Final Project of ML Finance

Bijie, Kai, Kus, Li

What problem we solved?

Hybrid ARIMA & SVM revisited

- Problem to solve
    - To achieve better prediction using the hybrid method
- Perfecting the original paper
    - Removed mysterious model 3, arima + svm
    - Created consistent tables
    - Used a different data set to replicate result
    - Used only one error Measurement (mean squared error)

Closing Price from these 8 companies

1) American National Insurance Company
2) Citi Group
3) General Electric Company
4) General Motors Company
5) JPMorgan Chase & Co
6) Eastman Kodak Company
7) Phillips Morris International Inc.
8) AT&T Inc.

Training Period	Testing Period
Jan 1st ~ 11th, 2015	Jan 12th ~ March 31st, 2015

Data Set

What we Did?

Models

ARIMA

$$y_t = \theta_0 + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots + \varphi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q}$$

SVR

$$y = w\varphi(x) + b$$

$$K(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$$

$$R(\alpha_i - \alpha_i^*) = \sum_{i=1}^{N} d_i(\alpha_i - \alpha_i^*) - \varepsilon \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i^*) \times (\alpha_j - \alpha_j^*) K(x_i, x_j)$$

Language, Platform, and Toolkits

jupyter

python

NumPy

scikit learn

PANDAS

matplotlib

Program - ARIMA

```
X = series.values
#size = int(len(X) * 0.995)
size = 100
#train, test = X[0:size], X[size:len(X)]
train = X[0:10]
test = X[10:size]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
    model = ARIMA(history, order=(1,0,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
# plot
pyplot.plot(test, color='blue')
pyplot.plot(predictions, color='red')
pyplot.show()
```

Create ARIMA model!

Fits ARIMA(p,d,q) by exact maximum likelihood via Kalman filter.

Forecast values based on history

Program - SVR

```
size = 100;
#train, test = series[0:size], series[size:len(series)]
train = series[0:10]
test = series[11:size]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
    test_data = np.arange(len(history),len(history)+1)
    test_data = np.expand_dims(test_data,axis=1)
    train_data = np.arange(0,len(history))
    train_data = np.expand_dims(train_data,axis=1)
    svr = SVR(kernel='rbf', C=1e3, gamma = 1/1250)
    yhat = svr.fit(train_data,history).predict(test_data)
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)

plt.plot(test, color='blue')
plt.plot(predictions, color='red')
plt.show()
```

Create SVR model!

Fit and predict

Program - Hybrid

ARIMA

Outlier

SVR

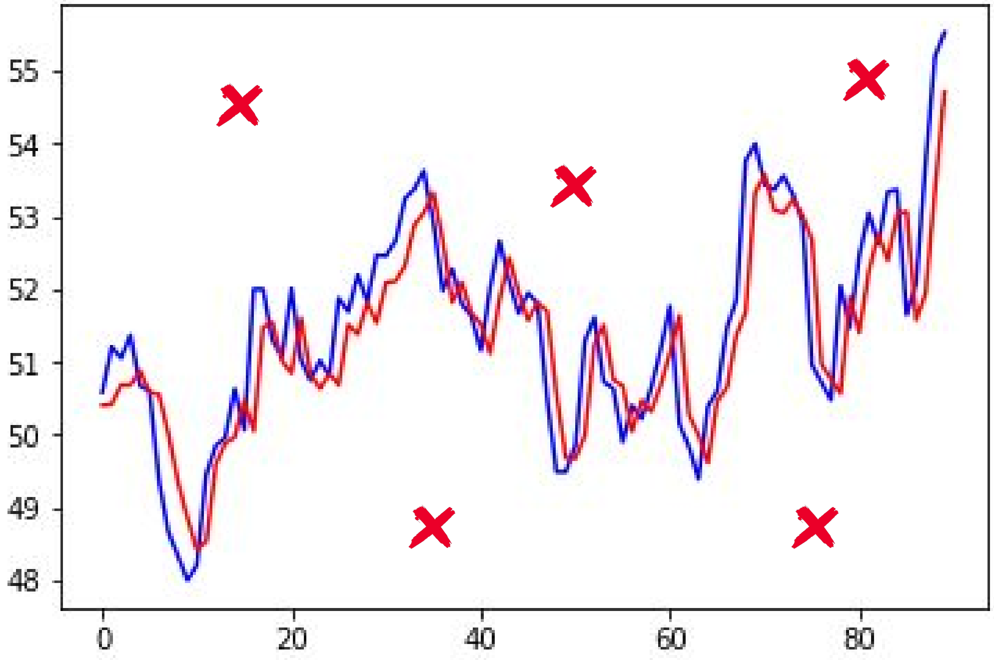
Programs - Hybrid

```
for t in range(start_point,len(train)):
    model = ARIMA(history, order=(0,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    if abs(yhat-train[t]) > epsilon:
        outlier.append(train[t])#if a point
    else:
        predictions.append(yhat)
        true_data.append(train[t])
        obs = train[t]
        history.append(obs)
```

Outlier and threshold

```
for t in range(1, len(outlier)):
    predicted_data = np.arange(len(history),len(history)+1)
    predicted_data = np.expand_dims(predicted_data,axis=1)
    train_data = np.arange(0,len(history))
    train_data = np.expand_dims(train_data,axis=1)
    svr = SVR(kernel='rbf', C=1e3, gamma = 1/1250)
    yhat = svr.fit(train_data,history).predict(predicted_data)
    true_data.append(outlier[t])
    predictions.append(yhat)
    obs = outlier[t]
    history.append(obs)
```

Programs - Hybrid



ARIMA Model

What are our results?

Experiment Settings :

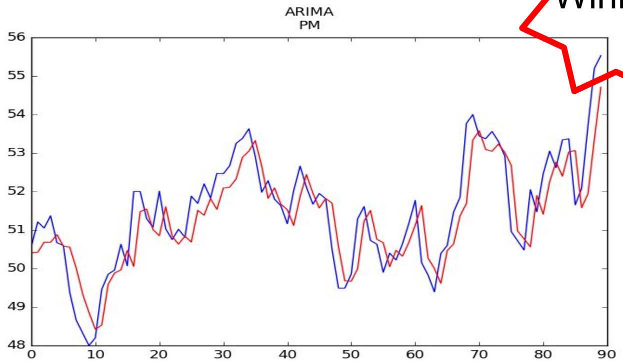
- Training & Testing: For training dataset, we use first 10 day for training and 90 day for testing

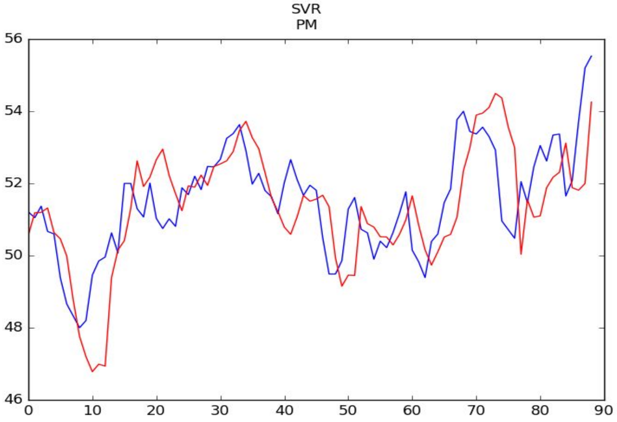
- Data Sets: Eight datasets in January 1st, 2015 to March 31st, 2015

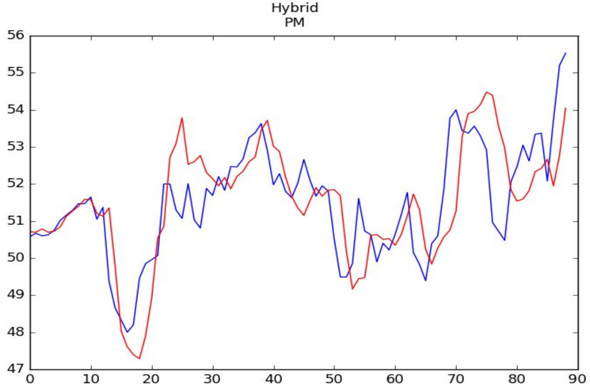
- Evaluation Metrics: We use MSE (mean squared error) of testing data to evaluate the accuracy of the model

Result PM:

Winner

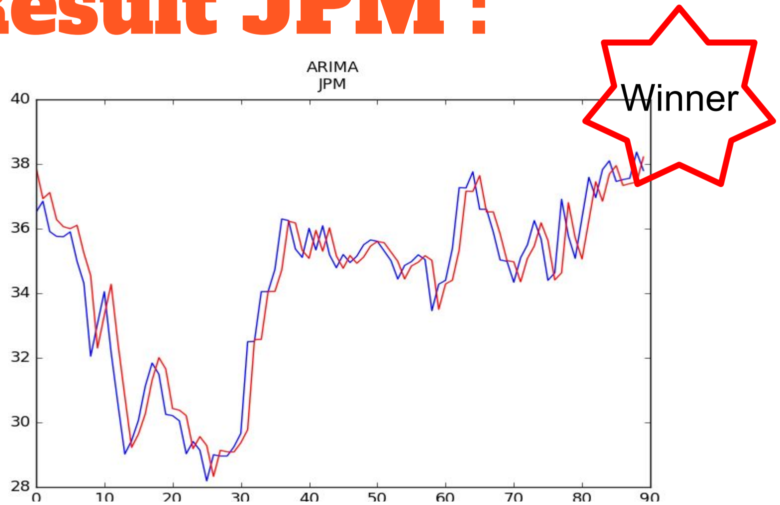


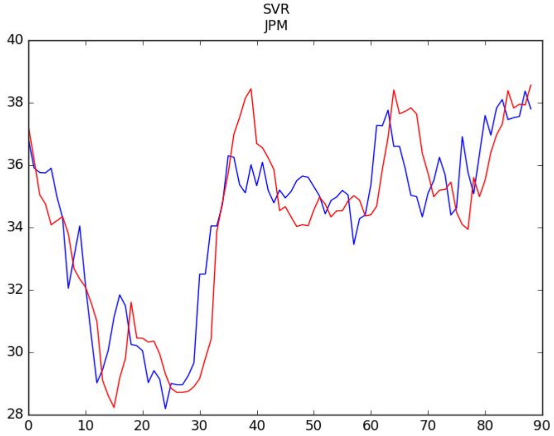


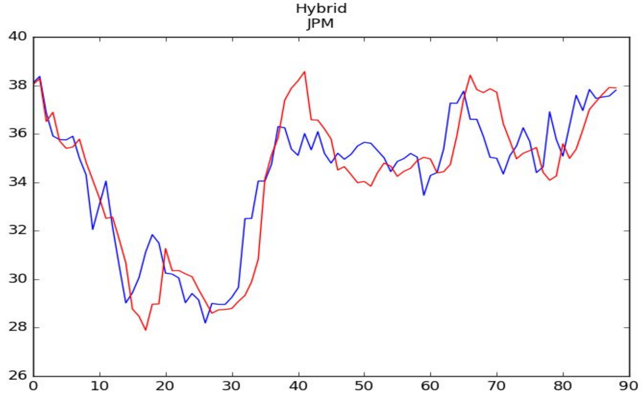


ARIMA	SVM	Hybrid
0.617	1.704	1.443

Result JPM :

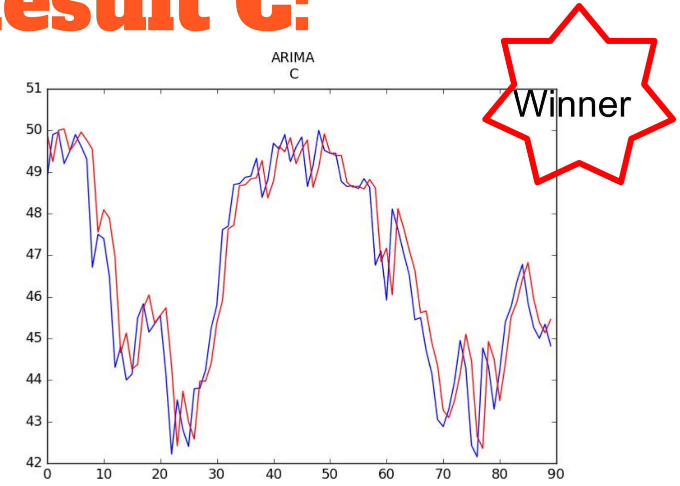


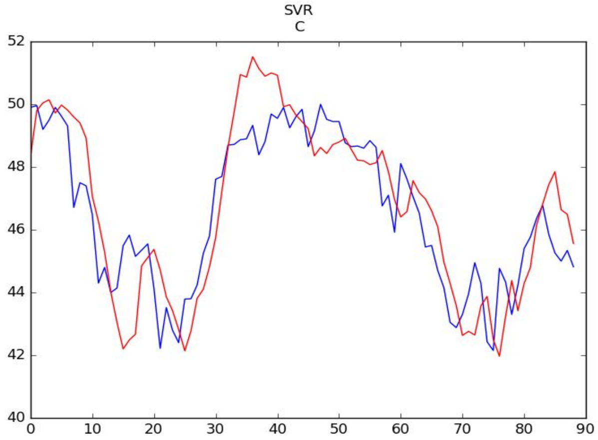


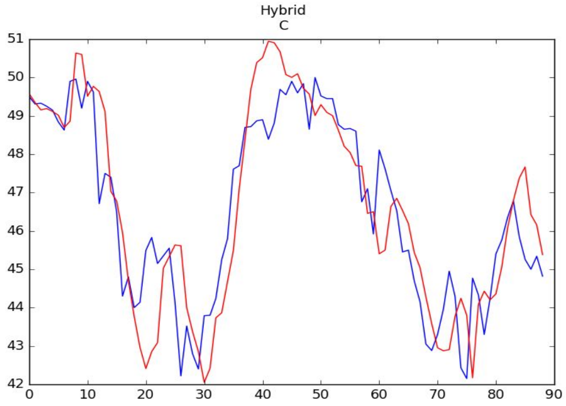


ARIMA	SVM	Hybrid
0.837	1.972	1.955

Result C:







ARIMA	SVM	Hybrid
0.863	1.819	1.609

What's different from the original?

Our MSE Results:

	ARIMA	SVM	Hybrid
PM	0.617	1.704	1.443
T	0.566	0.700	0.656
KODK	0.713	2.971	2.398
JPM	0.837	1.972	1.955
GM	0.342	1.031	0.946
GE	0.714	1.362	1.131
C	0.863	1.819	1.609
ANAT	1.215	2.133	1.867

The MSE From Paper

	ARIMA	SVM	Hybrid
PM	0.187	0.187	0.110
T	0.426	0.425	0.589
KODK	0.225	0.224	0.213
JPM	0.128	0.127	0.112
GM	0.374	0.318	0.204
GE	0.135	0.133	0.132
C	0.3078	0.333	0.262
ANAT	1.130	1.112	1.002

Differences & Discussion

As we can see, we can not replicate the result. In our experiment, most of the time, the ARIMA model yields the best result. Hybrid model performs better than SVM, but worse than ARIMA. This may due to following reasons:

1) Different data period. We are using datasets within three years instead of data ten years ago
2) Different market condition. The market condition might change during this period

3) “Efficient Market Hypothesis” at work.