Machine Learning and Heart Rate Zones

Project Report

Tom Springett

10/26/20

1. Project Definition

To improve physical fitness, there has been a surge of interest in high intensity training. This involves short bursts of all-out exercise followed by brief periods of rest. The 'on' duration can be as short as 30 seconds. These pulsed sessions have been found to have the maximum benefit for the shortest workout periods. These high intensity sessions (HIIT) not only benefits elite athletes but people at all levels of fitness and age. With the proper guidance, 90-year old's have been shown to benefit from HIIT.

Heart rate monitors are used to determine training intensity. The problem is determining what is high intensity. There are various calculations such as 220 – age for maximum heart rate. And then high intensity may be 70-90% of maximum. But these are guidelines and have not been calculated vigorously nor applied consistently. Another method is to go to a lab with an electrocardiogram and do a stress test. But this option is expensive and not widely available. A new technique is needed.

This project will investigate machine learning techniques for determining high intensity heart rates. First, data will be extracted from an Apple watch. It will then be be loaded and filtered into a pandas dataframe. The data that will be analyzed is beats per minute (BPM). This one-dimensional data will be visualized with histograms. Various industry benchmarks will be applied to see how well they match up with the distribution.

Next modeling will be done. The models selected will focus on clustering and unsupervised learning. These models will be implemented and their hyperparameters optimized. Their performance will be evaluated based on silhouette scores. The modeled results will then be compared to the industry benchmarks.

These algorithms could then be embedded into heart rate monitors and apps.

The remainder of this report will cover analyzing, implementing and discussing the project. A few key code snippets are included. The full code, along with commentary, is available in a Jupyter Notebook in a GitHub repository at: https://github.com/ThomasSpringett/HeartRate

2. Analyze

Explore the Data

Summary statistics and graphical analysis are used to understand and explore the dataset.

The heart rate data was obtained from an Apple Watch and exported from Healthkit. Due to the sensitivity and availability of health data, the dataset was collected from only one white 63-year-old male. The watch hardware limits are from 30 bpm to 210 bpm. Values outside of this range were removed. The dataset has over 349,000 data points. A random sample of 10,000 from this population was generated. This did not significantly affect the distribution or results but did improve execution time.

Maximum Heart Rate

The first step in determining high intensity heart rate zones is to determine maximum heart rate. There are several methods for this:

1. 220 - age. 0r 158bpm for a 62-year-old.
2. Heart Rate Monitor data collected over a period of months with various activities.
3. Using a treadmill or stationary bike with an ECG.

Option number 1 is the most common for folks that do not have access to the treadmill and ECG. It is remarkable how well it works. The body seems to have a built-in algorithm. After the maximum heart rate is determined, target heart rates can be determined:

Descriptive Statistics

The summary below shows that there were 349,781 heat beat measurements. The minimum was 5 and the maximum is 215. These values are outside of the sensor's valid range, which is 30 to 210. Consequently, they will be deleted. The mean of 78 bpm is significantly above the median at 69 bpm, suggesting there the distribution has a high-end tail. The standard deviation of 24 is also high, further suggesting a wide range of values.

Population	
	BPM
count	349781
mean	78
std	24
min	5
25%	61
50%	69
75%	86
max	200

Sample	
	BPM
count	10000
mean	77
std	23
min	38
25%	61
50%	69
75%	86
max	200

Sampling

For performance metrics, this analysis relies heavily on the silhouette score, which is computationally intensive. In order to reduce execution times, a sample was drawn from the original population. The sample size was 10,000. Descriptive statistics, shown above, were again generated on this sample and the distribution is virtually unchanged.

The sample was generated with the following code snippet:

```
import random
df_s=random.sample(list(df_limit.BPM), sample_size)
df_sample = pd.DataFrame(df_s)
```

Feature Addition - Hear Rate Intensity Zones

Heart rate zones are defined for different intensity levels. There are fat burning zones, muscle building zones and anerobic zones. Exercise intensity is adjusted to meet the desired zone.

The CDC defines zones as follows: Target Heart Rate is 64% to 76%. High Intensity target heart rate is 76-93% (CDC). Or 120 to 146 for the 62-year-old.

The Mayo Clinic defines moderate intensity as 50-70% of maximum heart rate and vigorous as 70% to 85%.

The Polar heart rate monitor company suggests 5 heart rate zones:

1. Very light at 50-60% of HRMAX. Boost recovery and prepare for higher zones.
2. Light 60-70%. Improve general endurance and increase capillary density.
2. Moderate at 70-80%. Lactic acid build-up, improve efficiency of blood circulation in the heart and muscles
4. Hard 80-90%. Improve speed endurance.
5. Maximum 90-100%. For elite athletes to further improve speed.

These zones will be compared against one another and serve as benchmarks for the clustering analysis.

Below is an example of the code to create zones (features) in the dataset. First, limits are defined in a dictionary:

```
mayo_limits =
{'moderate_low':0.50*max_hr,'moderate_hi':0.7*max_hr,'intensity_low':0.70*max_hr,'intensity_hi':0.85*max_hr}
cdc_limits =
{'moderate_low':0.64*max_hr,'moderate_hi':0.76*max_hr,'intensity_low':0.76*max_hr,'intensity_hi':0.93*max_hr}
polar_limits =
{'zone1':0.6*max_hr,'zone2':0.7*max_hr,'zone3':0.80*max_hr,'zone4':0.9*max_hr,'zone5':1*max_hr}
```

Next a method is defined for the zones:

```
def assignCDCZone(value):
    if (value > cdc_limits['moderate_low']) and (value <= cdc_limits['moderate_hi']):
        return "Moderate"
    elif (value > cdc_limits['intensity_low']) and (value <= cdc_limits['intensity_hi']):
        return  "Intensity"
    else:
        return 'Other'
```

And then the features are created with an assign function:

```
df_zones['Polar_Zone'] = df_zones.BPM.apply(assignPolarZone)
df_zones['CDC_Zone'] = df_zones.BPM.apply(assignCDCZone)
df_zones['Mayo_Zone'] = df_zones.BPM.apply(assignMayoZone)
```

After the zones are defined, the dataframe looks like this:

	BPM	Polar_Zone	CDC_Zone	Mayo_Zone
0	57	zone1	Other	Other
1	63	zone1	Other	Other
2	69	zone1	Other	Other
3	74	zone1	Other	Other
4	72	zone1	Other	Other

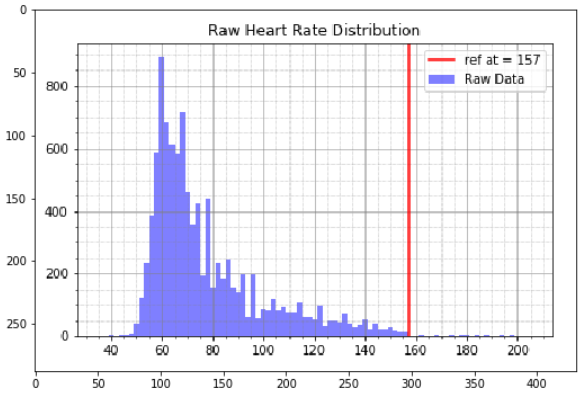
Benchmarking - Histogram Comparison

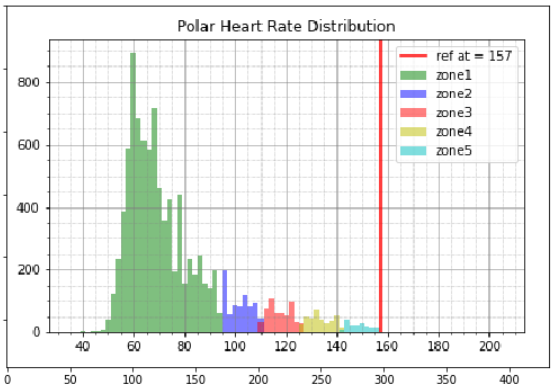
A snippet of code to generate the histograms is below. The full code is available in the Jupyter Notebook.

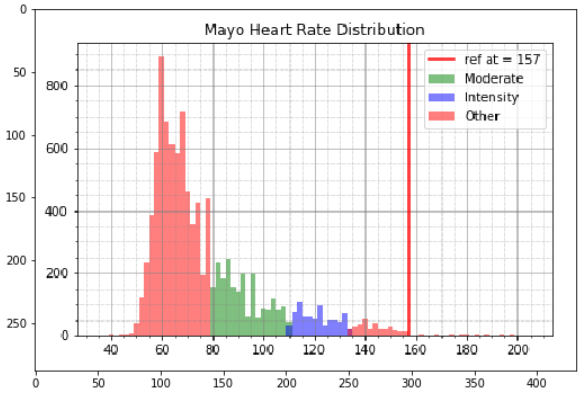
```
zone1 = df.loc[df.CDC_Zone == 'Moderate', 'BPM']
zone2 = df.loc[df.CDC_Zone == 'Intensity', 'BPM']
zone3 = df.loc[df.CDC_Zone == 'Other', 'BPM']

xref = [max_hr]
ref_colors=['r']
for ref, c in zip(xref,ref_colors):
  plt.axvline(x=ref, c=c, label= 'ref at = {}'.format(ref))

plt.hist(zone1, **kwargs, color='g', label='Moderate',bins=range(35, 205, 2))
plt.hist(zone2, **kwargs, color='b', label='Intensity',bins=range(35, 205, 2))
plt.hist(zone3, **kwargs, color='r', label='Other',bins=range(35, 205, 2))
plt.legend()
plt.savefig('./plots/cdc_histo.png')
```

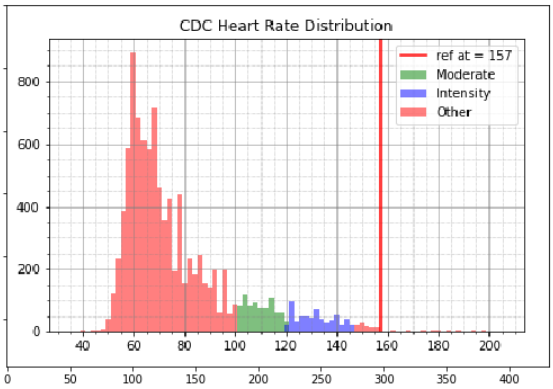
Below are 4 histograms. They show the heart rate data and how the zones differ. The histogram on the upper left is a reference histogram. The red line is draw at the calculated maximum hear rate at 158 BPM. It is an excellent reference, although there are data points going beyond this. The histogram on the upper right is for the 5 zones recommended by the Polar ™ sensor company. The distribution looks continuous with a tail on the high end. The zones have been superimposed on the raw distribution. The high intensity zone is zone 5, which is 90-100% of the maximum heart rate.

The bottom two graphs are for the Mayo Clinic and the CDC target zones. They only define 2 zones, moderate and intense. The CDC and Mayo limits are more conservative than the Polar zones. The Polar zones are more appropriate for athletes as it is common to achieve the calculated maximum heart rate indicated by the red reference line.









Data Exploration Summary

The dataset has been loaded, columns of interest selected, filtered and a smaller sample generated to speed execution. Summary statistics and graphical analysis have also been completed. This is a simple 1D dataset with no features or labels.

The most significant take-away, highlighted by the distribution, is that there does not appear to be any clusters. The is no clear separation in any of the data. However, the analysis will continue. The models, such as a Gaussian Mixture Model, may be able to tease apart at least a second distribution.

The maximum heart rate calculation agrees well with the data.

Training and Test Datasets

Note also that training and test datasets have not been generated. This is because it is un-supervised learning. There are no labels for model predictions to be compared against. Instead a 'score', explained more below, will be used to evaluate the model with the best fit.

3. Model Implementation

The next step is to build and evaluate models that identify target heart rate zones. Since the data is not labeled, unsupervised clustering techniques will be utilized. Three models are evaluated: Kmeans, GMM and DBSCAN.

Kmeans

Kmeans identifies clusters within a dataset. It works as follows. At first, the centroids of the clusters are located at random. Next the instances are labeled and the square of the distance from each instance to the centroid is calculated. Then the centroids are updated, the instances re-labeled, and the calculation is redone. This continues until the centroids stop moving. Convergence can be improved by increasing the number of initializations and selecting the best one.

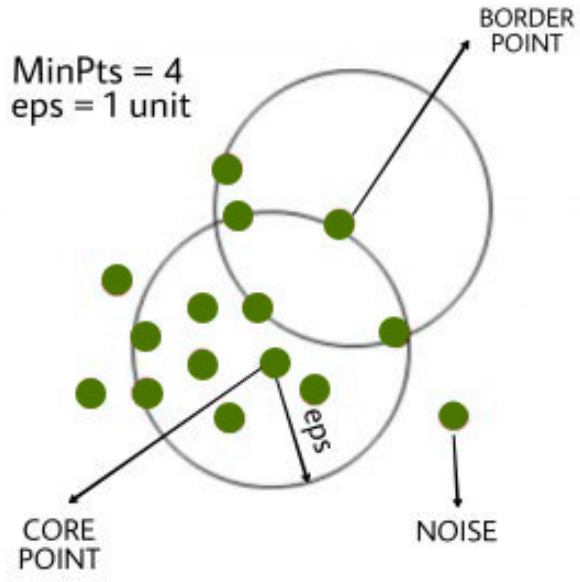
Limitations of Kmeans are that it works best with cluster of similar size and density. It also assigns instances to clusters even if they are outliers. Furthermore, one needs to know the number of clusters ahead of time.

Gaussian Mixture Model

Another problem with Kmeans is that every instance is assigned to a cluster and there is no strength associated with that membership. That is, there is no probability of how strongly that data point is associated with a given cluster. This is an issue that the Gaussian Mixture Model, or GMM, addresses. The model assumes that each cluster is gaussian and assigns a probability to each data point for belonging to that cluster. Unlike Kmeans, the sizes of the cluster are not assumed to be the same. Each cluster, or distribution, has its own mean, covariance and weight.

DBSCAN

Density-based spatial clustering of applications with noise, or DBSCAN, is a clustering technique that does not rely on specifying the number of clusters ahead of time. Instead, a characteristic distance is defined, *eps,* along with a number of minimum points. If points are within the *eps*, it is a core instance and considered to be within the cluster neighborhood. If the neighborhood achieves the minimum number of points, it is considered to be a cluster. Any instance that is not a core instance is considered an anomaly. These anomalies are labeled with a [-1]. Once the data has been fit to the model, the labels for the clusters are available with the dbscan.labels_ parameter.



https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/

Performance Metrics

For Kmeans two performance metrics were used. The first is the **inertia score**. It is the mean squared distance from an instance to the closest centroid. Lower numbers are better. However, a limitation with this method is that it assumes similar cluster sizes. It also is dependent on the number of clusters. As k increases the score decreases as there are more clusters closer to an instance. However, this was chosen as a starting point for selecting the number of clusters.

The second and more precise metric is the **silhouette score**. This metric could be used, not only for Kmeans, but as a yardstick for the other models. The silhouette score is equal to:

$$(b-a)/max(a,b)$$

where *a* is the mean intra-cluster distance and *b* is the mean nearest cluster distance – the inter cluster distance. It comprehends both within cluster variation, or cohesiveness, and cluster to cluster variation, or separation.

For GMM, the Bayes Information Criterion (**BIC**), which does allow for unequal and non-spherical clusters, was calculated. BIC is a maximum likelihood technique that will find a model with the fewest number of parameters that still does a good job of explaining the data. Unfortunately, Kmeans and DBSCAN do not have a built-in BIC method.

Hyperparameters

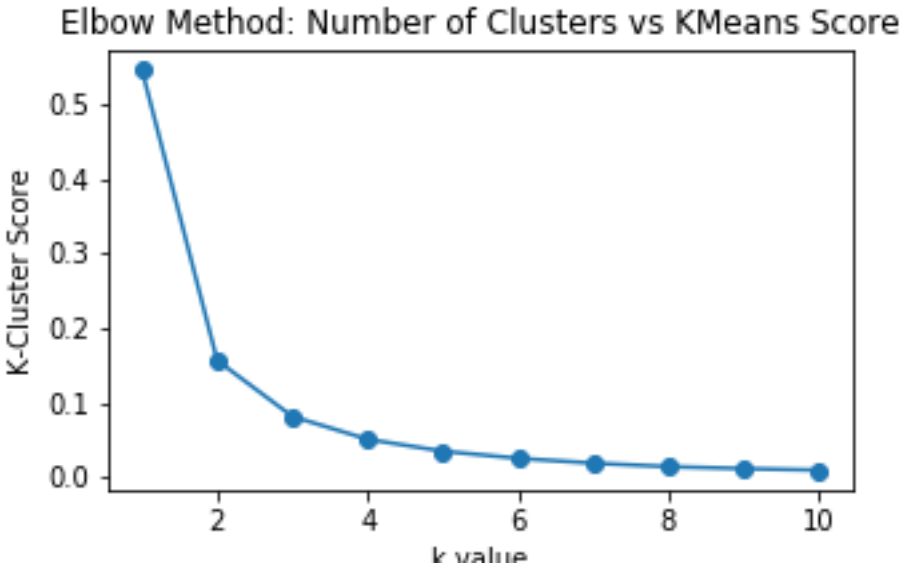
With the models chosen and the performance metrics defined, the next step is to define the hyperparameters.

For KMeans and GMM, the hyperparameter of interest is the number of clusters. For DBSCAN, *eps* and minimum number of samples, are the hyperparameters of interest. Each parameter was skewed and graphically analyzed to determine the best values. Results are below.

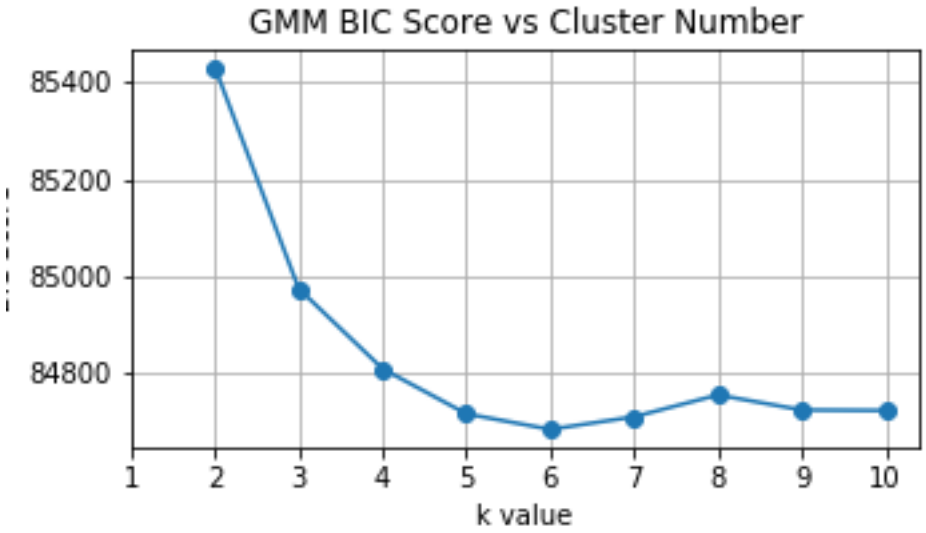
4. Model Results

Hyper-Parameter Tuning

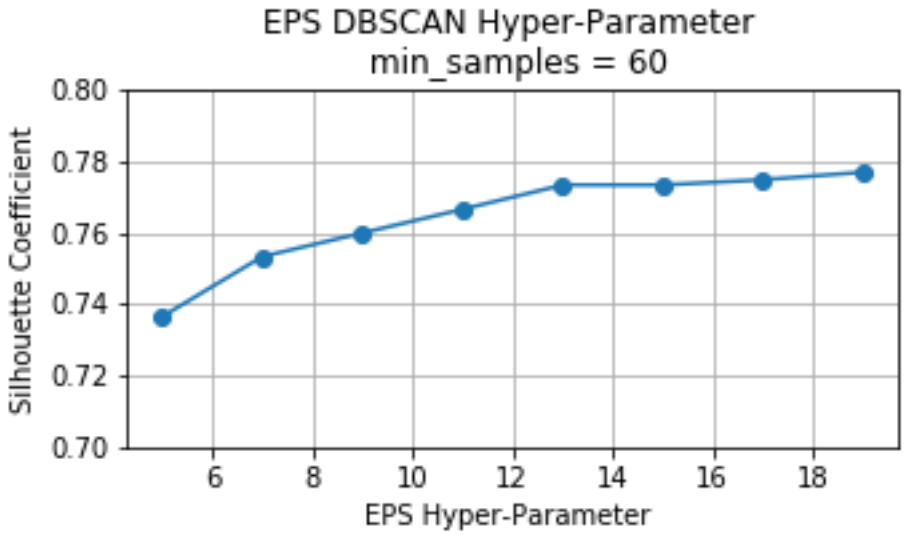
The graphs below are used to select the optimum hyperparameters. The performance metrics used include the inertia score, silhouette score and BIC.



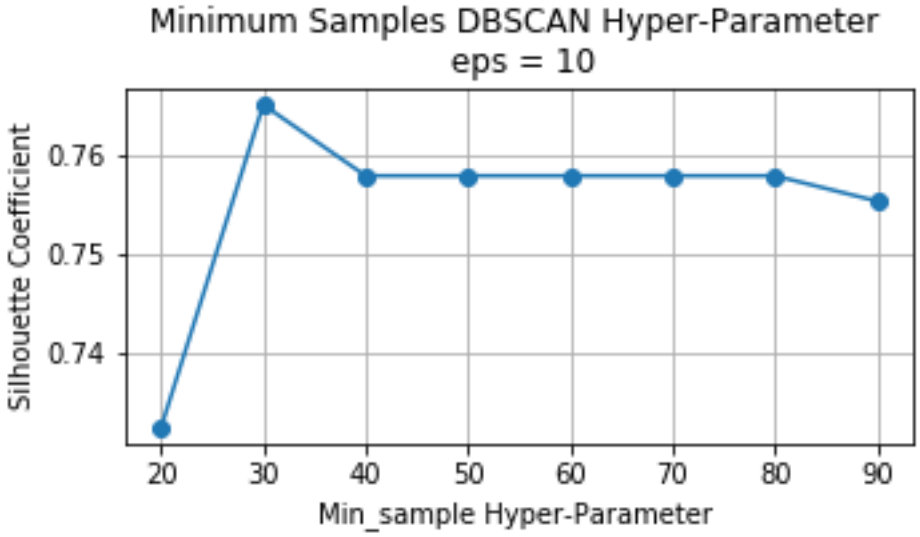
The plot above is for Kmeans and shows that for values > 5, the improvement in the inertia score are marginal.



The graph above for GMM also shows that for k values greater than 5 there is marginal benefits in the BIC score.

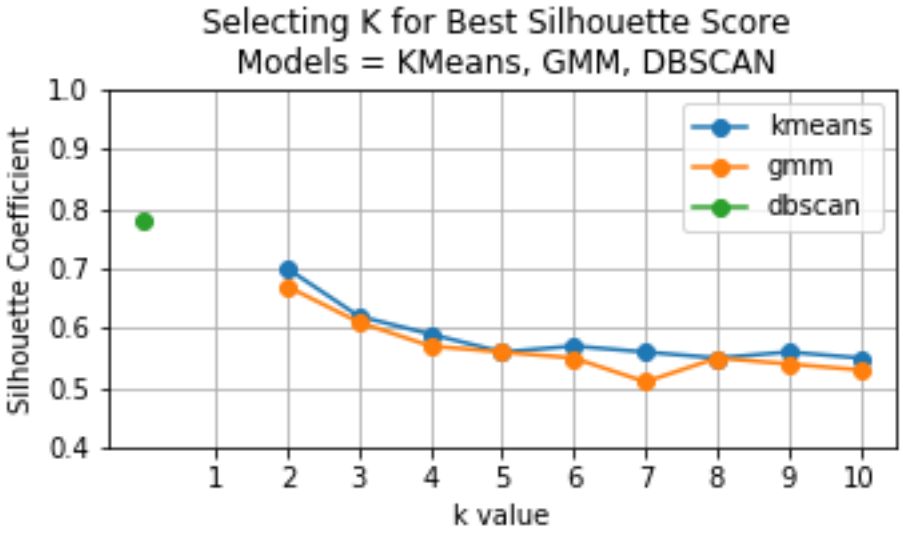


The plot above is for the DBSCAN hyperparameter EPS. A value of 10 was chosen.



And lastly the plot above for DBSCAN, 60 was chosen as an optimum sample size.

Now let's look at all three models on one plot to see how they compare:



From the graph above, DBSCAN has the highest silhouette score at close to 0.8. Since the number of clusters is not a hyperparameter for DBSCAN, only one data point was plotted. GMM and Kmeans had similar performance.

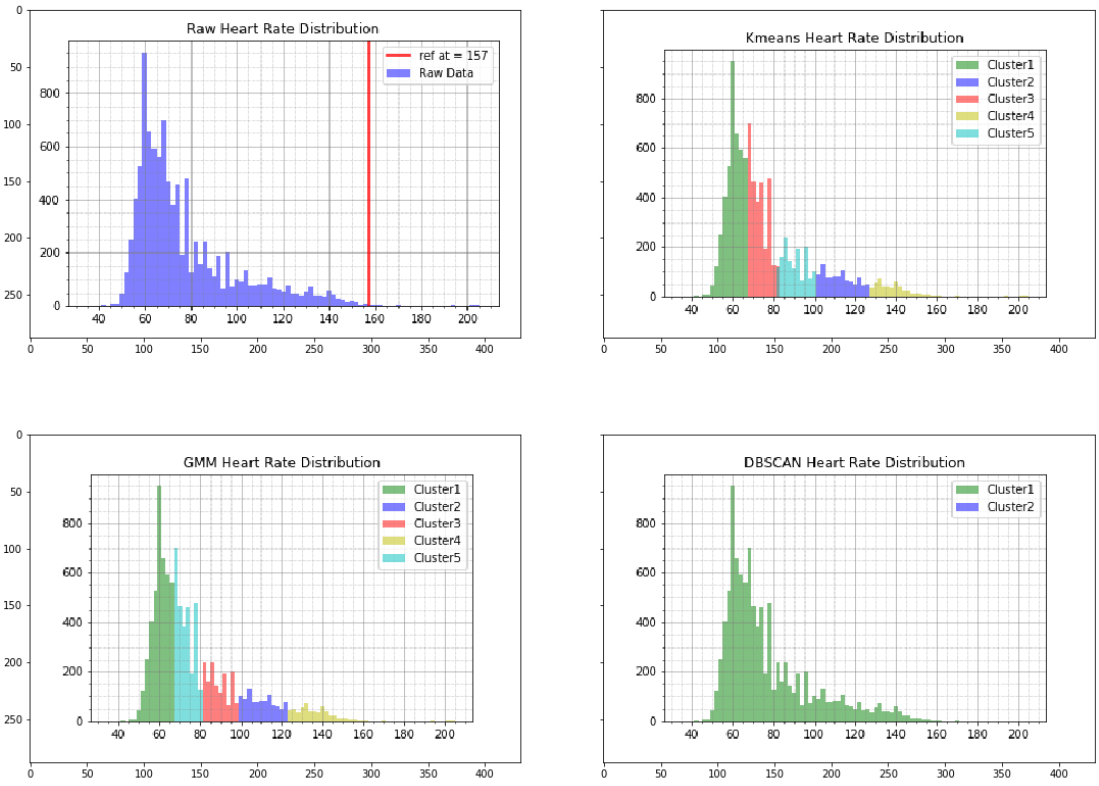
The code to generate the silhouette scores above:

```
def silhouetteScore(df,num_clusters):
    models = ['kmeans', 'gmm', 'dbscan']
    sil_scores = []
    k_value = []
%%time
def silhouetteScore(df,num_clusters):
    models = ['kmeans', 'gmm', 'dbscan']
    sil_scores = []
    k_value = []
    model_type = []
    bic_values = []
    dbscan_cluster_labels = []
    range_n_clusters = list(range(2,num_clusters))
    for n_clusters in range_n_clusters:
        for mymodel in models:
            if mymodel == 'kmeans':
                k_value.append(n_clusters)
                model_type.append(mymodel)
                kmeans = KMeans(n_clusters=n_clusters)
                cluster_label = kmeans.fit_predict(df)
                score = round(silhouette_score(df,cluster_label),2)
                sil_scores.append(score)
                bic_values.append(None)
                print("Model {} with n_clusters = {}, silhouette score is {}".format(mymodel,n_clusters, score))
                print("Model {} with n_clusters = {}, BIC is {}".format(mymodel,n_clusters, None))
```

```
            elif mymodel == 'gmm':
                k_value.append(n_clusters)
                model_type.append(mymodel)
                gmm = mixture.GaussianMixture(n_components=n_clusters,n_init=10)
                gmm.fit(df)
                bic_values.append(round(gmm.bic(df),0))
                cluster_label = gmm.fit_predict(df)
                score = round(silhouette_score(df,cluster_label),2)
                sil_scores.append(score)
                print("Model {} with n_clusters = {}, silhouette score is {}".format(mymodel,n_clusters, score))
                print("Model {} with n_clusters = {}, BIC is {}".format(mymodel,n_clusters,
round(gmm.bic(df),0)))

            elif mymodel == 'dbscan':
                k_value.append(n_clusters)
                model_type.append(mymodel)
                dbscan = DBSCAN(eps=20, min_samples= 100)
                cluster_label = dbscan.fit_predict(df)
                dbscan_cluster_labels.append(cluster_label)
                score = round(silhouette_score(df,cluster_label),2)
                sil_scores.append(score)
                bic_values.append(None)
                print("Model {} with n_clusters = {}, BIC is {}".format(mymodel,n_clusters, None))

    print ("\nFinished!")
    return sil_scores, bic_values,k_value, model_type,dbscan_cluster_labels
```

Graphical Analysis

For each of the 3 models, the histogram was partitioned by the clusters. The figure below shows the results.



The upper left is the raw data with a reference line at the maximum heart rate. The upper right shows the Kmeans result, the lower left is for GMM and the lower right is DBSCAN. Again, the Kmeans and GMM look similar. Below is the code and output to display the descriptive statistics for each model and cluster:

```
display (df2.groupby('Kmeans_Cluster')['BPM'].describe())
display (df2.groupby('GMM_Cluster')['BPM'].describe())
display (df2.groupby('DBSCAN_Cluster')['BPM'].describe())
```

Model Cluster Statistics

Kmeans_Cluster	count	mean	std	min	25%	50%	75%	max
0	4139	59.7195	3.994418	40	57	60	63	66
1	993	111.9043	7.28979	101	105	112	118	126
2	2922	72.54073	4.215346	67	69	72	76	81
3	539	141.0649	15.09664	127	131	138	144	209
4	1407	89.74982	5.45458	82	85	89	95	100

GMM_Cluster	count	mean	std	min	25%	50%	75%	max
0	4139	59.7195	3.994418	40	57	60	63	66
1	998	109.4429	6.873134	99	104	109	115	122
2	1428	88.31373	5.202922	81	83	87	92	98
3	635	138.5496	15.13823	123	129	135	142	209
4	2800	72.17214	3.910075	67	69	72	76	80

DBSCAN_Cluster	count	mean	std	min	25%	50%	75%	max
-1	22	198.1818	6.46335	186	193.25	200.5	203	209
0	9978	76.99098	22.2876	40	61	69	86	185

5. Discussion/Conclusions

DBSCAN achieved the best silhouette coefficient while Kmeans and GMM were comparable. The clusters identified by Kmeans and GMM did not have any separation, suggesting overfitting. However, this may not be the classic definition of over-fitting as any new data would certainly fall into one of the cluster definitions. It can be said that the clusters do not provide any additional insight over the Polar zones.

With DBSCAN, a [-1] represents an anomaly or noise. Consequently, from the summary statistics above, it can be seen that there is only one cluster identified [0]. However, DBSCAN does an excellent job of identifying these anomalies. From the table above, the anomalies have a range of 188 to 215 - above the predicted maximum heart rate of 158. These anomalies could be a result of a faulty reading of the heart rate sensor. Providing a method to exclude this data would be of value to algorithms embedded in software or hardware. They could be the result a loose or improperly located wrist band.

The zones or clusters defined by the CDC and Mayo clinic seem to be insufficient for high intensity training. They both only define 2 zones, moderate and intense. Their upper limits (for a 62-year-old), 133 and 146, respectively, are not close enough to the predicted maximum heart of 158. This heart rate is frequently obtained and to guard so heavily against it may prohibit athletes from achieving their full potential.

Lastly, the predicted maximum heart rate (220-age) compared extremely well with the data. Although it is not a hard limit. There are what appear to be valid, although few, readings above this limit. For example, a reading of 166 BPM is well below the hardware limit, and the anomaly range of 186-210 identified by DBSCAN, and nominally above the max_hr limit base on age of 158. It is likely a valid reading.

References:

"Scared of High Intensity Interval Training? A Hear Monitor Can Make it Fun and Easy". New York Times, 5/29/18.

"Really, Really Short Workouts". New York Times, 3/15/19.

"Preventing Muscle Loss as We Age". New York Times. 9/22/18.

"Feel the beat of Heart Rate Training". Harvard Health Publishing. 12/2017.

"What is Optical Heart Rate Tracking". Polar. 8/30/17.

"Polar H10 Heart Rate Sensor System", Polar. 11/11/19.

"How Apple Watch Measures Your Heart Rate". Apple.3/24/2020.

"Target Heart Rate and Maximum Heart Rate". CDC.

"Intensity Exercise: How to measure it". Mayo Clinic.

"Gaussian Mixture Model". YouTube Video.

"Gaussian Mixture Models Explained". Toward Data Science.

"Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow". Aurelien Geron. O'Reilly Press.

"An Intuitive Explanation of the Bayesian Information Criterion". Towards Data Science.