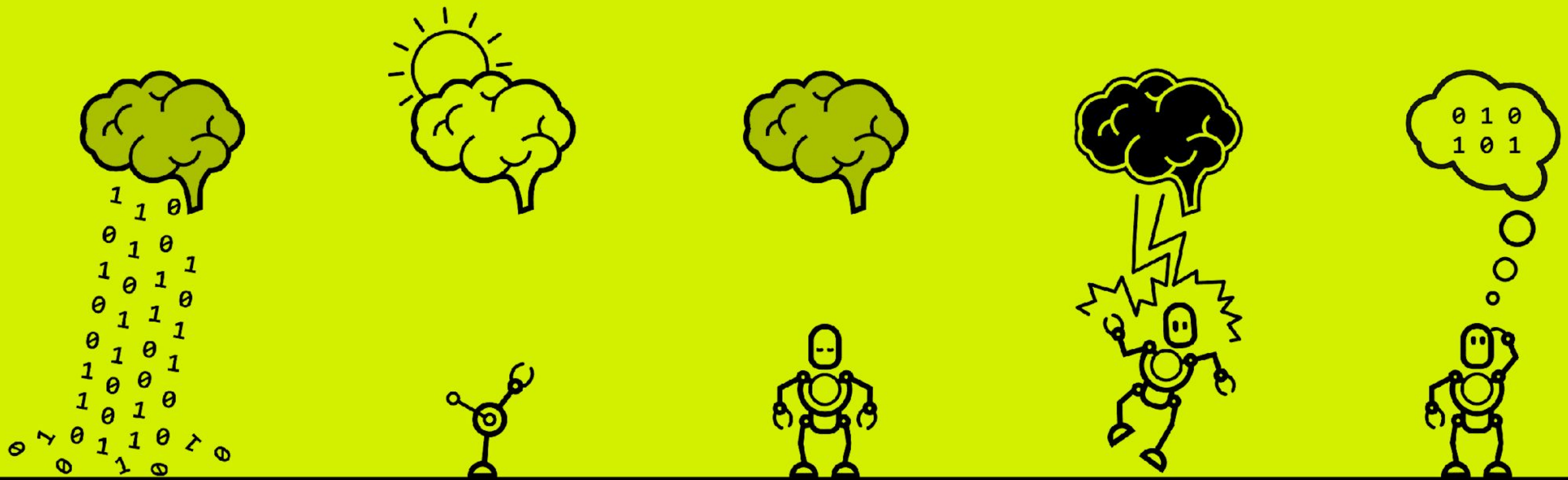
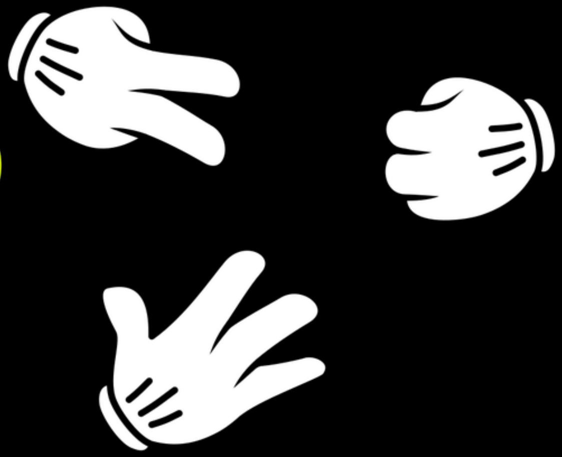


BR41N.IO

THE BRAIN-COMPUTER INTERFACE
DESIGNERS HACKATHON

BR41N.IO



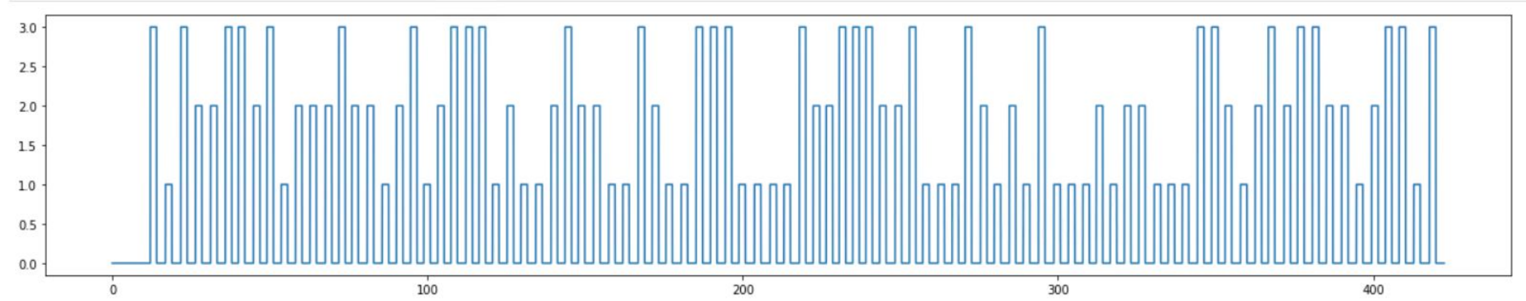
ECoG Hand Pose Data Analysis

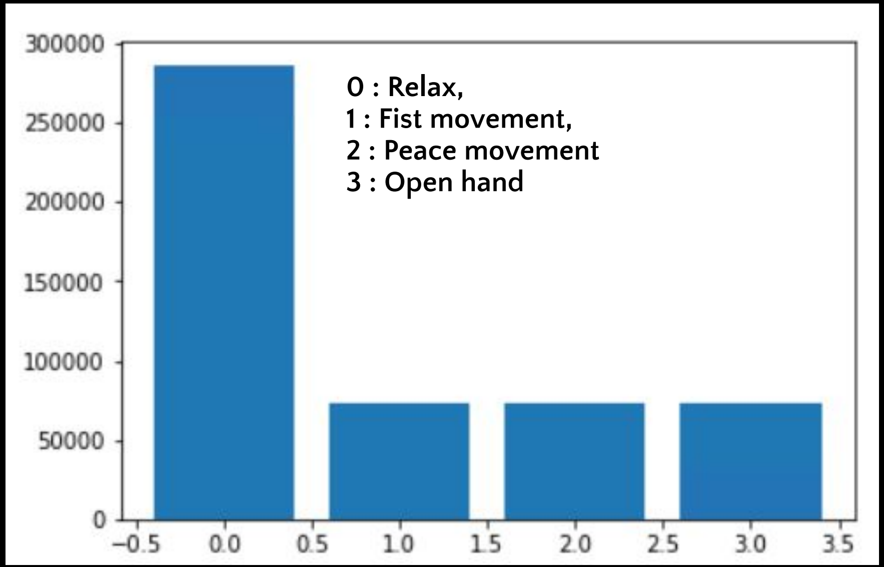
A Team Has No Name

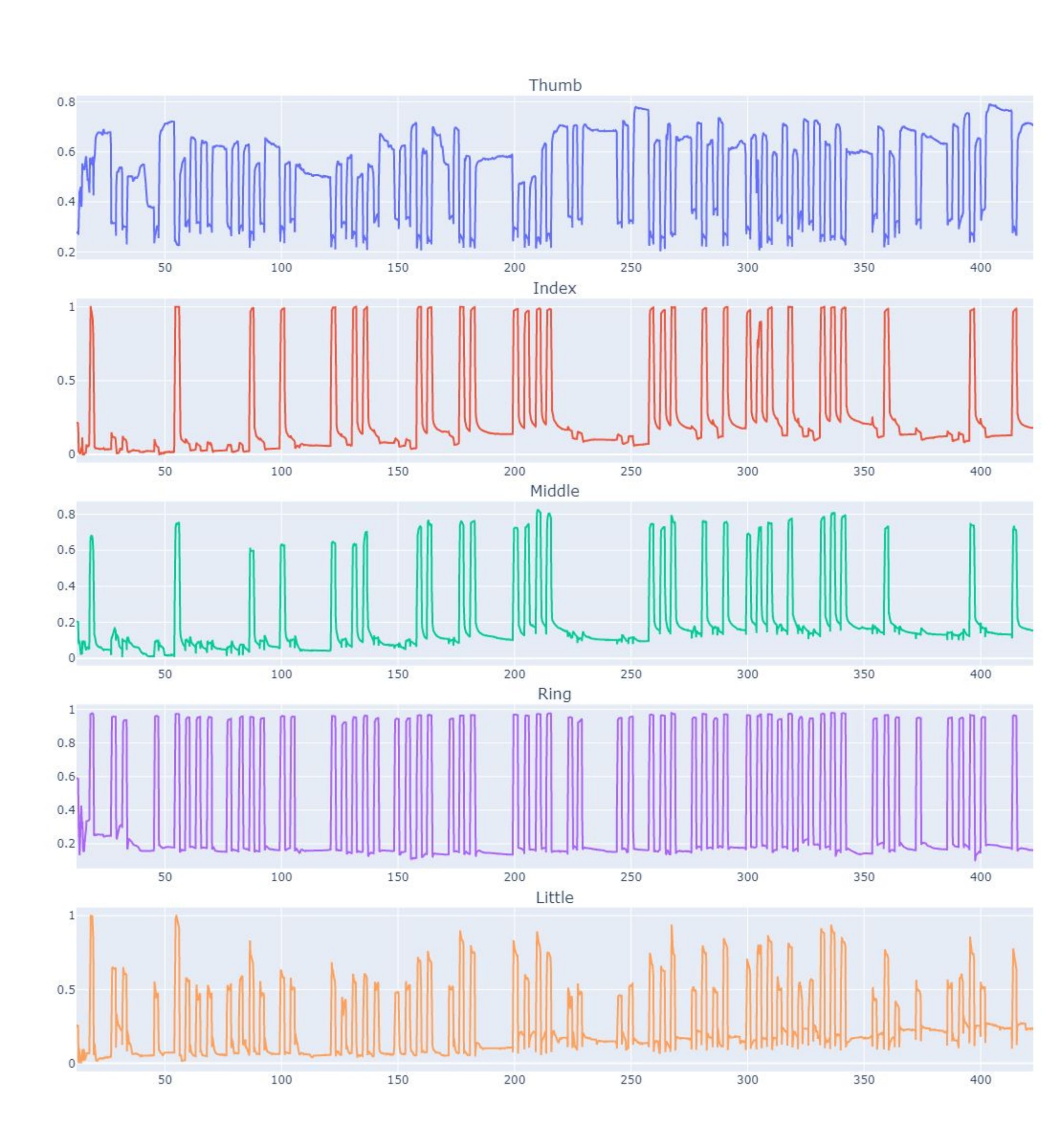
Muthu Jeyanthi, Woobean Lee,
Vivek Mohan, Amin Ranjbar,
Sayali Bachhav, Atharva Kand,
Noga Mudrik, Giulia Porro

Data Description

- 90 trials
- 60 electrodes (ECoG signals)
- 5 finger movement (Hand Glove Signal)
- Each trial with one of the rock-paper-scissors gestures
- Each cue presented for 2s, followed by a black screen for 2-3s
- **Sampling frequency:** 1200 Hz



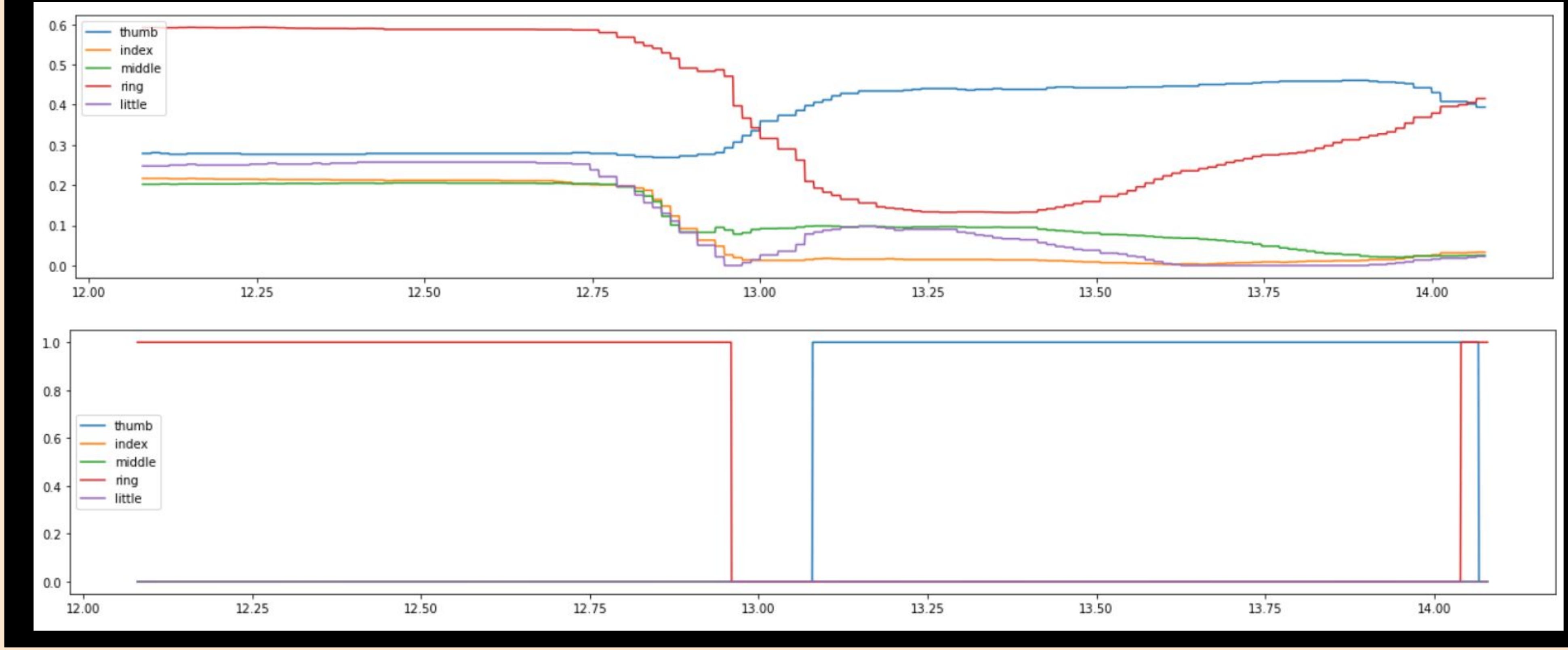




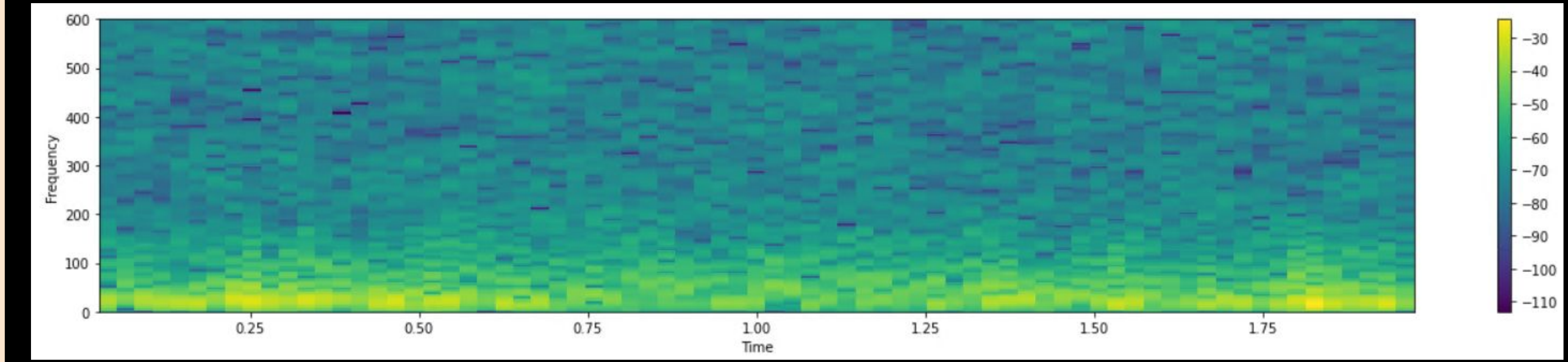
Hand Glove Data

INITIAL SITUATION

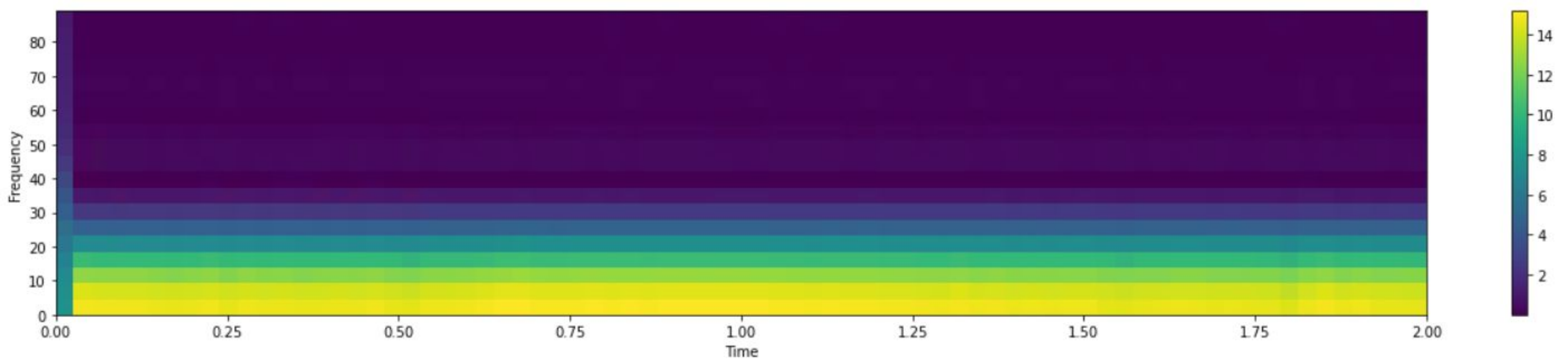
Change in value per Data glove



FFT signal of PCAed ECoG



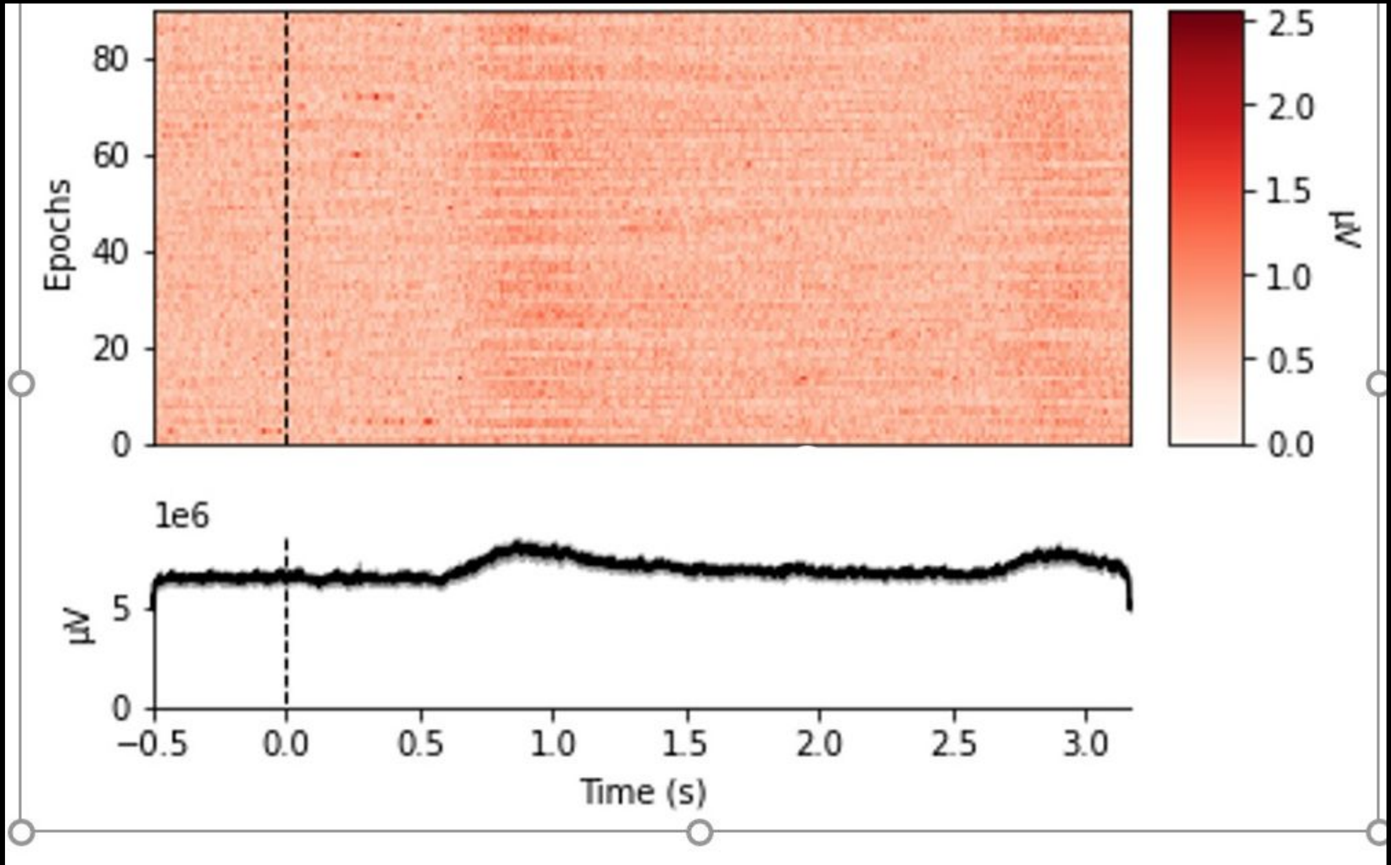
STFT signal of PCAed ECoG



BR41N.IO

Epochs Creation

- 90 epochs from -0.5 seconds to 3 seconds



Features Extracted

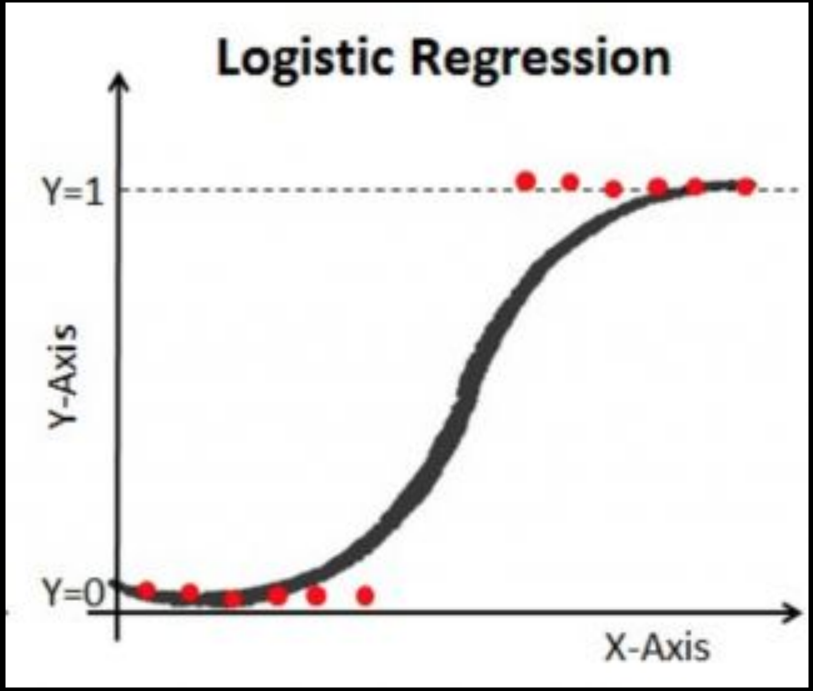
EcoG Data

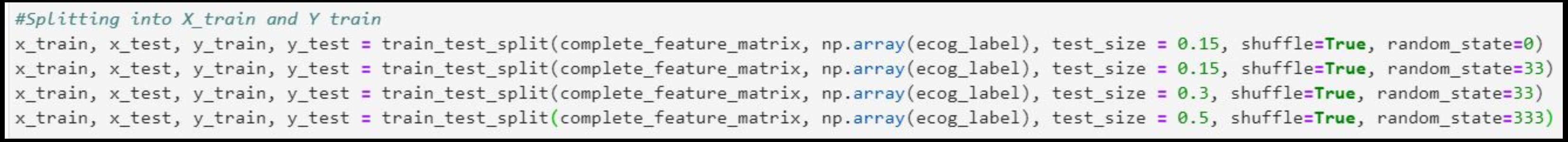
- Mean
- Kurtosis
- Skewness
- Standard Deviation
- Gradient
- Root Mean Square
- Average frequency band powers
- Ratio of band powers

Glove data

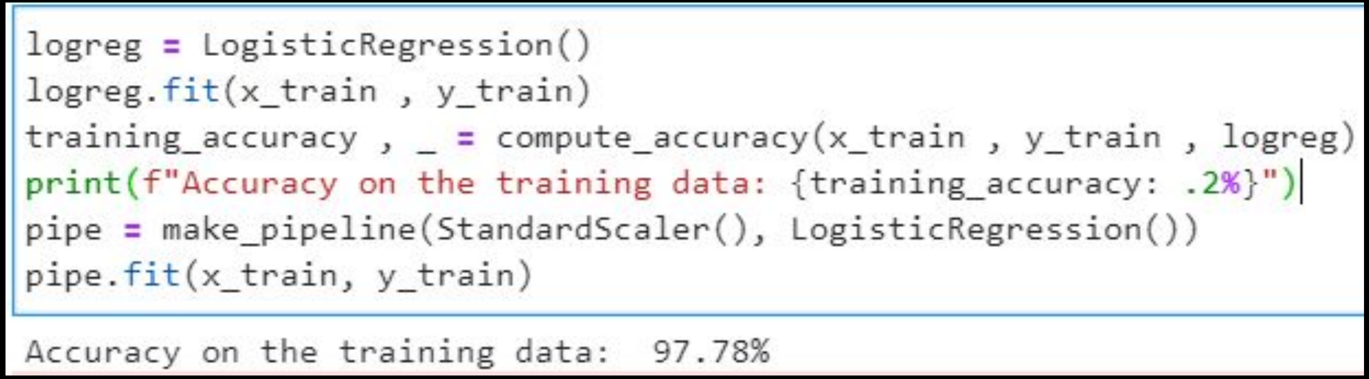
- Mean
- Kurtosis
- Skewness
- Standard Deviation
- Spike over time

Logistic Regression Model



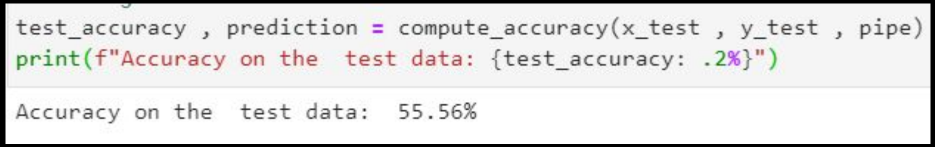


```
#Splitting into X_train and Y train
x_train, x_test, y_train, y_test = train_test_split(complete_feature_matrix, np.array(ecog_label), test_size = 0.15, shuffle=True, random_state=0)
x_train, x_test, y_train, y_test = train_test_split(complete_feature_matrix, np.array(ecog_label), test_size = 0.15, shuffle=True, random_state=33)
x_train, x_test, y_train, y_test = train_test_split(complete_feature_matrix, np.array(ecog_label), test_size = 0.3, shuffle=True, random_state=33)
x_train, x_test, y_train, y_test = train_test_split(complete_feature_matrix, np.array(ecog_label), test_size = 0.5, shuffle=True, random_state=333)
```



```
logreg = LogisticRegression()
logreg.fit(x_train , y_train)
training_accuracy , _ = compute_accuracy(x_train , y_train , logreg)
print(f"Accuracy on the training data: {training_accuracy: .2%}")
pipe = make_pipeline(StandardScaler(), LogisticRegression())
pipe.fit(x_train, y_train)
```

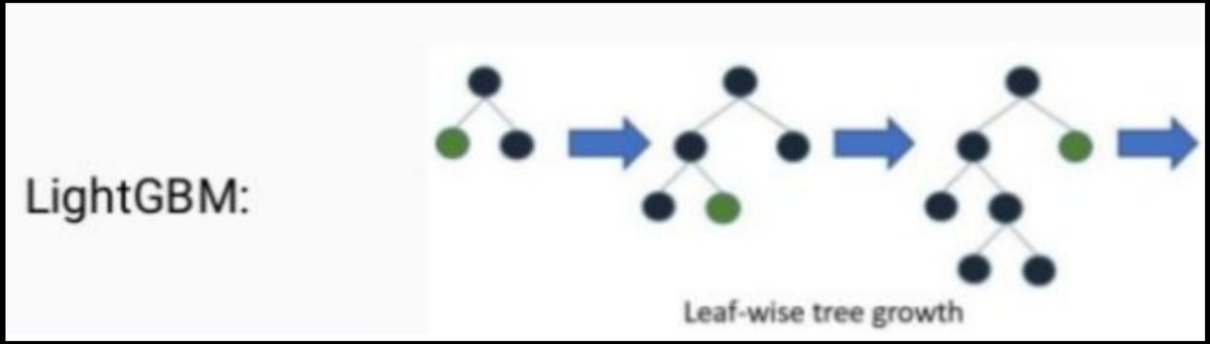
```
Accuracy on the training data:  97.78%
```

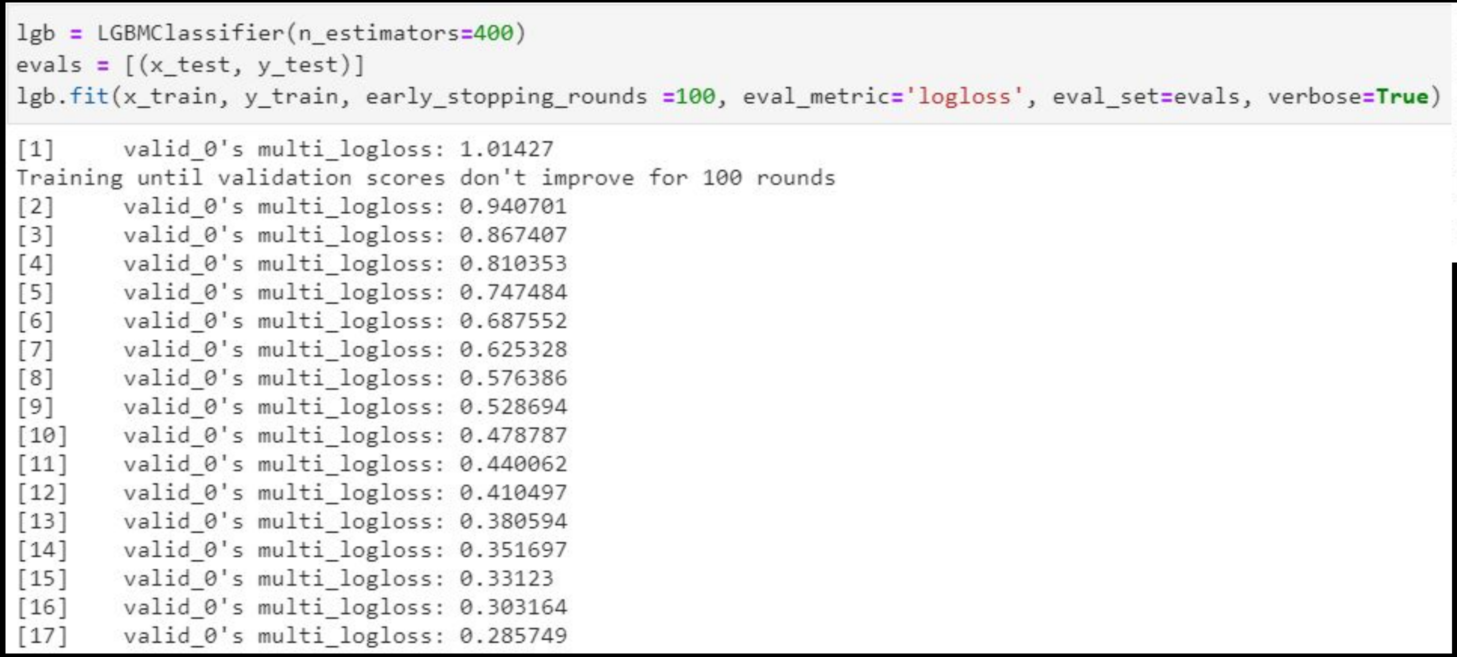


```
test_accuracy , prediction = compute_accuracy(x_test , y_test , pipe)
print(f"Accuracy on the  test data: {test_accuracy: .2%}")
```

```
Accuracy on the  test data:  55.56%
```

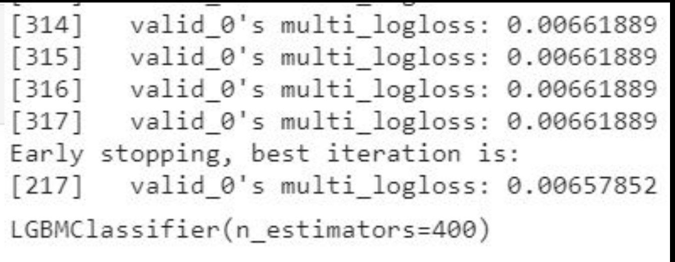
LightGBM Classifier Model





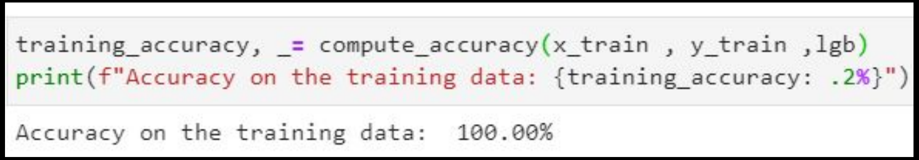
```
lgb = LGBMClassifier(n_estimators=400)
evals = [(x_test, y_test)]
lgb.fit(x_train, y_train, early_stopping_rounds =100, eval_metric='logloss', eval_set=evals, verbose=True)
```

```
[1]	valid_0's multi_logloss: 1.01427
Training until validation scores don't improve for 100 rounds
[2]	valid_0's multi_logloss: 0.940701
[3]	valid_0's multi_logloss: 0.867407
[4]	valid_0's multi_logloss: 0.810353
[5]	valid_0's multi_logloss: 0.747484
[6]	valid_0's multi_logloss: 0.687552
[7]	valid_0's multi_logloss: 0.625328
[8]	valid_0's multi_logloss: 0.576386
[9]	valid_0's multi_logloss: 0.528694
[10]	valid_0's multi_logloss: 0.478787
[11]	valid_0's multi_logloss: 0.440062
[12]	valid_0's multi_logloss: 0.410497
[13]	valid_0's multi_logloss: 0.380594
[14]	valid_0's multi_logloss: 0.351697
[15]	valid_0's multi_logloss: 0.33123
[16]	valid_0's multi_logloss: 0.303164
[17]	valid_0's multi_logloss: 0.285749
```



```
[314]	valid_0's multi_logloss: 0.00661889
[315]	valid_0's multi_logloss: 0.00661889
[316]	valid_0's multi_logloss: 0.00661889
[317]	valid_0's multi_logloss: 0.00661889
Early stopping, best iteration is:
[217]	valid_0's multi_logloss: 0.00657852

LGBMClassifier(n_estimators=400)
```

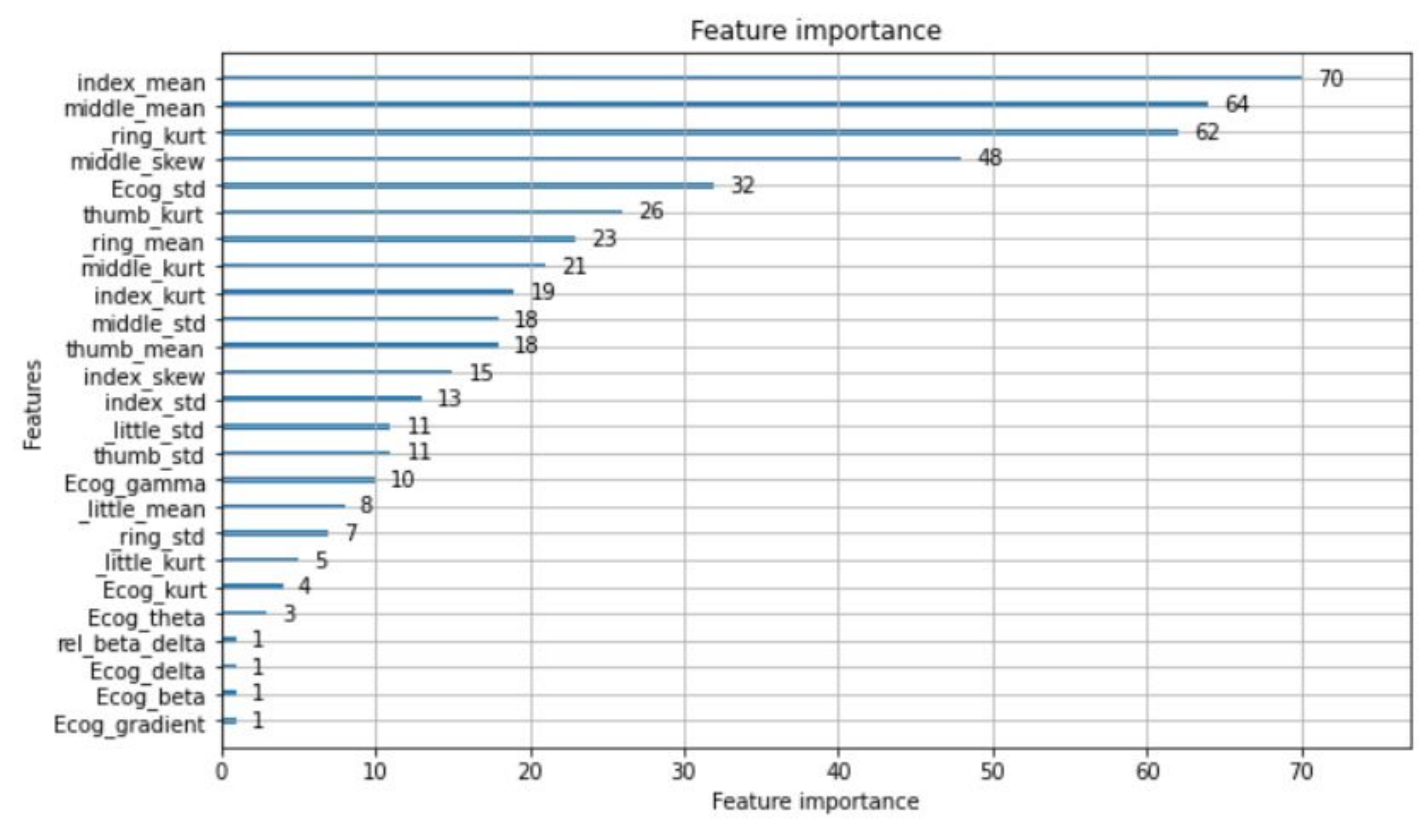


```
training_accuracy, _= compute_accuracy(x_train , y_train ,lgb)
print(f"Accuracy on the training data: {training_accuracy: .2%}")
```

```
Accuracy on the training data:  100.00%
```

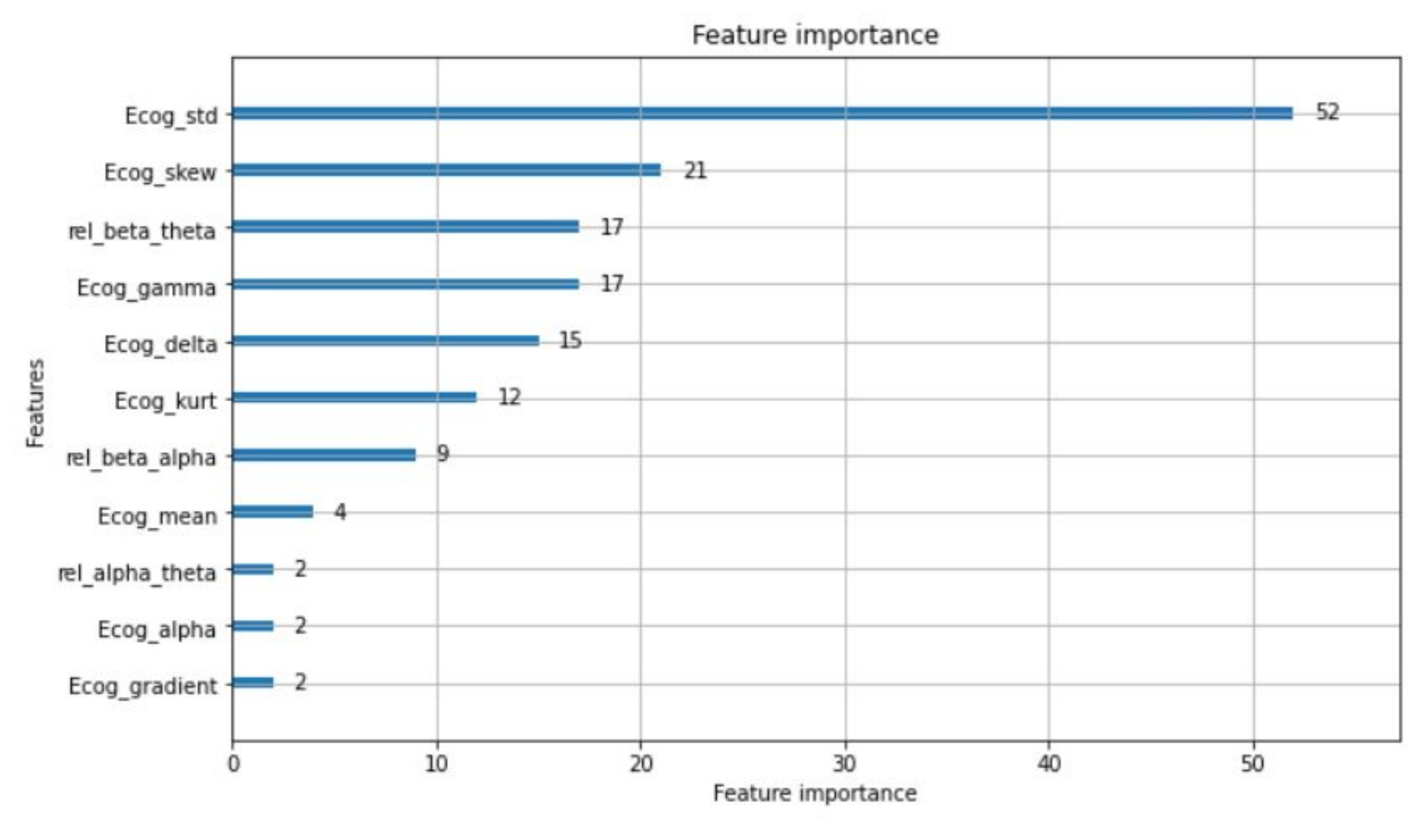
LightGBM Classifier Model

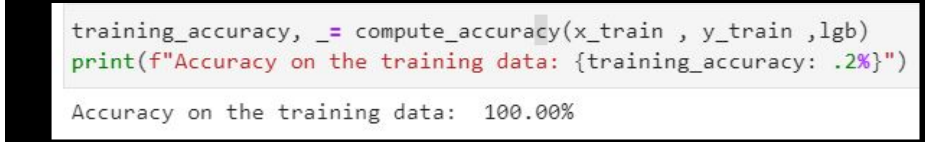
Feature Importance



LightGBM Classifier Model

Feature Importance





```
training_accuracy, _= compute_accuracy(x_train , y_train ,lgb)
print(f"Accuracy on the training data: {training_accuracy: .2%}")
```

```
Accuracy on the training data:  100.00%
```

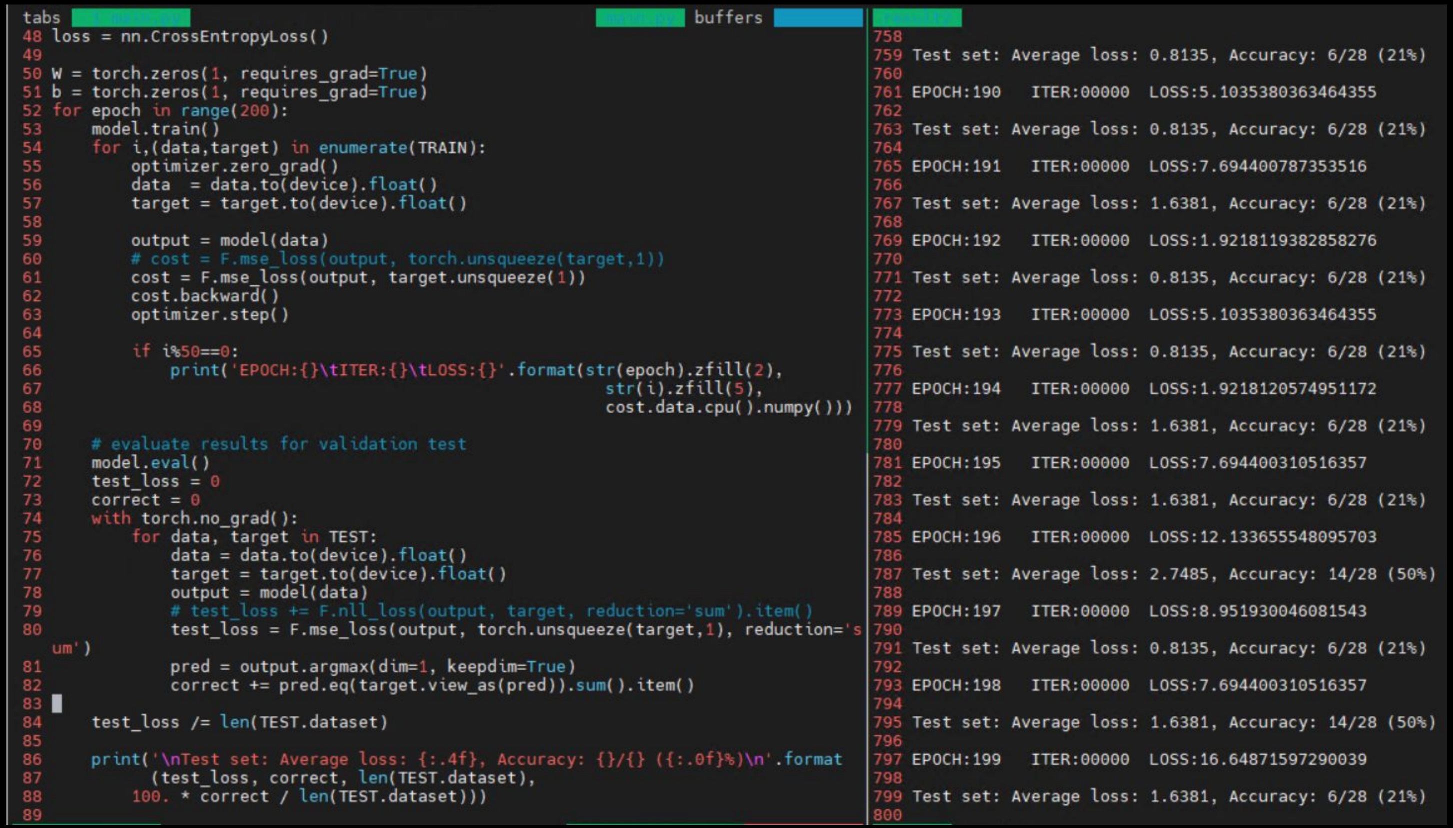
Additional Preprocessing & Trials

Standard Scaler
PCA
Unsupervised Learning

- KNN
- DBScan

Deep Learning

- CNN
- LSTM



```
loss = nn.CrossEntropyLoss()

W = torch.zeros(1, requires_grad=True)
b = torch.zeros(1, requires_grad=True)
for epoch in range(200):
    model.train()
    for i,(data,target) in enumerate(TRAIN):
        optimizer.zero_grad()
        data  = data.to(device).float()
        target = target.to(device).float()

        output = model(data)
        # cost = F.mse_loss(output, torch.unsqueeze(target,1))
        cost = F.mse_loss(output, target.unsqueeze(1))
        cost.backward()
        optimizer.step()

        if i%50==0:
            print('EPOCH:{}\tITER:{}\tLOSS:{}'.format(str(epoch).zfill(2),
                                                      str(i).zfill(5),
                                                      cost.data.cpu().numpy()))

    # evaluate results for validation test
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in TEST:
            data = data.to(device).float()
            target = target.to(device).float()
            output = model(data)
            # test_loss += F.nll_loss(output, target, reduction='sum').item()
            test_loss = F.mse_loss(output, torch.unsqueeze(target,1), reduction='sum')
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(TEST.dataset)

    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format
          (test_loss, correct, len(TEST.dataset),
        100. * correct / len(TEST.dataset)))

```

```

Test set: Average loss: 0.8135, Accuracy: 6/28 (21%)

EPOCH:190   ITER:00000  LOSS:5.1035380363464355

Test set: Average loss: 0.8135, Accuracy: 6/28 (21%)

EPOCH:191   ITER:00000  LOSS:7.694400787353516

Test set: Average loss: 1.6381, Accuracy: 6/28 (21%)

EPOCH:192   ITER:00000  LOSS:1.9218119382858276

Test set: Average loss: 0.8135, Accuracy: 6/28 (21%)

EPOCH:193   ITER:00000  LOSS:5.1035380363464355

Test set: Average loss: 0.8135, Accuracy: 6/28 (21%)

EPOCH:194   ITER:00000  LOSS:1.9218120574951172

Test set: Average loss: 1.6381, Accuracy: 6/28 (21%)

EPOCH:195   ITER:00000  LOSS:7.694400310516357

Test set: Average loss: 1.6381, Accuracy: 6/28 (21%)

EPOCH:196   ITER:00000  LOSS:12.133655548095703

Test set: Average loss: 2.7485, Accuracy: 14/28 (50%)

EPOCH:197   ITER:00000  LOSS:8.951930046081543

Test set: Average loss: 0.8135, Accuracy: 6/28 (21%)

EPOCH:198   ITER:00000  LOSS:7.694400310516357

Test set: Average loss: 1.6381, Accuracy: 14/28 (50%)

EPOCH:199   ITER:00000  LOSS:16.64871597290039

Test set: Average loss: 1.6381, Accuracy: 6/28 (21%)

```

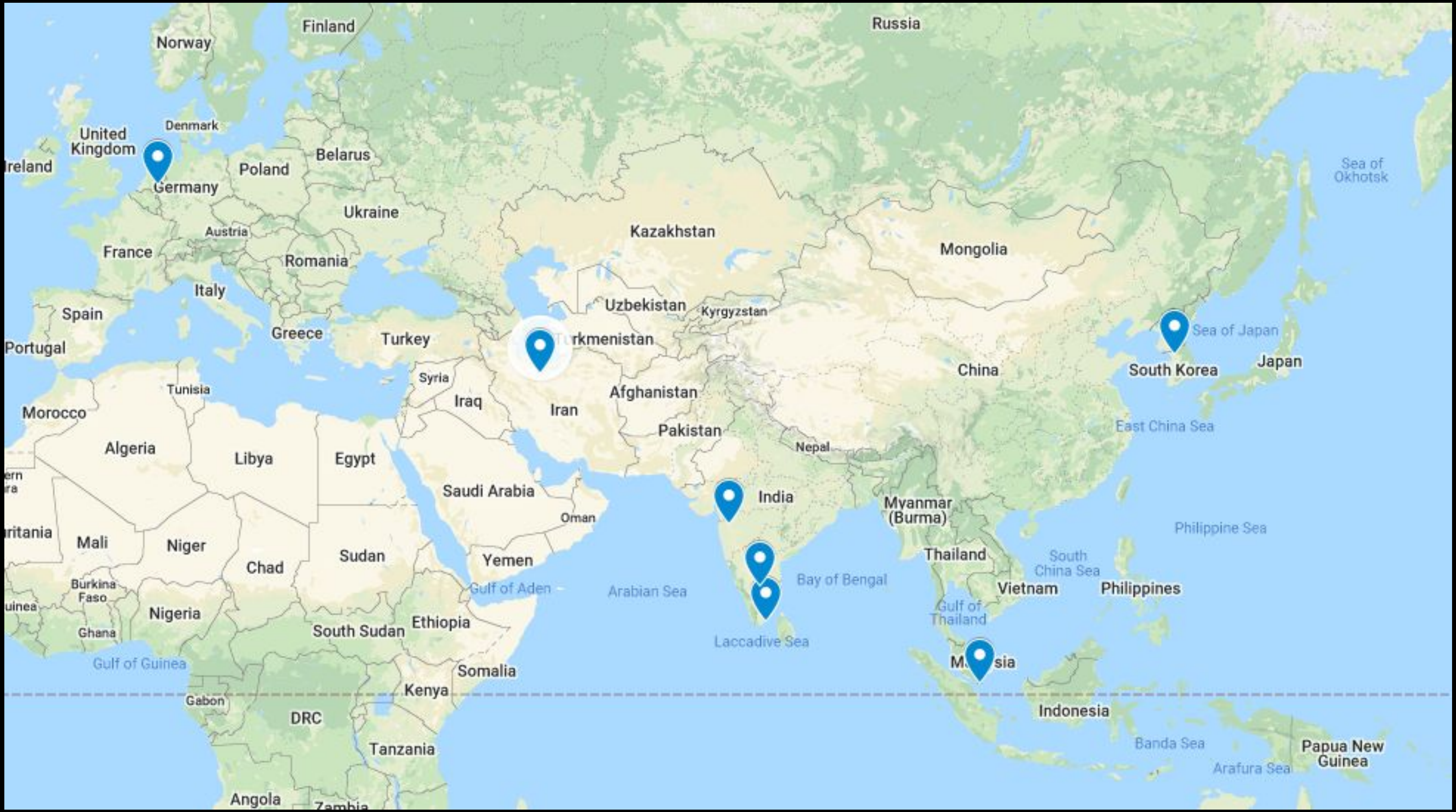
BR41N.IO

REFLECTION…

- Feature extraction is key!
- Leveraging existing deep learning models and libraries
- Working across geographies is fun despite the different time zones that separate us

GROUP PICTURE
PICS OF US WORKIN :D

Invisible
Warriors
Working
Using
Github



BR41N.IO

ECoG Hand Pose Data Analysis

A Team Has No Name
Muthu Jeyanthi, Woobean Lee,
Vivek Mohan, Amin Ranjbar,
Sayali Bachhav, Atharva Kand,
Noga Mudrik, Giulia Porro