Machine Learning Final Project

Group 39: 費蓋德 0516251
莫乃杰 0210895
方妮可 0416256

Compare TSMC’s growth and the 5 best Microprocessors companies in the world using data from Yahoo Finance

1. Content

- ➔ Data Acquisition
- ➔ Linear Regression
- ➔ Polynomial Regression
- ➔ Support Vector Machines(SVM)
- ➔ Result and Relevance of our work

What our Project is not about ?

- Stock Market Prediction
  - *Isn't stock market prediction a scam ?*
- Prediction of companies growth or decline
  - *3 years of data is insufficient*

Tip

If someone tells you he can predict stock market, ask him/her to tell you when you will die.

They know they can't

What our Project is about ?

- Confront Yahoo Finance
  - *Having the data is one thing, knowing the data is another one*
- Analyse the ups and downs of the 5 best semiconductors companies Worldwide + TSMC
  - *Some are growing, some are dying (We will figure out Which ones later)*

Tip

Having your data is one thing but nothing compared to knowing you data

Tip

Having your data is one thing but nothing compared to knowing you data

Data Acquisition

- Where do we get the data ?
  - Yahoo Finance (we wanted to analyse the 5 best SC companies)
- How did we get the data ?
  - By scraping yahoo finance
  - It's legal
  - And easy
- What does our data look like ?
  - Data from 12/03/2015 to 12/03/2018
  - 7 rows: Date - Open - High - Low - Close - Adj Close - Volume
  - Relevant Feature :Adj Close - Output Feature: Number of days

Data Acquisition(CODE)

Scraper Program: (funct.py)

funct.py

```python
import os
import numpy as np
import datetime
from datetime import date
import requests


def scraping(url, file='SecFile.txt'):


    link = "https://finance.yahoo.com/quote/REY.MI/history?period1=1431986400&period2=1526680800&interval=1d&filter=history&frequency=1d"

    f = requests.get(url)

    text_file = open("SecFile.txt", "w")
    text_file.write(f.text)
    text_file.close()

    # HTML parsing
    with open(file, 'r+') as myfile:
     s = myfile.read()
     s=s[s.index("Volume")+6:]
     s=s[s.index("\"prices\":[{")+11:]
     s=s[:s.index("}],\"isPending\":")]
     s=s.replace("},{","\n")
     s=s.replace("\"date\":","")
     s=s.replace("\"close\":","")
     s=s.replace("\"open\":","")
     s=s.replace("\"high\":","")
     s=s.replace("\"low\":","")
     s=s.replace("\"volume\":","")
     s=s.replace("\"adjclose\":","")
     s=s.replace(","," ")



    with open(file, "w") as text_file:
      text_file.write(s)

    with open(file,"r+") as f:
      new_f = f.readlines()
      f.seek(0)
      for line in new_f:
          if "\"" not in line:
              f.write(line)
          f.truncate()


    data = np.loadtxt(file)
    data=data.reshape(len(data),7)
    #swap the last two column...
    data[:, 5], data[:, 6] = data[:, 6], data[:, 5].copy()
    #save on file with two decimal rounded digits
    np.savetxt("SecFile.txt", data,fmt='%1.2f')
    #reload file with two decimal rounded digits
    data = np.loadtxt('SecFile.txt')

    # extract first row for next dates generator
    data_c=data[:,0]
    np.savetxt("row1.txt", data_c,fmt='%i')

    os.remove("SecFile.txt")

    return data
```

```python
def Dates_man():

    #----- dates management.... ------
    today = str(date.today().strftime("%Y,%m,%d"))
    #print(today)

    date_1 = datetime.datetime.strptime(today, "%Y,%m,%d")
    date_2=(addYears(date_1, -3))
    code_1=digit_initial_date_for_query(date_1)
    #print(code_1)

    ##### 3 years of data
    numDays=(date_1-date_2).days
    code_2=code_1-numDays*86400

    date_2=date_2.strftime("%Y,%m,%d")

    return today, date_1, code_1, date_2, code_2



def addYears(d, years):
    try:
#Return same day of the current year
            return d.replace(year = d.year + years)
    except ValueError:
#If not same day, it will return other, i.e.  February 29 to March 1 etc.
            return d + (date(d.year + years, 1, 1) - date(d.year, 1, 1))

def days_between(d1, d2):
    d1 = datetime.datetime.strptime(d1, "%b %d, %Y")
    d2 = datetime.datetime.strptime(d2, "%b %d, %Y")
    return abs((d2 - d1).days)


def codeGenerator(digit, init):
    delta=((init-digit)/86400)
    print("delta:  ", delta)
    return(init+delta)

def digit_initial_date_for_query(current_date):

    digit_init=976089600
    d_init = str(date(2000,12,6).strftime("%Y,%m,%d"))
    d_init=datetime.datetime.strptime(d_init, "%Y,%m,%d")
    delta_days=(current_date-d_init).days
    #print("AAAAAA:  ",delta_days)
    date_query_today=digit_init+(86400*delta_days)
    #print("quesy code:   ",date_query_today)
    return (date_query_today)


def dateGenerator(date_1,code_1, code_2):
    delta_days=((code_1-code_2)/86400)
    date_2=date_1 - datetime.timedelta(days=delta_days)
    date_2=date_2.strftime('%b %d, %Y')
    #print("date_2:  ", date_2)
    return (date_2)
```

```python
def saveResult(today, date_2, title):

 # adjust file for better viewing
 with open('Stock_parsed.dat', 'r+') as myfile:
  s = myfile.read()
  s=s.replace(".0\n","\n")
  #s=s.replace(",   ",", ")

 # create new directory folder
 dir="scraping/"+title+"/"+str(today)+"__"+str(date_2)+"/"
 dir=dir.replace(",",".")
 if not os.path.exists(dir):
    os.makedirs(dir)

 init_row="Date            Open    High    Low     Close   AdjCl    Volume\n"
 tmp_row="Date,Open,High,Low,Close,Adj Close,Volume\n"

 with open(dir+title+".dat", "w") as text_file:
    text_file.write(init_row)
    text_file.write(s)

 os.remove("Stock_parsed.dat")

 with open('CSV.txt', 'r+') as myfile:
  s = myfile.read()

 with open(dir+title+".csv", "w") as text_file:
    text_file.write(tmp_row)
    text_file.write(s)

 os.remove("CSV.txt")
```

```python
def addDigit(s):

    s=s.replace(".0",".00")
    s=s.replace(".1",".10")
    s=s.replace(".2",".20")
    s=s.replace(".3",".30")
    s=s.replace(".4",".40")
    s=s.replace(".5",".50")
    s=s.replace(".6",".60")
    s=s.replace(".7",".70")
    s=s.replace(".8",".80")
    s=s.replace(".9",".90")
    return s



def urlGenerator(code_1, code_2, titles):

 url=[]
 for i in range(len(titles)):
   url.append("https://finance.yahoo.com/quote/"+titles[i]+"/history?period1="+str(code_2)+"&period2="+str(code_1)+"&interval=1d&filter=history&frequency=1d")
   #print(url[i])
 return url


def code2Dates(date_1,code_1, data):
 #--------- dates -----------

 new_dates=[]

 # convert all column data....
 with open("row1.txt","r+") as f:
     new_f = f.readlines()
     f.seek(0)
     for line in new_f:
        new_dates.append(dateGenerator(date_1,code_1,int(line)))

 data=np.delete(data,0,1)
 all = data.astype(np.str)

 all=np.insert(all, 0, new_dates, axis=1)

 for i in range(all.shape[0]):
    for j in range(all.shape[1]-1):
        if (all[i][j][len(all[i][j])-2]=="."):
           all[i][j]=np.core.defchararray.add(all[i][j],"0")

 all=np.core.defchararray.replace(all,",  ",", ")
 #tmp=all[:,[0,4]]
 tmp=all
 tmp=np.core.defchararray.replace(tmp,", ","," ")
 tmp=np.core.defchararray.replace(tmp,"   ",",")
 #tmp=np.core.defchararray.replace(tmp,"   ",",")
 np.savetxt("csv.txt", tmp,fmt='%s', delimiter=",")
 np.savetxt("Stock_parsed.dat", all,fmt='%s', delimiter="   ")
 # remove tmp files...
 os.remove("row1.txt")
```

Scraper.py (Uses the functions from funct.py to get the Data according to the keywords and save it in a folder called scraping.)

```python
from funct import*


# the code for the companies we want
titles=["AMD", "AVGO", "PLAB", "XPER", "MU", "TSM"]

for i in range(len(titles)):

  print("scraping ", titles[i], " stock quotes in action...", end='', flush=True)

  # dates management....
  today, date_1, code_1, date_2, code_2 = Dates_man()

  # URL generation for querying web....
  url=urlGenerator(code_1, code_2, titles)

  # web scraping & HTML parsing
  data=scraping(url[i])


  # convert dates from original codes
  code2Dates(date_1, code_1, data)



  # adjust file for better viewing & save final data
  saveResult(today, date_2, titles[i])
  print("done!")

print("\nScarping process fully completed!")
```

Data Preprocessing

For each companies, we had something
Like this :

	Date	Open	High	Low	Close	Adj Close	Volume
0	Dec 03 2018	22.48	23.75	22.37	23.71	23.71	139255800.0
1	Nov 30 2018	21.30	21.36	20.52	21.30	21.30	82370700.0
2	Nov 29 2018	21.19	21.61	20.73	21.43	21.43	79853700.0
3	Nov 28 2018	21.82	21.88	20.18	21.34	21.34	134425300.0
4	Nov 27 2018	19.77	21.45	19.73	21.05	21.05	119230100.0
5	Nov 26 2018	19.96	20.19	19.11	20.08	20.08	82965000.0
6	Nov 23 2018	18.61	19.83	18.56	19.38	19.38	54611300.0
7	Nov 21 2018	20.05	20.31	18.50	18.73	18.73	81585600.0
8	Nov 20 2018	17.40	19.58	17.18	19.21	19.21	109869400.0
9	Nov 19 2018	20.40	20.59	19.09	19.11	19.11	93578200.0
10	Nov 16 2018	19.87	20.97	19.72	20.66	20.66	112376600.0

[755 rows x 7 columns]

For 6 companies we had no less
Than 755 * 6 rows of data = 4530.

Then we transform it into this:

	Day_count	Adj Close
0	755	23.71
1	754	21.30
2	753	21.43
3	752	21.34
4	751	21.05
5	750	20.08
6	749	19.38
7	748	18.73
8	747	19.21
9	746	19.11
10	745	20.66

Where 12/03/2015 is day 1 and
12/03/2018 is day 755

(keep in mind that the stock
Market doesn't open on weekends)

What are the six companies in question ?

1. Advanced Micro Devices (AMD)
2. Broadcom (AVGO)
3. Micron Technology (MU)
4. Photronics (PLAB)
5. TSMC
6. Xperi Corporation (XPER)

```python
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.svm import SVR

# read my data
df1 = pd.read_csv("scraping/AMD/2018.12.04__2015.12.04/AMD.csv")
df2 = pd.read_csv("scraping/AVGO/2018.12.04__2015.12.04/AVGO.csv")
df3 = pd.read_csv("scraping/MU/2018.12.04__2015.12.04/MU.csv")
df4 = pd.read_csv("scraping/PLAB/2018.12.04__2015.12.04/PLAB.csv")
df5 = pd.read_csv("scraping/TSM/2018.12.04__2015.12.04/TSM.csv")
df6 = pd.read_csv("scraping/XPER/2018.12.04__2015.12.04/XPER.csv")
print(df1)
# transform the datetime (optional)
df1['Date'] = pd.to_datetime(df1['Date'])
df2['Date'] = pd.to_datetime(df2['Date'])
df3['Date'] = pd.to_datetime(df3['Date'])
df4['Date'] = pd.to_datetime(df4['Date'])
df5['Date'] = pd.to_datetime(df5['Date'])
df6['Date'] = pd.to_datetime(df6['Date'])

# drop the unused features
df1.drop(['Date', 'High', 'Low', 'Close', 'Volume', 'Open'], axis=1, inplace=True)
df2.drop(['Date', 'High', 'Low', 'Close', 'Volume', 'Open'], axis=1, inplace=True)
df3.drop(['Date', 'High', 'Low', 'Close', 'Volume', 'Open'], axis=1, inplace=True)
df4.drop(['Date', 'High', 'Low', 'Close', 'Volume', 'Open'], axis=1, inplace=True)
df5.drop(['Date', 'High', 'Low', 'Close', 'Volume', 'Open'], axis=1, inplace=True)
df6.drop(['Date', 'High', 'Low', 'Close', 'Volume', 'Open'], axis=1, inplace=True)


# add new col with  day count (the earliest day being day 1 (2015-12-03))
# and the last (2018-12-03)
df1.insert(0, 'Day_count', range(len(df1), 0, -1))
df2.insert(0, 'Day_count', range(len(df2), 0, -1))
df3.insert(0, 'Day_count', range(len(df3), 0, -1))
df4.insert(0, 'Day_count', range(len(df4), 0, -1))
df5.insert(0, 'Day_count', range(len(df5), 0, -1))
df6.insert(0, 'Day_count', range(len(df6), 0, -1))
```

Data Preprocessing(code)

2. Linear Regression

Is used to predict a quantitative response Y from the predictor variable X.

Is made with an assumption that there's a linear relationship between X and Y.

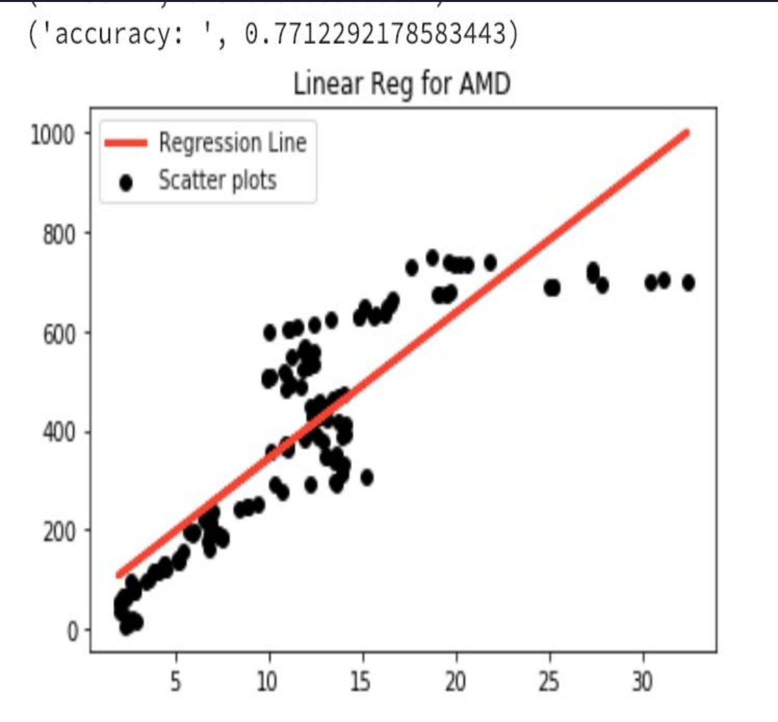
So in the graph below you can that he y the days and x is the stock price when the market is closed.

Code (Sklearn library)
Same code for each data just different dataset

```
# Linear regression
train_X, test_X, train_y, test_y = train_test_split(X, Y, test_size=0.20)
reg = LinearRegression().fit(train_X, train_y)
pred_y = reg.predict(test_X)

# accuracy
acc = reg.score(test_X, test_y)
print("accuracy: ", acc)

# Plot the graph
plt.title('Linear Reg for AMD')
plt.scatter(test_X, test_y,  color='black', label = 'Scatter plots')
plt.plot(test_X, pred_y, color='red', linewidth=3, label='Regression Line')
plt.legend()
plt.show()
```

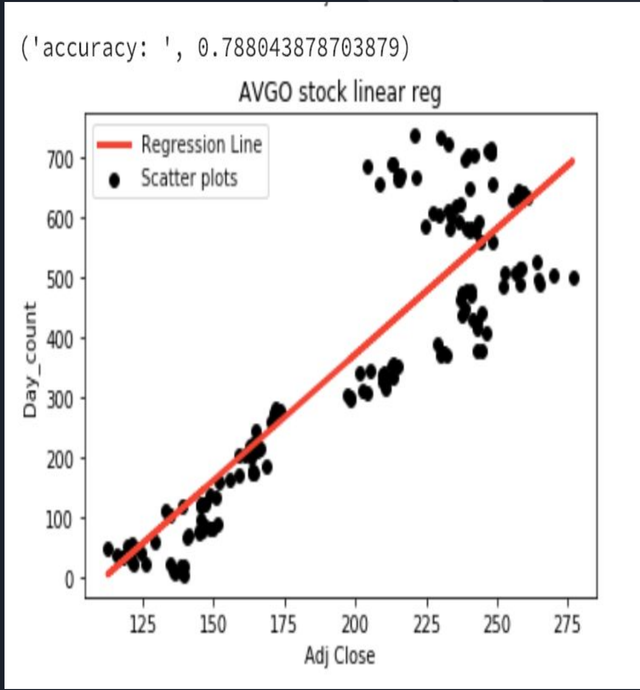
1. **Advanced Micro Devices (AMD)**

AMD stock prices has been increasing



2. **Broadcom (AVGO)**

Stock price has been increasing steadily until last 100 days it has been unsteady

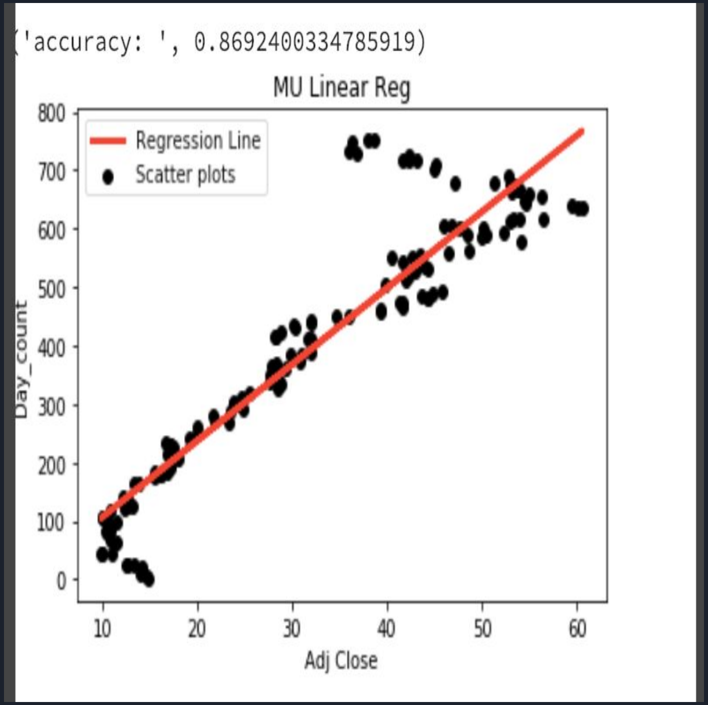


Tip

Build simple models

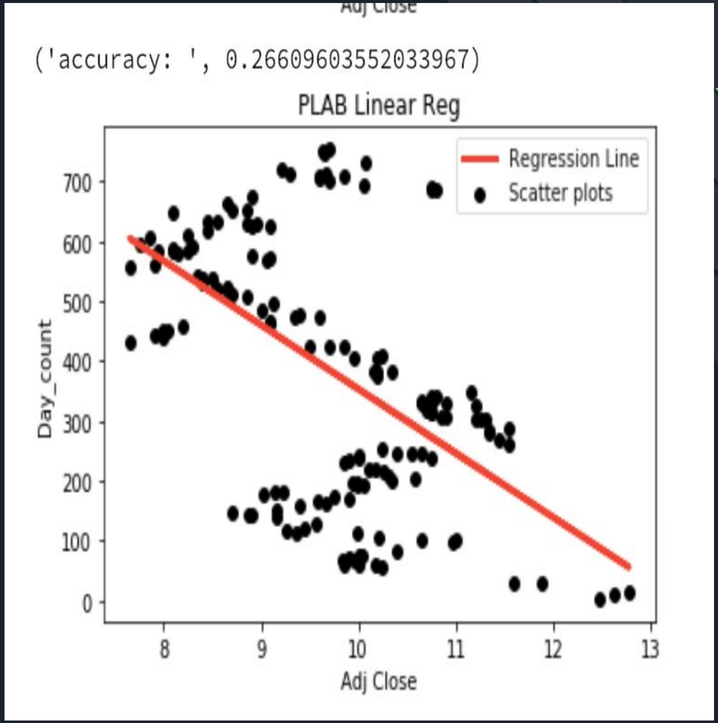
3. Micro Technology (MU)

Have the best result among the 6 companies. Linear reg is best for this graph because is around 90% accuracy



4. Photronics (PLAB)

The results are worst among the 6 companies, over the past 600 days the stock prices has been negative but you see its stock prices are increasing in the later days

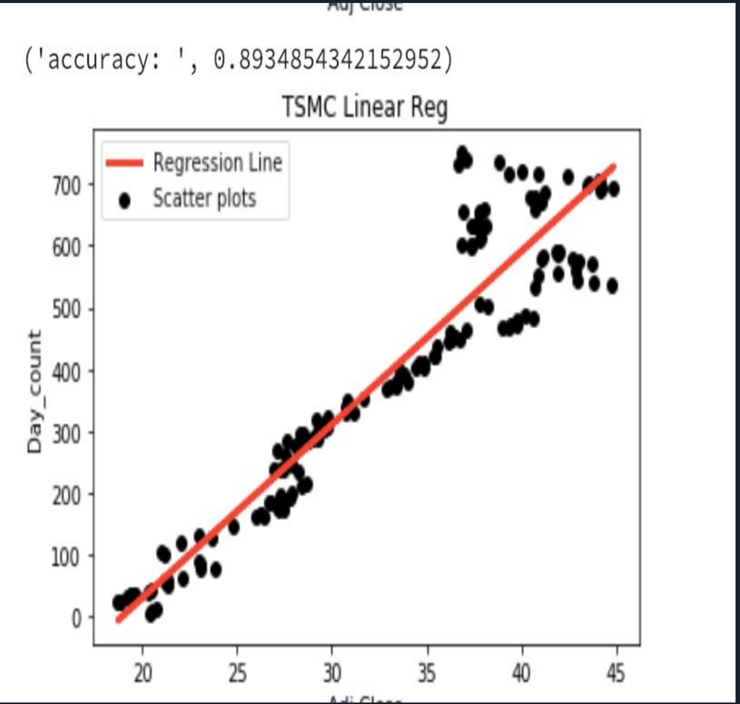


Tip

The key step in getting good model is explanatory data!!

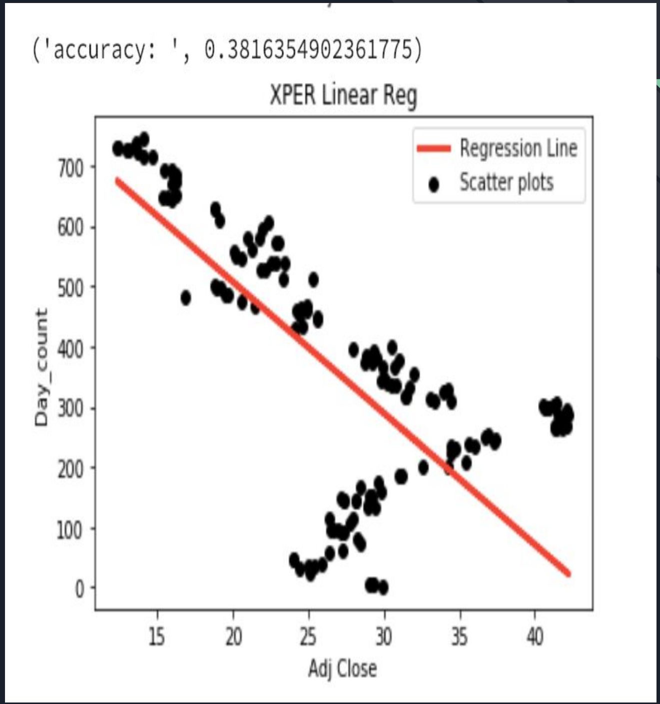
5. TSMC

TSMC, the fastest growing SC company (among the 6) in the world during the last 3 years.



6. XPER

There is a negative relationship between the days (Y) and end of stock price (X)

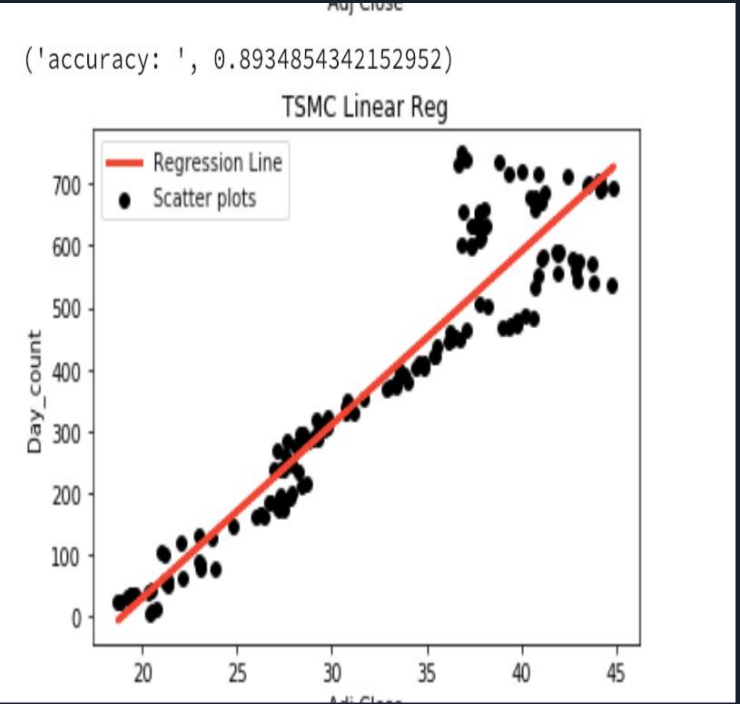


Tip

Be sure to graph the relevant variables

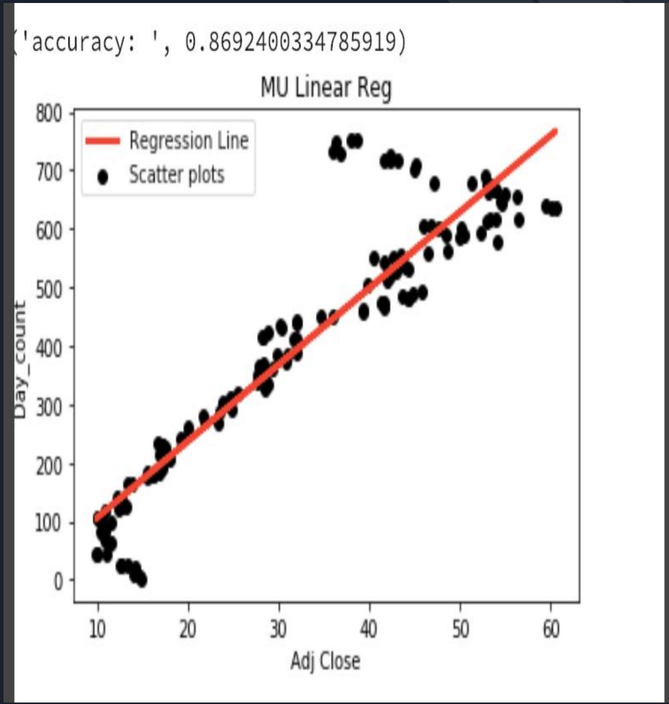
5. TSMC

TSMC, the fastest growing SC company



3. Micro Technology (MU)

MU, the second fastest growing company



Tip

Be sure to graph the relevant variables

3. Polynomial Regression

In statistics, **polynomial regression** is a form of regression analysis in which the relationship between the independent variable *x* and the dependent variable *y* is modelled as an ***nth*** degree polynomial in *x*.

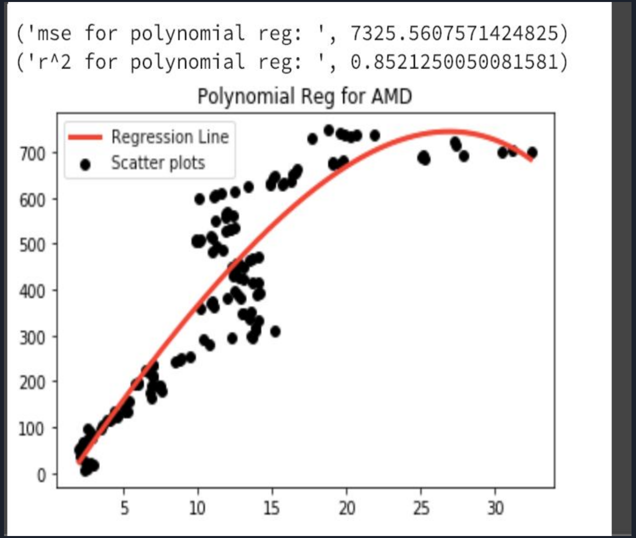
Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is considered to be a special case of multiple linear regression.

Code (Sklearn library)
Same code for each data just different dataset
The degree of the polynomial will change according to the data we have

```
# Polynomial reg
model = Pipeline([('poly', PolynomialFeatures(degree=4)),
                  ('linear', LinearRegression(fit_intercept=False))])



model = model.fit(train_X, train_y)
y_pred = model.predict(test_X)
r = r2_score(test_y, y_pred)
mse = mean_squared_error(test_y, y_pred)
print("mse for polynomial reg: ", mse)
print("r^2 for polynomial reg: ", r)


# PLOT THE GRAPH
X_grid = np.arange(min(test_X), max(test_X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.title('Polynomial Reg for AMD')
plt.scatter(test_X, test_y,  color='black', label = 'Scatter plots')
# test_X = np.linspace(0,1, 151)

plt.plot(X_grid, model.predict(X_grid), color='red', linewidth=3, label='Regression Line')
plt.legend()
plt.show()
```

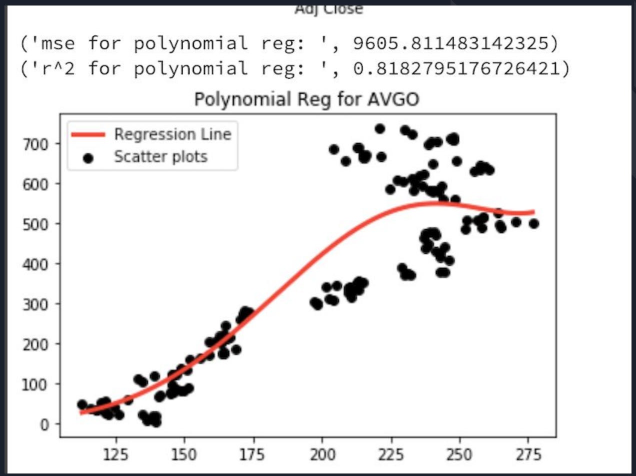
1. Advanced Micro Devices (AMD)

This basically tells us that we can predict with as much as 85% the behavior of AMD stock.



2. Broadcom (AVGO)

AVGO is probably the most expensive SC company in the world. This can clearly be seen from the graph. But for the last 200 days, they are not going that good

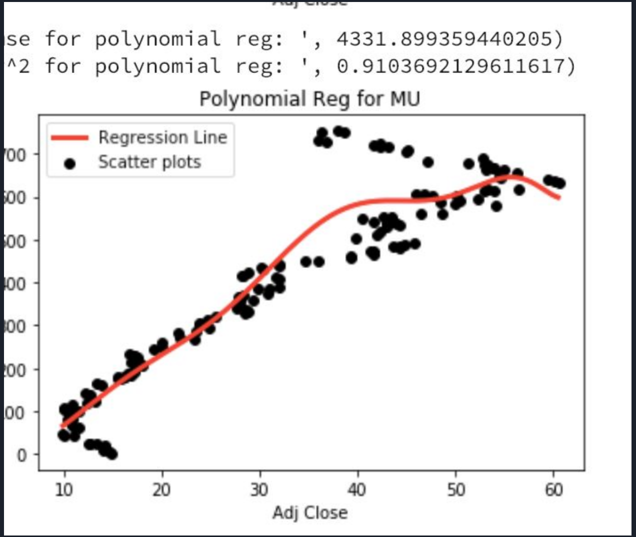


Tip

Sklearn provides a very good and handy polynomial regression library, try it out (:

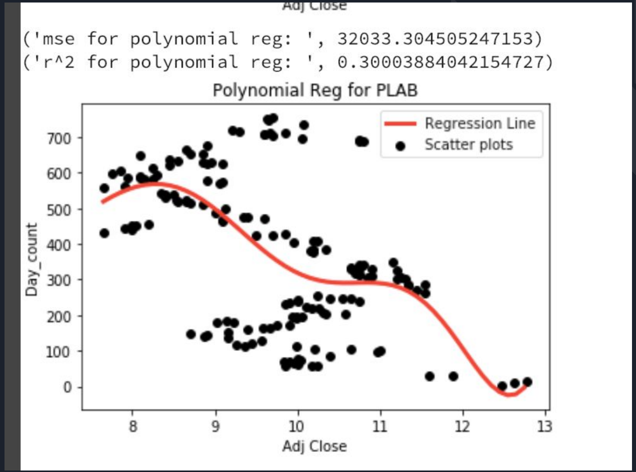
3. Micro Technology (MU)

MU’s growth was one of the most significant until recently. We can say their last 100 days on the market are bad



4. Photronics (PLAB)

PLAB has the prize for the most messy distribution.First of all, their price are the worst and for the last 3 years, they registered more setbacks than growth (How can they be among the 5 best in the world ???)

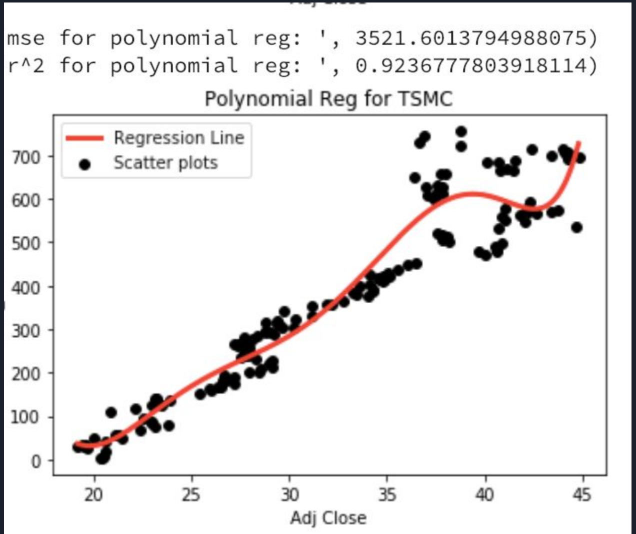


Tip

Numpy polyfit is also great too.

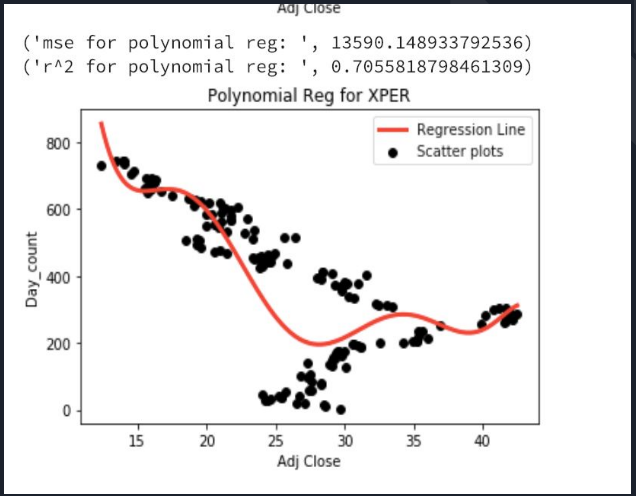
5. TSMC

TSMC, the fastest growing SC company (among the 6) in the world during the last 3 years. But it was not in the list of the 5 best (???). Like most companies, they suffer from the recent economic crisis (trade war) but overall they are doing good. Their growth can be predicted at more than 90% accuracy



6. XPER

XPER follow almost the same pattern as PLAB. This is a company that should probably be rethought. They lost too much in the recent years. Their mainly negative distribution can be predicted accurately.
Ps: It is still considered as one of the best in the world (???)



Tip

Numpy polyfit is also great too.

Support Vector Machines (SVM)

```
#SVM 1
array = np.array(df1)
X = array[:,1].reshape(-1, 1)
Y = array[:,0]
train_X, test_X, train_y, test_y = train_test_split(X, Y, test_size=0.20)

clf = SVR(kernel='rbf', C=10, degree=3, gamma='scale')
pred_y = clf.fit(train_X, train_y).predict(test_X)
r = r2_score(test_y, pred_y)
mse = mean_squared_error(test_y, pred_y)
print("r^2 for SVR: ", r)

# Plot the graph
X_grid = np.arange(min(test_X), max(test_X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.title('SVM for AMD')
plt.scatter(test_X, test_y,  color='black', label = 'Scatter plots')
plt.plot(X_grid, model.predict(X_grid), color='red', linewidth=3, label='Regression Line')
plt.legend()
plt.show()
```

In machine learning SVMs also called support vector networks, are supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis.

1. Advanced Micro Devices (AMD)

If we predict Y from X based solely from looking at the scatterplot and regression line, what Y value is associated with an X = 400

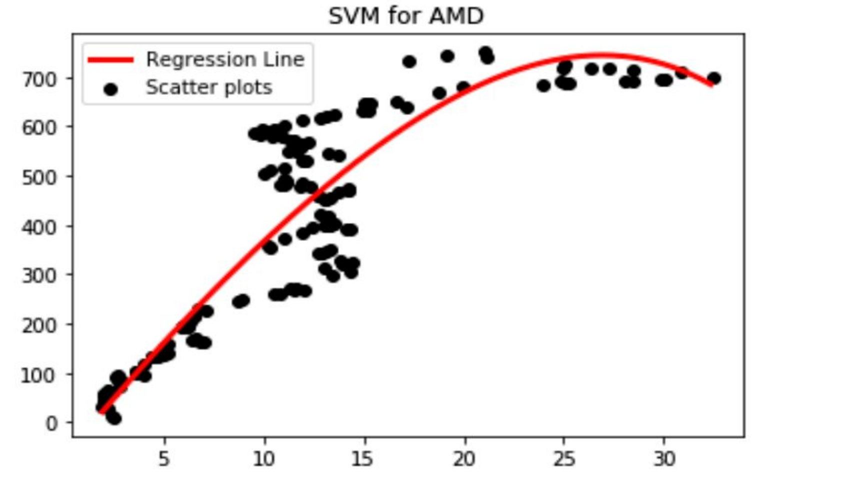
The two point observed from the graph are (400,10) and (400,15)

2. Broadcom (AVGO)

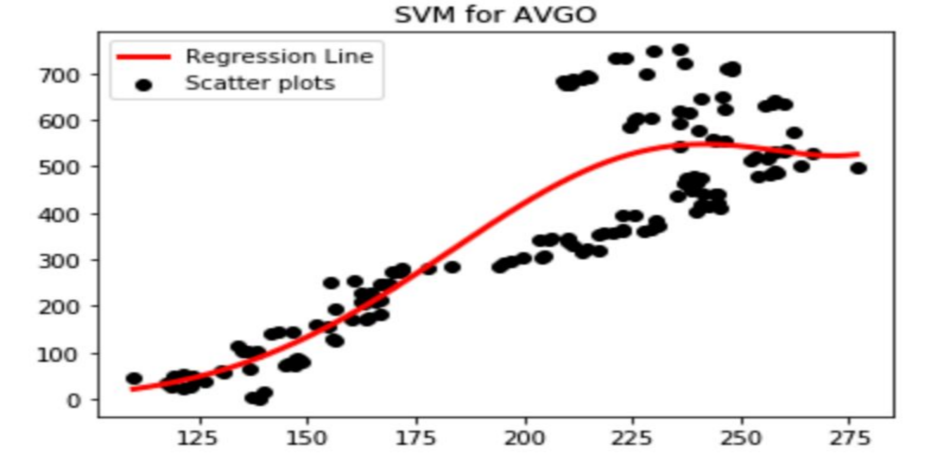
Here the plot shows non- linear relationship between parameters in non linear SVR, the kernel functions transform the linear separation.

The plot basically tells us how the data points fits to the regression line. That is how AVGO increase is predicted / train with the respect to the SVR

('r^2 for SVR: ', 0.8239343301955481)

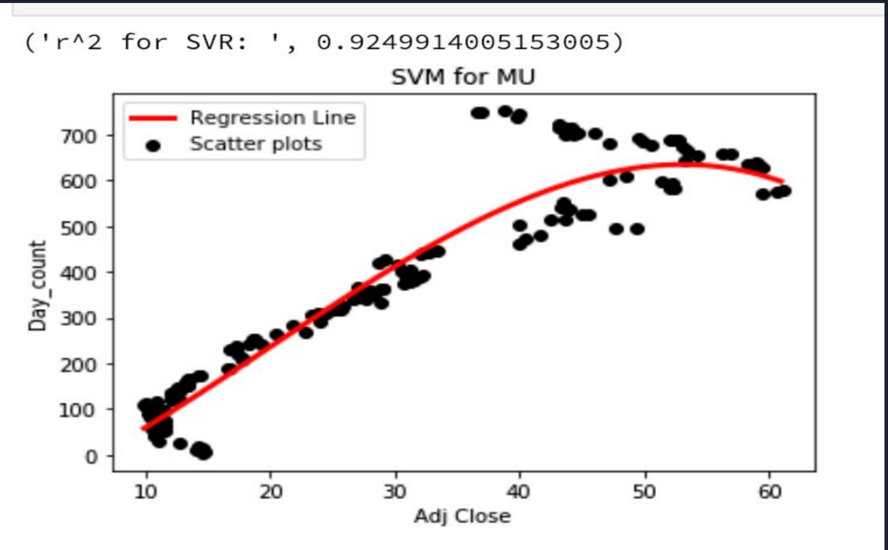


('r^2 for SVR: ', 0.7012302103273238)



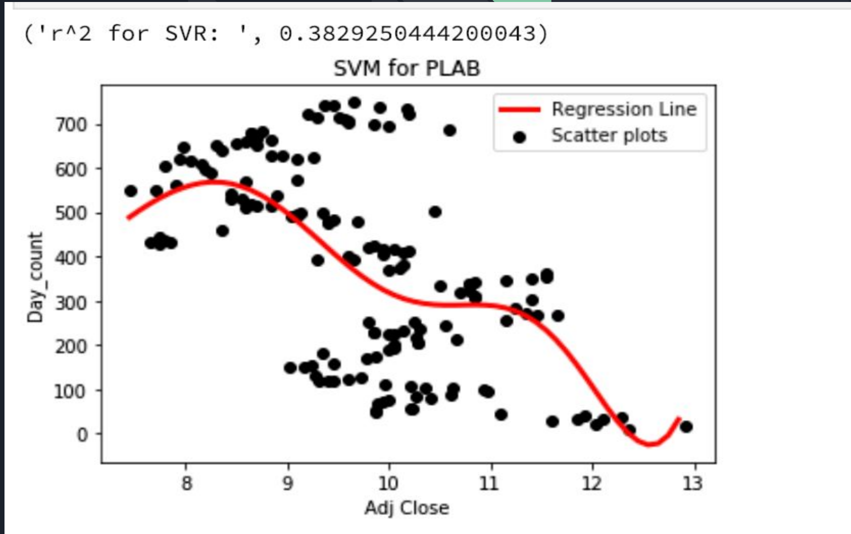
3. Micro Technology (MU)

Here we noticed that all the point are very close to regression line indicating a high positive correlation.



4. Photronics (PLAB)

Here the observations are negatively correlated and only few points are close to the regression line, this shows a bad result over the years

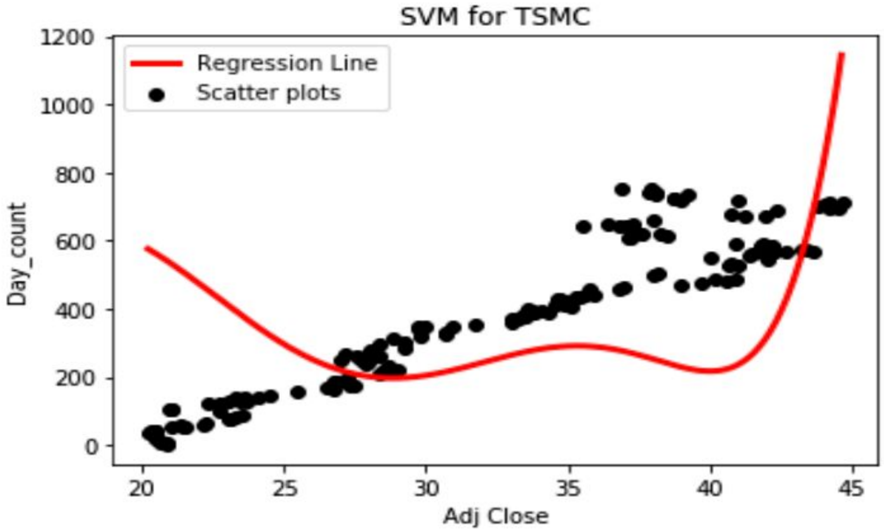


5. Taiwan Semiconductor Manufacturing Company (TSMC)

If we predict Y from X based solely from looking at the scatterplot and regression line, what Y value is associated with an X value.

Here taking a look at the relationship between Y and X , we noticed that there is a positive move over the years.

```
('r^2 for SVR: ', 0.9342337531407596)
```

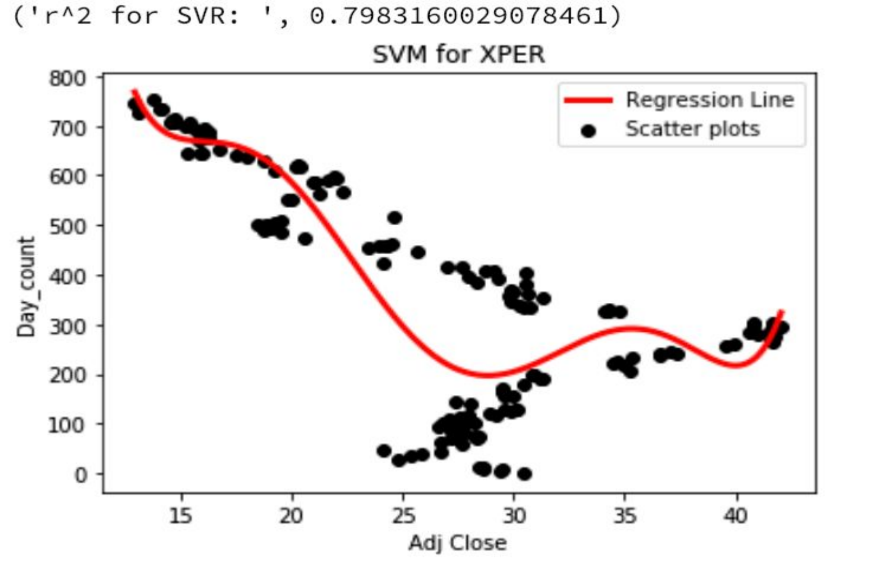


If you take a look you will easily notice that all the points are relatively closer to one and other, and not distance from the regression line…. This manifest that there is a positive move over the past years and up to date.

6. Xperi Corporation (XPER)

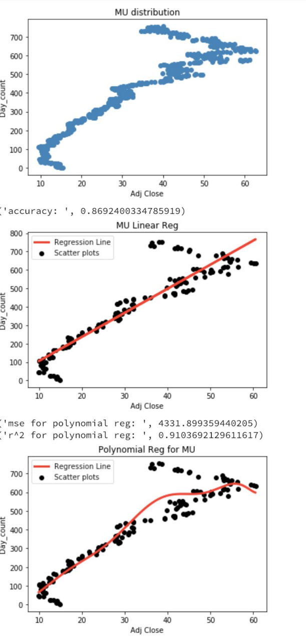
The relationship of X and Y through the regression line is negatively correlated.

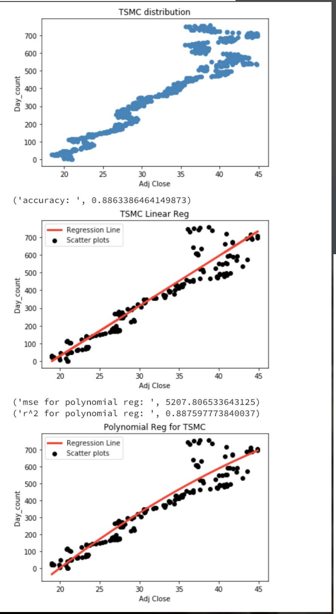
We expect negative relationship between X and Y due to the direction the regression line takes and the scatter plots.



Overall result

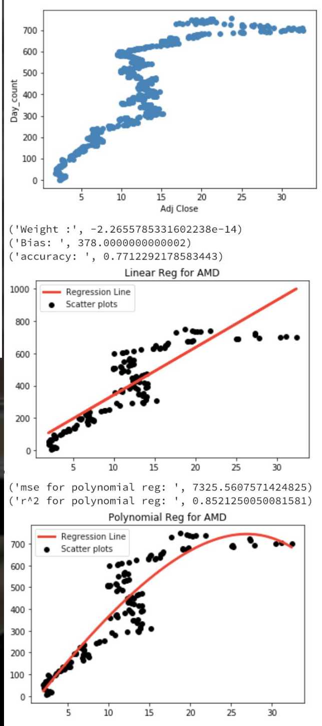
MU

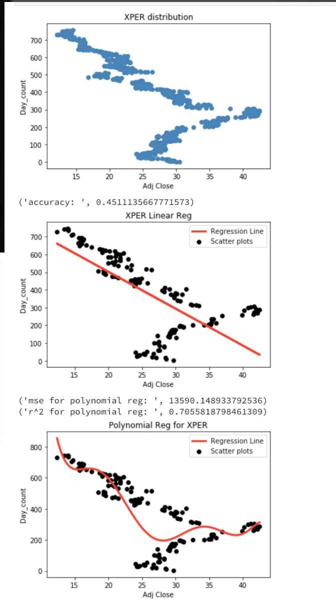




TSMC

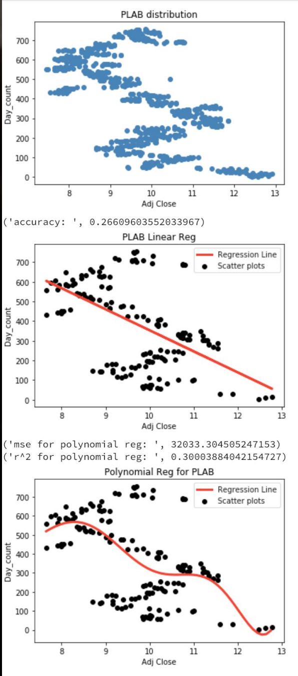
AMD

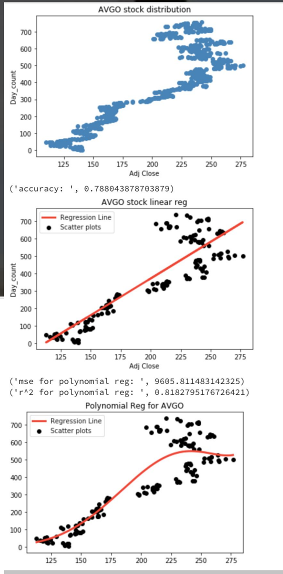




XPER

PLAB





AVGO

Relevance of our work

Always keep in mind that the media will not always tell the truth.

Fortunately you can manipulate data by yourself

Your growth is highly related to how you started. AVGO is better but has not been better the last 3 years

My financial engineer friend was wrong. The volume of stock transactions during a period of time has little or no impact on the price of that stock.

Thank you...!

Group Contribution Table

Work / Members	Data acquisition and preprocessing	Linear Regression	Polynomial Regression	Support Vector Machine Regression
方妮可 0416256		👍		
Omodou Njie 0210895				👍
費蓋德 0516251	👍		👍	