Predicting user churn on streaming services using recurrent neural networks

Helder Martins

Supervisors: Hedvig Kjellström (KTH), Sahar Asadi (Spotify)

Outline

- Introduction
- Methods
- Results
- Conclusions

Introduction

Introduction

- The user base of **streaming service providers** (SSPs) like Spotify increased rapidly in the last decade, attracting new competitors to the market.

- High cost for acquiring new users, raising the relevance of **user retention**.

- Detecting the users more likely to leave the SSP in the future, aka **churn**, is an important step for user retention.

Introduction

- **Churn prediction** is the task of predicting how likely a user is to abandon a service provider.
- **Logistic regression** and **random forests** are commonly used models for predicting churn by **aggregating** the user attributes over a period of time.
- **Long short-term memory** (LSTM) is a recurrent neural network well suited for **sequential** data like speech recognition and video classification.

Research Question I

"Which predictive modeling algorithm amongst **logistic regression**, **random forests** and **LSTM** obtains the best performance for predicting the chance of users churning in the future?"

Introduction

- Different aspects of the training data have a direct impact on the performance of classifiers
  - Balance between churning and retaining classes
  - Amount of historical information
  - Distance in the future predictions are being made
  - Data representation
- Understanding how accuracy changes provides insights on user behavior

Research Question II

“How do **different aspects of the data** influence the accuracy of the predictive models?”

Such as ...

- … the class distribution between churning and retaining users
- … the amount of historical user behavior information used for training
- … lower dimensional representation of the data

Contributions

- Evaluating the performance of **LSTM** compared **random forests** and **logistic regression** for predicting churn.

- Assessing the impact that the size of the **customer event history** has in accuracy of the trained predictive models.

- Analyzing the performance of models when predicting churn rate for **increasing ranges in the future**.

Contributions

- Experimenting on **changing the ratio** between the **retained** and **churning classes**, and the impact in accuracy that it yields.

- Evaluating if a **lower dimensional representation** of user data can improve the performance of predictive models.

- A **data pipeline framework** suitable for extracting sequential information from raw user data.

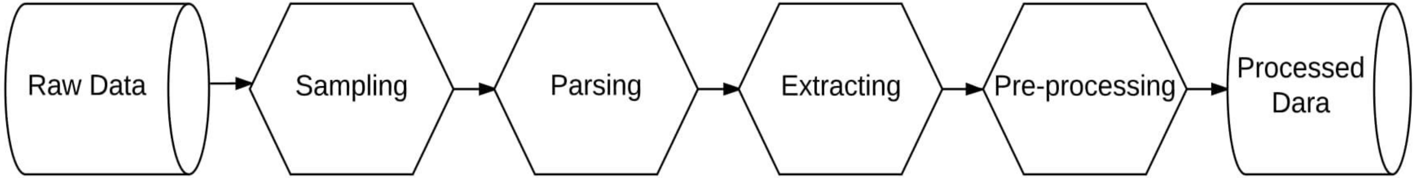
Methods

Dataset

- The source data is **streaming behavior**
    - Consumption time of media content
    - Contexts of the application being used
    - Time between streams
    - Total number of streams

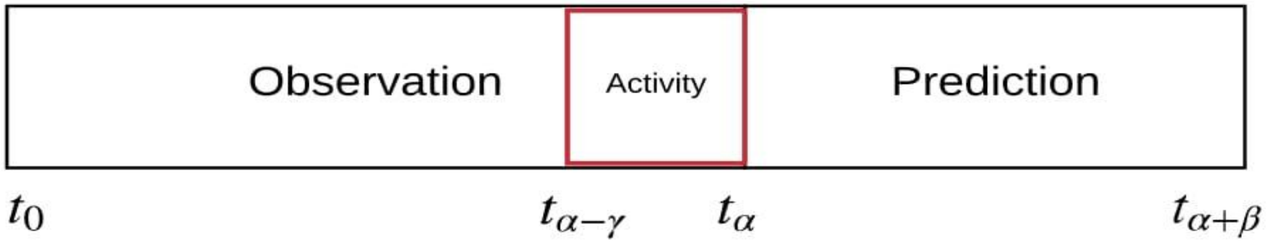
- In total, **52 features** were used
- **414794 active users** sampled from **3 different markets**
- **1.8B streams** extracted representing **86 days** (March - May 2017)

Data Pipeline



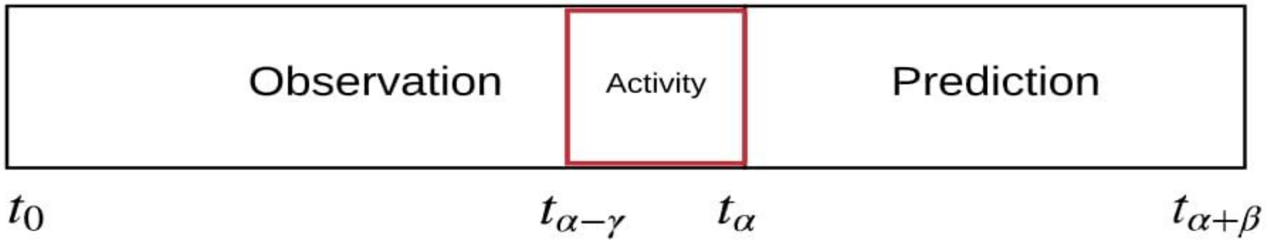
- The **data pipeline** is composed of 4 stages
  - **Sampling**: active users are selected
  - **Parsing**: context strings are parsed
  - **Extracting**: features are engineered and aggregated in time steps
  - **Pre-processing**: data is normalized and exported to files

Time Windows



- Temporal data is split into **time windows**
  - **Observation**: Used to create user samples
  - **Prediction**: Used for labelling churned or retained
  - **Activity**: Used for sampling users that are active
- Each time window is composed of several equally sized **time steps** of **8 hours** each

Churn Definition



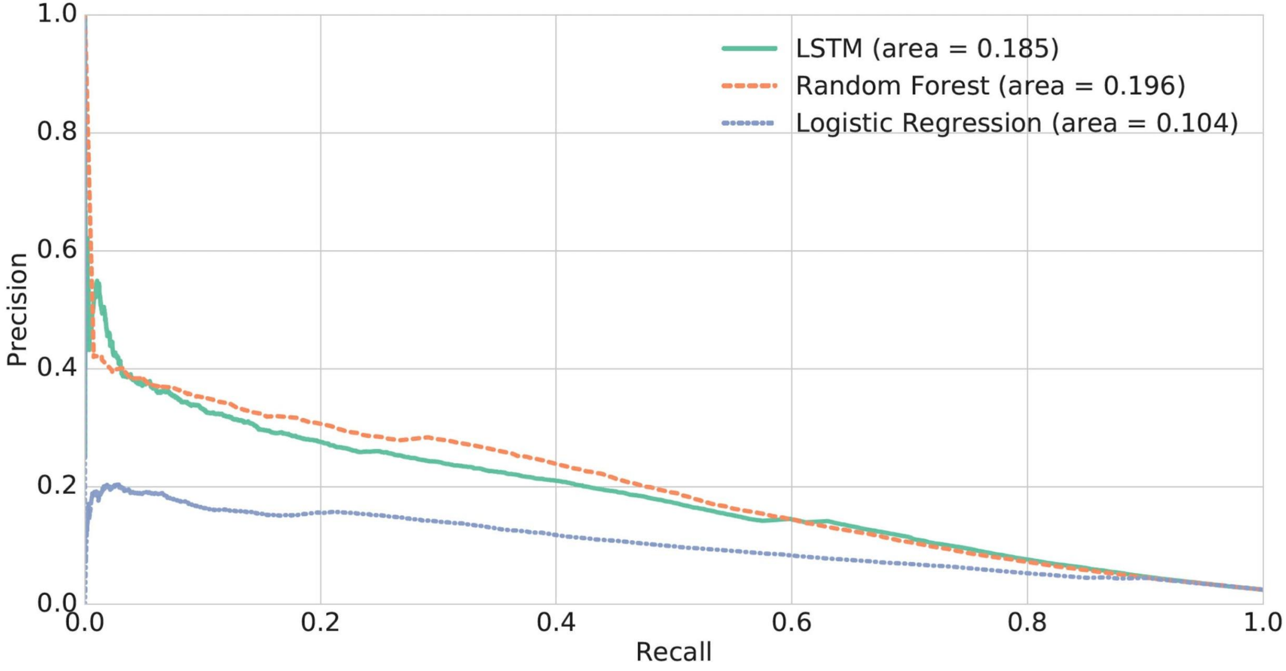
- A user **churns** iif there is **no stream of his during the prediction window**.
- Share of churners depends on size of prediction window
  - **2.4% when 30 days, 7.5% when 7 days**

Results

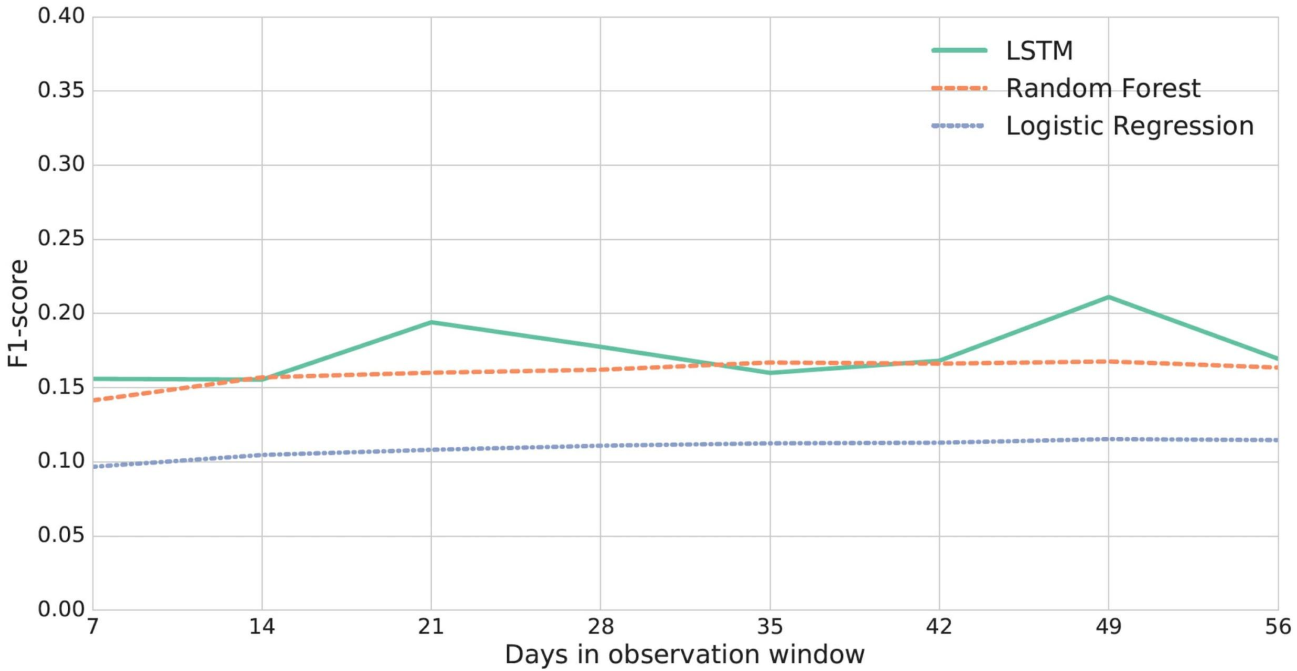
Disclaimer

- 5 experiments were performed:
  - LSTM vs. baselines
  - Observation / Prediction windows
  - Class Balance
  - Dimensionality Reduction
- F1-score is the main evaluation metric
  - Thresholded at 0.5 for churn / retain classification
- Default size of time windows (days):
  - 56 for observation, 30 for prediction
- Default class distribution in training data:
  - 50% churners, 50% retainers

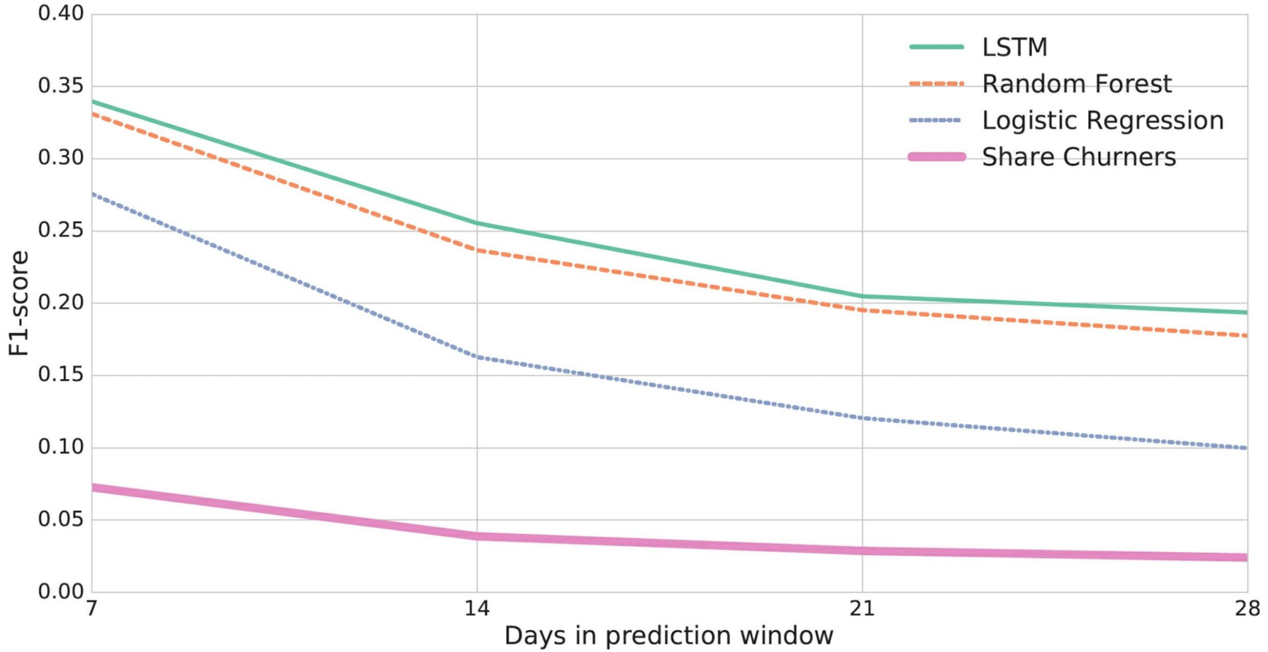
LSTM vs. Baselines



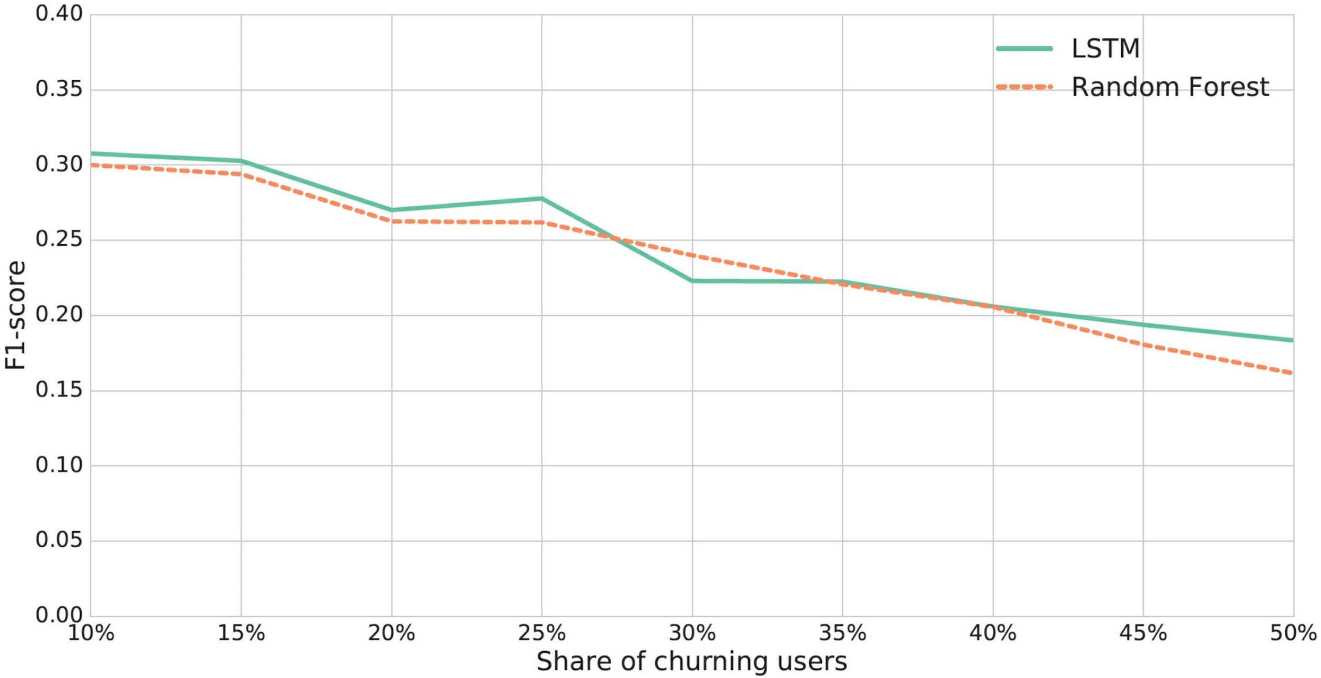
Observation Window



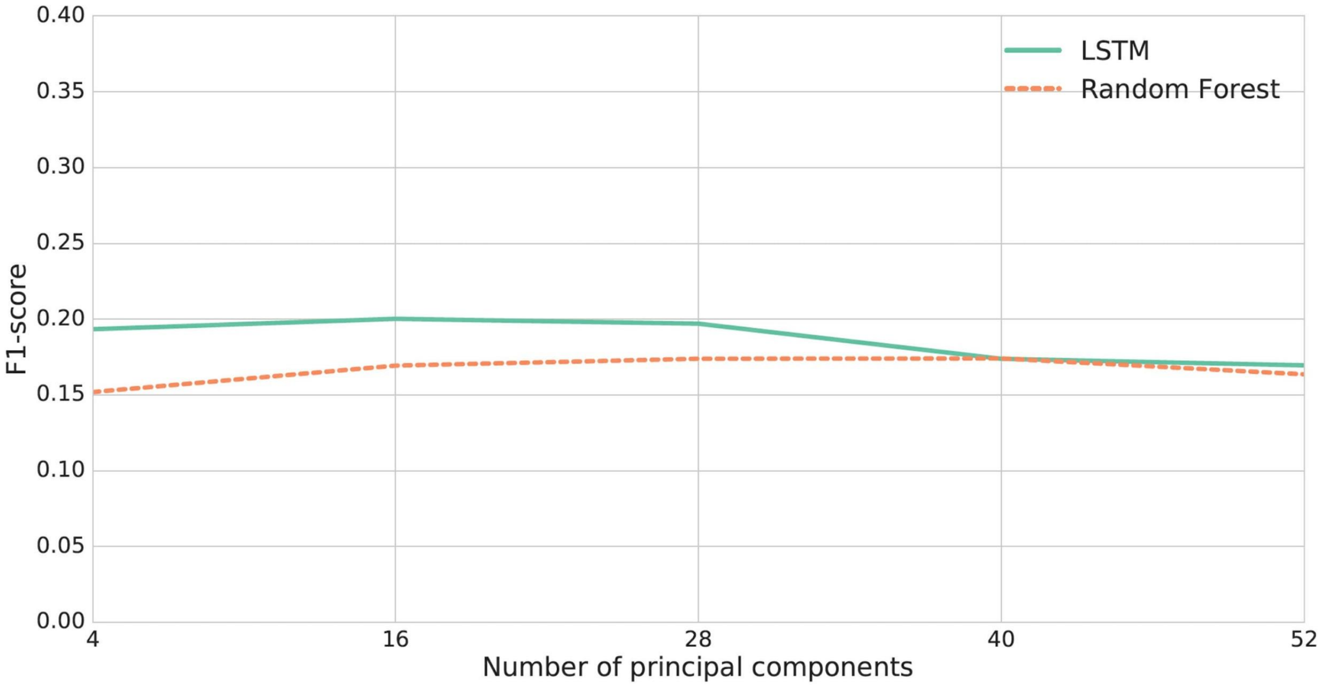
Prediction Window



Class Distribution



Dimensionality Reduction



Conclusions

LSTM vs. Baselines

- LSTM >> Logistic Regression

- LSTM ≅ Random Forest
  - RF thrives with good features
    - However feature engineering is expensive

  - Insufficient temporal data for LSTM

  - Limited number of hyperparameters tested
    - Costly to train an LSTM

The Class Imbalance Problem

- Churning class is extremely rare (2.4% - 7.5%)

- The lesser the gap between the classes…
  - … lower the precision
  - … higher the recall
  - … lower the F1-score

- The problem could be mitigated…
  - … by using more data
  - … with a smarter undersampling
  - … with a customized loss function

The Impact of Time Windows

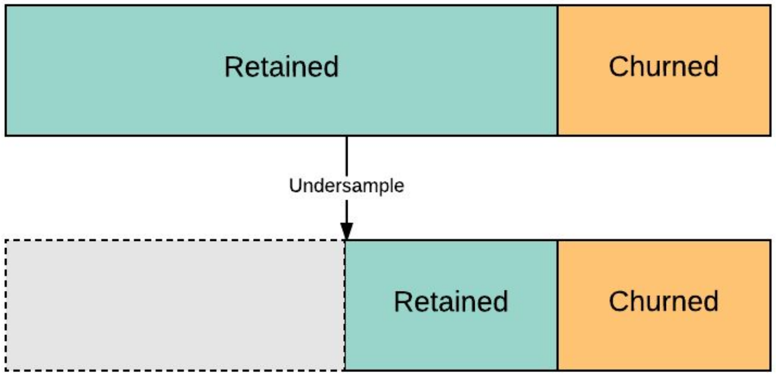
- The smaller the prediction window, higher the F1-score
  - More churners
  - Easier to predict the near future

- The larger the observation window, higher the F1-score
  - Albeit sligthly
  - LSTM has peaks of performance as more days are added to training

Additional Slides

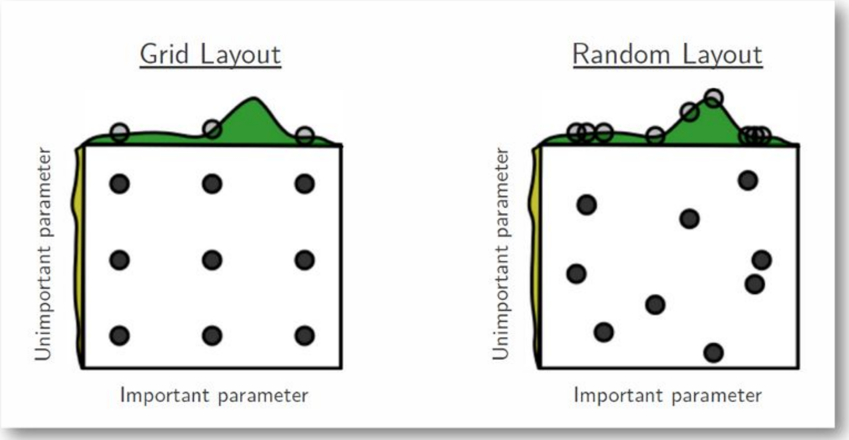
Class Imbalance

- Large gap between the retaining and churning classes
- Experiments will **randomly undersample** the majority class for training



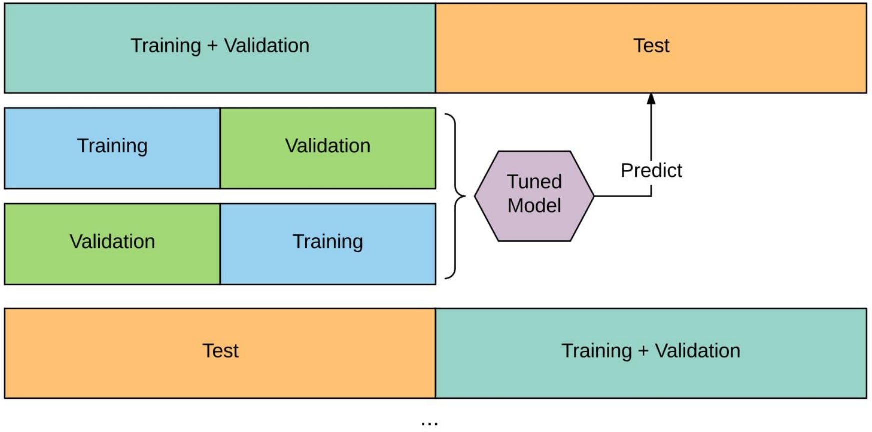
Hyperparameters Search

- Grid search for baseline models
  - RFs: Number estimators
  - LR: Regularization method / strength
- Random search for LSTM
  - Number layers and units
  - Optimizer



Cross Validation

- 2x2 cross-validation
  - 2 outer splits for train/val and test data
  - 2 inner splits for train and validation
- Inner splits are used for hyperparameter tuning
- Outer splits for evaluation



Future Work

- Sequence of binary classification at each time step instead of a single value
    - Encodes the transition of users between states
    - Sequence-to-sequence LSTM
    - Survival analysis

- System improvements
    - No cross-validation
    - Horizontal scaling (Downpour SGD, Tensorflow)