Trend Analysis of S&P500 Index

Statistical Methods in Finance -- Final Project

- Data: Qisheng Zhang
- Principal Component Analysis: Senyao Han; Qisheng Zhang; Jiayi Zou
- Support Vector Machine: Yang Xu; Suyang Gao
- Time Series: Sijia He, Yufei Feng; Mengyun Zeng
- Presentation: Jiayi Zou, Senyao Han; Yang Xu; Suyang Gao; Mengyun Zeng

1. Introduction

Study
We would like to analyse the future trend of S&P 500 daily log returns so we could get a better understanding about what and how we invest.

Data
The data we used are downloaded using R code from yahoo.finance. Due to the frequently replacement of companies in S&P 500 index, we combine different companies' daily returns together for different time periods to make sure we have as precise data as we could.

For example, on 30th September, 2016, Diamond Offshore Drilling (DO) is replaced by COTY Inc. (COTY). What we did is also remove the records of DO from 09/30/2016 and added COTY's data instead.

2. Objective

Objective
Our major objective is to find out or to estimate the future trend of S&P 500 Index. Also try to predict future returns.

Plan
We planned to use two methods to achieve our goal.

The first method is PCA-SVM. We used Principal Component Analysis to reduce the data dimension so we could better fit it to the SVM model. Then train our updated data with Support Vector Machines and test whether the model is a good fit. After we find the best model we could get, estimate trends of S&P 500 and see if we could get a good prediction of future returns.

The second one is Time Series. For Time Series we did not use PCA to reduce variables, because it could work pretty well with our original data. Other process for Time Series are the same as those described above. After implementing the two models, we made a comparison of which one is doing better.

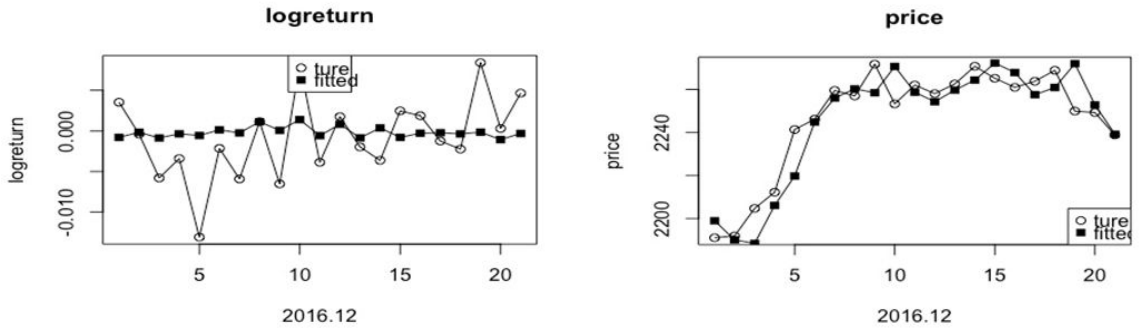
3. Time Series

We use log return value from Jan of 2010 till the end of Dec of 2016 as training data and those in Dec, 2016 as test data. Since it is closest to data for Jan, 2017, which is data to be predicted. At first, we need to check the process is stationary or not. If it's nonstationary we need to differencing the process first. Here we used the Augmented Dickey-Fuller Test. It shows that the p-value of the test is 0.01, smaller than 0.05, so we conclude that there is no unit roots, the process is stationary, and thus we can directly use ARMA model here.

In order to find the optimal bunch of data to build the model, we do the same fitting test but with different proportion of data and use fitting error rate as the selection criteria. According to the results, we find that using data from No. 1573 to No. 1761 can reach the best model. With this part of data, the ARIMA model fitted is:

```
ARIMA(0,0,5) with non-zero mean

Coefficients:
          ma1     ma2      ma3      ma4      ma5    mean
      -0.0488  0.0201  -0.0658  -0.0124  -0.0837  -4e-04
s.e.   0.0237  0.0239   0.0237   0.0239   0.0231   2e-04

sigma^2 estimated as 9.494e-05:  log likelihood=5659.63
AIC=-11305.26   AICc=-11305.2   BIC=-11266.95
```

The plot of the true data together with their fitted value using the ARIMA we find before is:



And we also grab out the predicted results of first three days, 01/01/2017 till 01/03/2017, which should be the most accurate compared to their true value and list them as below:

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
1762	-0.0010561599	-0.01354339	0.01143107	-0.02015374	0.01804142
1763	-0.0003258817	-0.01282795	0.01217619	-0.01944615	0.01879438
1764	-0.0014474129	-0.01395201	0.01105718	-0.02057154	0.01767672

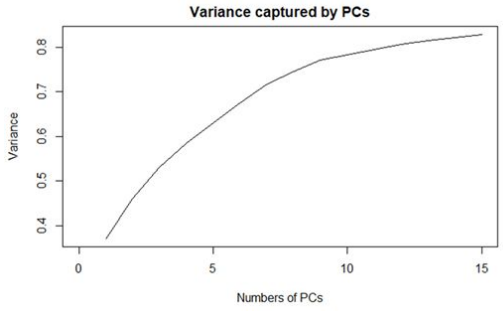
In contrast, we also repeat the similar steps for price value, instead of log return. And after predicting price value with fitted ARIMA model, we change the price into log return and plot it with true value.

As is shown in the plot above, the result is much worse than that of what we did in part 1, which is directly using log return value to build ARIMA model and than forecast. It implies that using time series to predict stock price is unreasonable since that is easily affected by numerous factors and thus cannot be fully decided according to time pattern.

4. PCA

We did the Principal Component Analysis (PCA) of this famous S&P 500 Index. Talking about Principal Component Analysis, we know that PCA is basically done to reduce the dimension of the original data set. In the case of S&P 500 index we have about 500 stocks. It will be difficult to follow all these 500 stocks when we analyze S&P 500. In this case, we are surrounded by data with a large number of variables, some of which might be correlated. This correlation between variables brings about a redundancy in the information that can be gathered by the data set. Thus in order to reduce the

computational and cost complexities, we used PCA to transform the original variables to the linear combination of these variables which are independent.



In the dimension reduction process, we normalized the data before we can apply the Kernel PCA function. This normalization was done to bring all the data to the same scale. The kernel we chose is Polynomial Kernel. The reason we used Kernel PCA because the result data was not well after implementing regular linear PCA. It seems that the data we used is linearly inseparable. Therefore we used an orthogonal transformation to reduce these 500 components to just 15 components. All these components will be orthogonal. First 15 components will be able to explain 80+% variability of the index. The variance is as follows:

5. SVM

Introduction

Formally defined by a separating hyperplane, Support Vector Machine (SVM) is a discriminatively defined classifier, firstly introduced by Boser, Guyon, and Vapnik. SVM algorithm generates an optimal hyperplane that categorizes new forms given labeled training data, which can be applied to both classification problems and regression problems [Alex and Bernhard, 1998]. Instead of using classification approach, we choose regression approach relying on defining the loss function that ignores errors, which are located within a certain range of the true value.

When we apply Support Vector Regression (SVR), first step is to define a function, with at most ε -deviation from the target y. Since only the points outside the ε-region contribute to the final cost, then we do not care about errors as long as they are less than ε [Paul, 2012]. The Support Vector Regression approach is really a robust method since the loss becomes larger and larger as the absolute value of error increase; while sometimes the approach could be error-insensitive since the loss equals zero as the absolute value of error within a given range. We want to compare SVR approach with time series model, the better approach should be the one that gives us a smaller error rate.

Application

The application is conducted by programming in Python code on IPython Notebook. A strong numerical and analytical package called Scikit-learn is implemented which included SVM library. We mainly used one function in this library called SVR. SVR has several parameter which can be defined when we create a classifier (clf) by calling clf = svr(…). There are a number of parameter which can make a multiple ways in analyzing the data.

Data was separated into from date 2010/1/4 to date 2016/12/30 as training data set and 2017/1/1 to 3/31/2017 as test data set. Moreover, we chose last 100 days in our training data as our validation data set to test the performance of different fit models.

Our prediction model is using 15 principal components (PCs) as our input variables vector to predict the response of the next days. Same input could generate whether log-return or price depends on the model we fit. The challenge part is trying to determine how many days should be considered as our data to fit the model. As the table below, we found out that 20-30 days prior is a practical, acquirable interval with high accuracy.

```
Using 10 days to predict, we have accuracy with[ 0.99200162]accuracy
Using 15 days to predict, we have accuracy with[ 0.99061138]accuracy
Using 20 days to predict, we have accuracy with[ 0.98924588]accuracy
Using 25 days to predict, we have accuracy with[ 0.98800977]accuracy
Using 30 days to predict, we have accuracy with[ 0.98646972]accuracy
```

Then, we predict each day's price and log-return using this method, which means using 1 day to 20 days prior to fit the data. Out of 1761 days in the training set, we acquired 1741 of days prediction of both price and log-return. The output is as shown.

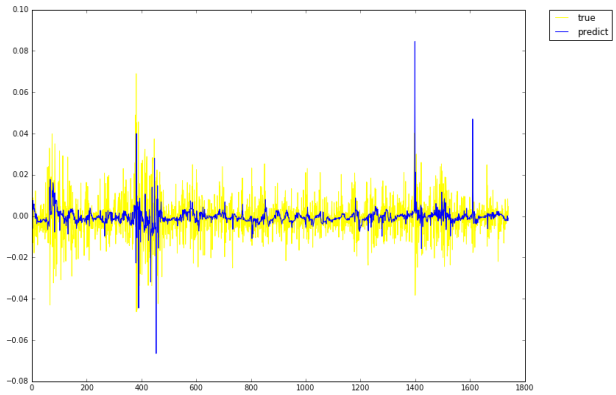


Figure Log-return Model Fit

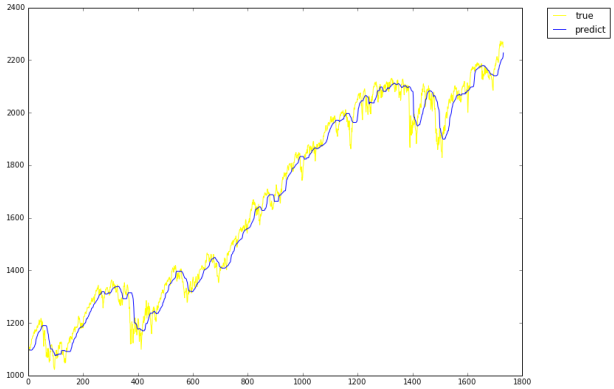


Figure. Price model fit

Even though both log-return and price prediction showed that the prediction value and trend is approximately following the true value which is the yellow background, the exact amplitude is hard to accurately predict. When we try another visualization method in turning all the positive return as red horizontal bar and negative return as the green. We have the plot as below.

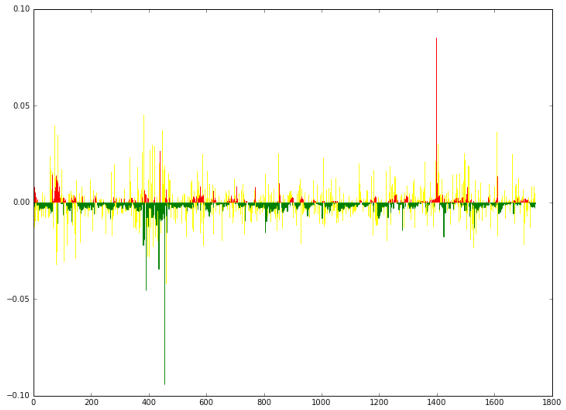


Figure. Log-return model fit in +/- visualization

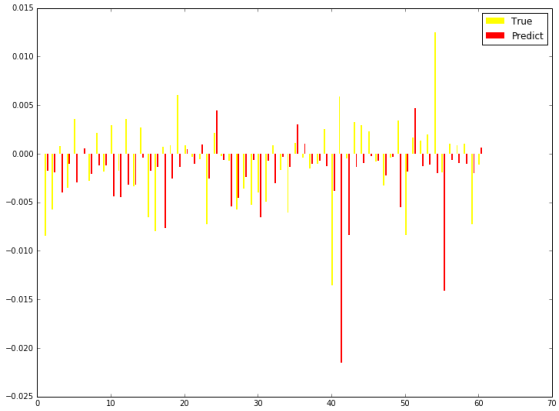


Figure. Prediction of the next 60 days

Therefore, we change our eventual perspective and try to figure out if just making positive or negative prediction is working out. We keep last model which fits the last day of our training model back to 20 days prior. And we predict the new 60 days using the test data with the 60 days of 15 PCs. The output is shown that 33 days out of 60 days in total is accurate prediction. Which gives us around 55% of right prediction.

6. Conclusion

Again, our project mainlys contains 3 parts: Time Series, PCA and SVM. The result of time series in log-return prediction is much worse than price prediction. Basically, the accuracy is not tolerable. It implies that using ARIMA model to predict stock log-return is unreasonable since that is easily affected by numerous factors and thus cannot be fully decided according to time pattern.

As for the part we conducted Kernel PCA to implement dimension reduction of our data and finally got 15 principal components. These 15 principal components we chose can explain about 82% of the whole variance, which means these principal components are feasible to use in SVM.

By comparing the conduction ability, flexibility and error rates we got from SVR approach with time series model, we find the SVR is more reasonable to be use in practical analysis. The SVR not only showed a high accuracy trend in price model fit, but also performed well in log-return prediction and eventually resulted with over 50% of correct prediction. Therefore, we can imply that by given a certain time period, the SVR approach provides a more efficient and precise data analysis for financial stock data.

7. Appendix

Reference:

[1] Alex J. Smola and Bernhard Scholkopf. A Tutorial on Support Vector Regression. NeuroCOLT2 Technical Report Series, NC2-TR-1998-030, October, 1998.

[2] Tutorial slides by Paul Paisitkriangkrai.
http://cs.adelaide.edu.au/~chhshen/teaching/ML_SVR.pdf

```
##########################################################################

### 2010.1.1-2017.3.31


setwd("C:/Users/Qisheng/Desktop/Study/GRAD 2017SP/Stat Method in Finance/Final Project/data2010-2015")

data <- read.csv("SymbolNameSector.csv", header = TRUE)

SP500 <- data$Symbol

SP500 <- as.character(levels(SP500))[SP500]

SP500[67] <- "BRK-B"; SP500[78] <- "BF-B"

#SP500data <-
download.file("http://chart.finance.yahoo.com/table.csv?s=^GSPC&a=0&b=1&c=2010&d=2&e=31&f=2017&g=d&ignore=.csv",  "C:/Users/Qisheng/Desktop/Study/GRAD 2017SP/Stat Method in Finance/Final Project/data2010-2015/SP500data.csv", quiet = F)


address <- paste("http://chart.finance.yahoo.com/table.csv?s=", SP500, "&a=0&b=1&c=2010&d=2&e=31&f=2017&g=d&ignore=.csv", sep = "")

location <- "C:/Users/Qisheng/Desktop/Study/GRAD 2017SP/Stat Method in Finance/Final Project/data2010-2015/"

filelocation <- paste(location,  SP500, ".csv", sep = "")

#for (i in 1:length(SP500)){

#  download.file(address[i], filelocation[i], quiet = F)

#}


SPIA <- read.csv("SP500data.csv", header = TRUE)

VEC <- log(SPIA[-1,]$Close) - log(SPIA[-1824,]$Close)

SPIA <- SPIA[-1824,]

SPIA$Return <- VEC

SPIA <- SPIA[,c(1,8)]

Cname <- c("Date", "S&P500")

colnames(SPIA) <- Cname
```

```
for (i in 1:length(SP500)){
  filename <- paste(SP500[i], ".csv", sep = "")
  SPIA.m <- read.csv(filename, header = TRUE)
  VEC <- log(SPIA.m[-1,]$Close) - log(SPIA.m[-1824,]$Close)
  SPIA.m <- SPIA.m[-1824,]
  SPIA.m$Return <- VEC
  SPIA <- merge(SPIA, SPIA.m, all.x = TRUE)[,c(1:(i+1),i+8)]
  Cname <- c(Cname, SP500[i])
  colnames(SPIA) <-Cname
  print(c(i,'success',dim(SPIA)))
}

write.csv(SPIA, file = "SP500_2010-2017 data_Return.csv")


# create new "variables" as putting two companies together

setwd("C:/Users/Qisheng/Desktop/Study/GRAD 2017SP/Stat Method in Finance/Final Project/data2010-2015")
data <- read.csv("SP500_2010-2017 data_Return.csv", header = T)

countCol <- c()
for (i in 1:ncol(data)) {
  result1 <- sum(is.na(data[,i]))
  if (result1 > 0) {countCol <- c(countCol, i)}
}
names <- colnames(data)
names.NA <- colnames(data[,countCol])
```

```
# download data

setwd("C:/Users/Qisheng/Desktop/Study/GRAD 2017SP/Stat Method in Finance/Final Project/data2010-2015")

SP500_R <- c("FII", "JCP", "PBI", NA, NA, "DO", "CSC", NA, "LM", "TER", NA, "CHDN", "DV", "SNPS", NA, "CSC", NA, NA, "DISH", "SHLD", "RDC", NA, NA, NA, "APO", NA, NA, "NE", "SVU", NA, "GNW", NA, NA, NA, NA, "ITT", "FHN")

SP500_RnoNA <- SP500_R[!is.na(SP500_R)]


address <- paste("http://chart.finance.yahoo.com/table.csv?s=", SP500_RnoNA, "&a=0&b=1&c=2010&d=2&e=31&f=2017&g=d&ignore=.csv", sep = "")

location <- "C:/Users/Qisheng/Desktop/Study/GRAD 2017SP/Stat Method in Finance/Final Project/data2010-2015/"

filelocation <- paste(location,  SP500_RnoNA, ".csv", sep = "")

#for (i in 1:length(SP500_RnoNA)){

#  download.file(address[i], filelocation[i], quiet = F)

#}


SPIA_R <- read.csv("SP500data.csv", header = TRUE)

SPIA_R$Return <- log(SPIA_R$Close) - log(SPIA_R$Open)

SPIA_R <- SPIA_R[,c(1,8)]

Cname <- c("Date", "S&P500")

colnames(SPIA_R) <- Cname


for (i in 1:length(SP500_RnoNA)){

  filename <- paste(SP500_RnoNA[i], ".csv", sep = "")

  SPIA.m <- read.csv(filename, header = TRUE)
```

```
  SPIA.m$Return <- log(SPIA.m$Close) - log(SPIA.m$Open)
  SPIA_R <- merge(SPIA_R, SPIA.m, all.x = TRUE)[,c(1:(i+1),i+8)]
  Cname <- c(Cname, SP500_RnoNA[i])
  colnames(SPIA_R) <-Cname
  print(c(i,'success',dim(SPIA_R)))
}

write.csv(SPIA_R, file = "SP500Replace_data_Return.csv")


# Replace NA's with new data.

table <- read.csv("SP500_2010-2017 data_Return.csv", header = T)
tableN <- read.csv("SP500Replace_data_Return.csv", header = T)
data <- read.csv("SP500_2010-2017 data_Return.csv", header = T)
countCol <- c()
for (i in 1:ncol(data)) {
  result1 <- sum(is.na(data[,i]))
  if (result1 > 0) {countCol <- c(countCol, i)}
}
tableN <- tableN[,c(-1,-2,-3)]
countCol <- countCol[!is.na(SP500_R)]
for (i in countCol) {
  for (j in 1:nrow(table)){
    if (is.na(table[j,i]) == TRUE) {table[j,i] <- tableN[j,which(countCol == i)]}
  }
}

write.csv(table, file = "SP500_2010-2017 data(replaceNA)_Return.csv")
```

```
# Remove all NA columns


table <- read.csv("SP500_2010-2017 data(replaceNA)_Return.csv", header = TRUE)
countCol <- c()
for (i in 1:ncol(table)) {
  result1 <- sum(is.na(table[,i]))
  if (result1 > 0) {countCol <- c(countCol, i)}
}
table <- table[,-countCol]
write.csv(table, file = "SP500_2010-2017 data(replace&RemoveNA)_Return.csv")
```

PCA Codes:

```
library(kernlab)

data<-sp500[,-1]

data<-as.data.frame(data)

#kpc <- kpca(~.,data=data,kernel="rbfdot",kpar=list(sigma=0.2),features=10)

kpc = kpca(~., data=data, kernel="polydot", kpar=list(scale=1, offset=0, degree=2), features=500)

eig(kpc)

pc<-pcv(kpc)[,1:15]

pc<-as.data.frame(cbind(sp500[,1], pc))

colnames(pc)<-c("SP500",
"PC1","PC2","PC3","PC4","PC5","PC6","PC7","PC8","PC9","PC10","PC11","PC12","PC13","PC14","PC15")

e<-eig(kpc)/sum(eig(kpc))

eigen<-e[1:15]

plot(x=1:15, cumsum(eigen), type="l", xlim=c(0,15), xlab="Numbers of PCs", ylab="Variance",
main="Variance captured by PCs")
```

```
install.packages("tseries")
install.packages("TSA")
install.packages("forecast")
library(forecast)
library(TSA)
library(tseries)
data<-read.csv("SP500_logreturn.csv",header = TRUE)
return<-data[,5]
end2016<-return[1:1761]
test<-return[1741:1761]
true<-return[1762:1824]
adf.test(end2016)

error<-NULL
for(i in 1:1700){
  fit<-auto.arima(end2016[i:1761],stepwise = F,approximation = F)
  error[i]<-mean(abs((unlist(fit[18])[(1742-i):(1762-i)]-test)/test))
}
which.min(error)

fit<-auto.arima(end2016[1573:1761],stepwise = F,approximation = F)
plot(c(1:21),end2016[1741:1761],type="o",pch=1,xlab="2016.12",ylab="logretu
rn",main="logreturn")
lines(c(1:21),unlist(fit[18])[(1742-1573):(1762-1573)],type="o",pch=15)
legend("top",c("ture","fitted"),pch=c(1,15))
forecast.Arima(fit,h=3)

#Calculate return by price
data1<-read.csv("SP500.csv",header = TRUE)
price<-data1[,5]
price2016<-price[1:1762]
trueprice<-price[1763:1824]
testprice<-price[1742:1762]
adf.test(price2016)
adf.test(diff(price2016))

errorprice<-NULL
for(i in 1:1700){
  fit<-auto.arima(price2016[i:1762],stepwise = F,approximation = F)
  errorprice[i]<-mean(abs((unlist(fit[18])[(1743-i):(1763-i)]-
testprice)/testprice))
}

fitprice<-auto.arima(price2016[1699:1762],stepwise = F,approximation = F)

plot(c(1:21),price2016[1742:1762],type="o",pch=1,xlab="2016.12",ylab="price"
,main="price")
lines(c(1:21),unlist(fitprice[18])[(1743-1699):(1763-1699)],type="o",pch=15)
legend("bottomright",c("ture","fitted"),pch=c(1,15))
```

```
cal<-unlist(fitprice[18])[(1742-1699):(1763-1699)]
calre<-NULL
for(i in 1:21){
  r<-cal[i+1]/cal[i]-1
  calre[i]<-log(1+r)
}

plot(c(1:21),end2016[1741:1761],type="o",pch=1,xlab="2016.12",ylab="logretu
rn",main="logreturn-calculated",ylim=c(-0.015,0.015))
lines(c(1:21),calre,type="o",pch=15)
legend("top",c("ture","fitted"),pch=c(1,15))

forecast.Arima(fitprice,h=3)
pre<-NULL
pre[1]<-log(2237.215/2362.72)
pre[2]<-log(2237.052/2237.215)
pre[3]<-log(2237.052/2237.052)

pre
```

5261 Project

May 7, 2017

In [3]:
```
## LOG_RETURN PREDICTION
import pandas as pd
import numpy as np

file = pd.read_csv('pc.csv')
data = pd.DataFrame(file)
```

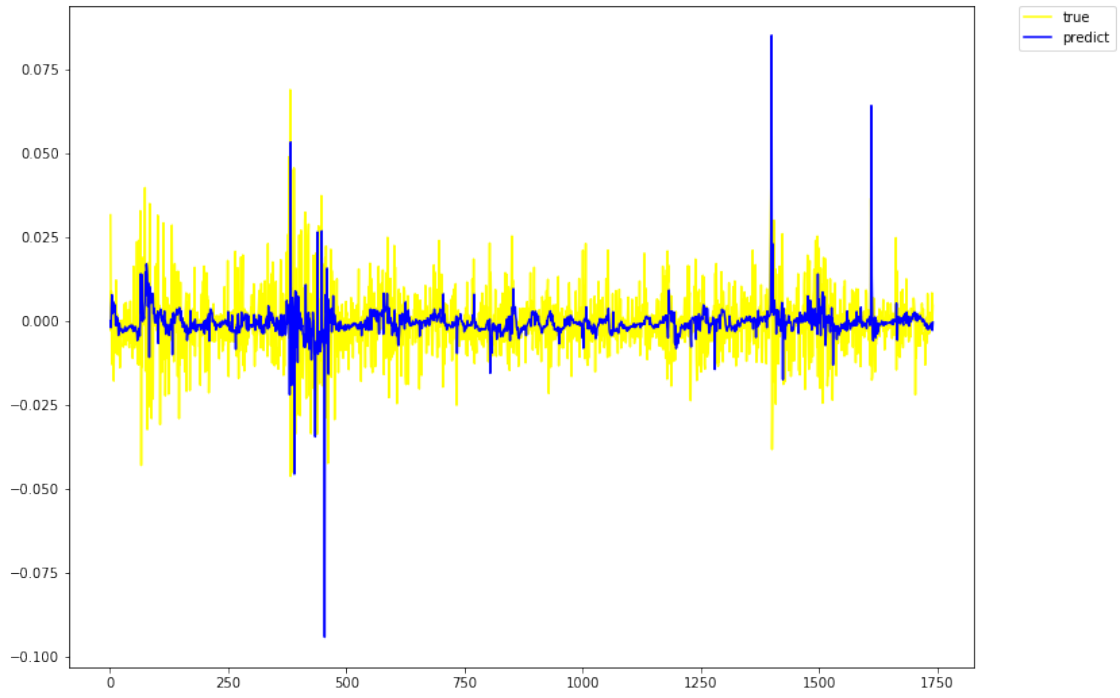
In [4]:
```
## Fit the model in training data set
import warnings
warnings.filterwarnings('ignore')

sum = 0
true = []
predic = []
for m in range(21,1761):
    list = []
    for i in range(m-21, m-1):
        vari = file.loc[i,'PC1':'PC15']
        list.append(vari)
    V = np.array(list)

    list = []
    for j in range(m-20, m):
        lr = file.loc[j, 'SP500']
        list.append(lr)
    L = np.array(list)

    pred = file.loc[m - 1,'PC1':'PC15']

    from sklearn.svm import SVR

    clf = SVR(C = 3, epsilon=0.001, kernel = 'rbf')
    clf.fit(V, L)

    estimate_value = clf.predict(pred)
    predic.append(estimate_value)
    true_value = file.loc[m,'SP500']
```

```
            true.append(true_value)


        length = len(predic)
        print(len(predic) == len(true))
```

```
True
```

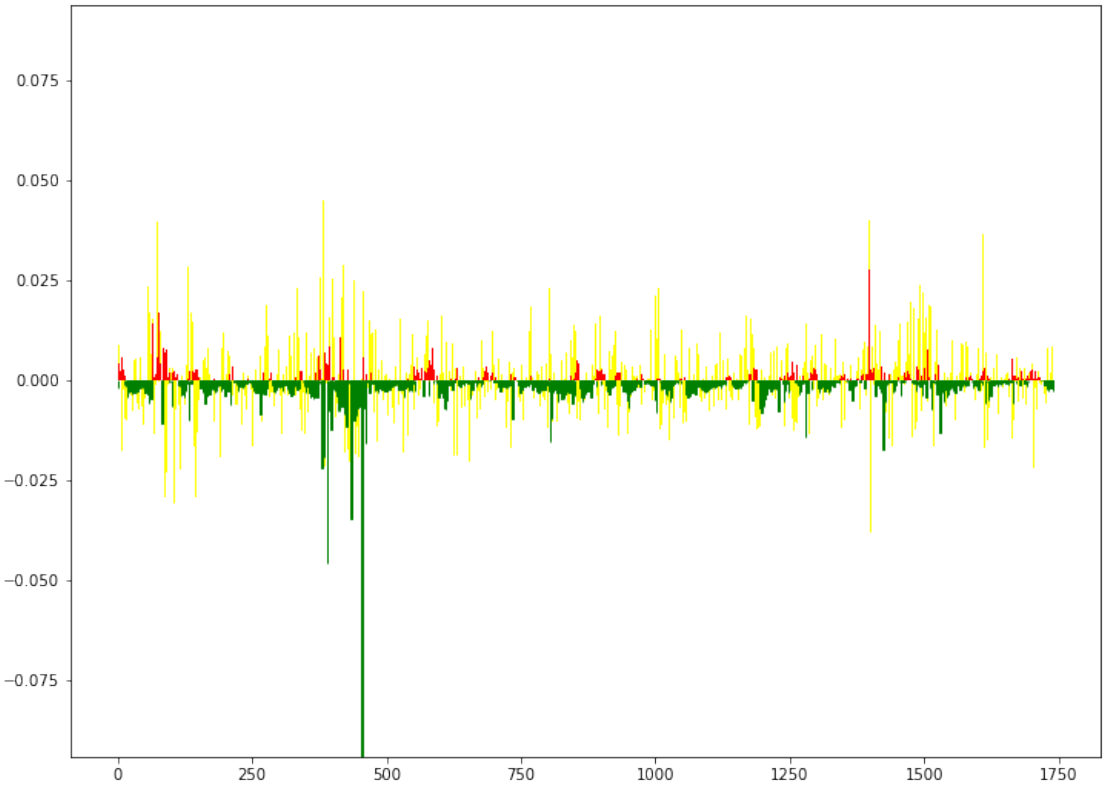
In [5]:
```
        import matplotlib.pyplot as plt
        import numpy as np
        %matplotlib inline

        plt.figure(figsize=(12,9))
        x = np.linspace(1,length, num=length)
        plt.plot(x, true, label='true', c='yellow')
        plt.plot(x, predic, label='predict', c='blue')

        plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[5]: <matplotlib.legend.Legend at 0x4d641d0>



In [6]:
```
        %matplotlib inline
        y = predic
        x = np.linspace(1,length, num=length)
```

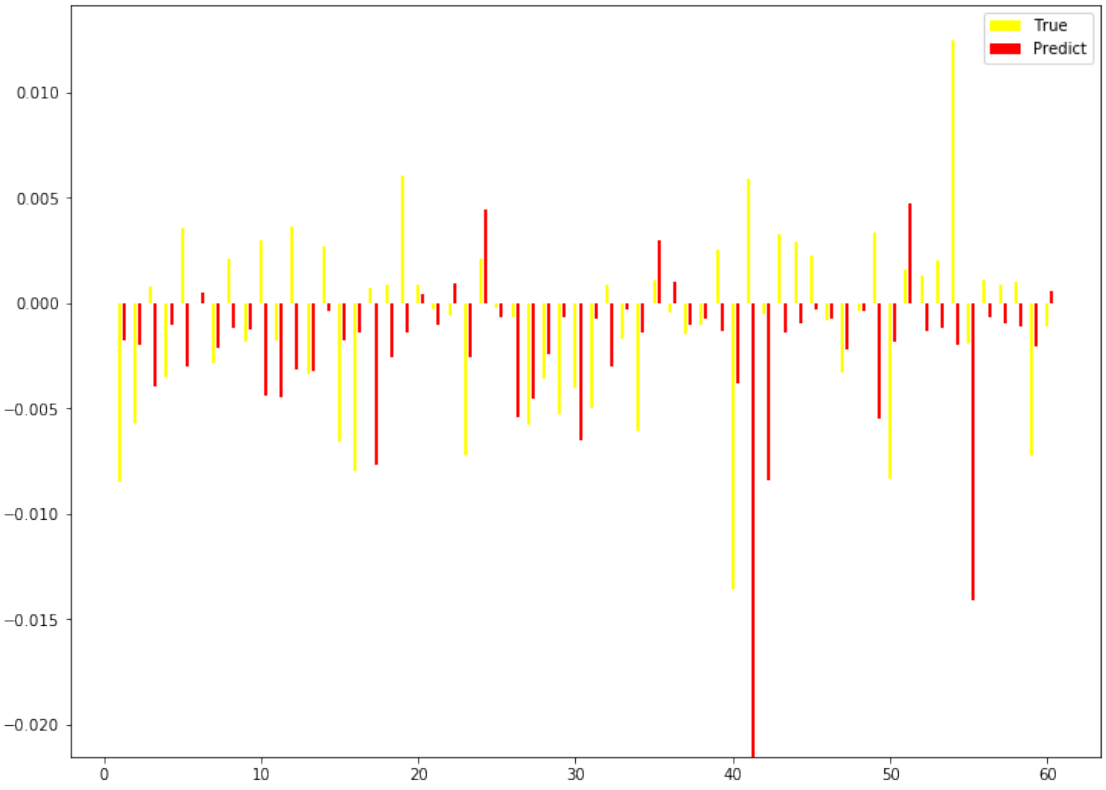
```
width = 1

plt.figure(figsize=(12,9))
plt.bar(x, true, width, label='true', color='yellow',  edgecolor='none')
barlist = plt.bar(x, y, width, color="red", edgecolor='none')
for i in range(0, length):
    if predic[i] <0:
        barlist[i].set_color('green')
```



In [7]:
```
## Prediction for 30 days in test data
days = 60
predic = []
true = file.loc[1761 : 1761 + days - 1, 'SP500']
clf = SVR(C = 200, epsilon=0.0001, kernel = 'linear')
clf.fit(V, L)

for k in range(1761 , 1761 + days):
    pred = file.loc[k - 1,'PC1':'PC15']
    esti = clf.predict(pred)
    predic.append(esti)

width = 0.2
```

```
x = np.linspace(1, days , num = days)
x_1 = np.linspace(1 + 0.3, days + 0.3 , num = days)
plt.figure(figsize=(12,9))
plt.bar(x, true, width, color='yellow', edgecolor='none')
plt.bar(x_1, predic, width, color='red', edgecolor='none')
legend = plt.subplot()
legend.legend(['True','Predict'], loc='upper right')
```

Out[7]: <matplotlib.legend.Legend at 0xc680dd8>



In [8]:
```
## PRICE PREDICTION
import pandas as pd
import numpy as np

file = pd.read_csv('train.csv')
data = pd.DataFrame(file)
```

In [9]:
```
## Find a good interval
import warnings
warnings.filterwarnings('ignore')

##-- Check for the intervals of prior days for prediction
```

```
        for m in range(2,7): ## 10 days to 30 days
            sum = 0
            for w in range(1661, 1761):

                list = []
                for i in range(w -1 - 5*m, w-1):
                    vari = file.loc[i,'PC1':'PC10']
                    list.append(vari)
                V = np.array(list)

                list = []
                for j in range(w - 5*m, w):
                    lr = file.loc[j, 'y']
                    list.append(lr)
                L = np.array(list)

                pred = file.loc[w - 1,'PC1':'PC10']

                from sklearn.svm import SVR

                clf = SVR()
                clf.fit(V, L)

                SVR(C=3.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.001, gamm
                    kernel='linear', max_iter=-1, shrinking=False, tol=0.001, verbo

                estimate_value = clf.predict(pred)
                true_value = file.loc[w,'y']
                error = (estimate_value-true_value)/(true_value)
                sum = sum + (1-abs(error))
            ave = sum/100
            print("Using "+ str(5*m) +" days to predict, we have accuracy with" + s
```

```
Using 10 days to predict, we have accuracy with[ 0.99200162]accuracy
Using 15 days to predict, we have accuracy with[ 0.99061138]accuracy
Using 20 days to predict, we have accuracy with[ 0.98924588]accuracy
Using 25 days to predict, we have accuracy with[ 0.98800977]accuracy
Using 30 days to predict, we have accuracy with[ 0.98646972]accuracy
```

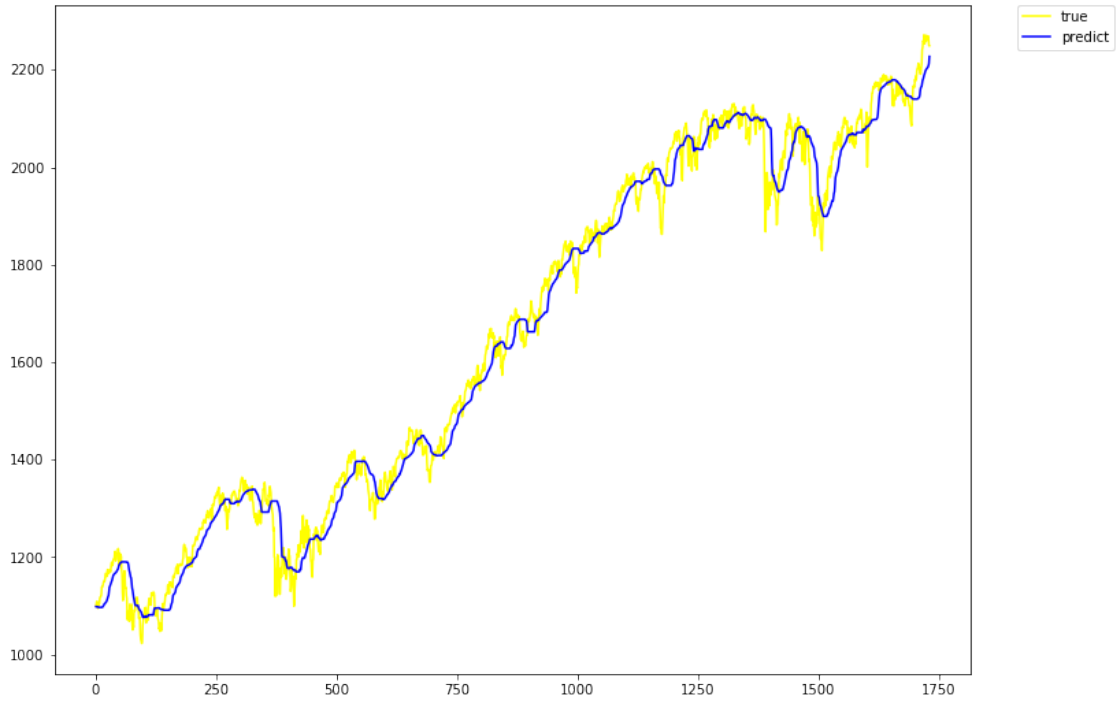
```
In [10]: ## Fit the model
         import warnings
         warnings.filterwarnings('ignore')

         sum = 0
         true = []
         predic = []
```

```
        for m in range(30,1761):
            list = []
            for i in range(m-30, m):
                var = file.loc[i,'PC1':'PC10']
                list.append(var)
            V = np.array(list)

            list = []
            for j in range(m-30, m):
                close = file.loc[j, 'y']
                list.append(close)
            Close = np.array(list)

            pred = file.loc[m,'PC1':'PC10']

            from sklearn.svm import SVR

            clf = SVR(C = 3, epsilon = 0.001, kernel = 'rbf')
            clf.fit(V, Close)

            #SVR(C=100.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.001, gamm
            #    kernel='rbf', max_iter=-1, shrinking=False, tol=0.001, verbose=Tr

            estimate_value = clf.predict(pred)
            predic.append(estimate_value)
            true_value = file.loc[m,'y']
            true.append(true_value)


        length = len(predic)
        print(len(predic) == len(true))
```

```
True
```

In [11]:
```
        import matplotlib.pyplot as plt
        import numpy as np
        %matplotlib inline

        plt.figure(figsize=(12,9))
        x = np.linspace(1,length, num=length)
        plt.plot(x, true, label='true', c='yellow')
        plt.plot(x, predic, label='predict', c='blue')
        plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

Out[11]:
```
<matplotlib.legend.Legend at 0xfc92f60>
```


true
predict
2200
2000
1800
1600
1400
1200
1000
0
250
500
750
1000
1250
1500
1750