Project -1 Report

DATA MINING

CSE 572: SPRING 2020

SUBMITTED TO:

Professor Ayan Banerjee
Ira A. Fulton School of Engineering
Arizona State University

SUBMITTED BY:

Rishti Gupta
rgupta75@asu.edu
1217211814

1. Introduction:

Continuous glucose monitoring (CGM) is a method of continuously monitoring glucose levels in the interstitial fluid as a basis for improving metabolic control. The task in the project is to extract features from the given glucose data at lunchtime of the patients and time series when the glucose values are recorded in Continuous Glucose Monitor (CGM).

In the project, the selection of a particular feature for each time series has been explained and the significance is validated by its value. After feature extraction, the feature matrix is created and subsequently passed to the PCA. Principal Component Analysis (PCA) on the feature matrix selected the top 5 most important features (considering the maximum variance), and then the features were plotted for each time series. The eigenvectors obtained from the PCA were orthogonal to each other. The plots obtained by performing PCA demonstrates the most relevant features for our analysis.

2. Project Description:

- Data was collected from a continuous glucose monitor(CGM) and provided. The given data includes:
- The first cell array has tissue glucose levels every 5 mins for 2.5 hrs during a lunch meal. The data starts from 30 mins before meal intake and continues up to 2 hrs after the start of meal consumption.
- The second cell array has timestamps of each time series in the first cell array.
- The third cell array has insulin basal infusion input time series at different times during the 2.5 hr time interval.
- The fourth cell array has timestamps for each basal or bolus insulin delivery time series.
- The fifth cell array has an insulin bolus infusion input time series at different times during the 2.5 hr time interval.

Phase 1 of the project expects to find features, analyze and discuss if the features selected are appropriate and apply PCA to the feature matrix to get the top 5 features.

3. Project Task: Feature Extraction

Four features were extracted from the CGM glucose level array and CGM timestamp cell array. Some preprocessing had to be done before feature extraction.

Preprocessing: The first step was to preprocess the data to combine (concatenate) all the data from different files. The observations with missing values were removed to make our analysis more effective.

Following features were extracted from the data:

- Rolling Mean and Rolling Standard Deviation
- Polyfit Regression
- Fast Fourier Transform (FFT)
- Interquartile Range (IQR)

3.1 Rolling Mean and Rolling Standard Deviation:

Python library used: pandas

Pandas `dataframe.rolling()` function provides the feature of rolling window calculations. The concept of rolling window calculation is most primarily used in signal processing and time-series data[1]. Two statistics were chosen for the CGM data:

Significance:

- Rolling Mean `(pandas.rolling_mean)`:
  Rolling mean was calculated to find the moving average for the CGM glucose levels for all the patients (extracted from the full concatenated data).
- Rolling Standard Deviation `(pandas.rolling_std)`:
  Rolling standard deviation was calculated to find the moving standard deviation for the CGM glucose levels for all the patients(extracted from the full concatenated data).

3.2 Polyfit Regression:

Python library used: NumPy

Numpy `numpy.polyfit()` function provides least-squares polynomial fit. It fits a polynomial p(x) = p[0] * x**deg + ... + p[deg] of degree deg to points (x, y) and returns a vector of coefficients p that minimizes the squared error in the order deg, deg-1, … 0. The Polyfit Regression used in the project was of degree 3. Thus, the feature extracted from this is the coefficients of the polynomial[2].

Significance: In the project, Polyfit Regression is used to fit the data and study the trend in glucose levels. The resulting curve is used to estimate the future glucose levels and in calculating the rate of change of glucose levels.

3.3 Fast Fourier Transform (FFT):

Python library used: NumPy

NumPy `numpy.fft.fft()` computes the one-dimensional discrete Fourier Transform. This function computes the one-dimensional n-point discrete Fourier Transform (DFT) with the efficient Fast Fourier Transform (FFT) algorithm. Fast Fourier Transform technique samples a signal over a period of time and decomposes it into frequency components. FFT transforms the data from the time domain to the frequency domain, which will help in the analysis of data[3].

Significance: The data was converted from time-domain (given format of the dataset) to frequency- domain for better understanding and analysis. The FFT or Fast Fourier Transform is an algorithm that essentially uses convolution techniques to efficiently find the magnitude and location of the tones (here our input CGM Series data) that make up the signal of interest. Therefore, this will be particularly useful for understanding the data better and subsequently in forecasting applications.

3.4 Interquartile Range (IQR):

Python library used: NumPy

Interquartile range using `numpy.median`. The interquartile range (IQR) is a measure of variability, based on dividing a data set into quartiles. Quartiles divide a rank-ordered data set into four equal parts. The values that divide each part are called the first, second, and third quartiles; and they are denoted by Q1, Q2, and Q3, respectively. Formula: IQR = Q3 – Q1[4].

Significance: In our project, we will use this function as it will generate how spread out the middle 50% of each distribution curve is. Thus, when the glucose level is high, it will give the time taken for each rise and fall of the glucose level.

4. Feature Results:

4.1 Rolling Mean:

```
In [51]: rolling_mean
```

Out[51]:

	cgmSeries_1	cgmSeries_2	cgmSeries_3	cgmSeries_4	cgmSeries_5	cgmSeries_6	cgmSeries_7	cgmSeries_8	cgmSeries_9	cgmSeries_10	...	cgmSeries_33
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	249.333333	256.000000	263.666667	269.000000	273.333333	277.000000	278.666667	279.000000	280.000000	278.333333	...	0.0
3	240.000000	247.000000	251.000000	255.000000	260.333333	264.666667	267.666667	269.333333	271.666667	271.000000	...	0.0
4	199.000000	201.666667	203.000000	204.666667	208.333333	212.000000	214.000000	213.333333	212.333333	210.000000	...	0.0
5	192.666667	194.333333	192.000000	190.333333	191.000000	193.000000	195.000000	197.000000	198.000000	195.333333	...	0.0
6	171.333333	173.000000	174.666667	175.666667	176.666667	180.666667	185.000000	188.333333	189.666667	189.000000	...	0.0
7	201.666667	204.000000	205.000000	205.000000	203.000000	205.666667	208.333333	209.000000	208.333333	205.333333	...	0.0
8	193.000000	194.333333	195.000000	195.000000	193.666667	195.333333	195.333333	193.666667	191.666667	190.000000	...	0.0
9	183.000000	183.000000	183.333333	183.333333	183.000000	183.333333	183.000000	182.000000	182.333333	185.333333	...	0.0
10	180.000000	179.333333	180.000000	180.666667	182.000000	183.000000	184.333333	185.666667	187.333333	192.000000	...	0.0
11	193.333333	194.333333	196.666667	199.000000	201.666667	203.333333	204.666667	204.333333	204.000000	204.666667	...	0.0
12	211.000000	213.666667	216.333333	218.000000	220.000000	221.000000	221.333333	219.666667	218.000000	214.000000	...	0.0
13	190.333333	194.666667	200.666667	207.000000	210.666667	211.666667	212.666667	211.000000	210.666667	206.333333	...	0.0
14	155.000000	158.333333	162.333333	166.666667	169.666667	171.000000	172.000000	171.333333	171.666667	168.666667	...	0.0
15	161.666667	165.000000	168.666667	174.000000	176.666667	176.666667	175.333333	173.000000	169.666667	163.333333	...	0.0
16	181.333333	183.666667	183.666667	183.666667	183.666667	182.666667	180.000000	177.666667	174.000000	168.666667	...	0.0
17	167.666667	168.333333	168.000000	166.666667	165.333333	163.666667	163.000000	164.000000	165.000000	165.000000	...	0.0
18	158.666667	158.666667	157.333333	153.666667	149.666667	146.000000	146.333333	148.333333	152.333333	156.333333	...	0.0
19	132.000000	134.000000	135.333333	134.666667	134.000000	135.000000	138.666667	142.666667	148.000000	151.000000	...	0.0
20	147.666667	153.666667	159.000000	164.333333	168.666667	175.000000	180.666667	185.000000	189.000000	186.666667	...	0.0
21	118.666667	123.666667	129.333333	136.666667	143.666667	153.000000	159.333333	163.333333	167.000000	163.666667	...	0.0
22	123.333333	128.666667	134.333333	143.000000	151.000000	158.000000	162.333333	164.000000	165.000000	160.666667	...	0.0
23	104.666667	112.000000	120.666667	129.333333	138.000000	143.666667	146.666667	146.000000	144.666667	142.666667	...	0.0
24	132.333333	143.333333	154.666667	165.000000	175.333333	183.000000	187.000000	184.333333	181.666667	178.666667	...	0.0
25	114.666667	123.333333	133.000000	140.333333	147.333333	153.666667	157.000000	156.333333	156.333333	156.666667	...	0.0
26	114.000000	119.333333	125.000000	131.000000	136.666667	142.333333	145.666667	146.000000	146.666667	147.333333	...	0.0
27	88.000000	91.333333	98.666667	106.000000	112.000000	116.666667	118.666667	121.000000	123.333333	126.666667	...	0.0
28	119.000000	119.000000	123.333333	128.333333	132.666667	136.000000	136.666667	138.000000	139.666667	140.666667	...	0.0
29	151.666667	148.666667	149.666667	149.000000	147.666667	147.333333	145.000000	143.666667	142.666667	142.333333	...	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
40	163.333333	176.333333	184.333333	189.333333	193.000000	198.000000	199.666667	204.666667	209.000000	206.666667	...	0.0

40	163.333333	176.333333	184.333333	189.333333	193.000000	198.000000	199.666667	204.666667	209.000000	206.666667	...	0.0
41	170.333333	185.000000	194.333333	201.333333	207.000000	211.333333	213.000000	217.333333	220.333333	222.666667	...	0.0
42	207.666667	216.333333	222.333333	228.333333	233.666667	235.333333	237.000000	235.666667	231.000000	226.333333	...	0.0
43	210.000000	216.000000	221.333333	227.666667	234.000000	236.666667	238.333333	238.666667	236.333333	234.333333	...	0.0
44	223.000000	225.000000	227.000000	229.333333	230.333333	228.666667	228.333333	228.333333	229.333333	232.666667	...	0.0
45	237.000000	240.666667	242.666667	245.666667	246.666667	244.000000	243.666667	245.333333	246.666667	249.666667	...	0.0
46	206.333333	211.000000	215.666667	221.000000	222.333333	218.666667	217.333333	218.666667	220.333333	223.333333	...	0.0
47	239.000000	245.333333	251.666667	258.000000	259.666667	255.666667	253.666667	259.333333	261.000000	262.666667	...	0.0
48	215.666667	220.000000	225.666667	231.333333	232.666667	229.333333	227.000000	227.666667	223.333333	223.666667	...	0.0
49	200.333333	201.000000	203.666667	208.000000	211.333333	211.666667	212.666667	216.666667	216.000000	220.000000	...	0.0
50	176.333333	177.333333	179.666667	184.333333	186.333333	186.666667	186.333333	182.666667	176.333333	175.333333	...	0.0
51	176.666667	177.666667	178.333333	180.333333	180.333333	181.333333	182.333333	180.333333	180.000000	180.666667	...	0.0
0	223.666667	227.000000	228.000000	228.333333	224.000000	220.666667	216.666667	209.000000	203.000000	197.333333	...	0.0
1	222.000000	226.333333	228.000000	228.666667	219.333333	215.000000	211.333333	203.666667	202.000000	200.333333	...	0.0
2	256.666667	260.000000	263.333333	262.666667	254.000000	249.333333	245.666667	239.333333	237.666667	235.333333	...	0.0
3	301.333333	302.000000	176.333333	175.666667	168.333333	164.666667	163.000000	158.666667	158.666667	158.333333	...	0.0
4	355.333333	350.000000	221.000000	218.000000	215.333333	214.666667	212.000000	207.000000	203.666667	199.333333	...	0.0
5	281.333333	276.666667	146.666667	144.333333	142.333333	142.000000	138.333333	133.666667	131.666667	130.666667	...	0.0

5	281.333333	276.666667	146.666667	144.333333	142.333333	142.000000	138.333333	133.666667	131.666667	130.666667	...	0.0
6	240.333333	233.333333	242.000000	246.666667	249.000000	251.666667	243.666667	236.666667	227.333333	223.333333	...	0.0
7	187.333333	186.000000	198.000000	202.333333	203.333333	202.333333	196.000000	193.333333	187.333333	186.000000	...	0.0
8	256.333333	256.000000	268.666667	277.333333	276.666667	275.333333	263.666667	260.333333	253.000000	249.333333	...	0.0
9	201.000000	204.666667	208.333333	213.666667	212.000000	212.666667	158.333333	218.666667	233.333333	239.000000	...	0.0
10	207.000000	209.000000	212.333333	221.000000	224.000000	230.333333	174.000000	230.333333	244.000000	249.666667	...	0.0
11	164.000000	163.666667	166.333333	170.333333	119.666667	186.666667	138.666667	196.000000	209.000000	215.333333	...	0.0
12	191.666667	193.666667	196.666667	200.333333	149.333333	212.333333	212.000000	208.666667	208.000000	207.666667	...	0.0
13	167.333333	168.666667	170.000000	171.666667	118.000000	178.666667	181.666667	180.666667	180.333333	179.000000	...	0.0
14	178.333333	182.666667	185.666667	190.333333	193.000000	194.333333	196.000000	196.666667	198.000000	196.333333	...	0.0
15	224.333333	225.666667	228.666667	232.000000	230.333333	228.333333	226.666667	222.666667	218.666667	211.333333	...	0.0
16	266.666667	268.666667	273.666667	277.000000	274.666667	271.666667	264.666667	259.666667	255.333333	247.000000	...	0.0
17	279.666667	281.333333	283.333333	284.666667	277.333333	270.333333	258.666667	250.666667	244.000000	234.000000	...	0.0

216 rows × 42 columns

Sample output: The following shows the stem and plot for the rolling mean of record no. 32 of the concatenated dataset:

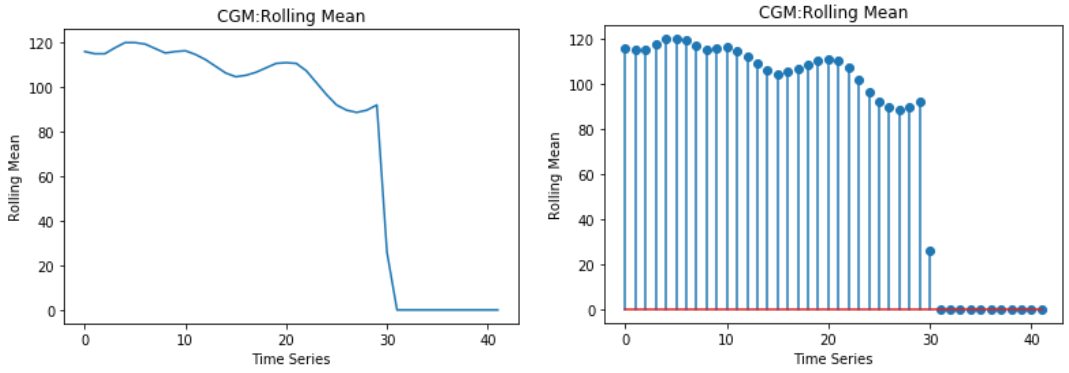


Fig 1. Rolling Mean of CGM series v/s time series

Intuition: Mean is a good feature considering that all the attributes are taken into account. As one of the attributes can have extremely large or small values, the mean can be biased towards that specific attribute. Basically it means that mean can be easily prone to noise. To avoid this particular issue and still take the feature, rolling (window) mean has been considered. It is a calculation of a series of averages of different subsets of the full data set. The window selected = 3 which reduces the dataset with 30 columns to 10 columns averaging out the values.

Rolling Standard Deviation:

Out[415]:

	cgmSeries_1	cgmSeries_2	cgmSeries_3	cgmSeries_4	cgmSeries_5	cgmSeries_6	cgmSeries_7	cgmSeries_8	cgmSeries_9	cgmSeries_10	...	cgmSeries_33
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
2	15.143756	18.083141	17.785762	19.924859	26.083200	29.816103	35.232561	42.567593	50.408333	57.500725	...	0.0
3	17.435596	22.715633	30.610456	38.039453	44.060564	47.648015	51.791248	56.518434	62.042996	67.638746	...	0.0
4	53.730811	56.083271	56.929781	58.824598	58.286648	58.557664	59.506302	57.011695	56.011903	54.110997	...	0.0
5	48.788660	50.292478	46.032597	42.735621	38.157568	35.538711	34.698703	35.000000	36.290495	33.306656	...	0.0
6	38.527047	39.344631	38.214308	36.637867	32.254199	30.088758	31.048349	32.562760	34.645827	32.046841	...	0.0
7	33.471381	32.908965	31.224990	29.546573	27.874720	25.106440	22.300972	19.974984	20.404248	17.559423	...	0.0
8	32.419130	32.316147	31.224990	30.049958	28.536526	27.465129	25.813433	22.501852	19.139836	15.716234	...	0.0
9	47.127487	48.569538	47.521925	46.003623	43.000000	42.548012	39.736633	34.394767	28.041636	18.823744	...	0.0
10	42.930176	43.385866	42.720019	42.099089	41.509035	42.035699	41.860881	40.377386	36.350149	30.347982	...	0.0
11	51.539629	53.454030	53.928966	54.442630	54.243279	53.715299	51.052261	46.198846	39.849718	28.183920	...	0.0
12	21.166010	20.526406	20.744477	22.649503	23.643181	23.895606	22.501852	19.655364	15.620499	13.000000	...	0.0
13	35.118846	34.645827	31.533051	27.874720	27.135463	26.312228	23.072350	18.520259	13.613719	6.806859	...	0.0
14	33.045423	34.645827	38.135723	42.253205	43.878620	44.192760	47.759816	50.639247	54.353779	58.620247	...	0.0
15	43.189505	44.575778	46.651188	50.477718	51.032669	49.571497	50.540413	51.971146	52.880368	54.811799	...	0.0
16	52.443621	53.519467	55.193599	56.871200	56.002976	54.307765	54.836119	56.571489	57.662813	61.158265	...	0.0
17	76.054805	80.039574	82.310388	86.315313	87.757241	87.202829	84.184322	80.074965	72.958893	67.268120	...	0.0
18	70.465121	73.819600	75.341445	77.674535	77.822447	77.323994	75.962710	73.227955	68.573561	65.041013	...	0.0
19	50.119856	53.702886	55.895736	59.214300	60.621778	63.498031	65.736849	65.378386	62.385896	57.419509	...	0.0
20	28.360771	24.946610	19.467922	11.150486	0.577350	13.856406	21.385353	26.000000	28.583212	24.214321	...	0.0

45	60.232881	64.384263	66.032820	69.284438	69.945217	69.907081	68.068593	67.678160	63.516402	56.127830	...	0.0
46	71.842420	73.627441	72.279550	74.101282	74.002252	72.224188	69.787774	69.428620	65.957057	59.542702	...	0.0
47	70.000000	68.806492	64.655497	62.745518	61.614392	58.968918	57.830211	61.010928	61.489837	61.492547	...	0.0
48	55.075705	53.000000	50.163067	47.226405	46.360903	44.455971	44.844175	55.293158	63.893140	70.088040	...	0.0
49	79.657601	83.144453	84.346508	83.468557	79.027421	71.065697	65.607418	68.529799	70.547856	72.807967	...	0.0
50	55.940445	59.002825	59.079043	58.858588	53.106811	45.003704	37.166293	24.131584	7.637626	5.507571	...	0.0
51	56.323471	59.366096	57.709040	54.884728	47.056703	39.310728	32.578111	21.361960	13.114877	10.503968	...	0.0
0	28.676355	28.478062	29.103264	28.360771	28.583212	28.936713	27.428695	28.792360	30.512293	30.989245	...	0.0
1	31.048349	29.365513	29.103264	28.041636	33.471381	35.369478	33.946036	35.133080	31.764760	27.227437	...	0.0
2	66.040392	61.024585	58.534890	52.519838	59.556696	62.131581	63.532144	65.041013	63.002645	58.620247	...	0.0
3	99.946652	93.616238	163.530833	160.935805	157.455814	155.500268	154.786950	152.007675	151.163267	149.078279	...	0.0
4	32.020827	31.432467	191.580270	189.113722	186.808280	186.360761	183.869519	179.424079	176.437902	172.653796	...	0.0
5	156.234866	152.582218	174.737899	174.316761	172.378460	173.225287	169.918608	164.943425	160.182188	155.052679	...	0.0
6	127.798018	122.316529	125.904726	132.258585	135.281189	139.173752	135.809916	133.155298	123.799569	116.091918	...	0.0
7	81.770003	77.252832	93.402355	106.025154	113.632448	119.738604	114.895605	113.094356	101.026399	94.063808	...	0.0
8	57.492028	55.027266	54.123316	62.978832	69.082077	77.086531	69.356567	63.877487	54.616847	51.393904	...	0.0
9	105.128493	106.025154	103.500403	102.275771	94.398093	87.374672	146.827563	56.615663	46.231303	45.530210	...	0.0
10	105.128493	106.127282	103.712744	102.132267	93.824304	84.913682	153.453576	50.401720	33.045423	29.091809	...	0.0
11	58.660038	58.483616	57.882064	55.536775	116.629042	49.943301	122.463600	23.430749	27.874720	35.472994	...	0.0
12	18.502252	19.756855	23.007245	29.905406	129.639243	30.892286	24.979992	23.965253	26.888659	26.576932	...	0.0
13	23.692474	25.794056	28.213472	32.654760	109.027519	39.501055	39.551654	41.501004	43.096790	44.034078	...	0.0
14	36.637867	39.513711	41.476901	43.661577	47.031904	48.086727	47.759816	50.063293	51.156622	52.309974	...	0.0
15	90.533603	89.745938	92.737982	95.252297	94.113407	90.522557	85.500487	80.686637	74.648063	68.237331	...	0.0
16	53.500779	50.500825	52.012819	53.507009	51.052261	47.056703	43.316663	36.473735	29.022979	20.297783	...	0.0
17	35.725808	33.261589	38.188131	42.665365	47.226405	49.013604	51.597804	49.003401	45.705580	40.261644	...	0.0

216 rows × 42 columns

Sample output: The following shows the stem and plot for the rolling standard deviation of record no. 32 of the concatenated dataset:

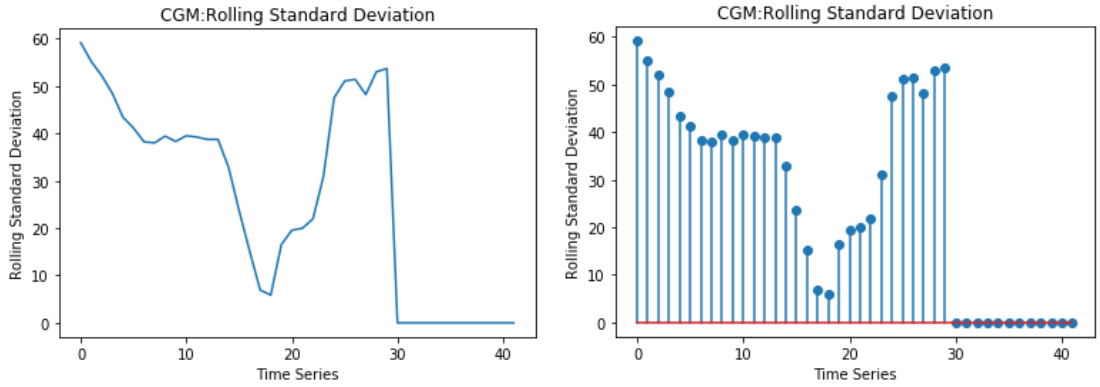


Fig 2. RollingStandard Deviation of CGM series v/s time series

Intuition: Variance/standard deviation measures how far attribute values are spread out from their mean value. We want to select the features which have highly contrasting values and ignore the features which have comparatively similar values. Variance plays a big role in selecting these features. As the mean is prone to noise, the variance will be affected if we take the whole average and

accordingly calculate the variance. In order to tackle this concern, rolling variance is considered which is basically calculated on the windowed mean.

4.2 **Polyfit Regression:**

The result for polyfit regression:

3-degree polynomial: $y = ax^3 + bx^2 + cx + d$.

The result enumerates the coefficients of the equation and is of the form *array[a, b, c, d]*:

```
In [471]: polyfit_reg = pd.DataFrame(polyfit_reg)
          polyfit_reg
```

Out[471]:

	0	1	2	3
0	0.005468	-0.307896	-3.284815	269.159435
1	0.019269	-1.684724	30.168128	216.178160
2	0.012348	-0.837411	7.532055	242.069163
3	0.003728	-0.225240	-2.986245	234.573778
4	0.007679	-0.676247	11.426392	116.273902
5	0.009694	-0.644537	4.906294	210.997315
6	0.007198	-0.536740	5.640660	162.929394
7	0.004856	-0.274221	-2.948853	240.057821
8	0.003645	-0.318468	2.043486	181.014933
9	0.010614	-0.914858	16.653135	99.767475
10	0.010956	-0.755213	7.218155	212.226216
11	0.005341	-0.252492	-4.805401	253.615927
12	0.007592	-0.563696	5.528153	184.070405
13	0.011496	-0.814262	9.990445	162.517400
14	0.001374	-0.187867	2.134435	111.348737
15	0.001003	-0.083175	-4.139840	223.100909
16	0.006379	-0.504835	4.348232	210.525219
17	-0.004231	-0.232130	14.103396	23.683882
18	0.004676	-0.304047	0.084320	177.365079

194	0.002838	-0.184962	-3.105174	228.800664
195	0.007646	-0.876993	20.280912	74.756233
196	-0.000636	-0.044414	-2.892188	206.849794
197	0.002315	-0.179949	-2.358954	219.438471
198	0.010795	-0.704387	4.493103	248.745763
199	0.003618	-0.451023	7.162393	176.140307
200	0.010715	-0.896837	10.711503	290.494379
201	-0.047133	3.011911	-52.779648	252.021578
202	0.004904	-0.478586	2.168076	332.863217
203	-0.001304	-0.252710	9.675663	63.629786
204	0.009543	-0.613743	1.768827	293.458136
205	0.005915	-0.453352	3.550368	191.318131
206	0.015752	-0.978470	6.333051	301.049733
207	0.022874	-1.714723	30.619334	48.238196
208	0.014044	-1.010844	12.788222	198.394812
209	0.004388	-0.475766	8.011440	139.838417
210	-0.005519	-0.037523	4.589354	185.060472
211	-0.004735	0.034029	2.171009	128.814725
212	0.009704	-0.792154	10.745385	198.179066
213	0.009984	-0.497829	-4.503622	338.954562
214	0.004381	-0.376155	0.834719	267.114031
215	-0.001684	0.022157	-5.033171	247.448975

216 rows × 4 columns

Sample output: The following shows the plot for the polynomial fit of degree 3 for the record no. 10 of the concatenated dataset:

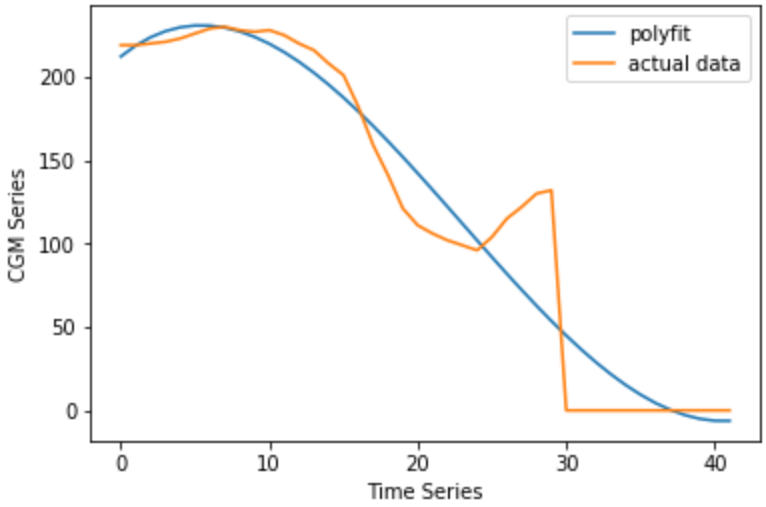


Fig 3. Polyfit (degree=3) foe CGM series v/s time series for record no. 10 of the concatenated data

Intuition:

Polynomial fit fits a model to the actual data. Thus the above plot clearly indicates the actual data of CGM Series being fitted by a 3-degree polynomial. The graph validates our model and hence proves to be a good feature for the principal component analysis.

4.3 **Fast Fourier Transform (FFT):**

The matrix after applying FFT to the CGM series data, FFT values and FFT frequencies were extracted. The resultant matrices are shown as follows:

FFT Values:

In [623]:
```
Fourier_peak = pd.DataFrame(Fourier_peak)
Fourier_peak
```

Out[623]:

	0	1	2	3	4	5	6	7	8
0	414.632630	414.632630	550.141246	550.141246	1247.650747	1247.650747	2246.983205	2246.983205	5196.0
1	394.575536	394.575536	678.028662	678.028662	1426.896447	1426.896447	3786.424944	3786.424944	9207.0
2	370.658125	370.658125	417.644159	417.644159	1310.895588	1310.895588	2539.290590	2539.290590	5858.0
3	381.167977	381.167977	442.493722	442.493722	1163.474466	1163.474466	1907.700671	1907.700671	4679.0
4	304.371198	304.371198	330.747420	330.747420	696.395327	696.395327	1704.024370	1704.024370	4305.0
5	370.014012	370.014012	454.822274	454.822274	977.809543	977.809543	2135.643601	2135.643601	4919.0
6	284.291146	284.291146	305.515836	305.515836	946.452635	946.452635	1714.853858	1714.853858	4250.0
7	363.585658	363.585658	413.480972	413.480972	1196.655071	1196.655071	2006.275686	2006.275686	4611.0
8	343.794966	343.794966	420.011549	420.011549	931.096653	931.096653	1653.851913	1653.851913	4478.0
9	328.854074	328.854074	354.512595	354.512595	583.594804	583.594804	1923.265855	1923.265855	4604.0
10	445.633956	445.633956	529.129102	529.129102	1008.167980	1008.167980	2266.157065	2266.157065	5260.0
11	332.353340	332.353340	415.229815	415.229815	1292.108876	1292.108876	2100.611259	2100.611259	4459.0
12	313.607921	313.607921	398.634866	398.634866	986.172442	986.172442	1901.804196	1901.804196	4691.0
13	270.780981	270.780981	317.201065	317.201065	932.255206	932.255206	1958.299065	1958.299065	4553.0
14	235.446247	235.446247	244.822849	244.822849	677.405735	677.405735	1105.076344	1105.076344	3058.0
15	355.744855	355.744855	468.406871	468.406871	1190.776951	1190.776951	1684.007265	1684.007265	4568.0
16	397.684543	397.684543	467.811082	467.811082	1158.763680	1158.763680	2050.736852	2050.736852	5289.0
17	377.908040	377.908040	407.316258	407.316258	976.834753	976.834753	2204.309803	2204.309803	4472.0
18	363.758668	363.758668	410.749647	410.749647	794.020663	794.020663	1587.466227	1587.466227	3746.0
19	246.688896	246.688896	251.055154	251.055154	751.193193	751.193193	2025.387750	2025.387750	4552.0
20	295.033056	295.033056	343.572509	343.572509	843.506530	843.506530	1873.440012	1873.440012	4772.0
21	245.306578	245.306578	276.555562	276.555562	567.374079	567.374079	940.405334	940.405334	2740.0
192	315.281811	315.281811	399.174497	399.174497	965.671073	965.671073	1862.712959	1862.712959	4766.0
193	516.361251	516.361251	560.927072	560.927072	1084.637939	1084.637939	3083.155876	3083.155876	6514.0
194	325.873823	325.873823	601.523789	601.523789	1065.834523	1065.834523	1837.777804	1837.777804	4634.0
195	344.366601	344.366601	641.460751	641.460751	737.486010	737.486010	2088.170991	2088.170991	5379.0
196	378.573132	378.573132	455.493781	455.493781	1261.552250	1261.552250	1585.363544	1585.363544	4668.0
197	415.041874	415.041874	429.783442	429.783442	1112.894355	1112.894355	1790.483922	1790.483922	4615.0
198	438.569159	438.569159	618.767718	618.767718	997.617050	997.617050	2507.913239	2507.913239	5539.0
199	495.093510	495.093510	656.797688	656.797688	1045.203631	1045.203631	1968.039955	1968.039955	5503.0
200	599.334591	599.334591	615.808841	615.808841	1494.270378	1494.270378	3175.446068	3175.446068	8003.0
201	1184.574386	1184.574386	1198.591363	1198.591363	1214.351888	1214.351888	1739.705482	1739.705482	1948.0
202	637.596775	637.596775	704.465861	704.465861	1819.795370	1819.795370	3002.096563	3002.096563	8082.0
203	364.982444	364.982444	516.788602	516.788602	686.323803	686.323803	1626.025218	1626.025218	4017.0
204	439.116351	439.116351	448.121450	448.121450	1663.931381	1663.931381	2675.488746	2675.488746	6303.0
205	412.716956	412.716956	490.284788	490.284788	935.305066	935.305066	1869.781665	1869.781665	4678.0
206	562.145833	562.145833	674.587788	674.587788	1100.518248	1100.518248	3152.536910	3152.536910	6466.0
207	274.565962	274.565962	308.294480	308.294480	452.568490	452.568490	2311.645604	2311.645604	4500.0
208	457.681837	457.681837	638.356073	638.356073	861.960720	861.960720	2461.691337	2461.691337	5675.0
209	480.681819	480.681819	547.392694	547.392694	717.203143	717.203143	1694.667801	1694.667801	4691.0
210	562.084997	562.084997	622.768077	622.768077	1641.432758	1641.432758	2007.160991	2007.160991	6739.0
211	383.113955	383.113955	507.684669	507.684669	1008.513325	1008.513325	1391.121679	1391.121679	4580.0
212	547.439831	547.439831	547.517338	547.517338	1039.373045	1039.373045	2326.022600	2326.022600	5899.0
213	530.274194	530.274194	748.902904	748.902904	1371.348797	1371.348797	2998.542078	2998.542078	5901.0
214	570.492055	570.492055	611.460387	611.460387	1417.359756	1417.359756	2349.492491	2349.492491	6225.0
215	553.306127	553.306127	606.511197	606.511197	1351.326672	1351.326672	1859.463059	1859.463059	5339.0

216 rows × 9 columns

FFT frequencies:

```
In [621]: Fourier_frequency = pd.DataFrame(Fourier_frequency)
          Fourier_frequency
```

Out[621]:

	0	1	2	3	4
0	0.0	0.023810	0.047619	0.071429	0.142857
1	0.0	0.023810	0.047619	0.095238	0.119048
2	0.0	0.023810	0.047619	0.095238	0.119048
3	0.0	0.023810	0.047619	0.071429	0.142857
4	0.0	0.023810	0.047619	0.071429	0.119048
5	0.0	0.023810	0.047619	0.071429	0.142857
6	0.0	0.023810	0.047619	0.095238	0.142857
7	0.0	0.023810	0.047619	0.071429	0.119048
8	0.0	0.023810	0.047619	0.071429	0.142857
9	0.0	0.023810	0.047619	0.071429	0.119048
10	0.0	0.023810	0.047619	0.071429	0.119048
11	0.0	0.023810	0.047619	0.095238	0.119048
12	0.0	0.023810	0.047619	0.071429	0.119048
13	0.0	0.023810	0.047619	0.119048	0.142857
14	0.0	0.023810	0.047619	0.071429	0.119048
15	0.0	0.023810	0.047619	0.071429	0.119048
16	0.0	0.023810	0.047619	0.071429	0.119048
17	0.0	0.023810	0.047619	0.071429	0.119048
18	0.0	0.023810	0.047619	0.071429	0.119048
19	0.0	0.023810	0.047619	0.119048	0.142857
20	0.0	0.023810	0.047619	0.071429	0.142857
21	0.0	0.023810	0.047619	0.071429	0.119048
22	0.0	0.023810	0.047619	0.071429	0.119048
...					
195	0.0	0.023810	0.047619	0.071429	0.119048
196	0.0	0.023810	0.047619	0.071429	0.119048
197	0.0	0.023810	0.047619	0.071429	0.119048
198	0.0	0.023810	0.047619	0.071429	0.119048
199	0.0	0.023810	0.047619	0.071429	0.095238
200	0.0	0.023810	0.047619	0.071429	0.119048
201	0.0	0.047619	0.071429	0.119048	0.142857
202	0.0	0.023810	0.047619	0.071429	0.119048
203	0.0	0.023810	0.047619	0.071429	0.119048
204	0.0	0.023810	0.047619	0.095238	0.119048
205	0.0	0.023810	0.047619	0.071429	0.119048
206	0.0	0.023810	0.047619	0.071429	0.095238
207	0.0	0.023810	0.071429	0.119048	0.142857
208	0.0	0.023810	0.047619	0.071429	0.119048
209	0.0	0.023810	0.047619	0.071429	0.142857
210	0.0	0.023810	0.047619	0.071429	0.142857
211	0.0	0.023810	0.047619	0.071429	0.142857
212	0.0	0.023810	0.047619	0.071429	0.119048
213	0.0	0.023810	0.047619	0.071429	0.119048
214	0.0	0.023810	0.047619	0.071429	0.119048
215	0.0	0.023810	0.047619	0.071429	0.119048

216 rows × 5 columns

Sample output: The following shows the stem plot for the Fast Fourier Transform for the record no. 10 of the concatenated dataset:

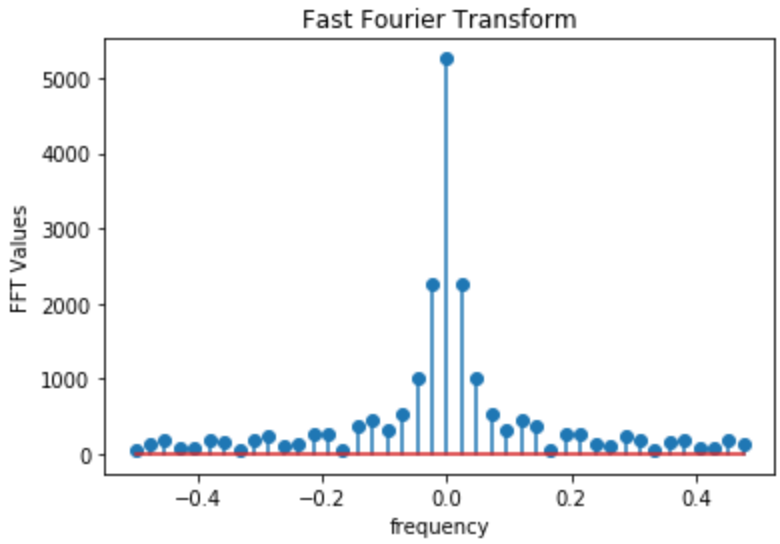


Fig 4. FFT values v/s frequency of record no. 10 of the concatenated data

Intuition:

The data provided has discrete CGM glucose level values. The time-domain data given is converted to frequency-domain by applying the Discrete Fourier Transform. A Fast Fourier Transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Thus, when the FFT was fit to the CGM Series data, it provided several peaks denoting the highest glucose levels. Out of all, the top 8 unique peaks were chosen as features for the feature matrix which could, in turn, be fed into the PCA for analysis.

4.4 **Inter- Quartile Range (IQR):**

The following are the result matrix for IQR:

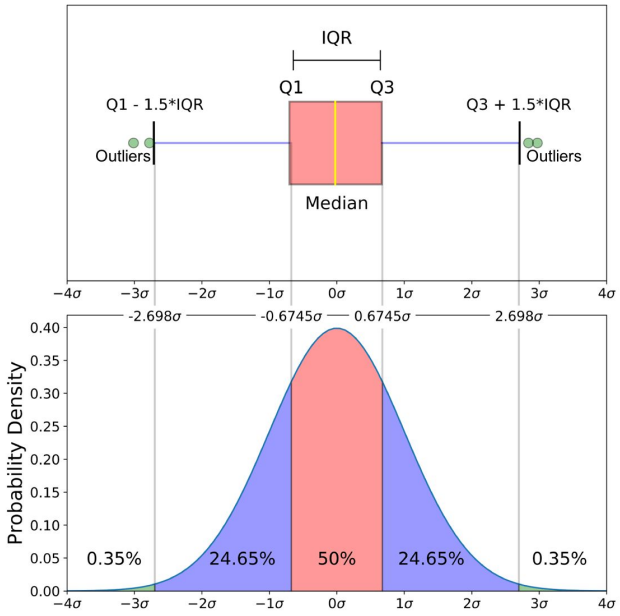
In [473]:
```
x = pd.DataFrame(x)
x
```

Out[473]:

	0
0	172.0
1	335.0
2	237.5
3	158.0
4	156.0
5	168.0
6	166.5
7	193.5
8	125.0
9	156.5
10	219.5
11	195.5
12	189.0
13	180.5
14	105.0
15	124.5
16	213.5
17	195.0
18	156.0
19	173.0
194	127.5
195	135.5
196	160.5
197	161.5
198	218.5
199	188.5
200	296.5
201	0.0
202	290.5
203	154.0
204	250.5
205	183.5
206	276.0
207	177.5
208	222.0
209	176.0
210	130.0
211	86.5
212	224.0
213	245.0
214	230.0
215	175.0

216 rows × 1 columns

Interquartile Range (IQR) is the measure of the mid-spread or middle 50% of the curve. It is a measure of variability while dividing the data set into quartiles. It defines how spread out the middle 50% of the data is. The following figure explains the IQR:



Intuition: IQR is a measure of variability which is based on dividing a data set into quartiles. Quartiles divide a rank-ordered data set into four equal parts. This helps to analyze how the glucose level values change across the spread of the time series.
According to the result of IQR, when the glucose level is high, it will give the time taken for each rise and fall of the glucose level.

5. Creating the Final Feature Matrix:

The feature matrix is created by appending the features extracted from the data. The following is a 216 x 103 feature matrix:

In [605]:
```
feature_matrix = pd.DataFrame(feature_matrix)
feature_matrix
```

Out[605]:

	0	1	2	3	4	5	6	7	8	9	...	93	9
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1247.650747	1247.6507
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1426.896447	1426.8964
2	249.333333	256.000000	263.666667	269.000000	273.333333	277.000000	278.666667	279.000000	280.000000	278.333333	...	1310.895588	1310.8955
3	240.000000	247.000000	251.000000	255.000000	260.333333	264.666667	267.666667	269.333333	271.666667	271.000000	...	1163.474466	1163.4744
4	199.000000	201.666667	203.000000	204.666667	208.333333	212.000000	214.000000	213.333333	212.333333	210.000000	...	696.395327	696.3953
5	192.666667	194.333333	192.000000	190.333333	191.000000	193.000000	195.000000	197.000000	198.000000	195.333333	...	977.809543	977.8095
6	171.333333	173.000000	174.666667	175.666667	176.666667	180.666667	185.000000	188.333333	189.666667	189.000000	...	946.452635	946.4526
7	201.666667	204.000000	205.000000	205.000000	203.000000	205.666667	208.333333	209.000000	208.333333	205.333333	...	1196.655071	1196.6550
8	193.000000	194.333333	195.000000	195.000000	193.666667	195.333333	195.333333	193.666667	191.666667	190.000000	...	931.096653	931.0966
9	183.000000	183.000000	183.333333	183.333333	183.000000	183.333333	183.000000	182.000000	182.333333	185.333333	...	583.594804	583.5948
10	180.000000	179.333333	180.000000	180.666667	182.000000	183.000000	184.333333	185.666667	187.333333	192.000000	...	1008.167980	1008.1679
11	193.333333	194.333333	196.666667	199.000000	201.666667	203.333333	204.666667	204.333333	204.000000	204.666667	...	1292.108876	1292.1088
12	211.000000	213.666667	216.333333	218.000000	220.000000	221.000000	221.333333	219.666667	218.000000	214.000000	...	986.172442	986.1724
13	190.333333	194.666667	200.666667	207.000000	210.666667	211.666667	212.666667	211.000000	210.666667	206.333333	...	932.255206	932.2552
14	155.000000	158.333333	162.333333	166.666667	169.666667	171.000000	172.000000	171.333333	171.666667	168.666667	...	677.405735	677.4057
15	161.666667	165.000000	168.666667	174.000000	176.666667	176.666667	175.333333	173.000000	169.666667	163.333333	...	1190.776951	1190.7769
16	181.333333	183.666667	183.666667	183.666667	183.666667	182.666667	180.000000	177.666667	174.000000	168.666667	...	1158.763680	1158.7636
17	167.666667	168.333333	168.000000	166.666667	165.333333	163.666667	163.000000	164.000000	165.000000	165.000000	...	976.834753	976.8347
18	158.666667	158.666667	157.333333	153.666667	149.666667	146.000000	146.333333	148.333333	152.333333	156.333333	...	794.020663	794.0206
19	132.000000	134.000000	135.333333	134.666667	134.000000	135.000000	138.666667	142.666667	148.000000	151.000000	...	751.193193	751.1931
20	147.666667	153.666667	159.000000	164.333333	168.666667	175.000000	180.666667	185.000000	189.000000	186.666667	...	843.506530	843.5065
21	118.666667	123.666667	129.333333	136.666667	143.666667	153.000000	159.333333	163.333333	167.000000	163.666667	...	567.374079	567.3740
22	123.333333	128.666667	134.333333	143.000000	151.000000	158.000000	162.333333	164.000000	165.000000	160.666667	...	1213.653267	1213.6532
192	206.333333	211.000000	215.666667	221.000000	222.333333	218.666667	217.333333	218.666667	220.333333	223.333333	...	965.671073	965.6710
193	239.000000	245.333333	251.666667	258.000000	259.666667	255.666667	253.666667	259.333333	261.000000	262.666667	...	1084.637939	1084.6379
194	215.666667	220.000000	225.666667	231.333333	232.666667	229.333333	227.000000	227.666667	223.333333	223.666667	...	1065.834523	1065.8345
195	200.333333	201.000000	203.666667	208.000000	211.333333	211.666667	212.666667	216.666667	216.000000	220.000000	...	737.486010	737.4860
196	176.333333	177.333333	179.666667	184.333333	186.333333	186.666667	186.333333	182.666667	176.333333	175.333333	...	1261.552250	1261.5522
197	176.666667	177.666667	178.333333	180.333333	180.333333	181.333333	182.333333	180.333333	180.000000	180.666667	...	1112.894355	1112.8943
198	223.666667	227.000000	228.000000	228.333333	224.000000	220.666667	216.666667	209.000000	203.000000	197.333333	...	997.617050	997.6170
199	222.000000	226.333333	228.000000	228.666667	219.333333	215.000000	211.333333	203.666667	202.000000	200.333333	...	1045.203631	1045.2036
200	256.666667	260.000000	263.333333	262.666667	254.000000	249.333333	245.666667	239.333333	237.666667	235.333333	...	1494.270378	1494.2703
201	301.333333	302.000000	176.333333	175.666667	168.333333	164.666667	163.000000	158.666667	158.666667	158.333333	...	1214.351888	1214.3518
202	355.333333	350.000000	221.000000	218.000000	215.333333	214.666667	212.000000	207.000000	203.666667	199.333333	...	1819.795370	1819.7953
203	281.333333	276.666667	146.666667	144.333333	142.333333	142.000000	138.333333	133.666667	131.666667	130.666667	...	686.323803	686.3238
204	240.333333	233.333333	242.000000	246.666667	249.000000	251.666667	243.666667	236.666667	227.333333	223.333333	...	1663.931381	1663.9313
205	187.333333	186.000000	198.000000	202.333333	203.333333	202.333333	196.000000	193.333333	187.333333	186.000000	...	935.305066	935.3050
206	256.333333	256.000000	268.666667	277.333333	276.666667	275.333333	263.666667	260.333333	253.000000	249.333333	...	1100.518248	1100.5182
207	201.000000	204.666667	208.333333	213.666667	212.000000	212.666667	158.333333	218.666667	233.333333	239.000000	...	452.568490	452.5684
208	207.000000	209.000000	212.333333	221.000000	224.000000	230.333333	174.000000	230.333333	244.000000	249.666667	...	861.960720	861.9607
209	164.000000	163.666667	166.333333	170.333333	119.666667	186.666667	138.666667	196.000000	209.000000	215.333333	...	717.203143	717.2031
210	191.666667	193.666667	196.666667	200.333333	149.333333	212.333333	212.000000	208.666667	208.000000	207.666667	...	1641.432758	1641.4327
211	167.333333	168.666667	170.000000	171.666667	118.000000	178.666667	181.666667	180.666667	180.333333	179.000000	...	1008.513325	1008.5133
212	178.333333	182.666667	185.666667	190.333333	193.000000	194.333333	196.000000	196.666667	198.000000	196.333333	...	1039.373045	1039.3730
213	224.333333	225.666667	228.666667	232.000000	230.333333	228.333333	226.666667	222.666667	218.666667	211.333333	...	1371.348797	1371.3487
214	266.666667	268.666667	273.666667	277.000000	274.666667	271.666667	264.666667	259.666667	255.333333	247.000000	...	1417.359756	1417.3597
215	279.666667	281.333333	283.333333	284.666667	277.333333	270.333333	258.666667	250.666667	244.000000	234.000000	...	1351.326672	1351.3266

216 rows × 103 columns

6. Creating the Covariance Matrix:

The covariance matrix is calculated using the formula:

$$Cov_{ij} = \frac{\sum_{k=1}^{N} (Z_{ik} - \mu_i)(Z_{jk} - \mu_j)}{N-1}$$

where,
Z - value of a cell
i, j - are layers of a stack
μ - is the mean of a layer
N - is the number of cells
k - denotes a particular cell

In matrix form, the above statistical equation can be written as:

$$Cov(X,Y) = E\left[(X - \mu_X)(Y - \mu_Y)\right]$$

The result of covariance matrix of size 103x103:

```
In [477]: cov_mat = pd.DataFrame(cov_mat)
          cov_mat
```

Out[477]:

	0	1	2	3	4	5	6	7	8	9	...	93	94	95	
0	1.004651	1.001098	0.929461	0.919710	0.883625	0.877136	0.861398	0.840126	0.836780	0.846519	...	0.348478	0.348478	0.363919	0.363
1	1.001098	1.004651	0.939952	0.932613	0.899376	0.894204	0.879568	0.858892	0.857117	0.866633	...	0.350180	0.350180	0.368137	0.368
2	0.929461	0.939952	1.004651	1.002560	0.973389	0.972521	0.958985	0.940153	0.899777	0.906668	...	0.346482	0.346482	0.352505	0.352
3	0.919710	0.932613	1.002560	1.004651	0.979017	0.980472	0.966593	0.948089	0.908871	0.915966	...	0.343368	0.343368	0.355058	0.355
4	0.883625	0.899376	0.973389	0.979017	1.004651	0.991728	0.984871	0.962713	0.924835	0.917713	...	0.322102	0.322102	0.353462	0.353
5	0.877136	0.894204	0.972521	0.980472	0.991728	1.004651	0.994423	0.980998	0.945156	0.937460	...	0.325464	0.325464	0.346335	0.346
6	0.861398	0.879568	0.958985	0.966593	0.984871	0.994423	1.004651	0.977229	0.939923	0.929777	...	0.330655	0.330655	0.340672	0.340
7	0.840126	0.858892	0.940153	0.948089	0.962713	0.980998	0.977229	1.004651	0.977505	0.944067	...	0.310415	0.310415	0.336970	0.336
8	0.836780	0.857117	0.899777	0.908871	0.924835	0.945156	0.939923	0.977505	1.004651	0.981077	...	0.288772	0.288772	0.376754	0.376
9	0.846519	0.866633	0.906668	0.915966	0.917713	0.937460	0.929777	0.944067	0.981077	1.004651	...	0.267598	0.267598	0.390262	0.390
10	0.829713	0.850113	0.887637	0.896285	0.892788	0.913762	0.906022	0.924766	0.968438	1.000120	...	0.250767	0.250767	0.394162	0.394
11	0.804970	0.823880	0.848169	0.854638	0.848298	0.869645	0.863492	0.882893	0.933259	0.970370	...	0.226593	0.226593	0.398684	0.398
12	0.797151	0.815193	0.834289	0.839173	0.830793	0.848797	0.843926	0.869328	0.927411	0.966532	...	0.230241	0.230241	0.408333	0.408
13	0.668793	0.683689	0.696044	0.701059	0.690846	0.711157	0.712151	0.747461	0.813917	0.852968	...	0.235278	0.235278	0.445080	0.445
14	0.626009	0.638320	0.644406	0.646357	0.634957	0.652862	0.658754	0.693953	0.764029	0.804488	...	0.224646	0.224646	0.440367	0.440
15	0.573995	0.581903	0.578831	0.576807	0.561373	0.575018	0.581392	0.614467	0.687128	0.730891	...	0.210675	0.210675	0.414608	0.414
16	0.516952	0.520302	0.506344	0.500687	0.480306	0.494631	0.498648	0.530223	0.607218	0.654669	...	0.196311	0.196311	0.392421	0.392
17	0.489060	0.489080	0.467658	0.460263	0.439559	0.452158	0.455610	0.481175	0.560976	0.611266	...	0.196053	0.196053	0.372696	0.372
18	0.441362	0.438931	0.414701	0.406874	0.389733	0.399385	0.404508	0.425411	0.503575	0.547884	...	0.218226	0.218226	0.345335	0.345
19	0.393759	0.388651	0.365672	0.356402	0.342013	0.349115	0.356862	0.375407	0.447254	0.489058	...	0.223548	0.223548	0.317608	0.317
20	0.364198	0.359941	0.338865	0.329583	0.319841	0.324318	0.337398	0.351558	0.418819	0.459875	...	0.211825	0.211825	0.309095	0.309
21	0.252292	0.247819	0.228291	0.215819	0.207701	0.210617	0.228223	0.246319	0.304436	0.337033	...	0.268860	0.268860	0.298737	0.298
22	0.208089	0.203620	0.190010	0.176500	0.167426	0.171988	0.190156	0.209580	0.254058	0.280385	...	0.279855	0.279855	0.273070	0.273

78	0.069330	0.065215	0.071095	0.067388	0.069301	0.064556	0.064790	0.062281	0.061280	0.066576	...	-0.017561	-0.017561	0.001423	0.001
79	0.069330	0.065215	0.071095	0.067388	0.069301	0.064556	0.064790	0.062281	0.061280	0.066576	...	-0.017561	-0.017561	0.001423	0.001
80	0.069330	0.065215	0.071095	0.067388	0.069301	0.064556	0.064790	0.062281	0.061280	0.066576	...	-0.017561	-0.017561	0.001423	0.001
81	0.069330	0.065215	0.071095	0.067388	0.069301	0.064556	0.064790	0.062281	0.061280	0.066576	...	-0.017561	-0.017561	0.001423	0.001
82	0.069330	0.065215	0.071095	0.067388	0.069301	0.064556	0.064790	0.062281	0.061280	0.066576	...	-0.017561	-0.017561	0.001423	0.001
83	0.069330	0.065215	0.071095	0.067388	0.069301	0.064556	0.064790	0.062281	0.061280	0.066576	...	-0.017561	-0.017561	0.001423	0.001
84	-0.238066	-0.237405	-0.241940	-0.240124	-0.213623	-0.215766	-0.215887	-0.212519	-0.219234	-0.239262	...	-0.414041	-0.414041	-0.492612	-0.492
85	-0.229069	-0.228572	-0.240427	-0.238759	-0.212746	-0.215248	-0.215494	-0.212475	-0.219324	-0.239553	...	-0.411352	-0.411352	-0.493040	-0.493
86	-0.229947	-0.228904	-0.242344	-0.240711	-0.217101	-0.219903	-0.223047	-0.216741	-0.218194	-0.234686	...	-0.427687	-0.427687	-0.483269	-0.483
87	-0.207270	-0.207009	-0.215157	-0.213926	-0.187103	-0.190789	-0.191295	-0.181051	-0.185048	-0.207185	...	-0.392902	-0.392902	-0.443965	-0.443
88	-0.198835	-0.200841	-0.206206	-0.206430	-0.188048	-0.182552	-0.182002	-0.168786	-0.173072	-0.196868	...	-0.349135	-0.349135	-0.473252	-0.473
89	0.331929	0.326176	0.236097	0.228055	0.184924	0.185666	0.172171	0.162495	0.174043	0.186136	...	0.685639	0.685639	0.588115	0.588
90	0.331929	0.326176	0.236097	0.228055	0.184924	0.185666	0.172171	0.162495	0.174043	0.186136	...	0.685639	0.685639	0.588115	0.588
91	0.331907	0.325340	0.240704	0.232509	0.191659	0.188103	0.174941	0.162966	0.175517	0.191311	...	0.692735	0.692735	0.612099	0.612
92	0.331907	0.325340	0.240704	0.232509	0.191659	0.188103	0.174941	0.162966	0.175517	0.191311	...	0.692735	0.692735	0.612099	0.612
93	0.348478	0.350180	0.346482	0.343368	0.322102	0.325464	0.330655	0.310415	0.288772	0.267598	...	1.004651	1.004651	0.602470	0.602
94	0.348478	0.350180	0.346482	0.343368	0.322102	0.325464	0.330655	0.310415	0.288772	0.267598	...	1.004651	1.004651	0.602470	0.602
95	0.363919	0.368137	0.352505	0.355058	0.353462	0.346335	0.340672	0.336970	0.376754	0.390262	...	0.602470	0.602470	1.004651	1.004
96	0.363919	0.368137	0.352505	0.355058	0.353462	0.346335	0.340672	0.336970	0.376754	0.390262	...	0.602470	0.602470	1.004651	1.004
97	0.394432	0.397813	0.409314	0.407796	0.389847	0.393462	0.391000	0.383523	0.404759	0.415307	...	0.721713	0.721713	0.862556	0.862
98	0.048452	0.063676	0.145190	0.164437	0.204539	0.201492	0.194991	0.207109	0.256847	0.282673	...	-0.206773	-0.206773	0.423769	0.423
99	0.017371	0.003789	-0.066405	-0.082957	-0.117326	-0.117226	-0.112180	-0.128409	-0.190503	-0.224161	...	0.150503	0.150503	-0.512404	-0.512
100	-0.212323	-0.204030	-0.164404	-0.154455	-0.130779	-0.125939	-0.127055	-0.103606	-0.032555	0.008197	...	-0.186774	-0.186774	0.416082	0.416
101	0.569533	0.572799	0.590056	0.590607	0.577025	0.571486	0.566238	0.541347	0.510453	0.489608	...	0.622213	0.622213	0.469536	0.469
102	0.355246	0.362269	0.376154	0.379118	0.384372	0.373251	0.364626	0.359636	0.389168	0.398174	...	0.610458	0.610458	0.895336	0.895

103 rows × 103 columns

7. EigenValues and EigenVectors:

Eigenvectors =

```
In [481]: eig_vecs = pd.DataFrame(eig_vecs)
          eig_vecs
```

Out[481]:

	0	1	2	3	4	
0	(0.025817791263460774+0j)	(-0.14128280175966232+0j)	(-0.08769280705509551+0j)	(-0.15281527500188458+0j)	(0.0710804327787093+0j)	(-0.02
1	(0.026618112325311756+0j)	(-0.14450283208134904+0j)	(-0.08563558103998113+0j)	(-0.15315978273058892+0j)	(0.07525115588354238+0j)	(-0.02
2	(0.03969821299568313+0j)	(-0.15770206707285422+0j)	(-0.06379568488466039+0j)	(-0.14115951952502012+0j)	(0.09532957940698011+0j)	(-0
3	(0.039990521166021135+0j)	(-0.1582517897634817+0j)	(-0.061593766710752444+0j)	(-0.14633535632780864+0j)	(0.09712999214417398+0j)	(-0.0
4	(0.04005360802221187+0j)	(-0.15452957006600304+0j)	(-0.06040179801081883+0j)	(-0.14782238205025042+0j)	(0.10784030035436366+0j)	(-0.0
5	(0.039925946499639264+0j)	(-0.15763423975138324+0j)	(-0.05963361526095405+0j)	(-0.14240908173648018+0j)	(0.11167375885093209+0j)	(-0
6	(0.039243733366271585+0j)	(-0.1573501769571673+0j)	(-0.06042341080580094+0j)	(-0.13014321933134154+0j)	(0.1132256752484619+0j)	(-0.0

96	(0.006470973124405647+0j)	(-0.116955689609808+0j)	(-0.06334158978806374+0j)	(-0.04515131634856039+0j)	(-0.18613780055745607+0j)	(-0.1
97	(0.008548192687918777+0j)	(-0.12907752566577307+0j)	(-0.0609305799043543 26+0j)	(-0.020602278872020027+0j)	(-0.18207142124088402+0j)	(-0.0
98	(0.03907955669169266+0j)	(-0.05304498825479636+0j)	(0.027153376174028705+0j)	(-0.03828768976390096+0j)	(0.05309499399853914+0j)	(-0.1
99	(-0.02068579397520047+0j)	(0.05912312522788034+0j)	(-0.016911071960958122+0j)	(-0.009983125564881587+0j)	(-0.01854218710057837+0j)	(0.1
100	(-0.011236644614794292+0j)	(-0.03378855138335319+0j)	(0.007973345078922545+0j)	(0.09830830066221523+0j)	(-0.005589873604691525 6+0j)	(-0.0
101	(0.03852969292706584+0j)	(-0.0964499293879072+0j)	(-0.0345878726468258+0j)	(-0.15361153260351773+0j)	(-0.09643495467620426+0j)	(-0.1
102	(0.00829425929232875+0j)	(-0.117695063923409+0j)	(-0.04978334958996154+0j)	(-0.04757294193816246+0j)	(-0.16603121377591168+0j)	(-0

103 rows × 103 columns

Eigenvalues =

In [480]:
```
eig_vals = pd.DataFrame(eig_vals)
eig_vals
```

Out[480]:

	0
0	(22.834591814975838+0j)
1	(21.52519067467568+0j)
2	(20.580783600527546+0j)
3	(8.050228584712595+0j)
4	(7.07657191519219+0j)
5	(4.82048069194149+0j)
6	(4.297076376037873+0j)
7	(2.385762602944019+0j)
8	(1.9935701560535135+0j)
9	(1.831152806081491+0j)
10	(1.4749738500688074+0j)
11	(1.1106951925601587+0j)
12	(0.9779542563248734+0j)
13	(0.6873215472831001+0j)
14	(0.5098306921792654+0j)
15	(0.4160979644793085+0j)
16	(0.3444590682884659+0j)
17	(0.2582609209759517 7+0j)
18	(0.2516026934982669+0j)
19	(0.2293820718750008 7+0j)
20	(0.2109151119717789 8+0j)
21	(0.1834194272581062+0j)

86	(-2.746487392265033e-17+0j)
87	(3.718503948539855e-18+2.272384982788868e-18j)
88	(3.718503948539855e-18-2.272384982788868e-18j)
89	(3.20278088803099e-18+0j)
90	(6.188677334624022e-19+2.9710388433192447e-18j)
91	(6.188677334624022e-19-2.9710388433192447e-18j)
92	(-2.116393117638896e-18+0j)
93	(-9.558339227750575e-19+1.035828276088647e-18j)
94	(-9.558339227750575e-19-1.035828276088647e-18j)
95	(3.0346391484311106e-19+3.8570313301871362e-19j)
96	(3.0346391484311106e-19-3.8570313301871362e-19j)
97	(-5.0239132003672634e-21+0j)
98	(2.813688120769557e-21+0j)
99	(2.8860176002246753e-24+0j)
100	(-2.2953357850004115e-32+0j)
101	(1.253594262687313e-33+6.442637658900852e-33j)
102	(1.253594262687313e-33-6.442637658900852e-33j)

103 rows × 1 columns

The eigenvectors corresponding to the top 5 eigenvalues were chosen as the principal components.

8. Principal Component Analysis (PCA):

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. To reduce the high dimensional data of 97 features extracted, we are using PCA to reduce it to a 5-dimensional space while we preserve the maximum variance in the data for reproducibility[5].

Thus, on decomposing, we get the following components:

PCA matrix =

In [482]:
```
X_std_pca = pd.DataFrame(X_std_pca)
X_std_pca
```

Out[482]:

	0	1	2	3	4
0	3.620040	9.120381	15.808993	1.303439	-9.085779
1	3.721119	7.464145	15.183584	1.366427	-11.185912
2	-1.250408	-6.394359	-6.877467	0.043922	3.559042
3	-1.825643	-4.646900	-7.147727	0.404718	3.850193
4	1.369359	-2.716040	2.241818	2.879994	1.847342
5	2.045520	-3.014931	3.579907	2.433587	0.142342
6	2.018905	-2.222157	4.406103	1.420835	1.032226
7	2.792684	-3.678830	4.825612	2.933269	-0.060347
8	2.800946	-2.890909	5.663904	1.895845	0.415833
9	1.804369	-2.292451	3.875766	1.115598	1.044598
10	1.636018	-3.794586	2.630014	1.522902	-0.594494
11	1.417641	-2.834717	1.983643	3.135291	0.366939
12	2.485246	-3.689520	3.996954	2.507615	1.084811
13	2.731580	-3.067105	5.089980	2.313736	1.481204
14	1.432210	0.295162	4.841478	2.056003	1.262989
15	1.547442	-0.609844	4.305853	3.021351	-1.205796
16	0.935560	-1.063062	2.489853	2.926114	-1.369922
17	-1.065252	-0.417866	-1.142994	-0.069460	-0.328963
18	-1.004091	0.737808	-0.600425	-0.060020	0.076262
19	-0.647336	1.132054	0.646904	-1.216081	0.800071
20	2.062677	-1.945606	4.773152	0.239076	0.841164
21	0.732246	1.318268	4.076703	0.854425	1.173207
22	0.528005	0.330647	2.929608	0.884051	-0.970304
⋮					
193	0.620649	-7.494424	-2.772127	3.398571	0.928031
194	1.041010	-3.852754	0.438722	3.877240	1.293367
195	-0.170738	-3.400448	-1.845425	2.253100	1.520098
196	0.607125	-2.288875	1.167113	0.050719	-0.005294
197	0.656799	-2.517254	1.377765	-0.024083	-0.089788
198	1.939146	-4.531870	2.603412	2.408418	-0.605304
199	1.302971	-4.736141	1.163029	1.086656	-0.120635
200	-0.210703	-7.928432	-4.700363	2.101311	-0.098873
201	-6.206184	4.271786	-11.713272	5.854998	-4.165201
202	-6.000006	0.184777	-14.082028	6.789695	-2.753992
203	-5.791436	6.219323	-9.051455	4.936450	-0.810714
204	-1.785308	-4.694580	-6.386898	5.517121	0.224079
205	-0.206667	-2.066001	-0.632259	3.908656	-0.039470
206	1.503159	-7.369540	-1.040681	5.390692	-0.101580
207	0.829927	-3.398404	0.512949	4.379633	3.489186
208	1.244854	-6.118561	-0.472368	4.509675	1.198613
209	0.914786	-3.376673	1.392703	1.310926	0.675015
210	-0.084262	-5.083404	-1.820239	-2.133130	-0.589180
211	-0.541180	-2.247589	-0.449075	-3.298460	0.504275
212	-0.467748	-4.798486	-1.927747	-2.151595	-0.559058
213	0.048880	-4.744445	-2.482069	4.224873	-1.518381
214	1.827619	-8.028948	-0.497878	2.885551	0.683404
215	1.515209	-6.533360	-0.198557	3.208692	0.808169

216 rows × 5 columns

PCA top 5 features:

The following shows the plot for the top 5 principal components:

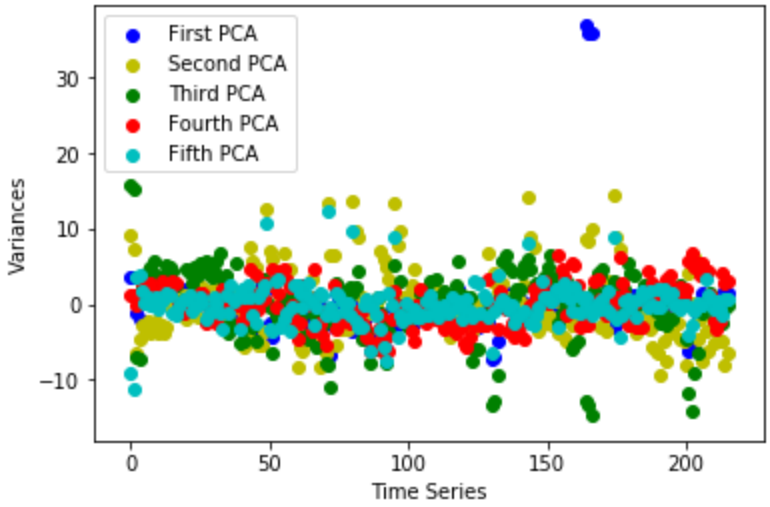


Fig 5. Scatter Plot of variances of top 5 Principle Components

PCA Explained Variance:

```
In [484]: pcaExpVariance = pd.DataFrame(pcaExpVariance)
          pcaExpVariance
```

Out[484]:

	0
0	22.834592
1	21.525191
2	20.580784
3	8.050229
4	7.076572

The following plot shows that the amount of information captured from the PCA is highest for the first vector. The actual learning rate decreases gradually.

After feeding the original feature matrix to the PCA model generated we get the transformed matrix. The following graph captures the variance for the top 5 principal components in a scree (bar) plot.

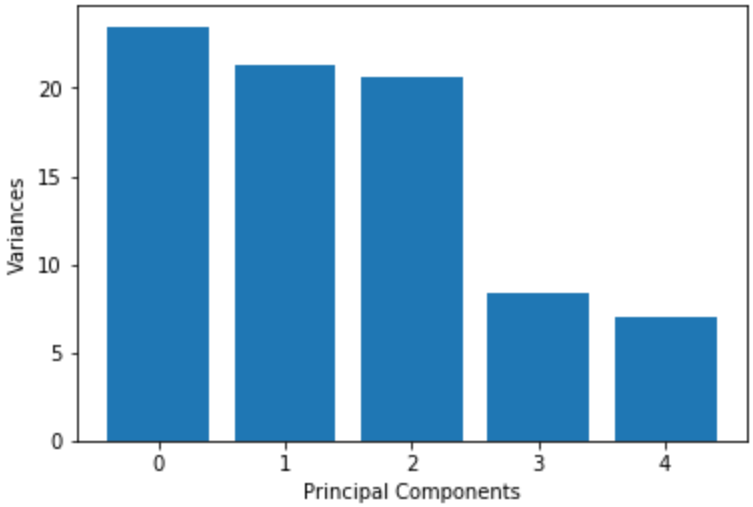


Fig 6. Scree(Bar) Plot of variances of top 5 Principle Components

9. Plots and Arguments:

PCA - Dimensionality Reduction Tool:
In PCA, the eigenvalues are ordered from largest to smallest so that it gives the components in order or significance to effectively reduce dimensionality.
If a dataset has n variables, then there will be n eigenvalues and eigenvectors created from the covariance matrix. It turns out that the eigenvector corresponding to the highest eigenvalue is the principal component of the dataset. Here, to reduce the dimensions, we choose the first 5 eigenvalues and ignore the rest. We do lose out some information in the process, but if the eigenvalues are small, we do not lose much.
Thus, PCA is quite efficient in Dimensionality Reduction.

The following are individual plots displaying the above-extracted eigenvectors of top 5 principal components with time:

9.1 **Principal Component 1:**

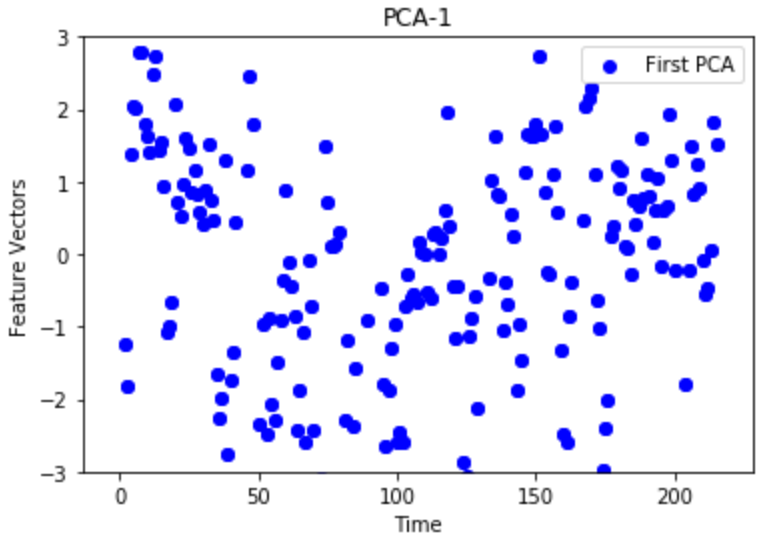


Fig 7. Features projected along the first principle component

Argument: The principal component is chosen as the first PC because it shows the maximum explained variance (22.834592).

9.2 **Principal Component 2:**

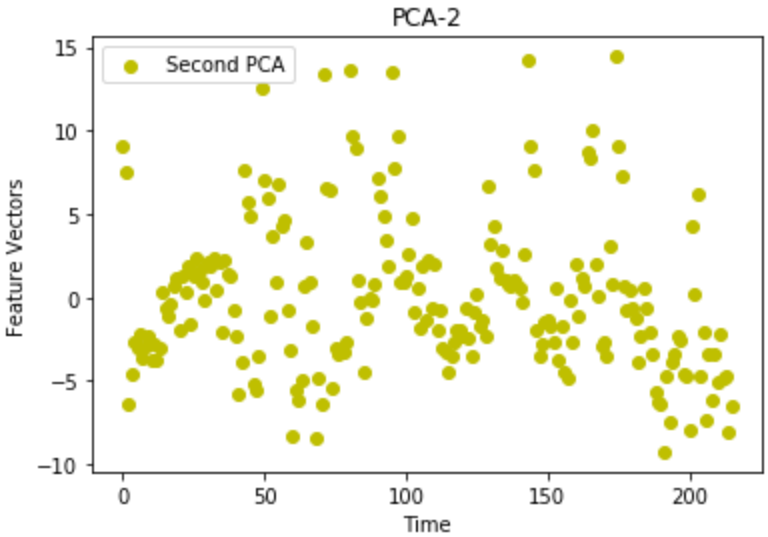


Fig 8. Features projected along the second principle component

Argument: The principal component is chosen as the second PC because it shows the second maximum explained variance (21.525191).

9.3 **Principal Component 3:**

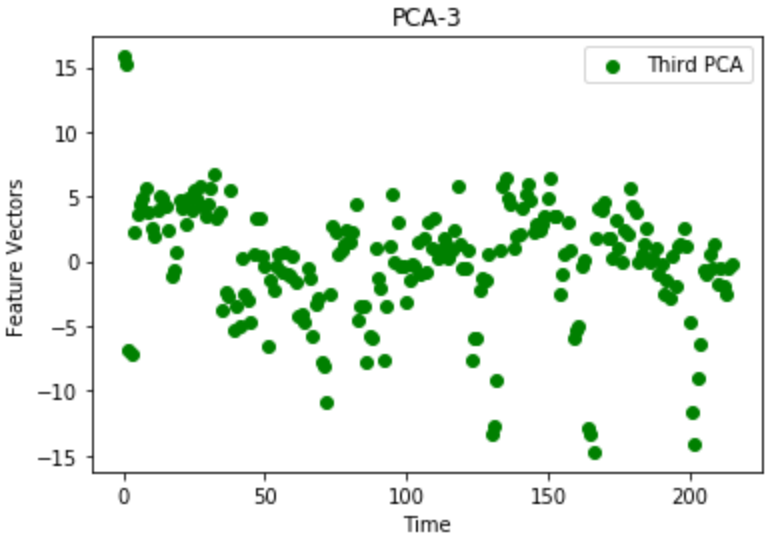


Fig 9. Features projected along the third principle component

Argument: The principal component is chosen as the second PC because it shows the third maximum explained variance (20.580784).

9.4 **Principal Component 4:**

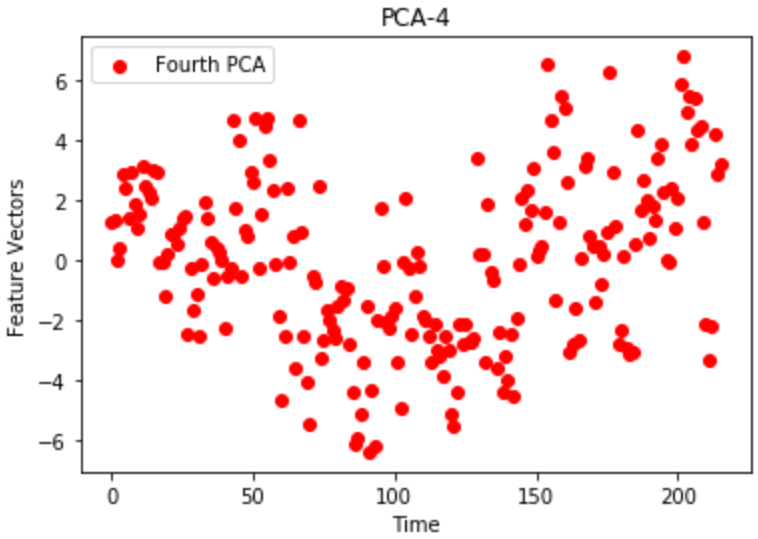


Fig 10. Features projected along the fourth principle component

Argument: The principal component is chosen as the second PC because it shows the fourth maximum explained variance (8.050229).

9.5 **Principal Component 5:**

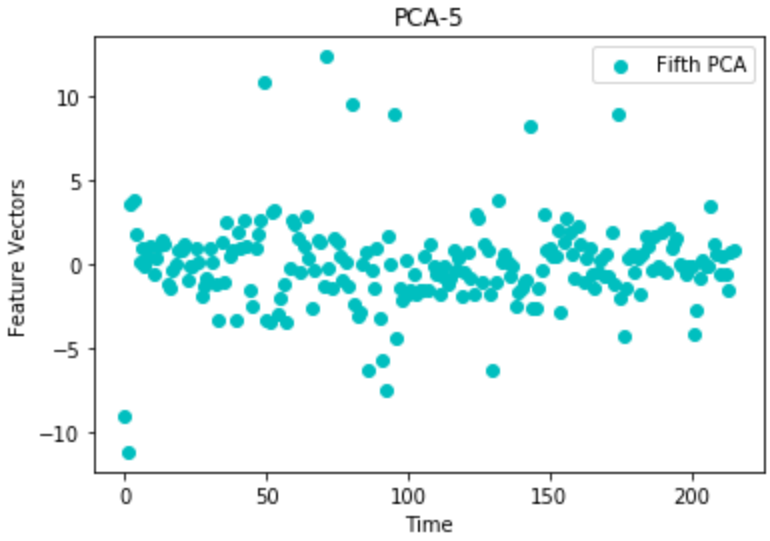


Fig 11. Features projected along the fifth principle component

Argument: The principal component is chosen as the second PC because it shows the fifth maximum explained variance (7.076572).

PCA Argument on no. of features selected:
The project picks up top 5 principal components denoting the most significant features. The ideal number of features that should be taken can be calculated by the explained variance ratio which is applied on the PCA matrix.
Picking top 5 features captures 77.5% of the data while ideally, to capture ~100% of the data, 30-35 features should have been selected. The following analysis validates the above explanation:

Explained Variance Ratio (5 components) = [22.1 42.9 62.8 70.6 77.4]

Explained Variance Ratio (30 components) = [22.1 42.9 62.8 70.6 77.4 82.1 86.3 88.6 90.5 92.3 93.7 94.8 95.7 96.4 96.9 97.3 97.6 97.8 98. 98.2 98.4 98.6 98.7 98.8 98.9 99. 99.1 99.2 99.3 99.4]

In the above array we see that the first feature explains roughly 22.1% of the variance within our data set while the first two explain 42.9 and so on. If we employ 30 features we capture 99.4% of the variance within the dataset, thus we gain very little by implementing an additional feature (works like a diminishing marginal return on total variance explained).

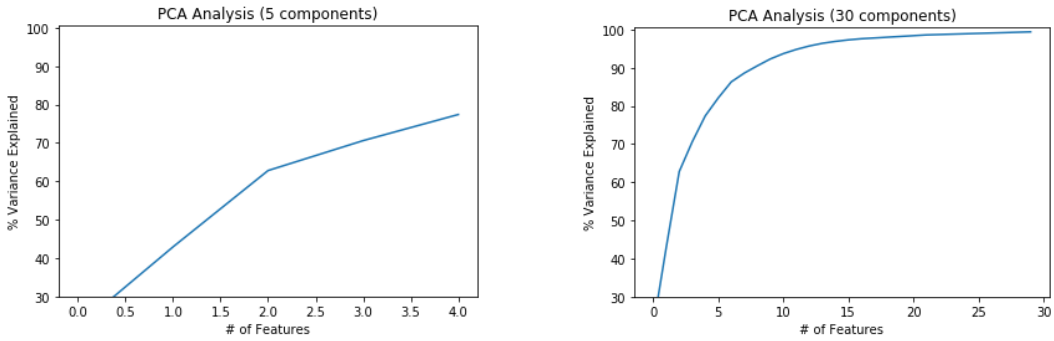


Fig 12. Amount of data captured 5 components were selected from PCA (left) and when 30 components were selected from PCA (right)

PCA reduces the 103 features of the feature matrix to top 5 components (which can capture 77.4% of the variance of the whole dataset). Taking top 30 features can capture 99.4% of the variance of the whole dataset, again reducing the 103 features to only 30 features that are important for analysis. Adding more features does not provide any additional benefit. Hence, PCA is a good tool for dimensionality reduction.

10. References:

[1]. https://www.programcreek.com/python/example/101350/pandas.rolling_std
[2]. https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft.html
[3].https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html
[4].https://www.geeksforgeeks.org/interquartile-range-and-quartile-deviation-using-numpy-and-scipy/
[5].https://chrisalbon.com/machine_learning/feature_engineering/feature_extraction_with_pca/