CSCA 5642 Introduction to Deep Learning Final Project

Hironari Saito

Catch Me If You Can ("Alice")

Intruder Detection through Webpage Session Tracking

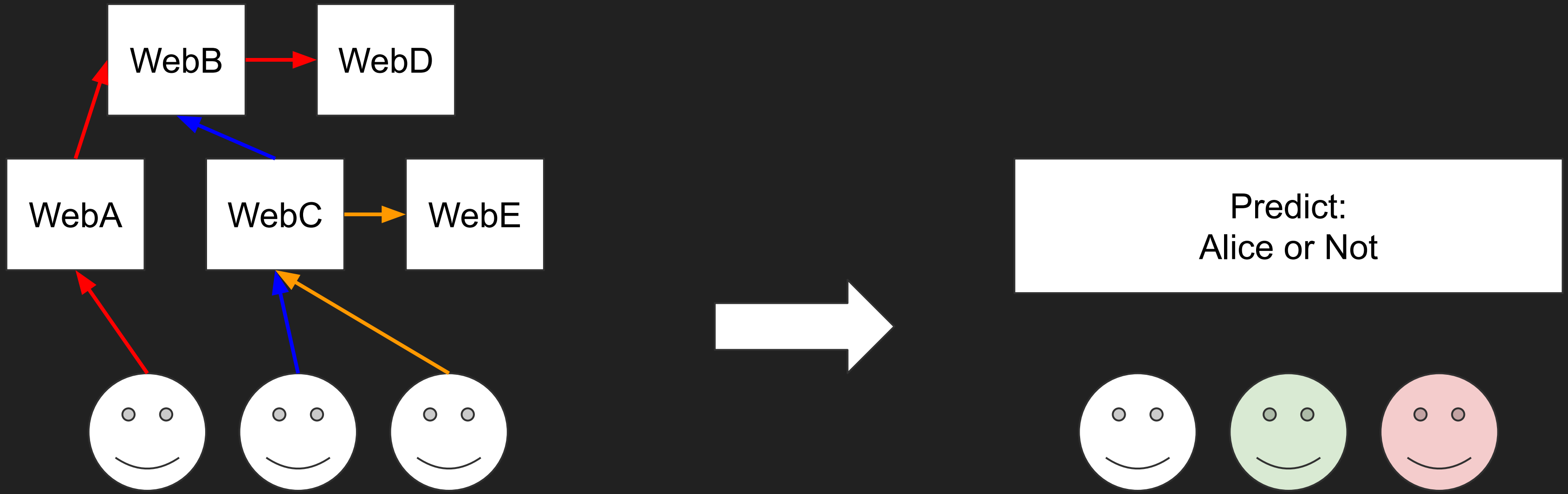
From Kaggle

Overview

- Problem
- Data
- Features
- Models
- Results
- Conclusion

Problem

This project addresses the problem of Web-User identification, specifically focusing on determining whether a given sequence of web page accesses corresponds to a particular user (Alice).



Data: Overview

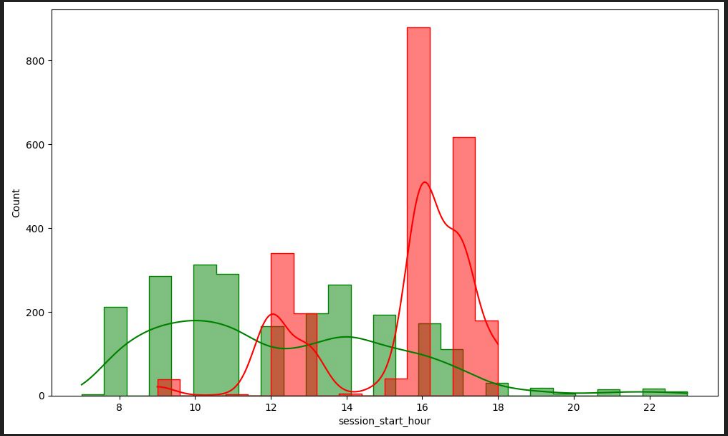
The data is obtained from the Blaise Pascal University proxy servers and is referenced in the paper "A Tool for Classification of Sequential Data" by Giacomo Kahn, Yannick Loiseau, and Olivier Raynaud. The dataset contains session_id, site1-10, time1-10, and target variables. While site1 and time1 are always present, site2-10 and time2-10 are stored optionally.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253561 entries, 0 to 253560
Data columns (total 22 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   session_id  253561 non-null  int64
 1   site1       253561 non-null  int64
 2   time1       253561 non-null  object
 3   site2       250098 non-null  float64
 4   time2       250098 non-null  object
 5   site3       246919 non-null  float64
 6   time3       246919 non-null  object
 7   site4       244321 non-null  float64
 8   time4       244321 non-null  object
 9   site5       241829 non-null  float64
 10  time5       241829 non-null  object
 11  site6       239495 non-null  float64
 12  time6       239495 non-null  object
 13  site7       237297 non-null  float64
 14  time7       237297 non-null  object
 15  site8       235224 non-null  float64
 16  time8       235224 non-null  object
 17  site9       233084 non-null  float64
 18  time9       233084 non-null  object
 19  site10      231052 non-null  float64
 20  time10      231052 non-null  object
 21  target      253561 non-null  int64
dtypes: float64(9), int64(3), object(10)
memory usage: 42.6+ MB
```

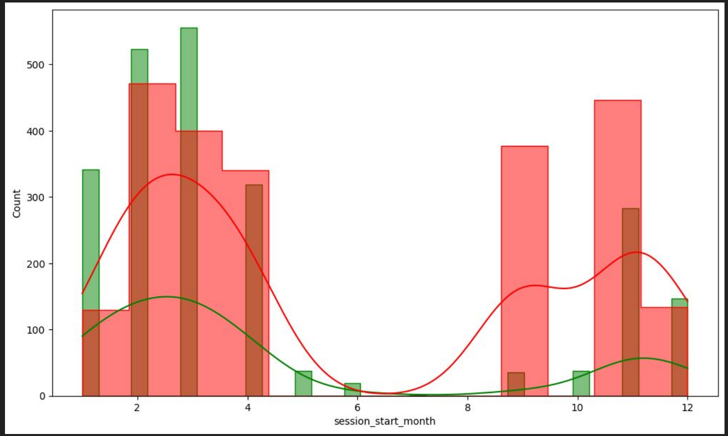
Features: Time Operation

We can create time-based features using session start time, end time, and session length to analyze user activity patterns over time.

- session start time
- session end time
- session length
- session start year, month, day, hour, minute, second, weekday
- is weekend
- midnight, morning, midday evening activity
- month start/end
- quarter start/end
- year start/end
- etc



session start time

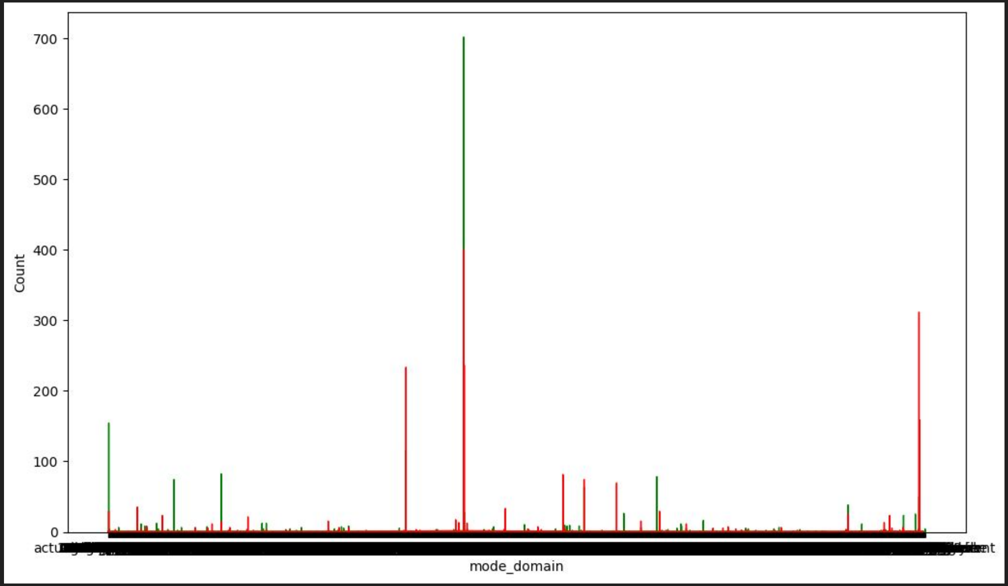


session start month

Features: Site Operation

We can create features related to the site by using the dictionary corresponding to site IDs.

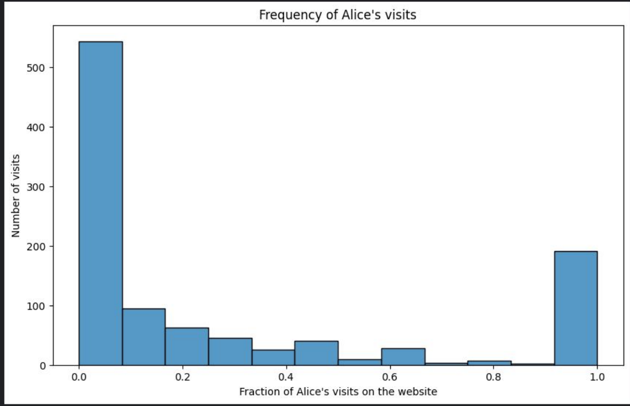
- number of sites
- avg_domain_name_length
- mode_tld
- mode_domain

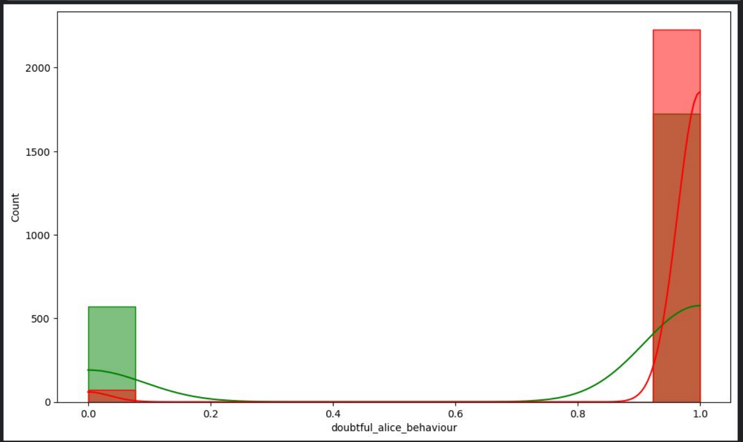


Features: User activity Operation

Sequential Pattern Mining is applied to identify frequent patterns in Alice's activity, and if suspicious behavior is detected, it is marked as 1 in the feature set. Additionally, since the user accesses a limited number of sites, the ratio of visits to those sites is calculated, and a threshold is set to create a corresponding feature.

- doubtful_alice_behaviour
- alice_site_1 (threshold: 0.01)
- alice_site_2 (threshold: 0.05)
- alice_site_3 (threshold: 0.1)
- alice_site_4 (threshold: 0.3)
- alice_site_5 (threshold: 0.5)

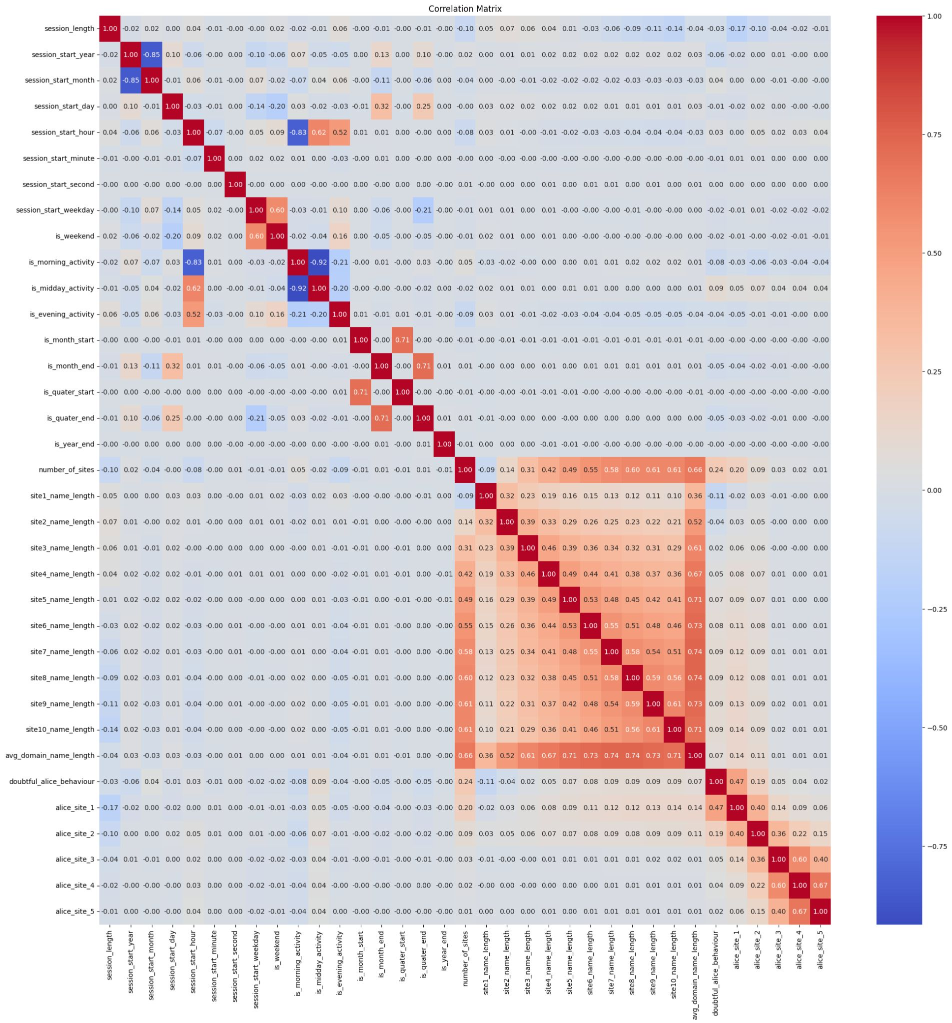




Data: Correlation Matrix

A correlation matrix was created for the features we developed. While some strong correlations were observed, it was decided to proceed with using the features as they are, given that we are utilizing deep learning models.

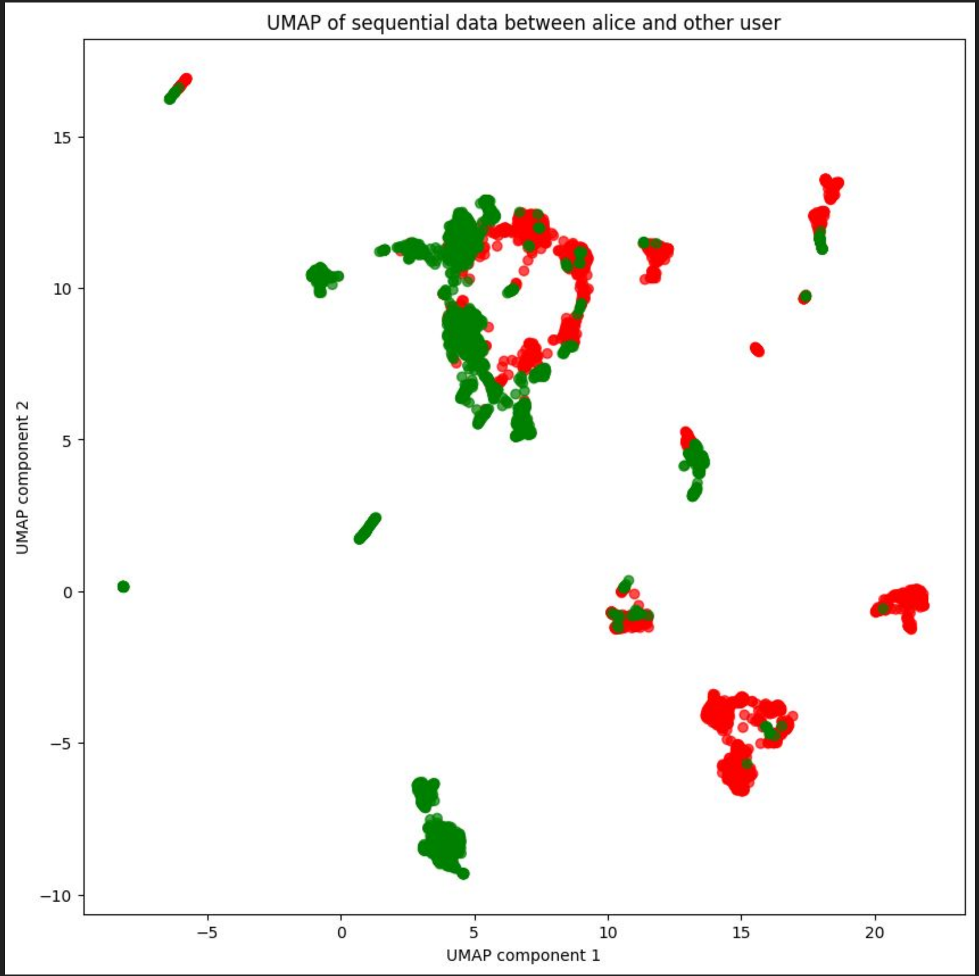
Class Label: imbalance
Dimensionality: 37 features



Data: UMAP

The data was balanced based on the label, and UMAP was used for visualization.

The results, shown in the table on the left, suggest that the features indicate a certain consistency in Alice's behavioral patterns.



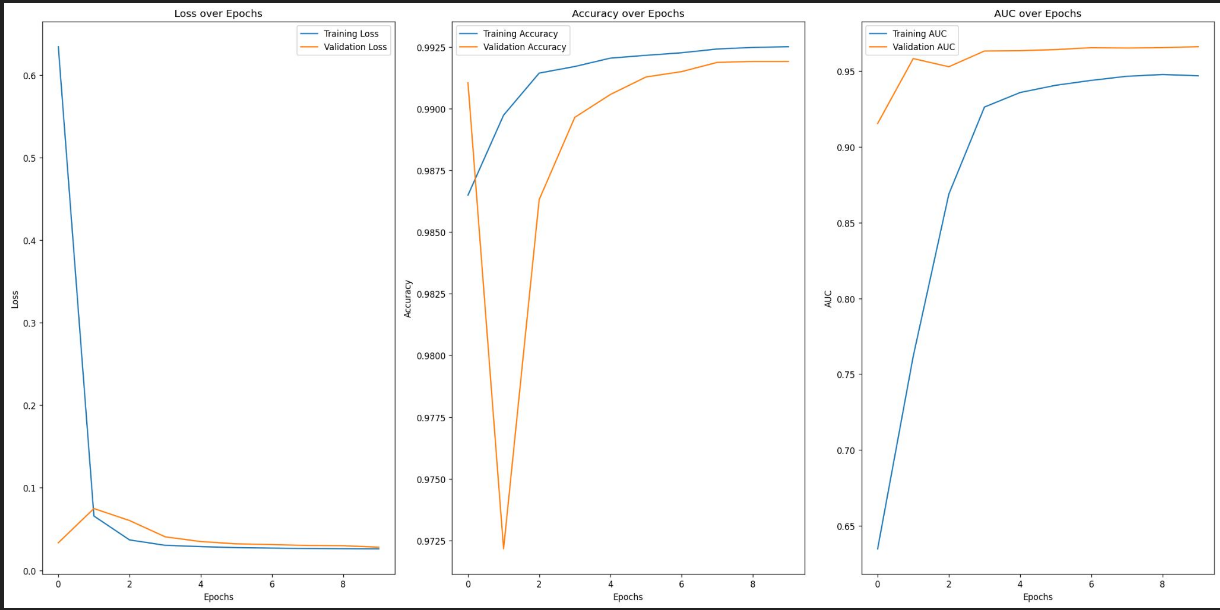
Models: Dense Layer-Only

Model:
Input(
  Embedding(mode_tld),
  Embedding(mode_domain),
  other_features)
-> Dense(64, relu)
-> Dense(32, relu)
-> Dense(1, sigmoid)
Exec:

- epochs=10,
- batch_size=32

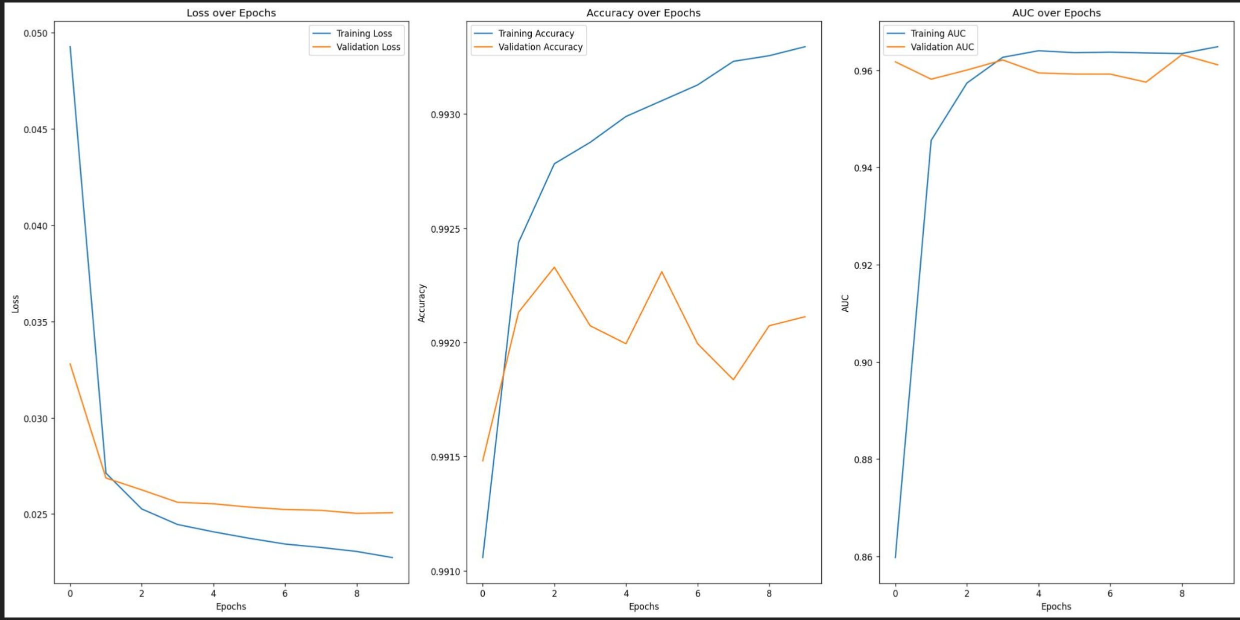
Optimizer: Adam
Metrics:

- lossbinary_crossentropy
- accuracy
- auc



Models: CNN Model

Model:
Input(
Embedding(mode_tld),
Embedding(mode_domain),
other_features)
-> CNN (64)
-> Dense(64, relu)
-> Dense(32, relu)
-> Dense(1, sigmoid)
Exec:
- epochs=10,
- batch_size=32
Optimizer: Adam
Metrics:
- lossbinary_crossentropy
- accuracy
- auc



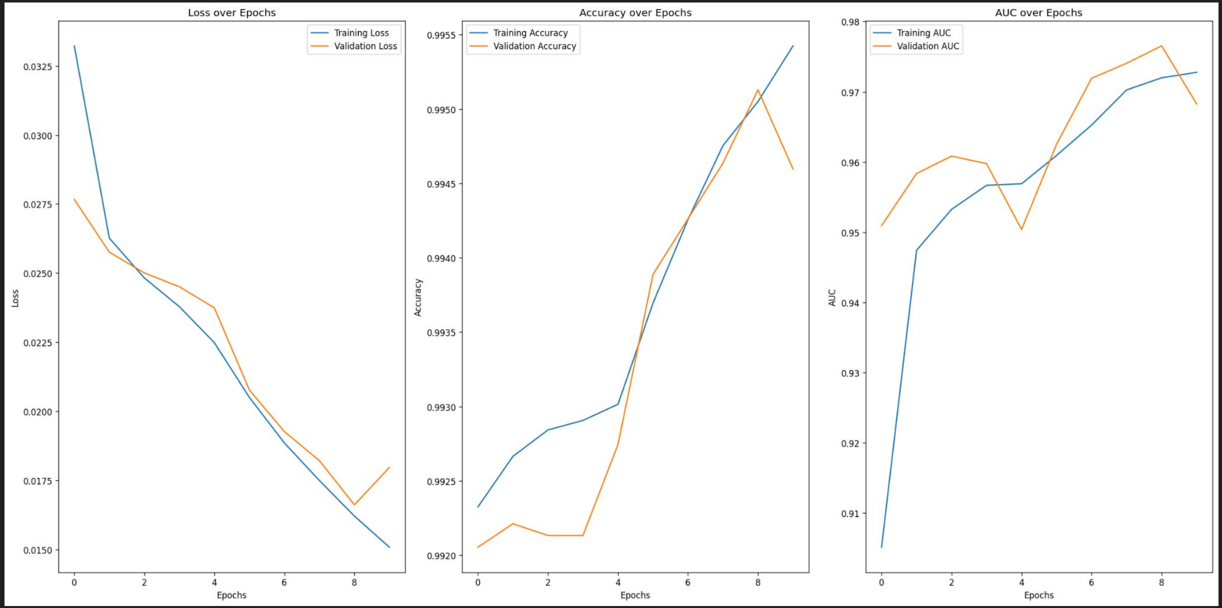
Models: LSTM Model

Model:
Input(
  Embedding(mode_tld),
  Embedding(mode_domain),
  other_features)
-> LSTM (64)
-> Dense(64, relu)
-> Dense(32, relu)
-> Dense(1, sigmoid)
Exec:

- epochs=10,
- batch_size=32

Optimizer: Adam
Metrics:

- lossbinary_crossentropy
- accuracy
- auc



Models: Tuned Dense Layer only

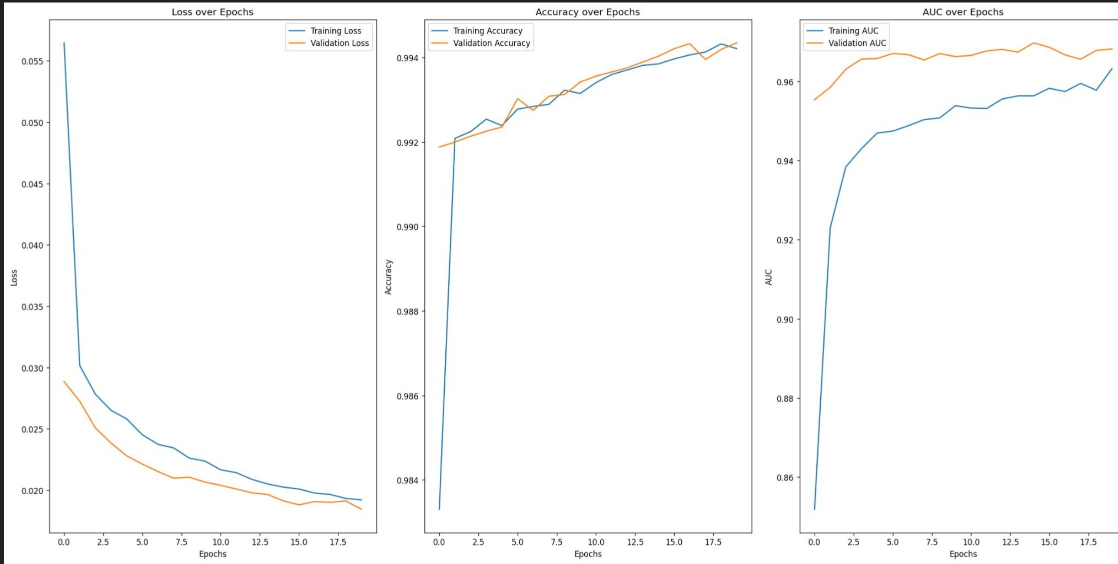
Standardize:
- all features will be standardized

Model:
Input
-> Dense(64, relu)
-> BatchNormalization -> Dropout(0.3)
-> Dense(32, relu)
-> BatchNormalization -> Dropout(0.3)
-> Dense(1, sigmoid)

Exec:
- epochs=20,
- batch_size=16

Optimizer: Adamax

Metrics:
- lossbinary_crossentropy
- accuracy
- auc



Models: Tuned CNN Layer only

Standardize:

- all features will be standardized

Model:
Input
-> CNN(64) -> BatchNormalization
-> Dense(64, relu)
-> BatchNormalization -> Dropout(0.3)
-> Dense(32, relu)
-> BatchNormalization -> Dropout(0.3)
-> Dense(1, sigmoid)

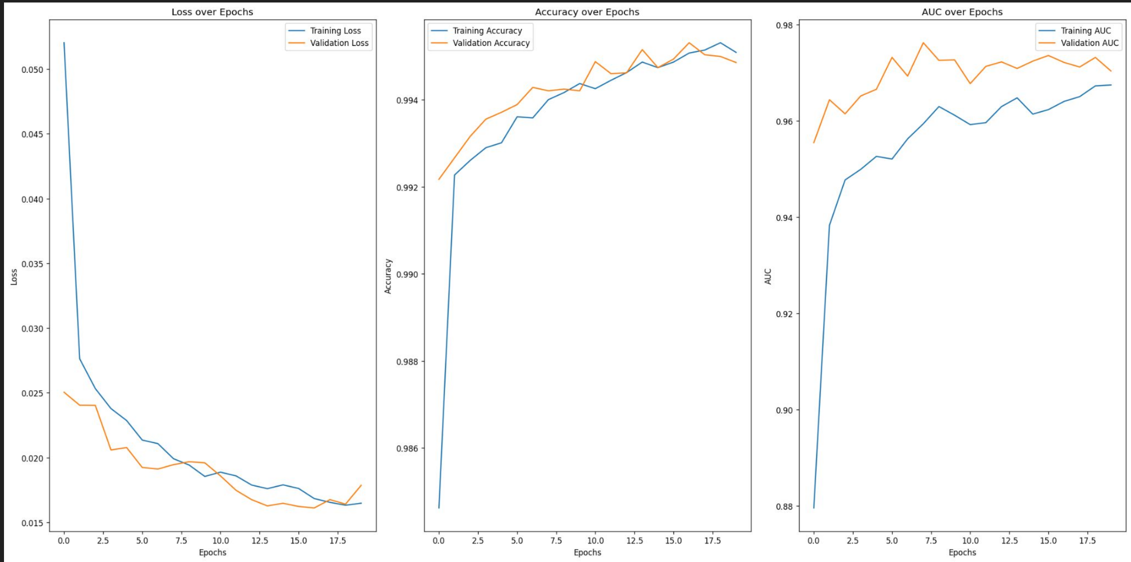
Exec:

- epochs=20,
- batch_size=16

Optimizer: Adamax

Metrics:

- lossbinary_crossentropy
- accuracy
- auc



Models: Tuned LSTM Layer only

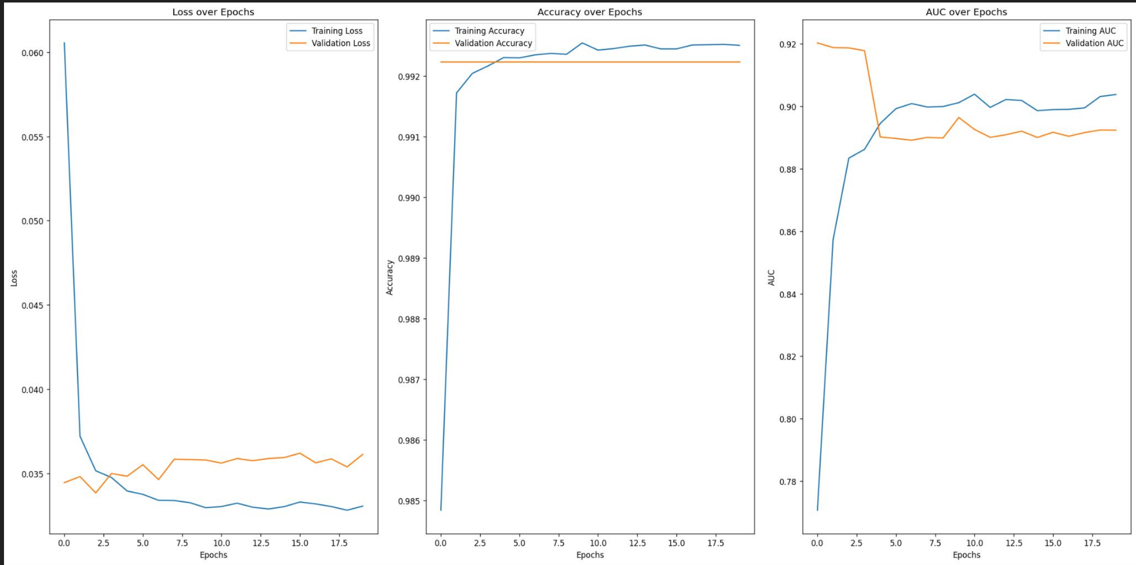
Standardize:
- all features will be standardized

Model:
Input
-> LSTM(64, recurrent_dropout=0.3)
-> Dense(64, relu)
-> BatchNormalization -> Dropout(0.3)
-> Dense(32, relu)
-> BatchNormalization -> Dropout(0.3)
-> Dense(1, sigmoid)

Exec:
- epochs=20,
- batch_size=16

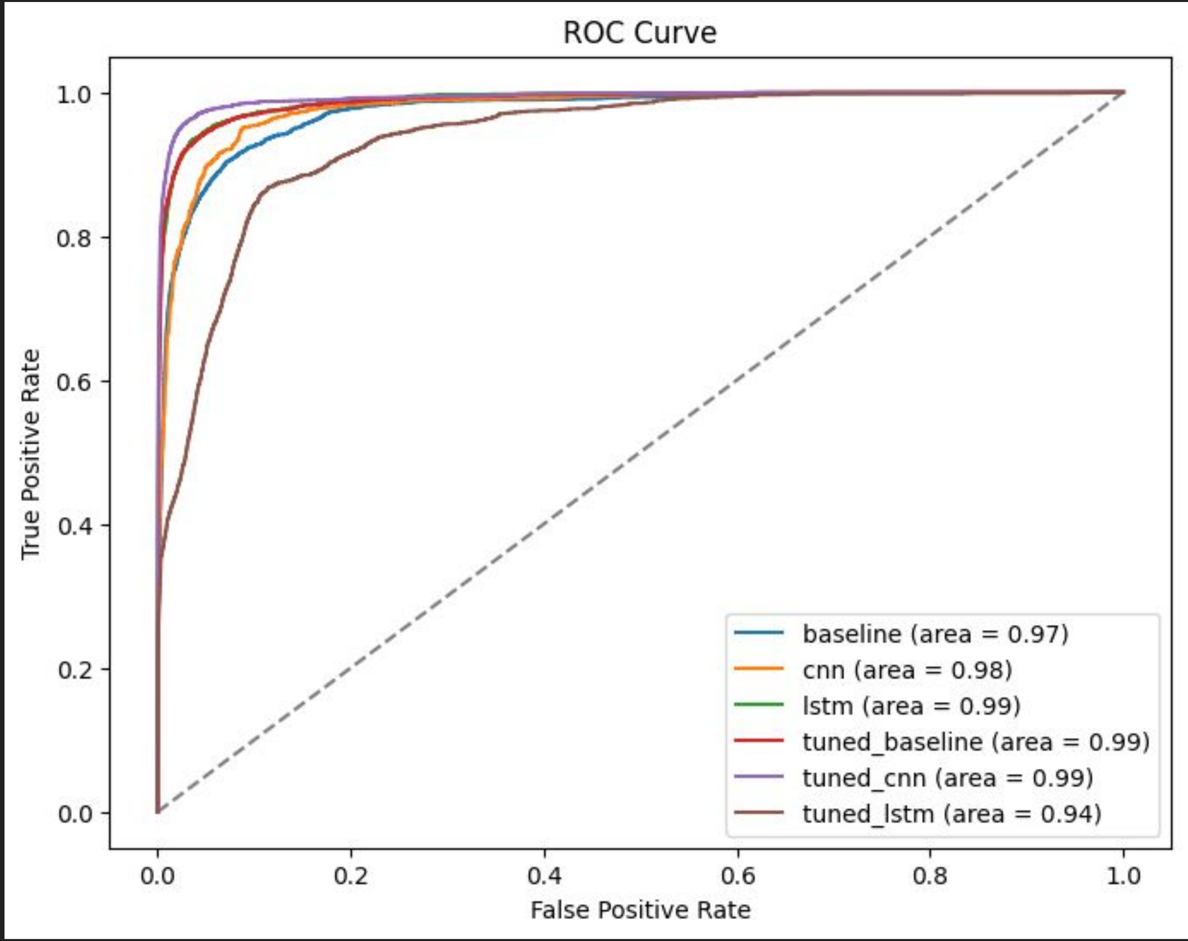
Optimizer: Adamax

Metrics:
- lossbinary_crossentropy
- accuracy
- auc



Results: Overall

Each model performed well, with all models achieving an ROC score above 0.90. The tuned CNN model had the best performance, while the tuned LSTM model performed the worst.



Conclusion

Multiple deep learning models were utilized to create features based on a specific user's sequence of sessions and to determine whether the user was Alice.

- The results showed that the CNN model performed the best, while the LSTM model had the lowest ROC score.
- By using the features sequentially and processing them in a way that suits the LSTM model, it may be possible to identify the specific user without manually engineering features. This will be the next challenge to explore.

Thank you for your time