Implementation ELEKTRokardiomatrix Method
And Tests scenarios

Supervisors:
Dr. Nasour Bagheri
Dr. Sadegh Sadeghi

The overall structure of the presentation

- Implementation phase
- Test phase
  - Classification Model → Tests 1,2
  - Datasets Combinations → Tests 3 to Test 6

Implementation phase

Implementation phase

Task: Authenticating users with their EKMs

Results of implementing the actual ELEKTRA paper for below datasets:

NSRDB

- **Epochs: 10**
- **Accuracy: 99.66%**

Notes:

- All these results are with EKMs with 5 bpf
- No contribution have been done (Just the results of main paper)

Implementation phase

```
[ ] # Train the model
    model.fit(train_x, numerical_train_labels, epochs=10, batch_size=32)
```

```
Epoch 1/10
1351/1351 [==============================] - 57s 40ms/step - loss: 0.4343 - accuracy: 0.8702
Epoch 2/10
1351/1351 [==============================] - 52s 39ms/step - loss: 0.0723 - accuracy: 0.9785
Epoch 3/10
1351/1351 [==============================] - 55s 41ms/step - loss: 0.0485 - accuracy: 0.9844
Epoch 4/10
1351/1351 [==============================] - 55s 40ms/step - loss: 0.0404 - accuracy: 0.9868
Epoch 5/10
1351/1351 [==============================] - 54s 40ms/step - loss: 0.0314 - accuracy: 0.9896
Epoch 6/10
1351/1351 [==============================] - 50s 37ms/step - loss: 0.0325 - accuracy: 0.9892
Epoch 7/10
1351/1351 [==============================] - 43s 32ms/step - loss: 0.0270 - accuracy: 0.9908
Epoch 8/10
1351/1351 [==============================] - 39s 29ms/step - loss: 0.0259 - accuracy: 0.9917
Epoch 9/10
1351/1351 [==============================] - 44s 32ms/step - loss: 0.0237 - accuracy: 0.9917
Epoch 10/10
1351/1351 [==============================] - 39s 29ms/step - loss: 0.0240 - accuracy: 0.9916
<keras.callbacks.History at 0x7fcdb047a3e0>
```

```
[ ] # Evaluate the model on the test set
    test_loss, test_accuracy = model.evaluate(test_x, numerical_test_labels, verbose=2)
    print(f'Test Loss: {test_loss:.4f}')
    print(f'Test Accuracy: {test_accuracy:.4f}')
```

```
337/337 - 3s - loss: 0.0119 - accuracy: 0.9966 - 3s/epoch - 8ms/step
Test Loss: 0.0119
Test Accuracy: 0.9966
```

Results of authenticating users of NSRDB dataset with 5 bpf in each EKM

Implementation phase

MITDB

- Epochs: 10
- Accuracy: 96.85%

```
[ ] # Train the model
    model.fit(train_x, numerical_train_labels, epochs=10, batch_size=32)

    Epoch 1/10
    542/542 [==============================] - 17s 28ms/step - loss: 1.9859 - accuracy: 0.4719
    Epoch 2/10
    542/542 [==============================] - 13s 23ms/step - loss: 0.6372 - accuracy: 0.8233
    Epoch 3/10
    542/542 [==============================] - 12s 23ms/step - loss: 0.3995 - accuracy: 0.8860
    Epoch 4/10
    542/542 [==============================] - 13s 23ms/step - loss: 0.3010 - accuracy: 0.9144
    Epoch 5/10
    542/542 [==============================] - 13s 23ms/step - loss: 0.2402 - accuracy: 0.9291
    Epoch 6/10
    542/542 [==============================] - 13s 23ms/step - loss: 0.2125 - accuracy: 0.9335
    Epoch 7/10
    542/542 [==============================] - 13s 23ms/step - loss: 0.1853 - accuracy: 0.9428
    Epoch 8/10
    542/542 [==============================] - 12s 22ms/step - loss: 0.1640 - accuracy: 0.9470
    Epoch 9/10
    542/542 [==============================] - 12s 22ms/step - loss: 0.1530 - accuracy: 0.9518
    Epoch 10/10
    542/542 [==============================] - 12s 23ms/step - loss: 0.1454 - accuracy: 0.9556
    <keras.callbacks.History at 0x7b55ddd1ae90>

[ ] # Evaluate the model on the test set
    test_loss, test_accuracy = model.evaluate(test_x, numerical_test_labels, verbose=2)
    print(f'Test Loss: {test_loss:.4f}')
    print(f'Test Accuracy: {test_accuracy:.4f}')

    134/134 - 1s - loss: 0.1056 - accuracy: 0.9685 - 908ms/epoch - 7ms/step
    Test Loss: 0.1056
    Test Accuracy: 0.9685
```

Results of authenticating users of MITDB dataset with 5 bpf in each EKM

Implementation phase

- PTBDB
  - Epochs: 30
  - Accuracy: 87.70%

```
[ ] # Train the model
    model.fit(train_x, numerical_train_labels, epochs=30, batch_size=128)

    53/53 [==============================] - 4s 67ms/step - loss: 0.1207 - accuracy: 0.9610
    Epoch 3/30
    53/53 [==============================] - 4s 71ms/step - loss: 0.1120 - accuracy: 0.9644
    Epoch 4/30
    53/53 [==============================] - 5s 97ms/step - loss: 0.1130 - accuracy: 0.9628
    Epoch 5/30
    53/53 [==============================] - 6s 121ms/step - loss: 0.1088 - accuracy: 0.9646
    Epoch 6/30
    53/53 [==============================] - 3s 65ms/step - loss: 0.1188 - accuracy: 0.9619
    Epoch 7/30
    53/53 [==============================] - 3s 65ms/step - loss: 0.1090 - accuracy: 0.9658
    Epoch 8/30
    53/53 [==============================] - 4s 85ms/step - loss: 0.1068 - accuracy: 0.9681
    Epoch 9/30
    53/53 [==============================] - 5s 96ms/step - loss: 0.1268 - accuracy: 0.9575
    Epoch 10/30
    53/53 [==============================] - 3s 65ms/step - loss: 0.1006 - accuracy: 0.9671
```

```
[ ] # Evaluate the model on the test set
    test_loss, test_accuracy = model.evaluate(test_x, numerical_test_labels, verbose=2)
    print(f'Test Loss: {test_loss:.4f}')
    print(f'Test Accuracy: {test_accuracy:.4f}')

    79/79 - 1s - loss: 1.2666 - accuracy: 0.8770 - 571ms/epoch - 7ms/step
    Test Loss: 1.2666
    Test Accuracy: 0.8770
```

Results of authenticating users of PTBDB dataset with 5 bpf in each EKM

Test phase

Test 1

Classification Model

Classification Model

In Test phase, We started with a classification model.

Task: Creating a binary classification model to classify the NSRDB dataset into two classes: "healthy" and "cvd". We used the MITDB dataset as our train set for this task.

This test aimed to explore a scenario where we used two different devices to collect our train and test samples. However, the results showed that relying on the same model is not sufficient. The performance of the model was only 70%, which is not satisfactory for this occasion.

(Assume that during the sampling of our train set, we discovered that the device malfunctioned and we had to replace it with another device to sample our test set as well. This unexpected change in devices could have affected the results and performance of the model.)

Test phase

1

Classifier Model

Result:

```
history = model.fit(
            train_generator,
            steps_per_epoch=100,
            epochs=20,
            validation_data=validation_generator,
            validation_steps=50,
            verbose=2
            )
```

Test phase

1

```
Epoch 10/20
100/100 - 10s - loss: 0.1145 - accuracy: 0.9569 - val_loss: 2.9820 - val_accuracy: 0.2788 - 10s/epoch - 98ms/step
Epoch 11/20
100/100 - 13s - loss: 0.1066 - accuracy: 0.9625 - val_loss: 3.3600 - val_accuracy: 0.4494 - 13s/epoch - 127ms/step
Epoch 12/20
100/100 - 12s - loss: 0.0850 - accuracy: 0.9694 - val_loss: 3.1712 - val_accuracy: 0.4712 - 12s/epoch - 119ms/step
Epoch 13/20
100/100 - 12s - loss: 0.0822 - accuracy: 0.9697 - val_loss: 3.2177 - val_accuracy: 0.5150 - 12s/epoch - 117ms/step
Epoch 14/20
100/100 - 13s - loss: 0.0732 - accuracy: 0.9753 - val_loss: 3.5484 - val_accuracy: 0.4737 - 13s/epoch - 129ms/step
Epoch 15/20
100/100 - 13s - loss: 0.0680 - accuracy: 0.9778 - val_loss: 3.2624 - val_accuracy: 0.4988 - 13s/epoch - 129ms/step
Epoch 16/20
100/100 - 14s - loss: 0.0557 - accuracy: 0.9789 - val_loss: 3.2195 - val_accuracy: 0.3262 - 14s/epoch - 137ms/step
Epoch 17/20
100/100 - 13s - loss: 0.0389 - accuracy: 0.9866 - val_loss: 3.6351 - val_accuracy: 0.5312 - 13s/epoch - 127ms/step
Epoch 18/20
100/100 - 11s - loss: 0.0597 - accuracy: 0.9791 - val_loss: 3.7732 - val_accuracy: 0.5088 - 11s/epoch - 107ms/step
Epoch 19/20
100/100 - 12s - loss: 0.0509 - accuracy: 0.9800 - val_loss: 4.1310 - val_accuracy: 0.4356 - 12s/epoch - 117ms/step
Epoch 20/20
100/100 - 13s - loss: 0.0459 - accuracy: 0.9827 - val_loss: 3.4883 - val_accuracy: 0.5056 - 13s/epoch - 128ms/step
```

Testing the model with users of NSRDB dataset

```
Total healthy count: 3612
Total cvd count: 1597
Accuracy: 69.34%
```

Classifier Model

Next time, we attempted to use an equal number of EKM for each user in the binary classification model. We will also test the model using the NSRDB dataset in a similar manner. Since the labels in this case represent two classes, namely "healthy" and "cvd" (cardiovascular disease), we anticipate that the number of EKM for each user will not have a significant impact. As expected, we did not observe substantial improvement.

Test phase

1

Classifier Model

This time, as you know, we trained the classifier model with 11 healthy users and 11 cardiovascular disease (CVD) patients from MITDB. However, for this experiment, we utilized 11 CVD users from PTBDB instead of the ones from MITDB. Unfortunately, the results turned out to be less than desired, with an accuracy of only around 50%.

Test phase

1

Result:

```
Epoch 10/20
100/100 - 8s - loss: 0.0165 - accuracy: 0.9959 - 8s/epoch - 77ms/step
Epoch 11/20
100/100 - 7s - loss: 0.0135 - accuracy: 0.9943 - 7s/epoch - 73ms/step
Epoch 12/20
100/100 - 7s - loss: 0.0183 - accuracy: 0.9937 - 7s/epoch - 68ms/step
Epoch 13/20
100/100 - 8s - loss: 0.0095 - accuracy: 0.9959 - 8s/epoch - 78ms/step
Epoch 14/20
100/100 - 7s - loss: 0.0130 - accuracy: 0.9962 - 7s/epoch - 70ms/step
Epoch 15/20
100/100 - 7s - loss: 0.0162 - accuracy: 0.9934 - 7s/epoch - 65ms/step
Epoch 16/20
100/100 - 7s - loss: 0.0118 - accuracy: 0.9962 - 7s/epoch - 71ms/step
Epoch 17/20
100/100 - 8s - loss: 0.0115 - accuracy: 0.9972 - 8s/epoch - 78ms/step
Epoch 18/20
100/100 - 7s - loss: 0.0095 - accuracy: 0.9962 - 7s/epoch - 71ms/step
Epoch 19/20
100/100 - 7s - loss: 0.0098 - accuracy: 0.9969 - 7s/epoch - 68ms/step
Epoch 20/20
100/100 - 7s - loss: 0.0139 - accuracy: 0.9965 - 7s/epoch - 69ms/step
```

Testing the model with users of NSRDB dataset

```
Total healthy count: 2741
Total cvd count: 2468
Accuracy: 52.62%
```

Test 2

Classification Model

Test phase

2

Datasets Combinations

Task: testing the performance of proposed model with classifying patient (users with CVD) and healthy users.

- Final dataset:
  amount of EKMs for each label (patient, healthy)
- Patient users
  MITDB (16,410 EKMs)
- Healthy users
  MITDB (5,203 EKMs)
  NSRDB (11, 207 EKMs)

Test phase

2

- **Scenario**

We reach to an agreement that another possible experiment to examine the models performance is to test if the model can classify <<users with CVD>> and <<healthy users>>.

To examine this, I used MITDB dataset first. I got healthy and patient users of the MITDB dataset and counted EKMs of them. We have 5203 EKMs for healthy user and 16410 EKMs for patient users. As you can see there is an imbalancement in these two labels. So I decided to cover that up by adding some EKMs from NSRDB dataset which all of its users are healthy. I added 11207 randomly chosen-EKMs from NSRDB dataset, so that we can have a balanced dataset.

In the next step, like before, vectorized EKMs of the final dataset and fed them to the model by proportion of 80% to 20% for train and test data, by 20 epochs.

In the end, I tested the classifier with 1000 randomly-chosen EKMs of PTBDB dataset too, to see the performance of the model more accurately.

Test phase

2

- Accuracy: 99.48%
  - **Tested with 20% of created dataset**

```
[ ] # Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32)

Epoch 1/10
1276/1276 [==============================] - 33s 25ms/step - loss: 0.2989 - accuracy: 0.8609
Epoch 2/10
1276/1276 [==============================] - 32s 25ms/step - loss: 0.1737 - accuracy: 0.9270
Epoch 3/10
1276/1276 [==============================] - 31s 24ms/step - loss: 0.1117 - accuracy: 0.9558
Epoch 4/10
1276/1276 [==============================] - 30s 24ms/step - loss: 0.0797 - accuracy: 0.9693
Epoch 5/10
1276/1276 [==============================] - 32s 25ms/step - loss: 0.0646 - accuracy: 0.9762
Epoch 6/10
1276/1276 [==============================] - 32s 25ms/step - loss: 0.0545 - accuracy: 0.9799
Epoch 7/10
1276/1276 [==============================] - 32s 25ms/step - loss: 0.0465 - accuracy: 0.9837
Epoch 8/10
1276/1276 [==============================] - 31s 25ms/step - loss: 0.0449 - accuracy: 0.9839
Epoch 9/10
1276/1276 [==============================] - 30s 23ms/step - loss: 0.0410 - accuracy: 0.9858
Epoch 10/10
1276/1276 [==============================] - 31s 24ms/step - loss: 0.0365 - accuracy: 0.9873
<keras.callbacks.History at 0x7ed31bd32e00>

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

319/319 - 2s - loss: 0.0169 - accuracy: 0.9948 - 2s/epoch - 6ms/step
Test Loss: 0.0169
Test Accuracy: 0.9948
```

Results of testing the classifier(model) which been trained by created dataset with 5 bpf in each EKM, with 20% of the created dataset EKMs

Test phase

2

- Accuracy: 82.42%
  - **Tested with 1000 EKMs of PTBDB**

```
[ ] # Evaluate the model on the test set
    test_loss, test_accuracy = model.evaluate(ptb_test_x, ptb_test_y, verbose=2)
    print(f'Test Loss: {test_loss:.4f}')
    print(f'Test Accuracy: {test_accuracy:.4f}')

    291/291 - 2s - loss: 0.8868 - accuracy: 0.8242 - 2s/epoch - 6ms/step
    Test Loss: 0.8868
    Test Accuracy: 0.8242
```

Results of testing the classifier(model) which been trained by created dataset with 5 bpf in each EKM, with 1000 randomly-chosen EKMs

Test 3

Datasets Combinations

Test phase

3

Datasets Combinations

Task: We tried to combine two datasets, NSRSB and MITDB, and split them into 20% for training and 80% for testing. The model is still being authentication using the same approach.
(This means we used a test set of the same type as the train set.)

Test phase

3

```
[ ] # Train the model
    model.fit(train_x, numerical_train_labels, epochs=10, batch_size=32)

    Epoch 1/10
    345/345 [==============================] - 7s 17ms/step - loss: 2.2761 - accuracy: 0.3903
    Epoch 2/10
    345/345 [==============================] - 7s 22ms/step - loss: 0.8561 - accuracy: 0.7372
    Epoch 3/10
    345/345 [==============================] - 6s 18ms/step - loss: 0.5491 - accuracy: 0.8300
    Epoch 4/10
    345/345 [==============================] - 9s 26ms/step - loss: 0.4002 - accuracy: 0.8752
    Epoch 5/10
    345/345 [==============================] - 6s 19ms/step - loss: 0.3060 - accuracy: 0.9018
    Epoch 6/10
    345/345 [==============================] - 7s 22ms/step - loss: 0.2616 - accuracy: 0.9144
    Epoch 7/10
    345/345 [==============================] - 6s 18ms/step - loss: 0.2221 - accuracy: 0.9283
    Epoch 8/10
    345/345 [==============================] - 7s 21ms/step - loss: 0.1963 - accuracy: 0.9387
    Epoch 9/10
    345/345 [==============================] - 6s 18ms/step - loss: 0.1818 - accuracy: 0.9409
    Epoch 10/10
    345/345 [==============================] - 7s 22ms/step - loss: 0.1637 - accuracy: 0.9483
```

```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(comb_test_x, comb_test_y, verbose=2)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

346/346 - 1s - loss: 0.4923 - accuracy: 0.9620 - 698ms/epoch - 13ms/step
Test Loss: 0.3973
Test Accuracy: 0.9620
```

Test 4

Datasets Combinations

Test phase

4

Datasets Combinations

Task: mixing all the datasets into one dataset in two ways:

- Imbalanced
- Balanced (All users has same amount of EKMs)

Notes:

- Balanceness is in term of equal amount of EKMs for each user.
- All EKMs are 5 bpf.

Test phase

4

- **Scenario**

I decided to mix all the datasets so we can see the performance of the proposed model in large number of users and in mixture of different datasets.

I conducted the experiment in Balance and Imbalance dataset in the term of amount of EKMs of each user.

- In Imbalanced experiment, I let the amount of EKMs of all users of the different datasets be the original (ex. for NSRDB users all has 3000 EKMs each).
- In balanced experiment, I found dataset with least EKMs for each user which was PTBDB, then calculate the average EKMs each user has in that of dataset (=32). Finally selected that amount of EKMs for all users of different datasets. I should mention that because of getting average amount of minimum-EKM-having dataset, our final dataset is partially balanced.

In the end, like before, vectorized EKMs of the final dataset and fed them to the model by proportion of 80% to 20% for train and test data, by 20 epochs.

Test phase

4

- Number of users in final dataset in both balance and imbalance datasets is equal to 356.

```
[ ] len(np.unique(numerical_y_mixed_labels))
356
```

Number of users in final dataset.

Test phase

4

- **Imbalanced**
  - Epochs: 10
  - Accuracy: 97.50%
- Balanceness is in term of equal amount of EKMs for each user.
- All EKMs are 5 bpf.
- Number of users in final dataset is equal to 356.

```
[ ]  # Train the model
     model.fit(X_train, y_train, epochs=10, batch_size=32)

     Epoch 1/10
     2123/2123 [==============================] - 59s 27ms/step - loss: 1.3368 - accuracy: 0.6861
     Epoch 2/10
     2123/2123 [==============================] - 57s 27ms/step - loss: 0.4754 - accuracy: 0.8736
     Epoch 3/10
     2123/2123 [==============================] - 61s 29ms/step - loss: 0.3345 - accuracy: 0.9082
     Epoch 4/10
     2123/2123 [==============================] - 61s 29ms/step - loss: 0.2611 - accuracy: 0.9248
     Epoch 5/10
     2123/2123 [==============================] - 61s 29ms/step - loss: 0.2266 - accuracy: 0.9342
     Epoch 6/10
     2123/2123 [==============================] - 59s 28ms/step - loss: 0.1962 - accuracy: 0.9418
     Epoch 7/10
     2123/2123 [==============================] - 58s 27ms/step - loss: 0.1772 - accuracy: 0.9467
     Epoch 8/10
     2123/2123 [==============================] - 57s 27ms/step - loss: 0.1590 - accuracy: 0.9524
     Epoch 9/10
     2123/2123 [==============================] - 58s 27ms/step - loss: 0.1471 - accuracy: 0.9556
     Epoch 10/10
     2123/2123 [==============================] - 59s 28ms/step - loss: 0.1391 - accuracy: 0.9572
     <keras.callbacks.History at 0x780614f88e20>

[ ]  # Evaluate the model on the unfair test set
     test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
     print(f'Test Loss: {test_loss:.4f}')
     print(f'Test Accuracy: {test_accuracy:.4f}')

     531/531 - 4s - loss: 0.0957 - accuracy: 0.9750 - 4s/epoch - 7ms/step
     Test Loss: 0.0957
     Test Accuracy: 0.9750
```

Accuracy of model with imbalance dataset

Test phase

4

- **Balanced** (All users has same amount of EKMs)
  - Epochs: 20
  - Accuracy: 87.20%
    - As we know balance dataset results are more accurate (because the dataset is balanced!).

```
Epoch 17/20
241/241 [==============================] - 7s 30ms/step - loss: 0.5785 - accuracy: 0.8256
Epoch 18/20
241/241 [==============================] - 6s 24ms/step - loss: 0.5635 - accuracy: 0.8272
Epoch 19/20
241/241 [==============================] - 8s 34ms/step - loss: 0.5206 - accuracy: 0.8402
Epoch 20/20
241/241 [==============================] - 6s 24ms/step - loss: 0.5084 - accuracy: 0.8407
<keras.callbacks.History at 0x78060e2d9330>

[ ] len(y_test)

1922

[ ] # Evaluate the model on the unfair test set
    test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
    print(f'Test Loss: {test_loss:.4f}')
    print(f'Test Accuracy: {test_accuracy:.4f}')

61/61 - 1s - loss: 0.4923 - accuracy: 0.8720 - 877ms/epoch - 14ms/step
Test Loss: 0.4923
Test Accuracy: 0.8720
```

Accuracy of model with balance dataset

Test 5

Datasets Combinations

Datasets Combinations

Task: experimenting the model with same amount of users from each dataset to set the fairness by the amount of users (with equal amount of EKMs for each user)

Test phase

5

- **Scenario**

We reach to an agreement that another possible experiment to examine the models performance is to get equal number of users with equal amount of EKMs for each individual user. I selected equal amount of users randomly so that the results will not be biased. Also, I chose equal amount of EKMs for each user randomly too for same reason as before.

Points & Notes

- As we know from before, NSRDB has the least amount of users among datasets which is 18 users; so I selected 18 users from other datasets, randomly.
- I collected all EKMs of all the selected users then calculated average amount of EKMs a user has in datasets, and the minimum average result was average amount of EKMs for PTBDB with 37 EKMs (because minimum was 4!) So I chose 37 EKMs, randomly, for all the users of different datasets.

In the end, like before, vectorized EKMs of the final dataset and fed them to the model by proportion of 80% to 20% for train and test data, by 30 epochs.

Test phase

5

- Results
  - Epochs: 30
  - Accuracy: 93.12%

```
[ ]  # Evaluate the model on the unfair test set
     test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
     print(f'Test Loss: {test_loss:.4f}')
     print(f'Test Accuracy: {test_accuracy:.4f}')

     12/12 - 0s - loss: 0.2024 - accuracy: 0.9312 - 392ms/epoch - 33ms/step
     Test Loss: 0.2024
     Test Accuracy: 0.9312
```

Accuracy of model with same amount of users from each dataset. All users also have same amount of EKMs.

Test 6

Datasets Combinations

Test phase

6

Datasets Combinations

Task: experimenting the model with minimum and low amounts of EKMs and same amount of users from each dataset.

- **Scenario**

  We decided to examine the model with minimum amount of EKMs a user can have which is 4 EKMs to see the performance of the model with very low amount of EKMs.

  Also I repeated the experiment with 10, 15 and 20 randomly-chosen EKMs for each user.

  In the end, like before, vectorized EKMs of the final dataset and fed them to the model by proportion of 80% to 20% for train and test data, by 80 epochs.

Test phase

6

```
[27] average_EKMs_amount_each_user = 4
```

```
[28] # Getting random EKMs from NSRDB's users' EKMs
     NSRDB_EKMs = []
     for user in NSRDB_users_EKMs_dict.keys():
       NSRDB_EKMs = NSRDB_EKMs + random.sample(NSRDB_users_EKMs_dict[user], \
                    average_EKMs_amount_each_user)
```

Setting the amount of EKMs for each user equal to 4.

Test phase

6

Results:

EKMs amount: 4

- Epochs80
- Accuracy 54.55%
- Loss 1.62

```
[57] # Train the model
     model.fit(X_train, y_train, epochs=80, batch_size=32)

     6/6 [==============================] - 0s 23ms/step - loss: 0.2875 - accuracy: 0.9419
     Epoch 53/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2452 - accuracy: 0.9535
     Epoch 54/80
     6/6 [==============================] - 0s 20ms/step - loss: 0.2392 - accuracy: 0.9477
     Epoch 55/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2433 - accuracy: 0.9302
     Epoch 56/80
     6/6 [==============================] - 0s 24ms/step - loss: 0.2487 - accuracy: 0.9302
     Epoch 57/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2251 - accuracy: 0.9535
     Epoch 58/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.1858 - accuracy: 0.9477
     Epoch 59/80
     6/6 [==============================] - 0s 22ms/step - loss: 0.2243 - accuracy: 0.9535
     Epoch 60/80
```

```
[58] # Evaluate the model on the unfair test set
     test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
     print(f'Test Loss: {test_loss:.4f}')
     print(f'Test Accuracy: {test_accuracy:.4f}')

     2/2 - 0s - loss: 1.6232 - accuracy: 0.5455 - 158ms/epoch - 79ms/step
     Test Loss: 1.6232
     Test Accuracy: 0.5455
```

Setting the amount of EKMs for each user equal to 4

Test phase

6

EKMs amount: 10

- Epochs80
- Accuracy 83.33%
- Loss 0.63

```
[57] # Train the model
     model.fit(X_train, y_train, epochs=80, batch_size=32)

     6/6 [==============================] - 0s 23ms/step - loss: 0.2875 - accuracy: 0.9419
     Epoch 53/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2452 - accuracy: 0.9535
     Epoch 54/80
     6/6 [==============================] - 0s 20ms/step - loss: 0.2392 - accuracy: 0.9477
     Epoch 55/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2433 - accuracy: 0.9302
     Epoch 56/80
     6/6 [==============================] - 0s 24ms/step - loss: 0.2487 - accuracy: 0.9302
     Epoch 57/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2251 - accuracy: 0.9535
     Epoch 58/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.1858 - accuracy: 0.9477
     Epoch 59/80
     6/6 [==============================] - 0s 22ms/step - loss: 0.2243 - accuracy: 0.9535
     Epoch 60/80
```

```
# Evaluate the model on the unfair test set
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

4/4 - 0s - loss: 0.6301 - accuracy: 0.8333 - 164ms/epoch - 41ms/step
Test Loss: 0.6301
Test Accuracy: 0.8333
```

Setting the amount of EKMs for each user equal to 10.

Test phase

6

EKMs amount: 15

- Epochs80
- Accuracy 92.50%
- Loss 0.57

```
[57] # Train the model
     model.fit(X_train, y_train, epochs=80, batch_size=32)

     6/6 [==============================] - 0s 23ms/step - loss: 0.2875 - accuracy: 0.9419
     Epoch 53/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2452 - accuracy: 0.9535
     Epoch 54/80
     6/6 [==============================] - 0s 20ms/step - loss: 0.2392 - accuracy: 0.9477
     Epoch 55/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2433 - accuracy: 0.9302
     Epoch 56/80
     6/6 [==============================] - 0s 24ms/step - loss: 0.2487 - accuracy: 0.9302
     Epoch 57/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2251 - accuracy: 0.9535
     Epoch 58/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.1858 - accuracy: 0.9477
     Epoch 59/80
     6/6 [==============================] - 0s 22ms/step - loss: 0.2243 - accuracy: 0.9535
     Epoch 60/80
```

```
# Evaluate the model on the unfair test set
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

5/5 - 0s - loss: 0.5792 - accuracy: 0.9250 - 80ms/epoch - 16ms/step
Test Loss: 0.5792
Test Accuracy: 0.9250
```

Setting the amount of EKMs for each user equal to 15.

Test phase

6

EKMs amount: 20

- Epochs 80
- Accuracy 95.11%
- Loss 0.23

```
[57] # Train the model
     model.fit(X_train, y_train, epochs=80, batch_size=32)

     6/6 [==============================] - 0s 23ms/step - loss: 0.2875 - accuracy: 0.9419
     Epoch 53/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2452 - accuracy: 0.9535
     Epoch 54/80
     6/6 [==============================] - 0s 20ms/step - loss: 0.2392 - accuracy: 0.9477
     Epoch 55/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2433 - accuracy: 0.9302
     Epoch 56/80
     6/6 [==============================] - 0s 24ms/step - loss: 0.2487 - accuracy: 0.9302
     Epoch 57/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.2251 - accuracy: 0.9535
     Epoch 58/80
     6/6 [==============================] - 0s 21ms/step - loss: 0.1858 - accuracy: 0.9477
     Epoch 59/80
     6/6 [==============================] - 0s 22ms/step - loss: 0.2243 - accuracy: 0.9535
     Epoch 60/80
```

```
# Evaluate the model on the unfair test set
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

9/9 - 0s - loss: 0.2323 - accuracy: 0.9511 - 306ms/epoch - 34ms/step
Test Loss: 0.2323
Test Accuracy: 0.9511
```

Setting the amount of EKMs for each user equal to 20.

Thanks For Your Attention.