IE423: QUALITY ENGINEERING

Project Part 1

Fall 2023 & 02.11.2023

Ömer Coşkun - 2019402177

Fatmanur Yaman - 2019402204

Hüseyin Emre Bacak - 2021402279

1. Executive Summary

In the study, 6 stock prices from Borsa Istanbul are examined and detailed outlier analysis is conducted. The control charts and box plots are used to explain the variability and determine the outliers. Then, Google Trends search volume data added and whether there is a correlation between search volume and stock prices are examined. In addition to this, the other factors and main reasons that can affect the stock prices are analyzed.

2. Introduction

There is uncertainty and variability in every aspect of the real world. Industrial engineers try to deal with the variability to decrease the uncertainty of the quality. There are several tools and methods developed by industrial engineers to detect the outliers, the instances with high variabilities. The Box Plot & Interquartile Range and Control Charts & 3-sigma rule are the main methods that make it easier to identify the outliers by using data visualization.

One of the areas that is being affected by various factors is the financial markets. The stock prices follow the random walk usually, but sometimes they show extremely high values. In the study, 6 stocks from Borsa Istanbul have been examined.

- **AKBNK** - AKBANK T.A.Ş. - Banking
- **VAKBN** - TÜRKİYE VAKIFLAR BANKASI T.A.O. - Banking
- **ARCLK** - Arçelik Pazarlama A.Ş. - Consumer Goods
- **TUPRS**- TÜRKİYE PETROL RAFİNERİLERİ A.Ş. - Petroleum Products
- **TCELL** - TURKCELL İLETİŞİM HİZMETLERİ A.Ş. - Telecommunication
- **THYAO**- TURK HAVA YOLLARI A.O. - Air Transporation

3. Methods

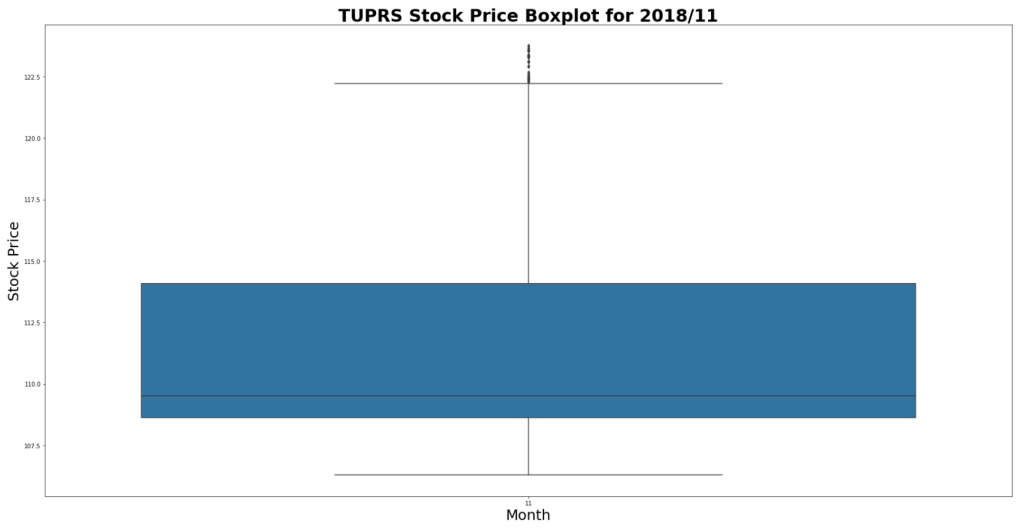
3.1. Control Chart

The control charts are commonly used in statistical process control since they show the limits by using variation and outliers. Control charts have 3 lines. The center line represents the average of the data. The Upper Control Limit line represents the last limit that the data can be accepted. It is obtained by adding 3 standard deviations to the mean. Lastly,

the Lower Control Limit line represents the first limit that the data can be accepted. It is obtained by extracting 3 standard deviations to the mean. The data points should be in the UCL and LCL. Otherwise, they are counted as outliers. If all the points are in the limits and there is no outlier, it can be said that the process is in a state of statistical control.

3.2. Box Plot

The box plot splits the data into 4 points as 25th quartile (lower quartile), median, 75th quartile (upper quartile), and outliers. Since the box plot shows the min, max, and outliers, it is one of the powerful tools that graphically describes the variation.



4. The Code

4.1. Importing Libraries

The necessary libraries and packages are imported first.

In [1]:
```
# importing the necessary libraries for analysis and visualizations
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import math
from datetime import datetime
import matplotlib.dates as mdates
from dateutil.relativedelta import relativedelta
from sklearn.preprocessing import MinMaxScaler
import copy

import warnings
warnings.filterwarnings("ignore")
```

4.2. Preprocessing

Null values are filled with their averages and a new filled dataframe is obtained.

In [2]:
```
# reading the data to a dataframe
df = pd.read_csv('all_ticks_wide.csv')

# converting to pd datetime
df['timestamp'] = pd.to_datetime(df['timestamp'])

# getting year, month, day as features
df['year'] = df['timestamp'].dt.year
df['month'] = df['timestamp'].dt.month
df['day'] = df['timestamp'].dt.day

df
```

In [4]:
```
# function to fill null values
# it takes the
def fill_null_values_with_average(df, timestamp):
    filled_df = df.copy()
    other_columns = list(filled_df.columns.difference([timestamp]))
    for column in other_columns:
        # creating a mask for NaN values in the column
        null_mask = filled_df[column].isnull()
        # finding the first non-null value before each NaN
        before = filled_df[column].where(~null_mask).ffill()
        # finding the first non-null value after each NaN
        after = filled_df[column].where(~null_mask[::-1]).bfill()[::-1]
        # taking the average of them
        average = (before + after) / 2
        # filling the nan
        filled_df[column].fillna(average, inplace=True)
    return filled_df

# filling the nan
filled_df = fill_null_values_with_average(df, 'timestamp')
filled_df
```

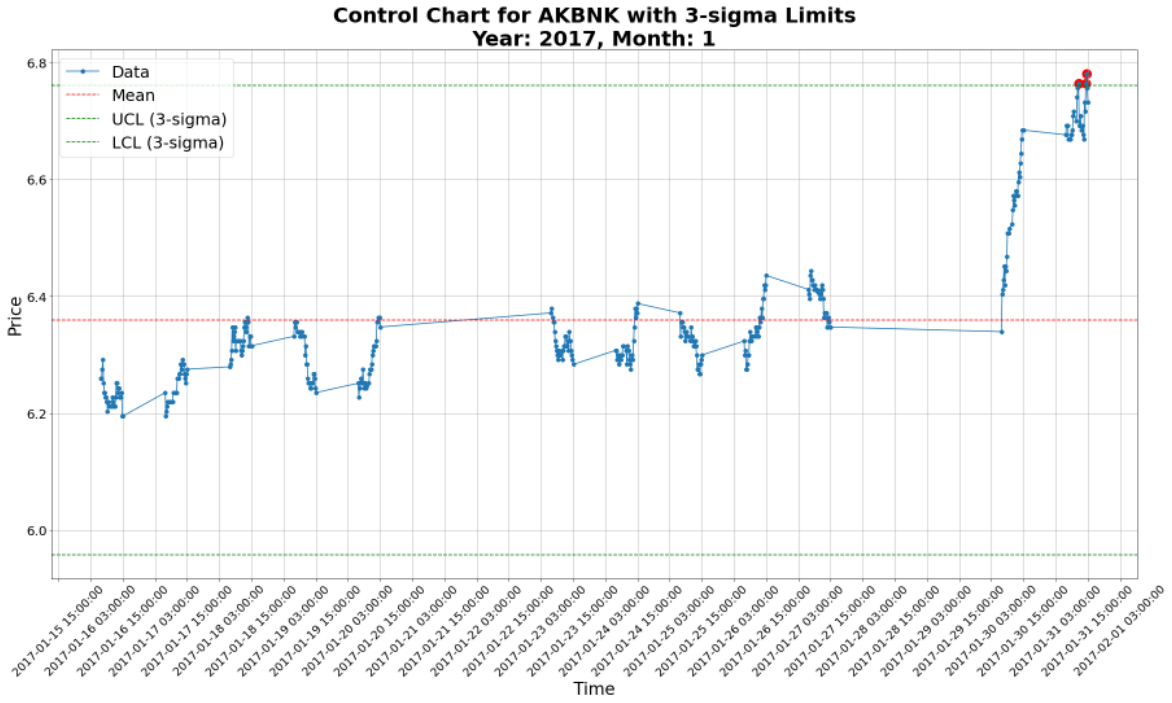
4.3. Analysis

4.3.1. Control Chart Analysis

The plot_control_chart function takes the data and calculates its mean and standard deviation. Then, 3 sigma values are calculated. UCL (upper control limit) is being calculated by adding 3 sigma to the mean, on the other hand LCL (lower control limit) is being calculated by extracting 3 sigma from the mean. According to these values, the limits are also being plotted to the chart with green lines. Lastly, the points that are out of the limits are being painted red. One of the outputs is in the below.

In [6]:
```
# copy df
df = filled_df.copy()
# start date
start = pd.to_datetime("2017-01-15 09:30:00+00:00")
# end data
end = pd.to_datetime("2019-01-15 09:30:00+00:00")
# filtering with time
ndf = df[(df['timestamp'] >= start) & (df['timestamp'] <= end)]
# filtering the necessary columns
df = ndf[['timestamp', 'year', 'month', 'AKBNK', 'VAKBN', 'ARCLK', 'TUPRS', 'TCELL', 'THYAO']].copy()

# years and months as array
years = [2017, 2018, 2019]
months = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

# the selected stocks
indexes_selected = ['AKBNK', 'VAKBN', 'ARCLK', 'TUPRS', 'TCELL', 'THYAO']

outliers_akbnk = []
outliers_vakbn = []
outliers_arclk = []
outliers_tuprs = []
outliers_tcell = []
outliers_thyao = []

def get_list(index):
    if index == "AKBNK":
        return outliers_akbnk
    elif index == "VAKBN":
        return outliers_vakbn
    elif index == "ARCLK":
        return outliers_arclk
    elif index == "TUPRS":
        return outliers_tuprs
    elif index == "TCELL":
        return outliers_tcell
    else:
        return outliers_thyao

# given a stock in a specific month, constructs the
# control charts for that month
def plot_control_chart(selected):
    for index in indexes_selected:
        # getting the prices for the stock
        prices = selected[index].to_list()
        # timestamps
        time = selected['timestamp'].to_list()
        # mean
        mu = np.mean(prices)
        # standard deviation
        sigma = np.std(prices)
        # UCL and LCL
        UCL = mu + 3 * sigma
        LCL = mu - 3 * sigma

        # create a list to store the outliers
        outliers = []
        # plot Mean, UCL, LCL
        plt.figure(figsize=(30, 15))
        plt.plot(time, prices, marker='o', label='Data')
        plt.axhline(mu, color='r', linestyle='--', label='Mean')
        plt.axhline(UCL, color='g', linestyle='--', label='UCL (3-sigma)')
        plt.axhline(LCL, color='g', linestyle='--', label='LCL (3-sigma)')

        # plot outliers as red dots
        for i, price in enumerate(prices):
            if price > UCL or price < LCL:
                plt.scatter(time[i], price, c='red', marker='o', s=200)
                outliers.append((time[i], price))
        # plot dates, x axis, y axis, title
        date_format = mdates.DateFormatter('%Y-%m-%d %H:%M:%S')
        plt.gca().xaxis.set_major_formatter(date_format)
        plt.gca().xaxis.set_major_locator(mdates.HourLocator(interval=12))
        plt.xlabel('Time', fontsize=26)
        plt.ylabel('Price', fontsize=26)
        plt.title(f'Control Chart for {index} with 3-sigma Limits\nYear: {selected["year"].iloc[0]}, Month: {selected["month"].iloc[0]}', fontsize=32, f
        plt.legend(fontsize=25)
        plt.grid(True)
        plt.xticks(rotation=45, fontsize=18)
        plt.yticks(fontsize=20)
        plt.show()
        # print outliers
        if outliers:
            print(f'Outliers for {index}:')
            l = get_list(index)
            for outlier in outliers:
                l.append((outlier[0],outlier[1]))
                print(f'Timestamp: {outlier[0]}, Price: {outlier[1]}')

# group by year, month
grouped = df.groupby(['year', 'month'])

# apply the plot_control_chart function to each group
grouped.apply(plot_control_chart)
```

Control charts for all 6 stocks are plotted monthly and the outliers are detected in the jupyter notebook. The data regarding the previous 2 years was used.



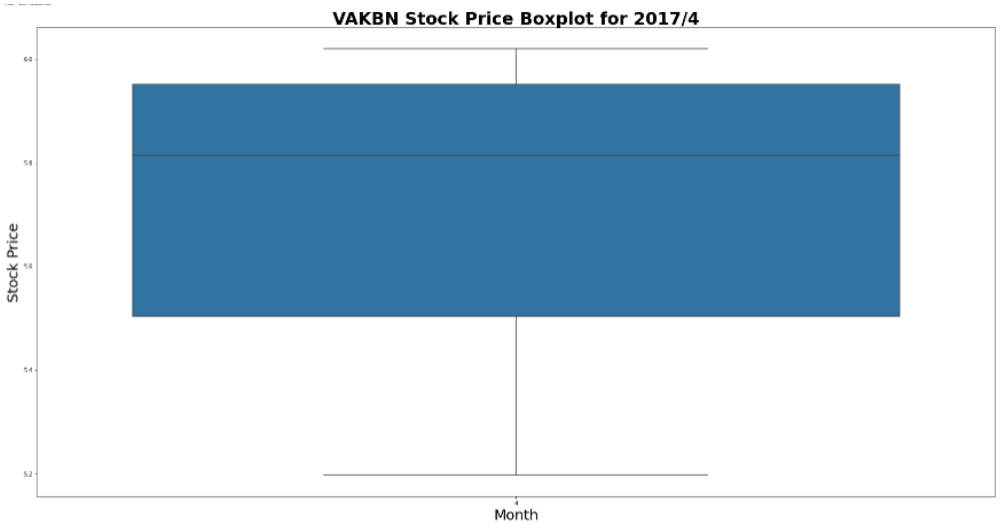

```
Outliers for AKBNK:
Timestamp: 2017-01-31 11:30:00+00:00, Price: 6.7643
Timestamp: 2017-01-31 14:15:00+00:00, Price: 6.7643
Timestamp: 2017-01-31 14:30:00+00:00, Price: 6.7803
```

4.3.2. Box Plot Analysis

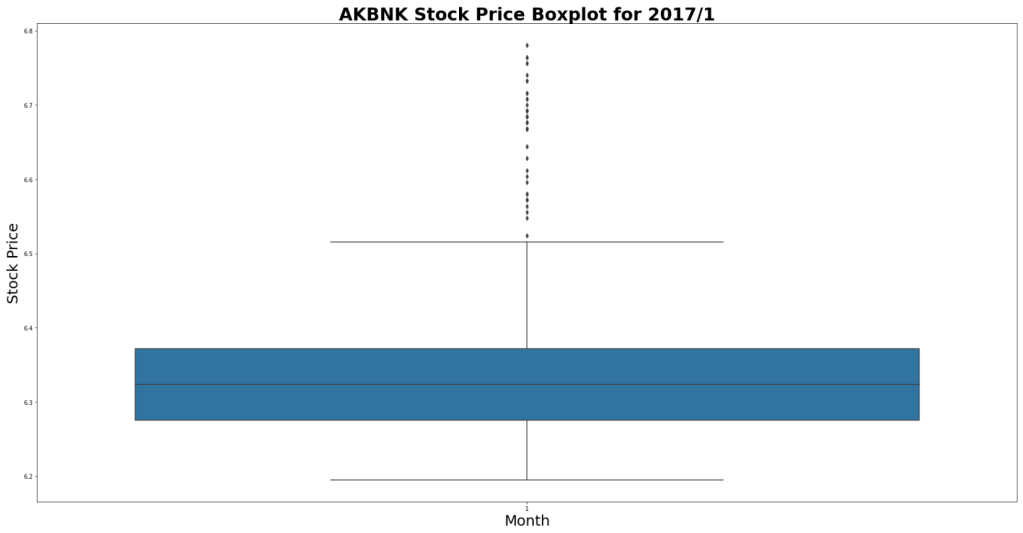
The function plot_box_plot uses the boxplot() function from seaborn. The function is a ready package for the boxplot. It shows the mean, the 25th percentile, the 75th percentile, and the outliers. In addition to this function, detecting and listing the outliers from both the upper and the lower ends is added. If there are no outliers, the plot_box_plot function gives output as “NO OUTLIERS”. There are examples below for different cases.

In [7]:
```
# box plot
def plot_box_plot(selected):
    # getting year and month info from df
    year = selected["year"].iloc[0]
    month = selected["month"].iloc[0]
    # looping over stocks
    for index in indexes_selected:
        # initializing the plot, title, x & y axis
        plt.figure(figsize=(30, 15))
        ax = sns.boxplot(data=selected, x='month', y=index)
        plt.title(str(index) + " " + f'Stock Price Boxplot for {year}/{month}', fontsize=30, fontweight='bold')
        plt.xlabel('Month', fontsize=25)
        plt.ylabel('Stock Price', fontsize=25)

        # identifying and marking both upper and lower outliers
        threshold = 1.5  # Adjust this threshold as needed
        upper_outliers = selected[selected[index] > (selected[index].quantile(0.75) + threshold * (selected[index].quantile(0.75) - selected[index].quan
        lower_outliers = selected[selected[index] < (selected[index].quantile(0.25) - threshold * (selected[index].quantile(0.75) - selected[index].quan

        # printing information about the outliers
        if not upper_outliers.empty:
            print('Upper Outliers for ' + str(index) + " in " + str(year) + "/" + str(month))
            for _, row in upper_outliers.iterrows():
                timestamp = row['timestamp']
                price = row[index]
                print(f'Timestamp: {timestamp}, Price: {price}')

        if not lower_outliers.empty:
            print('Lower Outliers for ' + str(index) + " in " + str(year) + "/" + str(month))
            for _, row in lower_outliers.iterrows():
                timestamp = row['timestamp']
                price = row[index]
                print(f'Timestamp: {timestamp}, Price: {price}')

        if lower_outliers.empty and upper_outliers.empty:
            print("NO OUTLIERS")

        plt.show()

# group by year, month
grouped = df.groupby(['year', 'month'])

# apply the plot_box_plot function to each group
grouped.apply(plot_box_plot)
```

The box plots for all 6 stocks are plotted monthly and the outliers are detected in the jupyter notebook. The data regarding the previous 2 years was used.



NO OUTLIERS




```
Upper Outliers for AKBNK in 2017/1
Timestamp: 2017-01-30 10:45:00+00:00, Price: 6.5238
Timestamp: 2017-01-30 11:00:00+00:00, Price: 6.5478
Timestamp: 2017-01-30 11:15:00+00:00, Price: 6.572
Timestamp: 2017-01-30 11:30:00+00:00, Price: 6.5639
Timestamp: 2017-01-30 11:45:00+00:00, Price: 6.5559
Timestamp: 2017-01-30 12:00:00+00:00, Price: 6.572
Timestamp: 2017-01-30 12:15:00+00:00, Price: 6.5799
Timestamp: 2017-01-30 12:30:00+00:00, Price: 6.5799
Timestamp: 2017-01-30 12:45:00+00:00, Price: 6.572
Timestamp: 2017-01-30 13:00:00+00:00, Price: 6.596
Timestamp: 2017-01-30 13:15:00+00:00, Price: 6.612
Timestamp: 2017-01-30 13:30:00+00:00, Price: 6.604
Timestamp: 2017-01-30 13:45:00+00:00, Price: 6.6281
Timestamp: 2017-01-30 14:00:00+00:00, Price: 6.6441
Timestamp: 2017-01-30 14:15:00+00:00, Price: 6.6681
Timestamp: 2017-01-30 14:30:00+00:00, Price: 6.6842
Timestamp: 2017-01-30 14:45:00+00:00, Price: 6.6842
Timestamp: 2017-01-30 15:00:00+00:00, Price: 6.6842
Timestamp: 2017-01-31 06:45:00+00:00, Price: 6.6761
Timestamp: 2017-01-31 07:00:00+00:00, Price: 6.6921
Timestamp: 2017-01-31 07:15:00+00:00, Price: 6.6921
Timestamp: 2017-01-31 07:30:00+00:00, Price: 6.6921
Timestamp: 2017-01-31 07:45:00+00:00, Price: 6.6681
Timestamp: 2017-01-31 08:00:00+00:00, Price: 6.6681
Timestamp: 2017-01-31 08:15:00+00:00, Price: 6.6681
Timestamp: 2017-01-31 08:30:00+00:00, Price: 6.6681
Timestamp: 2017-01-31 08:45:00+00:00, Price: 6.6761
Timestamp: 2017-01-31 09:00:00+00:00, Price: 6.6761
Timestamp: 2017-01-31 09:15:00+00:00, Price: 6.6842
Timestamp: 2017-01-31 09:30:00+00:00, Price: 6.7082
Timestamp: 2017-01-31 09:45:00+00:00, Price: 6.7163
Timestamp: 2017-01-31 10:45:00+00:00, Price: 6.7002
```

4.3.3. Google Trend Analysis

The outliers obtained from the control chart analysis and the search volume data from Google are used. These values are plotted together by using the function plot_trend() to the line chart and it is analyzed whether there is a correlation between the search volume and the stock price.

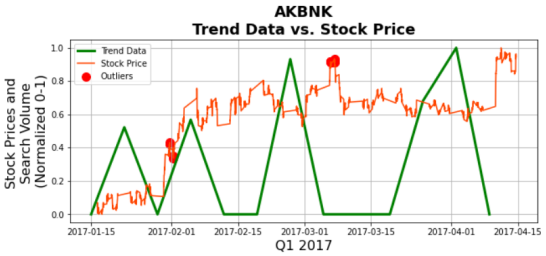
In [8]:
```python
# returns outliers obtained from control charts in part 3
def get_outliers(stock_name):
    if stock_name == "AKBNK":
        return outliers_akbnk
    elif stock_name == "VAKBN":
        return outliers_vakbn
    elif stock_name == "ARCLK":
        return outliers_arclk
    elif stock_name == "TUPRS":
        return outliers_tuprs
    elif stock_name == "TCELL":
        return outliers_tcell
    else:
        return outliers_thyao

'''
Below function plots the quarterly trend, stock prices, and outliers
given the stock name.
'''
def plot_trend(stock_name):
    df_trend = pd.read_csv(f"{stock_name}.csv", header=0)
    df_trend["Week"] = pd.to_datetime(df_trend["Week"])
```

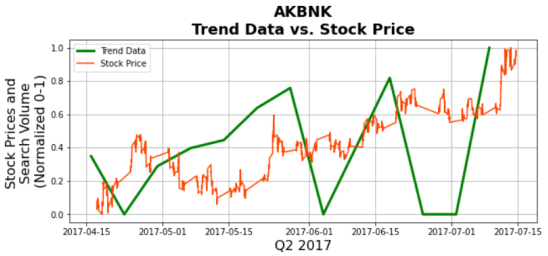
```python
    # start and end dates
    # notice that we observe them quarterly
    # but the outlier data we used is obtained from part 3, therefore monthly
    start = pd.to_datetime("2017-01-15")
    end = start + pd.DateOffset(months=3)

    # initialize the figure
    fig, axes = plt.subplots(4, 2, figsize=(22, 22))
    fig.subplots_adjust(hspace=0.5)

    # we have 8 quarters (2 years)
    for i in range(8):
        # get the outliers for quarter
        outliers = get_outliers(stock_name)
        outliers_quarter = [i for i in outliers if i[0] >= start.tz_localize('UTC') and i[0] <= end.tz_localize('UTC')]
        outliers_quarter = pd.DataFrame(outliers_quarter, columns=['timestamp', 'value'])
        # copy df
        df = filled_df.copy()
        # filtering with time and adding UTC timezone
        ndf = df[(df['timestamp'] >= start.tz_localize('UTC')) & (df['timestamp'] <= end.tz_localize('UTC'))]
        # filtering the necessary columns
        df = ndf[['timestamp', 'year', 'month', 'day', stock_name]].copy()

        # We must scale both of the df between 0-1 since their scale is very different
        scaler = MinMaxScaler(feature_range=(0, 1))
        df[stock_name] = scaler.fit_transform(df[stock_name].values.reshape(-1, 1))
        # scale outliers
        if len(outliers_quarter) > 0:
            outliers_quarter["value"] = scaler.transform(outliers_quarter["value"].values.reshape(-1, 1))
```

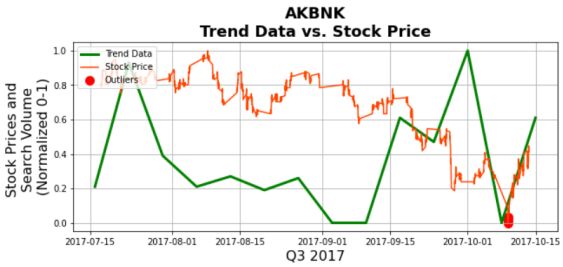
```python
        # filtering df_akbnk with the same timezone
        df_trend_filtered = df_trend[(df_trend['Week'] >= start) & (df_trend['Week'] <= end)]

        # min max scaler
        scaler = MinMaxScaler(feature_range=(0, 1))
        df_trend_filtered[f"IST:{stock_name}: (Türkiye)"] = scaler.fit_transform(df_trend_filtered[f"IST:{stock_name}: (Türkiye)"].values.reshape(-1, 1))

        row = i // 2
        col = i % 2

        # creating a figure and axis
        ax = axes[row, col]
        # plotting the data from df_akbnk_filtered
        ax.plot(df_trend_filtered["Week"], df_trend_filtered[f"IST:{stock_name}: (Türkiye)"], label="Trend Data", color="green", linewidth=3)

        # plotting the data from df_copy (assuming "timestamp" is your x-axis)
        ax.plot(df["timestamp"], df[stock_name], label="Stock Price", color="orangered")

        # plot outliers as red dots
        if len(outliers_quarter) > 0:
            ax.scatter(outliers_quarter['timestamp'], outliers_quarter['value'], c='red', marker='o', s=100, label='Outliers')

        # x, y axis, title and legend
        ax.set_xlabel(f"Q{i % 4 + 1} {start.year}", fontsize=16)
        ax.set_ylabel("Stock Prices and \nSearch Volume \n(Normalized 0-1)", fontsize=16)
        ax.grid(True)
        ax.set_title(f"{stock_name}\nTrend Data vs. Stock Price", fontsize=18, fontweight="bold")
        ax.legend(loc="upper left")

        # off-setting dates
        start = end
        end = start + relativedelta(months=3)

    plt.show()
```

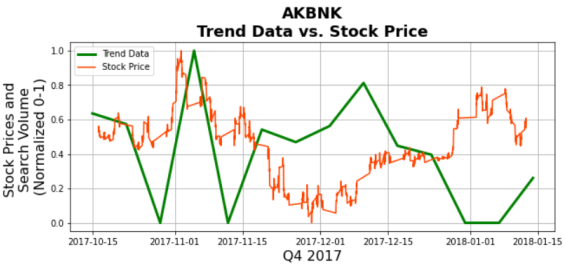
4.1 IST:AKBNK

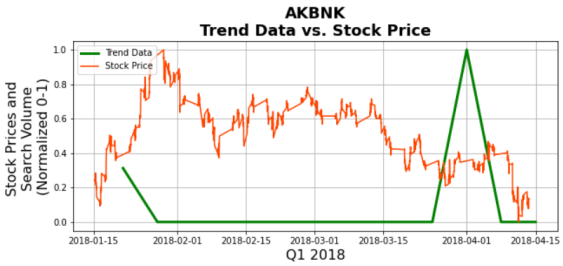
```
In [9]: plot_trend("AKBNK")
```

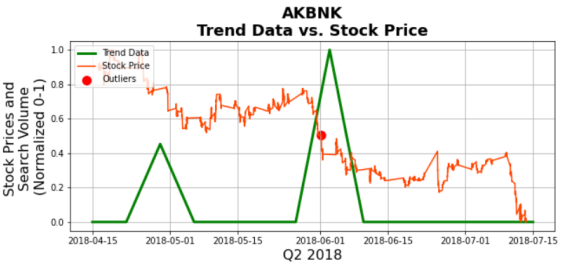


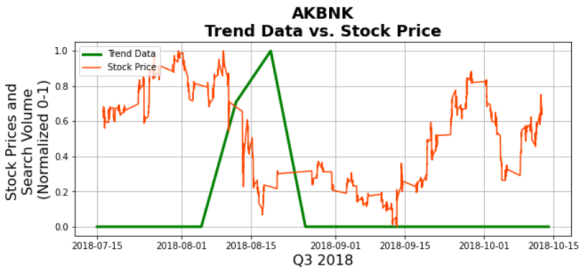


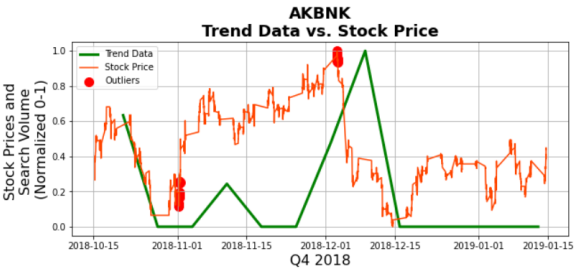






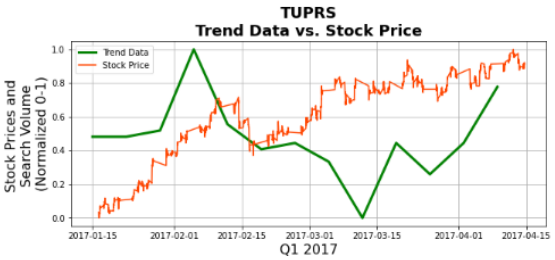


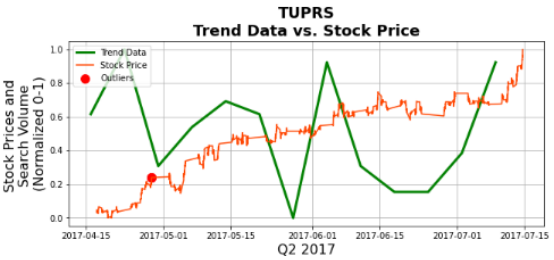


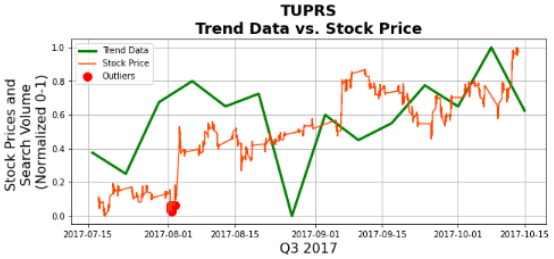


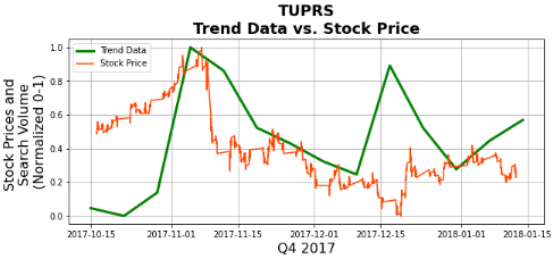
4.4 IST:TUPRS

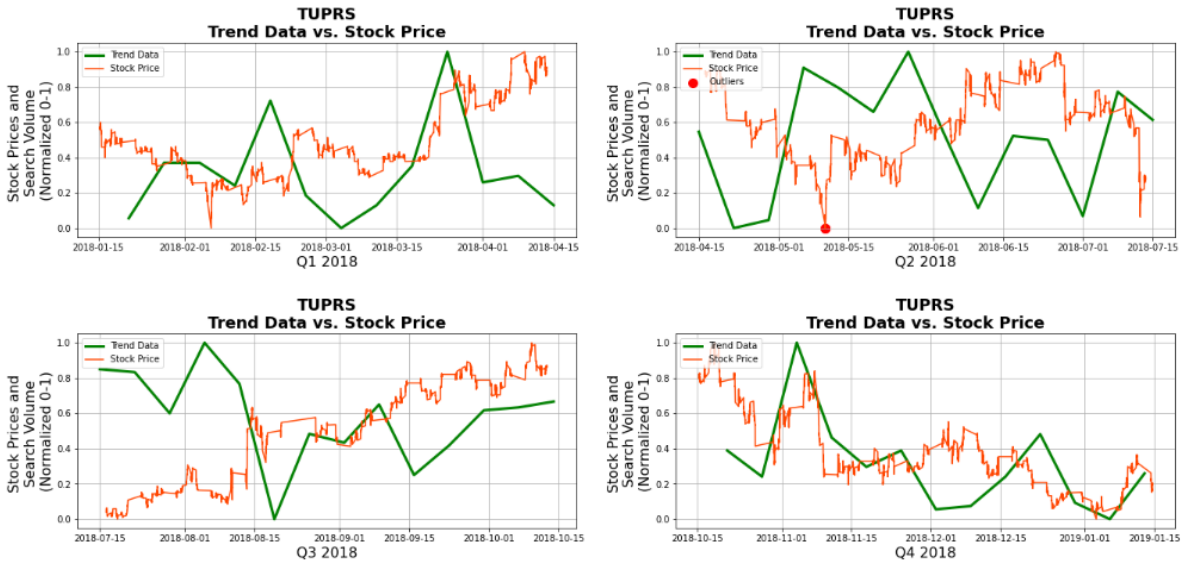
```
In [12]: plot_trend("TUPRS")
```





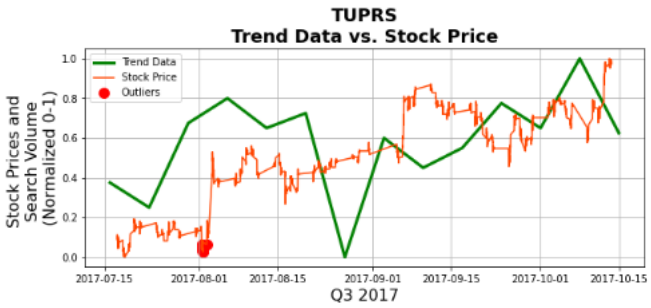




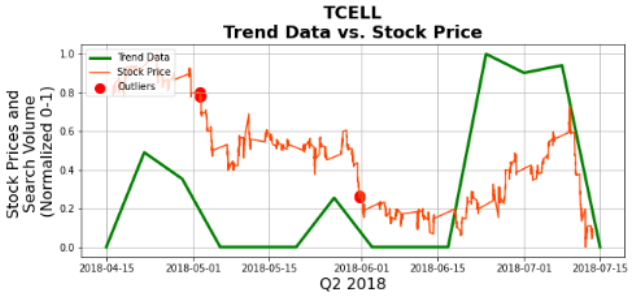


5. Results

Since the outliers are the unexpected decreases or increases of the stock prices, these are usually the points where the search volume increases. People are curious about the current situation of the stock prices under the extreme circumstances, unexpected news, and speculations. Thus, the search volume of the stock prices are correlated as in the cases below.



The outliers in the TUPRS stock prices and the increase in the search volume coincides. It means that there are some bad things happening in Tüpraş so that people are watching what is happening to the stock price under new conditions.

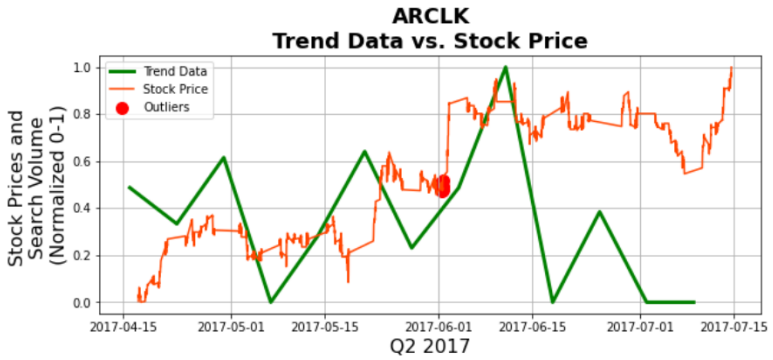


When the case of Turkcell is examined, it can be seen that the outliers are in the places after the large number of searches in Google. It means that something has happened and people are checking the prices everyday.

6. Discussion

One discussion point can be finding the effects of speculations and daily unexpected news on the stock prices. The outliers for some stocks have been examined and it is found that there are some reasonable causes that create an outlier situation.

➔ There is an increase in the stock prices of ARCLK in the second quarter of 2017 that causes outlier values. The main reason for the increase is the reward that is given to Arçelik in those times.

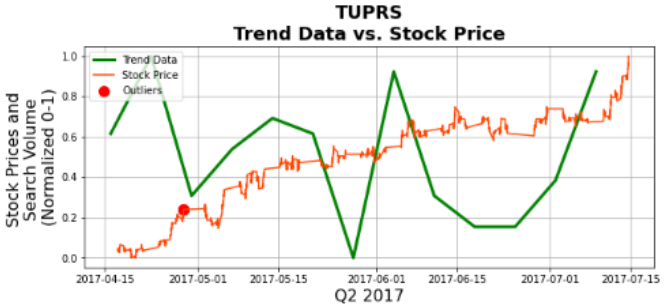


aa.com.tr
https://www.aa.com.tr › arcelik-asye-uc-odul

Arçelik AŞ'ye üç ödül - Anadolu Ajansı

30 May 2017 — **Arçelik** AŞ, Contact Center World Awards'ın Londra'da düzenlenen finalinden üç farklı kategoride birincilik ödülüyle döndü. - Anadolu Ajansı.

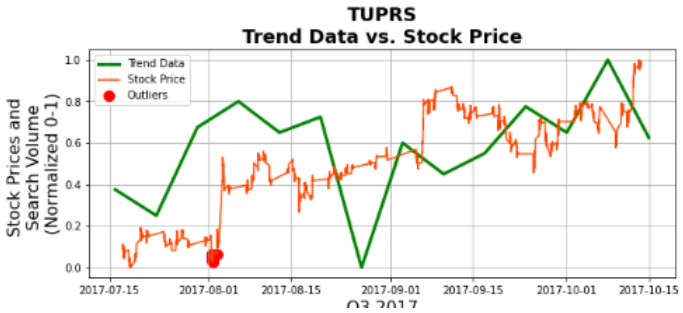
➔ There are huge increases in the stock prices of TUPRS that cause outliers. In those times, annual profit statements and distribution of the profit explanations are made to the public.



dunya.com
https://www.dunya.com › Şirket Haberleri

Tüpraş, 1.5 milyar lira kâr payı dağıtacak - Dünya Gazetesi

30 Mar 2017 — Karara göre yüzde 621,8 nispetinde ve 1 TL nominal değerde bir adet hisse senedine hissedarların vergi mükellefiyetine göre 6,218 TL brüt, 5,2853 TL net nakit ...

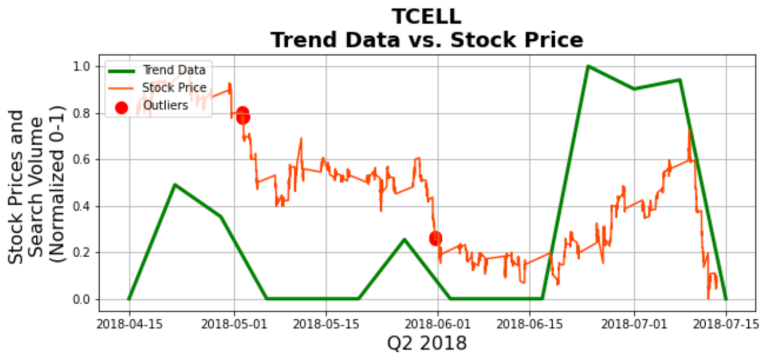


hurriyet.com.tr
https://www.hurriyet.com.tr › Ekonomi

Tüpraş'tan yılın ilk yarısında 2,3 milyar lira net kar - Hürriyet

3 Ağu 2017 — Sonuç olarak 6 aylık satış hasılatını 25 milyar liraya çıkaran **Tüpraş** ilk yarı yılda 2,3 milyar liralık net kar rakamına ulaşmıştır." **Haberin** Devamı. Açıklamada ...

➔ New investment of Turkcellto the new domestic car producing company causes a decrease in the stock prices in the 2nd quarter of 2018. The uncertainty of the company may cause such decrease and outliers.



haberturk.com
https://www.haberturk.com › Ekonomi › İş-Yaşam

Son Dakika: Yerli otomobilde şirket payları açıklandı - Habertürk

2 Haz 2018 — Türkiye'nin Otomobili için kurulan şirkette Anadolu Grubu, BMC, Kök Grubu, **Turkcell** ve Zorlu Holding'in payları yüzde 19'ar, TOBB'un payı yüzde 5 oldu. Yerli ...

7. Conclusion and Recommendations

Real world and real data are full of unexplainable and unexpected variabilities. The aim of the control charts, box plots, and detailed other analysis are finding the points with extreme standard deviations, called outliers, and searching for the reasons. After that, looking for new solutions can be another point to deal with the effects of high variabilities if unexpected results are not wanted.

The stock prices follow random walk as it is said in the economy lessons, but they are not independent and correlated from real life events, wars, speculations, news, politics etc. as it is seen in the study above. As industrial engineers, we are responsible for dealing with the variability, as it is not wanted in manufacturing in terms of quality, by searching for both the defected items & values and the reasons & solutions for them.

8. References

https://www.aa.com.tr/tr/sirkethaberleri/bilisim/arcelik-asye-3-odul/639522

https://www.dunya.com/sirketler/tupras-15-milyar-lira-kar-payi-dagitacak-haberi-355977

https://www.hurriyet.com.tr/ekonomi/tuprastan-yilin-ilk-yarisinda-2-3-milyar-lira-net-kar-40538944

https://www.haberturk.com/son-dakika-yerli-otomobilde-sirket-paylari-aciklandi-1997233-ekonomi