Introduction

Household spending is a key indicator of economic health and a critical factor in shaping the direction of the economy. It refers to the amount of money that individuals or families spend on goods and services to meet their daily needs and wants. It includes all types of purchases made by household, such as food, clothing, housing, transportation, health care, education and other expenses. It represents a significant portion of consumer demand and drives economic growth. When households have more money to spend, they will increase their expenses and lead to increased economic activity and job creation. It is really important to study for our economy.

In this paper, we aim to analyze the how household spending changes overtime using ordinary least squares (OLS) model. Our goal is to study and examine the spending patterns and how purchasing power differs in different areas, such as food, shelter, Transportation, income tax. We believe the model can help researchers and policymakers better understand consumer behaviors, which can also inform marketing strategies and product development.

In the following section, we will discuss the data sources and variables used in our analysis and followed by the methodology for estimating the OLS model. Then, we will present the results of our analysis, including the coefficients of the model, measures of goodness-of-fit, and statistical significance. We will interpret these results and provide insights into the key findings of the study.

Finally, we will discuss the implications of our findings for the economic policy, businesses decisions, and possible future research in the conclusion section. Overall, this paper aims to focus on the relationship between household spending and employment by industry and provide valuable insights for policymakers, businesses, and investors.

Data:

In this section, we will describe the data sources and variables used in our analysis. All our data are from Statistics Canada for the period 2010 to 2019. The data set includes information on total household expenditures, and household expenditures on food, shelter, transportation, and income tax.

Our primary independent variable is year that between 2010 and 2019. And our dependent variable is household expenditures as well as household expenditures in food, shelter, transportation, and income tax. To aviod multicorrelation problem, we will create multiple regression models for our analyze.

The following table provides a descriptive statistic for the key variables used in our analysis. (Table placeholder, Information will be filled later)

Modelling

The data was prepared was good quality but required some data cleaning. Some steps that were taken to clean the data was dropping unnecessary columns. Columns such as GEO data were not needed as all the locations were already determined to be Canada. Next the column names were changed to be more short and clear.
To prepare the data for analysis, the data was imported as a dataframe using pandas. Scikitlearn was used for its LinearRegression tool to quickly get the predicted spending of Canadian's per year. One challenge that was faced was loading the dataframe into Google Colab. I learned that Colab can accept data from Github so I created a variable that linked to my dataset. I also had trouble plotting the data. The y label kept bleeding into other subplots. I learned that the adjust function can allow you to create whitespace between subplots.

The model created is a predictive model for expenditure. The annual average expenditure for each topic such as food or shelter is graphed on a scatterplot. Scikitlearn is used to get the y predicted value. The linear regression works very well as the R^2 value is all above 0.8m showing a strong correlation for all topics.

The following modelling was completed to determine if there is

```
import pandas as pd
import numpy as np

# Canadian Data
# https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1110022201
# Import and clean the data set
url_can = "https://raw.githubusercontent.com/jkeomany/Project/main/can_data.csv"
df_can = pd.read_csv(url_can)

## Make a list of columns to drop
drop_list = ['GEO','DGUID','Statistic','UOM','UOM_ID','SCALAR_FACTOR','SCALAR_ID','VECTOR','COORDINATE','STATUS','SYMBOL','TERMINATED','DECIM
## The columns in drop list were removed from the statistical analysis because the columns did not provide value for the regression model and
```

```
## Make the Canadian household expenditures dataframe
df_can = df_can.drop(drop_list, axis=1)

## Rename long and unclear column names
df_can.rename(columns={"Household expenditures, summary-level categories": "HOUSEHOLD EXPENDITURE"}, inplace=True)

df_can.head()
```

	REF_DATE	HOUSEHOLD EXPENDITURE	VALUE
0	2010	Total expenditure	72075
1	2011	Total expenditure	73646
2	2012	Total expenditure	75695
3	2013	Total expenditure	79098
4	2014	Total expenditure	80727

```
# Choose Spending Areas

## Total expenditures
can_total = df_can.drop('HOUSEHOLD EXPENDITURE', axis=1)[0:9]
#print(can_total)

## Food
can_food = df_can.drop('HOUSEHOLD EXPENDITURE', axis=1)[df_can['HOUSEHOLD EXPENDITURE']=='Food expenditures']
#print(can_food)

## Shelter
can_shelter = df_can.drop('HOUSEHOLD EXPENDITURE', axis=1)[df_can['HOUSEHOLD EXPENDITURE']=='Shelter']
#print(can_shelter)

## Transportation
can_transport = df_can.drop('HOUSEHOLD EXPENDITURE', axis=1)[df_can['HOUSEHOLD EXPENDITURE']=='Transportation']
#print(can_transport)

## Income Tax
can_tax = df_can.drop('HOUSEHOLD EXPENDITURE', axis=1)[df_can['HOUSEHOLD EXPENDITURE']=='Income taxes']
#print(can_tax)


# Data Discripution

## Total expenditures
total_count = can_total['VALUE'].count()
print('Total number of Total Expenditures:', total_count)

total_sum = can_total['VALUE'].sum()
print('Total sum of Total Expenditures:', total_sum)

total_avg = can_total['VALUE'].mean()
print('Average Total Expenditures:', total_avg)

## Food
total_count_food = can_food['VALUE'].count()
print('Total number of Total Expenditures:', total_count_food)

total_sum_food = can_food['VALUE'].sum()
print('Total sum of Total Expenditures:', total_sum_food)

total_avg_food = can_food['VALUE'].mean()
print('Average Total Expenditures:', total_avg_food)

## Shelter
total_count_shelter = can_shelter['VALUE'].count()
print('Total number of Total Expenditures:', total_count_shelter)

total_sum_shelter = can_shelter['VALUE'].sum()
print('Total sum of Total Expenditures:', total_sum_shelter)

total_avg_shelter = can_shelter['VALUE'].mean()
print('Average Total Expenditures:', total_avg_shelter)

## Transportation
total_count_transport = can_transport['VALUE'].count()
print('Total number of Total Expenditures:', total_count_transport)
```

```
total_sum_transport = can_transport['VALUE'].sum()
print('Total sum of Total Expenditures:', total_sum_transport)

total_avg_transport = can_transport['VALUE'].mean()
print('Average Total Expenditures:', total_avg_transport)

## Income Tax
total_count_tax = can_tax['VALUE'].count()
print('Total number of Total Expenditures:', total_count_tax)

total_sum_tax = can_tax['VALUE'].sum()
print('Total sum of Total Expenditures:', total_sum_tax)

total_avg_tax = can_tax['VALUE'].mean()
print('Average Total Expenditures:', total_avg_tax)
```

```
Total number of Total Expenditures: 9
Total sum of Total Expenditures: 728630
Average Total Expenditures: 80958.88888888889
Total number of Total Expenditures: 9
Total sum of Total Expenditures: 76136
Average Total Expenditures: 8459.555555555555
Total number of Total Expenditures: 9
Total sum of Total Expenditures: 153944
Average Total Expenditures: 17104.88888888889
Total number of Total Expenditures: 9
Total sum of Total Expenditures: 106683
Average Total Expenditures: 11853.666666666666
Total number of Total Expenditures: 9
Total sum of Total Expenditures: 130038
Average Total Expenditures: 14448.666666666666
```

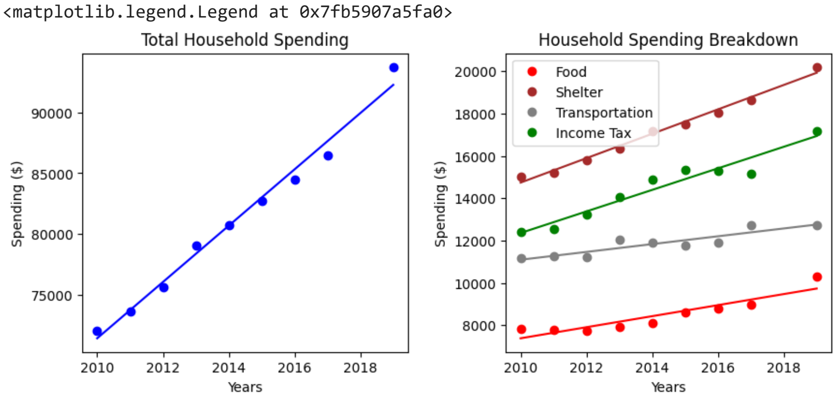
```
# Create an OLS regression
from sklearn.linear_model import LinearRegression

## Total Expenditures
can_totalx = can_total['REF_DATE'].to_numpy().reshape((-1,1))
can_totaly = can_total['VALUE'].to_numpy()
total_model = LinearRegression().fit(can_totalx, can_totaly)
total_rsq = total_model.score(can_totalx, can_totaly)
total_intercept = total_model.intercept_
total_coef = total_model.coef_
total_ypred = total_model.predict(can_totalx)
print('total',total_rsq)

## Food
can_foodx = can_food['REF_DATE'].to_numpy().reshape((-1,1))
can_foody= can_food['VALUE'].to_numpy()
food_model = LinearRegression().fit(can_foodx, can_foody)
food_rsq = food_model.score(can_foodx, can_foody)
food_intercept = food_model.intercept_
food_coef = food_model.coef_
food_ypred = food_model.predict(can_foodx)
print('food',food_rsq)

## Shelter
can_shelterx = can_shelter['REF_DATE'].to_numpy().reshape((-1,1))
can_sheltery= can_shelter['VALUE'].to_numpy()
shelter_model = LinearRegression().fit(can_shelterx, can_sheltery)
shelter_rsq = shelter_model.score(can_shelterx, can_sheltery)
shelter_intercept = shelter_model.intercept_
shelter_coef = shelter_model.coef_
shelter_ypred = shelter_model.predict(can_shelterx)
print('shelter',shelter_rsq)

## Transportation
can_transportx = can_transport['REF_DATE'].to_numpy().reshape((-1,1))
can_transporty= can_transport['VALUE'].to_numpy()
transport_model = LinearRegression().fit(can_transportx, can_transporty)
transport_rsq = transport_model.score(can_transportx, can_transporty)
transport_intercept = transport_model.intercept_
transport_coef = transport_model.coef_
transport_ypred = transport_model.predict(can_transportx)
print('transport',transport_rsq)

## Income Tax
can_taxx = can_tax['REF_DATE'].to_numpy().reshape((-1,1))
```

```
can_taxy= can_tax['VALUE'].to_numpy()
tax_model = LinearRegression().fit(can_taxx, can_taxy)
tax_rsq = tax_model.score(can_taxx, can_taxy)
tax_intercept = tax_model.intercept_
tax_coef = tax_model.coef_
tax_ypred = tax_model.predict(can_taxx)
print('tax',tax_rsq)
```

```
total 0.9856932818200037
food 0.8480160664926918
shelter 0.9890364816254562
transport 0.8254387565199827
tax 0.9358421630873699
```

```
# Graph Canadian Models

import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl

## Create a subplot for the total spending and
fig, (ax1,ax2) = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

## Adjust the subplot layout parameters
fig.subplots_adjust(hspace=0.125, wspace=0.3)

## Graph total household spending
ax1.plot(can_total.REF_DATE,can_total.VALUE,'o',color='blue')
ax1.plot(can_totalx,total_ypred,'-',color='blue')
ax1.title.set_text('Total Household Spending')
ax1.set_xlabel('Years')
ax1.set_ylabel('Spending ($)')

## Graph food spending
ax2.plot(can_food.REF_DATE,can_food.VALUE,'o',color='red',label='Food')
ax2.plot(can_foodx,food_ypred,'-',color='red')

## Graph shelter spending
ax2.plot(can_shelter.REF_DATE,can_shelter.VALUE,'o',color='brown',label='Shelter')
ax2.plot(can_shelterx,shelter_ypred,'-',color='brown')

## Graph transportation spending
ax2.plot(can_transport.REF_DATE,can_transport.VALUE,'o',color='grey',label='Transportation')
ax2.plot(can_transportx,transport_ypred,'-',color='grey')

## Graph income tax spending
ax2.plot(can_tax.REF_DATE,can_tax.VALUE,'o',color='green',label='Income Tax')
ax2.plot(can_taxx,tax_ypred,'-',color='green')

ax2.title.set_text('Household Spending Breakdown')
ax2.set_xlabel('Years')
ax2.set_ylabel('Spending ($)')
ax2.legend()
```

```
<matplotlib.legend.Legend at 0x7fb5907a5fa0>
```



Result

```
# OLS Summary result

## Total Expenditures

import statsmodels.api as sm

can_totalx = can_total['REF_DATE']
can_totaly = can_total['VALUE']
can_totalx = sm.add_constant(can_totalx)
total_model = sm.OLS(can_totaly, can_totalx).fit()
print(total_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  VALUE   R-squared:                       0.986
Model:                            OLS   Adj. R-squared:                  0.984
Method:                 Least Squares   F-statistic:                     482.3
Date:                Sat, 15 Apr 2023   Prob (F-statistic):           1.03e-07
Time:                        21:02:42   Log-Likelihood:                -72.583
No. Observations:                   9   AIC:                             149.2
Df Residuals:                       7   BIC:                             149.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -4.569e+06   2.12e+05    -21.579      0.000   -5.07e+06   -4.07e+06
REF_DATE    2308.7484    105.130     21.961      0.000    2060.155    2557.341
==============================================================================
Omnibus:                        0.478   Durbin-Watson:                   1.812
Prob(Omnibus):                  0.787   Jarque-Bera (JB):                0.393
Skew:                           0.394   Prob(JB):                        0.822
Kurtosis:                       2.348   Cond. No.                     1.47e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.9/dist-packages/scipy/stats/_stats_py.py:1736: UserWarning: kurtosistest only valid for n>=20 ... continuing anyw
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

```
## Food

can_foodx = can_food['REF_DATE']
can_foody = can_food['VALUE']
can_foodx = sm.add_constant(can_foodx)
total_model_food = sm.OLS(can_foody, can_foodx).fit()
print(total_model_food.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  VALUE   R-squared:                       0.848
Model:                            OLS   Adj. R-squared:                  0.826
Method:                 Least Squares   F-statistic:                     39.06
Date:                Sat, 15 Apr 2023   Prob (F-statistic):           0.000424
Time:                        21:02:44   Log-Likelihood:                -64.269
No. Observations:                   9   AIC:                             132.5
Df Residuals:                       7   BIC:                             132.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -5.169e+05   8.41e+04     -6.149      0.000   -7.16e+05   -3.18e+05
REF_DATE     260.8468     41.738      6.250      0.000     162.152     359.542
```

```
==============================================================================
Omnibus:                        2.213   Durbin-Watson:                   1.144
Prob(Omnibus):                  0.331   Jarque-Bera (JB):                1.362
Skew:                           0.873   Prob(JB):                        0.506
Kurtosis:                       2.235   Cond. No.                     1.47e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.9/dist-packages/scipy/stats/_stats_py.py:1736: UserWarning: kurtosistest only valid for n>=20 ... continuing anyw
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

```
## Shelter

can_shelterx = can_shelter['REF_DATE']
can_sheltery = can_shelter['VALUE']
can_shelterx = sm.add_constant(can_shelterx)
total_model_shelter = sm.OLS(can_sheltery, can_shelterx).fit()
print(total_model_shelter.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  VALUE   R-squared:                       0.989
Model:                            OLS   Adj. R-squared:                  0.987
Method:                 Least Squares   F-statistic:                     631.5
Date:                Sat, 15 Apr 2023   Prob (F-statistic):           4.03e-08
Time:                        21:02:48   Log-Likelihood:                -58.876
No. Observations:                   9   AIC:                             121.8
Df Residuals:                       7   BIC:                             122.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.143e+06   4.62e+04    -24.759      0.000   -1.25e+06   -1.03e+06
REF_DATE     576.1177     22.926     25.129      0.000     521.906     630.329
==============================================================================
Omnibus:                        2.345   Durbin-Watson:                   1.647
Prob(Omnibus):                  0.310   Jarque-Bera (JB):                1.471
Skew:                           0.844   Prob(JB):                        0.479
Kurtosis:                       1.965   Cond. No.                     1.47e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.9/dist-packages/scipy/stats/_stats_py.py:1736: UserWarning: kurtosistest only valid for n>=20 ... continuing anyw
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

```
## Transportation

can_transportx = can_transport['REF_DATE']
can_transporty = can_transport['VALUE']
can_transportx = sm.add_constant(can_transportx)
total_model_transport = sm.OLS(can_transporty, can_transportx).fit()
print(total_model_transport.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  VALUE   R-squared:                       0.825
Model:                            OLS   Adj. R-squared:                  0.801
Method:                 Least Squares   F-statistic:                     33.10
Date:                Sat, 15 Apr 2023   Prob (F-statistic):           0.000696
Time:                        21:02:51   Log-Likelihood:                -61.900
No. Observations:                   9   AIC:                             127.8
Df Residuals:                       7   BIC:                             128.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -3.599e+05   6.46e+04     -5.570      0.001   -5.13e+05   -2.07e+05
REF_DATE     184.5774     32.082      5.753      0.001     108.716     260.439
==============================================================================
Omnibus:                        0.643   Durbin-Watson:                   2.480
Prob(Omnibus):                  0.725   Jarque-Bera (JB):                0.584
```

```
Skew:                           0.338   Prob(JB):                        0.747
Kurtosis:                       1.951   Cond. No.                     1.47e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.9/dist-packages/scipy/stats/_stats_py.py:1736: UserWarning: kurtosistest only valid for n>=20 ... continuing anyw
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

```
## Income Tax

can_taxx = can_tax['REF_DATE']
can_taxy = can_tax['VALUE']
can_taxx = sm.add_constant(can_taxx)
total_model_tax = sm.OLS(can_taxy, can_taxx).fit()
print(total_model_tax.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  VALUE   R-squared:                       0.936
Model:                            OLS   Adj. R-squared:                  0.927
Method:                 Least Squares   F-statistic:                     102.1
Date:                Sat, 15 Apr 2023   Prob (F-statistic):           2.00e-05
Time:                        21:02:53   Log-Likelihood:                -65.956
No. Observations:                   9   AIC:                             135.9
Df Residuals:                       7   BIC:                             136.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -1.01e+06   1.01e+05     -9.962      0.000   -1.25e+06    -7.7e+05
REF_DATE     508.7516     50.348     10.105      0.000     389.698     627.805
==============================================================================
Omnibus:                        1.733   Durbin-Watson:                   1.718
Prob(Omnibus):                  0.420   Jarque-Bera (JB):                0.669
Skew:                          -0.658   Prob(JB):                        0.716
Kurtosis:                       2.771   Cond. No.                     1.47e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
/usr/local/lib/python3.9/dist-packages/scipy/stats/_stats_py.py:1736: UserWarning: kurtosistest only valid for n>=20 ... continuing anyw
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Conclusion

Summary

Double-click (or enter) to edit