Q4 – TP3

State of the Art

Growing tendencies in recommender systems, according to our literature review, include conversational recommender systems, which establish systems that talk to the users and convert the feedback, using Natural Language Processing techniques [3][4], and recommender systems for the health care domain [9]. Trending social topics include the fairness of recommender systems [2][6], as well as digital nudging [5].

Several different Artificial Intelligence approaches have been surveyed by [8], like deep neural networks, transfer learning, active learning and evolutionary algorithms. We can see that deep learning is indeed a popular topic in recommender systems, being cited by another research, like [10], [11] and [12]. As for the techniques, we can see that transformers are a very popular method in new works, like in [7], [8] and [13].

Therefore, we have decided to use a Transformer model for our project. Our code is based on the code provided by the paper “A Transformer-based recommendation system”, by Khalid Salama, which is heavily inspired by the model presented in [1].

References

[1] Chen, Qiwei, et al. "Behavior sequence transformer for e-commerce recommendation in alibaba." Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data. 2019.

[2] Deldjoo, Yashar, Alejandro Bellogin, and Tommaso Di Noia. "Explaining recommender systems fairness and accuracy through the lens of data characteristics." Information Processing & Management 58.5 (2021): 102662.

[3] Gao, Chongming, et al. "Advances and challenges in conversational recommender systems: A survey." AI Open 2 (2021): 100-126.

[4] Jannach, Dietmar, et al. "A survey on conversational recommender systems." ACM Computing Surveys (CSUR) 54.5 (2021): 1-36.

[5] Jesse, Mathias, and Dietmar Jannach. "Digital nudging with recommender systems: Survey and future directions." Computers in Human Behavior Reports 3 (2021): 100052.

[6] Li, Yunqi, Yingqiang Ge, and Yongfeng Zhang. "Tutorial on Fairness of Machine Learning in Recommender Systems." Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021.

[7] Liu, Davide, George Philippe Farajalla, and Alexandre Boulenger. "Transformer-based Banking Products Recommender System." Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021.

[8] McDonnell, Serena, et al. "Cyberbert: A deep dynamic-state session-based recommender system for cyber threat recognition." 2021 IEEE Aerospace Conference (50100). IEEE, 2021.

[9] Tran, Thi Ngoc Trang, et al. "Recommender systems in the healthcare domain: state-of-the-art and research issues." Journal of Intelligent Information Systems 57.1 (2021): 171-201.

[10] Shambour, Qusai. "A deep learning based algorithm for multi-criteria recommender systems." Knowledge-Based Systems 211 (2021): 106545.

[11] Shin, Kyuyong, et al. "One4all user representation for recommender systems in e-commerce." arXiv preprint arXiv:2106.00573 (2021).

[12] Wang, Shoujin, et al. "Graph learning based recommender systems: A review." arXiv preprint arXiv:2105.06339 (2021).

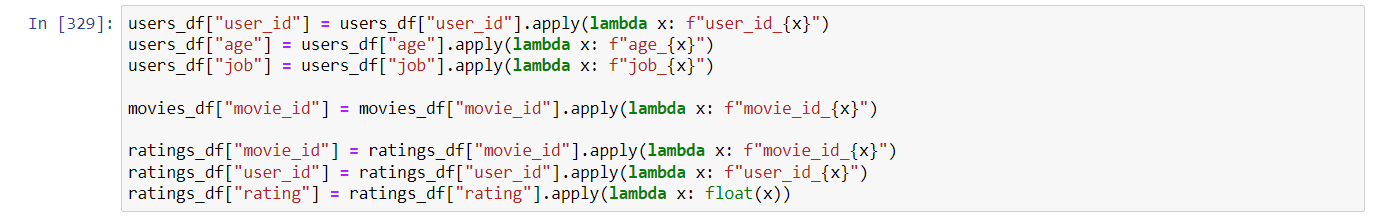
[13] Xia, Lianghao, et al. "Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 5. 2021.

Our Model

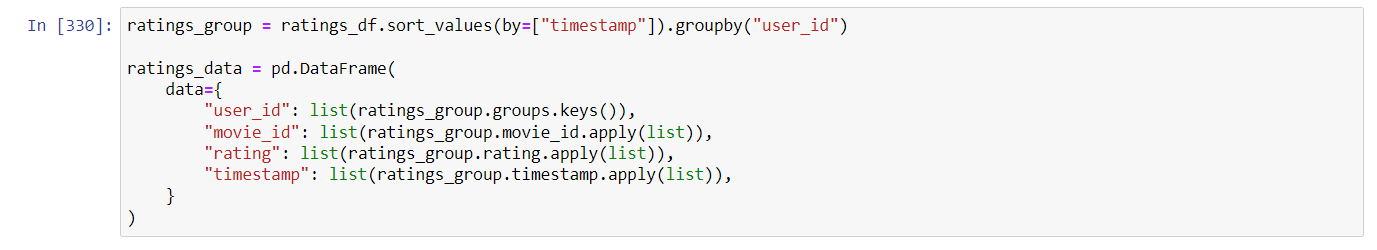
Our model is based on the code given by Khalid Salama, inspired by the model presented in [1]. The full code will be attached to this document. The first step is to load the libraries and the CSVs:



Then, we substitute the values for a whole string, containing the item type and the item id:



Then, we create a new dataframe, ordering the items by timestamp and grouping the items by their user ids.



After that, we have the sequence creator. In order to process the items in our Behavior Sequence Transformer, we have to concatenate many items in a single sequence. The step size defines the speed rate of each step. In our tests, we can see that the model behaves better with even sequence numbers, and with low step sizes. For the example below, we are going to work with sequence length 2, and step size 1.

After that, since we no longer need the timestamps, they are deleted from the dataset.



Then we create a new transformed dataset, containing the sequences of movie ids and ratings.



Then, we split the dataset into 80/20 for training and validation, as requested. Some lists and a dictionary are used for the metadata.

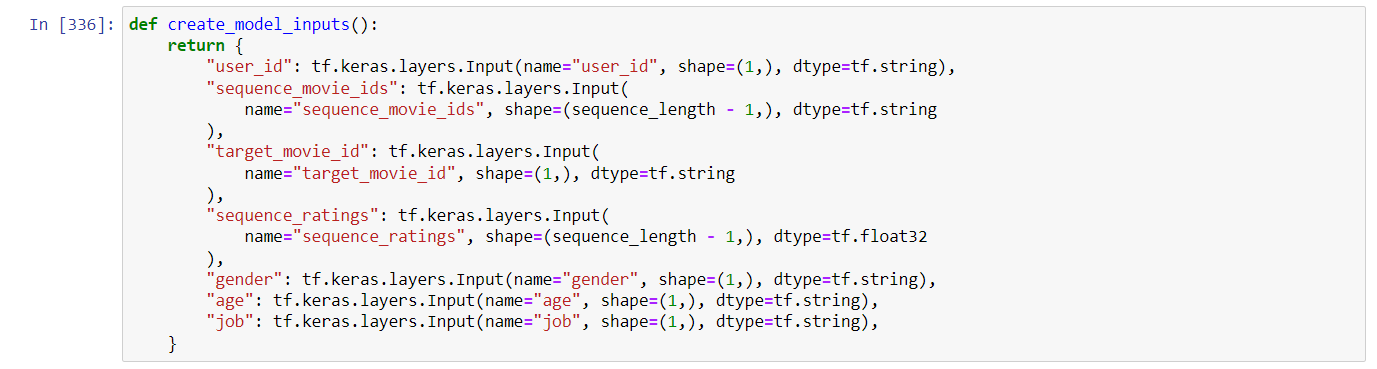
In our code, we have commented this first cell, so that the method would use the same sets for training and validation in every model compilation, so that it would be easier to compare the effectiveness of the model. The cell can be uncommented to see its functionality.



Then we have a method to read the dataset from the saved CSV.



Now we have a method to create the inputs for the neural network:



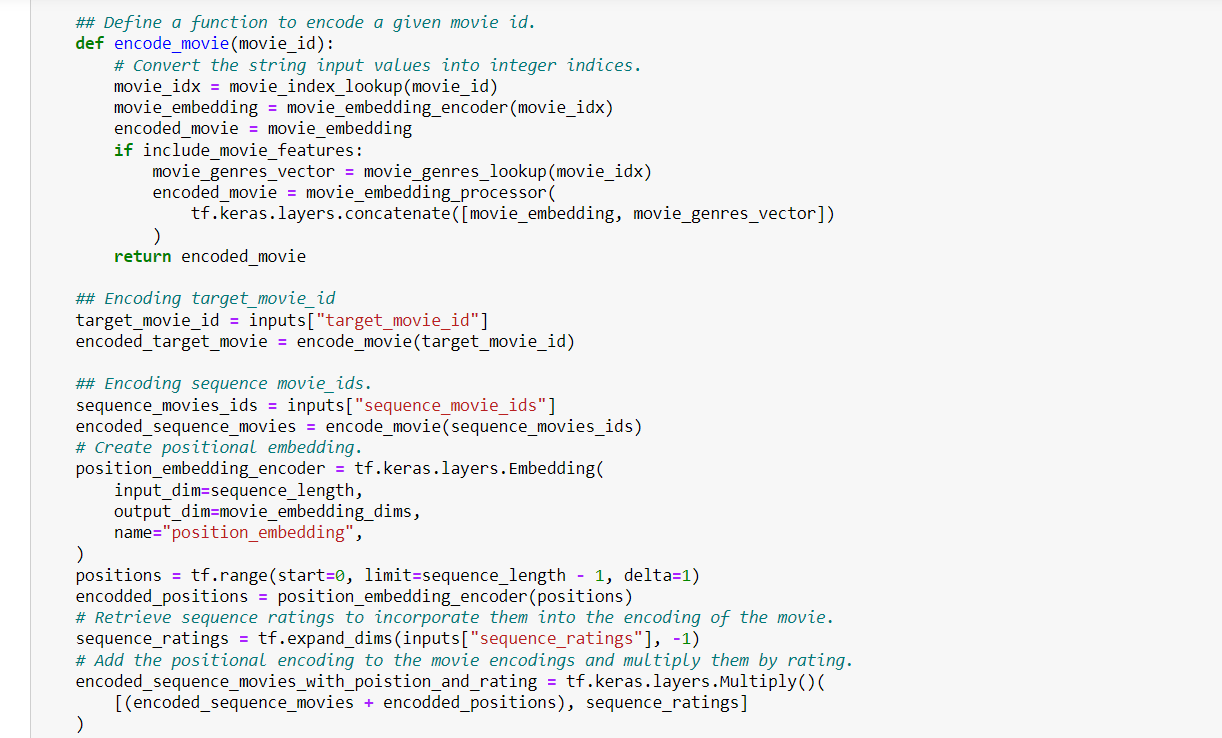
Next, we have a method to create the embeddings for the inputs, using keras.layers. StringLookup and keras.layers.Embedding

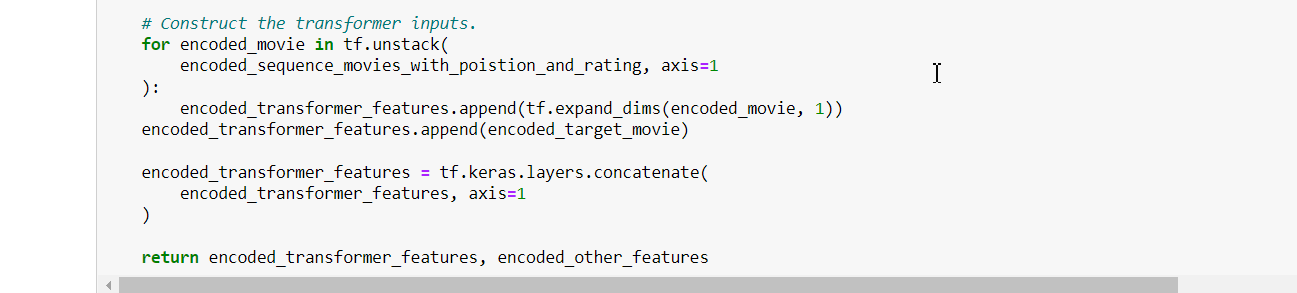






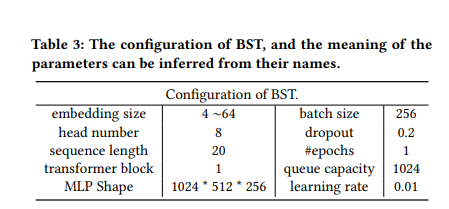
Then, we have a method to encode the movie id, respecting the sequence that we have created in the previous methods.



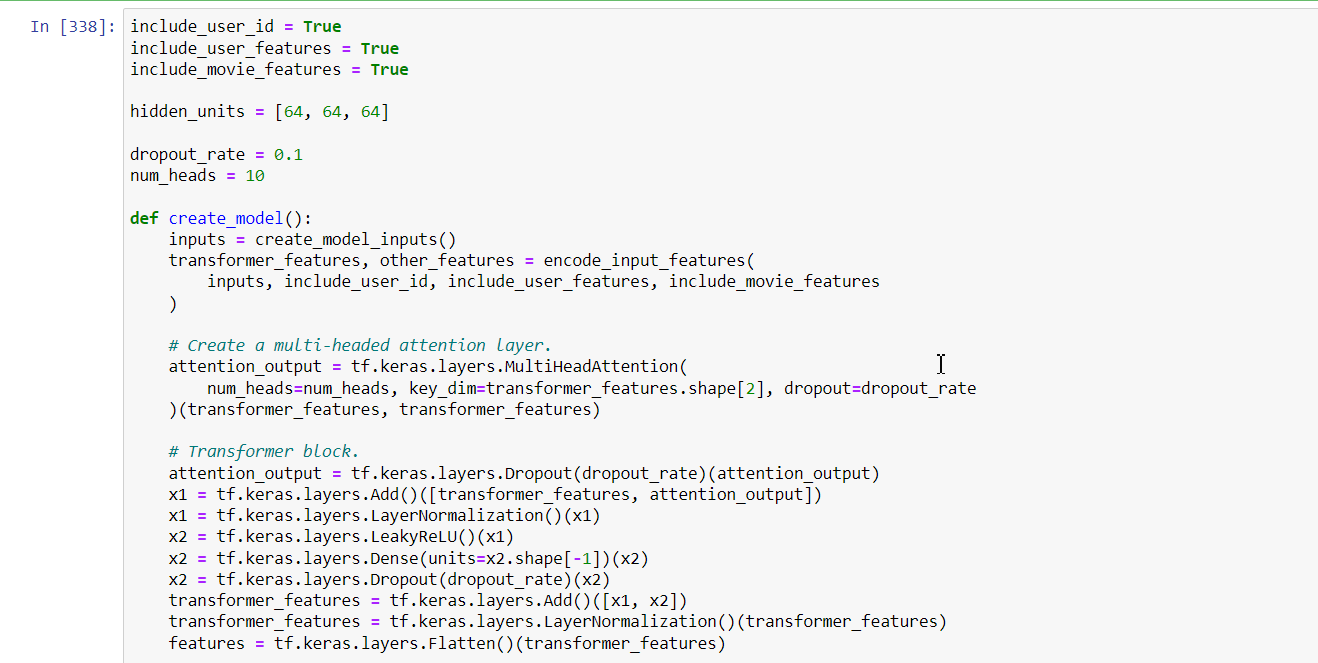


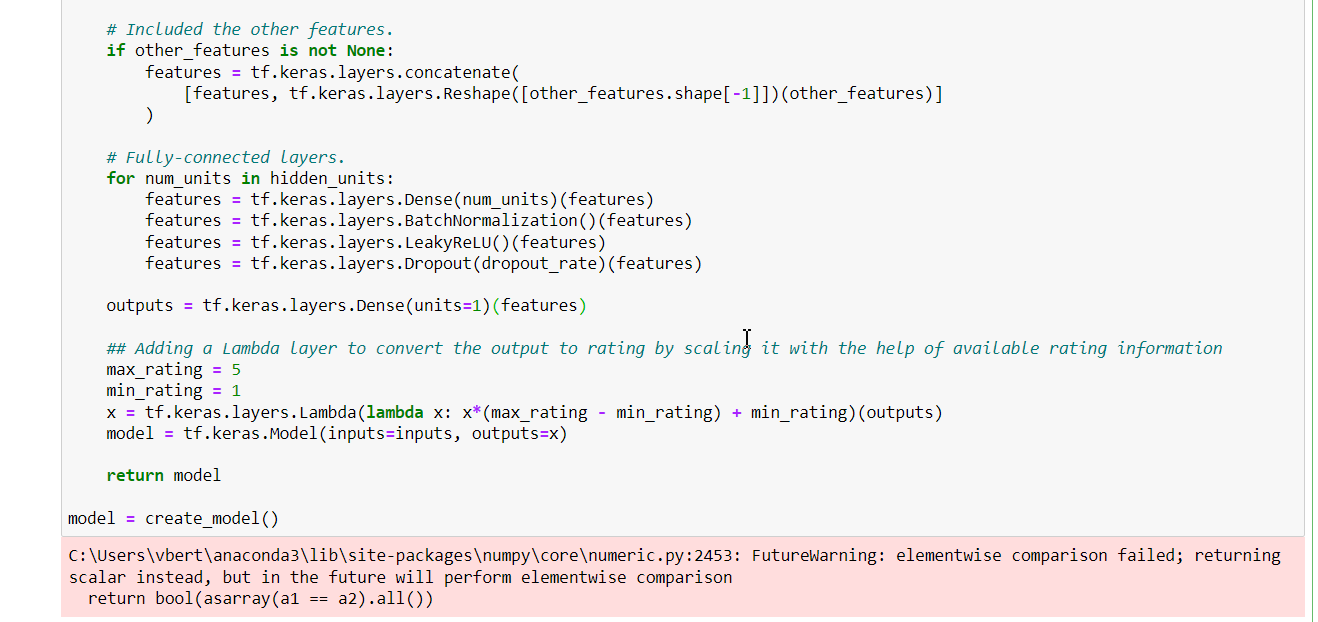
Now, for the model. Here, we describe that we want to use user id, user and movie features, and in the list “hidden\_units”, we establish the number of dense layers, as well as the number of neurons of each layer.

In our model, after several tests, we have found that using more layers will less neurons usually work best than less layers with more neurons. The original paper, [1], uses a 3-layer architecture, with 1024, 512 and 256 neurons, respectively. Those are the configurations of the original paper. The “embedding size” is our sequence length, and “transformer block” is our step size. The other parameters have the same name.

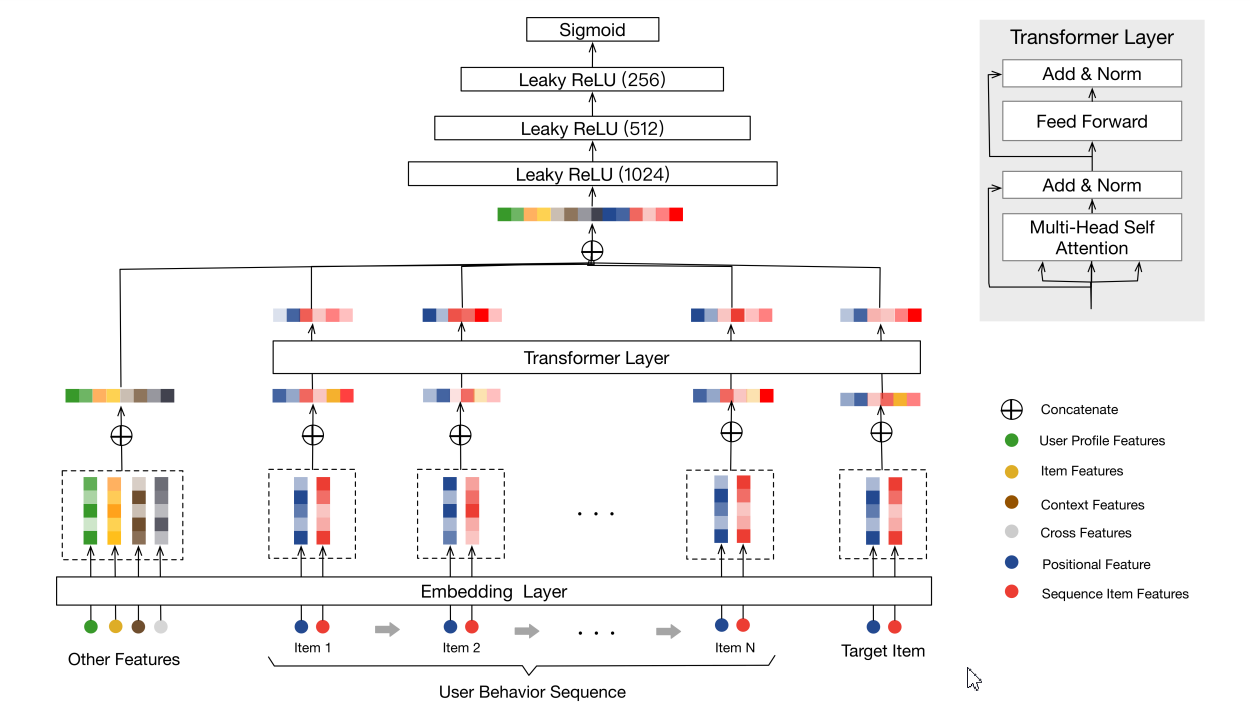


And for the code:

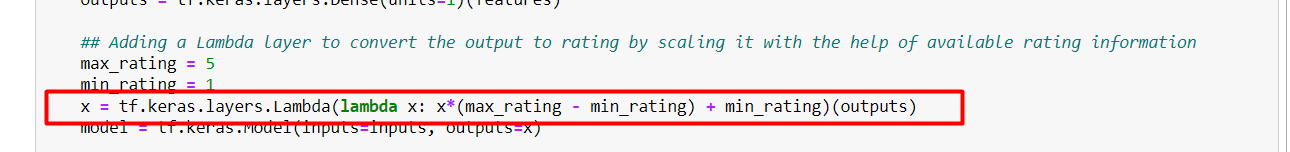




The method is described below, by the picture extracted from [1].



A significant change from the original method is that we apply a normalization based on the min and max ratings.

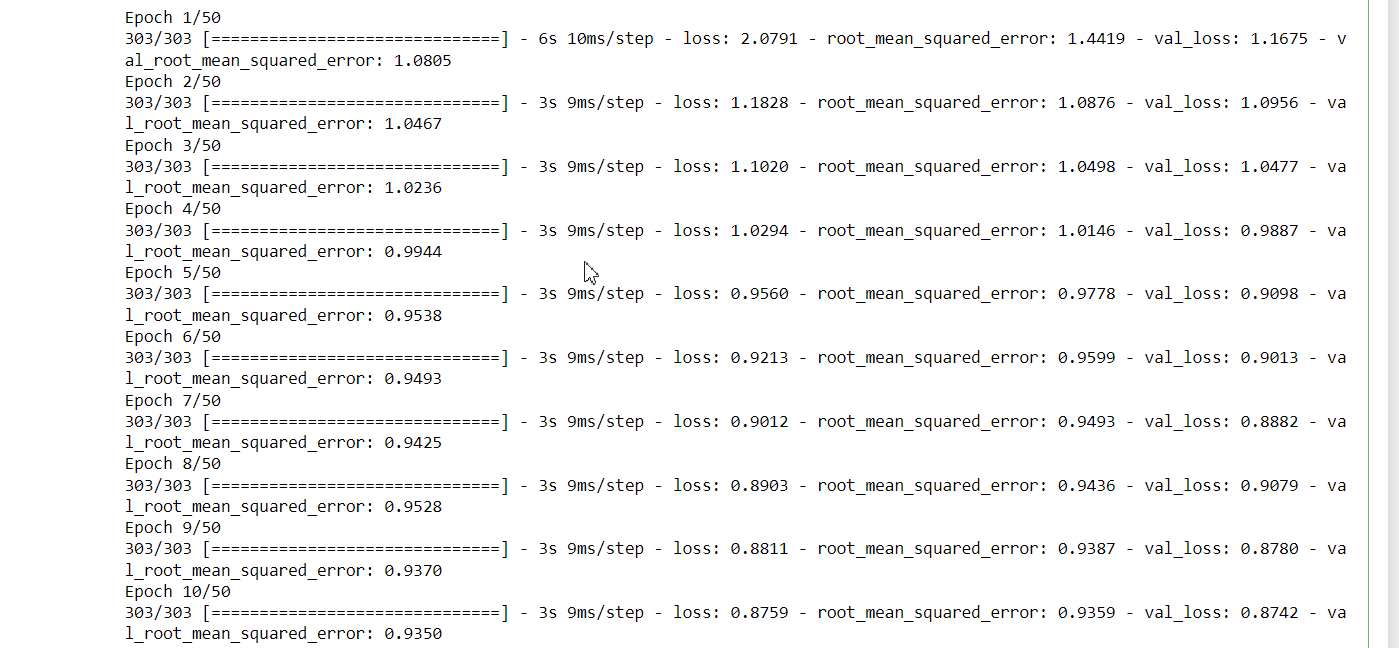


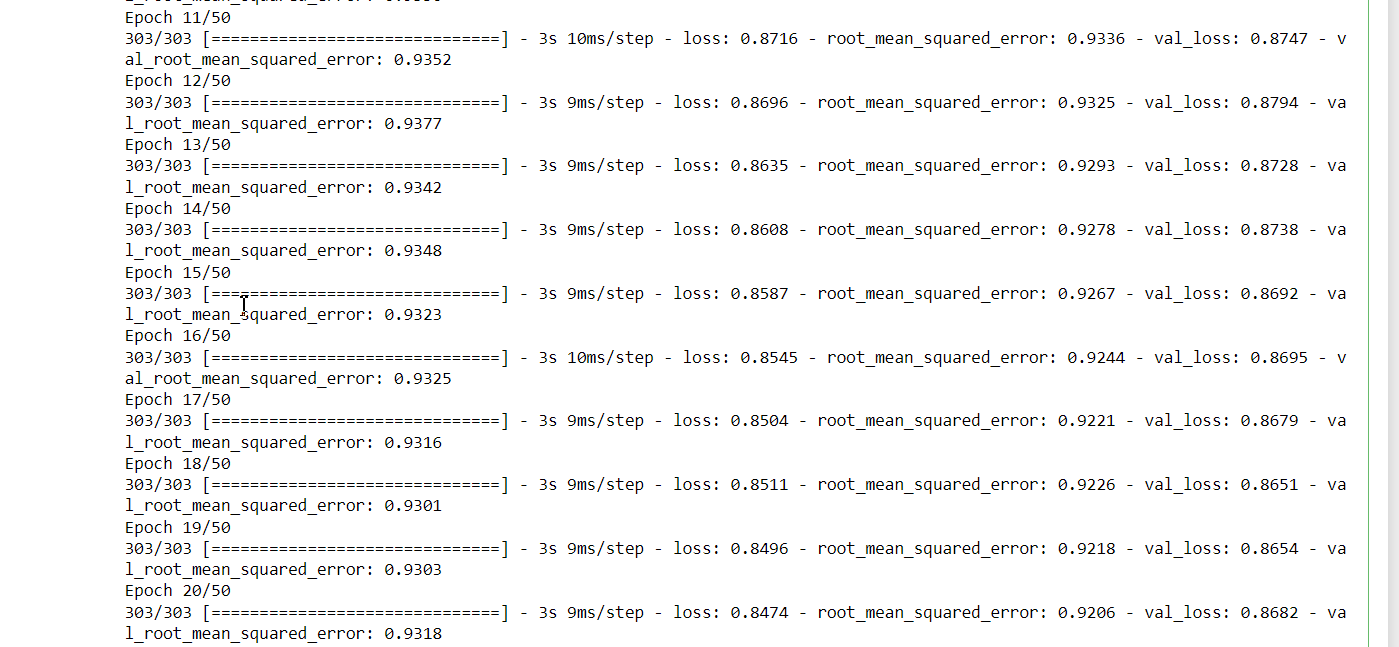
And for the model compilation, we have found the a higher learning rate, usually around 0.1, works better than a small one. The optimizer has also been changed to either Adagrad or Adadelta, that work better than Adam for our model.

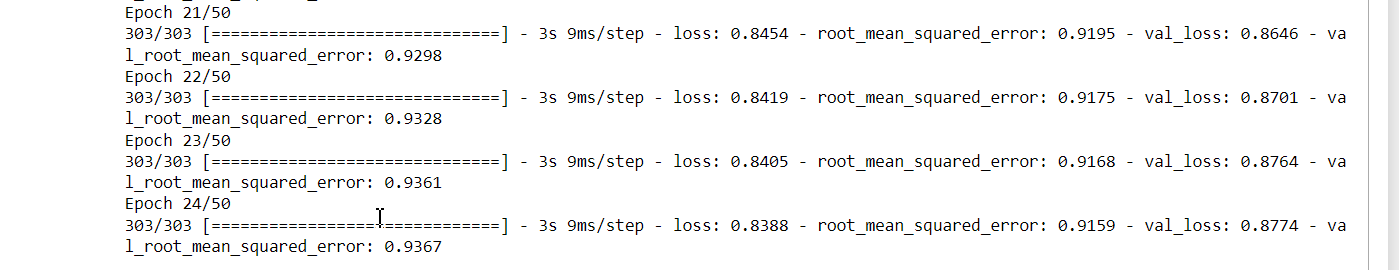
The callbacks used were the same as the ones from the other questions of the TP.



Here are our executions:







Our minimum RMSE for this execution is 0.**9298**. The model, as well as the .IPNYB file, are attached to the TP file.