Download and Import Required modules

In [1]:
```
!pip install alpha_vantage
import pandas as pd
import matplotlib.pyplot as plt
from alpha_vantage.timeseries import TimeSeries
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from alpha_vantage.techindicators import TechIndicators
import seaborn as sb
```

```
Requirement already satisfied: alpha_vantage in c:\users\turtw\anaconda3\lib\site-pa
ckages (2.3.1)
Requirement already satisfied: aiohttp in c:\users\turtw\anaconda3\lib\site-packages
(from alpha_vantage) (3.7.4.post0)
Requirement already satisfied: requests in c:\users\turtw\anaconda3\lib\site-package
s (from alpha_vantage) (2.24.0)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\turtw\anaconda3\lib\site-p
ackages (from aiohttp->alpha_vantage) (1.6.3)
Requirement already satisfied: async-timeout<4.0,>=3.0 in c:\users\turtw\anaconda3\l
ib\site-packages (from aiohttp->alpha_vantage) (3.0.1)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\turtw\anaconda3\lib\s
ite-packages (from aiohttp->alpha_vantage) (5.1.0)
Requirement already satisfied: attrs>=17.3.0 in c:\users\turtw\anaconda3\lib\site-pa
ckages (from aiohttp->alpha_vantage) (20.3.0)
Requirement already satisfied: chardet<5.0,>=2.0 in c:\users\turtw\anaconda3\lib\sit
e-packages (from aiohttp->alpha_vantage) (3.0.4)
Requirement already satisfied: typing-extensions>=3.6.5 in c:\users\turtw\anaconda3
\lib\site-packages (from aiohttp->alpha_vantage) (3.7.4.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\turtw\anaconda3\lib\si
te-packages (from requests->alpha_vantage) (2020.6.20)
Requirement already satisfied: idna<3,>=2.5 in c:\users\turtw\anaconda3\lib\site-pac
kages (from requests->alpha_vantage) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\users\t
urtw\anaconda3\lib\site-packages (from requests->alpha_vantage) (1.25.11)
```

Obtain the Necessary Data using the API

In [2]:
```
# replace with your own API key
kkey = 'YEWIOG54S98KWCOO'
stock = str(input("What stocks are you looking at (please key in the corresponding s


ts = TimeSeries(key = kkey, output_format='pandas')
data, meta = ts.get_daily(stock, outputsize='full')

#clean and set date as index
data = data.reset_index()
data = data.sort_values(by=['date'],ascending = True)
data = data.set_index("date")


#extract closing price and visualise
closingprice = data[['4. close']]


#Same for Bollinger BAnds
ti = TechIndicators(key='YEWIOG54S98KWCOO', output_format='pandas')
bbandsdata, meta = ti.get_bbands(symbol = stock, interval='daily', time_period=20, s
bbandsdata = bbandsdata.reset_index()
bbandsdata = bbandsdata.sort_values(by=['date'],ascending = True)
```

```
bbandsdata = bbandsdata.set_index("date")


#Same for RSI
ti = TechIndicators(key='YEWIOG54S98KWCOO', output_format='pandas')
rsi, meta = ti.get_rsi(symbol = stock, interval='daily', time_period=20, series_type
rsi = rsi.reset_index()
rsi = rsi.sort_values(by=['date'],ascending = True)
rsi = rsi.set_index("date")


closingprice
```

```
What stocks are you looking at (please key in the corresponding stock symbol eg. AAP
L, SPY)  :  SPY
```

Out[2]:

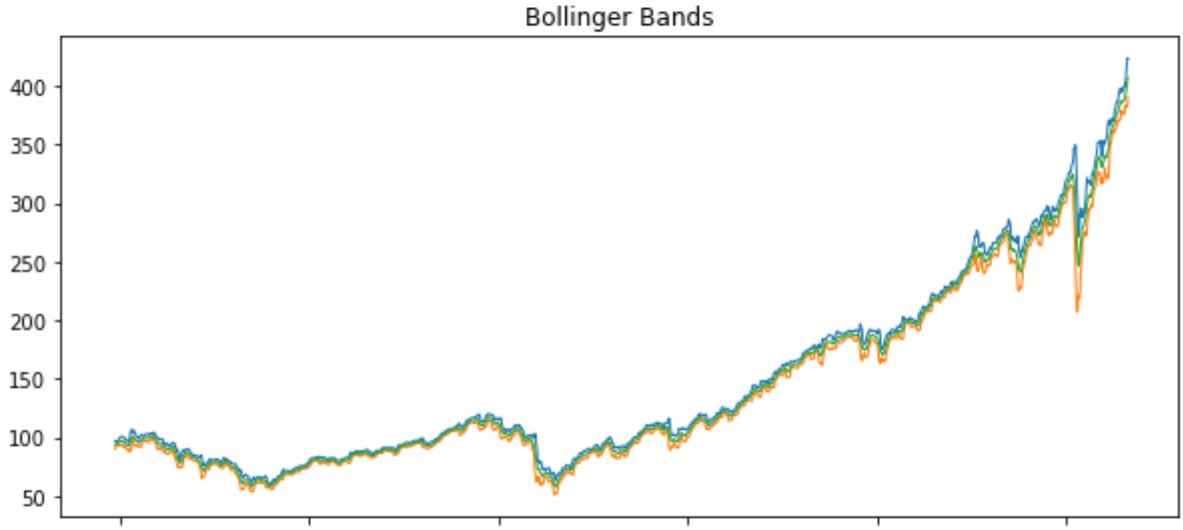
date	4. close
1999-11-01	135.5625
1999-11-02	134.5937
1999-11-03	135.5000
1999-11-04	136.5312
1999-11-05	137.8750
...	...
2021-04-16	417.2600
2021-04-19	415.2100
2021-04-20	412.1700
2021-04-21	416.0700
2021-04-22	412.2700

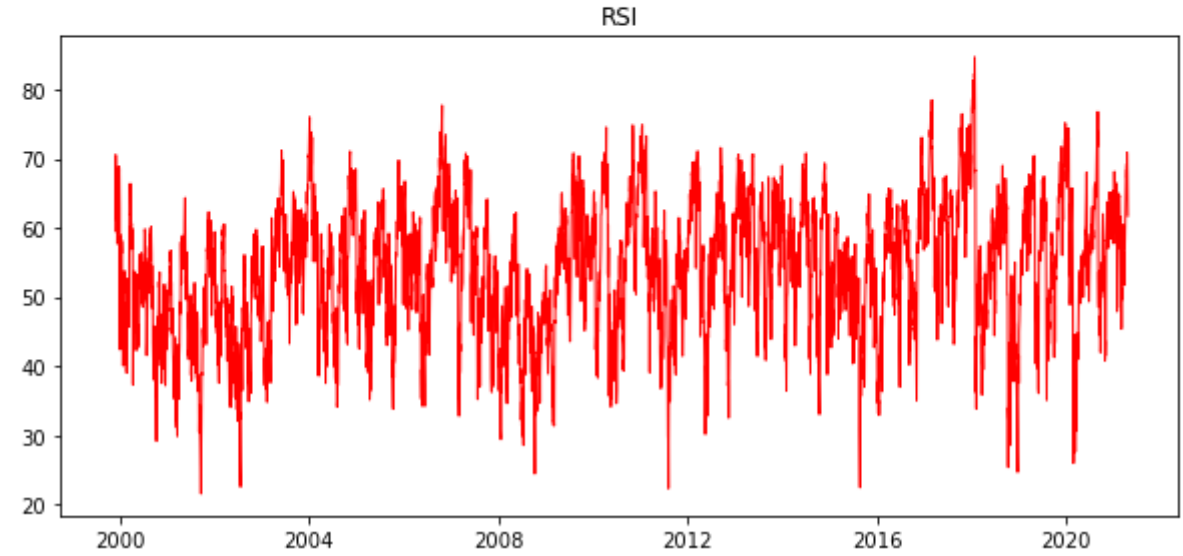
5403 rows × 1 columns

Visualise the Data gain appropriate insights

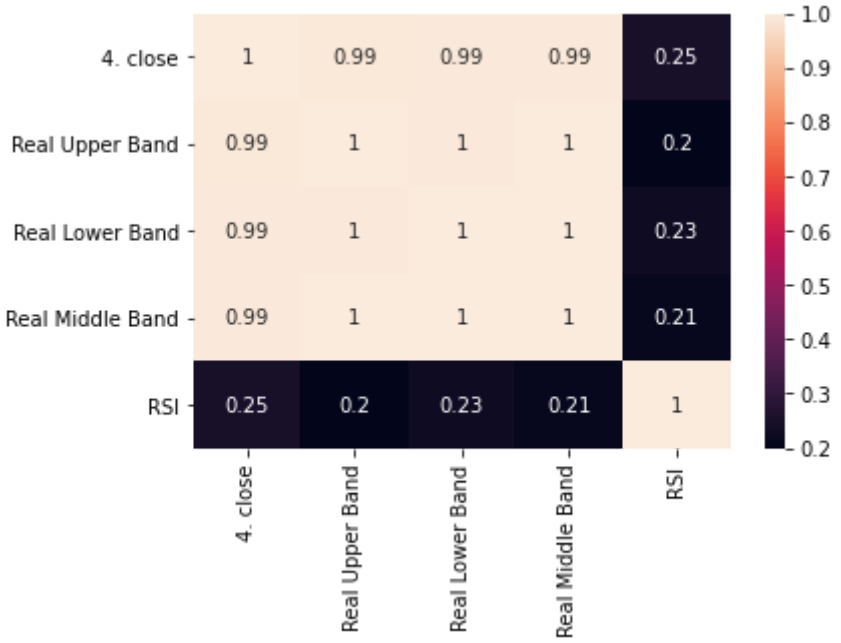
In [3]:
```
#visualising Data
fig, axs = plt.subplots(3, sharex=True, sharey=False, figsize = (10,15))

axs[0].plot(closingprice,linewidth =1)
axs[0].set_title('Wanted Price')
axs[1].plot(bbandsdata, linewidth =1)
axs[1].set_title('Bollinger Bands')
axs[2].plot(rsi,linewidth = 1, color = 'red')
axs[2].set_title('RSI')
```

Out[3]:
```
Text(0.5, 1.0, 'RSI')
```





In [4]:
```
corrMatrix = pd.concat([closingprice,bbandsdata,rsi], axis=1).corr()
sb.heatmap(corrMatrix, annot=True)
plt.show()
```



Function For data Cleaning

In [5]:

```
import datetime as dt
from dateutil.relativedelta import relativedelta

#Function to clean out a singlecolumn information eg, closeprice/openprice in to the
def dataclean(dataframe):

    dataframe['date'] = dataframe.index.date
    dataframe['date'] = pd.to_datetime(dataframe.date,format = "%Y-%m-%d")

    startdate = "2005-01-01"

    startdate = dt.datetime.strptime(startdate, "%Y-%m-%d")

    #startdate = startdate + relativedelta(months=+4)
    enddate =startdate + relativedelta(months=+2) - dt.timedelta(days = 1)

    mask = (dataframe['date']>=startdate) & (dataframe['date']<=enddate)

    a = []
    a.append(dataframe.loc[mask])

    jan = pd.DataFrame(np.row_stack(a))
    jan.rename(columns = {0:"jan",1:"date"},inplace=True)
    jan=jan[["date","jan"]]
    jan = jan.rename(columns={"jan":"2005-01"})
    #print(enddate)


    for x in range(35):

        startdate = startdate + relativedelta(months=+2)
        enddate =startdate + relativedelta(months=+2) - dt.timedelta(days = 1)

        mask = (dataframe['date']>=startdate) & (dataframe['date']<= enddate)


        b = []
        b.append(dataframe.loc[mask])
        feb  = pd.DataFrame(np.row_stack(b))
        feb.rename(columns = {0:"closingprice",1:"date"},inplace=True)
```

```
        feb.drop(['date'], axis='columns', inplace=True)
        jan[x] = feb
        #print("startdate",startdate)
        #print("endndate",enddate)
        jan = jan.rename(columns={x: startdate.strftime("%Y-%m")})
        b = []
        del feb
    jan = jan.drop(jan.columns[0], axis=1)
    jan.dropna(axis=0, how='any', thresh=None, subset=None, inplace=True)

    return(jan)
```

Functions For Linear Regression

In [6]:

```
def linearregression():

    predictionslist =[]
    actuallist = []
    explainedVariance =[]
    lr = LinearRegression()

#getting the X and Y values for fitting and prediction, past 2 months and next 2 mon
    for i in range(len(closeprice.columns)-1):
        X = pd.DataFrame(closeprice.iloc[:,i])
        Y = pd.DataFrame(closeprice.iloc[:,i+1])
        X2 = pd.DataFrame(rsi.iloc[:,i])
        X3 = pd.DataFrame(bbandsdata2.iloc[:,i])

        X = pd.concat([X,X2,X3],axis =1)

        Y2 = pd.DataFrame(rsi.iloc[:,i+1])
        Y3 = pd.DataFrame(bbandsdata2.iloc[:,i+1])
        Z = pd.concat([Y,Y2,Y3],axis =1)
        linreg = lr.fit(X,Y)
        predictionslist.append(linreg.predict(Z))
        explainedVariance.append(linreg.score(X,Y))


    for i in range(2,len(closeprice.columns)):
        actuallist.append(np.array(closeprice.iloc[:,i]))

    return(predictionslist,actuallist,explainedVariance)


def plotpredictions(month,actual,predictions,explainedVariance):
    if(month == 19):
        print("\nAvg Explained Variance R^2 : ",sum(explainedVariance)/len(explained
        print("-----------------------------------------------------------------------
        for i in range(len(actual)):

            plt.title("predictions for: "+str(closeprice.columns[i+1]))
            plt.plot(predictions[i],color = 'red',label = 'prediction')
            plt.plot(actual[i],color = 'blue',label = 'actual')
            plt.legend()
            plt.show()
            print("Explained Variance R^2 : ",explainedVariance[i],"\n")

    else:
        plt.plot(predictions[indexes[month-1]-1],color = 'red',label = 'prediction')
        plt.plot(actual[indexes[month-1]-1],color = 'blue',label = 'actual')
        plt.legend()
        plt.show()
        print("Explained Variance R^2: ",explainedVariance[indexes[month-1]-1],"\n")
```

```
        # Plot the Linear Regression line
        plt.title("Linear Regresssion Line")
        plt.scatter(actual[indexes[month-1]-1], predictions[indexes[month-1]-1], col
        plt.plot(actual[indexes[month-1]-1], actual[indexes[month-1]-1],color = 'r',
```

Function For BackTesting

In [7]:
```
def backtest(closeprice,predictionslist):
    profits = []
    bought = []
    sold = []

    for i in range(len(closeprice.columns)):

        if(i>1):
            #print((closeprice.iloc[:,i-1]).iloc[0])
            #print(predictionslist[i-2][-1])
            if(predictionslist[i-2][-1]-(closeprice.iloc[:,i-1]).iloc[0]>1):
                #print("buy");
                profits2 = (closeprice.iloc[:,i]).iloc[-1] - (closeprice.iloc[:,i-1]
                #print("profits: =",(closeprice.iloc[:,i]).iloc[-1], " - ", (closepr
                profits.append((closeprice.iloc[:,i]).iloc[-1] - (closeprice.iloc[:,
                bought.append(i-1)
                sold.append(i)
    return profits,bought,sold
```

Function for Printing Results

In [8]:
```
def printresult(dates,bought,profits,closeprice):
    indexes =[]
    p=0
    l=0

    for i in range(len(profits)):
      if(profits[i-1]>0):
        p = p+1
      if(profits[i-1]<0):
        l = l+1

    print("\t\t     BackTest Results from 2005 to 2010 :")
    print("\t\t\t\t\t1000 shares per trade")
    print("-------------------------------------------------------------------------
    print("Total Profit From",dates[0],"to",dates[len(dates)-1],": \t","$",1000*np.r
    print("Number of Profits: ", p )
    print("Number of Losses: ", l )
    print("Profit rate:", round(p/(l+p)*100, 3),"%")
    print("\n")

    for i in range(len(profits)):
        print("from",dates[bought[i]], "to", dates[sold[i]])
        print("bought: \t",(closeprice.iloc[:,bought[i]]).iloc[0])
        print("sold:   \t",(closeprice.iloc[:,sold[i]]).iloc[-1])
        print("Profit: \t","$",1000*np.round(profits[i], 3))
        print("\n")
        indexes.append(bought[i])
    return indexes
```

```
def userinput(profits,bought,dates):
    print("Which predictions of trade executed would u like to view:\n")

    for i in range(len(profits)):
        print(i+1,")","from",dates[bought[i]], "to", dates[sold[i]])
        print("\n")
    print(i+2,")","ALL predictions")
    month = int(input())
    return month
```

MAIN

In [9]:
```
#doing data cleaning and transform dataframe to 4months of data/column.
closeprice = dataclean(closingprice)
bbandsdata2 = bbandsdata[['Real Lower Band']]
bbandsdata2 = dataclean(bbandsdata2)
rsi = dataclean(rsi)

#obtaining linear regression results
predictions,actual,explainedVariance = linearregression()
profits,bought,sold = backtest(closeprice,predictions)

dates = list(closeprice.columns)

indexes = printresult(dates,bought,profits,closeprice)
closeprice
```

```
<ipython-input-5-6f7bc370edd7>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  dataframe['date'] = dataframe.index.date
<ipython-input-5-6f7bc370edd7>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  dataframe['date'] = pd.to_datetime(dataframe.date,format = "%Y-%m-%d")
<ipython-input-5-6f7bc370edd7>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  dataframe['date'] = dataframe.index.date
<ipython-input-5-6f7bc370edd7>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  dataframe['date'] = pd.to_datetime(dataframe.date,format = "%Y-%m-%d")
                    BackTest Results from 2005 to 2010 :
                                        1000 shares per trade
--------------------------------------------------------------------------------
---------------------------
Total Profit From 2005-01 to 2010-11 :   $ 71020.0
Number of Profits:  13
Number of Losses:  5
Profit rate: 72.222 %


from 2005-07 to 2005-09
```

```
bought:          119.53
sold:            119.72
Profit:          $ 190.0


from 2005-11 to 2006-01
bought:          120.49
sold:            129.46
Profit:          $ 8970.0


from 2006-05 to 2006-07
bought:          130.4
sold:            129.65
Profit:          $ -750.0


from 2006-09 to 2006-11
bought:          131.42
sold:            141.58
Profit:          $ 10160.0


from 2006-11 to 2007-01
bought:          136.86
sold:            139.5
Profit:          $ 2640.0


from 2007-01 to 2007-03
bought:          141.37
sold:            148.12
Profit:          $ 6750.0


from 2007-03 to 2007-05
bought:          140.51
sold:            150.55
Profit:          $ 10040.0


from 2008-03 to 2008-05
bought:          133.5
sold:            131.19
Profit:          $ -2310.0


from 2008-09 to 2008-11
bought:          127.99
sold:            87.16
Profit:          $ -40830.0


from 2008-11 to 2009-01
bought:          97.11
sold:            75.62
Profit:          $ -21490.0


from 2009-03 to 2009-05
bought:          70.6
sold:            90.12
Profit:          $ 19520.0


from 2009-05 to 2009-07
bought:          87.89
sold:            102.96
Profit:          $ 15070.0
```

```
from 2009-07 to 2009-09
bought:          92.33
sold:            108.08
Profit:          $ 15750.0


from 2009-09 to 2009-11
bought:          100.2
sold:            112.48
Profit:          $ 12280.0


from 2009-11 to 2010-01
bought:          104.32
sold:            110.74
Profit:          $ 6420.0


from 2010-03 to 2010-05
bought:          111.89
sold:            107.42
Profit:          $ -4470.0


from 2010-07 to 2010-09
bought:          102.76
sold:            118.7
Profit:          $ 15940.0


from 2010-09 to 2010-11
bought:          108.46
sold:            125.6
Profit:          $ 17140.0
```

Out[9]:

	2005-01	2005-03	2005-05	2005-07	2005-09	2005-11	2006-01	2006-03	2006-05	2006-07	...	2009-05	2009-07
0	120.3	121.23	116.4	119.53	122.49	120.49	126.7	129.37	130.4	127.8	...	87.89	92.33
1	118.83	121.17	116.6	120.49	122.27	121.75	127.3	129.36	131.38	127.07	...	90.88	89.81
2	118.01	121.22	117.5	119.48	123.7	122.27	127.38	128.76	130.89	127.44	...	90.57	89.8
3	118.61	122.73	117.46	119.95	123.91	122.11	128.44	128.17	131.36	126.61	...	92.14	88.06
4	118.44	122.79	117.09	121.32	123.5	122.23	128.77	127.97	132.52	126.85	...	90.86	88
5	119	122.33	117.82	121.94	124.6	122.23	128.9	128.24	132.36	127.41	...	92.98	88.17
6	118.18	120.97	116.6	122.26	124.35	122.39	129.31	127.38	132.62	126.05	...	91.24	87.96
7	118.57	121.24	117.24	122.43	123.66	123.34	128.8	128.59	132.55	124	...	90.97	90.1
8	117.62	120.39	115.95	122.91	123.21	123.76	128.68	128.83	130.95	123.52	...	88.68	90.61
9	118.24	121.14	115.72	122.84	123.15	123.69	128.33	130.18	129.24	123.34	...	89.44	93.26
10	119.47	120.14	116.8	122.35	123.5	123.24	127.82	130.76	129.5	123.97	...	88.71	93.11
11	118.22	119.12	117.58	123.02	123.09	123.49	128.31	131.03	129.31	125.69	...	91.23	94.13
12	117.5	119.36	118.79	123.44	122.05	124.64	125.97	130.62	126.85	124.83	...	91.12	95.13
13	116.78	118.54	119.29	122.72	120.91	125.13	126.42	130.41	126.21	123.95	...	90.51	95.57
14	116.55	118.1	119.12	123.54	121.34	125.76	126.55	129.59	127.1	126.21	...	89.21	95.55

	2005-01	2005-03	2005-05	2005-07	2005-09	2005-11	2006-01	2006-03	2006-05	2006-07	...	2009-05	2009-07
15	116.88	116.9	119.78	123.19	121.44	126.3	126.66	130.38	126.13	126.66	...	89.02	97.66
16	117.23	117	119.5	123.34	121.58	127.03	127.36	130.11	125.17	126.83	...	91.3	98.06
17	117.43	117.14	119.41	123.79	121.55	127.13	128.54	130.21	126.17	126.71	...	89.67	98.35
18	117.43	117.31	120.05	124.57	121.67	126.23	128.44	130.02	127.73	127.98	...	90.92	97.89
19	118.16	116.53	120.25	123.74	122.66	126.09	127.5	129.22	128.38	127.85	...	92.53	97.65
20	118.91	118.18	119.48	123.65	123.04	125.41	128.39	130.03	126.1	127.22	...	94.77	98.67
21	119.27	117.96	120.5	124.39	122.6	126.69	126.9	129.8	127.51	128.08	...	94.85	98.81
22	118.96	117.43	120.76	124.72	121.22	126.85	126.27	129.83	128.73	128.42	...	93.65	100.44
23	120.23	117.63	120.15	123.72	119.63	126.58	126.6	129.73	129	128.2	...	94.53	100.7
24	120.07	118.19	120.04	122.88	119.2	126.82	125.48	130.56	127.12	127.9	...	94.55	100.41
25	120.21	118.6	120.13	122.65	119.61	126.08	126.62	131.01	126.81	127.41	...	94.16	99.89
26	119.31	119.24	119.91	123.39	118.6	126	126.41	130.87	125.86	126.98	...	94.64	101.2
27	119.74	118	120.48	123.33	118.43	126.33	126.64	129.54	125.75	127.37	...	94.4	100.99
28	120.77	118.09	120.2	123.82	117.5	126.45	126.41	129.74	125.35	127.01	...	94.82	99.73
29	120.68	118.7	120.58	123.06	117.43	127.31	127.75	128.64	123.99	127.11	...	95.08	100.8
30	121.13	117.3	120.86	123.82	118.67	127.81	128.2	128.88	122.55	128.63	...	92.9	101.57
31	121.21	115.77	121.09	122.21	119.11	127.44	129.16	128.71	123.5	129.7	...	91.64	100.79
32	120.23	114.15	121.4	122.2	117.82	126.36	128.81	128.66	126.12	130.03	...	91.55	98.31
33	120.39	114.5	121.36	122.19	119.78	125.71	128.49	130.7	124.65	130.69	...	92.22	99.09
34	118.6	115.41	121.4	122.47	117.67	125.83	129.27	130.95	123.67	130.13	...	92.04	99.96
35	119.45	113.8	121.47	122.47	118.13	126.03	129.08	131.13	124.09	130.12	...	89.28	100.99
36	120.24	116.01	121.57	122.24	119.96	126.69	129.41	131.15	125.01	129.76	...	89.35	102.97
37	121.43	115.57	119.86	121.15	119.72	126.76	129.46	130.91	124.46	129.65	...	90.12	102.96

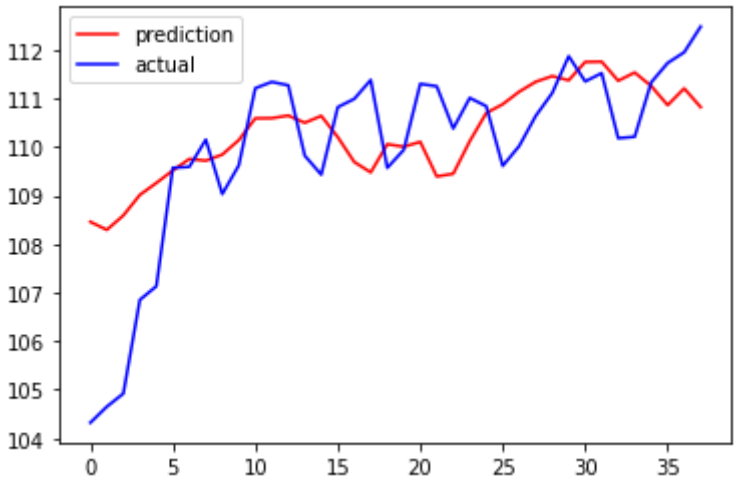
38 rows × 36 columns

In [10]:
```
month = userinput(profits,bought,dates)
```

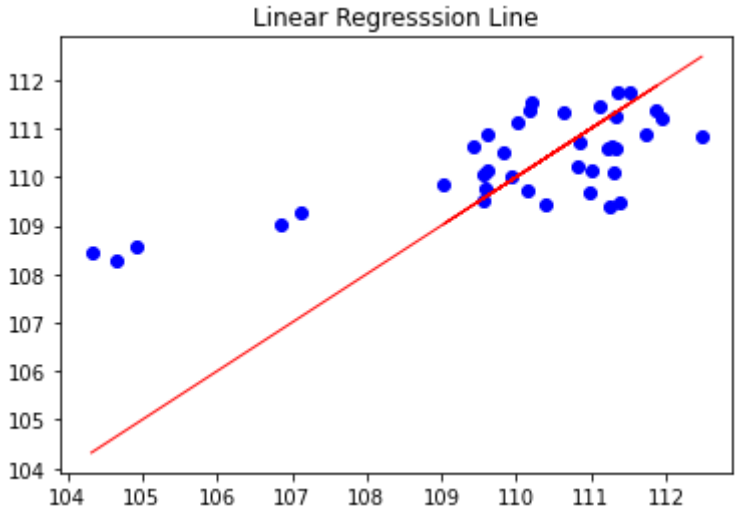
```
Which predictions of trade executed would u like to view:

1 ) from 2005-07 to 2005-09

2 ) from 2005-11 to 2006-01

3 ) from 2006-05 to 2006-07

4 ) from 2006-09 to 2006-11

5 ) from 2006-11 to 2007-01
```

```
6 ) from 2007-01 to 2007-03

7 ) from 2007-03 to 2007-05

8 ) from 2008-03 to 2008-05

9 ) from 2008-09 to 2008-11

10 ) from 2008-11 to 2009-01

11 ) from 2009-03 to 2009-05

12 ) from 2009-05 to 2009-07

13 ) from 2009-07 to 2009-09

14 ) from 2009-09 to 2009-11

15 ) from 2009-11 to 2010-01

16 ) from 2010-03 to 2010-05

17 ) from 2010-07 to 2010-09

18 ) from 2010-09 to 2010-11

19 ) ALL predictions
14
```

In [11]:
```
plotpredictions(month,actual,predictions,explainedVariance)
```



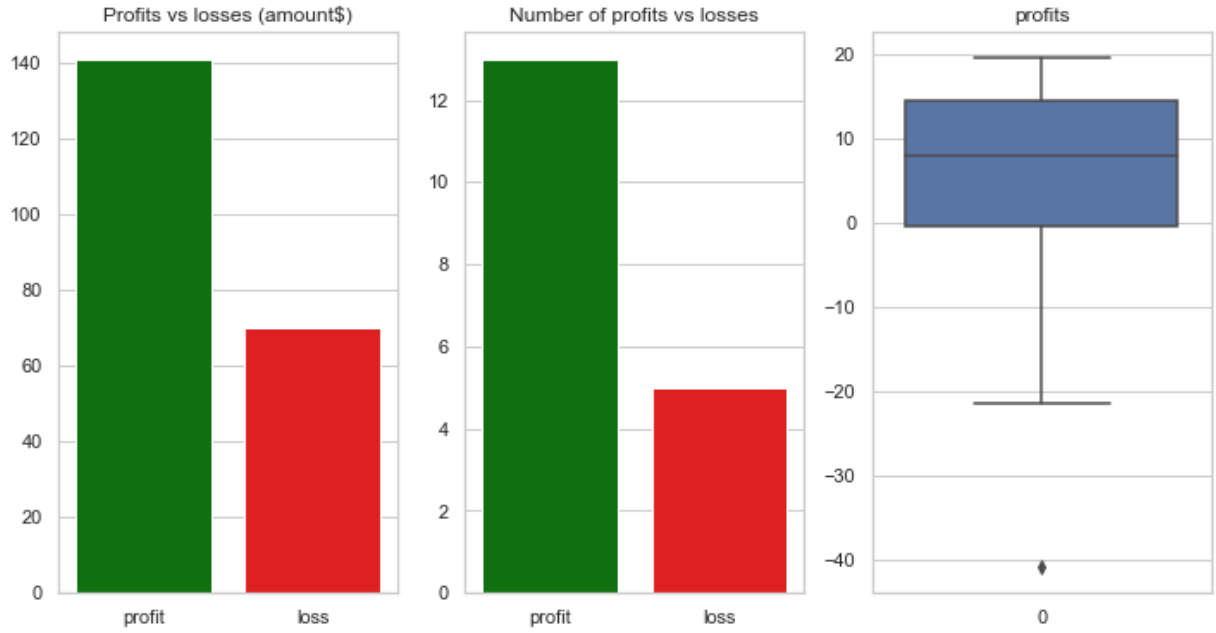

```
Explained Variance R^2:  0.514314512223555
```



In [12]:
```
p=0
l=0
np= 0
nl =0


for i in range((len(profits))):
  if(profits[i]>0):
    p = p+profits[i]
    np = np+1
  if(profits[i]<0):
    l= l+profits[i]
    nl = nl+1
```

In [13]:
```
#just some simple visualisation of results.
yaxis = [p,abs(l)]
y2 = [np,nl]
xaxis=["profit","loss"]


sb.set_theme(style="whitegrid")
clrs = ['green','red']
fig, axes = plt.subplots(1, 3,figsize = (12,6))
axes[0].set_title('Profits vs losses (amount$)')
axes[1].set_title('Number of profits vs losses')
axes[2].set_title('profits')
sb.barplot(ax = axes[0], x = xaxis, y = yaxis,palette = clrs)
sb.barplot(ax = axes[1],x = xaxis, y = y2,palette = clrs)
sb.boxplot(ax = axes[2], data = profits)
```

Out[13]:
```
<AxesSubplot:title={'center':'profits'}>
```


Profits vs losses (amount$)
140
120
100
80
60
40
20
0
profit
loss
Number of profits vs losses
12
10
8
6
4
2
0
profit
loss
profits
20
10
0
−10
−20
−30
−40
0

In [ ]:

In [ ]:

In [ ]: