Reference: https://www.kaggle.com/nroman/lgb-single-model-lb-0-9419
(https://www.kaggle.com/nroman/lgb-single-model-lb-0-9419)

Importing necessary library

In [1]:

```
import pandas as pd
import numpy as np
import multiprocessing
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
import lightgbm as lgb
import gc
from time import time
import datetime
from tqdm import tqdm_notebook
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold, TimeSeries
Split
from sklearn.metrics import roc_auc_score
warnings.simplefilter('ignore')
sns.set()
%matplotlib inline
```

Importing datasets

In [2]:

```
sub = pd.read_csv("../input/ieee-fraud-detection/samp
le_submission.csv")
```

In [3]:

```
train_id = pd.read_csv("../input/ieee-fraud-detectio
n/train_identity.csv")
train_tr = pd.read_csv("../input/ieee-fraud-detectio
n/train_transaction.csv")
```

In [4]:

```
train_id.head(5)
```

Out[4]:

	TransactionID	id_01	id_02	id_03	id_04	id_05	id_06	id_07	i
0	2987004	0.0	70787.0	NaN	NaN	NaN	NaN	NaN	N
1	2987008	-5.0	98945.0	NaN	NaN	0.0	-5.0	NaN	N
2	2987010	-5.0	191631.0	0.0	0.0	0.0	0.0	NaN	N
3	2987011	-5.0	221832.0	NaN	NaN	0.0	-6.0	NaN	N
4	2987016	0.0	7460.0	0.0	0.0	1.0	0.0	NaN	N

5 rows × 41 columns

In [5]:

```
train_id.head(5)
```

Out[5]:

	TransactionID	id_01	id_02	id_03	id_04	id_05	id_06	id_07	id
0	2987004	0.0	70787.0	NaN	NaN	NaN	NaN	NaN	N
1	2987008	-5.0	98945.0	NaN	NaN	0.0	-5.0	NaN	N
2	2987010	-5.0	191631.0	0.0	0.0	0.0	0.0	NaN	N
3	2987011	-5.0	221832.0	NaN	NaN	0.0	-6.0	NaN	N
4	2987016	0.0	7460.0	0.0	0.0	1.0	0.0	NaN	N

5 rows × 41 columns

In [6]:

```
train_id.shape, train_tr.shape
```

Out[6]:

```
((144233, 41), (590540, 394))
```

In [7]:

```
test_id = pd.read_csv("../input/ieee-fraud-detection/
test_identity.csv")
test_tr = pd.read_csv("../input/ieee-fraud-detection/
test_transaction.csv")
```

In [8]:

```
test_id.shape, test_tr.shape
```

Out[8]:

```
((141907, 41), (506691, 393))
```

Merging transaction and Identity

In [9]:

```
train = pd.merge(train_tr, train_id, on='TransactionI
D', how='left')
test = pd.merge(test_tr, test_id, on='TransactionID',
how='left')

del test_id, test_tr, train_id, train_tr
gc.collect()
```

Out[9]:

```
15
```

In [10]:

```
train.head(5)
```

Out[10]:

	TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card
0	2987000	0	86400	68.5	W	1392
1	2987001	0	86401	29.0	W	2755
2	2987002	0	86469	59.0	W	4663
3	2987003	0	86499	50.0	W	1813
4	2987004	0	86506	50.0	H	4497

5 rows × 434 columns

In [11]:

```
train.shape, test.shape
```

Out[11]:

```
((590540, 434), (506691, 433))
```

From below we can see that there are a lot of features with almost 99% nan values

In [12]:

```
train.isna().sum()
```

Out[12]:

```
TransactionID          0
isFraud                0
TransactionDT          0
TransactionAmt         0
ProductCD              0
                     ...
id_36             449555
id_37             449555
id_38             449555
DeviceType        449730
DeviceInfo        471874
Length: 434, dtype: int64
```

Sorting features on basis of TransactionDT

In [13]:

```
train = train.sort_values('TransactionDT')
```

Taking all features

Initially I will start with all the features and then will drop most of the features

Initially I will start with all the features and then will drop most of the features on the basis of count

In [14]:
```
useful_features = [col for col in train.columns]
```

From below we can see that length of features is 434

In [15]:
```
len(useful_features)
```

Out[15]:
```
434
```

In [16]:
```
train.shape
```

Out[16]:
```
(590540, 434)
```

In [17]:
```
train.isna().sum()
```

Out[17]:
```
TransactionID          0
isFraud                0
TransactionDT          0
TransactionAmt         0
ProductCD              0
                     ...
id_36             449555
id_37             449555
id_38             449555
DeviceType        449730
DeviceInfo        471874
Length: 434, dtype: int64
```

Displaying all the columns

In [18]:
```
pd.set_option('display.max_columns', None)
```

In [19]:
```
train.shape, test.shape
```

Out[19]:
```
((590540, 434), (506691, 433))
```

In [20]:
```
train.head(10)
```

Out[20]:

	TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card
0	2987000	0	86400	68.5	W	1392
1	2987001	0	86401	29.0	W	2755
2	2987002	0	86469	59.0	W	4663
3	2987003	0	86499	50.0	W	1813
4	2987004	0	86506	50.0	H	4497
5	2987005	0	86510	49.0	W	5937
6	2987006	0	86522	159.0	W	1230
7	2987007	0	86529	422.5	W	1269
8	2987008	0	86535	15.0	H	2803
9	2987009	0	86536	117.0	W	1739

In [21]:

```
target = train["isFraud"]
train.drop(["isFraud"], axis=1, inplace=True)
```

Concatinating train and test as one dataframe

In [22]:

```
train = pd.concat([train,test])
```

In [23]:

```
train.drop(["TransactionID", "TransactionDT"], axis=1
, inplace=True)
train.shape
```

Out[23]:

```
(1097231, 431)
```

Here I will treat all features as categorical except TransationAmt

In [24]:

```
neglect = ["TransactionAmt"]
```

In [25]:

```
useful_features = [col for col in train.columns if co
l not in neglect]
```

This block of code count every features and drop original features

Dropping below features as these seems to be repeating

In [26]:

```
dropping1 =["D8_count_dist", "V138_count_dist", "V139
_count_dist", "V140_count_dist", "V141_count_dist",\
           "V146_count_dist", "V147_count_dist", "V14
```

```
            "V146_count_dist", "V147_count_dist", "V14
8_count_dist", "V149_count_dist", "V144_count_dist",\
            "V145_count_dist", "V150_count_dist", "V15
1_count_dist", "V152_count_dist", "V153_count_dist",\
            "V154_count_dist", "V155_count_dist", "V15
6_count_dist", "V157_count_dist", "V158_count_dist",\
            "V159_count_dist", "V160_count_dist", "V16
1_count_dist", "V162_count_dist", "V163_count_dist",\
            "V164_count_dist", "V165_count_dist", "V16
6_count_dist", "V168_count_dist", "V170_count_dist",\
            "V171_count_dist", "V172_count_dist", "V17
3_count_dist", "V174_count_dist", "V175_count_dist",\
            "V176_count_dist", "V177_count_dist", "V17
8_count_dist", "V179_count_dist", "V180_count_dist",\
            "V181_count_dist", "V182_count_dist", "V18
3_count_dist", "V184_count_dist", "V185_count_dist",\
            "V186_count_dist", "V187_count_dist", "V18
8_count_dist", "V189_count_dist", "V190_count_dist",\
            "V191_count_dist", "V192_count_dist", "V19
3_count_dist", "V194_count_dist", "V195_count_dist",\
            "V196_count_dist", "V197_count_dist", "V19
8_count_dist", "V199_count_dist", "V200_count_dist",\
            "V201_count_dist", "V202_count_dist", "V20
3_count_dist", "V204_count_dist", "V205_count_dist",\
            "V206_count_dist", "V207_count_dist", "V20
8_count_dist", "V209_count_dist", "V210_count_dist",\
            "V211_count_dist", "V212_count_dist", "V21
3_count_dist", "V214_count_dist", "V215_count_dist",\
            "V216_count_dist", "V218_count_dist", "V21
9_count_dist", "V221_count_dist", "V222_count_dist",\
            "V223_count_dist", "V224_count_dist", "V22
5_count_dist", "V226_count_dist", "V227_count_dist",\
            "V228_count_dist", "V229_count_dist", "V23
0_count_dist", "V231_count_dist", "V232_count_dist",\
            "V233_count_dist", "V234_count_dist", "V23
5_count_dist", "V236_count_dist", "V237_count_dist",\
            "V205_count_dist", "V205_count_dist", "V20
5_count_dist", "V205_count_dist", "V205_count_dist",\
            "V238_count_dist", "V239_count_dist", "V24
0_count_dist", "V241_count_dist", "V242_count_dist",\
            "V243_count_dist", "V244_count_dist", "V24
5_count_dist","V246_count_dist", "V247_count_dist",\
            "V248_count_dist", "V249_count_dist", "V25
0_count_dist", "V251_count_dist", "V252_count_dist",\
            "V253_count_dist", "V254_count_dist", "V25
5_count_dist", "V256_count_dist", "V257_count_dist",\
            "V258_count_dist", "V259_count_dist", "V26
0_count_dist", "V261_count_dist", "V262_count_dist",\
            "V263_count_dist", "V264_count_dist", "V26
5_count_dist", "V266_count_dist", "V267_count_dist",\
            "V268_count_dist", "V269_count_dist", "V27
0_count_dist", "V271_count_dist", "V272_count_dist",\
            "V273_count_dist", "V274_count_dist", "V27
5_count_dist", "V276_count_dist", "V277_count_dist",\
            "V278_count_dist", "V323_count_dist", "V32
4_count_dist", "V325_count_dist", "V326_count_dist",\
            "V327_count_dist", "V328_count_dist", "V32
9_count_dist", "V330_count_dist", "V331_count_dist",\
            "V332_count_dist", "V333_count_dist", "V33
```

```
4_count_dist", "V335_count_dist", "V336_count_dist",\
           "V237_count_dist", "V238_count_dist", "V23
9_count_dist", "id_04_count_dist", "id_06_count_dist"
,\
           "id_08_count_dist", "id_10_count_dist", "i
d_22_count_dist", "id_27_count_dist", "id_29_count_di
st",\
           "id_36_count_dist", "id_37_count_dist", "i
d_38_count_dist"]
```

In [27]:

```
dropping = []
for i in dropping1:
    dropping.append(i.replace("_count_dist", ""))
```

In [28]:

```
dropping
```

Out[28]:

```
['D8',
 'V138',
 'V139',
 'V140',
 'V141',
 'V146',
 'V147',
 'V148',
 'V149',
 'V144',
 'V145',
 'V150',
 'V151',
 'V152',
 'V153',
 'V154',
 'V155',
 'V156',
 'V157',
 'V158',
 'V159',
 'V160',
 'V161',
 'V162',
 'V163',
 'V164',
 'V165',
 'V166',
 'V168',
 'V170',
 'V171',
 'V172',
 'V173',
 'V174',
 'V175',
 'V176',
 'V177',
 'V178',
```

```
'V179',
'V180',
'V181',
'V182',
'V183',
'V184',
'V185',
'V186',
'V187',
'V188',
'V189',
'V190',
'V191',
'V192',
'V193',
'V194',
'V195',
'V196',
'V197',
'V198',
'V199',
'V200',
'V201',
'V202',
'V203',
'V204',
'V205',
'V206',
'V207',
'V208',
'V209',
'V210',
'V211',
'V212',
'V213',
'V214',
'V215',
'V216',
'V218',
'V219',
'V221',
'V222',
'V223',
'V224',
'V225',
'V226',
'V227',
'V228',
'V229',
'V230',
'V231',
'V232',
'V233',
'V234',
'V235',
'V236',
'V237',
'V205',
```

```
'V205',
'V205',
'V205',
'V205',
'V238',
'V239',
'V240',
'V241',
'V242',
'V243',
'V244',
'V245',
'V246',
'V247',
'V248',
'V249',
'V250',
'V251',
'V252',
'V253',
'V254',
'V255',
'V256',
'V257',
'V258',
'V259',
'V260',
'V261',
'V262',
'V263',
'V264',
'V265',
'V266',
'V267',
'V268',
'V269',
'V270',
'V271',
'V272',
'V273',
'V274',
'V275',
'V276',
'V277',
'V278',
'V323',
'V324',
'V325',
'V326',
'V327',
'V328',
'V329',
'V330',
'V331',
'V332',
'V333',
'V334',
'V335',
'V336'
```

```
 'V236',
 'V237',
 'V238',
 'V239',
 'id_04',
 'id_06',
 'id_08',
 'id_10',
 'id_22',
 'id_27',
 'id_29',
 'id_36',
 'id_37',
 'id_38']
```

In [29]:

```
train = train.drop(dropping, axis=1)
```

In [30]:

```
i=0
for feature in useful_features:

        # Count encoded separately for train and test
    train[feature + '_count_dist'] = np.log(train[feature].map(train[feature].value_counts(dropna=False)))
    train.drop([feature], axis=1,inplace=True)
    print("Done" + str(i))
    i+=1
```

```
Done0
Done1
Done2
Done3
Done4
Done5
Done6
Done7
Done8
Done9
Done10
Done11
Done12
Done13
Done14
Done15
Done16
Done17
Done18
Done19
Done20
Done21
Done22
Done23
Done24
Done25
Done26
```

```
Done27
Done28
Done29
Done30
Done31
Done32
Done33


---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2896             try:
```



```
pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'D8'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-30-eccd76ef1c1f> in <module>
      3 
      4         # Count encoded separately for train and test
----> 5     train[feature + '_count_dist'] = np.log(train[feature].map(train[feature].value_counts(dropna=False)))
      6     train.drop([feature], axis=1,inplace=True)
      7     print("Done" + str(i))

/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in __getitem__(self, key)
   2978             if self.columns.nlevels > 1:
   2979                 return self._getitem_multilevel(key)
-> 2980             indexer = self.columns.get_loc(key)
   2981             if is_integer(indexer):
```

```
   2982                 indexer = [indexer]

/opt/conda/lib/python3.6/site-packages/pandas/core/in
dexes/base.py in get_loc(self, key, method, toleranc
e)
   2897                 return self._engine.get_loc(k
ey)
   2898             except KeyError:
-> 2899                 return self._engine.get_loc(s
elf._maybe_cast_indexer(key))
   2900         indexer = self.get_indexer([key], met
hod=method, tolerance=tolerance)
   2901         if indexer.ndim > 1 or indexer.size >
1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEng
ine.get_loc()
```

Notebook Data Output Comments

```
pandas/_libs/hashtable_class_helper.pxi in pandas._li
bs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._li
bs.hashtable.PyObjectHashTable.get_item()

KeyError: 'D8'
```

In [31]:

```
len(dropping)
```

Out[31]:

```
168
```

Below we can see that all I am left with is count

In [32]:

```
train.head(4)
```

Out[32]:

	TransactionAmt	D9	D10	D11	D12	D13	D14	D15	M1
0	68.5	NaN	13.0	13.0	NaN	NaN	NaN	0.0	T
1	29.0	NaN	0.0	NaN	NaN	NaN	NaN	0.0	NaN
2	59.0	NaN	0.0	315.0	NaN	NaN	NaN	315.0	T
3	50.0	NaN	84.0	NaN	NaN	NaN	NaN	111.0	NaN

In [33]:

```
train.shape
```

Out[33]:

```
(1097231, 271)
```

In [34]:
```
#X = train.sort_values('TransactionDT').drop(['isFraud', 'TransactionDT', 'TransactionID'], axis=1)#
#y = train.sort_values('TransactionDT')['isFraud']
#test = test.sort_values('TransactionDT').drop(['TransactionDT', 'TransactionID'], axis=1)
```

In [35]:
```
#del train
#gc.collect()
```

Again seperating data into train and test

In [36]:
```
X = train.iloc[:590540, :]
test = train.iloc[590540:, :]
```

In [37]:
```
y=target
```

Train test and split

In [38]:
```
# Training and Validation Set
#from sklearn.model_selection import train_test_split
#X_train, X_valid, y_train, y_valid = train_test_split(train, target, test_size=0.20, random_state=23)
```

Lightgbm

In [39]:
```
from catboost import CatBoostRegressor
categorical_var = np.where(train.dtypes != np.float)[0]
print('\nCategorical Variables indices : ',categorical_var)
```

```
Categorical Variables indices :  [  8   9  10  11  12  13  14  15  16 214 217 218 224 228 229 230 232 233
 234 235 236]
```

In [40]:
```
del train
```

In [41]:
```
params = {'num_leaves': 491,
          'min_child_weight': 0.03454472573214212,
```

```
             'feature_fraction': 0.3797454081646243,
             'bagging_fraction': 0.4181193142567742,
             'min_data_in_leaf': 106,
             'objective': 'binary',
             'max_depth': -1,
             'learning_rate': 0.006883242363721497,
             "boosting_type": "gbdt",
             "bagging_seed": 11,
             "metric": 'auc',
             "verbosity": -1,
             'reg_alpha': 0.3899927210061127,
             'reg_lambda': 0.6485237330340494,
             'random_state': 47
            }
```

In [42]:

```
folds = TimeSeriesSplit(n_splits=5)

aucs = list()
feature_importances = pd.DataFrame()
feature_importances['feature'] = X.columns

training_start_time = time()
for fold, (trn_idx, test_idx) in enumerate(folds.spli
t(X, y)):
    start_time = time()
    print('Training on fold {}'.format(fold + 1))

    trn_data = lgb.Dataset(X.iloc[trn_idx], label=y.i
loc[trn_idx])
    val_data = lgb.Dataset(X.iloc[test_idx], label=y.
iloc[test_idx])
    clf = lgb.train(params, trn_data, 10000, valid_se
ts = [trn_data, val_data], verbose_eval=1000, early_s
topping_rounds=500)

    feature_importances['fold_{}'.format(fold + 1)] =
clf.feature_importance()
    aucs.append(clf.best_score['valid_1']['auc'])

    print('Fold {} finished in {}'.format(fold + 1, s
tr(datetime.timedelta(seconds=time() - start_time))))
print('-' * 30)
print('Training has finished.')
print('Total training time is {}'.format(str(datetime
.timedelta(seconds=time() - training_start_time))))
print('Mean AUC:', np.mean(aucs))
print('-' * 30)
```

```
Training on fold 1


-----------------------------------------------------
----------------------
ValueError                                 Traceback
 (most recent call last)
<ipython-input-42-9e31bbac3c4c> in <module>
     12     trn_data = lgb.Dataset(X.iloc[trn_idx], l
```

```
abel=y.iloc[trn_idx])
     13     val_data = lgb.Dataset(X.iloc[test_idx],
 label=y.iloc[test_idx])
---> 14     clf = lgb.train(params, trn_data, 10000,
 valid_sets = [trn_data, val_data], verbose_eval=1000
, early_stopping_rounds=500)
     15
     16     feature_importances['fold_{}'.format(fold
+ 1)] = clf.feature_importance()

/opt/conda/lib/python3.6/site-packages/lightgbm/engin
e.py in train(params, train_set, num_boost_round, val
id_sets, valid_names, fobj, feval, init_model, featur
e_name, categorical_feature, early_stopping_rounds, e
vals_result, verbose_eval, learning_rates, keep_train
ing_booster, callbacks)
    195     # construct booster
    196     try:
--> 197         booster = Booster(params=params, trai
n_set=train_set)
    198         if is_valid_contain_train:
    199             booster.set_train_data_name(train
_data_name)

/opt/conda/lib/python3.6/site-packages/lightgbm/basi
c.py in __init__(self, params, train_set, model_file,
silent)
   1550             self.handle = ctypes.c_void_p()
   1551             _safe_call(_LIB.LGBM_BoosterCreat
e(
-> 1552                 train_set.construct().handle,
   1553                 c_str(params_str),
   1554                 ctypes.byref(self.handle)))

/opt/conda/lib/python3.6/site-packages/lightgbm/basi
c.py in construct(self)
    999                                     init_score=se
lf.init_score, predictor=self._predictor,
   1000                                     silent=self.s
ilent, feature_name=self.feature_name,
-> 1001                                     categorical_f
eature=self.categorical_feature, params=self.params)
   1002             if self.free_raw_data:
   1003                 self.data = None

/opt/conda/lib/python3.6/site-packages/lightgbm/basi
c.py in _lazy_init(self, data, label, reference, weig
ht, group, init_score, predictor, silent, feature_nam
e, categorical_feature, params)
    727
feature_name,
    728
categorical_feature,
--> 729
self.pandas_categorical)
    730         label = _label_from_pandas(label)
    731         self.data_has_header = False
```

```
/opt/conda/lib/python3.6/site-packages/lightgbm/basi
c.py in _data_from_pandas(data, feature_name, categor
ical_feature, pandas_categorical)
    275             msg = ("DataFrame.dtypes for data
must be int, float or bool.\n"
    276                    "Did not expect the data t
ypes in fields ")
--> 277             raise ValueError(msg + ', '.join(
bad_fields))
    278         data = data.values.astype('float')
    279     else:

ValueError: DataFrame.dtypes for data must be int, fl
oat or bool.
Did not expect the data types in fields M1, M2, M3, M
4, M5, M6, M7, M8, M9, id_12, id_15, id_16, id_23, id
_28, id_30, id_31, id_33, id_34, id_35, DeviceType, D
eviceInfo
```

In [43]:

```
feature_importances['average'] = feature_importances
[['fold_{}'.format(fold + 1) for fold in range(folds.
n_splits)]].mean(axis=1)
feature_importances.to_csv('feature_importances.csv')

plt.figure(figsize=(16, 16))
sns.barplot(data=feature_importances.sort_values(by=
'average', ascending=False).head(50), x='average', y=
'feature');
plt.title('50 TOP feature importance over {} folds av
erage'.format(folds.n_splits));
```

```
-----------------------------------------------------
----------------------
KeyError                                  Traceback
 (most recent call last)
<ipython-input-43-1195e33b4c10> in <module>
----> 1 feature_importances['average'] = feature_impo
rtances[['fold_{}'.format(fold + 1) for fold in range
(folds.n_splits)]].mean(axis=1)
      2 feature_importances.to_csv('feature_importanc
es.csv')
      3
      4 plt.figure(figsize=(16, 16))
      5 sns.barplot(data=feature_importances.sort_val
ues(by='average', ascending=False).head(50), x='avera
ge', y='feature');

/opt/conda/lib/python3.6/site-packages/pandas/core/fr
ame.py in __getitem__(self, key)
```


Did you find this Kernel useful?

Show your appreciation with an upvote

0

Show your appreciation with an upvote

Data

Data Sources

- IEEE-CIS Fraud Detection
  - sample_submission.csv 507k x 2
  - test_identity.csv 142k x 41
  - test_transaction.csv 507k x 393
  - train_identity.csv 144k x 41
  - train_transaction.csv 591k x 394

IEEE-CIS Fraud Detection

Can you detect fraud from customer transactions?

Last Updated: 2 months ago

About this Competition

In this competition you are predicting the probability that an online transaction is fraudulent, as denoted by the binary target `isFraud` .

The data is broken into two files `identity` and `transaction` , which are joined by `TransactionID` . Not all transactions have corresponding identity information.

Categorical Features - Transaction

- `ProductCD`
- `card1 - card6`
- `addr1` , `addr2`
- `P_emaildomain`
- `R_emaildomain`
- `M1 - M9`

Categorical Features - Identity

- `DeviceType`
- `DeviceInfo`
- `id_12 - id_38`

The `TransactionDT` feature is a timedelta from a given reference datetime (not an actual timestamp).

You can read more about the data from this post by the competition host.

Output Files

New Dataset New Notebook Download All

Output Files

- ieee_cis_fraud_detection_1.csv

About this file

This file was created from a Kernel, it does not have a description.

ieee_cis_fraud_detection_1.csv

1	TransactionID	isFraud
2	3663549	0.5
3	3663550	0.5
4	3663551	0.5
5	3663552	0.5
6	3663553	0.5
7	3663554	0.5
8	3663555	0.5
9	3663556	0.5
10	3663557	0.5
11	3663558	0.5
12	3663559	0.5
13	3663560	0.5
14	3663561	0.5
15	3663562	0.5
16	3663563	0.5
17	3663564	0.5
18	3663565	0.5
19	3663566	0.5
20	3663567	0.5
21	3663568	0.5
22	3663569	0.5
23	3663570	0.5
24	3663571	0.5
25	3663572	0.5
26	3663573	0.5
27	3663574	0.5
28	3663575	0.5
29	3663576	0.5
30	3663577	0.5
31	3663578	0.5

Comments (0)

Click here to comment...