CS498 AML HW5 REPORT (Fall 2018 UIUC)

PENGYU CHENG *pcheng11*
CAITING WU *cwu72*
HAOTIAN QIU *hqiu9*

For this assignment, we first provide several basic points:

- **We used <u>Hierarchical K-means</u>**
- **We split train-test set <u>only once</u> BEFORE we did any analysis. (Train set 70%. Test set 30%)**

- **Tables:**

Overlap	Segment Siz	Overlap Rati	K Value	Classifier	Accuracy
FALSE	48	N/A	40	rf	0.7857143
FALSE	96	N/A	40	rf	0.7321429
FALSE	192	N/A	40	rf	0.7678571
FALSE	48	N/A	120	rf	0.7440476
FALSE	96	N/A	120	rf	0.7678571
FALSE	192	N/A	120	rf	0.7559524
FALSE	48	N/A	240	rf	0.7738095
FALSE	192	N/A	240	rf	0.7083333
FALSE	48	N/A	480	rf	0.7321429
FALSE	96	N/A	480	rf	0.7440476
FALSE	48	N/A	1000	rf	0.6845238
FALSE	48	N/A	40	svm	0.7559524
FALSE	96	N/A	40	svm	0.7321429
FALSE	192	N/A	40	svm	0.7142857
FALSE	48	N/A	120	svm	0.7202381
FALSE	96	N/A	120	svm	0.6904762
FALSE	192	N/A	120	svm	0.7380952
FALSE	48	N/A	240	svm	0.702381
FALSE	192	N/A	240	svm	0.7261905
FALSE	48	N/A	480	svm	0.7083333
FALSE	96	N/A	480	svm	0.75
FALSE	48	N/A	1000	svm	0.7261905
TRUE	48	0.2	40	rf	0.7857143
TRUE	48	0.5	40	rf	0.7916667
TRUE	48	0.8	40	rf	0.7857143
TRUE	96	0.2	40	rf	0.7380952
TRUE	96	0.5	40	rf	0.8095238
TRUE	96	0.8	40	rf	0.797619
TRUE	192	0.2	40	rf	0.7738095
TRUE	192	0.5	40	rf	0.7857143
TRUE	192	0.8	40	rf	0.7916667
TRUE	48	0.2	120	rf	0.7440476
TRUE	48	0.5	120	rf	0.8095238
TRUE	48	0.8	120	rf	0.7797619
TRUE	96	0.2	120	rf	0.7142857
TRUE	96	0.5	120	rf	0.827381
TRUE	96	0.8	120	rf	0.8035714
TRUE	192	0.2	120	rf	0.75
TRUE	192	0.5	120	rf	0.797619
TRUE	192	0.8	120	rf	0.7916667
TRUE	48	0.2	240	rf	0.797619
TRUE	48	0.5	240	rf	0.8035714
TRUE	48	0.8	240	rf	0.7797619
TRUE	96	0.2	240	rf	0.7321429
TRUE	96	0.5	240	rf	0.7559524
TRUE	96	0.8	240	rf	0.8214286
TRUE	192	0.2	240	rf	0.6666667
TRUE	192	0.5	240	rf	0.8035714
TRUE	192	0.8	240	rf	0.8333333
TRUE	48	0.2	480	rf	0.75
TRUE	48	0.5	480	rf	0.7857143
TRUE	48	0.8	480	rf	0.7797619
TRUE	96	0.2	480	rf	0.7321429
TRUE	96	0.5	480	rf	0.8035714
TRUE	96	0.8	480	rf	0.7916667
TRUE	192	0.2	480	rf	0.7321429
TRUE	192	0.5	480	rf	0.7678571
TRUE	192	0.8	480	rf	0.8035714
TRUE	48	0.2	1000	rf	0.7321429
TRUE	48	0.5	1000	rf	0.7678571
TRUE	48	0.8	1000	rf	0.8035714
TRUE	96	0.2	1000	rf	0.6785714
TRUE	96	0.5	1000	rf	0.7857143
TRUE	96	0.8	1000	rf	0.7797619
TRUE	192	0.2	1000	rf	0.7083333
TRUE	192	0.8	1000	rf	0.8095238
TRUE	48	0.2	40	svm	0.7083333
TRUE	48	0.5	40	svm	0.7440476
TRUE	48	0.8	40	svm	0.5238095
TRUE	96	0.2	40	svm	0.6309524
TRUE	96	0.5	40	svm	0.7559524
TRUE	96	0.8	40	svm	0.4761905
TRUE	192	0.2	40	svm	0.7261905
TRUE	192	0.5	40	svm	0.7797619
TRUE	192	0.8	40	svm	0.8214286
TRUE	48	0.2	120	svm	0.702381
TRUE	48	0.5	120	svm	0.7797619
TRUE	48	0.8	120	svm	0.672619
TRUE	96	0.2	120	svm	0.7380952
TRUE	96	0.5	120	svm	0.7857143
TRUE	96	0.8	120	svm	0.7142857
TRUE	192	0.2	120	svm	0.7261905
TRUE	192	0.5	120	svm	0.7857143
TRUE	192	0.8	120	svm	0.8035714
TRUE	48	0.2	240	svm	0.6845238
TRUE	48	0.5	240	svm	0.7916667
TRUE	48	0.8	240	svm	0.6845238
TRUE	96	0.2	240	svm	0.6785714
TRUE	96	0.5	240	svm	0.7678571
TRUE	96	0.8	240	svm	0.7380952
TRUE	192	0.2	240	svm	0.6666667
TRUE	192	0.5	240	svm	0.7857143
TRUE	192	0.8	240	svm	0.827381
TRUE	48	0.2	480	svm	0.75
TRUE	48	0.5	480	svm	0.7619048
TRUE	48	0.8	480	svm	0.7261905
TRUE	96	0.2	480	svm	0.6904762
TRUE	96	0.5	480	svm	0.7559524
TRUE	96	0.8	480	svm	0.7678571
TRUE	192	0.2	480	svm	0.7797619
TRUE	192	0.5	480	svm	0.8333333
TRUE	192	0.8	480	svm	0.8154762
TRUE	48	0.2	1000	svm	0.7619048
TRUE	48	0.5	1000	svm	0.797619
TRUE	48	0.8	1000	svm	0.7678571
TRUE	96	0.2	1000	svm	0.7559524
TRUE	96	0.5	1000	svm	0.7857143
TRUE	96	0.8	1000	svm	0.7916667
TRUE	192	0.2	1000	svm	0.7321429
TRUE	192	0.8	1000	svm	0.8571429

CS498 AML HW5 REPORT (Fall 2018 UIUC)

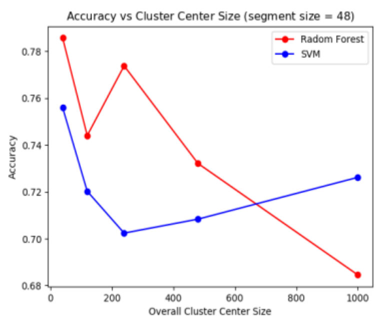
PENGYU CHENG *pcheng11*
CAITING WU *cwu72*
HAOTIAN QIU *hqiu9*

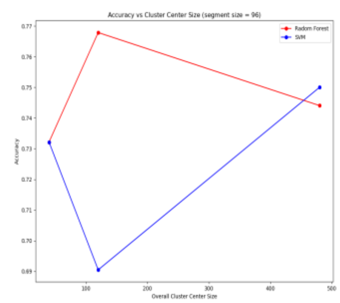
<u>In-depth analysis and experimentation:</u>

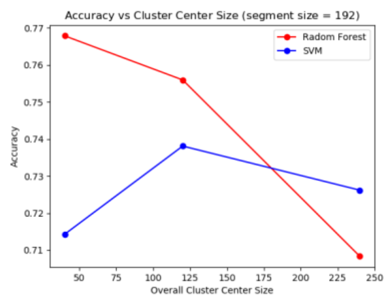
- For the analysis and experimentation, we tried some combinations of segment size, overlap percent, whether we use overlap, type of classifiers and number of clusters(k).
- We provide both the tables and plots we derived from the experimentation.
- The intuition of the *<u>choice of the segment size</u>* is the following:
    1. Size 192 corresponds to 2s of the activity, as we flatten the data as *(x1y1z1)(x2y2z2)…(xnynzn)*
    2. Size 96 corresponds to 1s of the activity
    3. Size 48 corresponds to 0.5s of the activity

(Note the range of x axis is not the same regardless of the choice of overlapping or not as with the increasing size of segments, we will get a relatively less segments for clustering.)

- **<u>Non-overlapping:</u>**

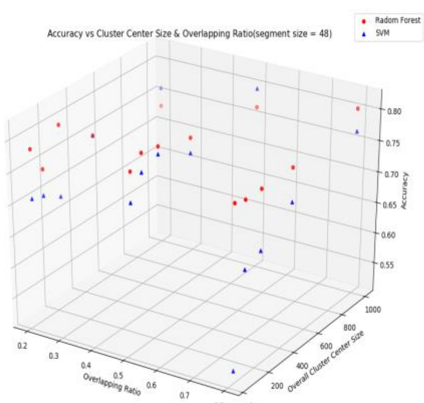


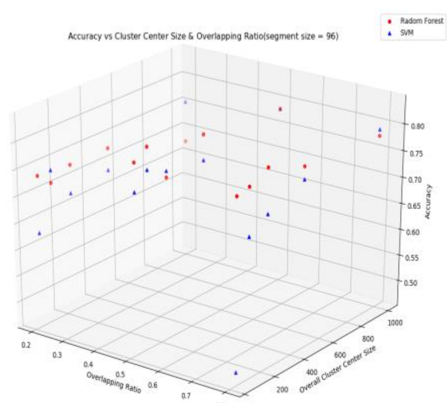




<u>(As we posted on piazza</u>, if the clusters in the first layer are too many, we cannot get enough segments to be clustered for the second layer. Therefore, we only have 250 clusters for the third plot.)

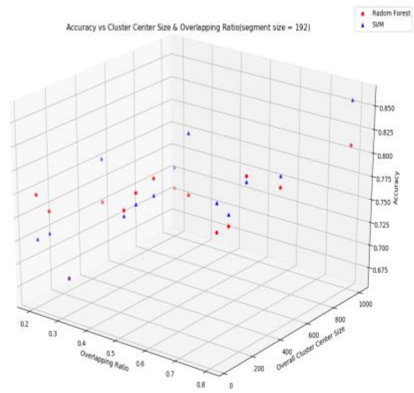
- **<u>Overlapping:</u>**





CS498 AML HW5 REPORT (Fall 2018 UIUC)

PENGYU CHENG *pcheng11*
CAITING WU *cwu72*
HAOTIAN QIU *hqiu9*



Based on the above diagrams, we have the following observations:

- When the segment size is small, overlapping method does not tend to offer better accuracy than non-overlapping method does.

- Under the non-overlapping methods, changing the segment size does not offer significant improvement on accuracy.

  Under both non-overlapping method and overlapping method, when the segment size is relatively small (48 or 96 in our case), the results yielded by using SVM and the one generated by Random Forest do not tend to have significant difference. Nevertheless, with smaller segment size, Random Forest produce slightly better results than SVM does.

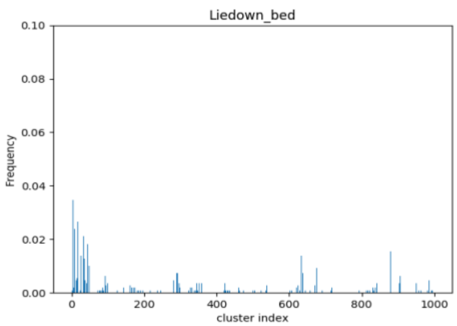
- Under both non-overlapping method and overlapping method, the SVM performs better than Random Forest when the segment size is large (e.g. when size is 192). We postulate what causes such divergence is that SVM tend to perform better when the input feature is large. Because we are swapping the original 3D data recorded by an accelerator with a column vector filled with cluster center numbers, the bigger the segment size is, the more feature we feed into the classifiers.

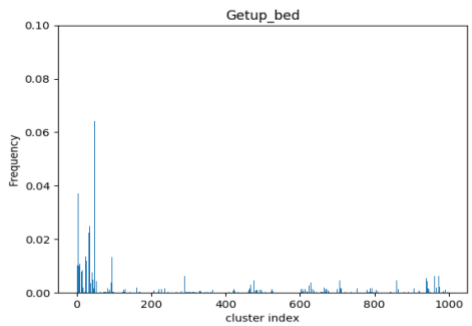
CS498 AML HW5 REPORT (Fall 2018 UIUC)

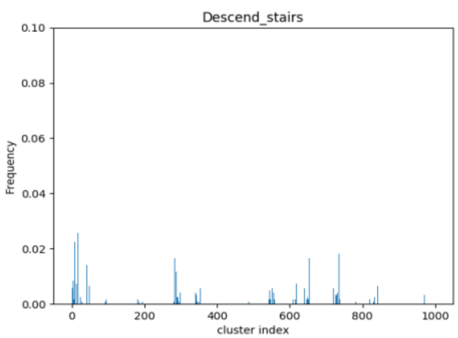
PENGYU CHENG *pcheng11*
CAITING WU *cwu72*
HAOTIAN QIU *hqiu9*

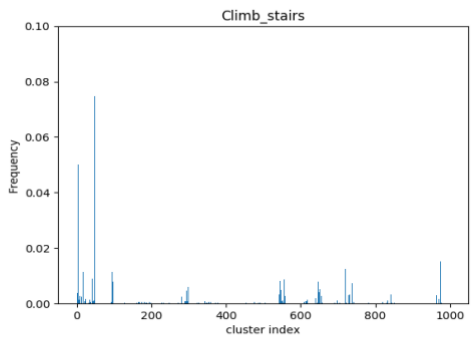
The following plot is based on the hyper-parameters we chose on page 1 (in-depth analysis)

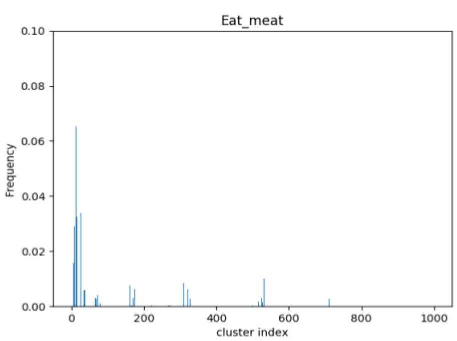
- **We get the best result when use:**
  - **Segment size = 192, Overlap percent = 80%, Number of clusters (k) = 1000, Type of classifiers = SVM**
- **Histograms (Normalized)**

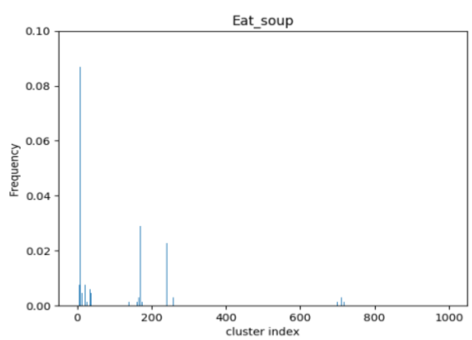


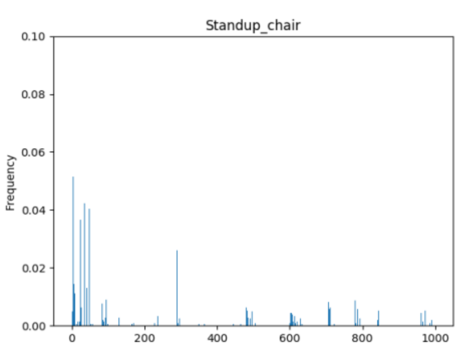


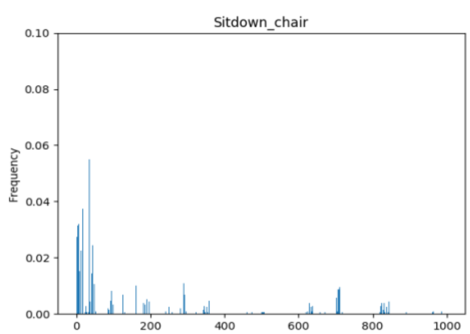










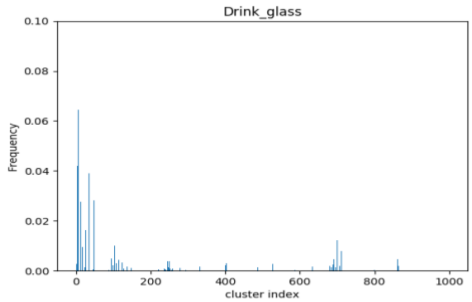


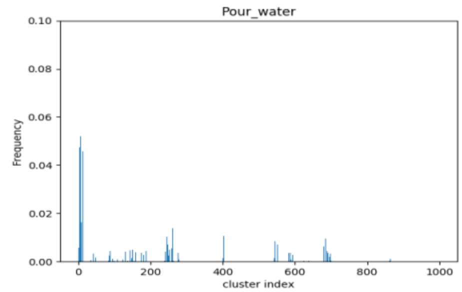
CS498 AML HW5 REPORT (Fall 2018 UIUC)

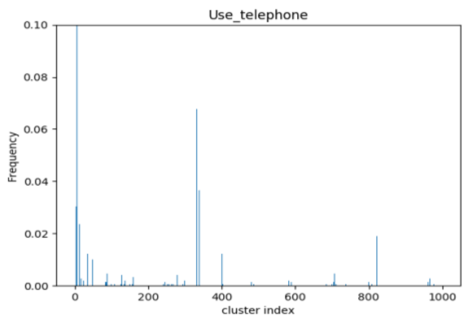
PENGYU CHENG *pcheng11*

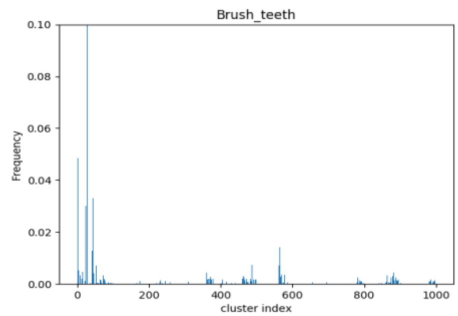
CAITING WU *cwu72*

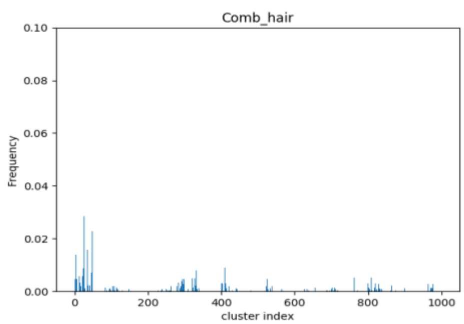
HAOTIAN QIU *hqiu9*

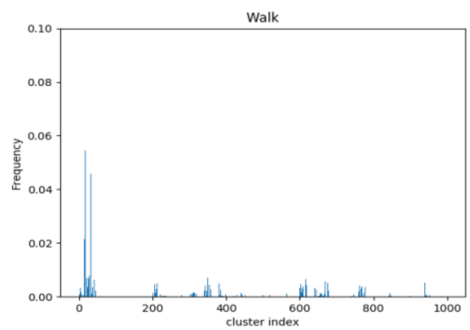




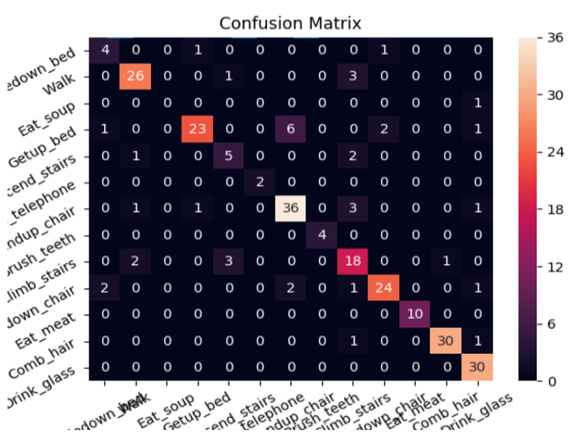








- **Confusion Matrix**
    - **(X axis is the predicted label and Y axis is the true label)**



From the confusion matrix, it is not hard to find that some similar activities result in some non-accuracy. For example, we predict one laydown bed activity to be get up bed as they are similar and two descend stairs activities to be climb stairs.

CS498 AML HW5 REPORT (Fall 2018 UIUC)

PENGYU CHENG *pcheng11*
CAITING WU *cwu72*
HAOTIAN QIU *hqiu9*

1. **segmentation of the vector**
   - **Two parts: overlap and non-overlap**

```
def cut_segments(data, segment_size, overlap_percent, overlap=False):
    reconstruct_data = []
    segments = []
    if overlap:
        for signal in data:
            temp = []
            for start in range(0, len(signal), segment_size-int(overlap_percent*segment_size)):
                segment = signal[:-1][start:start+segment_size]
                if len(segment) == segment_size:
                    temp.append(segment)
                    segments.append(segment)
            temp.append(signal[-1])
            reconstruct_data.append(temp)
    else:
        for signal in data:
            temp = []
            for start in range(0, len(signal), segment_size):
                segment = signal[:-1][start:start+segment_size]
                if len(segment) == segment_size:
                    temp.append(segment)
                    segments.append(segment)
            temp.append(signal[-1])
            reconstruct_data.append(temp)

    print('Total segments number:{}'.format(len(segments)))
    return segments, reconstruct_data
```

2. **k-means (We used hierarchical k-means)**

```
def hierarchical_kmeans(segments, fst_level_cluster_num, snd_level_cluster_num, segment_size):
    temp_dic = {}
    dic = {}
    fst_kmeans = KMeans(n_clusters=fst_level_cluster_num, random_state=0).fit(segments)
    first_level = zip(fst_kmeans.labels_, segments)
    print('Building first level hierarchical kmeans...')
    for center, segment in tqdm(list(first_level)):
        if center in temp_dic:
            temp_dic[center].append(segment)
        else:
            temp_dic[center] = [segment]
    print('Building second level hierarchical kmeans...')
    for center in tqdm(range(fst_level_cluster_num)):
        snd_kmeans = KMeans(n_clusters=snd_level_cluster_num, random_state=0).fit(temp_dic[center])
        dic[center] = snd_kmeans
    return fst_kmeans, dic
```

3. **generating the histogram (quantization) and plot histogram**

```
def quantization(reconstruct_data, fst_kmeans, dic, fst_level_cluster_num, snd_level_cluster_num):
    new_data = []
    dimension = fst_level_cluster_num * snd_level_cluster_num
    print("Vector Quantization...")
    for sample in tqdm(reconstruct_data):
        new_sample = np.zeros(dimension+1) #+1 for label
        for segment in sample[:-1]:
            fst_center = fst_kmeans.predict([segment])[0]
            snd_center = dic[fst_center].predict([segment])[0]
            new_sample[fst_center*snd_level_cluster_num+snd_center] += 1
            new_sample[fst_center] += 1
        new_sample[-1] = sample[-1]
        new_data.append(new_sample)
    return np.array(new_data)
```

```
def plot(classes, class_names, new_train_data, fst_level_cluster_num, snd_level_cluster_num):
    print('Saving figures...')
    for num in range(classes):
        sub_data = new_train_data[np.where(new_train_data[:,-1] == num+1)]
        plot_hist(np.mean(sub_data, axis=0)[:-1], class_names[num], fst_level_cluster_num*snd_level_cluster_num)

def plot_hist(data, name, dimension):
    plt.figure()
    plt.bar(range(dimension), data/sum(data))
    plt.ylim(top=0.2)
    plt.ylabel('Frequency')
    plt.xlabel('cluster index')
    plt.title(name[12:])
    plt.savefig('/Users/pengyucheng/Desktop/cs498_aml/cs498_aml_hw5/report/' + name[12:] + '.png')
```

CS498 AML HW5 REPORT (Fall 2018 UIUC)

PENGYU CHENG *pcheng11*
CAITING WU *cwu72*
HAOTIAN QIU *hqiu9*

4. **classification (We tried SVM and Random Forest Classifier from scikit-learn)**

```
def classify_predict(train_data, test_data, class_names, classifier):
    if classifier == 'svm':
        clf = svm.LinearSVC()
    if classifier == 'rf':
        clf = RandomForestClassifier()
    clf.fit(train_data[:,:-1], train_data[:,-1])
    y_pred = clf.predict(test_data[:,:-1])
    cnf_matrix = confusion_matrix(test_data[:,-1], y_pred)
    accuracy = accuracy_score(test_data[:,-1], y_pred)
    plot_confusion_matrix(cnf_matrix, classes=class_names, title='Confusion Matrix')
    return accuracy
```

Relevant code snippet:

- **Experimentation:**

```
def experiement(fst_level_cluster_num, snd_level_cluster_num, \
                segment_size, overlap, overlap_size, train_data, \
                test_data, clf, class_names):
    train_segments, reconstruct_train_data = cut_segments(train_data, segment_size, overlap_size, overlap=overlap)
    fst_kmeans, dic = hierarchical_kmeans(train_segments, fst_level_cluster_num, snd_level_cluster_num, 96)
    new_train_data = quantization(reconstruct_train_data, fst_kmeans, dic, fst_level_cluster_num, snd_level_cluster_num)
    test_segments, reconstruct_test_data = cut_segments(test_data, segment_size, overlap_size, overlap=overlap)
    new_test_data = quantization(reconstruct_test_data, fst_kmeans, dic, fst_level_cluster_num, snd_level_cluster_num)

    return classify_predict(new_train_data, new_test_data, class_names, clf)
```

- **Main driver (contains train test split)**

```
def main():
    classes = 14

    train_data, class_names = read_data("HMP_Dataset")
    print("Split dataset before quantization...")
    train_data, test_data = train_test_split(train_data, test_size=0.2)

    segment_sizes = [24, 48, 96, 192]
    overlap_percents = [0.2, 0.5, 0.8]
    cluster_centers = [(10,4), (20,6), (30, 8), (40, 12), (50, 20)]

    for overlap in [False, True]:
        for clf in ["rf", "svm"]:
            for k in cluster_centers:
                for segment_size in segment_sizes:
                    print("classifier:{0}".format(clf))
                    if overlap:
                        for overlap_percent in overlap_percents:
                            try:
                                accuracy = experiement(k[0], k[1], segment_size, overlap, overlap_percent, train_data, test_data, clf, class_nam
                                storing_info = [overlap, clf, k[0], k[1], segment_size, overlap_percent, accuracy]
                            except:
                                continue
                            with open('analysis.json', 'a+') as f:
                                jasonfile = json.dumps(storing_info)
                                f.write(jasonfile)
                                f.write('\n')

                    else:
                        try:
                            accuracy = experiement(k[0], k[1], segment_size, overlap, 0.1, train_data, test_data, clf, class_names)
                            storing_info = [overlap, clf, k[0], k[1], segment_size, 'N/A', accuracy]
                        except:
                            continue
                        with open('analysis.json', 'a+') as f:
                            jasonfile = json.dumps(storing_info)
                            f.write(jasonfile)
                            f.write('\n')
```