STATS 415 Kaggle Report - Group 20

Alexander Hawthorne, Evelyn Paskhaver, Jeanne Yang, Li Yuan
Kaggle Account: lygitdata

Method

Data Cleaning

We employed one-hot encoding to transform categorical variables into a numeric format to ensure their inclusion in the analysis. We created dummy variables for predictors `self_eval`, `teacher_eval`, and `district`. After applying one-hot encoding to these categorical variables, the data frame has 65 predictor columns. We scaled the categorical dummy variable columns to ensure uniformity and comparability among the different variables.

Neural Network Model Structure

Neural network is a flexible method effective in handling high-dimensional data. When trained appropriately, a neural network generally exhibits good performance on unseen data and can achieve high prediction accuracy. This is the reason we chose Neural Networks to do this task.

Our neural network model is designed with TensorFlow and Keras. The model architecture was defined as a sequential stack of layers, including one dense layer with Exponential Linear Unit activation (TensorFlow, 2023), one batch normalization layer, which normalizes outputs from the previous layer (TensorFlow, 2023), and the final linear output layer. The core of the training is the usage of mini-batch stochastic gradient descent with a maximum epoch of 300.

An AdamW optimizer (TensorFlow, 2023) was utilized to compile the model with hyperparameters such as learning rate, decay steps, and decay rate.

We used the adjusted R-squared as the validation metric of the model's performance. For robust estimation of the model's performance, we employed 20-fold cross-validation during the training process.

Moreover, we set the early stop for the training process with 20 patience, meaning that we will stop the training if the next 20 validation adjusted R-squares of the model are a lower R-squared than the previous maximum adjusted R-squared, and we used model checkpointing (TensorFlow, 2023) to save the best-performing model based on the validation adjusted R-squared.

Hyperparameter Tuning

For the hyperparameters used in the neural network model, we employed a grid search with 20-fold cross-validation (Scikit-learn, 2019) on the neural network model we defined in the previous section to find the combination of parameters that would yield the best-performing model based on the adjusted R-squared. The lists of hyperparameter values we tested are as follows:

- Learning rate = [0.001, 0.002, 0.003, 0.004, 0.005]
- Dense layer units = [2, 3, 4, 5, 6]
- Decay steps = [1000, 2000, 5000, 10000, 20000]
- Decay rate = [0.8, 0.855, 0.9, 0.955]
- Batch size = [32, 64, 128, 256, 512]

After evaluating all the 2,500 combinations of hyperparameters, we found the set of hyperparameters that performed the best as follows:

- Learning rate = 0.001
- Dense layer units = 2
- Decay steps = 20,000
- Decay rate = 0.955
- Batch size = 512

The final model we used, which is generated by the Keras, is shown in Table 1 below.

Table 1: Summary of the Final Model We Used for Submission

Layer (type)	**Output Shape**	**Param #**
dense_0 (Dense)	(None, 2)	130
batch_normalization_0 (BatchNormalization)	(None, 2)	8
dense_1 (Dense)	(None, 1)	3
Total params: 141 (564.00 Byte) Trainable params: 137 (548.00 Byte) Non-trainable params: 4 (16.00 Byte)		

Contributions

Alexander Hawthorne: Worked on data cleaning and one-hot encoding of categorical predictors, and contributed to the development of the model structure and layers.
Evelyn Paskhaver: Contributed greatly to the hyperparameter tuning phase, particularly in the use of the grid search method, and assisted in the development of the neural network.
Jeanne Yang: Assisted in the hyperparameter tuning phase and grid search understanding, and contributed significantly to the detailed explanation of our methods in the report.
Li Yuan: Led the development of the neural network architecture and hyperparameter tuning phase, and contributed to the accuracy and detail of the methods in the report.

References

Scikit-learn. (2019). *sklearn.model_selection.GridSearchCV — scikit-learn 0.22 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

TensorFlow. (2023, September 3). *Training checkpoints | TensorFlow Core*. TensorFlow. https://www.tensorflow.org/guide/checkpoint

TensorFlow. (2023, September 27). *tf.keras.activations.elu | TensorFlow Core v2.5.0*. TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/activations/elu

TensorFlow. (2023, September 27). *tf.keras.optimizers.AdamW | TensorFlow v2.14.0*. TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/AdamW

TensorFlow. (2023, October 4). *tf.keras.layers.BatchNormalization | TensorFlow Core v2.4.1*. TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization