Big Data Intelligence: Methods and Technologies
Assignment 2: Hyper-Parameter Tuning

Elsa Scola Martín **Liana Mehrabyan**

1 Introduction

Given a dataset from the Kaggle competition *"House Prices: Advanced Regression Techniques"* [1] the goal of this assignment is to apply different algorithms and optimization techniques to predict House Prices based on attributes such as:

- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access
- Etc.

2 The Kaggle Competition

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

The dataset is composed of 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa.

The objetive of the competition is to predict the sales price for each house, therefore, for each Id in the test set, the value of the SalePrice variable must be predicted.

Submissions are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of predicted value and the logarithm of the observed sales price.

3 Approaches

3.1 Default Parameters

As a first step, DecisionTreeRegressor() and KNeighborsRegressor() algorithms were applied with their default parameters and evaluated using holdout method (2/3 of data for training and 1/3 for validation). This resulted in training error of almost 0 and validation $RMSE_{DT} = 0.045$ for Decision Tree and $RMSE_{KNN} = 0.030$ for KNN.

To have a more stable evaluation, 2-fold cross-validation was also used to evaluate the mean value for $RMSE$ for 2 iterations. The same approach will be used to evaluate scores for hyper-parameter tuning. As a result, for default parameters, mean $RMSE_{DT} = 0.048$ and $RMSE_{KNN} = 0.036$. These will be benchmark values we will try to optimize using hyper-parameter tuning.

3.2 Random Search

Using random search to optimize $min\_samples\_split$ and $criterion$ parameters in Decision Tree Regressor resulted in $RMSE_{DT} = 0.046$ with optimal parameters $min\_samples\_split = 0.1$ and $criterion = mse$. I.e. we were able to optimize the metric by 0.002.

As for KNN regressor, $RMSE_{KNN} = 0.034$ with 8 neighbors, $p = 1$ (i.e.Manhattan distance) again resulting in 0.002 improvement. As a result, this parameter setting had a Kaggle score improvement of $RMSE = 0.1698$ (Figure 1) This is the best result achieved using the algorithms proposed for this assignment.

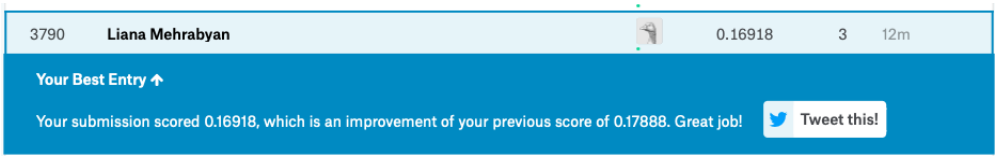

Figure 1: KNN Rank

3.3 scikit-optimize

Using scikit-optimize module, BayesSearchCV method was used for hyper-parameter tuning of the same regressors.

In case of both RandomizedSearchCV and BayesianSearchCV, a fixed number of parameter settings is tried out from the specified parameter space and not every single option, in our case 20 of them. However, RandomizedSearchCV chooses parameters randomly from given distribution, whereas BayesianSearch takes into account information on the hyperparameter combinations it has seen so far when choosing the next set to evaluate [2].

Comparing execution times for Random Search and skopt BayesianSearch yielded the following results: $t_{DT} = 0.2058$, $t_{KNN} = 2.8791$ for RandomizedSearch and $t_{DT} = 12.3086$, $t_{KNN} = 26.5392$ for BayesianSearch. Considering that both methods resulted in the same optimal hyperparameters, it is was optimal to use RandomizedSearch for the following problem. Such long execution time of Bayesian method can be explained by multiple reasons, one of which is calculation of a kernel function between all pairs of observations.

3.4 Further Improvement Attempts

Suspecting potential over fitting of the model, max_depth parameter was tuned from different range of values. The resulting optimal value of 60, however, didn't improve neither the $RMSE$ of validation nor the Kaggle score.

Having high number of features, which could also contribute to model complexity, Principal Component Analysis was used to reduce data dimensions. Trying out different number of components didn't result in a better score either. This could be explained by the fact that the explained variance ratio captured by small amount of components was very low (< 0.2 for 10 components).

Finally, experimenting with GradientBoostingRegressor parameters resulted in a Kaggle score of 0.14, best result we were able to achieve at this point (Figure 2).

Figure 2: GB Rank

References

[1] House Prices: Advanced Regression Techniques.
Available[Online]:
https://www.kaggle.com/c/house-prices-advanced-regression-techniques

[2] Kraus M., Using Bayesian Optimisation to reduce the time spent on hyperparameter tuning
Available[Online]: https://bit.ly/35ogKHj