FRA – Assignment

Group 1

PGP-BABI Aug'17

Contents

About the dataset

The IBM dataset contains the values following fields related to the IBM stocks:

- Date field
- 5 fields related to the Stock prices: Open, High, Low, Close and Adjusted Closing Price
- Volume

The period for the given data, ranges from 27-06-2011 to 28-06-2013.
Note: For the following problem statements we will only be considering the Date field and the Adjusted Closing Price.
We have the financial data of IBM. The objective is to predict the future stock prices. As we have a time series data, we will be using the price history (the past observations) to predict future stock prices/movement.

Q1. Build a univariate model for forecasting returns

Objective:

- To forecast, ie. To predict tomorrow's stock price based on today's stock price
- To check if there is momentum or contrarian in the data
    - Momentum: if today's price increases, tomorrow's price will also increase.
    - Contrarian: if today's price increases, tomorrow's price will decrease.

Data: Time series data of Stock price.
For this problem statement we have considered adjusted closing price in the time period between 02-07-2012 and 31-12-2012.

Any Time series data will have the following components: Trend, Seasonality and Error.
The error component is what we are interested to know more about through our models. (Because the trend and seasonality are known).

We should first take out the effect of **trend** from the data. This can be achieved by taking a difference of the Time series values. For the price change to be comparable across all values, we compute the relative change in the price.

Steps followed for building a linear regression model:

1. Create a field called Returns. The computation for this field will be based on the relative change between the Adjusted Closing Price, of the current and the past observation. If r_t is the return for the current data point and P_t is the price, then:
$$r_t = (P_t - P_{t-1})/P_{t-1}$$
2. Create a field called Lagged returns. This field will be the returns achieved from Step1, with a lag of one day.

A snapshot of the excel table with returns and lagged-returns

Date	Adj Close	Returns (Y)	Lagged Returns (X)
02-07-12	166.972031		
03-07-12	167.057327	0.0005108	
05-07-12	166.511627	-0.003267	0.00051084
06-07-12	163.2034	-0.019868	-0.003266543
09-07-12	161.719818	-0.00909	-0.019867844
10-07-12	158.812347	-0.017978	-0.009090387
11-07-12	157.951126	-0.005423	-0.017978446
12-07-12	156.109451	-0.01166	-0.005422884
13-07-12	158.599167	0.0159485	-0.011659778
16-07-12	157.558945	-0.006559	0.015948528
17-07-12	156.586914	-0.006169	-0.006558811
18-07-12	160.509094	0.0250479	-0.006169317
19-07-12	166.554291	0.0376626	0.025047942
20-07-12	164.090149	-0.014795	0.037662645

3. Build a regression model, with 'Returns' as the Y-variable (dependent variable) and 'Lagged Returns' as the X-variable (Independent variable).
   The regression equation will take the form:

$$r_t = \alpha + \beta r_{t-1} + \epsilon_t$$

Results from performing the linear regression:

Regression Statistics								
Multiple R	0.184577998							
R Square	0.034069037							
Adjusted R Square	0.026086137							
Standard Error	0.010455888							
Observations	123							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.000467	0.000467	4.267752	0.040976966			
Residual	121	0.013228	0.000109					
Total	122	0.013695						
	Coefficients	*andard Err*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-3.42367E-05	0.000943	-0.03631	0.971093	-0.001900846	0.001832373	-0.001900846	0.001832373
X Variable 1	0.185142722	0.08962	2.065854	0.040977	0.007715422	0.362570021	0.007715422	0.362570021

The p-value is significant at 95% c.f. So, at 95% confidence level the model is significant and shows a **momentum** pattern.

The regression equation for the model is: $\mathbf{r_t = -0.0000342 + 0.185\ r_{t-1} + e_t}$

R-square of the model is **3.41%**

At 95% confidence we can say that the market has some momentum and the true beta lies between [0.0077, 0.3626]

We now know the alpha and beta values, which are **-0.0000342 and 0.185** respectively. Using these values, we can make predictions on the expected returns and devise trading strategies.

Q2. Convert predictions to trading strategies

Objective:

- Devise rules for buying/selling the stock
- Analyze the prescription based on the profit/loss from an initial wealth which is fed into the model

Time period to be considered for prediction: 29-06-2012 till 31-12-2012
Time period for the training data: 02-07-2012 till 31-12-2012

We follow the steps mentioned in the previous section for building a linear regression model.
The regression equation will take the form:

$$r_t = \alpha + \beta r_{t-1} + \epsilon_t$$

Results from the linear regression model:

Regression Statistics	
Multiple R	0.167198201
R Square	0.027955239
Adjusted R Sc	0.020116168
Standard Erro	0.0105402
Observations	126

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.000396183	0.000396	3.566142	0.06130549
Residual	124	0.01377588	0.000111		
Total	125	0.014172064			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.000132198	0.000938996	0.140787	0.888267	-0.0017263	0.0019907	-0.0017263	0.001991
X Variable 1	0.167264359	0.088573559	1.888423	0.061305	-0.0080475	0.3425762	-0.0080475	0.342576

The p-value is significant at 95% c.f. So, at 90% confidence level the model is significant and shows a **momentum** pattern.

The regression equation for the model is: **$r_t = 0.0001322 + 0.1672\ r_{t-1} + e_t$**
R-square of the model is 2.79%
At 90% confidence we can say that the market has some momentum and the true beta lies between [-0.008, 0.3426]
Using the alpha and beta values, we can make predictions on the expected returns and devise trading strategies.

alpha	0.000132198
beta	0.167264359

Using the equation "$r_t = 0.0001322 + 0.1672\ r_{t-1}$" we can proceed to compute the prediction values for all the data points in the period considered.

The forecast will be for the next day. ie, the prediction for 3rd July, will be made at the end of day on 2nd July.

Prescriptive Analytics:

For a prediction that the price will go up; the prescription will be to Buy the stock

For a prediction that the price will go down; the prescription will be to Sell the stock (and buy it back for a lesser amount later).
Note: All the actions are carried out on the assumption that the buy/sell positions are held for a day.

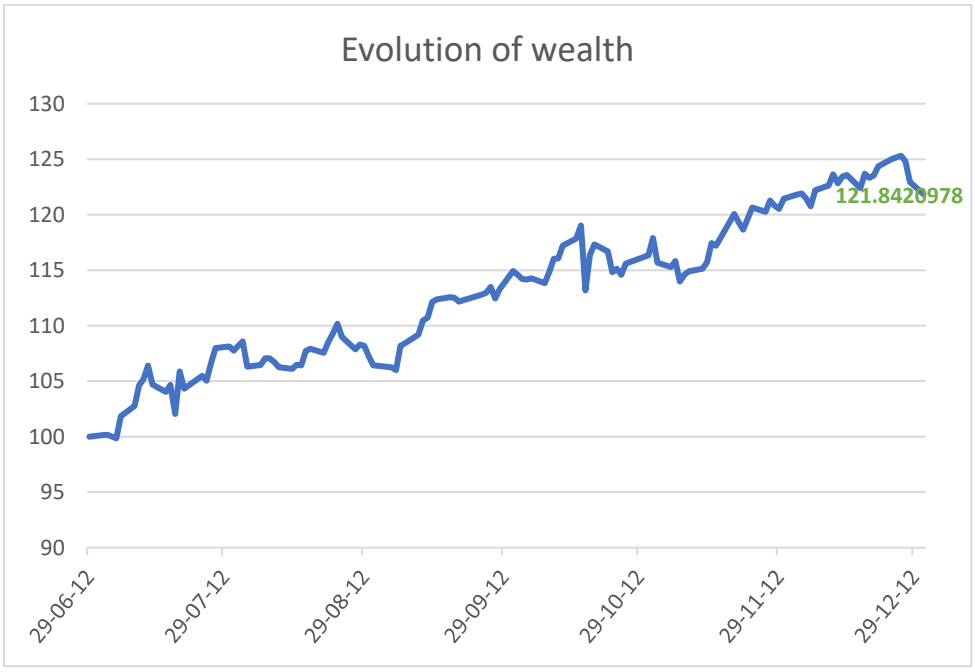
Snapshot of the excel workings:

Date	Adj Close	Return (Y)	Lagged return (X	Predictions	Prescription	Verbose prescriptic	Square-off	Verbose Sqr-off	Profit	Ini wealth	# units	Actual profit	Final wealth
27-06-12	164.559082												
28-06-12	163.194855	-0.00829											
29-06-12	166.758911	0.0218393	-0.008290196	0.00378513	Buy	Buy at 166.76	Sell	Sell at 166.97	0.21312	100	0.599668	0.127801266	100.1278
02-07-12	166.972031	0.001278	0.021839267	0.00034596	Buy	Buy at 166.97	Sell	Sell at 167.06	0.085296	100.1278	0.599668	0.051149291	100.179
03-07-12	167.057327	0.0005108	0.001278013	0.00021764	Buy	Buy at 167.06	Sell	Sell at 166.51	-0.5457	100.179	0.599668	-0.327238884	99.85171
05-07-12	166.511627	-0.003267	0.00051084	-0.0004142	Sell	Sell at 166.51	Buy	Buy at 163.2	3.308227	99.85171	0.599668	1.983838213	101.8355
06-07-12	163.2034	-0.019868	-0.003266543	-0.003191	Sell	Sell at 163.2	Buy	Buy at 161.72	1.483582	101.8355	0.623979	0.925724518	102.7613
09-07-12	161.719818	-0.00909	-0.019867844	-0.0013883	Sell	Sell at 161.72	Buy	Buy at 158.81	2.907471	102.7613	0.635428	1.847488013	104.6088
10-07-12	158.812347	-0.017978	-0.009090387	-0.002875	Sell	Sell at 158.81	Buy	Buy at 157.95	0.861221	104.6088	0.658694	0.567281226	105.176
11-07-12	157.951126	-0.005423	-0.017978446	-0.0007749	Sell	Sell at 157.95	Buy	Buy at 156.11	1.841675	105.176	0.665877	1.226329277	106.4024
12-07-12	156.109451	-0.01166	-0.005422884	-0.0018181	Sell	Sell at 156.11	Buy	Buy at 158.6	-2.48972	106.4024	0.681588	-1.696961258	104.7054
13-07-12	158.599167	0.0159485	-0.011659778	0.00279982	Buy	Buy at 158.6	Sell	Sell at 157.56	-1.04022	104.7054	0.660189	-0.686743031	104.0187

Performance Analytics

Based on the above calculations, the evolution of wealth over a six-month period (starting 29/06/2012 till 31/12/2012) is plotted using an Equity curve (which is a plot of time by initial wealth).

Equity curve – 29th June till 31st Dec, 2012



From the results of the model and from the above chart, we can see that the Regression model and the predictions based on that may yield a profit of $21.84 after a 6-month period.

Q3. Comparing the model with a benchmark

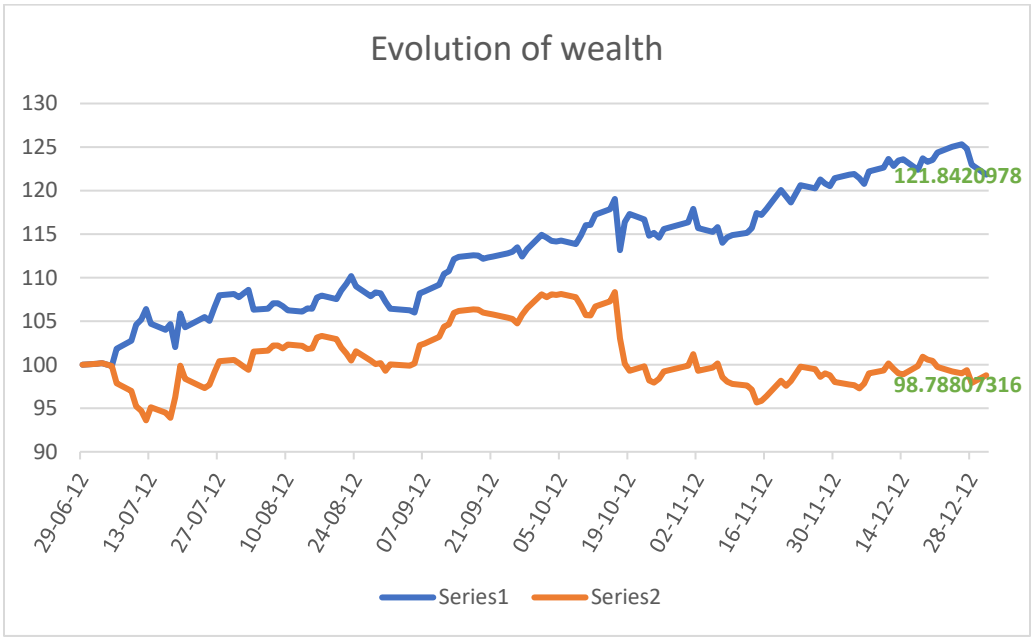
Objective: To compare the regression model with a benchmark (buy-and-hold strategy).

The buy-and-hold strategy is where the stock which is bought at the beginning of the time period is held-onto until the end of the time period considered.

Snapshot of the excel workings:

Date	Adj Close	Return (Y)	Predictions	Prescription	Verbose prescriptic	Square-off	Verbose Sqr-off	Profit	Ini wealth	# units	actual pro	Final wealth	buy-and-hold model	
27-06-12	164.559082							34.041264					ini wealth	#units
28-06-12	163.194855	-0.00829												
29-06-12	166.758911	0.0218393	0.00378513	Buy	Buy at 166.76	Sell	Sell at 166.97	0.21312	100	0.599668	0.127801	100.1278013	100	0.599668
02-07-12	166.972031	0.001278	0.00034596	Buy	Buy at 166.97	Sell	Sell at 167.06	0.085296	100.1278	0.599668	0.051149	100.1789506	100.1278	
03-07-12	167.057327	0.0005108	0.00021764	Buy	Buy at 167.06	Sell	Sell at 166.51	-0.5457	100.179	0.599668	-0.32724	99.85171167	100.179	
05-07-12	166.511627	-0.003267	-0.0004142	Sell	Sell at 166.51	Buy	Buy at 163.2	3.308227	99.85171	0.599668	1.983838	101.8355499	99.85171	
06-07-12	163.2034	-0.019868	-0.003191	Sell	Sell at 163.2	Buy	Buy at 161.72	1.483582	101.8355	0.623979	0.925725	102.7612744	97.86787	
09-07-12	161.719818	-0.00909	-0.0013883	Sell	Sell at 161.72	Buy	Buy at 158.81	2.907471	102.7613	0.635428	1.847488	104.6087624	96.97822	
10-07-12	158.812347	-0.017978	-0.002875	Sell	Sell at 158.81	Buy	Buy at 157.95	0.861221	104.6088	0.658694	0.567281	105.1760436	95.2347	
11-07-12	157.951126	-0.005423	-0.0007749	Sell	Sell at 157.95	Buy	Buy at 156.11	1.841675	105.176	0.665877	1.226329	106.4023729	94.71825	
12-07-12	156.109451	-0.01166	-0.0018181	Sell	Sell at 156.11	Buy	Buy at 158.6	-2.489716	106.4024	0.681588	-1.69696	104.7054117	93.61386	
13-07-12	158.599167	0.0159485	0.00279982	Buy	Buy at 158.6	Sell	Sell at 157.56	-1.040222	104.7054	0.660189	-0.68674	104.0186686	95.10686	
16-07-12	157.558945	-0.006559	-0.0009649	Sell	Sell at 157.56	Buy	Buy at 156.59	0.972031	104.0187	0.660189	0.641724	104.6603927	94.48307	
17-07-12	156.586914	-0.006169	-0.0008997	Sell	Sell at 156.59	Buy	Buy at 160.51	-3.92218	104.6604	0.668385	-2.62153	102.0388652	93.90018	
18-07-12	160.509094	0.0250479	0.00432183	Buy	Buy at 160.51	Sell	Sell at 166.55	6.045197	102.0389	0.63572	3.843054	105.8819188	96.25218	
19-07-12	166.554291	0.0376626	0.00643182	Buy	Buy at 166.55	Sell	Sell at 164.09	-2.464142	105.8819	0.63572	-1.5665	104.3154141	99.8773	
20-07-12	164.090149	-0.014795	-0.0023424	Sell	Sell at 164.09	Buy	Buy at 162.71	1.381287	104.3154	0.63572	0.878112	105.193526	98.39963	

Equity curve comparing the evolution of the initial wealth through the regression model with that of the buy-and-hold-strategy



Series 1 – represents the curve using the regression model

Series 2 – represents the curve using the benchmark strategy (buy and hold strategy)

For $100 invested at the start of the period at 2nd July 2012, at the end of 6 months, the regression model predicts a profit of $21.68 (total wealth in hand at the end of 6 months is $121.608)

Whereas the benchmark strategy of buy-and-hold would yield a loss of $1.21 (total wealth in hand at the end of 6 months is $98.788).

The regression model acts better than the buy-and-hold strategy especially when the market is down. The equity curve of both are the same when the market just keeps going up (i.e. when there is no downward movement in the market)

Q4. Use the trained model to make predictions on a test data

Objective:

- Build a model on a train data (2nd July 2012 to 31st December 2012).
- Evaluate the performance of the model on the test data (2nd January 2013 to 28th June 2013)
- Compare the model performance on the test data with a benchmark strategy

Similar to the steps followed in the solution for Question 2, we use the alpha and beta values to make predictions on the expected returns on the test set.

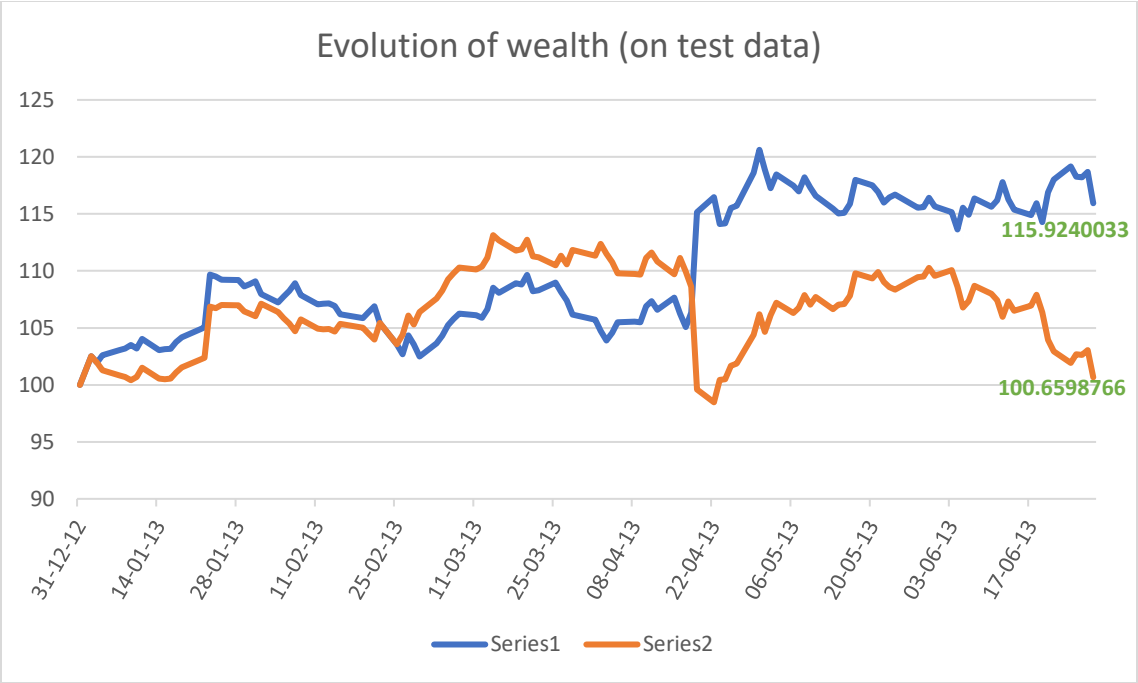
alpha	0.000132198
beta	0.167264359

Using the equation "$r_t = 0.0001322 + 0.1672\ r_{t-1}$" we can proceed to compute the prediction values for all the data points in the test data.

We then compare the wealth movement using the regression model with that using the buy-and-hold-strategy.

Screenshot of the excel workings:

Date	Adj Close	Return (Y)	Predictions	Prescriptic	Profit	Ini wealth	# units	actual profits	Final weal	buy-and-hold strategy	
27-06-12	164.559082				(1 unit)		(fractions ok)			ini wealth	#units
31-12-12	164.737915	0.00906067	0.001647725	Buy	4.128143	100	0.6070248	2.505885181	102.5059	100	0.607025
02-01-13	168.866058	0.025058852	0.004323651	Buy	-0.92886	102.50589	0.6070248	-0.563842877	101.942	102.50589	
03-01-13	167.937195	-0.00550059	-0.000787854	Sell	1.100815	101.94204	0.6070248	0.668222006	102.6103	101.94204	
04-01-13	166.83638	-0.006554921	-0.000964206	Sell	0.731003	102.61026	0.61503531	0.449592655	103.0599	101.27382	
07-01-13	166.105377	-0.004381556	-0.00060068	Sell	0.232223	103.05986	0.62044865	0.144082447	103.2039	100.83008	
08-01-13	165.873154	-0.001398046	-0.000101645	Sell	0.473023	103.20394	0.62218591	0.294308246	103.4982	100.68912	
09-01-13	165.400131	-0.002851715	-0.000344792	Sell	-0.48164	103.49825	0.62574465	-0.301386158	103.1969	100.40198	
10-01-13	165.881775	0.002911993	0.000619271	Buy	1.350235	103.19686	0.62211091	0.839995921	104.0369	100.69435	
11-01-13	167.23201	0.008139743	0.001493687	Buy	-1.57387	104.03686	0.62211091	-0.979120449	103.0577	101.51398	
14-01-13	165.658142	-0.009411284	-0.001441974	Sell	0.103195	103.05774	0.62211091	0.064198735	103.1219	100.5586	
15-01-13	165.554947	-0.000622939	2.80027E-05	Buy	0.077377	103.12194	0.62288647	0.048197086	103.1701	100.49596	
16-01-13	165.632324	0.00046738	0.000210374	Buy	0.911652	103.17013	0.62288647	0.567855692	103.738	100.54293	
17-01-13	166.543976	0.005504071	0.001052833	Buy	0.705246	103.73799	0.62288647	0.439288188	104.1773	101.09632	
18-01-13	167.249222	0.004234593	0.000840495	Buy	1.384613	104.17728	0.62288647	0.862456697	105.0397	101.52443	
22-01-13	168.633835	0.008278741	0.001516937	Buy	7.430633	105.03973	0.62288647	4.628440722	109.6682	102.36492	
23-01-13	176.064468	0.044063714	0.007502487	Buy	-0.25803	109.66817	0.62288647	-0.160720903	109.5075	106.8755	



Series 1 – represents the curve using the regression model
Series 2 – represents the curve using the benchmark strategy (buy and hold strategy)

For $100 invested at the start of the period at 31st Dec 2012, at the end of 6 months, the regression model predicts a profit of $15.92 (total wealth-in-hand at the end of 6 months is $115.924)

Whereas the benchmark strategy of buy-and-hold would yield a profit of $0.659 (total wealth-in-hand at the end of 6 months is $100.659)

Q5. Forecast based on a sliding window

The linear regression model used above, is considered for building a sliding window model.

Objective:

- To build a linear regression model for a 6-month window. And use it to predict five data points post the 6-month period (in 2013).
- Then rebuild the model by sliding the 6-month window by 5 days. (The 5-day period is considered because typically the weekly stock prices are for the weekdays).

Explanation of the logic used for sliding window regression modeling (using R code)

Step 1:

The lagged returns and the returns where considered as the X(independent) variable and the Y (dependent variable) respectively. The objective of the regression model is to predict the returns based on the lagged returns. The data had values from July 2012 till end of June 2013.

```
ibmData = read.csv("dataset_for_Q5.csv")
```

Step 2:

We need a sliding window of 6 months (~120 days) to be used for building the model. The model built, is to be used for predicting 5 data points in the test data.

As the test data contains around 125 observations from 02/02/2013 to 28/06/2013, predicting 5 data points each time will require the number of windows to be 125/5 = 25.

```
w_size = 120  #sets the Window size
n_windows = 25  #sets the Number of windows
```

Step 3:

The training data and the test data were initiated afresh in a loop. The model will be retrained during every loop (for every 5 days). Each time the prediction will be made for 5 data points in the test data. These new 5 data points were infused back into the data set replacing the oldest observations.

```
###Rolling Window Loop ###
forecasts = foreach(i=1:n_windows, .combine = rbind)%do%
 {
  # Select data for the training data and the test data
 trainData = ibmData[i:(w_size + i - 1), ]
 testData = idmData[(w_size + i):(w_size + i+4), ]

 # Regression Model – the model will be rebuilt during every loop
 modelLR = lm(return ~ lagged_return, data = trainData)
 f1 = predict(modelLR, testData)

 # Replace the oldest five observations with the newly forecasted points
 ibmData[(w_size - 114),]$return <- getElement(f1, as.character(w_size+i))
 ibmData[(w_size - 113),]$return <- getElement(f1, as.character(w_size+i+1))
 ibmData[(w_size - 112),]$return <- getElement(f1, as.character(w_size+i+2))
```

```
ibmData[(w_size - 111),]$return <- getElement(f1, as.character(w_size+i+3))
ibmData[(w_size - 110),]$return <- getElement(f1, as.character(w_size+i+4))
return(f1)
}
```

Step 4:

The final step is to use the forecasts which will be a 25 X 5 matrix whose data points contain the projected returns on the test data. The predictions can be used for making the prescriptions and profit estimations, which can be evaluated using equity curve.

```
forecasts
write.csv(forecasts,"sw_forecasts_new.csv")
```

The below matrix is the result of running a sliding window regression model. Each data point represents the predicted Returns(Y).
There were totally 25 windows, each with a result of 5 data points.

Forecast results

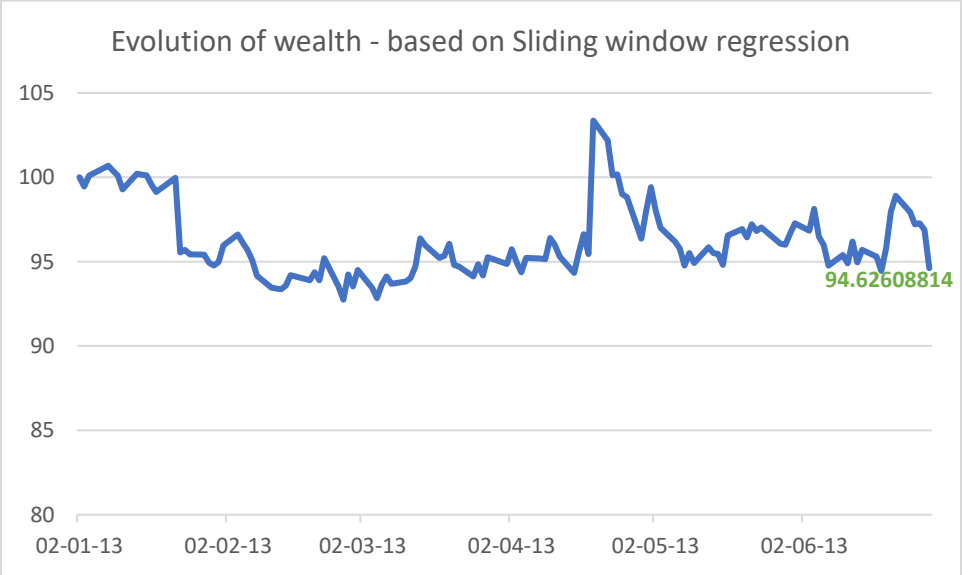
result.1	0.00176	0.004517	-0.00075	-0.00093	-0.00056
result.2	0.004947	-0.00044	-0.00062	-0.00024	0.000285
result.3	-0.00036	-0.00053	-0.00019	0.000266	4.27E-05
result.4	-0.00047	-0.00014	0.000313	9.30E-05	0.000964
result.5	-0.0004	0.000101	-0.00014	0.00083	0.001713
result.6	0.000151	-0.00011	0.00093	0.001875	-0.0013
result.7	-8.33E-05	0.000944	0.001876	-0.00125	0.000314
result.8	0.000784	0.001781	-0.00156	0.00011	0.000318
result.9	0.001076	-0.00116	-4.04E-05	9.85E-05	0.00074
result.10	-0.00164	-5.47E-06	0.000197	0.001135	0.000898
result.11	5.77E-05	0.000251	0.001144	0.000919	0.001636
result.12	0.000265	0.001153	0.000929	0.001642	0.007953
result.13	0.001168	0.000943	0.001661	0.008014	-6.93E-05
result.14	0.000837	0.001542	0.007779	-0.00016	0.000568
result.15	0.001434	0.007283	-0.00016	0.00052	4.87E-05
result.16	0.009116	0.000149	0.000968	0.000399	-0.00055
result.17	0.000167	0.000834	0.000371	-0.0004	0.000103
result.18	0.000884	0.000426	-0.00034	0.00016	0.000105
result.19	0.000447	-0.00031	0.000185	0.00013	0.002114
result.20	-0.00058	-5.48E-05	-0.00011	0.001967	-0.00087
result.21	-5.53E-05	-0.00011	0.002026	-0.00089	-0.00057
result.22	-0.00018	0.001962	-0.00095	-0.00064	-0.00058
result.23	0.002047	-0.00084	-0.00053	-0.00047	-0.00077
result.24	-0.00083	-0.00053	-0.00048	-0.00077	0.001818
result.25	-0.00063	-0.00057	-0.00087	0.00178	-0.00106

The prescription strategy is built based on the Predicted values from the sliding window-regression model.

Screenshot of the excel workings:

Date	Adj Close	Lagged return (X)	Predictions (Y) based on sliding window regr	Prescription	Verbose prescription	Square-off	Verbose Sqr-off	Profit	Ini wealth	# units	actual profits	Final wealth
31-05-13	180.49765	0.006925925	-0.000572051	Sell	Sell at 180.5	Buy	Buy at 181.3	-0.806915	100.0353	0.55421939	-0.447208	99.58809
03-06-13	181.304565	-0.006400515	-0.000177347	Sell	Sell at 181.3	Buy	Buy at 178.91	2.394775	99.588089	0.54928616	1.3154168	100.9035
04-06-13	178.90979	0.004470501	0.001962301	Buy	Buy at 178.91	Sell	Sell at 175.92	-2.993561	100.90351	0.56399097	-1.688341	99.21516
05-06-13	175.916229	-0.013208575	-0.000952275	Sell	Sell at 175.92	Buy	Buy at 176.84	-0.919754	99.215164	0.56399097	-0.518733	98.69643
06-06-13	176.835983	-0.016732237	-0.000635134	Sell	Sell at 176.84	Buy	Buy at 179.05	-2.212632	98.696431	0.55812414	-1.234923	97.46151
07-06-13	179.048615	0.005228364	-0.00057551	Sell	Sell at 179.05	Buy	Buy at 177.89	1.154023	97.461508	0.54432986	0.6281692	98.08968
10-06-13	177.894592	0.01251234	0.00204748	Buy	Buy at 177.89	Sell	Sell at 176.99	-0.902404	98.089677	0.55139212	-0.497578	97.5921
11-06-13	176.992188	-0.006445305	-0.000841771	Sell	Sell at 176.99	Buy	Buy at 174.58	2.412201	97.592099	0.55139212	1.3300686	98.92217
12-06-13	174.579987	-0.005072689	-0.000527386	Sell	Sell at 174.58	Buy	Buy at 176.81	-2.22995	98.922167	0.56662948	-1.263555	97.65861
13-06-13	176.809937	-0.013628856	-0.000468281	Sell	Sell at 176.81	Buy	Buy at 175.45	1.362275	97.658612	0.55233667	0.7524344	98.41105
14-06-13	175.447662	0.012773228	-0.000773279	Sell	Sell at 175.45	Buy	Buy at 176.18	-0.728897	98.411046	0.56091398	-0.408849	98.0022
17-06-13	176.176559	-0.007704742	-0.00083481	Sell	Sell at 176.18	Buy	Buy at 177.76	-1.587876	98.002198	0.55627263	-0.883292	97.11891
18-06-13	177.764435	0.004154498	-0.000534285	Sell	Sell at 177.76	Buy	Buy at 175.22	2.542343	97.118906	0.54633485	1.3889706	98.50788
19-06-13	175.222092	0.009012981	-0.000477785	Sell	Sell at 175.22	Buy	Buy at 171.24	3.982727	98.507876	0.56218868	2.239044	100.7469
20-06-13	171.239365	-0.014301753	-0.000769337	Sell	Sell at 171.24	Buy	Buy at 169.6	1.639939	100.74692	0.58833972	0.9648413	101.7118
21-06-13	169.599426	-0.022729594	0.001818042	Buy	Buy at 169.6	Sell	Sell at 167.93	-1.665985	101.71176	0.59971761	-0.999121	100.7126
24-06-13	167.933441	-0.009576881	-0.000625669	Sell	Sell at 167.93	Buy	Buy at 169.18	-1.249481	100.71264	0.59971761	-0.749336	99.96331
25-06-13	169.182922	0.009823058	-0.000567895	Sell	Sell at 169.18	Buy	Buy at 169.08	0.10411	99.963305	0.59085931	0.0615144	100.0248
26-06-13	169.078812	0.007440335	-0.000866021	Sell	Sell at 169.08	Buy	Buy at 169.76	-0.685485	100.02482	0.59158696	-0.405524	99.6193
27-06-13	169.764297	-0.000615369	0.001779703	Buy	Buy at 169.76	Sell	Sell at 165.82	-3.939315	99.619296	0.58680946	-2.311627	97.30767
28-06-13	165.824982	0.004054234	-0.001058371	Sell	Sell at 165.82	Buy	Buy at 0		97.307668		0	

Equity curve based on the predictions from the sliding window regression model



We could see that the sliding window regression model has actually predicted a loss of $2.692

For an amount of $100 invested on 2nd Jan 2013, at the end of the 6-month period on 28th June 2013 it would be $97.308

Q6. Use different model specification: Logistic regression

We so far tried to estimate the predicted returns which is a continuous value. Using logistic regression, we can only predict binary outcomes. Hence, the estimated outcome from logistic regression model will feed in to our prescription of whether to buy or sell the stock in hand.

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). It is a special type of regression where the binary response variable is related to one or more explanatory variables, which can be discrete and/or continuous.
With the provided dataset, we build the Logistic Regression model to predict whether to buy or sell the stock.

Objective: To get predictions as 0s and 1s. (0 referring to a prediction to Sell and 1 refers to Buy).

Steps followed for logistic regression:

- Create a new field called 'Actual'. Assign values 0 or 1 in this field, based on the Return value being positive/negative.
- Perform logistic regression with lagged-return as the X variable and 'Actual' as the Y variable.

Alternate Hypothesis: Lagged-return is a significant predictor of the decision to buy/sell.

Interpreting Logistic Regression model

Overall Significance of the Logistic Regression to test Applicability

```
Likelihood ratio test

Model 1: Actual ~ Lagged_return
Model 2: Actual ~ 1
  #Df  LogLik Df  Chisq Pr(>Chisq)
1   2 -84.637
2   1 -86.939 -1 4.6051    0.03188 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation:
The overall test for model significance based on the chi-square test is significant indicating the existence of a predictor variable. This implies that, the likelihood of the buy/sell decision depends upon the lagged-return.

McFadden R-Square

```
  McFadden
    0.0265
```

Interpretation:
Based on the McFadden R-Square we conclude that 2.65% of the uncertainty of the intercept only model (Model 2) has been explained by the Full Model (Model 1). The value indicates a rather weak goodness of fit.

Confusion Matrix (Misclassification Matrix)

Classify the Training Data, and Develop a Confusion Matrix

Training Data

	Predicted	
Actual	0	1
0	53	15
1	36	22

Interpretation:

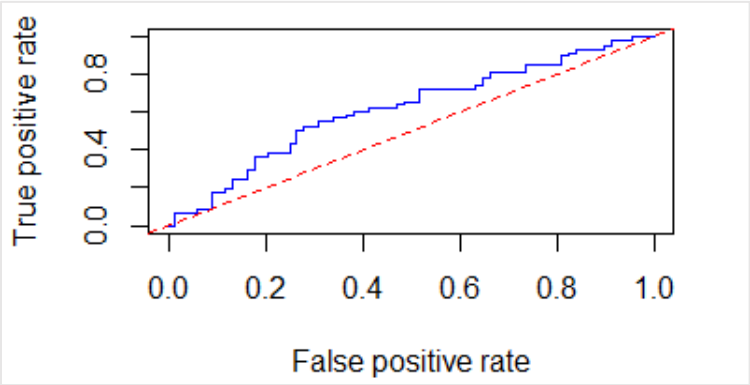
Accuracy = **59.52%**

i.e. Out of the 126 data points the model has correctly predicted 75 records.

False positives = 15; the model has wrongly predicted 'buy' where the actual decision was to sell.

False negatives = 36; The model has wrongly predicted 'sell' where the actual decision was to buy.

ROC Plot



AUC indicates that **61.76%** of the points are within the area of the curve (i.e., under the curve)

Predicting the Test Data

Test Data

	Predicted	
Actual	0	1
0	36	28
1	43	18

Accuracy for the test data = **43.2%**

Comparing the results of the Train and Test data to validate the model:

Performance Measures (in %)	Training Data	Test Data
Accuracy	59.52%	43.2%
Sensitivity	37.93%	29.51%
Specificity	77.94%	56.25%
AUC (ROC)	61.76%	46.69%

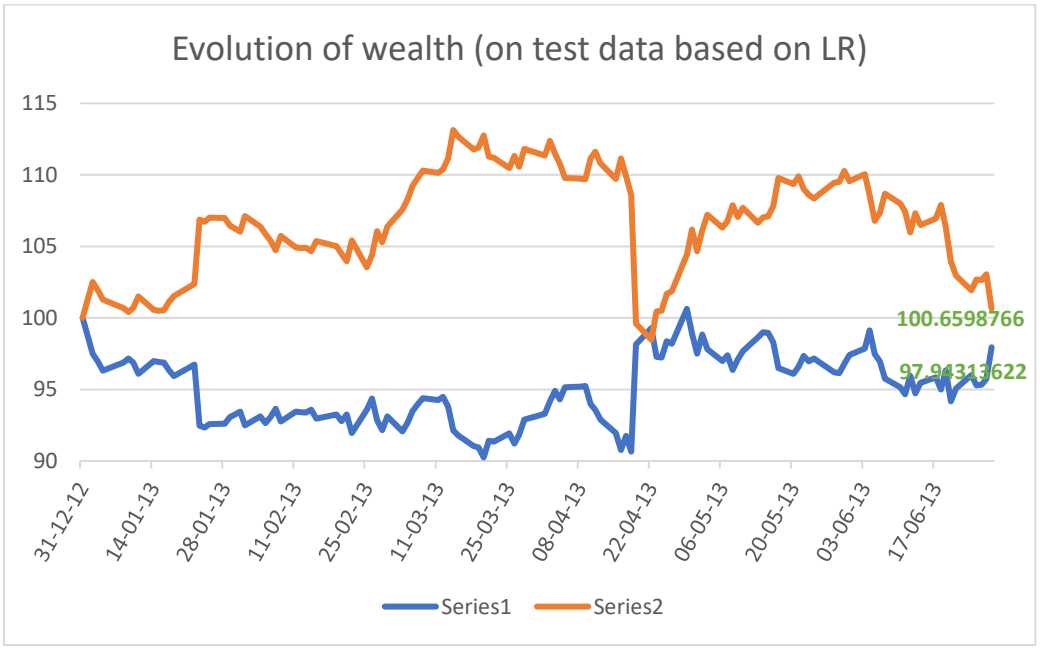
From the above table, it could be seen that, the Accuracy rates are not that satisfactory. Also, the model is not consistent across the Training set and the Test set.
However, using the above performance measures to evaluate may not be the right approach as we are dealing with Time series data. **Equity curve** can be a good measure of performance in such cases.

Prescription, Profit measurement, Wealth movement based on the Buy/Sell strategy from logistic regression

Now that we have predicted 0's and 1's on the test data, using this we can, come up with Buy/Sell (1/0) strategy similar to what we did under solution for Question 2.

Snapshot of excel workings:

Date	Adj Close	Return (Y)	Actual	LR-Predictions	Prescription	Verbose prescription	Square-off	Verbose Sqr-off	Profit	Ini wealth	# units	actual pro	Final weal	Buy-and-hold-strategy	
31-12-12	164.737915	0.0090607	1	0	Sell	Sell at 164.74	Buy	Buy at 168.87	-4.12814	100	0.6070248	-2.50589	97.49411	100	0.607025
02-01-13	168.866058	0.0250589	1	1	Buy	Buy at 168.87	Sell	Sell at 167.94	-0.92886	97.4941148	0.57734583	-0.53628	96.95784	102.5059	
03-01-13	167.937195	-0.005501	0	1	Buy	Buy at 167.94	Sell	Sell at 166.84	-1.10082	96.9578396	0.57734583	-0.63555	96.32229	101.942	
04-01-13	166.83638	-0.006555	0	0	Sell	Sell at 166.84	Buy	Buy at 166.11	0.731003	96.3222887	0.57734583	0.422042	96.74433	101.2738	
07-01-13	166.105377	-0.004382	0	0	Sell	Sell at 166.11	Buy	Buy at 165.87	0.232223	96.7443302	0.58242744	0.135253	96.87958	100.8301	
08-01-13	165.873154	-0.001398	0	0	Sell	Sell at 165.87	Buy	Buy at 165.4	0.473023	96.8795833	0.58405825	0.276273	97.15586	100.6891	
09-01-13	165.400131	-0.002852	0	0	Sell	Sell at 165.4	Buy	Buy at 165.88	-0.48164	97.1558563	0.58739891	-0.28292	96.87294	100.402	
10-01-13	165.881775	0.002912	1	0	Sell	Sell at 165.88	Buy	Buy at 167.23	-1.35024	96.8729391	0.58398784	-0.78852	96.08442	100.6944	
11-01-13	167.23201	0.0081397	1	0	Sell	Sell at 167.23	Buy	Buy at 165.66	1.573868	96.0844183	0.57455758	0.904278	96.9887	101.514	
14-01-13	165.658142	-0.009411	0	1	Buy	Buy at 165.66	Sell	Sell at 165.55	-0.10319	96.9886961	0.58547497	-0.06042	96.92828	100.5586	
15-01-13	165.554947	-0.000623	0	0	Sell	Sell at 165.55	Buy	Buy at 165.63	-0.07738	96.928278	0.58547497	-0.0453	96.88298	100.496	
16-01-13	165.632324	0.0004674	1	0	Sell	Sell at 165.63	Buy	Buy at 166.54	-0.91165	96.8829757	0.58492795	-0.53325	96.34972	100.5429	
17-01-13	166.543976	0.0055041	1	0	Sell	Sell at 166.54	Buy	Buy at 167.25	-0.70525	96.3497249	0.57852423	-0.408	95.94172	101.0963	
18-01-13	167.249222	0.0042346	1	1	Buy	Buy at 167.25	Sell	Sell at 168.63	1.384613	95.941723	0.57364526	0.794277	96.736	101.5244	
22-01-13	168.633835	0.0082787	1	0	Sell	Sell at 168.63	Buy	Buy at 176.06	-7.43063	96.7359997	0.57364526	-4.26255	92.47345	102.3649	
23-01-13	176.064468	0.0440637	1	1	Buy	Buy at 176.06	Sell	Sell at 175.81	-0.25803	92.4734523	0.52522496	-0.13552	92.33793	106.8755	
24-01-13	175.806442	-0.001466	0	1	Buy	Buy at 175.81	Sell	Sell at 176.28	0.473053	92.3379306	0.52522496	0.248459	92.58639	106.7189	
25-01-13	176.279495	0.0026908	1	0	Sell	Sell at 176.28	Buy	Buy at 176.25	0.034439	92.5863899	0.52522496	0.018088	92.60448	107.006	



Series 1 – represents the curve using the logistic regression model
Series 2 – represents the curve using the benchmark strategy (buy and hold strategy)

For $100 invested at the start of the period at 31st Dec 2012, at the end of 6 months, the logistic regression model predicts a loss of $2.057 (total wealth in hand at the end of 6 months is $97.943)

Whereas the benchmark strategy of buy-and-hold would yield a profit of $0.659 (total wealth in hand at the end of 6 months is $100.659)

Hence, it can be concluded that the logistic regression model performs poorly when compared to the benchmark model and the linear regression model.

Appendix

R Codes

R Code - logistic regression

```
#setwd("C:/Users/keerthy/Documents/Great Lakes/")
trainData=read.csv("log-reg training data.csv")
str(trainData)
```

```
## 'data.frame':    126 obs. of  5 variables:
##  $ Date         : Factor w/ 126 levels "01-08-12","01-10-12",..: 117 4 8 15 20 3
2 36 41 45 50 ...
##  $ Adj_Close    : num  167 167 167 167 163 ...
##  $ Return       : num  0.021839 0.001278 0.000511 -0.003267 -0.019868 ...
##  $ Lagged_return: num  -0.00829 0.021839 0.001278 0.000511 -0.003267 ...
##  $ Actual       : int  1 1 1 0 0 0 0 0 0 1 ...
```

Step0 - Running the logistic regression

```
logit=glm(Actual~ Lagged_return, data = trainData, family = binomial)
summary(logit)
```

```
##
## Call:
## glm(formula = Actual ~ Lagged_return, family = binomial, data = trainData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7472  -1.0809  -0.8688   1.2074   1.5714
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.1658     0.1821  -0.911   0.3625
## Lagged_return  38.4220    18.8575   2.037   0.0416 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 173.88  on 125  degrees of freedom
## Residual deviance: 169.27  on 124  degrees of freedom
## AIC: 173.27
##
## Number of Fisher Scoring iterations: 4
```

Step1 - Overall significance of the Logistic regression model to test applicability

```
library(lmtest)

lrtest(logit)
```

```
## Likelihood ratio test
##
## Model 1: Actual ~ Lagged_return
## Model 2: Actual ~ 1
```

```
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   2 -84.637
## 2   1 -86.939 -1 4.6051    0.03188 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Step2 - McFaden or pseudo r^2 and interpretation

```
library(pscl)
round(pR2(logit)[4],4)
```

```
## McFadden
##   0.0265
```

```
#pR2(logit)
```

Step3- Explanatory power of odds ratio

```
odds_main=data.frame(exp(coef(logit)))
coef_main=data.frame(summary(logit)$coefficients)
write.csv(coef_main, "coef_main.csv")
coef_main
```

```
##                 Estimate Std..Error    z.value   Pr...z..
## (Intercept)   -0.1658304  0.1820946 -0.9106828 0.36246252
## Lagged_return 38.4219672 18.8574550  2.0374948 0.04160048
```

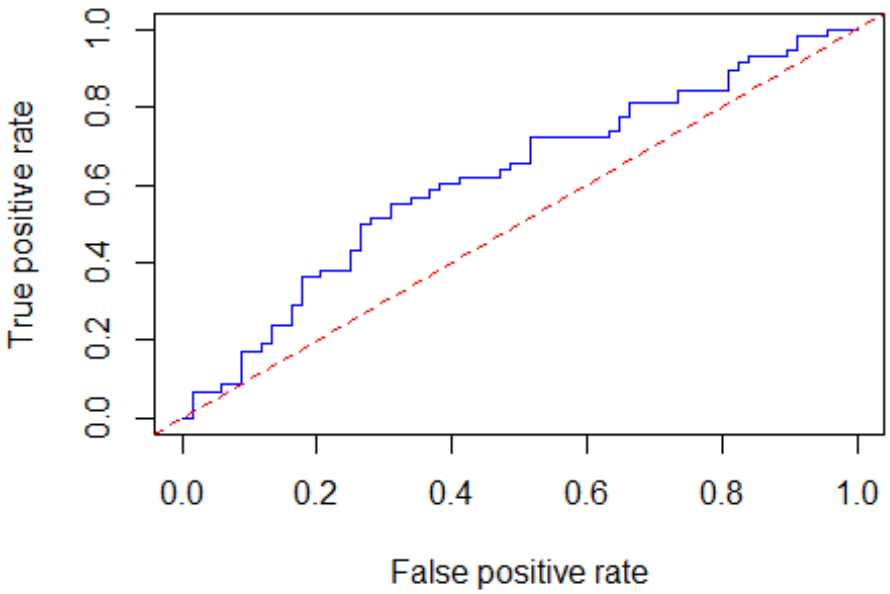
Step4 - Predict for training set and compute confusion matrix (misclassification matrix)

```
library(caret)
library(ROCR)

##Predict using logit model on new training set
trainData$predict.score=predict(logit, trainData, type="response")
trainData$predict.class=floor(predict(logit, type = "response",data=trainData)+0.5)


tabTrain=with(trainData,table(Actual,predict.class))
LR.confmatTrain=confusionMatrix(tabTrain,positive = "1")
LR.confmatTrain
```

```
## Confusion Matrix and Statistics
##
##       predict.class
## Actual  0  1
##      0 53 15
##      1 36 22
##
##                Accuracy : 0.5952
##                  95% CI : (0.5042, 0.6817)
##     No Information Rate : 0.7063
##     P-Value [Acc > NIR] : 0.997119
##
##                   Kappa : 0.1631
##  Mcnemar's Test P-Value : 0.005101
##
```

```
##             Sensitivity : 0.5946
##             Specificity : 0.5955
##          Pos Pred Value : 0.3793
##          Neg Pred Value : 0.7794
##              Prevalence : 0.2937
##          Detection Rate : 0.1746
##    Detection Prevalence : 0.4603
##       Balanced Accuracy : 0.5951
##
##        'Positive' Class : 1
##
```

```
## PLotting the ROCR curve
# library(ROCR)

predTrain=prediction(trainData$predict.score,trainData$Actual)
perfTrain=performance(predTrain,"tpr","fpr")
plot(perfTrain, col="blue")
abline(0, 1, lty = 8, col = "red")
```

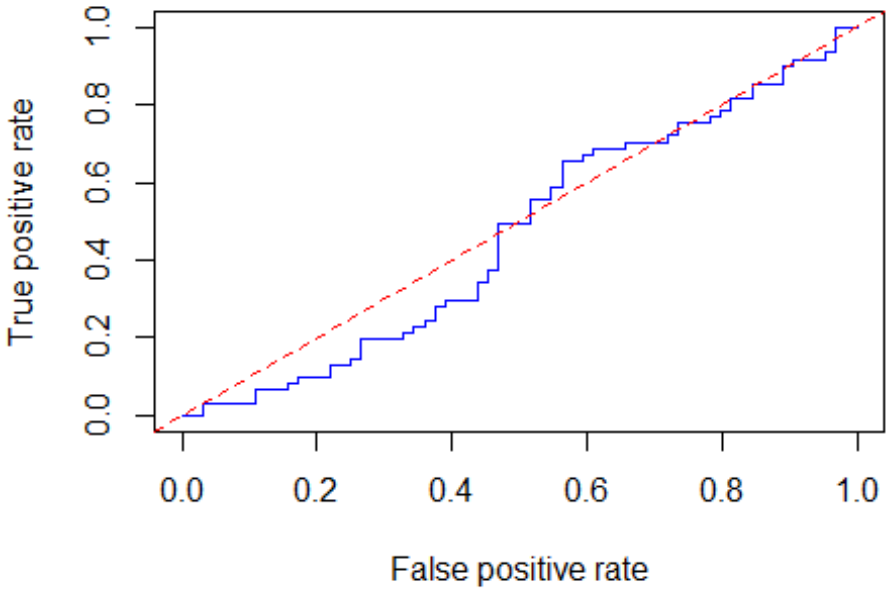



```
## Calculating model performance measures
LR.train.KS <- max(attr(perfTrain, 'y.values')[[1]]-attr(perfTrain, 'x.values')
[[1]])
LR.train.KS

## [1] 0.2429006

LR.train.auc=performance(predTrain,"auc")
LR.train.auc=as.numeric(LR.train.auc@y.values)
LR.train.auc
```

```
## [1] 0.6176471

write.csv(trainData, "log-reg_train_results.csv")
```

Step 5 - Predicting on the test set

```
testdata = read.csv("log-reg test data.csv")

testdata$predict.score=predict(logit,testdata,type="response")
testdata$predict.class=floor(predict(logit, type = "response",newdata=testdata)+0.
5)


tabtest=with(testdata,table(Actual,predict.class))
LR.confmattest=confusionMatrix(tabtest,positive = "1")
LR.confmattest

## Confusion Matrix and Statistics
##
##       predict.class
## Actual  0  1
##      0 36 28
##      1 43 18
##
##                Accuracy : 0.432
##                  95% CI : (0.3437, 0.5236)
##     No Information Rate : 0.632
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : -0.1432
##  Mcnemar's Test P-Value : 0.09661
##
##             Sensitivity : 0.3913
##             Specificity : 0.4557
##          Pos Pred Value : 0.2951
##          Neg Pred Value : 0.5625
##              Prevalence : 0.3680
##          Detection Rate : 0.1440
##    Detection Prevalence : 0.4880
##       Balanced Accuracy : 0.4235
##
##        'Positive' Class : 1
##

predtest_LR=prediction(testdata$predict.score,testdata$Actual)
perftest_LR=performance(predtest_LR,"tpr","fpr")
plot(perftest_LR, col= "blue")
abline(0, 1, lty = 8, col = "red")
```




```
LR.test.KS <- max(attr(perftest_LR, 'y.values')[[1]]-attr(perftest_LR, 'x.values')
[[1]])
LR.test.KS

## [1] 0.0932377

LR.test.auc=performance(predtest_LR,"auc")
LR.test.auc=as.numeric(LR.test.auc@y.values)
LR.test.auc

## [1] 0.466957

write.csv(testdata, "log-reg_test_results.csv")
```

R-Code for Sliding window regression model

```
#set working directory and read the csv file

#setwd("C:/Users/keerthy/Documents/Great Lakes/")
q5data=read.csv("dataset_Q5.csv")
str(q5data)
```

```
## 'data.frame':    250 obs. of  4 variables:
##  $ Date         : Factor w/ 250 levels "01-02-13","01-03-13",..: 236 11 19 35 44
 69 77 87 95 104 ...
##  $ Adj_Close    : num  167 167 167 167 163 ...
##  $ return       : num  0.021839 0.001278 0.000511 -0.003267 -0.019868 ...
##  $ lagged_return: num  -0.00829 0.021839 0.001278 0.000511 -0.003267 ...
```

```
library(AER)
library(xts)
library(foreach)
library(ggplot2)
library(forecast)

X = q5data

#Number of windows and window size
w_size = 120
n_windows = 25

###Rolling Window Loop ###
forecasts = foreach(i=1:n_windows, .combine = rbind)%do%
  {

  # Select data for the training data and the test data

  # The length of the training set is set to be 120 - sliding window
  X_train = X[i:(w_size + i - 1), ]

  # predictions to be made for 5 data points in the test set
  X_test = X[(w_size + i):(w_size + i+4), ]

  # Regression Model
  m1 = lm(return ~ lagged_return, data = X_train)
  f1 = predict(m1, X_test)

  return(f1)
  }
forecasts
```

```
##                    127           128           129           130
## result.1   1.647725e-03  4.323651e-03 -7.878544e-04 -9.642064e-04
## result.2   5.035954e-03 -9.456898e-04 -1.152063e-03 -7.266524e-04
## result.3  -9.276366e-04 -1.119686e-03 -7.238011e-04 -1.803456e-04
## result.4  -1.180480e-03 -7.798563e-04 -2.298960e-04 -4.978556e-04
## result.5  -7.853528e-04 -2.306870e-04 -5.009393e-04  5.705945e-04
## result.6  -7.637183e-05 -3.404649e-04  7.066477e-04  1.656391e-03
## result.7  -2.939404e-04  7.082924e-04  1.617329e-03 -1.434563e-03
```

```
## result.8   8.338239e-04  1.681064e-03 -1.163360e-03  2.609317e-04
## result.9   1.742254e-03 -1.057771e-03  3.442888e-04  5.182342e-04
## result.10 -9.466295e-04  3.575511e-04  5.193533e-04  1.266793e-03
## result.11  2.059017e-04  3.843037e-04  1.208425e-03  1.000709e-03
## result.12  4.623986e-04  1.342077e-03  1.120357e-03  1.826684e-03
## result.13  1.410612e-03  1.192502e-03  1.887331e-03  8.035575e-03
## result.14  1.063183e-03  1.814727e-03  8.464829e-03  3.901953e-06
## result.15  1.126370e-03  5.576183e-03 -8.531626e-05  4.315116e-04
## result.16  1.026738e-02  2.570976e-04  1.170918e-03  5.363600e-04
## result.17  3.049592e-04  1.027844e-03  5.258719e-04 -3.142962e-04
## result.18  1.056823e-03  5.593527e-04 -2.732798e-04  2.717631e-04
## result.19  5.190722e-04 -3.172033e-04  2.302244e-04  1.701586e-04
## result.20 -4.532781e-04  8.449455e-05  2.548818e-05  2.168521e-03
## result.21  2.150319e-05 -3.391838e-05  1.978919e-03 -7.629180e-04
## result.22 -4.806890e-05  1.980191e-03 -7.826542e-04 -4.820238e-04
## result.23  2.077800e-03 -6.545288e-04 -3.572189e-04 -3.013235e-04
## result.24 -6.343076e-04 -3.522358e-04 -2.992052e-04 -5.728552e-04
## result.25 -3.645733e-04 -3.111302e-04 -5.869087e-04  1.860486e-03
##                    131
## result.1  -6.006799e-04
## result.2  -1.426661e-04
## result.3  -4.451359e-04
## result.4   5.645880e-04
## result.5   1.542488e-03
## result.6  -1.532164e-03
## result.7   9.361431e-05
## result.8   4.376353e-04
## result.9   1.321769e-03
## result.10  1.078403e-03
## result.11  1.662427e-03
## result.12  8.076673e-03
## result.13  2.131615e-04
## result.14  7.762844e-04
## result.15  7.262570e-05
## result.16 -5.257214e-04
## result.17  2.356796e-04
## result.18  2.119590e-04
## result.19  2.351667e-03
## result.20 -7.506649e-04
## result.21 -4.645736e-04
## result.22 -4.255041e-04
## result.23 -5.897567e-04
## result.24  1.855651e-03
## result.25 -7.648391e-04

write.csv(forecasts,"sw_forecasts.csv")
```