BOĞAZİÇİ ÜNİVERSİTESİ

IE423

QUALITY ENGINEERING

Project Part 1

Instructor: Mustafa Gökçe Baydoğan

Date: 11.12.2023

Group Members

Ferhat Peynirci (2019402105)

Recep Eren Durgut (2019402000)

Metehan Yalçın (2018402042)

In this assignment, the objective is to explore the application of control charts in pairs trading, focusing on the hourly time series data of BIST30 stock indices. It will be identified, highly correlated stock pairs, model their relationship through linear regression, and apply control charts to monitor residuals for potential trading opportunities.

The data is fetched and manipulated through statistical tools. There are 30 stock bonds (BIST30) given and the data examined is within the dates from 2018-01 to 2018-04.

Given bonds: THYAO', 'AKBNK', 'ARCLK', 'ASELS', 'BIMAS', 'DOHOL', 'EKGYO', 'EREGL', 'FROTO', 'GUBRF', 'GARAN', 'KRDMD', 'KCHOL', 'KOZAL', 'KOZAA', 'PGSUS', 'PETKM', 'SAHOL', 'SASA', 'SISE', 'TAVHL', 'TKFEN', 'TUPRS', 'TTKOM', 'TCELL', 'HALKB', 'ISCTR', 'VAKBN', 'VESTL', 'YKBNK'

Task 1: Basic Pairs Trading Strategy Using Constant Variance Assumption

In this task, it is going to be investigated that how a basic pairs trading strategy is constructed with the assumption of constant variance.

Initially, the data between 2018-01 – 2018-04 is imported and the data frame objects are created. The stocks are separated into smaller data frames, and they are merged again in one main data frame in order to get a correct correlation matrix.

After obtaining a correlation matrix, a threshold is set (0.85). and the pairs whose correlation coefficients are beyond that value are called highly correlated stocks and their linear regression models are obtained.

Linear Regression model between highly correlated stocks

In [ ]:
```
from sklearn.linear_model import LinearRegression

trading_pairs = []  # List to store trading pairs

for pair in highly_correlated_pairs:
    pair_data = df_merged[['timestamp', pair[0], pair[1]]].dropna()

    X = pair_data[pair[0]].values.reshape(-1, 1)
    y = pair_data[pair[1]].values

    model = LinearRegression()
    model.fit(X, y)

    trading_pairs.append({'pair': pair, 'model': model, 'data': pair_data})
```

It is asked to run a trading simulation, and for the simulation, there must be signals triggering buy/sell actions. These actions are determined by residuals (k = 2).

Trigger limits for trading

In [ ]:
```
for pair_info in trading_pairs:
    pair_info['residuals'] = pair_info['data'][pair_info['pair'][1]] - pair_info['model'].predict(pair_info['data'][pair_info['pair'][0]].values.reshape(-1, 1))

# Assuming constant variance assumption
for pair_info in trading_pairs:
    mean_residuals = pair_info['residuals'].mean()
    std_residuals = pair_info['residuals'].std()

    # Set control limits
    pair_info['upper_limit'] = mean_residuals + 2 * std_residuals
    pair_info['lower_limit'] = mean_residuals - 2 * std_residuals
```

Simulation:

Trading simulation

In [ ]:
```
trading_capital = 1000  # Initial trading capital
position_size = 100  # Number of shares to trade for each signal

for pair_info in trading_pairs:
    signals = []

    for i, residual in enumerate(pair_info['residuals']):
        if residual > pair_info['upper_limit']:
            signals.append(-1)  # Short signal
        elif residual < pair_info['lower_limit']:
            signals.append(1)  # Long signal
        else:
            signals.append(0)  # No signal

    # Assuming equal capital allocation for each pair
    capital_per_pair = trading_capital / len(trading_pairs)

    # Trading simulation
    for i in range(1, len(signals)):
        if signals[i] != signals[i-1]:
            if signals[i] == 1:
                # Buy the underperforming stock
                capital_per_pair -= position_size * pair_info['data'][pair_info['pair'][1]].iloc[i]
            elif signals[i] == -1:
                # Short sell the outperforming stock
                capital_per_pair += position_size * pair_info['data'][pair_info['pair'][1]].iloc[i]

    print(f"Potential gains for pair {pair_info['pair']}: ${capital_per_pair - trading_capital}")
```

There are nine basic pairs combined of 2 stocks whose threshold is higher than 0.85. According to the simulation results, ('KRDMD', 'TAVHL') is the best option for investment since the signals are doing the buy/sell work and their residuals are high, standard deviation is high as well.

Gains: $13015.731111111112

```
Potential gains for pair ('AKBNK', 'GARAN'): $5410.091111111111
Potential gains for pair ('AKBNK', 'ISCTR'): $-1349.868888888889
Potential gains for pair ('AKBNK', 'VAKBN'): $-3468.9188888888884
Potential gains for pair ('ARCLK', 'HALKB'): $-1779.5488888888888
Potential gains for pair ('GARAN', 'ISCTR'): $-1340.4988888888888
Potential gains for pair ('KRDMD', 'TAVHL'): $13015.731111111112
Potential gains for pair ('KOZAA', 'KOZAL'): $-1047.4588888888889
Potential gains for pair ('HALKB', 'KCHOL'): $-3630.698888888889
Potential gains for pair ('ISCTR', 'VAKBN'): $-1462.658888888889
```

Task 2: Advanced Pairs Trading Strategy Using Time Series Analysis

This task requires incorporating time series analysis. Therefore, an ARIMA (autoregressive integrated moving average) model is implemented. ACF and PACF are checked before and after ARIMA. The code below is trying to find the best fitting ARIMA model. The visualization is below.

Part 2 with Time Series on Resudials Approach

In [ ]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from pmdarima import auto_arima

for pair_info in trading_pairs:

    #checking ACF and PACF before ARIMA
    print(pair_info['pair'])
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))
    sm.graphics.tsa.plot_acf(pair_info['residuals'], lags=40, ax=ax1)
    sm.graphics.tsa.plot_pacf(pair_info['residuals'], lags=40, ax=ax2)
    plt.show()

    auto_arima_model = auto_arima(pair_info['residuals'], seasonal=False, trace=True, suppress_warnings=True) #finding best fittin ARIMA model
    order = auto_arima_model.order

    model = sm.tsa.ARIMA(pair_info['residuals'], order=order)
    results = model.fit()

    #Visualtization of residuals after ARIMA fit
    pair_info['residuals'] = results.resid
    fig, ax = plt.subplots(figsize=(10, 6))
    ax.plot(pair_info['residuals'])
    ax.set_title('Residuals of Time Series Model')
    plt.show()
    # checking ACF and PACF after ARIMA fit
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))
    sm.graphics.tsa.plot_acf(pair_info['residuals'], lags=40, ax=ax1)
    sm.graphics.tsa.plot_pacf(pair_info['residuals'], lags=40, ax=ax2)
    plt.show()
```

```
('AKBNK', 'GARAN')
```

Autocorrelation

Performing stepwise search in order to minimize AIC. For instance, for pairs ('AKBNK', 'GARAN'):

```
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-2431.154, Time=0.31 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-2428.692, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-2427.832, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-2428.131, Time=0.06 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-2430.691, Time=0.02 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=-2432.491, Time=0.07 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=-2434.019, Time=0.05 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AIC=-2432.275, Time=0.09 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-2429.974, Time=0.08 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=-2430.843, Time=0.22 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=-2436.018, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=-2430.118, Time=0.01 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=-2434.490, Time=0.03 sec
 ARIMA(0,1,3)(0,0,0)[0]             : AIC=-2434.274, Time=0.04 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=-2431.973, Time=0.04 sec
 ARIMA(1,1,3)(0,0,0)[0]             : AIC=-2432.262, Time=0.10 sec

Best model:  ARIMA(0,1,2)(0,0,0)[0]
Total fit time: 1.202 seconds
```

ARIMA fit is done by using residuals. The simulation results are:

```
Potential gains for pair ('AKBNK', 'GARAN'): $39.27111111111162
Potential gains for pair ('AKBNK', 'ISCTR'): $-1284.968888888889
Potential gains for pair ('AKBNK', 'VAKBN'): $2210.711111111111
Potential gains for pair ('ARCLK', 'HALKB'): $-1608.8388888888896
Potential gains for pair ('GARAN', 'ISCTR'): $-1607.168888888889
Potential gains for pair ('KRDMD', 'TAVHL'): $-7641.388888888889
Potential gains for pair ('KOZAA', 'KOZAL'): $-257.01888888888914
Potential gains for pair ('HALKB', 'KCHOL'): $-8250.248888888887
Potential gains for pair ('ISCTR', 'VAKBN'): $1950.431111111112
```

The best result belongs to the pairs ('AKBNK', 'VAKBN'): $2210.711111111111

In the next step, prices approach is used. Prices are fitted into the linear regression. Similar steps are conducted.

In [ ]:
```
for stock in df_merged[stock_names]:
    df_merged[stock].fillna(df_merged[stock].mean(), inplace=True)
    auto_arima_model = auto_arima(df_merged[stock], seasonal=False, trace=True, suppress_warnings=True) #Finding best arima fit for the data
    order = auto_arima_model.order

    model = sm.tsa.ARIMA(df_merged[stock], order=order)
    results = model.fit()

    df_merged[stock] = results.resid
```

```
('VESTL', 'YKBNK')
```

In [ ]:
```
for pair_info in trading_pairs:
    pair_info['residuals'] = pair_info['data'][pair_info['pair'][1]] - pair_info['model'].predict(pair_info['data'][pair_info['pair'][0]].values.reshape(-1, 1))

# Assuming constant variance assumption
for pair_info in trading_pairs:
    mean_residuals = pair_info['residuals'].mean()
    std_residuals = pair_info['residuals'].std()

    # Set control limits
    pair_info['upper_limit'] = mean_residuals + 2 * std_residuals
    pair_info['lower_limit'] = mean_residuals - 2 * std_residuals
```

In [ ]:
```
trading_capital = 1000  # Initial trading capital
position_size = 100  # Number of shares to trade for each signal

for pair_info in trading_pairs:
    signals = []

    for i, residual in enumerate(pair_info['residuals']):
        if residual > pair_info['upper_limit']:
            signals.append(-1)  # Short signal
        elif residual < pair_info['lower_limit']:
            signals.append(1)  # Long signal
        else:
            signals.append(0)  # No signal

    # Assuming equal capital allocation for each pair
    capital_per_pair = trading_capital / len(trading_pairs)

    # Trading simulation
    for i in range(1, len(signals)):
        if signals[i] != signals[i-1]:
            if signals[i] == 1:
                # Buy the underperforming stock
                capital_per_pair -= position_size * pair_info['data'][pair_info['pair'][1]].iloc[i]
            elif signals[i] == -1:
                # Short sell the outperforming stock
                capital_per_pair += position_size * pair_info['data'][pair_info['pair'][1]].iloc[i]
    if capital_per_pair - trading_capital > 0:
        print(f"Potential gains for pair {pair_info['pair']}: ${capital_per_pair - trading_capital}")
```

The best result belongs to the pairs ('ARCLK', 'PGSUS'): $1244.4630541871934

Discussion

By using basic pairs strategy using constant variance assumption, the results are much higher compared to the results in task 2, linear regression approach. Since the first result is very profitable & unrealistic and linear regression approach is clearer than basic pairs strategy because it considers many more constraints, it could be concluded that linear regression approach should be used in real life. In real life, reliability is more important. However, there are unwanted attributions in linear regression model compared to basic pairs strategy. Time series models are harder to conduct. There is more time and operations are needed.

References

https://tr.investing.com/equities/