

Northeastern University

Data Management and Database Design

INFO6210

Spring 2019

Hyperparameter Database Team 11

Project Report

By Darshan Durve & Shriram Karthikeyan

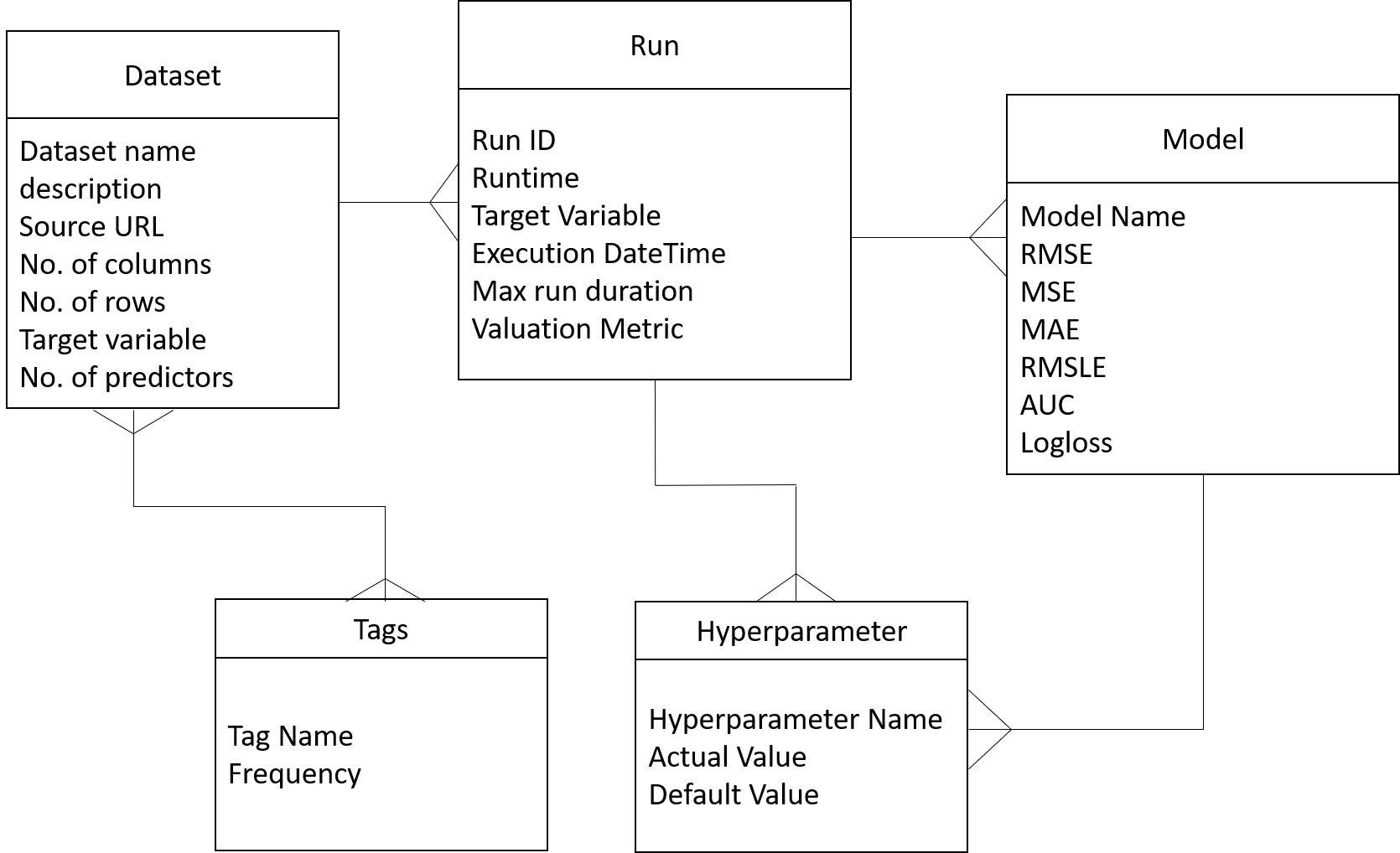
# Abstract

The goal of this hyperparameter project is to create a database consisting of hyperparameters that were extracted by running various machine learning models (classification/regression) on multiple datasets. Using the database of hyperparameters that is generated we will suggest hyperparameters for the dataset that the user would like to run ML models on; this aspect of the project is not currently within our scope therefore we will limit ourselves to building a database of hyperparameters for one dataset.

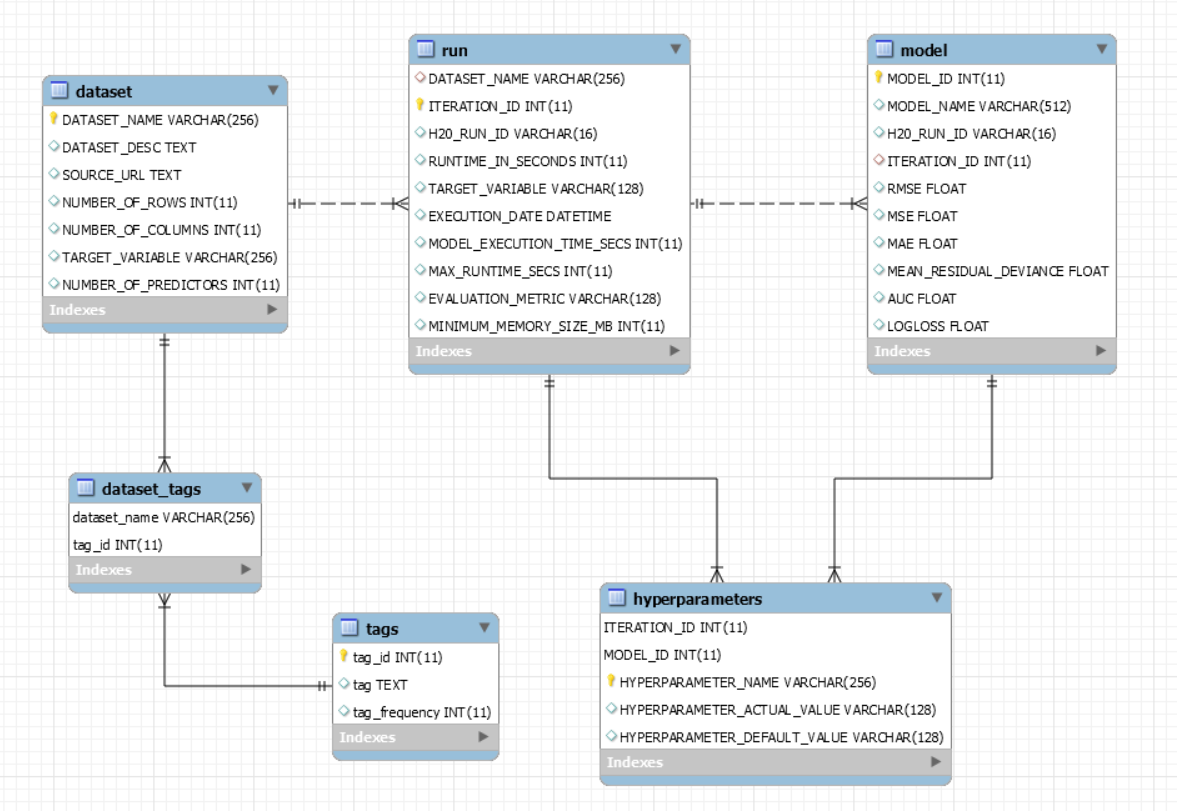
# The Dataset

We have got the CalCOFI dataset from Kaggle.com. The data set represents the longest (1949-present) and most complete (more than 50,000 sampling stations) time series of oceanographic and larval fish data in the world. It includes abundance data on the larvae of over 250 species of fish; larval length frequency data and egg abundance data on key commercial species; and oceanographic and plankton data.

# Conceptual Diagram



# ER Diagram



Above E-R diagram explains the relationship between the models for each run with their hyperparameters. Each run has different models generated by H2O and each model has different hyperparameters associated to it. We have also stored tags to associate the dataset with keywords.

# Normalization

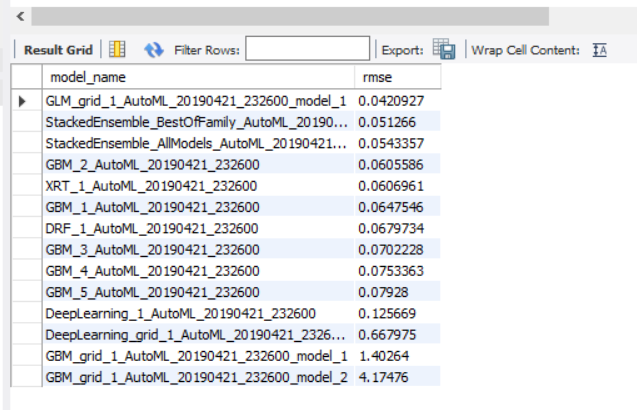
We have strived to follow 3NF in our physical data model and normalized table to a reasonable extent. For instance the hyperparameter table contains a composite key of Iteration\_ID, Model\_ID and the Hyperparameter\_Name out of which Iteration\_ID and Model\_ID are foreign keys referencing the Run and Model tables respectively.

# Use Cases

## Retrieve the rmse value for the models of the lowest runtime.

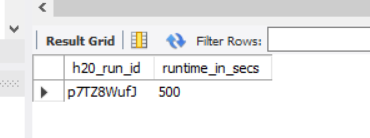
* select m.model\_name, m.rmse from model m join run r on m.iteration\_id=r.iteration\_id

where r.max\_runtime\_secs=(select min(max\_runtime\_secs) from run);



## Retrieve the runid from the meta-data for the lowest runtime.

- Select h20\_run\_id, max\_runtime\_secs as runtime\_in\_secs from run where max\_runtime\_secs=(select min(max\_runtime\_secs) from run);

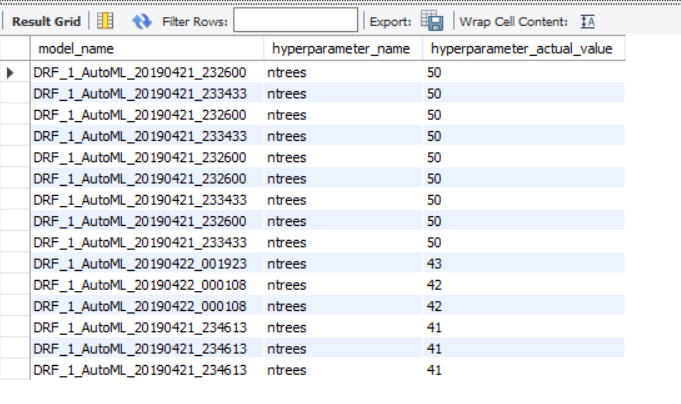


## Retrieve the ntrees from the Random Forest algorithm.

- select m.model\_name, h.hyperparameter\_name, h.hyperparameter\_actual\_value

from model m join hyperparameters h on m.model\_id=h.model\_id

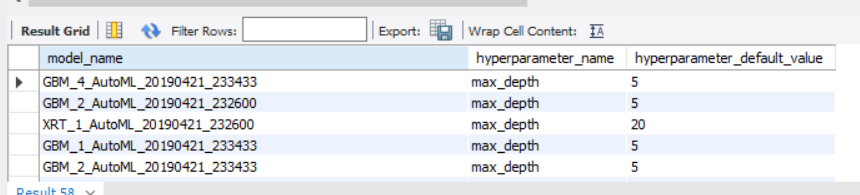
where m.model\_name like 'drf%' and h.hyperparameter\_name = 'ntrees' order by h.hyperparameter\_actual\_value desc;



## 4. The default max\_depth of the models

- select m.model\_name, h.hyperparameter\_name, h.hyperparameter\_default\_value from hyperparameters h join model m on h.model\_id=m.model\_id

where h.hyperparameter\_name = 'max\_depth';

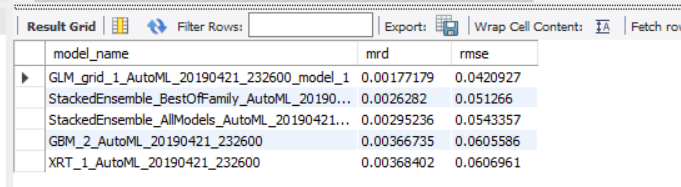


## The mean residual deviance for the top 5 models of the lowest runtime.

#Based on rmse

select model\_name, mean\_residual\_deviance as mrd, rmse from model m join run r on m.iteration\_id=r.iteration\_id

where r.max\_runtime\_secs=(select min(max\_runtime\_secs) from run) order by rmse limit 5;

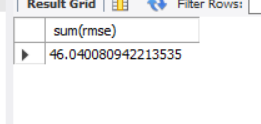


## Sum of rmse values of any least or highest runtime.

#For highest

select sum(rmse) from model m join run r on m.iteration\_id=r.iteration\_id

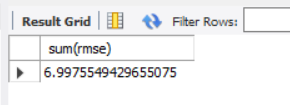
where r.max\_runtime\_secs in (select max(max\_runtime\_secs) from run);



#For lowest

select sum(rmse) from model m join run r on m.iteration\_id=r.iteration\_id

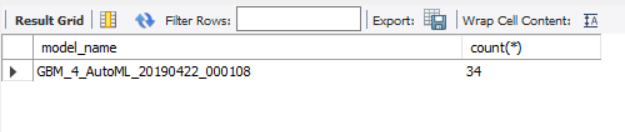
where r.max\_runtime\_secs in (select min(max\_runtime\_secs) from run);



## The number of models of the same algorithm for a particular runtime(1100).

select model\_name, count(\*) from model m join run r on m.iteration\_id=r.iteration\_id

where r.max\_runtime\_secs = '1100' and model\_name like 'GBM%';

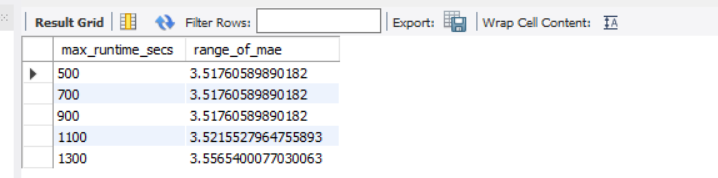


## The difference between the highest and lowest mae values of the models for a runtime.

* select r.max\_runtime\_secs, max(m.mae)-min(m.mae) as range\_of\_mae

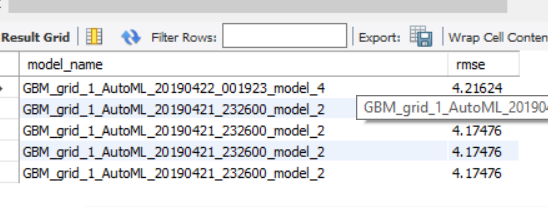
from run r join model m on m.iteration\_id=r.iteration\_id group by r.max\_runtime\_secs

order by max\_runtime\_secs;



## List of top 5 models with low performance based on their rmse values

* select model\_name, rmse from model order by rmse desc limit 5;



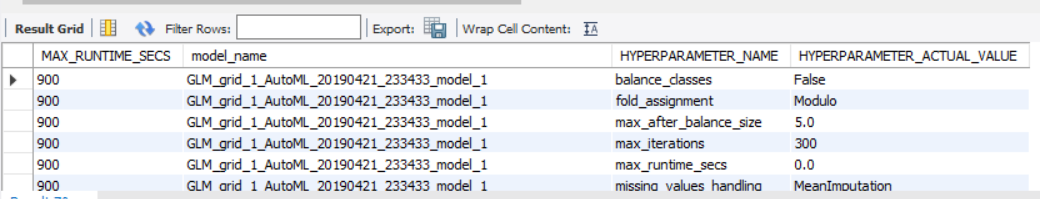
## Suggest me a list of run times, model\_names, hyperparameters and their values to get the lowest rmse

* select r.MAX\_RUNTIME\_SECS, m.model\_name, h.HYPERPARAMETER\_NAME, h.HYPERPARAMETER\_ACTUAL\_VALUE

from run r, model m, hyperparameters h

where r.iteration\_id=m.iteration\_id and m.model\_id=h.model\_id

and m.rmse = (select min(rmse) from model);



# Functions

## Suggest a dataset based on the given tag –

CREATE DEFINER=`root`@`localhost` FUNCTION `fetch\_dataset`(tag\_name varchar(128)) RETURNS varchar(128) CHARSET utf8mb4

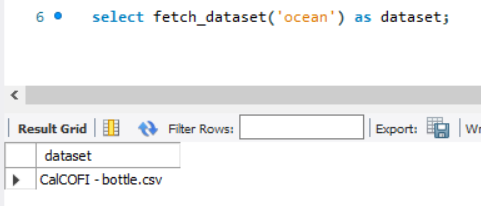
BEGIN

declare dataset varchar(128);

select distinct d.dataset\_name into dataset from tags t join dataset\_tags d on t.tag\_id=d.tag\_id where tag like concat('%',tag\_name,'%');

RETURN dataset;

END



## Suggest best value of specified hyperparameter of specified model at specified runtime

CREATE DEFINER=`root`@`localhost` FUNCTION `best\_hyperparameter`(runtime varchar(128), model varchar(128), hyper varchar(128)) RETURNS varchar(128) CHARSET utf8mb4

BEGIN

declare best\_value varchar(128);

select distinct HYPERPARAMETER\_ACTUAL\_VALUE into best\_value from model m join run r on m.iteration\_id=r.iteration\_id

join hyperparameters h on m.model\_id=h.model\_id

where r.max\_runtime\_secs = runtime and

m.model\_name like concat('%',model,'%')

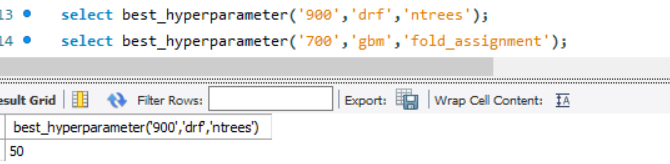
and h.hyperparameter\_name like concat('%',hyper,'%')

and m.rmse = (select min(m.rmse) from model m join run r on m.iteration\_id=r.iteration\_id

where r.max\_runtime\_secs = runtime and m.model\_name like concat('%',model,'%'));

RETURN best\_value;

END



## Get number of models for a given range of RMSE values

CREATE DEFINER=`root`@`localhost` FUNCTION `count\_models`(FROM\_RMSE float, TO\_RMSE float) RETURNS int(11)

DETERMINISTIC

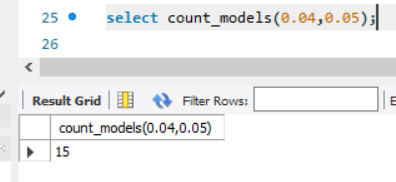
BEGIN

DECLARE MODEL\_COUNT int;

select count(model\_name) into model\_count from model where rmse between from\_rmse and to\_rmse;

RETURN MODEL\_COUNT;

END



## Get the best model for a given runtime

CREATE DEFINER=`root`@`localhost` FUNCTION `best\_model`(runtime int) RETURNS varchar(128) CHARSET utf8mb4

DETERMINISTIC

BEGIN

DECLARE model varchar(128);

select model\_name into model from model m join run r on m.iteration\_id=r.iteration\_id

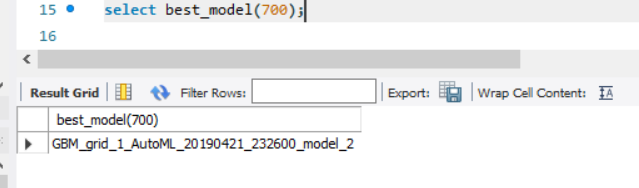
where max\_runtime\_secs = 900

group by m.model\_name, rmse

order by rmse desc limit 1;

RETURN model;

END



# Views

## View 1

1. Create a view to get the highest rmse value for the drf algorithm with the least runtime

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `hyperparameter\_db\_11`.`max\_mae` AS

SELECT

MAX(`m`.`MAE`) AS `max(mae)`

FROM

((`hyperparameter\_db\_11`.`model` `m`

JOIN `hyperparameter\_db\_11`.`hyperparameters` `h`)

JOIN `hyperparameter\_db\_11`.`run` `r`)

WHERE

((`h`.`MODEL\_ID` = `m`.`MODEL\_ID`)

AND (`m`.`ITERATION\_ID` = `r`.`ITERATION\_ID`)

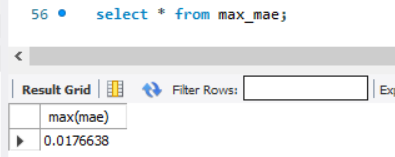
AND (`m`.`MODEL\_NAME` LIKE '%drf%')

AND (`r`.`MAX\_RUNTIME\_SECS` = (SELECT

MIN(`hyperparameter\_db\_11`.`run`.`MAX\_RUNTIME\_SECS`)

FROM

`hyperparameter\_db\_11`.`run`)))



## View 2

1. Create a view to display max and min values of Tweedie power for all deep learning algorithms for any all runtime

* CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `hyperparameter\_db\_11`.`tweedie\_power` AS

SELECT

`r`.`MAX\_RUNTIME\_SECS` AS `max\_runtime\_secs`,

`m`.`MODEL\_NAME` AS `model\_name`,

`h`.`HYPERPARAMETER\_NAME` AS `hyperparameter\_name`,

MAX(`h`.`HYPERPARAMETER\_ACTUAL\_VALUE`) AS `max(h.HYPERPARAMETER\_ACTUAL\_VALUE)`,

MIN(`h`.`HYPERPARAMETER\_ACTUAL\_VALUE`) AS `min(h.HYPERPARAMETER\_ACTUAL\_VALUE)`

FROM

((`hyperparameter\_db\_11`.`model` `m`

JOIN `hyperparameter\_db\_11`.`hyperparameters` `h`)

JOIN `hyperparameter\_db\_11`.`run` `r`)

WHERE

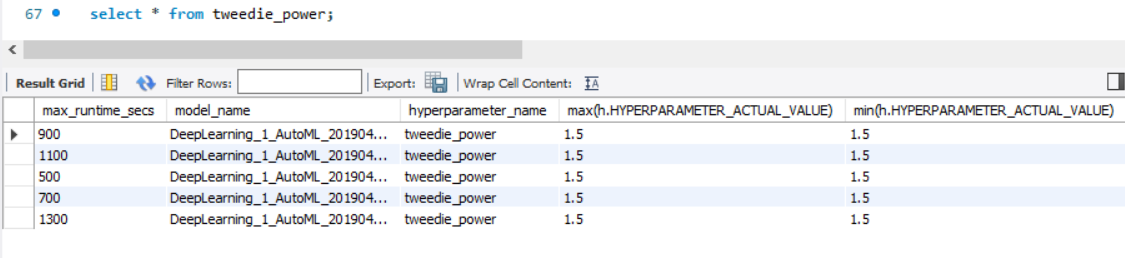
((`h`.`MODEL\_ID` = `m`.`MODEL\_ID`)

AND (`m`.`ITERATION\_ID` = `r`.`ITERATION\_ID`)

AND (`h`.`HYPERPARAMETER\_NAME` = 'tweedie\_power')

AND (`m`.`MODEL\_NAME` LIKE '%deep%'))

GROUP BY `r`.`MAX\_RUNTIME\_SECS`



## View 3

View that contains the best hyperparameters by dataset

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `best\_hp\_by\_dataset` AS

SELECT

`d`.`DATASET\_NAME` AS `dataset\_name`,

`h`.`HYPERPARAMETER\_NAME` AS `HYPERPARAMETER\_NAME`,

`h`.`HYPERPARAMETER\_ACTUAL\_VALUE` AS `HYPERPARAMETER\_ACTUAL\_VALUE`,

`h`.`HYPERPARAMETER\_DEFAULT\_VALUE` AS `HYPERPARAMETER\_DEFAULT\_VALUE`

FROM

(((`dataset` `d`

LEFT JOIN `run` `r` ON ((`d`.`DATASET\_NAME` = `r`.`DATASET\_NAME`)))

LEFT JOIN `model` `m` ON ((`r`.`ITERATION\_ID` = `m`.`MODEL\_ID`)))

LEFT JOIN `hyperparameters` `h` ON (((`m`.`MODEL\_ID` = `h`.`MODEL\_ID`)

AND (`m`.`ITERATION\_ID` = `h`.`ITERATION\_ID`))))

WHERE

(`m`.`RMSE` = (SELECT

MIN(`m2`.`RMSE`)

FROM

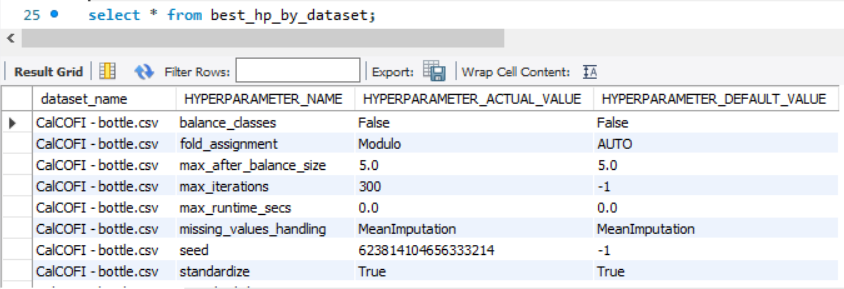
((`dataset` `d2`

LEFT JOIN `run` `r2` ON ((`d2`.`DATASET\_NAME` = `r2`.`DATASET\_NAME`)))

LEFT JOIN `model` `m2` ON ((`r2`.`ITERATION\_ID` = `m2`.`MODEL\_ID`)))

WHERE

(`d2`.`DATASET\_NAME` = `d`.`DATASET\_NAME`)))



This view contains the best set of hyperparameters for all datasets which makes it easier for the end user to query without having to join tables and see only the necessary information.

## View 4

This view gives the summary count of the number of runs, models and hyperparameters by dataset

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `dataset\_summary` AS

SELECT

`d`.`DATASET\_NAME` AS `dataset\_name`,

COUNT(DISTINCT `r`.`ITERATION\_ID`) AS `number\_of runs`,

COUNT(DISTINCT `m`.`MODEL\_ID`) AS `number\_of\_models`,

COUNT(DISTINCT `h`.`HYPERPARAMETER\_NAME`) AS `number\_of\_hyperparameters`

FROM

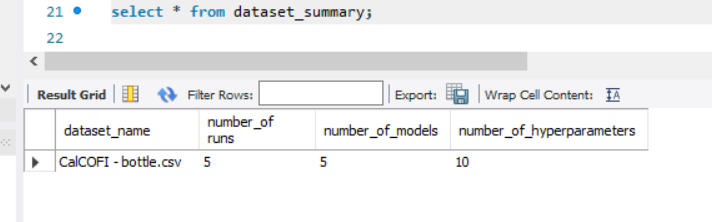
(((`dataset` `d`

LEFT JOIN `run` `r` ON ((`d`.`DATASET\_NAME` = `r`.`DATASET\_NAME`)))

LEFT JOIN `model` `m` ON ((`r`.`ITERATION\_ID` = `m`.`MODEL\_ID`)))

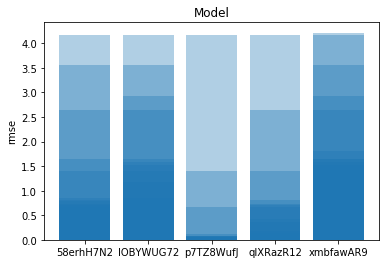
LEFT JOIN `hyperparameters` `h` ON (((`m`.`MODEL\_ID` = `h`.`MODEL\_ID`)

AND (`m`.`ITERATION\_ID` = `h`.`ITERATION\_ID`))))

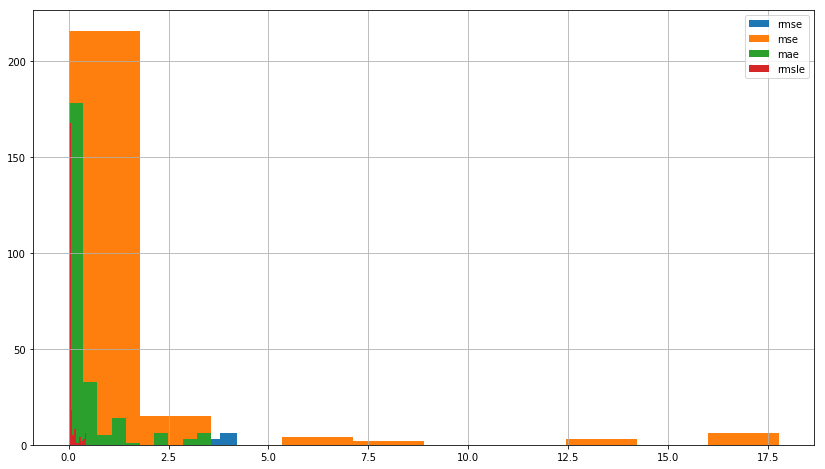


# Analytics

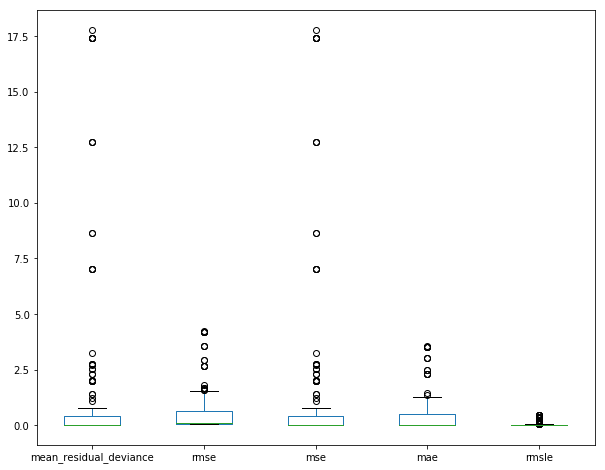
## Range of RMSE values for different run IDs



## Frequency of Metrics



## Box plot for performance metrics



# Conclusion

This hyperparameter project has taught us how to model a database around a relatively new concept for us as database students. We have tried to create a database that can be implemented in a production environment to hold data about hyperparameters, models, and run for thousands of datasets and millions of records at the hyperparameter level. To make this project better we would like to delve deeper into making the functionality to search for tags and finding an appropriate dataset and suggestion of hyperparameters our focus.

# Citations & References

https://stackoverflow.com/questions/19587118/iterating-through-directories-with-python

https://stackoverflow.com/questions/7099290/how-to-ignore-hidden-files-using-os-listdir

https://stackoverflow.com/questions/845058/how-to-get-line-count-cheaply-in-python

https://stackoverflow.com/questions/48996822/python-drop-rows-from-a-pandas-dataframe-that-contain-numbers

https://stackoverflow.com/questions/11346283/renaming-columns-in-pandashttps://www.nltk.org/book/ch05.html

https://www.nltk.org/book/ch05.html

https://stackoverflow.com/questions/2661778/tag-generation-from-a-text-content

https://stackoverflow.com/questions/14753321/add-auto-increment-id-to-existing-table

https://stackoverflow.com/questions/8384737/extract-file-name-from-path-no-matter-what-the-os-path-format

[*https://www.interviewqs.com/ddi\_code\_snippets/add\_new\_col\_df\_default\_value*](https://www.interviewqs.com/ddi_code_snippets/add_new_col_df_default_value)

# License

The code in this Hyperparameter project document by Shriram Karthikeyan and Darshan Durve is licensed under CC BY 3.0 https://creativecommons.org/licenses/by/3.0/us/

The code in the Hyperparameter project document by Shriram Karthikeyan and Darshan Durve is licensed under the MIT License https://opensource.org/licenses/MIT