HyperparameterDatabase for predicting the water temperature in degree Celsius

FNU Karan
karan.k@husky.neu.edu
INFO 6105, Spring 2019, Northeastern University

Abstract: ***In Bayesian 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 best hyperparameter for the dataset which would help the database team in modelling the database schema in an efficient way. We would leverage H2O to generate models for the dataset for getting best hyperparameters. In the dataset we are predicting the water temperature in degree Celsius.***

Keywords—Hyperparameter, H2O, Prediction.

I. Introduction:

Hyperparameters are adjustable parameters you choose to train a model that governs the training process itself. For example, to train a deep neural network, you decide the number of hidden layers in the network and the number of nodes in each layer prior to training the model. These values usually stay constant during the training process.

After selecting the dataset, following analysis has been performed:

- Trying to find the important hyperparameters.
- Trying to find the range of hyperparameters
- Comparing the range of values across the models for different hyperparameters

To extract the hyperparameters H2O is used. H2O generates different models for the dataset by using various algorithms like Gradient Boosting Machine, Generalized Linear Models, Distribution Random Forests, Extremely Randomized Forest, Deep Learning. Hyperparameters for all the models are extracted and analysis are done using Grid Search. Manual analysis on the extracted hyperparameters is also performed.

II. Dataset:

The CalCOFI data set represents the longest (1949-present) and most complete (more than 50,000 sampling stations) time series of oceanographic and larval fish data in the world. It includes abundance data on the larvae of over 250 species of fish; larval length frequency data and egg abundance data on key commercial species; and oceanographic and plankton data. The physical, chemical, and biological data collected at regular time and space intervals quickly became valuable for documenting climatic cycles in the California Current and a range of biological responses to them. CalCOFI research drew world attention to the biological response to the dramatic Pacific-warming event in 1957-58 and introduced the term “El Niño” into the scientific literature. In the dataset we are determining the water temperature in degree celsius

Source:https://www.kaggle.com/sohier/calcofi

The following are the columns in the dataset:

```
In[]:list(df2)
Out[]:['cst_cnt',
 'btl_cnt',
 'sta_id',
 'depth_id',
 'depthm',
 't_degc',
 'salnty',
 'o2ml_l',
 'stheta',
```

```
'o2sat',
'oxy_µmol/kg',
'btlnum',
'recind',
't_prec',
't_qual',
's_prec',
's_qual',
'p_qual',
'o_qual',
'sthtaq',
'o2satq',
'chlora',
'chlqua',
'phaeop',
'phaqua',
'po4um',
'po4q',
'sio3um',
'sio3qu',
'no2um',
'no2q',
'no3um',
'no3q',
'nh3um',
'nh3q',
'c14as1',
'c14a1p',
'c14a1q',
'c14as2',
'c14a2p',
'c14a2q',
'darkas',
'darkap',
'darkaq',
'meanas',
'meanap',
'meanaq',
'inctim',
'lightp',
'r_depth',
'r_temp',
'r_potemp',
'r_salinity',
'r_sigma',
'r_sva',
'r_dynht',
'r_o2',
'r_o2sat',
'r_sio3',
'r_po4',
'r_no3',
'r_no2',
'r_nh4',
'r_chla',
'r_phaeo',
'r_pres',
'r_samp',
'dic1',
'dic2',
'ta1',
'ta2',
'ph2',
'ph1',
'dic_quality_comment']
```

III. Data Cleaning

In the dataset there are many null values in each column. To remove these null values there are two approaches:

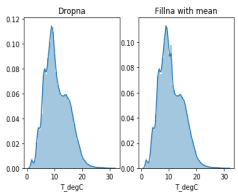
- By using .dropna() i.e. by dropping the missing values from the dataset.
- By using .fillna() i.e. by filling the missing values with mean

To decide which approach is best we have to check the difference between the graphs of each distribution. If they have a normal distribution(close to one), it is better to fill the values with the mean. If not, there will be unwanted spike in the graph.

Let us plot the graphs for water temperature

```
In [11]:
# Let's compare water temperature using fillna() vs mean()
fig, ax=plt.subplots(1,2)
sns.distplot(calcofi.T_degC.dropna(), ax=ax[0])
ax[0].set_title("Dropna")
sns.distplot(calcofi.T_degC.fillna(calcofi.T_degC.mean()), ax=ax[1])
ax[1].set_title("Fillna with mean")

Out[11]:
Text(0.5,1,'Fillna with mean')
```

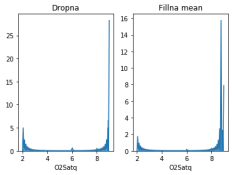


We have observed that there is no unwanted spike in the distributions.

Let's plot the graph for oxygen saturation

```
In [12]:
# Let's compare oxygen saturation using fillna() vs mean()
fig, ax=plt.subplots(1,2)
sns.distplot(calcofi.O2Satq.dropna(), ax=ax[0])
ax[0].set_title("Dropna")
sns.distplot(calcofi.O2Satq.fillna(calcofi.O2Satq.mean()), ax=ax[1])
ax[1].set_title("Fillna mean")

Out[12]:
Text(0.5,1,'Fillna mean')
```



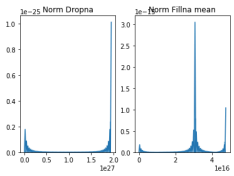
Although there is something weird but we can ignore that.

Let's normalize with box-cox so that we can observe more clearly

```
In [13]:
# normalize oxygen sat (w/ Box-Cox)
normalized_ox_dropna = stats.boxcox(calcofi.O2Satq.dropna())[0]
# normalize oxygen sat (w/ Box-Cox)
normalized_ox_fillna = stats.boxcox(calcofi.O2Satq.fillna(calcofi.O2Satq.mean()))[0]

fig, ax=plt.subplots(1,2)
sns.distplot(normalized_ox_dropna, ax=ax[0])
ax[0].set_title("Norm Dropna")
sns.distplot(normalized_ox_fillna, ax=ax[1])
ax[1].set_title("Norm Fillna mean")

Out[13]:
Text(0.5,1,'Norm Fillna mean')
```



We can see a clear difference now. It should be using .fillna() instead of using .dropna() could lead to errors if we try to correlate variables.

After using .fillna() i.e. filling the null values with the mean, there are still some columns that have constant values since there are no other values other than the null values in it before filling with the mean value. So these columns should be dropped.

IV. Method:

In Bayesian statistics, a hyperparameter is a parameter of a prior distribution; the term is used to distinguish them from parameters of the model for the underlying system under analysis. These are adjustable parameters you choose to train a model that governs the training process itself. For example, to train a deep neural network, you decide the number of hidden layers in the network and the number of nodes in each layer prior to training the model. These values usually stay constant during the training process. Better the hyperparameters better the result. Conceptually, hyperparameters can be considered orthogonal to the learning model itself in the sense that, although they live outside the model, there is a direct relationship between them. Determining these hyperparameters is tedious and cumbersome task. The main aim of this research is to determine the important hyperparameters with proper tuning values obtained from the dataset. To achieve this, this research leveraged H2O.ai's module for python called H2O which gives flexibility to run the process for numerous run times. Models are generated by running H2OAutoML for various runtimes (500s, 700s, 900s, 1100s and 1300s).

V. Code and Documentation:

Github Link: The complete code with documentation can be found on the below link:
https://github.com/INFO6105-Spring19/hyperparameter-db-project-ds11

VI. Results:

The H2O AutoML is ran for diffrenet runtimes i.e. 500s, 700s, 900s, 1100s, 1300s. After running H2O, leaderboard is generated for each runtime and hyperparameters are generated for each model in Jason file. The metadata information for each run is also obtained for each run in Jason file. After getting hyperparameters, range of each hyperparameter is determined. Also, Compared the range of values across the models for different hyperparameters. In the project, Grid search is used to find the important hyperparameters for following algorithms.

- Gradient Boosting Machine
- Generalized Linear Models
- Distribution Random Forests

VII. Discussion:

In the current project we have found hyperparameters for the following Algorithms:

- Gradient Boosting Machine
- Generalized Linear Models
- Distribution Random Forests
- Extremely Randomized Forest
- Deep Learning

VIII. Acknowledgement:

We are thankful to Prof. Nick Bear Brown, Assistant Professor and Prabhu Subramanian, Teaching Assistant, Northeastern University, Boston, MA for their valuable guidance, encouragement, and cooperation during the implementation of the project.

IX. References:

- https://github.com/skunkworksneu/IIT-AGNE-Workshop
- https://github.com/prabhuSub/Hyperparamter-Samples
- http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html
- https://www.kaggle.com/sohier/calcofi
- https://www.kaggle.com/jonasnconde/data-cleaning-challenge-scale-and-normalize-data