Hyperparameter DB

Sharwari Makarand Gothe, Sharvari Karnik, Cassian Gonsalves
{gothe.s, karnik.sh, gonsalves.c}@husky.neu.edu
INFO 6510, Spring 2019, Northeastern University

Abstract*—In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed. In any machine learning algorithms, these parameters need to be initialized before training a model. Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained. Our aim is to find proper hyperparameter with proper tuning for our dataset which would help the database team in modelling the database schema in an efficient way. We would create H2O models for this dataset for getting proper hyperparameters. In this paper the dataset is on Bank Marketing in which the goal is to predict if the client will subscribe a term deposit.

Keywords*—Hyperparameter, H2O, Bank Marketing, Prediction, Classification.

1. Introduction

1.1 Background

Hyperparameters are variables that we need to set before applying a learning algorithm to a dataset. In machine learning scenarios, a significant part of model performance depends on the hyperparameter values selected. The goal of hyperparameter exploration is to search across various hyperparameter configurations to find the one that results in the optimal performance. The challenge with hyperparameters is that there are no magic number that works everywhere. The best numbers depend on each task and each dataset. The hyperparameter database to be developed as a part of this project is an open resource with algorithms, tools, and data that allows users to visualize and understand how to choose hyperparameters that maximize the predictive power of their models. Phase I of the project involves selecting a unique dataset containing predicted target variables, hyperparameters, meta-data etc. by running different models (with varying hyperparameters) on it using H2O.

Hyperparameters can be divided into 2 categories:

1. Optimizer hyperparameters
   - They are related more to the optimization and training process - If our learning rate is too small than optimal value then it would take a much longer time (hundreds or thousands) of epochs to reach the ideal state
   - If our learning rate is too large than optimal value then it would overshoot the ideal state and our algorithm might not converge.
2. Model specific Hyperparameters
   - They are more involved in the structure of the model

Currently, the hyperparameter database analyzes the effect of hyperparameters on the following algorithms: Distributed Random Forest (DRF), Generalized Linear Model (GLM), Gradient Boosting Machine (GBM). Naïve Bayes

Classifier, Stacked Ensembles, XGBoost and Deep Learning Models (Neural Networks).

1.2 Dataset

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

- Age (numeric)
- Job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
- Marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown' ; note: 'divorced' means divorced or widowed)
- Education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
- Default: has credit in default? (categorical: 'no', 'yes', 'unknown')
- Housing: has housing loan? (categorical: 'no', 'yes', 'unknown')
- Loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

This is a classification dataset. The project is to determine whether the client will subscribe or not. The target variable is y - has the client subscribed a term deposit? (binary: 'yes', 'no')

2. Data Cleaning and Exploration

In this task, we are inspecting and auditing the data to identify the data problems, and then fix the problems. Different generic and major data problems could be found in the data might include:

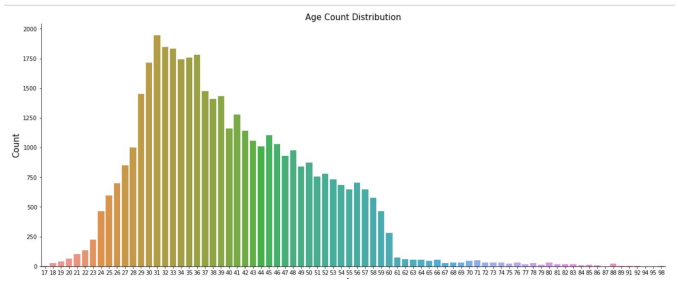
1. Lexical errors, e.g., typos and spelling mistakes
2. Irregularities, e.g., abnormal data values and data formats
3. Violations of the Integrity constraint.
4. Outliers
5. Duplications
6. Missing values
7. Inconsistency, e.g., inhomogeneity in values and types in representing the same data

The following libraries are used throughout the notebook for purpose of data cleaning and visualization

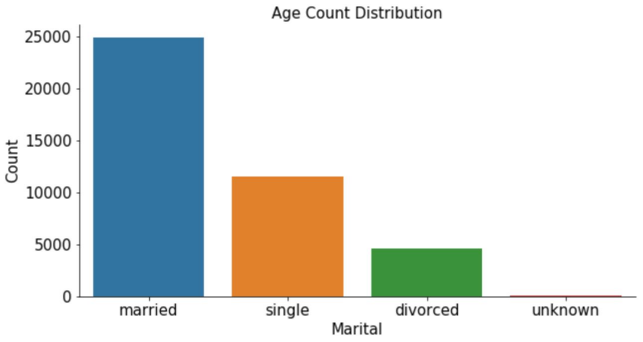
Serial No.	Package	Plots used in this kernel	Remark	Nature of plots
1	Matplotlib	1. vendor_id histogram, 2. store and fwd flag histogram	Matplotlib is oldest and most widely used python visualization package, its a decade old but still its the first name come to our mind when it comes to plotting. Many libraries are built on top of it, and uses its functions in the backend. Its style is very simple and that's the reason plotting is fast in this. It is used to create axis and design the layout for plotting using other libraries like seaborn.	
2	Seaborn	1.Violin plot (passenger count vs trip duration), 2. Boxplots(Weekday vs trip duration, 3. tsplot (hours, weekday vs avg trip duration), 4. distplots of lat-long, and trip_duration	Seaborn is my favorite plotting library (Not at all a fan of house greyjoy though :P) Plots from this package are soothing to eyes. Its build as a wrapper on matplotlib and many matplotlib's functions are also work with it.colors are amazing in this package's plots	
3	Pandas	1. Parallel coordinates (for cluster characteristics)	Pandas also offer many plotting functions and it's also a package built on matplotlib, so you need to know matplotlib to tweak the defaults of pandas. It offers Alluvial plots (which are nowhere near what R offers as alluvial plots) which are used in this notebook to show cluster characteristics.	
4	Bokeh	1. Time series plot (day of the year vs avg trip duration)	Bokeh is one great package which offers interactive plots, you can use bokeh with other libraries like seaborn, data-shader or holoviews, but bokeh its offers various different kind of plots. zoom, axis, and interactive legends makes bokeh different than others	Interactive
5	Folium	1.pickup locations in Manhattan, 2. cluster's location in the USA, 3. clusters location in Manhattan	This package offers geographical-maps and that too are interactive in nature. This package offers a different kind of terrains for maps- stemmer terrain, open street maps to name a few. you can place bubble at the locations, shift the zoom, and scroll the plot left-right-up-down and add interactions, for example - cluster plots shown in this notebook offers information about clusters like number of vehicles going out, most frequently visited clusters etc. *kaggle started supporting this package during this competition only*	interactive
6	Pygmaps	1. location visualizations 2. cluster visualizations	Pygmaps is available as archive package and can't even be installed using pip install command, but this package was the predecessor of gamps package but offers few great interactions which even gmaps doesn't offer. for example, a scattering of a cluster can be plotting with this one better than with gmaps. This package was way underdeveloped and developed version of it is know as gmaps yet, it was able to generate beautiful vizs. plots made my this package are best viewed in browsers.	interactive
7	Plotly	1.bubble plot	This is another great package which offers colorful visualizations, but some of these beautiful plots require to interact with plotly server so you need API key and it will call the API.	interactive
8	Gmaps	*To be updated*	gmaps provide great features like- route features, and we all are too used to gmaps, so feel like home.	interactive
9	Ggplot2	1. Weather plots of NYC for given period	gglots are now available in python as well, and its kind of in developing state and documentation is less of this package which makes it a little difficult but at the same time it provides very beautiful plots, so there is a tradeoff ;)	interactive
10	Basemaps	*Will not be added in this kernel*	As long as you are not developing any maps related library, there are no benefits of using basemaps. They offer many options but using them is difficult, due to lots of arguments, different style, and less amount of documentation, and many lines of codes sometimes will be required to plot a map.	
11	No package	1. heatmaps of NYC taxi traffic	Instead of depending on data-shadder, I tried plotting the heatmap of traffic data with a row image, you will get to know the basics of image processing(reading image, color schemes that's all :P) and how such basic exercise can result in traffic heatmap	
12	Datashader	1.locations' heatmap	If you really have a very large size of data that you want to plot on a map, data shader is one of the best easiest option available in marker. But I found that row image processing and generating plots using a scipy.mics or cv2 is considerably faster than using this package.	interactive
13	Holoviews	1. Pairplot for feature interaction.	Holoviews is another alternative for making interactive visualizations, this package offers artistic plots. But you need really good RAM etc to use this package, else the notebook will get hang. plots exported to HTML works perfectly fine	interactive

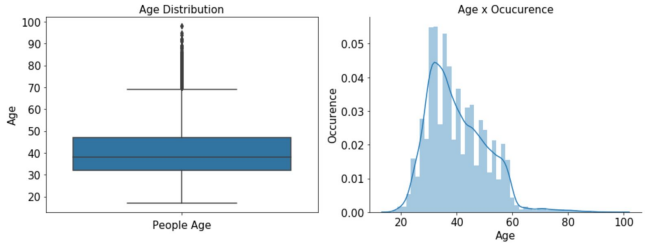
Libraries used in data cleaning and visualization

Figuring out some inference based on the age of the clients. Following are few graphs which will help in figuring out the distribution and effect of age on our independent variable.



Seaborn Plot to see the age count distribution

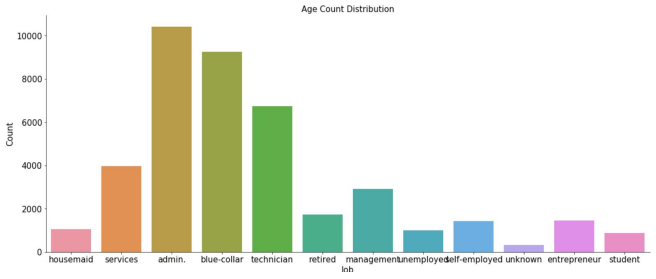




Age distribution Vs the occurrence count

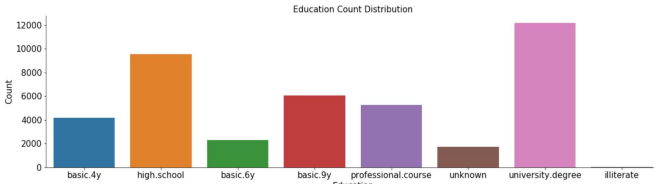
Based on the above graphs our conclusion says that in our opinion due to almost high dispersion and just looking at this this graph we cannot conclude if age have a high effect to our variable y, need to keep searching for some pattern. high middle dispersion means we have people with all ages and maybe all of them can subscript a term deposit, or not. The outliers was calculated, so my thinking is fit the model with and without them

Exploring Jobs:

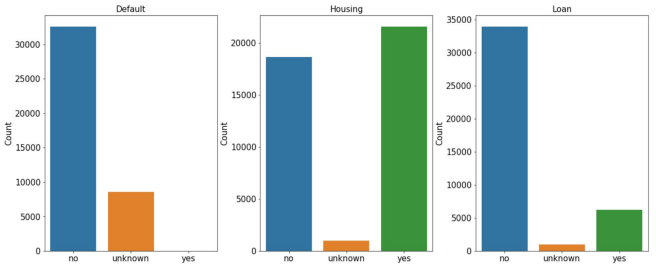


Exploring Marital Status:

Exploring Education data:



Exploring Default, Housing and Loan Data:



The above graphs and analysis shows that The age don't mean too much, has a medium dispersion and don't make sense relate with other variables will not tell any insight

Jobs, Marital and Education is probably the best analysis is just the count of each variable, if we relate it with the other ones its is not conclusive, all this kind of variables has yes, unknown and no for loan, default and housing.

Default, loan and housing, its just to see the distribution of people.

3. Algorithm and Code Source

H2O is an open source predictive analytics platform for data scientists and business analysts

who need scalable and fast machine learning. Unlike traditional analytics tools, H2O provides a combination of extraordinary math and high performance parallel processing with unrivaled ease of use. H2O intelligently combines the following features to solve today's most challenging business problems:

- Best of breed Open Source Technology
- Easy-to-use WebUI and Familiar Interfaces
- Data Agnostic Support for all Common Database and File Types
- Massively scalable Big Data Munging and Analysis
- Real-time Data Scoring

The H2O python module is not intended as a replacement for other popular machine learning frameworks such as scikit-learn, pylearn2, and their ilk, but is intended to bring H2O to a wider audience of data and machine learning devotees who work exclusively with Python.

H2O from Python is a tool for rapidly turning over models, doing data munging, and building applications in a fast, scalable environment without any of the mental anguish about parallelism and distribution of work.

In this project the dataset is cleaned and then passed to H2o to get all the possible models. The models are generated for various runtime. In this project the dataset has been run for 200, 700 and 1500 seconds.

For 200 seconds:

The dataset is passed to automl and H2o connects with its server to generate various models. The log file is generated to save all the logs created in the process. Log files will help to track the actions and the results in the future. Following image shows that the server is running at the given port number and the address at which the H2O server is connected successfully.

```
  Server is running at http://127.0.0.1:26213
Connecting to H2O server at http://127.0.0.1:26213... successful.
```

H2O cluster uptime:	04 secs
H2O cluster version:	3.14.0.1
H2O cluster version age:	1 year, 8 months and 8 days !!!
H2O cluster name:	H2O_from_python_Sharvari_Karnik_qf0ust
H2O cluster total nodes:	1
H2O cluster free memory:	1.745 Gb
H2O cluster total cores:	8
H2O cluster allowed cores:	8
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:26213
H2O connection proxy:	None
H2O internal security:	False
H2O API Extensions:	Algos, AutoML, Core V3, Core V4
Python version:	3.7.2 final

As mentioned earlier, each model does not generate separate meta-data. For every run-time a single metadata is generated which is stored in a separate file.

The leaderboard displays all the algorithms used to generate the models. The leaderboard for this dataset for the runtime of 200 seconds is as follows:

```
# Printing the leaderboard
aml_leaderboard_df
```

	model_id	auc	logloss
0	GBM_grid_0_AutoML_20190419_145419_model_0	0.948582	0.178329
1	StackedEnsemble_0_AutoML_20190419_145419	0.947018	0.198430
2	GBM_grid_0_AutoML_20190419_145419_model_1	0.946754	0.180883
3	GBM_grid_0_AutoML_20190419_145419_model_2	0.946714	0.182316
4	GBM_grid_0_AutoML_20190419_145419_model_3	0.944999	0.188787
5	XRT_0_AutoML_20190419_145419	0.943848	0.183741
6	DRF_0_AutoML_20190419_145419	0.941649	0.189263
7	GLM_grid_0_AutoML_20190419_145419_model_1	0.933566	0.213060
8	GLM_grid_0_AutoML_20190419_145419_model_0	0.933566	0.213060

Every model is stored in the database for future reference. The hyperparameters of the best model are found using further analysis techniques.

For 700 seconds

Similar to process followed for 200 seconds, the procedure is again followed for 700 seconds. There is a minute difference in the readings. Following images show the connection with H2O

server and the leaderboard generated by it in 700 seconds.

```
Server is running at http://127.0.0.1:15420
Connecting to H2O server at http://127.0.0.1:15420... successful.
```

H2O cluster uptime:	03 secs
H2O cluster version:	3.14.0.1
H2O cluster version age:	1 year, 8 months and 8 days !!!
H2O cluster name:	H2O_from_python_Sharvari_Karnik_t0weln
H2O cluster total nodes:	1
H2O cluster free memory:	1.745 Gb
H2O cluster total cores:	8
H2O cluster allowed cores:	8
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:15420
H2O connection proxy:	None
H2O internal security:	False
H2O API Extensions:	Algos, AutoML, Core V3, Core V4
Python version:	3.7.2 final

```
aml_leaderboard_df
```

	model_id	auc	logloss
0	GBM_grid_0_AutoML_20190419_150223_model_1	0.947737	0.179908
1	GBM_grid_0_AutoML_20190419_150223_model_0	0.947290	0.180794
2	StackedEnsemble_0_AutoML_20190419_150223	0.947071	0.198294
3	GBM_grid_0_AutoML_20190419_150223_model_4	0.947039	0.179225
4	GBM_grid_0_AutoML_20190419_150223_model_2	0.946913	0.180341
5	GBM_grid_1_AutoML_20190419_150223_model_1	0.944443	0.192693
6	GBM_grid_0_AutoML_20190419_150223_model_3	0.944122	0.188420
7	XRT_0_AutoML_20190419_150223	0.943559	0.183518
8	GBM_grid_1_AutoML_20190419_150223_model_3	0.942706	0.222069
9	DRF_0_AutoML_20190419_150223	0.939738	0.192561
10	GBM_grid_1_AutoML_20190419_150223_model_2	0.937204	0.203695
11	GLM_grid_0_AutoML_20190419_150223_model_0	0.933566	0.213060
12	GLM_grid_0_AutoML_20190419_150223_model_1	0.933566	0.213060
13	DeepLearning_0_AutoML_20190419_150223	0.928769	0.239359
14	GBM_grid_1_AutoML_20190419_150223_model_0	0.747353	2.794924

For 1500 seconds

Similar to process followed for 200 seconds, the procedure is again followed for 1500 seconds. There is a minute difference in the readings. Following images show the connection with H2O server and the leaderboard generated by it in 1500 seconds.

```
Server is running at http://127.0.0.1:52355
Connecting to H2O server at http://127.0.0.1:52355... successful.
```

H2O cluster uptime:	04 secs
H2O cluster version:	3.14.0.1
H2O cluster version age:	1 year, 8 months and 8 days !!!
H2O cluster name:	H2O_from_python_Sharvari_Karnik_f4bpr8
H2O cluster total nodes:	1
H2O cluster free memory:	1.745 Gb
H2O cluster total cores:	0
H2O cluster allowed cores:	0
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:52355
H2O connection proxy:	None
H2O internal security:	False
H2O API Extensions:	Algos, AutoML, Core V3, Core V4
Python version:	3.7.2 final

```
aml_leaderboard_df
```

	model_id	auc	logloss
0	GBM_grid_0_AutoML_20190419_152003_model_0	0.947845	0.179557
1	GBM_grid_0_AutoML_20190419_152003_model_1	0.947844	0.179753
2	GBM_grid_1_AutoML_20190419_152003_model_4	0.947558	0.184119
3	GBM_grid_0_AutoML_20190419_152003_model_4	0.946255	0.180033
4	StackedEnsemble_0_AutoML_20190419_152003	0.945930	0.197898
5	GBM_grid_1_AutoML_20190419_152003_model_0	0.945708	0.191594
6	GBM_grid_0_AutoML_20190419_152003_model_2	0.945488	0.183640
7	GBM_grid_0_AutoML_20190419_152003_model_3	0.944647	0.188188
8	GBM_grid_1_AutoML_20190419_152003_model_1	0.943908	0.192912
9	XRT_0_AutoML_20190419_152003	0.942575	0.190489
10	GBM_grid_1_AutoML_20190419_152003_model_5	0.939407	0.235658
11	DRF_0_AutoML_20190419_152003	0.937810	0.190303
12	GBM_grid_1_AutoML_20190419_152003_model_2	0.936006	0.220501
13	GLM_grid_0_AutoML_20190419_152003_model_1	0.933566	0.213060
14	GLM_grid_0_AutoML_20190419_152003_model_0	0.933566	0.213060
15	GBM_grid_1_AutoML_20190419_152003_model_3	0.932064	0.208754
16	DeepLearning_0_AutoML_20190419_152003	0.931733	0.204163

4. RESULT

The model is run for 200, 700 and 1500 seconds. The best model generated for 200 seconds runtime is StackedEnsemble_AllModels_0_AutoML_20190426_01.. where the AUC = 0.948108 and logloss = 0.193033. The best model generated for 700 seconds is 0 GBM_grid_0_AutoML_20190426_015716_model_10 with AUC= 0.948317 and logloss=0.187848. The best model generated for 1500 seconds runtime is 0 GBM_grid_0_AutoML_20190426_021307_model_26 with AUC = 0.948503 and logloss = 0.175856. The dataset is made available globally i.e. the

notebook can be executed on any computer just by executing the code.

5. Conclusion

The Bank Marketing dataset is an example of classification. The model is run for 200, 700 and 1500 seconds. The best model generated for 200 seconds runtime is StackedEnsemble_AllModels_0_AutoML_20190426_01.. where the AUC=0.948108 and logloss =0.193033. The best model generated for 700 seconds is 0 GBM_grid_0_AutoML_20190426_015716_model_10 with AUC=0.948317 and logloss=0.187848. The best model generated for 1500 seconds runtime is 0 GBM_grid_0_AutoML_20190426_021307_model_26 with AUC=0.948503 and logloss=0.175856. The dataset is made available globally i.e. the notebook can be executed on any computer just by executing the code. We are storing the hyperparameters for Bank Marketing dataset which will help in the future to set the hyperparameters before training the data. THe output saved for each model stores the meta-data, run-id and the files inside the same directory consisting of JSON files. More information can be inferred from this data to further analyse the important hyperparameters. The CSV files including the JSON file is stored in the directory as well.

6. Acknowledgment

We want to thank Prof. Nick Brown for his constant support and guidance during this project. We would also like to thank Prabhu Subramanian, manager of this project who has helped us on every step and guided us wherever required.

7. References

- http://localhost:8890/notebooks/Desktop/DSProject/hyperparameter-db-project-ds18/BankDB_200.ipynb
- https://www.kaggle.com/henriqueyamahata/bank-marketing
- https://towardsdatascience.com/hyperparameters-in-deep-learning-927f7b2084dd
- https://www.quora.com/What-are-hyperparameters-in-machine-learning
- http://university.h2o.ai/data-science-101/lesson1.html
- https://h2o-release.s3.amazonaws.com/h2o/rel-slater/9/docs-website/h2o-py/docs/h2o.html
- https://stat.ethz.ch/R-manual/R-devel/library/base/html/scale.html
- https://www.jeremyjordan.me/hyperparameter-tuning/
- https://github.com/prabhuSub/Hyperparamter-Samples/blob/master/Hyperparameter_Generated/PUBGDataset_1000_Hyperparameter_Generated.ipynb
- https://www.h2o.ai/
- https://en.wikipedia.org/wiki/H2O_(software)
- https://github.com/h2oai
- http://docs.h2o.ai/