Info 6210 Project Portfolio

Yuan Chai Hsiang-Hua Chen

Abstract

Hyperparameters are parameters that are specified prior to running machine learning algorithms that have a large effect on the predictive power of statistical models parameter of a prior distribution, the term which used to distinguish them from parameters of the model for the underlying system under analysis. Knowledge of the relative importance of a hyperparameter to an algorithm and its range of values is crucial to hyperparameter tuning and creating effective models.

The hyperparameter database 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, calculating the individual conditional expectation of every hyperparameter on the quality of a model. The data science part need to generating models using H2O to find best hyperparameters.

Background

The data we collected and stored concerns predicting housing transaction price which contains values of cities, floors, unit area households counts and parking capacity, rooms, heat fuel, heat type and front door structure. We separated and grouped data into different entities and attributes and build the one-to-many connections between them, which presented the data in more structured and organized way and allows us to query data, sort data, and manipulate data in various ways for the future performance.

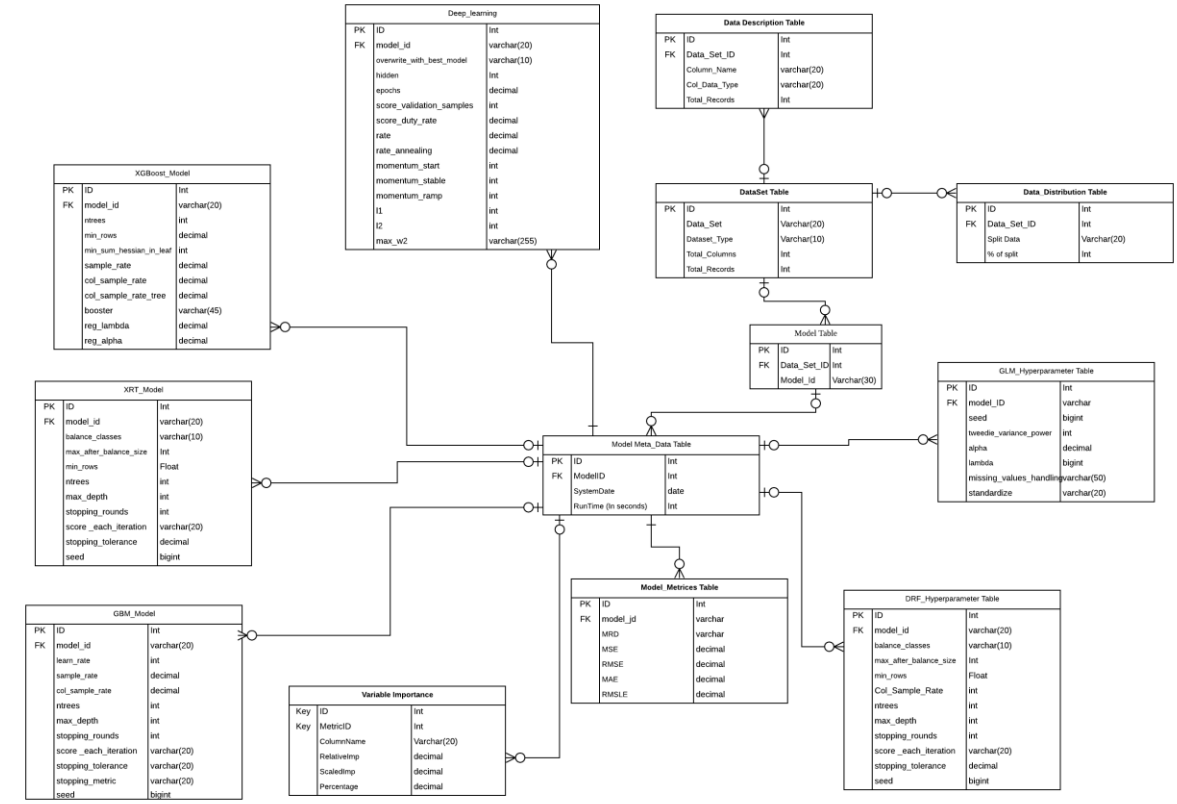
Dataset

The dataset is from the website https://www.kaggle.com/econdata/predciting-price-transaction#trainPrice.csv . Housing price always been a popular item that people expect to predict. Since it is critical for us to find out the factors that affecting transaction price. This data set covers different aspects of factors which influence the housing price, which requires the scientific and specific method to calculate the best result.

ERD

Info 6210 Project Portfolio

Yuan Chai Hsiang-Hua Chen



Normalization

1NF

For all of our tables, We check them one by one and eliminate all the redundant data to ensure there are no repeating groups. We divided Alpha and lambda attributes in GLM Hyperparameter table into atomic as alpha one to seven and lambda one to five. And divided hiddens into hidden one to three in Deep Learning model. We make sure there are no same values in each table.

2NF

We check all the tables that whether there are any functional dependencies on part of any candidate key and make sure there are no partial dependencies.

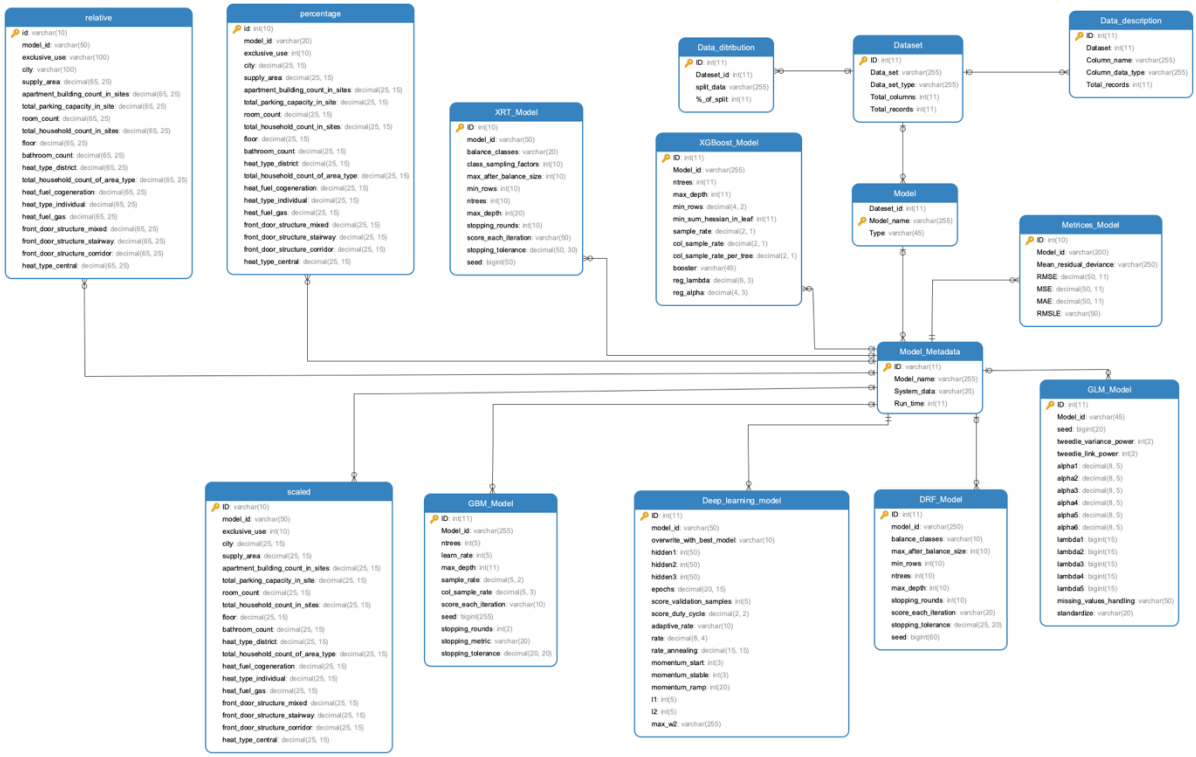
3NF

We check all our tables and make sure there are no non-prime attribute is transitively dependent of any key. All the fields are directly depend on the primary key.

Physical Model

Info 6210 Project Portfolio

Yuan Chai Hsiang-Hua Chen



Use Case

1. Select the best model

```
SELECT me.Model_id, me.RMSE, mm.Model_name, mm.Run_time
FROM Metrices_Model me inner join Model_Metadata mm
ON me.Model_id    = mm.ID
order by RMSE desc
LIMIT 1;
```

2. Select the hyperparameter with the same model but different runtime.

```
SELECT mm.ID, mm.Model_name, mm.Run_time,
me.RMSE,
dl. hidden1, hidden2,hidden3,epochs
FROM Model_Metadata mm left join Deep_learning_model dl
ON mm.ID = dl.model_id
```

```
JOIN Metrices_Model me
ON    dl.model_id = me.Model_id
WHERE mm.ID LIKE "DL%"
ORDER BY mm.Model_name, me.RMSE, mm.Run_time desc;
```

3. Select the average rmse with the same type model

```
SELECT avg(mm.RMSE) as GLM_AVERAGE_rmse
FROM Metrices_Model mm
WHERE mm.Model_id LIKE "G%"
```

4. Select ID, name, runtime and rmse which is higher than the average rmse with the same "XG" ID

```
SELECT metr.ID, metr.Model_id,mm.Model_name, mm.Run_time, metr.RMSE
FROM Metrices_Model metr INNER JOIN Model_Metadata mm
ON metr.Model_id = mm.ID
WHERE metr.Model_id LIKE "XG%"
HAVING metr.RMSE > (
SELECT avg(metr.RMSE)
FROM Metrices_Model metr
WHERE metr.Model_id LIKE "XG%"
)
ORDER BY metr.RMSE DESC;
```

5. Select the counts of models which runtime is 2000

```
SELECT COUNT(*)
FROM Model_Metadata mm
WHERE mm.Run_time = 2000
order BY mm.Run_time
;
```

6. Select the top 10 rmse in XRT model

```
SELECT metr.ID, metr.Model_id,mm.Model_name, mm.Run_time, metr.RMSE
FROM Metrices_Model metr INNER JOIN Model_Metadata mm
ON metr.Model_id = mm.ID
WHERE metr.Model_id LIKE "XRT%"
ORDER BY metr.RMSE
LIMIT 10;
```

7. Select the range of the learning rate of all the model

```
SELECT
MIN(gm.ntrees) AS min_ntrees,
MAX(gm.ntrees) AS max_ntrees,
MIN(gm.max_depth) AS min_max_depth,
MAX(gm.max_depth) AS max_max_depth
FROM GBM_Model gm;
```

8. Select all cities which variable is higher than 0.08

```
SELECT *
FROM percentage pp
having pp.city > 0.08
order by pp.city desc;
```

9. Select all the runtime from the dataset

```
SELECT distinct mm.Run_time
from Model_Metadata mm
order by mm.Run_time;
```

10. Select the type, counts of the type and the best RMSE by order from model data tables

```
SELECT mo.TYPE AS MODEL_TYPE, COUNT(*) AS AMOUNT, MAX(me.RMSE) AS BEST_RMSE
FROM Model_Metadata mm,    Model mo, Metrices_Model me
WHERE mm.Model_name = mo.Model_name AND    mm.ID = me.Model_id
GROUP BY mo.type
ORDER BY max(me.RMSE) desc ;
```

Views

1. Select all the counts of deep learning models in model

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
```

```
    SQL SECURITY DEFINER
VIEW `finall`.`case1` AS
    SELECT
        COUNT(0) AS `COUNT(*)`
    FROM
        `finall`.`model` `m`
    WHERE
        (`m`.`Type` = 'DL')
```

2. Select GLM ID, model name and standardize from metadata by runtime order

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `finall`.`case2` AS
    SELECT
        `mm`.`ID` AS `ID`,
        `mm`.`Model_name` AS `Model_name`,
        `dl`.`standardize` AS `standardize`
    FROM
        (`finall`.`model_metadata` `mm`
        JOIN `finall`.`glm_model` `dl` ON ((`mm`.`ID` = `dl`.`Model_id`)))
    ORDER BY `mm`.`Model_name` , `mm`.`Run_time` DESC
```

3. Select top 20 DL MSE and runtime in metrices model

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `finall`.`case3` AS
    SELECT
        `metr`.`ID` AS `ID`,
        `metr`.`Model_id` AS `Model_id`,
        `mm`.`Model_name` AS `Model_name`,
```

```
        `mm`.`Run_time` AS `Run_time`,
        `metr`.`MSE` AS `MSE`
    FROM
        (`finall`.`metrices_model` `metr`
        JOIN `finall`.`model_metadata` `mm` ON ((`metr`.`Model_id` = `mm`.`ID`)))
    WHERE
        (`metr`.`Model_id` LIKE 'DL%')
    ORDER BY `metr`.`MSE`
    LIMIT 20
```

4. Select average RMSE in DRF model

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `finall`.`case4` AS
    SELECT
        AVG(`mm`.`RMSE`) AS `GLM_AVERAGE_rmse`
    FROM
        `finall`.`metrices_model` `mm`
    WHERE
        (`mm`.`Model_id` LIKE 'DRF%')
```

Functions

1. Get_Average_Stopping_tolerance_in_XRT_Model`

```
CREATE DEFINER=`root`@`localhost` FUNCTION
`get_Average_Stopping_tolerance_in_XRT_Model`(id int) RETURNS varchar(500) CHARSET utf8
BEGIN
  DECLARE a varchar(500);
 -- DECLARE b BIGINT;
 SELECT AVG(stopping_tolerance) INTO a FROM XRT_Model
 WHERE ID = id;

RETURN (a);
END
```

2. get_Deep_learning_model_Epochs

```
        CREATE DEFINER=`root`@`localhost` FUNCTION `get_Deep_learning_model_Epochs`(id
```

```
int) RETURNS varchar(500) CHARSET utf8
    BEGIN
      DECLARE a varchar(500);
     -- DECLARE b BIGINT;
     SELECT epochs INTO a FROM Deep_learning_model
     WHERE ID = id limit 1;

    RETURN (a);
END
```

3. **Get Max_depth_Bigger_Than_Enter_In_XGBoost_Model**

```
    CREATE DEFINER=`root`@`localhost` FUNCTION
`Max_depth_Bigger_Than_Enter_In_XGBoost_Model`(EnteredNum int) RETURNS varchar(50)
CHARSET utf8
    BEGIN
     DECLARE b BIGINT;
     SELECT count(max_depth) INTO b FROM XGBoost_Model
     WHERE max_depth > EnteredNum;

    RETURN (b);
END
```

4. **Type_Max_Get_Max_MAE_In_Metrices_Model**

```
CREATE DEFINER=`root`@`localhost` FUNCTION
`Type_Max_Get_Max_MAE_In_Metrices_Model`(Enter VARCHAR(50)) RETURNS varchar(500)
CHARSET utf8
BEGIN

  DECLARE namemodel VARCHAR(500);

  SELECT max(MAE) INTO namemodel FROM Metrices_Model
  where maxx = "max";

  RETURN namemodel;
END
```

Info 6210 Project Portfolio

Yuan Chai Hsiang-Hua Chen

Analytics & Conclusions

By storing Hyperparameters data set in the database enables us obtain the structural and organized data to call the different functions for analyzing and select the best model for prediction, which make it more visualized to check and use the data and achieve different utilization. By creating the use cases, functions and views, we can select single or combined date set, get the best model, calculate the average or the max data for improving the different performance.

Citations

https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/
https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)
https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a
https://towardsdatascience.com/hyperparameters-in-deep-learning-927f7b2084dd
https://www.w3schools.com/sql/sql_create_index.asp
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql?view=sql-server-2017
https://www.w3schools.com/sql/sql_view.asp

License

Info 6210 Project Portfolio

Yuan Chai Hsiang-Hua Chen