COMP 468 Machine Learning in Python, Spring 2023
Instructor: Zafer Aydın
Homework 4

Setup

1. Login to your Kaggle account.
2. Click on the Code link.
3. Click on the New Notebook button.
4. Change the title of the page on the upper left corner so that it obeys this format: “ML in Python, Spring 2023, Homework 4, Your Name Surname” (e.g. ML in Python, Spring 2023, Homework 4, Zafer Aydın).
5. Click on the “Add Data” button on the upper right corner.
6. Click on the “Search keyword or URL text box” below Add Data. Search for “Housing Prices Competition for Kaggle Learn Users” by entering this text to the search box.
7. Go to page numbered 2.
8. Click on the + button next to the dataset uploaded by user A.P. to add dataset. When you bring your mouse on the + button you should see Add Dataset not Add Notebook Output.
9. Click on the × button next to Add Data at the upper right corner to close the window for adding data.
10. Click on the +Code button at the lower left side of the code cell to add a new code cell.
11. You can start from the template code called ML in Python, Spring 2023, Homework 4, Template (which is made available as ml-in-python-spring-2023-homework-4-template.ipynb).

Assignment

Implement the steps below on top of the template code. Put the code of each question into a separate cell (e.g. question 1a goes to one cell, question 1b goes to another cell, question 2 goes to another cell, etc.)

1. Define an XGBoost regressor model with default parameters, n_jobs=-1, random_state=0.
(a) What are default values of n_estimators and learning_rate? You can access default value of n_estimators as print(model.n_estimators) once you define a model. Include your commands for printing default n_estimators into your notebook. The default value of learning_rate is given in the tutorial page of XGBoost lesson. Include these default values as markdown cell into your notebook.

(b) Train the model on train set (X_train_imputed, y_train) and compute predictions on test set (X_test_imputed). Store the predictions as a pandas data frame, which will have an Id column that contains X_test.index as the values and a SalePrice column that contains predictions on test set as the values. Save this data frame as a csv file such as submission_default.csv by setting index=False. Submit your predictions to competition. You can find instructions for submitting a csv file to competition in question 4 of homework 2. The link of the competition is https://www.kaggle.com/competitions/home-data-for-ml-course. Enter your leaderboard score as a markdown cell to your notebook.

2. Optimize the n_estimators and learning_rate hyper-parameters of XGBRegressor by grid search optimization. Use GridSearchCV class of scikit-learn, which can be accessed at https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. The link includes some code examples for using GridSearchCV. Implement the following steps

- Import GridSearchCV from sklearn.model_selection.
- Define a Python list called n_estimators_values that contains 100, 500, and 1000 as the values.
- Define a Python list called learning_rate_values that contains 0.01, 0.05, and 0.1 as the values.
- Define a Python dictionary called parameters that contains 'n_estimators' and 'learning_rate' as the keys and n_estimators_values and learning_rate_values as the values of the dictionary.
- Define a GridSearchCV object called opt by sending XGBRegressor model and the Python dictionary called parameters as input to the constructor method called GridSearchCV. Note that the default value of 5 is used for cv parameter of GridSearchCV, which will perform a 5-fold

cross-validation on training set for each hyper-parameter combination in parameter grid once we call the fit method in the next step.
- Call the fit method of GridSearchCV object opt by passing training set as input (X_train_imputed and y_train). This method will optimize the hyper-parameters and once it finishes it will train the XGBRegressor model using the optimum hyper-parameters to make it available for computing predictions.
- Print the optimum n_estimators and learning_rate found. You can access the optimums through the best_params_ attribute (e.g. opt.best_params_). Include these optimums as a markdown cell.
- Call the predict method of GridSearchCV object opt and compute predictions on test set (X_test_imputed). Submit your predictions to competition as in question 1b. You can save your csv file as submission_grid_search.csv. Enter your leaderboard score as a markdown cell to your notebook.

3. Optimize the n_estimators and learning_rate hyper-parameters of XGBRegressor by randomized search optimization. Use RandomizedSearchCV class of scikit-learn, which can be accessed at https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html#sklearn.model_selection.RandomizedSearchCV. The link includes some code examples for using RandomizedSearchCV. Implement the following steps
- Import RandomizedSearchCV from sklearn.model_selection, randint from scipy.stats, and loguniform from scipy.stats.
- Define a Python dictionary called distributions as in the code examples for randomized search. Set n_estimators to randint(100,1001) and learning_rate to loguniform(0.01, 0.1). This way randomized search algorithm will sample n_estimators as a random integer in range 100 to 1000 from randint distribution and learning_rate as a real-value in range 0.01 to 0.1 from loguniform distribution.
- Define a RandomizedSearchCV object called opt by sending XGBRegressor model and the Python dictionary called distributions as input to the constructor method called RandomizedSearchCV and by setting n_iter to 9, n_jobs to -1, and random_state to 0. This will make a total of 9 iterations for randomized search (a total of 9 hyper-parameter combinations will be tried) and will perform a 5-fold cross-validation on training set in each iteration.
- Call the fit method of RandomizedSearchCV object opt by passing training set as input (X_train_imputed and y_train). This method will optimize the hyper-parameters and once it finishes it will train the XGBRegressor model using the optimum hyper-parameters to make it available for computing predictions.
- Print the optimum n_estimators and learning_rate found. You can access the optimums through the best_params_ attribute (e.g. opt.best_params_). Include these optimums as a markdown cell.
- Call the predict method of RandomizedSearchCV object opt and compute predictions on test set (X_test_imputed). Submit your predictions to competition as in question 1b. You can save your csv file as submission_randomized_search.csv. Enter your leaderboard score as a markdown cell to your notebook.

4. Define an XGBRegressor model with n_estimators set to 1000, learning_rate to 0.05, n_jobs to -1, and random_state to 0. Train the model on the training set (X_train_imputed, y_train) and compute predictions on test set (X_test_imputed). Convert the predictions to a pandas data frame and save it as a csv file called submisson_selected.csv. Submit your csv file to competition as in question 1b. Enter your leaderboard score as a markdown cell to your notebook.

5. Import cross_val_score from sklearn.model_selection. Implement a Python function called score_dataset that performs the following
- Receives train set (i.e. X_train_set and y_train_set), n_estimators, and learning_rate parameters of XGBoost as input.

- Sets the n_estimators parameter of XGBoost to n_estimators that is input to your Python function, learning_rate parameter of XGBoost to learning_rate that is input to your Python function, n_jobs to -1 and random_state to 0.
- Performs a 5-fold cross-validation on training set (X_train_set and y_train_set) using cross_val_score and XGBoost regressor as the model.
- Computes and returns the average mean absolute error (MAE) as the output obtained as the average of five scores from each fold of cross-validation.

6. Compute and print the average MAE scores by calling the score_dataset function (i.e. by performing 5-fold cross-validation) using the training set (X_train_imputed and y_train) for the following scenarios
- n_estimators and learning_rate set to defaults as in question 1
- n_estimators and learning_rate set to optimums found by grid search as in question 2
- n_estimators and learning_rate set to optimums found by randomized search as in question 3
- n_estimators and learning_rate set to values used in question 4

Include these MAE scores as a markdown cell to your notebook.

7. Fill the table below that includes your cross-validation (computed on training set) and test set scores and include to your solution document (e.g. a Word file).

n_estimators and learning_rate	CV MAE on Training Set	Test Set MAE
Defaults		
Optimized by grid search		
Optimized by randomized search		
n_estimators=1000, learning_rate=0.05		

8. Include your answers for the following questions to your solution document (e.g. a Word file).

(a) Which approach gives the best cross-validation MAE score? Which approach gives the best test set score (i.e. leaderboard score)?
(b) Can we say that optimizing hyper-parameters gives better results than using default values?
(c) If the approach that gives the best cross-validation MAE score is different from the approach that gives the best leaderboard score what can be the reason for this? Hint: Can you consider the training set as small?

Submission

Once you finish, click File and Download notebook. Submit your notebook with .ipynb extension to Canvas. Submit your solutions to questions 7 and 8 as a Word file to Canvas.