Main.py

```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler
import pickle
import warnings
warnings.filterwarnings('ignore')
a=pd.read_csv("Gait_Data.csv")
print(a)
```

```
             alx      aly      alz       glx      gly      glz     arx  \
0        2.18490  -9.6967  0.63077  0.103900 -0.84053 -0.68762 -8.6499
1        2.38760  -9.5080  0.68389  0.085343 -0.83865 -0.68369 -8.6275
2        2.40860  -9.5674  0.68113  0.085343 -0.83865 -0.68369 -8.5055
3        2.18140  -9.4301  0.55031  0.085343 -0.83865 -0.68369 -8.6279
4        2.41730  -9.3889  0.71098  0.085343 -0.83865 -0.68369 -8.7008
...          ...      ...      ...       ...      ...      ...     ...
910381  0.32374  -9.9511 -1.04640 -0.569570 -0.75422 -0.52456 -6.5886
910382  0.40570 -10.0510 -1.11310 -0.569570 -0.75422 -0.52456 -6.0806
910383  0.48707 -10.0110 -1.09980 -0.593690 -0.75047 -0.53045 -5.3752
910384  0.41283  -9.8038 -1.17360 -0.593690 -0.75047 -0.53045 -4.6322
910385  0.29146  -9.8527 -1.11470 -0.593690 -0.75047 -0.53045 -3.9156

              ary       arz       grx      gry       grz  Activity
0       -4.5781  0.187760 -0.449020 -1.01030  0.034483         0
1       -4.3198  0.023595 -0.449020 -1.01030  0.034483         0
2       -4.2772  0.275720 -0.449020 -1.01030  0.034483         0
3       -4.3163  0.367520 -0.456860 -1.00820  0.025862         0
4       -4.1459  0.407290 -0.456860 -1.00820  0.025862         0
...         ...       ...       ...      ...       ...       ...
910381 -10.5290  2.129800 -0.182350 -0.92813 -0.553880         0
910382 -10.5710  1.877700 -0.182350 -0.92813 -0.553880         0
910383 -10.0910  1.870400 -0.078431 -0.96509 -0.560340         0
910384  -9.7003  1.973300 -0.078431 -0.96509 -0.560340         0
910385  -9.5098  2.024500 -0.078431 -0.96509 -0.560340         0

[910386 rows x 13 columns]
```

############################## features ####################################

```
X=a.drop(['Activity'],axis=1)
```

```
print(X)
```

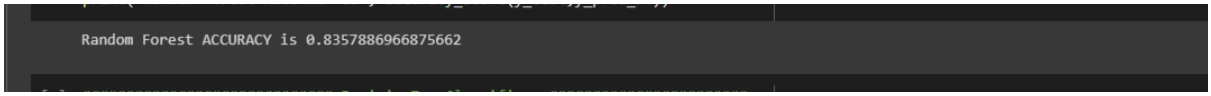
```
            alx      aly      alz       glx      gly      glz     arx  \
0       2.18490  -9.6967  0.63077  0.103900 -0.84053 -0.68762 -8.6499
1       2.38760  -9.5080  0.68389  0.085343 -0.83865 -0.68369 -8.6275
2       2.40860  -9.5674  0.68113  0.085343 -0.83865 -0.68369 -8.5055
3       2.18140  -9.4301  0.55031  0.085343 -0.83865 -0.68369 -8.6279
4       2.41730  -9.3889  0.71098  0.085343 -0.83865 -0.68369 -8.7008
...         ...      ...      ...       ...      ...      ...     ...
910381  0.32374  -9.9511 -1.04640 -0.569570 -0.75422 -0.52456 -6.5886
910382  0.40570 -10.0510 -1.11310 -0.569570 -0.75422 -0.52456 -6.0806
910383  0.48707 -10.0110 -1.09980 -0.593690 -0.75047 -0.53045 -5.3752
910384  0.41283  -9.8038 -1.17360 -0.593690 -0.75047 -0.53045 -4.6322
910385  0.29146  -9.8527 -1.11470 -0.593690 -0.75047 -0.53045 -3.9156

            ary       arz       grx      gry       grz
0       -4.5781  0.187760 -0.449020 -1.01030  0.034483
1       -4.3198  0.023595 -0.449020 -1.01030  0.034483
2       -4.2772  0.275720 -0.449020 -1.01030  0.034483
3       -4.3163  0.367520 -0.456860 -1.00820  0.025862
4       -4.1459  0.407290 -0.456860 -1.00820  0.025862
...         ...       ...       ...      ...       ...
910381 -10.5290  2.129800 -0.182350 -0.92813 -0.553880
910382 -10.5710  1.877700 -0.182350 -0.92813 -0.553880
910383 -10.0910  1.870400 -0.078431 -0.96509 -0.560340
910384  -9.7003  1.973300 -0.078431 -0.96509 -0.560340
910385  -9.5098  2.024500 -0.078431 -0.96509 -0.560340

[910386 rows x 12 columns]
0         0
1         0
2         0
3         0
4         0
```

```
            ary       arz       grx      gry       grz
0       -4.5781  0.187760 -0.449020 -1.01030  0.034483
1       -4.3198  0.023595 -0.449020 -1.01030  0.034483
2       -4.2772  0.275720 -0.449020 -1.01030  0.034483
3       -4.3163  0.367520 -0.456860 -1.00820  0.025862
4       -4.1459  0.407290 -0.456860 -1.00820  0.025862
...         ...       ...       ...      ...       ...
910381 -10.5290  2.129800 -0.182350 -0.92813 -0.553880
910382 -10.5710  1.877700 -0.182350 -0.92813 -0.553880
910383 -10.0910  1.870400 -0.078431 -0.96509 -0.560340
910384  -9.7003  1.973300 -0.078431 -0.96509 -0.560340
910385  -9.5098  2.024500 -0.078431 -0.96509 -0.560340

[910386 rows x 12 columns]
0         0
1         0
2         0
3         0
4         0
         ..
910381    0
910382    0
910383    0
910384    0
910385    0
Name: Activity, Length: 910386, dtype: int64
```

```
############################   labels  ######################################
Y=a['Activity']
print(Y)

############################## traing and testing part ######################
x_train,x_test,y_train,y_test = train_test_split(X,Y,shuffle=True,test_size=0.25, random_state=0)


############################## Algorithm Implementation ######################


############################## Gaussian Naive Bayes  ######################
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB.fit(x_train, y_train)  #train the data
y_pred=NB.predict(x_test)
##print(y_pred)
##print(y_test)
print('Naive Bayes ACCURACY is', accuracy_score(y_test,y_pred))
```

```
Naive Bayes ACCURACY is 0.6296875617868425
```

```
############################## RandomForestClassifier  ######################
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(x_train, y_train)  #train the data
y_pred_rf=clf.predict(x_test)
##print(y_pred)
##print(y_test)
print('Random Forest ACCURACY is', accuracy_score(y_test,y_pred_rf))
```



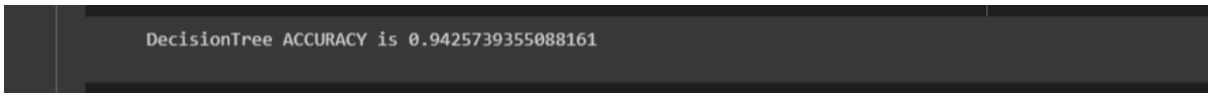

################################ KNN ##########################################

```
from sklearn.neighbors import KNeighborsClassifier

# Create KNN classifier with k=3
knn = KNeighborsClassifier(n_neighbors=3)

# Train the classifier
knn.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
```


################################3#### SVM ####################################

```
from sklearn import svm

# Create a SVM classifier with a linear kernel
clf = svm.SVC(kernel='linear', C=1)

# Train the classifier on the training data
clf.fit(x_train, y_train)
```

```
# Make predictions on the testing data
y_pred = clf.predict(x_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print('SVM Accuracy:', accuracy)

############################## DecisionTreeClassifier  ######################
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(random_state=0)
DT.fit(x_train, y_train)  #train the data
y_pred_DT=DT.predict(x_test)
##print(y_pred)
##print(y_test)
print('DecisionTree ACCURACY is', accuracy_score(y_test,y_pred_DT))
```




```
from keras.models import Sequential
from keras.layers import LSTM, Dense
model_LSTM = Sequential()
model_LSTM.add(LSTM(32, input_shape=(12,1)))
model_LSTM.add(Dense(16, activation='relu'))
model_LSTM.add(Dense(1, activation='sigmoid'))

# Compile the model
model_LSTM.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
history = model_LSTM.fit(x_train, y_train, epochs=1, batch_size=32, validation_data=(x_test, y_test))
```

```
# Evaluate the model
test_loss, test_acc = model_LSTM.evaluate(x_train, Y_test)
##print("Test Loss:", test_loss)
print("Test Accuracy:", test_acc)
```


```
filename = 'model.pkl'
#pickle.dump(DT, open(filename, 'wb'))
```