 Data and AI Expert Labs and Learning

Build a Rapid Prototype with Watson Studio AutoAI

Summary

In this lab you will:

- Run a Watson Studio AutoAI Experiment
- Generate a Python notebook from the AutoAI
- Try different strategies in the notebook for data preprocessing and observe changes in the pipeline performance
- Try different estimators in the notebook and observe changes in the pipeline performance
- Try different hyperparameter values and observe the effects on the pipelines
- Observe different evaluation measures for the AutoAI-generated pipelines
- Deploy a model using Watson Machine Learning and test it

Instructions

1. Create a new project in Watson Studio

    a. Select Create an empty project
    b. Name your project
    c. Select your Cloud Object Storage from the Select storage dropdown
    d. Click Create to finish creating the empty project

2. Add a data source to project

    a. Click the 0001 icon on the menu bar at the top right
    b. Click on browse to add a local .csv file
    c. Upload the provided file Real estate valuation data set.csv

3. Explore the data

    a. You should see the file listed as a Data asset on the project page. Select it to open a Preview and Profile of the data.
    b. Review the description of the data source and the individual attributes here: http://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set
    c. Click Profile (next to Preview) in the menu bar of Watson Studio and select Create profile
    d. A data profile will be generated. You may need to refresh the page in your browser to show the profile. Take a moment to explore the provided statistics.

4. Add an AutoAI Experiment

    a. Return to the project menu by clicking the name of the project in the top left menu bar
    b. Click Add to Project +
    c. Select AutoAI Experiment
    d. Give the experiment a name
    e. Under Associate Services, select Associate a Machine Learning service instance
    f. Click on WatsonMachineLearning from the drop down menu in the new tab and Select
    g. In the original tab, select the Reload button
    h. Click Create

5. Add a data source to the AutoAI Experiment

    a. Click Select from project
    b. Select Real estate valuation data set.csv and click Select asset
    c. Once the columns have been parsed, click on the column Y house price of unit area to select it as a prediction column
    d. Notice that the experiment has been inferred to be a Regression and the data types of the features have been automatically detected.
    e. Click Experiment Settings. Observe the default training and holdout split and the ability to deselect columns for use in the model.
    f. Click Prediction. View the options for optimized metrics. Leave the default experiment settings for now. Click Cancel.
    g. Click Run experiment.
    h. Watch the animation and the changing Pipeline leaderboard as the experiment runs
    i. When the experiment finishes running (should take slightly over 2min), notice the two algorithms that have been selected as top performing. Are these algorithms that you would have tried on your own?

6. Save the top-ranked pipeline as a Notebook

    a. Rollover the mouse on the top-ranked pipeline to show the Save as button. Click Notebook to save that pipeline as a Notebook.
    b. Click Create on the New notebook screen
    c. In the rest of this lab, we'll work our way through the notebook examining how each step of the Data Science workflow is represented in the notebook.
        i. Review the notebook and run each cell.
        ii. In particular, we'll focus on the Compose Pipeline cell.
        iii. If you're not familiar with `scikit-learn`'s `Pipeline` class, review the documentation here: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
        iv. Documentation for IBM's autoai_libs can be found here: https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-lib-python.html
    d. Under 8. Fit pipeline, predict on Holdout set, calculate score, perform cross-validation, notice the output indicating the pipeline's accuracy on the holdout set and for cross-validation. These are the values that we will observe as we make changes to the notebook.

7. Make changes to preprocessing strategies in the notebook

    a. Save this version of the notebook
    b. Return to the Preview of the .csv file to review the feature attributes
    c. In the Compose Pipeline cell of the notebook, find the code block under the comment # composing steps for preprocessor Pipeline
    d. Find the line that appends num_scaler to the pipeline using the autoai_libs OptStandardScaler
    e. Review the parameters to the OptStandardScaler from IBM's documentation: (also see next page) https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-lib-python.html
    f. Set the parameters `num_scaler_with_mean`, `num_scaler_with_std`, and `use_scaler_flag` to `True`

g. Re-run the notebook. Does it affect the output scores for holdout and cross-validation?
    i. Recall that the pipeline was optimized with the given inputs and we've now changed the inputs. We should be aware of this as we continue to make changes to the notebook. Performing optimization with these new inputs could give us different results.
h. Save this version of the notebook.

8. Make changes to estimator in the notebook

    a. Find the code block under the comment `# assembling cognito_ Pipeline`
    b. Find the line that appends the estimator to the pipeline. Notice the XGBRegressor estimator is appended here.
    c. Recall from the Pipeline leaderboard for this AutoAI experiment that the ExtraTreesRegressor also performed well for this dataset. Replace the XGBRegressor with the ExtraTreesRegressor from scikit-learn. Use `sklearn.ensemble.ExtraTreesRegressor()` with the default parameters.
    d. Re-run the notebook and notice the output for the holdout and cross-validation scores. Save this version of the notebook and revert back to the previous version.

9. Explore Feature Engineering

    a. Return to the AutoAI Experiment Pipeline leaderboard
    b. Click on the top-ranked pipeline
    c. On the left menu bar, select Feature Transformations
    d. Examine the newly created features
    e. Click on Feature Importance from the left menu bar. This importance is determined by a decision tree-based algorithm.
    f. Notice NewFeature_6 uses the transformations sqrt and sin.
    g. Inspect the Cognito transformations via the notebook
        i. In the Compose Pipeline cell, find the code block with the comment `# composing steps for cognito Pipeline`
        ii. Notice the two transforms that are applied are the `sqrt` and `sin` functions.
        iii. Look up the documentation for the `TA1` and `FS1` classes from autoai_libs here: https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-lib-python.html

10. Examine hyperparameter settings

    a. Open a version of the notebook which uses the XGBRegressor
    b. Find the line of code in the cell titled Compose Pipeline which appends the estimator
        i. You may want to add some line breaks here to make it easier to read this line of code.

   ii. The optimization procedure is not exposed to the notebook. The values that are set for the hyperparameters for the XGBRegressor in this notebook are the result of the optimization.
c. Compare the hyperparameters here to the default hyperparameters for the XGBRegressor. View the documentation for the estimator here: https://xgboost.readthedocs.io/en/latest/parameter.html
d. Notice that the hyperparameter `max_depth` has a default value of 6, but has been set to 11. Change the `max_depth` to 6 and re-run the notebook to observe changes in the pipeline performance.
e. The default `learning_rate` is 0.3, but the hyperparameter has been set to 0.2. Try the value 0.3 instead and observe how the performance changes.
f. Save the notebook version.

11. Examine performance metrics

    a. Return to the AutoAI experiment with the Pipeline leaderboard
    b. Observe how the pipelines are ranked by RMSE
    c. Use the toggle button at the top right to select Holdout instead of Cross validation. Observe how this changes the ranking of the pipelines.
    d. Change the option in the Rank by dropdown menu next to the Holdout/Cross validation option to F1 instead of Accuracy. Observe how this changes the ranking of the pipelines for both Holdout and Cross validation.
    e. Click on the Pipeline comparison option and view it for both Holdout and Cross validation options to compare metrics across pipelines
    f. Select a pipeline for which you would like to deploy the model
    g. Click this pipeline to show more details about Model Evaluation.

12. Deploy a model

    a. Click the Save As button at the top right and select Model
    b. Give a Description if desired and click Save
    c. You should get a notification on the top right that the model has been saved and an option to View in project
    d. The Overview tab will give a Summary of your model and Input Schema. Select the Deployments tab.
    e. Click Add Deployment + at top right.
    f. Give a Name. Notice that the model will be deployed as a Web service. Click Save.
    g. Notice that the status is Initializing. Once the status changes to Ready, click the ellipsis under Actions and select View.
    h. Select the Test tab.
    i. Enter the following values into the form and click Predict
        i. X1 transaction date = 2012.667
        ii. X2 house age = 18
        iii. X3 distance to nearest MRT station = 1084
        iv. X4 number of convenience stores = 4
        v. X5 latitude = 24.96
        vi. X6 longitude = 121.53
        (There isn't any new data to test the model on, but these are the mean values from each feature rounded to the nearest integer where appropriate.)
    j. Observe the prediction value

Further Readings

Alberto Costa and Giacomo Nannicini. 2018. RBFOpt: an open-source library for black-box optimization with costly function evaluations. *Math. Prog. Comp.* 10, 59 (Dec 2018), 597–629. DOI: https://doi.org/10.1007/s12532-018-0144-7

AutoAI implementation details. (January 2020). Retrieved April 16, 2020 from https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-details.html

Udayan Khurana, Horst Samulowitz, and Deepak Turaga. 2018. Feature Engineering for Predictive Modeling Using Reinforcement Learning. In *32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, February 2 – 7, 2018, New Orleans, LA. AAAI Press, Palo Alto, CA, 3407-3414. https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16564

Udayan Khurana, Deepak Turaga, Horst Samulowitz and Srinivasan Parthasrathy. 2016. Cognito: Automated Feature Engineering for Supervised Learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, December 12 – 15, 2016, Barcelona, Spain. IEEE, Piscataway, NJ, 1304-1307. https://doi.org/10.1109/ICDMW.2016.0190

Ashish Sabharwal, Horst Samulowitz, and Gerald Tesauro. 2016. Selecting near-optimal learners via incremental data allocation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (*AAAI'16*). AAAI Press, 2007–2015. DOI: https://dl.acm.org/doi/10.5555/3016100.3016179