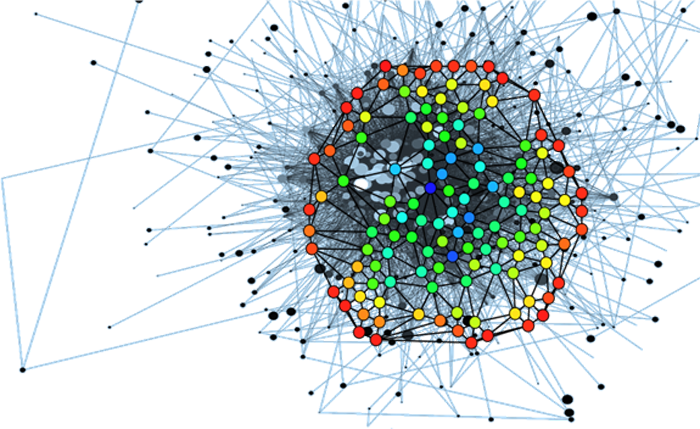
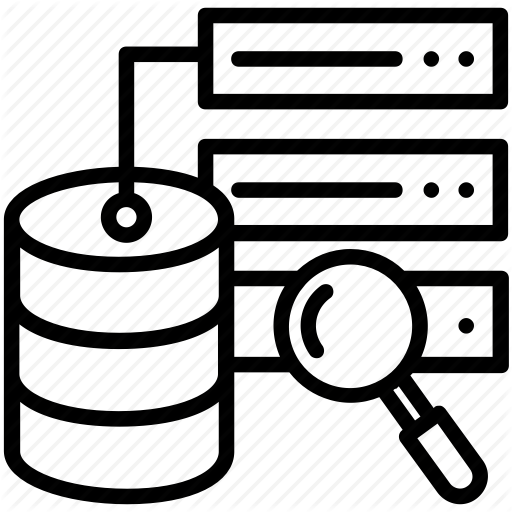
PORTFOLIO

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

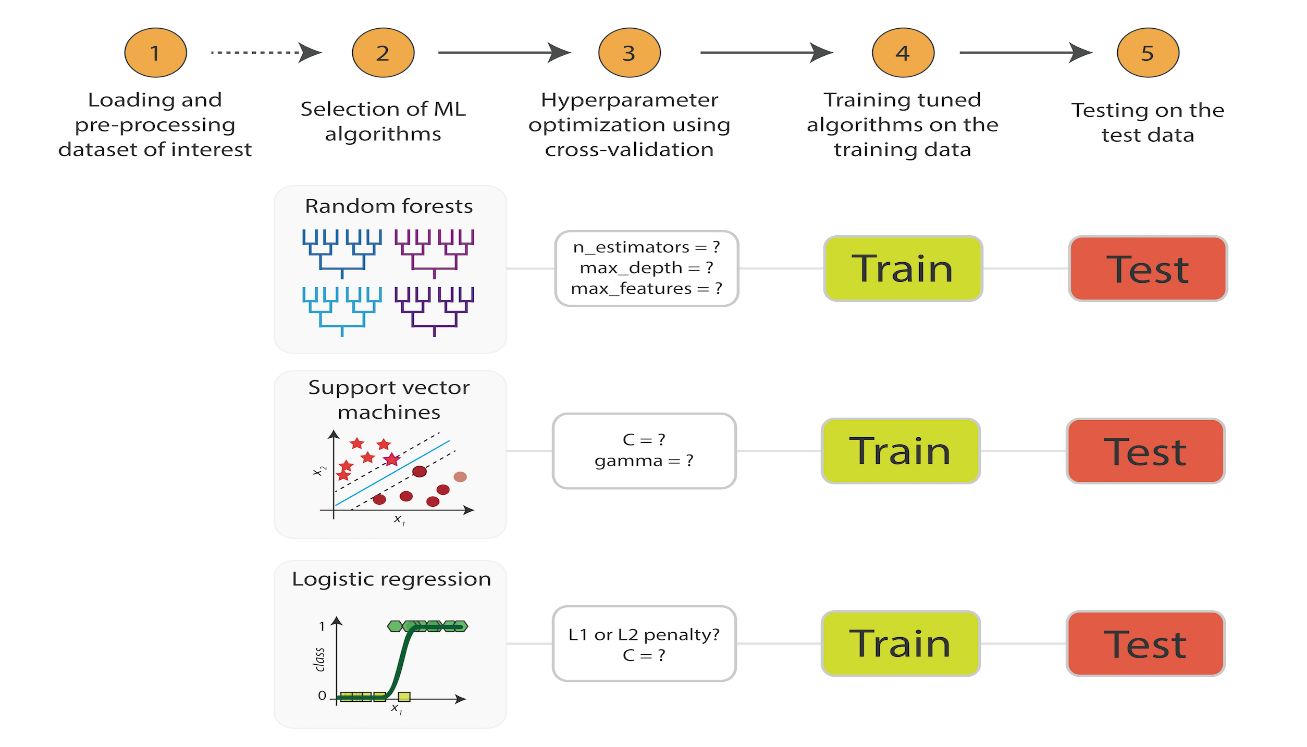
In the above example,

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.

<https://data.world/nrippner/ols-regression-challenge/workspace/file?filename=cancer_reg.csv>



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**

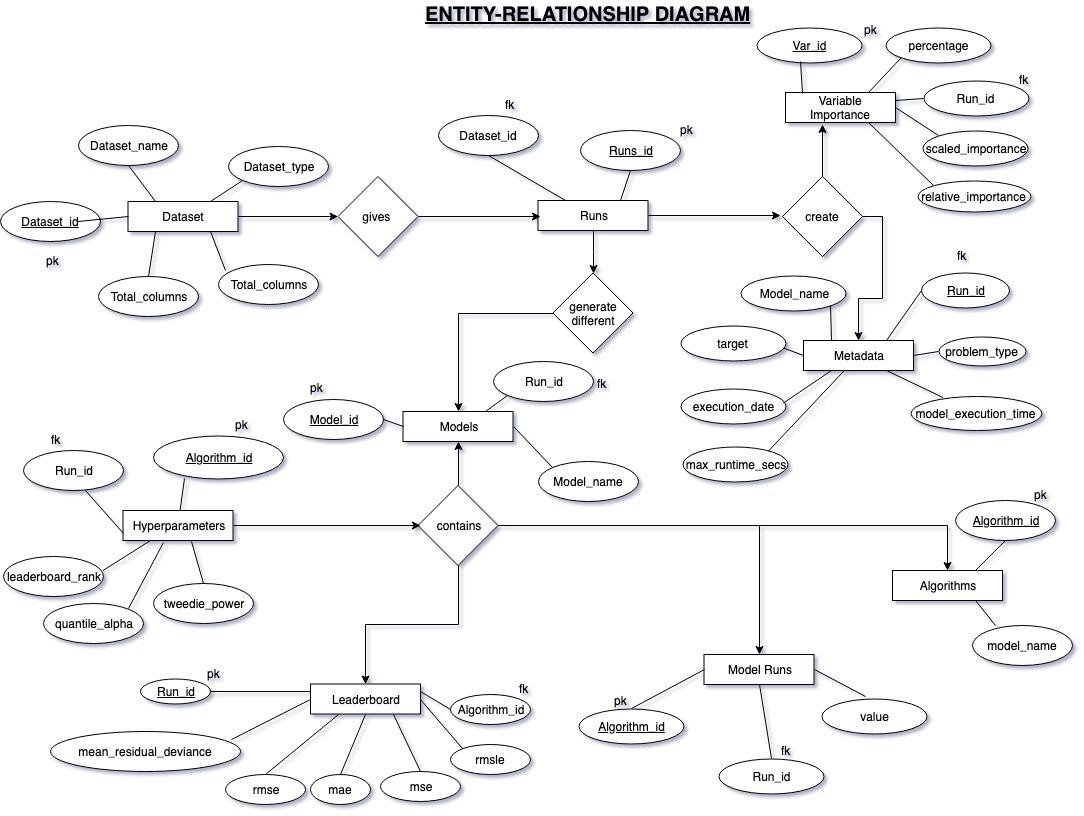
**2. CAb9R3kai-----128**

**3. WdShVGuoh---223**

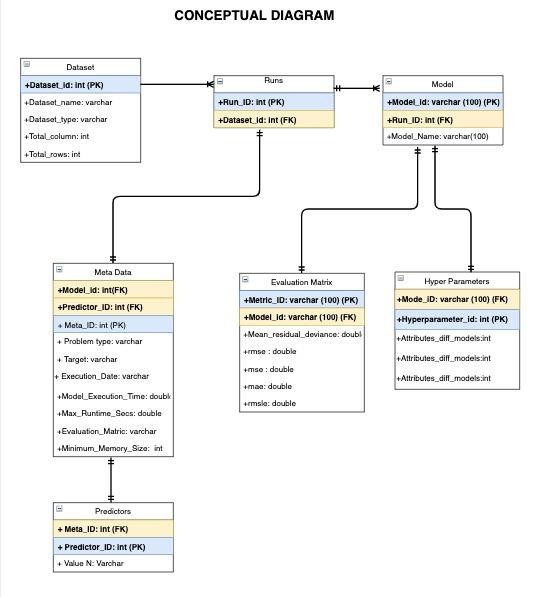
**4. ON7BbTEGe---343**

**5. gCje7dhU4-----503**

**ENTITY-RELATIONSHIP DIAGRAM**



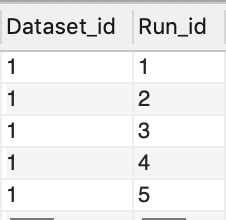
**NORMALIZATION:**

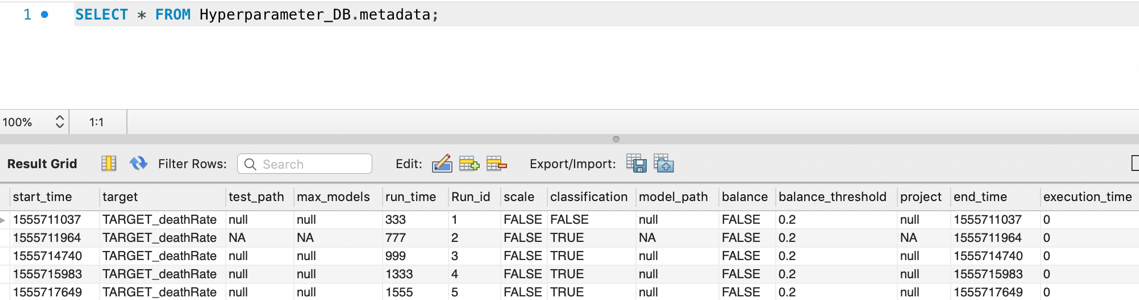


The above diagram was the conceptual schema before normalization.

**For normalization**

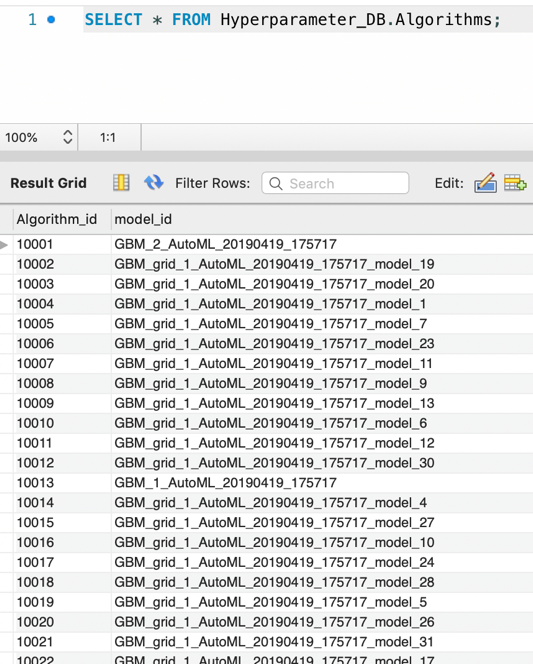
**First Normal Form:** Firstly, we created a bridge table which contains the run\_id and dataset\_id. The primary key is Run\_id and the foreign key is dataset\_id.



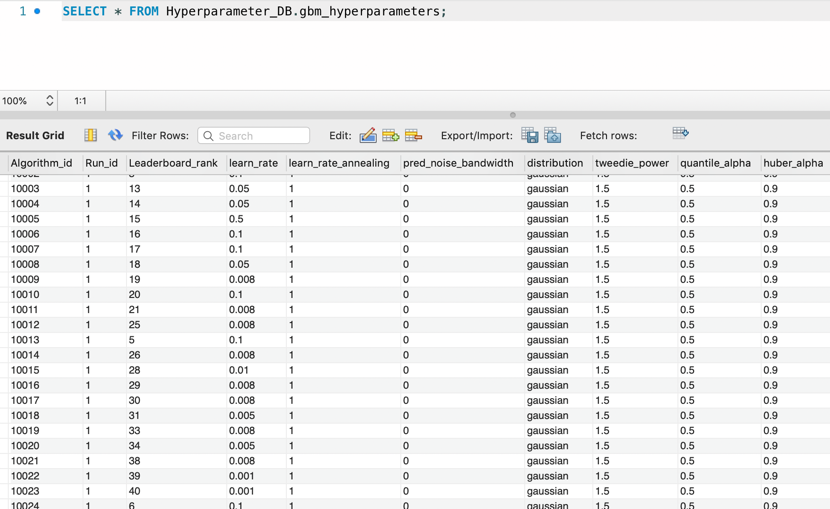


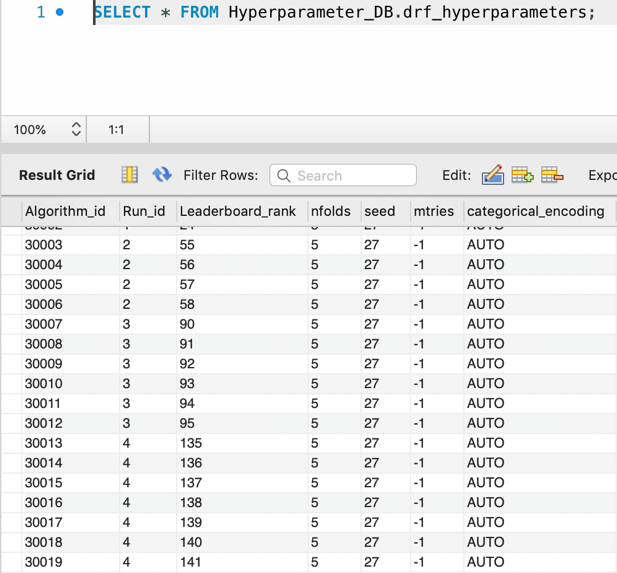
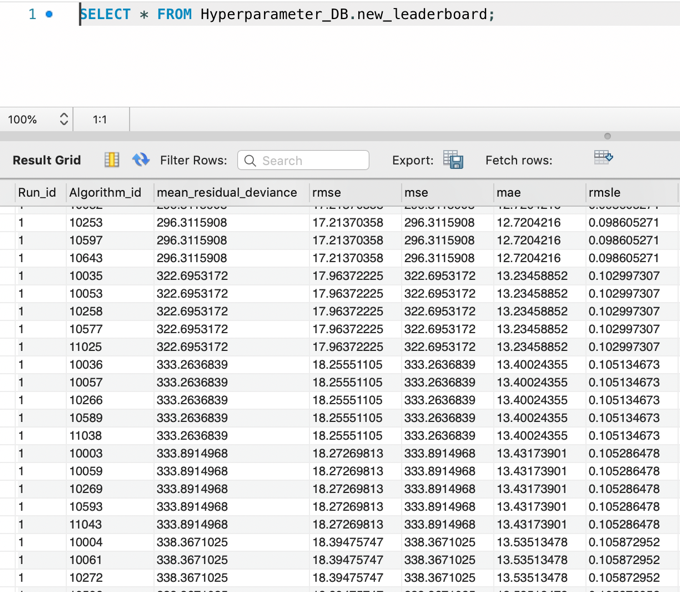
Next, we created separate table for all the hyperparameters and made Algorithm\_id as the primary key.

**Second Normal Form**: In all the hyperparameters tables there was a partial dependency. In order to normalize them we formed a Bridge table Algorithms which contains all the algorithms along with their Id.





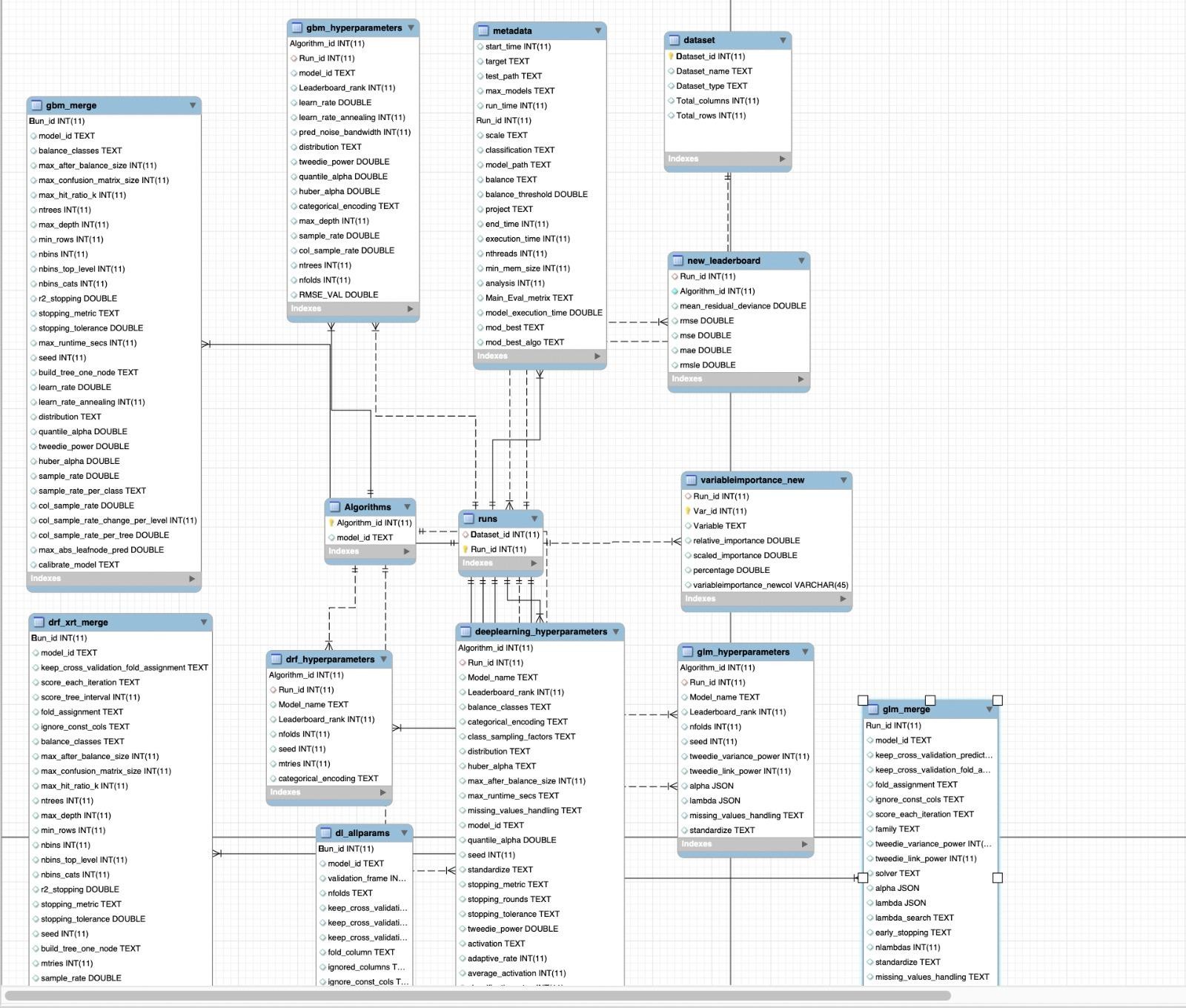


**Third Normal Form:** All requirements of 2NF were met. We have eliminated all fields that do not directly depend on the primary key; that is no transitive dependencies.

The final conceptual schema is shown below:

**CONCEPTUAL SCHEMA**



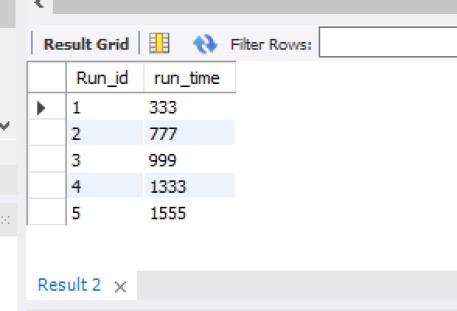
**USE CASES**

**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:**



**2.To find the variable importance for 2nd run**

SELECT Run\_id, Variable, relative\_importance, scaled\_importance, percentage

FROM variableimportance\_new WHERE Run\_id = 2 limit 10; **Result:**



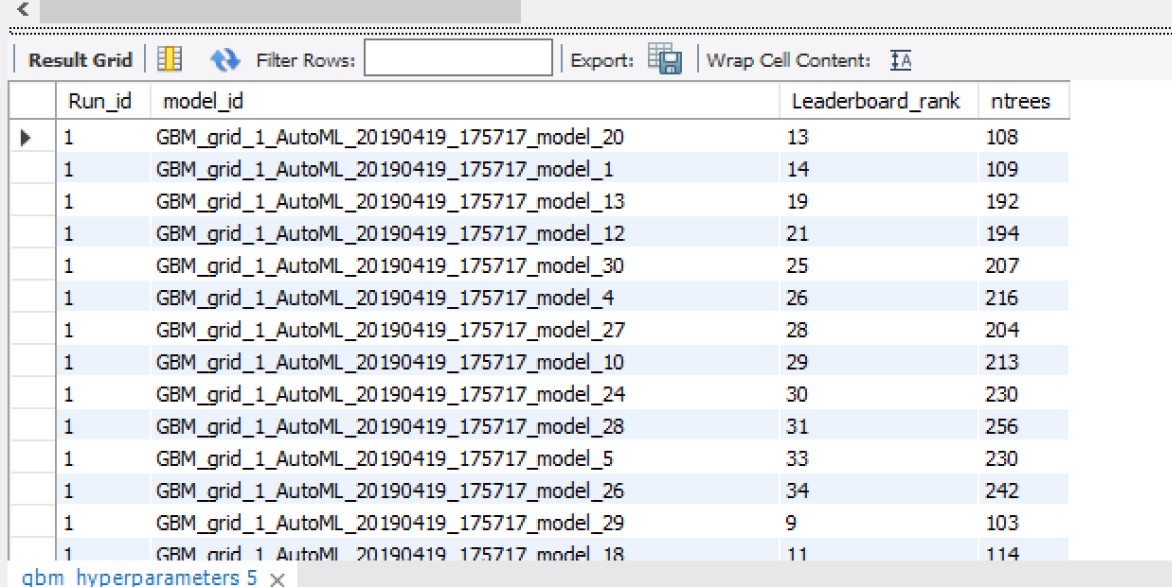
**3.Which models for gbm takes more than 100 ntrees for first 3 runs?**

SELECT Run\_id, model\_id, Leaderboard\_rank, ntrees

FROM gbm\_hyperparameters

WHERE ntrees> 100 AND Run\_id BETWEEN 1 AND 3;

**Result:**



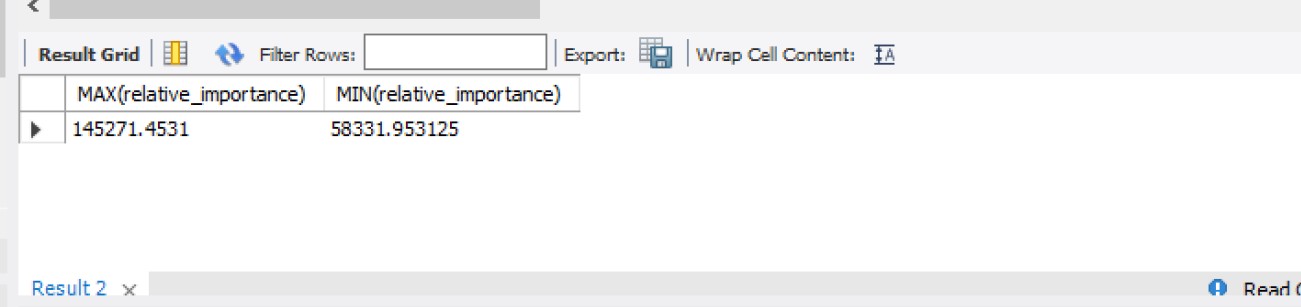
**4.What is the range of relative importance for the variable BirthRate for all the run IDs?**

SELECT MAX(relative\_importance), MIN(relative\_importance)

FROM variableimportance\_new

WHERE Variable = 'BirthRate' AND Run\_id BETWEEN 1 AND 5;

**Result:**

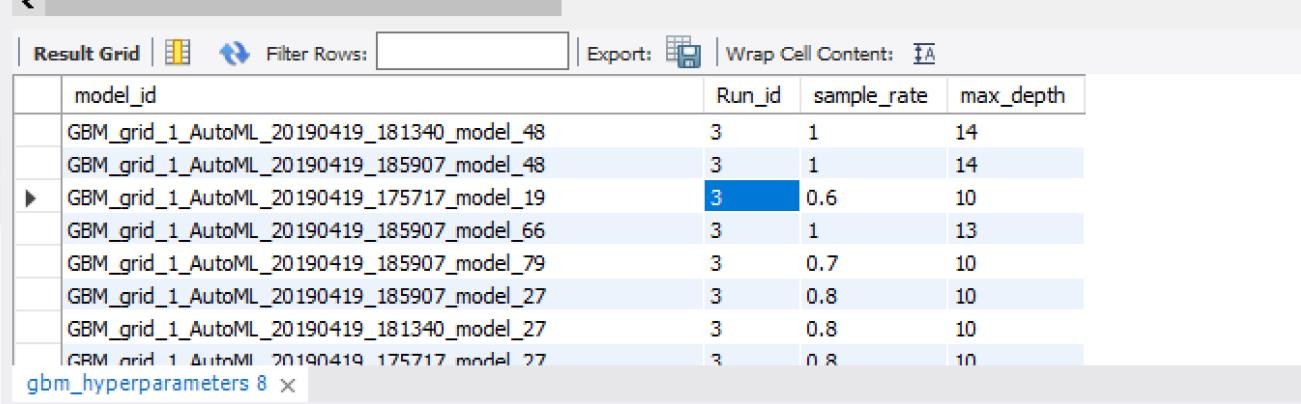


**5.What is sample\_rate and max\_depth for GBM hyperparameter for the 3rd Run?**

SELECT model\_id,Run\_id,sample\_rate, max\_depth FROM hyperparameter\_db.gbm\_hyperparameters

WHERE Run\_id = 3;

**Result:**

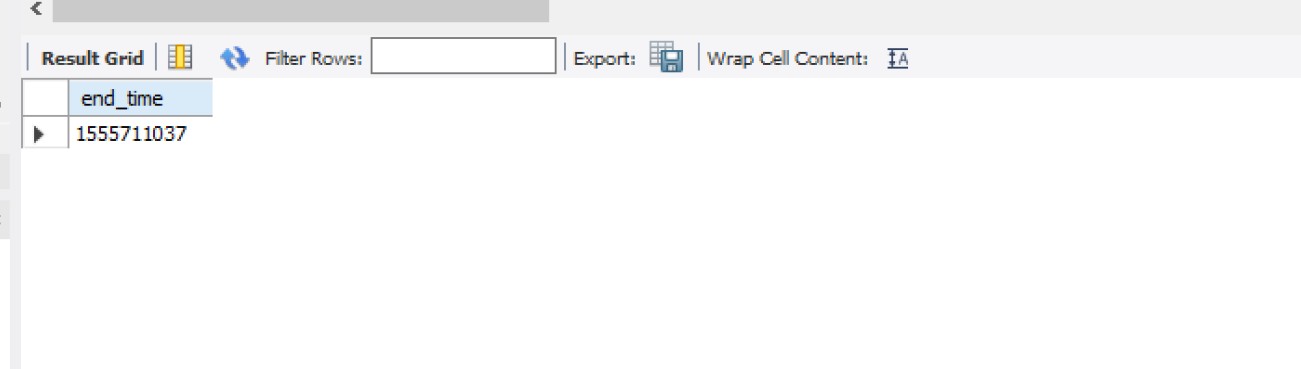


**6.What is the difference of end\_time between run 1 and run 5?**

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

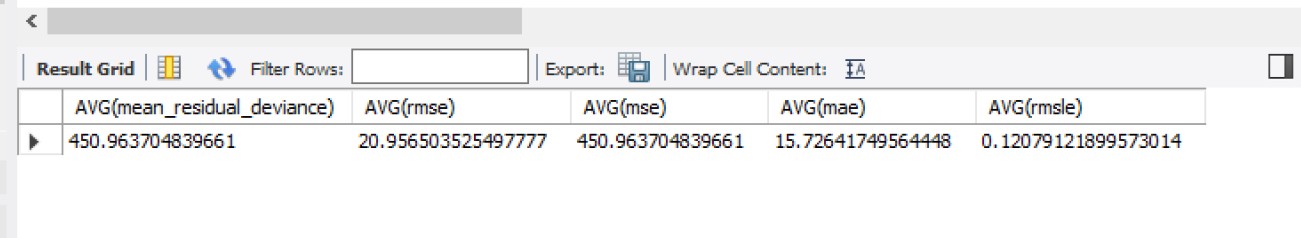
**Result:**



**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:**

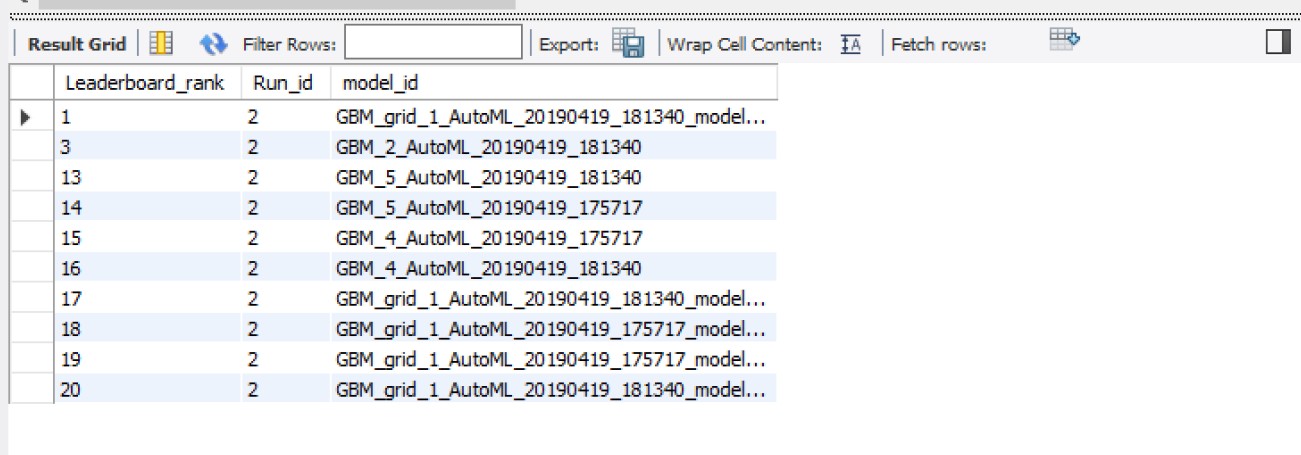


**8.Which models of gbm had leaderboard rank above 50 FOR 2nd run limiting to 10?**

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:**



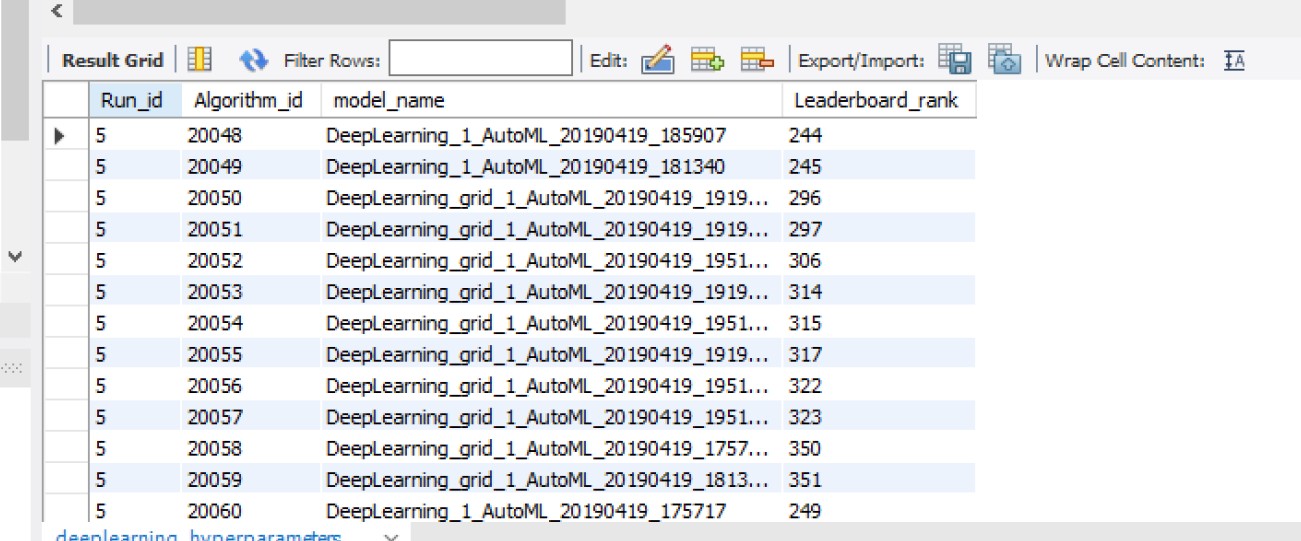
**9. Display the ranks of a leaderboard of all models for DRF hyperparameter for the 5th run?**

SELECT Run\_id, Algorithm\_id, model\_name, Leaderboard\_rank

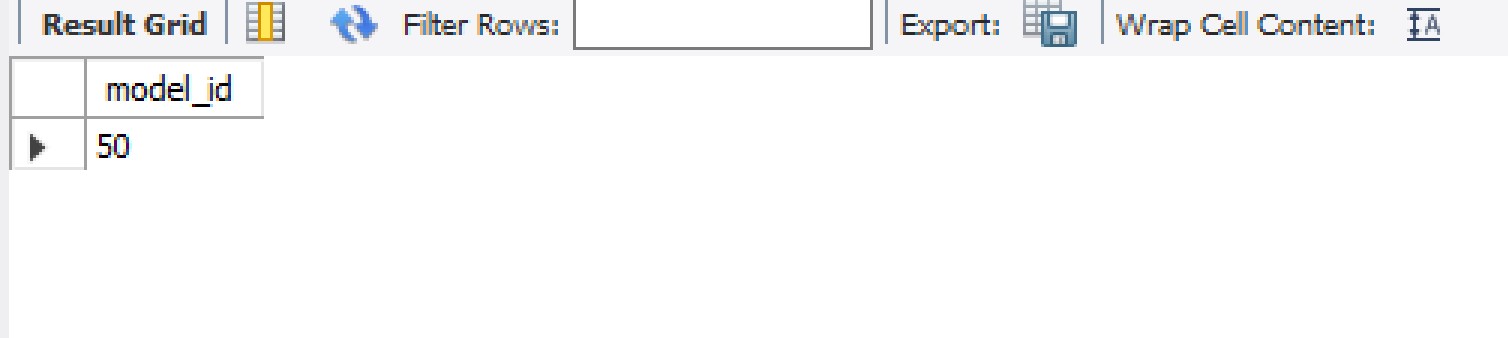
FROM deeplearning\_hyperparameters

WHERE Run\_id = 5;

**Result:**



**10**. **Find the count of all the models for the first run of GLM?** SELECT count(\*) model\_id FROM hyperparameter\_db.dl\_allparams **Result:**



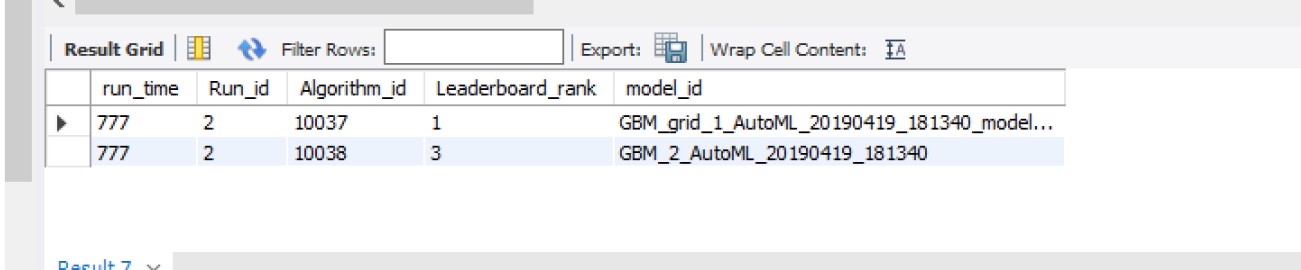
**11. What are the top three models for 2nd run of GBM models?**

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

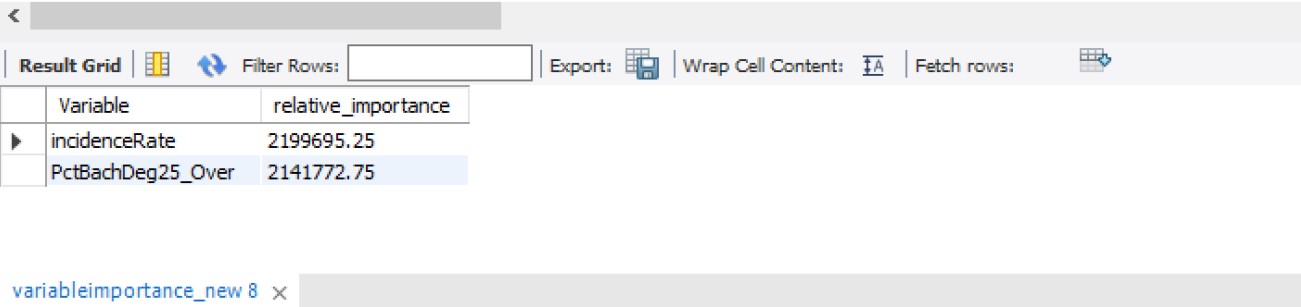
**Result:**



**12. Which variable showed highest importance?** SELECT distinct Variable, relative\_importance FROM variableimportance\_new

ORDER BY relative\_importance DESC LIMIT 2;

**Result:**

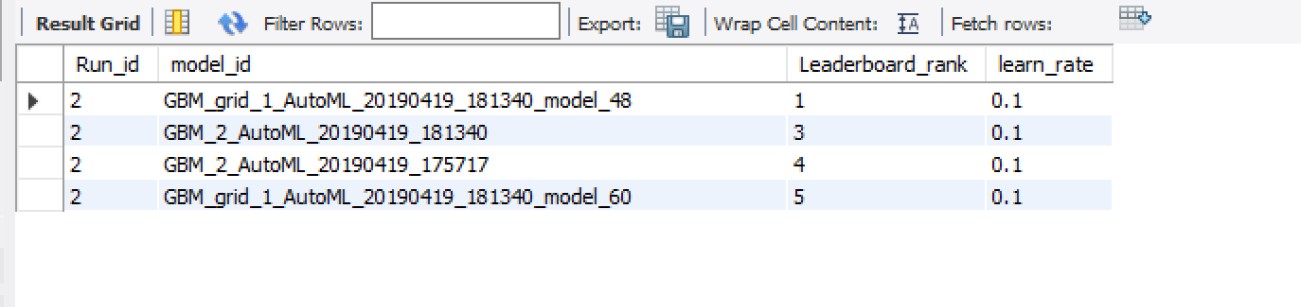


**13. What should I set the learning rate for GBM?** SELECT Run\_id,model\_id,Leaderboard\_rank,learn\_rate FROM gbm\_hyperparameters

WHERE Run\_id=2

ORDER BY Leaderboard\_rank limit 4;

**Result:**



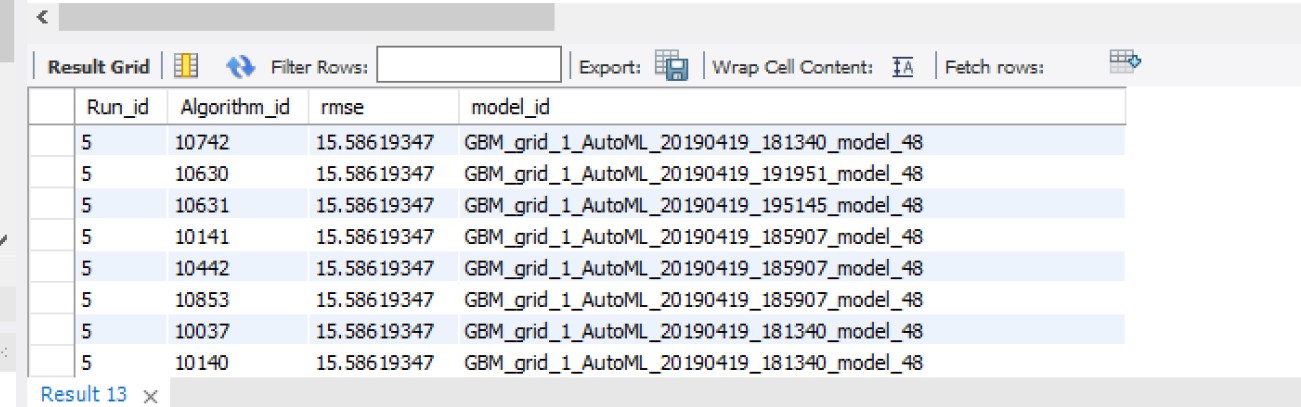
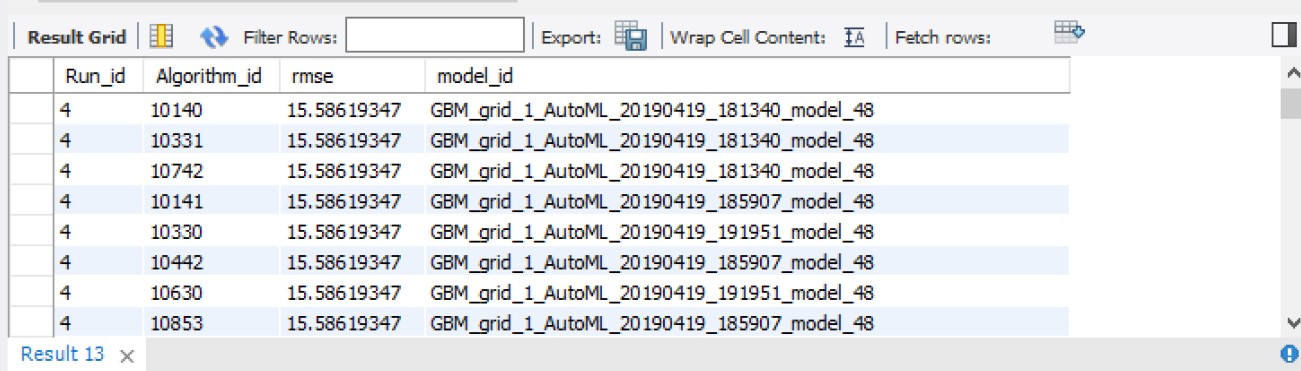
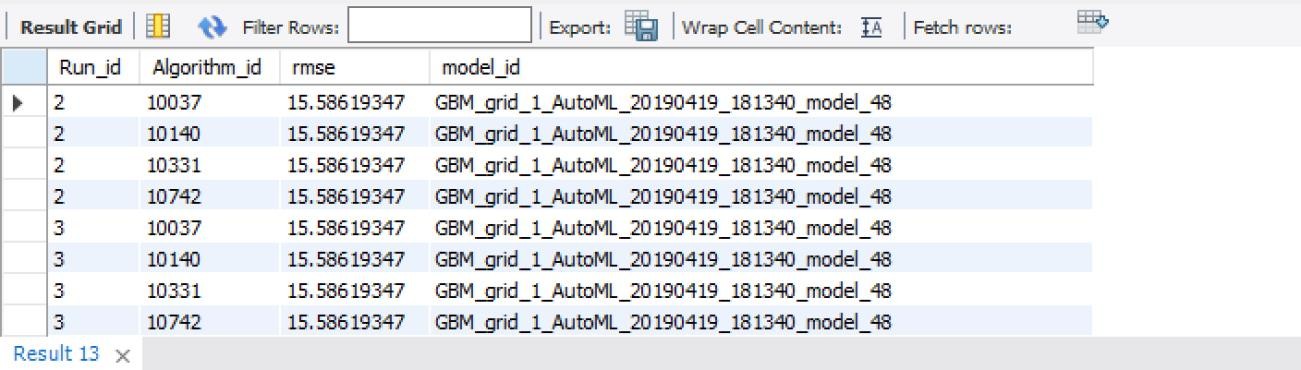
**14. Which model performed the Best for all Runs.**

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:**



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

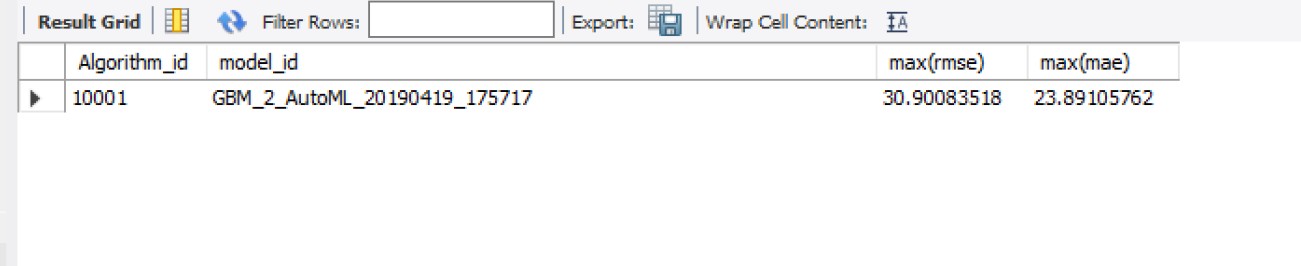
SELECT new\_leaderboard.Algorithm\_id, model\_id,

max(rmse),

max(mae) FROM hyperparameter\_db.new\_leaderboard

inner join algorithms on algorithms.Algorithm\_id=new\_leaderboard.Algorithm\_id

**Result:**



**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

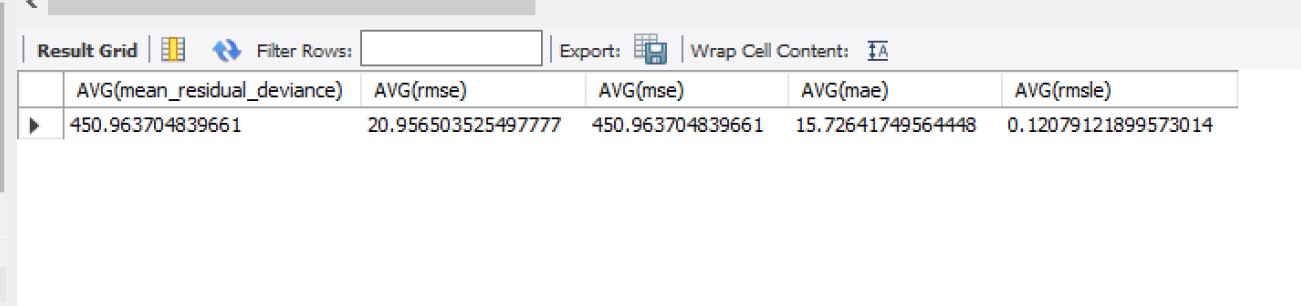
AVG(`new\_leaderboard`.`mean\_residual\_deviance`) AS

`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:**



**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`)))

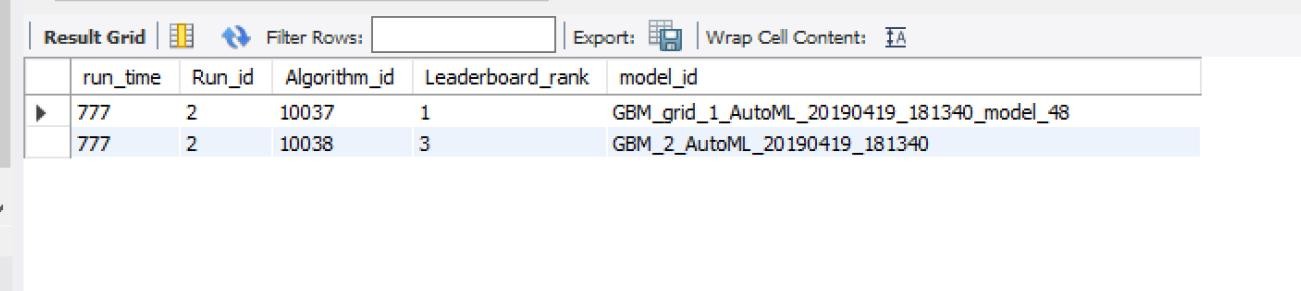
JOIN `algorithms` ON ((`algorithms`.`Algorithm\_id` =

`gbm\_hyperparameters`.`Algorithm\_id`))) WHERE

((`metadata`.`run\_time` = 777)

AND (`gbm\_hyperparameters`.`Leaderboard\_rank` < 4))

**Result:**



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

CREATE

ALGORITHM = UNDEFINED DEFINER = `root`@`localhost` SQL SECURITY DEFINER

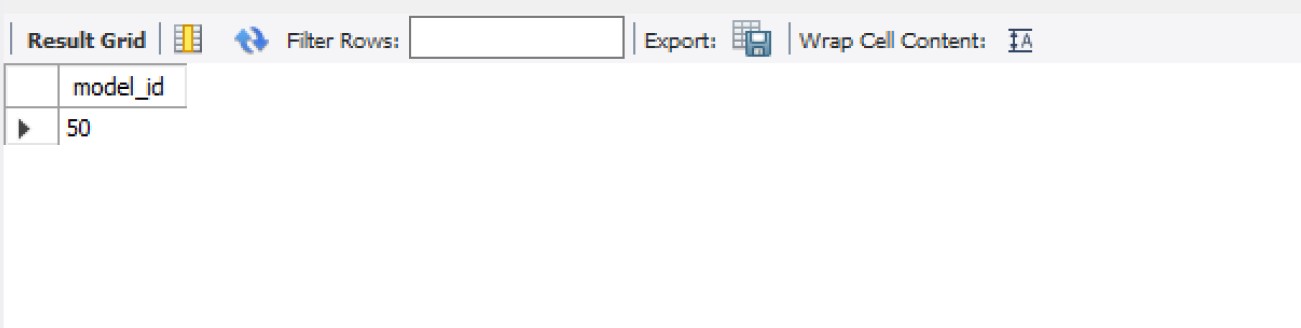
VIEW `view\_3` AS SELECT

COUNT(0) AS `model\_id`

FROM

`dl\_allparams`

**Result:**



**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`,

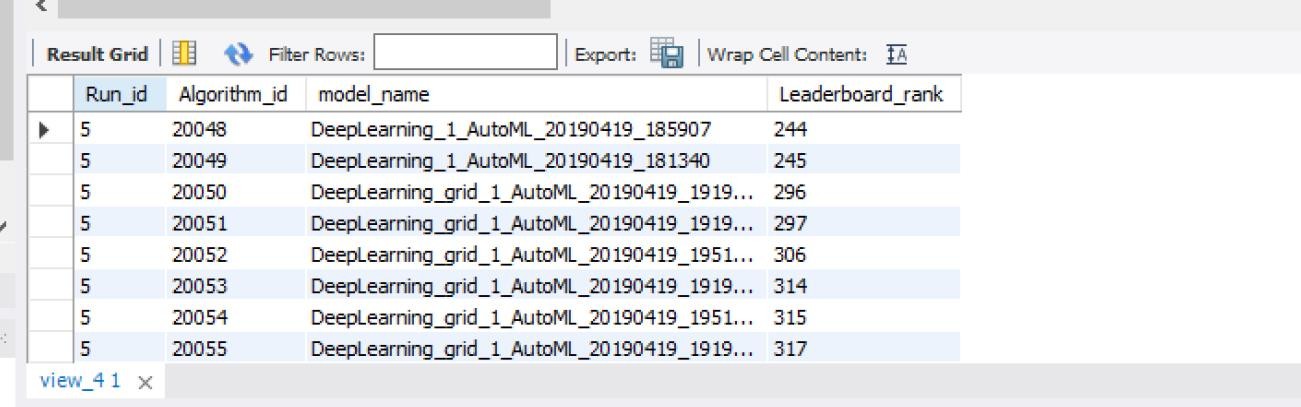
`deeplearning\_hyperparameters`.`Model\_name` AS `model\_name`,

`deeplearning\_hyperparameters`.`Leaderboard\_rank` AS `Leaderboard\_rank` FROM

`deeplearning\_hyperparameters` WHERE

(`deeplearning\_hyperparameters`.`Run\_id` = 5)

**Result:**



**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:**



**6. Which variable showed highest importance?**

CREATE

ALGORITHM = UNDEFINED DEFINER = `root`@`localhost` SQL SECURITY DEFINER

VIEW `view\_6` AS SELECT DISTINCT

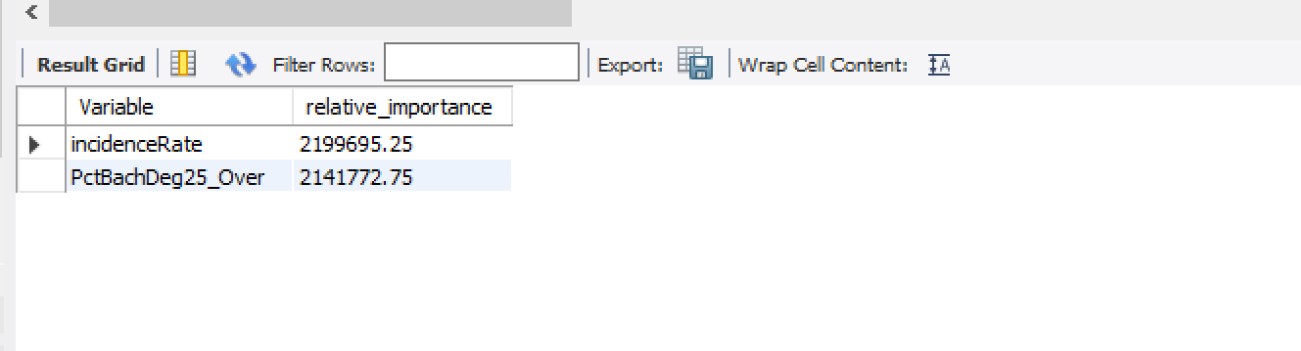
`variableimportance\_new`.`Variable` AS `Variable`,

`variableimportance\_new`.`relative\_importance` AS `relative\_importance` FROM

`variableimportance\_new`

ORDER BY `variableimportance\_new`.`relative\_importance` DESC LIMIT 2

**Result:**



**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";

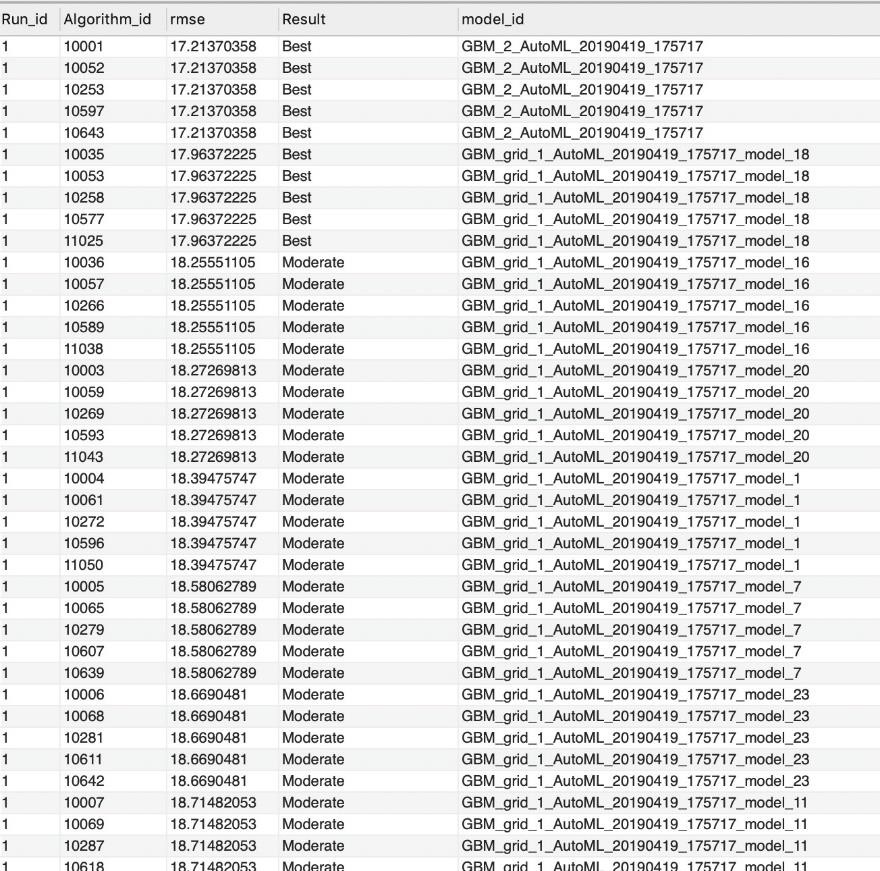
else

set result="Worst";

end if;

RETURN (result); END

**Result:**



**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;

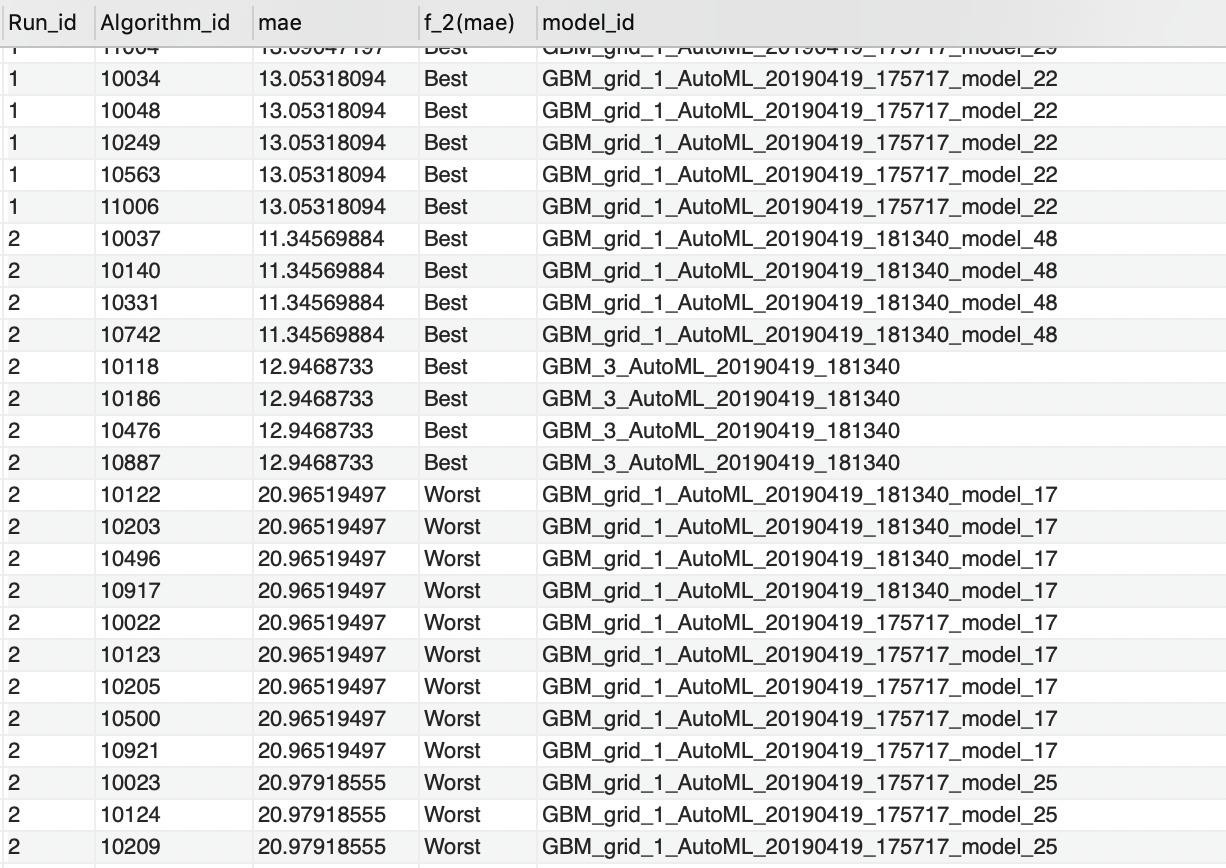
if mae > 18 then set result="Worst"; else

set result="Best ";

end if;

RETURN (result); END

**Result:**



**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:**



**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:**



**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:**



**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";

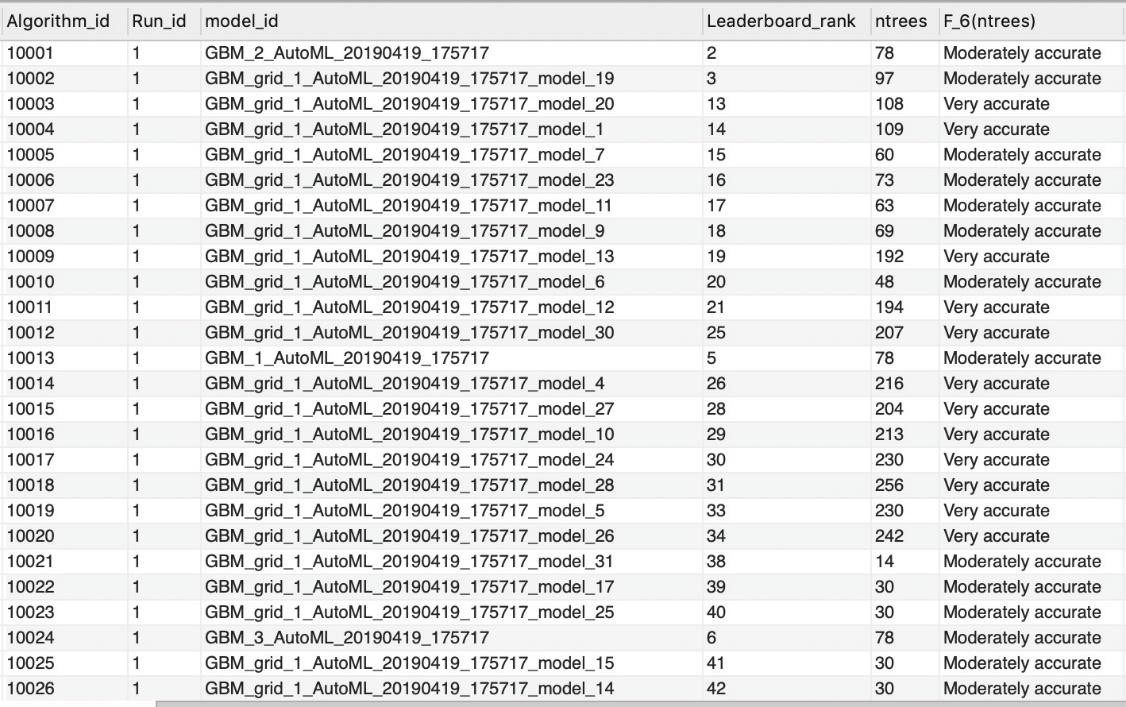
else

set result="Moderately accurate";

end if;

RETURN (result); END

**Result:**

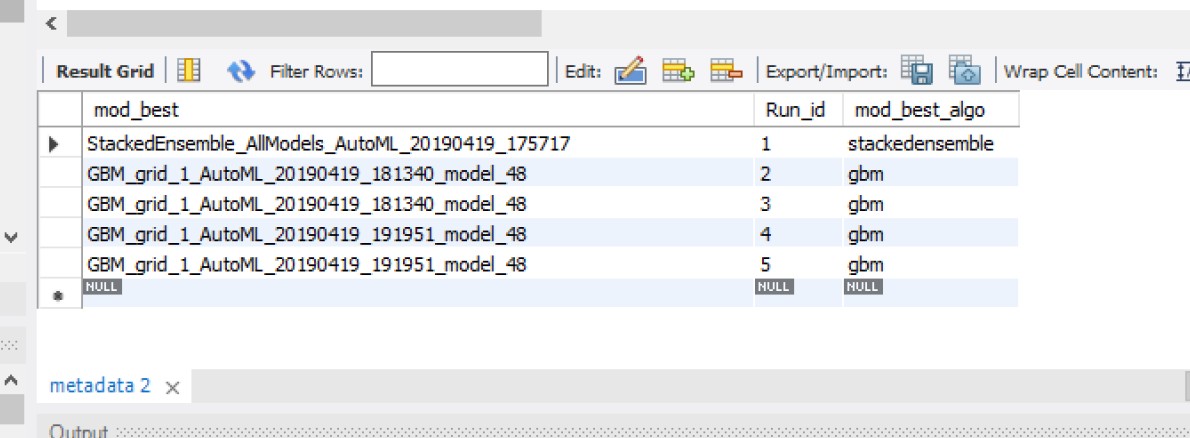


**ANALYSIS**

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:



• **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.

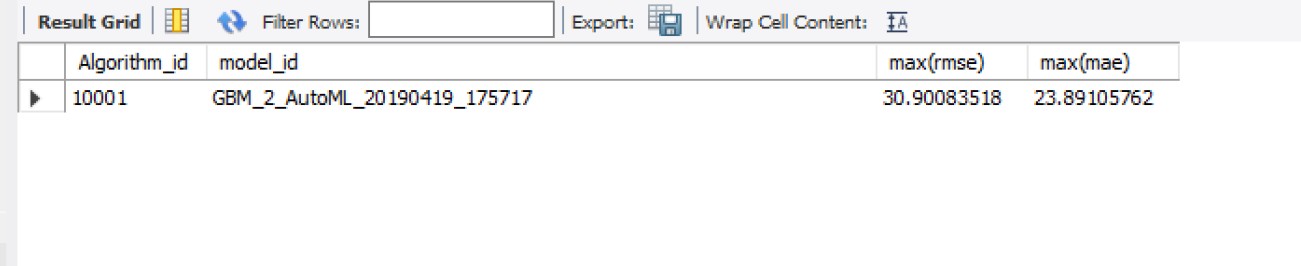
SELECT new\_leaderboard.Algorithm\_id, model\_id,

max(rmse),

max(mae) FROM hyperparameter\_db.new\_leaderboard

inner join algorithms on algorithms.Algorithm\_id=new\_leaderboard.Algorithm\_id

**Result:**



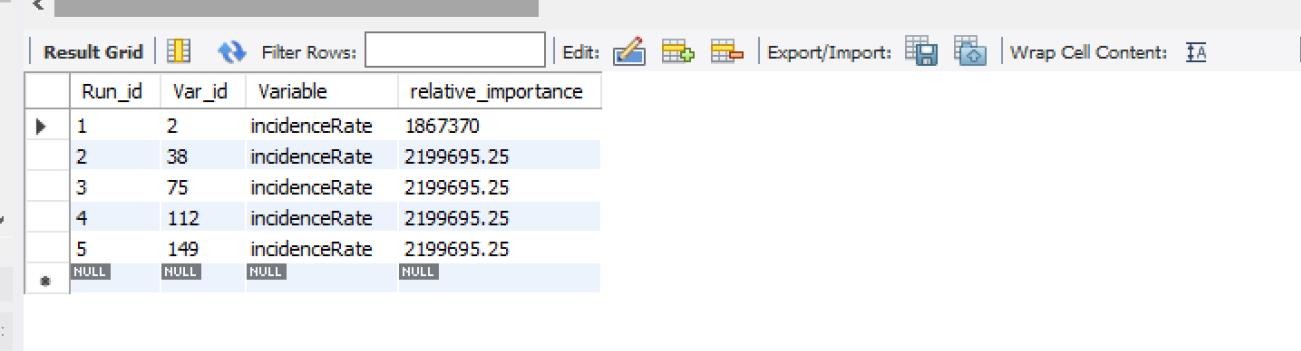
• **Most Important Variable**

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:**



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.

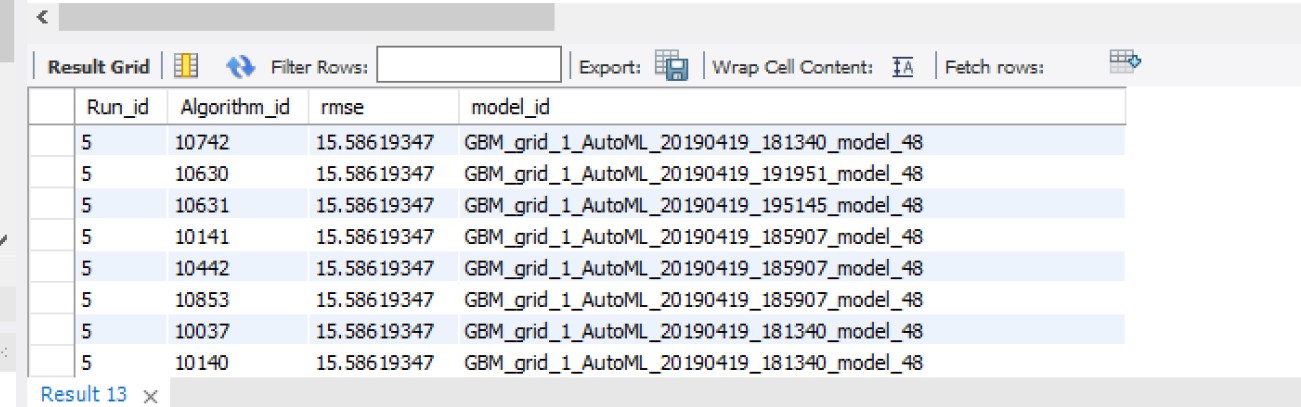
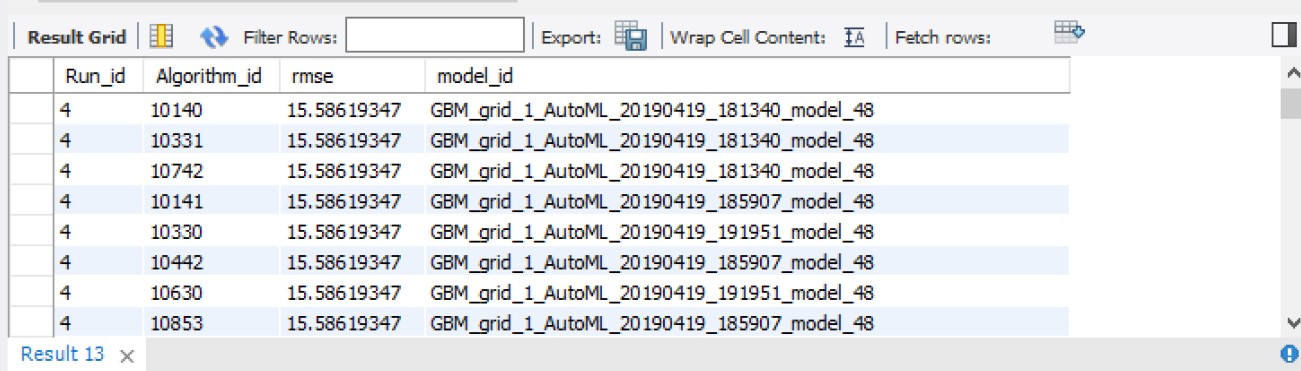
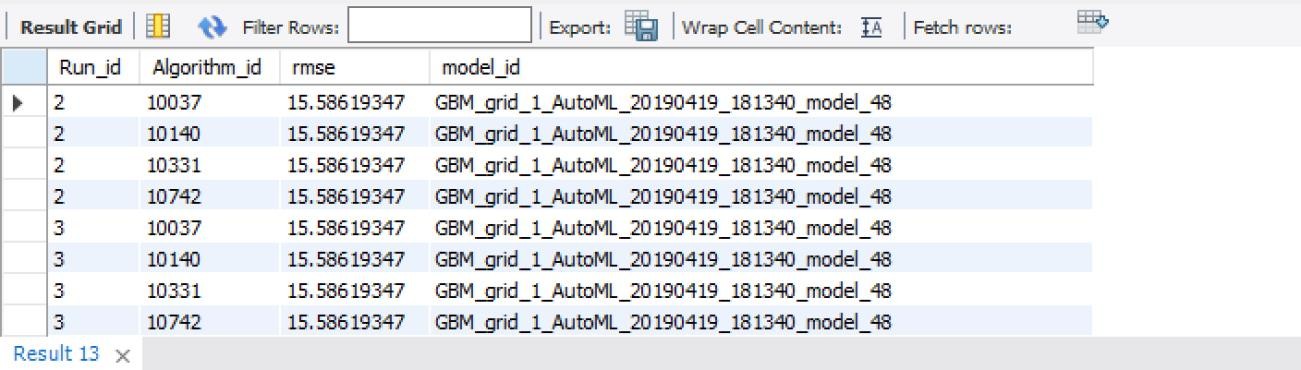
• **Choosing the best hyperparameter**

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



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

The code in the document by <'PURVANG JAYESH THAKKAR'>,'IRA ABHIJEET PANTBALEKUNDRI'> AND

<'NIKITA GAWDE'>is licensed under the MIT License Copyright <2019> <'PURVANG JAYESH THAKKAR'>,<'IRA ABHIJEET PANTBALEKUNDRI'> AND <'NIKITA GAWDE'>. 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.