

Hyperparameter Database Project

Team DB-08 (Bhavana Shanmugam, Yadukrishnan Sethumadhavan)

1. Abstract

The project is a part of skunkworks hyperparameters project. From the database prospective, the project has two main parts where part one is dedicated to creating a database that stores the metadata and data set details of Melbourne Housing Data which is our chosen dataset for the project. Part two is to store the evaluations and hyperparameters that are being generated by the machine learning algorithms on Melbourne housing dataset.

Introduction

Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. Hyperparameters are parameters that are specified prior to running machine learning algorithms that have a large effect on the predictive power of statistical models. Knowledge of the relative importance of a hyperparameter to an algorithm and its range of values is crucial to hyperparameter tuning and creating effective models.

The hyperparameter database is a public 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.

The hyperparameter database is created by running millions of hyperparameter values, over thousands of public datasets and calculating the individual conditional expectation of every hyperparameter on the quality of a 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).

The hyperparameter database also uses these data to build models that can predict hyperparameters without search and for visualizing and teaching statistical concepts such as power and bias/variance tradeoff.

As part of the DMDD team we create a database starting from creating a conceptual model and storing all the data into a physical database. Following are the steps that were followed:

- Cleaned and scraped JSON files containing Hyperparameter information and stored the data in data frames and saved them in CSV Format (later used to populate the database).
- Designed the database and created a conceptual model and ER diagram of the database.
- Performed database normalization – 1NF, 2NF, 3NF.
- Created tables in the database.

- Inserted data into the respective tables using the CSV generated from parsing the JSON files obtained from the data science team members.
- Created foreign key relationships
- Queried database using SQL, created use cases, views (virtual tables), functions and indexes.
- Performed analytics on the database using queries, for e.g. getting the best value for the hyperparameter from the database.

2. Explain Data Source

Data related to Melbourne Housing is obtained from Kaggle:
<https://www.kaggle.com/anthonyypino/melbourne-housing-market>.

The dataset contains 21 columns. The problem is a regression problem, target field being Price.

Dataset Information:

Suburb: Suburb

Address: Address

Rooms: Number of rooms

Price: Price in Australian dollars

Method: S - property sold; SP - property sold prior; PI - property passed in; PN - sold prior not disclosed; SN - sold not disclosed; NB - no bid; VB - vendor bid; W - withdrawn prior to auction; SA - sold after auction; SS - sold after auction price not disclosed. N/A - price or highest bid not available.

Type: br - bedroom(s); h - house, cottage, villa, semi, terrace; u - unit, duplex; t - townhouse; dev site - development site; o res - other residential.

SellerG: Real Estate Agent

Date: Date sold

Distance: Distance from CBD in Kilometers

Regionname: General Region (West, North West, North, North east ...etc)

Propertycount: Number of properties that exist in the suburb.

Bedroom2: Scraped # of Bedrooms (from different source)

Bathroom: Number of Bathrooms

Car: Number of car spots

Landsize: Land Size in Meters

BuildingArea: Building Size in Meters

YearBuilt: Year the house was built

CouncilArea: Governing council for the area

Latitude: Self explanatory

Longitude: Self explanatory

This data is validated and preprocessed to assure the credibility and integrity. Data manipulations are done suitably for machine learning models by the data science part of our group. H2O is used for hyperparameter tuning. The output file containing the hyperparameters information is provided by the data science group as JSON files which we parse to obtain the required information to populate the database tables.

Tools & Application Used

H2O: An open source, in-memory, distributed, fast, and scalable machine learning and predictive analytics platform to build machine learning models and provide easy productions of those models.

MySQL Server & MySQL Workbench: A relational database management system software. It is used for storing the Hyperparameters data.

Data Science Part

H2O software, which is an open source, in-memory, distributed, fast, and scalable machine learning and predictive analytics platform is used to build machine learning models and provide easy productions of those models. To search for Hyperparameter, the AUTOML is run on the whole dataset, using different run-ids. The result gives us the best model from the Leaderboard which has performed better in-terms of RMSE, MSE, MAE, RMSLE metrics. This process is repeated for five different run-times. After getting the required values and range of the Hyperparameters, the hyperparameter metadata is saved in JSON files.

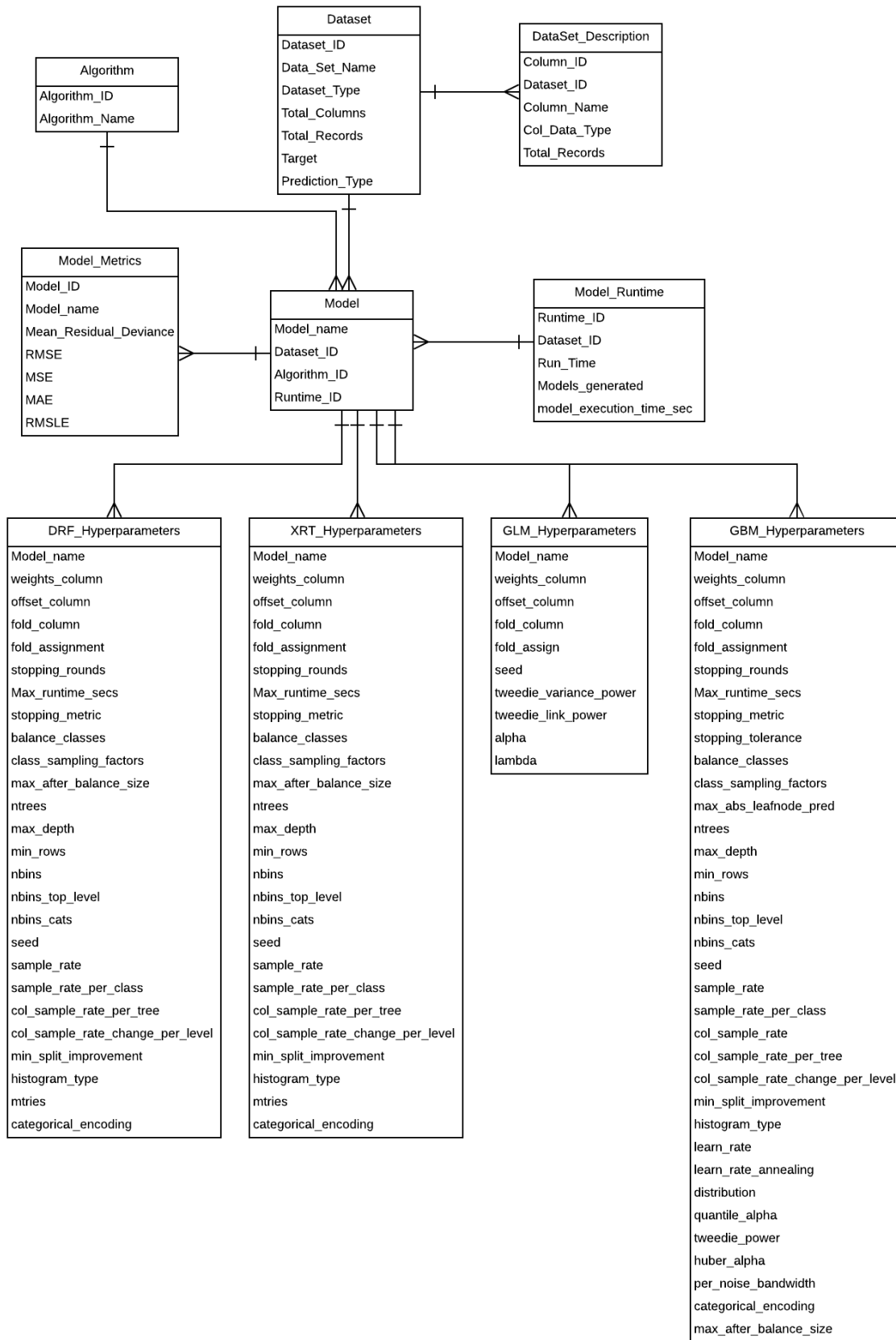
Database Part:

After receiving the hyperparameters metadata files from all the runs in JSON format, the JSON files are parsed to get the information to be modeled in the database. The parsed data is stored in CSV files which are later used to populate the database tables.

Conceptual diagram and E-R diagram is created which provides information about the entities, relationships and attributes for the table. The tables are created and populated using the CSV files generated. After this, as per the project requirement, use cases, views, functions and indexes are created.

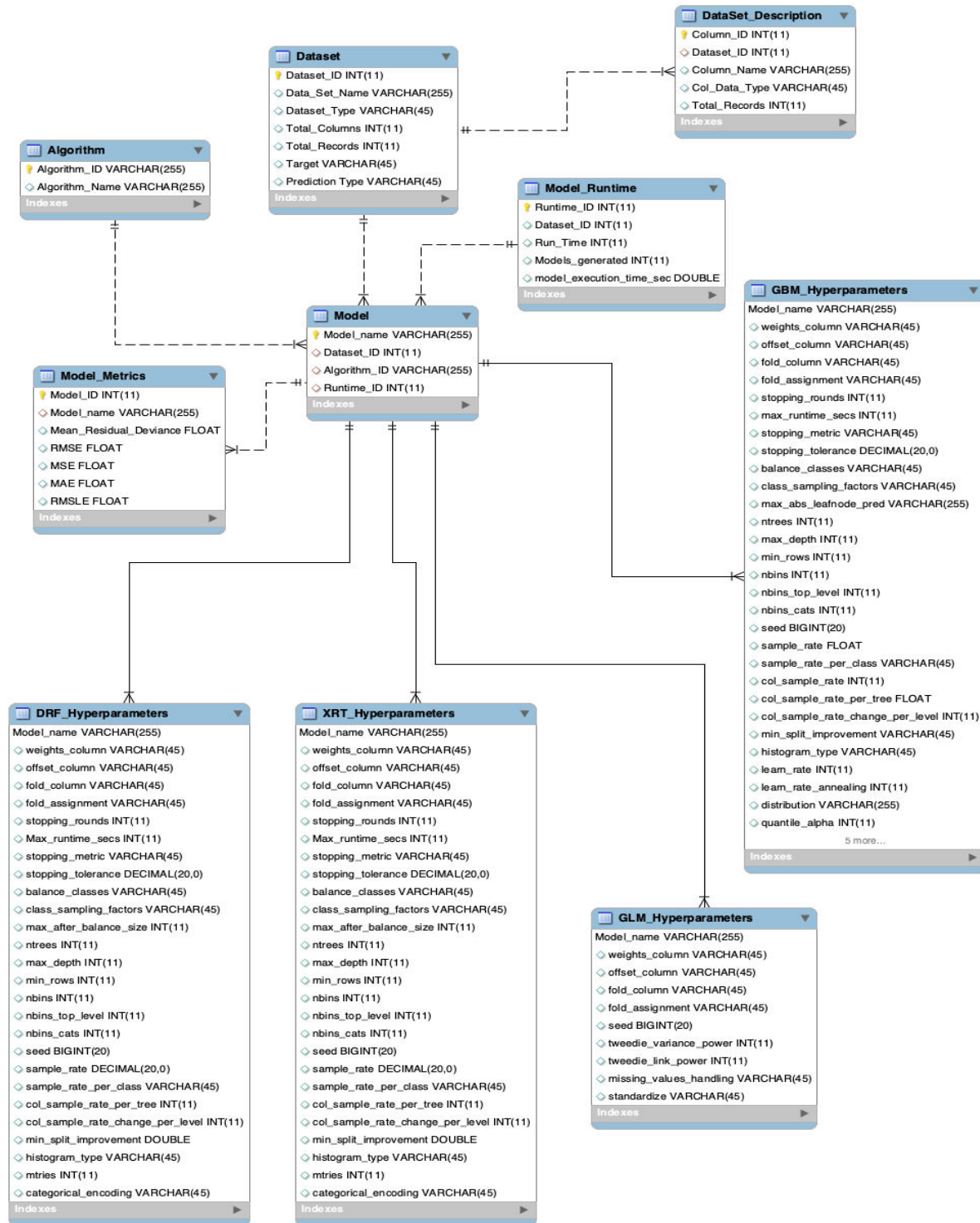
3. Conceptual Schema

A conceptual data model captures the key entities (a person, place, concept, event or thing about which the organization wants to collect data), and the relationships between these entities. Attributes are often not added to conceptual data models as these are higher level models.



4. ER Diagram

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems



JSON to CSV

- Scraped the JSON files to create CSV files used to populate the database tables.

Imported JSON files of all the models -

```
In [605]: import json
import pandas as pd
import numpy as np

In [606]: L500= open("hlE1w6gySMelbourneHousingPrices_500_hy_parameter.json")
L800= open("ilkMo6dfRMelbourneHousingPrices_800_hy_parameter.json")
L1000=open("q077P1c6BMelbourneHousingPrices_1000_hy_parameter.json")
L1300=open("aGXJjUoV5MelbourneHousingPrices_1300_hy_parameter.json")
L1500=open("ljms5Ed0lMelbourneHousingPrices_1500_hy_parameter.json")
d=json.load(L500)+json.load(L800)+json.load(L1000)+json.load(L1300)+json.load(L1500)
```

Created lists to hold the hyperparameter information generated by H2O AutoML -

```
In [607]: #GBM
model_name=[]
weights_column=[]
offset_column=[]
fold_column=[]
fold_assignment=[]
stopping_rounds=[]
max_runtime_secs=[]
stopping_metric=[]
stopping_tolerance=[]
balance_classes=[]
class_sampling_factors=[]
max_after_balance_size=[]
ntrees=[]
max_depth=[]
min_rows=[]
nbins=[]
nbins_top_level=[]
nbins_cats=[]
seed=[]
sample_rate=[]
sample_rate_per_class=[]
col_sample_rate_per_tree=[]
col_sample_rate_change_per_level=[]
min_split_improvement=[]
histogram_type=[]
learn_rate=[]
learn_rate_annealing=[]
col_sample_rate=[]
max_abs_leafnode_pred=[]
pred_noise_bandwidth=[]
distribution=[]
tweedie_power=[]
quantile_alpha=[]
huber_alpha=[]
categorical_encoding=[]
```

Scraping all the hyperparameter information for algorithms to lists -

```

for i in range(len(d)):
    if d[i]['model_id']['actual']['name'].startswith('GBM'):
        model_name1=d[i]['model_id']['actual']['name']
        weights_column1= d[i]['weights_column']['actual']
        offset_column1= d[i]['offset_column']['actual']
        fold_column1= d[i]['fold_column']['actual']
        fold_assignment1= d[i]['fold_assignment']['actual']
        stopping_rounds1= d[i]['stopping_rounds']['actual']
        max_runtime_secs1= d[i]['max_runtime_secs']['actual']
        stopping_metric1= d[i]['stopping_metric']['actual']
        stopping_tolerance1= d[i]['stopping_tolerance']['actual']
        balance_classes1= d[i]['balance_classes']['actual']
        class_sampling_factors1= d[i]['class_sampling_factors']['actual']
        max_after_balance_size1= d[i]['max_after_balance_size']['actual']
        ntrees1= d[i]['ntrees']['actual']
        max_depth1= d[i]['max_depth']['actual']
        min_rows1= d[i]['min_rows']['actual']
        nbins1= d[i]['nbins']['actual']
        nbins_top_level1= d[i]['nbins_top_level']['actual']
        nbins_cats1= d[i]['nbins_cats']['actual']
        seed1= d[i]['seed']['actual']
        sample_rate1= d[i]['sample_rate']['actual']
        sample_rate_per_class1= d[i]['sample_rate_per_class']['actual']
        col_sample_rate_per_tree1= d[i]['col_sample_rate_per_tree']['actual']
        col_sample_rate_change_per_level1= d[i]['col_sample_rate_change_per_level']['actual']
        min_split_improvement1= d[i]['min_split_improvement']['actual']
        histogram_type1= d[i]['histogram_type']['actual']
        learn_rate1= d[i]['learn_rate']['actual']
        learn_rate_annealing1= d[i]['learn_rate_annealing']['actual']
        col_sample_rate1= d[i]['col_sample_rate']['actual']
        max_abs_leafnode_pred1= d[i]['max_abs_leafnode_pred']['actual']
        pred_noise_bandwidth1= d[i]['pred_noise_bandwidth']['actual']
        distribution1= d[i]['distribution']['actual']
        tweedie_power1= d[i]['tweedie_power']['actual']
        quantile_alpha1= d[i]['quantile_alpha']['actual']
        huber_alpha1= d[i]['huber_alpha']['actual']
        categorical_encoding1= d[i]['categorical_encoding']['actual']

        model_name.append(model_name1)
        weights_column.append(weights_column1)
        offset_column.append(offset_column1)
        fold_column.append(fold_column1)
        fold_assignment.append(fold_assignment1)
        stopping_rounds.append(stopping_rounds1)
        max_runtime_secs.append(max_runtime_secs1)
        stopping_metric.append(stopping_metric1)
        stopping_tolerance.append(stopping_tolerance1)
        balance_classes.append(balance_classes1)
        class_sampling_factors.append(class_sampling_factors1)
        max_after_balance_size.append(max_after_balance_size1)
        ntrees.append(ntrees1)
        max_depth.append(max_depth1)
        min_rows.append(min_rows1)
        nbins.append(nbins1)
        nbins_top_level.append(nbins_top_level1)
        nbins_cats.append(nbins_cats1)
        seed.append(seed1)
        sample_rate.append(sample_rate1)
        sample_rate_per_class.append(sample_rate_per_class1)
        col_sample_rate_per_tree.append(col_sample_rate_per_tree1)
        col_sample_rate_change_per_level.append(col_sample_rate_change_per_level1)
        min_split_improvement.append(min_split_improvement1)
        histogram_type.append(histogram_type1)
        learn_rate.append(learn_rate1)
        learn_rate_annealing.append(learn_rate_annealing1)
        col_sample_rate.append(col_sample_rate1)
        max_abs_leafnode_pred.append(max_abs_leafnode_pred1)
        pred_noise_bandwidth.append(pred_noise_bandwidth1)
        distribution.append(distribution1)
        tweedie_power.append(tweedie_power1)
        quantile_alpha.append(quantile_alpha1)
        huber_alpha.append(huber_alpha1)
        categorical_encoding.append(categorical_encoding1)

```

Storing it into data frame which is later exported to CSV –


```

distribution.append(distribution1)
tweedie_power.append(tweedie_power1)
quantile_alpha.append(quantile_alpha1)
huber_alpha.append(huber_alpha)
categorical_encoding.append(categorical_encoding1)

gbm_df = pd.DataFrame(data={"Model Name":model_name,"weights_column":weights_column,
'offset_column':offset_column, 'fold_column':fold_column,
'fold_assessment':fold_assignment, 'stopping_rounds':stopping_rounds,
'max_runtime_secs':max_runtime_secs, 'stopping_metric':stopping_metric,
'stopping_tolerance':stopping_tolerance, 'balance_classes':balance_classes,
'class_sampling_factors':class_sampling_factors, 'max_after_balance_size':max_after_balance_size,
'ntrees':ntrees, 'max_depth':max_depth, 'min_rows':min_rows, 'nbins':nbins,
'nbins_top_level':nbins_top_level, 'nbins_cats':nbins_cats, "seed":seed,
'sample_rate':sample_rate, 'sample_rate_per_class':sample_rate_per_class,
'col_sample_rate_per_tree':col_sample_rate_per_tree, 'col_sample_rate_change_per_level':col_sample_rate_change_per_level,
'min_split_improvement':min_split_improvement, 'histogram_type':histogram_type,
'learn_rate':learn_rate,'learn_rate_annealing':learn_rate_annealing,
'col_sample_rate':col_sample_rate, 'max_abs_leafnode_pred':max_abs_leafnode_pred,
'pred_noise_bandwidth':pred_noise_bandwidth, 'distribution':distribution,
'tweedie_power':tweedie_power,'quantile_alpha':quantile_alpha,
'huber_alpha':huber_alpha,'categorical_encoding':categorical_encoding})
gbm_df.fillna("NULL", inplace = True)
gbm_df.to_csv("GBM.csv",sep=',', index=False)

```

In [608]: gbm_df.head(5)

Out[608]:

	Model Name	weights_column	offset_column	fold_column	fold_assessment	stopping_rounds	max_runtime_secs	stopping_metric	stopping_tolerance
0	GBM_4_AutoML_20190416_233703	NULL	NULL	NULL	Modulo	0	0.0	deviance	0.001
1	GBM_2_AutoML_20190416_233703	NULL	NULL	NULL	Modulo	0	0.0	deviance	0.001
2	GBM_3_AutoML_20190416_233703	NULL	NULL	NULL	Modulo	0	0.0	deviance	0.001
3	GBM_1_AutoML_20190416_233703	NULL	NULL	NULL	Modulo	0	0.0	deviance	0.001
4	GBM_5_AutoML_20190416_233703	NULL	NULL	NULL	Modulo	0	0.0	deviance	0.001

5 rows x 35 columns

Repeated the same process and scraped the data from JSON for all algorithm types.

5. Normalization

Check that tables are in First normal form (1NF)

- Each table has a primary key: minimal set of attributes which can uniquely identify a record
 - Generated Primary keys for the tables which didn't had one.
- The values in each column of a table are atomic (No multi-value attributes allowed)
 - Cleaned the data so there we no atomic values in every table
- There are no repeating groups: two columns do not store similar information in the same table
 - Drop the tables which from the dataset which showed redundancy

Check that tables are in Second normal form (2NF)

- All requirements for 1st NF must be met.
 - Satisfied
- No partial dependencies.
 - While tidying the datasets cleaned the partial dependencies
- No calculated data
 - The variables depended upon other variables were removed

Check that tables are in Third normal form (3NF) and the final SQL

- All requirements for 2nd NF must be met.
 - Satisfied

- Eliminate fields that do not directly depend on the primary key; that is no transitive dependencies.
 - Satisfied.

Table 1: Dataset

This table has one primary key(Dataset_ID), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 2: Dataset_Description

This table has one primary key(Column_ID), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 3: Model_Runtime

This table has one primary key(Runtime_ID), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 4: Model

This table has one primary key(Model_name), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 5: Model_Metrics

This table has one primary key(Model_ID), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 6: Algorithm

This table has one primary key(Algorithm_ID), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 7: GBM_Hyperparameters

This table has one primary key(Model_name), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 8: GLM_Hyperparameters

This table has one primary key(Model_name), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 9: DRF_Hyperparameters

This table has one primary key(Model_name), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

Table 10: XRT_Hyperparameters

This table has one primary key(Model_name), and several non-key attributes. All non-key attributes are based on and only on primary key. It is in 3NF.

6. Physical Model

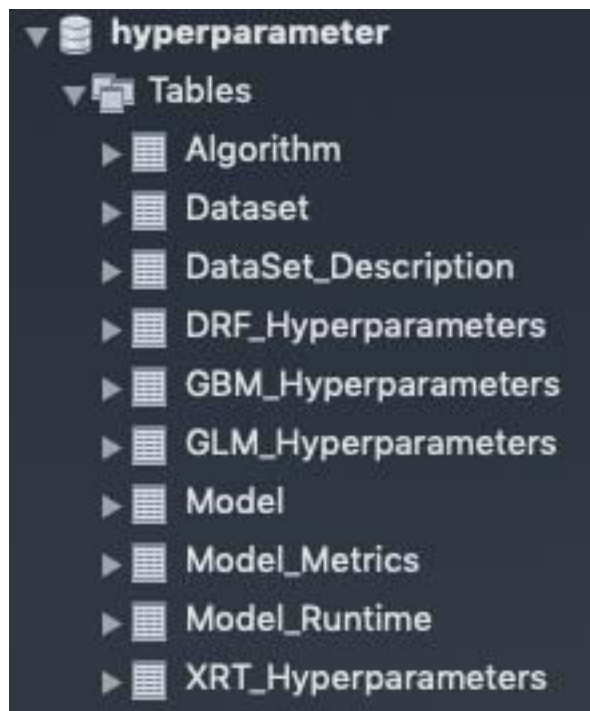
Physical model is created using MySQL database

The SQL files contains all the SQL physical model information.

CREATE TABLES IN SQL:

```
CREATE TABLE `GLM_Hyperparameters` (  
  `Model_name` varchar(255) NOT NULL,  
  `weights_column` varchar(45) DEFAULT NULL,  
  `offset_column` varchar(45) DEFAULT NULL,  
  `fold_column` varchar(45) DEFAULT NULL,  
  `fold_assignment` varchar(45) DEFAULT NULL,  
  `seed` bigint(20) DEFAULT NULL,  
  `tweedie_variance_power` int(11) DEFAULT NULL,  
  `tweedie_link_power` int(11) DEFAULT NULL,  
  `missing_values_handling` varchar(45) DEFAULT NULL,  
  `standardize` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`Model_name`),  
  CONSTRAINT `Model3` FOREIGN KEY (`Model_name`) REFERENCES `Model`  
  (`Model_name`) ON DELETE NO ACTION ON UPDATE NO ACTION  
)
```

TABLES CREATED:



7. Use Cases

Use cases contains the use cases SQL queries.

1) Find the best model amongst all the models of GBM algorithm

```
SELECT Model_name FROM model_metrics
WHERE RMSE=(SELECT MIN(RMSE) FROM model_metrics a JOIN
GBM_Hyperparameters b ON a.Model_name=b.Model_name)
```

Model_name	rmse
GBM_grid_1_AutoML_20190419_190006_model_7	300602

2) Count the number of models formed for runtime 500s

```
SELECT count(Model_name) FROM Model a, Model_Runtime b
WHERE a.runtime_ID=b.runtime_ID AND b.Run_Time =500
```

count(Model_name)
13

3) For what runtime is the best model generated?

```
SELECT run_time, rmse FROM model_runtime a, model_metrics b, model c
```

WHERE a.runtime_ID=c.runtime_ID AND b.model_name = c.model_name AND
b.rmse=(SELECT MIN(RMSE) FROM model_metrics)

run_time	rmse
▶ 1500	300602

4) How many DRF models are generated for a particular runtime?

SELECT b.Model_name FROM model_runtime a, model_metrics b, model c
WHERE a.runtime_ID=c.runtime_ID AND b.model_name = c.model_name AND
a.run_time=500 AND b.model_name LIKE "DRF%"

Model_name
▶ DRF_1_AutoML_20190416_233703

5) Find average of RMSE and count of model runs for all model runs of GBM Models for run time 500s

SELECT avg(rmse), count(a.model_name) FROM model_metrics a, model_runtime b, model c
WHERE b.runtime_ID=c.runtime_ID AND a.model_name = c.model_name AND
b.run_time=500 AND a.model_name LIKE "GBM%"

avg(rmse)	count(a.model_name)
▶ 345126.5	6

6) What is the Maximum RMSE value for GLM model?

SELECT MAX(RMSE) FROM model_metrics
WHERE RMSE=(SELECT MAX(RMSE) FROM model_metrics a JOIN
GLM_Hyperparameters b ON a.Model_name=b.Model_name) LIMIT 1

MAX(RMSE)
▶ 651416

7) List the number of GBM models which has distribution="gaussian"?

SELECT COUNT(model_Name) FROM GBM_Hyperparameters
WHERE distribution LIKE 'gaussian%' AND model_name LIKE "GBM%"

count(model_Name)
▶ 59

8) what should be the seed value to get best rmse for GLM?

```
SELECT seed FROM GLM_Hyperparameters a, model_metrics b
WHERE a.model_name=b.model_name AND b.rmse =(SELECT MIN(RMSE) FROM
model_metrics a JOIN GLM_Hyperparameters b ON a.Model_name=b.Model_name)
```

seed
3605434411294760589
-4293711706793077853
1695358159836644228
1411290820467823314
2106761952620105859
-312107903609785575

9) Which model has maximum accuracy?

```
SELECT model_name FROM model_metrics ORDER BY rmse LIMIT 1
```

model_name
GBM_grid_1_AutoML_20190419_190006_model_7

10) Display how many models were created for each runtime?

```
SELECT a.run_time as 'Runtime', COUNT( b.model_name) as 'No of models' FROM
model_runtime a, Model_metrics b, model c
WHERE a.runtime_ID=c.runtime_ID AND b.model_name = c.model_name GROUP BY
a.run_time
```

Runtime	No of models
500	12
800	26
1000	13
1300	21
1500	20

8. Views

Views contain the Views SQL queries.

1) Display performance measures of all the models present in the database for a particular Algorithm-Type (GBM/GLM/Linear etc.)

Create view Models_for_algos AS

select M.Model_name, A.Algorithm_Name, MM.Mean_Residual_Deviance, MM.RMSE, MM.MSE, MM.MAE, MM.RMSLE

from Algorithm A

join Model M on A.Algorithm_ID = M.Algorithm_ID

join Model_Metrics MM on MM.Model_Name = M.Model_Name

where A.Algorithm_Name ="GBM";

Model_name	Algorithm_Name	Mean_Residual_Deviance	RMSE	MSE	MAE	RMSLE
GBM_1_AutoML_20190416_233703	GBM	93506400000	305788	93506400000	173001	0.206323
GBM_1_AutoML_20190417_092107	GBM	91480100000	302457	91480100000	171898	0.206078
GBM_1_AutoML_20190417_103626	GBM	92516000000	304164	92516000000	172391	0.20572
GBM_1_AutoML_20190417_121910	GBM	92281400000	303779	92281400000	172470	0.205012
GBM_1_AutoML_20190419_190006	GBM	93246100000	305362	93246100000	172482	0.206082
GBM_1_AutoML_20190419_193106	GBM	92075900000	303440	92075900000	171953	0.205569
GBM_2_AutoML_20190416_233703	GBM	91903500000	303156	91903500000	171953	0.205054
GBM_2_AutoML_20190417_092107	GBM	93072100000	305077	93072100000	172842	0.206126
GBM_2_AutoML_20190417_103626	GBM	93500300000	305778	93500300000	173022	0.206536
GBM_2_AutoML_20190417_121910	GBM	91810000000	303002	91810000000	171920	0.204805
GBM_2_AutoML_20190419_190006	GBM	91889100000	303132	91889100000	171957	0.204748
GBM_2_AutoML_20190419_193106	GBM	92939500000	304860	92939500000	172467	0.205375
GBM_3_AutoML_20190416_233703	GBM	92203400000	303650	92203400000	171106	0.20058
GBM_3_AutoML_20190417_092107	GBM	93761400000	306205	93761400000	172664	0.205604
GBM_3_AutoML_20190417_103626	GBM	93744500000	306177	93744500000	172498	0.205731
GBM_3_AutoML_20190417_121910	GBM	91838000000	303048	91838000000	171935	0.205196
GBM_3_AutoML_20190419_190006	GBM	92794000000	304621	92794000000	172327	0.204726
GBM_3_AutoML_20190419_193106	GBM	91750000000	302903	91750000000	171672	0.204166
GBM_4_AutoML_20190416_233703	GBM	91662600000	302758	91662600000	171471	0.204274
GBM_4_AutoML_20190417_092107	GBM	93537100000	305838	93537100000	173089	0.206132
GBM_4_AutoML_20190417_103626	GBM	94047400000	306672	94047400000	172980	0.205805
GBM_4_AutoML_20190417_121910	GBM	92942000000	304864	92942000000	172690	0.205287
GBM_4_AutoML_20190419_190006	GBM	90932600000	301550	90932600000	170770	0.203594
GBM_4_AutoML_20190419_193106	GBM	93275200000	305410	93275200000	172305	0.205156
GBM_5_AutoML_20190416_233703	GBM	96806000000	311135	96806000000	176071	0.208741
GBM_5_AutoML_20190417_092107	GBM	96319600000	310354	96319600000	174895	0.207882
GBM_5_AutoML_20190417_103626	GBM	95291000000	308692	95291000000	174519	0.2077
GBM_5_AutoML_20190417_121910	GBM	96779900000	311095	96779900000	175250	0.208079
GBM_5_AutoML_20190419_190006	GBM	96280800000	310291	96280800000	175470	0.208547
GBM_5_AutoML_20190419_193106	GBM	96285500000	310299	96285500000	174727	0.207146
GBM_grid_1_AutoML_20190416_233703_model_1	GBM	296232000000	544272	296232000000	368958	0.430541
GBM_grid_1_AutoML_20190417_092107_model_1	GBM	91581000000	302623	91581000000	171073	0.203611
GBM_grid_1_AutoML_20190417_092107_model_2	GBM	118243000000	343866	118243000000	206802	0.250432
GBM_grid_1_AutoML_20190417_103626_model_1	GBM	94531100000	307459	94531100000	173774	0.206751
GBM_grid_1_AutoML_20190417_103626_model_3	GBM	110434000000	332316	110434000000	193822	0.232977
GBM_grid_1_AutoML_20190417_103626_model_4	GBM	164224000000	405246	164224000000	258496	0.311862
GBM_grid_1_AutoML_20190417_121910_model_3	GBM	137412000000	370691	137412000000	227275	0.275108
GBM_grid_1_AutoML_20190419_190006_model_1	GBM	98649300000	314085	98649300000	179877	0.214534
GBM_grid_1_AutoML_20190419_190006_model_10	GBM	131667000000	362860	131667000000	211065	0.256765
GBM_grid_1_AutoML_20190419_190006_model_2	GBM	99034700000	314698	99034700000	179176	0.213443
GBM_grid_1_AutoML_20190419_190006_model_3	GBM	150704000000	388206	150704000000	234823	0.286161
GBM_grid_1_AutoML_20190419_190006_model_5	GBM	97463600000	312192	97463600000	178298	0.211984
GBM_grid_1_AutoML_20190419_190006_model_6	GBM	91832800000	303039	91832800000	171230	0.205578
GBM_grid_1_AutoML_20190419_190006_model_7	GBM	90361700000	300602	90361700000	170227	0.20324
GBM_grid_1_AutoML_20190419_190006_model_8	GBM	92262100000	303747	92262100000	171426	0.205685
GBM_grid_1_AutoML_20190419_190006_model_9	GBM	93554000000	305866	93554000000	173230	0.206363
GBM_grid_1_AutoML_20190419_193106_model_1	GBM	96808300000	311140	96808300000	177760	0.212076
GBM_grid_1_AutoML_20190419_193106_model_2	GBM	93411600000	305633	93411600000	172939	0.20784
GBM_grid_1_AutoML_20190419_193106_model_3	GBM	95124200000	308422	95124200000	175192	0.214356
GBM_grid_1_AutoML_20190419_193106_model_4	GBM	134918000000	367312	134918000000	221059	0.269377
GBM_grid_1_AutoML_20190419_193106_model_5	GBM	109705000000	331218	109705000000	190927	0.227341
GBM_grid_1_AutoML_20190419_193106_model_6	GBM	93225600000	305329	93225600000	172157	0.205503
GBM_grid_1_AutoML_20190419_193106_model_7	GBM	94126600000	306801	94126600000	172213	0.20486
GBM_grid_1_AutoML_20190419_193106_model_8	GBM	141214000000	375785	141214000000	230758	0.281441
GBM_grid_1_AutoML_20190419_193106_model_9	GBM	331043000000	575363	331043000000	394968	0.457192

2) Display all the performance measures for all the models of a particular run

Create view Models_for_runtime AS

```
select M.Model_name, R.Run_Time, MM.Mean_Residual_Deviance, MM.RMSE, MM.MSE,
MM.MAE, MM.RMSLE
from Model_Runtime R
join Model M on R.Runtime_ID = M.Runtime_ID
join Model_Metrics MM on MM.Model_Name = M.Model_Name
where R.Run_Time=1000;
```

Model_name	Run_Time	Mean_Residual_Deviance	RMSE	MSE	MAE	RMSLE
DeepLearning_grid_1_AutoML_20190417_121910_model_2	1000	97626900000	312453	97626900000	181825	0.22061
DeepLearning_grid_1_AutoML_20190417_121910_model_3	1000	156482000000	395578	156482000000	249444	0.302083
DRF_1_AutoML_20190417_121910	1000	97522100000	312285	97522100000	174325	0.20681
GBM_1_AutoML_20190417_121910	1000	92281400000	303779	92281400000	172470	0.205012
GBM_2_AutoML_20190417_121910	1000	91810000000	303002	91810000000	171920	0.204805
GBM_3_AutoML_20190417_121910	1000	91838000000	303048	91838000000	171935	0.205196
GBM_4_AutoML_20190417_121910	1000	92942000000	304864	92942000000	172690	0.205287
GBM_5_AutoML_20190417_121910	1000	96779900000	311095	96779900000	175250	0.208079
GBM_grid_1_AutoML_20190417_121910_model_3	1000	137412000000	370691	137412000000	227275	0.275108
GLM_grid_1_AutoML_20190417_121910_model_1	1000	424343000000	651416	424343000000	456794	0.523713
StackedEnsemble_AllModels_AutoML_20190417_121910	1000	412365000000	642157	412365000000	449593	0.515595
StackedEnsemble_BestOfFamily_AutoML_20190417_121910	1000	420695000000	648610	420695000000	454611	0.521235
XRT_1_AutoML_20190417_121910	1000	108518000000	329421	108518000000	187737	0.22404

3) Create view for 50 best models of all algorithms

Create view 50_best_models AS

```
select M.Model_name, MM.Mean_Residual_Deviance, MM.RMSE, MM.MSE, MM.MAE,
MM.RMSLE
from Model M
join Model_Metrics MM on MM.Model_Name = M.Model_Name
ORDER BY rmse , mse
LIMIT 50;
```


Model_name	Mean_Residual_Deviance	RMSE	MSE	MAE	RMSLE
GBM_grid_1_AutoML_20190419_190006_model_7	90361700000	300602	90361700000	170227	0.20324
GBM_4_AutoML_20190419_190006	90932600000	301550	90932600000	170770	0.203594
GBM_1_AutoML_20190417_092107	91480100000	302457	91480100000	171898	0.206078
GBM_grid_1_AutoML_20190417_092107_model_1	91581000000	302623	91581000000	171073	0.203811
GBM_4_AutoML_20190416_233703	91662600000	302758	91662600000	171471	0.204274
GBM_3_AutoML_20190419_193106	91750000000	302903	91750000000	171672	0.204166
GBM_2_AutoML_20190417_121910	91810000000	303002	91810000000	171920	0.204805
GBM_grid_1_AutoML_20190419_190006_model_6	91832800000	303039	91832800000	171230	0.205578
GBM_3_AutoML_20190417_121910	91838000000	303048	91838000000	171935	0.205196
GBM_2_AutoML_20190419_190006	91889100000	303132	91889100000	171957	0.204748
GBM_2_AutoML_20190416_233703	91903500000	303156	91903500000	171953	0.205054
GBM_1_AutoML_20190419_193106	92075900000	303440	92075900000	171663	0.205569
GBM_3_AutoML_20190416_233703	92203400000	303650	92203400000	171106	0.20358
GBM_grid_1_AutoML_20190419_190006_model_8	92262100000	303747	92262100000	171426	0.205585
GBM_1_AutoML_20190417_121910	92281400000	303779	92281400000	172470	0.205012
GBM_1_AutoML_20190417_103626	92516000000	304164	92516000000	172391	0.20572
GBM_3_AutoML_20190419_190006	92794000000	304621	92794000000	172327	0.204726
GBM_2_AutoML_20190419_193106	92939500000	304860	92939500000	172467	0.205375
GBM_4_AutoML_20190417_121910	92942000000	304864	92942000000	172690	0.205287
GBM_2_AutoML_20190417_092107	93072100000	305077	93072100000	172842	0.206126
GBM_grid_1_AutoML_20190419_193106_model_6	93225600000	305329	93225600000	172157	0.205503
GBM_1_AutoML_20190419_190006	93246100000	305362	93246100000	172482	0.206082
GBM_4_AutoML_20190419_193106	93275200000	305410	93275200000	172305	0.205156
GBM_grid_1_AutoML_20190419_193106_model_2	93411600000	305633	93411600000	172939	0.20784
GBM_2_AutoML_20190417_103626	93500300000	305778	93500300000	173022	0.206536
GBM_1_AutoML_20190416_233703	93506400000	305788	93506400000	173001	0.206323
GBM_4_AutoML_20190417_092107	93537100000	305838	93537100000	173089	0.206132
GBM_grid_1_AutoML_20190419_190006_model_9	93554000000	305866	93554000000	173230	0.206363
GBM_3_AutoML_20190417_103626	93744500000	306177	93744500000	172498	0.205731
GBM_3_AutoML_20190417_092107	93761400000	306205	93761400000	172664	0.205604
GBM_4_AutoML_20190417_103626	94047400000	306672	94047400000	172980	0.205805
GBM_grid_1_AutoML_20190419_193106_model_7	94126600000	306801	94126600000	172213	0.20486
GBM_grid_1_AutoML_20190417_103626_model_1	94531100000	307459	94531100000	173774	0.206751
GBM_grid_1_AutoML_20190419_193106_model_3	95124200000	308422	95124200000	175192	0.214356
GBM_5_AutoML_20190417_103626	95291000000	308692	95291000000	174519	0.2077
GBM_5_AutoML_20190419_190006	96280800000	310291	96280800000	175470	0.208547
GBM_5_AutoML_20190419_193106	96285500000	310299	96285500000	174727	0.207146
GBM_5_AutoML_20190417_092107	96319600000	310354	96319600000	174895	0.207882
GBM_5_AutoML_20190417_121910	96779900000	311095	96779900000	175250	0.208079
GBM_5_AutoML_20190416_233703	96805000000	311135	96805000000	176071	0.208741
GBM_grid_1_AutoML_20190419_193106_model_1	96808300000	311140	96808300000	177760	0.212076
DRF_1_AutoML_20190417_103626	97308600000	311943	97308600000	174490	0.206746
GBM_grid_1_AutoML_20190419_190006_model_5	97463600000	312192	97463600000	178298	0.211984
DRF_1_AutoML_20190417_121910	97522100000	312285	97522100000	174325	0.20681
DeepLearning_grid_1_AutoML_20190417_121910_model_2	97628900000	312453	97628900000	181825	0.22061
DRF_1_AutoML_20190417_092107	97669800000	312522	97669800000	174488	0.206973
DRF_1_AutoML_20190416_233703	97719600000	312601	97719600000	174594	0.207177
DRF_1_AutoML_20190419_190006	98482800000	313820	98482800000	175685	0.208185
GBM_grid_1_AutoML_20190419_190006_model_1	98649300000	314085	98649300000	179677	0.214534
GBM_grid_1_AutoML_20190419_190006_model_2	99034700000	314698	99034700000	179176	0.213443

4) Create view for models of all algorithms and performance measure

Create view Model_algo_info AS

Select M.Model_name, A.Algorithm_Name, MM.Mean_Residual_Deviance, MM.RMSE,
MM.MSE, MM.MAE, MM.RMSLE

from Model M

join Algorithm A on M.Algorithm_ID = A.Algorithm_ID

join Model_Metrics MM on MM.Model_Name = M.Model_Name;

9. Functions

Functions contain the Views SQL queries.

1) No of models created for a runtime

```
DELIMITER $$
```

```
CREATE FUNCTION no_of_models(runtime INT) RETURNS INT
```

```
DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE models INT ;
```

```
    SET models = ( select Models_generated FROM Model_Runtime WHERE Run_Time  
=runtime);
```

```
    RETURN models;
```

```
END$$
```

```
DELIMITER ;
```

```
=> SELECT no_of_models(500);
```



2) No of ntrees <50 from DRF

```
DELIMITER $$
```

```
CREATE FUNCTION ntrees(no_of_ntrees INT) RETURNS INT
```

```
DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE lessthan50 INT ;
```

```
    SET lessthan50= ( select count(ntrees) FROM DRF_Hyperparameters WHERE  
ntrees<=no_of_ntrees);
```

```
    RETURN lessthan50;
```

```
END$$
```

DELIMITER ;

=> SELECT ntrees(50);

A screenshot of a SQL query result. The first row shows the function call 'ntrees(50)' in a dark box. The second row shows the result '6' in a dark box.

3) Type of prediction

DELIMITER \$\$

CREATE FUNCTION dataset_type(name VARCHAR(255)) RETURNS VARCHAR(255)

DETERMINISTIC

BEGIN

DECLARE dtype VARCHAR(255) ;

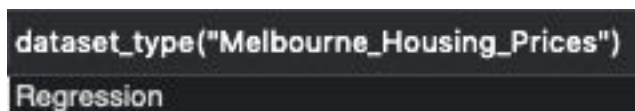
SET dtype= (select Prediction_Type FROM Dataset WHERE Data_Set_Name LIKE name);

RETURN dtype;

END\$\$

DELIMITER ;

=> select dataset_type("Melbourne_Housing_Prices");

A screenshot of a SQL query result. The first row shows the function call 'dataset_type("Melbourne_Housing_Prices")' in a dark box. The second row shows the result 'Regression' in a dark box.

4) Pass dataset_id to get number of columns

DELIMITER \$\$

CREATE FUNCTION no_of_columns(data_id INT) RETURNS VARCHAR(255)

DETERMINISTIC

BEGIN

DECLARE dcolumn INT ;

SET dcolumn = (select Total_Columns FROM Dataset WHERE Dataset_ID = data_id);

RETURN dcolumn;

END\$\$

DELIMITER ;

=> SELECT no_of_columns(1);



A screenshot of a SQL query result. The first row shows the function call 'no_of_columns(1)' in a dark background. The second row shows the result '20' in a lighter background.

10. Indexes

Index contain the Views SQL queries.

1) ALTER TABLE Algorithm

ADD INDEX ind_Algorithm(Algorithm_ID);

2) ALTER TABLE Dataset

ADD INDEX ind_Dataset(Dataset_ID);

3) ALTER TABLE DataSet_Description

ADD INDEX ind_DataSet_Description(Column_ID);

4) ALTER TABLE DRF_Hyperparameters

ADD INDEX ind_DRF_Hyperparameters(Model_name);

5) ALTER TABLE GBM_Hyperparameters

ADD INDEX ind_GBM_Hyperparameters(Model_name);

6) ALTER TABLE GLM_Hyperparameters

ADD INDEX ind_GLM_Hyperparameters(Model_name);

7) ALTER TABLE XRT_Hyperparameters

ADD INDEX ind_XRT_Hyperparameters(Model_name);

8) ALTER TABLE Model

ADD INDEX ind_Model(Model_name);

9) ALTER TABLE Model_Metrics

ADD INDEX ind_Model_Metrics(Model_name);

10) ALTER TABLE Model_Runtime

ADD INDEX ind_Model_Runtime(Runtime_ID);

11. Analytics

The hyperparameter database also uses these data to build models that can predict hyperparameters without search and for visualizing and teaching statistical concepts such as power and bias/variance tradeoff

12. Conclusion

In the project, we were able to create an actual physical database storing the hyperparameters actual values. The following points are covered in the project:

- 1) Conceptual Diagram
- 2) ER Diagram
- 3) Normalization (till 3NF)
- 4) Physical Model
- 5) 10 Use cases
- 6) 4 Views
- 7) 4 Functions
- 8) 4 Indexes
- 9) Analytics

13. Citations & References

Thanks to Prof.Brown, Prabhu, Chitra and Ami for the guidance.

1. Prof.Brown [Github](#)
2. [Project Sample](#) by Prabhu
3. [JSON Library](#)
4. [Data Source](#)
5. [H2O Documentation](#)

14. License

Copyright <2019> <Yadukrishnan Sethumadhavan> <Bhavana Shanmugam>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.