STATS 415 Fall 2023 **Kaggle Report**(December 3, 2023)
Group 7: Jiahe Huang, Jingjia Peng, Xinhe Wang
Kaggle Account: Jiahe Huang

1 Summary

Students' performance on standardized tests is a high concern in education. In our Kaggle Report, we develop a prediction of students' performance based on objective evaluation and scientific measurement[1]. We first preprocess data using **feature selection** and **One-Hot Encoding**. After that, we apply **gradient boosting regressor** and **neural network** separately and then combine them to give the prediction. We also perform cross-validation to fine-tune our model parameter. Our prediction achieves $R^2 = 0.886$ on the Kaggle test dataset which is powerful.

2 Customized Feature Selection

2.1 Why we use this

We treat `self_eval`, `teacher_eval` and `district` as categorical variables. Even though they are given in numerical value, we should separate them into different categories to better explain the group variance.
Besides, we still have `SRP i`, a time-series measurement of the same variable. After visualizing `SRP i` of several students, we found that `SRP i` behaves like a sinusoidal wave. This means we should not simply regress with raw data of 50 samples on a wave, but explore the wave feature for valuable information.

2.2 How we use this

First, we use **One Hot Encoding** for the three categorical predictors. This is implemented by creating dummy variables for every group in predictors, then setting "1" if the observation belongs to this group and "0" otherwise.
Then, we replace 50 columns of `SRP i` with the following wavelet features.

- `Average`: average SRP level over 50 days.
- `Max` and `Min`: peak and trough value of SRP value.
- `Amplitude`: gap between peak and trough.
- `Std`: standard variance of the samples
- `LatestAvg`: Average value of the latest 5-day SRP level before the exam.
- `LatestAmp`: amplitude of the latest 5-day SRP level before the exam.
- `LatestStd`: standard variance of the latest 5-day SRP level before the exam.

After transforming predictors, we perform **standard scaling** on each dimension.

3 Gradient Boosting Regressor

3.1 Why we use this

Boosting is an ensemble learning method that combines several weak regression tree models to provide a strong predictor. Gradient boosting here uses a **gradient descent** algorithm to minimize the loss function, which improves the model accuracy and training speed. We first use gradient boosting because the *tree-based* structure explores the feature's importance well by splitting a subtree by the most significant feature at a time. It provides more flexibility than linear regression to explore the *non-linear relationship*.

3.2 How we use this

We fine-tune the model parameter: the loss function for gradient descent is `square error`; the learning rate of gradient descent is 0.1; `max depth` of individual regression trees is 3 and `number of trees` is 150. These parameters achieve the best performance in **cross-validation**, which is visualized in Fig 1.

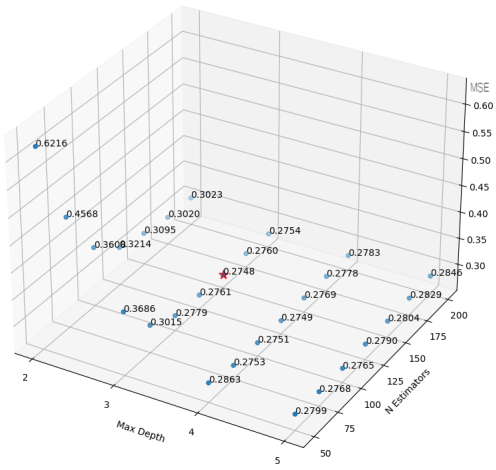

Figure 1: Mean Square Error on the test set under different combinations of `max depth` and `number of trees`, minimum MSE is marked by a red star.

4 Neural Network

4.1 Why we use this

Deeping learning is flexible in handling high-dimensional data and capable of learning complex, undiscovered patterns. We use a neural network here in the mind that there must exist some *hidden features and correlations* not extracted in the feature engineering and boosting model. The neural network is a good compensation for this shortcoming.

4.2 How we use this

We fine-tune the model parameter and structure as follows.

- We add **random noise** to 10% of training data. This improves the robustness of the model since slightly altering the data helps the model to generalize better rather than memorizing specific data points.
- The network consists of an input layer (11 input units), a hidden layer with 256 neurons, and an output layer (1 output unit). We randomly **drop off** 30% of the neurons to prevent overfitting. This drop-off works because it forces the model not to rely too heavily on any specific neuron.
- Loss function for training: **negative R-square**.

- Activation function of hidden layer: sigmoid function with ℓ_2 **regularization** and tuning parameter of 0.001.
- We add Adam optimizer with a learning rate of 0.0005. An introduction of this popular optimizer is cited here[2].
- We train 500 epochs in total, and use a batch size of 256 in each epoch.
- Finally, we introduce **early stopping**, a form of regularization to avoid overfitting. It stops the training if the test error begins to rise, which indicates that the model starts to learn specific patterns in the training data rather than generalize.

All of the hyper-parameters are selected by cross-validation with a $0.8 : 0.2$ train-test split.

5 Combination for Prediction

We combine the result of the two models by **taking the average** of their predicted `y`. The reason why we pick up both tree model and deep learning is that they contribute well to exploration in a different field: boosting intensively rely on feature engineering and emphasize important features, while neural network learn hidden patterns that we haven't discovered in the feature selection part. By combining them, we obtain a *flexible, comprehensive but not overfitting* result.

6 Discussion and Conclusion

Advantages: In this report, we build our prediction model by feature engineering, gradient boosting, and neural networks. Our result has *appropriate flexibility* by conducting noise addition, dropping-off, and regularization in the deep learning part, as well as *good generalizability* by using cross-validation in parameter tuning of both models.
Future work: The *size of the training data set* limits the performance of training and possibly leads to overfitting. We should collect more records of subjects, more sampling points of `SRP i` and more variables such as `history test score`, to give a more accurate prediction of `y:performance on standardized test`.

7 Acknowledgement

Thanks to our professor Jackson Loper and graduate student instructors Gabriel, Sunrit, and Yidan for their help. All group members equally contribute to the whole project, detailed task allocation is recorded in the open-ended report.

References

[1] STAT415 Teaching Team, "Final Project Stats 415 Fall 2023,"`https://www.kaggle.com/competitions/final-project-stats-415-fall-2023` Accessed Dec. 2, 2023.

[2] Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization." ArXiv, 2014, `https://doi.org/10.48550/arXiv.1412.6980`. Accessed Dec. 2, 2023.