**1. Abstract**

The project is a part of skunkworks hyperparameters project. The project has two main parts where part one is dedicated to create a database that store the metadata and data set details of Melbourne Housing Data. Part two is to store evaluation and hyperparameters that are used to perform algorithms on data in part one.

As a part of the DMDD team we created a conceptual model and stored all the data into a physical database by going through the following procedures

• Scraped JSON files and stores the data in dataframes further saving them in CSV Format.

• Created a conceptual model of the database.

• Perform database normalization – 1NF, 2NF, 3NF.

• Created tables as per the requirement.

• Inserted data into the respective tables from the file which is generated and stored by the Data Science member.

• Queried database using SQL, created Views (virtual tables) and Functions for various use cases of the database.

• Performed analytics on the database using queries e.g. getting the best value for the hyperparameter from the database.

## 2. Explain Data Source

Data related to Melbourne Housing is obtained from [Kaggle.com](https://www.kaggle.com/harlfoxem/housesalesprediction). This data is validated and preprocessed to assure the credibility and integrity. Data for machine learning models will be generated by our group. The output file containing the hyperparameters information is provided by the data science group as JSON files which we parse to obtain the required information to populate the database tables.

## 3. Conceptual Schema

## A screenshot of a cell phone Description automatically generated

## 4. ER Diagram

A screenshot of a cell phone

Description automatically generated

**5. Normalization (till 3NF)**

**1NF**

**Requirement:**

1. No repeating groups
2. Atomic
3. Each field has a unique name
4. Primary key

**2NF**

**Requirement:**

1. Already 1NF
2. No partial dependencies
3. No calculated data

**3NF**

**Requirement:**

1. Already 2NF
2. No transitive dependencies

*Table 1: Dataset*

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

*Table 2: Dataset\_Description*

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

*Table 3: Model\_Runtime*

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

*Table 4: Model*

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

*Table 5: Model\_Metrics*

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

*Table 6: Algorithm*

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

*Table 7: GBM\_Hyperparameters*

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

*Table 8: GLM\_Hyperparameters*

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

*Table 9: DRF\_Hyperparameters*

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

*Table 10: XRT\_Hyperparameters*

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

**6. Physical model**

Physical model is created using MySQL database

The [SQL files](https://github.com/INFO6105-Spring19/hyperparameter-db-project-db08/tree/master/SQL) contains all the SQL physical model information.

## 7. Use Cases

[Use cases](https://github.com/INFO6105-Spring19/hyperparameter-db-project-db08/blob/master/SQL/Hyper_usecases.sql) contains the use cases SQL queries.

#1. Find the best model amongst all the models of GBM algorithm

SELECT Model\_name FROM model\_metrics

WHERE RMSE=(SELECT MIN(RMSE) FROM model\_metrics a JOIN GBM\_Hyperparameters b ON a.Model\_name=b.Model\_name)



#2. Count the number of models formed for runtime 500s

SELECT count(Model\_name) FROM Model a, Model\_Runtime b

WHERE a.runtime\_ID=b.runtime\_ID AND b.Run\_Time =500



#3. For what runtime is the best model generated?

SELECT run\_time, rmse FROM model\_runtime a, model\_metrics b, model c

WHERE a.runtime\_ID=c.runtime\_ID AND b.model\_name = c.model\_name AND b.rmse=(SELECT MIN(RMSE) FROM model\_metrics )



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

SELECT b.Model\_name FROM model\_runtime a, model\_metrics b, model c

WHERE a.runtime\_ID=c.runtime\_ID AND b.model\_name = c.model\_name AND a.run\_time=500 AND b.model\_name LIKE "DRF%"



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

SELECT avg(rmse), count(a.model\_name) FROM model\_metrics a, model\_runtime b, model c

WHERE b.runtime\_ID=c.runtime\_ID AND a.model\_name = c.model\_name AND b.run\_time=500 AND a.model\_name LIKE "GBM%"



#6. What is the Maximum RMSE value for GLM model?

SELECT MAX(RMSE) FROM model\_metrics

WHERE RMSE=(SELECT MAX(RMSE) FROM model\_metrics a JOIN GLM\_Hyperparameters b ON a.Model\_name=b.Model\_name) LIMIT 1



#7. List the number of GBM models which has distribution=”gaussian”?

SELECT COUNT(model\_Name) FROM GBM\_Hyperparameters

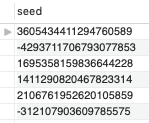
WHERE distribution LIKE 'gaussian%' AND model\_name LIKE "GBM%"



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

SELECT seed FROM GLM\_Hyperparameters a, model\_metrics b

WHERE a.model\_name=b.model\_name AND b.rmse =(SELECT MIN(RMSE) FROM model\_metrics a JOIN GLM\_Hyperparameters b ON a.Model\_name=b.Model\_name)



#9. Which model has maximum accuracy?

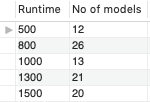
SELECT model\_name FROM model\_metrics ORDER BY rmse LIMIT 1



#10. Display how many models were created for each runtime?

SELECT a.run\_time as 'Runtime', COUNT( b.model\_name) as 'No of models' FROM model\_runtime a, Model\_metrics b, model c

WHERE a.runtime\_ID=c.runtime\_ID AND b.model\_name = c.model\_name GROUP BY a.run\_time



## 8. Views

[Views](https://github.com/INFO6105-Spring19/hyperparameter-db-project-db08/blob/master/SQL/hyper_views.sql) contain the Views SQL queries.

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

Create view Models\_for\_algos AS

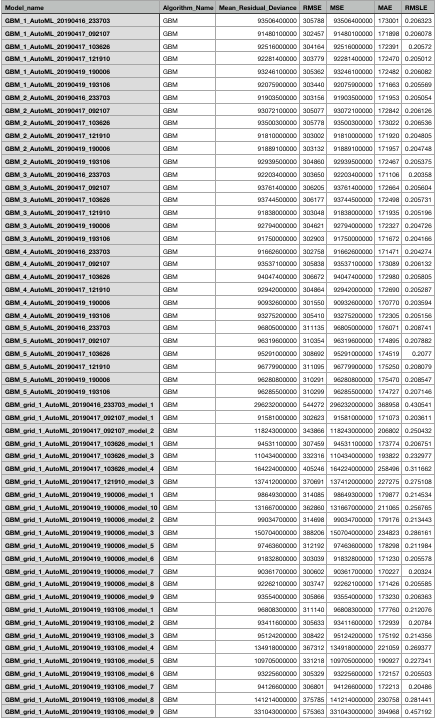
select M.Model\_name, A.Algorithm\_Name, MM.Mean\_Residual\_Deviance, MM.RMSE, MM.MSE, MM.MAE, MM.RMSLE

from Algorithm A

join Model M on A.Algorithm\_ID = M.Algorithm\_ID

join Model\_Metrics MM on MM.Model\_Name = M.Model\_Name

where A.Algorithm\_Name ="GBM";



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

Create view Models\_for\_runtime AS

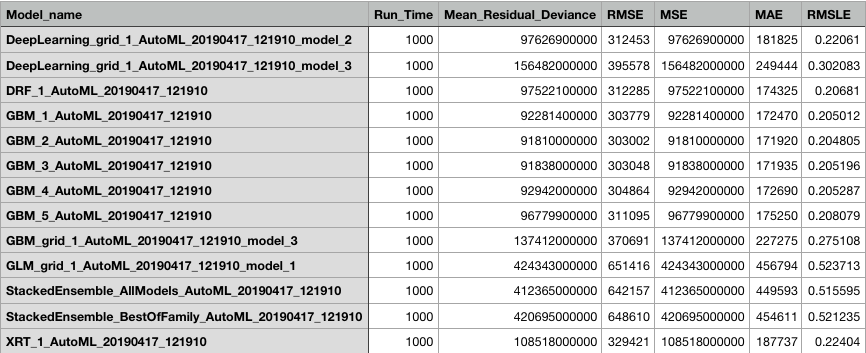
select M.Model\_name, R.Run\_Time, MM.Mean\_Residual\_Deviance, MM.RMSE, MM.MSE, MM.MAE, MM.RMSLE

from Model\_Runtime R

join Model M on R.Runtime\_ID = M.Runtime\_ID

join Model\_Metrics MM on MM.Model\_Name = M.Model\_Name

where R.Run\_Time=1000;



3) Create view for 50 best models of all algorithms

Create view 50\_best\_models AS

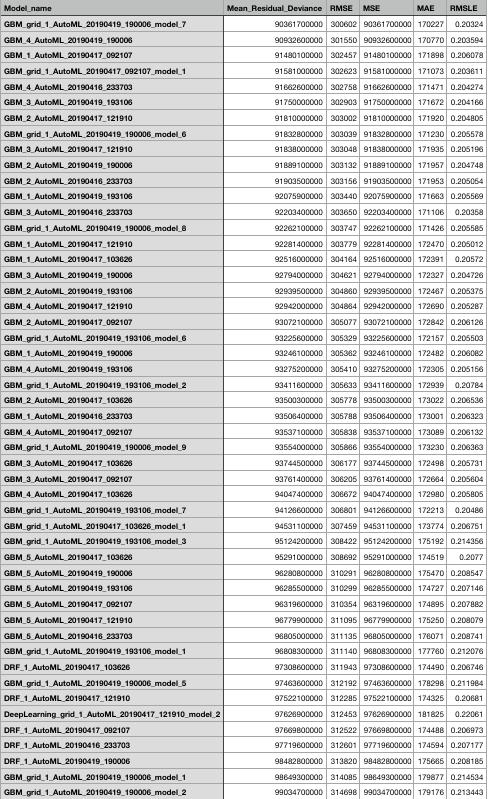
select M.Model\_name, MM.Mean\_Residual\_Deviance, MM.RMSE, MM.MSE, MM.MAE, MM.RMSLE

from Model M

join Model\_Metrics MM on MM.Model\_Name = M.Model\_Name

ORDER BY rmse , mse

LIMIT 50;



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

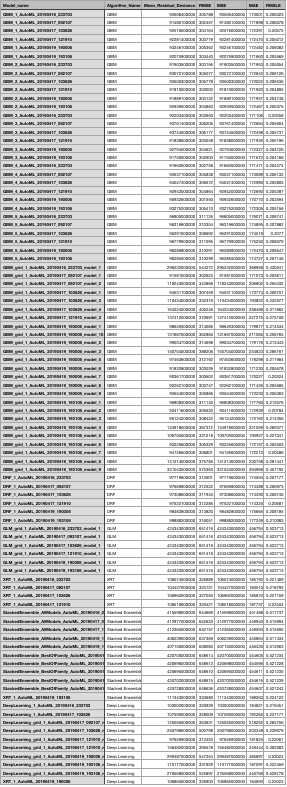
Create view Model\_algo\_info AS

select M.Model\_name, A.Algorithm\_Name, MM.Mean\_Residual\_Deviance, MM.RMSE, MM.MSE, MM.MAE, MM.RMSLE

from Model M

join Algorithm A on M.Algorithm\_ID = A.Algorithm\_ID

join Model\_Metrics MM on MM.Model\_Name = M.Model\_Name;



## 9. Funtions

[Functions](https://github.com/INFO6105-Spring19/hyperparameter-db-project-db08/blob/master/SQL/Hyper_Functions.sql) contain the Views SQL queries.

1) #no of models created for a runtime

DELIMITER $$

CREATE FUNCTION no\_of\_models(runtime INT) RETURNS INT

DETERMINISTIC

BEGIN

DECLARE models INT ;

SET models = ( select Models\_generated FROM Model\_Runtime WHERE Run\_Time =runtime);

RETURN models;

END$$

DELIMITER ;

2) #no of ntrees <50 from DRF

DELIMITER $$

CREATE FUNCTION ntrees(no\_of\_ntrees INT) RETURNS INT

DETERMINISTIC

BEGIN

DECLARE lessthan50 INT ;

SET lessthan50= ( select count(ntrees) FROM DRF\_Hyperparameters WHERE ntrees<=no\_of\_ntrees);

RETURN lessthan50;

END$$

DELIMITER ;

3) #Type of prediction

DELIMITER $$

CREATE FUNCTION dataset\_type(name VARCHAR(255)) RETURNS VARCHAR(255)

DETERMINISTIC

BEGIN

DECLARE dtype VARCHAR(255) ;

SET dtype= ( select Prediction\_Type FROM Dataset WHERE Data\_Set\_Name LIKE name);

RETURN dtype;

END$$

DELIMITER ;

4) #pass datased\_id to get number of columns

DELIMITER $$

CREATE FUNCTION no\_of\_columns(data\_id INT) RETURNS VARCHAR(255)

DETERMINISTIC

BEGIN

DECLARE dcolumn INT ;

SET dcolumn = ( select Total\_Columns FROM Dataset WHERE Dataset\_ID = data\_id);

RETURN dcolumn;

END$$

DELIMITER ;

## 10. Indexes

[Index](https://github.com/INFO6105-Spring19/hyperparameter-db-project-db08/blob/master/SQL/Hyper_Index.sql) contain the Views SQL queries.

USE hyperparameter;

ALTER TABLE Algorithm

ADD INDEX ind\_Algorithm(Algorithm\_ID);

ALTER TABLE Dataset

ADD INDEX ind\_Dataset(Dataset\_ID);

ALTER TABLE DataSet\_Description

ADD INDEX ind\_DataSet\_Description(Column\_ID);

ALTER TABLE DRF\_Hyperparameters

ADD INDEX ind\_DRF\_Hyperparameters(Model\_name);

ALTER TABLE GBM\_Hyperparameters

ADD INDEX ind\_GBM\_Hyperparameters(Model\_name);

ALTER TABLE GLM\_Hyperparameters

ADD INDEX ind\_GLM\_Hyperparameters(Model\_name);

ALTER TABLE XRT\_Hyperparameters

ADD INDEX ind\_XRT\_Hyperparameters(Model\_name);

ALTER TABLE Model

ADD INDEX ind\_Model(Model\_name);

ALTER TABLE Model\_Metrics

ADD INDEX ind\_Model\_Metrics(Model\_name);

ALTER TABLE Model\_Runtime

ADD INDEX ind\_Model\_Runtime(Runtime\_ID);

## 11. Citations and References

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

1. Prof.Brown [Github](https://github.com/nikbearbrown/INFO_6210)
2. [Project Sample](https://github.com/prabhuSub/Hyperparamter-Samples) by Prabhu
3. [JSON Library](https://docs.python.org/3.7/library/json.html)
4. [Data Source](https://www.kaggle.com/anthonypino/melbourne-housing-market)
5. [H2O Documentation](http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html)

## 12. License

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

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

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

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