CODE 1

temp.py | untitled2.py*

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar  9 12:23:57 2020

@author: M
"""
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

Xtest=pd.read_csv('XTEST.csv')
Xtrain=pd.read_csv('XTRAIN.csv')
Ytest=pd.read_csv('YTEST.csv')
Ytrain=pd.read_csv('YTRAIN.csv')
X = pd.concat([Xtrain, Xtest], axis = 0)
Y = pd.concat([Ytrain, Ytest], axis = 0)
X.head()
y2 = Y
y2 = np.asarray(y2)
y2 = pd.DataFrame(y2)
y2[0][367]
for i in range(0, len(y2)):
    if(y2[0][i] == 1):
        y2[0][i] = 'Walking'
    elif(y2[0][i] == 2):
        y2[0][i] = 'Walking Upstairs'
    elif(y2[0][i] == 3):
        y2[0][i] = 'Walking Downstairs'
    elif(y2[0][i] == 4):
        y2[0][i] = 'Sitting'
    elif(y2[0][i] == 5):
        y2[0][i] = 'Standing'
    elif(y2[0][i] == 6):
        y2[0][i] = 'Laying'
X.shape
#PCA algorithm being used for the dimensionality reduction process
from sklearn.decomposition import PCA
```

Editor - C:\Users\M\Desktop\UCI HAR Dataset (1)\UCI HAR Dataset\untitled2.py

temp.py | untitled2.py*

```
    elif(y2[0][i] == 4):
        y2[0][i] = 'Sitting'
    elif(y2[0][i] == 5):
        y2[0][i] = 'Standing'
    elif(y2[0][i] == 6):
        y2[0][i] = 'Laying'
X.shape
#PCA algorithm being used for the dimensionality reduction process
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
pca.fit(Y)

y_trans = pca.transform(Y)
y_trans = pd.DataFrame(y_trans, columns = ['0'])
y_trans.head()
y_trans['label'] = y2[0]

y_trans.tail()
import matplotlib
sns.lmplot(x = '0', y = '0', data = y_trans, hue = 'label',  fit_reg = False, x_jitter = 4.25, y_jitter = 2.4, size = 15)
plt.xlabel('F1 Feature')
plt.ylabel('F2 Feature')
import matplotlib
matplotlib.rcParams.update({'font.size': 20})
plt.show()
plt.figure(figsize = (18, 10))
sns.countplot(x = 'label', data = y_trans)
plt.tight_layout()
plt.show()

#F1 strip plot
plt.figure(figsize = (15, 8))
sns.stripplot(x = 'label', y= '0', data = y_trans, jitter = True, dodge = True)
plt.tight_layout()
plt.xlabel('Class Label')
plt.ylabel(' Feature')
plt.show()
```

CODE 2

temp.py | untitled2.py*

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar  9 12:23:57 2020

@author: M
"""
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

Xtest=pd.read_csv('XTEST.csv')
Xtrain=pd.read_csv('XTRAIN.csv')
Ytest=pd.read_csv('YTEST.csv')
Ytrain=pd.read_csv('YTRAIN.csv')

classifier=svm.SVC()

parameters=[{'kernel': ['rbf'], 'gamma': [0.001, 0.0001], 'C': [1, 10, 100, 1000]}, {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
model=GridSearchCV(classifier,parameters,n_jobs=-1,cv=4,verbose=4)
str(model.fit(Ytrain.as_matrix(),Ytrain.as_matrix().ravel().T))


from sklearn.metrics import accuracy_score
ypred=model.predict(Ytest)
accuracy=accuracy_score(Ytest,ypred)

print ('Best Parameters ')
str(model.best_params_)
print ('Accuracy Score: ')
str(accuracy*100) + ' %'
```

CODE 3

temp.py | untitled2.py*

```
"""
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

Xtest=pd.read_csv('XTEST.csv')
Xtrain=pd.read_csv('XTRAIN.csv')
Ytest=pd.read_csv('YTEST.csv')
Ytrain=pd.read_csv('YTRAIN.csv')

xtrain=Xtrain.values #converting into array
xtrain
xtest=Xtest.values

from sklearn.preprocessing import OneHotEncoder
onehotencoder = OneHotEncoder()
ytrain = onehotencoder.fit_transform(Ytrain).toarray()
ytrain

import keras
from keras.models import Sequential #sequential is required to initialise the neural network
from keras.layers import Dense      #dense is used to build the layers
from keras.layers import Dropout    #Dropout Layer in order to prevent Regularization in the network

#Initialising the deep learning model
#Defining the model as a sequence of layers
classifier = Sequential()
classifier.add(Dense(48, input_dim = 561, kernel_initializer='uniform', activation='relu', ))
#creating a network of 561 X 48 X 24 X 12 X6
classifier.add(Dropout(0.1))
classifier.add(Dense(24, kernel_initializer='uniform', activation='relu'))
classifier.add(Dropout(0.1))
classifier.add(Dense(12, kernel_initializer='uniform', activation='relu'))
classifier.add(Dropout(0.1))
classifier.add(Dense(6, kernel_initializer='uniform', activation='softmax'))
```

```
classifier.add(Dense(12, kernel_initializer='uniform', activation='relu'))
classifier.add(Dropout(0.1))
classifier.add(Dense(6, kernel_initializer='uniform', activation='softmax'))

classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics = ['accuracy'])
#Fitting the ANN to the training set
classifier.fit(xtrain, ytrain, batch_size=20, epochs=10, verbose = 4)
pred = classifier.predict(xtest)
predictions = []
for i in range(len(pred)):
    predictions.append(pred[i].argmax() + 1)
    from sklearn import metrics
print ('(str(metrics.accuracy_score(ytest, predictions)*100) + ' %')')
```