HYPERPARAMETER DATABASE PROJECT

DB - Team 05

Abstract:

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.

Objectives:

Our objectives as a database team is to perform the following steps

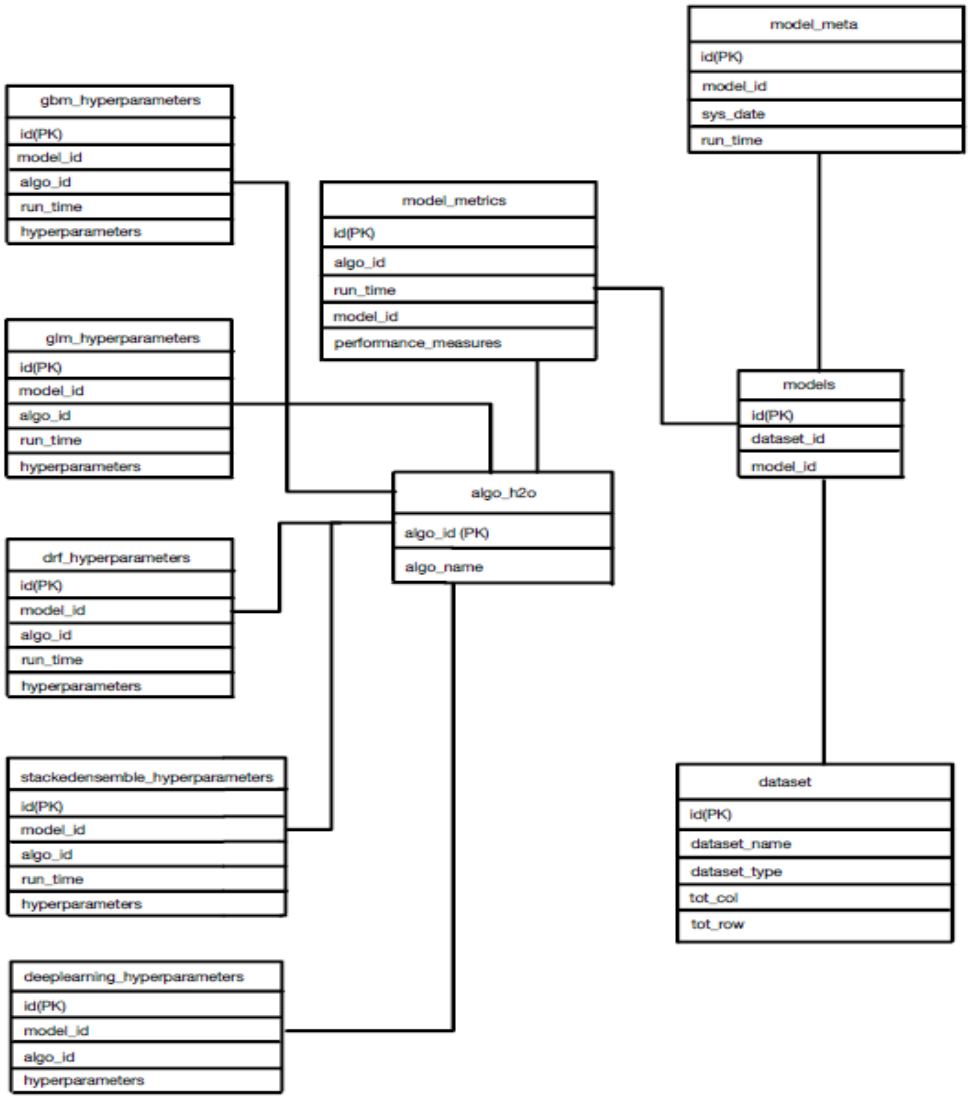
- Create a conceptual model and an entity relationship diagram
- Parse the json files and extract the hyperparameters into a csv file
- Normalization upto the 3rd normal form
- Create a physical database and populate it
- Create sql queries of real world relevance and run them
- Create functions and indexes
- Perform analytics on the database i.e which model has the best accuracy, etc

Dataset Description:

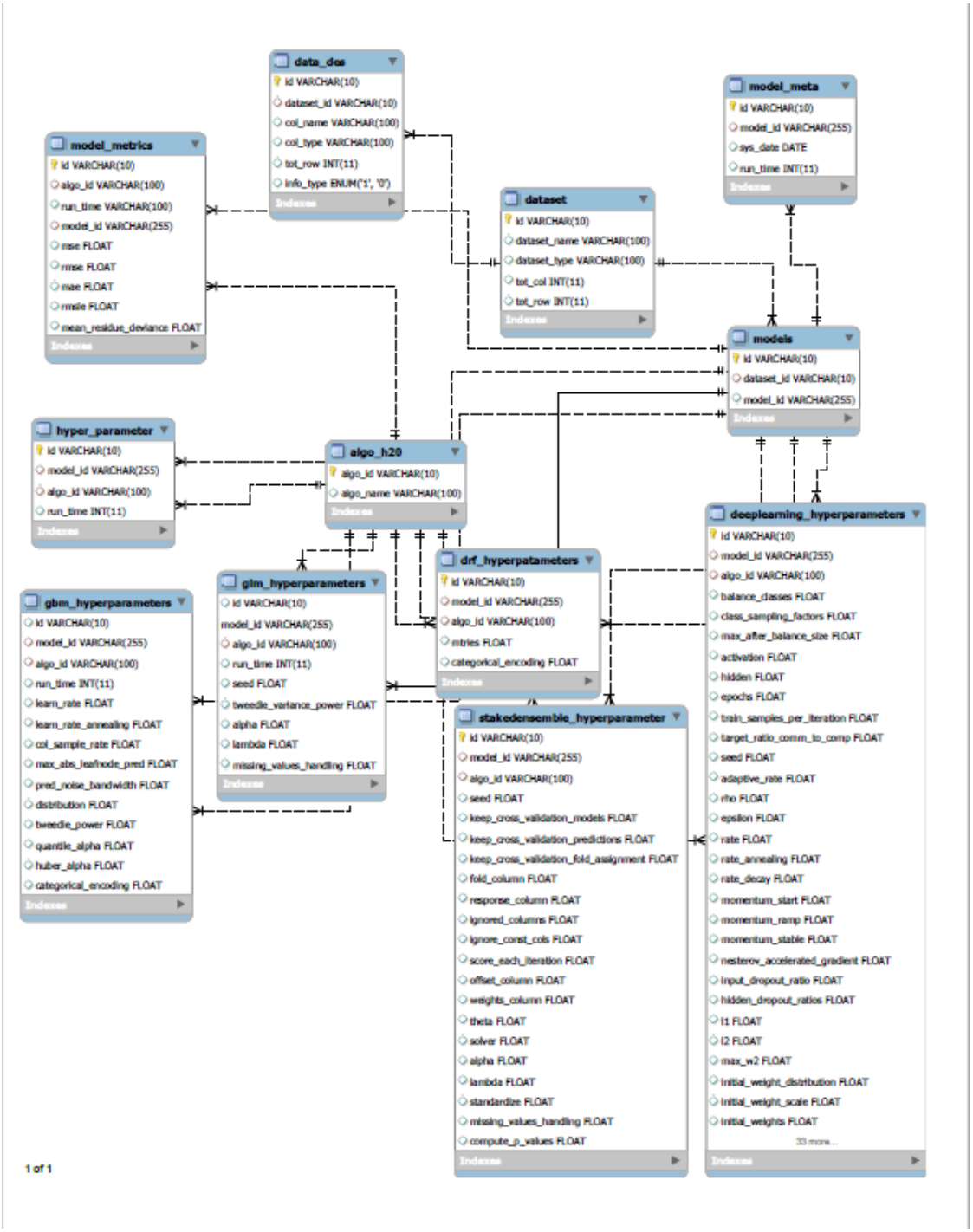
The dataset here is a sample of the transactions made in a retail store. The store wants to know better the customer purchase behaviour against different products. Specifically, here the problem is a regression problem where we are trying to predict the dependent variable (the amount of purchase) with the help of the information contained in the other variables.

Our dataset of 538,000 observations about the Black Friday in a retail store, it contains 12 columns of different kinds of variables either numerical or categorical.

Conceptual Schema:



Entity Relationship Diagram:



Normalization:

Normalization is done so that each table represents a single subject, no data item will be unnecessarily stored in more than one table, all the attributes in a table are dependent on the primary key.

First Normal Form:

- All our key attributes have been defined.
- There are no repeating groups in all the tables
- All our attributes are dependent on primary key

Second Normal Form:

- All the tables are in first normal form
- There are no partal dependencies i.e none of the attributes are dependent on only a portion of primary key

Third Normal Form:

- All the tables are in second normal form
- All our non-key atrributes are dependent only on the keys

Physical Model:

For creating a physical model we used MySql Workbench and below is a snippet of the tables that we have created

```
drop table dataset;
CREATE TABLE dataset(
id VARCHAR(10) primary key,
dataset_name VARCHAR(100),
dataset_type VARCHAR(100),
tot_col INT,
tot_row INT);

drop table data_des;
CREATE TABLE data_des(
id VARCHAR(10) primary key,
dataset_id VARCHAR(10),
col_name VARCHAR(100),
col_type VARCHAR(100),
tot_row INT,
info_type ENUM("1", "0"),
FOREIGN KEY(dataset_id) REFERENCES dataset(id));

CREATE TABLE models(
id VARCHAR(10) primary key not null,
dataset_id VARCHAR(10),
model_id VARCHAR(255),
```

Usecases:

Usecases are the real world applications of the database on which sql queries are run

Usecase 1: To find models for a given run time and the name of the algorithm

Input:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `models_based_on_runtime`(in runtime int)
BEGIN
select h.model_id as Hyperparameter_models, a.algo_name
from hyperparameters h
join algo_h20 a on
h.algo_id = a.algo_id
where h.runtime like concat('%',runtime ,'%')
;
END
```

Output:

```
call hyperparameter_project.models_based_on_runtime(500);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Hyperparameter_models	algo_name
GBM_5_AutoML_20190416_222141	GBM
GBM_2_AutoML_20190416_222141	GBM
GBM_1_AutoML_20190416_222141	GBM
GBM_1_AutoML_20190416_151701	GBM
GBM_5_AutoML_20190416_190838	GBM
GBM_3_AutoML_20190416_190838	GBM
GBM_4_AutoML_20190416_190838	GBM
GBM_2_AutoML_20190416_190838	GBM
GBM_grid_1_AutoML_20190416_235225_model_2	GBM
GBM_grid_1_AutoML_20190416_235225_model_1	GBM
DRF_1_AutoML_20190416_151701	DRF
DRF_1_AutoML_20190416_190838	DRF
DRF_1_AutoML_20190416_235225	DRF
GBM_1_AutoML_20190419_174950	GBM
GBM_2_AutoML_20190419_174950	GBM
GBM_3_AutoML_20190419_174950	GBM

Usecase 2: List of hyperparamter models based on algorithms

Input:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `models_based_on_algorithm`(in algo_id int)
BEGIN
select h.model_id as Hyperparameter_models, a.algo_name as algorithm
from hyperparameters h
join algo_h20 a on
h.algo_id = a.algo_id
where h.algo_id like concat('%',algo_id ,'%')
;
END
```

Output:

```
call hyperparameter_project.models_based_on_algorithm(1);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Hyperparameter_models	algorithm
GBM_5_AutoML_20190416_222141	GBM
GBM_2_AutoML_20190416_222141	GBM
GBM_1_AutoML_20190416_222141	GBM
GBM_1_AutoML_20190416_151701	GBM
GBM_5_AutoML_20190416_190838	GBM
GBM_3_AutoML_20190416_190838	GBM
GBM_4_AutoML_20190416_190838	GBM
GBM_2_AutoML_20190416_190838	GBM
GBM_1_AutoML_20190416_190838	GBM
GBM_1_AutoML_20190416_154932	GBM
GBM_5_AutoML_20190416_154932	GBM
GBM_4_AutoML_20190416_154932	GBM
GBM_3_AutoML_20190416_154932	GBM
GBM_2_AutoML_20190416_154932	GBM
GBM_5_AutoML_20190416_235225	GBM
GBM_3_AutoML_20190416_235225	GBM

Usecase 3: Total number of hyperparameter models for a specific runtime

Input:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `number_of_models_based_on_runtime`(in runtime int)
BEGIN
select count(h.model_id) as Total_Hyperparameter_models
from hyperparameters h
where h.runtime like concat('%',runtime,'%')
group by runtime
;
END
```

Output:

```
call hyperparameter_project.number_of_models_based_on_runtime(500);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_Hyperparameter_models
29

Usecase 4: Total number of hyperparameter models for a specific algorithm

Input:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `number_of_models_per_algorithm`(in algorithm varchar(20))
BEGIN
select  a.algo_name, count(h.model_id)
from hyperparameters h
inner join algo_h20 a on
h.algo_id = a.algo_id
where a.algo_name like concat('%',algorithm ,'%')
group by h.algo_id;
END
```

Output:

```
call hyperparameter_project.number_of_models_per_algorithm('glm');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

algo_name	count(h.model_id)
GLM	5

Usecase 5: List of DRF models generated for a given runtime

Input:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `list_of_drf_models_with_runtime` AS
    SELECT
        `hyperparameters`.`model_id` AS `model_id`,
        `hyperparameters`.`runtime` AS `runtime`,
        `algo_h20`.`algo_name` AS `algorithm`
    FROM
        (`hyperparameters`
        JOIN `algo_h20` ON ((`hyperparameters`.`algo_id` = `algo_h20`.`algo_id`)))
    WHERE
        (`hyperparameters`.`model_id` LIKE 'DRF%')
```

Output:

```
SELECT * FROM hyperparameter_project.list_of_drf_models_with_runtime;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

model_id	runtime	algorithm
DRF_1_AutoML_20190416_151701	500	DRF
DRF_1_AutoML_20190416_154932	2000	DRF
DRF_1_AutoML_20190416_190838	1500	DRF
DRF_1_AutoML_20190416_235225	2500	DRF

Usecase 6: List of GLM models generated for a given runtime

Input:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `list_of_glm_models_with_runtime` AS
    SELECT
        `hyperparameters`.`model_id` AS `model_id`,
        `hyperparameters`.`runtime` AS `runtime`,
        `algo_h20`.`algo_name` AS `algorithm`
    FROM
        (`hyperparameters`
        JOIN `algo_h20` ON ((`hyperparameters`.`algo_id` = `algo_h20`.`algo_id`)))
    WHERE
        (`hyperparameters`.`model_id` LIKE 'GLM%')
```

Output:

```
SELECT * FROM hyperparameter_project.list_of_glm_models_with_runtime;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

model_id	runtime	algorithm
GLM_grid_1_AutoML_20190419_174950_model	2500	GLM
GLM_grid_1_AutoML_20190419_184714_model_1	500	GLM
GLM_grid_1_AutoML_20190419_200713_model_1	1500	GLM
GLM_grid_1_AutoML_20190419_200713_model_1	1000	GLM
GLM_grid_1_AutoML_20190419_205801_model	2000	GLM

Usecase 7: List of Deeplearning models generated for a given runtime

Input:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `list_of_deeplearning_models_with_runtime` AS
    SELECT
        `hyperparameters`.`model_id` AS `model_id`,
        `hyperparameters`.`runtime` AS `runtime`,
        `algo_h20`.`algo_name` AS `algorithm`
    FROM
        (`hyperparameters`
        JOIN `algo_h20` ON ((`hyperparameters`.`algo_id` = `algo_h20`.`algo_id`)))
    WHERE
        (`hyperparameters`.`model_id` LIKE 'DEEPLEARNING%')
```

Output:

```
SELECT * FROM hyperparameter_project.list_of_deeplearning_models_with_runtime;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

model_id	runtime	algorithm
DeepLearning_1_AutoML_20190416_222141	2000	DEEPLEARNING
DeepLearning_1_AutoML_20190416_235225	2500	DEEPLEARNING

Usecase 8: Finding the datatype of a specific column

Input:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Datatype`(attribute varchar(20))
BEGIN
select col_type as 'Datatype of Column'
from data_des
where col_name like concat('%',attribute ,'%')
;
END
```

Output:

```
call hyperparameter_project.Datatype('purchase');
```

Datatype of Column
int

Usecase 9: List of columns and their datatypes

Input:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `columnname_and_columntype` AS
    SELECT
        `data_des`.`col_name` AS `name of column`,
        `data_des`.`col_type` AS `datatype`
    FROM
        `data_des`
```

Output:

```
SELECT * FROM hyperparameter_project.columnname_and_columntype;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name of column	datatype
user_id	float
Product_Category_2	int
Product_Category_3	int
Purchase	int
Product_ID	float
Gender	text
Age	int
Occupation	int
City_Category	string
Stay_In_Current_City_Years	int
Marital_Status	binary

Usecase 10: It gives of the model along with the name of the algorithm used

Input:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `model_details` AS
    SELECT
        `model_metrics`.`model_id` AS `model_id`,
        `algo_h20`.`algo_name` AS `algo_name`,
        `model_metrics`.`mae` AS `mae`,
        `model_metrics`.`rmse` AS `rmse`,
        `model_metrics`.`mse` AS `mse`,
        `model_metrics`.`rmsle` AS `rmsle`,
        `model_metrics`.`mean_residue_deviance` AS `mean_residue_deviance`
    FROM
        (`model_metrics`
        JOIN `algo_h20` ON ((`model_metrics`.`algo_id` = `algo_h20`.`algo_id`)))
```

Output:

```
SELECT * FROM hyperparameter_project.model_details;
```

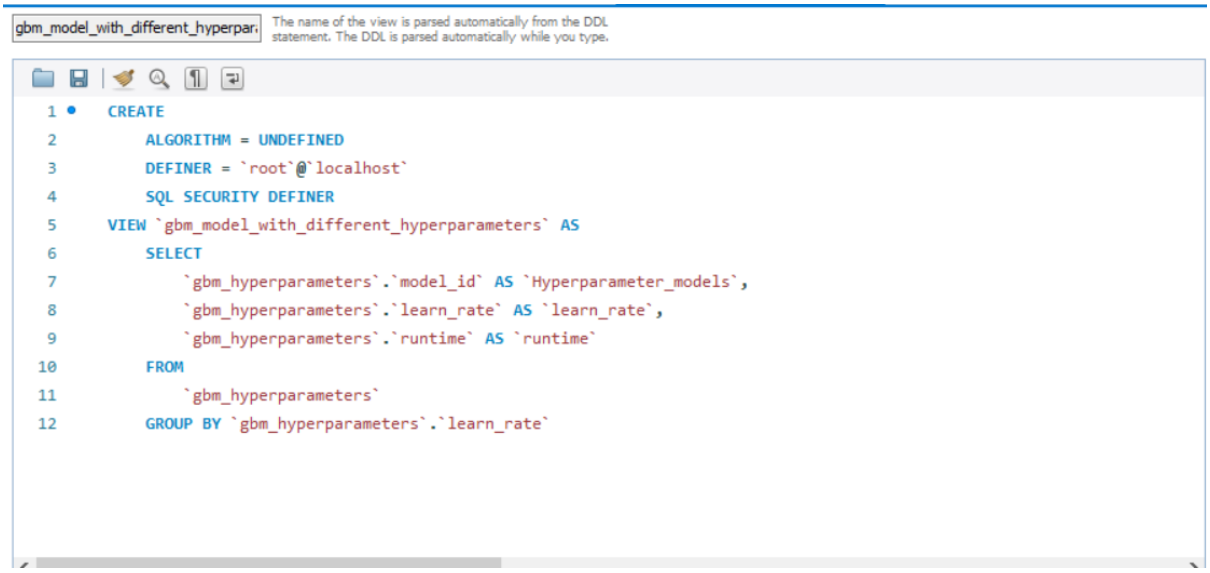
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

model_id	algo_name	mae	rmse	mse	rmsle	mean_residue_deviance
GBM_grid_1_AutoML_20190419_205801_model_2	GBM	4038.099071	4971.641844	24717222.62	0.665558779	24717222.62
GLM_grid_1_AutoML_20190419_174950_model_1	GLM	4047.516837	4981.021637	24810576.55	0.666416712	24810576.55
GLM_grid_1_AutoML_20190419_200713_model_1	GLM	4047.516837	4981.021637	24810576.55	0.666416712	24810576.55
GLM_grid_1_AutoML_20190419_205801_model_1	GLM	4047.516837	4981.021637	24810576.55	0.666416712	24810576.55
DRF_1_AutoML_20190417_172402	DRF	2240.492733	2975.032485	8850818.284	0.423903644	8850818.284
DRF_1_AutoML_20190419_174950	DRF	2290.047757	3041.318864	9249620.432	0.435612092	9249620.432
DRF_1_AutoML_20190419_184714	DRF	2199.219806	2915.14335	8498060.75	0.416115391	8498060.75
DRF_1_AutoML_20190419_200713	DRF	2093.235416	2777.184788	7712755.349	0.385330651	7712755.349
DRF_1_AutoML_20190419_205801	DRF	2165.275356	2870.013378	8236976.791	0.407344774	8236976.791
DeepLearning_1_AutoML_20190419_184714	DEEPLEARNING	2010.320172	2708.861926	7337932.932	0.35517041	7337932.932
DeepLearning_1_AutoML_20190419_205801	DEEPLEARNING	2020.327307	2708.163766	7334150.984	0.353313218	7334150.984
StackedEnsemble_AllModels_AutoML_20190417...	STAKEDENSEM...	2821.637544	3582.611744	12835106.91	0.523714828	12835106.91
StackedEnsemble_BestOfFamily_AutoML_20190...	STAKEDENSEM...	3248.211483	4074.906117	16604859.86	0.577518561	16604859.86
StackedEnsemble_BestOfFamily_AutoML_20190...	STAKEDENSEM...	3248.211483	4074.906117	16604859.86	0.577518561	16604859.86
StackedEnsemble_AllModels_AutoML_20190419...	STAKEDENSEM...	2571.864559	3292.441258	10840169.44	0.488331308	10840169.44
StackedEnsemble_BestOfFamily_AutoML_20190...	STAKEDENSEM...	3250.816018	4077.874558	16629060.91	0.578001622	16629060.91

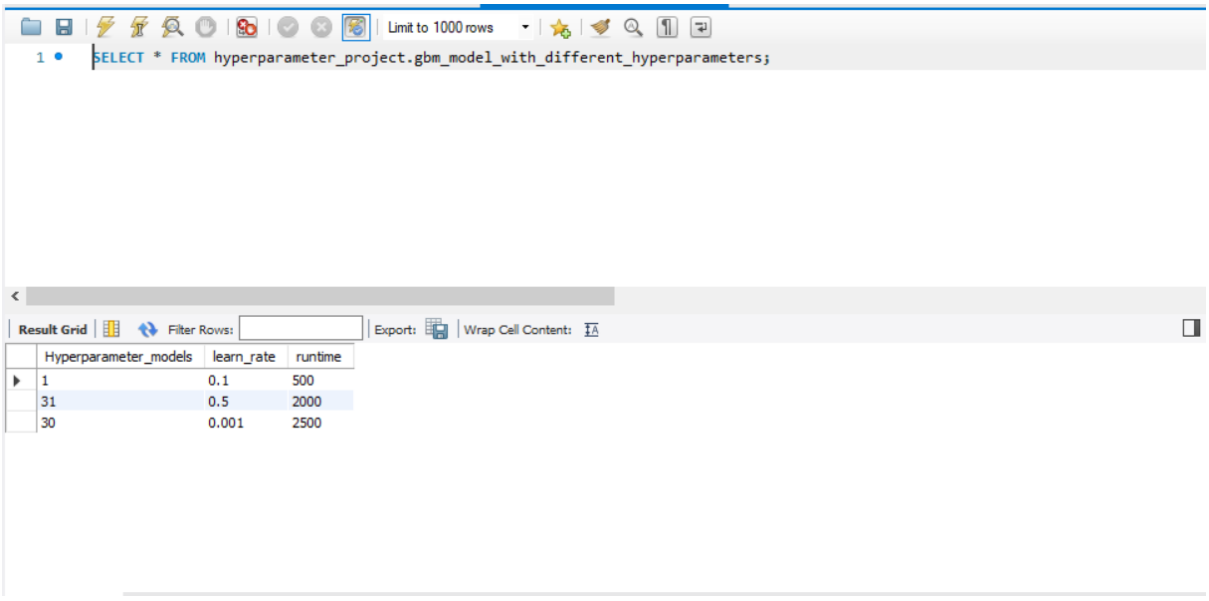
Views:

View 1: GBM with different hyperparameters

Input:



Output:



View 2: List of GBM models with runtimes

Input:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `list_of_gbm_models_with_runtime` AS
    SELECT
        `hyperparameters`.`model_id` AS `model_id`,
        `hyperparameters`.`runtime` AS `runtime`,
        `algo_h20`.`algo_name` AS `algorithm`
    FROM
        (`hyperparameters`
        JOIN `algo_h20` ON ((`hyperparameters`.`algo_id` = `algo_h20`.`algo_id`)))
    WHERE
        (`hyperparameters`.`model_id` LIKE 'GBM%')
```

Output:

```
SELECT * FROM hyperparameter_project.list_of_gbm_models_with_runtime;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

model_id	runtime	algorithm
GBM_5_AutoML_20190416_222141	500	GBM
GBM_2_AutoML_20190416_222141	500	GBM
GBM_1_AutoML_20190416_222141	500	GBM
GBM_1_AutoML_20190416_151701	500	GBM
GBM_5_AutoML_20190416_190838	500	GBM
GBM_3_AutoML_20190416_190838	500	GBM
GBM_4_AutoML_20190416_190838	500	GBM
GBM_2_AutoML_20190416_190838	500	GBM
GBM_1_AutoML_20190416_190838	1000	GBM
GBM_1_AutoML_20190416_154932	1000	GBM
GBM_5_AutoML_20190416_154932	1000	GBM
GBM_4_AutoML_20190416_154932	1000	GBM
GBM_3_AutoML_20190416_154932	1000	GBM
GBM_2_AutoML_20190416_154932	1000	GBM
GBM_5_AutoML_20190416_235225	1000	GBM

View 3: Number of models based on algorithm

Input:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `number_of_models_based_on_algorythm` AS
    SELECT
        `a`.`algo_name` AS `algo_name`,
        COUNT(`h`.`model_id`) AS `count(h.model_id)`
    FROM
        (`hyperparameters` `h`
        JOIN `algo_h20` `a` ON ((`h`.`algo_id` = `a`.`algo_id`)))
    GROUP BY `h`.`runtime`
```

Output:

```
SELECT * FROM hyperparameter_project.number_of_models_based_on_algorythm;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

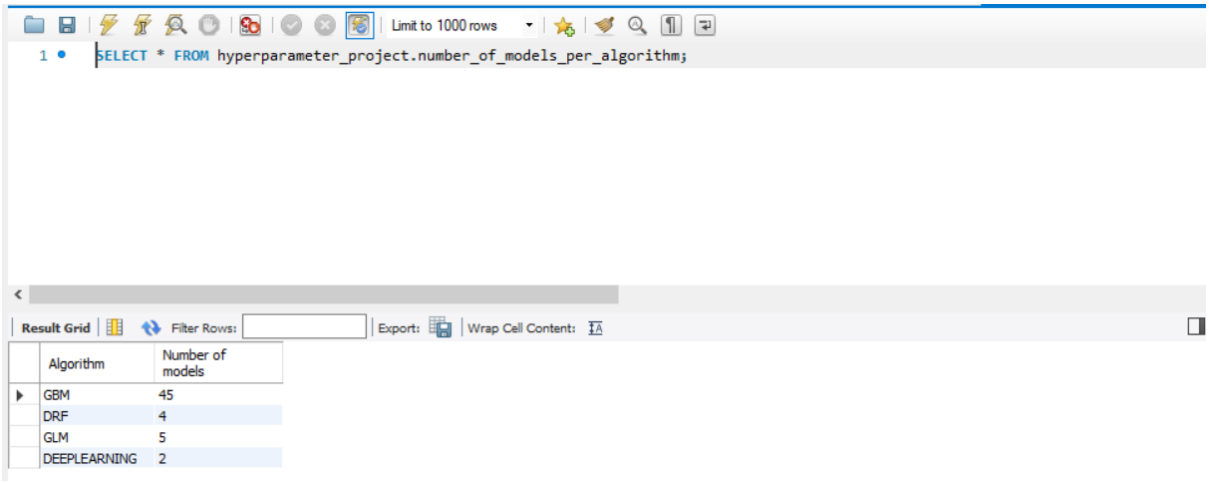
algo_name	count(h.model_id)
GBM	10
GBM	11
GBM	8
DRF	11
DRF	16

View 4: Number of models per algorithm

Input:

```sql
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `number_of_models_per_algorithm` AS
    SELECT
        `a`.`algo_name` AS `Algorithm`,
        COUNT(`h`.`model_id`) AS `Number of models`
    FROM
        (`hyperparameters` `h`
        JOIN `algo_h20` `a`)
    WHERE
        (`h`.`algo_id` = `a`.`algo_id`)
    GROUP BY `h`.`algo_id`
```

Output:



```sql
SELECT * FROM hyperparameter_project.number_of_models_per_algorithm;
```

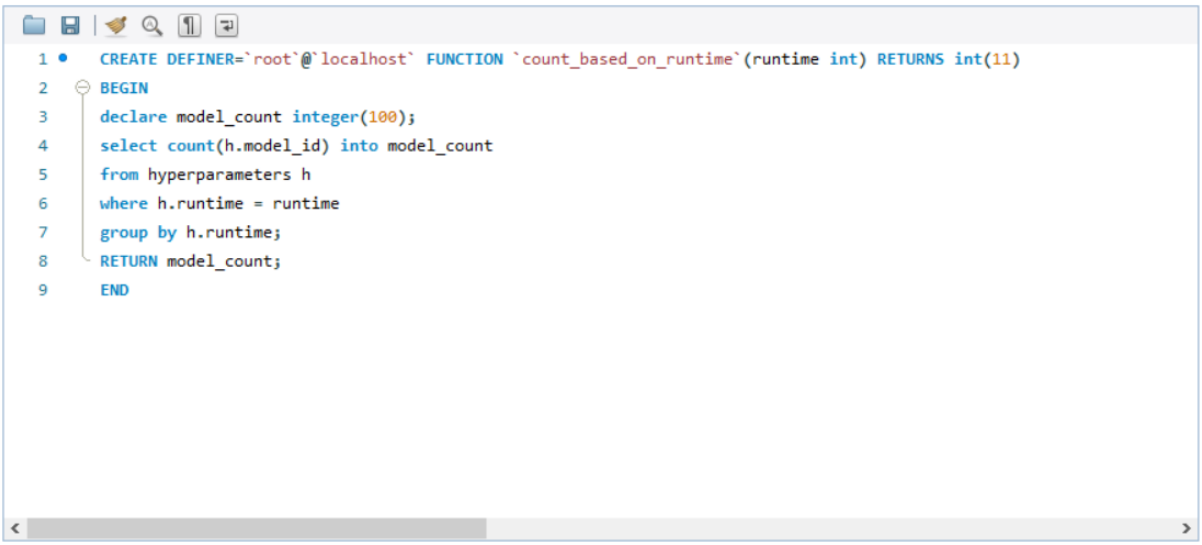
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Algorithm	Number of models
GBM	45
DRF	4
GLM	5
DEEPLEARNING	2

Functions:

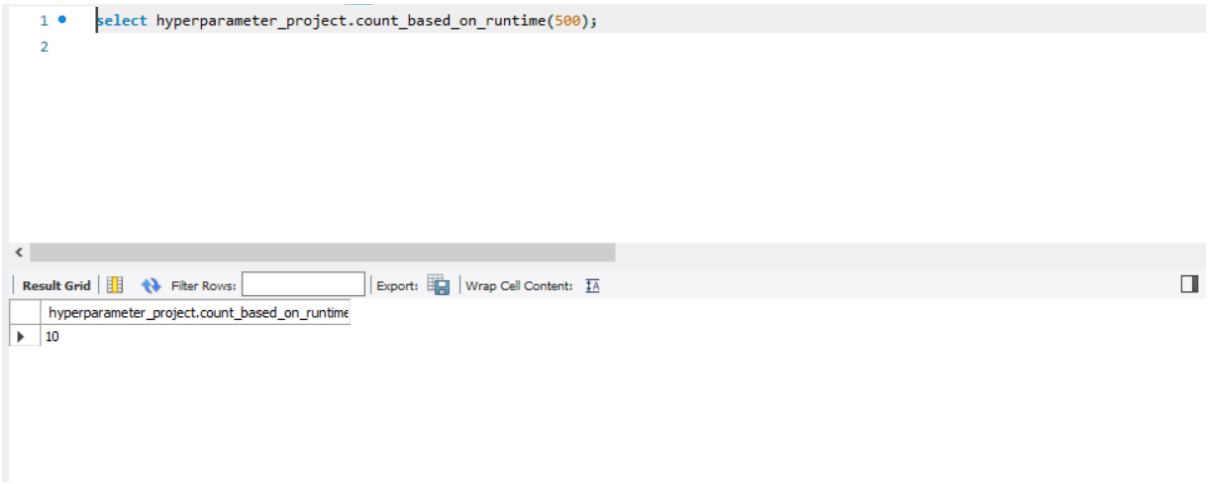
Function 1:

Input:



```
CREATE DEFINER=`root`@`localhost` FUNCTION `count_based_on_runtime`(runtime int) RETURNS int(11)
BEGIN
declare model_count integer(100);
select count(h.model_id) into model_count
from hyperparameters h
where h.runtime = runtime
group by h.runtime;
RETURN model_count;
END
```

Output:



```
select hyperparameter_project.count_based_on_runtime(500);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

hyperparameter_project.count_based_on_runtime
10

Function 2:

Input:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `MAE_for_model`(modelname TEXT) RETURNS double
BEGIN
DECLARE mae_value DOUBLE;
SELECT MAE INTO mae_value FROM model_metrics
WHERE model_id = modelname;
RETURN mae_value;
END
```

Output:

```
select hyperparameter_project.MAE_for_model('DRF_1_AutoML_20190419_174950');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

hyperparameter_project.MAE_for_model('DRF_1_
2290.047757

Function 3:

Input:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `MSE_for_model`(modelname TEXT) RETURNS double
BEGIN
DECLARE mse_value DOUBLE;
SELECT mse INTO mse_value FROM model_metrics
WHERE model_id = modelname;
RETURN mse_value;
END
```

Output:

```
select hyperparameter_project.MSE_for_model('GLM_grid_1_AutoML_20190419_174950_model_1');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

hyperparameter_project.MSE_for_model('GLM_gri
24810576.55

Function 4:

Input:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `RMSE_for_model`(modelname TEXT) RETURNS double
BEGIN
DECLARE rmse_value DOUBLE;
SELECT rmse INTO rmse_value FROM model_metrics
WHERE model_id = modelname;
RETURN rmse_value;
END
```

Output:

```
select hyperparameter_project.RMSE_for_model('DRF_1_AutoML_20190417_172402');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

hyperparameter_project.RMSE_for_model('DRF_1
2975.032485

Conclusion:

The database created by us can be thus used to find the hyperparameters that hold significance while tuning the model by querying the database for the desired results.

References:

References Nik Brown Github. https://github.com/nikbearbrown/INFO_6210

Normalization - 1NF, 2NF, 3NF and 4NF https://www.youtube.com/watch?v=UrYLYV7WSHM&t=71s

W3schools https://www.w3schools.com/sql

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

http://docs.h2o.ai/h2o/latest-stable/h2o-docs/grid-search.html?highlight=hyperparameters#supported-gridsearch-hyperparameters

Contributions:

We would like to thank our Prof. Nicholas Brown and our project managers who helped us and guided us through the entire project.