02 Statistical modeling

Author: Miao Cai miao.cai@slu.edu

Statistical modeling

We then use four different models to model the risk during the trip:

- Logistic regression
- Poisson regression
- XGBoost
- Deep learning (Neural networks)

import packages and read data

```
# !pip install h2o
import sys
import numpy as np
import h2o
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
h2o.init(nthreads = -1, max_mem_size = 8)

print("Python version: " + sys.version)
print("numpy version:", np.__version__)
print("h2o version:", h2o.__version__)
```

```
Checking whether there is an H2O instance running at http://localhost:54321 ..... not fo
Attempting to start a local H2O server...
  Java Version: openjdk version "11.0.4" 2019-07-16; OpenJDK Runtime Environment (build
  Starting server from /usr/local/lib/python3.6/dist-packages/h2o/backend/bin/h2o.jar
  Ice root: /tmp/tmp294_9azi
  JVM stdout: /tmp/tmp294_9azi/h2o_unknownUser_started_from_python.out
  JVM stderr: /tmp/tmp294_9azi/h2o_unknownUser_started_from_python.err
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321 ... successful.
```

H2O cluster uptime:	02 secs
H2O cluster timezone:	Etc/UTC
H2O data parsing timezone:	UTC
H2O cluster version:	3.26.0.11
H2O cluster version age:	11 days
H2O cluster name:	H2O_from_python_unknownUser_rtskve
H2O cluster total nodes:	1
H2O cluster free memory:	8 Gb

```
df = h2o.import_file('https://raw.githubusercontent.com/caimiao0714/optimization_stats_cas
df[df['y']  > 0,'y_binary'] = 1
df[df['y'] == 0,'y_binary'] = 0
df['y_binary'] = df['y_binary'].asfactor()
df['log_Distance'] = df['Distance'].log()
df.head(5)
```

```
Parse progress: |████████████████████████████████████████████████████████| 100%
```

	C1	y	Distance	Precipitation	Traffic	y_binary	log_Distance
0	0		1018	0	0.299886	0	6.9256
1	0		973	0	0.565617	0	6.88038
2	0		1021	0	0.414564	0	6.92854
3	0		998	0	0.559767	0	6.90575
4	0		985	0	0.777217	0	6.89264

```
lk = h2o.import_file('https://raw.githubusercontent.com/caimiao0714/optimization_stats_cas
lk.head(5)
```

```
2 tk.head(5)
```

```
Parse progress: |████████████████████████████████████████████████████████| 100%
```

	C1	# Node A	Node Z	Distance	Precipitation	Traffic
0		Ann_Arbor	Ithaca	800	0	0.254345
1		Ann_Arbor	Princeton	800	0	0.243435
2		Ann_Arbor	Salt_Lake_City	2400	0	0.254188
3		Atlanta	Houston	1200	0	0.424037
4		Atlanta	Pittsburgh	900	0	0.573477

Split into train and test sets

```
1 df_splits = df.split_frame(ratios = [0.7, 0.15], seed = 123)
2
3 df_train = df_splits[0]
4 df_test  = df_splits[1]
5 df_valid = df_splits[2]
6
7 print(str(df_train.nrow) + " rows in training set;\n" +
8       str(df_test.nrow) + " rows in test set;\n" +
9       str(df_valid.nrow) + " rows in validation set.")
```

```
7021 rows in training set;
1482 rows in test set;
1497 rows in validation set.
```

Logistic regression

```
1 fit_logit = H2OGeneralizedLinearEstimator(family='binomial',
2                                           model_id='fit_logit')
3 fit_logit.train(x = ['Precipitation', 'Traffic', 'Distance'],
4                 y = 'y_binary',
5                 training_frame = df_train)
6 logit_test_fit = fit_logit.model_performance(df_test)
7 fit_logit._model_json['output']['coefficients_table']
```

```
glm Model Build progress: |████████████████████████████████████████████████| 100%
```

Coefficients: glm coefficients

	names	coefficients	standardized_coefficients
0	Intercept	-3.604438	-2.144072
1	Distance	0.001008	0.032266
2	Precipitation	0.256380	0.092045
3	Traffic	0.830543	0.187609

```
print("Logistic regression model evaluation:")
print("train AUC: " + str(fit_logit.auc()))
print("test  AUC: " + str(logit_test_fit.auc()))
print("---")
print("train Accuracy" + str(fit_logit.accuracy()))
print("test  Accuracy" + str(logit_test_fit.accuracy()))
print("---")
print("train MSE" + str(fit_logit.mse()))
print("test  MSE" + str(logit_test_fit.mse()))
print("---")
print("train R-square: " + str(fit_logit.r2()))
print("test  R-square: " + str(logit_test_fit.r2()))
```

```
Logistic regression model evaluation:
train AUC: 0.5596289078650459
test  AUC: 0.5638801871833545
---
train Accuracy[[0.18502530292639074, 0.8936048995869534]]
test  Accuracy[[0.17768208465318328, 0.8940620782726046]]
---
train MSE0.09478002969662627
test  MSE0.09376565320196198
---
train R-square: 0.004278462347631851
test  R-square: 0.004429388061521045
```

Poisson regression

```
fit_poisson = H2OGeneralizedLinearEstimator(family='Poisson',
                                            model_id='fit_poisson')
fit_poisson.train(x = ['Precipitation', 'Traffic', 'Distance'],
                  #offset_column = 'Distance',
                  y = 'y',
                  training_frame = df_train)
poisson_test_fit = fit_poisson.model_performance(df_test)
fit_poisson._model_json['output']['coefficients_table']
```

```
glm Model Build progress: |████████████████████████████████████████████████| 100%
```

Coefficients: glm coefficients

	names	coefficients	standardized_coefficients
0	Intercept	-4.371852	-2.102051
1	Distance	0.001747	0.055937
2	Precipitation	0.334264	0.120008
3	Traffic	0.947354	0.213995

```
print("Poisson regression model evaluation:")
print("train MSE: " + str(fit_poisson.mse()))
print("test  MSE: " + str(poisson_test_fit.mse()))
print("---")
print("train R-square: " + str(fit_poisson.r2()))
print("test  R-square: " + str(poisson_test_fit.r2()))
```

```
Poisson regression model evaluation:
train MSE: 0.16471763615686122
test  MSE: 0.17174430698927012
---
train R-square: 0.006623137684569902
test  R-square: 0.0043653505149615635
```

XGBoost

```
from h2o.estimators import H2OXGBoostEstimator
xgboost_params = {
      "ntrees" : 50,
      "max_depth" : 5,
      "learn_rate" : 0.001,
      "sample_rate" : 0.7,
      "col_sample_rate_per_tree" : 0.9,
      "min_rows" : 5,
      "seed": 4241,
      "score_tree_interval": 10
}
fit_xgboost = H2OXGBoostEstimator(**xgboost_params)
fit_xgboost.train(x = ['Precipitation', 'Traffic', 'Distance'],
                  y = 'y_binary',
                  training_frame = df_train,
                  validation_frame = df_valid)
```

```
xgboost Model Build progress: |████████████████████████████████████████| 100%
```

```
xgboost_test_fit = fit_xgboost.model_performance(df_test)
print("XGBoost regression model evaluation:")
print("train AUC: " + str(fit_xgboost.auc()))
print("test  AUC: " + str(xgboost_test_fit.auc()))
print("---")
print("train Accuracy" + str(fit_xgboost.accuracy()))
print("test  Accuracy" + str(xgboost_test_fit.accuracy()))
print("---")
print("train MSE" + str(fit_xgboost.mse()))
print("test  MSE" + str(xgboost_test_fit.mse()))
print("---")
print("train R-square: " + str(fit_xgboost.r2()))
print("test  R-square: " + str(xgboost_test_fit.r2()))
```

```
XGBoost regression model evaluation:
train AUC: 0.6058716756560456
test  AUC: 0.552575221410063
---
train Accuracy[[0.48714303970336914, 0.8937473294402507]]
test  Accuracy[[0.4882924258708954, 0.8940620782726046]]
---
train MSE0.2352391414490155
test  MSE0.2352131798112998
---
train R-square: -1.4713294603237945
test  R-square: -1.4974105268199778
```

Neural networks

```
from h2o.estimators.deeplearning import H2OAutoEncoderEstimator, H2ODeepLearningEstimator
```

```
fit_DL = H2ODeepLearningEstimator(epochs = 1000,
                                  # hidden = [10, 10],
                                  model_id = 'Deep learning',
                                  seed = 1)
fit_DL.train(x = ['Precipitation', 'Traffic', 'Distance'],
             y = 'y_binary',
             training_frame = df_train,
             validation_frame = df_valid)
```

```
deeplearning Model Build progress: |████████████████████████████████████████| 100%
```

```
DL_test_fit = fit_DL.model_performance(df_test)
print("Deep learning model evaluation:")
print("train AUC: " + str(fit_DL.auc()))
print("test  AUC: " + str(DL_test_fit.auc()))
print("---")
print("train Accuracy" + str(fit_DL.accuracy()))
print("test  Accuracy" + str(DL_test_fit.accuracy()))
print("---")
print("train MSE" + str(fit_DL.mse()))
print("test  MSE" + str(DL_test_fit.mse()))
print("---")
print("train R-square: " + str(fit_DL.r2()))
print("test  R-square: " + str(DL_test_fit.r2()))
```

```
Deep learning model evaluation:
train AUC: 0.5614672763588284
test  AUC: 0.5351742274819198
---
train Accuracy[[0.21823377000378666, 0.8933200398803589]]
test  Accuracy[[0.2164417325947588, 0.8940620782726046]]
---
train MSE0.09477659936866129
test  MSE0.09432099305838623
---
train R-square: 0.0043145000176650905
test  R-square: -0.0014670145316892924
```

Prediction for links data

```
risk_logit = fit_logit.predict(lk).as_data_frame(True).p1.tolist()
risk_poisson = fit_poisson.predict(lk).as_data_frame(True).predict.tolist()
risk_xgboost = fit_xgboost.predict(lk).as_data_frame(True).p1.tolist()
risk_DL = fit_DL.predict(lk).as_data_frame(True).p1.tolist()

lk_risks = lk.cbind(h2o.H2OFrame(risk_logit).set_names(['risk_logit'])).\
             cbind(h2o.H2OFrame(risk_poisson).set_names(['risk_poisson'])).\
             cbind(h2o.H2OFrame(risk_xgboost).set_names(['risk_xgboost'])).\
             cbind(h2o.H2OFrame(risk_DL).set_names(['risk_DL']))
lk_risks.head(5)
```

```
glm prediction progress: |████████████████████████████████████████| 100%
glm prediction progress: |████████████████████████████████████████| 100%
xgboost prediction progress: |████████████████████████████████████| 100%
deeplearning prediction progress: |███████████████████████████████| 100%
Parse progress: |█████████████████████████████████████████████████| 100%
Parse progress: |█████████████████████████████████████████████████| 100%
Parse progress: |█████████████████████████████████████████████████| 100%
Parse progress: |█████████████████████████████████████████████████| 100%
```

C1	# Node A	Node Z	Distance	Precipitation	Traffic	risk_logit	risk_poisson	risk_xgboost	
0	Ann_Arbor	Ithaca	800	0	0.254345	0.0699758	0.0650006	0.480249	0.
1	Ann_Arbor	Princeton	800	0	0.243435	0.0693884	0.0643322	0.480249	0.
2	Ann_Arbor	Salt_Lake_City	2400	0	0.254188	0.2739	1.0635	0.482543	8.
3	Atlanta	Houston	1200	0	0.424037	0.114756	0.153534	0.482781	
4	Atlanta	Pittsburgh	900	0	0.573477	0.0978591	0.104734	0.480885	

```
from google.colab import drive
from google.colab import files
drive.mount('/content/drive/', force_remount=True)
lk_risks.as_data_frame().to_csv('lk_risks.csv')
files.download('lk_risks.csv')
```

```
Mounted at /content/drive/
```