This tutorial was taken from an H2O tutorial online: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/starting-h2o.html>

In [1]:

import h2o

from h2o.estimators.gbm import H2OGradientBoostingEstimator

h2o.init(max\_mem\_size=4)

Checking whether there is an H2O instance running at http://localhost:54321 ..... not found.

Attempting to start a local H2O server...

; Java HotSpot(TM) 64-Bit Server VM (build 13.0.2+8, mixed mode, sharing)

Starting server from c:\users\m\anaconda2\envs\python36\lib\site-packages\h2o\backend\bin\h2o.jar

Ice root: C:\Users\m\AppData\Local\Temp\tmp8r46t\_2b

JVM stdout: C:\Users\m\AppData\Local\Temp\tmp8r46t\_2b\h2o\_m\_started\_from\_python.out

JVM stderr: C:\Users\m\AppData\Local\Temp\tmp8r46t\_2b\h2o\_m\_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: | 09 secs |
| H2O cluster timezone: | America/Los\_Angeles |
| H2O data parsing timezone: | UTC |
| H2O cluster version: | 3.28.0.2 |
| H2O cluster version age: | 14 days, 20 hours and 35 minutes |
| H2O cluster name: | H2O\_from\_python\_m\_6gh42v |
| H2O cluster total nodes: | 1 |
| H2O cluster free memory: | 4 Gb |
| H2O cluster total cores: | 2 |
| H2O cluster allowed cores: | 2 |
| H2O cluster status: | locked, healthy |
| H2O connection url: | http://127.0.0.1:54321 |
| H2O connection proxy: | {'http': None, 'https': None} |
| H2O internal security: | False |
| H2O API Extensions: | Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4 |
| Python version: | 3.6.9 final |

This data is from kaggle when googling 'airline data h2o' because the tutorial file was not a valid web page.

In [2]:

flights2 = h2o.import\_file("flights.csv")

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

In [3]:

flights2

flights2.shape

Out[3]:

(5819079, 31)

In [4]:

flights2.head()

| **YEAR** | **MONTH** | **DAY** | **DAY\_OF\_WEEK** | **AIRLINE** | **FLIGHT\_NUMBER** | **TAIL\_NUMBER** | **ORIGIN\_AIRPORT** | **DESTINATION\_AIRPORT** | **SCHEDULED\_DEPARTURE** | **DEPARTURE\_TIME** | **DEPARTURE\_DELAY** | **TAXI\_OUT** | **WHEELS\_OFF** | **SCHEDULED\_TIME** | **ELAPSED\_TIME** | **AIR\_TIME** | **DISTANCE** | **WHEELS\_ON** | **TAXI\_IN** | **SCHEDULED\_ARRIVAL** | **ARRIVAL\_TIME** | **ARRIVAL\_DELAY** | **DIVERTED** | **CANCELLED** | **CANCELLATION\_REASON** | **AIR\_SYSTEM\_DELAY** | **SECURITY\_DELAY** | **AIRLINE\_DELAY** | **LATE\_AIRCRAFT\_DELAY** | **WEATHER\_DELAY** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2015 | 1 | 1 | 4 | AS | 98 | N407AS | ANC | SEA | 5 | 2354 | -11 | 21 | 15 | 205 | 194 | 169 | 1448 | 404 | 4 | 430 | 408 | -22 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | AA | 2336 | N3KUAA | LAX | PBI | 10 | 2 | -8 | 12 | 14 | 280 | 279 | 263 | 2330 | 737 | 4 | 750 | 741 | -9 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | US | 840 | N171US | SFO | CLT | 20 | 18 | -2 | 16 | 34 | 286 | 293 | 266 | 2296 | 800 | 11 | 806 | 811 | 5 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | AA | 258 | N3HYAA | LAX | MIA | 20 | 15 | -5 | 15 | 30 | 285 | 281 | 258 | 2342 | 748 | 8 | 805 | 756 | -9 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | AS | 135 | N527AS | SEA | ANC | 25 | 24 | -1 | 11 | 35 | 235 | 215 | 199 | 1448 | 254 | 5 | 320 | 259 | -21 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | DL | 806 | N3730B | SFO | MSP | 25 | 20 | -5 | 18 | 38 | 217 | 230 | 206 | 1589 | 604 | 6 | 602 | 610 | 8 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | NK | 612 | N635NK | LAS | MSP | 25 | 19 | -6 | 11 | 30 | 181 | 170 | 154 | 1299 | 504 | 5 | 526 | 509 | -17 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | US | 2013 | N584UW | LAX | CLT | 30 | 44 | 14 | 13 | 57 | 273 | 249 | 228 | 2125 | 745 | 8 | 803 | 753 | -10 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | AA | 1112 | N3LAAA | SFO | DFW | 30 | 19 | -11 | 17 | 36 | 195 | 193 | 173 | 1464 | 529 | 3 | 545 | 532 | -13 | 0 | 0 |  | nan | nan | nan | nan | nan |
| 2015 | 1 | 1 | 4 | DL | 1173 | N826DN | LAS | ATL | 30 | 33 | 3 | 12 | 45 | 221 | 203 | 186 | 1747 | 651 | 5 | 711 | 656 | -15 | 0 | 0 |  | nan | nan | nan | nan | nan |

Out[4]:

In [5]:

flights2.columns

Out[5]:

['YEAR',

'MONTH',

'DAY',

'DAY\_OF\_WEEK',

'AIRLINE',

'FLIGHT\_NUMBER',

'TAIL\_NUMBER',

'ORIGIN\_AIRPORT',

'DESTINATION\_AIRPORT',

'SCHEDULED\_DEPARTURE',

'DEPARTURE\_TIME',

'DEPARTURE\_DELAY',

'TAXI\_OUT',

'WHEELS\_OFF',

'SCHEDULED\_TIME',

'ELAPSED\_TIME',

'AIR\_TIME',

'DISTANCE',

'WHEELS\_ON',

'TAXI\_IN',

'SCHEDULED\_ARRIVAL',

'ARRIVAL\_TIME',

'ARRIVAL\_DELAY',

'DIVERTED',

'CANCELLED',

'CANCELLATION\_REASON',

'AIR\_SYSTEM\_DELAY',

'SECURITY\_DELAY',

'AIRLINE\_DELAY',

'LATE\_AIRCRAFT\_DELAY',

'WEATHER\_DELAY']

In [6]:

flights2['YEAR'] = flights2['YEAR'].asfactor()

flights2['MONTH'] = flights2['MONTH'].asfactor()

flights2['DAY\_OF\_WEEK'] = flights2['DAY\_OF\_WEEK'].asfactor()

flights2['FLIGHT\_NUMBER'] = flights2['FLIGHT\_NUMBER'].asfactor()

flights2['CANCELLED'] = flights2['CANCELLED'].asfactor()

#flights2['DEPARTURE\_DELAY'] = flights2['DEPARTURE\_DELAY'].asfactor()

In [7]:

predictors = ['YEAR', 'ORIGIN\_AIRPORT','DESTINATION\_AIRPORT','MONTH', 'DAY\_OF\_WEEK',

'FLIGHT\_NUMBER','DISTANCE','AIRLINE']

response = 'DEPARTURE\_DELAY'

In [8]:

train, valid = flights2.split\_frame(ratios=[0.8], seed=1234)

In [9]:

bin\_num = [8,16,32,64,128,256,512,1024,2048,4096]

label = ["8","16","32","64","128","256","512","1024","2048","4096"]

The next command shows the attributes available in the H2OGradientBoostingEstimator function used to train the GBM model and test on the validation set with.

In [10]:

dir(H2OGradientBoostingEstimator)

Out[10]:

['\_ModelBase\_\_generate\_partial\_plots',

'\_ModelBase\_\_generate\_user\_splits',

'\_ModelBase\_\_grabValues',

'\_ModelBase\_\_plot\_1dpdp',

'\_ModelBase\_\_plot\_2dpdp',

'\_ModelBase\_\_predFor3D',

'\_ModelBase\_\_setAxs1D',

'\_\_class\_\_',

'\_\_delattr\_\_',

'\_\_dict\_\_',

'\_\_dir\_\_',

'\_\_doc\_\_',

'\_\_eq\_\_',

'\_\_format\_\_',

'\_\_ge\_\_',

'\_\_getattr\_\_',

'\_\_getattribute\_\_',

'\_\_gt\_\_',

'\_\_hash\_\_',

'\_\_init\_\_',

'\_\_init\_subclass\_\_',

'\_\_le\_\_',

'\_\_lt\_\_',

'\_\_module\_\_',

'\_\_ne\_\_',

'\_\_new\_\_',

'\_\_reduce\_\_',

'\_\_reduce\_ex\_\_',

'\_\_repr\_\_',

'\_\_setattr\_\_',

'\_\_sizeof\_\_',

'\_\_str\_\_',

'\_\_subclasshook\_\_',

'\_\_weakref\_\_',

'\_additional\_used\_columns',

'\_bc',

'\_check\_and\_save\_parm',

'\_check\_targets',

'\_compute\_algo',

'\_get\_metrics',

'\_keyify\_if\_h2oframe',

'\_metrics\_class',

'\_plot',

'\_print\_model\_scoring\_history',

'\_requires\_training\_frame',

'\_resolve\_model',

'\_train',

'\_verify\_training\_frame\_params',

'actual\_params',

'aic',

'algo',

'auc',

'aucpr',

'balance\_classes',

'biases',

'build\_tree\_one\_node',

'calibrate\_model',

'calibration\_frame',

'categorical\_encoding',

'catoffsets',

'check\_constant\_response',

'checkpoint',

'class\_sampling\_factors',

'coef',

'coef\_norm',

'col\_sample\_rate',

'col\_sample\_rate\_change\_per\_level',

'col\_sample\_rate\_per\_tree',

'convert\_H2OXGBoostParams\_2\_XGBoostParams',

'cross\_validation\_fold\_assignment',

'cross\_validation\_holdout\_predictions',

'cross\_validation\_metrics\_summary',

'cross\_validation\_models',

'cross\_validation\_predictions',

'custom\_distribution\_func',

'custom\_metric\_func',

'deepfeatures',

'default\_params',

'detach',

'distribution',

'download\_model',

'download\_mojo',

'download\_pojo',

'end\_time',

'export\_checkpoints\_dir',

'feature\_frequencies',

'fit',

'fold\_assignment',

'fold\_column',

'full\_parameters',

'get\_params',

'get\_xval\_models',

'gini',

'have\_mojo',

'have\_pojo',

'histogram\_type',

'huber\_alpha',

'ignore\_const\_cols',

'ignored\_columns',

'is\_cross\_validated',

'join',

'keep\_cross\_validation\_fold\_assignment',

'keep\_cross\_validation\_models',

'keep\_cross\_validation\_predictions',

'key',

'learn\_rate',

'learn\_rate\_annealing',

'logloss',

'mae',

'max\_abs\_leafnode\_pred',

'max\_after\_balance\_size',

'max\_confusion\_matrix\_size',

'max\_depth',

'max\_hit\_ratio\_k',

'max\_runtime\_secs',

'mean\_residual\_deviance',

'min\_rows',

'min\_split\_improvement',

'mixin',

'model\_id',

'model\_performance',

'monotone\_constraints',

'mse',

'nbins',

'nbins\_cats',

'nbins\_top\_level',

'nfolds',

'normmul',

'normsub',

'ntrees',

'ntrees\_actual',

'null\_degrees\_of\_freedom',

'null\_deviance',

'offset\_column',

'param\_names',

'params',

'partial\_plot',

'pprint\_coef',

'pr\_auc',

'pred\_noise\_bandwidth',

'predict',

'predict\_contributions',

'predict\_leaf\_node\_assignment',

'quantile\_alpha',

'r2',

'r2\_stopping',

'residual\_degrees\_of\_freedom',

'residual\_deviance',

'respmul',

'response\_column',

'respsub',

'rmse',

'rmsle',

'rotation',

'run\_time',

'sample\_rate',

'sample\_rate\_per\_class',

'save\_model\_details',

'save\_mojo',

'score\_each\_iteration',

'score\_history',

'score\_tree\_interval',

'scoring\_history',

'seed',

'set\_params',

'show',

'staged\_predict\_proba',

'start',

'start\_time',

'std\_coef\_plot',

'stopping\_metric',

'stopping\_rounds',

'stopping\_tolerance',

'summary',

'train',

'training\_frame',

'training\_model\_metrics',

'tweedie\_power',

'type',

'validation\_frame',

'varimp',

'varimp\_plot',

'weights',

'weights\_column',

'xval\_keys',

'xvals']

The time() give the UTC amount of seconds that have elapsed in floating point values.

In [11]:

import time

start = time.time()

for key, num in enumerate(bin\_num):

flights2\_gbm = H2OGradientBoostingEstimator(nbins\_cats=num, seed=1234)

flights2\_gbm.train(x=predictors, y=response, training\_frame=train, validation\_frame=valid)

end = time.time()

predictionTime = (end-start)

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

gbm Model Build progress: |███████████████████████████████████████████████| 100%

This is an alternate way of reading in the file for python 3.6

In [12]:

flights2\_gbm

Model Details

=============

H2OGradientBoostingEstimator : Gradient Boosting Machine

Model Key: GBM\_model\_python\_1580859444109\_10

Model Summary:

|  |  | **number\_of\_trees** | **number\_of\_internal\_trees** | **model\_size\_in\_bytes** | **min\_depth** | **max\_depth** | **mean\_depth** | **min\_leaves** | **max\_leaves** | **mean\_leaves** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  | 50.0 | 50.0 | 110678.0 | 5.0 | 5.0 | 5.0 | 25.0 | 32.0 | 31.86 |

ModelMetricsRegression: gbm

\*\* Reported on train data. \*\*

MSE: 1329.2825288940203

RMSE: 36.459327049385045

MAE: 18.118516320527252

RMSLE: NaN

Mean Residual Deviance: 1329.2825288940203

ModelMetricsRegression: gbm

\*\* Reported on validation data. \*\*

MSE: 1353.0947850896957

RMSE: 36.784436723833295

MAE: 18.204540718600583

RMSLE: NaN

Mean Residual Deviance: 1353.0947850896957

Scoring History:

|  |  | **timestamp** | **duration** | **number\_of\_trees** | **training\_rmse** | **training\_mae** | **training\_deviance** | **validation\_rmse** | **validation\_mae** | **validation\_deviance** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  | 2020-02-04 16:22:55 | 0.016 sec | 0.0 | 37.031308 | 18.764566 | 1371.317777 | 37.278862 | 18.815101 | 1389.713527 |
| 1 |  | 2020-02-04 16:23:10 | 15.642 sec | 3.0 | 36.896040 | 18.616960 | 1361.317750 | 37.148286 | 18.670488 | 1379.995136 |
| 2 |  | 2020-02-04 16:23:30 | 35.627 sec | 7.0 | 36.782714 | 18.487281 | 1352.968056 | 37.043504 | 18.545082 | 1372.221222 |
| 3 |  | 2020-02-04 16:23:55 | 1 min 0.051 sec | 12.0 | 36.701522 | 18.389381 | 1347.001735 | 36.970735 | 18.451143 | 1366.835236 |
| 4 |  | 2020-02-04 16:24:20 | 1 min 25.398 sec | 17.0 | 36.646788 | 18.324858 | 1342.987095 | 36.923406 | 18.389741 | 1363.337878 |
| 5 |  | 2020-02-04 16:24:47 | 1 min 52.681 sec | 22.0 | 36.605118 | 18.275130 | 1339.934657 | 36.889600 | 18.343364 | 1360.842615 |
| 6 |  | 2020-02-04 16:25:13 | 2 min 18.714 sec | 27.0 | 36.573955 | 18.239270 | 1337.654157 | 36.865003 | 18.310246 | 1359.028479 |
| 7 |  | 2020-02-04 16:25:48 | 2 min 53.146 sec | 34.0 | 36.529584 | 18.190719 | 1334.410508 | 36.831864 | 18.267077 | 1356.586210 |
| 8 |  | 2020-02-04 16:26:19 | 3 min 24.133 sec | 40.0 | 36.497495 | 18.156603 | 1332.067112 | 36.808323 | 18.236518 | 1354.852620 |
| 9 |  | 2020-02-04 16:26:49 | 3 min 54.804 sec | 46.0 | 36.473456 | 18.131413 | 1330.312994 | 36.792856 | 18.214934 | 1353.714281 |
| 10 |  | 2020-02-04 16:27:09 | 4 min 14.821 sec | 50.0 | 36.459327 | 18.118516 | 1329.282529 | 36.784437 | 18.204541 | 1353.094785 |

Variable Importances:

|  | **variable** | **relative\_importance** | **scaled\_importance** | **percentage** |
| --- | --- | --- | --- | --- |
| 0 | ORIGIN\_AIRPORT | 319118368.0 | 1.000000 | 0.320754 |
| 1 | MONTH | 198638896.0 | 0.622461 | 0.199657 |
| 2 | DESTINATION\_AIRPORT | 194843408.0 | 0.610568 | 0.195842 |
| 3 | AIRLINE | 132232376.0 | 0.414368 | 0.132910 |
| 4 | FLIGHT\_NUMBER | 76783288.0 | 0.240611 | 0.077177 |
| 5 | DAY\_OF\_WEEK | 67829360.0 | 0.212552 | 0.068177 |
| 6 | DISTANCE | 5454050.5 | 0.017091 | 0.005482 |

Out[12]:

In [26]:

predictionTime

Out[26]:

2393.457242488861

In [61]:

print('The minutes to run the above code on 5.8 million observations using H2O GBM with 10 bins: ', predictionTime/60)

The minutes to run the above code on 5.8 million observations using H2O GBM with 10 bins: 39.89095404148102

In [53]:

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn import preprocessing, tree

from sklearn.metrics import classification\_report, confusion\_matrix

In [81]:

flights = pd.read\_csv('flights.csv', encoding='unicode\_escape')

In [82]:

type(flights)

Out[82]:

pandas.core.frame.DataFrame

In [83]:

flights.dtypes

Out[83]:

YEAR int64

MONTH int64

DAY int64

DAY\_OF\_WEEK int64

AIRLINE object

FLIGHT\_NUMBER int64

TAIL\_NUMBER object

ORIGIN\_AIRPORT object

DESTINATION\_AIRPORT object

SCHEDULED\_DEPARTURE int64

DEPARTURE\_TIME float64

DEPARTURE\_DELAY float64

TAXI\_OUT float64

WHEELS\_OFF float64

SCHEDULED\_TIME float64

ELAPSED\_TIME float64

AIR\_TIME float64

DISTANCE int64

WHEELS\_ON float64

TAXI\_IN float64

SCHEDULED\_ARRIVAL int64

ARRIVAL\_TIME float64

ARRIVAL\_DELAY float64

DIVERTED int64

CANCELLED int64

CANCELLATION\_REASON object

AIR\_SYSTEM\_DELAY float64

SECURITY\_DELAY float64

AIRLINE\_DELAY float64

LATE\_AIRCRAFT\_DELAY float64

WEATHER\_DELAY float64

dtype: object

In [85]:

flights = flights.astype({"YEAR":'category', "MONTH":'category',"DAY\_OF\_WEEK":'category',"FLIGHT\_NUMBER":'category',"DISTANCE":'category'})

In [86]:

flights.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5819079 entries, 0 to 5819078

Data columns (total 31 columns):

YEAR category

MONTH category

DAY int64

DAY\_OF\_WEEK category

AIRLINE object

FLIGHT\_NUMBER category

TAIL\_NUMBER object

ORIGIN\_AIRPORT object

DESTINATION\_AIRPORT object

SCHEDULED\_DEPARTURE int64

DEPARTURE\_TIME float64

DEPARTURE\_DELAY float64

TAXI\_OUT float64

WHEELS\_OFF float64

SCHEDULED\_TIME float64

ELAPSED\_TIME float64

AIR\_TIME float64

DISTANCE category

WHEELS\_ON float64

TAXI\_IN float64

SCHEDULED\_ARRIVAL int64

ARRIVAL\_TIME float64

ARRIVAL\_DELAY float64

DIVERTED int64

CANCELLED int64

CANCELLATION\_REASON object

AIR\_SYSTEM\_DELAY float64

SECURITY\_DELAY float64

AIRLINE\_DELAY float64

LATE\_AIRCRAFT\_DELAY float64

WEATHER\_DELAY float64

dtypes: category(5), float64(16), int64(5), object(5)

memory usage: 1.2+ GB

In [87]:

X = flights[['YEAR','MONTH','DAY\_OF\_WEEK','FLIGHT\_NUMBER','DISTANCE']]

y = flights['DEPARTURE\_DELAY']

(4655263, 5)

(1163816, 5)

(4655263,)

(1163816,)

In [95]:

flightsXY = pd.concat([X,y], axis=1)

In [97]:

flightsXY.shape

Out[97]:

(5819079, 6)

In [99]:

flightsXY = flightsXY.dropna()

flightsXY.shape

Out[99]:

(5732926, 6)

In [102]:

flightsXY

Out[102]:

|  | **YEAR** | **MONTH** | **DAY\_OF\_WEEK** | **FLIGHT\_NUMBER** | **DISTANCE** | **DEPARTURE\_DELAY** |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 2015 | 1 | 4 | 98 | 1448 | -11.0 |
| 1 | 2015 | 1 | 4 | 2336 | 2330 | -8.0 |
| 2 | 2015 | 1 | 4 | 840 | 2296 | -2.0 |
| 3 | 2015 | 1 | 4 | 258 | 2342 | -5.0 |
| 4 | 2015 | 1 | 4 | 135 | 1448 | -1.0 |
| ... | ... | ... | ... | ... | ... | ... |
| 5819074 | 2015 | 12 | 4 | 688 | 2611 | -4.0 |
| 5819075 | 2015 | 12 | 4 | 745 | 1617 | -4.0 |
| 5819076 | 2015 | 12 | 4 | 1503 | 1598 | -9.0 |
| 5819077 | 2015 | 12 | 4 | 333 | 1189 | -6.0 |
| 5819078 | 2015 | 12 | 4 | 839 | 1576 | 15.0 |

5732926 rows × 6 columns

In [103]:

X = flightsXY.iloc[:,0:5].values

y = flightsXY.iloc[:,5].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2, random\_state=20)

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

(4586340, 5)

(1146586, 5)

(4586340,)

(1146586,)

In [104]:

X\_train

Out[104]:

array([[2015, 7, 3, 4098, 946],

[2015, 5, 3, 2981, 135],

[2015, 1, 6, 4096, 472],

...,

[2015, 12, 3, 246, 991],

[2015, 7, 5, 5046, 83],

[2015, 3, 3, 3380, 351]], dtype=object)

In [105]:

y\_train

Out[105]:

array([363., 3., -6., ..., -4., -3., 20.])

In [106]:

knn = KNeighborsClassifier()

knn.fit(X\_train,

y\_train)

Out[106]:

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2,

weights='uniform')

In [107]:

y\_pred = knn.predict(X\_test)

In [109]:

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

print('Accuracy: ',accuracy\_score(y\_test, y\_pred))

[[0 0 0 ... 0 0 0]

[0 0 0 ... 0 0 0]

[0 0 0 ... 0 0 0]

...

[0 0 0 ... 0 0 0]

[0 0 0 ... 0 0 0]

[0 0 0 ... 0 0 0]]

Accuracy: 0.06603516875315066

In [ ]: