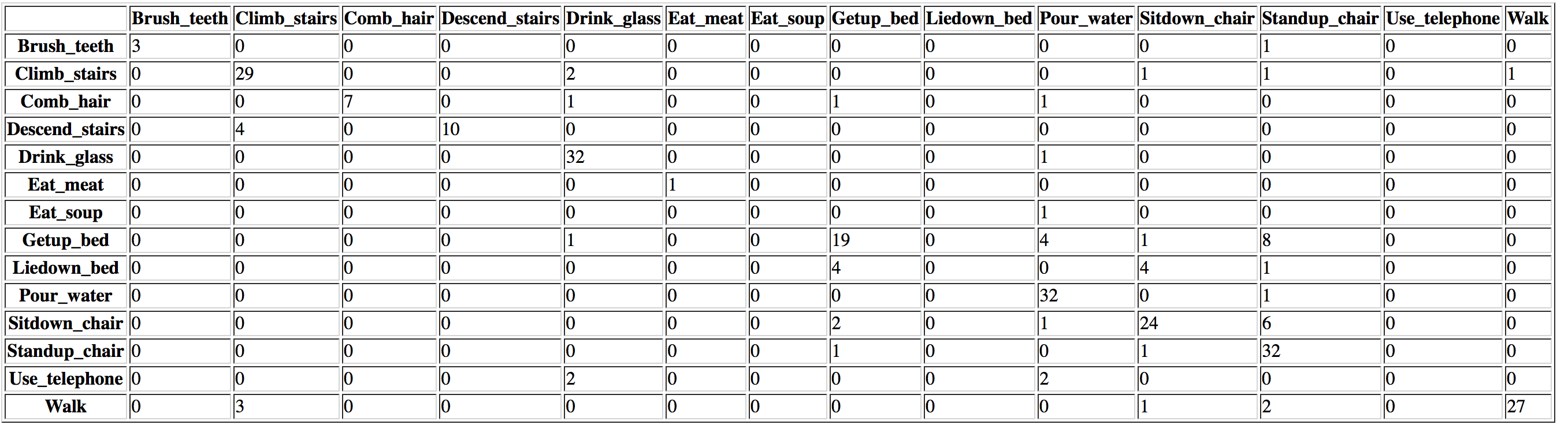
# Experiment Table

|  |  |  |  |
| --- | --- | --- | --- |
| Fixed Sample Length | Overlap % | K-Value (Number of Cluster Centres) | Classifier Accuracy |
| 16 | 0% | 240 | 72.68 |
| 16 | 50% | 240 | 78.55 |
| 16 | 70% | 240 | 79.96 |
|  |  |  |  |
| 16 | 0% | 480 | 73.79 |
| 16 | 50% | 480 | 77.22 |
| 16 | 70% | 480 | 79.75 |
|  |  |  |  |
| 16 | 0% | 780 | 69.16 |
| 16 | 50% | 780 | 77.37 |
| 16 | 70% | 780 | 77.23 |
|  |  |  |  |
| 32 | 0% | 240 | 71.06 |
| 32 | 50% | 240 | 74.97 |
| 32 | 70% | 240 | **80.81** |
|  |  |  |  |
| 32 | 0% | 480 | 66.86 |
| 32 | 50% | 480 | 74.15 |
| 32 | 70% | 480 | 78.42 |
|  |  |  |  |
| 32 | 0% | 780 | 65.20 |
| 32 | 50% | 780 | 72.94 |
| 32 | 70% | 780 | 76.39 |
|  |  |  |  |
| 64 | 0% | 240 | 66.76 |
| 64 | 50% | 240 | 75.34 |
| 64 | 70% | 240 | 76.40 |
|  |  |  |  |
| 64 | 0% | 480 | 62.92 |
| 64 | 50% | 480 | 71.29 |
| 64 | 70% | 480 | 77.71 |
|  |  |  |  |
| 64 | 0% | 780 | 62.45 |
| 64 | 50% | 780 | 68.42 |
| 64 | 70% | 780 | 75.21 |
|  |  |  |  |

# Histogram

|  |  |  |  |
| --- | --- | --- | --- |
| **Brush Teeth** |  | **Get Up Bed** |  |
| **Climb Stairs** |  | **Lie Down Bed** |  |
| **Comb Hair** |  | **Pour Water** |  |
| **Descend Stairs** |  | **Sit down Chair** |  |
| **Drink Glass** |  | **Stand Up Chair** |  |
| **Eat Meat** |  | **Use Telephone** |  |
| **Eat Soup** |  | **Walk** |  |

# Confusion Matrix



# Code Screen Shot

|  |  |
| --- | --- |
| Segmentation of the Vector | start\_idx = 0  end\_idx = start\_idx+segment\_length  run\_ind=**True**    **while**(run\_ind):  data\_extract=data[start\_idx:end\_idx,:].flatten().reshape(1,segment\_length\*3).astype(int)  data\_append=np.append(data\_extract,folder\_file\_extract).reshape(1,(segment\_length\*3)+3)  **if** (final\_data.shape[1] > 0):  final\_data=np.vstack((final\_data,data\_append))  **else**:  final\_data=np.vstack(data\_append)  start\_idx += int(np.round(segment\_length - ((segment\_length)\*(overlap/100))))  end\_idx = int(np.round(start\_idx+segment\_length))  **if** (end\_idx > data.shape[0]):  run\_ind = **False**  num\_elements\_per\_file+=1  num\_elements\_per\_folder+=1 |
| K-Means | kmeans\_predict\_all = np.array([[]])  kmeans = KMeans(n\_clusters=n\_cluster).fit(final\_data[:,0:segment\_length\*3].astype(int))  **for** file\_name **in** range(len(dir\_filename)):  i+=1  folder\_file\_extract = [folder\_name,folder[folder\_name],dir\_filename[file\_name]]  extract\_final\_data=final\_data[(final\_data[:,3\*segment\_length+1] == folder[folder\_name]) & (final\_data[:,3\*segment\_length+2] == dir\_filename[file\_name])][:,0:3\*segment\_length]  kmeans\_predict=kmeans.predict(extract\_final\_data)  a,b=np.histogram(kmeans\_predict,np.arange(n\_cluster+1)+1)  a=a.reshape(1,n\_cluster)  a\_append=np.append(a,folder\_file\_extract).reshape(1,n\_cluster+3)  **if** (kmeans\_predict\_all.shape[1] > 0):  kmeans\_predict\_all=np.vstack((kmeans\_predict\_all,a\_append))  **else**:  kmeans\_predict\_all=np.vstack(a\_append) |
| Generating the Histogram | **def** plot\_histogram(Category,data,cluster\_centers=480):  data\_mean = np.mean(data,axis=0)  bin\_probability = data\_mean/float(data\_mean.sum())  b = np.arange(cluster\_centers+1)+1  bin\_middles = (b[1:]+b[:-1])/2  bin\_width = b[1]-b[0]  *#plt.subplots(figsize=(12,8))*  plt.bar(bin\_middles, bin\_probability, width=bin\_width,color='blue')  plt.title(Category)  plt.show() |
| Classification | **def** predict(trn\_fold,tst\_fold,cluster=480,no\_tree=1000,max\_depth=10):  clf = RandomForestClassifier(n\_estimators=no\_tree,max\_depth=max\_depth)  X=trn\_fold[:,0:cluster]  y=trn\_fold[:,cluster]  X\_test=tst\_fold[:,0:cluster]  y\_true=tst\_fold[:,cluster]  clf.fit(X,y)  y\_pred=clf.predict(X\_test)  conf\_mat=confusion\_matrix(y\_true=y\_true, y\_pred=y\_pred)  **return** (np.sum(y\_true == y\_pred)/y\_pred.shape[0])\*100,conf\_mat |

# Source Code Screen Shot

|  |
| --- |
| **import** **numpy** **as** **np**  **import** **pandas** **as** **pd**  **import** **matplotlib.pyplot** **as** **plt**  **from** **os** **import** listdir  **from** **os.path** **import** isfile,join  **from** **sklearn.cluster** **import** KMeans  **from** **sklearn.model\_selection** **import** KFold  **from** **sklearn.ensemble** **import** RandomForestClassifier  **from** **sklearn.metrics** **import** confusion\_matrix  **import** **warnings**  warnings.filterwarnings('ignore')  **import** **math**  %**matplotlib** inline  **def** load\_dataset(foldername,filename):  file = "data/HMP\_Dataset/"+foldername+"/"+filename  data = np.array(pd.read\_csv(file,sep=' ',header=**None**,names=['X','Y','Z']))  **return** data  **def** resize\_data\_by\_factor(segment\_length=32,overlap=0,print\_Ind=**False**):  final\_data=np.array([[]])  folder = ['Brush\_teeth','Climb\_stairs','Comb\_hair','Descend\_stairs','Drink\_glass','Eat\_meat','Eat\_soup',  'Getup\_bed','Liedown\_bed','Pour\_water','Sitdown\_chair','Standup\_chair','Use\_telephone','Walk']  **for** folder\_name **in** range(len(folder)):  *#print ("Processing for :{} : {}".format(folder\_name,folder[folder\_name]))*  num\_elements\_per\_folder=0  file\_folder\_name = "data/HMP\_Dataset/"+folder[folder\_name]  dir\_filename = [f **for** f **in** listdir(file\_folder\_name) **if** isfile(join(file\_folder\_name, f))]  **for** file\_name **in** range(len(dir\_filename)):  num\_elements\_per\_file=0  folder\_file\_extract = [folder\_name,folder[folder\_name],dir\_filename[file\_name]]  data=load\_dataset(folder[folder\_name],dir\_filename[file\_name])  start\_idx = 0  end\_idx = start\_idx+segment\_length  run\_ind=**True**    **while**(run\_ind):  *#print ("Folder:{} File:{} Total:{}. num:{} Start:{} End:{}".format(folder[folder\_name],dir\_filename[file\_name],data.shape[0],num\_elements,start\_idx,end\_idx))*  data\_extract=data[start\_idx:end\_idx,:].flatten().reshape(1,segment\_length\*3).astype(int)  data\_append=np.append(data\_extract,folder\_file\_extract).reshape(1,(segment\_length\*3)+3)  **if** (final\_data.shape[1] > 0):  final\_data=np.vstack((final\_data,data\_append))  **else**:  final\_data=np.vstack(data\_append)  start\_idx += int(np.round(segment\_length - ((segment\_length)\*(overlap/100))))  end\_idx = int(np.round(start\_idx+segment\_length))  **if** (end\_idx > data.shape[0]):  run\_ind = **False**  num\_elements\_per\_file+=1  num\_elements\_per\_folder+=1  *#if (print\_Ind):*  *#print ("Folder:{} File:{} Total:{}. num:{} Start:{} End:{}".format(folder[folder\_name],dir\_filename[file\_name],data.shape[0],num\_elements\_per\_file,start\_idx,end\_idx))*  **if** (print\_Ind):  print ("Category:**{}**:**{}** File:**{}** Segment:**{}**".format(folder[folder\_name],folder\_name,len(dir\_filename),num\_elements\_per\_folder))  **if** (print\_Ind):  print ("Resize of Data Completed: **{}**".format(final\_data.shape))  **return** final\_data  **def** kmeans\_prediction(final\_data,n\_cluster=480,segment\_length=32,print\_Ind=**False**):  i=0  kmeans\_predict\_all = np.array([[]])  kmeans = KMeans(n\_clusters=n\_cluster).fit(final\_data[:,0:segment\_length\*3].astype(int))  folder = ['Brush\_teeth','Climb\_stairs','Comb\_hair','Descend\_stairs','Drink\_glass','Eat\_meat','Eat\_soup',  'Getup\_bed','Liedown\_bed','Pour\_water','Sitdown\_chair','Standup\_chair','Use\_telephone','Walk']  **for** folder\_name **in** range(len(folder)):  *#print ("Processing for :{} : {}".format(folder\_name,folder[folder\_name]))*  file\_folder\_name = "data/HMP\_Dataset/"+folder[folder\_name]  dir\_filename = [f **for** f **in** listdir(file\_folder\_name) **if** isfile(join(file\_folder\_name, f))]  **for** file\_name **in** range(len(dir\_filename)):  i+=1  folder\_file\_extract = [folder\_name,folder[folder\_name],dir\_filename[file\_name]]  extract\_final\_data=final\_data[(final\_data[:,3\*segment\_length+1] == folder[folder\_name]) & (final\_data[:,3\*segment\_length+2] == dir\_filename[file\_name])][:,0:3\*segment\_length]  **if** (print\_Ind):  print ("Processing Folder:**{}** File Name:**{}** Records:**{}**".format(folder[folder\_name],dir\_filename[file\_name],extract\_final\_data.shape))  kmeans\_predict=kmeans.predict(extract\_final\_data)  a,b=np.histogram(kmeans\_predict,np.arange(n\_cluster+1)+1)  a=a.reshape(1,n\_cluster)  a\_append=np.append(a,folder\_file\_extract).reshape(1,n\_cluster+3)  **if** (kmeans\_predict\_all.shape[1] > 0):  kmeans\_predict\_all=np.vstack((kmeans\_predict\_all,a\_append))  **else**:  kmeans\_predict\_all=np.vstack(a\_append)  **if** (print\_Ind):  print ("KMeans Prediction Completed: **{}**".format(kmeans\_predict\_all.shape))  **return** kmeans\_predict\_all  **def** plot\_histogram(Category,data,cluster\_centers=480):  data\_mean = np.mean(data,axis=0)  bin\_probability = data\_mean/float(data\_mean.sum())  b = np.arange(cluster\_centers+1)+1  bin\_middles = (b[1:]+b[:-1])/2  bin\_width = b[1]-b[0]  *#plt.subplots(figsize=(12,8))*  plt.bar(bin\_middles, bin\_probability, width=bin\_width,color='blue')  plt.title(Category)  plt.show()    **def** kfold\_data(kmeans\_predict\_all,n\_cluster=480,print\_Ind=**False**):  folder = ['Brush\_teeth','Climb\_stairs','Comb\_hair','Descend\_stairs','Drink\_glass','Eat\_meat','Eat\_soup',  'Getup\_bed','Liedown\_bed','Pour\_water','Sitdown\_chair','Standup\_chair','Use\_telephone','Walk']  *#folder = ['Brush\_teeth']*  kmeans\_predict\_extract = kmeans\_predict\_all[:,0:n\_cluster+1].astype(int)  train\_data\_fold = np.array([[]])  test\_data\_fold = np.array([[]])  kf = KFold(n\_splits=3)  **for** f **in** range(len(folder)):  fold=1  kmeans\_predict\_extract\_fold=kmeans\_predict\_extract[kmeans\_predict\_extract[:,n\_cluster] == f]  **for** trn\_idx,test\_idx **in** kf.split(kmeans\_predict\_extract\_fold):  **if** (print\_Ind):  print ("Category:**{}** Fold:**{}** :: **{}**:**{}** ::: ".format(f,fold,trn\_idx.shape,test\_idx.shape))  category\_trn\_repeat = np.repeat(f,trn\_idx.shape[0]).reshape(trn\_idx.shape[0],1)  category\_test\_repeat = np.repeat(f,test\_idx.shape[0]).reshape(test\_idx.shape[0],1)  fold\_trn\_repeat = np.repeat(fold,trn\_idx.shape[0]).reshape(trn\_idx.shape[0],1)  fold\_test\_repeat = np.repeat(fold,test\_idx.shape[0]).reshape(test\_idx.shape[0],1)  **if** (train\_data\_fold.shape[1]>0):  train\_data\_fold=np.vstack((train\_data\_fold,np.hstack((fold\_trn\_repeat,kmeans\_predict\_extract\_fold[trn\_idx]))))  **else**:  train\_data\_fold=np.vstack(np.hstack((fold\_trn\_repeat,kmeans\_predict\_extract\_fold[trn\_idx])))  **if** (test\_data\_fold.shape[1]>0):  test\_data\_fold=np.vstack((test\_data\_fold,np.hstack((fold\_test\_repeat,kmeans\_predict\_extract\_fold[test\_idx]))))  **else**:  test\_data\_fold=np.vstack(np.hstack((fold\_test\_repeat,kmeans\_predict\_extract\_fold[test\_idx])))  fold+=1  **return** train\_data\_fold,test\_data\_fold  **def** predict(trn\_fold,tst\_fold,cluster=480,no\_tree=1000,max\_depth=10):  clf = RandomForestClassifier(n\_estimators=no\_tree,max\_depth=max\_depth) *#n\_estimators = no of tree, max\_depth = depth of the tree*  X=trn\_fold[:,0:cluster]  y=trn\_fold[:,cluster]  X\_test=tst\_fold[:,0:cluster]  y\_true=tst\_fold[:,cluster]  clf.fit(X,y)  y\_pred=clf.predict(X\_test)  conf\_mat=confusion\_matrix(y\_true=y\_true, y\_pred=y\_pred)  **return** (np.sum(y\_true == y\_pred)/y\_pred.shape[0])\*100,conf\_mat  **def** predict\_fold(train\_data\_fold,test\_data\_fold,cluster=480,print\_Ind=**False**):  avg\_acc=0  **for** fold **in** range(3):  trn\_fold=train\_data\_fold[train\_data\_fold[:,0] == fold+1][:,1:cluster+2]  tst\_fold=test\_data\_fold[test\_data\_fold[:,0] == fold+1][:,1:cluster+2]  acc,conf\_mat=predict(trn\_fold,tst\_fold,cluster=cluster)  avg\_acc+=acc  **if** (print\_Ind):  print ("Fold:**{}** Trn:**{}** Test:**{}** Accuracy:**{}**".format(fold,trn\_fold.shape,tst\_fold.shape,acc))  avg\_acc=(avg\_acc/3)  **if** (print\_Ind):  print ("Average Accuracy:**{}**".format(avg\_acc/3))  **return** avg\_acc,conf\_mat  **def** predict\_segement\_overlap\_cluster():  segment\_list = [16,32,64]  overlap\_list = [0,50,70]  cluster\_list = [240,480,780]  *# segment\_list = [16]*  *# overlap\_list = [0]*  *# cluster\_list = [240]*  labels=['Brush\_teeth','Climb\_stairs','Comb\_hair','Descend\_stairs','Drink\_glass','Eat\_meat','Eat\_soup',  'Getup\_bed','Liedown\_bed','Pour\_water','Sitdown\_chair','Standup\_chair','Use\_telephone','Walk']  **for** s **in** (range(len(segment\_list))):  **for** o **in** (range(len(overlap\_list))):  final\_data=resize\_data\_by\_factor(segment\_length=segment\_list[s],overlap=overlap\_list[o],print\_Ind=**True**)  df\_final\_data=pd.DataFrame(final\_data)  final\_data\_file\_name = "submission/final\_data/final\_data\_"+str(segment\_list[s])+"\_"+str(overlap\_list[o])+".csv"  df\_final\_data.to\_csv(final\_data\_file\_name,index=**False**)  *#final\_data=np.array(pd.read\_csv("final\_data.csv")) Comment It out (Only Uncomment for Re-Run)*  **for** k **in** range(len(cluster\_list)):  kmeans\_predict\_all=kmeans\_prediction(final\_data,n\_cluster=cluster\_list[k],segment\_length=segment\_list[s],print\_Ind=**False**)  train\_data\_fold,test\_data\_fold=kfold\_data(kmeans\_predict\_all,n\_cluster=cluster\_list[k],print\_Ind=**False**)  average\_accuracy,confusion\_mat=predict\_fold(train\_data\_fold,test\_data\_fold,cluster=cluster\_list[k],print\_Ind=**False**)  df\_confusion\_mat=pd.DataFrame(confusion\_mat,columns=labels,index=labels)  confusion\_mat\_file\_name = "submission/confusion\_matrix/confusion\_matrix\_"+str(overlap\_list[k])+"\_"+str(cluster\_list[o])+"\_"+str(segment\_list[s]) +".html"  df\_confusion\_mat.to\_html(confusion\_mat\_file\_name,index=**True**)  print ("Segment:**{}** Overlap%:**{}** Cluster:**{}** Accuracy:**{}** Confusion Matrix:**{}**".format(segment\_list[s],overlap\_list[o],cluster\_list[k],average\_accuracy,confusion\_mat\_file\_name))  **def** plot\_histogram\_all\_category(file\_name,no\_cluster):  labels=['Brush\_teeth','Climb\_stairs','Comb\_hair','Descend\_stairs','Drink\_glass','Eat\_meat','Eat\_soup',  'Getup\_bed','Liedown\_bed','Pour\_water','Sitdown\_chair','Standup\_chair','Use\_telephone','Walk']  file\_name="submission/final\_data/"+str(file\_name)+".csv"  final\_data=np.array(pd.read\_csv(file\_name))  kmeans\_predict\_all=kmeans\_prediction(final\_data,n\_cluster=no\_cluster,segment\_length=32,print\_Ind=**False**)  kmeans\_predict\_all\_hist=kmeans\_predict\_all[:,0:no\_cluster+1].astype(int)  **for** i **in** range(14):  print ("Processing for i:**{}**".format(i))  x=kmeans\_predict\_all\_hist[kmeans\_predict\_all\_hist[:,no\_cluster] == i][:,0:no\_cluster]  plot\_histogram(Category=labels[i],data=x,cluster\_centers=no\_cluster)  predict\_segement\_overlap\_cluster()  plot\_histogram\_all\_category("submission/final\_data/final\_data\_32\_70.csv",240) |