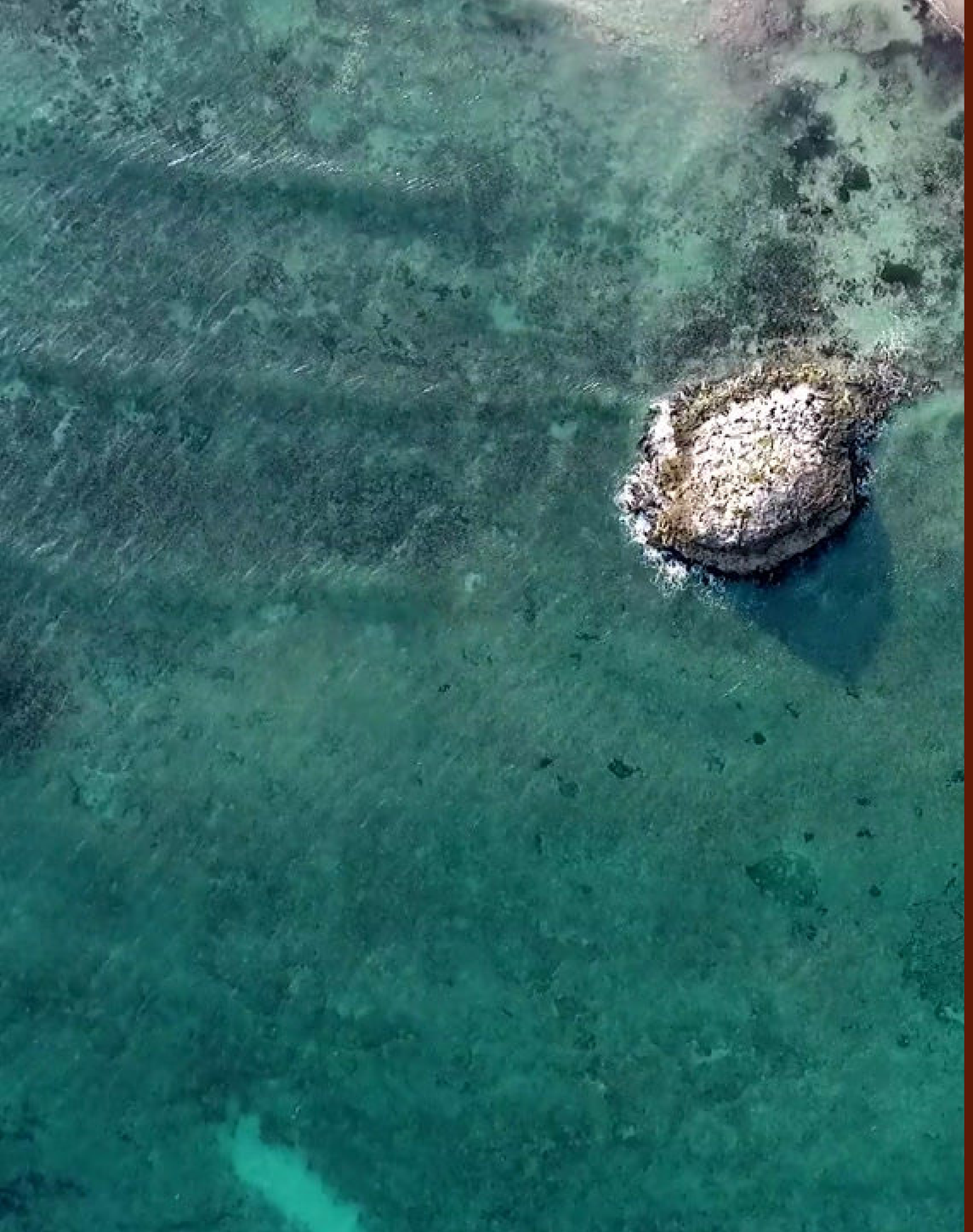


Waves Height Prediction

By Assaf Golani
GitHub

Research Intro

I like to surf and I follow the surf condition on daily basis, this is why I chose to answer this question, it's my passion.
Swell height means wave height
Swell period means wave period
Swell rate is the rate given by MSW to the swell according to their measurements

Research Question

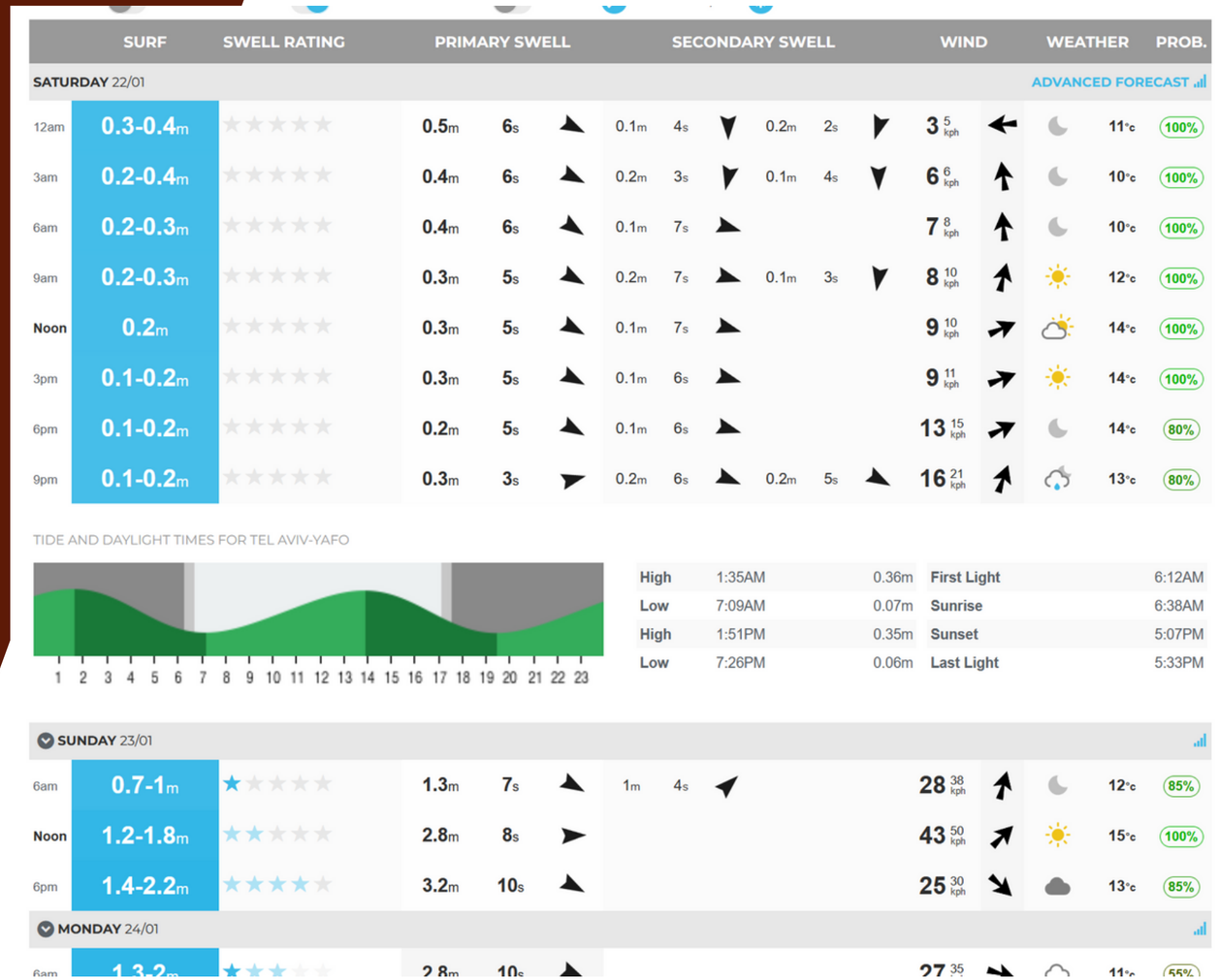
Is it possible to predict swell height by a given date according to the history of swell height date series?

Data source

I have crawled MSW site, In order to crawl history data from this site I had to create an account and buy the pro package which costs me 2.5$ per month.
The available data was from the 01/01/2020 up to this day but I have stopped crawling at the end of 20201.
I have retrieved data from 3 surf beaches that I like the most and which are close to each other.
https://magicseaweed.com/

MSW crawling

- I used both selenium and beautiful-soup to login and acquiring the data from MSW tables.
- I have created function to calculate end date of each month in order to insert them in the URL.



	SURF	SWELL RATING	PRIMARY SWELL	SECONDARY SWELL	WIND	WEATHER	PROB.
SATURDAY 22/01						ADVANCED FORECAST	
12am	0.3-0.4m		0.5m 6s	0.1m 4s 0.2m 2s	3 5 kph	11°c	100%
3am	0.2-0.4m		0.4m 6s	0.2m 3s 0.1m 4s	6 6 kph	10°c	100%
6am	0.2-0.3m		0.4m 6s	0.1m 7s	7 8 kph	10°c	100%
9am	0.2-0.3m		0.3m 5s	0.2m 7s 0.1m 3s	8 10 kph	12°c	100%
Noon	0.2m		0.3m 5s	0.1m 7s	9 10 kph	14°c	100%
3pm	0.1-0.2m		0.3m 5s	0.1m 6s	9 11 kph	14°c	100%
6pm	0.1-0.2m		0.2m 5s	0.1m 6s	13 15 kph	14°c	80%
9pm	0.1-0.2m		0.3m 3s	0.2m 6s 0.2m 5s	16 21 kph	13°c	80%

TIDE AND DAYLIGHT TIMES FOR TEL AVIV-YAFO

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

High	1:35AM	0.36m	First Light	6:12AM
Low	7:09AM	0.07m	Sunrise	6:38AM
High	1:51PM	0.35m	Sunset	5:07PM
Low	7:26PM	0.06m	Last Light	5:33PM

SUNDAY 23/01							
6am	0.7-1m		1.3m 7s	1m 4s	28 38 kph	12°c	85%
Noon	1.2-1.8m		2.8m 8s		43 50 kph	15°c	100%
6pm	1.4-2.2m		3.2m 10s		25 30 kph	13°c	85%
MONDAY 24/01							
6am	1.3-2m		2.8m 10s		27 35	11°c	55%

```python
#login
def login(self):
    email = self.find_element(By.NAME, 'email')
    password = self.find_element(By.NAME, 'password')
    email.send_keys("")
    password.send_keys("")
    self.find_element(By.ID,"msw-js-login").click()
    time.sleep(5)
```

```python
### code ref: https://stackoverflow.com/a/13565185/8703160
def end_of_month(any_day):
    # this will never fail
    # get close to the end of the month for any day, and add 4 days 'over'
    next_month = any_day.replace(day=28) + datetime.timedelta(days=4)
    # subtract the number of remaining 'overage' days to get last day of curre
    return next_month - datetime.timedelta(days=next_month.day)

def end_date_of_month_gen():
    end_arr = []
    for month in range(1, 13):
        for year in range (2020, 2022):
            end_date = (end_of_month(datetime.date(year, month, 1)))
            end_arr.append(str(end_date))
    return end_arr
```

```python
def start_day_of_month():
    start_arr = []
    for month in range (1,13):
        for year in range(2020,2022):
            start_day_of_month = datetime.date(year, month, 1)
            start_arr.append(str(start_day_of_month))
    return start_arr
```

```python
def scrape_data(soup):
    date = []
    # hours = ['00:00', '03:00', '06:00', '09:00', '12:00', '15:00', '18:00','21:00']
    hours = []
    primarySwellHeight = []
    primarySwellPeriod = []
    secondarySwellHeight = []
    secondarySwellPeriod = []
    swellRating = []
#     for i in range (1,9):
#         for day in soup.find_all('h6', {"class" : "nomargin pull-left heavy table-header-title"}):
#             date.append(day.find('small').get_text().strip())
    table  = soup.find('table', {'class':'table table-primary table-forecast allSwellsActive msw-js-table msw-units-large'})
    for tbody in table.find_all('tbody'):
        for tr in tbody.find_all('tr'):
            if tr.has_attr('data-timestamp'):
                date.append(time.strftime('%Y-%m-%d %H:%M', time.localtime(int(tr['data-timestamp']))))
            for td_time in tr.find_all('td', {'class':'nopadding-left row-title background-clear msw-js-tooltip'}):
                hours.append(td_time.find('small').get_text().strip())


            for i, td_swellHeight in enumerate(tr.find_all('td', {'class':['text-center background-gray-lighter','text-center background-gr
                if i % 2 == 0:
                    for h4_info in td_swellHeight.find_all('h4'):
                        primarySwellHeight.append(h4_info.text.strip())
                elif i % 2 != 0:
                    for h4_info in td_swellHeight.find_all('h4'):
                        primarySwellPeriod.append(h4_info.text.strip())
                else:
                    continue

#             for i, td_swellHeight in enumerate(tr.find_all('td', {'class':'text-center table-forecast-sub-swells'})):
#                 if i % 2 == 0:
#                     for h4_info in td_swellHeight.find_all('h5'):
#                         secondarySwellHeight.append(h4_info.text.strip())
#                 elif i % 2 != 0:
#                     for h4_info in td_swellHeight.find_all('h5'):
#                         secondarySwellPeriod.append(h4_info.text.strip())
            count = 0
            for td_rate in tr.find_all('ul'):
                count = len(td_rate.find_all('li', {'class':'active'}))
                swellRating.append(count)
    return pd.DataFrame({'date': pd.Series(date),'hour': pd.Series(hours), 'swell_rating': pd.Series(swellRating), 'primary_swell_height':
```

Final Data Frame after retreiveing all the data:

	date	hour	swell_rating	primary_swell_height	primary_swell_period
0	2020-01-01 02:00	12am	1	1m	8s
1	2020-01-01 05:00	3am	0	1m	7s
2	2020-01-01 08:00	6am	1	0.9m	7s
3	2020-01-01 11:00	9am	1	0.8m	7s
4	2020-01-01 14:00	12pm	0	0.8m	7s
...	...	...	...	...	...
17538	2021-12-31 09:00	9am	1	1.1m	7s
17539	2021-12-31 12:00	12pm	1	1m	8s
17540	2021-12-31 15:00	3pm	1	0.9m	8s
17541	2021-12-31 18:00	6pm	1	0.9m	8s
17542	2021-12-31 21:00	9pm	0	0.8m	8s

17543 rows × 5 columns

Data Cleaning

Removing unecessery column like unamed:0 and hour, cleaning the height and period units, transformed type of columns to int/float

	Unnamed: 0	date	hour	swell_rating	primary_swell_height	primary_swell_period
0	0	2020-01-01 02:00	12am	1	1m	8s
1	1	2020-01-01 05:00	3am	0	1m	7s
2	2	2020-01-01 08:00	6am	1	0.9m	7s
3	3	2020-01-01 11:00	9am	1	0.8m	7s
4	4	2020-01-01 14:00	12pm	0	0.8m	7s

Clean info for ML

	date	swell_rating	primary_swell_height	primary_swell_period
0	2020-01-01 02:00:00	1	1	8
1	2020-01-01 05:00:00	0	1	7
2	2020-01-01 08:00:00	1	0.9	7
3	2020-01-01 11:00:00	1	0.8	7
4	2020-01-01 14:00:00	0	0.8	7

Removing unecessery column like unamed:0 and hour, cleaning the height and period units, transformed type of columns to int/float, date is now splitted to different columns

	Unnamed: 0	date	hour	swell_rating	primary_swell_height	primary_swell_period
0	0	2020-01-01 02:00	12am	1	1m	8s
1	1	2020-01-01 05:00	3am	0	1m	7s
2	2	2020-01-01 08:00	6am	1	0.9m	7s
3	3	2020-01-01 11:00	9am	1	0.8m	7s
4	4	2020-01-01 14:00	12pm	0	0.8m	7s

clean info for EDA

	hour	swell_rating	primary_swell_height	primary_swell_period	year	month	day
0	00	1	1.0	8	2020	1	1
1	03	0	1.0	7	2020	1	1
2	06	1	0.9	7	2020	1	1
3	09	1	0.8	7	2020	1	1
4	12	0	0.8	7	2020	1	1

Removing unecessery column like unamed:0 and hour, cleaning the height and period units, transformed type of columns to int/float, date is now index

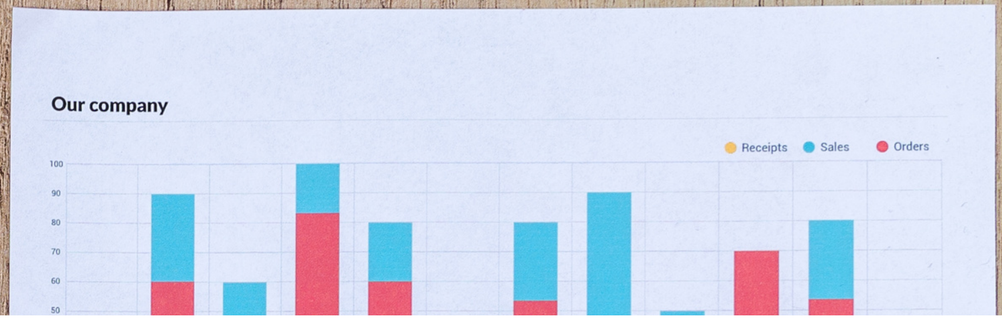
	Unnamed: 0	date	hour	swell_rating	primary_swell_height	primary_swell_period
0	0	2020-01-01 02:00	12am	1	1m	8s
1	1	2020-01-01 05:00	3am	0	1m	7s
2	2	2020-01-01 08:00	6am	1	0.9m	7s
3	3	2020-01-01 11:00	9am	1	0.8m	7s
4	4	2020-01-01 14:00	12pm	0	0.8m	7s

clean AVG info for EDA

date	swell_rating	primary_swell_height	primary_swell_period
2020-01-01	0.875000	0.816667	7.125000
2020-01-02	0.000000	0.704167	5.833333
2020-01-03	0.041667	1.483333	6.791667
2020-01-04	0.000000	1.754167	7.666667
2020-01-05	0.000000	1.658333	7.375000

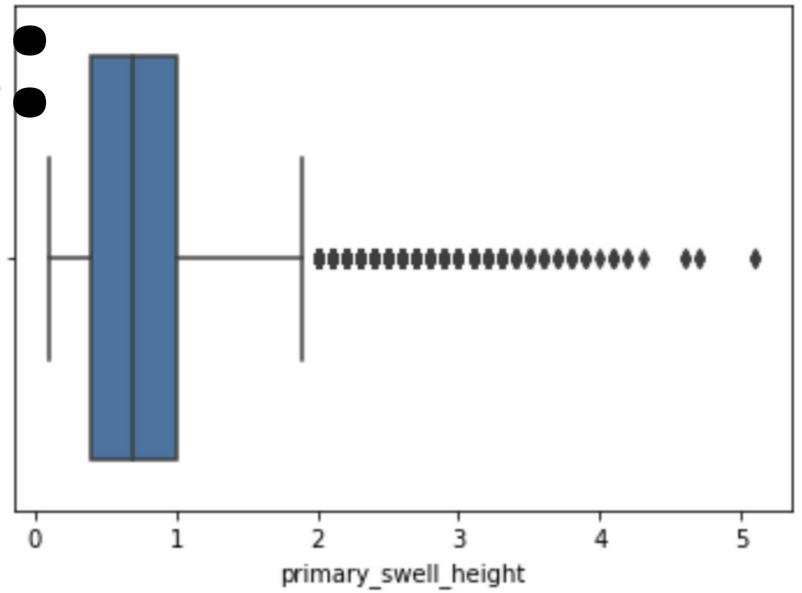
Outliers:

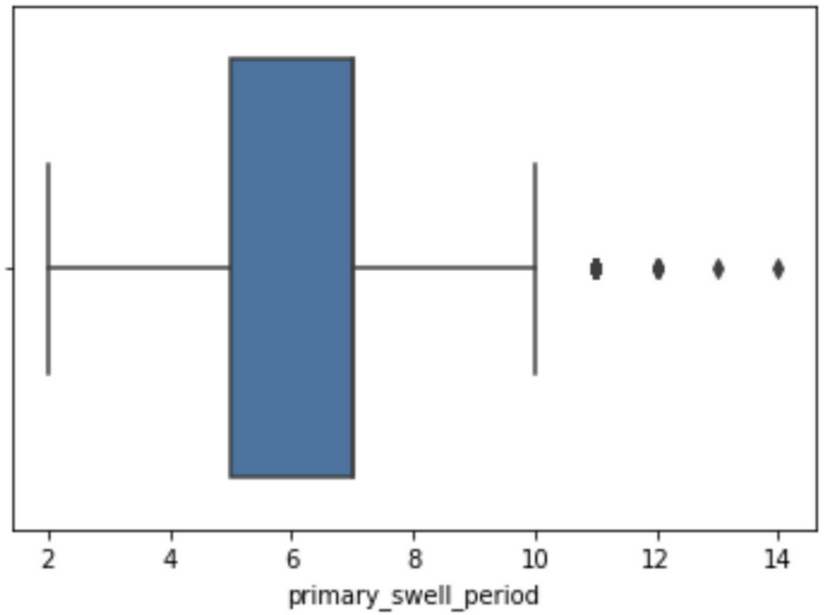
I dropped outliers using IQR.



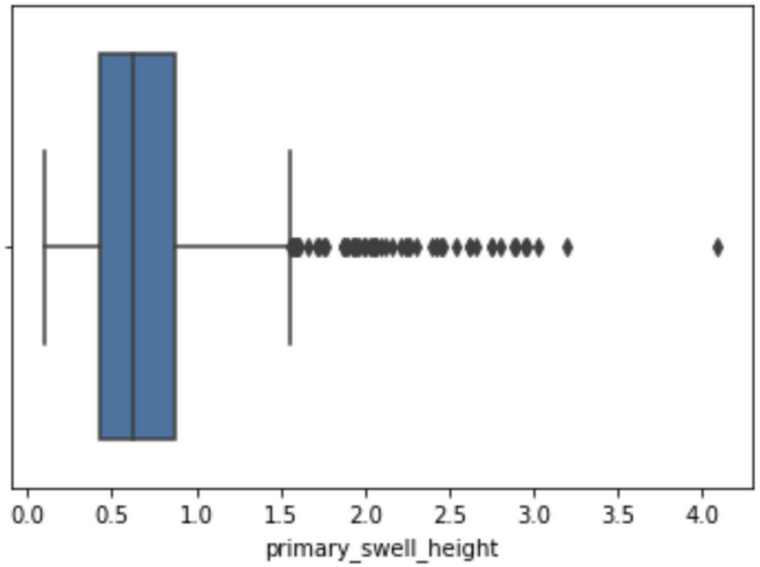

```
def outliers(df,col_name):
    sns.boxplot(x = df[col_name])
    Q1 = np.percentile(df[col_name], 25)
    Q3 = np.percentile(df[col_name], 75)
    IQR = Q3-Q1
    if col_name == 'primary_swell_period':
        IQR_range = 7
        df[col_name][(df[col_name] < Q1 - 1.5*IQR ) | (df[col_name] > Q3 + IQR_range*IQR)] = np.nan

    if col_name == 'primary_swell_height':
        IQR_range = 4
        df[col_name][(df[col_name] < Q1 - 1.5*IQR ) | (df[col_name] > Q3 + IQR_range*IQR)] = np.nan

    df = df[df[col_name].notna()]

    return df
```

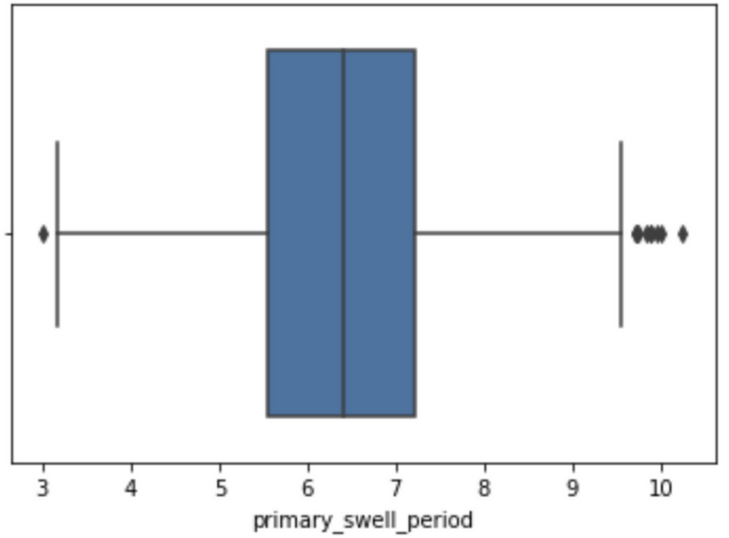
Outliers:





Avarage:



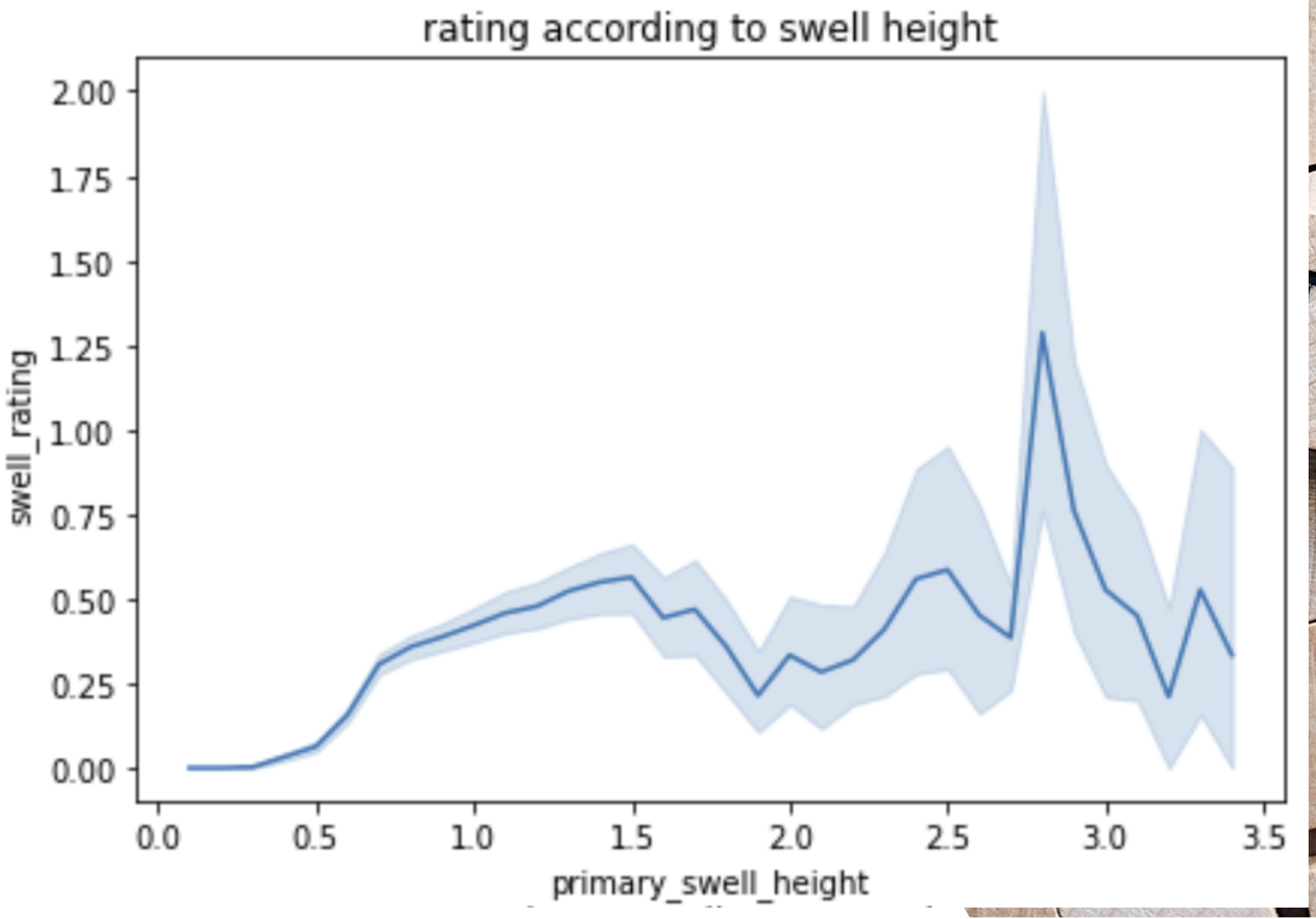


Final Dataframe for ML:

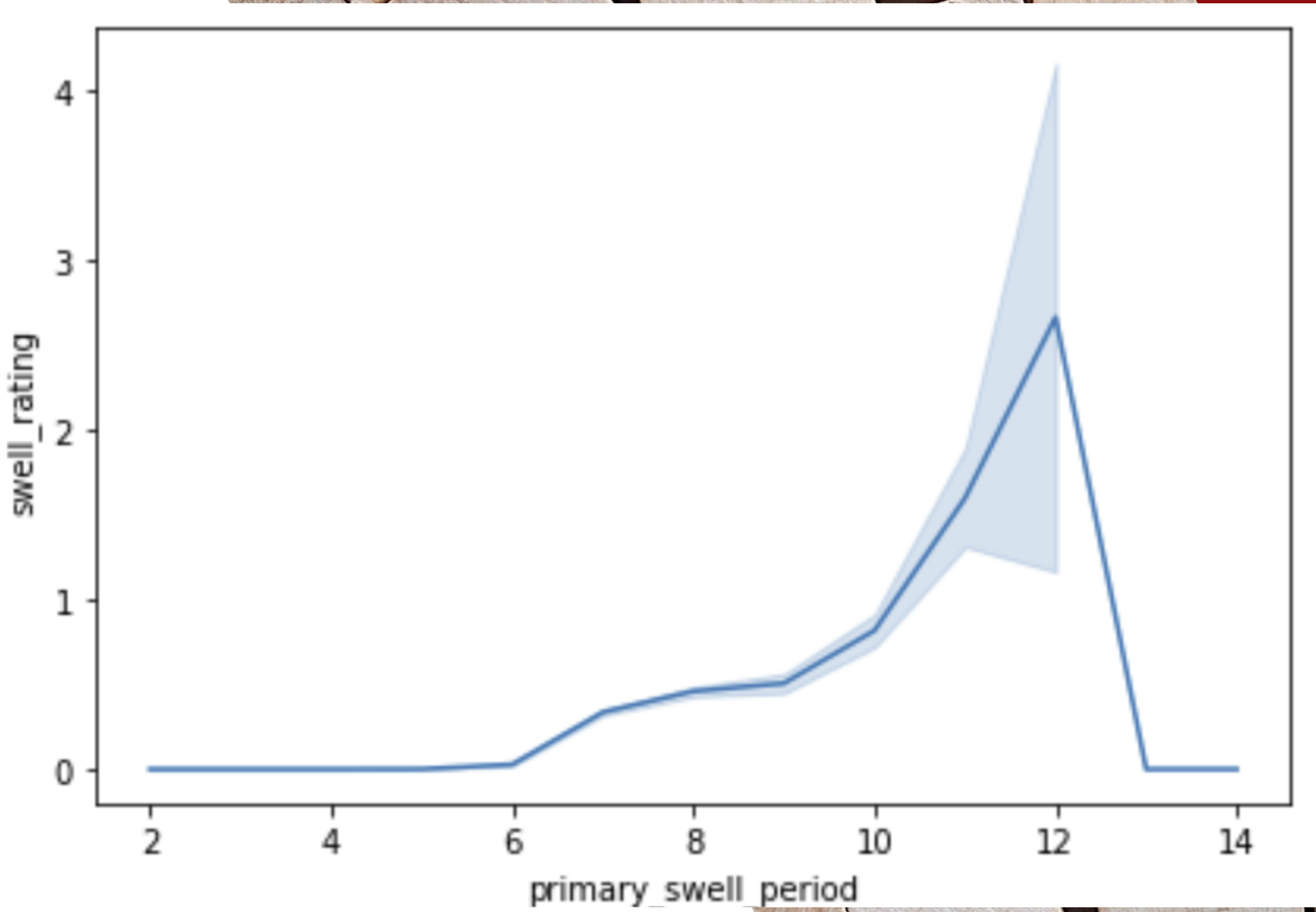
	date	swell_rating	primary_swell_height	primary_swell_period
0	2020-01-01 02:00:00	1	1.0	8
1	2020-01-01 05:00:00	0	1.0	7
2	2020-01-01 08:00:00	1	0.9	7
3	2020-01-01 11:00:00	1	0.8	7
4	2020-01-01 14:00:00	0	0.8	7
...	...	...	...	...
17532	2021-12-30 15:00:00	1	0.7	8
17534	2021-12-30 21:00:00	0	0.9	7
17538	2021-12-31 09:00:00	1	1.1	7
17540	2021-12-31 15:00:00	1	0.9	8
17541	2021-12-31 18:00:00	1	0.9	8

EDA

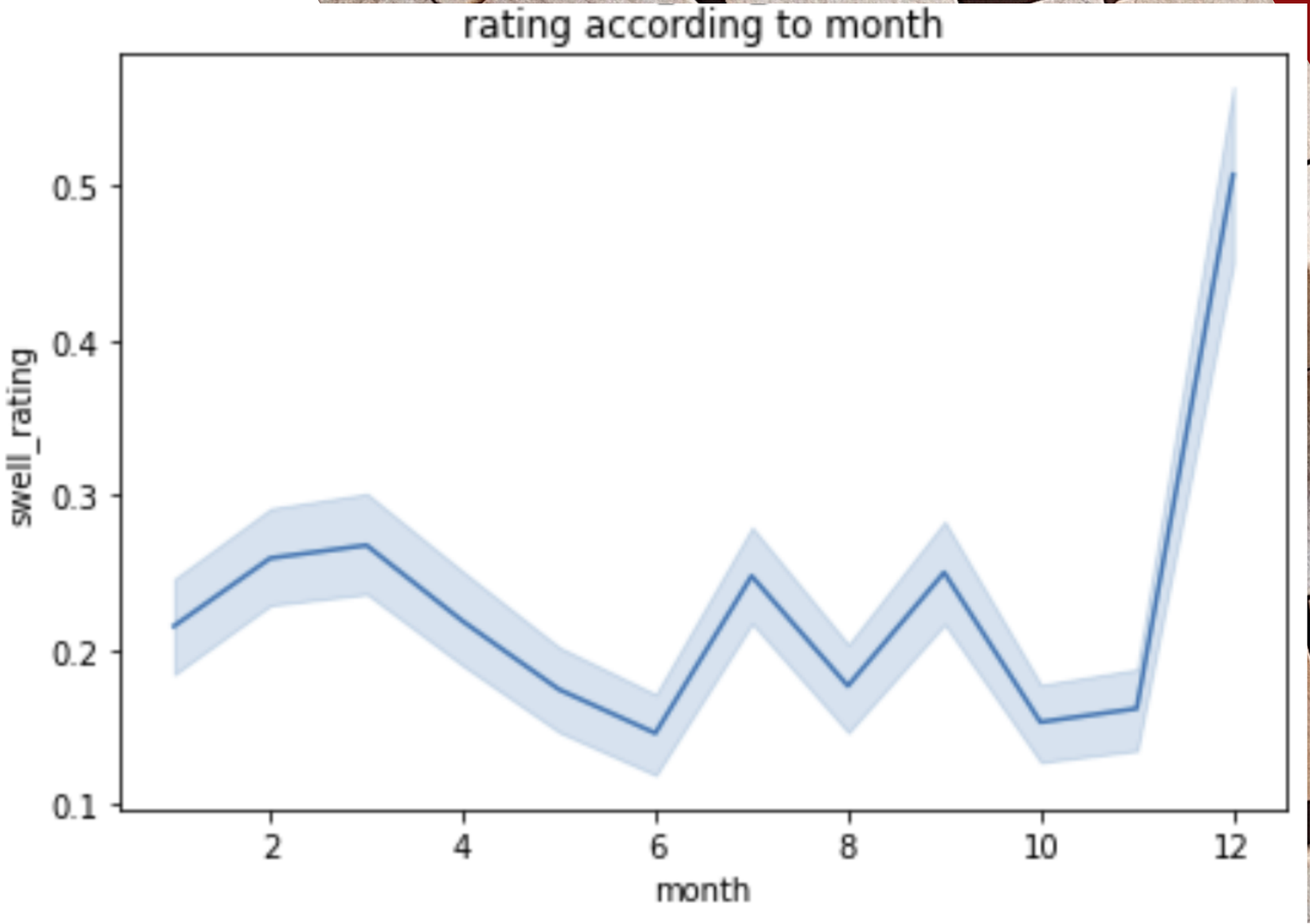
Swell Height/ Swell Rate



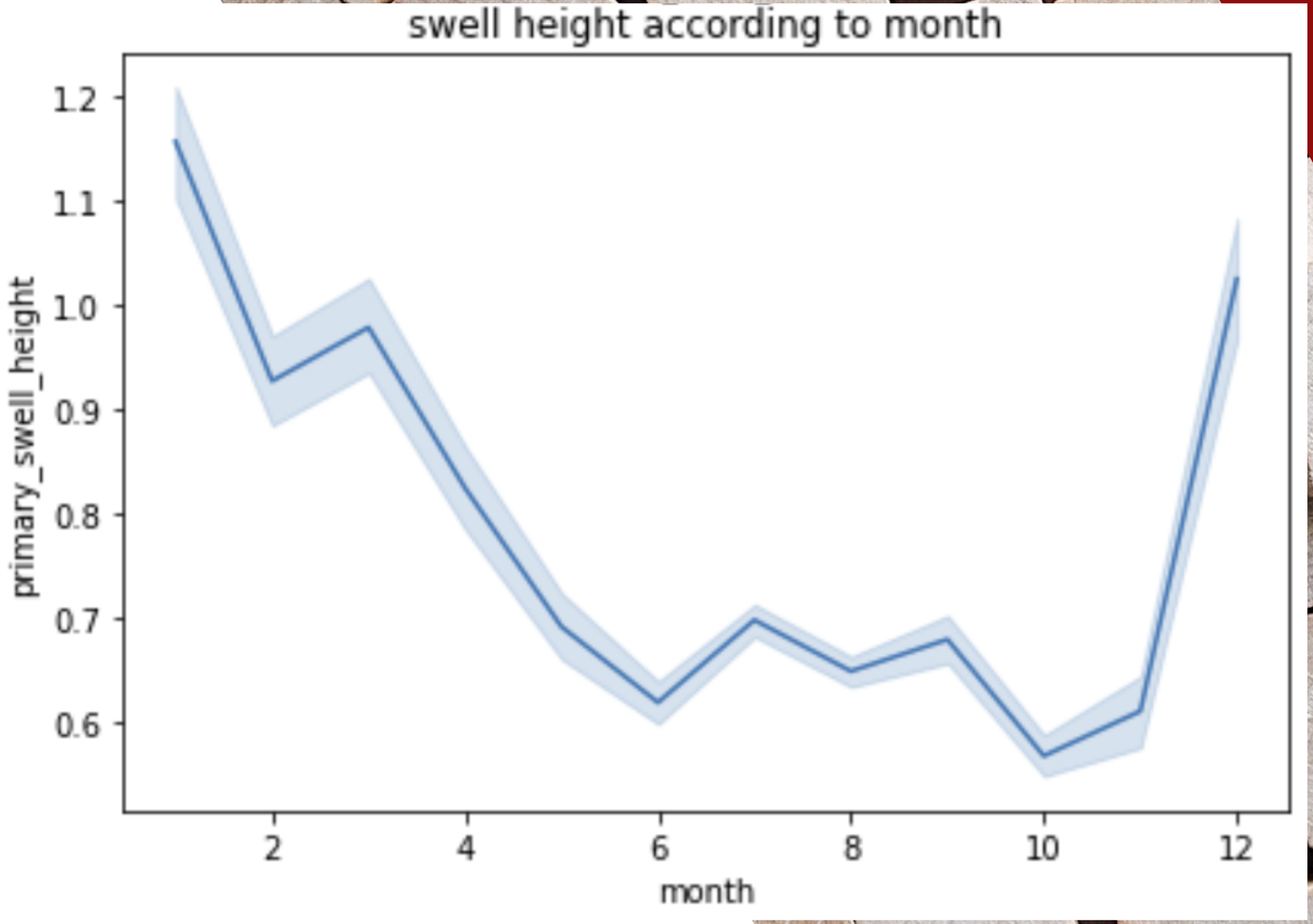
Swell Period/ Swell Rate



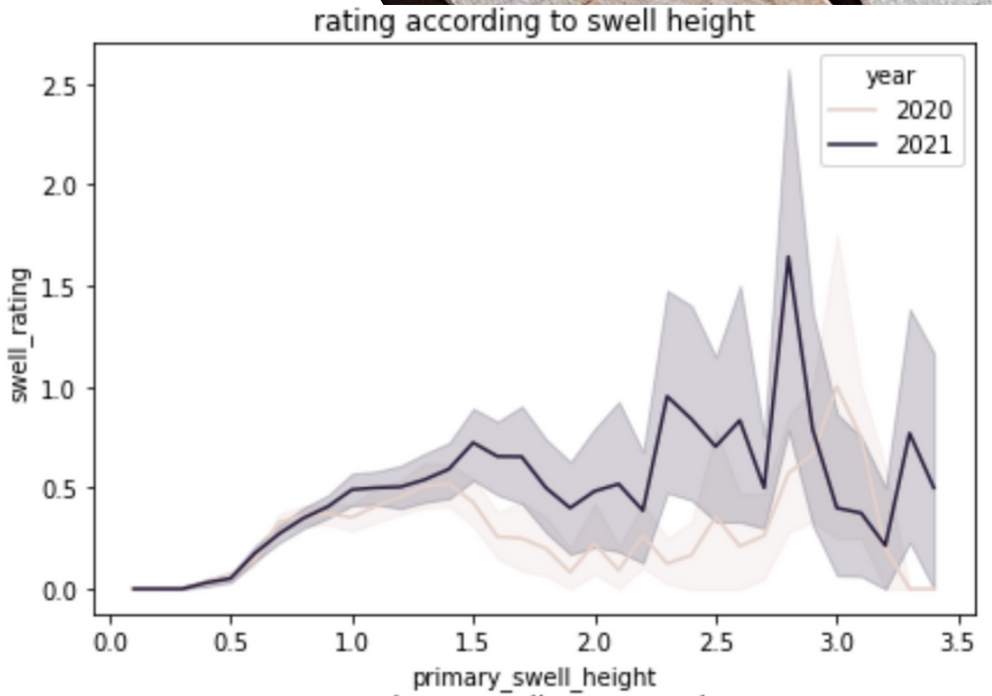
Month/ Swell Rate

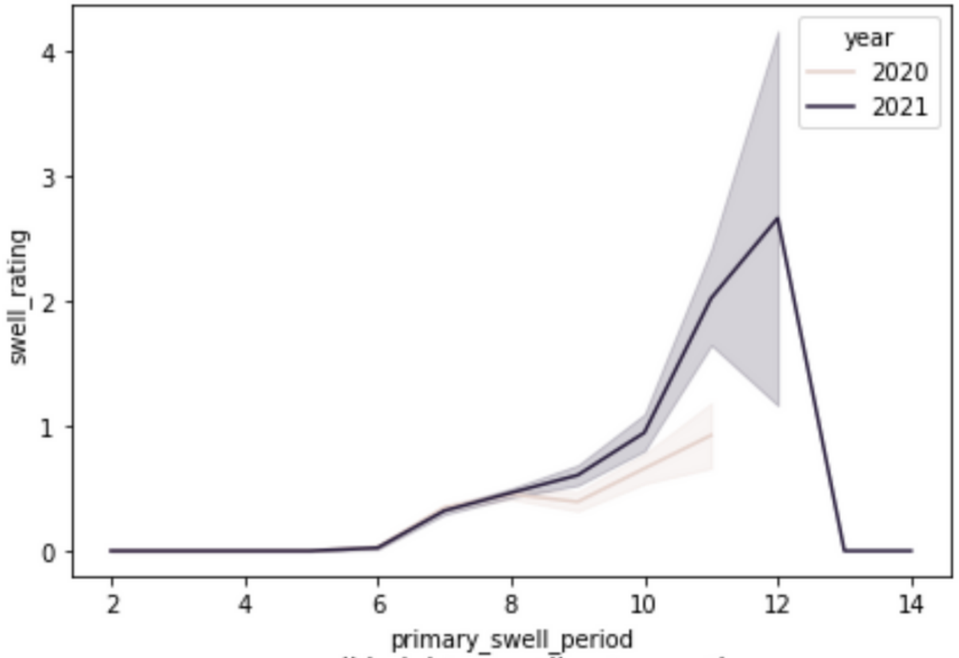


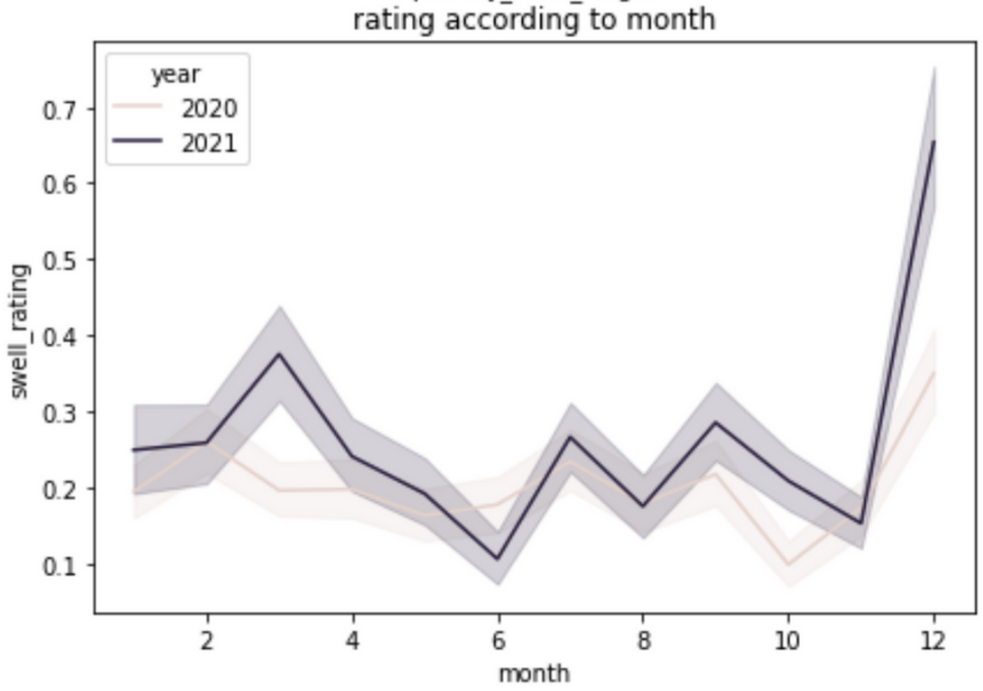
Month/ Swell Height

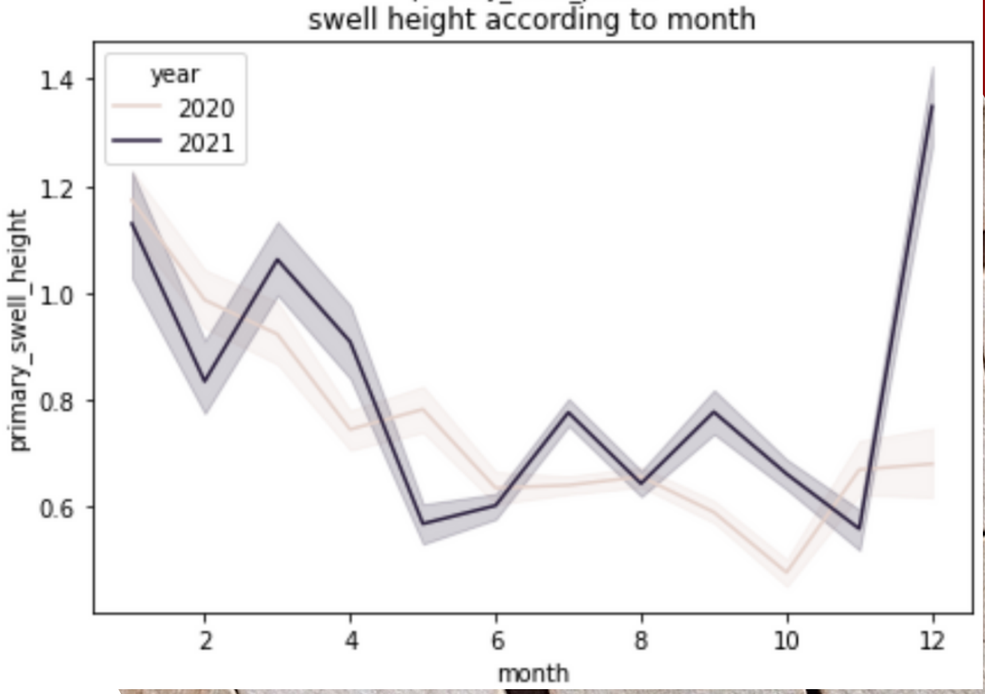


Stats compared by year

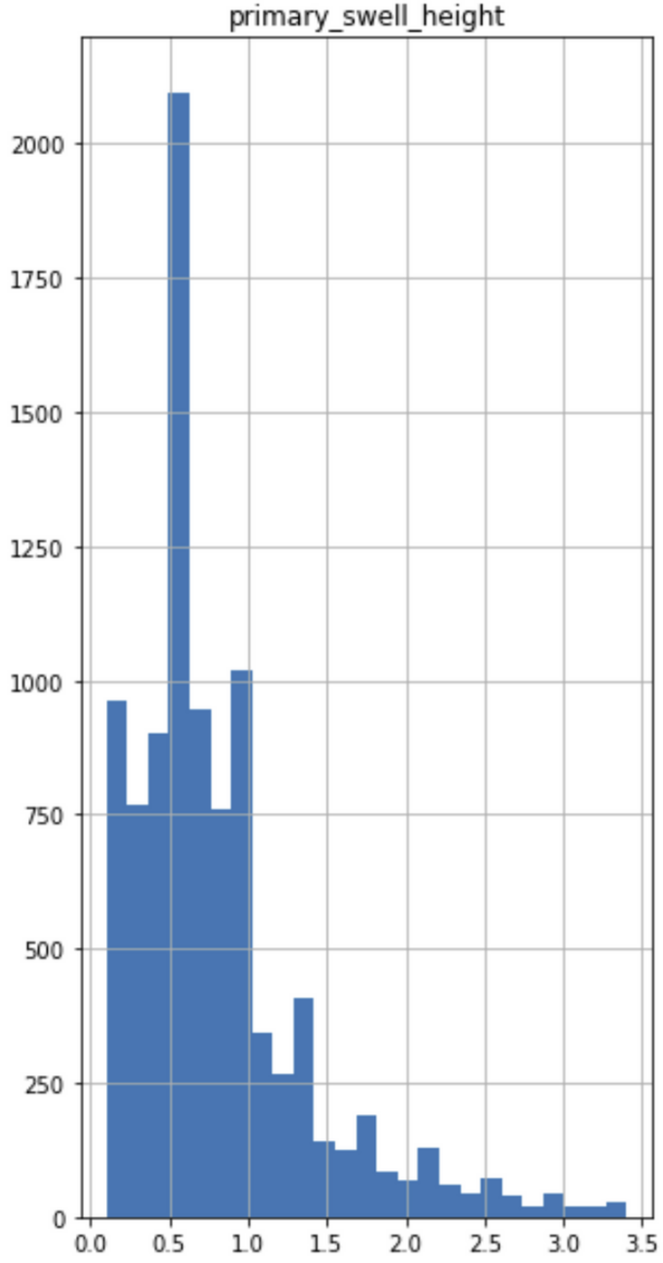


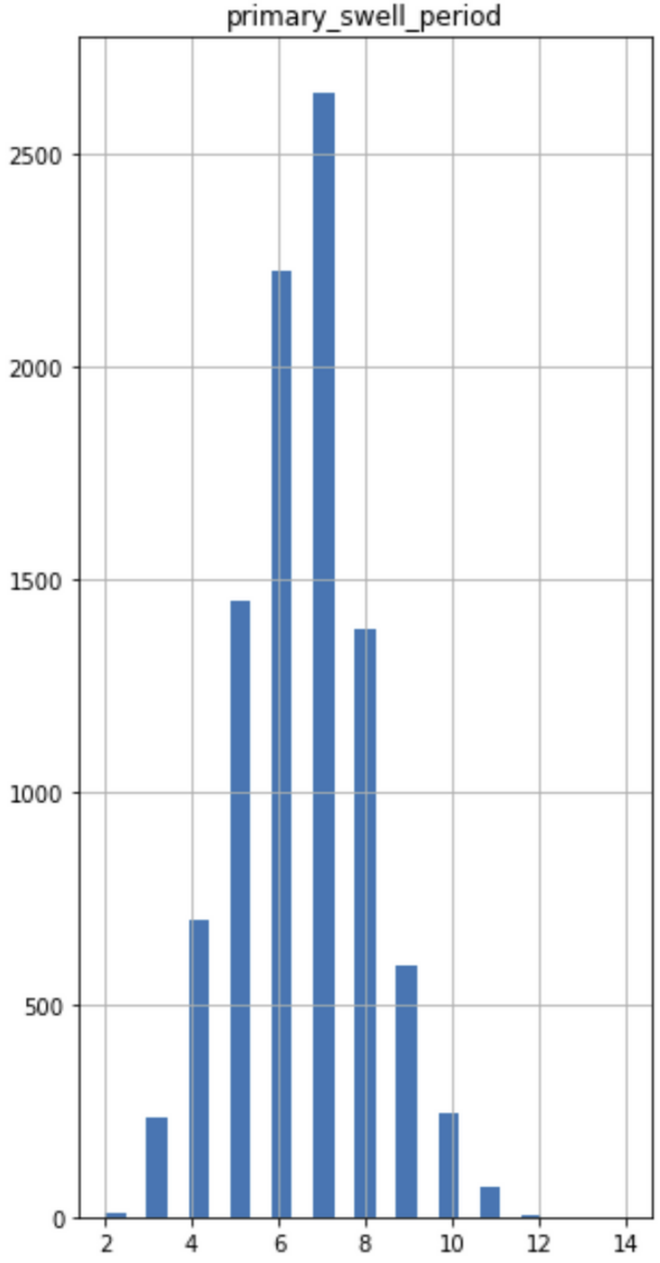


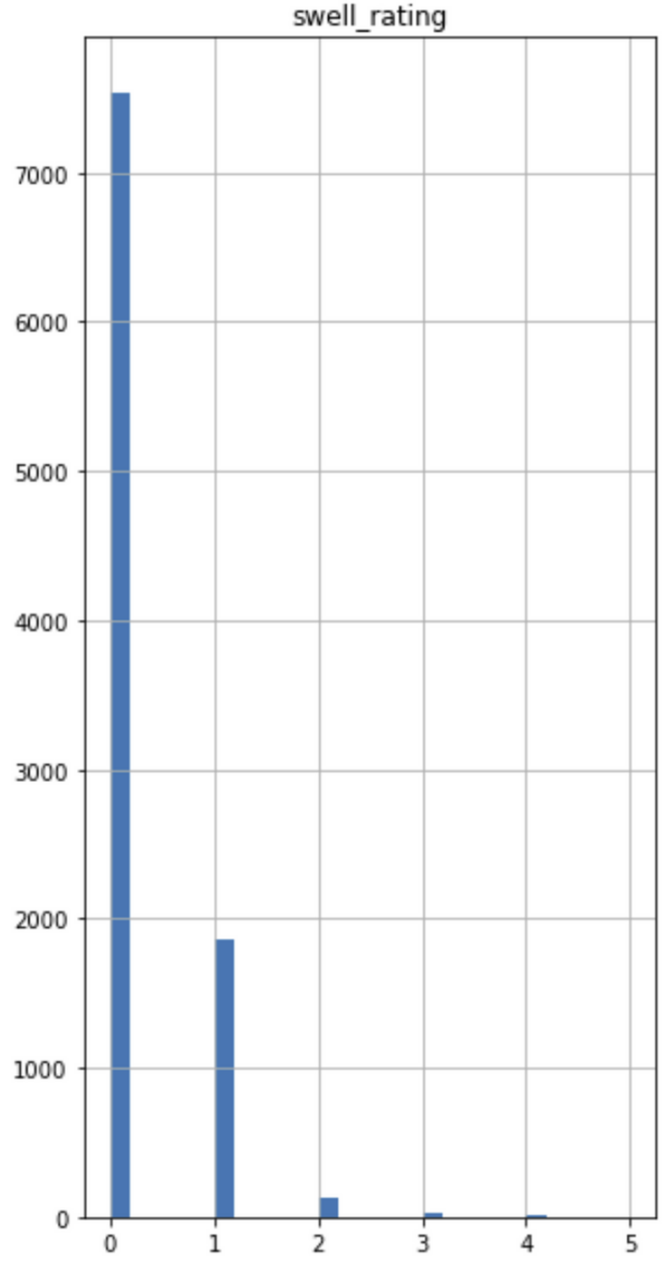




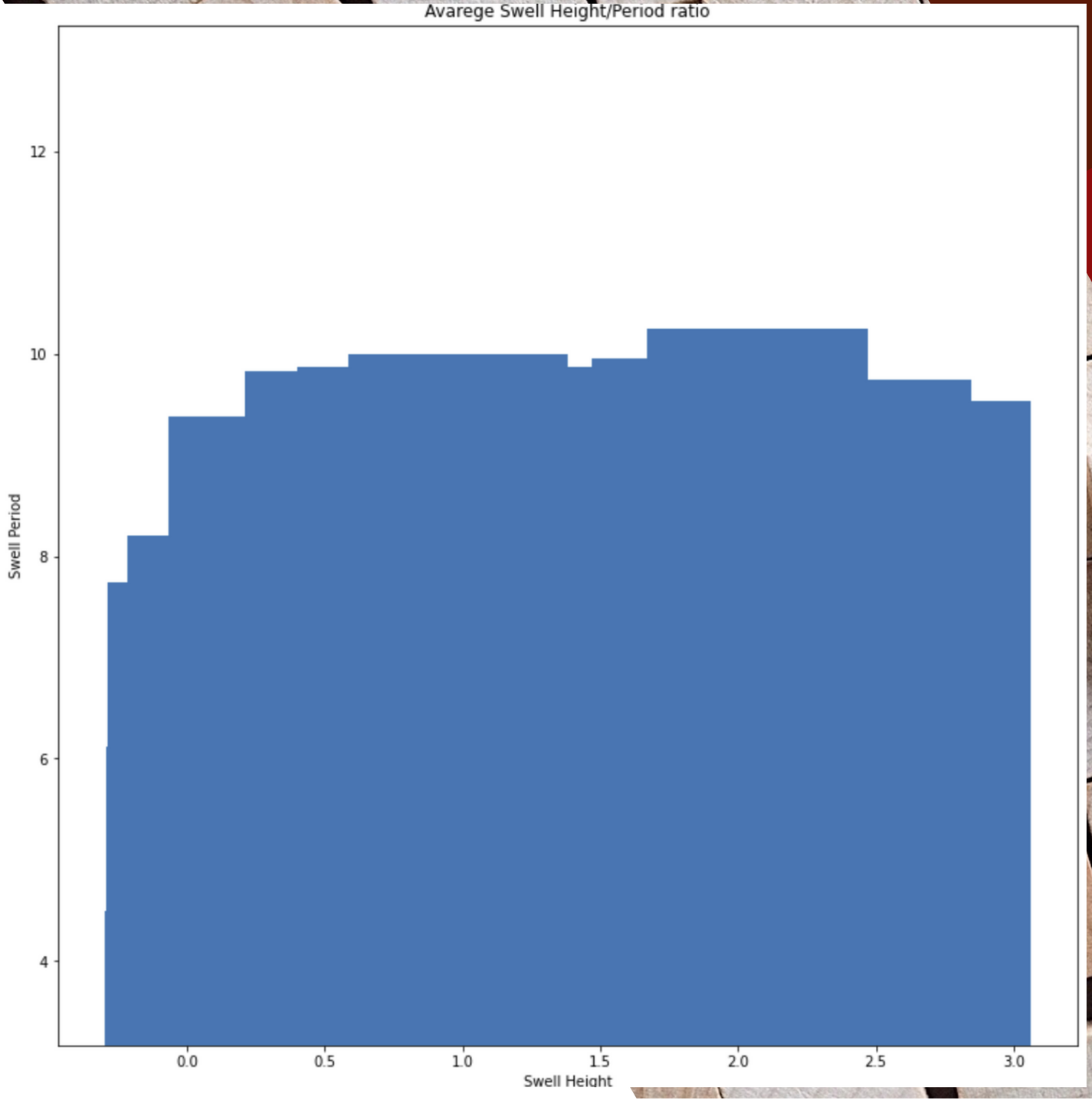
Histogram of stats:



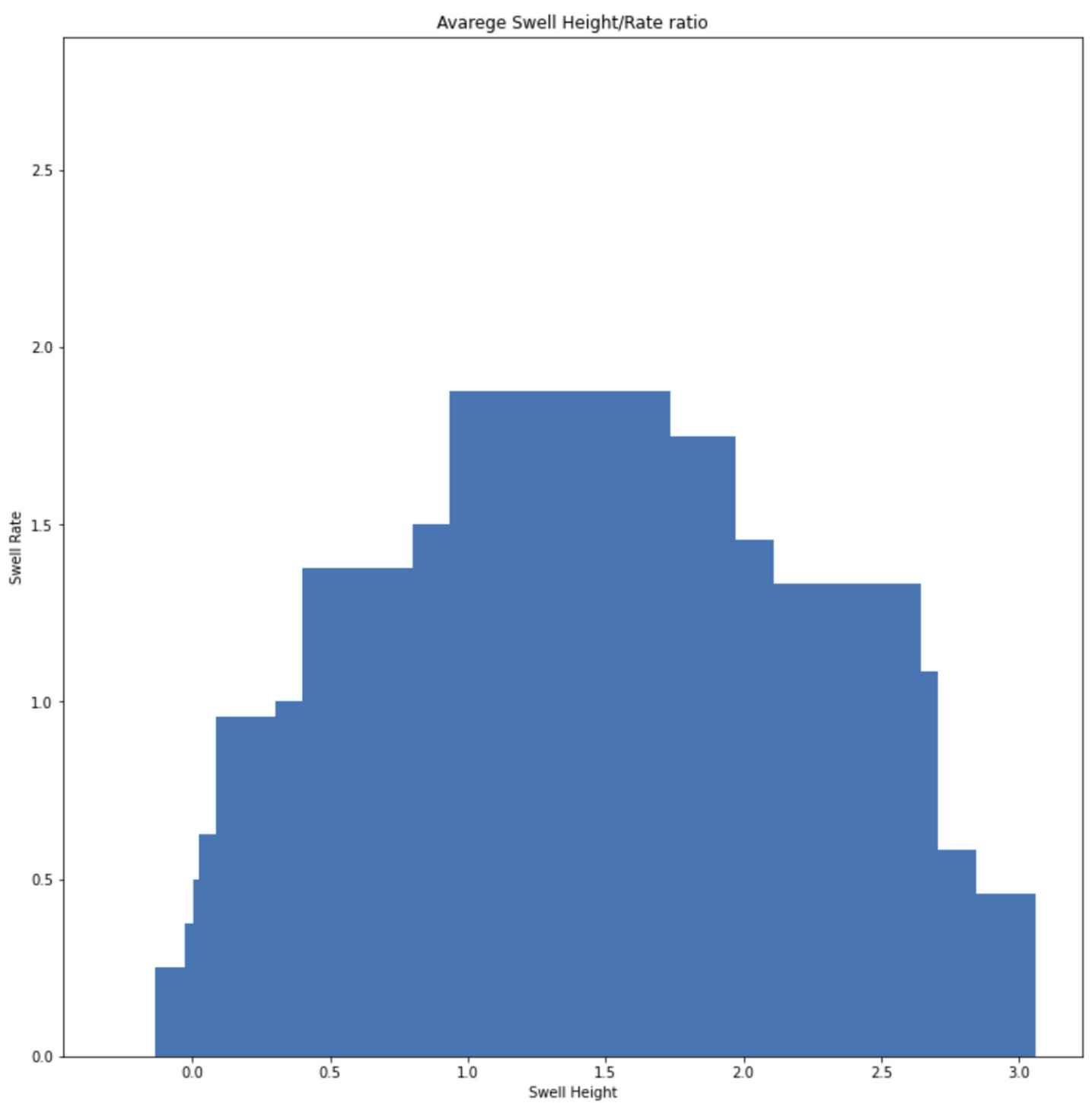




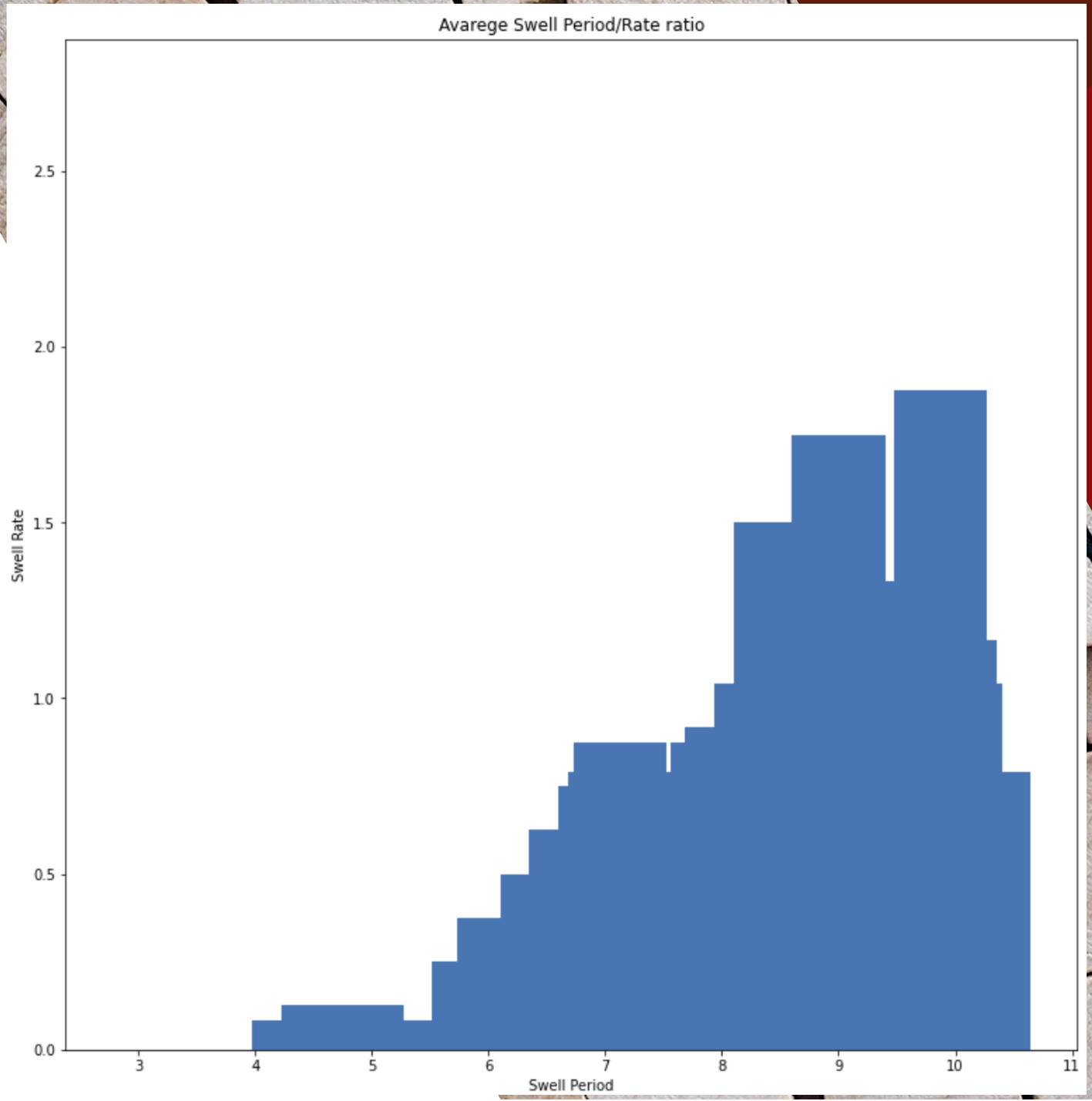
Avarage Height/Period:



Avarage Height/Rate:



Avarage Period/Rate:



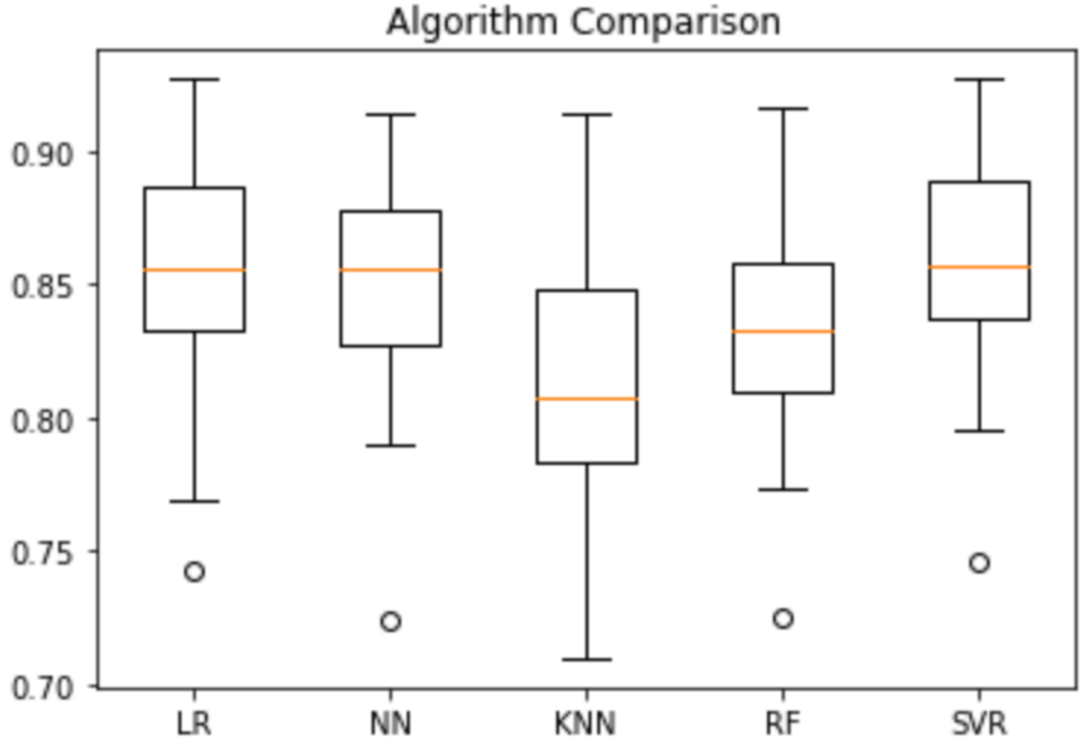
Machine Learning

For machine Learning I have compared several models to chose the one that fits the best for my purpose.

cv = cross validation

cv avg (cv standard deviation)

```
LR: 0.850122 (0.056061)
NN: 0.844237 (0.052109)
KNN: 0.812036 (0.066824)
RF: 0.833202 (0.055524)
SVR: 0.855298 (0.052778)
```



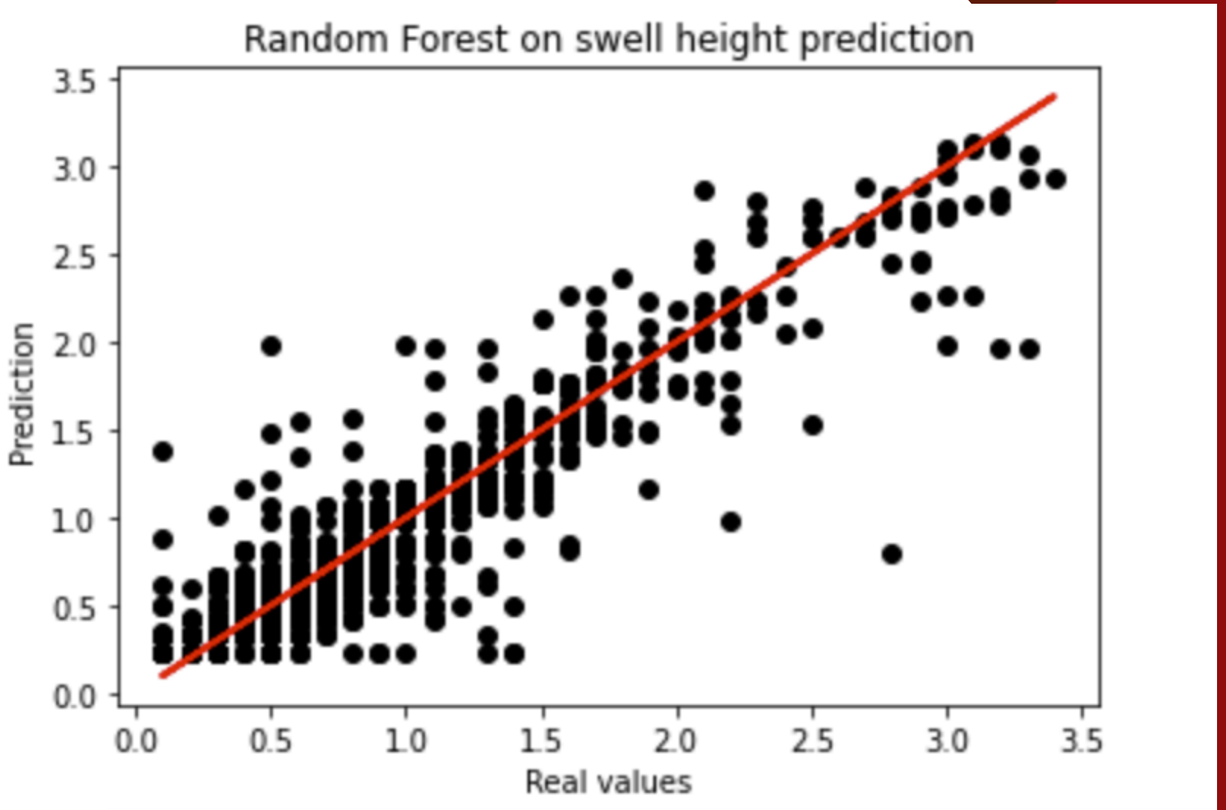
I chose to use Random Forest model because it's one of the best techniques with high performance which is widely used in various industries for its efficiency.

It also ensure that model predictions won't overfit

First Approach: Check all the values from 1/1/20 to 1/6/21 as train and all the values from 2/6/21 to 31/12/21 as test

I have created 2 more column called yesterday and yesterday-diff, although it's not realy "yesterday" but previous row value and it's diff. this way I could trace the difference between each row.

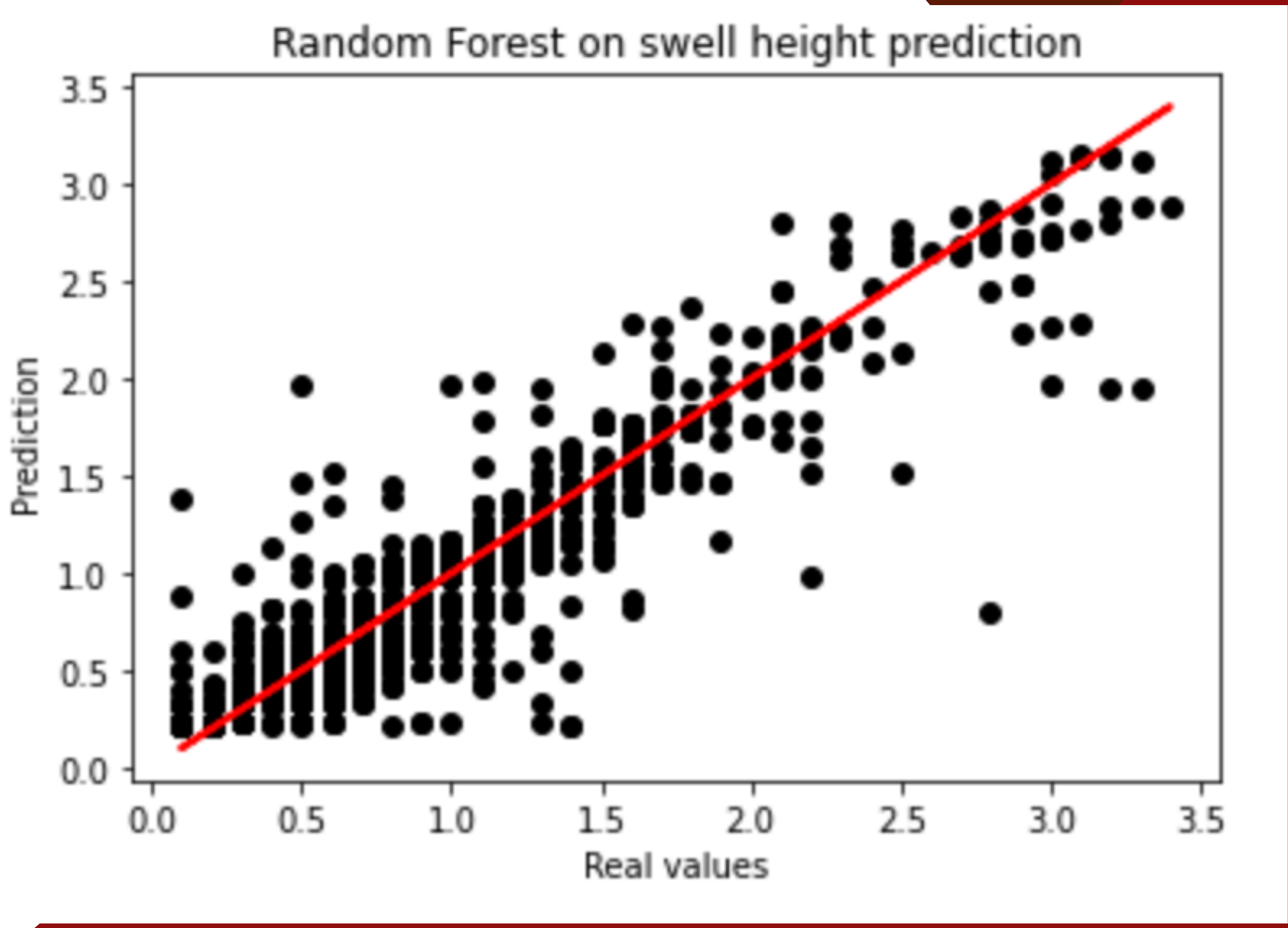
```
explained_variance:  0.8748
mean_squared_log_error:  0.0084
r2:  0.8748
MAE:  0.101
MSE:  0.0313
RMSE:  0.177
```



Second Approach: train and test dates are the same but:

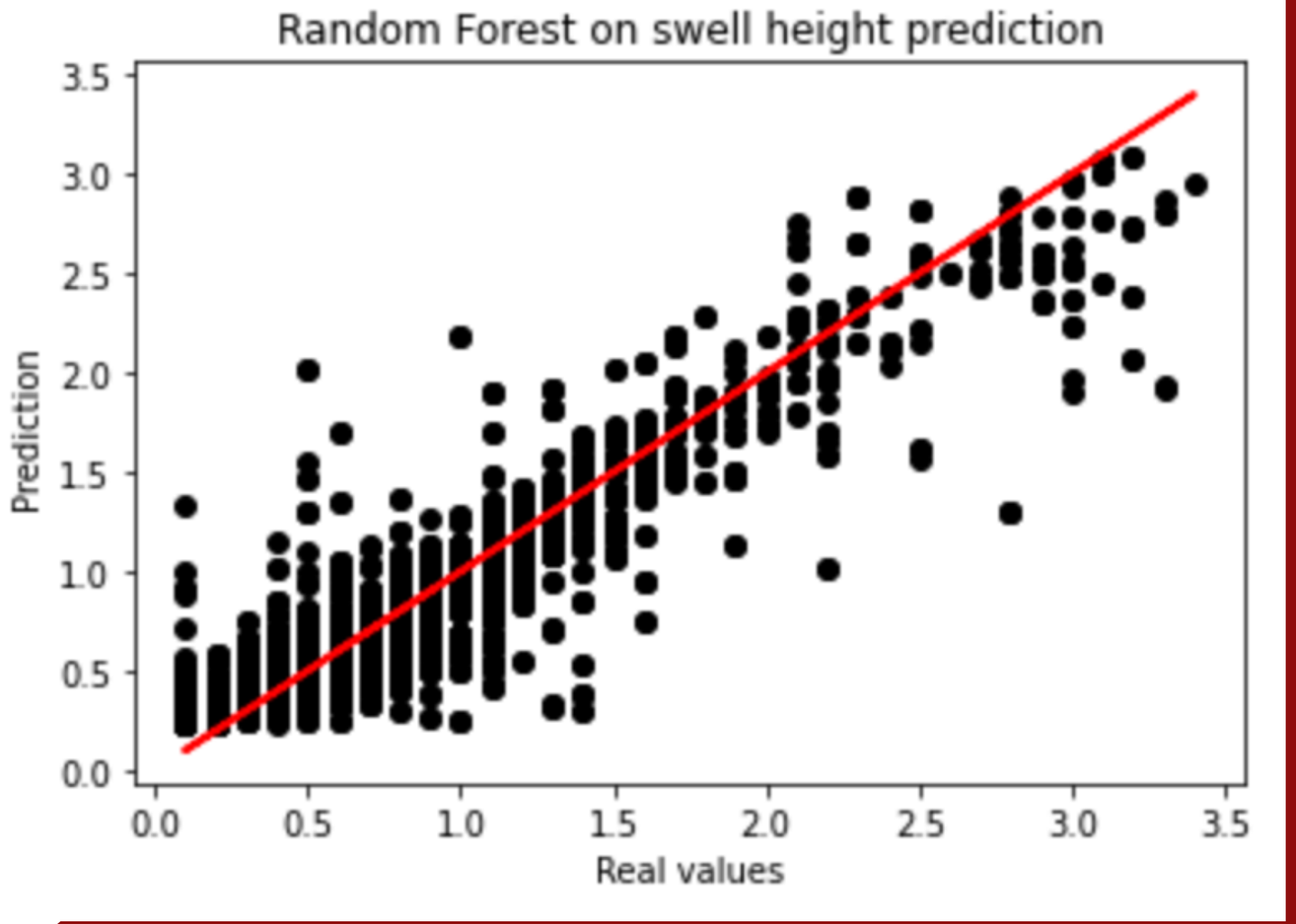
I have created 2 more column called yesterday-1 and yesterday-1 diff, this way I could trace the difference between each row and it's 2 above row.

```
explained_variance:  0.8756
mean_squared_log_error:  0.0084
r2:  0.8756
MAE:  0.1002
MSE:  0.0311
RMSE:  0.1765
```



Third approach: adding the swell period to the model, the rest is the same as second approach

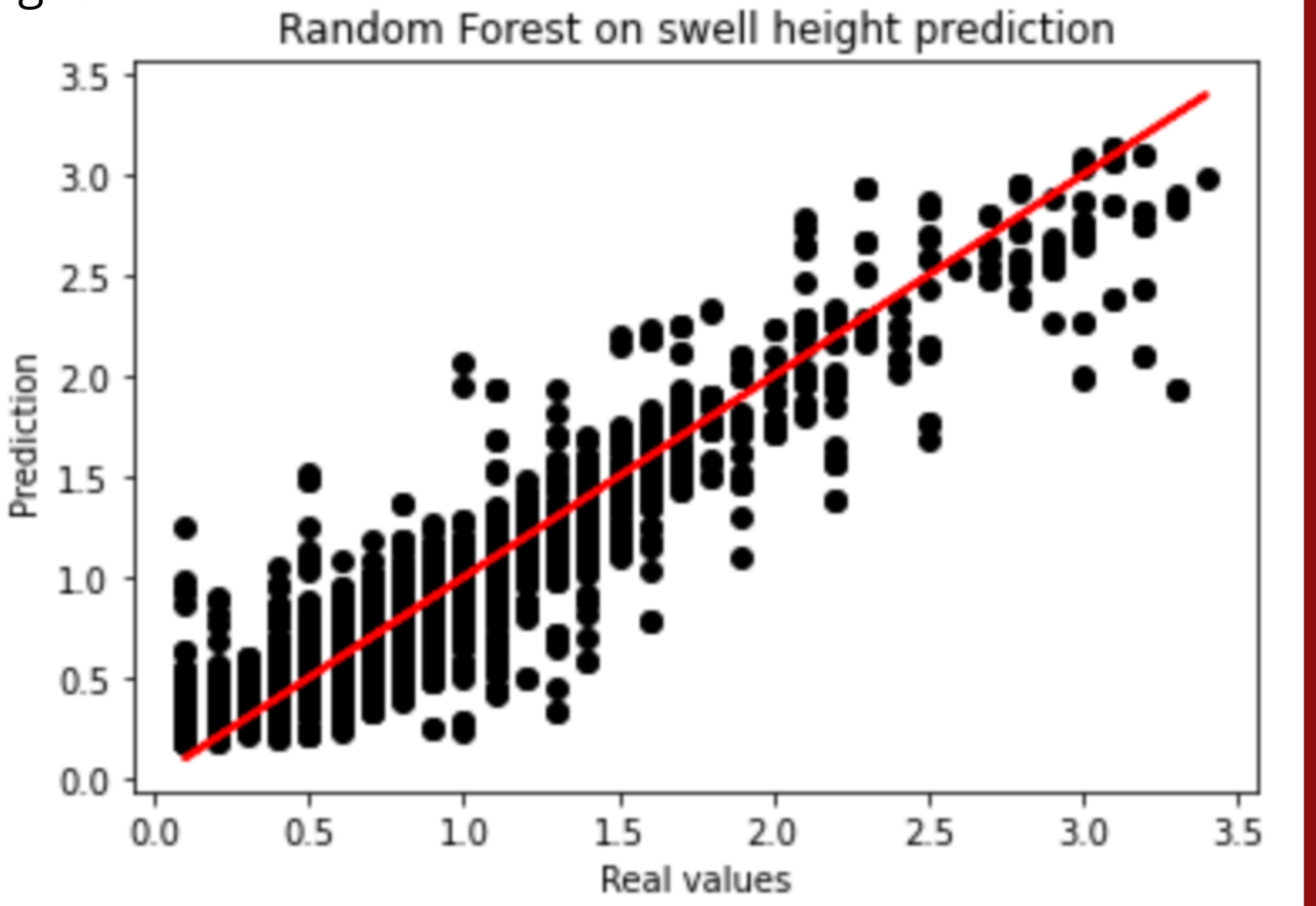
```
explained_variance:  0.8612
mean_squared_log_error:  0.0104
r2:  0.8612
MAE:  0.1192
MSE:  0.0388
RMSE:  0.197
```



Fourth approach: testing the data each in a diff of week.

if the model is trying to predict the height value on Jan 8, it must be fed information about the height on Jan 1.

```
explained_variance:  0.8858
mean_squared_log_error:  0.0088
r2:  0.8858
MAE:  0.1131
MSE:  0.0319
RMSE:  0.1787
```



Feature importance chart: