ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF SCIENCE AND LETTERS

GRADUATION PROJECT

DATA ANALYSIS OF STOCK MARKETS USING ECONOPHYSICAL METHODS

Alper ANAPALI

Physics Engineering

SPRING 2019

FOREWORD

I would like to express my appreciation for my advisor Dr. M. Murat Sunar for his support in this research. It has been a privilege to complete my study under his supervisions.

Also for their influence on me and making me eager to learn more about coding and data science I would like to thank Doç. Dr. Tolga Birkandan and Doç. Dr. Altan Çakır.

May 2019 Alper ANAPALI

(Physics Engineering Student)

TABLE OF CONTENTS

PAGE

ABBREVIATIONS AND SYMBOLS

LSTM **:** Long Short Term Memory

GNP **:** Gross National Product

ML **:** Machine Learning

K **:** Boltzman Constant

LIST OF TABLES

PAGE

LIST OF FIGURES

Data Analysis of Stock Markets Using Econophysical Methods

SUMMARY

Within the past decade data analysis become so popular and a vast amount of data has become available allowing empirical studies of market behavior to be performed. Technological advancements in computer science made it possible to process very large amounts of data in relatively small time.

Different business, science, healthcare and social science domains make use of data science. From detecting alpha particles coming from sun to diagnosing lung cancer it has a major field of application. With so much higher processing speed than a group of human analyist can handle, finding correlations in nature or in a specific samples became possible.

Developments in computer science has created a high demand in data analysis, companies started trying to imporve their efficieny by analyzing their own data. Economics and finance area which contain the largest data sources changed their persperctive to predict and calculate the probablity of events in future markets to use machine learning. Many economists could not keep up with the new tools due to their lack of statistical and mathematical knowledge.

In mid 1990s many physicists started to use their mathematics and statistics abilites to analyze stock markets. Econophysics, a field uses physics concepts, greatly thermodynamics to interperate future economical events and market behaviour first stated and founded by H. Eugene Stanly a statisctical physicist who have introduced many models using fluid mechanics. From then on many physics concepts such as quantum statistics or electrodynamics have been applied to model economic movements .

This data analysis model uses an ideal gas approach to predict stock market data of S&P 500 with long-short term memory neural networks. As programming language python is chosen which is one of the most popular programming language in data analysis.

As an initial step, from S&P 500 leading technology companies' stock price datas are extracted. Features are created from this data and properties of the data are investigated and graphed.

During analysis, linear and polynomial regressions are applied to stock data to explore its smilarities with an ideal gas.

Finally, an lstm model from keras is applied to predict future prices.Prediction is based on the two variables, temperature and volume. The predictions may be used as an indicator in future analysis of stock market.

If this model is successfull, a real gas model could also be applied to stock market data in future.

1. INTRODUCTION

1.1 Purpose of This Work

This work aims to create a stock market indicator to give a different insight to traders based on the realtion between ideal gas behavior and stock properties such as price, volume and GNP. The validity and application of the thermodynamic model of the ideal gas in economics will be investigated.

1.2 Background

A similar ideal-gas approach to stock market was made by Bikas K. Chakrabarti and Arnab Chatterjee (2003). In their model it is considered that the ideal-gas models of trading markets, where each agent is identified with a gas molecule and each trading as an elastic or money-conserving (two-body) collision. Their conclusion and results gave a solid support to a machine learning approach like this in this work. It is stated that a simple ideal-gas like market model compare well with real market observations [1].

Tinko Eftimov discussed the validity and application of the thermodynamic model of the ideal gas in economics and shown that while the aggregate-demand curve is described by an isothermal process, the isobaric and isochoric processes have important analogies in economics [2].

1.3 Hypothesis

Stock market price, volume and that nations GNP behaves like an ideal gas' pressure, volume and temperature. Using LSTM over temperature (GNP) divided by volume can give an insight and help creating an indicator for predicting the future price changes. This kind of LSTM id different due to the difference between trained quantities and predicted ones.

2. ECONOPHYSICAL APPROACH

2.1 Defining ideal gas

The ideal gas law is a simple equation to calculate the relationship between pressure, volume and temperature combining Charles's Law, Boyle's Law, and Gay-Lussac's Law. In this work we are not going to focus on constants and how many moles of gases exist.

Asumptions of an ideal gas:

1. Volume of particles are neglected.
2. There is no repulsion or attarction between particles.
3. Particles move randomly in accordance with Newton's Laws of Motion
4. Particles make elastics collisions and do not lose any energy.

The Ideal Gas Law makes it possible to calculate simply to determine physical properties of a system. [4]

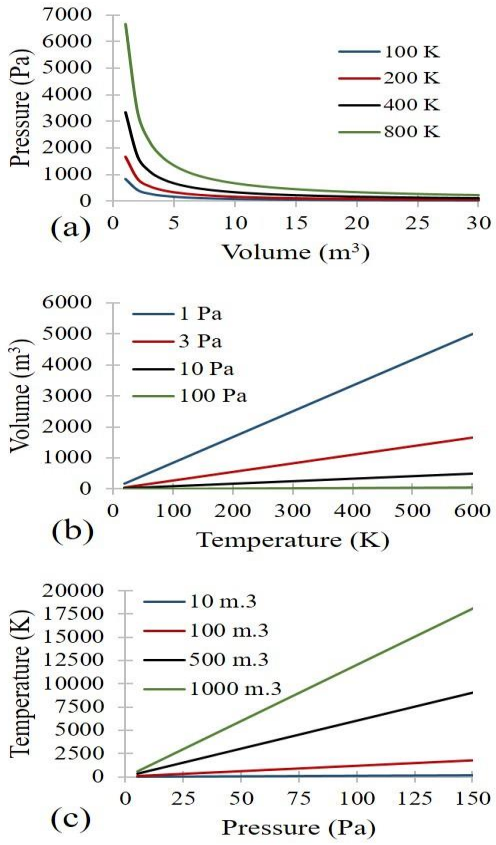

Figure 2.1: (a) the change in pressure and volume at constant temperature, (b) the change in temperature and volume at constant pressure, and (c) the change in pressure and temperature at constant volume. [5]

2.2 Ideal gas pressure, volume and temperature in stock market

The relationship between the money stock and output in an economy has been widely researched and the connections between temperature and the velocity of circulation has been pointed to be identical to equation [3]

Pressure and volume of the ideal gas represents price level and volume in an economy, number of atoms N shows the number of notes, deposits, and money instruments in circulation, k the nominal amount of value that each note is carry, and T is the velocity of circulation (In this case will be represented with GNP).

Money is a homogenous commodity and moves in the opposite direction to other economic factors such as labour, materials and output. A change in the stock of money, not accompanied by an offsetting change in volume output, could result in changes in both the index of trading value and prices in an economy.[5]

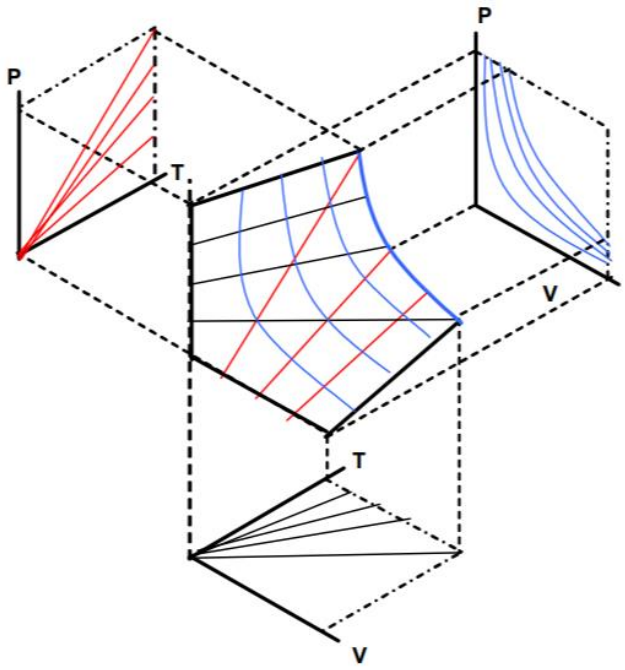

Figure 2.2 : Price, volume and the index of trading value [5]

$$[P.V]_{money} = [N.k.T]_{money} \quad (2.1)$$

To construct an ideal economic equation a dimensional analysis should be performed. Thermodynamic systems pressure P is measured by F Force per unit of area (L x L), and energy J i a product of force F times distance changed L. The Boltzman constant k is defined as energy (J) per molecule per degree of temperature (T).

$$\left(\frac{J}{L^3}\right)\times\left(L^3\right)=N\times\left(\frac{J}{NT}\right)\times T \qquad (2.2)$$

Similarly to describe an economic system, dimensions price P is measured as value J per volume V, and the monetary constant k is measured as value J per carrier per index of the trading value T.

Therefore the structure of the ideal gas equation and the ideal economic equation matched so far, to define our variables; pressure P with price per unit, volume V with units of output/consumption, temperature T with the index of trading, the Boltzmann constant k with the monetary constant per unit of carrier and index of trading value. It may be suggested that value in an economic system might have some equivalence to heat content in a thermodynamic system. [5]

3. INTRODUCTION TO USED PYTHON MODULES

Python has emerged over the last decades as one of the best tool to compute, analyze and graph scientific tasks. Usefullness of Python comes from the large and active (open source) ecosystem of third party packages and modules. When it comes to data science object oriented programming languages are prefered. Python has been the favorite platform for data scientists along the R language because of its particular tools such as Pandas , Numpy, Matplotlib and Scikit-Learn libraries. In this work Python version 3.7.3 is used. [6]

3.1 Pandas

When importing the pandas module “as pd” is used as a convention. Pandas’ main object is the pd.DataFrame which is two-dimensional and each column is a pd.Series object. Series object is basically interchangeable with a one-dimensional NumPy array. The main difference is the presence of the index: the Numpy Array has an implicitly defined integer index used to access the values, the Pandas Series has an explicitly defined index associated with the values.

[1]:
```
import numpy as np
import pandas as pd

dictionary_num = {"A": 1, "B" : 2, "C" : 3}
dictionary_sym = {"A": "a", "B" : "b", "C" : "c"}

d_n = pd.Series(dictionary_num)
d_s = pd.Series(dictionary_sym)

data_frame = pd.DataFrame( {"Numbers" : dictionary_num, "Sybols" : dictionary_sym})
data_frame.to_csv("example_DataFrame")
data_frame
```

[1]:

	Numbers	Sybols
A	1	a
B	2	b
C	3	c

DataFrame object of pandas has many features and can be investigated or changed with pandas functions. In this work following pandas functions are used for the analysis and data preperation processes ;

pd.read_csv() function is used for reading csv format files from the directory.

```
df= pd.read_csv("example_DataFrame")
```

```
data_frame.head()
```

	Numbers	Sybols
A	1	a
B	2	b
C	3	c

```
data_frame.describe()
```

	Numbers
count	3.0
mean	2.0
std	1.0
min	1.0
25%	1.5
50%	2.0
75%	2.5
max	3.0

```
data_frame["Numbers"]
```

```
A    1
B    2
C    3
Name: Numbers, dtype: int64
```

```
data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, A to C
Data columns (total 2 columns):
Numbers    3 non-null int64
Sybols     3 non-null object
dtypes: int64(1), object(1)
memory usage: 152.0+ bytes
```

```
data_frame.columns
```

```
Index(['Numbers', 'Symbols'], dtype='object')
```

```
data_frame.drop("Numbers", 1, inplace= True)
data_frame
```

	Symbols
A	a
B	b
C	c

3.2 Matplotlib

In this project matplotlib module is used for graphing the data. Matplotlib's most important feature is its ability to play well with many operating systems and graphics backends. Matplotlib supports dozens of backends and output types, which means you can count on it to work regardless of which operating system you are using or which output format you wish. This cross-platform, everything-to-everyone approach has been one of the great strengths of Matplotlib.

In this work another graphing module for python Seaborn is also used to graph heatmaps which is one of the strongest attributes of the module.

Module is imported as "plt" as a Python convention. Style function of matplotlib is used for changing graphics apperance and style ("ggplot" is used).

```
import matplotlib.pyplot as plt
from matplotlib import style
```

```
# https://matplotlib.org/gallery/style_sheets/style_sheets_reference.html
# other styles can be found in the link.
style.use("ggplot")
```

plt.figure(), plt.title(), plt.xlabel(), plt.ylabel() functions are used to adjust the attributes of the graphs figure.

plt.plot() and plt.scatter() functions are used for graphing line and scatter plots in the project.

```
plt.figure(figsize=(5,5))
plt.title("Matplotlib Demonstration", fontsize=20, color="g")
plt.xlabel("Index", fontsize=20)
plt.ylabel("Value", fontsize=20)
plt.plot(data_frame["Numbers"])
plt.show()
```

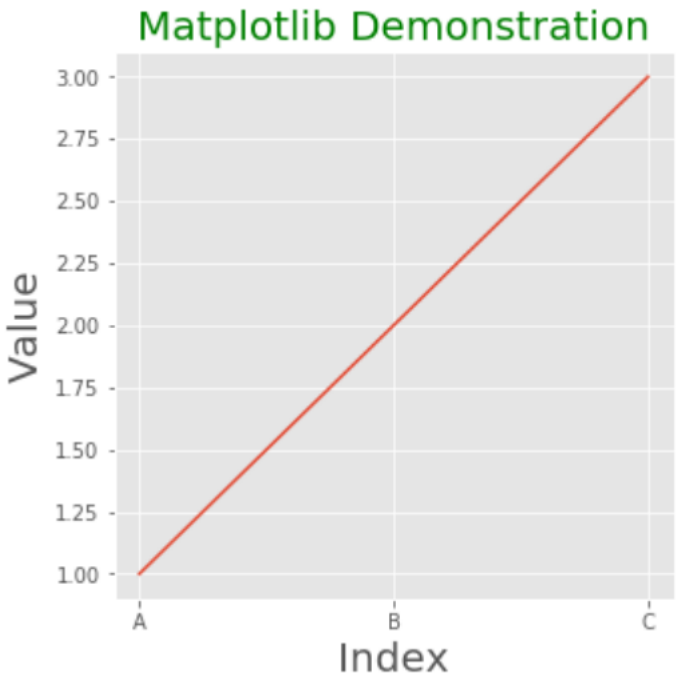

Figure 3.1: Matplotlib demonstration graph

3.3 Numpy

Takes its name from Numerical Python. Numpy module is used for mathmetical calculations and creating arrays (vector and matrix like objects of numpy). Numpy is utilized in many areas from random number generation to matrix multiplications even Fourirer transformations. In this project only mathematical functions of numpy are used.

By convention numpy is imported as “np”

```
import numpy as np
```

Mathematical functions mean(), max() and min() from numpy is used to calculate Mean Squared Errors and for Scaling the data.

```
np.mean(data_frame["Numbers"])
```

```
2.0
```

```
np.max(data_frame["Numbers"])
```

```
3
```

```
np.min(data_frame["Numbers"])
```

```
1
```

3.4 Scikit-Learn, TensorFlow and Keras

Scikit-Learn library contains many machine learning algorithms and additionaly model selection and scaling methods. It is built on numpy, scipy and matplotlib. For machine learning it has clasification, regression and clustering applications. In this project it will mostly be focused on regression models from Scikit-Learn.

The following functions from Scikit-Learn module are used in the project.

```
from sklearn import preprocessing, svm, model_selection
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

The following are a set of methods intended for regression in which the target value is expected to be a linear combination of the features. In mathematical notation, if y is the predicted value in the Equation (3.1) .

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p \tag{3.1}$$

LinearRegression() of scikit-learn fits a linear model with coefficients w=(w1,...,wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

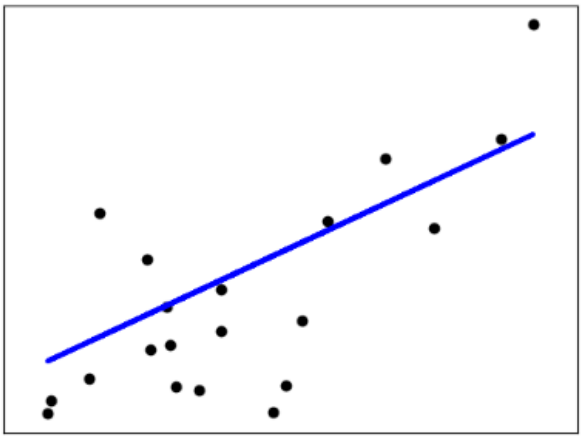

Figure 3.2: LinearRegression() of Scikit-Learn.

Scikit-Learn function **sklearn.preprocessing.PolynomialFeatures()** is also used to fit polynomial models in our model.

This function generates a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. For example, if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2].

```
# X being the values to fit
poly = PolynomialFeatures(2)
poly.fit_transform(X)
```

Degree of the polynomial can be set in the fucntion.However we need a second degree polynomial fit for the ideal gas case. (for example for a 3-dim polynomial fit PolynomialFeatures(3) is used)

For the LSTM model Keras libary's Squential and LSTM functions are used on Tensorflow backend.

```
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
import tensorflow as tf
```

Time series analysis refers to the analysis of change in the trend of the data over a period of time. Time series analysis has a variety of applications. One such application is the prediction of the future value of an item based on its past values. LSTM (Long short term memory) analiysis is one of the most used ones in the stock market predictions. The ability of LSTM to remember previous information makes it ideal for such tasks.

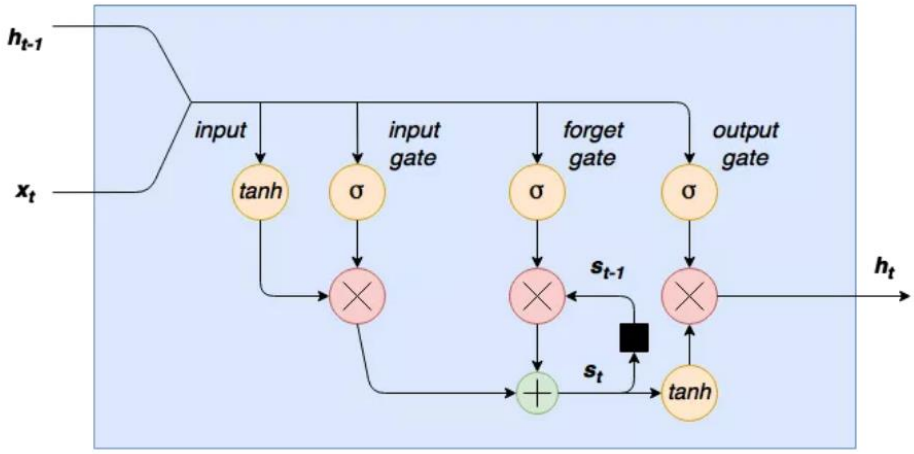

Figure 3.3: LSTM cell diagram

In the Figure 3.3 inputs x_t and H_{t-1} are concatenated (H_{t-1} is the previous output). This concatenated value then goes into the tanh() function. In the further steps it is fed into a sigmoid function.A sigmoid function used in almost every deep learning network. [7]

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{3.2}$$

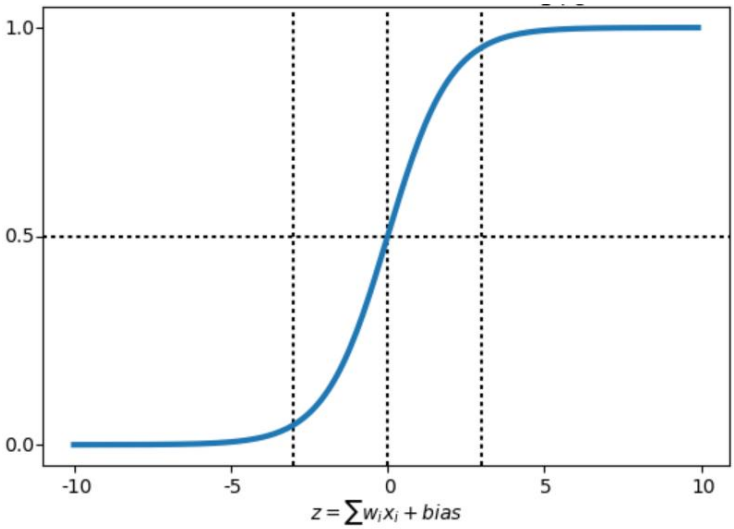

Figure 3.4: Sigmoid function

Input gate is a layer of sigmoid activated nodes whose output is multiplied by the concataneted input. These input gate sigmoids can act to shut down any elements of the input vector that aren't required. A sigmoid function outputs values between 0 and 1, so the weights connecting the input to these nodes can be trained to output values close to zero to "switch off" certain input values.[7]

The next step in the flow of data through this cell is the internal state / forget gate loop. LSTM cells have an internal state variable. This variable, lagged one time step and addwed to the input data. This addition operation, instead of a multiplication operation, helps to reduce the risk of vanishing gradients.

However, this recurrence loop is controlled by a forget gate – this works the same as the input gate, but instead helps the network learn which state variables should be "remembered" or "forgotten" (Where is takes the memory word in its name from). [7]

3.5 Other used modules

Datetime module is used to set starting and ending dates fot intented intervals of stock data.

```
import datetime as dt
start = dt.datetime(2005,1,1)
end = dt.datetime(2019,1,1)
```

Pythons built-in module os is used to create files in the operating system.

```
import os
```

A very usefull library to graph heatmaps Seaborn module is used.

4.DATA PROCESSING

Collecting data from the source is the first step in data processing data might be organised or not. It is important that the data sources available are trustworthy and well-built.

Once the data is collected it should be prepared for intended use. Raw data should be organized for the suitble structre to be processed by the algorithms.

4.1 pandas_datareader

In this project data is collected from Yahoo's stock price data using pandas_datareader() which is quite reliable. Since it is collected inplace from internet this program does not depend on which computer its running on.

```
import pandas_datareader.data as web
web.DataReader("AAPL", "yahoo", start, end)
```

There are many companies to collect data from to construct the model not only depending on 1 stock market data. Here we create a list of company tickers (ticker is the short name for the companies most of the time has 3 or 4 letters).

```
sp_500_tech_companies_tickers = ["MSFT", "AAPL", "AMZN", "GOOGL", "INTC", "ORCL", "NVDA", "TSLA"]
```

A good data analysis notebook should be interactive and could run in any environment. So that we should define functions and classes for further use and for program to be understandable for anyone whos reading the code.

A function to get data from internet is defined below.

```
def get_data(ticker):

    '''
    This function gets the data from yahoo saves it and labels it.
    ticker must be list type even if it has a single element.
    '''

    if not os.path.exists("stock_data"):
        os.makedirs("stock_data")

    for ticker in ticker:

        if not os.path.exists("stock_data/{}.csv".format(ticker)):
            df = web.DataReader(ticker, "yahoo", start, end)
            df.to_csv("stock_data/{}.csv".format(ticker))


        else:
            print("{}.csv already exists".format(ticker))

get_data(sp_500_tech_companies_tickers)
```

Now with the data exracted from internet and stored in csv files we can prepare it for the exploriatory data investigation.

Firstly csv files must be read and converted to a pandas DataFrame object. Secondly the specific columns we want from the data should be extracted. Lastly the final form the DataFrame should be restored in a varible (by convention “df” is used).

Compile_data (column) function is defined below. Its input column should be in string format. Compile_data() function creates a new csv file from the stock data with the only intended column names.In this case columns are “Adjusted Close Price” and “Volume”.

```
def compile_data(column):

    """ Specify the column name you want to merge your data in str type"""

    main_df = pd.DataFrame()

    for count,ticker in enumerate(sp_500_tech_companies_tickers):

        df = pd.read_csv("stock_data/{}.csv".format(ticker))

        column_names = list(df.columns)
        column_names.remove(column)
        column_names.remove("Date")

        df.set_index("Date", inplace=True)

        df.rename(columns={column : ticker}, inplace=True)
        df.drop(column_names, 1, inplace=True)

        if main_df.empty :
            main_df = df

        else:
            main_df = main_df.join(df, how ="outer")

    main_df.to_csv("stock_data/main_df_{}.csv".format(column))

    print(main_df.head(10))
    print("stock_data/main_df_{}.csv".format(column),"has created")
```

```
compile_data("Volume")
compile_data("Adj Close")

df_adj = pd.read_csv("stock_data/main_df_Adj Close.csv")
df_vol = pd.read_csv("stock_data/main_df_Volume.csv")
df_adj.head()
```

Table 4.1: Adj close prices of major S&P 500 technology companies

	Date	MSFT	AAPL	AMZN	GOOGL	INTC	ORCL	NVDA	TSLA
0	2005-01-03	19.6	3.0	44.5	101.5	15.3	11.8	7.3	nan
1	2005-01-04	19.7	3.0	42.1	97.3	15.0	11.5	6.9	nan
2	2005-01-05	19.6	3.1	41.8	96.9	14.9	11.6	7.0	nan
3	2005-01-06	19.6	3.1	41.0	94.4	14.9	11.7	6.9	nan
4	2005-01-07	19.6	3.3	42.3	97.0	15.1	11.8	6.8	nan

There are some nan values in TSLA column. It should be investigated and droped out if there is not enough data or data is corrupted. isnul() function is used to check the nan values.

```
sns.heatmap(df_adj.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2150f1d9470>
```

Figure 4.1: Nan values heatmap

From the heat map we see only “TSLA” has missing values and it is because Tesla Motors was not public untill 2010.

4.2 Compiling and investigating data

Now that we are ready to explore data starting with describe() function would be appropriate to have a common view over the values.

```
df_adj.describe()
```

Table 4.2: Describe() function applied on adjusted close values

	MSFT	AAPL	AMZN	GOOGL	INTC	ORCL	NVDA	TSLA
count	3523.0	3523.0	3523.0	3523.0	3523.0	3523.0	3523.0	2142.0
mean	35.7	60.6	376.4	456.8	22.5	28.0	43.2	172.5
std	23.3	56.1	444.1	296.3	10.1	11.5	64.8	115.1
min	11.9	3.0	26.1	87.6	8.8	10.2	5.4	15.8
25%	20.7	13.0	76.5	238.2	15.4	17.7	12.1	32.9
50%	24.2	39.4	205.1	310.0	18.2	28.0	17.1	203.3
75%	43.1	101.8	439.0	606.3	29.2	37.6	27.9	256.6
max	114.6	230.3	2039.5	1285.5	55.7	51.9	288.8	385.0

Ploting the table.

```
plt.figure(figsize=(15,10))
for ticker in sp_500_tech_companies_tickers:
    plt.plot(df_adj[ticker], label=ticker)
    plt.title("Top Tech Companies Adjusted Close")
    plt.legend(loc=2, prop={"size":18})
    plt.ylabel("Adj Close")

    plt.show
```

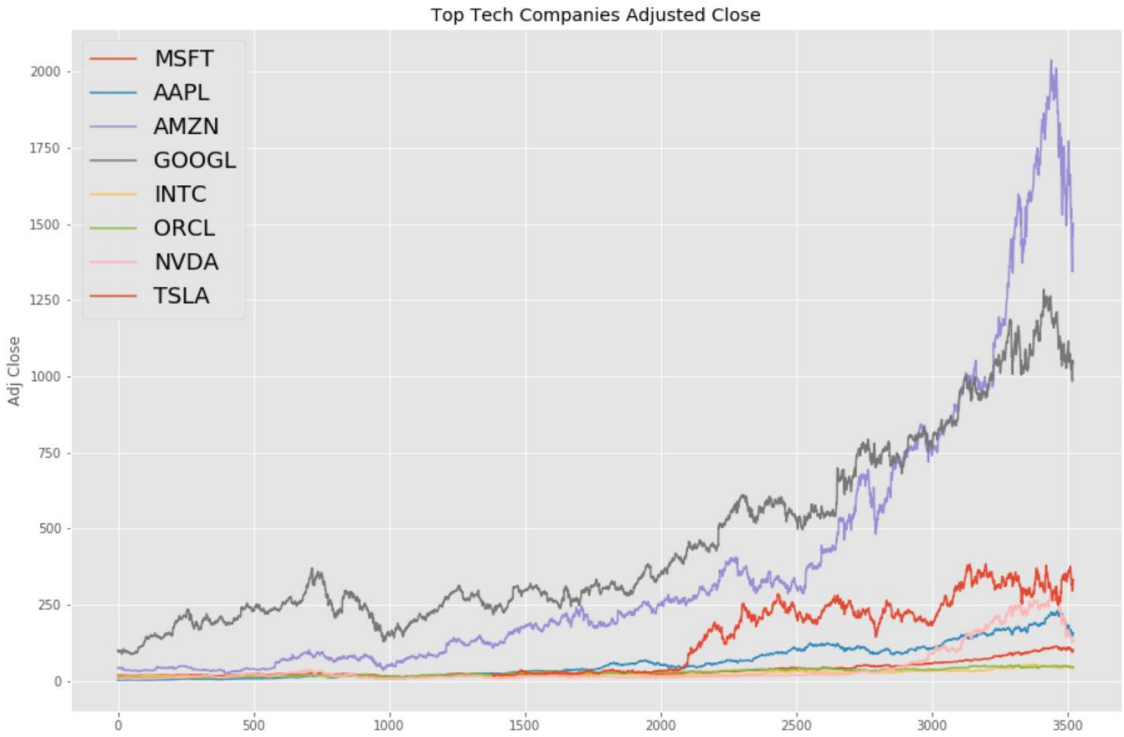

Figure 4.2: Top tech companies adjusted close prices

Now checking Volume values so that we might start our ideal-gas economic modeling.

```
i = 0
plt.figure(figsize=(40,20))
fig, axs = plt.subplots(9, 1, sharex=True, figsize=(15,15))
for ticker in sp_500_tech_companies_tickers:
    i = i + 1
    fig.subplots_adjust(hspace=0.05)

    axs[i].plot(df_vol[ticker], label=ticker)
    axs[i].set_ylabel(ticker)
plt.show()
```

Using for loops to plot multiple graphs as subplots in the figure.

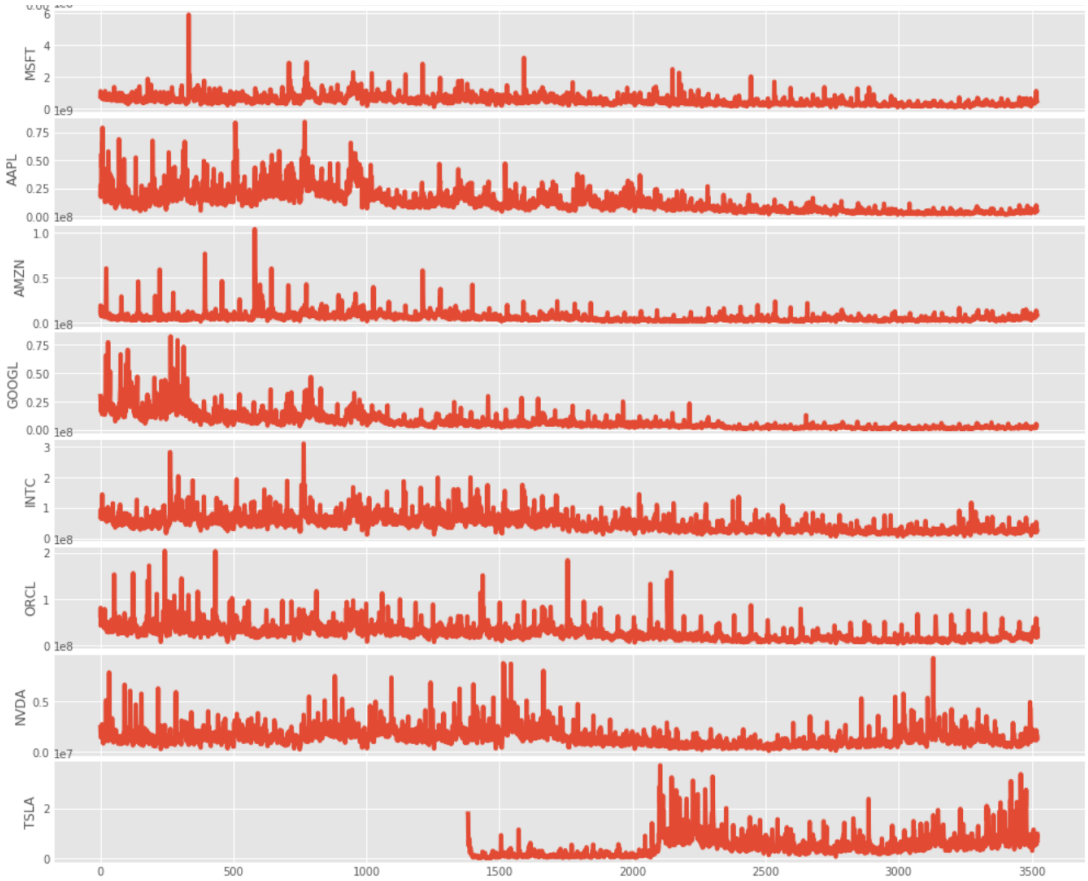

Figure 4.3: Top tech companies volume values

Now that we have the overall idea of individual compaines values. Ideal gas like economic model should be constructed by comparing volume and price realations of each company.Starting with “AAPL” ticker which corresponds to Apple’s stock data.

```
plt.figure(figsize=(40,20))
plt.xlabel("adj close", fontsize=40)
plt.ylabel("Volume", fontsize=40)
plt.title("AAPL P - V", fontsize=40)
plt.scatter(df_vol["AAPL"], df_adj["AAPL"])
```

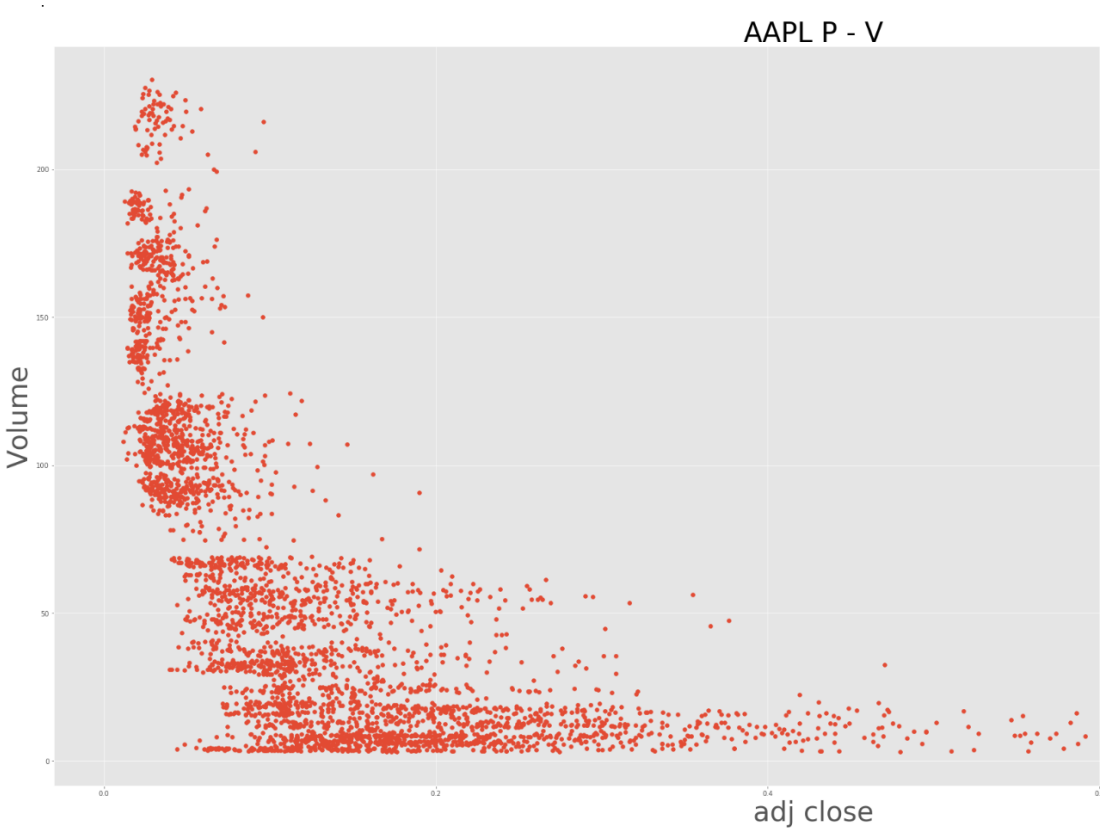

Figure 4.4: AAPL Adj close to Volume graph

This graph shows exactly what we expected from the Figure 2.1 P-V graph of an ideal-gas.

Now to prove that it is not just a coincidence occured in Apples stock prices other stock data must be check and graphed.

```
i = 0
plt.figure(figsize=(60,50))
fig, axs = plt.subplots(9, 1, sharex=True, figsize=(15,15))
plt.xlabel("Volume",size=25)

for ticker in sp_500_tech_companies_tickers:
    i = i + 1
    fig.subplots_adjust(hspace=0.05)
    axs[i].scatter(df_vol[ticker], df_adj[ticker], label=ticker)
    axs[i].set_ylabel(ticker)
plt.show()
```

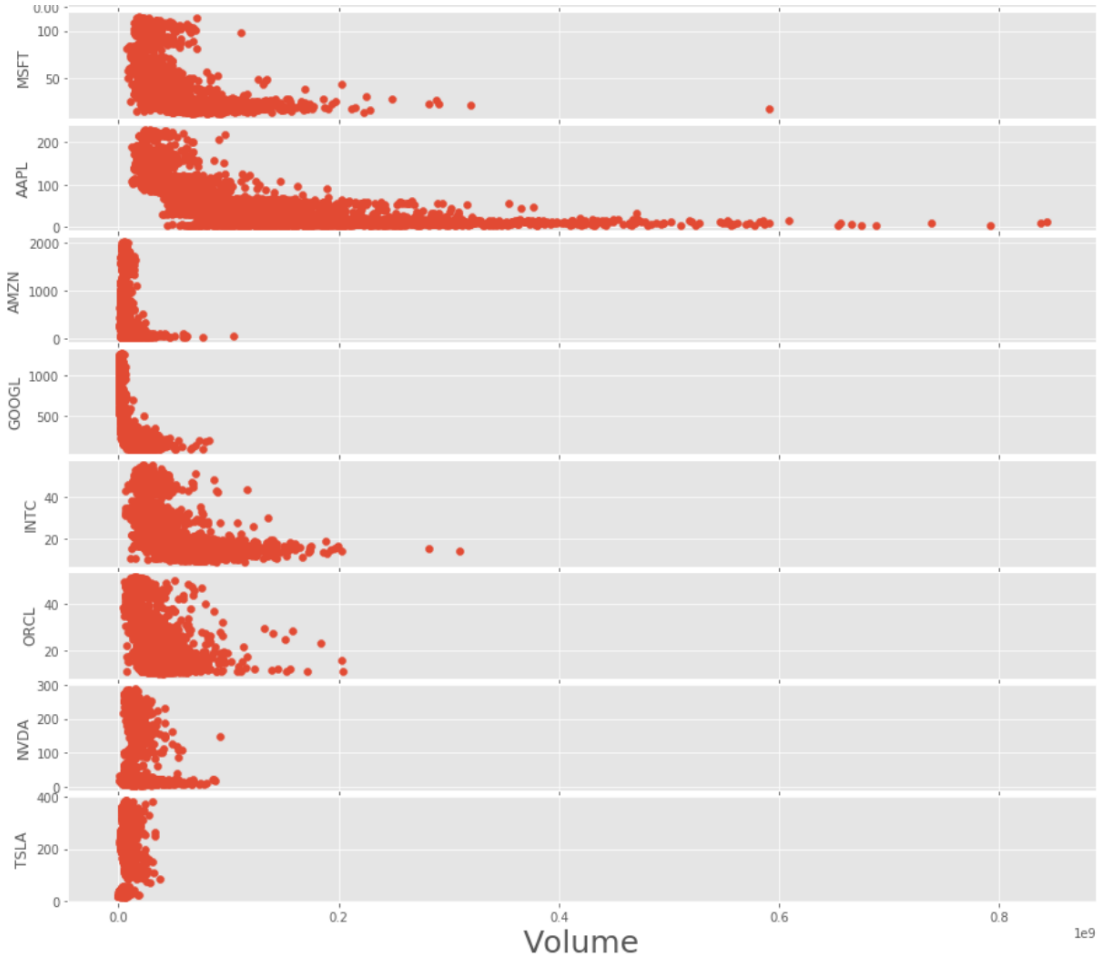

Figure 4.5: S&P 500 Tech Companies Adj close to Volume graph

From the overall view one can tell that the ideal gas realtion of P – V continues. However Tesla Motors shows exeptions and not viable to keep in this analysis since it differs in structre with other big companies and has missing values within this date range.

Exploratory analysis shows that the ideal gas model of econmics is promising. Fitting some models will help understanding the behavior of P – V in stock data.

Starting with linear regression and polynomial regression to understand the structre of the data.

5.MACHINE LEARNING

Machine learning is a data analytics method that humans and animals use naturally, depends learning from the past experiences and based on algorithms, mathematics and statistics. Applications of methods and algorithms in ML are investigated over three categories.

1.Supervised Learning

2.Unsupervised Learning

3.Reinforced Learning

In our case supervised learning method will be used. ML algorithms will learn form the labeled train data to predict the test data to see the score and error for further predictions. [9]

5.1 Linear Regression

Defining a function **linear_reg_sckit()** to make further aplications easier. Precprocesing.scale() is a built-in function of scikit-learn and scales the values around zero so that machine learning tools are applied easier and more accurate.

model_selection.train_test_split() splits the train and test data by the chosen **test_size** variable.

After each graph of stock data mean squarred error is calculated to check the compatibiliy with the particular stock.

```
def linear_reg_sckit(column, test_size = 0.25):

    df_adj[column] = preprocessing.scale(df_adj[column])
    df_vol[column] = preprocessing.scale(df_vol[column])

    X = df_vol[column].dropna()
    y = df_adj[column].dropna()

    X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, y,test_size=test_size)

    model = LinearRegression()
    model.fit(X_train.values.reshape(-1, 1), Y_train)

    score = model.score(X_test.values.reshape(-1, 1),Y_test)

    predictions = model.predict(X_test.values.reshape(-1, 1))

    mse = metrics.mean_squared_error(Y_test, predictions)

    plt.figure(figsize=((10,10)))
    plt.xlabel("Volume",size=25)
    plt.ylabel("Adjusted Close",size=25)
    plt.plot(X_test, predictions, color="b")
    plt.title("Linear regression {}".format(column), size=30)
    plt.text(1,3,"mean squarred error: {}".format(mse),fontsize=15)


    plt.scatter(X_train, Y_train,color ="g")
    plt.scatter(X_test, Y_test)
    return "mean squarred error:", mse
```

```
for column in sp_500_tech_companies_tickers:
    linear_reg_sckit(column, test_size = 0.2)
```

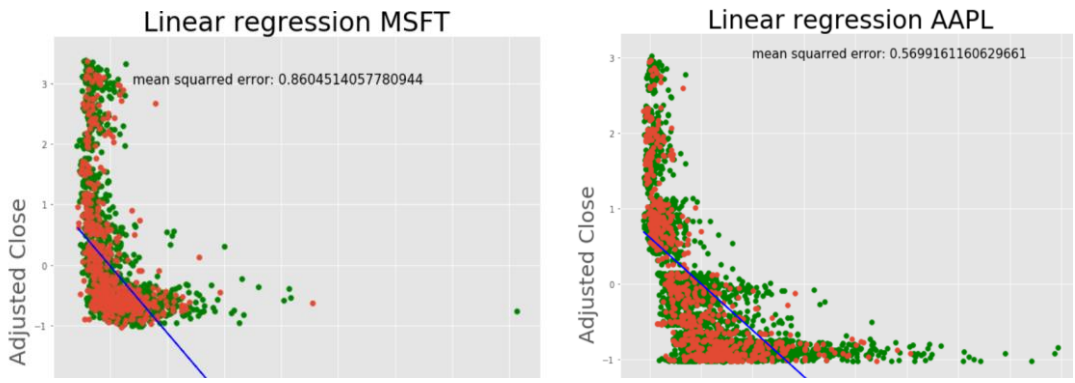

Figure 4.6: S&P 500 Tech Companies P – V Linear Regression

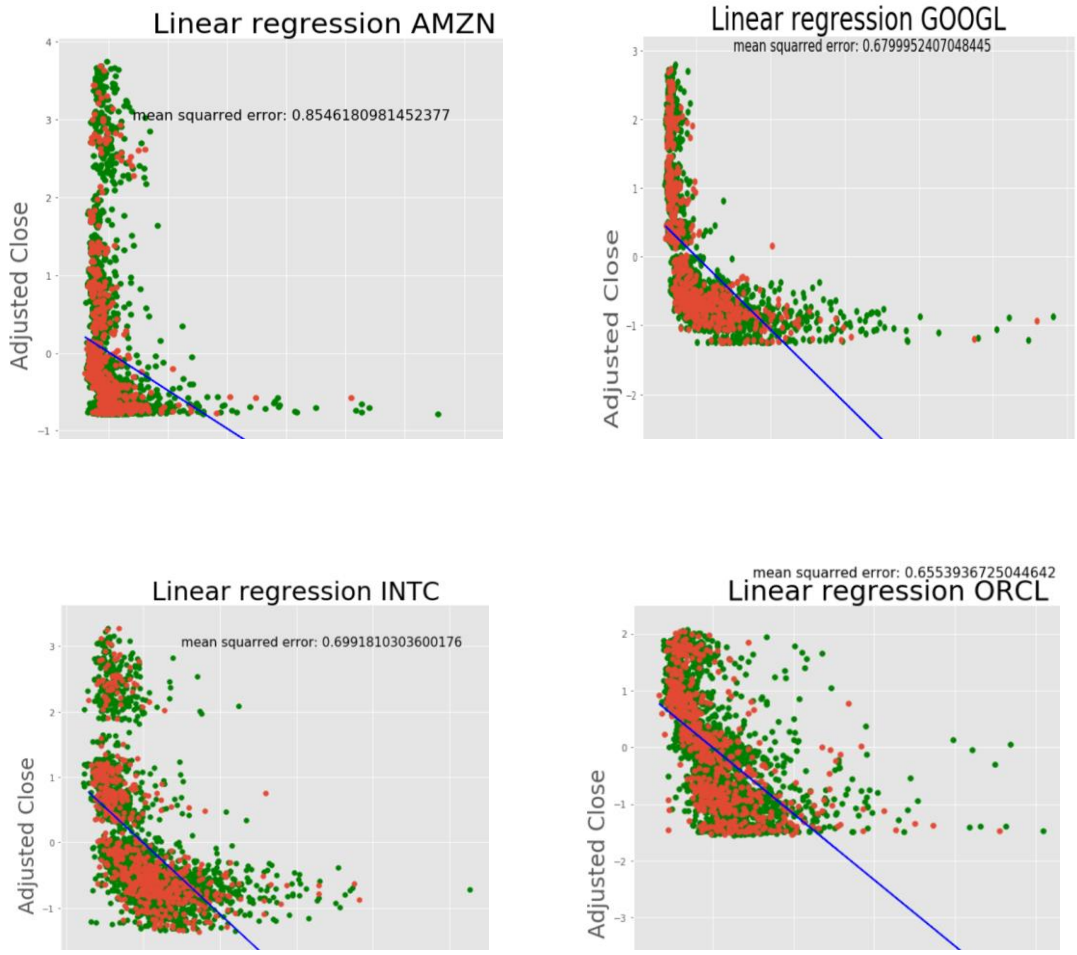

Figure 4.6: S&P 500 Tech Companies P – V Linear Regression

As we see from the Figure 4.6 linear regression is not what we are looking for. Ideal gas like pressure volume behaviour should act like a 2nd degree polinomial. Using PolynomialFeatures function from scikit-learn preprocessing module we are trying to achive an ideal gas P – V curve.

5.2 Polynomial Regression

```
def poly_sckit(column, test_size = 0.25, deg=2):

    df_adj[column] = preprocessing.scale(df_adj[column])
    df_vol[column] = preprocessing.scale(df_vol[column])

    X = df_vol[column].dropna()
    y = df_adj[column].dropna()

    X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, y,test_size=test_size)

    poly_reg = preprocessing.PolynomialFeatures(degree=deg)

    X_poly = poly_reg.fit_transform(X_train.values.reshape(-1, 1))

    pol_reg = LinearRegression()

    pol_reg.fit(X_poly, Y_train)



    #a =pol_reg.predict(poly_reg.fit_transform([[5.5]]))
    pred = pol_reg.predict(poly_reg.fit_transform(X_test.values.reshape(-1, 1)))

    mse = metrics.mean_squared_error(Y_test, pred)

    plt.figure(figsize=((10,10)))
    plt.xlabel("Volume",size=18)
    plt.ylabel("Adjusted Close",size=18)
    plt.title("Polynomial regression {}".format(column), size=20)
    plt.text(1,3,"mean squarred error: {}".format(mse),fontsize=15)
    plt.scatter(X_train, Y_train,color ="g")
    plt.scatter(X_test, Y_test)
    plt.scatter(X_test, pred, color="b")
```

As it is know from the source code of the scikit-learn PolynomiaFeatures. This function produces a new matrix including all polynomial combinations of the features equal to the specified degree. If the sample is two dimensional and of the form (x, y), the degree-2 polynomial features are (1, x, y, a^2, x.y, y^2)

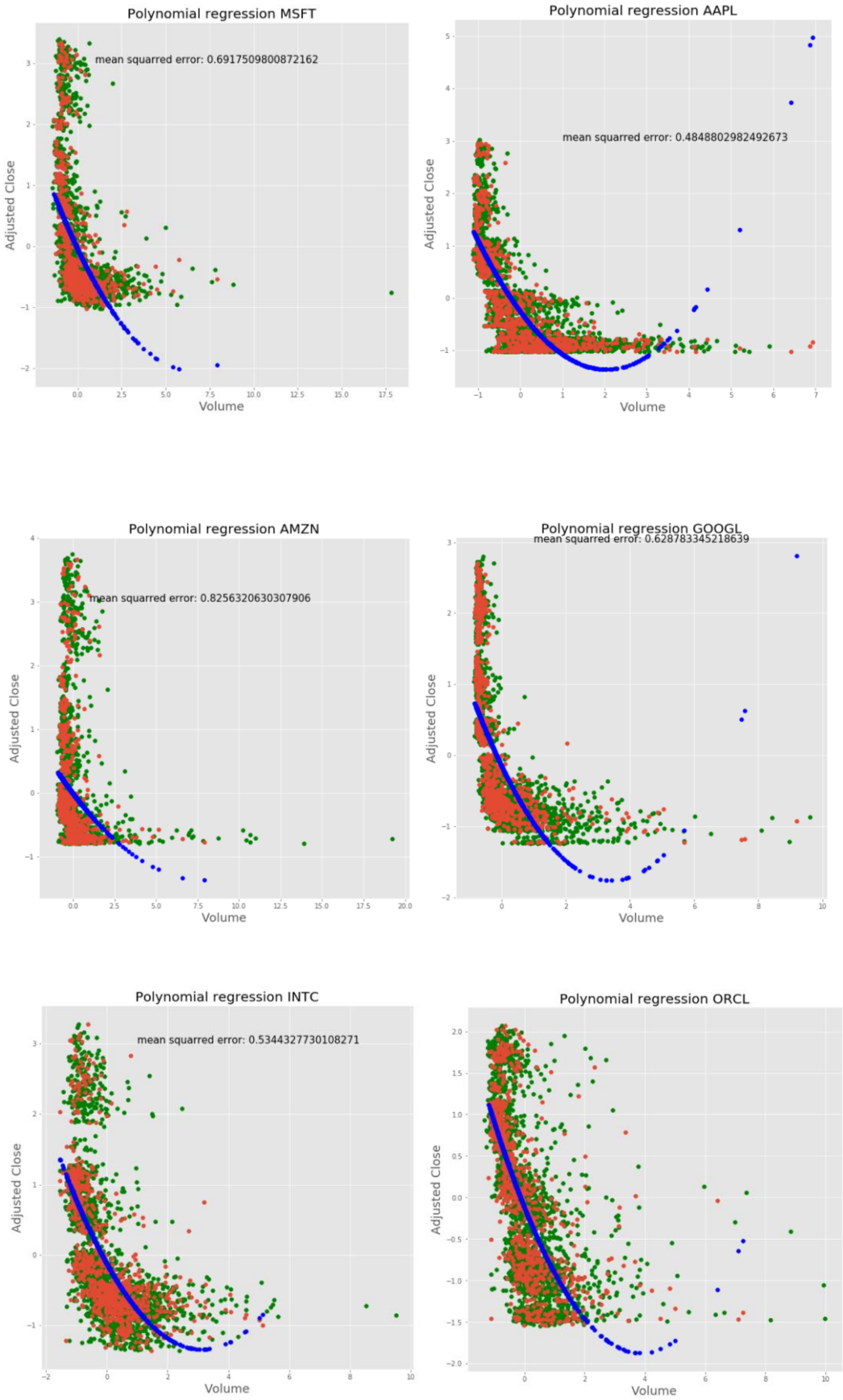

Figure 4.7: S&P 500 Tech Companies P – V Polynomial Regression

These results give us a clear judgement about the price-volume behaviour of chosen stock market. As expected the ideal gas behaviour is observed.

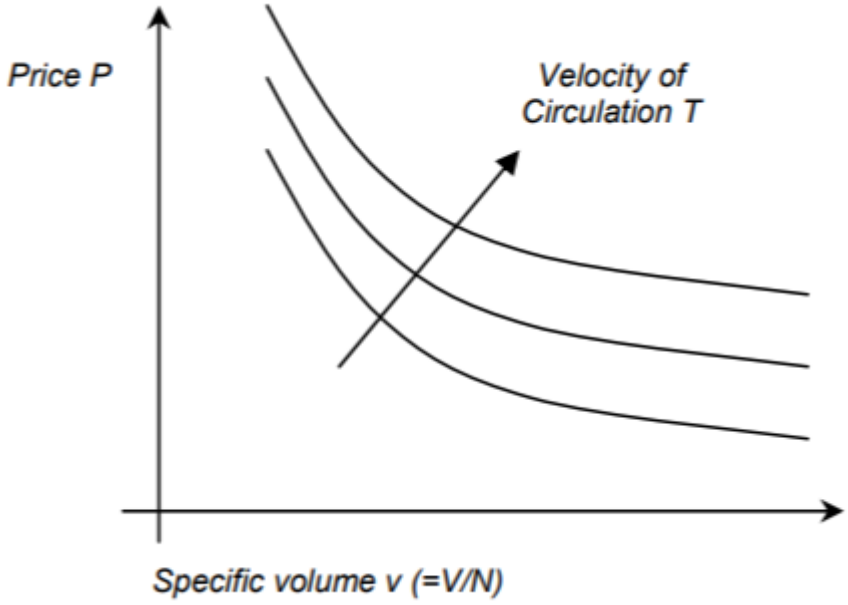

Figure 4.8: Ideal gas economic model Price – Volume graph [5]

Now that the pressure volume realtionship is observed adding the temperature is the last step to construct our ideal gas model to appl LSTM later on.

How much a market can provide or the customers are capable of buying can be measured by the internal value Ue = P.V. The internal value represents the internal energy of the ideal-gas in this case. Since the internal energy must be proportional to the temperature, the internal value must be in accordance with the purchase power of the nation or simply the GNP. [2]

Scaling and adding GNP Series to the DataFrame and comparing P V and T in the graph;

```
# very intresting when T rises P*V aslo does which strengthens our ideal gas aprroach
df_adj_r = pd.read_csv("stock_data/main_df_Adj Close.csv")
df_vol_r = pd.read_csv("stock_data/main_df_Volume.csv")
pv = df_vol_r["AAPL"] * df_adj_r["AAPL"]


plt.figure(figsize=(20,10))
plt.title("P * V increasement with T", fontsize=20)
plt.xlabel("time", fontsize=30)
plt.ylabel("p * v ", fontsize=30)
ax = plt.plot(pv)
ax1 =plt.plot(gnp["gnp"]*10**10)

plt.annotate('Spikes with T increase', xy=(500,1),
             arrowprops=dict(facecolor='g', shrink=0.5),
             )
```

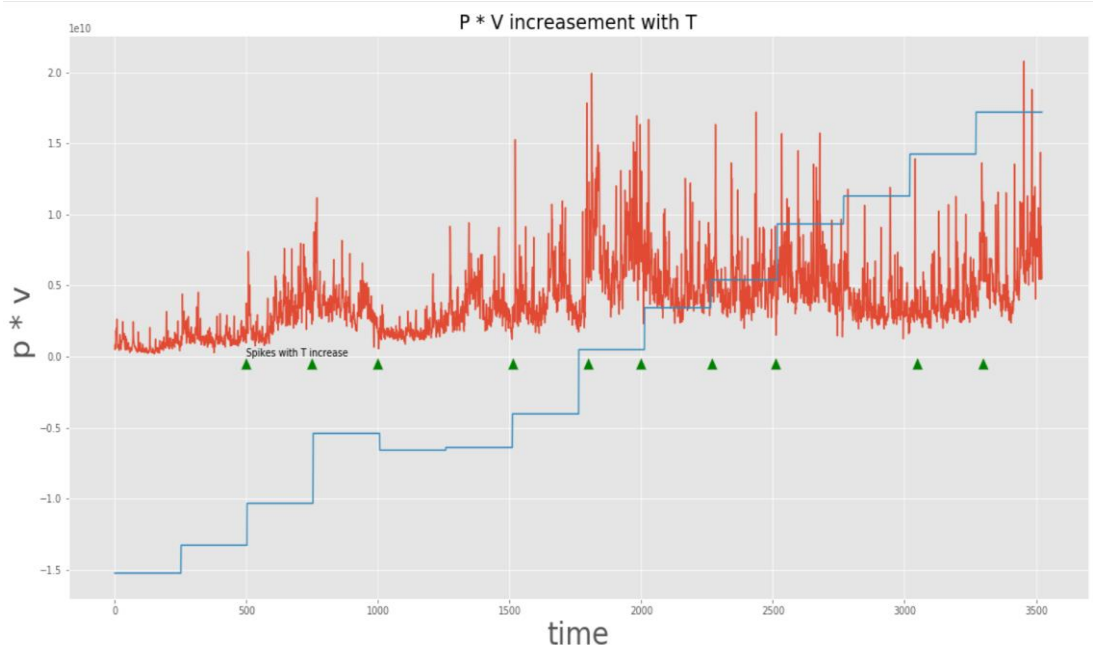

Figure 4.9: P.V increasement with T

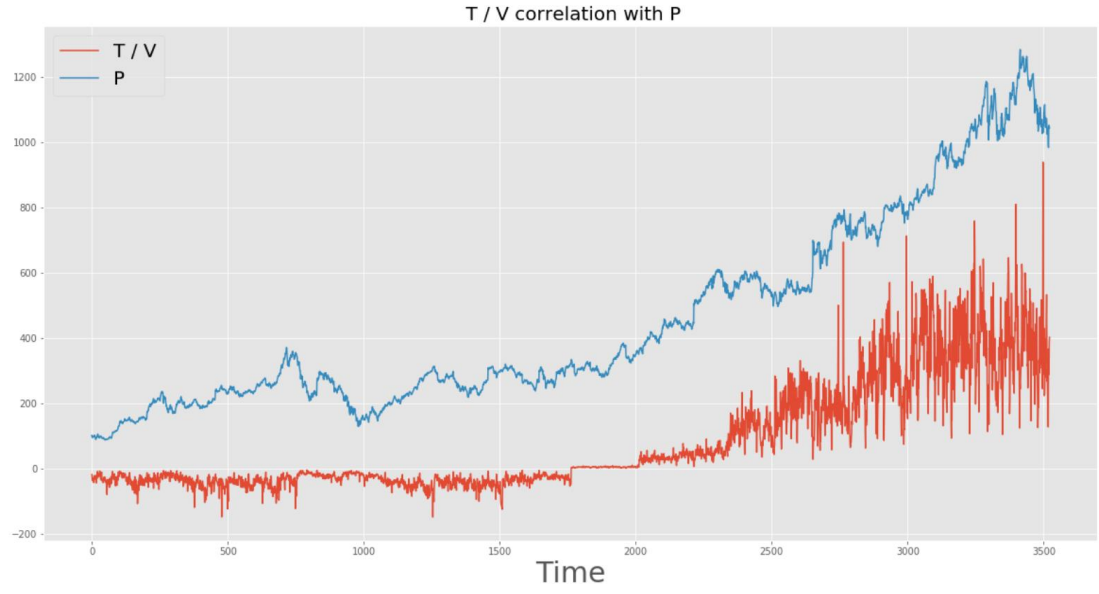

Figure 4.10: T/V and P graph

5.3 LSTM

We are constructing the lstm model to observe the "GOOGL" data. First scaling must be done. Using the MinMacScaler of scikit-learn would give better results here instead of the scaler() function.

```
cl = df_adj_r["GOOGL"]
cl_t_v =  gnp["gnp"]*9**9 / df_vol_r["GOOGL"]

scl = preprocessing.MinMaxScaler()
#Scale the data
cl = cl.values.reshape(cl.shape[0],1)
cl = scl.fit_transform(cl)

cl_t_v = cl_t_v.values.reshape(cl_t_v.shape[0],1)
cl_t_v = scl.fit_transform(cl_t_v)
```

Lstm needs a "look back" value to know the date interval of predictions. We process the data with by defining a function. Our look back value will be 7 days.

```
def processData(data,lb):
    X,Y = [],[]
    for i in range(len(data)-lb-1):
        X.append(data[i:(i+lb),0])
        Y.append(data[(i+lb),0])
    return np.array(X),np.array(Y)

X,y = processData(cl,7)
X_train,X_test = X[:int(X.shape[0]*0.80)],X[int(X.shape[0]*0.80):]
y_train,y_test = y[:int(y.shape[0]*0.80)],y[int(y.shape[0]*0.80):]


print(X_train.shape[0])
print(X_test.shape[0])
print(y_train.shape[0])
print(y_test.shape[0])
```

```
2812
703
2812
703
```

The main idea here is we are trying to predict the feature Price value by training our data with Volume and Temperature values. This makes a great difference between predicting the data from the Long and short term changed and predicting it from an entire different set of variables (T and V). This model will give us the opputurnity to test the ideal gas approach by creating a “ideal gas idicator”.

```
def lstm(column, epochs=5):

    scl = preprocessing.MinMaxScaler()

    a = df_adj_r[column]
    b = gnp["gnp"]*9**9 / df_vol_r[column]

    b = b.values.reshape(b.shape[0],1)
    b = scl.fit_transform(b)

    a = a.values.reshape(a.shape[0],1)
    a = scl.fit_transform(a)
    i = 0
    for val in [a, b]:
        i = i + 1
        X,y = processData(val,7)
        X_train,X_test = X[:int(X.shape[0]*0.80)],X[int(X.shape[0]*0.80):]
        y_train,y_test = y[:int(y.shape[0]*0.80)],y[int(y.shape[0]*0.80):]

        model = Sequential()
        model.add(LSTM(256,input_shape=(7,1)))

        model.add(Dense(1))
        model.compile(optimizer='adam',loss='mse')

        X_train = X_train.reshape((X_train.shape[0],X_train.shape[1],1))
        X_test = X_test.reshape((X_test.shape[0],X_test.shape[1],1))

        history = model.fit(X_train,y_train,epochs=epochs,validation_data=(X_test,y_test),shuffle=False)

        plt.figure(figsize=(20,10))
        plt.plot(history.history['loss'],label = "loss")
        plt.plot(history.history['val_loss'], label="val loss")
        plt.legend(loc=2, prop={"size":20})

        plt.figure(figsize=(20,10))
        Xt = model.predict(X_test)
        plt.plot(scl.inverse_transform(y_test.reshape(-1,1)), label="test_set")
        plt.plot(scl.inverse_transform(Xt), label="prediction")
        plt.legend(loc=2, prop={"size":20})
```

```
lstm("AAPL", epochs=30)
```

....

```
.
2812/2812 [==============================] - 1s 495us/step - loss: 1.5031e-04 - val_loss: 0.0012
Epoch 28/30
2812/2812 [==============================] - 1s 503us/step - loss: 1.3519e-04 - val_loss: 0.0011
Epoch 29/30
2812/2812 [==============================] - 1s 521us/step - loss: 1.1123e-04 - val_loss: 0.0011
Epoch 30/30
  32/2812 [..............................] - ETA: 1s - loss: 3.5137e-0
```

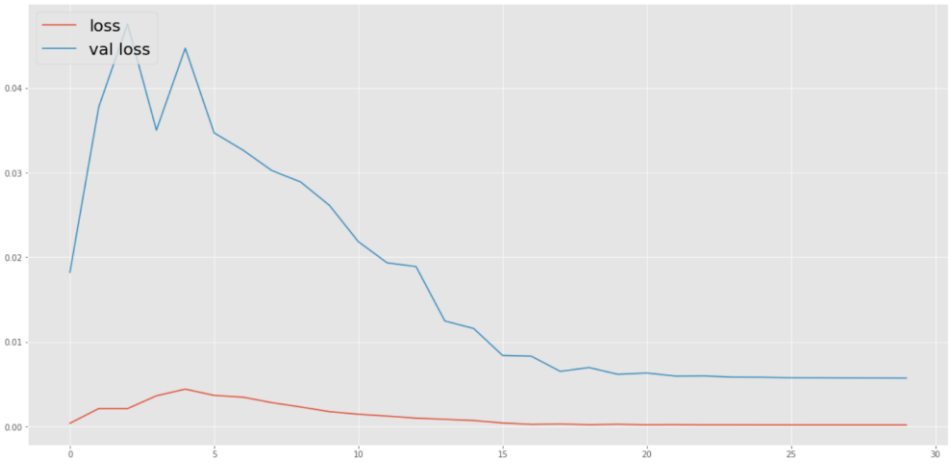

Figure 5.1: Lstm loss and value loss

Figure 5.2: Lstm predictions over test data

Now checking the loss and value loss so that we have an idea of the fitting.If the loss is decreasing towards the val loss it means our lstm model did not under or over fit. Eventhough it was not our purpose to predict the price with lstm from the older price data it looks like it did quite a good job.

5.4 An indicator based on LSTM

Finally we will create an indicator that predicts the Price of a stock from its previous volume and GNP relations.Epoch number taken here will be 30, it could be increased proportionally to your process speed.

Shuffle method of the fit function is marked as False since we do not want to shuffle our data because it is a stock market prediction and volume of companies vary in time.

```
epo=30

scl = preprocessing.MinMaxScaler()

h = df_adj_r[column]

g = gnp["gnp"]*9**9 / df_vol_r[column]

g = g.values.reshape(g.shape[0],1)
g = scl.fit_transform(g)

h = h.values.reshape(h.shape[0],1)
h = scl.fit_transform(h)

q,w = processData(h,7)

X_trainq,X_testq = q[:int(q.shape[0]*0.80)],q[int(q.shape[0]*0.80):]
y_trainq,y_testq = w[:int(w.shape[0]*0.80)],w[int(w.shape[0]*0.80):]

model = Sequential()
model.add(LSTM(256,input_shape=(7,1)))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mse')

X_trainq = X_trainq.reshape((X_trainq.shape[0],X_trainq.shape[1],1))
X_testq = X_testq.reshape((X_testq.shape[0],X_testq.shape[1],1))

history = model.fit(X_trainq,y_trainq,epochs=epo,validation_data=(X_testq,y_testq),shuffle=False)

j,k = processData(g,7)

X_train_j,X_test_j = j[:int(q.shape[0]*0.80)],j[int(q.shape[0]*0.80):]
y_train_j,y_test_j = k[:int(w.shape[0]*0.80)],k[int(w.shape[0]*0.80):]

model = Sequential()
model.add(LSTM(256,input_shape=(7,1)))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mse')

X_train_j = X_train_j.reshape((X_train_j.shape[0],X_train_j.shape[1],1))
X_test_j = X_test_j.reshape((X_test_j.shape[0],X_test_j.shape[1],1))

history = model.fit(X_train_j,y_train_j,epochs=epo,validation_data=(X_test_j,y_test_j),shuffle=False)
```

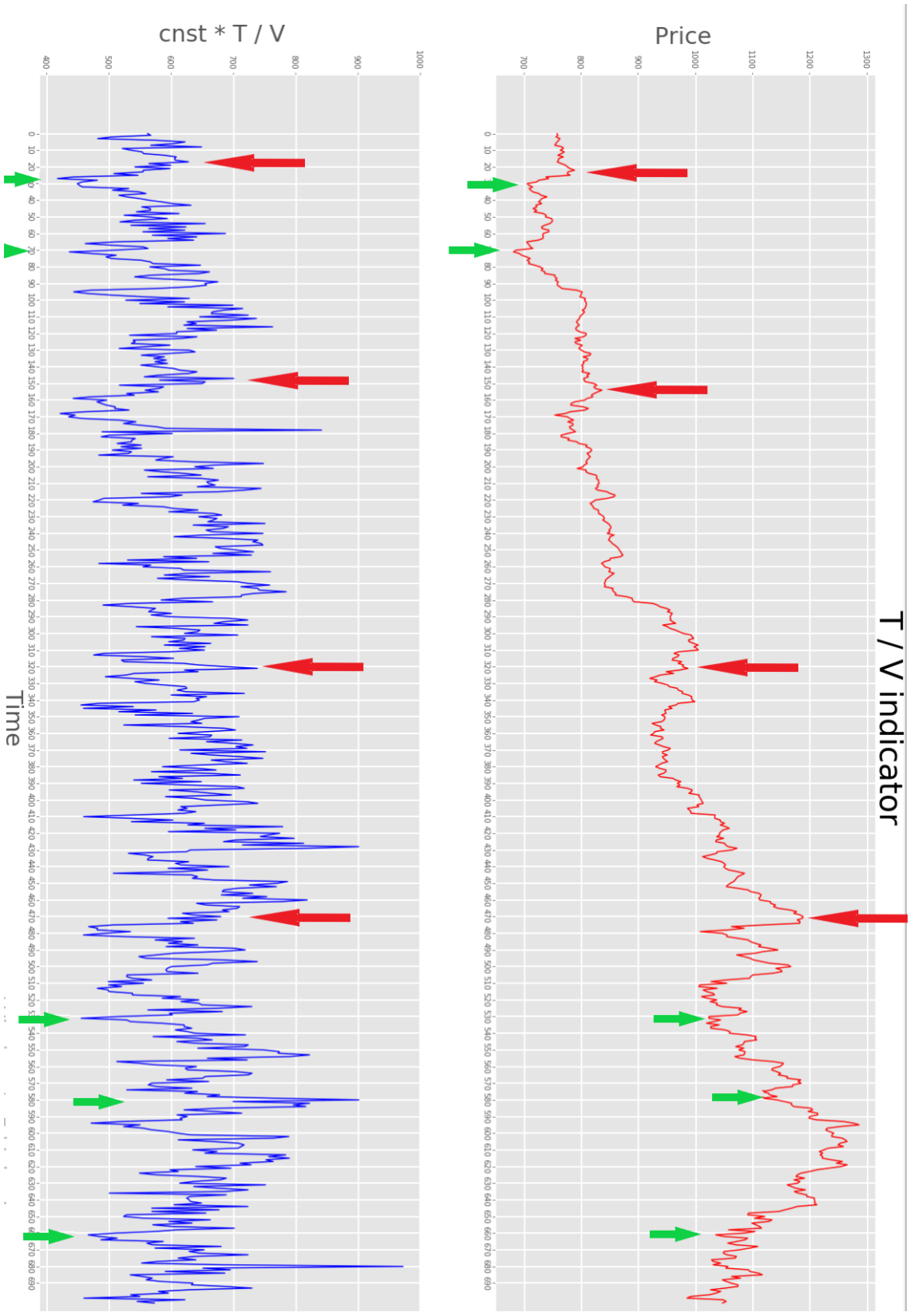

Figure 5.3: Lstm ideal gas predictor

6.CONCLUSION

In this work, stock market movements and ideal-gas behavior is compared and atempted to design an indicator to help stock market traders and to give them a different insight. The project procceded with a jupyter notebook so that the codes used are easy to catch up with and gives the possibility for implementations.

What makes the price predictions in this work different is the train data used is actually different than the target variables. Using Volume / Temperature which are in this case stock volume and GNP to predict stock Pressure (Price) in this ideal gas approach gave very promising results. In the Figure 4.13 lstm indicator signals for the some of the very important trend changes correctly for the “GOOGL” data.

For any further improvement a real gas model can be appleid with lstm networks and the analysis could be made on different tickers in stock market or maybe even in some other countries stock exchanges.

REFERENCES

[1] Bikas K. Chakrabarti and Arnab Chatterjee., " Ideal Gas-Like Distributions in Economics:Effects of Saving Propensity", p.283, 2003.

[2] Tinko Eftimov.," Ideal Gas Laws In Thermoeconomics And Financial Bubble Formation", Journal of the Technical University Sofia, 2013.

[3] Pikler, A. G., 'Optimum allocation in econometrics & physics', Weltwirtschaftliches Archiv. ,1954

[4] Kevin M. Tenny., Jeffrey S. Cooper., "Ideal Gas Behavior.", 2019.

[5] John Bryant., "A Thermodynamic Theory of Economics", 2007.

[6] Jake VanderPlas., "Python Data Science Handbook", Preface, 2016.

[7] "Keras LSTM tutorial – How to easily build a powerful deep learning language model "., Retrieved from https://adventuresinmachinelearning.com/keras-lstm-tutorial/, (2018).

[8] Nahua Kang., "Multi-Layer Neural Networks with Sigmoid Function— Deep Learning "., (2017).

[9] Alpaydin, E.,." Introduction to machine learning", MIT press, 2009.