Real-time Confusion Detection based on mouse pattern

Contents

1. Introduction to the problem

We are aiming to improve student's lab experience by creating a smart lab that can access lab components and/or simulations from home on LMS.

One of the features we aimed to achieve is to detect if students are confused in order to improve their experience by giving them AI-based assistance. However, detecting students' confusion is a real challenge without seeing the student's reaction or face.

In this document we discuss how we could train an AI model that detects if a student is confused based on his mouse coordinates

2. Data Collection and Format

Mouse data was collected from the website by mocking experiment sessions with and without confusion patterns.

We found that we could get some helpful features from the website other than just the coordinates such as:

- **Timestamp**: shows the time sample (incremented with 50ms each).
- **Clicked:** shows if the student clicked on a clickable button or not (users tend to click more clicks cluelessly when confused)
- **isConfused:** which is the label we are going to generate in order to achieve supervised learning.

Labeling data was real-time by toggling using a certain hot key to be set 1 confused or 0 not confused.

Data format: Recorded Session data was converted from .json to .csv format that arrived as follows

Sample:

[t(ms), x, y, clicked, isConfused]
50,328,177,0,1
100,328,177,0,1
150,328,177,0,1
200,328,177,0,1

3. Feature Engineering & Data Preprocessing

In order to help the model detect patterns easier, we need to do feature engineering to extract some more features.

Which features to extract was the real question here, we started searching for studies or research papers on close topics that might help us make the decision. After trial and monitoring different model behaviors, we finally extracted the following features:

- **distance covered**: measures distance covered per timestamp(50ms)
- **cursor_speed**: Calculates the change in distance per timestamp
- **acceleration**: Changes of velocity (inconsistency usually means confusion)
- **idle_time**: accumulates the time when the coordinates didn't change (usually more idle time indicates confusion)
- **movement_angle**: measures the angle between to lines formed between each 3 different points (more angles mean less straightness towards the target which indicates confusion)
- **angle_label**: Classifying angles each 30 degrees into labels from 0 to 5 in order to categorize large changes vs small changes
- **prev_angle_label**: Indicates the previous angle label in order to help the model notice the change (large vs small changes)

Preprocessing also included **Scaling** (if needed), **label encoding**, and **creating sequences** for time-series (RNN-model)

4. Model Selection & Training

Choosing the model is never a straightforward task. We studied suitable models for this problem which were:

- **Logistic Regression**
- **SVM**
- **XGBoost**
- **RNN or LSTMS**

We started by splitting data into train and test using train_test_split.

Then fit each model of these to the train data before evaluating its performance.

5. Model Evaluation & metrics

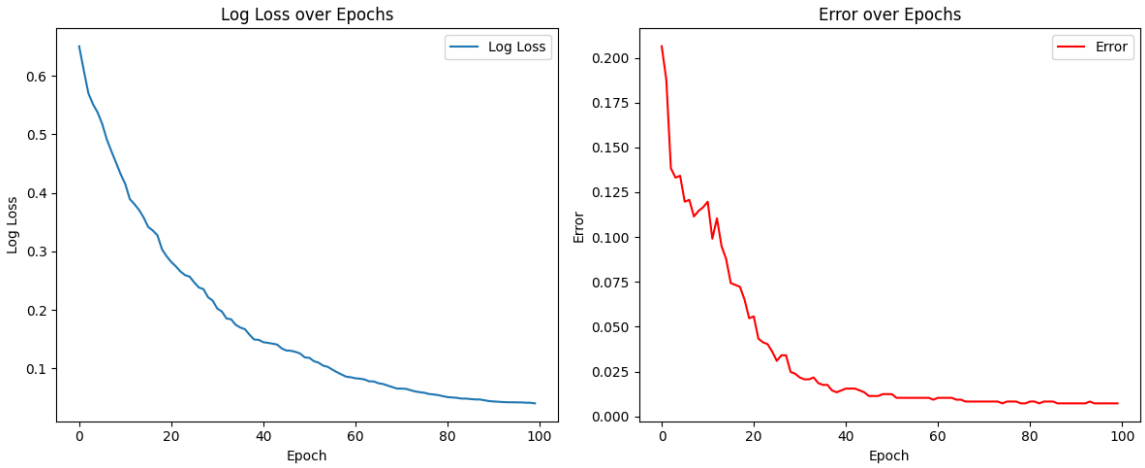
After trying multiple feature refinement for each of these models we reached final accuracies on test data evaluation to be:

- **Logistic Regression:** 0.6842105263157895
- **SVM: 0.**7079463364293086
- **XGBoost: 0.9927760577915377**
- **LSTM:** accuracy: 0.6614

XGBoost was clearly outperforming the other models, so we decided to choose it as our Machine Learning model and to also compare it to our only deep learning LSTM model.

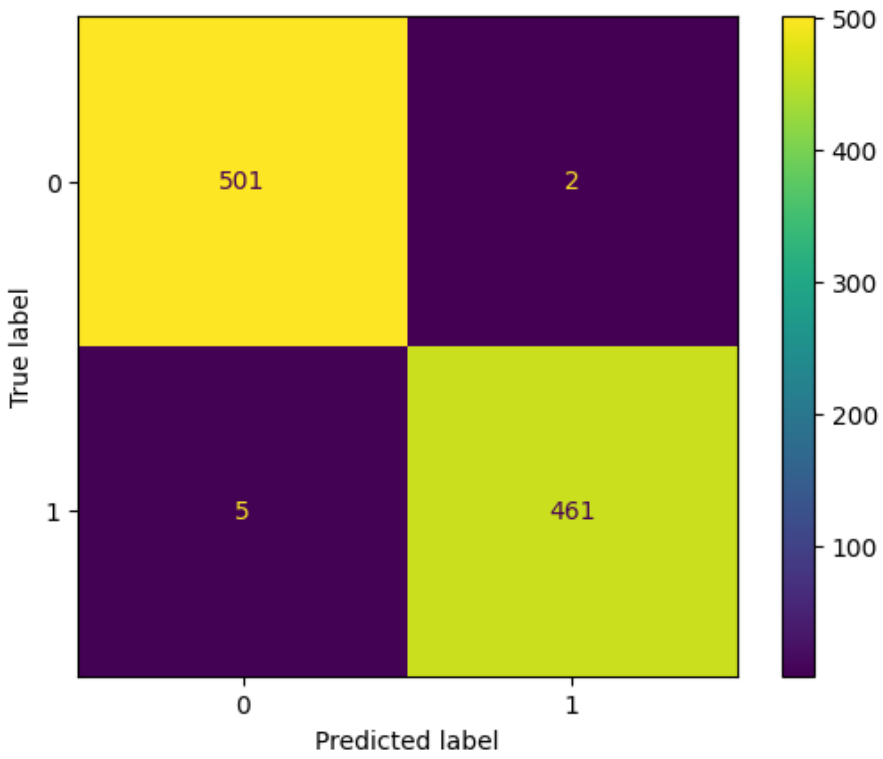
XGBoost: Notebook(XGB-99%-Confusion)

- **Training validation**:



We notice that loss and error were decreasing which is a good indication

- **Confusion Matrix:**

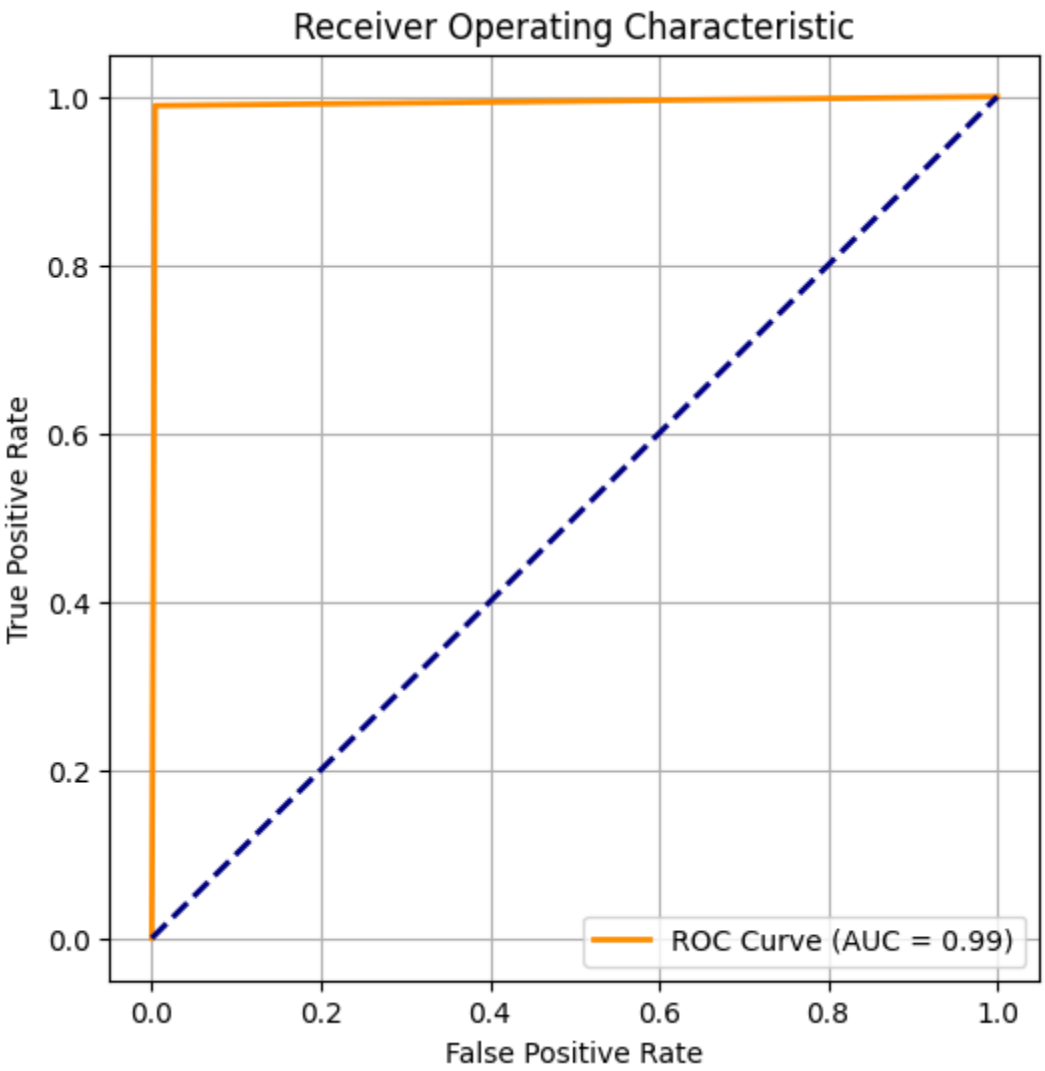


For a 1000 points of test data, only 7 were wrongly classified

- **F2_score:**
- F2 Score: 0.9905457670820799

  Used f2 score as false negatives (not detecting confusion) are much worse than false positives.
  0.99 score is close to perfection.

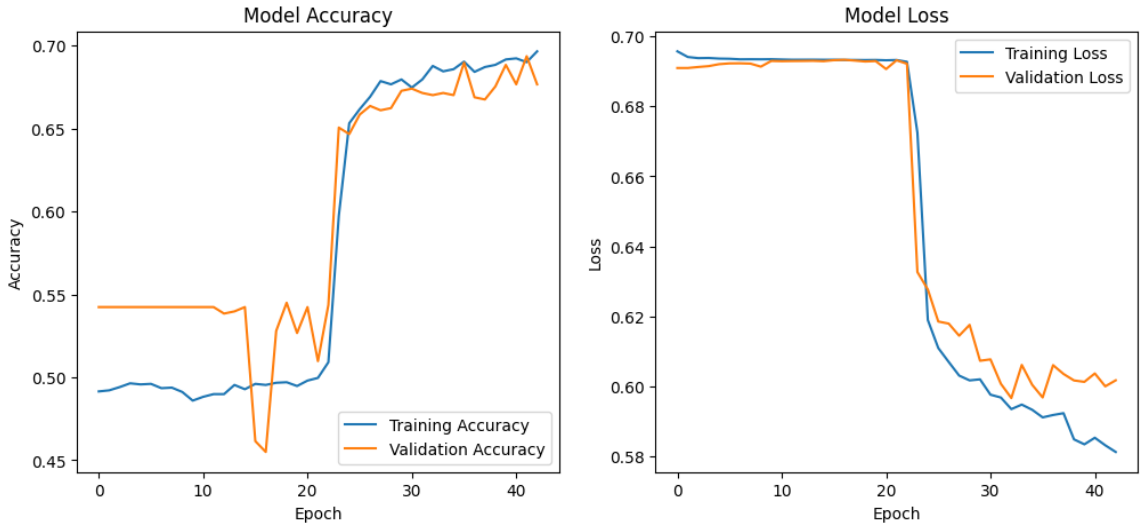
- **ROC curve:**



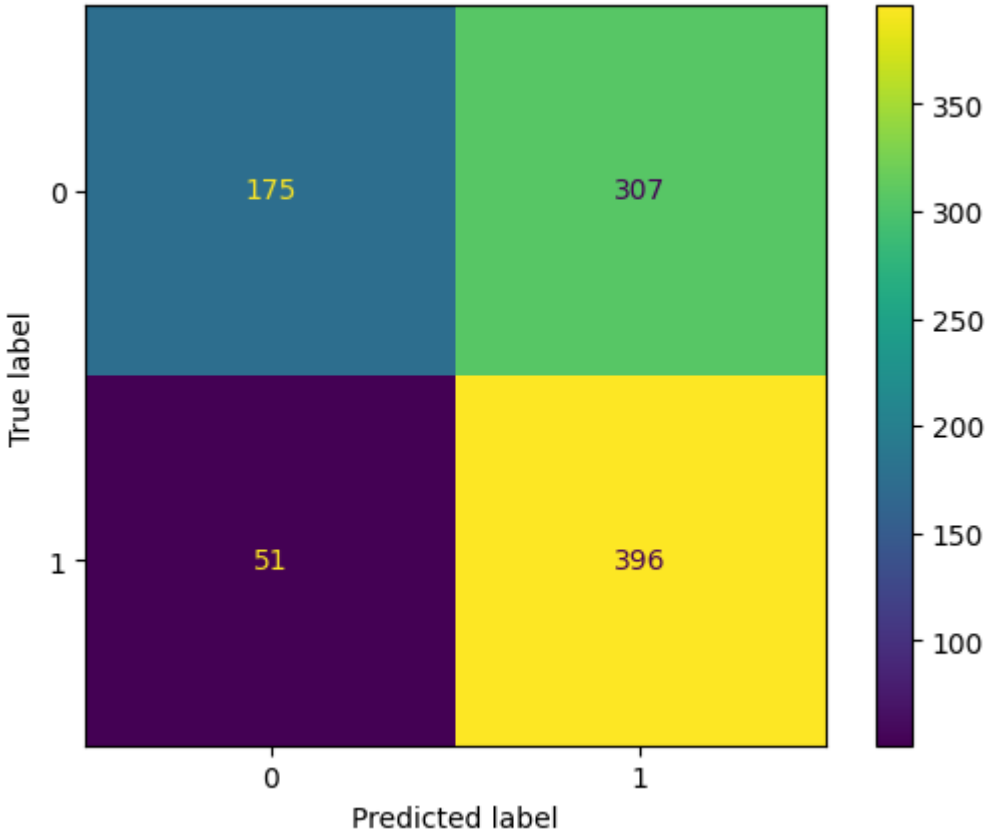
Close to 1 AUC means that the model is close to perfection on classifying confusion

LSTM: Notebook([LSTM-Confusion]())

- **Training validation**:

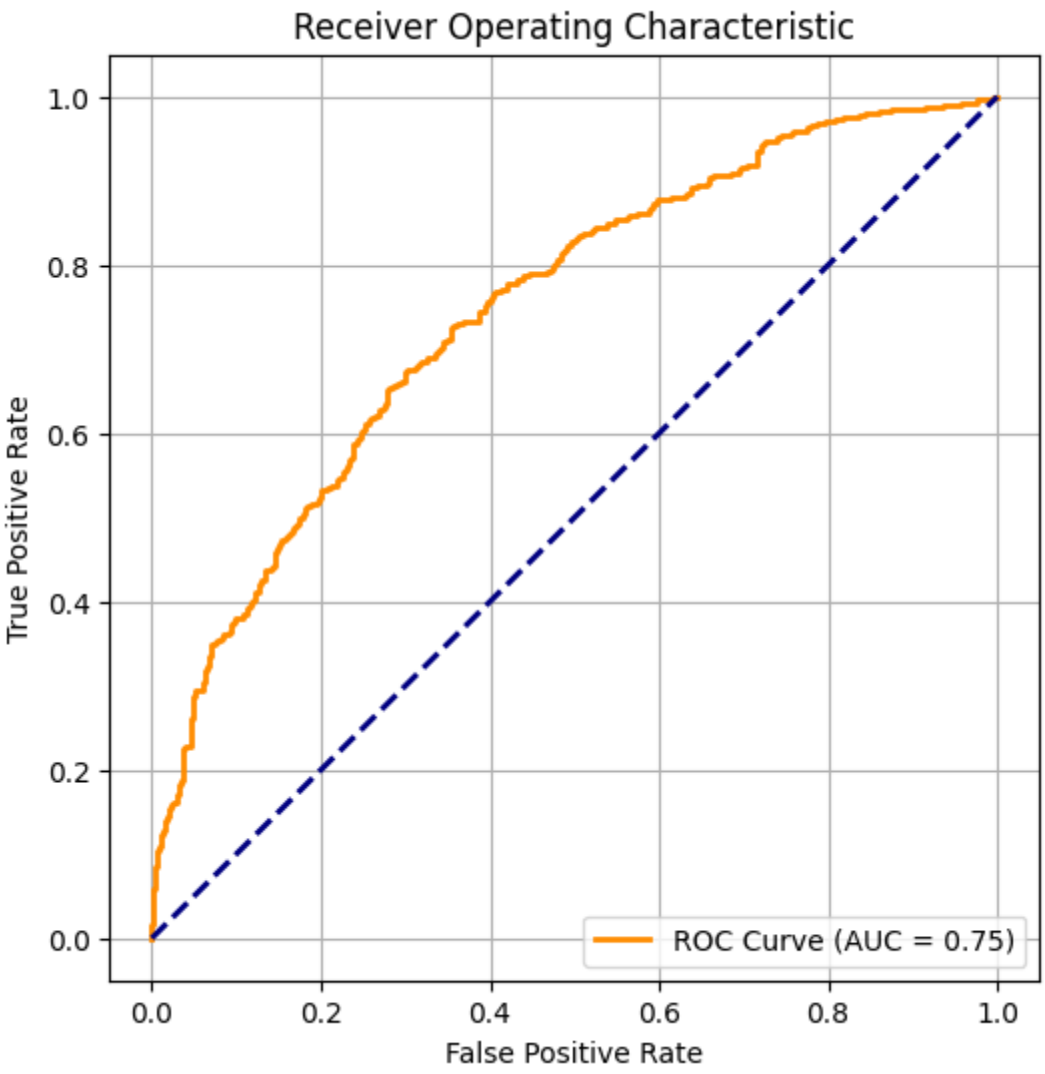


- **Confusion Matrix:**



- **F2_score:**
- F2 Score: 0.7948615014050583

- **ROC curve:**



Possible reasons why XGBoost outperformed LSTM:

- **Less Data points:** keeping in mind that we only have 1 demo experiment, we couldn't get much realistic data which is favorable for deep learning in general.
- **Unclear patterns**
  In order to increase our datapoints and detect micro patterns we sampled it each 50 ms. However, this lead sometimes to points having the same coordinates as the cursor didn't move fast

enough to detect which affects RNNs a lot apparently if change doesn't happen frequently on consecutive datapoints.

6. Final Application Workflow

After choosing XGBoost model we created an app that uses fast-api in order to call the model for classification.

The app handles the flow as following:

1. Detect user movement and upload CSV every 10 s (sampled each 50ms)
2. Data preprocessing and feature extraction (same flow of training)
3. Model prediction (penalizing false negatives by setting threshold > 0.3 majority vote)
4. Output: Prediction array, probabilities, and confusion decision
5. Taking action of AI-assistance based on confusion

7. Conclusion and Future Improvements

The system effectively identifies confused users based on mouse behavior. Future improvements may include real-time tracking, webcam integration for facial emotion recognition, and training with larger, more diverse datasets.

Huggingface Deployment: [Mouse-based-confusion-detector]