# ANOMALY DETECTION USING K-MEANS CLUSTERING

CHINMAY BAKE

In [1]:

```
from pyspark.sql import SparkSession
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import ClusteringEvaluator
spark = SparkSession.builder.appName('K_Means_Clustering').getOrCreate()
from pyspark.ml.clustering import KMeans

pd_df = pd.read_csv(r'C:\Users\chinm\Downloads\Subscribers.csv')
main_df = spark.read.csv(r'C:\Users\chinm\Downloads\Subscribers.csv',inferSchema=True,header=True)
```

#### DESCRIPTION OF THE PROBLEM

We have time-series data of the count of subscribers in a telecom organisation. Our objective is to build an Anomaly Detecting model which could help define a threshold or a baseline for customer churn operations and specifically assist in distinguishing seasonal data from anomalous data.

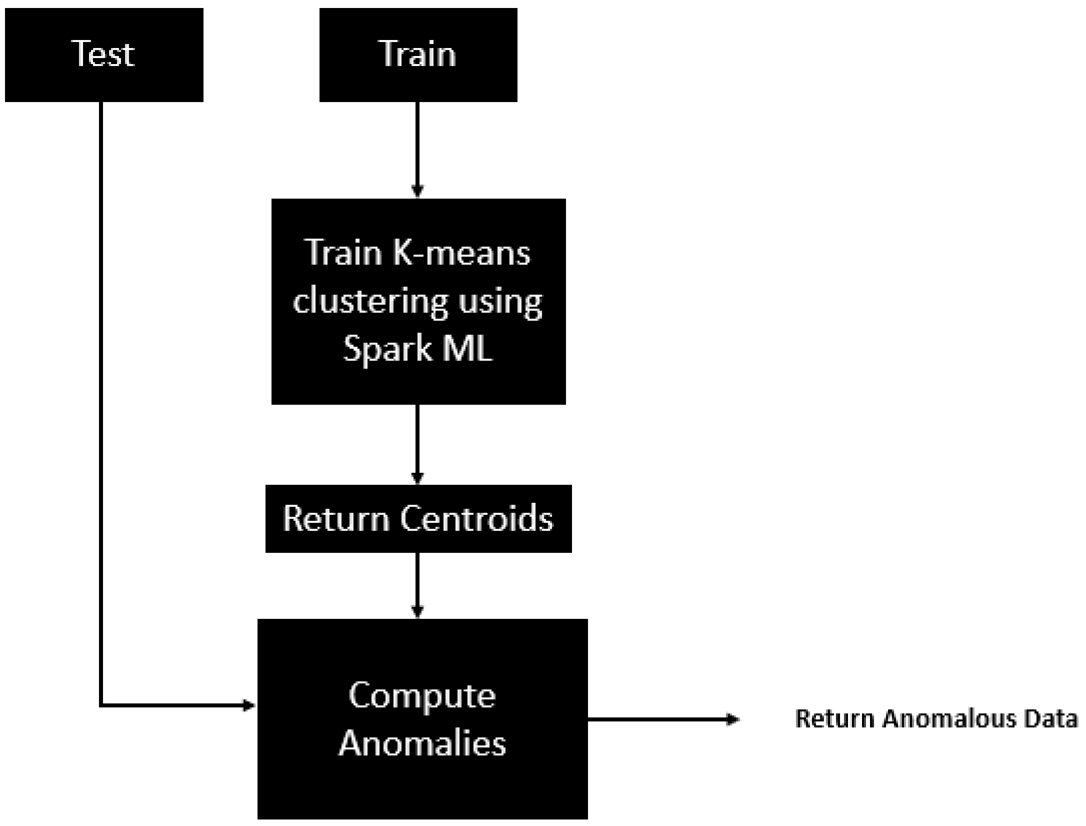
In [2]:

```
main_df.show(10,False)
```

```
+---------------+---------------+
|ACTIVATION_DATE|ALL_SUBSCRIBERS|
+---------------+---------------+
|01-05-2020     |7131           |
|02-05-2020     |6668           |
|03-05-2020     |4155           |
|04-05-2020     |5979           |
|05-05-2020     |5835           |
|06-05-2020     |5849           |
|07-05-2020     |5775           |
|08-05-2020     |5954           |
|09-05-2020     |5752           |
|10-05-2020     |3803           |
+---------------+---------------+
only showing top 10 rows
```

## APPROACH

We first train a K-Means culstering model using Spark ML. It includes evaluating optimal value for k, training the model with that optimal value and returning cluster centroids from the trained model



In [3]:
```
def k_means(transformed_df):

    errors = []
    sil = []
    val_list = []
    ctr = []

    for k in range(2,10):
        kmeans = KMeans(featuresCol='features',k=k).setSeed(2)
        model = kmeans.fit(transformed_df)
        intra_distance = model.computeCost(transformed_df)
        errors.append(intra_distance)

        predictions = model.transform(transformed_df)
        evaluatorObj = ClusteringEvaluator()
        sil.append(evaluatorObj.evaluate(predictions))


        index = errors.index(min(errors))

        if index == errors.index(errors[-1]) or index == errors.index(errors[0]):
            k_value = index

        else:
            for value in [index,index - 1,index + 1]:
                val_list.append(sil[value])
            k_value = val_list.index(max(val_list))


    kmeans = KMeans(featuresCol='features',k=k_value)
    model = kmeans.fit(transformed_df)
    predictions = model.transform(transformed_df)
    centers = model.clusterCenters()

    for center in centers:
        ctr.append(center)

    return ctr
```

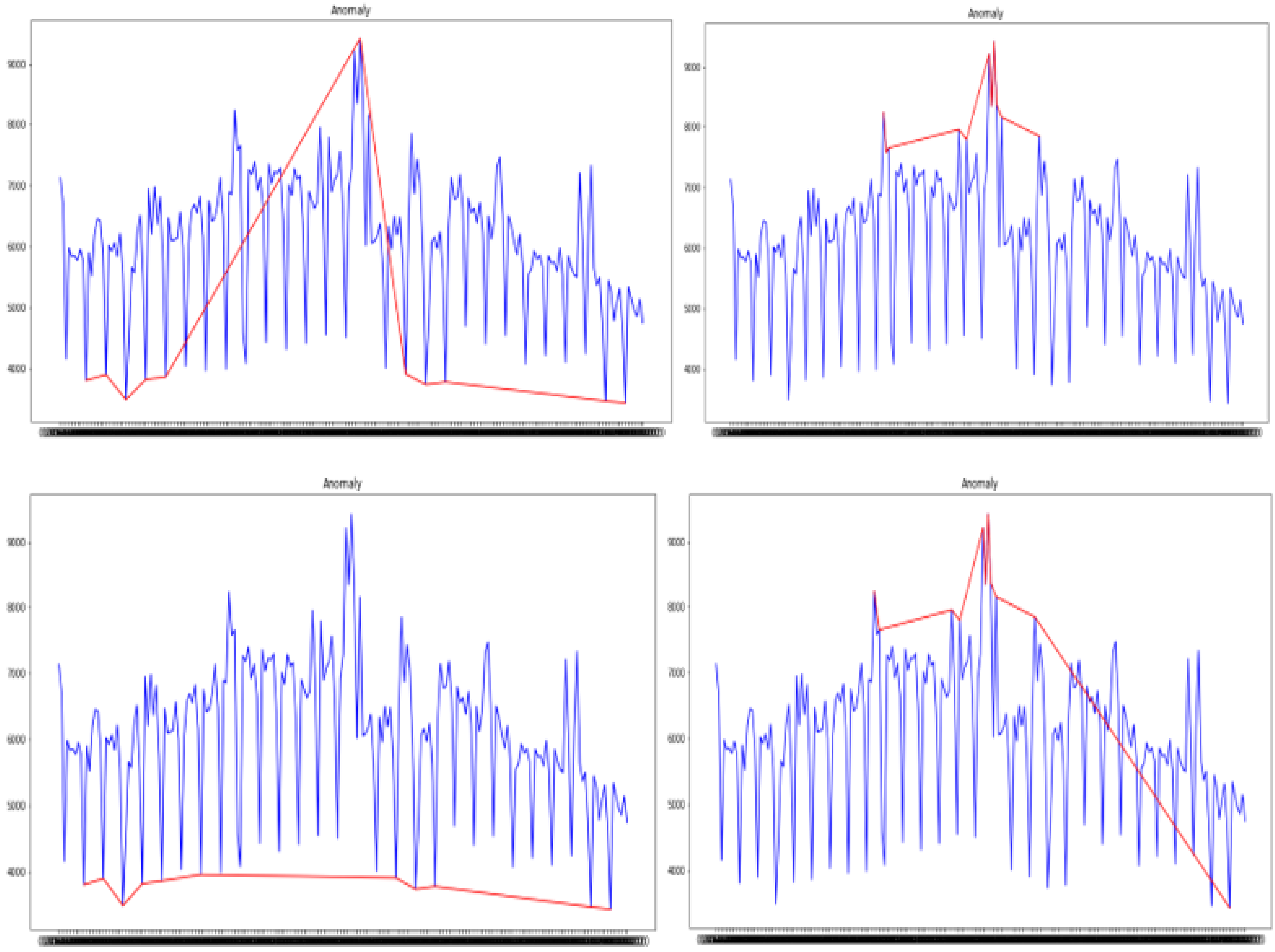
**ALGORITHM FOR ANOMALY DETECTION**

- Find K clusters
- Anomaly score $d\left(x^{(i)}\right)$ – **Distance to closest centroid**
- If $d\left(x^{(i)}\right) > \tau\,;\ x^{(i)}$ **is an anomaly**

$\tau$ could be any threshold value based on suitable assessment of the data. We, in this case have set its value to $\mathbf{95^{th}}$ $\boldsymbol{percentile}$ $of\ the\ array\ of\ anomaly\ scores$.

Threshold values could impact the precision and recall of the model. Smaller threhold values could result in a more sensitive detection mechanism.

In [9]:
```
def anomaly(ctr,sub_col,testing_df):

    for x in range(0,len(ctr),1):

        df1 = pd.DataFrame()
        dp =  list()

        for k in np.sqrt((ctr[x]-sub_col)**2):
            if k > np.percentile(np.sqrt((ctr[x]-sub_col)**2),95):
                dp.append(np.where(np.sqrt((ctr[x]-sub_col)**2) == k))

        for j in range(0,len(dp),1):
                df1 = df1.append(testing_df.loc[dp[j][0][0],:])

        return df1
```




Please note: the above graphics were generated in the analysis phase, and are not generated by any of the given code blocks.

In [ ]:
```
def predict():

    main_df = spark.read.csv(r'C:\Users\chinm\Downloads\Subscribers.csv',inferSchema=True,header=True)
    training_df,testing_df = main_df.randomSplit([0.85,0.15])
    testing_df = testing_df.toPandas()

    input_cols = ['ALL_SUBSCRIBERS']
    vec_assembler = VectorAssembler(inputCols = input_cols, outputCol="features")
    transformed_df = vec_assembler.transform(training_df)


    ctr = k_means(transformed_df)

    sub_col = testing_df['ALL_SUBSCRIBERS']
    sub_col = np.array(sub_col)


    anomalies = anomaly(ctr,sub_col,testing_df)

    return anomalies
```

In [11]:
```
predict()
```

Out[11]:

| | ACTIVATION_DATE | ALL_SUBSCRIBERS |
|---|---|---|
| 8 | 08-11-2020 | 3464.0 |
| 18 | 15-11-2020 | 3429.0 |

## ANALYSIS OF THE RESULT

The predict function returns data points classified as anomalous by the anomaly function defined above. Accuracy of the prediction is contextual. If in future, the model classifies data points with similar dates, or rather time of the year (for a yearly time-series), then the decision making process could pin point towards such dates, further identifying any trend or pattern in them. As against that, if certain data points have very unqiue time occurences, then such instances could be invesitaged for other impacting factors.

## REFERENCES


- https://www.janospoor.com/posts/2019-07-20-detecting-process-anomalies/ (https://www.janospoor.com/posts/2019-07-20-detecting-process-anomalies/)
- https://www.youtube.com/watch?v=69QYeZC_dFU&list=PLeEuH8so9u0JYokE0gY-tpwI6Cper9k6C&index=6&t=2296s (https://www.youtube.com/watch?v=69QYeZC_dFU&list=PLeEuH8so9u0JYokE0gY-tpwI6Cper9k6C&index=6&t=2296s)
- https://web.stanford.edu/class/cs345a/slides/12-clustering.pdf (https://web.stanford.edu/class/cs345a/slides/12-clustering.pdf)