# Data preparation:

In this work, we will study the daily data of stock market returns of APPLE since 1990. This data base has the following structure:

- It is a dictionary of 500 data frame.
- Each data frame contains some information about a company.
- These information are :
  - Open
  - High
  - Low
  - Volume
  - Close

We will make a prediction on past price and volume dynamics. Therefore, we used four families of technical analytics.
Let's start with the **Log returns:**

$$R_n = 10000 \times \log \frac{P_t}{P_{t-n}}$$

This method gives many advantages. First, log-normality: if we assume that prices are distributed log normally (which, in practice, may or may not be true for any given price series), then $log(1 + r_i)$ is conveniently normally distributed.

We used as well the technique of moving averages that smoothes the price data to form a trend following indicator. It doesn't predict price direction, but rather defines the current direction with a lag. Despite this lag, moving averages help smooth price and filter out the noise. Hence, we started by implementing the simple moving average through computing the average closing price over a specific number of periods:

$$\mathrm{SMA}(n) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{close}_{t-i}$$

$$\mathrm{SMA\%change}(n) = 100 \times \frac{\mathrm{close}_t - \mathrm{SMA}(n)_t}{\mathrm{SMA}(n)_t}$$

After that, we computed the independent variables referring to the technique of exponential moving average which reduces the lag by applying more weight to recent closing prices. The weighting applied to the most recent closing price depends on the number of periods in the moving average:

$$\mathrm{EMA}_{t,n} = \alpha \times \mathrm{close}_t + (1-\alpha) \times \mathrm{EMA}_{t-1}$$

$$\text{where } \alpha = \frac{2}{n+1} \text{ and } \mathrm{EMA}_{t,1} = \mathrm{close}_{t-n}.$$

Finally, we calculated the volume weighted exponential moving averages so that we introduce the volume dynamics' effect on formulating the independent variables of our model:

$$\mathrm{VWEMA}_{t,n} = \alpha \times \mathrm{close}_t \times \mathrm{Vol}_t + (1-\alpha) \times \mathrm{VWEMA}_{t-1}$$

The ensuing step was to define the target referred to as Y which consists in the (t + 2) stock market return. Both dependent and independent variables were thus put in a single data frame named *Data* to proceed for the data analysis phase.

## Data analysis:

**1)** In this question, I tried to describe the independent and the target variables. To realize this object, I used the following script:

```
#### Statistics description
Data.describe()
```

And the results were:

| | LogRet_1 | LogRet_2 | LogRet_3 | LogRet_4 | LogRet_5 |
|---|---|---|---|---|---|
| count | 6724.000000 | 6724.000000 | 6724.000000 | 6724.000000 | 6724.000000 |
| mean | 0.000701 | 0.001403 | 0.002100 | 0.002796 | 0.003496 |
| std | 0.029444 | 0.041437 | 0.050416 | 0.057871 | 0.065156 |
| min | -0.731247 | -0.791265 | -0.874534 | -0.817376 | -0.885802 |
| 25% | -0.013394 | -0.018787 | -0.022766 | -0.026735 | -0.029471 |
| 50% | 0.000000 | 0.001331 | 0.002365 | 0.003485 | 0.005235 |
| 75% | 0.014861 | 0.022146 | 0.028330 | 0.033600 | 0.039012 |
| max | 0.286892 | 0.390587 | 0.390587 | 0.419481 | 0.511540 |

| | LogRet_10 | LogRet_20 | SMAChange_3 | SMAChange_4 | SMAChange_5 |
|---|---|---|---|---|---|
| count | 6724.000000 | 6724.000000 | 6724.000000 | 6724.000000 | 6724.000000 |
| mean | 0.006995 | 0.013823 | 0.075119 | 0.115890 | 0.156888 |
| std | 0.093385 | 0.134885 | 2.170514 | 2.704795 | 3.147440 |
| min | -0.988155 | -1.109678 | -39.736710 | -42.658309 | -44.772112 |
| 25% | -0.040757 | -0.053890 | -0.979140 | -1.221411 | -1.398857 |
| 50% | 0.009107 | 0.017984 | 0.087155 | 0.141967 | 0.170089 |
| 75% | 0.060160 | 0.093604 | 1.122756 | 1.452018 | 1.741427 |
| max | 0.612939 | 0.789742 | 19.943092 | 23.823604 | 28.353746 |

| | ... | EMA_4 | EMA_5 | EMA_10 | EMA_20 |
|---|---|---|---|---|---|
| count | ... | 6724.000000 | 6724.000000 | 6724.000000 | 6724.000000 |
| mean | ... | 22.198112 | 22.190955 | 22.155010 | 22.083760 |
| std | ... | 33.935207 | 33.926559 | 33.883464 | 33.798758 |
| min | ... | 0.429300 | 0.429531 | 0.435999 | 0.456823 |
| 25% | ... | 1.233003 | 1.233462 | 1.231942 | 1.231760 |
| 50% | ... | 2.072919 | 2.072239 | 2.075679 | 2.061546 |
| 75% | ... | 26.840194 | 26.869253 | 26.671723 | 26.477388 |
| max | ... | 128.357673 | 128.182512 | 127.494855 | 126.312135 |

| | VWEMA_3 | VWEMA_4 | VWEMA_5 | VWEMA_10 | VWEMA_20 |
|---|---|---|---|---|---|
| count | 6724.000000 | 6724.000000 | 6724.000000 | 6724.000000 | 6724.000000 |
| mean | 22.199117 | 22.189005 | 22.179123 | 22.135087 | 22.055091 |
| std | 33.932272 | 33.918262 | 33.904856 | 33.847591 | 33.746171 |
| min | 0.426898 | 0.427360 | 0.428385 | 0.434642 | 0.457155 |
| 25% | 1.230870 | 1.231290 | 1.234024 | 1.234794 | 1.230934 |
| 50% | 2.075597 | 2.076495 | 2.075408 | 2.066826 | 2.061531 |
| 75% | 26.950957 | 26.848462 | 26.838695 | 26.669546 | 26.399959 |
| max | 128.388237 | 128.178032 | 127.998067 | 127.400103 | 126.306817 |

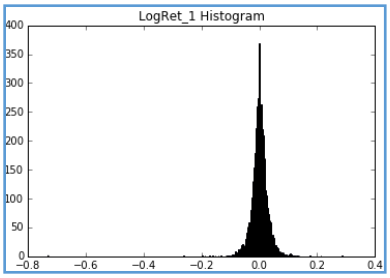
| | Target |
|---|---|
| count | 6724.000000 |
| mean | 0.001396 |
| std | 0.041434 |
| min | -0.791265 |
| 25% | -0.018787 |
| 50% | 0.001331 |
| 75% | 0.022123 |
| max | 0.390587 |

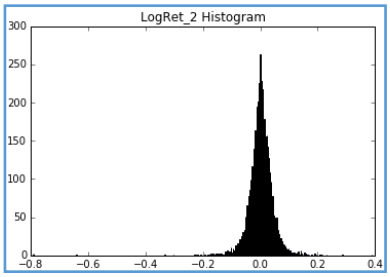
The independent and dependent variables show distinct descriptive statistic values: Some variables such as: *LogRet_i*, SMAChange_i present a small standard deviation which means that the values are close to the mean of the data set, on average.
Also, the EMA_i and the WEMA_i are characterized by a large standard deviation meaning that the values in the data set are farther away from the mean, on average.

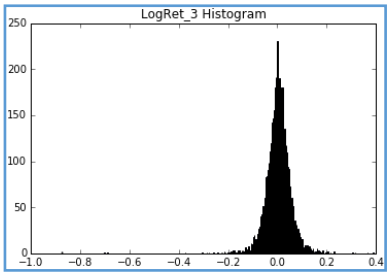
**2)** In this question, I tried to plot the histograms relative to our case study. So, I implemented the flowing Python code:

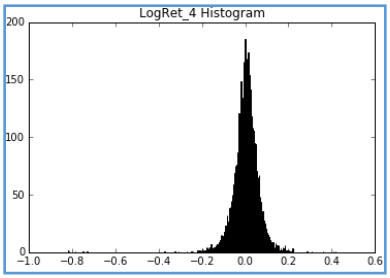
```
#### Histogram plots
i=1
for variable in ['LogRet_1', 'LogRet_2', 'LogRet_3', 'LogRet_4', 'LogRet_5',
                 'LogRet_10', 'LogRet_20',
                'SMAChange_3', 'SMAChange_4', 'SMAChange_5', 'SMAChange_10',
                'SMAChange_20', 'EMA_3', 'EMA_4', 'EMA_5', 'EMA_10', 'EMA_20',
              'VWEMA_3', 'VWEMA_4', 'VWEMA_5', 'VWEMA_10', 'VWEMA_20','Target']:
            plt.figure(str(i))
            plt.hist(Data[variable], bins=500)
            plt.title(variable+"Histogram")
            i=i+1
```

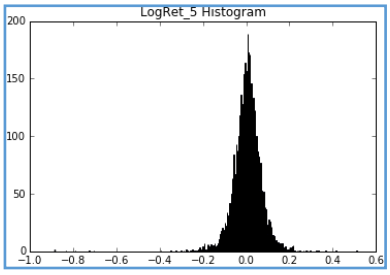
The output of my script was:

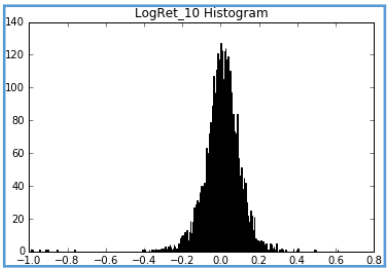


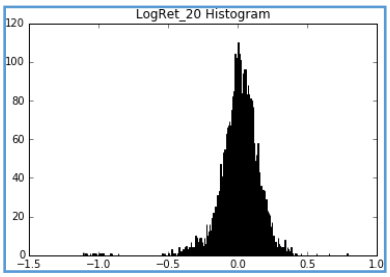



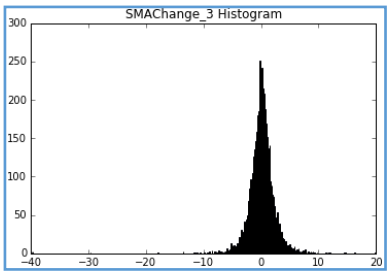



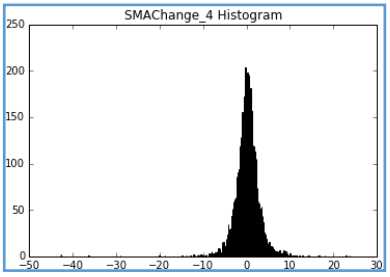



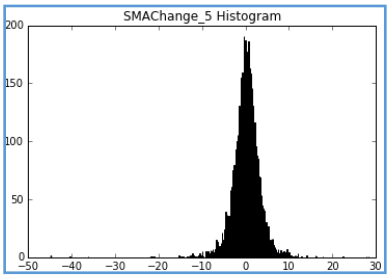



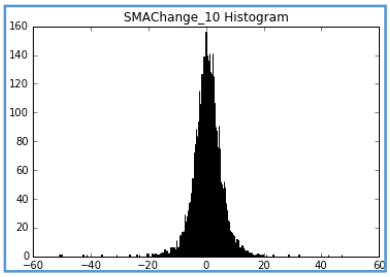



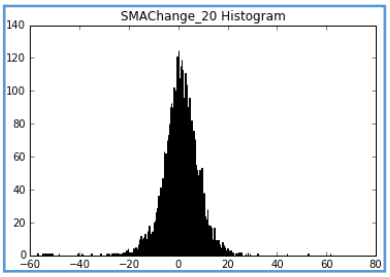



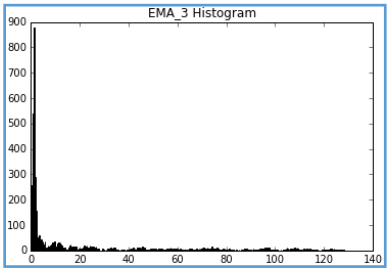



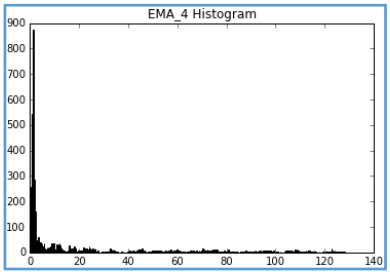



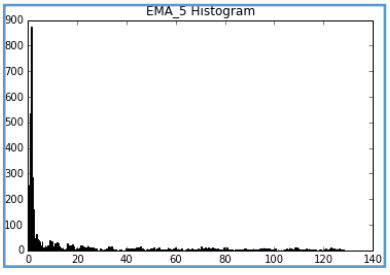



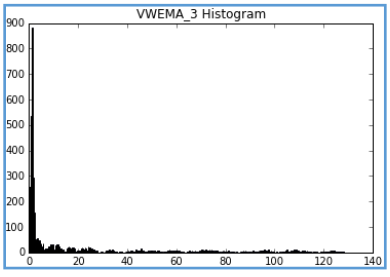



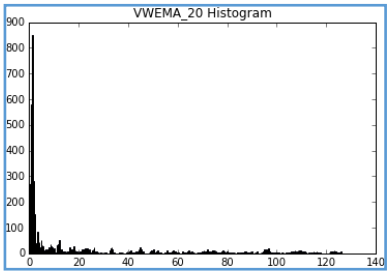



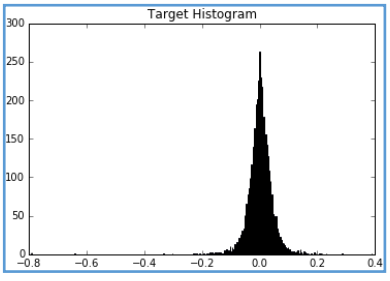



We can conclude that the variables are normally distributed. In fact, the center is located at the median of the distribution where about half of the observations are on either side.

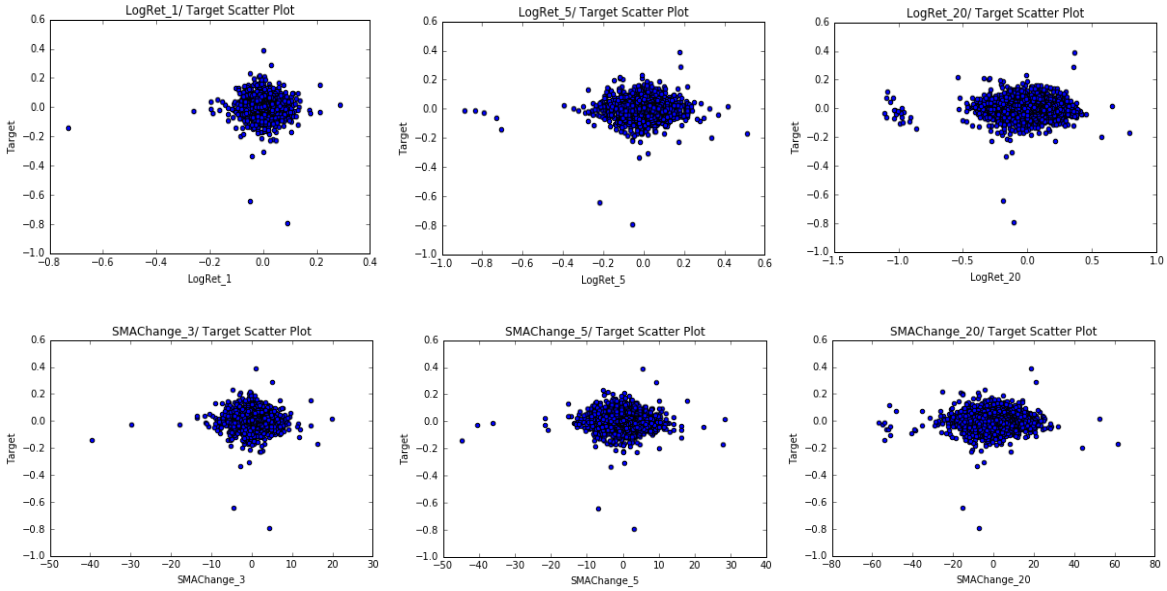
The height of each column indicates the frequency of observations. Besides, some figures are more variable (LogRet_20, SMAChage_20), so they have the greater spread.

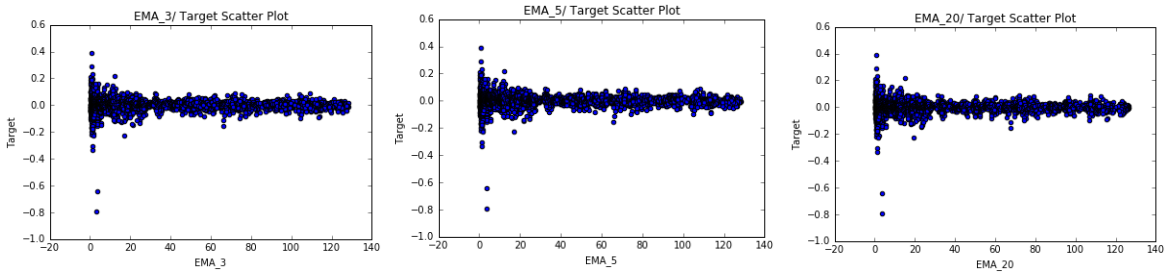
It is apparent as well that we have two types of histograms: Symmetric, unimodal and bell-shaped ones (LogRet_i, SAMChange_i and the target) and Skewed right ones (EMA_i and WEMA_i).

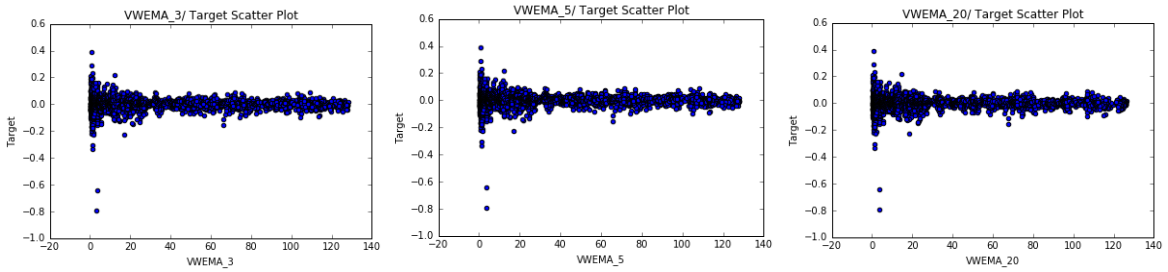
**3)** To show other kind of information about our data, I tried to generate the scatter-plot between the independent variables and the dependent variable Y. To realize this aim, I implemented the following script:

```
#### Scatter plots
i=1
for variable in ['LogRet_1', 'LogRet_2', 'LogRet_3', 'LogRet_4', 'LogRet_5',
                 'LogRet_10', 'LogRet_20',
                'SMAChange_3', 'SMAChange_4', 'SMAChange_5', 'SMAChange_10',
                'SMAChange_20', 'EMA_3', 'EMA_4', 'EMA_5', 'EMA_10', 'EMA_20',
                'VWEMA_3', 'VWEMA_4', 'VWEMA_5', 'VWEMA_10', 'VWEMA_20']:
            plt.figure(str(i))
            plt.xlabel(variable)
            plt.ylabel('Target')
            plt.scatter(Data[variable],Data['Target'])
            plt.title(variable+"/ Target Scatter Plot")
            i=i+1
```

And the results were:





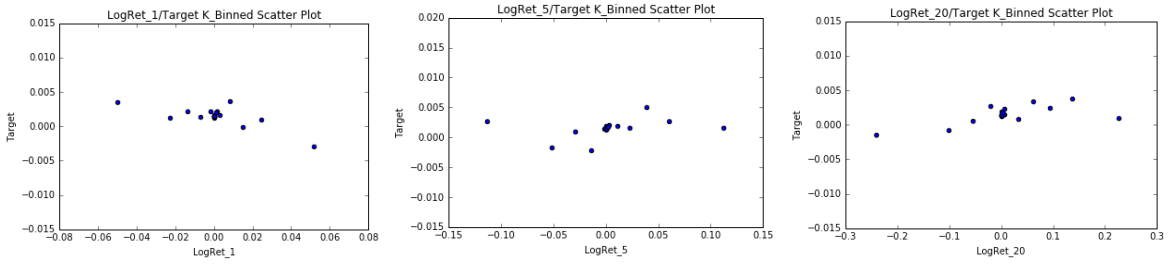



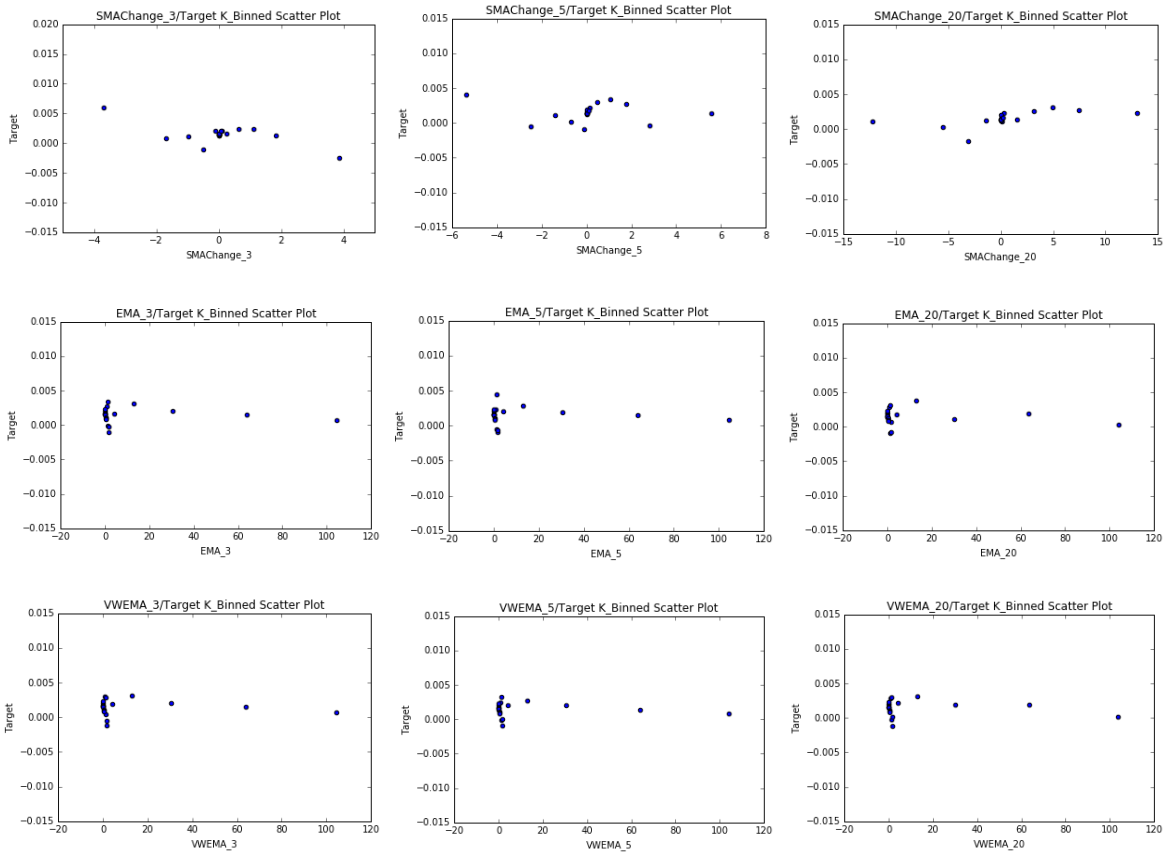

**4)** To read the scatter plots, I observed the overall pattern. Nevertheless, the previous figures don't exhibit any visual correlation between the independent variables and the target especially the first six ones.

**5)** As a result, I plotted the k-Binned scatter plot between two paired variables X and Y instead which consists of sorting X and discretizing it into k equal frequency bins, then computing the averages of X and Y in each bin and plotting the k Y averages as a function of the k X averages:

```
i=1
for variable in ['LogRet_1', 'LogRet_2', 'LogRet_3', 'LogRet_4', 'LogRet_5', 'LogRet_10', 'LogRet_20',
                 'SMAChange_3', 'SMAChange_4', 'SMAChange_5', 'SMAChange_10',
        'SMAChange_20', 'EMA_3', 'EMA_4', 'EMA_5', 'EMA_10', 'EMA_20',
        'VWEMA_3', 'VWEMA_4', 'VWEMA_5', 'VWEMA_10', 'VWEMA_20']:
            plt.figure(str(i))
            plt.xlabel(variable)
            plt.ylabel('Target')
            (Xq,XSe,Yq,YSe)=bscatter(Data[variable],Data['Target'],10)
            plt.scatter(Xq,Yq)
            plt.scatter(XSe,YSe)
            plt.title(variable+"/Target K_Binned Scatter Plot")
            i=i+1
```

The result of this script was:






**6)** In this question, I computed the linear correlation between each independent variable and the target.

To do that, I used the following script:

```
#### Linear Correlation
from scipy.stats.stats import pearsonr
from scipy.stats import ttest_ind
corr=[]
for variable in ['LogRet_1', 'LogRet_2', 'LogRet_3', 'LogRet_4', 'LogRet_5', 'LogRet_10', 'LogRet_20',
                 'SMAChange_3', 'SMAChange_4', 'SMAChange_5', 'SMAChange_10',
       'SMAChange_20', 'EMA_3', 'EMA_4', 'EMA_5', 'EMA_10', 'EMA_20',
       'VWEMA_3', 'VWEMA_4', 'VWEMA_5', 'VWEMA_10', 'VWEMA_20']:
            matrix =numpy.corrcoef(Data['Target'],Data[variable])
            linear_correlation=matrix[0,1]
            t,p=ttest_ind(Data['Target'],Data[variable])

            if p<0.05 :
                sig="significant"
            else:
                sig="non significant"

            print ('Correlation between Target and '+variable +' is '+str(linear_correlation)+
            ' and the relationship between the two variables is : '+sig)
            corr.append(linear_correlation)
```

The output was:

```
Correlation between Target and LogRet_1 is -0.017236507412 and the relationship
between the two variables is : non significant
Correlation between Target and LogRet_2 is -0.0246678994986 and the relationship
between the two variables is : non significant
Correlation between Target and LogRet_3 is -0.00317048427994 and the relationship
between the two variables is : non significant
Correlation between Target and LogRet_4 is 0.0149853016196 and the relationship
between the two variables is : non significant
Correlation between Target and LogRet_5 is 0.0149196505287 and the relationship
between the two variables is : significant
Correlation between Target and LogRet_10 is -0.00805466125839 and the relationship
between the two variables is : significant
Correlation between Target and LogRet_20 is 0.0227565484565 and the relationship
between the two variables is : significant

Correlation between Target and SMAChange_3 is -0.0246685097732 and the relationship
between the two variables is : significant
Correlation between Target and SMAChange_4 is -0.0170259211421 and the relationship
between the two variables is : significant
Correlation between Target and SMAChange_5 is -0.00678249987692 and the relationship
between the two variables is : significant
Correlation between Target and SMAChange_10 is 0.00302732835742 and the relationship
between the two variables is : significant
Correlation between Target and SMAChange_20 is 0.0147751833041 and the relationship
between the two variables is : significant

Correlation between Target and EMA_3 is -0.00390111159653 and the relationship
between the two variables is : significant
Correlation between Target and EMA_4 is -0.00391382917713 and the relationship
between the two variables is : significant
Correlation between Target and EMA_5 is -0.00390672059242 and the relationship
between the two variables is : significant
Correlation between Target and EMA_10 is -0.00392180422477 and the relationship
between the two variables is : significant
Correlation between Target and EMA_20 is -0.00421700890773 and the relationship
between the two variables is : significant

Correlation between Target and VWEMA_3 is -0.00392003100572 and the relationship
between the two variables is : significant
Correlation between Target and VWEMA_4 is -0.00394238816957 and the relationship
between the two variables is : significant
Correlation between Target and VWEMA_5 is -0.00395846419873 and the relationship
between the two variables is : significant
Correlation between Target and VWEMA_10 is -0.00394609410266 and the relationship
between the two variables is : significant
Correlation between Target and VWEMA_20 is -0.00423898145579 and the relationship
between the two variables is : significant
```

The strength of the relationship is indicated by the correlation coefficient. Its significance is expressed in probability levels: p-value of the **t-test** (significant at p-value <0.05). Therefore, the smaller the p-value, the more significant the relationship is and the larger the correlation, the stronger the relationship is.

**7)** To compute the mutual information between two continuous variables, I discretized both variables with an equal frequency discretization via the following Python code:

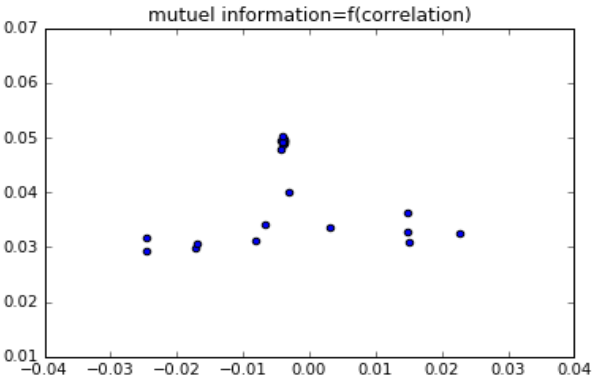
```
#### Mutual Information
from math import log
mut=[]
for n in list(Data)[0:len(list(Data))-1]:
    tar=pd.qcut(Data['Target'], 20, labels=False)
    variable=pd.qcut(Data[n], 20, labels=False)
    prob=pd.crosstab(variable, tar, margins=True)
    i=0
    Info=0
    for i in range(0,19):
        for j in range(0,19):
            Info=Info+prob.iloc[i,j]*(log((prob.iloc[i,j]*prob.iloc[20,20]))-log((prob.iloc[i,20]*prob.iloc[20,j])))/prob.iloc[2
    mut.append(Info)
```

The generated variable was:

| Indice | Type | Taille | Valeur |
|---|---|---|---|
| 0 | float64 | 1 | 0.029733510750628325 |
| 1 | float64 | 1 | 0.031674455361075174 |
| 2 | float64 | 1 | 0.039930456091033774 |
| 3 | float64 | 1 | 0.030982834615113579 |
| 4 | float64 | 1 | 0.032912459032296788 |
| 5 | float64 | 1 | 0.031181357816767508 |
| 6 | float64 | 1 | 0.032627900950620153 |
| 7 | float64 | 1 | 0.029343347044484373 |
| 8 | float64 | 1 | 0.030559314374972606 |
| 9 | float64 | 1 | 0.034149393258501942 |
| 10 | float64 | 1 | 0.033571191860252071 |
| 11 | float64 | 1 | 0.036391727915722712 |
| 12 | float64 | 1 | 0.049007067622847351 |
| 13 | float64 | 1 | 0.049070245330852684 |
| 14 | float64 | 1 | 0.049407009681898262 |
| 15 | float64 | 1 | 0.049576629776814068 |
| 16 | float64 | 1 | 0.04953538565079927 |

**8)** To compare the mutual information with the linear correlation, I plotted a graphic of the mutual information against the linear correlation (one point for each variable):

```
#### Plot of mutual information against the linear correlation
plt.scatter(x= corr,y= mut)
plt.title("mutuel information=f(correlation)")
```




This output shows that the Mutual information and the linear correlation are not antagonistic. In fact, The Mutual Information doesn't mean whether the association is linear or not, while the linear correlation may be zero and the variables may still be stochastically dependent.

## Linear Model:

**1)** Using the Python code *linear model.py*, I generated a linear model to predict Y:

```
#### Fitting a linear regression
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr = lr.fit(Data.ix[:,:-1], Data.Target)
```

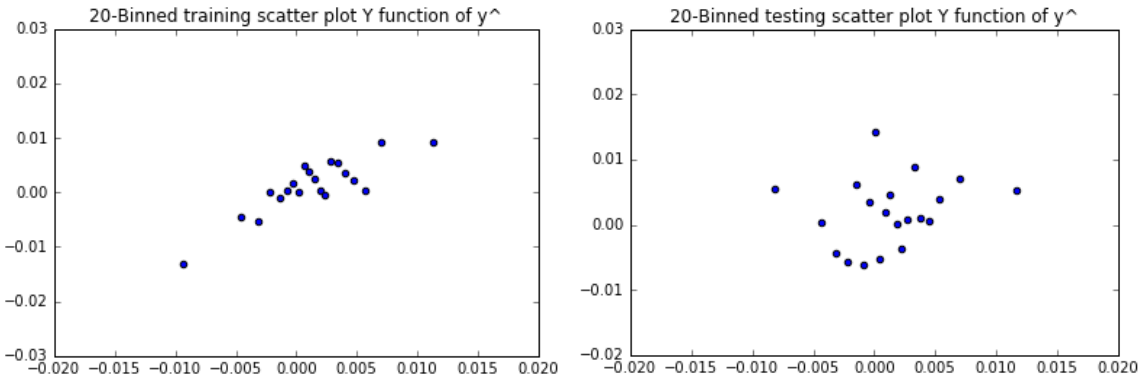
**2)** For a linear model, there should be a linear and additive relationship between dependent (response Y) variable and independent (predictor X) variable(s). A linear relationship suggests that a change in response Y due to one unit change in X is constant, regardless of the value of X. An additive relationship suggests that the effect of X on Y is independent of other variables. The independent variables should not be correlated (no multicollinearity). Moreover, there should be no correlation between the residual (error) terms (no autocorrelation). The residuals must have constant variance (homoskedasticity). They must be as well normally distributed.

**4)** When implementing a model, it is common to use a portion of the available data for fitting and use the rest of the data for testing the choosing model, as I did via the subsequent Python code:

```
#### Linear model using the training data and test data
from sklearn.cross_validation import train_test_split

#Data split
train, test = train_test_split(Data,test_size=0.2)

#Linear model for training
trainFit = LinearRegression()
trainFit = trainFit.fit(train.ix[:,:-1], train.Target)

#Prediction
pred = trainFit.predict(test.ix[:,:-1])
```

**5)** Computing the 20-Binned scatter plot between Y and Y^ for both training and testing generated the next graphics:

```
#### 20-binned training scatter plot
predTrain = trainFit.predict(train.ix[:,:-1])
k=20
Xq,XSe,Yq,YSe=bscatter(predTrain, train.iloc[:,-1],k)
plt.scatter(Xq,Yq)
plt.title("20-Binned training scatter plot Y function of y^")


#### 20-binned testing scatter plot
k=20
Xq,XSe,Yq,YSe=bscatter(pred, test.iloc[:,-1],k)
plt.scatter(Xq,Yq)
plt.title("20-Binned testing scatter plot Y function of y^")
```




When observing these plots, it is clear that we are faced by a problem of over-fitting since the implemented model works well on the training data but fails to predict on a new data (the testing sets) which was not used while fitting.

**6)** To evaluate the forecast accuracy of our model, I calculated the relative R-squared value:

```
#### R-squared
Rsquared = trainFit.score(train.ix[:,:-1], train.iloc[:,-1], sample_weight=None)

Rsquared
0.012640989254668944
```

Knowing that R-squared is a statistical measure of how close the data are to the fitted regression line, I confirm that the implemented model fails to generate reliable predicted values because the resulting R-squared is manifestly close to zero.

7) Finally, I implemented a Python code that determines a trading strategy and computes the cumulative return gotten:

```
#### Trading strategy implementation
a=df.ix[-2:,5:-1]
a=a[['LogRet_1', 'LogRet_2', 'LogRet_3', 'LogRet_4', 'LogRet_5', 'LogRet_10', 'LogRet_20',
                'SMAChange_3', 'SMAChange_4', 'SMAChange_5', 'SMAChange_10',
       'SMAChange_20', 'EMA_3', 'EMA_4', 'EMA_5', 'EMA_10', 'EMA_20',
       'VWEMA_3', 'VWEMA_4', 'VWEMA_5', 'VWEMA_10', 'VWEMA_20']]
predTrading = trainFit.predict(a)
print("The profit (/loss) would be " + str(Data.ix[:,-1][-1]))
```

```
The profit (/loss) would be 0.00889438293285
```