# An Intelligent Predictor System for Trading

Kantichai Rujitrakarnchotikul 57010127

Nutchaphut Suebbuk 57010393

Toranun Siridontanakasame 57010597

Metasit Rinthon 57011026

# Introduction

- ที่มา
  - ตั้งแต่อดีตจนถึงปัจจุบัน การซื้อ-ขายสินทรัพย์ทางการเงินในตลาดหลักทรัพย์ ตลาดอนุพันธ์ และอื่นๆ ถือเป็นช่องทางหนึ่งในการลงทุนเพื่อแสวงหาผลกำไรที่นิยมในหมู่นักลงทุน เนื่องจากตัวเลขผลกำไรที่อาจจะสูง จึงดึงดูดนักลงทุนหน้าใหม่ให้เข้ามาในตลาดหลักทรัพย์ ซึ่งนักลงทุนหน้าใหม่นั้นอาจจะมีความรู้ในตลาดหลักทรัพย์ที่น้อยจึงอาจจะสูญเสียเงินในการลงทุนได้
- วัตถุประสงค์
  - มีเครื่องมือที่ช่วยนักเทรดทำนายค่าว่าเป็นขาขึ้นหรือขาลงตาม Model ต่างๆ ช่วยในการตัดสินใจของนักเทรดโดยตีออกมาเป็นแนวโน้มความน่าจะเป็น(ขึ้น ลง เท่าเดิม) และเปอร์เซ็นต์ความเปลี่ยนแปลงของราคา

# Introduction (ต่อ)

- ลักษณะ data ที่ต้องการ
  - ข้อมูลหุ้น 50 ตัวที่มีผลประกอบการดี มี market cap และสภาพคล่องที่สูงเนื่องจากมีแนวคิดว่าหุ้นประเภทดังกล่าวเกิดจากความต้องซื้อและขายจริง ซึ่งควรจะมีการเคลื่อนไหวที่เป็นรูปแบบ
- ลักษณะ data ที่ใช้
  - เป็นไฟล์ประเภท CSV
  - รายชื่อหุ้นจาก SET 50 อันดับแรก (SET50)
  - มีข้อมูลย้อนหลัง ตั้งแต่วันที่ 1 มกราคม 2560 จนถึงวันที่ 15 พฤษภาคม 2561

# Data

- แหล่งที่มาของข้อมูล + Link
  - http://siamchart.com/
- ลักษณะของข้อมูล
  - เป็นไฟล์ CSV ที่มี attribute ticker, date, open, high, low, close และ volume
- Data cleaning
  - สำหรับหุ้นที่เพิ่งเข้าตลาดใหม่ ในช่วงเวลาก่อนหน้านั้น จะแทนที่ด้วยราคาเดียวกันกับวันแรกที่เข้าตลาด
- Data transformation
  - เมื่อ open nextday **มากกว่า** close today แทนค่าเป็น UP หรือ 1
  - เมื่อ open nextday **น้อยกว่า** close today แทนค่าเป็น DOWN หรือ -1
  - เมื่อ open nextday **เท่ากับ** close today แทนค่าเป็น DRAW หรือ 0

ตัวอย่างข้อมูลดิบ SCB

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | DATE | OPEN | HIGH | LOW | CLOSE | VOL |
| 2 | 20170104 | 154 | 156 | 153.5 | 156 | 11832000 |
| 3 | 20170105 | 156.5 | 157.5 | 155.5 | 156.5 | 8183500 |
| 4 | 20170106 | 156.5 | 157 | 155 | 156.5 | 3712600 |
| 5 | 20170109 | 156.5 | 158 | 155.5 | 156.5 | 6420200 |
| 6 | 20170110 | 156 | 158.5 | 156 | 157.5 | 11395000 |
| 7 | 20170111 | 158 | 158.5 | 157 | 157 | 6406500 |
| 8 | 20170112 | 158.5 | 158.5 | 154.5 | 155.5 | 7774000 |
| 9 | 20170113 | 156.5 | 158.5 | 155.5 | 157.5 | 6183100 |
| 10 | 20170116 | 157 | 157 | 155 | 155 | 5236700 |
| 11 | 20170117 | 155 | 156.5 | 153.5 | 155 | 8864100 |
| 12 | 20170118 | 155.5 | 156.5 | 155 | 156 | 4313600 |
| 13 | 20170119 | 155 | 155.5 | 153 | 155 | 6844800 |
| 14 | 20170120 | 151.5 | 152 | 148.5 | 150.5 | 28187500 |
| 15 | 20170123 | 151 | 152 | 149.5 | 150.5 | 8783700 |
| 16 | 20170124 | 150 | 151.5 | 150 | 150.5 | 7154200 |
| 17 | 20170125 | 151 | 153 | 150.5 | 152 | 16493300 |
| 18 | 20170126 | 153.5 | 154 | 152.5 | 153.5 | 9258000 |
| 19 | 20170127 | 153.5 | 153.5 | 152.5 | 152.5 | 4390400 |
| 20 | 20170130 | 152.5 | 153 | 152 | 152.5 | 3537000 |
| 21 | 20170131 | 152 | 152 | 150.5 | 151 | 7703200 |
| 22 | 20170201 | 150.5 | 153 | 150.5 | 152 | 6969200 |
| 23 | 20170202 | 152 | 153 | 151 | 151 | 6529100 |
| 24 | 20170203 | 151 | 153.5 | 151 | 153 | 6597700 |
| 25 | 20170206 | 154 | 155 | 153.5 | 154.5 | 9901200 |
| 26 | 20170207 | 154.5 | 155 | 153.5 | 154 | 3802600 |
| 27 | 20170208 | 153.5 | 154.5 | 153 | 154 | 5747300 |
| 28 | 20170209 | 154.5 | 156 | 153.5 | 154 | 6157400 |
| 29 | 20170210 | 154.5 | 155 | 154 | 154 | 9620500 |
| 30 | 20170214 | 155 | 156 | 153 | 154 | 9754000 |
| 31 | 20170215 | 154.5 | 155 | 153 | 153.5 | 4413500 |
| 32 | 20170216 | 154 | 154.5 | 153.5 | 154 | 4592500 |
| 33 | 20170217 | 154 | 155 | 153.5 | 154 | 6132300 |
| 34 | 20170220 | 154 | 154.5 | 153 | 154 | 6282600 |
| 35 | 20170221 | 153.5 | 154 | 152 | 152.5 | 5458500 |

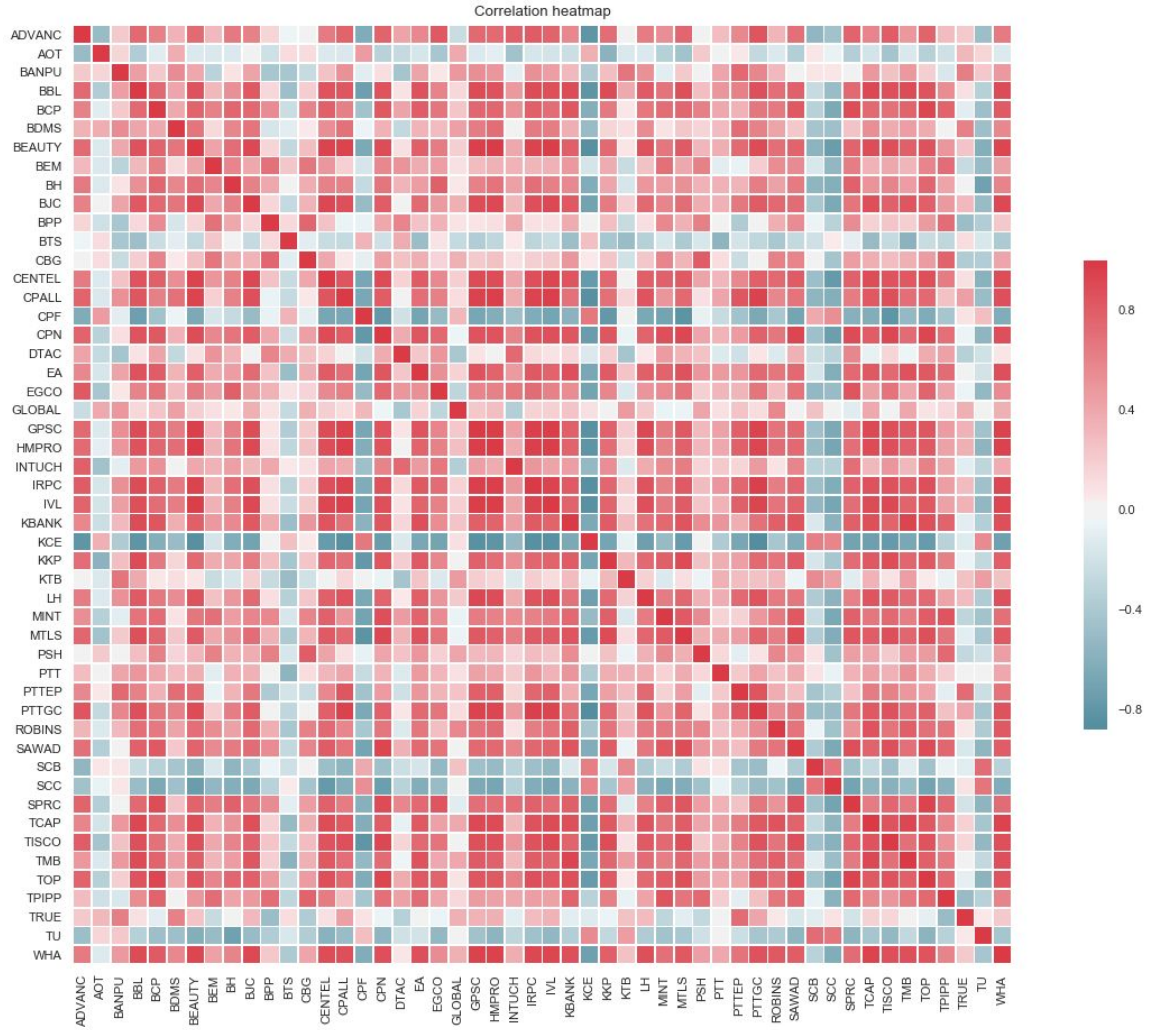
ตัวอย่างข้อมูล SCB

ที่เพิ่ม attribute next_open เข้ามา

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | DATE | OPEN | HIGH | LOW | CLOSE | VOL | NEXT_OPEN |
| 2 | 20170104 | 154 | 156 | 153.5 | 156 | 11832000 | UP |
| 3 | 20170105 | 156.5 | 157.5 | 155.5 | 156.5 | 8183500 | DRAW |
| 4 | 20170106 | 156.5 | 157 | 155 | 156.5 | 3712600 | DRAW |
| 5 | 20170109 | 156.5 | 158 | 155.5 | 156.5 | 6420200 | DOWN |
| 6 | 20170110 | 156 | 158.5 | 156 | 157.5 | 11395000 | UP |
| 7 | 20170111 | 158 | 158.5 | 157 | 157 | 6406500 | UP |
| 8 | 20170112 | 158.5 | 158.5 | 154.5 | 155.5 | 7774000 | UP |
| 9 | 20170113 | 156.5 | 158.5 | 155.5 | 157.5 | 6183100 | DOWN |
| 10 | 20170116 | 157 | 157 | 155 | 155 | 5236700 | DRAW |
| 11 | 20170117 | 155 | 156.5 | 153.5 | 155 | 8864100 | UP |
| 12 | 20170118 | 155.5 | 156.5 | 155 | 156 | 4313600 | DOWN |
| 13 | 20170119 | 155 | 155.5 | 153 | 155 | 6844800 | DOWN |
| 14 | 20170120 | 151.5 | 152 | 148.5 | 150.5 | 28187500 | UP |
| 15 | 20170123 | 151 | 152 | 149.5 | 150.5 | 8783700 | DOWN |
| 16 | 20170124 | 150 | 151.5 | 150 | 150.5 | 7154200 | UP |
| 17 | 20170125 | 151 | 153 | 150.5 | 152 | 16493300 | UP |
| 18 | 20170126 | 153.5 | 154 | 152.5 | 153.5 | 9258000 | DRAW |
| 19 | 20170127 | 153.5 | 153.5 | 152.5 | 152.5 | 4390400 | DRAW |
| 20 | 20170130 | 152.5 | 153 | 152 | 152.5 | 3537000 | DOWN |
| 21 | 20170131 | 152 | 152 | 150.5 | 151 | 7703200 | DOWN |
| 22 | 20170201 | 150.5 | 153 | 150.5 | 152 | 6969200 | DRAW |
| 23 | 20170202 | 152 | 153 | 151 | 151 | 6529100 | DRAW |
| 24 | 20170203 | 151 | 153.5 | 151 | 153 | 6597700 | UP |
| 25 | 20170206 | 154 | 155 | 153.5 | 154.5 | 9901200 | DRAW |
| 26 | 20170207 | 154.5 | 155 | 153.5 | 154 | 3802600 | DOWN |
| 27 | 20170208 | 153.5 | 154.5 | 153 | 154 | 5747300 | UP |
| 28 | 20170209 | 154.5 | 156 | 153.5 | 154 | 6157400 | UP |
| 29 | 20170210 | 154.5 | 155 | 154 | 154 | 9620500 | UP |
| 30 | 20170214 | 155 | 156 | 153 | 154 | 9754000 | UP |
| 31 | 20170215 | 154.5 | 155 | 153 | 153.5 | 4413500 | UP |
| 32 | 20170216 | 154 | 154.5 | 153.5 | 154 | 4592500 | DRAW |
| 33 | 20170217 | 154 | 155 | 153.5 | 154 | 6132300 | DRAW |
| 34 | 20170220 | 154 | 154.5 | 153 | 154 | 6282600 | DOWN |
| 35 | 20170221 | 153.5 | 154 | 152 | 152.5 | 5458500 | DRAW |

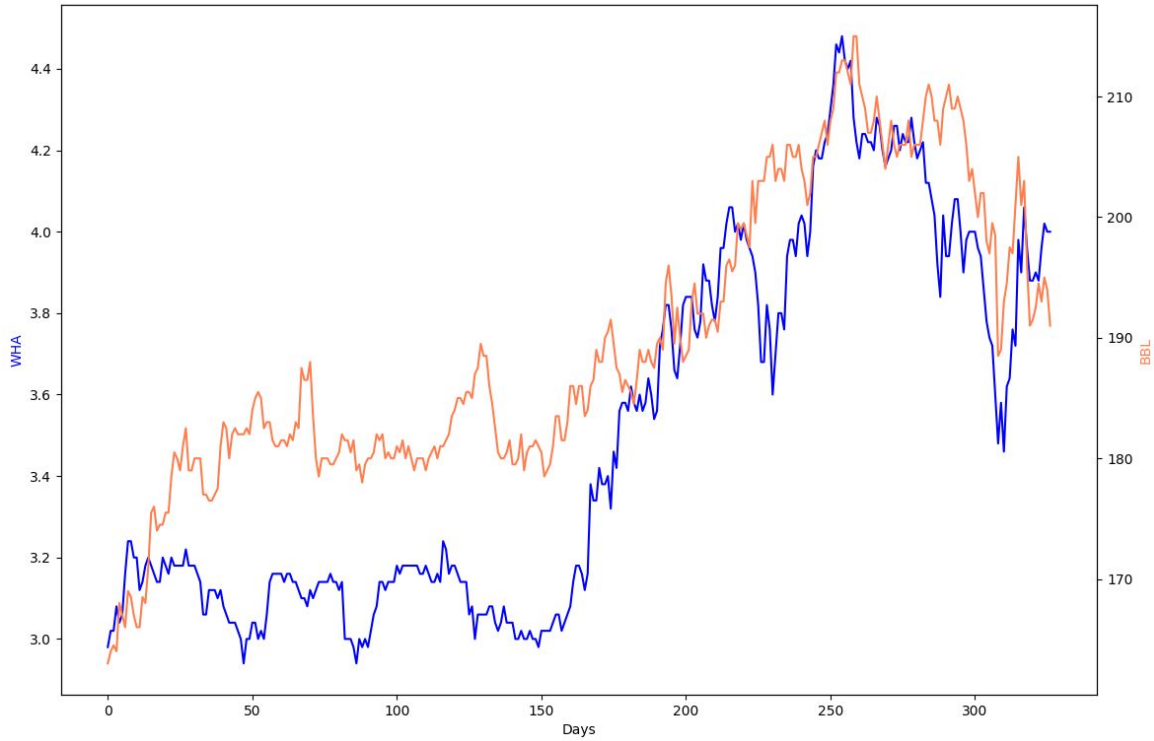
# Data Exploration

- แผนภาพแสดง Heatmap
  - ความสัมพันธ์ของราคาหุ้น 50 ตัว
  - สีแดง (1) คือคู่ที่มีความสัมพันธ์เหมือนกัน
  - สีขาว (0) คือคู่ที่ไม่มีความสัมพันธ์กัน
  - สีน้ำเงิน (-1) คือคู่ที่มีความสัมพันธ์แบบตรงกันข้าม



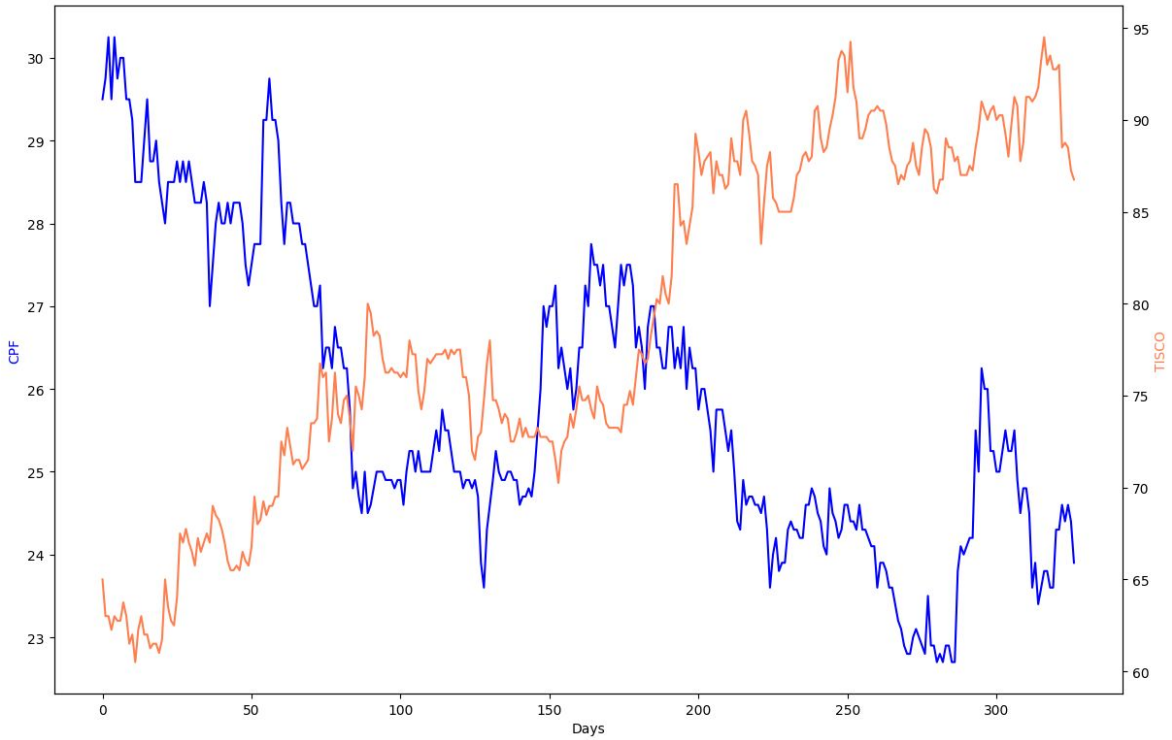
# Data Exploration (ต่อ)

- ตัวอย่างแผนภาพแสดง Correlation ไปในทิศทางเดียวกัน
  - WHA และ BBL ในระยะเวลา 1 ปี



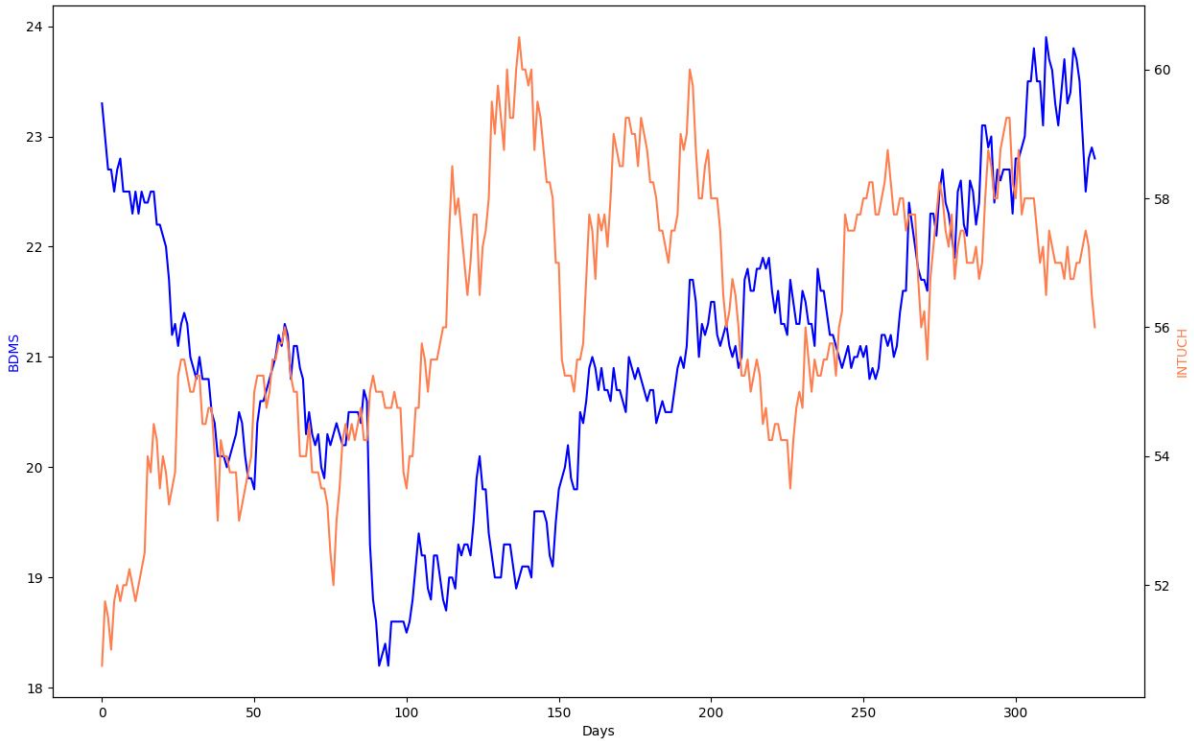
# Data Exploration (ต่อ)

- ตัวอย่างแผนภาพแสดง Correlation ที่มีลักษณะตรงกันข้ามกัน
  - CPF และ TISCO ในระยะเวลา 1 ปี



# Data Exploration (ต่อ)

- ตัวอย่างแผนภาพแสดง Correlation ไม่มีความสัมพันธ์กัน
  - BDMS และ INTUCH ในระยะเวลา 1 ปี



# Evaluation Method

- Probability
  - เนื่องจากจุดประสงค์ของ Model Gaussian คือ ทำนายราคาเปิดของราคาหุ้นแต่ละตัวในวันถัดไปเป็นอย่างไร(ขึ้น ลง เท่าเดิม) โดยวัดความแม่นยำจากจำนวนครั้งที่ทายถูก ส่วนด้วย จำนวนครั้งทั้งหมดที่ทำนาย
- Root Mean Square Error
  - ใช้ Root Mean Square Error เป็นตัววัดความแม่นยำของ Model Linear regression และ polynomial regression

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

- Mean Absolute Error
  - ใช้ Mean Absolute Error เป็นตัววัดความแม่นยำ โดยไม่คำนึงถึงการกระจายตัวของ Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} |e_t|$$

# Models & Algorithms

- Source Code
  - ฟังก์ชั่นสำหรับอ่านค่าจากไฟล์ CSV

```
def import_data(stockName):
    # import total dataset
    data = pd.read_csv('StocksData/' + stockName + '.csv')
    data = data.drop(['DATE'], 1)
    data = data.dropna()

    # get a list of column names
    headers = list(data.columns.values)

    # separate into independent and dependent variables
    x = data[headers[:-1]]
    y = data[headers[-1:]].values.ravel()
    y[y == "UP"] = 1
    y[y == "DRAW"] = 0
    y[y == "DOWN"] = -1

    return x, y
```

# Models & Algorithms (ต่อ)

- Model ที่ใช้
  - Linear Regression
- 10-fold cross-validation
  - SCB prediction
  - RMSE of Linear Regression accuracy result: 0.820846708618

```
if __name__ == '__main__':
    # get training and testing sets
    stockName = 'SCB'
    x, y = import_data(stockName)

    # set to 10 folds
    skf = KFold(n_splits=10)

    # blank lists to store predicted values and actual values
    predicted_y = []
    expected_y = []

    # partition data
    accuracy = 0
    for train_index, test_index in skf.split(x, y):
        # specific ".loc" syntax for working with dataframes
        x_train, x_test = x.loc[train_index], x.loc[test_index]
        y_train, y_test = y[train_index], y[test_index]

        # create and fit classifier
        classifier = LinearRegression()
        classifier.fit(x_train, y_train)

        # store result from classification
        predicted_y = classifier.predict(x_test)

        # store expected result for this specific fold
        expected_y = y_test

        accuracy += np.sqrt(metrics.mean_squared_error(expected_y, predicted_y))

print(stockName,"prediction")
print("RMSE of Linear Regression result:" , accuracy/10)
```

# Models & Algorithms (ต่อ)

- Model ที่ใช้
  - Polynomial Regression
- 10-fold cross-validation
  - SCB prediction
  - RMSE of Linear Regression accuracy result: 0.973619205627

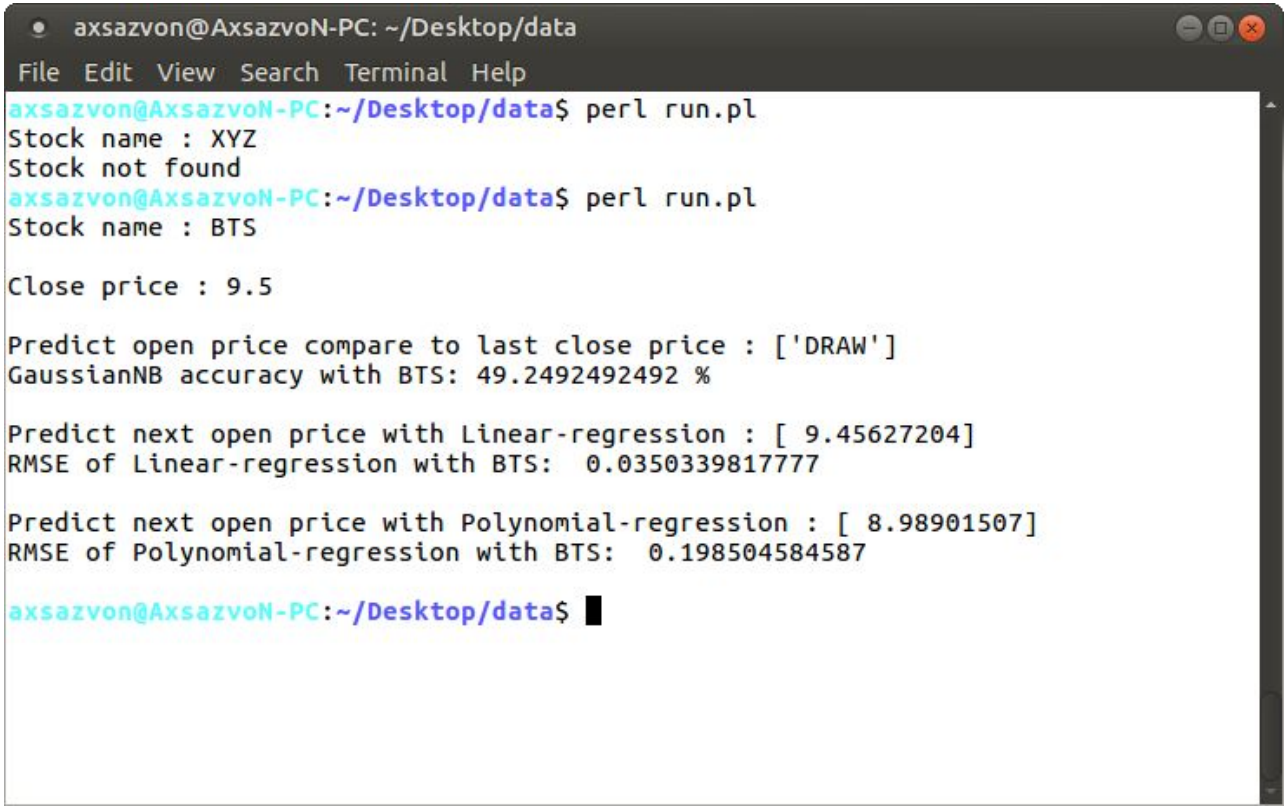
```
if __name__ == '__main__':

    stockName = 'SCB'
    x, y = import_data(stockName)

    skf = KFold(n_splits=10)

    predicted_y = []
    expected_y = []

    accuracy = 0
    for train_index, test_index in skf.split(x, y):

        x_train, x_test = x.loc[train_index], x.loc[test_index]
        y_train, y_test = y[train_index], y[test_index]

        classifier = PolynomialFeatures(degree = 3)
        X_poly = classifier.fit_transform(x_train)

        lr = LinearRegression()
        lr.fit(X_poly, y_train)

        classifier.fit(X_poly, y_train)

        predicted_y = lr.predict(classifier.fit_transform(x_test))

        expected_y = y_test

        accuracy += np.sqrt(metrics.mean_squared_error(expected_y, predicted_y))

print "Polynomial-regression accuracy with " + stockName + ": " , accuracy/10
```

# Models & Algorithms (ต่อ)

- Model ที่ใช้
  - GaussianNB
- Accuracy
  - SCB prediction
  - gaussian Naive Bayes ไม่สามารถใช้ RMSE ได้ เนื่องจากผลลัพธ์ของ Gaussian นั้นจะเป็นค่าบอกว่าเพิ่มขึ้น ลดลง หรือเท่าเดิม ไม่สามารถวัดเป็นค่าตวเลขได้ จึงใช้ค่าความน่าจะเป็นแทน
  - Accuracy score of Gaussian result: 35.7357357357 %

```python
if __name__ == '__main__':
    # get training and testing sets
    stockName = 'SCB'
    x, y = import_data(stockName)

    # set to 10 folds
    skf = StratifiedKFold(n_splits=10)

    # blank lists to store predicted values and actual values
    predicted_y = []
    expected_y = []

    accuracy = 0
    # partition data
    for train_index, test_index in skf.split(x, y):
        # specific ".loc" syntax for working with dataframes
        x_train, x_test = x.loc[train_index], x.loc[test_index]
        y_train, y_test = y[train_index], y[test_index]

        # create and fit classifier
        classifier = GaussianNB()
        classifier.fit(x_train, y_train)

        # store result from classification
        predicted_y.extend(classifier.predict(x_test))

        # store expected result for this specific fold
        expected_y.extend(y_test)

    accuracy = metrics.accuracy_score(expected_y, predicted_y)

print(stockName,"prediction")
print("Accuracy score of Gaussian result:" , accuracy*100, '%')
```

# Application



# Summary

- ผลที่ได้รับจากการทำโครงงานนี้
  - สร้างเครื่องมือช่วยทำนายการเทรด ที่สามารถนำมาใช้ได้จริง ส่งผลให้มีส่วนช่วยในการตัดสินใจของนักลงทุน
- บทเรียนที่ได้รับจากการลงมือจริง
  - เห็นความแตกต่างของการวัดผลด้วย RMSE และ MAE
  - เข้าใจการทำงาน และความแตกต่างของอัลกอริทึ่มที่เลือกใช้