standard deviations is predicted as the significance give is 0.05.

- Evaluating the forecast:

Performing model evaluation using MAE, MSE , RMSE and bias as below.

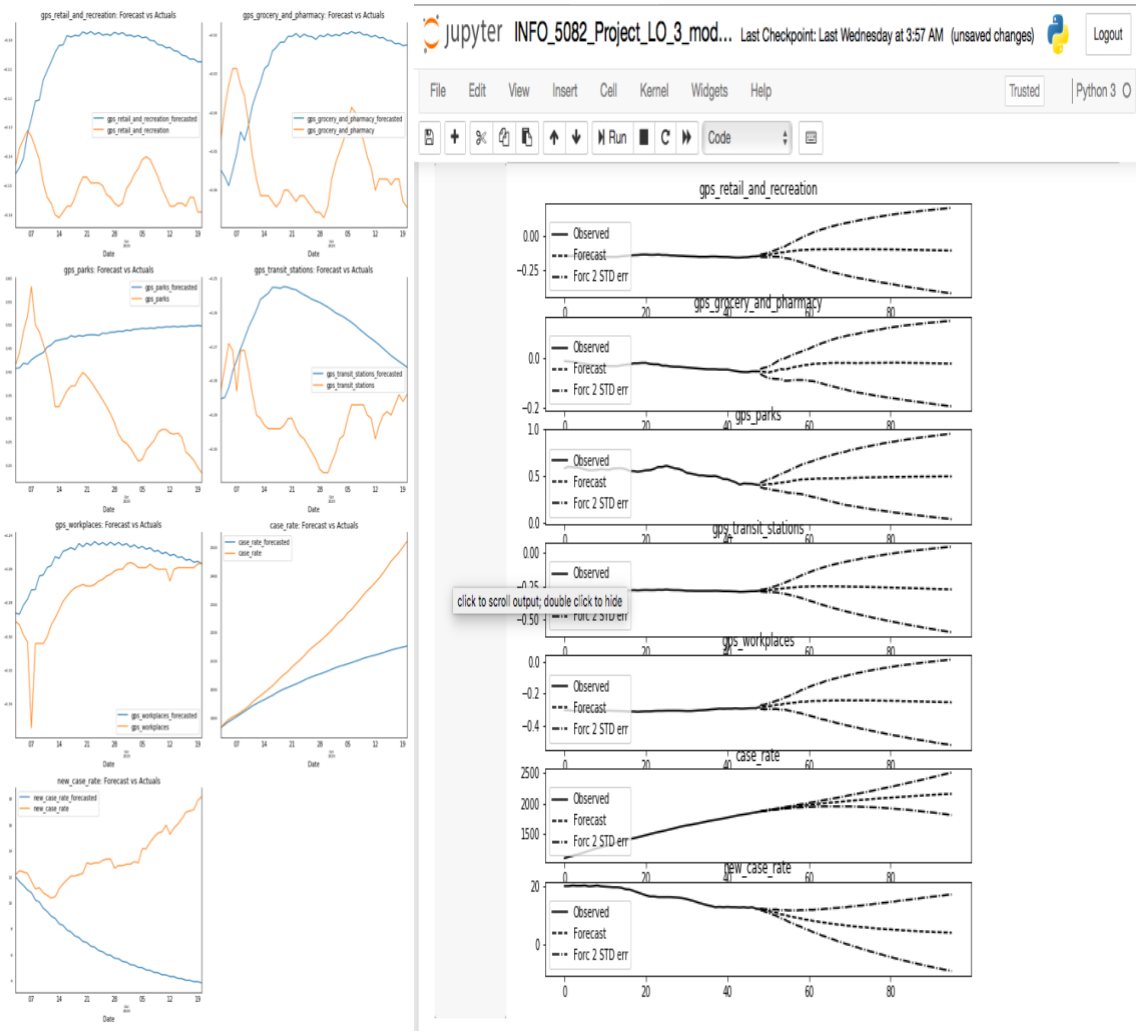


Considering the ‘emp_combined’ variable, RMSE = 0.3572 and bias = -0.2907 says the VECM model is has 35% accuracy and it is slightly over biased for analyzing the overall employment rates.

Data visualizations for third study objective – analyzing the people's mobility with covid-19 case rate

❖ VECM model visualizations and evaluation:

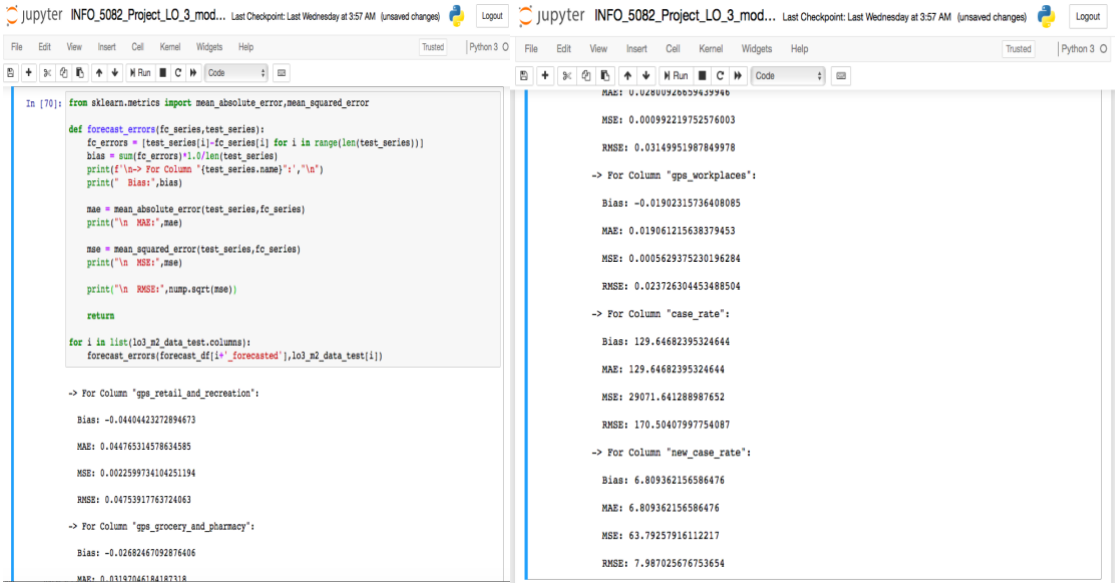
Visualizations for forecasted and actual values are plotted. Also the upper and lower values of forecasted values are plotted as below,



From the above graphs we can see that this model has better forecast. For the given 0.05 significance the upper and lower values for forecast are predicted.

- Evaluating the forecast:

Performing model evaluation using MAE, MSE , RMSE and bias as below.



Considering the ‘case_rate’ variable, RMSE = 170.50 and bias = 129.65 says that model is has a relatively better accuracy and it is under biased for analyzing the overall case_rates.

And considering the ‘new_case_rate’ variable, RMSE = 7.98 and bias = 6.802907 says that model is has a relatively better accuracy and it is under biased for analyzing and predicting new_case_rates in a day.

Conclusion

From the analysis of small business revenue data

Overall revenue in all small business is slowly increasing after sudden impact of covid-19 from March to April 2020.

By observing the granger's causality matrix, we can say the overall revenue generated in small businesses is very much impacted and caused by customers spending in almost every services like general merchandize, arts, etc., and also with opening businesses has shown impact with more revenue. It has been forecasted to maintain the same pattern till November 2020.

Revenue generated in transportation (SS 40) has shown similar impact and causality by customer spending and business opening except with professional and business services. It also ha been forecasted to maintain the same pattern till November 2020.

Revenue generated in Professional and business service (SS 60) has shown similar impact and causality by customer spending and business opening except with spending retails with no groceries and leisure and hospitality services. It also has been forecasted to maintain the same pattern till November 2020

Revenue generated in Education and health service (SS 65) has shown similar impact and causality by customer spending and business opening except with spending in arts and entertainment and leisure and hospitality services. It also has been forecasted to maintain the same pattern till November 2020.

Revenue generated in Leisure and hospitality service (SS 70) has shown similar impact and causality by customer spending and business opening except by transportation services. It also has been forecasted to maintain the same pattern till November 2020.

Both models VAR and VECM predicted with not so good accuracy, so this analysis need much more data for better understanding and predictions in small business revenues.

From the estimation of employment rates

Overall employment rate across the nation has also seen a sudden impact of covid-19 from in April 2020.

By observing the granger's causality matrix, we can say the overall employment rates has a very direct impact and caused by covid case rates along with daily new case rates. It is also impacted by employment rate in leisure and hospitality services and the model accuracy for overall employment is good but can improve its performance with more data.

Employment rate in transportation (SS 40), employment rate in Professional and business service (SS 60), employment rate in Education and health service (SS 65) and employment rate in Leisure and hospitality service (SS 70)has shown impact and causality by covid-19 cases and employment rate in all other services.

The VECM model predicted with a better accuracy rate, but can be improved by giving more data for better understanding and predictions for employment rate.

From the analysis of covid-19 rates with people's mobility

Overall covid-19 infections rate across the nation has been a increasing from the March 2020.

From model 1 where it has shown that people's mobility out of residential locations or inside residence has no impact on covid-19 which is strange.

From model 2, observing the granger's causality matrix, we can say the overall case_rates has a impact and caused by time spent in retail and recreational areas and in pharmacy.

The VECM model predicted with relatively better accuracy rate, but can be improved by giving more input data or including more variables of time spent in other locations like schools for better understanding and predictions for employment rate.

Bibliography

Chetty, R., Friedman, J. N., Hendren, N., Stephene, M. (2020). *The Economic Impacts of COVID-19: Evidence from a New Public Database Built Using Private Sector Data*. Retrieved from: https://opportunityinsights.org/wp-content/uploads/2020/05/tracker_paper.pdf

Sara. H. (2020). "Public Sector Impacts of the Great Recession and COVID-19". Retrieved from https://laborcenter.berkeley.edu/public-sector-impacts-great-recession-and-covid-19/

Sorensen, S. B. (September 2020). ECONOMICS. *Cointegration.* Retrieved from https://uh.edu/~bsorense/coint.pdf

Hjalmarsson, E., Osterholm, P. (June 2007). Rennhack., R. K. *Testing for Cointegration Using the Johansen Methodology when Variables are Near-Integrated*. Retrieved from https://www.imf.org/external/pubs/ft/wp/2007/wp07141.pdf WP/07/141

Singh, A. (2018). Analytics Vidya. *A Gentle Introduction to Handling a Non-Stationary Time Series in Python*. Retrieved from https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python/