REGRESSION & CLASSIFICATION TO GENERATE BUY/SELL SIGNALS

How to Generate Buy & Sell Signals of Stock Trading

An example with Crude Oil daily data

Sarit Maitra

Sep 18, 2020 · 6 min read ★

Image by Author

Buy and sell signals can be generated by two moving averages — a long-period and a short-period average. When the short moving average rises or falls below the long moving average, buy or sell signals can be generated based on set parameters. *

Here, with a simple example, we have shown as how to generate report on buy/sell signals and visualize the chart.

```
print("....Data Loading...."); print();
print('\033[4mCrude Oil Spot Price\033[0m');
data = web.DataReader('CL=F', data_source = 'yahoo', start = '2000-
01-01');
data;
```

```
....Data Loading....

Crude Oil Spot Price
```

Date	High	Low	Open	Close	Volume	Adj Close
2000-08-23	32.799999	31.950001	31.950001	32.049999	79385.0	32.049999
2000-08-24	32.240002	31.400000	31.900000	31.629999	72978.0	31.629999
2000-08-25	32.099998	31.320000	31.700001	32.049999	44601.0	32.049999
2000-08-28	32.919998	31.860001	32.040001	32.869999	46770.0	32.869999
2000-08-29	33.029999	32.560001	32.820000	32.720001	49131.0	32.720001
...	...	...	...	...	...	...
2020-09-11	37.820000	36.669998	37.009998	37.330002	363787.0	37.330002
2020-09-14	37.680000	36.820000	37.320000	37.259998	347563.0	37.259998
2020-09-15	38.570000	37.060001	37.279999	38.279999	348861.0	38.279999
2020-09-16	40.340000	38.349998	38.349998	40.160000	348861.0	40.160000
2020-09-17	40.290001	39.419998	40.189999	39.939999	41913.0	39.939999

5040 rows × 6 columns

Let's focus on closing price of daily stock.

```
df = data[['Close']];
# Plot the closing price
df.Close.plot(figsize=(10, 5));
plt.ylabel("Prices (USD)"); plt.title("Crude Oil Price Series");
plt.show();
```

Explanatory variables

I have used two EMAs of different time period to find the price movements; in simple moving average (SMA), each value in the time period carries equal weight, and values outside of the time period are not included in the average. EMA is a cumulative calculation, including all data where the past values have a diminishing contribution to the average; more recent values have a greater contribution. EMA allows the moving average to be more responsive to changes in the data.

```
df['ema10'] =
(df['Close'].ewm(span=10,adjust=True,ignore_na=True).mean());
df['ema20'] =
(df['Close'].ewm(span=20,adjust=True,ignore_na=True).mean());
df['price_tomorrow'] = df['Close'].shift(-1);
df.dropna(inplace=True);
X = df[['ema10', 'ema20']]; y = df['price_tomorrow'];
```

Plot EMA with original data

```
fig = go.Figure(data=[go.Candlestick(x=data.index[-100:],
open=data['Open'][-100:], high=data['High'][-100:], low=data['Low']
[-100:], close=data['Close'][-100:])]);
fig.add_trace(go.Scatter(x = df.index[-100:], y = df.ema10[-100:],
marker = dict(color = "blue"), name = "EMA10"));
fig.add_trace(go.Scatter(x = df.index[-100:], y = df.ema20[-100:],
marker = dict(color = "gray"), name = "EMA10"));
fig.update_xaxes(showline=True, linewidth=2, linecolor='black',
```

```
mirror=True);
fig.update_yaxes(showline=True, linewidth=2, linecolor='black',
mirror=True);
fig.update_layout(autosize = False, width = 1200, height = 600);
fig.update_layout(title='Crude oil prices, yaxis_title='(US$)');
fig.show();
```


Here, shorter the period, the more weight is given to recent prices. In fact a short time period generates a line closely following the actual candles on the chart. Visually this means that there are more occurrences of the price moving above or below the EMA which can be interpreted as a signal to open or close a position.

Time series data split

```
# Split the data into train and test data set
tscv = TimeSeriesSplit();
print(tscv);
TimeSeriesSplit(max_train_size = 0.80, n_splits=5);
for train_index, test_index in tscv.split(X):
   #print("TRAIN:", train_index, "TEST:", test_index);
   X_train, X_test = X[train_index], X[test_index];
   y_train, y_test = y[train_index], y[test_index];

print('Length train set: {}'.format(len(y_train)));
print('Length test set: {}'.format(len(y_test)));
```

Regression rule

```
# Create a linear regression model
model = ElasticNet(max_iter=5000, random_state=0).fit(X_train,
y_train);
print("Linear Regression model:");
print("Crude oil Price (y) = %.2f * 10 Days Moving Average (x1) \
+ %.2f * 20 Days Moving Average (x2) \
+ %.2f (constant)" % (model.coef_[0], model.coef_[1],
model.intercept_));
```

Plot actual and predicted output

```
y_pred = DataFrame(model.predict(X_test), index = df[-
len(y_test):].index, columns = ['price']);
y_test = DataFrame(y_test, index = df[-len(y_test):].index, columns =
['price']);

fig = go.Figure();
fig.add_trace(go.Scatter(x = y_pred.index, y = y_pred.price,
marker = dict(color ="red"), name = "Actual price"));
fig.add_trace(go.Scatter(x = y_test.index, y = y_test.price,
marker=dict(color = "green"), name = "Predicted price"));
fig.update_xaxes(showline = True, linewidth = 2, linecolor='black',
mirror = True, showspikes = True,);
fig.update_yaxes(showline = True, linewidth = 2, linecolor='black',
mirror = True, showspikes = True,);
fig.update_layout(title= "Crude oil price (predicted vs actual",
yaxis_title = 'price (US$)', hovermode = "x", hoverdistance = 100,
spikedistance = 1000);
fig.update_layout(autosize = False, width = 1000, height = 400,);
fig.show();
```

Test accuracy

```
accuracy = model.score(X_test, y_test);
print("Accuracy: ", round(accuracy*100,2).astype(str) + '%');
```

```
oil = DataFrame();
oil['price'] = df[-len(y_test):]['Close'];
oil['pred_next_day'] = y_pred;
oil['actual_price_next_day'] = y_test;
oil['returns'] = oil['price'].pct_change().shift(-1);
oil['signal'] = np.where(oil.pred_next_day.shift(1) <
oil.pred_next_day,1,0);
oil['strategy_returns'] = oil.signal * oil['returns'];

((oil['strategy_returns']+1).cumprod()).plot(figsize=(10,5));
plt.ylabel('Cumulative Returns');
plt.show();
```

> The Sharpe ratio is a well-known and well-reputed measure of risk-adjusted return on an investment or portfolio, developed by the economist William Sharpe.

The Formula for the Sharpe Ratio is:

Average return / Std dev of return

- Usually, any Sharpe ratio > 1.0 is considered acceptable.
- A ratio > 2.0 is rated as very good.
- A ratio of 3.0 or higher is considered excellent.
- A ratio < 1.0 is considered sub-optimal.

Buy/Sell Report

```
df.loc[:,'pred_price'] = model.predict(df[['ema10', 'ema20']]);
df.loc[:,'signal'] = np.where(df.pred_price.shift(1) <
df.pred_price,"Buy","Sell");
df.loc[:,'price_direction'] = df['signal'].replace(('Sell', 'Buy'),
(0, 1));
df.tail();
```

Buy/Sell signals plot

```
buys = df.loc[df['price_direction'] == 1]; sells =
df.loc[df['price_direction'] == 0];

# Plot
fig = plt.figure(figsize=(20, 5));
plt.plot(df.index, df['Close'], lw=2., label='Price');

# Plot the buy and sell signals on the same plot
plt.plot(buys.index, df.loc[buys.index]['Close'], '^', markersize=5,
color='k', lw=2., label='Buy');
plt.plot(sells.index, df.loc[sells.index]['Close'], 'v', markersize =
5, color='g', lw=2., label='Sell');
plt.ylabel('Price (USD)'); plt.xlabel('Date');
plt.title('Buy and Sell signals plot'); plt.legend(loc='best');

# Display everything
plt.show()
```


Classification rule

```
# creating predictors and target variables for classification
print('Data shape:', df.shape); print();
X = np.array(df[['ema10', 'ema20']]);

# Target variable
y = np.array(df['price_direction']);
```

Time series split

```
tscv = TimeSeriesSplit()
#print(tscv)
TimeSeriesSplit(max_train_size = 0.80, n_splits=5)
for train_index, test_index in tscv.split(X):
    #print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

print('Length train set: {}'.format(len(y_train)))
print('Length test set: {}'.format(len(y_test)))
```

Logistic Regression

```
print('\033[4mLogistic Regression\033[0m')
clf = LogisticRegression(solver='liblinear', C=0.05, random_state=0
).fit (X_train,y_train);
model_scores = cross_val_score(clf, X_train, y_train, cv=5);
model_mean = model_scores.mean();
print ('Accuracy score (%):', model_mean*100);
```

Prediction & accuracy

```
y_pred = clf.predict(X_test)
# evaluate predictions
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

Classification report

```
print('Confusion matrix:'); print(confusion_matrix(y_test, y_pred));
print(); print ('Classification report:'); print
(classification_report(y_test, y_pred));
```


The recall means how many of this class we find over the whole number of element of this class. The precision here is how many are correctly classified among that class. The f1-score is the harmonic mean between precision & recall. The support is the number of occurrence of the given class in our data.

Conclusion

We have shown here a simplified version of generating buy/sell signals based on moving average. This is a simple and fundamental process. Other technical indicators such as MACD, ROC etc. can be added based on trading strategy.

Connect me here.

Note: The programs described here are experimental and should be used with caution for any commercial purpose. All such use at your own risk.

Trading Algorithms | Machine Learning | Buy Sell Stock