**Term Group Project: Python for Marketing and Sales.**

**Predicting opening value of stock for the next day.**

|  |  |  |  |
| --- | --- | --- | --- |
| Program & Batch: | **PGDM, 2017-19** | | |
| Term: | **V** | | |
| Course Name: | **Python for Marketing and Sales** | | |
| Name of the faculty: | **Prof. Tushar Jaruhar** | | |
| Topic/Title: | **Project Proposal** | | |
| Original or Revised Write-up: | **Original** | | |
| Group Number: | **3** | | |
| Contact No. and email of Group Coordinator: | **+91 9003961463 :** [**pgp17kartiknagrajan@imt.ac.in**](mailto:pgp17kartiknagrajan@imt.ac.in) | | |
| Group Members: | **SL.** | **Roll No.** | **Name** |
| **1** | 170101003 | Bopin Valsan |
| **2** | 170103097 | Harshvardhan Singh |
| **3** | 170103105 | Aayush Vats |
| **4** | 170101036 | Gaurav Bhatt |
| **5** | 170101071 | Jitender Joshi |
|  | **6** | 170103097 | Kartik ganesh |



Contents

[**Business Case** 2](#_Toc522742160)

[Why this business case 3](#_Toc522742161)

[Issues addressed : 3](#_Toc522742162)

[**Model** 4](#_Toc522742163)

[THE SVM Model : 4](#_Toc522742164)

[**Algorithm** 6](#_Toc522742165)

[**Model assumptions** 7](#_Toc522742166)

[**Documentation of the code** 7](#_Toc522742167)

[EXPLAINING THE SVR Function : 8](#_Toc522742168)

[Libraries Used : 10](#_Toc522742169)

[**GRAPHICAL OUTPUTS :** 10](#_Toc522742170)

[RBF MODEL GRAPH OUTPUT 10](#_Toc522742171)

[LINEAR MODEL OUPUT : 11](#_Toc522742172)

[POLYNOMIAL MODEL OUTPUT : 11](#_Toc522742173)

[**OUTPUT FROM THE SCRIPT :** 12](#_Toc522742174)

[**OVERALL SUPPORT VECTRO REGRESSION RESULT :** 13](#_Toc522742175)

[Conclusion 13](#_Toc522742176)

[Recommendations and future steps 13](#_Toc522742177)

[Exceptions 14](#_Toc522742178)

[Version control 14](#_Toc522742179)

[**APPENDIX** 15](#_Toc522742180)

# **Business Case**

Charles, a brilliant coder, has been in his job for ten months now. He works for Goldman Sachs as a developer and is an expert in writing python codes for reconciliations of reports from different sources. His deep interest has always laid in python programming which has won him many competitions in his under-graduation. He has been recognized by many for his ability to come up with solutions to complex problems using python.

He has been friends with Adarsh V T, a stock market enthusiast. Adarsh V T has been in the stock market for three years now. Starting with a nominal amount, Adarsh V T made huge profits on the stock exchanges. He left his job to focus entirely on stock markets. Charles and Adarsh V T often meet and discuss stock exchanges and the complicated process of knowing where to invest and from where to divest. These discussions have led to Charles taking a keen interest in stock market exchange processes and how the prices of stocks rise and fall.

It was Eid, a holiday for both Adarsh V T and Charles. Both decided to have lunch together and celebrate Eid. While having lunch, they started conversing on stock price prediction and the challenges one face in stock markets. Adarsh V T emphasizing the complexity of the stock price prediction process said,” Stock price prediction is one of the most widely studied and challenging problems, and it has attracted researchers from many fields including economics, history, finance, mathematics, and computer science.” Adarsh V T stated that the challenge of stock forecasting is so appealing because improvement of just a few percentage points can increase profit by millions of dollars. This made Charles curious about the stock price prediction, and he started thinking about how can technology be leveraged to predict stock prices.

Charles, after coming back to his home started to read research papers about stock price prediction. After thorough readings and analyzing the topic, he realized that the volatile nature of the stock market makes it challenging to apply simple time series or regression techniques. Recently, researchers have started to focus on machine learning and big data for forecasting stock prices on various exchanges. One of the most popular machine learning techniques used for stock price prediction is **“Support Vector Machines (SVM).”** Charles planned on using SVM techniques at time “t” to predict whether a given stock’s price is higher or lower on the day “t+m”.

## Why this business case

Charles after pondering over the topic for a long timeobserved that every day billions of dollars were traded on the exchange, and behind each dollar spend an investor was hoping to earn profits. Despite its prevalence, Stock Market prediction remained a secretive and empirical art and only a few people, if any, knew about the strategies to be successful in such a volatile place. This made many people vulnerable to the negative impacts of stock markets.

Charles whenever he thought of the negative impacts of stock markets, always imagined the 2008 financial crisis. He started conceptualizing whether the 2008 financial crisis could have been avoided. He knew for a fact that if there is an algorithm which could improve the prediction of stock prices by only a few points, it can earn millions of dollars for a firm. So, he thought of beginning small and building an algorithm which could help his friend Adarsh V T predict stock prices.

## Issues addressed :

The major issue addressed here is the fact that most models to predict stock prices are dependent on data type and data distributions. Despite being a crude model here the values taken still ensure a value that is predicted to be reasonably correlated

 What correlation are we trying to make?

We are trying to using the SVR with different correlations to ensure that we can predict accurately the opening value of stock based on past data.

What is the relevance of this model to the business case?

The model helps crudely to determine when stocks can be bought and sold at the level of individuals, firms etc.

# **Model**

In the project here we have taken up the use of SVR (Support Vector Machine – Regression ) as the method to predict the value of opening stock values for a given company’s data.

The reason for trying out the use of Regression instead of Time Series Analysis used to predict stock is to because the SVR uses the same or similar principles for classification as that of SVM. (Support Vector Machine).

The main reason why this has been taken up is since the final idea is to ensure that we

* Minimize the error
* Maximize the margin ( individualizing the hyperplane )
* Error is marginally tolerated.

One of the bigger advantages of using SVM regression is that it can take into account input data that classical statistics might deem intolerant towards.

The complexity of the financial market makes the relationship between past and future data non-linear and AR, ARMA, ARIMA tend to be of lower standing as compared to that of the ANN, SVM and GARCH models.

Scalability : The other main reason is that with the use of SVR it is the fact that the model can be trained further which would further develop better classification.

Moreover multi-kernel approaches have been developed that can further increase the accuracy of results and forecasts. Instead of working on predictions alone forecasting more accurate results over time would be a big win.

With better classifiers it would be able to better determine weightage given to different values and how much exactly the weightage given to the value.

This can be further improved to understand the future values for more days ahead to understand the direction the stock will go in and further include factors impacting the forecasted value.

## THE SVM Model :

Support Vector Machines are one of the best binary classifiers. They create a decision boundary such that most points in one category fall on one side of the boundary while most points in the other category fall on the other side of the boundary.

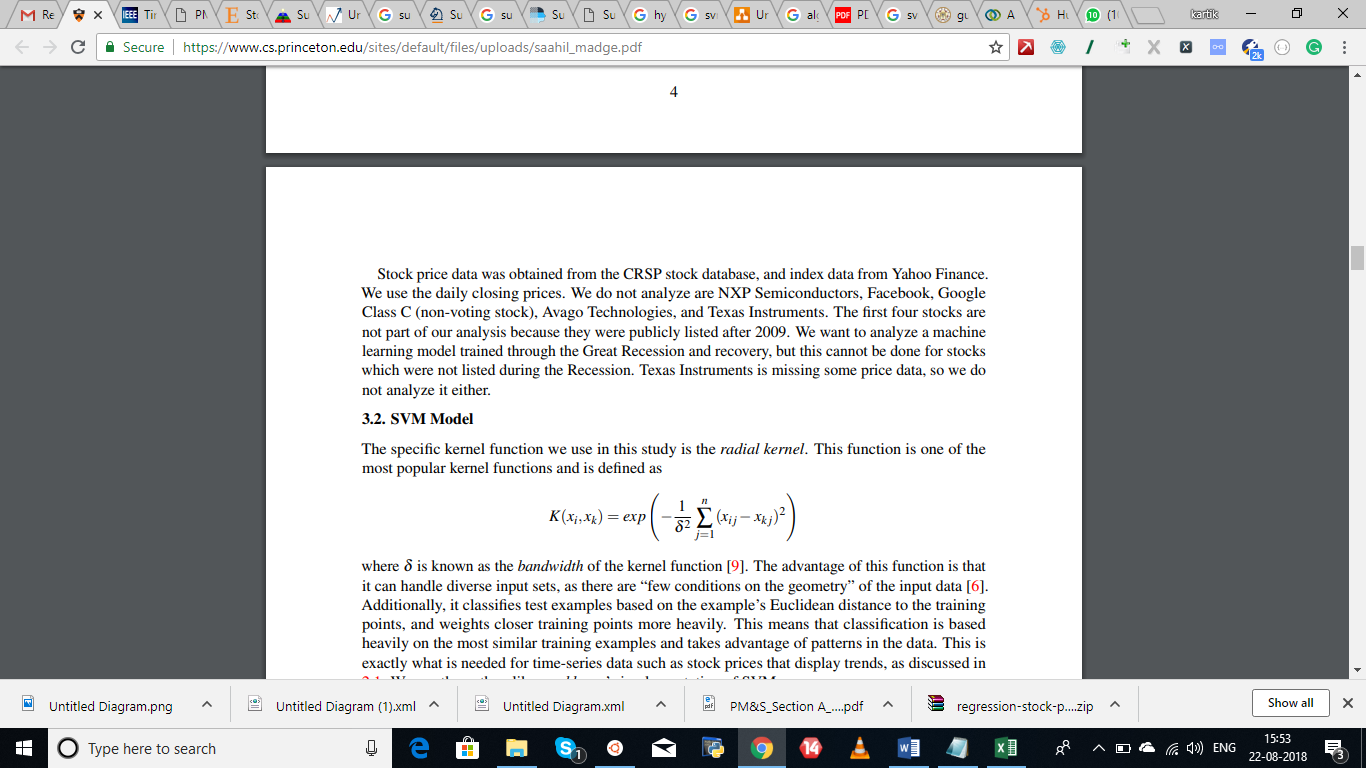
Elements in one category split out to have a positive sum while the other will have a sum less than zero.

The focus on using the SVM model with RBF Kernel for price forecasting.

The optimal hyperplane is such that we maximize the distance from the plane to any point. This is known as the margin. The maximum margin hyperplane (MMH) best splits the data. However, since it may not be a perfect differentiation, we can add error variables €.The crucial element is that only the points closest to the boundary matter for hyperplane selection; all others are irrelevant. These points are known as the support vectors, and the hyperplane is known as a Support Vector Classifier (SVC).

SVCs are limited in that they are only linear boundaries. SVMs fix this by applying non-linear kernel functions to map the inputs into a higher-dimensional space and linearly classify in that space. A linear classification in the higher-dimensional space will be non-linear in the original space. The SVM replaces the inner product with a more general kernel function K which allows the input to be mapped to higher-dimensions.

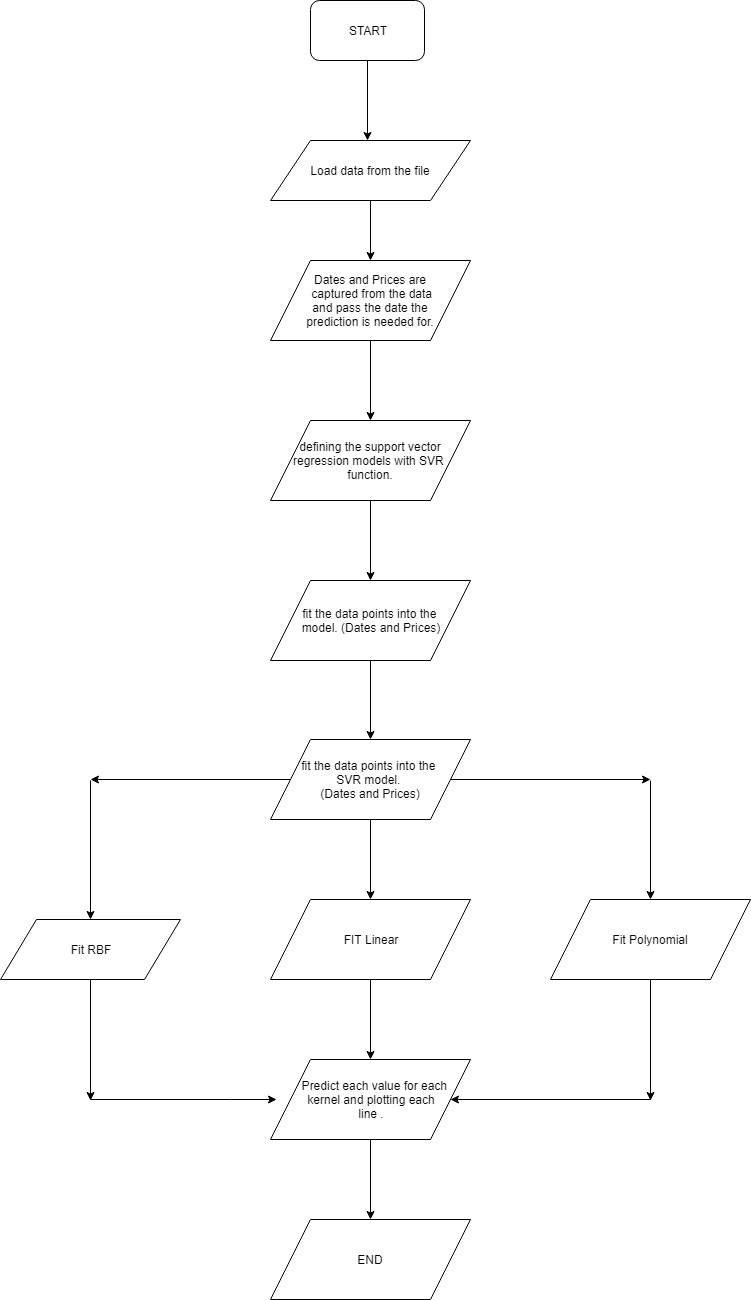
Thus in an SVM, y = β0 +∑αi yi K(x(i) , x) .What we are using is the RBF model ( Radial Kernel ) :



Here δ is known as the bandwidth of the kernel function. The advantage of this function is that it can handle diverse input sets, as there are “few conditions on the geometry” of the input data. Additionally, it classifies test samples based on the sample’s Euclidean distance to the training points, and weights closer training points more heavily.

We use the python library sklearn’s implementation of SVM.

# **Algorithm**



# **Model assumptions**

Since this is a very crude and basic approach in this project the focus is to avoid external factors and market impacts that happen such as mergers, acquisitions , new announcements and other external factors that as of now cannot be accounted for or cant be included.

It also assumes that the data is available for each day as we do not expect values for stocks to be not available for a day irrespective of the volume sold or bought that day.

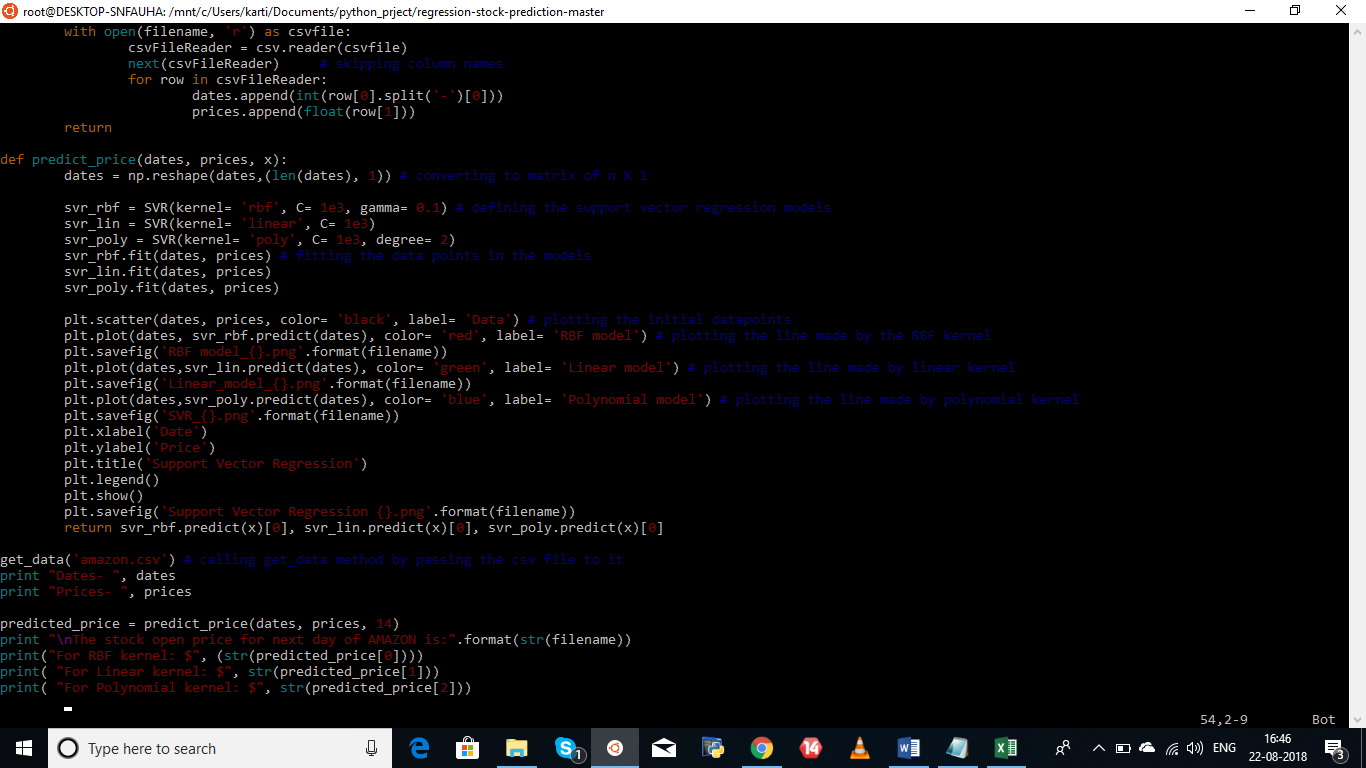
The model also assumes that the data is real and follows the unstructured and unorganized pattern a stock market does or there would be better methods to do the same.

Weekend model assumption is not taken into account and as of now the value will be shown for the opening value on Sat and Sun as well.

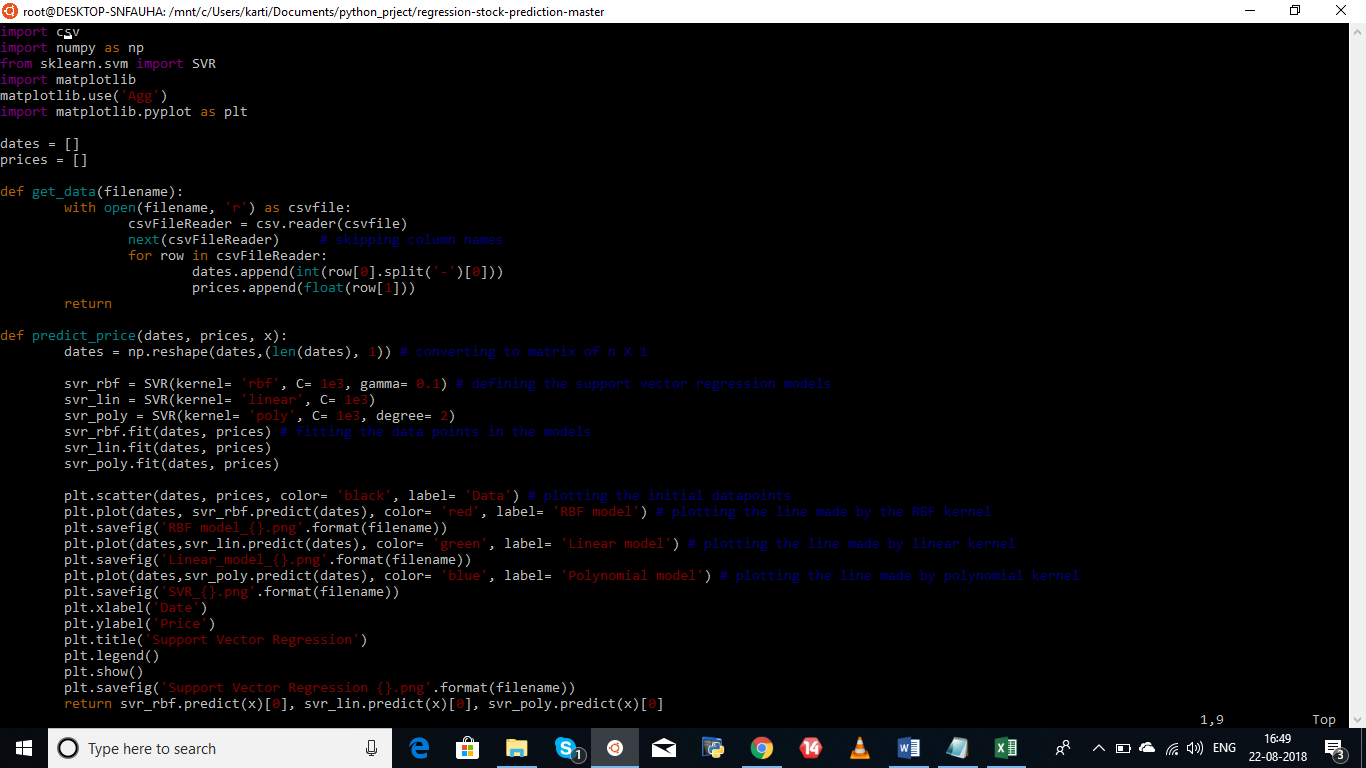
All the prices have the same unit i.e all are in dollars or rupees and so on without change across the data.

Date format is as dd-mm-yy with the – and not / or some other value in words.

# **Documentation of the code**



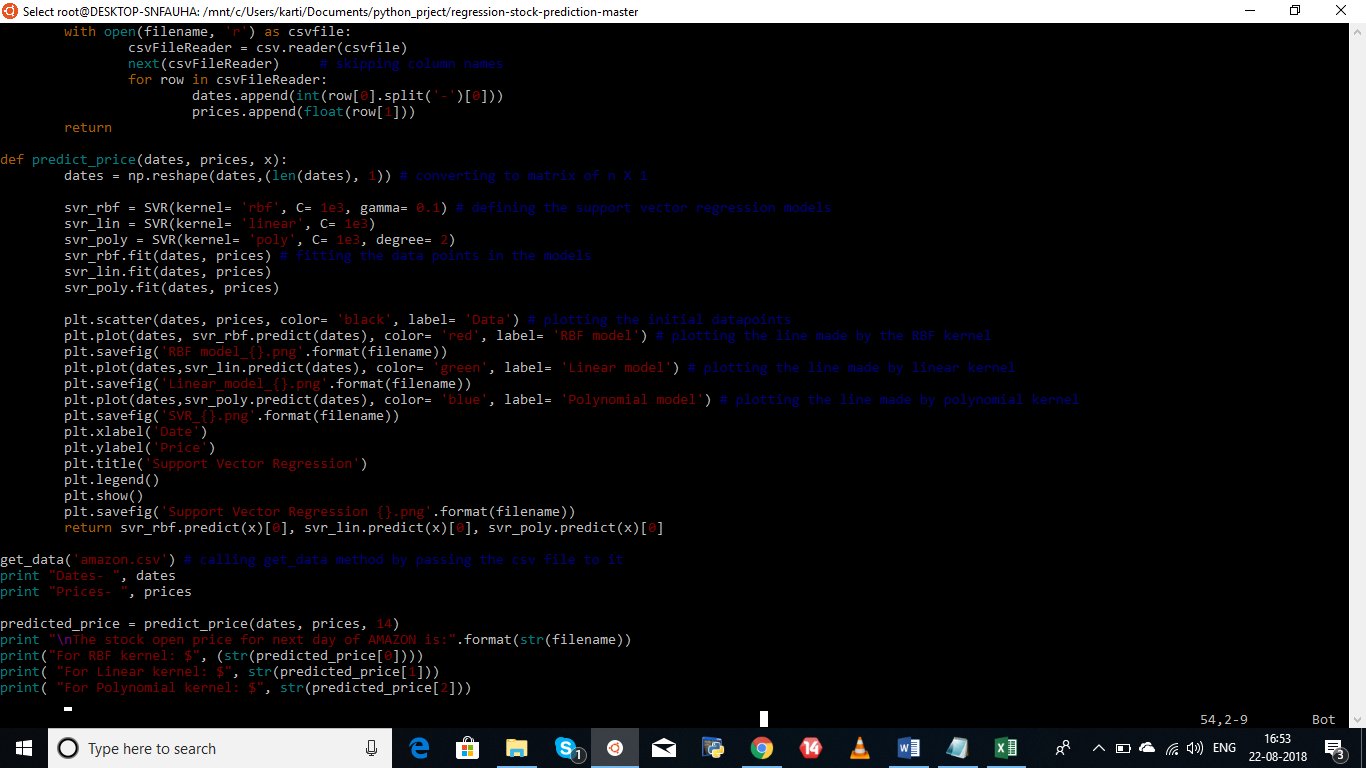
This is where the process begins where we get the data from the file. The dates and prices as output are printed here once the function get\_data() is run.

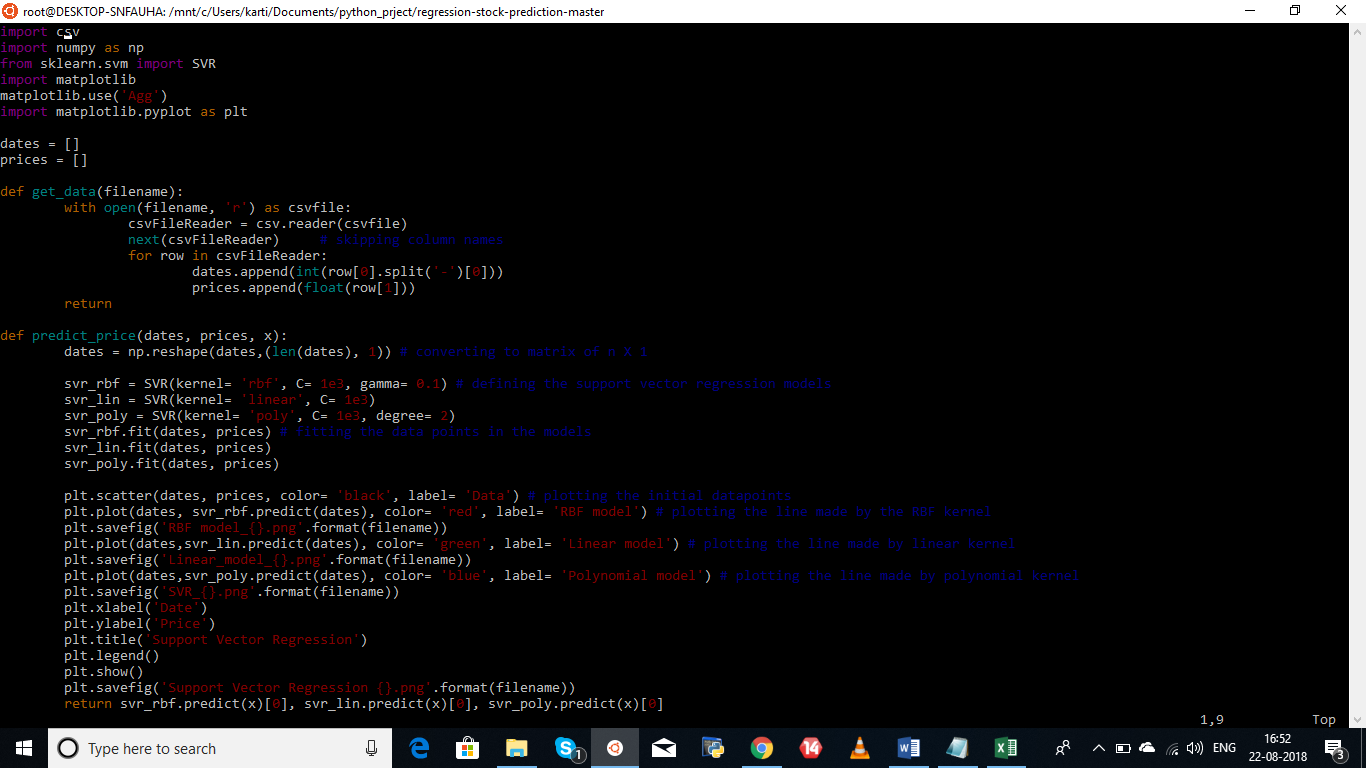


As mentioned above this is where we decide to retrieve the data from the file. This is the first step of our algorithm.

The second step is focused on predicting the opening price of the stock based on the data loaded.

This is where we call the predict\_price function passing the values of the dates and prices from get\_data(). We also pass an argument with the next date that we want to predict the value for.





The above figure shows the actual SVR used and has the variables svr\_rbf, svr\_lin, svr\_poly which define the support vector models.

The model is then fit in with data points : prices and dates . Post this step the primary focus is to look at how good the outputs are and the values that are predicted. For this we use plots and each kernel has been plotted separately to see the impacts. Similarly the output in itself has been predicted as separate kernel values and accuracy in the future could be further developed by giving weights as well if necessary.

## Explaining the svr function :

This function belongs to the class sklearn.svm

There are a few parameters passed through the function and the following express each one of them as shown in the python documentation.

**SVR**(kernel=’rbf’, degree=3, gamma=’auto’, coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache\_size=200, verbose=False, max\_iter=-1)

Parameters :

**C** : float, optional (default=1.0)Penalty parameter C of the error term.

**epsilon** : float, optional (default=0.1) : Epsilon in the epsilon-SVR model. It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.

**kernel** : string, optional (default=’rbf’)Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable. If none is given, ‘rbf’ will be used. If a callable is given it is used to precompute the kernel matrix.

**degree** : int, optional (default=3) : Degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.

**gamma** : float, optional (default=’auto’) : Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’. If gamma is ‘auto’ then 1/n\_features will be used instead.

**coef0** : float, optional (default=0.0) : Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.

**shrinking** : boolean, optional (default=True) : Whether to use the shrinking heuristic.

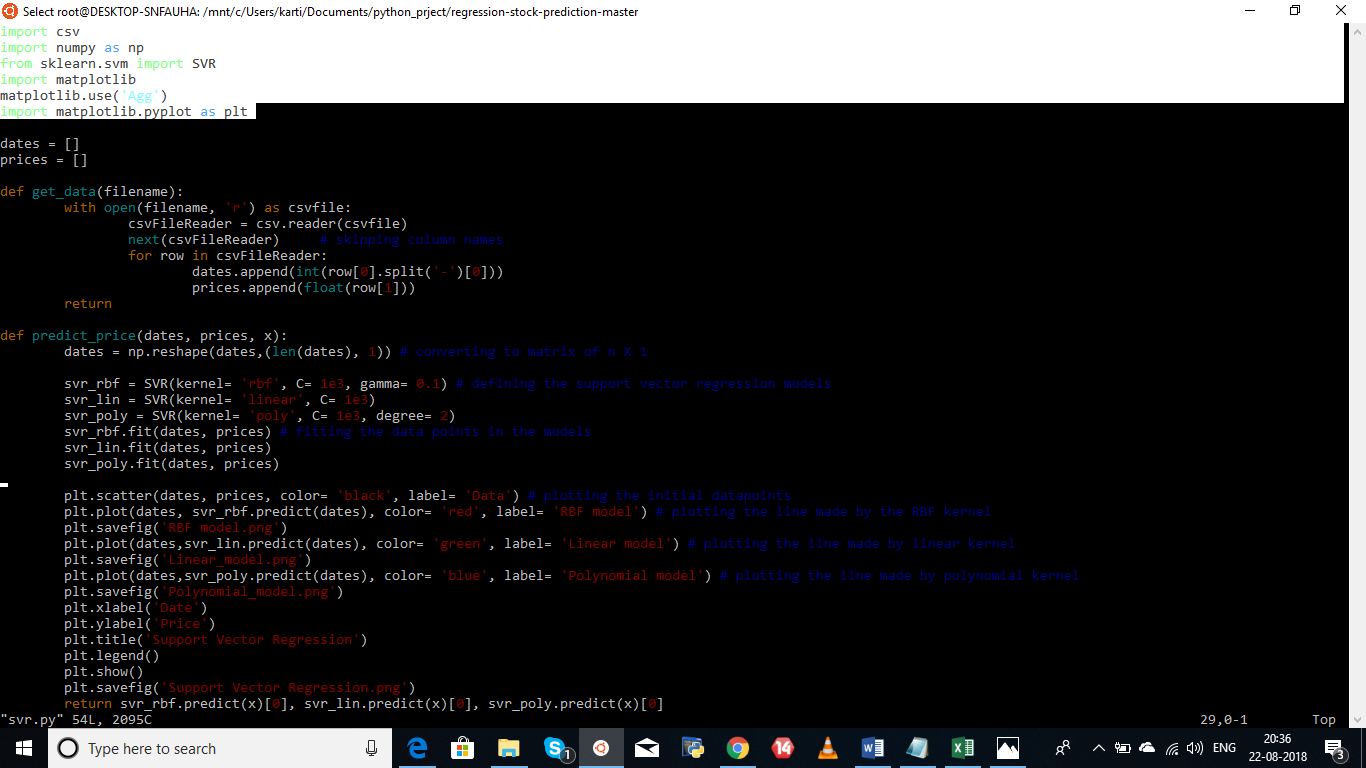
**tol** : float, optional (default=1e-3) : Tolerance for stopping criterion.

**cache\_size** : float, optional : Specify the size of the kernel cache (in MB).

The major reason to mention above fact is when we create tunnels for cross environment testing or runs caching overload might cause the trigger to fail.

**verbose** : bool, default: False : Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.

**max\_iter** : int, optional (default=-1) : Hard limit on iterations within solver, or -1 for no limit.



### 

### Libraries Used :

Import csv majorly used to import the data that is in the csv format

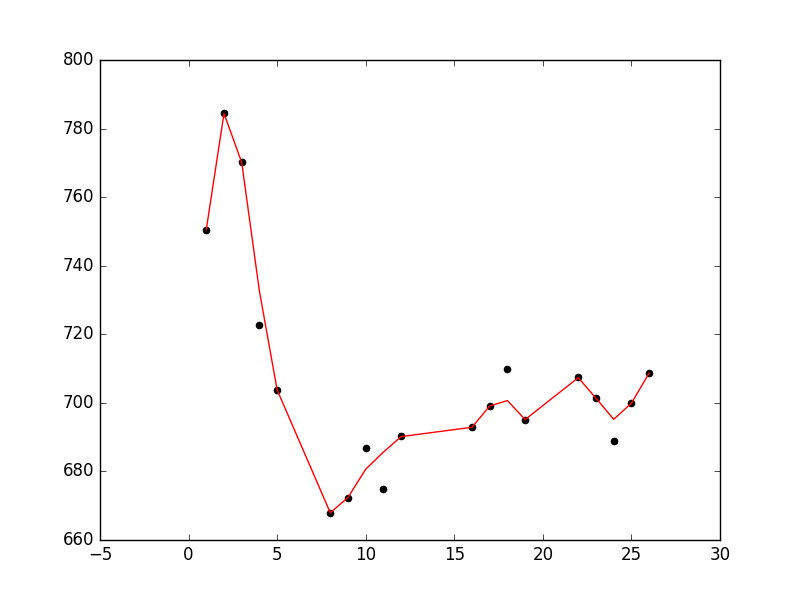
Import numpy is used to provide a high-performance multidimensional array and basic tools to compute with and manipulate these arrays.

from sklearn.svm import SVR : The predefined classes and functions are called so that the Support Vector Regression can be computed directly by passing suitable arguments.

Import Matplotlib is done primarily for plotting graphical outputs. The Aggregator statement is passed directly before to remove the tkinter or tclerror that comes up on display options on Ubuntu sytems or Ubuntu platforms.

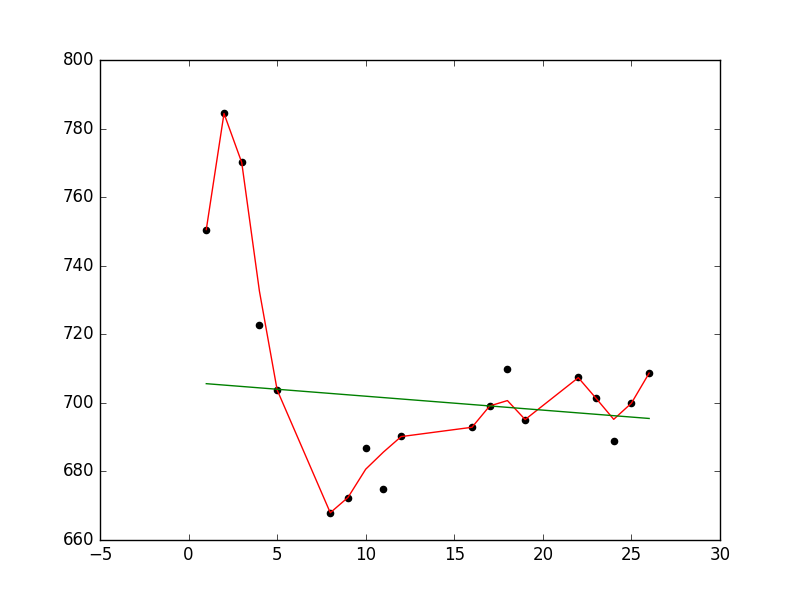
# **GRAPHICAL OUTPUTS :**

## RBF MODEL GRAPH OUTPUT



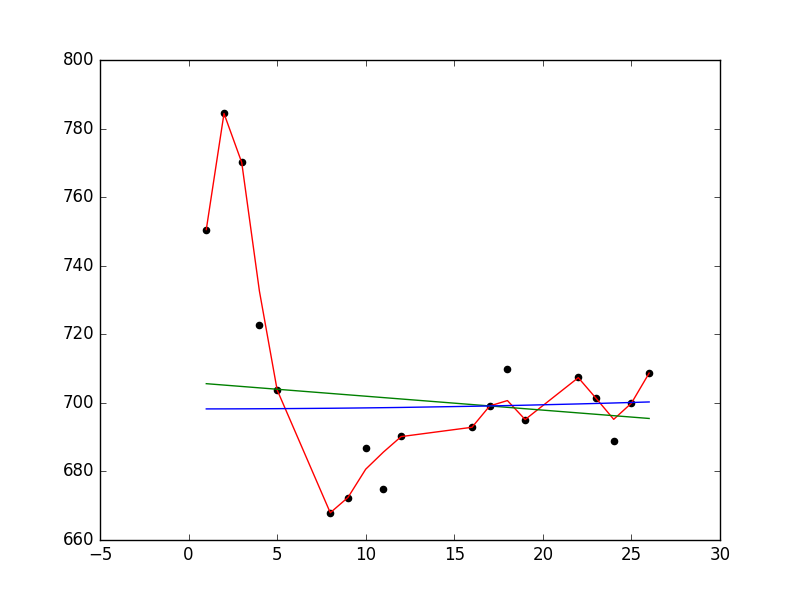
As we can see for Google’s data the model formed fits the data closely using the RBF model.

## LINEAR MODEL OUPUT :



Compared to the RBF model the linear as expected is way off in most cases.

## POLYNOMIAL MODEL OUTPUT :



The polynomial kernel is equally off for the given data set and we should thus stick to the output of the RBF model.

Dates- [26, 25, 24, 23, 22, 19, 18, 17, 16, 12, 11, 10, 9, 8, 5, 4, 3, 2, 1]

Prices- [708.58, 700.01, 688.92, 701.45, 707.45, 695.03, 710.0, 699.0, 692.98, 690.26, 675.0, 686.86, 672.32, 667.85, 703.87, 722.81, 770.22, 784.5, 750.46]

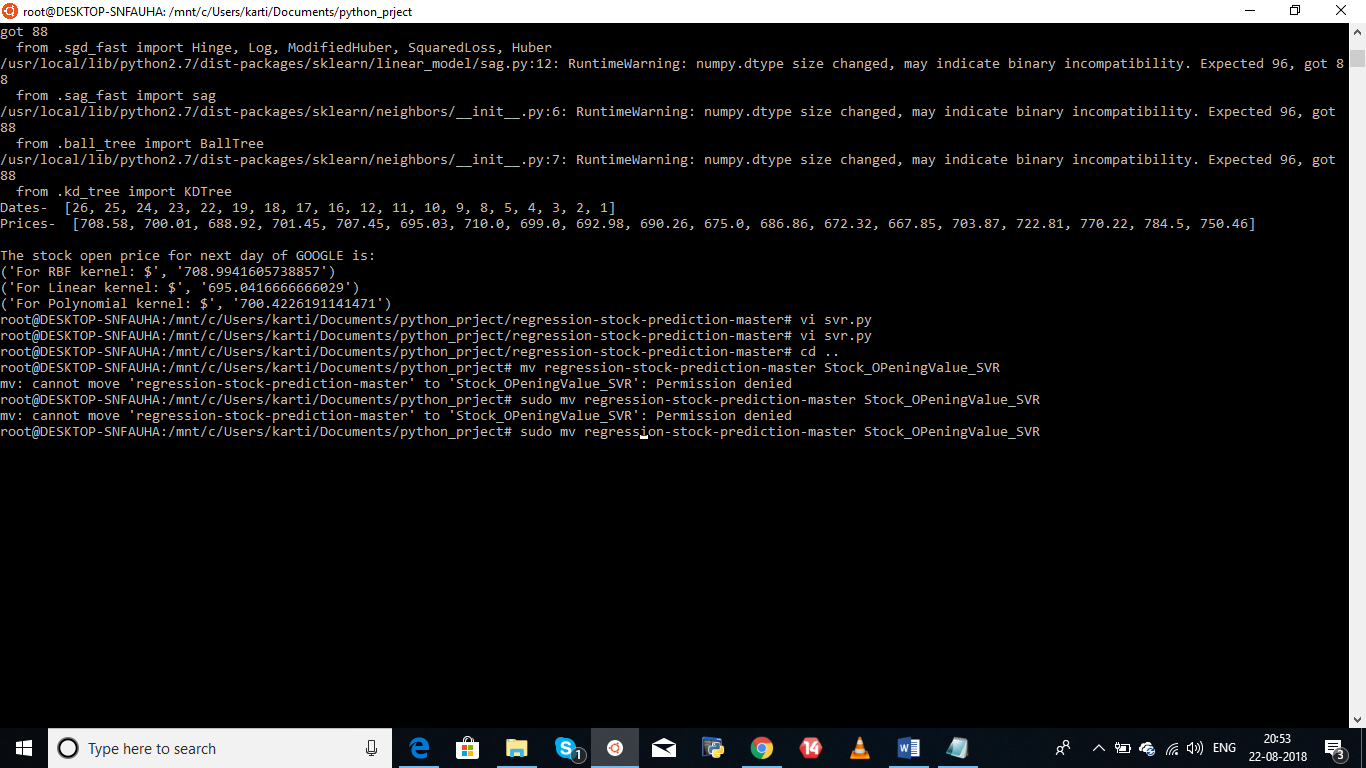
# **OUTPUT FROM THE SCRIPT :**

The stock open price for next day of GOOGLE is:

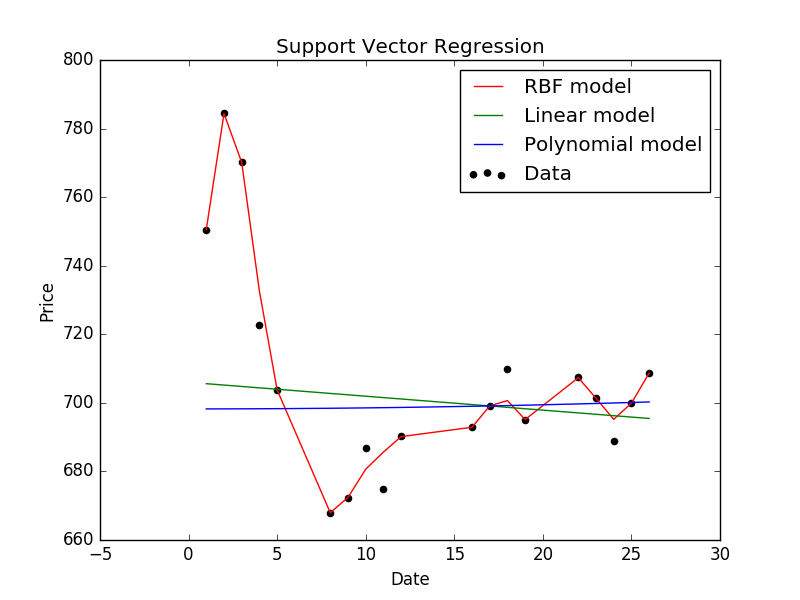
('For RBF kernel: $', '708.9941605738857')

('For Linear kernel: $', '695.0416666666029')

('For Polynomial kernel: $', '700.4226191141471')



# **OVERALL SUPPORT VECTRO REGRESSION RESULT :**



# Conclusion

As per this model, we can get a predictive value on the opening value of the stock the next day and this can help Bopin Valsan with the help of Gaurav Bhatt’s expertise ; to have a degree of confidence while trading on stocks and making better decisions as to when to buy and when to sell.

Moreover this being such a crude model with just two data points input; it is very marginal a scale at which this can operate . There is a far more expansive approach of the same if we include a lot more factors in the same model for a better and more realistic answer in real life scenarios.

# Recommendations and future steps

The main focus would be to develop the model further with addition of external factors that affect the stock market. Over time having a training data set and a test data set based on different scenarios primarily dipping stock, rising stock, flatline stock, and a turnaround are the main aspects that need to be trained for.

Classification parameters and weights must be tested on clearly and more importantly the focus must be clear on the fact that the error minimization must be done consistently irrespective of the number of hyperplanes that are used.

Adjust models speed as for large amounts of time duration if needed the process is slower and needs to be made faster.

# Exceptions

Preferable to have a working system with both python 2 and 3.

Pandas will outgrow 2.3 soon and will move to 2.6 and upgrading pandas is safe or it may fail in due time.

When no data is passed or available in the location specified.

When the date requested to be checked for is a non existent one.

## Version control

Release Date : 22-08-2018 (latest change)

Version : 0.1.1

# **APPENDIX**

Python Documentation for the svm library : Scikit Library

<http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html>

Data Sources : Yahoo Finance

<https://finance.yahoo.com/quote/> {Company stock name}

Example 1 of data set : GOOG.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Open | High | Low | Close | Volume |
| 25-Feb-16 | 700.01 | 705.98 | 690.58 | 705.75 | 1631855 |
| 24-Feb-16 | 688.92 | 700 | 680.78 | 699.56 | 1958611 |
| 23-Feb-16 | 701.45 | 708.4 | 693.58 | 695.85 | 1999699 |
| 22-Feb-16 | 707.45 | 713.24 | 702.51 | 706.46 | 1946067 |
| 19-Feb-16 | 695.03 | 703.08 | 694.05 | 700.91 | 1582260 |
| 18-Feb-16 | 710 | 712.35 | 696.03 | 697.35 | 1859130 |
| 17-Feb-16 | 699 | 709.75 | 691.38 | 708.4 | 2466808 |
| 16-Feb-16 | 692.98 | 698 | 685.05 | 691 | 2497024 |
| 12-Feb-16 | 690.26 | 693.75 | 678.6 | 682.4 | 2129831 |
| 11-Feb-16 | 675 | 689.35 | 668.87 | 683.11 | 3007223 |
| 10-Feb-16 | 686.86 | 701.31 | 682.13 | 684.12 | 2627379 |
| 09-Feb-16 | 672.32 | 699.9 | 668.77 | 678.11 | 3604335 |
| 08-Feb-16 | 667.85 | 684.03 | 663.06 | 682.74 | 4212541 |
| 05-Feb-16 | 703.87 | 703.99 | 680.15 | 683.57 | 5069985 |
| 04-Feb-16 | 722.81 | 727 | 701.86 | 708.01 | 5145855 |
| 03-Feb-16 | 770.22 | 774.5 | 720.5 | 726.95 | 6162333 |
| 02-Feb-16 | 784.5 | 789.87 | 764.65 | 764.65 | 6332431 |
| 01-Feb-16 | 750.46 | 757.86 | 743.27 | 752 | 4801816 |

Example 2 of data set AZ :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Opening Price | Highest Price | Lowest Price | Closing Price | Direction |
| 21-02-2017 | 848.84 | 857.98 | 847.25 | 856.44 | up |
| 22-02-2017 | 856.95 | 858.43 | 852.18 | 855.61 | down |
| 23-02-2017 | 857.57 | 860.86 | 848 | 852.19 | down |
| 24-02-2017 | 844.69 | 845.81 | 837.75 | 845.24 | down |
| 27-02-2017 | 842.38 | 852.5 | 839.67 | 848.64 | up |
| 28-02-2017 | 851.45 | 854.09 | 842.05 | 845.04 | down |
| 01-03-2017 | 853.05 | 854.83 | 849.01 | 853.08 | up |
| 02-03-2017 | 853.08 | 854.82 | 847.28 | 848.91 | down |
| 03-03-2017 | 847.2 | 851.99 | 846.27 | 849.88 | up |
| 06-03-2017 | 845.23 | 848.49 | 841.12 | 846.61 | down |
| 07-03-2017 | 845.48 | 848.46 | 843.75 | 846.02 | down |
| 08-03-2017 | 848 | 853.07 | 846.79 | 850.5 | up |
| 09-03-2017 | 851 | 856.4 | 850.31 | 853 | up |
| 10-03-2017 | 857 | 857.35 | 851.72 | 852.46 | down |
| 13-03-2017 | 851.77 | 855.69 | 851.71 | 854.59 | up |
| 14-03-2017 | 853.55 | 853.75 | 847.55 | 852.53 | down |
| 15-03-2017 | 854.33 | 854.45 | 847.11 | 852.97 | up |
| 16-03-2017 | 855.3 | 855.5 | 850.51 | 853.42 | up |
| 17-03-2017 | 853.49 | 853.83 | 850.64 | 852.31 | down |
| 20-03-2017 | 851.51 | 857.8 | 851.01 | 856.97 | up |
| 21-03-2017 | 858.84 | 862.8 | 841.31 | 843.2 | down |
| 22-03-2017 | 840.43 | 849.37 | 839.05 | 848.06 | up |
| 23-03-2017 | 848.2 | 850.89 | 844.8 | 847.38 | down |
| 24-03-2017 | 851.68 | 851.8 | 843.53 | 845.61 | down |
| 27-03-2017 | 838.07 | 850.3 | 833.5 | 846.82 | up |
| 28-03-2017 | 851.75 | 858.46 | 850.1 | 856 | up |
| 29-03-2017 | 859.05 | 876.44 | 859.02 | 874.32 | up |
| 30-03-2017 | 874.95 | 877.06 | 871.66 | 876.34 | up |
| 31-03-2017 | 877 | 890.35 | 876.65 | 886.54 | up |
| 03-04-2017 | 888 | 893.49 | 885.42 | 891.51 | up |
| 04-04-2017 | 891.5 | 908.54 | 890.28 | 906.83 | up |
| 05-04-2017 | 910.82 | 923.72 | 905.62 | 909.28 | up |
| 06-04-2017 | 913.8 | 917.19 | 894.49 | 898.28 | down |
| 07-04-2017 | 899.65 | 900.09 | 889.31 | 894.88 | down |
| 10-04-2017 | 899.63 | 908.51 | 899 | 907.04 | up |
| 11-04-2017 | 907.04 | 911.24 | 897.5 | 902.36 | down |
| 12-04-2017 | 903.09 | 904.09 | 895.25 | 896.23 | down |
| 13-04-2017 | 891.45 | 894.97 | 884.49 | 884.67 | down |

Example 3 : Data Set of Twtr

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 07-03-2017 | 15.52 | 15.73 | 15.16 | 15.18 | down |
| 08-03-2017 | 15.19 | 15.36 | 15.08 | 15.24 | up |
| 09-03-2017 | 15.24 | 15.39 | 15.03 | 15.22 | down |
| 10-03-2017 | 15.23 | 15.27 | 14.94 | 15.12 | down |
| 13-03-2017 | 15.1 | 15.27 | 15.1 | 15.21 | up |
| 14-03-2017 | 15.2 | 15.58 | 15.15 | 15.32 | up |
| 15-03-2017 | 15.25 | 15.3 | 14.85 | 15.03 | down |
| 16-03-2017 | 15.08 | 15.28 | 15.04 | 15.19 | up |
| 17-03-2017 | 15.2 | 15.23 | 15.03 | 15.08 | down |
| 20-03-2017 | 15.11 | 15.11 | 14.82 | 15.09 | up |
| 21-03-2017 | 15.08 | 15.1 | 14.5 | 14.54 | down |
| 22-03-2017 | 14.5 | 14.99 | 14.32 | 14.98 | up |
| 23-03-2017 | 14.99 | 15.05 | 14.73 | 14.93 | down |
| 24-03-2017 | 15.06 | 15.37 | 15.03 | 15.14 | up |
| 27-03-2017 | 15.02 | 15.06 | 14.75 | 14.99 | down |
| 28-03-2017 | 15 | 15.17 | 14.8 | 14.94 | down |
| 29-03-2017 | 14.86 | 15.06 | 14.7 | 15.04 | up |
| 30-03-2017 | 15.05 | 15.08 | 14.9 | 14.92 | down |
| 31-03-2017 | 14.93 | 15.06 | 14.91 | 14.95 | up |
| 03-04-2017 | 14.97 | 14.98 | 14.65 | 14.84 | down |
| 04-04-2017 | 14.75 | 14.76 | 14.58 | 14.69 | down |
| 05-04-2017 | 14.61 | 14.82 | 14.41 | 14.53 | down |
| 06-04-2017 | 14.53 | 14.62 | 14.3 | 14.39 | down |
| 07-04-2017 | 14.36 | 14.43 | 14.25 | 14.29 | down |
| 10-04-2017 | 14.3 | 14.46 | 14.2 | 14.36 | up |
| 11-04-2017 | 14.3 | 14.4 | 14.2 | 14.31 | down |
| 12-04-2017 | 14.34 | 14.78 | 14.26 | 14.42 | up |
| 13-04-2017 | 14.49 | 14.5 | 14.22 | 14.3 | down |