HYPERPARAMETERS DB

I409076
HSIANG-HUA CHEN

DATASET

- Housing price always been a popular item that people wants to predict. Since it is critical for us to find out the factors that affecting transaction price. 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.
- And from the processing of data, I found out the supply_area is most related to the transaction price.

FINISHED

- ✓Clean the data
- ✓Denote the data into H2O
- ✓Run H2O AutoML for different runtime (300, 500, 1000, 1500, 2000 seconds)
- ✓Store the best model of each runtime
- ✓Store the Leaderboard, Hyperparameter, Variable Importance from each model for every run

```python
def board_to_csv(board, runtime):
    board_csv = board.as_data_frame()
    system_date = datetime.date.today()
    board_csv['system_date'] = system_date
    board_csv['runtime'] = runtime
    print ('board_to_csv done')
    return board_csv
```

```python
def get_modelList(board_csv):
    model_list = []
    for index, row in board_csv.iterrows():
        model_list.append(row['model_id'])
    return model_list
```

```python
def get_all_params(board_csv):
    all_params = []
    model_list = get_modelList(board_csv)
    for i in model_list:
        print (i)
        model = h2o.get_model(i)
        params = model.params
        all_params.append(params)
    print ("get_all_params done")
    print ('model_list : ', len(model_list))
    print ('all_params : ', len(all_params))
    return all_params
```

```python
def get_leaderBoard(runtime):
    aml = H2OAutoML(max_runtime_secs=runtime,
                    project_name = project)
    aml.train(x=X, y=y, training_frame=df)

    board = aml.leaderboard
    print ('get_leaderBoard done')
    return board
```

```python
def get_BestModel(board_csv):
    id = board_csv['model_id'][0]
    best_model = h2o.get_model(id)
    return best_model
```

```python
def get_all_varimp(board_csv):
    model_list = get_modelList(board_csv)
    tup=[]
    gg = 1
    for mid in model_list:
        model = h2o.get_model(mid)
        varimp = model.varimp()
        print("tryyyyyyyyyyyyyyyy          ", gg)
        gg+= 1
        try:
            for var_item in varimp:
                vv = []
                vv.append(mid)
                ass = []
                for tit in var_item:
                    ass.append(tit)
                item = vv + ass
                tup.append(item)

            print('done')
        except:
            print(mid)
            print('pass')
            pass
        continue

    new_varimp = pd.DataFrame(tup, columns = ['model_id',
                             'variable',
                             'relative_importance',
                             'scaled_importance',
                             'percentage'
                            ])

    return new_varimp
```

```python
runtime = 300
leaderBoard = get_leaderBoard(runtime)

board_csv = board_to_csv(leaderBoard, runtime)
board_csv.to_csv('result/300/leaderboard.csv', sep='\t')

params =  get_all_params(board_csv)
with open('result/300/params.json', 'w') as f:
    json.dump(params, f)

all_varimp = get_all_varimp(board_csv)
all_varimp.to_csv('result/300/all_varimp.csv', sep='\t')
```

```
AutoML progress: |████████████████████████████████████████████████████████| 100%
get_leaderBoard done
board_to_csv done
GBM_1_AutoML_20190416_015849
XGBoost_1_AutoML_20190416_015849
XGBoost_grid_1_AutoML_20190416_020809_model_4
GBM_1_AutoML_20190416_020809
XGBoost_1_AutoML_20190416_020809
XGBoost_grid_1_AutoML_20190416_020809_model_7
XGBoost_2_AutoML_20190416_015849
XGBoost_grid_1_AutoML_20190416_015849_model_3
GBM_2_AutoML_20190416_020809
GBM_grid_1_AutoML_20190416_015849_model_7
GBM_4_AutoML_20190416_015849
XGBoost_2_AutoML_20190416_020809
GBM_4_AutoML_20190416_020809
XRT_1_AutoML_20190416_020809
GBM_3_AutoML_20190416_020809
DRF_1_AutoML_20190416_020809
XGBoost_grid_1_AutoML_20190416_015849_model_4
XRT_1_AutoML_20190416_015849
XGBoost_grid_1_AutoML_20190416_020809_model_2
GBM_3_AutoML_20190416_015849
XGBoost_grid_1_AutoML_20190416_015849_model_1
GBM_2_AutoML_20190416_015849
```

```
bestModel_300 = get_BestModel(board_csv)
bestModel_300
```

```
Model Details
=============
H2OGradientBoostingEstimator :  Gradient Boosting Machine
Model Key:  GBM_1_AutoML_20190416_015849


ModelMetricsRegression: gbm
** Reported on train data. **

MSE: 9064188861621198.0
RMSE: 95206033.74587767
MAE: 67496135.99104144
RMSLE: 0.25946421720441404
Mean Residual Deviance: 9064188861621198.0

ModelMetricsRegression: gbm
** Reported on cross-validation data. **

MSE: 2.5146739460939084e+16
RMSE: 158577235.00218776
MAE: 96028946.28873461
RMSLE: 0.3325625379269681
Mean Residual Deviance: 2.5146739460939084e+16
Cross-Validation Metrics Summary:
```

	mean	sd	cv_1_valid	cv_2_valid
mae	96028944.0000000	3612221.8	100648072.0000000	89991768.0
mean_residual_deviance	25146739300000000.0000000	3702095920000000.0000000	27711208500000000.0000000	189316811(
mse	25146739300000000.0000000	3702095920000000.0000000	27711208500000000.0000000	189316811(
r2	0.7517724	0.0152310	0.7486387	0.7622951
residual_deviance	25146739300000000.0000000	3702095920000000.0000000	27711208500000000.0000000	189316811(
rmse	157673632.0000000	11953314.0000000	166466832.0000000	137592448.(
rmsle	0.3324099	0.0071243	0.3404068	0.3175372

```
Scoring History:
```

TO DO LIST

- Analyze the best model from each run
- Compare the different best model of different runtime
- Find the best model
- Make a conclusion
- Complete the document