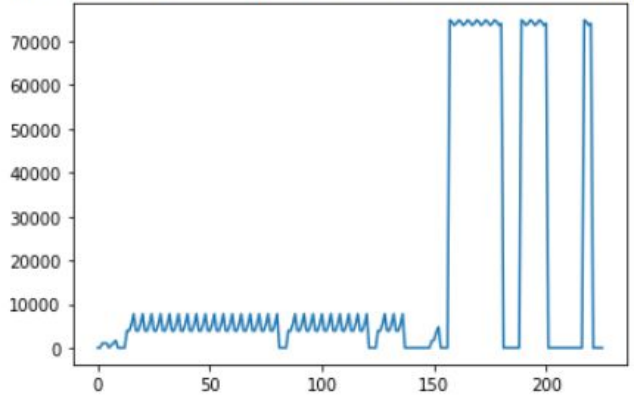
Telecom Churn Case Study Steps to Solution

- **Import Libraries**
  a. Pandas
  b. Matplotlib
  c. Seaborn
  d. Sklearn
- **Import csv in a DataFrame**
- **Data Quality Checks**
  a. Shape : (99999, 226)
  b. Column Names
  c. Null Values in a feature
  d. Call Describe function

- **Plot (No. of Null Values vs Feature Index)**
    - Dropped features having no. of null values greater than 50% of total present values (set a t**hreshold=0.5 *len(df)**)
    - Boils down feature numbers to **186 from 226**

<matplotlib.axes._subplots.AxesSubplot at 0x7f35aa6409b0>



Number of Null Values Vs Feature Index

- Evaluated average of Recharge amount of 1st month and 2nd month (in order to filter out the high value customers) [A feature was thus added Average(amt 1st, amt 2nd)]

- Filtered out those customers who have mean recharge amount greater or equal to 70th percentile of mean value of recharge amount 1st and 2nd.
    - Left with high Value Customers **(29k rows 187 features)**

```
filtered_df.shape
```

```
(29979, 187)
```

Dependent Variable

- Created dependent variable on given conditions
  - **Those who have not made any calls (either incoming or outgoing) AND have not used mobile internet even once in the churn phase.**
  - **Code snippet:**

```
filtered_df['churn_flag'] = np.where(
                            ((filtered_df['total_ic_mou_9'] == 0.00) | (filtered_df['total_og_mou_9'] == 0.00))
                            & ((filtered_df['vol_2g_mb_9'] == 0.00) | (filtered_df['vol_3g_mb_9'] == 0.00))
                            , 1, 0
                           )
```

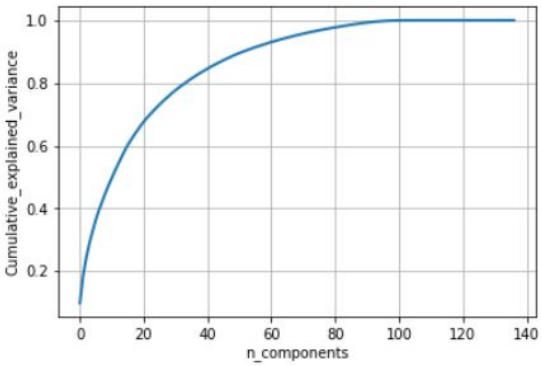
- After tagging churners, **removed all the attributes corresponding to the churn phase**
- We were left with **187 features but there were missing values present in data matrix,** hence we **imputed them with median values and dropped some Date features.**
- **To Study the Variability and to reduce the dimensions PCA has to be used**
- **Before Implementing PCA, there are couple of techniques need to be done**
  - Column Standardization (mean=0 ,stddev=1) of all features (to scale down)
  - Code Snippet:

```
[ ]  # Data-preprocessing: Standardizing the data

     from sklearn.preprocessing import StandardScaler
     standardized_data = StandardScaler().fit_transform(filtered_df)
     print(standardized_data.shape)
```

Principal Component Analysis

- In a nutshell, PCA evaluates:
  - **Covariance Matrix of data**
  - **Eigen Vectors and Eigen Values (for every feature)**
- Based on Eigen Vectors we study the geometric variablity and Eigen Values explains the variability of data in percentage on different vector
- Plotted a curve to study the variability :



- Previous Curve indicate our data won't lose the variability if we don't drop our first 70-75 features (as they account for 100% variability in our data)
- So, we'll drop all features but those having the highest significance on variability
- Shape of reduced data:

```
shape of pca_reduced.shape =  (29979, 70)
```

Training Model (Logistic Regression)

- Trained first 75%, leaving out 25% to test the model
- Results for Confusion Matrix on testing data were:

```
[ ] sklearn.metrics.confusion_matrix(y_test,predictions)

array([[6531,  108],
       [ 658,  198]])
```

- Accuracy:

```
sklearn.metrics.accuracy_score(y_test,predictions)

0.8977985323549033
```