Detecting Malwares with LGBM

Python notebook using data from Microsoft Malware Prediction · 7,996 views

Version 28
28 commits

Notebook
Data
Output
Log
Comments

Submission
✔ Ran successfully
Submitted by FabienDaniel a month ago

Public Score
0.677

Notebook Content

1. Utility functions

Before starting, we define a utility function that helps managing memory.

In [1]:

```
def reduce_mem_usage(df, verbose=True):
    numerics = ['int16', 'int32', 'int64', 'float16', 'float32',
 'float64']
    start_mem = df.memory_usage().sum() / 1024**2
    for col in df.columns:
        col_type = df[col].dtypes
        if col_type in numerics:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.
iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max <
np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max <
np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max <
np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max <
np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max
< np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
```

Hide

```
                else:
                    df[col] = df[col].astype(np.float64)
    end_mem = df.memory_usage().sum() / 1024**2
    if verbose: print('Mem. usage decreased to {:5.2f} Mb ({:.1
f}% reduction)'.format(end_mem, 100 * (start_mem - end_mem) / st
art_mem))
    return df
```

2. Loading the data

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import lightgbm as lgb
from sklearn.model_selection import KFold
import warnings
import gc
import time
import sys
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from sklearn.metrics import mean_squared_error
warnings.simplefilter(action='ignore', category=FutureWarning)
from sklearn import metrics
# Plotly library
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
from plotly import tools
init_notebook_mode(connected=True)
pd.set_option('display.max_columns', 500)
```

2.1 Get the files and select the variables

Following Theo Viel (https://www.kaggle.com/theoviel/load-the-totality-of-the-data), we set the types of each fields in the train set in order to reduce the memory usage:

In [3]:

Hide

```
dtypes = {
        'MachineIdentifier':
'category',
        'ProductName':
'category',
        'EngineVersion':
'category',
```

```
        'AppVersion':
'category',
        'AvSigVersion':
'category',
        'IsBeta':
'int8',
        'RtpStateBitfield':
'float16',
        'IsSxsPassiveMode':
'int8',
        'DefaultBrowsersIdentifier':
'float16',
        'AVProductStatesIdentifier':
'float32',
        'AVProductsInstalled':
'float16',
        'AVProductsEnabled':
'float16',
        'HasTpm':
'int8',
        'CountryIdentifier':
'int16',
        'CityIdentifier':
'float32',
        'OrganizationIdentifier':
'float16',
        'GeoNameIdentifier':
'float16',
        'LocaleEnglishNameIdentifier':
'int8',
        'Platform':
'category',
        'Processor':
'category',
        'OsVer':
'category',
        'OsBuild':
'int16',
        'OsSuite':
'int16',
        'OsPlatformSubRelease':
'category',
        'OsBuildLab':
'category',
        'SkuEdition':
'category',
        'IsProtected':
'float16',
        'AutoSampleOptIn':
'int8',
        'PuaMode':
'category',
        'SMode':
'float16',
        'IeVerIdentifier':
'float16',
        'SmartScreen':
'category',
```

```
        'Firewall':
'float16',
        'UacLuaenable':
'float32',
        'Census_MDC2FormFactor':
'category',
        'Census_DeviceFamily':
'category',
        'Census_OEMNameIdentifier':
'float16',
        'Census_OEMModelIdentifier':
'float32',
        'Census_ProcessorCoreCount':
'float16',
        'Census_ProcessorManufacturerIdentifier':
'float16',
        'Census_ProcessorModelIdentifier':
'float16',
        'Census_ProcessorClass':
'category',
        'Census_PrimaryDiskTotalCapacity':
'float32',
        'Census_PrimaryDiskTypeName':
'category',
        'Census_SystemVolumeTotalCapacity':
'float32',
        'Census_HasOpticalDiskDrive':
'int8',
        'Census_TotalPhysicalRAM':
'float32',
        'Census_ChassisTypeName':
'category',
        'Census_InternalPrimaryDiagonalDisplaySizeInInches':
'float16',
        'Census_InternalPrimaryDisplayResolutionHorizontal':
'float16',
        'Census_InternalPrimaryDisplayResolutionVertical':
'float16',
        'Census_PowerPlatformRoleName':
'category',
        'Census_InternalBatteryType':
'category',
        'Census_InternalBatteryNumberOfCharges':
'float32',
        'Census_OSVersion':
'category',
        'Census_OSArchitecture':
'category',
        'Census_OSBranch':
'category',
        'Census_OSBuildNumber':
'int16',
        'Census_OSBuildRevision':
'int32',
        'Census_OSEdition':
'category',
        'Census_OSSkuName':
'category',
```

```
        'Census_OSInstallTypeName':
'category',
        'Census_OSInstallLanguageIdentifier':
'float16',
        'Census_OSUILocaleIdentifier':
'int16',
        'Census_OSWUAutoUpdateOptionsName':
'category',
        'Census_IsPortableOperatingSystem':
'int8',
        'Census_GenuineStateName':
'category',
        'Census_ActivationChannel':
'category',
        'Census_IsFlightingInternal':
'float16',
        'Census_IsFlightsDisabled':
'float16',
        'Census_FlightRing':
'category',
        'Census_ThresholdOptIn':
'float16',
        'Census_FirmwareManufacturerIdentifier':
'float16',
        'Census_FirmwareVersionIdentifier':
'float32',
        'Census_IsSecureBootEnabled':
'int8',
        'Census_IsWIMBootEnabled':
'float16',
        'Census_IsVirtualDevice':
'float16',
        'Census_IsTouchEnabled':
'int8',
        'Census_IsPenCapable':
'int8',
        'Census_IsAlwaysOnAlwaysConnectedCapable':
'float16',
        'Wdft_IsGamer':
'float16',
        'Wdft_RegionIdentifier':
'float16',
        'HasDetections':
'int8'
        }
```

First, we make a census of the variables, by type, and define the set we want to keep before reading the data:

In [4]:
```
numerics = ['int8', 'int16', 'int32', 'int64', 'float16', 'float32',
 'float64']
numerical_columns = [c for c,v in dtypes.items() if v in numerics]
categorical_columns = [c for c,v in dtypes.items() if v not in numer
ics]
```

We read the data, limiting the size of the training set to 4'000'000 rows:

In [5]:
```
nrows = 4000000
#_____________________________________________________________________
__________
retained_columns = numerical_columns + categorical_columns
train = pd.read_csv('../input/train.csv',
                    nrows = nrows,
                    usecols = retained_columns,
                    dtype = dtypes)
#_____________________________________________________________
retained_columns += ['MachineIdentifier']
retained_columns.remove('HasDetections')
test = pd.read_csv('../input/test.csv',
                   usecols = retained_columns,
                   dtype = dtypes)
```

2.2 Define the type of each variable

In practice, among the numerical variables, many corresponds to identifiers. *In the current dataset, the truly numerical variables are in fact rare*. Below, I make a list of the variables which are truly numerical, according the the description of the data.

In [6]:
```
true_numerical_columns = [
    'Census_ProcessorCoreCount',
    'Census_PrimaryDiskTotalCapacity',
    'Census_SystemVolumeTotalCapacity',
    'Census_TotalPhysicalRAM',
    'Census_InternalPrimaryDiagonalDisplaySizeInInches',
    'Census_InternalPrimaryDisplayResolutionHorizontal',
    'Census_InternalPrimaryDisplayResolutionVertical',
    'Census_InternalBatteryNumberOfCharges'
]
```

We also list binary variables, since they can be treated as numericals by tree methods:

In [7]:
```
binary_variables = [c for c in train.columns if train[c].nunique() =
= 2]
```

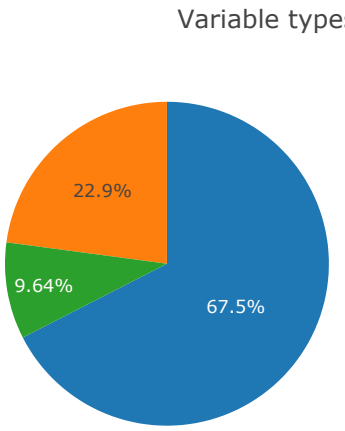
to finally make a census of the categorical variables:

In [8]:
```
categorical_columns = [c for c in train.columns
                       if (c not in true_numerical_columns) & (c not
 in binary_variables)]
```

In [9]:

Hide

```
variables = {
    'categorical_columns': len(categorical_columns),
    'binary_variables': len(binary_variables),
    'true_numerical_columns': len(true_numerical_columns)
}
pie_trace = go.Pie(labels=list(variables.keys()), values=list(va
riables.values()))
layout = dict(title= "Variable types", height=400, width=800)
fig = dict(data=[pie_trace], layout=layout)
iplot(fig)
```



Most of the current variables are categories and we need to choose a method to treat them. **Depending on the cardinality of each variable**, we can opt for **one-hot-encoding, frequency or target encoding**. In the particular case of Light-GBM, we can also use the **built-in LGBM treatment of categoricals**:

In [10]:

Hide

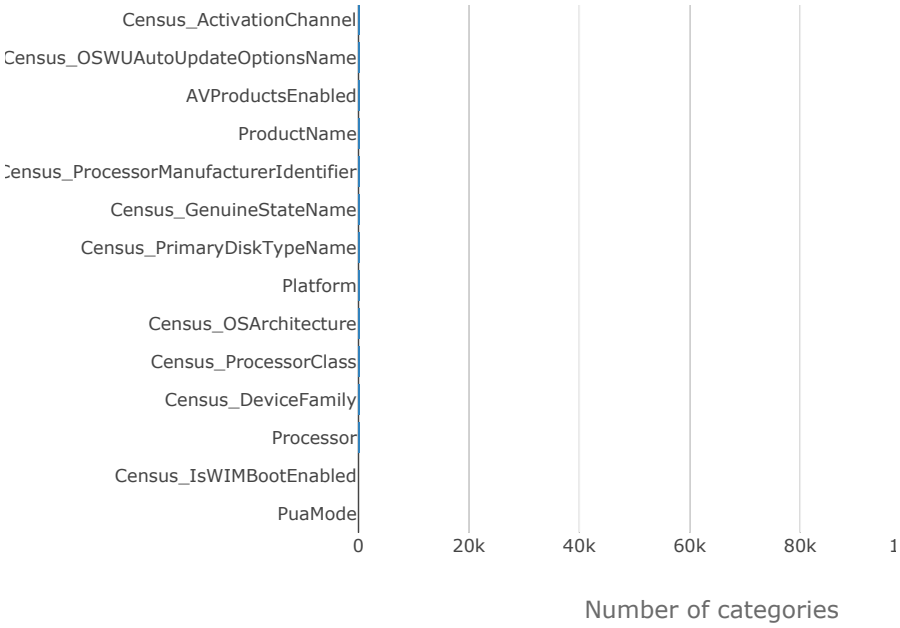
```
cardinality = []
for c in categorical_columns:
    if c == 'MachineIdentifier': continue
    cardinality.append([c, train[c].nunique()])
cardinality.sort(key = lambda x:x[1], reverse=False)

trace = go.Bar(y=[x[0] for x in cardinality],
               x=[x[1] for x in cardinality],
               orientation='h', marker=dict(color='rgb(49,130,18
9)'), name='train')

layout = go.Layout(
    title='Categorical cardinality', height=1600, width=800,
    xaxis=dict(
        title='Number of categories',
        titlefont=dict(size=16, color='rgb(107, 107, 107)'),
        domain=[0.25, 1]
    ),
    barmode='group',
```

```
    bargap=0.1,
    bargroupgap=0.1
)

fig = go.Figure(data=[trace], layout=layout)
iplot(fig)
```



3. Feature Engineering

3.1 Frequency encoding

For variables with large cardinality, an efficient encoding consists in ranking the categories with respect to their frequencies. These variables are then treated as numerical.

In [11]:
```
def frequency_encoding(variable):
    t = pd.concat([train[variable], test[variable]]).value_count
s().reset_index()
    t = t.reset_index()
    t.loc[t[variable] == 1, 'level_0'] = np.nan
    t.set_index('index', inplace=True)
    max_label = t['level_0'].max() + 1
    t.fillna(max_label, inplace=True)
    return t.to_dict()['level_0']
```

Hide

In [12]:
```
frequency_encoded_variables = [
    'Census_OEMModelIdentifier',
    'CityIdentifier',
    'Census_FirmwareVersionIdentifier',
    'AvSigVersion',
    'Census_ProcessorModelIdentifier',
    'Census_OEMNameIdentifier',
    'DefaultBrowsersIdentifier'
]
```

In [13]:
```
for variable in tqdm(frequency_encoded_variables):
```

```
    freq_enc_dict = frequency_encoding(variable)
    train[variable] = train[variable].map(lambda x: freq_enc_dict.ge
t(x, np.nan))
    test[variable] = test[variable].map(lambda x: freq_enc_dict.get(
x, np.nan))
    categorical_columns.remove(variable)
```

```
100%|██████████| 7/7 [00:41<00:00,  5.82s/it]
```

3.2 Label encoding

In [14]:
```
indexer = {}
for col in tqdm(categorical_columns):
    if col == 'MachineIdentifier': continue
    _, indexer[col] = pd.factorize(train[col])

for col in tqdm(categorical_columns):
    if col == 'MachineIdentifier': continue
    train[col] = indexer[col].get_indexer(train[col])
    test[col] = indexer[col].get_indexer(test[col])
```

```
100%|██████████| 49/49 [00:01<00:00, 29.36it/s]
100%|██████████| 49/49 [21:53<00:00, 14.21s/it]
```

In [15]:
```
train = reduce_mem_usage(train)
test = reduce_mem_usage(test)
```

```
Mem. usage decreased to 694.06 Mb (66.7% reduction)
Mem. usage decreased to 1361.03 Mb (66.7% reduction)
```

In [16]:
```
train[:5]
```

Out[16]:

	MachineIdentifier	ProductName	EngineVersion	AppVersion	AvSigVersion
0	0000028988387b115f69f31a3bf04f09	0	0	0	249.0
1	000007535c3f730efa9ea0b7ef1bd645	0	1	1	0.0
2	000007905a28d863f6d0d597892cd692	0	0	0	635.0
3	00000b11598a75ea8ba1beea8459149f	0	0	0	73.0
4	000014a5f00daa18e76b81417eeb99fc	0	0	0	51.0

In [17]:
```
target = train['HasDetections']
del train['HasDetections']
```

3. Training the model

In [18]:
```
param = {'num_leaves': 60,
         'min_data_in_leaf': 60,
         'objective':'binary',
         'max_depth': -1,
         'learning_rate': 0.1,
         "boosting": "gbdt",
         "feature_fraction": 0.8,
         "bagging_freq": 1,
         "bagging_fraction": 0.8 ,
         "bagging_seed": 11,
         "metric": 'auc',
         "lambda_l1": 0.1,
         "random_state": 133,
         "verbosity": -1}
```

We set the max number of iteration over folds:

In [19]:
```
max_iter = 5
```

In [20]:
```
gc.collect()
```

Out[20]:
```
210
```

In [21]:
```
folds = KFold(n_splits=5, shuffle=True, random_state=15)
oof = np.zeros(len(train))
categorical_columns = [c for c in categorical_columns if c not in [
'MachineIdentifier']]
features = [c for c in train.columns if c not in ['MachineIdentifie
r']]
predictions = np.zeros(len(test))
start = time.time()
feature_importance_df = pd.DataFrame()
start_time= time.time()
score = [0 for _ in range(folds.n_splits)]

for fold_, (trn_idx, val_idx) in enumerate(folds.split(train.values,
 target.values)):
    print("fold n°{}".format(fold_))
    trn_data = lgb.Dataset(train.iloc[trn_idx][features],
                           label=target.iloc[trn_idx],
                           categorical_feature = categorical_columns
                          )
    val_data = lgb.Dataset(train.iloc[val_idx][features],
                           label=target.iloc[val_idx],
                           categorical_feature = categorical_columns
```

```
                            categorical_feature = categorical_columns
                           )

    num_round = 10000
    clf = lgb.train(param,
                    trn_data,
                    num_round,
                    valid_sets = [trn_data, val_data],
                    verbose_eval=100,
                    early_stopping_rounds = 200)

    oof[val_idx] = clf.predict(train.iloc[val_idx][features], num_it
eration=clf.best_iteration)

    fold_importance_df = pd.DataFrame()
    fold_importance_df["feature"] = features
    fold_importance_df["importance"] = clf.feature_importance(import
ance_type='gain')
    fold_importance_df["fold"] = fold_ + 1
    feature_importance_df = pd.concat([feature_importance_df, fold_i
mportance_df], axis=0)

    # we perform predictions by chunks
    initial_idx = 0
    chunk_size = 1000000
    current_pred = np.zeros(len(test))
    while initial_idx < test.shape[0]:
        final_idx = min(initial_idx + chunk_size, test.shape[0])
        idx = range(initial_idx, final_idx)
        current_pred[idx] = clf.predict(test.iloc[idx][features], nu
m_iteration=clf.best_iteration)
        initial_idx = final_idx
    predictions += current_pred / min(folds.n_splits, max_iter)

    print("time elapsed: {:<5.2}s".format((time.time() - start_time)
 / 3600))
    score[fold_] = metrics.roc_auc_score(target.iloc[val_idx], oof[v
al_idx])
    if fold_ == max_iter - 1: break

if (folds.n_splits == max_iter):
    print("CV score: {:<8.5f}".format(metrics.roc_auc_score(target,
oof)))
else:
     print("CV score: {:<8.5f}".format(sum(score) / max_iter))
```

```
fold n°0


/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:1186: UserW
arning:

Using categorical_feature in Dataset.

/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:752: UserWa
rning:

categorical_feature in param dict is overridden.
```

```
Training until validation scores don't improve for 200 rounds.
[100]   training's auc: 0.734534        valid_1's auc: 0.72771
[200]   training's auc: 0.744401        valid_1's auc: 0.731472
[300]   training's auc: 0.750132        valid_1's auc: 0.732301
[400]   training's auc: 0.754319        valid_1's auc: 0.732635
[500]   training's auc: 0.757766        valid_1's auc: 0.732752
[600]   training's auc: 0.760704        valid_1's auc: 0.732706
[700]   training's auc: 0.76338 valid_1's auc: 0.732501
Early stopping, best iteration is:
[532]   training's auc: 0.758831        valid_1's auc: 0.732792
time elapsed: 0.36 s
fold n°1


/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:1186: UserW
arning:

Using categorical_feature in Dataset.

/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:752: UserWa
rning:

categorical_feature in param dict is overridden.


Training until validation scores don't improve for 200 rounds.
[100]   training's auc: 0.73433 valid_1's auc: 0.726763
[200]   training's auc: 0.74409 valid_1's auc: 0.730554
[300]   training's auc: 0.750039        valid_1's auc: 0.731747
[400]   training's auc: 0.754322        valid_1's auc: 0.732233
[500]   training's auc: 0.757824        valid_1's auc: 0.732496
[600]   training's auc: 0.76091 valid_1's auc: 0.732535
[700]   training's auc: 0.763749        valid_1's auc: 0.732429
Early stopping, best iteration is:
[558]   training's auc: 0.759663        valid_1's auc: 0.732565
time elapsed: 0.7  s
fold n°2


/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:1186: UserW
arning:

Using categorical_feature in Dataset.

/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:752: UserWa
rning:

categorical_feature in param dict is overridden.


Training until validation scores don't improve for 200 rounds.
[100]   training's auc: 0.734747        valid_1's auc: 0.72659
[200]   training's auc: 0.744595        valid_1's auc: 0.730293
[300]   training's auc: 0.750581        valid_1's auc: 0.731602
[400]   training's auc: 0.75484 valid_1's auc: 0.731815
[500]   training's auc: 0.75843 valid_1's auc: 0.73197
[600]   training's auc: 0.761437        valid_1's auc: 0.731949
```

```
[700]   training's auc: 0.764169        valid_1's auc: 0.731816
Early stopping, best iteration is:
[541]   training's auc: 0.759779        valid_1's auc: 0.732022
time elapsed: 1.0  s
fold n°3

/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:1186: UserW
arning:

Using categorical_feature in Dataset.

/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:752: UserWa
rning:

categorical_feature in param dict is overridden.


Training until validation scores don't improve for 200 rounds.
[100]   training's auc: 0.734686        valid_1's auc: 0.725928
[200]   training's auc: 0.744539        valid_1's auc: 0.729779
[300]   training's auc: 0.750035        valid_1's auc: 0.730761
[400]   training's auc: 0.754432        valid_1's auc: 0.731311
[500]   training's auc: 0.757937        valid_1's auc: 0.731444
[600]   training's auc: 0.761066        valid_1's auc: 0.731434
[700]   training's auc: 0.763712        valid_1's auc: 0.731327
Early stopping, best iteration is:
[547]   training's auc: 0.759493        valid_1's auc: 0.731492
time elapsed: 1.4  s
fold n°4

/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:1186: UserW
arning:

Using categorical_feature in Dataset.

/opt/conda/lib/python3.6/site-packages/lightgbm/basic.py:752: UserWa
rning:

categorical_feature in param dict is overridden.


Training until validation scores don't improve for 200 rounds.
[100]   training's auc: 0.734742        valid_1's auc: 0.727503
[200]   training's auc: 0.744478        valid_1's auc: 0.730999
[300]   training's auc: 0.750276        valid_1's auc: 0.732004
[400]   training's auc: 0.754471        valid_1's auc: 0.732274
[500]   training's auc: 0.75795 valid_1's auc: 0.732358
[600]   training's auc: 0.76114 valid_1's auc: 0.732413
[700]   training's auc: 0.763933        valid_1's auc: 0.732293
Early stopping, best iteration is:
[539]   training's auc: 0.759323        valid_1's auc: 0.732515
time elapsed: 1.7  s
```


Data

Data Sources

Microsoft Malware Pre...

sampl... 7.85m x 2

test.csv

train.csv

Microsoft Malware Prediction

Can you predict if a machine will soon be hit with malware?

Last Updated: a month ago

About this Competition

The goal of this competition is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine. The telemetry data containing these properties and the machine infections was generated by combining heartbeat and threat reports collected by Microsoft's endpoint protection solution, Windows Defender.

Each row in this dataset corresponds to a machine, uniquely identified by a `MachineIdentifier`. `HasDetections` is the ground truth and indicates that Malware was detected on the machine. Using the information and labels in `train.csv`, you must predict the value for `HasDetections` for each machine in `test.csv`.

The sampling methodology used to create this dataset was designed to meet certain business constraints, both in regards to user privacy as well as the time period during which the machine was running. Malware detection is inherently a time-series problem, but it is made complicated by the introduction of new machines, machines that come online and offline, machines that receive patches, machines that receive new operating systems, etc. While the dataset provided here has been roughly split by time, the complications and sampling requirements mentioned above may mean you may see imperfect agreement between your cross validation, public, and private scores! Additionally, this dataset is not representative of Microsoft customers' machines in the wild; it has been sampled to include a much larger proportion of malware machines.

Columns

Output Files

New Dataset | New Kernel | Download All

Output Files

submit.csv

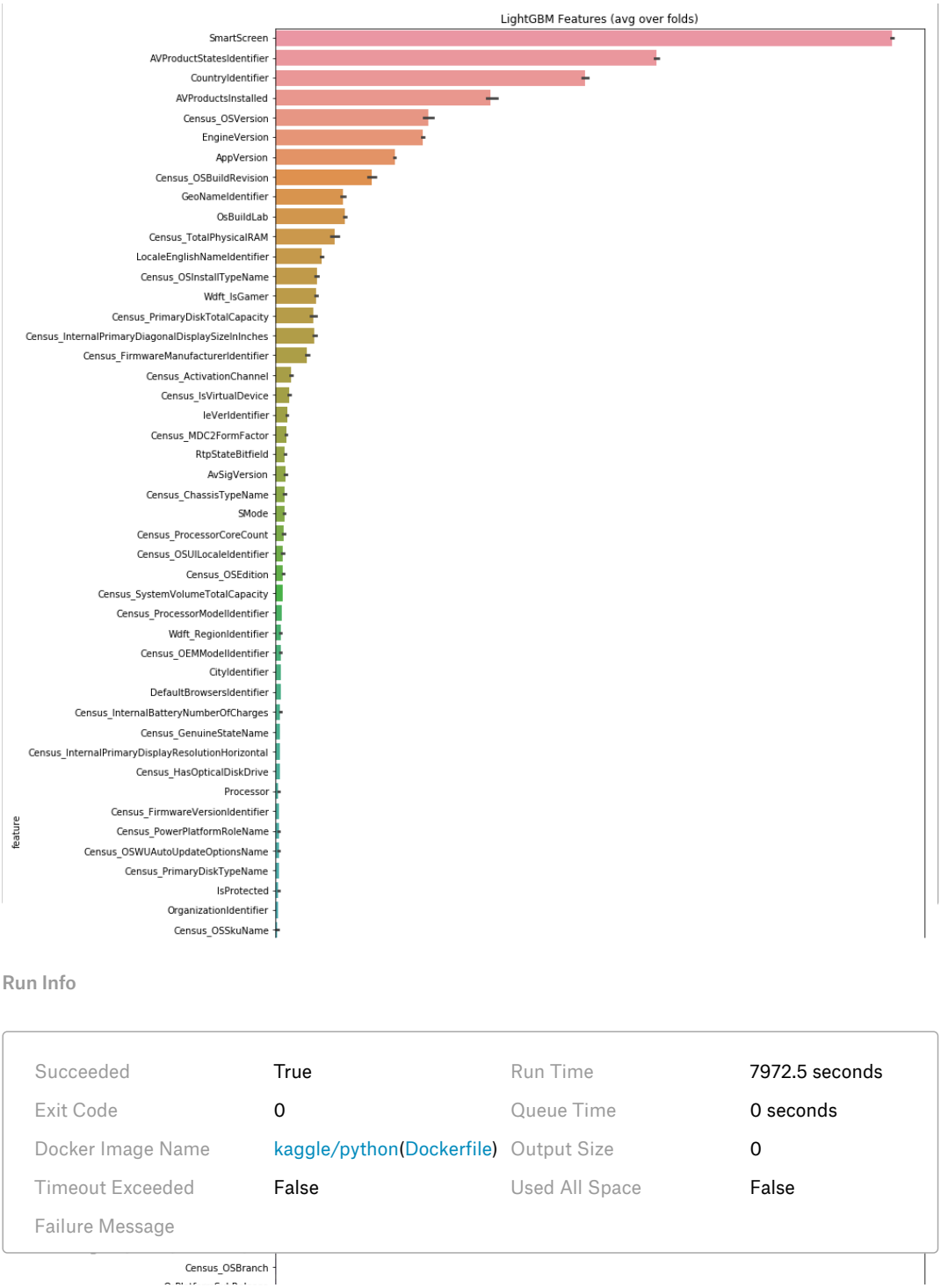
About this file

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

submit.csv

Whoops, something went wrong loading your data.

Output Visualizations



Run Info

Succeeded	True	Run Time	7972.5 seconds
Exit Code	0	Queue Time	0 seconds
Docker Image Name	kaggle/python(Dockerfile)	Output Size	0
Timeout Exceeded	False	Used All Space	False
Failure Message			

Census_OSBranch

Log

Download Log

```
Time  Line #  Log Message
4.4s       1  [NbConvertApp] Converting notebook script.ipynb to html
4.4s       2  [NbConvertApp] Executing notebook with kernel: python3
1667.0s    3  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
1667.0s    4  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
1667.1s    5  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
1667.1s    6  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
1667.1s    7  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
1667.1s    8  [LightGBM] [Warning] Met negative value in categorical features, will
```

```
            convert it to NaN
1667.1s    9 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.1s   10 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.1s   11 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.1s   12 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.1s   13 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.1s   14 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.2s   15 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.2s   16 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
1667.2s   17 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.7s   18 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.7s   19 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.7s   20 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.7s   21 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.8s   22 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.8s   23 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.8s   24 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.8s   25 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.8s   26 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.9s   27 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.9s   28 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
2942.9s   29 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.2s   30 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.2s   31 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.3s   32 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.3s   33 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.3s   34 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.3s   35 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.3s   36 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.3s   37 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.3s   38 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.4s   39 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.4s   40 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.4s   41 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
4167.4s   42 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
5423.3s   43 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
5423.3s   44 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
5423.4s   45 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
5423.4s   46 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
5423.4s   47 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
5423.4s   48 [LightGBM] [Warning] Met negative value in categorical features, will
            convert it to NaN
```

```
5423.4s   49  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
5423.4s   50  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
5423.4s   51  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
5423.5s   52  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
5423.5s   53  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
5423.5s   54  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
5423.5s   55  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6652.9s   56  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6652.9s   57  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6652.9s   58  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6652.9s   59  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6652.9s   60  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6652.9s   61  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6653.0s   62  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6653.0s   63  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6653.0s   64  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6653.0s   65  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6653.0s   66  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6653.1s   67  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
6653.1s   68  [LightGBM] [Warning] Met negative value in categorical features, will
              convert it to NaN
7971.6s   69  [NbConvertApp] Support files will be in __results___files/
              [NbConvertApp] Making directory __results___files
7971.6s   70  [NbConvertApp] Writing 358076 bytes to __results__.html
7971.6s   71
7971.6s   73  Complete. Exited with code 0.
```

Comments (32)

Sort by: All Comments | Hotness

Click here to enter a comment...

Alex Ruberti • Posted on Latest Version • 17 days ago • Options • Reply

1

Hi Fabien,
i am trying to replicate your code on my jupyter.
it keeps on 'dead kernel' -ing on input 21
can you suggest me how to handle that?

Alex Ruberti • Posted on Latest Version • 17 days ago • Options • Reply

1

sorted out.
for all MAC users:
at the beginning of the notebook, write:

```
import os
os.environ["KMPDUPLICATELIB_OK"]='TRUE'
```

SaurabhVe... • Posted on Latest Version • 9 days ago • Options • Reply 0

For me this worked:

```
os.environ['KMP_DUPLICATE_LIB_OK']='True'
```

Prabakaran • Posted on Latest Version • 21 days ago • Options • Reply 2

Why do we need to add 'MachineIdentifier' column again?
retained_columns += ['MachineIdentifier']

LongYin • Posted on Version 21 • a month ago • Options • Reply 1

Why you do reset_index() twice here?

```
def frequency_encoding(variable):

t = train[variable].value_counts().reset_index()

t = t.reset_index()

t.loc[t[variable] == 1, 'level_0'] = np.nan

t.set_index('index', inplace=True)

max_label = t['level_0'].max() + 1

t.fillna(max_label, inplace=True)

return t.to_dict()['level_0']
```

FabienDani... Kernel Author • Posted on Version 21 • a month ago • Options • Reply 2

to create a column with the order of the rows. Thus function is pretty messy, sure it could be written in a more pythonic way !

Möbius • Posted on Version 14 • a month ago • Options • Reply 4

Clean work. Why you didn't use more training data? (parameters like 'histogram*pool*size', 'max_bin' help to manage memory usage)

FabienDani... Kernel Author • Posted on Version 14 • a month ago • Options • Reply 0

Thanks for mentioning those parameters. I'll definitely have a look on how to use them.

tabacof • Posted on Version 19 • a month ago • Options • Reply

2

Very nice kernel, cool informative visualizations and straight to the point. :)

Just one minor issue, as you use early stopping on the validation set for each fold, then you predict on the same set, the AUCs by fold are probably overfitted or misleading. A simple solution would be using a test set to arrive at the expected performance on the leaderboard (there seems enough data for that!).

Alex M. • Posted on Version 4 • a month ago • Options • Reply

0

Nice work getting a LGBM benchmark up so quickly!

FabienDani... Kernel Author • Posted on Version 6 • a month ago • Options • Reply

0

it's the magic of copy paste :)

Panchajanya ... • Posted on Version 6 • a month ago • Options • Reply

0

Did you tune the hyperparameters yet? I'm just using the ELO param dictionary haha..

FabienDani... Kernel Author • Posted on Version 6 • a month ago • Options • Reply

0

a bit soon to tune hyperparameters ;) I haven't done it yet !!

Panchajan... • Posted on Version 6 • a month ago • Options • Reply

1

This will be fun. So many features to investigate!

Ishan Singh • Posted on Version 10 • a month ago • Options • Reply

0

Great Work, thank you for sharing!

Panchajanya ... • Posted on Version 11 • a month ago • Options • Reply

0

Why did you use the max_iter variable?

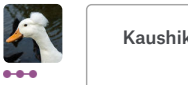
FabienDani... Kernel Author • Posted on Version 11 • a month ago • Options • Reply

1

that's just to reduce the runtime and to speed up the exploration of the dataset.

Panchajan... • Posted on Version 11 • a month ago • Options • Reply 0

Yeah that's sensible: so the n-fold split and shuffle still happens, but we fix the number of iterations

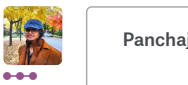


Kaushik P • Posted on Version 16 • a month ago • Options • Reply 0

This is so damn clean and amazing :D, thanks a lot @FabienDaniel

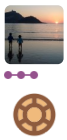
FabienDani... Kernel Author • Posted on Version 16 • a month ago • Options • Reply 3

thanks a lot for the complimemt :)



Panchajanya ... • Posted on Version 16 • a month ago • Options • Reply 0

I'm really enjoying watching this kernel develop! It's a proper treatise on the subject, the changing nature of the features, the increasingly more efficient data-handling... almost like a very good Kaggle tutorial!



FabienDani... Kernel Author • Posted on Version 16 • a month ago • Options • Reply 1

I'm really happy you see it that way, thanks !

LongYin • Posted on Version 21 • a month ago • Options • Reply 0

Thanks for sharing.

Vivekanand J... • Posted on Latest Version • 19 days ago • Options • Reply 0

This is a very good kernel to learn! Thanks a lot for sharing.

Rasool • Posted on Latest Version • 16 days ago • Options • Reply 0

Hi @Fabien, Thanks for sharing your great kernel!
Don't you think it would be better if you add AVProductStatesIdentifier to frequency*encoded*variables?

goshunka • Posted on Latest Version • 15 days ago • Options • Reply 0

what is "reduce mem usage"?

Aditya Soni • Posted on Latest Version • 14 days ago • Options • Reply 0

It's a function to optimize the dtypes of the columns to an appropriate type.. Have a look, it must be defined at the very beginning!

DimitreOliveir... • Posted on Latest Version • 12 days ago • Options • Reply 0

Good job, Thanks!

tamanyan • Posted on Latest Version • 12 days ago • Options • Reply 0

Awesome Kernel :)

RMMG • Posted on Latest Version • 10 days ago • Options • Reply 0

Thanks for sharing, was wondering how does LightGBM handle max_depth when you set it equal to -1

CoreyLevin... • Posted on Latest Version • 2 days ago • Options • Reply 0

when max depth = -1, then there is no max depth, it can grow as large as it wants (can lead to overfitting).

timyee90 • Posted on Latest Version • 21 hours ago • Options • Reply 0

HasDetections in training is binary. Why are predictions floating point? I'm having trouble understanding what is going on.