CSC110 Project Report: The Effect of COVID-19 on Small Businesses in Ontario

Arihant Bapna, Hongzip Kim, Nicholas Macasaet

December 14, 2021

Problem Description and Research Question

Question: What Has Been the Effect of COVID-19 on Small Businesses in Ontario?
Since the start of the COVID-19 pandemic in 2019, it has had a dramatic effect on many aspects of people's lives. In Canada, specifically, there were many businesses impacted by the pandemic, as they were forced to shut down due to lockdown. These effects are expressed notably through small businesses, as many had to shut down due to provincial restrictions. Upon the introduction of various COVID-19 vaccines as well as newly implemented safety procedures, businesses have started to open up again.
According to Statistics Canada, "small businesses make up 98.0% of all employer businesses in Canada" (Tam 2021). In other words, the Canadian economy heavily relied on small-businesses, and because of the pandemic, the Canadian economy underwent a recession. In our research, we would like to discover how the pandemic affected small businesses in Canada. However, for the purpose of this project, we will be narrowing our scope and focus on the impact of COVID-19 on small businesses in Ontario. The fundamental reason why we chose to conduct our research for small businesses in Ontario is because it is the largest populated province in Canada, and contains Toronto, Canada's financial capital (Toronto 2021). We would like to examine the performance of small businesses, specifically retail businesses, and the factors that influenced it, such as consumer price index, consumer behaviour, as well as the unemployment rate in Ontario. Through this data, we hope to better understand the analytic and trend of the performance of small businesses in Ontario. In addition, we will be using raw data provided by Statistics Canada, which includes the retail trade sales by province and territory from January 1991 to August 2021 (Government of Canada 2021f), as well as the Gross domestic product (GDP) by the industry on a monthly basis from January 1997 to August 2021 (Government of Canada 2021d). With these data sets, we hope to help advance our research in determining how small businesses were affected in Ontario.

Dataset Description

We compile 9 raw datasets for our project, 8 of which are from Statistics Canada in and 1 from OpenCovid. Out of the 11 raw datasets, 5 are csv files and 1 is in json format. Data from Statistics Canada consists of a metadata json as well as a data csv. Only the data files are explained below since the metadata files are used for aggregation purposes (to ensure files are up to date, and to download new data when released)

1. Covid: The Covid Dataset comes from OpenCovid Api as a json, from which we care about the active cases and cases keys (Group 2021)

2. GDP: The GDP data comes from Statisticsanada's WDS service with the *productId* = 36100434. After processing we are left with REF_DATE and Value columns. (Government of Canada 2021c)

3. Retail: The Retail data comes from StatisticsCanada's WDS service with the *productId* = 20100008. After processing we are left with REF_DATE and Value (Government of Canada 2021e)

4. CPI: The Consumer Price Index data comes from StatisticsCanada's WDS service with the *productId* = 18100004. After processing we are left with REF_DATE and Value (Government of Canada 2021a)

5. Employment: The Employment data comes from StatisticsCanada's WDS service with the *productId* = 14100287. After processing we are left with REF_DATE, Value, Age and Sex. (Government of Canada 2021b)

Computational Overview

The program attempts to download the required data sets from StatisticsCanada and OpenCovid or a Backup Server (written in Flask as a REST endpoint) hosted by me in case StatisticsCanada goes down, process the given data and store the processed data as csv files. Then read the data from the given files into plots for analysis purposes. The plots are then passed to a WebApp through Dash and then served to the client through Waitress. When the program is run again, if there is new data available at any of the sources, the program will download it and display new plots. Below are the described methods for doing so

1. Networking: Since we are downloading data from online resources, I set up a Requester class which is the parent to any and all network requests made. It handles unforseen network errors and retries the connection 1 time with a gap of 5 seconds to allow for small network drops. These values can easily be changed (I had them set to 3 times with a gap of 5 seconds before but it was painfully slow to watch since StatisticsCanada has just been down). The class also asserts all requests to ensure a status of 200 was met and otherwise sets the request as a fail (self.fail = True), which prompts the program to try backup routes for needed files.

2. Cube: A cube is just trying to mirror a StatCan data table, and upon initialization of a cube object with a pid and local directory, it tries to pull data from StatCan (metadata.json and data.zip). The zipped data file is then extracted and saved, deleting the .zip shortly after. On a program rerun the cube object will see the metadata file, read it and check if newer data is available from StatCan. If so, it will download the data through the same process. There are tons of fail-safes implemented to ensure the Cube can safely download, extract and update the data as quickly and painlessly as possible. Even so, there are cases that I cannot account for and hopefully you do not come across any such cases.

3. Data Aggregation: The Aggregate class initializes the working directory (folder where all the data goes) for the project if it already hasn't been so in it's constructor. The reading happens in initialize_all_files if it needs to download data from the servers, or initialize_gdp_file Before reading the files, it starts a multiprocess. Process instance, this is because without it the program couldn't handle the overhead of opening so many large files one after another. Calling the garbage collector and deleting the instances of the reader's did not help either. So I resorted to using processes, where every process happens sequentially through .join, ensuring the program doesn't slow down.

4. Data Processing: After the data has been downloaded into the working directory, we load up the raw data into memory as a pandas Dataframe, and process it by first transforming it to filter out specific values we want in columns. Then we drop unnecessary columns from the dataframe, rename the columns to match the common theme and ensure our data is indexed by a timeseries. After processing the raw data, the data is stored into a sub folder of the working_dir called processed_data. It was interesting to note that, the raw data was about 1.25 GB and after processing was only 430 KB. The data from each processed file is then read into a tuple of 5 elements and sent to the Graphing class where it's used to plot all the graphs.

5. Graphing: We use the plotly library for creating plots for our WebApp. If the program has reached this step, it's assuming the rest of the program ran without any issue and that the Graphing class was initialized properly. An instance of this graphic class is then sent to the webapp, so that it can use it to pull plotly figures onto the webpage. Every method in Graphing returns a plotly figure that can easily be imported into our dash app.

6. WebApp: The Dash Webapp implements Dash Bootstrap Componenent for future scalability and Pretty-ness. The set_layout method is what decides how the page is going to be look when it's presented to the client. The page is technically mobile browser compatible although poorly optimized for mobile devices. The layout calls on all the graph generating methods of Graphing for a grand total of 8 graphs being presented to the client

7. Serving: The WebApp is served to the client through Waitress which attempts to start the server for the WebApp on port 8050 of localhost. It should also open the default browser automatically.

8. Prints: The program also documents it's progress in the terminal in different categories, mainly: INFO, WARNING, ERROR, FATAL ERROR, DATA and GRAPH. It is important to note that FATAL ERROR may not actually be fatal, if StatCan is down then a FATAL ERROR may be called but it will then try and connect to the Backup server for the project and download data from there.

Instructions of Usage

1. To begin, all required packages and external resources may be pre-installed with the use of our requirements.txt file. For ease of use, type pip3 install -r requirements.txt within the project directory (from terminal/command prompt).

2. If the initial setup step was ignored, all packages and resources are automatically installed upon start of the main.py file. This file will be the only required file to run to operate our application.

3. Once the main.py file has been started, required datasets are downloaded to the specified directory (data by default). First, existing datasets are evaluated for any needed updates. If needed, a request to download data from Statistics Canada will be made to update the local data on the machine. Unfortunately Statistics Canada has been known to have server up-time issues, to resolve this issue a back-up server has been created by me to always hold the latest datasets.

Changes Since Proposal

Within our original proposal, we intended to specifically focus on the city of Toronto. However, due to insufficient data we have opted to focus on the province of Ontario due for a greater sample size of data. Similarly, due to Ontario containing the highest population out of all of the provinces in Canada, this would allow us to better observe changes in our data sets had we chosen another province or Toronto. Our group did not decide to observe the entirety of Canada due to Ontario providing more centralized data sets that would produce better statistical results.

Discussion

1. What has been the effect of COVID-19 on small businesses? Well from our interactive graphs it's clear that retail trade took a major hit. April was down a lot not only in raw retail but the overall GDP contributions of retail also was down. Analyzing the CPI before covid and after covid, we see a sharp increase in the value. So much so that the EWM is almost an order of magnitude more steeper during Covid. This change is a clear sign of inflation caused by the pandemic, further hurting small businesses. Finally, looking at the unemployment data, we see that the youngest of the workforce took the biggest hit during the pandemic as their unemployment rate almost tripled. This is an indicator of small businesses unable to retain their employees, and having to let go their youngest / most inexperienced staff. The first graph helps us see this more clearly. Hence we can conclude that COVID-19 drastically affected small businesses and their output towards the economy. We also saw that males and females had about the same unemployment rate, regardless of the Age range during the pandemic.

2. Do the results of your computational exploration help answer this question? Most of our computational exploration did in fact help answer this question, but our initial attempts to do so were definitely quite difficult to work with. The correlation coefficients for our data weren't close to 1 at all but that is to be expected of real world raw data. The rest of the produced outputs are quite self explanatory and attempt to explain how small businesses were affected by the pandemic.

3. What limitations did you encounter, with the datasets you found, the algorithms/libraries you used, or other obstacles? The data obtained is not very large since COVID-19 is an ongoing phenomenon, we only have access to data for 2 years at this point. Usually in statistics it's hard to make any concrete conclusions with a small sample size. But that's the great thing about the project. It will automatically keep updating in perpetuity as long as the endpoints are alive. Another issue we had was the Data Aggregation seemed to slow down the program way too much. So much so that it could not move forward after a bit. In order to tackle this obstacle I implemented python Processes which improved performance drastically throughout the program.

4. What are some next steps for further exploration? With more data coming in slowly and aggregating automatically, the program should be able to show us better trends for the future. I would also want to look into more Statistical analysis tools to extract more data from our datasets since I feel like a lot of it is going to waste. I would also look into implementing a prediction model that would look at all the other variables and try and predict a similar drop (the one in April 2020) happening again. In the future we could also do hypothesis testing.

Overall the project was extremely fun to work on and the analysis was extremely intriguing. The outputs were quite surprising as well. The most fun part about the project was getting to see it finally work on multiple devices.

References

Government of Canada, Statistics Canada. 2021a. "Consumer Price Index over Time."

———. 2021b. "Employment over Time."

———. 2021c. "GDP over Time."

———. 2021d. *Gross Domestic Product (GDP) at Basic Prices.* Statistics Canada.

———. 2021e. "Retail over Time."

———. 2021f. *Retail Trade Sales by Province and Territor.* Statistics Canada.

Group, Canada Open Data Working. 2021. "Open Covid Dataset."

Tam, et al., Stephanie. 2021. *Impact of COVID-19 on Small Businesses in Canada, Second Quarter of 2021.* Data Ontario.

Toronto, City of. 2021. *Strong Economy.* Toronto Global.