

## CSE 572 Data mining Project 1

Background literature review:

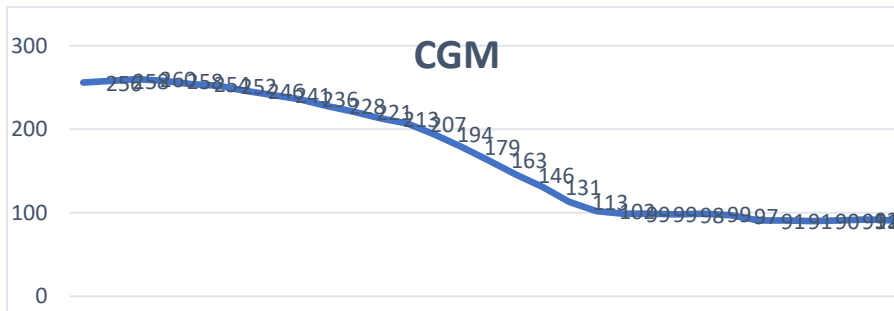
Basal and Bolus understanding:

Some people might take only basal, or "background" insulin. This is a long-acting insulin that boosts activity for around 24 hours at a time, but to a lower peak than rapid-, intermediate-, or regular-acting insulin. Basal provides a constant supply of insulin to bring down high resting blood glucose levels. Bolus insulin, on the other hand, has a much more powerful but shorter-lived effect on blood sugar, making it an ideal supplement for people with diabetes to take after meals and in moments of extremely high blood sugar. A basal-bolus insulin regimen involves a person with diabetes taking both basal and bolus insulin throughout the day. Basal Takes around 90 minutes to 4 hours to actually activate.

Data understanding and explanations:

As the Data starts 30 minutes before, The CGM level should ideally increase if a meal is taken, but as we know the diabetic patients takes either basal or bolus insulin infusion as well so ideally when the meal is taken the CGM level should increase and should be contained with a limit or go downwards depending upon the intake of insulin.

Patient 1 Case Lunch event 1:




Right side is Lunch event 1 of patient 1. CGM is in mg dl unit. Now starting from right in the graph the patient takes the meal & gradually the glucose level increases, but the insulin infusion is there which

contains the glucose level with in some range, doesn't let it to shoot up very high. Probably the insulin infusion should have been stronger (could be basal or bolus) to maintain the blood glucose level within a prescribed range. Below are some images showing glucose levels in adults.

Target Blood Sugar Levels for Diabetes	
Age 20+	
Fasting	less than 100
Before Meal	70-130
After Meal (1-2hrs)	less than 180
Before Exercise	if taking insulin, at least 100
Bedtime	100-140
Amounts shown above mg/dL	
A1c	less than or around 7.0%

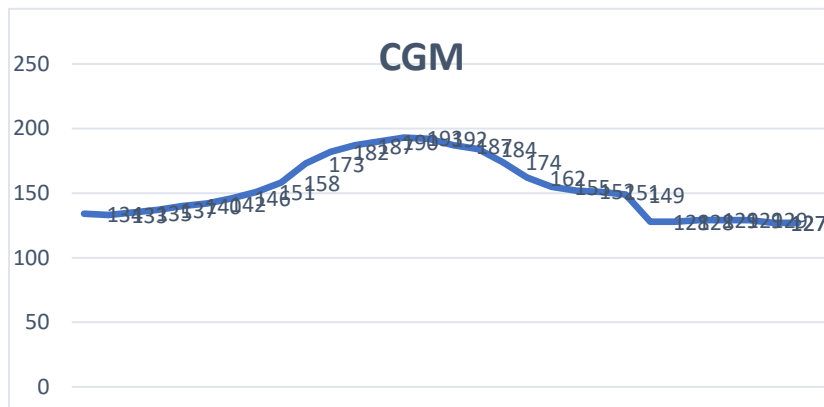
These are general medical guidelines. Please follow your doctor's instructions.



WebMD

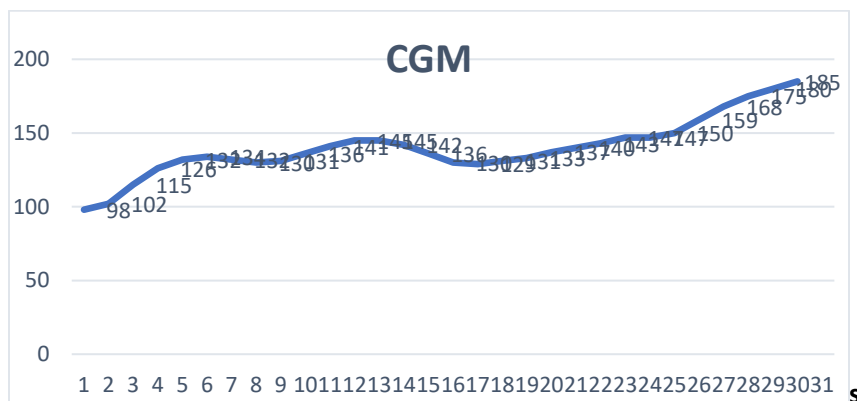
Condition	Fasting	Just Ate	3 Hours After Eating
Normal	80-100	170-200	120-140
Pre-diabetic	101-125	190-230	140-160
Diabetic	126 and above	220-300	200 and above

Patient 1 Case Lunch event 2:



Probably proper infusion methodology was applied and the glucose level was maintained with in range.

Patient 1 Case Lunch event 3:



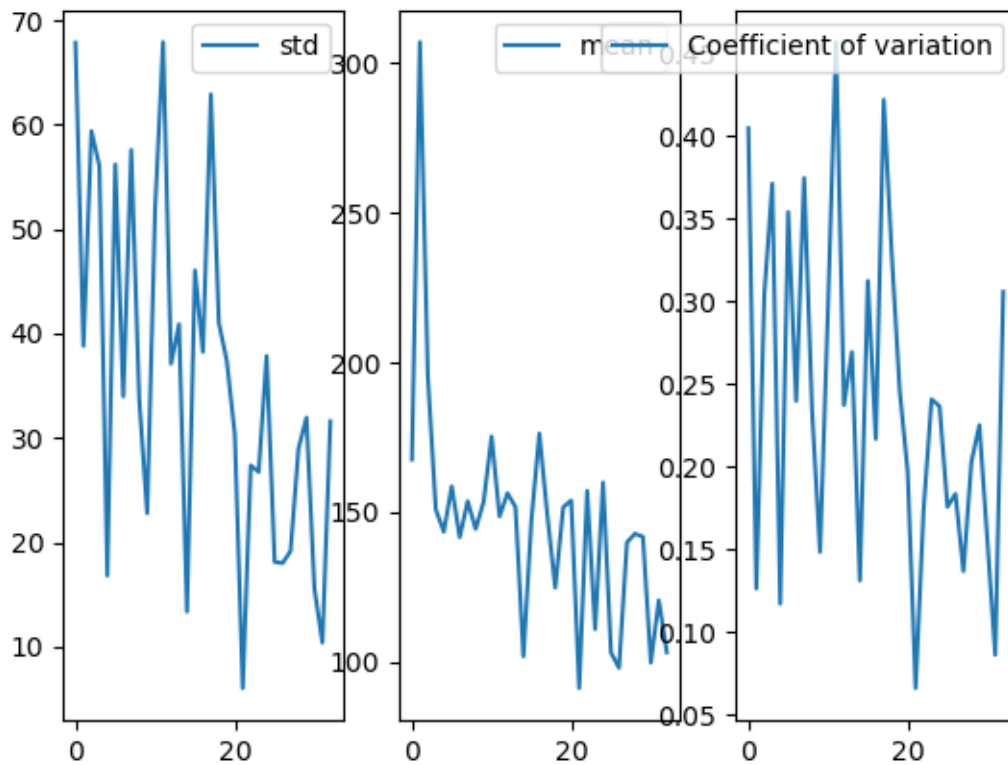
Either light Meal was taken or Basal/Bolus infusion was strong enough to draw glucose level in blood down.

### Tasks:

- Extract 4 different types of time series features from only the CGM data cell array and CGM timestamp cell array (10 points each) total 40

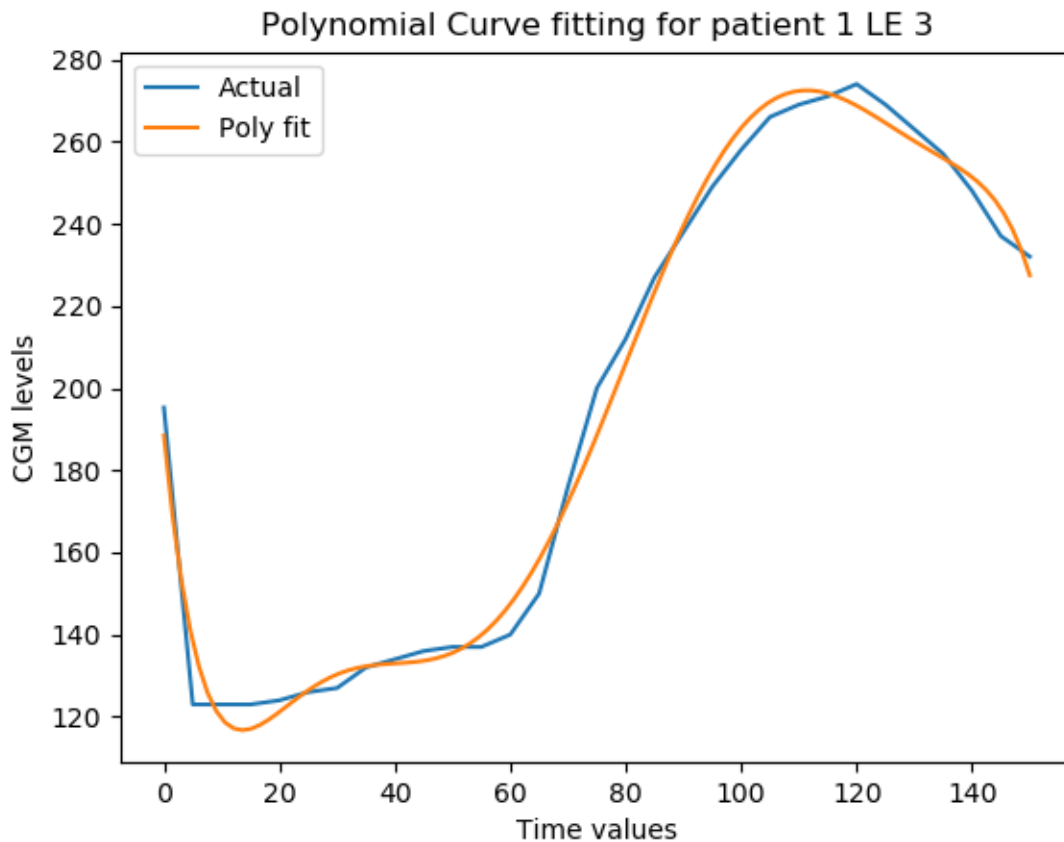
The 4 different type of features I am deciding based on literature review and intuition are:

- Coefficient of variation** (CV=standard deviation / mean): As a rule of thumb, a CV  $\geq 1$  indicates a relatively high variation, while a CV  $< 1$  can be considered low. This means that distributions with a coefficient of variation higher than 1 are considered to be high variance whereas those with a CV lower than 1 are considered to be low-variance. Reference:  
<https://www.cambridge.org/core/journals/journal-of-french-language-studies/article/e-in-normandy-the-sociolinguistics-phonology-and-phonetics-of-the-loi-de-position/1B6B9D55EE7865E322D6EEC62D8CD2E9>



Above is the CV values for patient 1 across all the meal events.

2. **Polynomial Curve fitting the CGM Data (Coefficients):** I would try to fit a polynomial to different Lunch events CGM data, would choose the polynomial of an order which represents the data with the least error, error can be modeled by calculating goodness of fit (Chi Method) can be calculated and range of factors can be modeled based on the error.

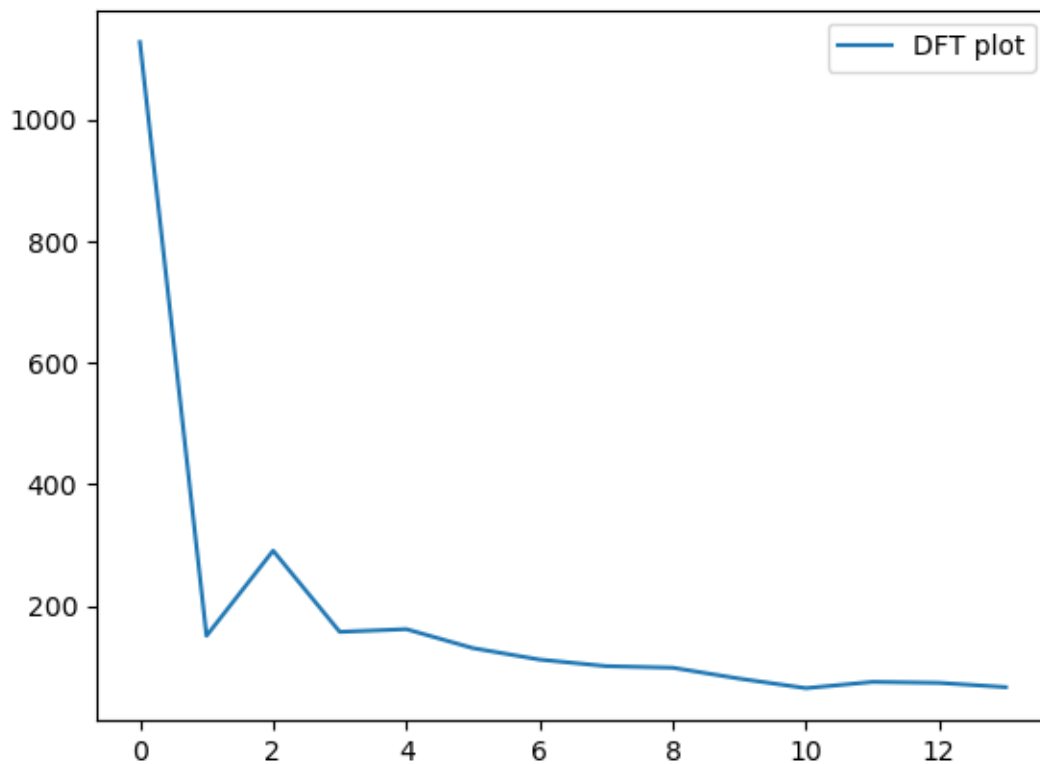


```
('Actual:', array([ nan, 123., 123., 123., 124., 126., 127., 132., 134., 136., 137.,
137., 140., 150., 176., 200., 212., 227., 238., 249., 258., 266.,
269., 271., 274., 269., 263., 257., 248., 237., 232.]))
```

```
('Coefficients:', array([ 8.80855938e+01, 1.45973240e+00, -1.05037932e-01, 2.50516456e-03,
-1.97365124e-05, 5.07423756e-08]))
```

```
('stats:', [array([682.02929708]), 6, array([2.30255894e+00, 7.89025775e-01, 2.67934772e-01,
6.14244114e-02,
9.88180575e-03, 1.01134227e-03]), 6.8833827526759706e-15])
```

- 3. Discrete Fourier Transform Amplitudes:** Using the DFT will convert the time domain data into the frequency domain data; any peaks in the frequency would represent the presence of external introduced variability in the CGM data, which possibly could be due to the Meal event. I will consider Top 5 DFT amplitudes from the set of peaks.

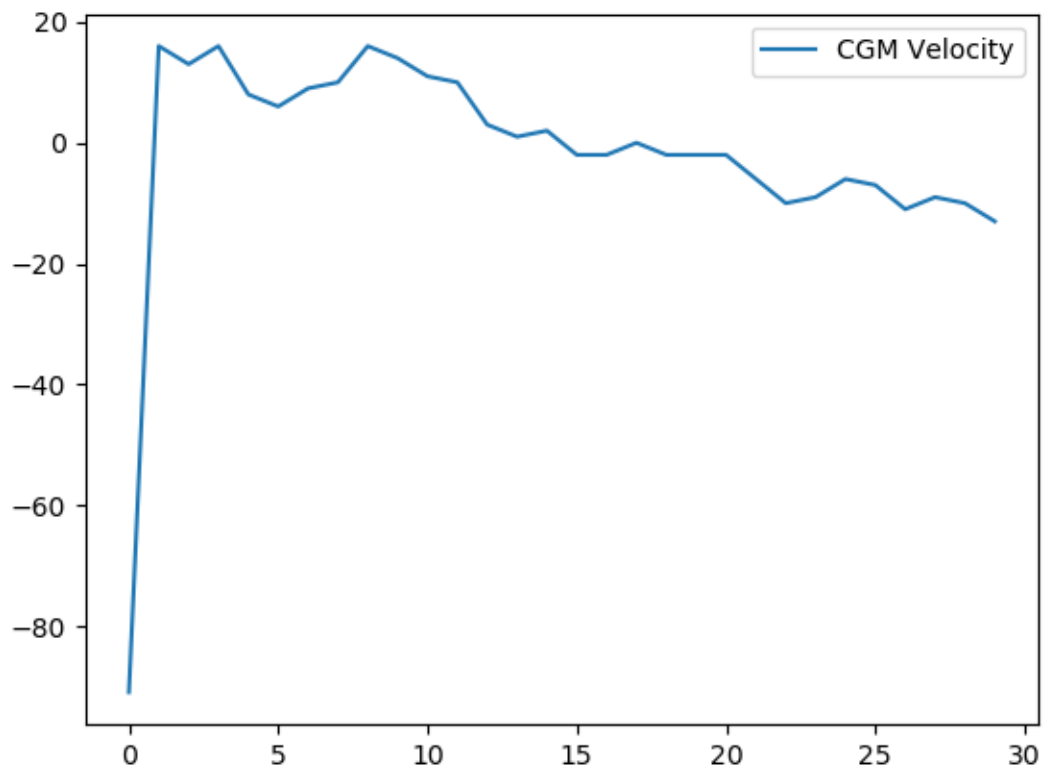


```
('Actual:', array([ 91., 92., 91., 90., 91., 91., 97., 99., 98., 99., 99.,
102., 113., 131., 146., 163., 179., 194., 207., 213., 221., 228.,
236., 241., 246., 252., 254., 258., 260., 258., 256.]))

('F2Coefficients:', array([ 9.24900354e+01, -1.62050652e+00, 2.21445507e-01, -1.05780473e-02,
2.31386255e-04, -2.43091410e-06, 1.21760896e-08, -2.34768220e-11]))

('F2stats:', [array([285.94969872]), 8, array([2.65288540e+00, 9.11411744e-01, 3.48618268e-01,
9.76564499e-02,
2.10560286e-02, 3.55814540e-03, 4.52573869e-04, 3.78271642e-05]), 6.8833827526759706e-15])
```

4. **Top 3 Peaks of Velocity (ROC) of CGM Data:** Will calculate the velocity of the CGM Data by differentiating the Data and will check how many zero crossings are there; there will be some conditions for zero crossing to be worthy of consideration. Once those range have identified we can use this feature in predicting Meal or No Meal.



```
('CGMVelocity:', array([-90.9, 16., 13., 16., 8., 6., 9., 10., 16.,
14., 11., 10., 3., 1., 2., -2., -2., 0.,
-2., -2., -2., -6., -10., -9., -6., -7., -11.,
-9., -10., -13. ]))
```

```
('ArrayPeaksIndex:', array([ 1, 3, 8, 14, 17, 24, 27], dtype=int64))
```

```
('PeakArray:', array([16., 16., 16., 2., 0., -6., -9.]))
```

```
('PeakArray:', array([16., 16., 16., 2., 0., -6., -9.]))
```

```
('Top 3 Peaks from CGM Velocity:', array([16., 16., 16.]))
```

**I am currently thinking on the feature based on the following model:**

**Differential equation-based Model for Blood Glucose levels:** This approach I am proposing to use in the next project as it includes the insulin infusion rates as well.

It is proposed that the kinetics of the reaction steps are first order, so that

$$\frac{dC}{dt} = -k_1 C \quad (1) \quad \text{and} \quad \frac{dG}{dt} = k_1 C - k_2 (G - G_0) \quad (2)$$

where  $C$  represents the concentration of carbohydrates in the stomach and  $G$  represents the concentration of glucose in the blood.  $G_0$  is the baseline value  $G$  at  $t = 0$ . The initial conditions are  $C(0) = A_0$  and  $G(0) = G_0$ . The rate constant  $k_1$  is related to how quickly the food is digested while the rate constant  $k_2$  is associated with how quickly insulin is released.

Reference:

<https://scholarcommons.usf.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=4830&context=ujmm>

Idea would be to model the blood glucose levels as differential equations and calculate all the parameters of the glucose equations using the data, and once the new data is given the feature values from it could be calculated & compared to the range of modeled value based on the training data.

Approach for working with Numerical NaNs values:

standard and often very good approach is to replace the missing values with mean, median or mode. For numerical values you should go with mean, and if there are some outliers try median (since it is much less sensitive to them).

```
from sklearn.preprocessing import Imputer
```

```
imputer = Imputer(missing_values=np.nan, strategy='median', axis=0)
```

```
census_data[['fnlwgt']] = imputer.fit_transform(census_data[['fnlwgt']])
```

b) For each time series explain why you chose such feature (5 points each) total 20

1. **Coefficient of variation** (CV=standard deviation / mean): As a rule of thumb, a CV  $\geq 1$  indicates a relatively high variation, while a CV  $< 1$  can be considered low. This means that distributions with a coefficient of variation higher than 1 are considered to be high variance whereas those with a CV lower than 1 are considered to be low-variance. Reference:

<https://www.cambridge.org/core/journals/journal-of-french-language-studies/article/e-in-normandy-the-sociolinguistics-phonology-and-phonetics-of-the-loi-de-position/1B6B9D55EE7865E322D6EEC62D8CD2E9>

I am choosing CV as if there is no variation in the data or very less variation then the modeled value, it will classify as a No Meal.

2. **Polynomial Curve fitting the CGM Data (Coefficients):** I would try to fit a polynomial to different Lunch events CGM data, would choose the polynomial of an order which represents the data with the least error, error can be modeled by calculating goodness of fit (Chi Method) can be calculated and range of factors can be modeled based on the error.

I am choosing coefficient of polynomial, I will model them according to the meal data, Now for the test data if I find the coefficient are out of modeled range, it will classified as the No meal data.

3. **Discrete Fourier Transform Amplitudes:** Using the DFT will convert the time domain data into the frequency domain data; any peaks in the frequency would represent the presence of external introduced variability in the CGM data, which possibly could be due to the Meal event. I will consider Top 5 DFT amplitudes from the set of peaks.

I am choosing the fft coefficients; I will model these in the acceptable range, if the values goes out of range it will mean no meal data.

Also if the FFT curve doesn't have any peaks it will also mean No meal data.

4. **Top 3 Peaks of Velocity (ROC) of CGM Data:** Will calculate the velocity of the CGM Data by differentiating the Data and will check how many zero crossings are there; there will be some conditions for zero crossing to be worthy of consideration. Once those range have identified we can use this feature in predicting Meal or No Meal.

I am choosing Top 3 peaks of CGM velocity, if the peaks are less than the modeled range the test data will be classified as No meal data.



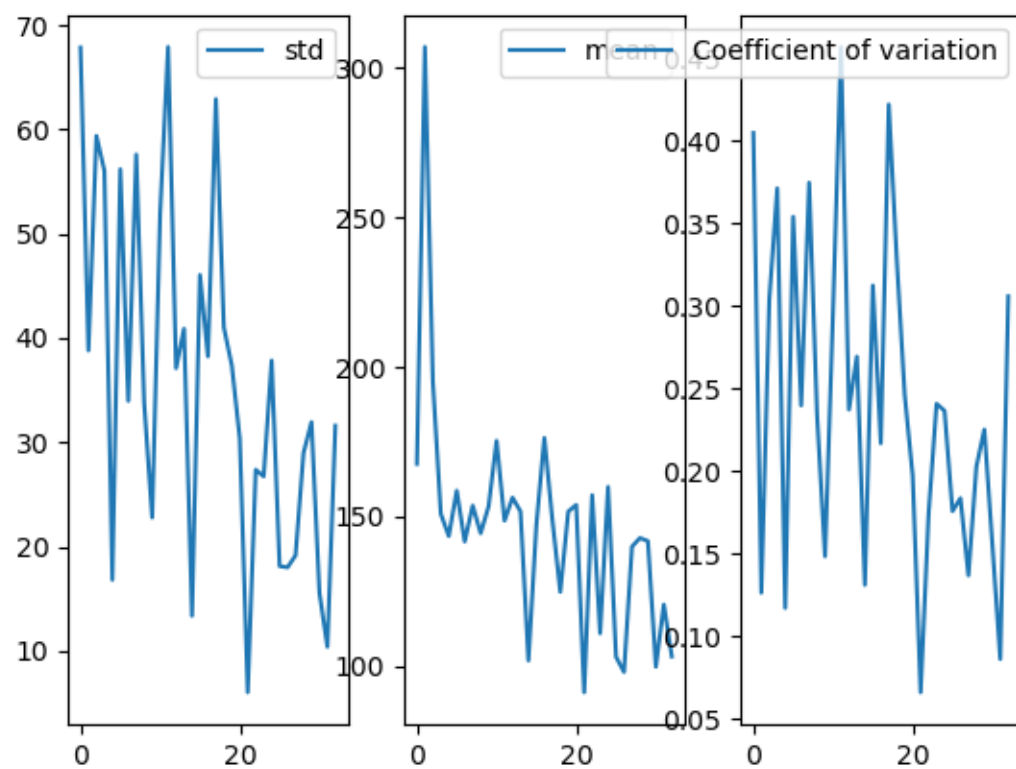
C. Show values of each of the features and argue that your intuition in step b is validated or disproved? (5 points each ) total 20

5. **Coefficient of variation** (CV=standard deviation / mean): As a rule of thumb, a  $CV \geq 1$  indicates a relatively high variation, while a  $CV < 1$  can be considered low. This means that distributions with a coefficient of variation higher than 1 are considered to be high variance whereas those with a CV lower than 1 are considered to be low-variance. Reference:

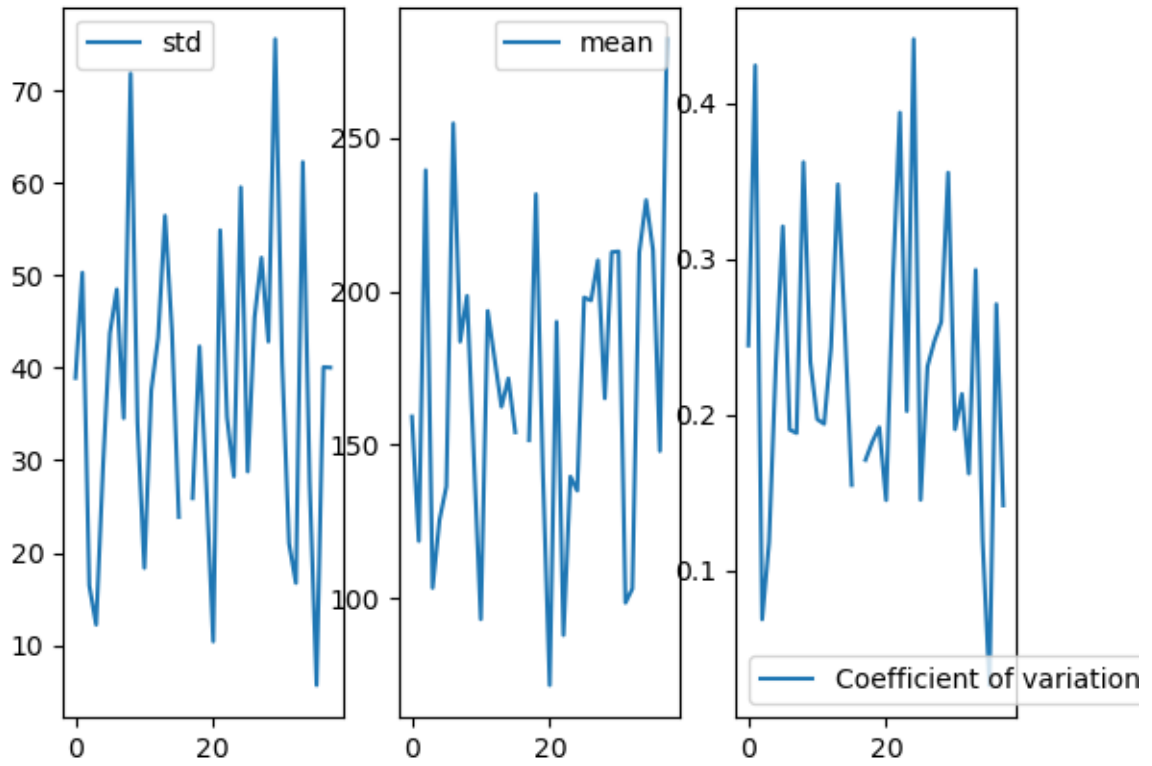
<https://www.cambridge.org/core/journals/journal-of-french-language-studies/article/e-in-normandy-the-sociolinguistics-phonology-and-phonetics-of-the-loi-de-position/1B6B9D55EE7865E322D6EEC62D8CD2E9>

I am choosing CV as if there is no variation in the data or very less variation then the modeled value, it will classify as a No Meal.

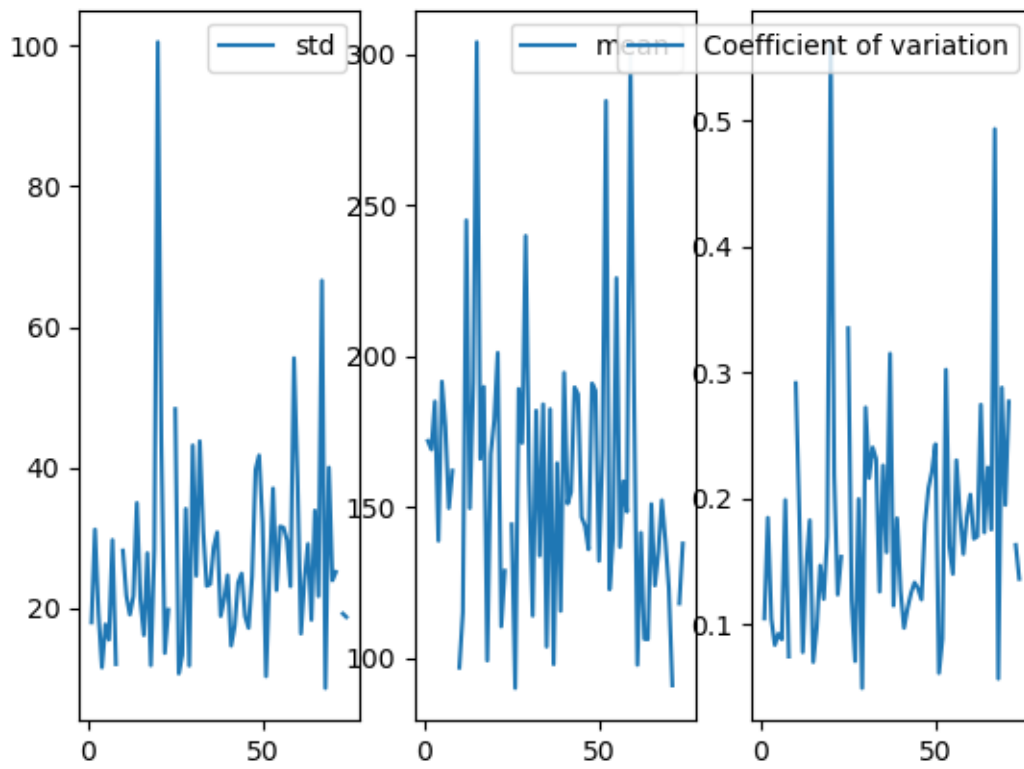
Patient 1 All events Data



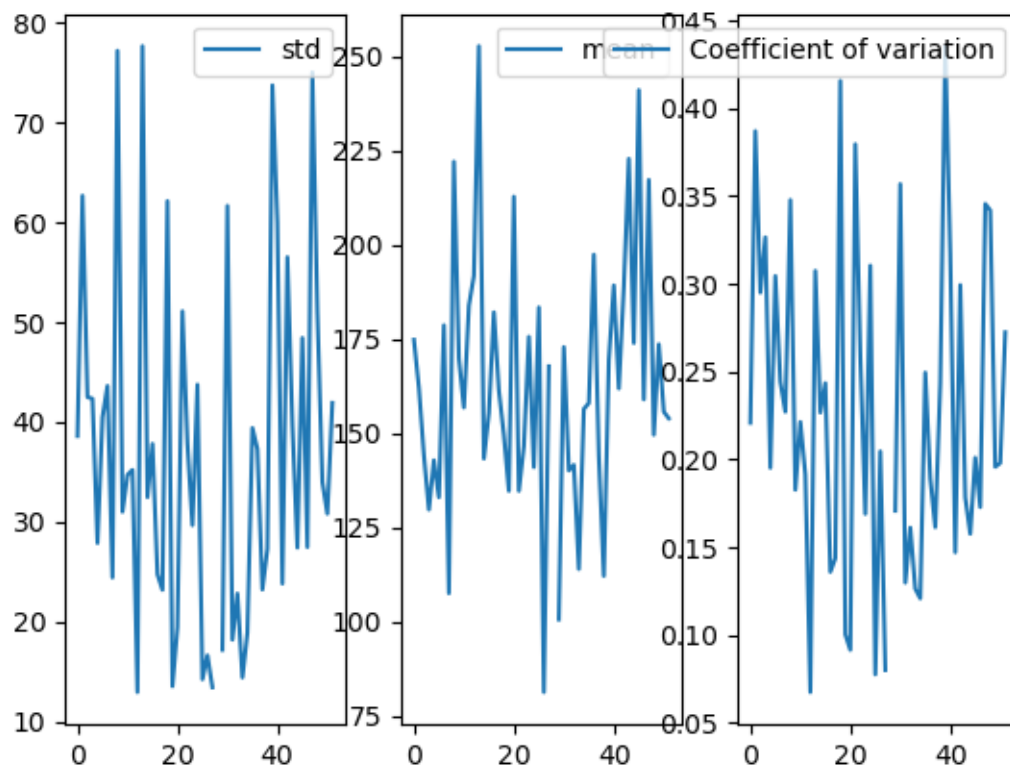
Patient 2 All events Data



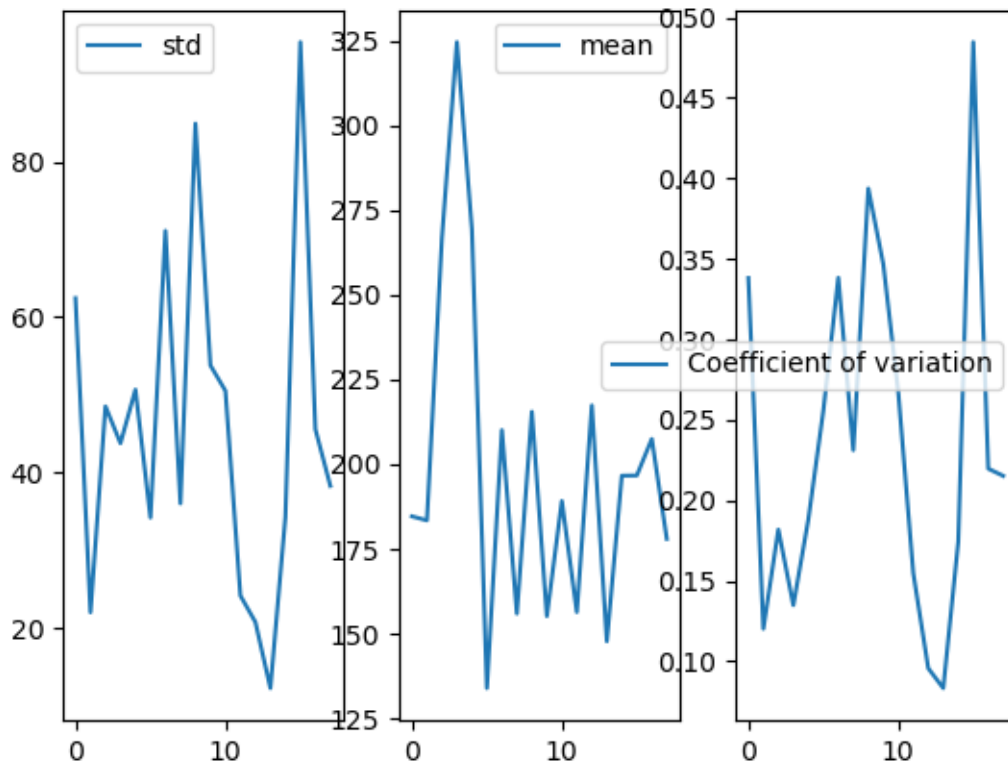
Patient 3 All events Data



Patient 4 All events Data



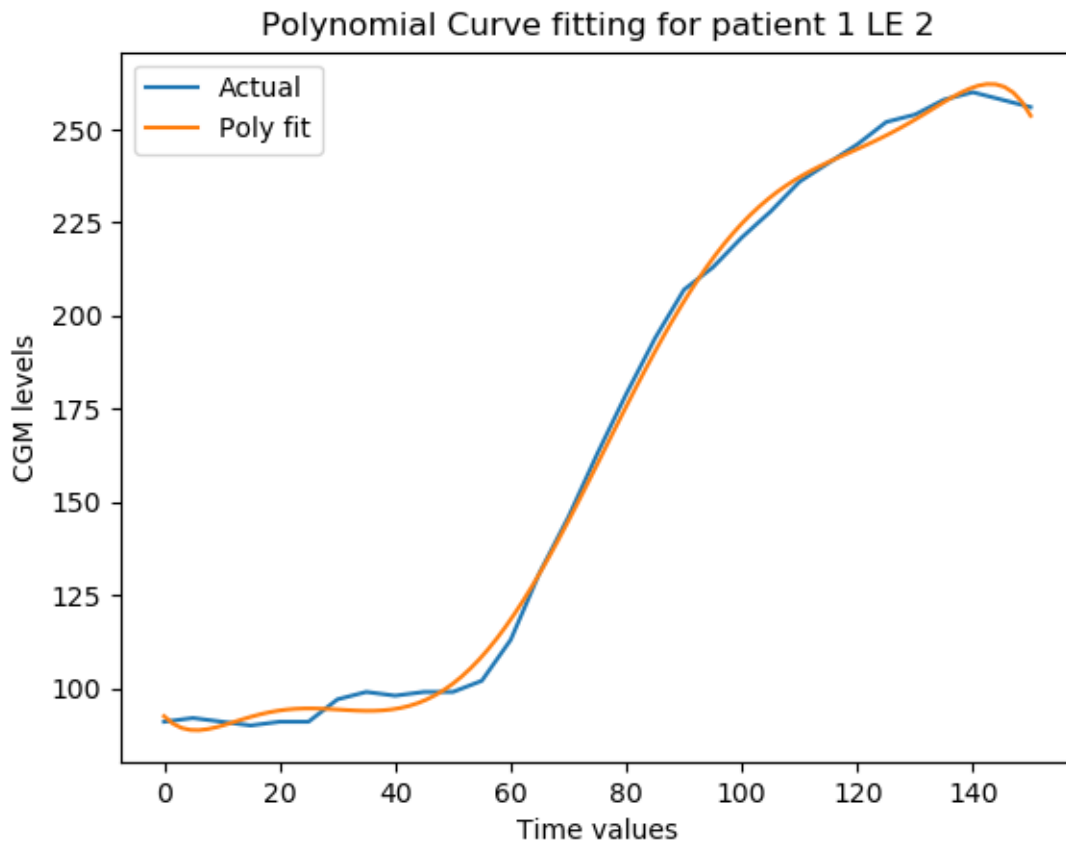
Patient 5 All events Data



If we see the CV data for all the features is within a tight range 0.1 to 0.45; this will classify as meal; as whole of the data is a meal data anything outside of this can be no meal. Also for modeling probabilistic inference can be approached.

2. **Polynomial Curve fitting the CGM Data (Coefficients):** I would try to fit a polynomial to different Lunch events CGM data, would choose the polynomial of an order which represents the data with the least error, error can be modeled by calculating goodness of fit (Chi Method) can be calculated and range of factors can be modeled based on the error.

I am choosing coefficient of polynomial, I will model them according to the meal data, Now for the test data if I find the coefficient are out of modeled range, it will classified as the No meal data.



The Polynomial of 7 degree gives 682 as the standard root mean square error.

```
('stats:', [array([682.02929708]), 6, array([2.30255894e+00, 7.89025775e-01, 2.67934772e-01,
6.14244114e-02,
9.88180575e-03, 1.01134227e-03]), 6.8833827526759706e-15])
```

For other case it gives 285.94969872

```
('F2Coefficients:', array([ 9.24900354e+01, -1.62050652e+00, 2.21445507e-01, -1.05780473e-02,
2.31386255e-04, -2.43091410e-06, 1.21760896e-08, -2.34768220e-11]))
('F2stats:', [array([285.94969872]), 8, array([2.65288540e+00, 9.11411744e-01, 3.48618268e-01,
9.76564499e-02,
2.10560286e-02, 3.55814540e-03, 4.52573869e-04, 3.78271642e-05]), 6.8833827526759706e-15])
```

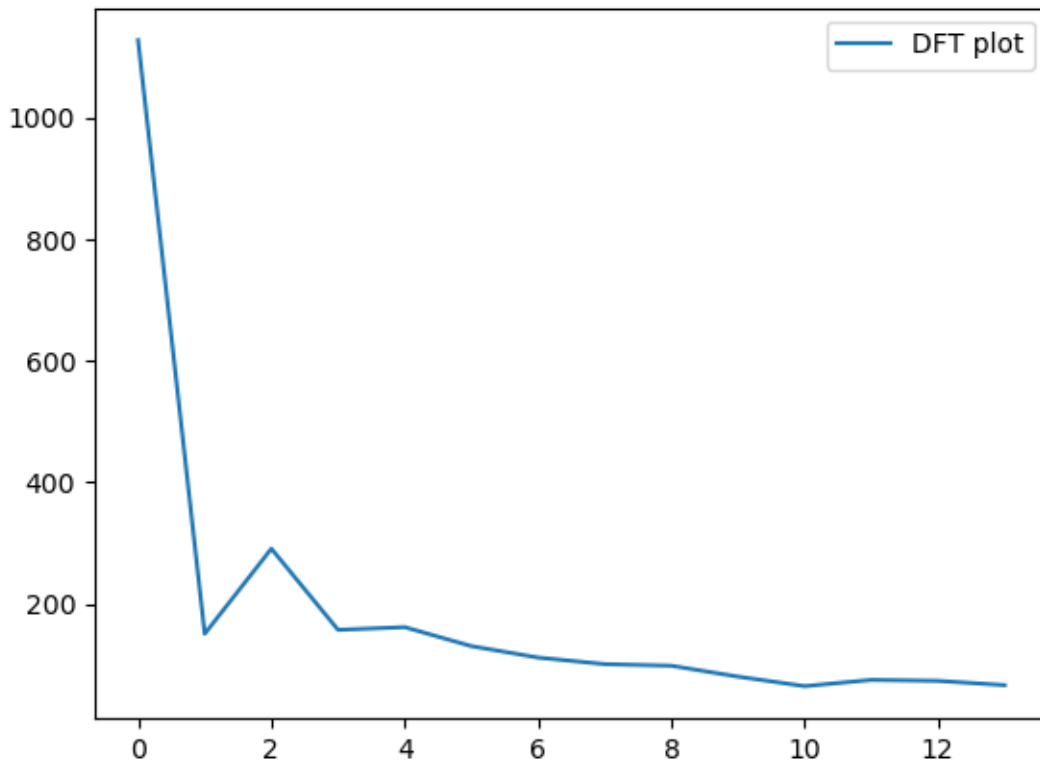
Its within 0 to 1000 range which will imply a good fit. Also the coefficient will be modeled acrossing to the analysis on the whole data, by working on the meal data we find these values to be in range. All the data is there in the code, can't be put in the report only.

- Discrete Fourier Transform Amplitudes:** Using the DFT will convert the time domain data into the frequency domain data; any peaks in the frequency would represent the presence of

external introduced variability in the CGM data, which possibly could be due to the Meal event. I will consider Top 5 DFT amplitudes from the set of peaks.

I am choosing the fft coefficients; I will model these in the acceptable range, if the values goes out of range it will mean no meal data.

Also if the FFT curve doesn't have any peaks it will also mean No meal data.

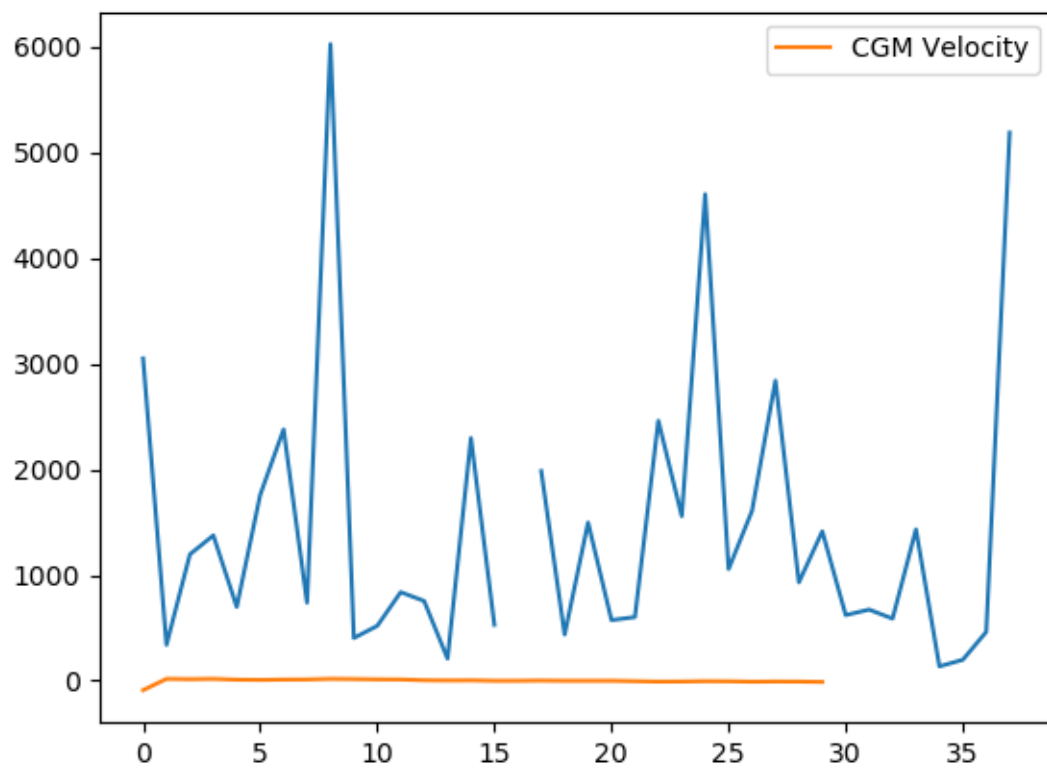


```
('scipyfft:', array([469.      , -2.80901699, -0.58778525, -1.69098301,
0.95105652]))
```

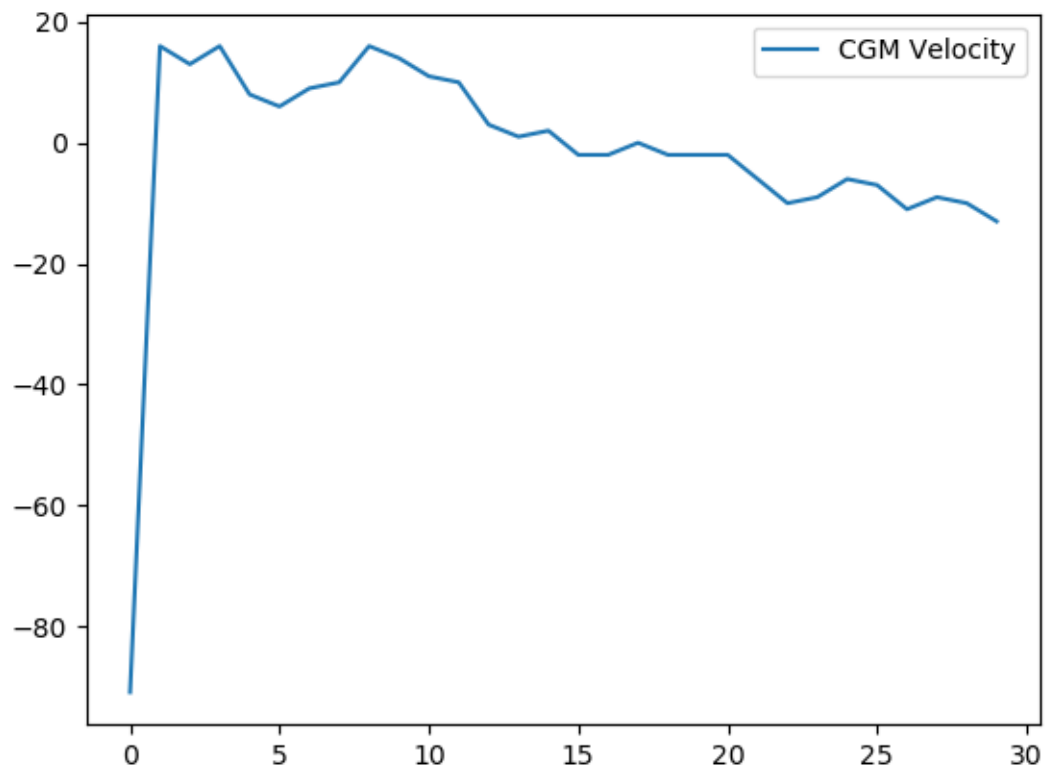
The values of the FFT components are also in the prescribed range. On working on whole data we can model the range of the values for our prediction algorithms.

- 4. Top 3 Peaks of Velocity (ROC) of CGM Data:** Will calculate the velocity of the CGM Data by differentiating the Data and will check how many zero crossings are there; there will be some conditions for zero crossing to be worthy of consideration. Once those range have identified we can use this feature in predicting Meal or No Meal.

I am choosing Top 3 peaks of CGM velocity, if the peaks are less than the modeled range the test data will be classified as No meal data.







```
('CGMVelocity:', array([-90.9, 16., 13., 16., 8., 6., 9., 10., 16.,
14., 11., 10., 3., 1., 2., -2., -2., 0.,
-2., -2., -2., -6., -10., -9., -6., -7., -11.,
-9., -10., -13.]))

('ArrayPeaksIndex:', array([ 1, 3, 8, 14, 17, 24, 27], dtype=int64))

('PeakArray:', array([16., 16., 16., 2., 0., -6., -9.]))

('PeakArray:', array([16., 16., 16., 2., 0., -6., -9.]))

('Top 3 Peaks from CGM Velocity:', array([16., 16., 16.]))
```

We are able to calculate finite values for the CGM velocity, we can model the range of these values as well.

So all feature values are validated with the data that I have worked upon, but I can model the range completely by working on more data.

- D. Create a feature matrix where each row is a collection of features from each time series. SO if there are 75 time series and your feature length after concatenation of the 4 types of feates is 17 then the feature matrix size will be 75 X 17 (10 points)

Feature matrix is Created. It is of (216,17) Dimensions.

```
[[ 4.04789088e-01 1.67348238e+02 -1.07139551e+00 ... 1.80000000e+01
 1.70000000e+01 8.00000000e+00]
 [ 1.26447239e-01 3.06309358e+02 -1.45362026e+00 ... 1.60000000e+01
 1.60000000e+01 1.60000000e+01]
 [ 3.04094568e-01 1.96747940e+02 -1.64851775e+00 ... 2.60000000e+01
 1.50000000e+01 5.00000000e+00]
 ...
 [ 1.19436236e-01 9.22156535e+01 -1.01057897e+00 ... 1.00000000e+01
 4.00000000e+00 0.00000000e+00]
 [ 7.08597554e-02 1.87485908e+02 1.38907185e+00 ... 1.40000000e+01
 9.00000000e+00 7.00000000e+00]
 [ 1.99509235e-01 1.73783622e+02 -1.24227075e+00 ... 2.17200000e+01
 1.30000000e+01 1.00000000e+00]]
```

Above can be viewed in the Code results as the variable name `feature_matrix`.

```
##### Part 4: ##### (d): Feature matrix

feature_matrix = []

cgmLpAll = cgmLp1.append(cgmLp2).append(cgmLp3).append(cgmLp4).append(cgmLp5)

stdarr = []
meanarr = []
CVA11 = []

for i in range(0,len(cgmLpAll)):
    #print(cgmLp1.iloc[i])
    stdarr.append(numpy.std(cgmLpAll.iloc[i]))
    meanarr.append(numpy.mean(cgmLpAll.iloc[i]))
    tCV = numpy.std(cgmLpAll.iloc[i]) / numpy.mean(cgmLpAll.iloc[i])
    #print(tCV)
    CVA11.append(tCV)

print(CVA11)

#CVA11 = np.asarray(CVA11)

CVmat = np.asmatrix(np.asarray(CVA11).reshape(216L,1))

coefficientp = []
```

```

fit_statsp = []

timearrAll = numpy.arange(0,206,5)

for i in range(0,len(cgmLpAll)):
    OrdarrLEp = numpy.where(numpy.isnan(cgmLpAll.iloc[i].values),
numpy.mean(cgmLpAll.iloc[i]), cgmLpAll.iloc[i].values)[::-1]
    c, stats = numpy.polynomial.polynomial.polyfit(timearrAll, OrdarrLEp, 7,
full=True)
    coefficientp.append(c)
    fit_statsp.append(stats)

print("Coefficienct patient:", coefficienttp)

print("fit_stats patient:", fit_statsp[1][1])
Polymat = numpy.asmatrix(coefficienttp)

CoeffPAll = []
for i in range(0,len(cgmLpAll)):
    OrdarrLEp = numpy.where(numpy.isnan(cgmLpAll.iloc[i].values),
numpy.mean(cgmLpAll.iloc[i]), cgmLpAll.iloc[i].values)[::-1]
    Coeff = scipy.fftpack.rfft(OrdarrLEp, n=5)
    CoeffPAll.append(Coeff)

print(CoeffPAll)

fftmat = numpy.asmatrix(CoeffPAll)

Top3PeaksPAll = []

for i in range(0,len(cgmLpAll)):
    OrdarrLEp = numpy.where(numpy.isnan(cgmLpAll.iloc[i].values),
numpy.mean(cgmLpAll.iloc[i]), cgmLpAll.iloc[i].values)[::-1]
    CGMVelocity = numpy.diff(OrdarrLEp, n=1)
    peaksarrixd = scipy.signal.find_peaks(CGMVelocity)
    peakArray = CGMVelocity[peaksarrixd[0]]
    if (len(peakArray) == 0):
        Top3Peaks = np.array([0,0,0])
    else:
        Top3Peaks = peakArray[heapq.nlargest(3, range(len(peakArray)),
peakArray.take)].reshape(3L,)
        # print(peakArray, len(peakArray))
        Top3PeaksPAll.append(Top3Peaks)

print("Top3PeaksP1:", Top3PeaksPAll)

PeakVelocitymat = numpy.asmatrix(Top3PeaksPAll)

feature_matrix = np.concatenate((CVmat, Polymat,fftmat,PeakVelocitymat), axis=1)
feature_matrix = np.nan_to_num(feature_matrix)
print(feature_matrix, "shape of feature matrix:", feature_matrix.shape)

```

- E. Provide this feature matrix to PCA and derive the new feature matrix. Chose the top 5 features and plot them for each time series. (5 points)

This has been done in the code.

```
#Using the PCA to generate the new PCA matrix
pca = PCA(n_components=5)
pca_reduced_new_mat = pca.fit_transform(feature_matrix)

plt.plot(pca_reduced_new_mat[:,0])
plt.xlabel('Meal events across the data')
plt.ylabel('PCA component1 ');
plt.show()

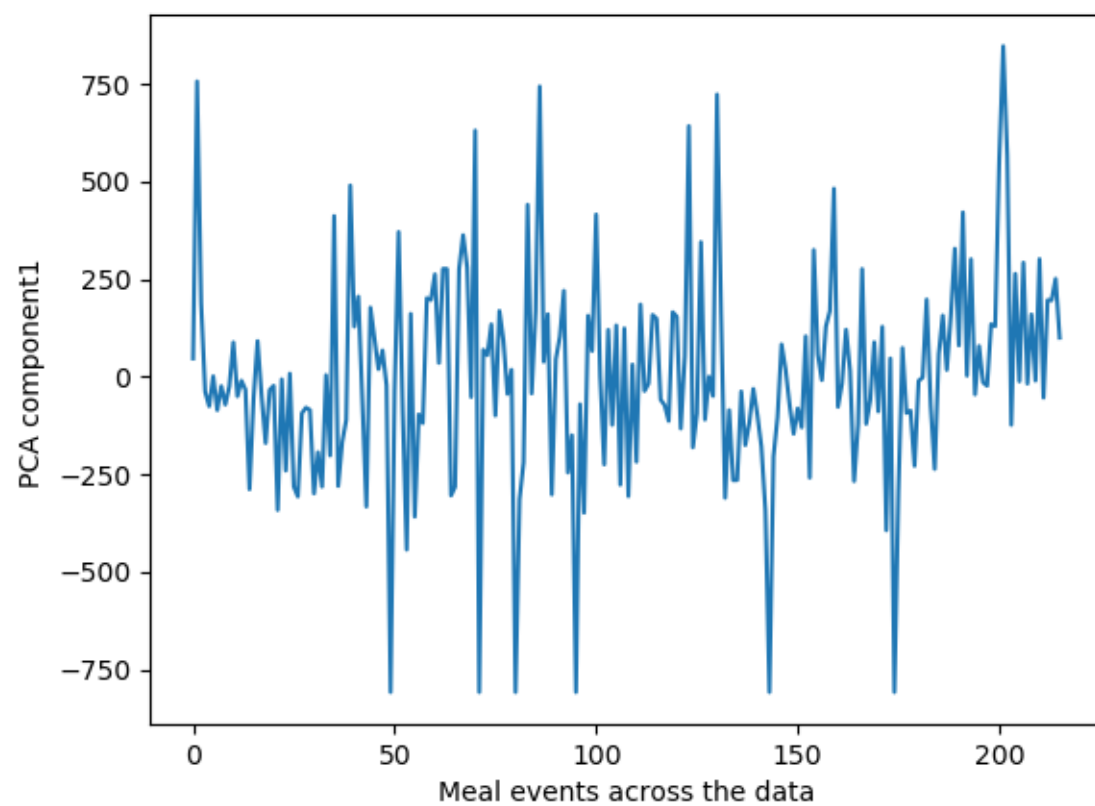
plt.plot(pca_reduced_new_mat[:,1])
plt.xlabel('Meal events across the data')
plt.ylabel('PCA component2 ');
plt.show()

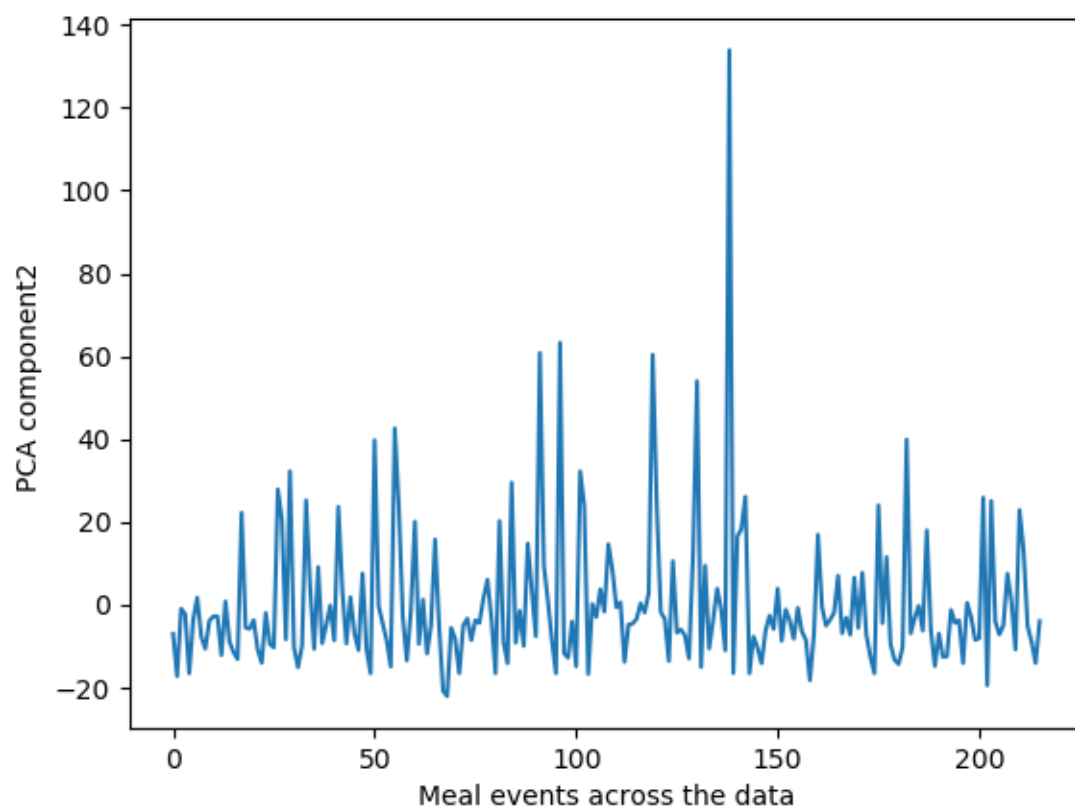
plt.plot(pca_reduced_new_mat[:,2])
plt.xlabel('Meal events across the data')
plt.ylabel('PCA component3 ');
plt.show()

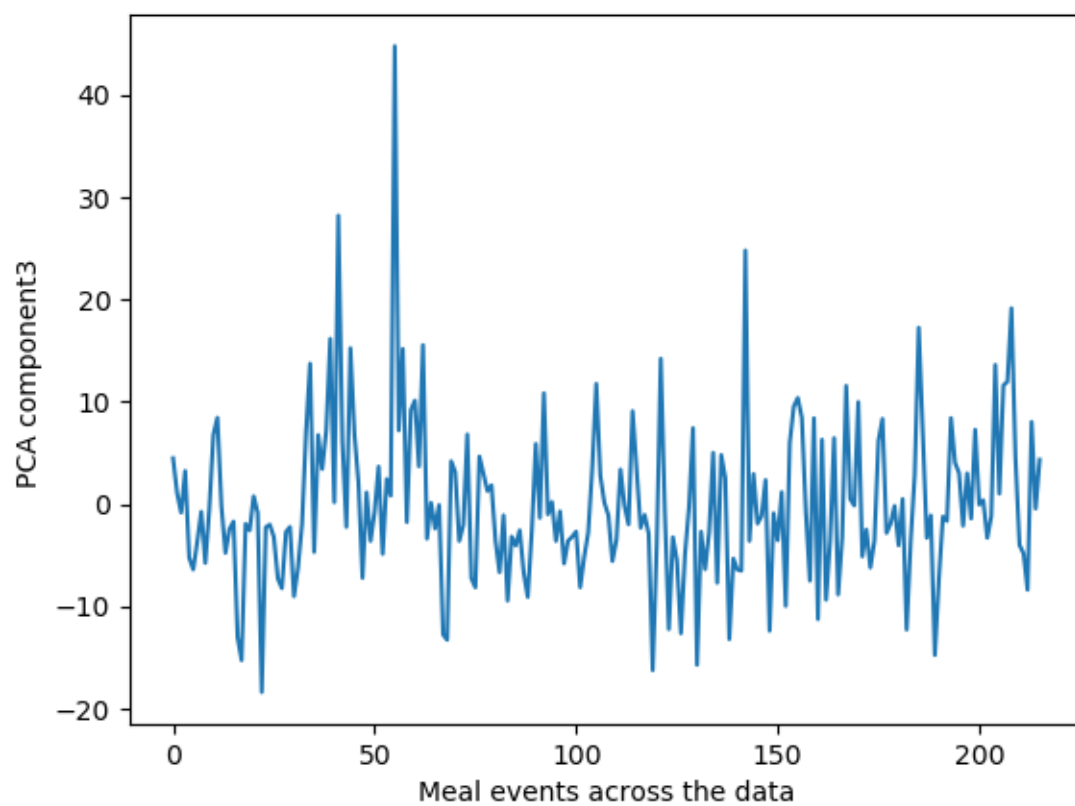
plt.plot(pca_reduced_new_mat[:,3])
plt.xlabel('Meal events across the data')
plt.ylabel('PCA component4 ');
plt.show()

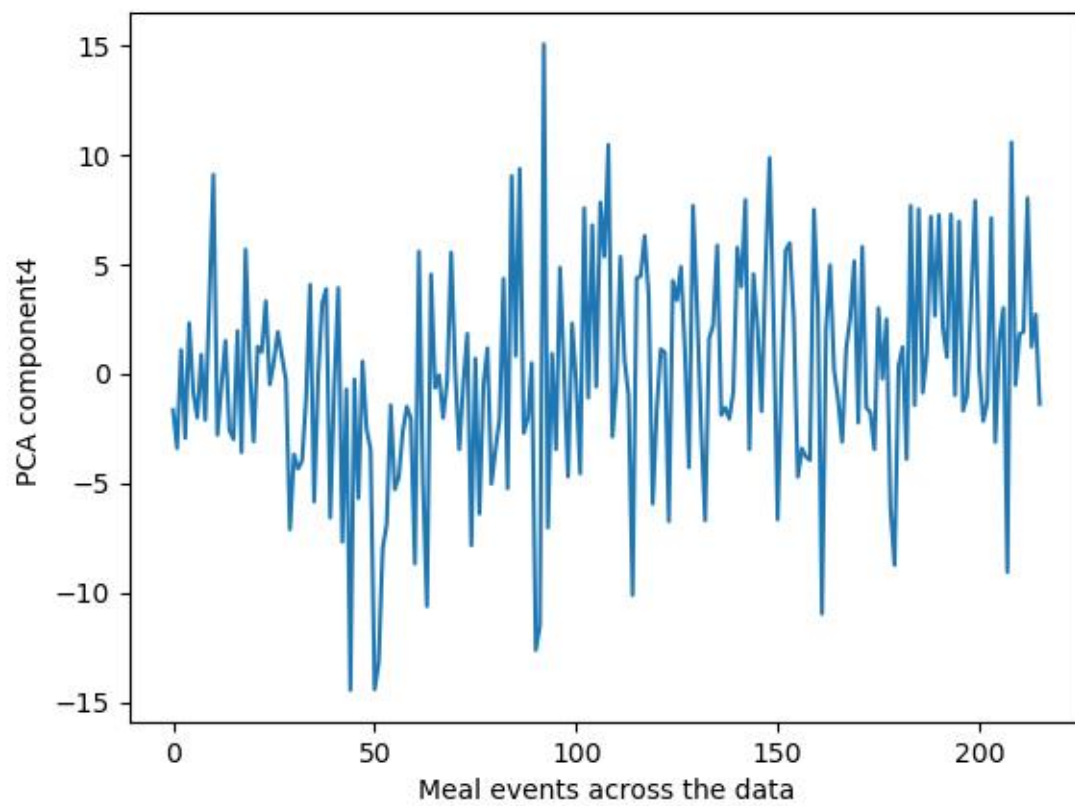
plt.plot(pca_reduced_new_mat[:,4])
plt.xlabel('Meal events across the data')
plt.ylabel('PCA component5 ');
plt.show()

print(pca_reduced_new_mat, "shape:", pca_reduced_new_mat.shape)
```

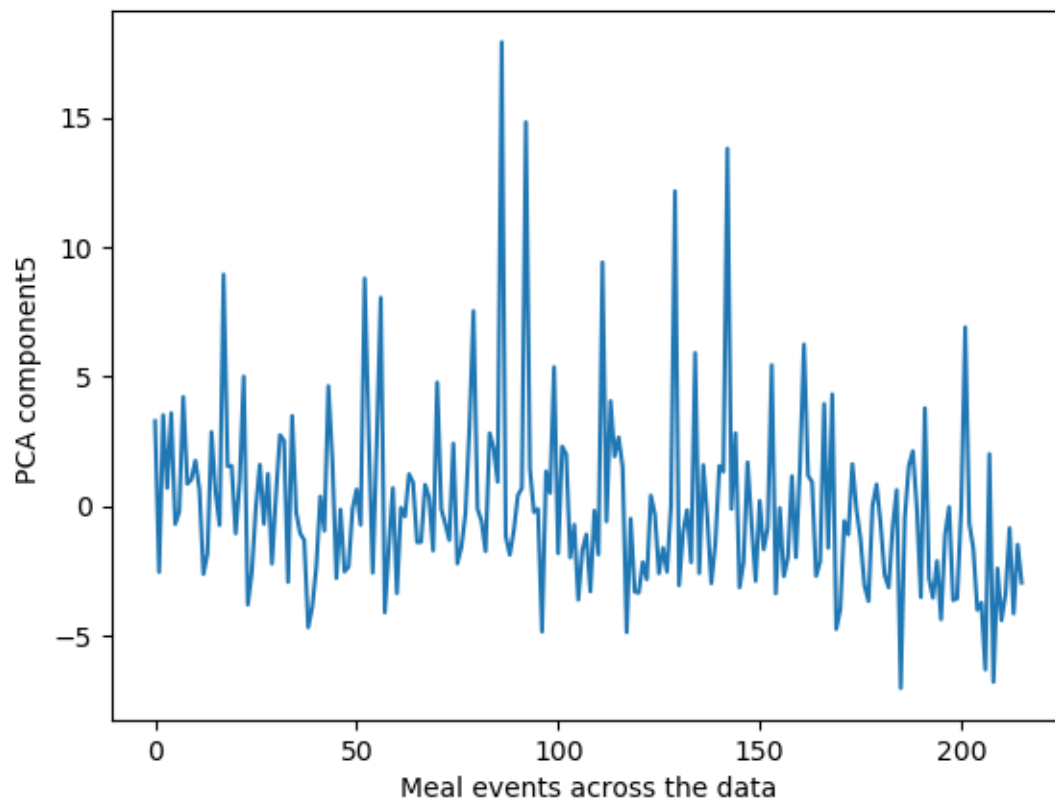












- F. For each feature in the top 5 argue why it is chosen as a top five feature in PCA? (3 points each)  
total 15

Based on the variance data we can argue that in the direction of first component the variance is maximum and it decreases.

PCA helps us to identify patterns in data based on the correlation between features. In a nutshell, PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one.

Since we want to reduce the dimensionality of our dataset by compressing it onto a new feature subspace, we only select the subset of the eigenvectors (principal components) that contains most of the information (variance). The eigenvalues define the magnitude of the eigenvectors, so we have to sort the eigenvalues by decreasing magnitude; we are interested in the top  $k$  eigenvectors based on the values of their corresponding eigenvalues.

But before we collect those  $k$  most informative eigenvectors, let's plot the variance explained ratios of the eigenvalues. The variance explained ratio of an eigenvalue  $\lambda_j$  is simply the fraction of an eigenvalue  $\lambda_j$  and the total sum of the eigenvalues:

$$\frac{\lambda_j}{\sum_{j=1}^d \lambda_j}$$

Using the NumPy cumsum function, we can then calculate the cumulative sum of explained variances, which we will then plot via matplotlib's step function:

