DS5100

2020 Tokyo Olympics

Reilly Meinert, Max Ryoo, Said Mrad, Sydney Masterson

Project Scenario

Team	Gold	Silver	Bronze	Total
Argentina	-	1	2	3
Armenia	-	2	2	4
Australia	17	7	22	46

- Brief Introduction About the data set
    - Olympics
        - https://olympics.com/en/olympic-games/tokyo-2020/medals
    - GDP
        - https://www.worldometers.info/gdp/gdp-by-country/
    - Kaggle Integration
        - https://www.kaggle.com/arjunprasadsarkhel/2021-olympics-in-tokyo?select=Teams.xlsx

Dataset Introduction

- Pandas Dataframe to csv
  - https://github.com/hyunsukr/DS5100-Final/tree/main/data
  - Dimension
    - Tokyo Olympics: 74 x 14
    - Historical Olympics: 1315 x 9
- Data was webscrapped and engineered to produce a final dataframe with the information below.

	Name	Gold	Silver	Bronze	Total	Country	GDP	GDP abbreviated	GDP growth	Population	GDP per capita	NOC	Discipline	Continents
0	USA United States of America	39	41	33	113	United States	$19,485,394,000,000.00	$19.485 trillion	2.27%	325,084,756	$59,939.00	United States of America	47	North America
1	CHN People's Republic of China	38	32	18	88	China	$12,237,700,479,375.00	$12.238 trillion	6.90%	1,421,021,791	$8,612.00	People's Republic of China	33	Asia
2	JPN Japan	27	14	17	58	Japan	$4,872,415,104,315.00	$4.872 trillion	1.71%	127,502,725	$38,214.00	Japan	48	Asia
3	GBR Great Britain	22	21	22	65	United Kingdom	$2,637,866,340,434.00	$2.638 trillion	1.79%	66,727,461	$39,532.00	Great Britain	28	Europe
4	ROC ROC	20	28	23	71	Russia	$1,578,417,211,937.00	$1.578 trillion	1.55%	145,530,082	$10,846.00	ROC	34	Asia
5	AUS Australia	17	7	22	46	Australia	$1,323,421,072,479.00	$1.323 trillion	1.96%	24,584,620	$53,831.00	Australia	35	Australia
6	NED Netherlands	10	12	14	36	Netherlands	$830,572,618,850.00	$831 billion	3.16%	17,021,347	$48,796.00	Netherlands	27	Europe
7	FRA France	10	12	11	33	France	$2,582,501,307,216.00	$2.583 trillion	1.82%	64,842,509	$39,827.00	France	33	Europe

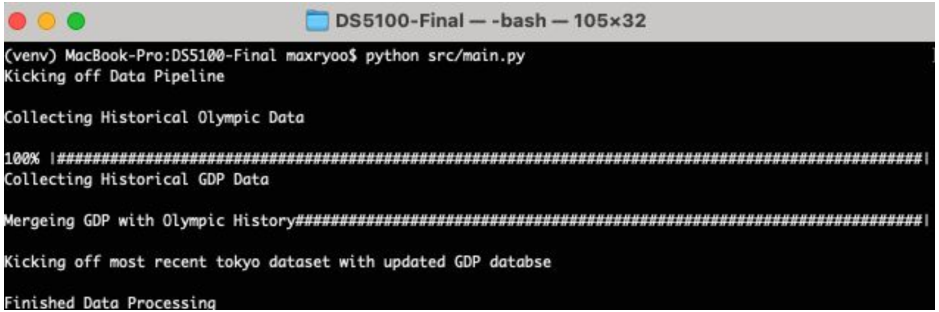
Dataset Web Scraping

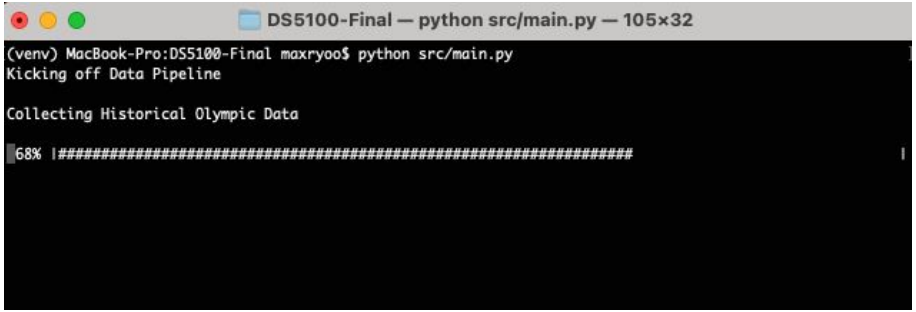
- Class called Web_Scrapper
- Three data sets were webscrapped
    - Olympics
    - GDP
        - Only gives 2020 (recent GDP)
    - GDP Historical Data

```python
class Web_Scrapper():
    def __init__(self, baselink="https://olympics.com/tokyo-2020/olympic-games/en/results/all-sports/", history = {}):
        self.baselink = baselink
        with open('src/resources/history.json') as json_file:
            history = json.load(json_file)
        self.history = history

    def scrape_gdp_history(self, years):
        ## Not Tested Yet
        time_series = pd.DataFrame()
        pbar.start()
        for i in pbar(range(0,len(years))):
            if int(years[i]) > 1949:
                df = self.scrape_gdp_economy(years[i])
                time_series = time_series.append(df)

        return time_series


    def scrape_gdp_economy(self, year):
        URL = 'https://countryeconomy.com/gdp?year=' + year
        r = requests.get(URL) #http requests tot ehs specified url and save it in R
        soup = BeautifulSoup(r.content, 'html5lib')
        tables = soup.find_all('table', {'id':'tbA'})
        tables_percap = soup.find_all('table', {'id':'tbPC'})
        tempList = []
        for table in tables:
            for child in table.children:
                for td in child:
                    for tr in td:
                        tempList.append(tr.get_text())

        second_tempList = []
        for table in tables_percap:
            for child in table.children:
                for td in child:
                    for tr in td:
                        second_tempList.append(tr.get_text())
        tempList = tempList[5:len(tempList)-1]
        second_tempList = second_tempList[6:len(second_tempList) - 1]
```

Data Processing - Interaction

- Interaction with user
- Progress bar to show % data pull
- Give feedback to user how much the data pull is complete.





Data Processing - Data Engineering

```
{
    "United States of America" : "United States" ,
    "People's Republic of China" : "China",
    "Japan" : "Japan",
    "Great Britain" : "United Kingdom",
    "ROC" : "Russia",
    "Australia" : "Australia",
    "Netherlands": "Netherlands",
```

- Joined datasets based on country name
    - Some country names were different
    - Had to map countries names through a json (dictionary) through data cleaning
- Added geographical location for the data
    - Continents each country is located
    - Utilized a third party package
        - pycountry-convert

pycountry-convert 0.7.2

```
pip install pycountry-convert
```

```python
class Cleaner():
    def __init__(self):
        # Opening JSON Mapping file
        with open('src/resources/mapping.json') as json_file:
            mapping = json.load(json_file)

        with open('src/resources/mapping_continents.json') as json_file:
            cont_map = json.load(json_file)

        self.continent_maps = cont_map
        self.country_maps = mapping

    def join_gdp(self, gdp, olympic, join_cols=['Country']):
        temp_olympic = olympic.copy()
        temp_olympic["Country"] = temp_olympic["Name"].str[4:].map(self.country_maps)
        joined = pd.merge(temp_olympic, gdp, how='left', on=join_cols)
        return joined

    def join_aggregate_teams(self, teams, olympic):
        teams = teams.groupby("NOC")["Discipline"].count().reset_index()
        temp = olympic.copy()
        temp['tempName'] = temp["Name"].str[4:]
        joined = pd.merge(temp, teams, how='inner', left_on='tempName', right_on='NOC')
        joined = joined.drop(columns=["tempName"])
        return joined
```

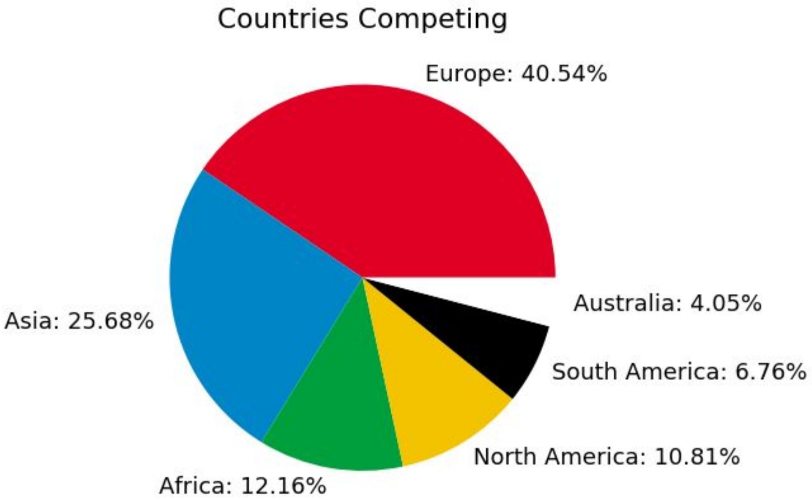
Testing

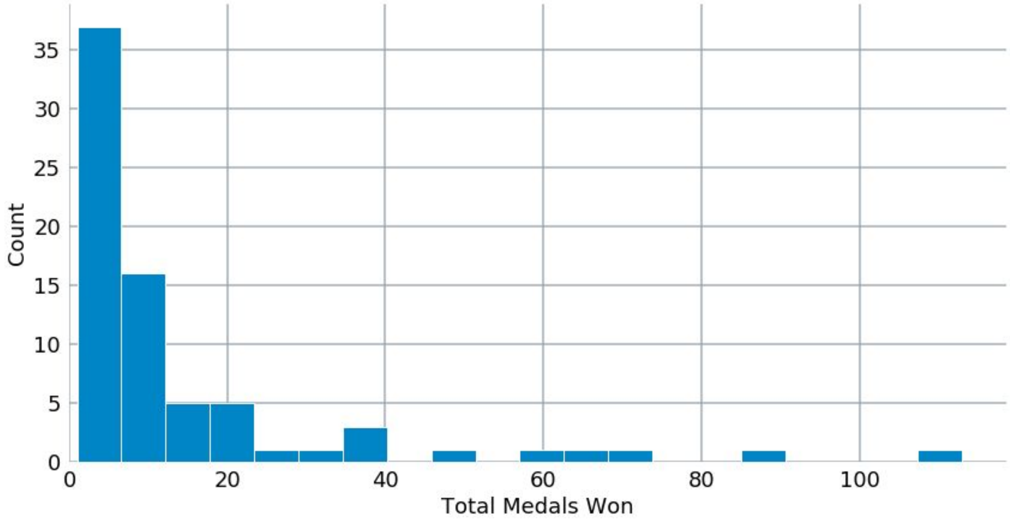
- Pytest to test the code / coverage
- Data engineering functions and data quality testing
- All methods relating to data
- Code coverage of 100% except main

```
============================= test session starts ==============================
platform darwin -- Python 3.7.1, pytest-6.2.5, py-1.11.0, pluggy-1.0.0 -- /Users/maxryoo/Documents/MSDS/DS5100/DS5100-Final/venv/bin/python
cachedir: .pytest_cache
rootdir: /Users/maxryoo/Documents/MSDS/DS5100/DS5100-Final
plugins: mock-3.6.1, cov-3.0.0
collected 11 items

tests/utils/test_cleaner.py::test_join_gdp_single PASSED                 [  9%]
tests/utils/test_cleaner.py::test_get_continents_map PASSED              [ 18%]
tests/utils/test_cleaner.py::test_get_continents_map_exceptions PASSED   [ 27%]
tests/utils/test_cleaner.py::test_convert_continent PASSED               [ 36%]
tests/utils/test_cleaner.py::test_convert_continent_not_available PASSED [ 45%]
tests/utils/test_cleaner.py::test_join_gdp PASSED                        [ 54%]
tests/utils/test_cleaner.py::test_join_aggregate_teams PASSED            [ 63%]
tests/utils/test_webscrapper.py::test_scrape_gdp PASSED                  [ 72%]
tests/utils/test_webscrapper.py::test_scarpe_summary PASSED              [ 81%]
tests/utils/test_webscrapper.py::test_scrape_gdp_history PASSED          [ 90%]
tests/utils/test_webscrapper.py::test_scrape_history PASSED                                [100%]

=============================================== warnings summary ===============================================
tests/utils/test_cleaner.py::test_get_continents_map
  /Users/maxryoo/Documents/MSDS/DS5100/DS5100-Final/venv/lib/python3.7/site-packages/pkg_resources/_vendor/pyparsing.py:943: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
    collections.MutableMapping.register(ParseResults)

tests/utils/test_cleaner.py::test_get_continents_map
  /Users/maxryoo/Documents/MSDS/DS5100/DS5100-Final/venv/lib/python3.7/site-packages/pkg_resources/_vendor/pyparsing.py:3226: DeprecationWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
    elif isinstance( exprs, collections.Iterable ):

-- Docs: https://docs.pytest.org/en/stable/warnings.html

---------- coverage: platform darwin, python 3.7.1-final-0 -----------
Name                        Stmts   Miss  Cover
-----------------------------------------------
src/__init__.py                 0      0   100%
src/main.py                    34     34     0%
src/utils/__init__.py           0      0   100%
src/utils/cleaner.py           50      0   100%
```

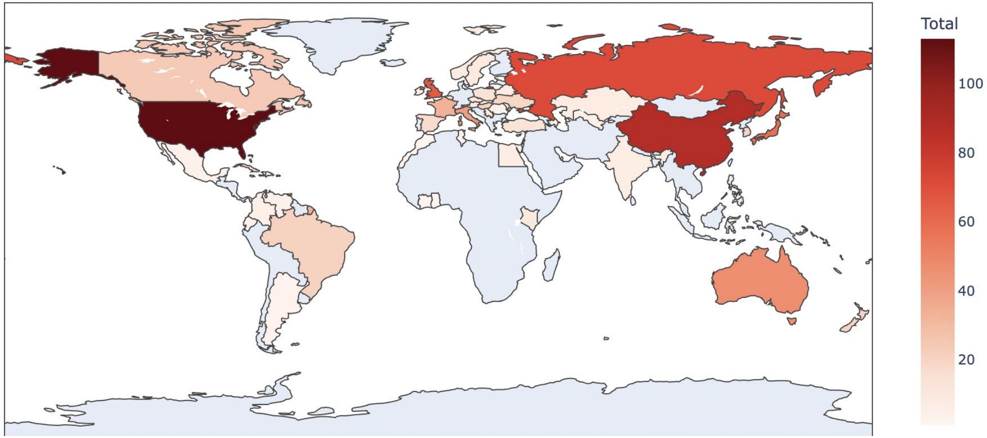
```python
def test_scarpe_summary():
    webscrapper = Web_Scrapper()
    scrapped_olympic = webscrapper.scrape_summary('https://olympics.com/en/olympic-games/tokyo-2020/medals')

    computed_total = scrapped_olympic["Gold"].astype(int) + scrapped_olympic["Silver"].astype(int) + scrapped_olympic["Bronze"].astype(int)

    sanity_check_medal_count = sum(scrapped_olympic["Total"].astype(int) > 500)

    assert set(scrapped_olympic) == set(['Name', 'Gold', 'Silver', 'Bronze', 'Total'])
    assert scrapped_olympic.equals(scrapped_olympic.dropna())
    assert computed_total.equals(scrapped_olympic["Total"].astype(int))
    assert sanity_check_medal_count == 0
```

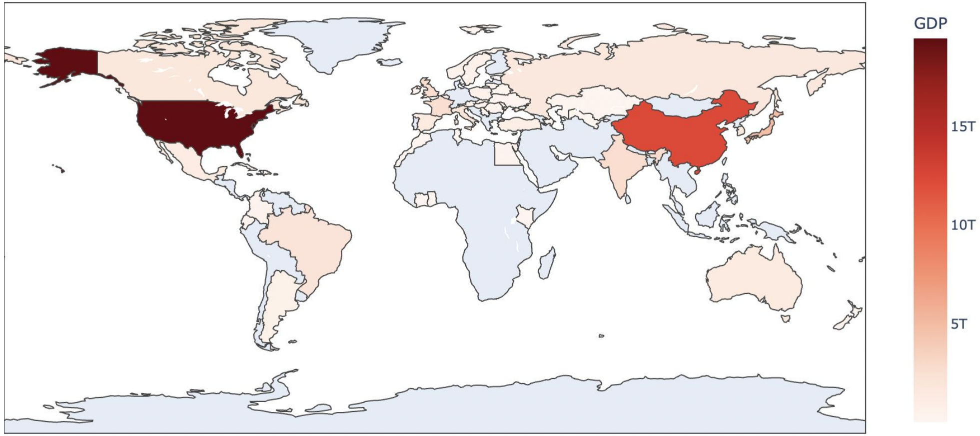
Exploratory Data Analysis





World Statistics

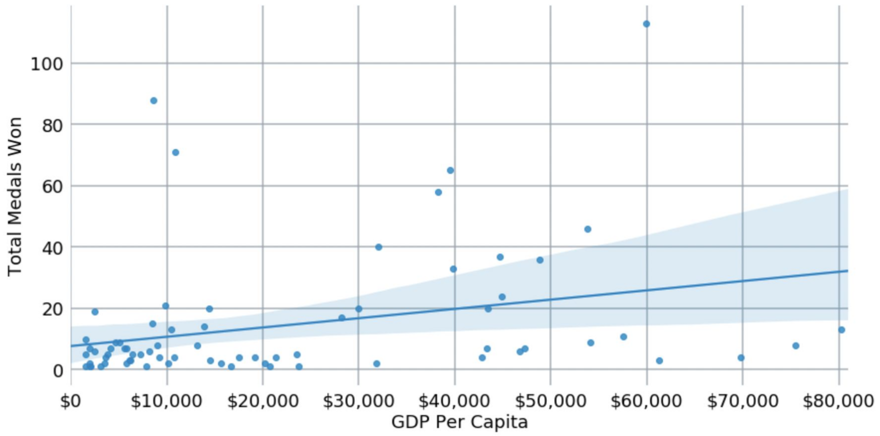


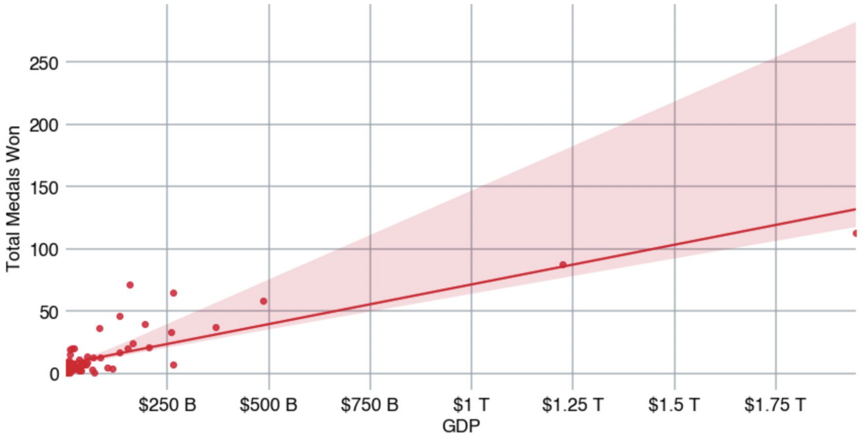


Conclusions:

- US has highest GDP and number of medals won
- China second for both
- Relationship not as strong for rest of world
    - Russia, Australia, & Great Britain have high medal counts but lower GDPs compared to US and China

Medal Count versus GDP





Medals Won versus GDP Per Capita: .2975

Medals Won versus GDP: 0.8362

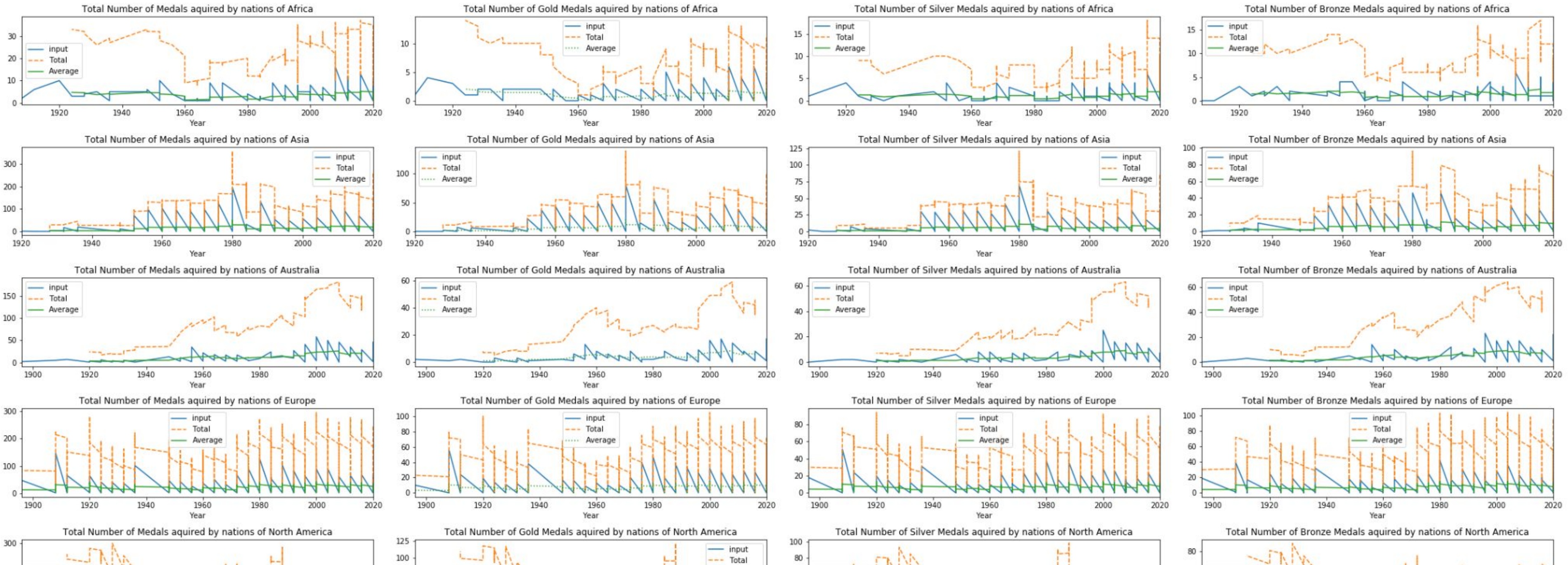
Conclusion: The relationship between GDP and medals won is much stronger than the relationship between GDP per capita and medals won.

Model Building

- Multiple Linear Regression
  - R^2 = 0.8479990568528668
  - Mean Squared Error = 109.1391895919251
  - Root Mean Squared Error = 10.446970354697342
- Possible Next Steps
  - Multicollinearity
  - Linear Regression assumption checking
  - GDP and Population had a beta of 0, which may raise eyebrows

	Feature	Coefficient
7	Continents_Europe	6.200415
8	Continents_North America	4.484664
5	Continents_Asia	3.548857
6	Continents_Australia	2.443772
4	Discipline	0.784625
0	GDP	0.000000
2	Population	0.000000
9	Continents_South America	-6.659379
1	GDP growth	-0.138194
3	GDP per capita	-0.000042

Time Series Analysis



Conclusion / Next Steps

- Code in github is available with virtual environments
  - Making the github repo a package repo will make it so that we can deploy the package.
    - Setup.py
- Dive deeper in the Multiple Linear Regression model such as multicollinearity etc.