COMP4211 Project Report

Kaggle Competition: TMDB Box Office Prediction

Ng Chi Him

20420921

chngax@connect.ust.hk

## Introduction

For the course project, I have participated in a Kaggle competition titled TMDB Box Office Prediction. The objective is to predict the ticket sales number of movies based on dataset provided by The Movie Database. To tackle this regression problem, I have experimented with three machine learning methods - Feedforward Linear Regression, Neural Network, and Random Forest.

## Computing Environment

### Software

Python 3.6 and Jupyter Notebook is used for development. PyTorch was used to build and train the neural network while Scikit-Learn was used to execute Linear Regression and Random Forest algorithms.

### Hardware

CPU: Intel i7-6820HQ (Skylake, 2.7Ghz 4 Core 8 Threads)

RAM: 16GB

GPU: NVIDIA RTX 2070 (8GB RAM)

## Dataset and Pre-processing

The dataset contains 3000 rows of training data with ground truth and 4398 rows of testing data with input fields only for generating submissions. There are 21 input fields in total, such as budget, genre, popularity, etc.

### Excluded Fields

Some fields do not seem to have correlation to the output, such as homepage url and imdb id. Also, as text mining is not in the project scope, long text fields such as overview, title and tagline will be excluded. Fields with content that are too sparse will be excluded as well, such as cast and crew, in order to save processing time in later stages. Please refer to source code for exact list of excluded fields.

### One-hot encoding

For categorical fields such as genre and language, they cannot be processed by the proposed learning methods. Therefore, one-hot encoding is used to transform them to vector form, where binary values are assigned to denote whether the category is active for data entry.

After processing, which consumed around 15 minutes of computation, the input size grew to 20082 fields, all in numeric form.

**Train-Validation splitting**

All data will be shuffled and split into training set (80%) and calidation set (20%) at the beginning of each implementation for the validation stage. The whole set will be used to train the final model for submission in the testing stage.

**NaN Values**

All NaN (no-a-number) values will be replaced by 0 before feeding into our models.

## Scoring Method

Root-Mean-Squared-Logarithmic-Error (RMSLE) is used as the loss function or the scoring function whenever possible. For Random Forest, Mean-Squared-Error (MSE) is used as criterion is the RMSLE is not available as an option.

All “Score” or “Loss” stated in results below are RMSLE.

## Linear Regression

As the simplest solution, Linear Regression was not expected to perform well and used as base line only.

**Validation Result**

Training set score: 0.03461

Validation set score: 7.020

**Testing Result**

In order to save submission quota limited by Kaggle (10 per day), testing output was not generated and submitted as the validation score is signification worse than other models.

## Feedforward Neural Network

### Design

The neural network has 5 fully-connected layers, with SELU and Dropout between each layer.

```
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(in_features=20082, out_features=8192),
            nn.SELU(),
            nn.Dropout(p=0.5),
            nn.Linear(in_features=8192, out_features=4096),
            nn.SELU(),
            nn.Dropout(p=0.5),
            nn.Linear(in_features=4096, out_features=2048),
            nn.SELU(),
            nn.Dropout(p=0.5),
            nn.Linear(in_features=2048, out_features=1024),
            nn.SELU(),
            nn.Dropout(p=0.5),
            nn.Linear(in_features=1024, out_features=1),
            nn.SELU()
        )
        self.loss_func = nn.MSELoss()
```

### Experiment: SELU vs ELU vs ReLU

Different activation layers are tested with configuration below:

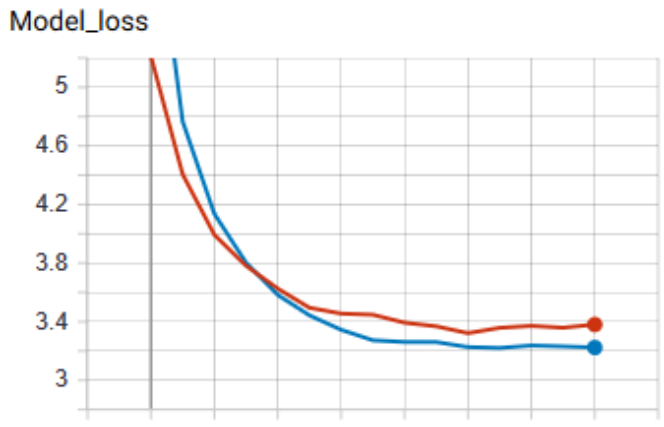
Optimizer: Adam, Learning rate: 1e-4 (default 1e-3 was not able to converge)

Beta1: 0.9, Beta2: 0.999 (0.99 for SELU as suggested by original paper [1])

Batch size: 32, Epochs: 15

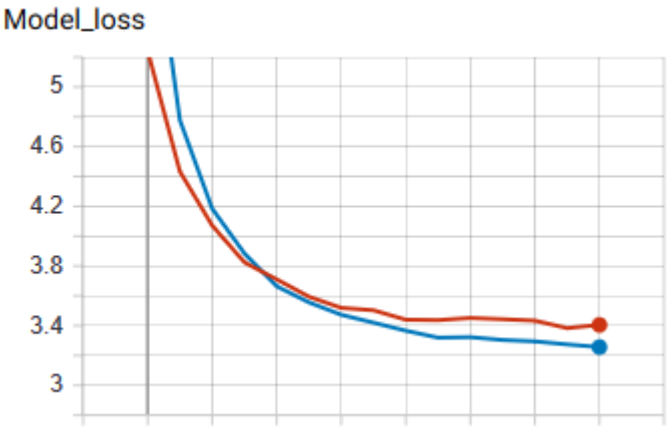
| | SELU | ELU | ReLU |
|---|---|---|---|
| Train Loss | 3.2239 | 3.2567 | 3.2486 |
| Val. Loss | 3.3807 | 3.4054 | 3.3972 |
| Time Used | 1m41s | 1m40s | 1m40s |

SELU Loss Graph



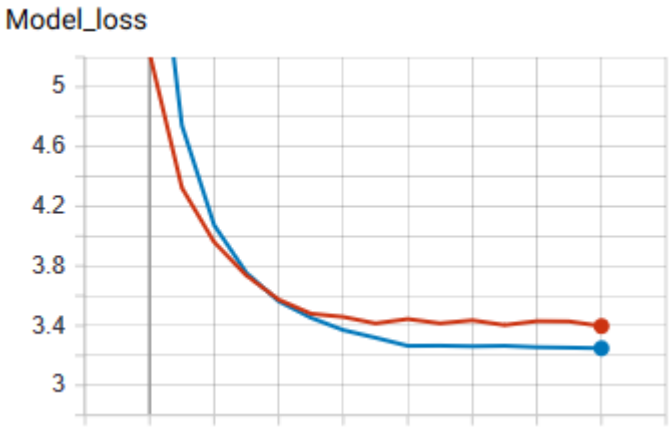

Red: Val., Blue: Train

ELU Loss Graph




Red: Val., Blue: Train

ReLU Loss Graph




Red: Val., Blue: Train

SELU was selected for the final model as it gave the lowest loss.

**Hyperparameter Tuning**

Different learning rates are tested with configuration below:

Activation: SELU

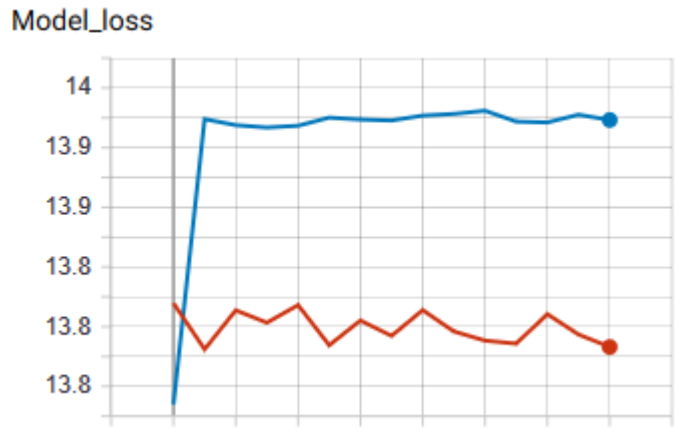
Optimizer: Adam

Beta1: 0.9, Beta2: 0.99

Batch size: 32, Epochs: 15

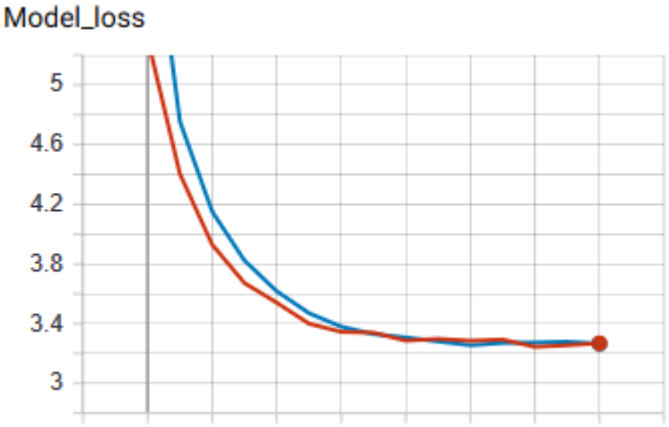
| | 7e-3 | 1e-4 | 3e-4 | 2e-4 | 2.5e-4 |
|---|---|---|---|---|---|
| Train Loss | 13.9386 | 3.2662 | 3.2815 | 3.2492 | 3.2748 |
| Val. Loss | 13.7864 | 3.2689 | 3.2906 | 3.2524 | 3.2598 |
| Time Used | 1m44s | 1m43s | 1m39s | 1m40s | 1m41s |

7e-3 Loss Graph



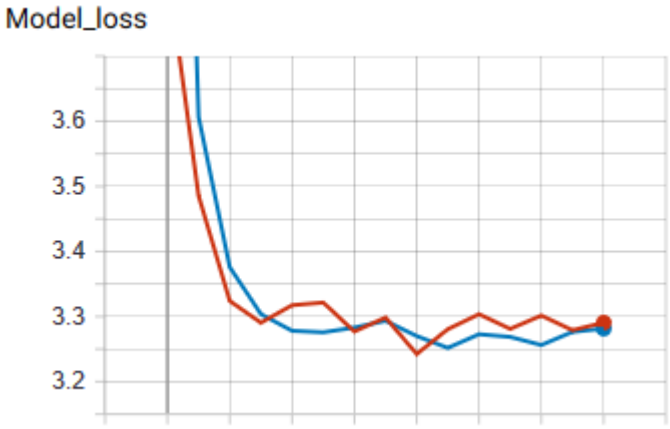

Red: Val., Blue: Train

1e-4 Loss Graph

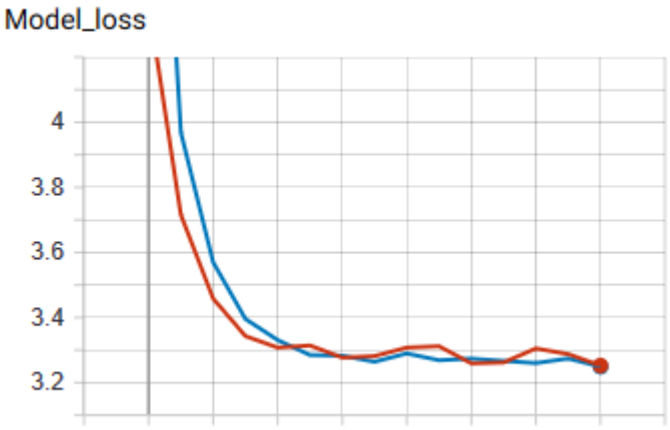



Red: Val., Blue: Train

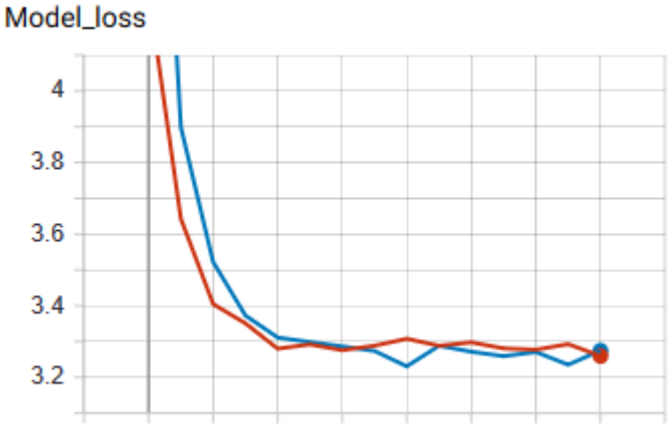
3e-4 Loss Graph




2e-4 Loss Graph




2.5e-4 Loss Graph




2e-4 was selected for the final model.

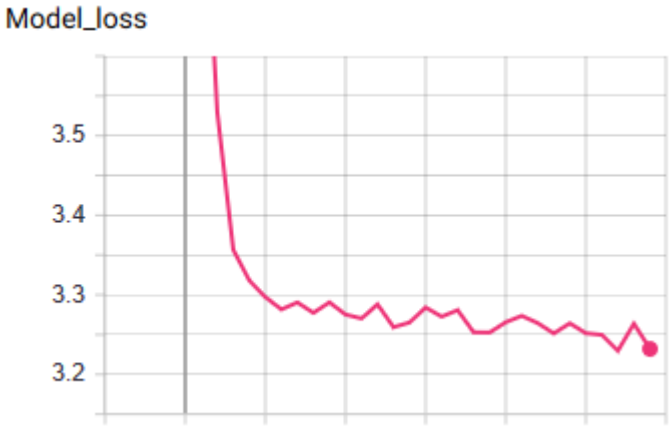
**Final Model**

Activation: SELU

Optimizer: Adam, Learning rate: 2e-4

Beta1: 0.9, Beta2: 0.99

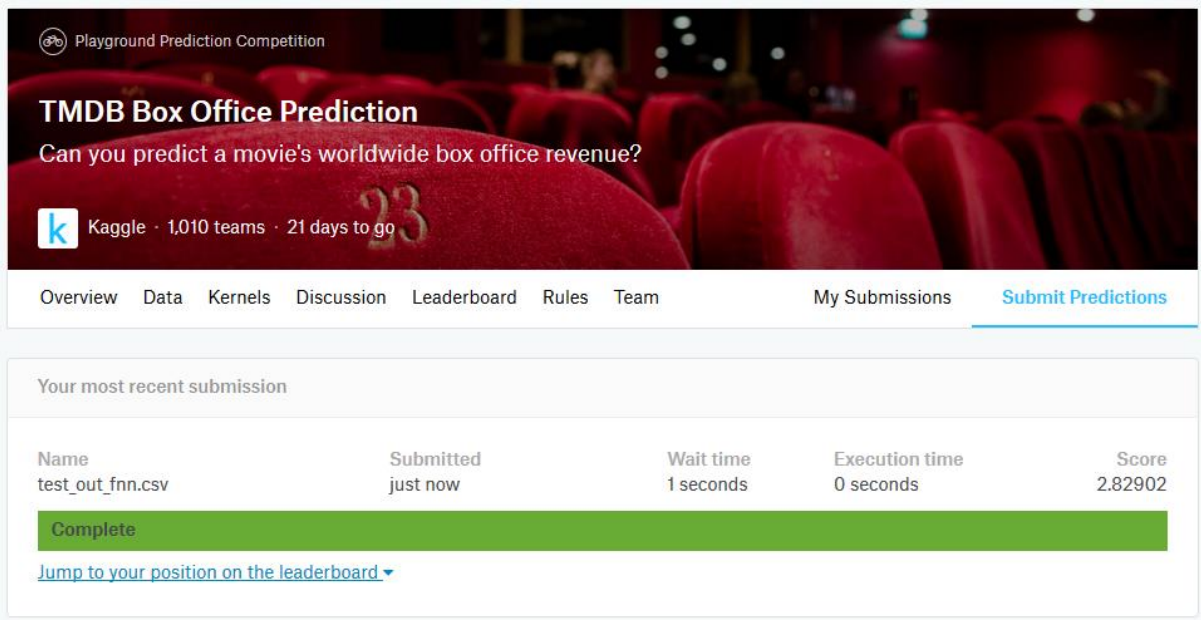
Batch size: 32, Epochs: 30




Training Loss: 3.232

Time Consumed: 4m16s

Kaggle Score: 2.82902



## Random Forest

### Hypermeter Tuning

Different numbers of estimators were tested with configuration below:

```
criterion='mse',
max_depth=None,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features='auto',
max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_impurity_split=None,
bootstrap=True,
oob_score=False,
n_jobs=None,
random_state=None,
verbose=0,
warm_start=False
```

| | 25 | 50 | 75 | 100 | 125 |
|---|---|---|---|---|---|
| Train Score | 1.8348 | 1.8338 | 1.8524 | 1.8512 | 1.8624 |
| Val Score | 2.128 | 2.0845 | 2.0818 | 2.0776 | 2.1096 |

125 was selected for the final model as there seems to be diminishing of return.

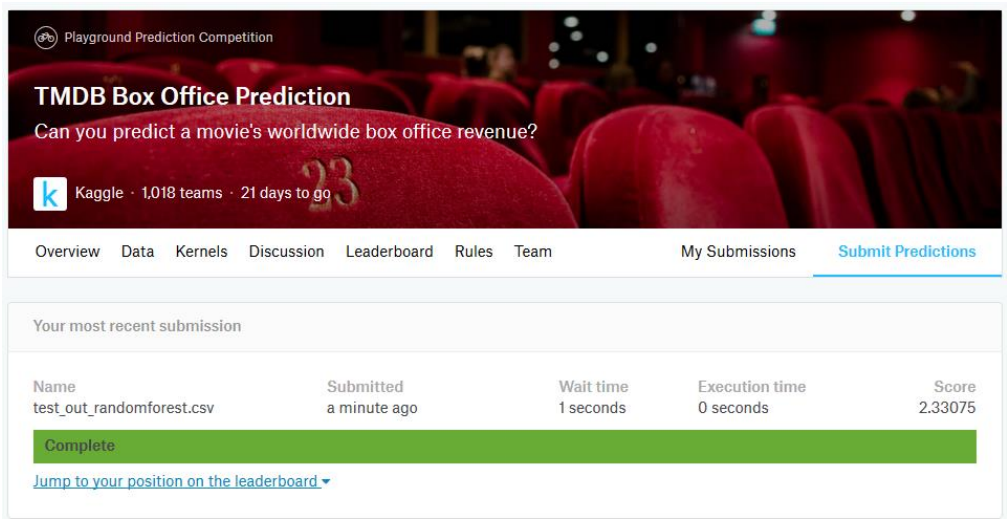
### Final Model

Configuration:

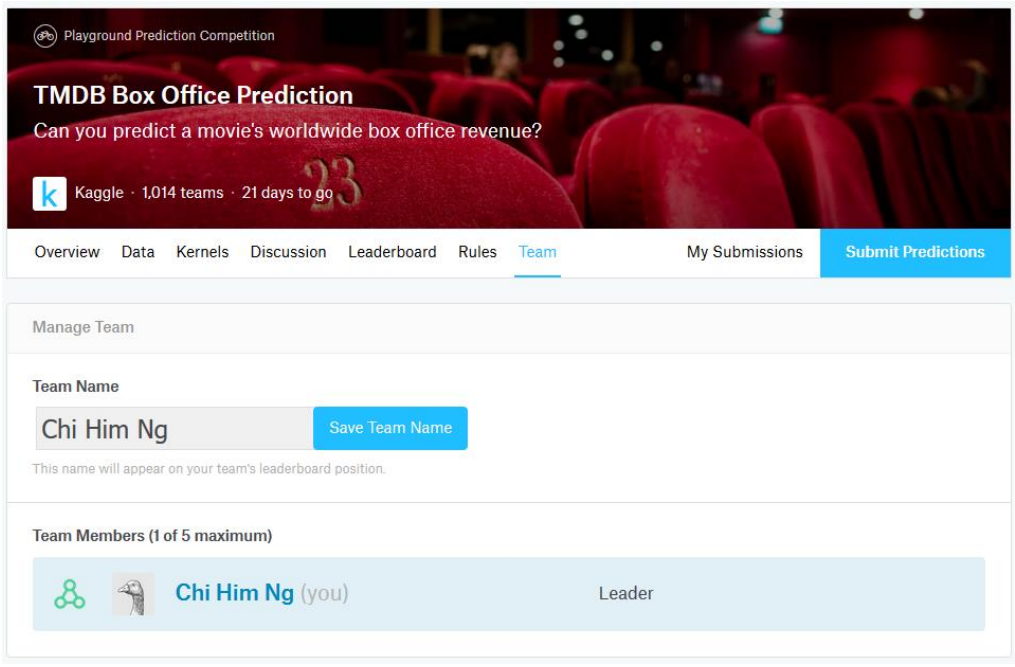
```
forest = RandomForestRegressor(
    n_estimators=125,
    criterion='mse',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features='auto',
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    min_impurity_split=None,
    bootstrap=True,
    oob_score=False,
    n_jobs=None,
    random_state=None,
    verbose=0,
    warm_start=False
)
```

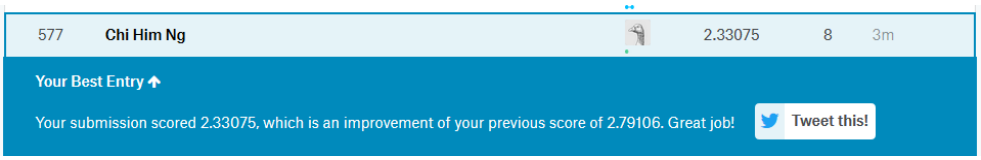
Train Score: 1.8320

Kaggle Score: 2.33075




## Kaggle Team Proof




## References

[1] Klambauer, Günter, et al. "Self-normalizing neural networks." *Advances in neural information processing systems*. 2017.