Marketing - Regression

Luke Nigel J. Laylo

St. Lawrence College

ADMN5015 - Artificial Intelligence in Marketing

Milton Davila

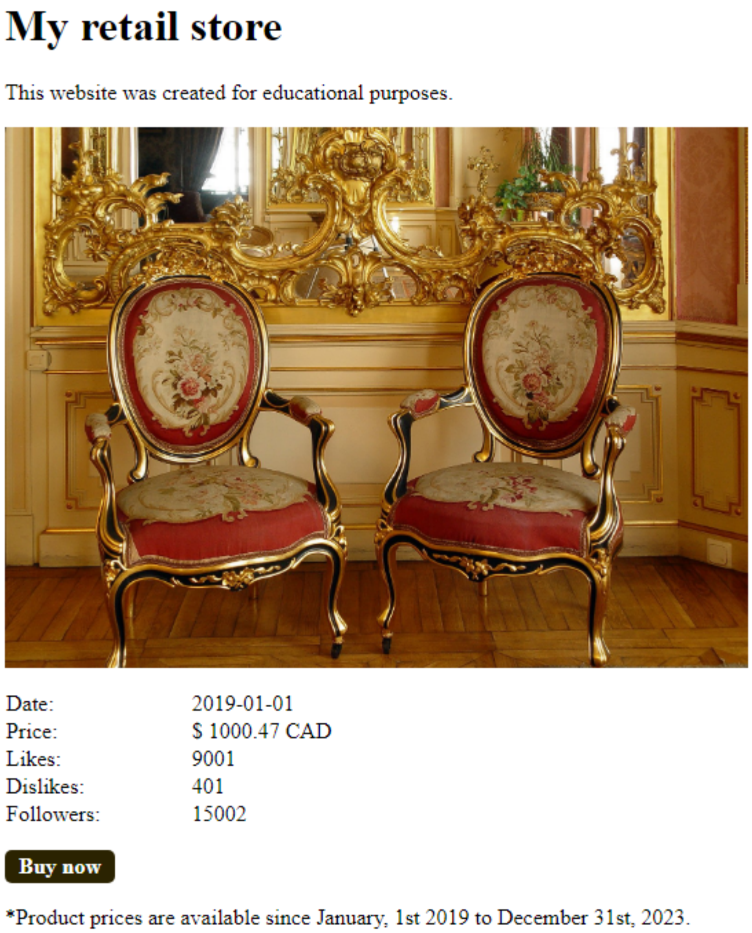
01/18/2023

Introduction

Web scraping is the process of extracting data from websites using code. Python is a popular programming language for web scraping due to its libraries like BeautifulSoup, which make it easy to navigate and extract information from HTML files. In this project, we will use Python to scrape data from a website. We then extract the data from the data we want to get (date,price,likes,dislikes, and followers) for each day from January 1st, 2019 to December 31st, 2023. Once the data has been extracted we form it into a dataframe and export it to csv.

Website Extraction

The website we are going to extract data from is (https://admn5015-340805.uc.r.appspot.com/2019-01-01.html)



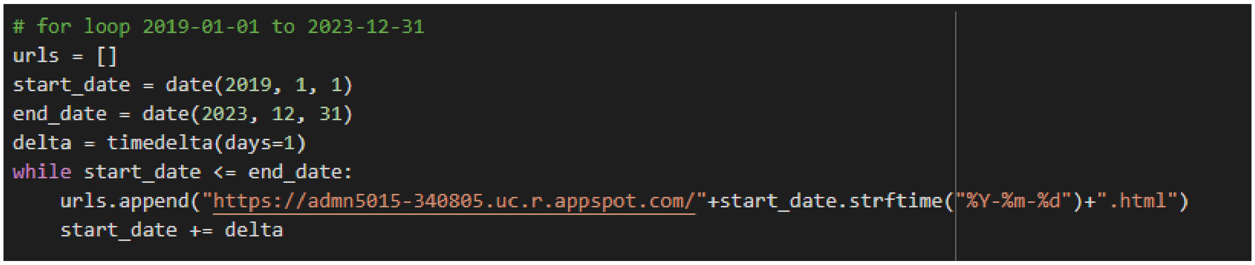
My retail store

This website was created for educational purposes.

Date:	2019-01-01
Price:	$ 1000.47 CAD
Likes:	9001
Dislikes:	401
Followers:	15002

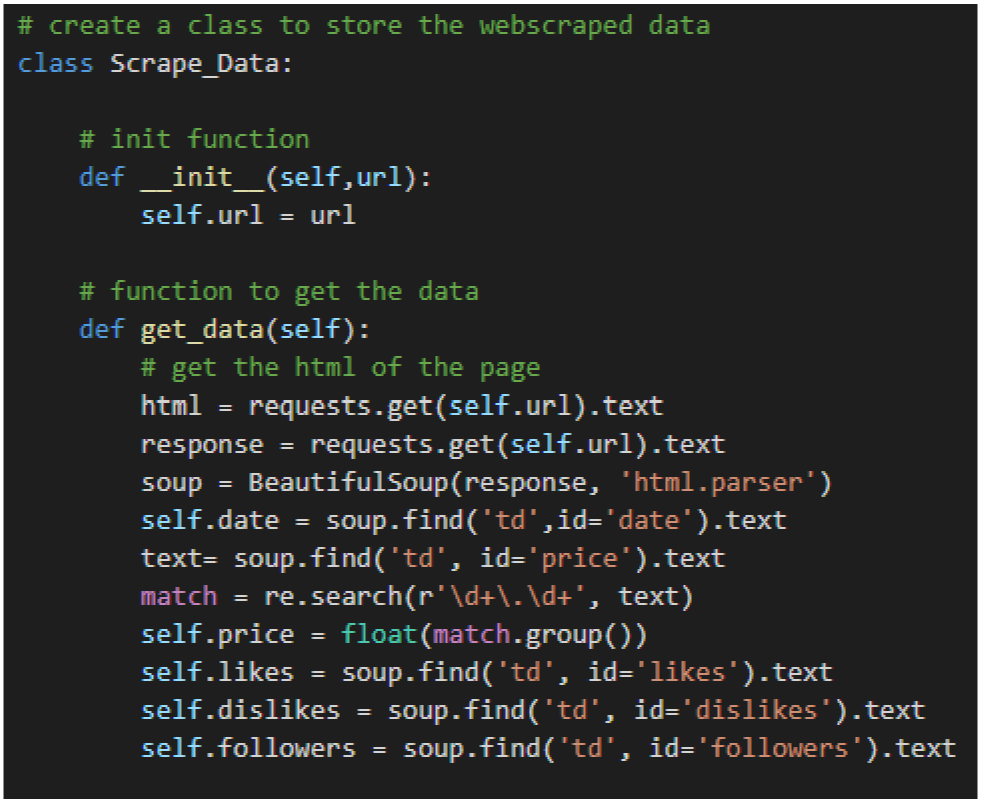
*Product prices are available since January, 1st 2019 to December 31st, 2023.

The website has a dynamic url where the pattern changes by the date. We can also see that the data has different fields (date, price, likes, dislikes, and followers). To start we would need to get all the url from the website by looping through all the dates.



```python
# for loop 2019-01-01 to 2023-12-31
urls = []
start_date = date(2019, 1, 1)
end_date = date(2023, 12, 31)
delta = timedelta(days=1)
while start_date <= end_date:
    urls.append("https://admn5015-340805.uc.r.appspot.com/"+start_date.strftime("%Y-%m-%d")+".html")
    start_date += delta
```

We then created a class to request the HTML doc and get the data from the desired webpage.



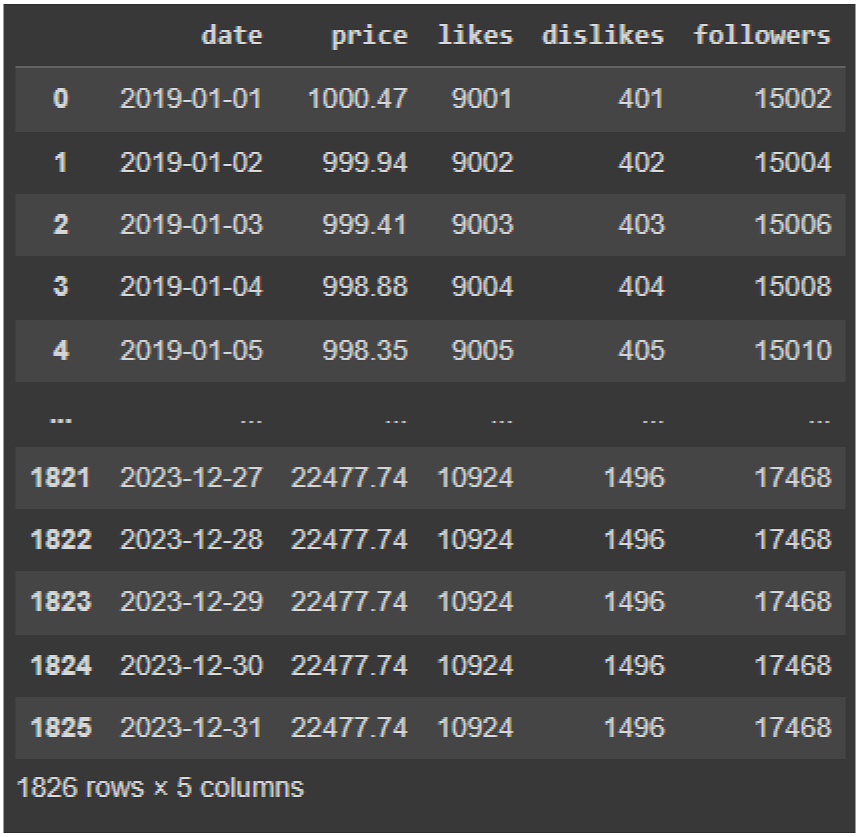
```python
# create a class to store the webscraped data
class Scrape_Data:

    # init function
    def __init__(self,url):
        self.url = url

    # function to get the data
    def get_data(self):
        # get the html of the page
        html = requests.get(self.url).text
        response = requests.get(self.url).text
        soup = BeautifulSoup(response, 'html.parser')
        self.date = soup.find('td',id='date').text
        text= soup.find('td', id='price').text
        match = re.search(r'\d+\.\d+', text)
        self.price = float(match.group())
        self.likes = soup.find('td', id='likes').text
        self.dislikes = soup.find('td', id='dislikes').text
        self.followers = soup.find('td', id='followers').text
```

We then extracted the data into the array and made it into a dataframe, where it will be formatted in csv.

The final data is now structured as a dataframe in which we can save it into a csv file.

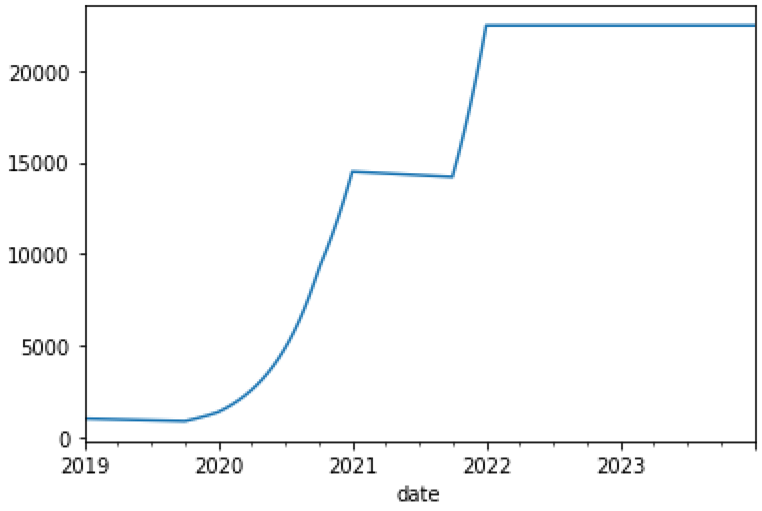


	date	price	likes	dislikes	followers
0	2019-01-01	1000.47	9001	401	15002
1	2019-01-02	999.94	9002	402	15004
2	2019-01-03	999.41	9003	403	15006
3	2019-01-04	998.88	9004	404	15008
4	2019-01-05	998.35	9005	405	15010
...	...	...	...	...	...
1821	2023-12-27	22477.74	10924	1496	17468
1822	2023-12-28	22477.74	10924	1496	17468
1823	2023-12-29	22477.74	10924	1496	17468
1824	2023-12-30	22477.74	10924	1496	17468
1825	2023-12-31	22477.74	10924	1496	17468

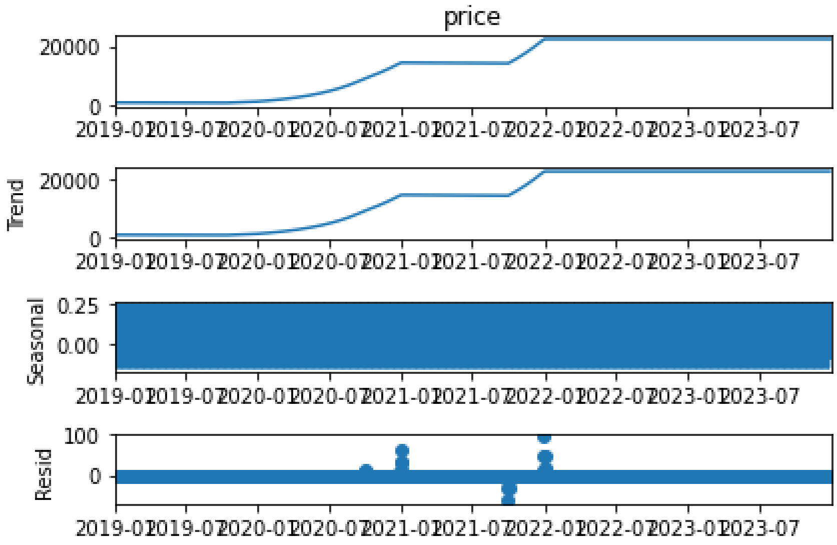
1826 rows × 5 columns

Data Exploration

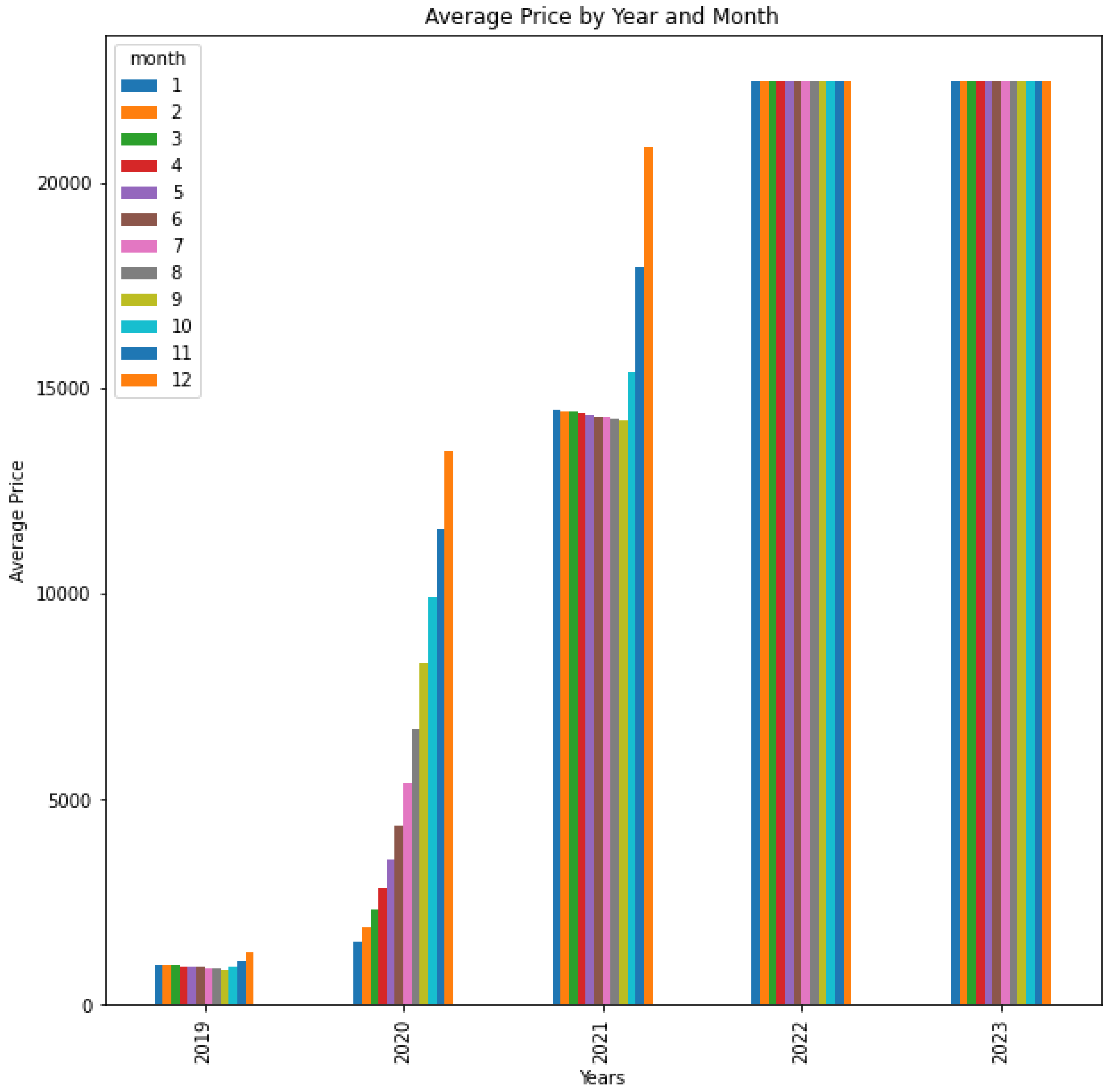
Looking at the data, over the years the product has had an increasing trend in price, from 2019 to 2022, and then remained the same from 2022 to the present.



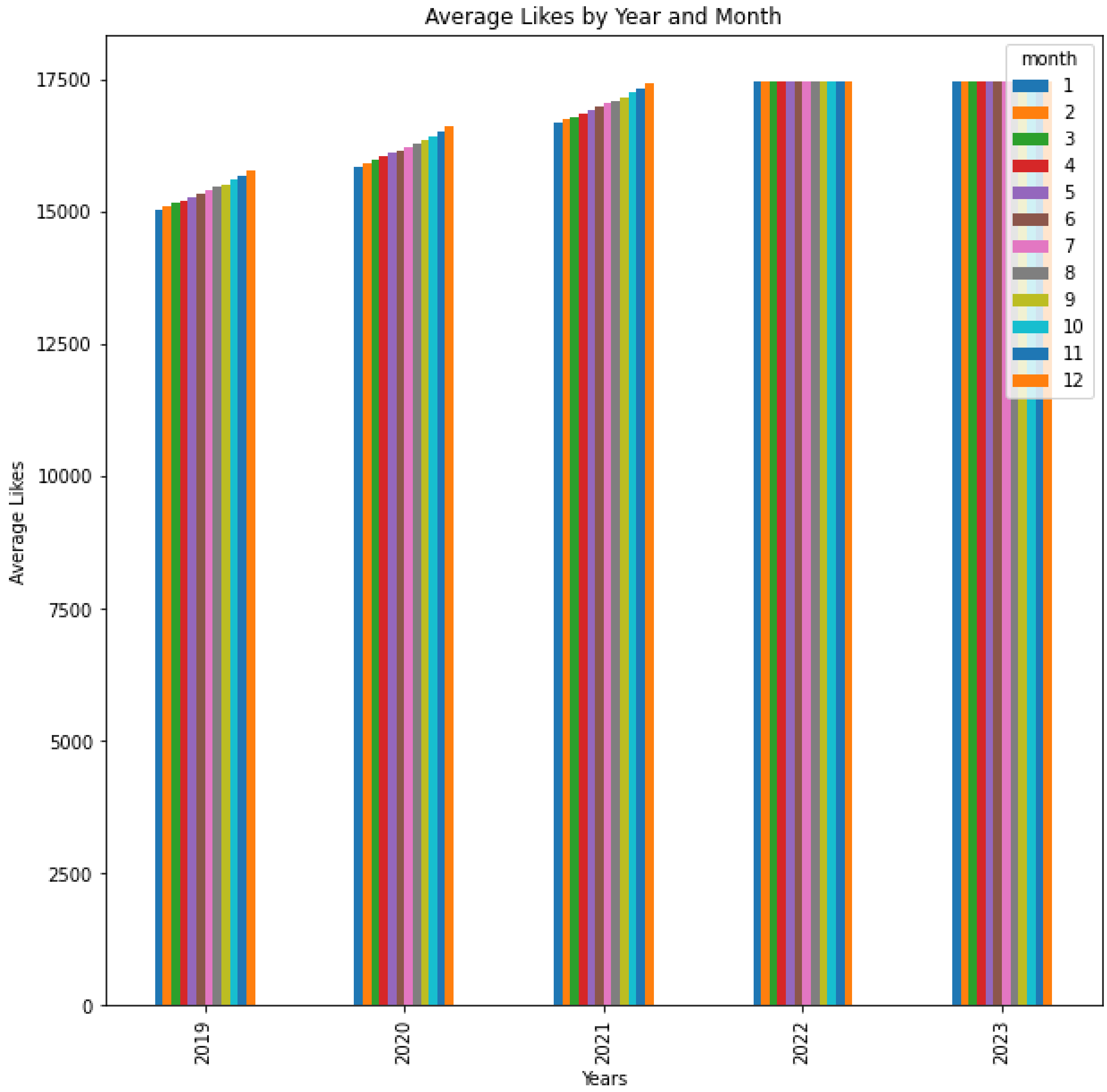
When we look at the seasonality of the price of the product it could be seen that the prices tend to change on a yearly basis.

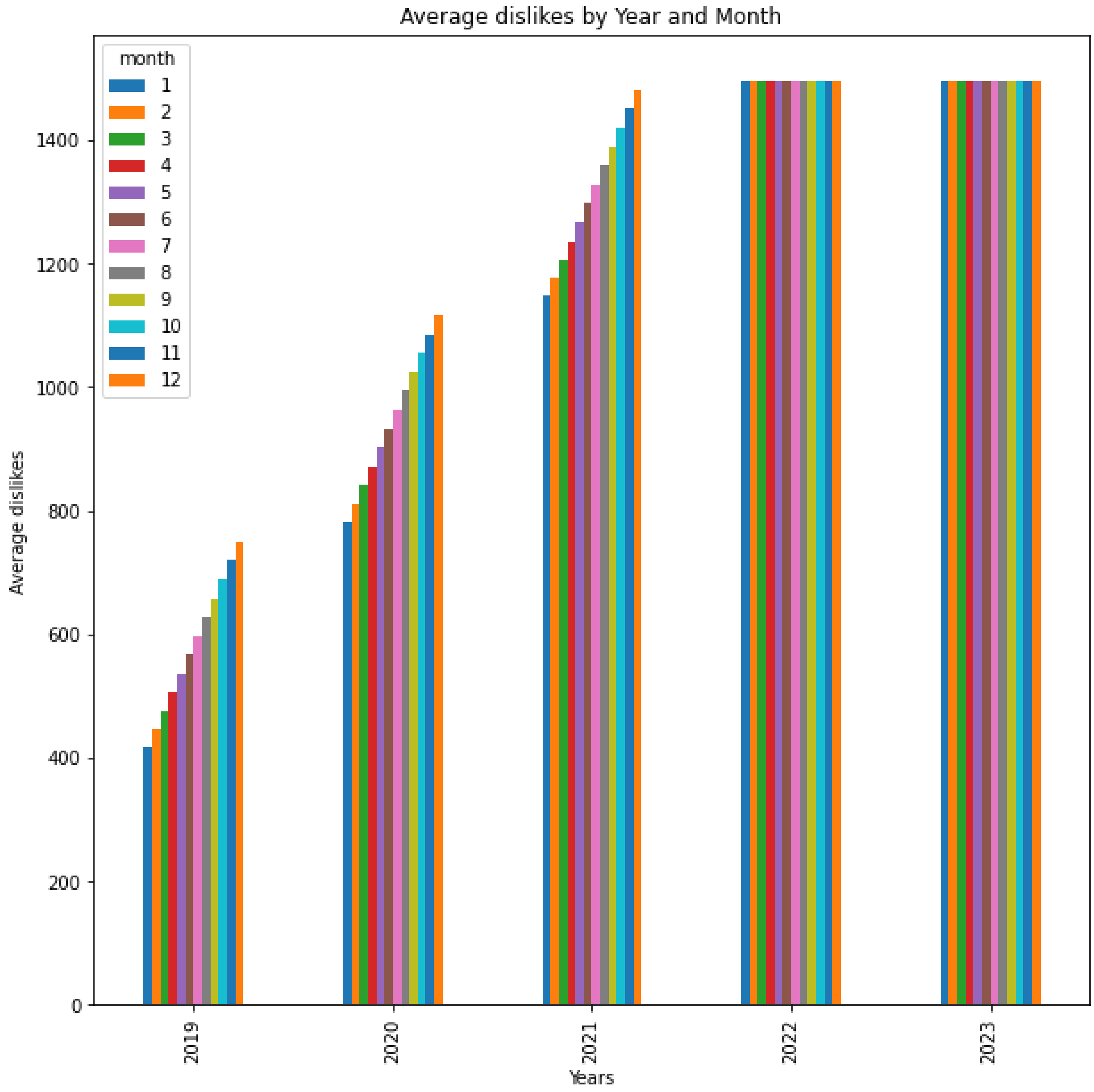


Judging the data from the average price by year and month we could see that in 2019 the average price is at its lowest and has increased in the last few months. Then in 2020 it keeps increasing at an exponential rate where it grows to almost three times the average price in 2019. Again in 2021 the trend increases in the last few months could be seen. Lastly, the prices in 2022 and 2023 have not changed and reached their peak.

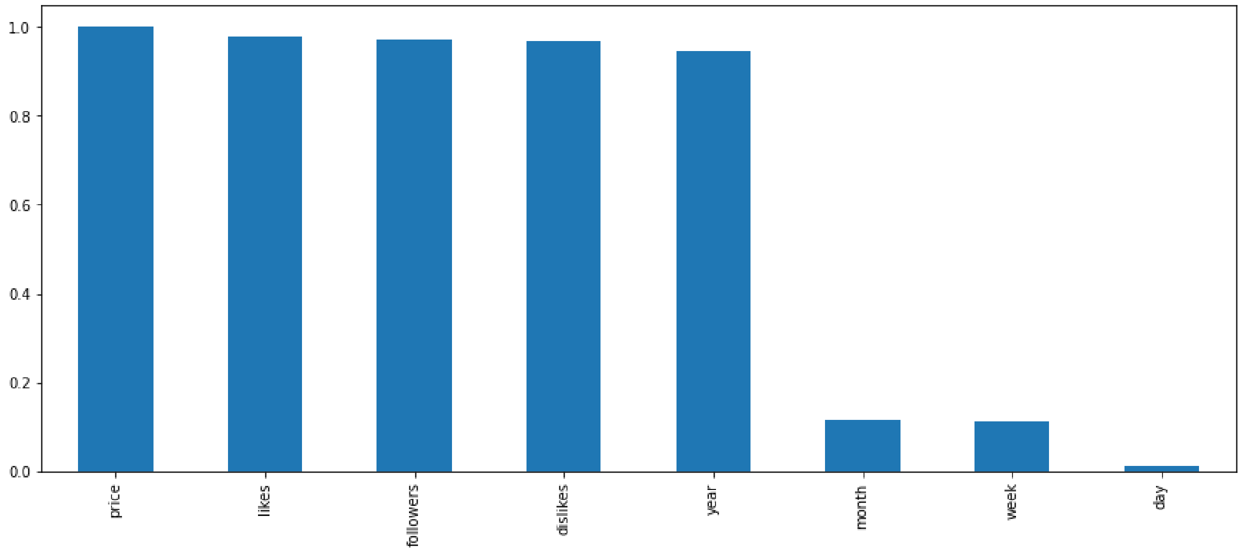


When it comes to the other factors of the product it could be seen that it has the same trend.


Average Likes by Year and Month
month
1
2
3
4
5
6
7
8
9
10
11
12
Average Likes
0
2500
5000
7500
10000
12500
15000
17500
2019
2020
2021
2022
2023
Years


Average dislikes by Year and Month
month
1
2
3
4
5
6
7
8
9
10
11
12
Average dislikes
0
200
400
600
800
1000
1200
1400
2019
2020
2021
2022
2023
Years

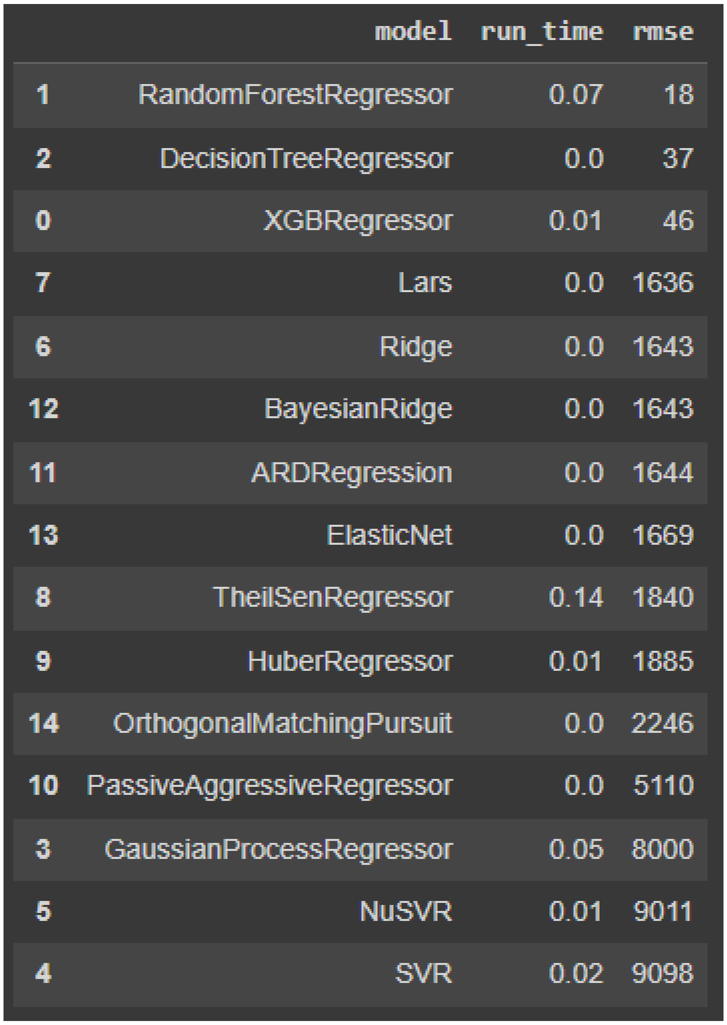
To prove this trend is the same, we can check it using a correlation where all the features have a high correlation to each other.



Modeling

In starting the model we have to split the data into training and testing data sets. For our model we are going to use the features (date, likes, dislikes, and followers) and our target would be the price. Our test data size would be 30% and 70% will be used for training.

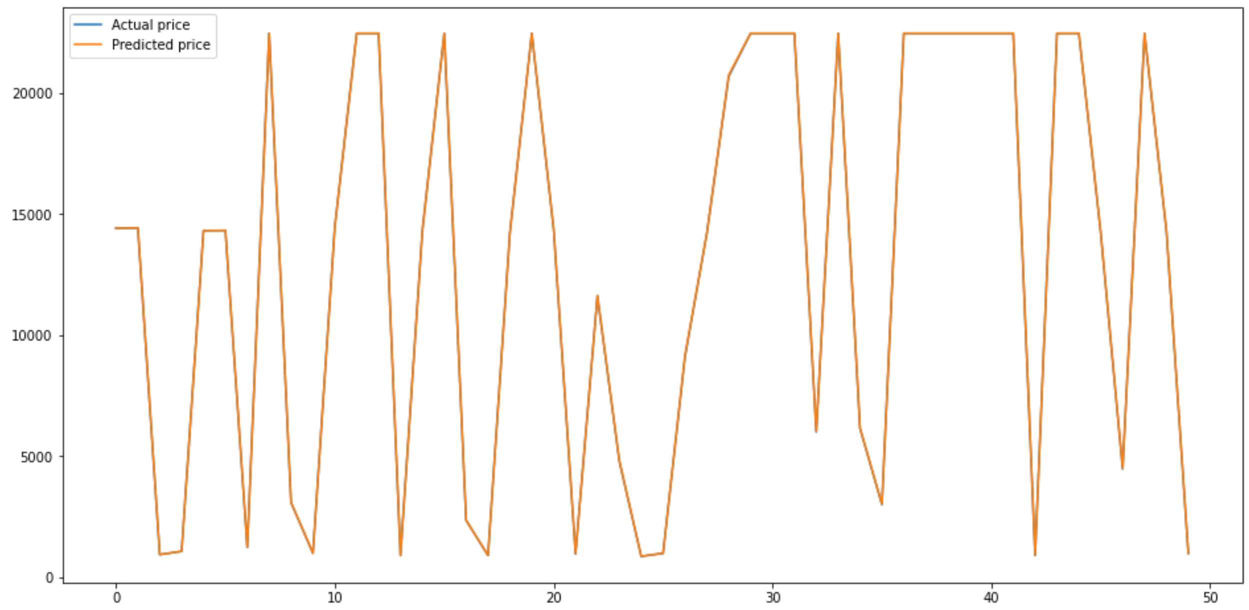
In regression we have to look at the rmse to check how well the model is learning. The figure below shows the model, run time and rmse of the different models we want to check.



	model	run_time	rmse
1	RandomForestRegressor	0.07	18
2	DecisionTreeRegressor	0.0	37
0	XGBRegressor	0.01	46
7	Lars	0.0	1636
6	Ridge	0.0	1643
12	BayesianRidge	0.0	1643
11	ARDRegression	0.0	1644
13	ElasticNet	0.0	1669
8	TheilSenRegressor	0.14	1840
9	HuberRegressor	0.01	1885
14	OrthogonalMatchingPursuit	0.0	2246
10	PassiveAggressiveRegressor	0.0	5110
3	GaussianProcessRegressor	0.05	8000
5	NuSVR	0.01	9011
4	SVR	0.02	9098

We have calculated the rmse which the RandomForestRegressor has the lowest rmse which has an rmse of 18.

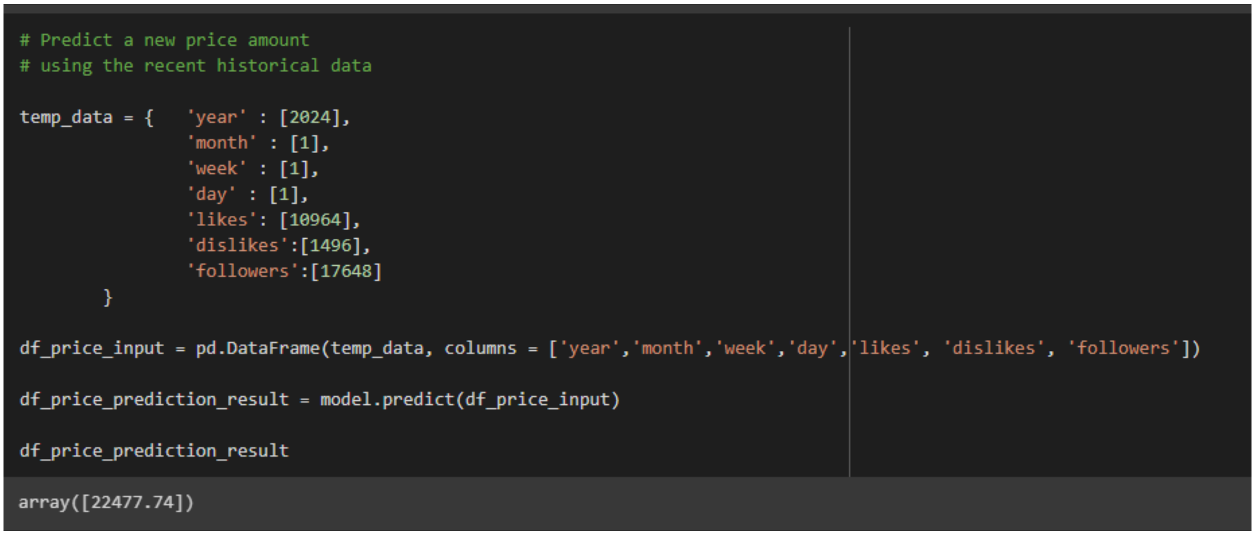
We then evaluate the model with the actual price and predicted price and has seen that the have almost the same values when it comes to RandomForestRegressor.



Prediction

For the prediction, using the model we added the values from the historical dataset which we can see that from the previous years there was no significant change when it comes to the price, likes, dislikes, and followers. We then just followed the trend and added those details for our temp_data for us to use.

From the data exploration earlier we can conclude that the data affecting the price is usually due to the number of likes, dislikes, and followers. As these features don’t change the price of the product doesn't entirely change.



```
# Predict a new price amount
# using the recent historical data

temp_data = {   'year' : [2024],
                'month' : [1],
                'week' : [1],
                'day' : [1],
                'likes': [10964],
                'dislikes':[1496],
                'followers':[17648]
        }

df_price_input = pd.DataFrame(temp_data, columns = ['year','month','week','day','likes', 'dislikes', 'followers'])

df_price_prediction_result = model.predict(df_price_input)

df_price_prediction_result
```

```
array([22477.74])
```