DATABASE PROJECT REPORT

HYPERPARAMETER PROJECT -DB13

INFO6210 Data Mgt and Database Design SEC 03 Spring 2019

TEAM MEMBERS:

PURVANG JAYESH THAKKAR	001387983
IRA PANTBALEKUNDRI	001423854
NIKITA GAWDE	001409592

INTRODUCTION

What are Hyperparameters?

Hyperparameters are configuration variables that are external to the model and whose values cannot be estimated from data. They can't be learned directly from the data in standard model training. They are almost always specified by the machine learning engineer prior to training.
Hyperparameters have to be given manually along with the input training data with the Machine learning algorithm.
For example,
Input→**Machine Learning Algorithm**→Model

In the above example,
The input will consist of 2 things: one being the training data and two is the configuration parameters which we shall define while passing with the Machine Algorithm to get the derived Model and it's model parameters.

Input→Training Data + Configuration Parameters (Hyperparameters)

The output which we get is the representation of the input data; Once the model is ready we can provide any test data and it shall predict the desired output or predicted output for it. The model consist of certain parameters and those parameters are nothing, but they model parameters and they vary from algorithm to algorithm. They are different from the Hyperparameters.

Hyperparameters are advanced and are usually given by Machine Learning engineers to Machine learning algorithms while training the data. There are no fixed ranges but must be manually supplied by us. It cannot automatically generate them.
Example:
If we have SVM (Support Vector Machine) Algorithm, the hyperparameters supplied for it would Sigma, Kernel and C. We need to supply different values for each of these hyperparameters.
The model parameters which are generated after the training are like Support vector or weights(co efficient of the support vector)

ABSTRACT

The goal of this project is to provide a database which will store all the hyperparameters for a particular model for a given dataset.

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.

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.

We think the apart from storing the Hyperparameter values in the Database, we can also probably visualize some plots by comparing which are best or not, and have a comparison done using matplotlib in python.

DATA SOURCE

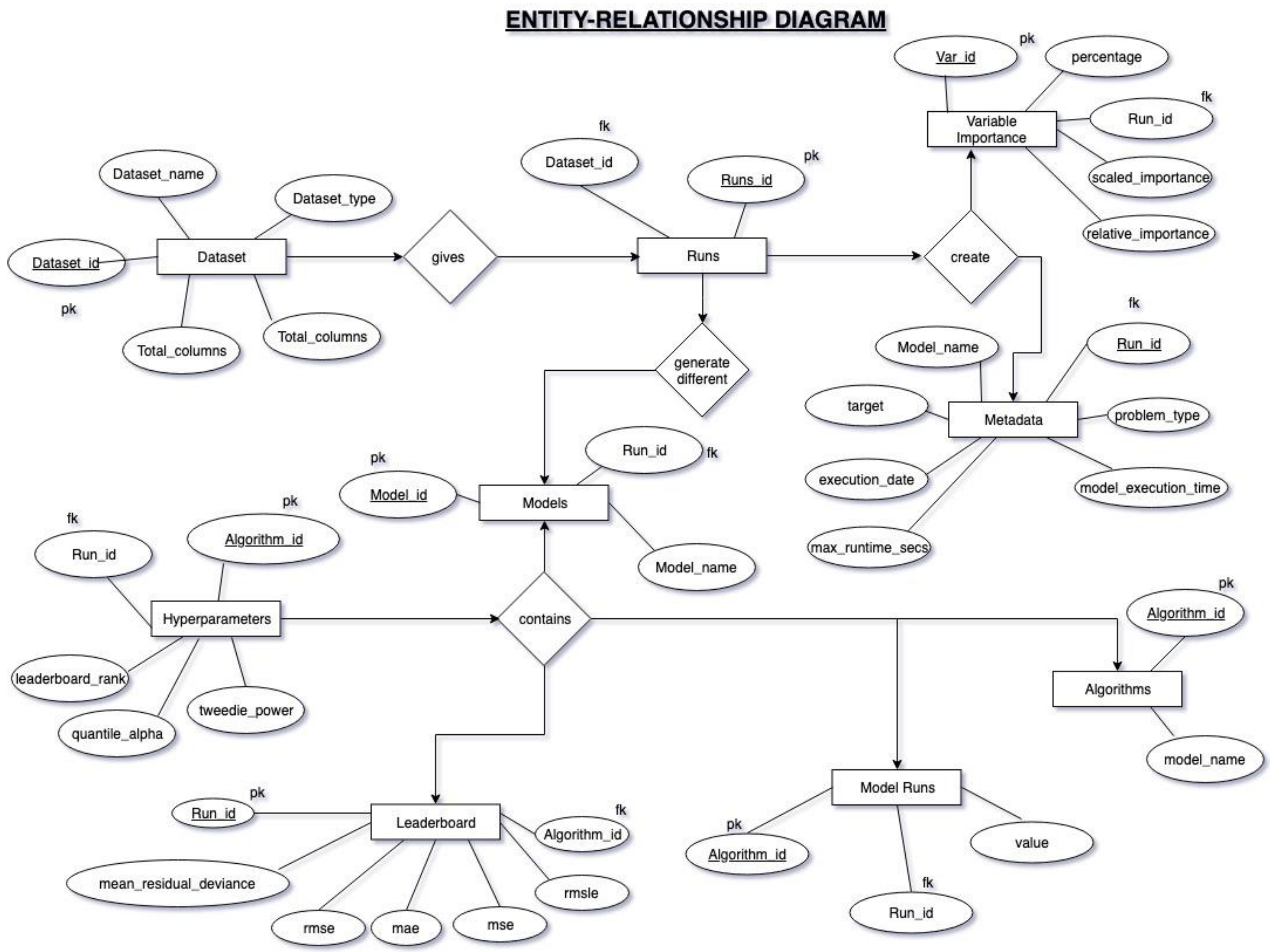
The dataset was obtained from Data world and aggregated from multiple sources including American Community Service, cancer.org.

The goal of the dataset is to determine the cancer mortality rate by using multiple regression models such as GBM, Deep Learning, Stacked Ensembles, DRF and so on.
Our objective is to store the JSON files and analyze the mortality rate is estimated using different variables of the dataset as predictors. These predictors are stored in metadata.

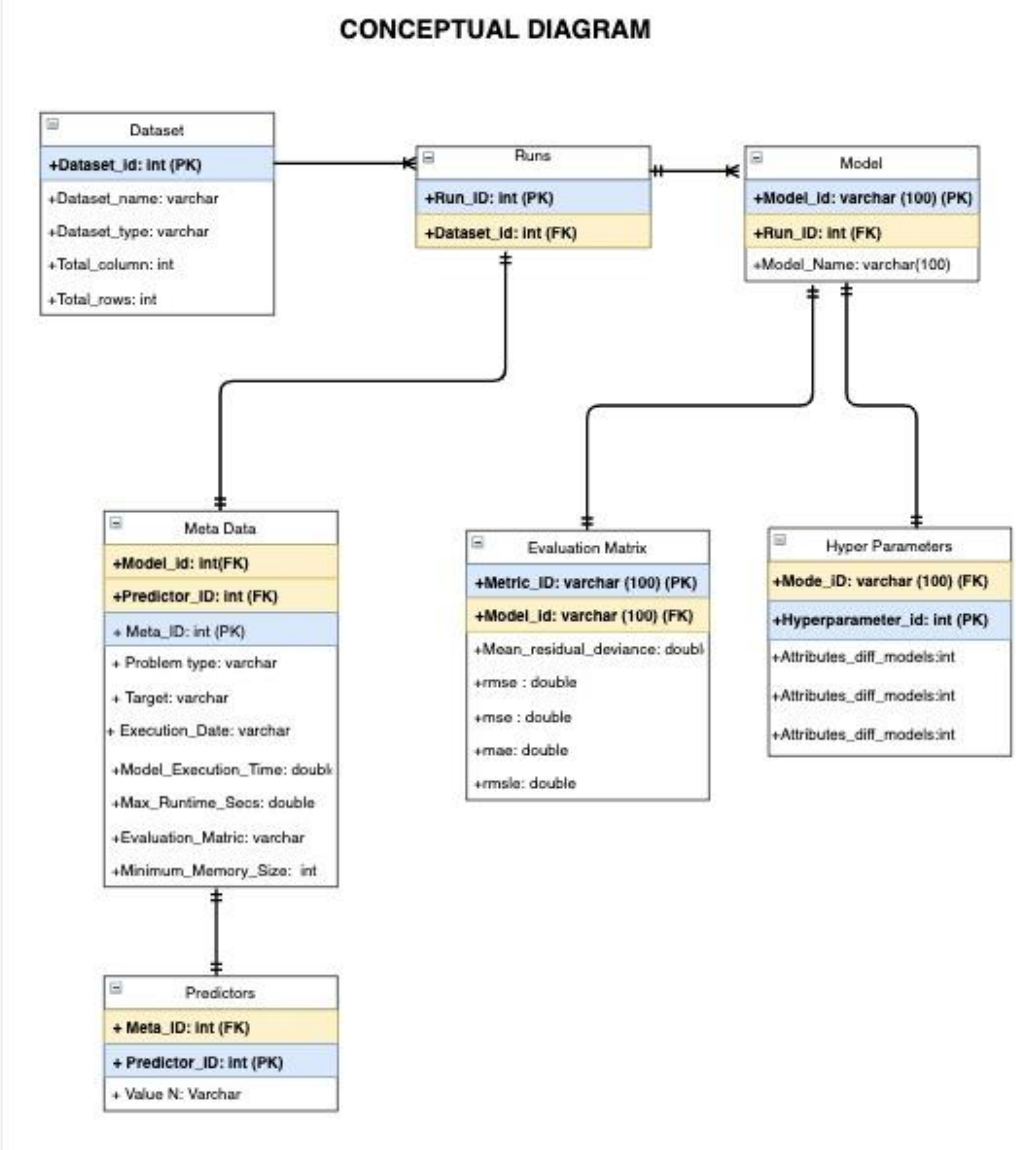
Also, the Data Science team handed as with 5 below runs with 1251 JSON files and we have stored all of them in our database, extracting each one of them and converting them into csvs.

1. **MKmhZIltm----54**
2. **CAb9R3kai-----128**
3. **WdShVGuoh---223**
4. **ON7BbTEGe---343**
5. **gCje7dhU4-----503**

ENTITY-RELATIONSHIP DIAGRAM



NORMALIZATION:



The above diagram was the conceptual schema before normalization.

For normalization

First Normal Form: Firstly, we created a Main table which contains the run_id and dataset_id. The primary key is Run_id and the foreign key is dataset_id.

Next, we created separate table for hyperparameters, and the metadata liked with the algorithm_id as primary key.

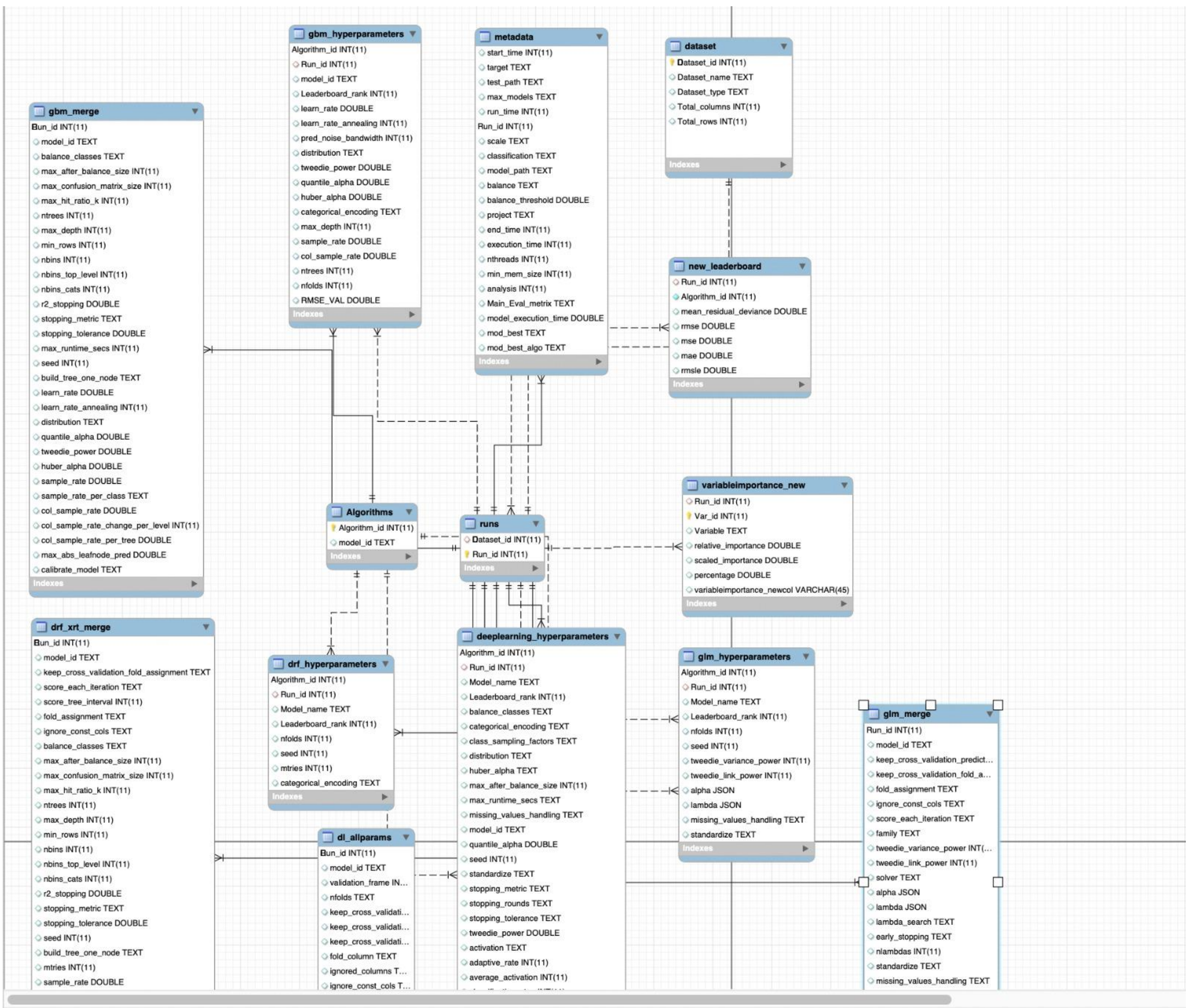
Second Normal Form: In all the tables there was a partial dependency due to presence of composite keys so to normalize we formed a different table Algorithms which contained just the algorithm_id and the name of that model. Then we joined all our tables using JOIN operations.

Third Normal Form: All requirements of 2NF are met.

We have eliminated all fields that do not directly depend on the primary key; that is no transitive dependencies.

The final conceptual schema is shown below:

CONCEPTUAL SCHEMA



USE CASES

1.Find runtime of all data sets

SELECT metadata.Run_id,run_time

FROM metadata

INNER JOIN runs ON metadata.Run_id = runs.Run_id

WHERE runs.Dataset_id = 1;

Result:

Result Grid Filter Rows:

Run_id	run_time
1	333
2	777
3	999
4	1333
5	1555

Result 2

2.To find the variable importance for 2[nd] run

SELECT Run_id, Variable, relative_importance, scaled_importance, percentage

FROM variableimportance_new

WHERE Run_id = 2 limit 10;

Result:

Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:

Run_id	Variable	relative_importance	scaled_importance	percentage
2	incidenceRate	2199695.25	1	0.204773521
2	PctBachDeg25_Over	2035948.625	0.925559404	0.189530058
2	AvgHouseholdSize	191581.7188	0.087094664	0.017834681
2	PctOtherRace	184559.1094	0.083902127	0.017180934
2	PctBlack	183525.4688	0.083432225	0.01708471
2	PctUnemployed16_Over	181384.5	0.082458922	0.016885404
2	MedianAgeFemale	164529.7188	0.074796597	0.015316362
2	PctHS18_24	159675.0938	0.072589643	0.014864437
2	PctMarriedHouseholds	147014.1094	0.066833853	0.013685803

variableimportance_new 4

3.Which models for gbm takes more than 100 ntrees for first 3 runs?

SELECT Run_id, model_id, Leaderboard_rank, ntrees

FROM gbm_hyperparameters

WHERE ntrees> 100 AND Run_id BETWEEN 1 AND 3;

Result:

Run_id	model_id	Leaderboard_rank	ntrees
1	GBM_grid_1_AutoML_20190419_175717_model_20	13	108
1	GBM_grid_1_AutoML_20190419_175717_model_1	14	109
1	GBM_grid_1_AutoML_20190419_175717_model_13	19	192
1	GBM_grid_1_AutoML_20190419_175717_model_12	21	194
1	GBM_grid_1_AutoML_20190419_175717_model_30	25	207
1	GBM_grid_1_AutoML_20190419_175717_model_4	26	216
1	GBM_grid_1_AutoML_20190419_175717_model_27	28	204
1	GBM_grid_1_AutoML_20190419_175717_model_10	29	213
1	GBM_grid_1_AutoML_20190419_175717_model_24	30	230
1	GBM_grid_1_AutoML_20190419_175717_model_28	31	256
1	GBM_grid_1_AutoML_20190419_175717_model_5	33	230
1	GBM_grid_1_AutoML_20190419_175717_model_26	34	242
1	GBM_grid_1_AutoML_20190419_175717_model_29	9	103
1	GBM_grid_1_AutoML_20190419_175717_model_18	11	114

gbm_hyperparameters 5

4.What is the range of relative importance for the variable BirthRate for all the run IDs?

SELECT MAX(relative_importance),

MIN(relative_importance)

FROM variableimportance_new

WHERE Variable = 'BirthRate' AND Run_id BETWEEN 1 AND 5;

Result:

MAX(relative_importance)	MIN(relative_importance)
145271.4531	58331.953125

Result 2

5.What is sample_rate and max_depth for GBM hyperparameter for the 3rd Run?

SELECT model_id,Run_id,sample_rate, max_depth FROM hyperparameter_db.gbm_hyperparameters

WHERE Run_id = 3;

Result:

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

model_id	Run_id	sample_rate	max_depth
GBM_grid_1_AutoML_20190419_181340_model_48	3	1	14
GBM_grid_1_AutoML_20190419_185907_model_48	3	1	14
GBM_grid_1_AutoML_20190419_175717_model_19	3	0.6	10
GBM_grid_1_AutoML_20190419_185907_model_66	3	1	13
GBM_grid_1_AutoML_20190419_185907_model_79	3	0.7	10
GBM_grid_1_AutoML_20190419_185907_model_27	3	0.8	10
GBM_grid_1_AutoML_20190419_181340_model_27	3	0.8	10
GBM_grid_1_AutoML_20190419_175717_model_27	3	0.8	10

gbm_hyperparameters 8

6.What is the difference of end_time between run 1 and run 5?

SELECT end_time

FROM metadata

GROUP BY Run_id

HAVING SUM(case when Run_id = 1 then end_time else 0 end) -

SUM(case when Run_id = 2 then end_time else 0 end) > 0

Result:

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

end_time
1555711037

7.Find the average of all the evaluation matrices from leaderboard?

SELECT AVG(mean_residual_deviance), AVG(rmse), AVG(mse), AVG(mae), AVG(rmsle)

FROM new_leaderboard;

Result:

AVG(mean_residual_deviance)	AVG(rmse)	AVG(mse)	AVG(mae)	AVG(rmsle)
450.963704839661	20.956503525497777	450.963704839661	15.72641749564448	0.12079121899573014

8.Which models of gbm had leaderboard rank above 50 FOR 2nd run limiting to 10?

SELECT gbm_hyperparameters.Leaderboard_rank, gbm_hyperparameters.Run_id, Algorithms.model_id

FROM Algorithms

INNER JOIN gbm_hyperparameters ON gbm_hyperparameters.Algorithm_id=Algorithms.Algorithm_id

WHERE gbm_hyperparameters.Run_id = 2 AND gbm_hyperparameters.Leaderboard_rank <50;

Result:

Leaderboard_rank	Run_id	model_id
1	2	GBM_grid_1_AutoML_20190419_181340_model...
3	2	GBM_2_AutoML_20190419_181340
13	2	GBM_5_AutoML_20190419_181340
14	2	GBM_5_AutoML_20190419_175717
15	2	GBM_4_AutoML_20190419_175717
16	2	GBM_4_AutoML_20190419_181340
17	2	GBM_grid_1_AutoML_20190419_181340_model...
18	2	GBM_grid_1_AutoML_20190419_175717_model...
19	2	GBM_grid_1_AutoML_20190419_175717_model...
20	2	GBM_grid_1_AutoML_20190419_181340_model...

9. Display the ranks of a leaderboard of all models for DRF hyperparameter for the 5th run?

SELECT Run_id, Algorithm_id, model_name, Leaderboard_rank

FROM deeplearning_hyperparameters

WHERE Run_id = 5;

Result:

Run_id	Algorithm_id	model_name	Leaderboard_rank
5	20048	DeepLearning_1_AutoML_20190419_185907	244
5	20049	DeepLearning_1_AutoML_20190419_181340	245
5	20050	DeepLearning_grid_1_AutoML_20190419_1919...	296
5	20051	DeepLearning_grid_1_AutoML_20190419_1919...	297
5	20052	DeepLearning_grid_1_AutoML_20190419_1951...	306
5	20053	DeepLearning_grid_1_AutoML_20190419_1919...	314
5	20054	DeepLearning_grid_1_AutoML_20190419_1951...	315
5	20055	DeepLearning_grid_1_AutoML_20190419_1919...	317
5	20056	DeepLearning_grid_1_AutoML_20190419_1951...	322
5	20057	DeepLearning_grid_1_AutoML_20190419_1951...	323
5	20058	DeepLearning_grid_1_AutoML_20190419_1757...	350
5	20059	DeepLearning_grid_1_AutoML_20190419_1813...	351
5	20060	DeepLearning_1_AutoML_20190419_175717	249

10. Find the count of all the models for the first run of GLM?

SELECT count(*) model_id FROM hyperparameter_db.dl_allparams

Result:

model_id
50

11. What are the top three models for 2nd run of GBM models?

SELECT metadata.run_time,

metadata.Run_id,

gbm_hyperparameters.Algorithm_id,

gbm_hyperparameters.Leaderboard_rank,

Algorithms.model_id

FROM metadata

INNER JOIN gbm_hyperparameters on gbm_hyperparameters.Run_id=metadata.Run_id

INNER JOIN Algorithms on Algorithms.Algorithm_id=gbm_hyperparameters.Algorithm_id

WHERE metadata.run_time=777 AND gbm_hyperparameters.Leaderboard_rank < 4

Result:

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

run_time	Run_id	Algorithm_id	Leaderboard_rank	model_id
777	2	10037	1	GBM_grid_1_AutoML_20190419_181340_model...
777	2	10038	3	GBM_2_AutoML_20190419_181340

Result 7

12. Which variable showed highest importance?

SELECT distinct Variable, relative_importance

FROM variableimportance_new

ORDER BY relative_importance DESC LIMIT 2;

Result:

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

Variable	relative_importance
incidenceRate	2199695.25
PctBachDeg25_Over	2141772.75

variableimportance_new 8

13. What should I set the learning rate for GBM?

SELECT Run_id,model_id,Leaderboard_rank,learn_rate

FROM gbm_hyperparameters

WHERE Run_id=2

ORDER BY Leaderboard_rank limit 4;

Result:

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

Run_id	model_id	Leaderboard_rank	learn_rate
2	GBM_grid_1_AutoML_20190419_181340_model_48	1	0.1
2	GBM_2_AutoML_20190419_181340	3	0.1
2	GBM_2_AutoML_20190419_175717	4	0.1
2	GBM_grid_1_AutoML_20190419_181340_model_60	5	0.1

14. Which model performed the Best for all Runs.

SELECT Distinct new_leaderboard.Algorithm_id,rmse,model_id

FROM hyperparameter_db.new_leaderboard

inner join algorithms on algorithms.Algorithm_id=new_leaderboard.Algorithm_id

order by rmse, Run_id

Result:

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

Run_id	Algorithm_id	rmse	model_id
2	10037	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
2	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
2	10331	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
2	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10037	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10331	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48

Result 13 ×

Run_id	Algorithm_id	rmse	model_id
4	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
4	10331	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
4	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
4	10141	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
4	10330	15.58619347	GBM_grid_1_AutoML_20190419_191951_model_48
4	10442	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
4	10630	15.58619347	GBM_grid_1_AutoML_20190419_191951_model_48
4	10853	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48

Result 13

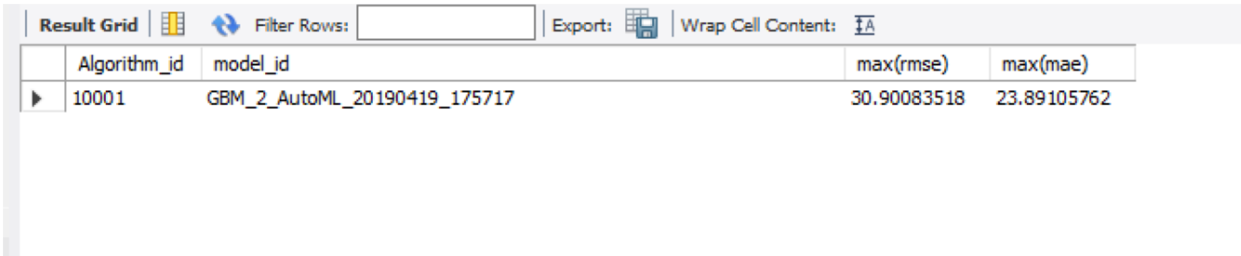
Run_id	Algorithm_id	rmse	model_id
5	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
5	10630	15.58619347	GBM_grid_1_AutoML_20190419_191951_model_48
5	10631	15.58619347	GBM_grid_1_AutoML_20190419_195145_model_48
5	10141	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
5	10442	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
5	10853	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
5	10037	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
5	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48

Result 13

15.Find the Highest RMSE and MAE value for GBM model(This gives the worst model performance metrics)

```
SELECT new_leaderboard.Algorithm_id,
model_id,
max(rmse),
max(mae) FROM hyperparameter_db.new_leaderboard
inner join algorithms on algorithms.Algorithm_id=new_leaderboard.Algorithm_id
```

Result:



Algorithm_id	model_id	max(rmse)	max(mae)
10001	GBM_2_AutoML_20190419_175717	30.90083518	23.89105762

VIEWS

1.Find the average of all the evaluation matrices from leaderboard?

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `view_1` AS
    SELECT
        AVG(`new_leaderboard`.`mean_residual_deviance`) AS
`AVG(mean_residual_deviance)`,
        AVG(`new_leaderboard`.`rmse`) AS `AVG(rmse)`,
        AVG(`new_leaderboard`.`mse`) AS `AVG(mse)`,
        AVG(`new_leaderboard`.`mae`) AS `AVG(mae)`,
        AVG(`new_leaderboard`.`rmsle`) AS `AVG(rmsle)`
    FROM
        `new_leaderboard`
```

Result:

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

AVG(mean_residual_deviance)	AVG(rmse)	AVG(mse)	AVG(mae)	AVG(rmsle)
450.963704839661	20.956503525497777	450.963704839661	15.72641749564448	0.12079121899573014

2.What are the top three models for 2nd run of GBM models?

```
CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `root`@`localhost`
   SQL SECURITY DEFINER
VIEW `view_2` AS
   SELECT
      `metadata`.`run_time` AS `run_time`,
      `metadata`.`Run_id` AS `Run_id`,
      `gbm_hyperparameters`.`Algorithm_id` AS `Algorithm_id`,
      `gbm_hyperparameters`.`Leaderboard_rank` AS `Leaderboard_rank`,
      `algorithms`.`model_id` AS `model_id`
   FROM
      ((`metadata`
      JOIN `gbm_hyperparameters` ON ((`gbm_hyperparameters`.`Run_id` =
`metadata`.`Run_id`)))
      JOIN `algorithms` ON ((`algorithms`.`Algorithm_id` =
`gbm_hyperparameters`.`Algorithm_id`)))
   WHERE
      ((`metadata`.`run_time` = 777)
         AND (`gbm_hyperparameters`.`Leaderboard_rank` < 4))
```

Result:

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

run_time	Run_id	Algorithm_id	Leaderboard_rank	model_id
777	2	10037	1	GBM_grid_1_AutoML_20190419_181340_model_48
777	2	10038	3	GBM_2_AutoML_20190419_181340

3.Find the count of all the models for the first run of GLM?

```
CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `root`@`localhost`
   SQL SECURITY DEFINER
VIEW `view_3` AS
   SELECT
      COUNT(0) AS `model_id`
   FROM
```

```
`dl_allparams`
```

Result:

model_id
50

4. Display the ranks of a leaderboard of all models for DRF hyperparameter for the 5th run?

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `view_4` AS
    SELECT
        `deeplearning_hyperparameters`.`Run_id` AS `Run_id`,
        `deeplearning_hyperparameters`.`Algorithm_id` AS `Algorithm_id`,
        `deeplearning_hyperparameters`.`Model_name` AS `model_name`,
        `deeplearning_hyperparameters`.`Leaderboard_rank` AS `Leaderboard_rank`
    FROM
        `deeplearning_hyperparameters`
    WHERE
        (`deeplearning_hyperparameters`.`Run_id` = 5)
```

Result:

Run_id	Algorithm_id	model_name	Leaderboard_rank
5	20048	DeepLearning_1_AutoML_20190419_185907	244
5	20049	DeepLearning_1_AutoML_20190419_181340	245
5	20050	DeepLearning_grid_1_AutoML_20190419_1919...	296
5	20051	DeepLearning_grid_1_AutoML_20190419_1919...	297
5	20052	DeepLearning_grid_1_AutoML_20190419_1951...	306
5	20053	DeepLearning_grid_1_AutoML_20190419_1919...	314
5	20054	DeepLearning_grid_1_AutoML_20190419_1951...	315
5	20055	DeepLearning_grid_1_AutoML_20190419_1919...	317

view_4 1

5. Which models of gbm had leaderboard rank above 50 FOR 2nd run limiting to 10?

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `view_5` AS
    SELECT
        `gbm_hyperparameters`.`Leaderboard_rank` AS `Leaderboard_rank`,
        `gbm_hyperparameters`.`Run_id` AS `Run_id`,
        `algorithms`.`model_id` AS `model_id`
    FROM
        (`algorithms`
        JOIN `gbm_hyperparameters` ON ((`gbm_hyperparameters`.`Algorithm_id` =
`algorithms`.`Algorithm_id`)))
    WHERE
        ((`gbm_hyperparameters`.`Run_id` = 2)
            AND (`gbm_hyperparameters`.`Leaderboard_rank` < 50))
```

Result:

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

Leaderboard_rank	Run_id	model_id
1	2	GBM_grid_1_AutoML_20190419_181340_model...
3	2	GBM_2_AutoML_20190419_181340
13	2	GBM_5_AutoML_20190419_181340
14	2	GBM_5_AutoML_20190419_175717
15	2	GBM_4_AutoML_20190419_175717
16	2	GBM_4_AutoML_20190419_181340
17	2	GBM_grid_1_AutoML_20190419_181340_model...
18	2	GBM_grid_1_AutoML_20190419_175717_model...

view_5 1 ×

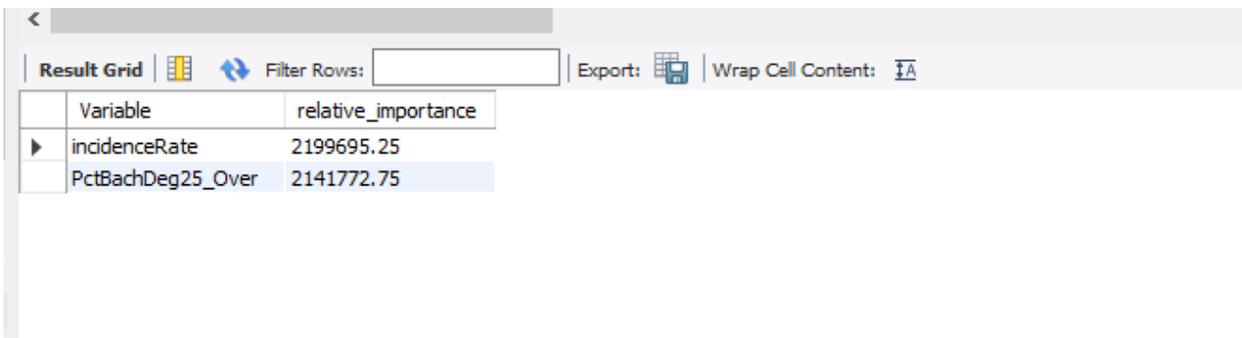
Output

6. Which variable showed highest importance?

```
CREATE

    ALGORITHM = UNDEFINED

    DEFINER = `root`@`localhost`

    SQL SECURITY DEFINER

VIEW `view_6` AS

    SELECT DISTINCT
```

```
    `variableimportance_new`.`Variable` AS `Variable`,
    `variableimportance_new`.`relative_importance` AS `relative_importance`
FROM
    `variableimportance_new`
ORDER BY `variableimportance_new`.`relative_importance` DESC
LIMIT 2
```

Result:



Variable	relative_importance
incidenceRate	2199695.25
PctBachDeg25_Over	2141772.75

FUNCTIONS

1.Based on RMSE, it classifies the models as best, moderate or worst.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `F_1`(rmse double) RETURNS text CHARSET utf8mb4
    DETERMINISTIC
BEGIN
declare result text;

if rmse < 18  then
set result="Best";
elseif rmse < 19 then
set result="Moderate";
else
set result="Worst";
end if;

RETURN (result);
END
```

Result:

Run_id	Algorithm_id	rmse	Result	model_id
1	10001	17.21370358	Best	GBM_2_AutoML_20190419_175717
1	10052	17.21370358	Best	GBM_2_AutoML_20190419_175717
1	10253	17.21370358	Best	GBM_2_AutoML_20190419_175717
1	10597	17.21370358	Best	GBM_2_AutoML_20190419_175717
1	10643	17.21370358	Best	GBM_2_AutoML_20190419_175717
1	10035	17.96372225	Best	GBM_grid_1_AutoML_20190419_175717_model_18
1	10053	17.96372225	Best	GBM_grid_1_AutoML_20190419_175717_model_18
1	10258	17.96372225	Best	GBM_grid_1_AutoML_20190419_175717_model_18
1	10577	17.96372225	Best	GBM_grid_1_AutoML_20190419_175717_model_18
1	11025	17.96372225	Best	GBM_grid_1_AutoML_20190419_175717_model_18
1	10036	18.25551105	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10057	18.25551105	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10266	18.25551105	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10589	18.25551105	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	11038	18.25551105	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10003	18.27269813	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10059	18.27269813	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10269	18.27269813	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10593	18.27269813	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	11043	18.27269813	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10004	18.39475747	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10061	18.39475747	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10272	18.39475747	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10596	18.39475747	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	11050	18.39475747	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10005	18.58062789	Moderate	GBM_grid_1_AutoML_20190419_175717_model_7
1	10065	18.58062789	Moderate	GBM_grid_1_AutoML_20190419_175717_model_7
1	10279	18.58062789	Moderate	GBM_grid_1_AutoML_20190419_175717_model_7
1	10607	18.58062789	Moderate	GBM_grid_1_AutoML_20190419_175717_model_7
1	10639	18.58062789	Moderate	GBM_grid_1_AutoML_20190419_175717_model_7
1	10006	18.6690481	Moderate	GBM_grid_1_AutoML_20190419_175717_model_23
1	10068	18.6690481	Moderate	GBM_grid_1_AutoML_20190419_175717_model_23
1	10281	18.6690481	Moderate	GBM_grid_1_AutoML_20190419_175717_model_23
1	10611	18.6690481	Moderate	GBM_grid_1_AutoML_20190419_175717_model_23
1	10642	18.6690481	Moderate	GBM_grid_1_AutoML_20190419_175717_model_23
1	10007	18.71482053	Moderate	GBM_grid_1_AutoML_20190419_175717_model_11
1	10069	18.71482053	Moderate	GBM_grid_1_AutoML_20190419_175717_model_11
1	10287	18.71482053	Moderate	GBM_grid_1_AutoML_20190419_175717_model_11
1	10618	18.71482053	Moderate	GBM_grid_1_AutoML_20190419_175717_model_11

2.Based on MAE, it classifies the models as best, moderate or worst.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `F_2`(mae double) RETURNS text CHARSET utf8mb4
    DETERMINISTIC
BEGIN
declare result text;

if mae > 18  then
set result="Worst";
else
set result="Best ";
end if;

RETURN (result);
END
```

Result:

Run_id	Algorithm_id	mae	f_2(mae)	model_id
1	10034	13.05318094	Best	GBM_grid_1_AutoML_20190419_175717_model_22
1	10048	13.05318094	Best	GBM_grid_1_AutoML_20190419_175717_model_22
1	10249	13.05318094	Best	GBM_grid_1_AutoML_20190419_175717_model_22
1	10563	13.05318094	Best	GBM_grid_1_AutoML_20190419_175717_model_22
1	11006	13.05318094	Best	GBM_grid_1_AutoML_20190419_175717_model_22
2	10037	11.34569884	Best	GBM_grid_1_AutoML_20190419_181340_model_48
2	10140	11.34569884	Best	GBM_grid_1_AutoML_20190419_181340_model_48
2	10331	11.34569884	Best	GBM_grid_1_AutoML_20190419_181340_model_48
2	10742	11.34569884	Best	GBM_grid_1_AutoML_20190419_181340_model_48
2	10118	12.9468733	Best	GBM_3_AutoML_20190419_181340
2	10186	12.9468733	Best	GBM_3_AutoML_20190419_181340
2	10476	12.9468733	Best	GBM_3_AutoML_20190419_181340
2	10887	12.9468733	Best	GBM_3_AutoML_20190419_181340
2	10122	20.96519497	Worst	GBM_grid_1_AutoML_20190419_181340_model_17
2	10203	20.96519497	Worst	GBM_grid_1_AutoML_20190419_181340_model_17
2	10496	20.96519497	Worst	GBM_grid_1_AutoML_20190419_181340_model_17
2	10917	20.96519497	Worst	GBM_grid_1_AutoML_20190419_181340_model_17
2	10022	20.96519497	Worst	GBM_grid_1_AutoML_20190419_175717_model_17
2	10123	20.96519497	Worst	GBM_grid_1_AutoML_20190419_175717_model_17
2	10205	20.96519497	Worst	GBM_grid_1_AutoML_20190419_175717_model_17
2	10500	20.96519497	Worst	GBM_grid_1_AutoML_20190419_175717_model_17
2	10921	20.96519497	Worst	GBM_grid_1_AutoML_20190419_175717_model_17
2	10023	20.97918555	Worst	GBM_grid_1_AutoML_20190419_175717_model_25
2	10124	20.97918555	Worst	GBM_grid_1_AutoML_20190419_175717_model_25
2	10209	20.97918555	Worst	GBM_grid_1_AutoML_20190419_175717_model_25

3.Based on MSE, it classifies the models as best, moderate or worst.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `F_3`(mse double) RETURNS text CHARSET utf8mb4
    DETERMINISTIC
BEGIN
declare result text;

if mse < 300  then
set result="Best";
elseif mse <500 then
set result="Moderate";
else
set result="Worst";
end if;

RETURN (result);
END
```

Result:

Run_id	Algorithm_id	mse	f_3(mse)	model_id
1	10001	296.3115908	Best	GBM_2_AutoML_20190419_175717
1	10052	296.3115908	Best	GBM_2_AutoML_20190419_175717
1	10253	296.3115908	Best	GBM_2_AutoML_20190419_175717
1	10597	296.3115908	Best	GBM_2_AutoML_20190419_175717
1	10643	296.3115908	Best	GBM_2_AutoML_20190419_175717
1	10035	322.6953172	Moderate	GBM_grid_1_AutoML_20190419_175717_model_18
1	10053	322.6953172	Moderate	GBM_grid_1_AutoML_20190419_175717_model_18
1	10258	322.6953172	Moderate	GBM_grid_1_AutoML_20190419_175717_model_18
1	10577	322.6953172	Moderate	GBM_grid_1_AutoML_20190419_175717_model_18
1	11025	322.6953172	Moderate	GBM_grid_1_AutoML_20190419_175717_model_18
1	10036	333.2636839	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10057	333.2636839	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10266	333.2636839	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10589	333.2636839	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	11038	333.2636839	Moderate	GBM_grid_1_AutoML_20190419_175717_model_16
1	10003	333.8914968	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10059	333.8914968	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10269	333.8914968	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10593	333.8914968	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	11043	333.8914968	Moderate	GBM_grid_1_AutoML_20190419_175717_model_20
1	10004	338.3671025	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10061	338.3671025	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10272	338.3671025	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10596	338.3671025	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	11050	338.3671025	Moderate	GBM_grid_1_AutoML_20190419_175717_model_1
1	10005	345.2397326	Moderate	GBM_grid_1_AutoML_20190419_175717_model_7

4.Show the importance of variables based on relative importance.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `F_4`(relative_importance double) RETURNS text
CHARSET utf8mb4
    DETERMINISTIC
BEGIN
declare result text;

if relative_importance < 10000  then
set result="not important";
elseif relative_importance >10000 and  relative_importance< 50000 then
set result="Moderately important ";
else
set result="very important";
end if;

RETURN (result);
END
```

Result:

Run_id	Var_id	Variable	relative_importance	F_4(relative_importance)
1	18	PctBachDeg18_24	137780.2969	very important
1	19	PctNoHS18_24	132300.6406	very important
1	20	PctAsian	119957.7188	very important
1	21	PctEmpPrivCoverage	103336.0781	very important
1	22	PctPrivateCoverageAlone	92164.375	very important
1	23	binnedInc	83327.45313	very important
1	24	PctHS25_Over	620397.125	very important
1	25	studyPerCap	43831.15625	Moderately important
1	26	isPoor	25948.84766	Moderately important
1	27	PctSomeCol18_24	22120.4043	Moderately important
1	28	MedianAge_1	9289.574219	not important
1	29	MedianAge	8865.966797	not important
1	30	MedianAge_2	1764.491699	not important
1	31	MedianAge_3	941.8779907	not important
1	32	medIncome	383928.9375	very important
1	33	PctPrivateCoverage	373168.4375	very important
1	34	povertyPercent	369052.875	very important
1	35	PctPublicCoverageAlone	281258.5938	very important
1	36	popEst2015	278202.875	very important
1	37	avgAnnCount	255146.9531	very important
2	38	incidenceRate	2199695.25	very important
2	39	PctBachDeg25_Over	2035948.625	very important
2	40	AvgHouseholdSize	191581.7188	very important
2	41	PctOtherRace	184559.1094	very important
2	42	PctBlack	183525.4688	very important
2	43	PctUnemployed16_Over	181384.5	very important

5. Show the importance of variables based on scaled importance.

```
 CREATE DEFINER=`root`@`localhost` FUNCTION `F_5`(scaled_importance double) RETURNS text
CHARSET utf8mb4
   DETERMINISTIC
BEGIN
declare result text;

if scaled_importance <0.02  then
set result="not important";
elseif scaled_importance >0.02 and  scaled_importance<0.05  then
set result="Moderately important ";
else
set result="very important";
end if;

RETURN (result);
END
```

Result:

Run_id	Var_id	Variable	scaled_importance	F_5(scaled_importance)
1	16	BirthRate	0.067827669	very important
1	17	PctPublicCoverage	0.066433198	very important
1	18	PctBachDeg18_24	0.064330026	very important
1	19	PctNoHS18_24	0.061771559	very important
1	20	PctAsian	0.056008612	very important
1	21	PctEmpPrivCoverage	0.048247919	Moderately important
1	22	PctPrivateCoverageAlone	0.043031818	Moderately important
1	23	binnedInc	0.038905833	Moderately important
1	24	PctHS25_Over	0.289665243	very important
1	25	studyPerCap	0.020464896	Moderately important
1	26	isPoor	0.012115593	not important
1	27	PctSomeCol18_24	0.010328082	not important
1	28	MedianAge_1	0.00433733	not important
1	29	MedianAge	0.004139546	not important
1	30	MedianAge_2	0.000823846	not important
1	31	MedianAge_3	0.000439766	not important
1	32	medIncome	0.179257551	very important
1	33	PctPrivateCoverage	0.174233442	very important
1	34	povertyPercent	0.172311873	very important
1	35	PctPublicCoverageAlone	0.131320465	very important
1	36	popEst2015	0.129893741	very important
1	37	avgAnnCount	0.119128863	very important
2	38	incidenceRate	1	very important
2	39	PctBachDeg25_Over	0.925559404	very important
2	40	AvgHouseholdSize	0.087094664	very important
2	41	PctOtherRace	0.083902127	very important

6. Finding the accuracy of the model based on the number of trees.

```
 CREATE DEFINER=`root`@`localhost` FUNCTION `F_6`(ntrees int) RETURNS text CHARSET utf8mb4
    DETERMINISTIC
BEGIN
declare result text;

if ntrees >100  then
set result="Very accurate";

else
set result="Moderately accurate";
end if;

RETURN (result);
END
```

Result:

Algorithm_id	Run_id	model_id	Leaderboard_rank	ntrees	F_6(ntrees)
10001	1	GBM_2_AutoML_20190419_175717	2	78	Moderately accurate
10002	1	GBM_grid_1_AutoML_20190419_175717_model_19	3	97	Moderately accurate
10003	1	GBM_grid_1_AutoML_20190419_175717_model_20	13	108	Very accurate
10004	1	GBM_grid_1_AutoML_20190419_175717_model_1	14	109	Very accurate
10005	1	GBM_grid_1_AutoML_20190419_175717_model_7	15	60	Moderately accurate
10006	1	GBM_grid_1_AutoML_20190419_175717_model_23	16	73	Moderately accurate
10007	1	GBM_grid_1_AutoML_20190419_175717_model_11	17	63	Moderately accurate
10008	1	GBM_grid_1_AutoML_20190419_175717_model_9	18	69	Moderately accurate
10009	1	GBM_grid_1_AutoML_20190419_175717_model_13	19	192	Very accurate
10010	1	GBM_grid_1_AutoML_20190419_175717_model_6	20	48	Moderately accurate
10011	1	GBM_grid_1_AutoML_20190419_175717_model_12	21	194	Very accurate
10012	1	GBM_grid_1_AutoML_20190419_175717_model_30	25	207	Very accurate
10013	1	GBM_1_AutoML_20190419_175717	5	78	Moderately accurate
10014	1	GBM_grid_1_AutoML_20190419_175717_model_4	26	216	Very accurate
10015	1	GBM_grid_1_AutoML_20190419_175717_model_27	28	204	Very accurate
10016	1	GBM_grid_1_AutoML_20190419_175717_model_10	29	213	Very accurate
10017	1	GBM_grid_1_AutoML_20190419_175717_model_24	30	230	Very accurate
10018	1	GBM_grid_1_AutoML_20190419_175717_model_28	31	256	Very accurate
10019	1	GBM_grid_1_AutoML_20190419_175717_model_5	33	230	Very accurate
10020	1	GBM_grid_1_AutoML_20190419_175717_model_26	34	242	Very accurate
10021	1	GBM_grid_1_AutoML_20190419_175717_model_31	38	14	Moderately accurate
10022	1	GBM_grid_1_AutoML_20190419_175717_model_17	39	30	Moderately accurate
10023	1	GBM_grid_1_AutoML_20190419_175717_model_25	40	30	Moderately accurate
10024	1	GBM_3_AutoML_20190419_175717	6	78	Moderately accurate
10025	1	GBM_grid_1_AutoML_20190419_175717_model_15	41	30	Moderately accurate
10026	1	GBM_grid_1_AutoML_20190419_175717_model_14	42	30	Moderately accurate

ANALYSIS

The analysis for our dataset revealed the following insights:

- **Best algorithm comparable for all the runs**

Comparing the leaderboard for all the runs gives many interesting revelations.

SELECT mod_best,Run_id,mod_best_algo FROM hyperparameter_db.metadata;

Result:

mod_best	Run_id	mod_best_algo
StackedEnsemble_AllModels_AutoML_20190419_175717	1	stackedensemble
GBM_grid_1_AutoML_20190419_181340_model_48	2	gbm
GBM_grid_1_AutoML_20190419_181340_model_48	3	gbm
GBM_grid_1_AutoML_20190419_191951_model_48	4	gbm
GBM_grid_1_AutoML_20190419_191951_model_48	5	gbm
NULL	NULL	NULL

- **Worst algorithm comparable for all the runs**

As described before the model which gives the worst values i.e. very high values of rmse and mae, hence very less accuracy in predicting the mortality rate.

SELECT new_leaderboard.Algorithm_id,

model_id,

max(rmse),

max(mae) FROM hyperparameter_db.new_leaderboard

inner join algorithms on algorithms.Algorithm_id=new_leaderboard.Algorithm_id

Result:

Algorithm_id	model_id	max(rmse)	max(mae)
10001	GBM_2_AutoML_20190419_175717	30.90083518	23.89105762

- ## Most Important Variable

For finding out which variable was the most important we compared all the 35 predictors used. The relative importance of all these variables were ordered by a SQL query for all the runs. The insights uncovered were:

Select Run_id, Var_id, Variable, relative_importance

from variableimportance_new

where Variable like "%incidence%";

Result:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Run_id	Var_id	Variable	relative_importance
1	2	incidenceRate	1867370
2	38	incidenceRate	2199695.25
3	75	incidenceRate	2199695.25
4	112	incidenceRate	2199695.25
5	149	incidenceRate	2199695.25
NULL	NULL	NULL	NULL

incidence_Rate: The incidence rates of cancer as Mean per capita (100,000) cancer diagnoses for the years 2010-2016.

This shows that the model has been trained in accordance with the incidence rates of cancer which is essential for determining the mortality rate of cancer.

- ## Choosing the best hyperparameter

The best hyperparameters for a model will be the ones which have the least value of rmse i.e. root mean squared error. So by checking the leaderboard generated for all the runs we discovered the following:

SELECT Distinct new_leaderboard.Algorithm_id,rmse,model_id

FROM hyperparameter_db.new_leaderboard

inner join algorithms on algorithms.Algorithm_id=new_leaderboard.Algorithm_id

order by rmse, Run_id

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

Run_id	Algorithm_id	rmse	model_id
2	10037	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
2	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
2	10331	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
2	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10037	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10331	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
3	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48

Result 13

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

Run_id	Algorithm_id	rmse	model_id
4	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
4	10331	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
4	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
4	10141	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
4	10330	15.58619347	GBM_grid_1_AutoML_20190419_191951_model_48
4	10442	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
4	10630	15.58619347	GBM_grid_1_AutoML_20190419_191951_model_48
4	10853	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48

Result 13

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

Run_id	Algorithm_id	rmse	model_id
5	10742	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
5	10630	15.58619347	GBM_grid_1_AutoML_20190419_191951_model_48
5	10631	15.58619347	GBM_grid_1_AutoML_20190419_195145_model_48
5	10141	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
5	10442	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
5	10853	15.58619347	GBM_grid_1_AutoML_20190419_185907_model_48
5	10037	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48
5	10140	15.58619347	GBM_grid_1_AutoML_20190419_181340_model_48

Result 13

Run Number	**Model Name**	**RMSE**
1	'GBM_2_AutoML_20190419_175717'	17.21370358
2	'GBM_grid_1_AutoML_20190419_181340_model_48'	15.58619347
3	'GBM_grid_1_AutoML_20190419_181340_model_48'	15.58619347
4	'GBM_grid_1_AutoML_20190419_181340_model_48'	15.58619347
5	'GBM_grid_1_AutoML_20190419_185907_model_69'	17.03479754

CONCLUSION

Concluding we stored and queried the dataset which was used for predicting the cancer mortality rate. The variable which was essential during the model building was "incidenceRate" which gave the number of cancer diagnoses for a period of 6 years. Also, the model was run for 5 times with different run times and each run gave differential outputs. Our insights suggest that GBM (Gradient Boosting Machine) algorithm is the most efficient model with the least RMSE value of 15.58

The hyperparameters of GBM: learn_rate, ntrees, n_folds, max_depth, tweedie_power, distribution, sample_rate are the ones which impacted the performance of the model during each run. The metadata files also gave the best model for each run and also the execution time for that run. Also the defining factor of our database was the presence of leaderboard ranks along with their error percentages which gave concrete insights about the model execution.

CITATIONS AND REFERNCES

[1] Machine Learning Data Science - What is difference between model parameter and hyperparameter? https://www.youtube.com/watch?v=tyDgjKe5C9Y

[2] https://en.wikipedia.org/wiki/Hyperparameter

[3] https://github.com/skunkworksneu/Projects

[4] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/grid-search.html#supported-grid-search-hyperparameters

[5] https://github.com/nikbearbrown/INFO_6210

[6] https://github.com/skunkworksneu/Projects

[7] https://github.com/prabhuSub/Hyperparamter-Samples/tree/master/Hyperparameter_Generated

[8] https://stackoverflow.com/questions/45068309/mysql-error-importing-from-text-tilda-delimited-file

[9] https://www.geeksforgeeks.org/database-normalization-normal-forms/

[10] https://www.w3schools.com/sql/sql_create_index.asp

[11] https://www.w3schools.com/sql/sql_stored_procedures.asp

[12] https://www.geeksforgeeks.org/sql-views/

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

LICENSE