kernel

September 21, 2021

1 PUBg Finish Placement Prediction

2 Player Unknown Battleground

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
import gc
import os
import sys

sns.set_style('darkgrid')
sns.set_palette('bone')
pd.options.display.float_format = '{:,.3f}'.format
```

[3]:
```
def toTapleList(list1,list2):
    return list(itertools.product(list1,list2))
```

[4]:
```
# Memory saving function credit to https://www.kaggle.com/gemartin/
 ↪load-data-reduce-memory-usage
def reduce_mem_usage(df):
    """ iterate through all the columns of a dataframe and modify the data type
        to reduce memory usage.
    """
    start_mem = df.memory_usage().sum() / 1024**2

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            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)
                #el
                if c_min > np.finfo(np.float32).min and c_max < np.finfo(np.
↪float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        #else:
            #df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB --> {:.2f} MB (Decreased by {:
↪.1f}%)'.format(
        start_mem, end_mem, 100 * (start_mem - end_mem) / start_mem))
    return df
```

3 Load data

[5]:
```
%%time
train = pd.read_csv('train_V2.csv')
train = reduce_mem_usage(train)
test = pd.read_csv('test_V2.csv')
test = reduce_mem_usage(test)
print(train.shape, test.shape)
```

```
Memory usage of dataframe is 1.99 MB --> 0.68 MB (Decreased by 65.9%)
Memory usage of dataframe is 1.60 MB --> 0.54 MB (Decreased by 66.5%)
(8999, 29) (7505, 28)
CPU times: user 89.9 ms, sys: 14.6 ms, total: 105 ms
Wall time: 105 ms
```

[6]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8999 entries, 0 to 8998
Data columns (total 29 columns):
Id                  8999 non-null object
groupId             8999 non-null object
matchId             8999 non-null object
assists             8999 non-null int8
boosts              8999 non-null int8
damageDealt         8999 non-null float32
DBNOs               8999 non-null int8
headshotKills       8999 non-null int8
heals               8999 non-null int8
killPlace           8999 non-null int8
killPoints          8999 non-null int16
kills               8999 non-null int8
killStreaks         8999 non-null int8
longestKill         8999 non-null float32
matchDuration       8999 non-null int16
matchType           8999 non-null object
maxPlace            8999 non-null int8
numGroups           8999 non-null int8
rankPoints          8999 non-null int16
revives             8999 non-null int8
rideDistance        8999 non-null float32
roadKills           8999 non-null int8
swimDistance        8999 non-null float32
teamKills           8999 non-null int8
vehicleDestroys     8999 non-null int8
walkDistance        8999 non-null float32
weaponsAcquired     8999 non-null int8
winPoints           8999 non-null int16
winPlacePerc        8999 non-null float32
dtypes: float32(6), int16(4), int8(15), object(4)
memory usage: 694.4+ KB
```

[7]:
```
null_cnt = train.isnull().sum().sort_values()
print('null count:', null_cnt[null_cnt > 0])
# dropna
train.dropna(inplace=True)
```

```
null count: Series([], dtype: int64)
```

[8]:
```
train.describe(include=np.number).drop('count').T
```

[8]:

	mean	std	min	25%	50%	75%	max
assists	0.228	0.573	0.000	0.000	0.000	0.000	6.000
boosts	1.129	1.759	0.000	0.000	0.000	2.000	11.000
damageDealt	130.226	170.846	0.000	0.000	80.340	183.650	2,325.000
DBNOs	0.652	1.133	0.000	0.000	0.000	1.000	13.000
headshotKills	0.226	0.582	0.000	0.000	0.000	0.000	8.000
heals	1.375	2.666	0.000	0.000	0.000	2.000	29.000
killPlace	47.901	27.483	1.000	24.000	48.000	72.000	100.000
killPoints	507.422	627.486	0.000	0.000	0.000	1,172.000	2,026.000
kills	0.922	1.552	0.000	0.000	0.000	1.000	21.000
killStreaks	0.540	0.714	0.000	0.000	0.000	1.000	5.000
longestKill	22.007	48.729	0.000	0.000	0.000	20.195	624.200
matchDuration	1,577.711	257.182	314.000	1,367.000	1,439.000	1,849.000	2,204.000
maxPlace	44.574	23.809	5.000	28.000	30.000	49.000	100.000
numGroups	43.108	23.268	1.000	27.000	30.000	47.000	100.000
rankPoints	887.821	737.079	-1.000	-1.000	1,443.000	1,500.000	2,395.000
revives	0.162	0.465	0.000	0.000	0.000	0.000	6.000
rideDistance	611.111	1,509.545	0.000	0.000	0.000	0.015	17,210.000
roadKills	0.003	0.058	0.000	0.000	0.000	0.000	2.000
swimDistance	4.335	29.668	0.000	0.000	0.000	0.000	768.400
teamKills	0.026	0.175	0.000	0.000	0.000	0.000	
vehicleDestroys	0.008	0.091	0.000	0.000	0.000	0.000	
walkDistance	1,137.408	1,166.894	0.000	153.900	678.100	1,928.000	
weaponsAcquired	3.646	2.396	0.000	2.000	3.000	5.000	
winPoints	609.908	739.887	0.000	0.000	0.000	1,495.000	
winPlacePerc	0.472	0.307	0.000	0.200	0.458	0.741	

```
teamKills            3.000
vehicleDestroys      1.000
walkDistance     9,325.000
weaponsAcquired     43.000
winPoints        1,861.000
winPlacePerc         1.000
```

4 Feature Engineering

[9]:
```
all_data = train.append(test, sort=False).reset_index(drop=True)
del train, test
gc.collect()
```

[9]: 27

4.1 new feature

[10]:
```
def fillInf(df, val):
    numcols = df.select_dtypes(include='number').columns
    cols = numcols[numcols != 'winPlacePerc']
    df[df == np.Inf] = np.NaN
    df[df == np.NINF] = np.NaN
    for c in cols: df[c].fillna(val, inplace=True)
```

[11]:
```
all_data['_totalDistance'] = all_data['rideDistance'] +␣
 ↪all_data['walkDistance'] + all_data['swimDistance']
all_data['_healthItems'] = all_data['heals'] + all_data['boosts']
all_data['_headshotKillRate'] = all_data['headshotKills'] / all_data['kills']
all_data['_killPlaceOverMaxPlace'] = all_data['killPlace'] /␣
 ↪all_data['maxPlace']
all_data['_killsOverWalkDistance'] = all_data['kills'] /␣
 ↪all_data['walkDistance']
#all_data['_killsOverDistance'] = all_data['kills'] / all_data['_totalDistance']
#all_data['_walkDistancePerSec'] = all_data['walkDistance'] /␣
 ↪all_data['matchDuration']

fillInf(all_data, 0)
```

4.2 rank as percent

[12]:
```
match = all_data.groupby('matchId')
all_data['killsPerc'] = match['kills'].rank(pct=True).values
all_data['killPlacePerc'] = match['killPlace'].rank(pct=True).values
all_data['walkDistancePerc'] = match['walkDistance'].rank(pct=True).values
#all_data['damageDealtPerc'] = match['damageDealt'].rank(pct=True).values

all_data['walkPerc_killsPerc'] = all_data['walkDistancePerc'] /␣
↪all_data['killsPerc']
#all_data['walkPerc_kills'] = all_data['walkDistancePerc'] / all_data['kills']
#all_data['kills_walkPerc'] = all_data['kills'] / all_data['walkDistancePerc']
```

4.3 drop feature

[13]:
```
#all_data.drop(['killStreaks','DBNOs'], axis=1, inplace=True)
all_data.drop(['boosts','heals','revives','assists'], axis=1, inplace=True)
all_data.drop(['headshotKills','roadKills','vehicleDestroys','teamKills'],␣
↪axis=1, inplace=True)
all_data.drop(['rideDistance','swimDistance','matchDuration'], axis=1,␣
↪inplace=True)
all_data.drop(['rankPoints','killPoints','winPoints'], axis=1, inplace=True)
```

4.4 grouping

- need to predict the order of places for groups within each match.
- train on group-level instead of the user-level

[14]:
```
match = all_data.groupby(['matchId'])
group = all_data.groupby(['matchId','groupId','matchType'])

# target feature (max, min)
agg_col = list(all_data.columns)
exclude_agg_col =␣
↪['Id','matchId','groupId','matchType','maxPlace','numGroups','winPlacePerc']
for c in exclude_agg_col:
    agg_col.remove(c)
print(agg_col)

# target feature (sum)
sum_col = ['kills','killPlace','damageDealt','walkDistance','_healthItems']
```

```
['damageDealt', 'DBNOs', 'killPlace', 'kills', 'killStreaks', 'longestKill',
'walkDistance', 'weaponsAcquired', ' Id', '_totalDistance', '_healthItems',
```

```
'_headshotKillRate', '_killPlaceOverMaxPlace', '_killsOverWalkDistance',
'killsPerc', 'killPlacePerc', 'walkDistancePerc', 'walkPerc_killsPerc']
```

[15]:
```
''' match sum, match max, match mean, group sum
'''
match_data = pd.concat([
    match.size().to_frame('m.players'),
    match[sum_col].sum().rename(columns=lambda s: 'm.sum.' + s),
    match[sum_col].max().rename(columns=lambda s: 'm.max.' + s),
    match[sum_col].mean().rename(columns=lambda s: 'm.mean.' + s)
    ], axis=1).reset_index()
match_data = pd.merge(match_data,
    group[sum_col].sum().rename(columns=lambda s: 'sum.' + s).reset_index())
match_data = reduce_mem_usage(match_data)

print(match_data.shape)
```

```
Memory usage of dataframe is 2.31 MB --> 1.63 MB (Decreased by 29.3%)
(16461, 24)
```

[16]:
```
''' ranking of kills and killPlace in each match
'''
minKills = all_data.sort_values(['matchId','groupId','kills','killPlace']).
 ↪groupby(
    ['matchId','groupId','kills']).first().reset_index().copy()
for n in np.arange(5):
    c = 'kills_' + str(n) + '_Place'
    nKills = (minKills['kills'] == n)
    minKills.loc[nKills, c] = minKills[nKills].
 ↪groupby(['matchId'])['killPlace'].rank().values
    match_data = pd.merge(match_data,␣
 ↪minKills[nKills][['matchId','groupId',c]], how='left')
    match_data[c].fillna(0, inplace=True)
match_data = reduce_mem_usage(match_data)
del minKills, nKills

print(match_data.shape)
```

```
Memory usage of dataframe is 2.26 MB --> 1.95 MB (Decreased by 13.9%)
(16461, 29)
```

[17]:
```
''' group mean, max, min
'''
all_data = pd.concat([
    group.size().to_frame('players'),
    group.mean(),
    group[agg_col].max().rename(columns=lambda s: 'max.' + s),
```

```
    group[agg_col].min().rename(columns=lambda s: 'min.' + s),
    ], axis=1).reset_index()
all_data = reduce_mem_usage(all_data)

print(all_data.shape)
```

```
Memory usage of dataframe is 5.21 MB --> 3.47 MB (Decreased by 33.4%)
(16461, 60)
```

4.5 aggregate feature

[18]:
```
numcols = all_data.select_dtypes(include='number').columns.values
numcols = numcols[numcols != 'winPlacePerc']
```

[19]:
```
''' match summary, max
'''
all_data = pd.merge(all_data, match_data)
del match_data
gc.collect()

all_data['enemy.players'] = all_data['m.players'] - all_data['players']
for c in sum_col:
    all_data['enemy.' + c] = (all_data['m.sum.' + c] - all_data['sum.' + c]) /␣
 ↪all_data['enemy.players']
    #all_data['p.sum_msum.' + c] = all_data['sum.' + c] / all_data['m.sum.' + c]
    #all_data['p.max_mmean.' + c] = all_data['max.' + c] / all_data['m.mean.' +␣
 ↪c]
    all_data['p.max_msum.' + c] = all_data['max.' + c] / all_data['m.sum.' + c]
    all_data['p.max_mmax.' + c] = all_data['max.' + c] / all_data['m.max.' + c]
    all_data.drop(['m.sum.' + c, 'm.max.' + c], axis=1, inplace=True)

fillInf(all_data, 0)
print(all_data.shape)
```

```
(16461, 92)
```

[20]:
```
''' match rank
'''
match = all_data.groupby('matchId')
matchRank = match[numcols].rank(pct=True).rename(columns=lambda s: 'rank.' + s)
all_data = reduce_mem_usage(pd.concat([all_data, matchRank], axis=1))
rank_col = matchRank.columns
del matchRank
gc.collect()

# instead of rank(pct=True, method='dense')
```

```
match = all_data.groupby('matchId')
matchRank = match[rank_col].max().rename(columns=lambda s: 'max.' + s).
 ↪reset_index()
all_data = pd.merge(all_data, matchRank)
for c in numcols:
    all_data['rank.' + c] = all_data['rank.' + c] / all_data['max.rank.' + c]
    all_data.drop(['max.rank.' + c], axis=1, inplace=True)
del matchRank
gc.collect()

print(all_data.shape)
```

```
Memory usage of dataframe is 12.48 MB --> 8.90 MB (Decreased by 28.7%)
(16461, 146)
```

4.6 killPlace rank of group and kills

[21]:
```
''' TODO: incomplete
'''
killMinorRank = all_data[['matchId','min.kills','max.killPlace']].copy()
group = killMinorRank.groupby(['matchId','min.kills'])
killMinorRank['rank.minor.maxKillPlace'] = group.rank(pct=True).values
all_data = pd.merge(all_data, killMinorRank)

killMinorRank = all_data[['matchId','max.kills','min.killPlace']].copy()
group = killMinorRank.groupby(['matchId','max.kills'])
killMinorRank['rank.minor.minKillPlace'] = group.rank(pct=True).values
all_data = pd.merge(all_data, killMinorRank)

del killMinorRank
gc.collect()
```

[21]: 20

4.7 drop constant feature

[22]:
```
# drop constant column
constant_column = [col for col in all_data.columns if all_data[col].nunique()␣
 ↪== 1]
print('drop columns:', constant_column)
all_data.drop(constant_column, axis=1, inplace=True)
```

```
drop columns: ['rank.maxPlace', 'rank.numGroups']
```

4.8 encode

[23]:
```
'''
solo  <-- solo,solo-fpp,normal-solo,normal-solo-fpp
duo   <-- duo,duo-fpp,normal-duo,normal-duo-fpp,crashfpp,crashtpp
squad <-- squad,squad-fpp,normal-squad,normal-squad-fpp,flarefpp,flaretpp
'''
mapper = lambda x: 'solo' if ('solo' in x) else 'duo' if ('duo' in x) or␣
↪('crash' in x) else 'squad'
all_data['matchType'] = all_data['matchType'].apply(mapper)

all_data = pd.concat([all_data, pd.get_dummies(all_data['matchType'],␣
↪prefix='matchType')], axis=1)
```

[24]:
```
cols = [col for col in all_data.columns if col not in␣
↪['Id','matchId','groupId']]
for i, t in all_data.loc[:, cols].dtypes.iteritems():
    if t == object:
        all_data[i] = pd.factorize(all_data[i])[0]

all_data = reduce_mem_usage(all_data)
all_data.head()
```

```
Memory usage of dataframe is 9.07 MB --> 8.79 MB (Decreased by 3.1%)
```

[24]:

	matchId	groupId	matchType	players	damageDealt	DBNOs	\
0	0000a43bce5eec	7bd08592bb25e2	0	1	0.000	0.000	
1	0000eb01ea6cdd	338a69335655a6	0	1	210.900	1.000	
2	0000eb01ea6cdd	e89636a86735d3	0	1	300.000	2.000	
3	00086e740a5804	4018d80e8ad32a	1	1	177.400	0.000	
4	0008c31a9be4a7	26d4045668cf95	0	1	0.000	0.000	

	killPlace	kills	killStreaks	longestKill	…	\
0	41.000	0.000	0.000	0.000	…	
1	29.000	1.000	1.000	92.570	…	
2	11.000	3.000	3.000	19.330	…	
3	62.000	0.000	0.000	0.000	…	
4	74.000	0.000	0.000	0.000	…	

	rank.min._killsOverWalkDistance	rank.min.killsPerc	\
0	1.000	1.000	
1	0.500	0.500	
2	1.000	1.000	
3	1.000	1.000	
4	1.000	1.000	

	rank.min.killPlacePerc	rank.min.walkDistancePerc	\

```
0                    1.000                      1.000
1                    1.000                      1.000
2                    0.500                      0.500
3                    1.000                      1.000
4                    1.000                      1.000

   rank.min.walkPerc_killsPerc  rank.minor.maxKillPlace  \
0                        1.000                    1.000
1                        1.000                    1.000
2                        0.500                    1.000
3                        1.000                    1.000
4                        1.000                    1.000

   rank.minor.minKillPlace  matchType_duo  matchType_solo  matchType_squad
0                    1.000          0.000           0.000            1.000
1                    1.000          0.000           0.000            1.000
2                    1.000          0.000           0.000            1.000
3                    1.000          0.000           1.000            0.000
4                    1.000          0.000           0.000            1.000

[5 rows x 149 columns]
```

5 Predict

[25]:
```
X_train = all_data[all_data['winPlacePerc'].notnull()].reset_index(drop=True)
X_test = all_data[all_data['winPlacePerc'].isnull()].drop(['winPlacePerc'],␣
 ↪axis=1).reset_index(drop=True)
del all_data
gc.collect()

Y_train = X_train.pop('winPlacePerc')
X_test_grp = X_test[['matchId','groupId']].copy()

# drop matchId,groupId
X_train.drop(['matchId','groupId'], axis=1, inplace=True)
X_test.drop(['matchId','groupId'], axis=1, inplace=True)

X_train_cols = X_train.columns

print(X_train.shape, X_test.shape)
```

```
(8972, 146) (7489, 146)
```

[26]:
```
#print(pd.DataFrame([[val for val in dir()], [sys.getsizeof(eval(val)) for val␣
 ↪in dir()]],
```

```
#                   index=['name','size']).T.sort_values('size',␣
↪ascending=False).reset_index(drop=True)[:10])
```

[27]:
```
from keras import optimizers, regularizers
from keras.callbacks import LearningRateScheduler, EarlyStopping,␣
↪ModelCheckpoint, ReduceLROnPlateau
from keras.layers import Dense, Dropout, BatchNormalization, PReLU
from keras.models import load_model
from keras.models import Sequential

def createModel():
    model = Sequential()
    model.add(Dense(512, kernel_initializer='he_normal', input_dim=X_train.
↪shape[1], activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))

    model.add(Dense(256, kernel_initializer='he_normal'))
    model.add(PReLU(alpha_initializer='zeros', alpha_regularizer=None,␣
↪alpha_constraint=None, shared_axes=None))
    model.add(BatchNormalization())
    model.add(Dropout(0.2))

    model.add(Dense(128, kernel_initializer='he_normal'))
    model.add(PReLU(alpha_initializer='zeros', alpha_regularizer=None,␣
↪alpha_constraint=None, shared_axes=None))
    model.add(BatchNormalization())
    model.add(Dropout(0.1))

    model.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))

    optimizer = optimizers.Adam(lr=0.005)
    model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])
    #model.compile(optimizer=optimizer, loss='binary_crossentropy',␣
↪metrics=['mae'])

    return model
```

```
Using TensorFlow backend.
```

import tensorflow as tf print(tf.**version**)

[28]:
```
def step_decay_schedule(initial_lr=1e-3, decay_factor=0.75, step_size=10,␣
↪verbose=0):
    ''' Wrapper function to create a LearningRateScheduler with step decay␣
↪schedule. '''
    def schedule(epoch):
```

```
        return initial_lr * (decay_factor ** np.floor(epoch/step_size))

    return LearningRateScheduler(schedule, verbose)

lr_sched = step_decay_schedule(initial_lr=0.001, decay_factor=0.97,␣
 ↪step_size=1, verbose=1)
early_stopping = EarlyStopping(monitor='val_mean_absolute_error', mode='min',␣
 ↪patience=10, verbose=1)
```

[29]:
```
from sklearn import preprocessing
import tensorflow as tf
np.random.seed(42)
tf.random.set_seed(1234)

#scaler = preprocessing.StandardScaler().fit(X_train)#.astype(float))
#X_train = scaler.transform(X_train)#.astype(float))
#X_test = scaler.transform(X_test)#.astype(float))

model = createModel()
history = model.fit(
        X_train, Y_train,
        epochs=200,
        batch_size=2**15,
        validation_split=0.2,
        callbacks=[lr_sched, early_stopping],
        verbose=2)
```

```
Train on 7177 samples, validate on 1795 samples
Epoch 1/200

Epoch 00001: LearningRateScheduler setting learning rate to 0.001.
 - 1s - loss: 0.1295 - mae: 0.3076 - val_loss: 0.0833 - val_mae: 0.2234
Epoch 2/200

Epoch 00002: LearningRateScheduler setting learning rate to
0.0009699999999999999.
 - 0s - loss: 0.0466 - mae: 0.1778 - val_loss: 0.1110 - val_mae: 0.2623
Epoch 3/200

Epoch 00003: LearningRateScheduler setting learning rate to
0.0009408999999999999.

/Users/vipulgaur/opt/anaconda3/lib/python3.7/site-
packages/keras/callbacks/callbacks.py:846: RuntimeWarning: Early stopping
conditioned on metric `val_mean_absolute_error` which is not available.
Available metrics are: val_loss,val_mae,loss,mae,lr
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
```

```
 - 0s - loss: 0.0296 - mae: 0.1353 - val_loss: 0.1203 - val_mae: 0.2736
Epoch 4/200

Epoch 00004: LearningRateScheduler setting learning rate to 0.000912673.
 - 0s - loss: 0.0283 - mae: 0.1279 - val_loss: 0.1238 - val_mae: 0.2774
Epoch 5/200

Epoch 00005: LearningRateScheduler setting learning rate to 0.00088529281.
 - 0s - loss: 0.0283 - mae: 0.1253 - val_loss: 0.1246 - val_mae: 0.2785
Epoch 6/200

Epoch 00006: LearningRateScheduler setting learning rate to 0.0008587340257.
 - 0s - loss: 0.0288 - mae: 0.1255 - val_loss: 0.1242 - val_mae: 0.2785
Epoch 7/200

Epoch 00007: LearningRateScheduler setting learning rate to
0.0008329720049289998.
 - 0s - loss: 0.0283 - mae: 0.1241 - val_loss: 0.1235 - val_mae: 0.2777
Epoch 8/200

Epoch 00008: LearningRateScheduler setting learning rate to
0.0008079828447811299.
 - 0s - loss: 0.0270 - mae: 0.1209 - val_loss: 0.1225 - val_mae: 0.2768
Epoch 9/200

Epoch 00009: LearningRateScheduler setting learning rate to
0.0007837433594376959.
 - 0s - loss: 0.0261 - mae: 0.1192 - val_loss: 0.1212 - val_mae: 0.2757
Epoch 10/200

Epoch 00010: LearningRateScheduler setting learning rate to
0.0007602310586545651.
 - 0s - loss: 0.0253 - mae: 0.1180 - val_loss: 0.1195 - val_mae: 0.2740
Epoch 11/200

Epoch 00011: LearningRateScheduler setting learning rate to
0.0007374241268949281.
 - 0s - loss: 0.0246 - mae: 0.1163 - val_loss: 0.1177 - val_mae: 0.2720
Epoch 12/200

Epoch 00012: LearningRateScheduler setting learning rate to
0.0007153014030880803.
 - 0s - loss: 0.0242 - mae: 0.1166 - val_loss: 0.1157 - val_mae: 0.2698
Epoch 13/200

Epoch 00013: LearningRateScheduler setting learning rate to
0.0006938423609954377.
 - 0s - loss: 0.0236 - mae: 0.1155 - val_loss: 0.1136 - val_mae: 0.2674
```

```
Epoch 14/200

Epoch 00014: LearningRateScheduler setting learning rate to
0.0006730270901655746.
 - 0s - loss: 0.0237 - mae: 0.1160 - val_loss: 0.1116 - val_mae: 0.2650
Epoch 15/200

Epoch 00015: LearningRateScheduler setting learning rate to
0.0006528362774606074.
 - 0s - loss: 0.0237 - mae: 0.1162 - val_loss: 0.1096 - val_mae: 0.2626
Epoch 16/200

Epoch 00016: LearningRateScheduler setting learning rate to
0.0006332511891367892.
 - 0s - loss: 0.0237 - mae: 0.1173 - val_loss: 0.1075 - val_mae: 0.2600
Epoch 17/200

Epoch 00017: LearningRateScheduler setting learning rate to
0.0006142536534626855.
 - 0s - loss: 0.0234 - mae: 0.1164 - val_loss: 0.1053 - val_mae: 0.2572
Epoch 18/200

Epoch 00018: LearningRateScheduler setting learning rate to
0.0005958260438588049.
 - 0s - loss: 0.0235 - mae: 0.1165 - val_loss: 0.1032 - val_mae: 0.2542
Epoch 19/200

Epoch 00019: LearningRateScheduler setting learning rate to
0.0005779512625430407.
 - 0s - loss: 0.0232 - mae: 0.1158 - val_loss: 0.1009 - val_mae: 0.2510
Epoch 20/200

Epoch 00020: LearningRateScheduler setting learning rate to
0.0005606127246667495.
 - 0s - loss: 0.0229 - mae: 0.1152 - val_loss: 0.0986 - val_mae: 0.2478
Epoch 21/200

Epoch 00021: LearningRateScheduler setting learning rate to
0.0005437943429267 47.
 - 0s - loss: 0.0226 - mae: 0.1148 - val_loss: 0.0963 - val_mae: 0.2445
Epoch 22/200

Epoch 00022: LearningRateScheduler setting learning rate to
0.0005274805126389446.
 - 0s - loss: 0.0230 - mae: 0.1152 - val_loss: 0.0940 - val_mae: 0.2411
Epoch 23/200

Epoch 00023: LearningRateScheduler setting learning rate to
```

```
0.0005116560972597763.
 - 0s - loss: 0.0230 - mae: 0.1148 - val_loss: 0.0917 - val_mae: 0.2378
Epoch 24/200

Epoch 00024: LearningRateScheduler setting learning rate to
0.0004963064143419829.
 - 0s - loss: 0.0223 - mae: 0.1132 - val_loss: 0.0895 - val_mae: 0.2344
Epoch 25/200

Epoch 00025: LearningRateScheduler setting learning rate to
0.00048141722191172336.
 - 0s - loss: 0.0224 - mae: 0.1133 - val_loss: 0.0873 - val_mae: 0.2311
Epoch 26/200

Epoch 00026: LearningRateScheduler setting learning rate to
0.0004669747052543717.
 - 0s - loss: 0.0220 - mae: 0.1123 - val_loss: 0.0852 - val_mae: 0.2279
Epoch 27/200

Epoch 00027: LearningRateScheduler setting learning rate to
0.0004529654640967405.
 - 0s - loss: 0.0222 - mae: 0.1119 - val_loss: 0.0832 - val_mae: 0.2247
Epoch 28/200

Epoch 00028: LearningRateScheduler setting learning rate to
0.0004393765001738383.
 - 0s - loss: 0.0220 - mae: 0.1113 - val_loss: 0.0813 - val_mae: 0.2216
Epoch 29/200

Epoch 00029: LearningRateScheduler setting learning rate to
0.0004261952051686231.
 - 0s - loss: 0.0221 - mae: 0.1115 - val_loss: 0.0794 - val_mae: 0.2185
Epoch 30/200

Epoch 00030: LearningRateScheduler setting learning rate to
0.0004134093490135644.
 - 0s - loss: 0.0221 - mae: 0.1114 - val_loss: 0.0776 - val_mae: 0.2156
Epoch 31/200

Epoch 00031: LearningRateScheduler setting learning rate to
0.00040100706854315747.
 - 0s - loss: 0.0220 - mae: 0.1110 - val_loss: 0.0758 - val_mae: 0.2127
Epoch 32/200

Epoch 00032: LearningRateScheduler setting learning rate to
0.00038897685648686274.
 - 0s - loss: 0.0220 - mae: 0.1108 - val_loss: 0.0741 - val_mae: 0.2099
Epoch 33/200
```

```
Epoch 00033: LearningRateScheduler setting learning rate to
0.00037730755079225687.
 - 0s - loss: 0.0218 - mae: 0.1105 - val_loss: 0.0724 - val_mae: 0.2072
Epoch 34/200

Epoch 00034: LearningRateScheduler setting learning rate to
0.00036598832426848916.
 - 0s - loss: 0.0217 - mae: 0.1104 - val_loss: 0.0708 - val_mae: 0.2046
Epoch 35/200

Epoch 00035: LearningRateScheduler setting learning rate to
0.00035500867454043444.
 - 0s - loss: 0.0216 - mae: 0.1105 - val_loss: 0.0692 - val_mae: 0.2021
Epoch 36/200

Epoch 00036: LearningRateScheduler setting learning rate to
0.00034435843143042214.
 - 0s - loss: 0.0216 - mae: 0.1101 - val_loss: 0.0676 - val_mae: 0.1997
Epoch 37/200

Epoch 00037: LearningRateScheduler setting learning rate to
0.00033402766187509475.
 - 0s - loss: 0.0216 - mae: 0.1103 - val_loss: 0.0662 - val_mae: 0.1973
Epoch 38/200

Epoch 00038: LearningRateScheduler setting learning rate to
0.00032400683201884195.
 - 0s - loss: 0.0218 - mae: 0.1111 - val_loss: 0.0647 - val_mae: 0.1950
Epoch 39/200

Epoch 00039: LearningRateScheduler setting learning rate to
0.00031428662705827666.
 - 0s - loss: 0.0215 - mae: 0.1098 - val_loss: 0.0634 - val_mae: 0.1927
Epoch 40/200

Epoch 00040: LearningRateScheduler setting learning rate to
0.00030485802824652834.
 - 0s - loss: 0.0216 - mae: 0.1104 - val_loss: 0.0620 - val_mae: 0.1906
Epoch 41/200

Epoch 00041: LearningRateScheduler setting learning rate to
0.00029571228738913253.
 - 0s - loss: 0.0214 - mae: 0.1098 - val_loss: 0.0608 - val_mae: 0.1885
Epoch 42/200

Epoch 00042: LearningRateScheduler setting learning rate to
0.00028684091877715853.
```

```
 - 0s - loss: 0.0216 - mae: 0.1102 - val_loss: 0.0596 - val_mae: 0.1864
Epoch 43/200

Epoch 00043: LearningRateScheduler setting learning rate to
0.00027823569121384375.
 - 0s - loss: 0.0212 - mae: 0.1093 - val_loss: 0.0584 - val_mae: 0.1844
Epoch 44/200

Epoch 00044: LearningRateScheduler setting learning rate to
0.00026988886204774284.
 - 0s - loss: 0.0213 - mae: 0.1098 - val_loss: 0.0573 - val_mae: 0.1824
Epoch 45/200

Epoch 00045: LearningRateScheduler setting learning rate to
0.00026179196186310554.
 - 0s - loss: 0.0215 - mae: 0.1101 - val_loss: 0.0562 - val_mae: 0.1805
Epoch 46/200

Epoch 00046: LearningRateScheduler setting learning rate to
0.0002539382030072124.
 - 0s - loss: 0.0211 - mae: 0.1093 - val_loss: 0.0552 - val_mae: 0.1787
Epoch 47/200

Epoch 00047: LearningRateScheduler setting learning rate to
0.00024632005691699 96.
 - 0s - loss: 0.0211 - mae: 0.1094 - val_loss: 0.0542 - val_mae: 0.1769
Epoch 48/200

Epoch 00048: LearningRateScheduler setting learning rate to
0.00023893045520948612.
 - 0s - loss: 0.0211 - mae: 0.1097 - val_loss: 0.0532 - val_mae: 0.1752
Epoch 49/200

Epoch 00049: LearningRateScheduler setting learning rate to
0.00023176254155320153.
 - 0s - loss: 0.0211 - mae: 0.1095 - val_loss: 0.0523 - val_mae: 0.1734
Epoch 50/200

Epoch 00050: LearningRateScheduler setting learning rate to
0.00022480966530660546.
 - 0s - loss: 0.0213 - mae: 0.1099 - val_loss: 0.0514 - val_mae: 0.1718
Epoch 51/200

Epoch 00051: LearningRateScheduler setting learning rate to
0.00021806537534740 73.
 - 0s - loss: 0.0211 - mae: 0.1090 - val_loss: 0.0505 - val_mae: 0.1701
Epoch 52/200
```

```
Epoch 00052: LearningRateScheduler setting learning rate to
0.00021152341408698508.
 - 0s - loss: 0.0209 - mae: 0.1089 - val_loss: 0.0497 - val_mae: 0.1685
Epoch 53/200

Epoch 00053: LearningRateScheduler setting learning rate to
0.00020251777116643755.
 - 0s - loss: 0.0210 - mae: 0.1091 - val_loss: 0.0489 - val_mae: 0.1670
Epoch 54/200

Epoch 00054: LearningRateScheduler setting learning rate to
0.00019902238031444425.
 - 0s - loss: 0.0210 - mae: 0.1097 - val_loss: 0.0481 - val_mae: 0.1655
Epoch 55/200

Epoch 00055: LearningRateScheduler setting learning rate to
0.00019305170890501 09.
 - 0s - loss: 0.0210 - mae: 0.1089 - val_loss: 0.0474 - val_mae: 0.1640
Epoch 56/200

Epoch 00056: LearningRateScheduler setting learning rate to
0.00018726015763786058.
 - 0s - loss: 0.0209 - mae: 0.1092 - val_loss: 0.0466 - val_mae: 0.1626
Epoch 57/200

Epoch 00057: LearningRateScheduler setting learning rate to
0.00018164235290872477.
 - 0s - loss: 0.0210 - mae: 0.1093 - val_loss: 0.0459 - val_mae: 0.1612
Epoch 58/200

Epoch 00058: LearningRateScheduler setting learning rate to
0.00017619308232 1463.
 - 0s - loss: 0.0209 - mae: 0.1089 - val_loss: 0.0452 - val_mae: 0.1598
Epoch 59/200

Epoch 00059: LearningRateScheduler setting learning rate to
0.00017090728985 18191.
 - 0s - loss: 0.0209 - mae: 0.1094 - val_loss: 0.0446 - val_mae: 0.1584
Epoch 60/200

Epoch 00060: LearningRateScheduler setting learning rate to
0.00016578007115626453.
 - 0s - loss: 0.0208 - mae: 0.1084 - val_loss: 0.0439 - val_mae: 0.1571
Epoch 61/200

Epoch 00061: LearningRateScheduler setting learning rate to
0.00016080666902157 66.
 - 0s - loss: 0.0207 - mae: 0.1084 - val_loss: 0.0433 - val_mae: 0.1558
```

```
Epoch 62/200

Epoch 00062: LearningRateScheduler setting learning rate to
0.0001559824689509293.
 - 0s - loss: 0.0206 - mae: 0.1082 - val_loss: 0.0426 - val_mae: 0.1546
Epoch 63/200

Epoch 00063: LearningRateScheduler setting learning rate to
0.0001513029948824014.
 - 0s - loss: 0.0208 - mae: 0.1086 - val_loss: 0.0420 - val_mae: 0.1534
Epoch 64/200

Epoch 00064: LearningRateScheduler setting learning rate to
0.00014676390503592937.
 - 0s - loss: 0.0206 - mae: 0.1085 - val_loss: 0.0414 - val_mae: 0.1522
Epoch 65/200

Epoch 00065: LearningRateScheduler setting learning rate to
0.0001423609878848515.
 - 0s - loss: 0.0207 - mae: 0.1081 - val_loss: 0.0409 - val_mae: 0.1510
Epoch 66/200

Epoch 00066: LearningRateScheduler setting learning rate to
0.00013809015824830593.
 - 0s - loss: 0.0205 - mae: 0.1076 - val_loss: 0.0403 - val_mae: 0.1499
Epoch 67/200

Epoch 00067: LearningRateScheduler setting learning rate to
0.00013394745350085675.
 - 0s - loss: 0.0207 - mae: 0.1083 - val_loss: 0.0398 - val_mae: 0.1487
Epoch 68/200

Epoch 00068: LearningRateScheduler setting learning rate to
0.00012992902989583105.
 - 0s - loss: 0.0206 - mae: 0.1080 - val_loss: 0.0392 - val_mae: 0.1476
Epoch 69/200

Epoch 00069: LearningRateScheduler setting learning rate to
0.00012603115899895612.
 - 0s - loss: 0.0206 - mae: 0.1085 - val_loss: 0.0387 - val_mae: 0.1465
Epoch 70/200

Epoch 00070: LearningRateScheduler setting learning rate to
0.00012225022422898742.
 - 0s - loss: 0.0208 - mae: 0.1085 - val_loss: 0.0382 - val_mae: 0.1454
Epoch 71/200

Epoch 00071: LearningRateScheduler setting learning rate to
```

```
0.0001185827175021178.
 - 0s - loss: 0.0207 - mae: 0.1085 - val_loss: 0.0377 - val_mae: 0.1444
Epoch 72/200

Epoch 00072: LearningRateScheduler setting learning rate to
0.00011502523597705427.
 - 0s - loss: 0.0207 - mae: 0.1080 - val_loss: 0.0372 - val_mae: 0.1434
Epoch 73/200

Epoch 00073: LearningRateScheduler setting learning rate to
0.00011157447889774264.
 - 0s - loss: 0.0202 - mae: 0.1072 - val_loss: 0.0368 - val_mae: 0.1424
Epoch 74/200

Epoch 00074: LearningRateScheduler setting learning rate to
0.00010822724453081035.
 - 0s - loss: 0.0204 - mae: 0.1078 - val_loss: 0.0363 - val_mae: 0.1414
Epoch 75/200

Epoch 00075: LearningRateScheduler setting learning rate to
0.00010498042719488605.
 - 0s - loss: 0.0208 - mae: 0.1081 - val_loss: 0.0359 - val_mae: 0.1404
Epoch 76/200

Epoch 00076: LearningRateScheduler setting learning rate to
0.00010183101437903946.
 - 0s - loss: 0.0202 - mae: 0.1072 - val_loss: 0.0354 - val_mae: 0.1395
Epoch 77/200

Epoch 00077: LearningRateScheduler setting learning rate to
9.877608394766827e-05.
 - 0s - loss: 0.0203 - mae: 0.1074 - val_loss: 0.0350 - val_mae: 0.1386
Epoch 78/200

Epoch 00078: LearningRateScheduler setting learning rate to
9.581280142923822e-05.
 - 0s - loss: 0.0205 - mae: 0.1083 - val_loss: 0.0346 - val_mae: 0.1377
Epoch 79/200

Epoch 00079: LearningRateScheduler setting learning rate to
9.293841738636107e-05.
 - 0s - loss: 0.0202 - mae: 0.1074 - val_loss: 0.0342 - val_mae: 0.1368
Epoch 80/200

Epoch 00080: LearningRateScheduler setting learning rate to
9.015026486477024e-05.
 - 0s - loss: 0.0204 - mae: 0.1074 - val_loss: 0.0338 - val_mae: 0.1359
Epoch 81/200
```

```
Epoch 00081: LearningRateScheduler setting learning rate to
8.744575691882712e-05.
 - 0s - loss: 0.0202 - mae: 0.1070 - val_loss: 0.0334 - val_mae: 0.1351
Epoch 82/200

Epoch 00082: LearningRateScheduler setting learning rate to
8.48223842112623e-05.
 - 0s - loss: 0.0204 - mae: 0.1077 - val_loss: 0.0331 - val_mae: 0.1343
Epoch 83/200

Epoch 00083: LearningRateScheduler setting learning rate to
8.227771268492445e-05.
 - 0s - loss: 0.0206 - mae: 0.1081 - val_loss: 0.0327 - val_mae: 0.1335
Epoch 84/200

Epoch 00084: LearningRateScheduler setting learning rate to
7.98093813043767e-05.
 - 0s - loss: 0.0204 - mae: 0.1071 - val_loss: 0.0324 - val_mae: 0.1327
Epoch 85/200

Epoch 00085: LearningRateScheduler setting learning rate to
7.741509986524539e-05.
 - 0s - loss: 0.0203 - mae: 0.1074 - val_loss: 0.0320 - val_mae: 0.1319
Epoch 86/200

Epoch 00086: LearningRateScheduler setting learning rate to
7.509264686928803e-05.
 - 0s - loss: 0.0205 - mae: 0.1072 - val_loss: 0.0317 - val_mae: 0.1311
Epoch 87/200

Epoch 00087: LearningRateScheduler setting learning rate to
7.283986746320939e-05.
 - 0s - loss: 0.0204 - mae: 0.1070 - val_loss: 0.0314 - val_mae: 0.1304
Epoch 88/200

Epoch 00088: LearningRateScheduler setting learning rate to
7.06546714393131e-05.
 - 0s - loss: 0.0202 - mae: 0.1070 - val_loss: 0.0311 - val_mae: 0.1296
Epoch 89/200

Epoch 00089: LearningRateScheduler setting learning rate to
6.853503129613371e-05.
 - 0s - loss: 0.0205 - mae: 0.1076 - val_loss: 0.0308 - val_mae: 0.1289
Epoch 90/200

Epoch 00090: LearningRateScheduler setting learning rate to
6.64789803572497e-05.
```

```
 - 0s - loss: 0.0202 - mae: 0.1071 - val_loss: 0.0305 - val_mae: 0.1283
Epoch 91/200

Epoch 00091: LearningRateScheduler setting learning rate to
6.44846109465322e-05.
 - 0s - loss: 0.0200 - mae: 0.1063 - val_loss: 0.0302 - val_mae: 0.1276
Epoch 92/200

Epoch 00092: LearningRateScheduler setting learning rate to
6.255007261813624e-05.
 - 0s - loss: 0.0202 - mae: 0.1069 - val_loss: 0.0299 - val_mae: 0.1270
Epoch 93/200

Epoch 00093: LearningRateScheduler setting learning rate to
6.067357043959214e-05.
 - 0s - loss: 0.0205 - mae: 0.1075 - val_loss: 0.0296 - val_mae: 0.1263
Epoch 94/200

Epoch 00094: LearningRateScheduler setting learning rate to
5.8853363326404384e-05.
 - 0s - loss: 0.0203 - mae: 0.1070 - val_loss: 0.0294 - val_mae: 0.1257
Epoch 95/200

Epoch 00095: LearningRateScheduler setting learning rate to
5.7087762426612245e-05.
 - 0s - loss: 0.0205 - mae: 0.1071 - val_loss: 0.0291 - val_mae: 0.1251
Epoch 96/200

Epoch 00096: LearningRateScheduler setting learning rate to
5.537512955381388e-05.
 - 0s - loss: 0.0202 - mae: 0.1068 - val_loss: 0.0289 - val_mae: 0.1245
Epoch 97/200

Epoch 00097: LearningRateScheduler setting learning rate to
5.371387566719946e-05.
 - 0s - loss: 0.0201 - mae: 0.1064 - val_loss: 0.0286 - val_mae: 0.1239
Epoch 98/200

Epoch 00098: LearningRateScheduler setting learning rate to
5.210245939718348e-05.
 - 0s - loss: 0.0201 - mae: 0.1069 - val_loss: 0.0284 - val_mae: 0.1234
Epoch 99/200

Epoch 00099: LearningRateScheduler setting learning rate to
5.053938561526797e-05.
 - 0s - loss: 0.0202 - mae: 0.1066 - val_loss: 0.0281 - val_mae: 0.1228
Epoch 100/200
```

```
Epoch 00100: LearningRateScheduler setting learning rate to
4.9023204046809934e-05.
 - 0s - loss: 0.0201 - mae: 0.1066 - val_loss: 0.0279 - val_mae: 0.1223
Epoch 101/200

Epoch 00101: LearningRateScheduler setting learning rate to
4.755250792540563e-05.
 - 0s - loss: 0.0200 - mae: 0.1067 - val_loss: 0.0277 - val_mae: 0.1218
Epoch 102/200

Epoch 00102: LearningRateScheduler setting learning rate to
4.612593268764346e-05.
 - 0s - loss: 0.0204 - mae: 0.1072 - val_loss: 0.0275 - val_mae: 0.1213
Epoch 103/200

Epoch 00103: LearningRateScheduler setting learning rate to
4.474215470701415e-05.
 - 0s - loss: 0.0200 - mae: 0.1068 - val_loss: 0.0273 - val_mae: 0.1208
Epoch 104/200

Epoch 00104: LearningRateScheduler setting learning rate to
4.339989006580373e-05.
 - 0s - loss: 0.0202 - mae: 0.1066 - val_loss: 0.0271 - val_mae: 0.1203
Epoch 105/200

Epoch 00105: LearningRateScheduler setting learning rate to
4.2097893363829616e-05.
 - 0s - loss: 0.0201 - mae: 0.1073 - val_loss: 0.0269 - val_mae: 0.1198
Epoch 106/200

Epoch 00106: LearningRateScheduler setting learning rate to
4.083495656291473e-05.
 - 0s - loss: 0.0203 - mae: 0.1071 - val_loss: 0.0267 - val_mae: 0.1194
Epoch 107/200

Epoch 00107: LearningRateScheduler setting learning rate to
3.960990786602728e-05.
 - 0s - loss: 0.0200 - mae: 0.1061 - val_loss: 0.0265 - val_mae: 0.1190
Epoch 108/200

Epoch 00108: LearningRateScheduler setting learning rate to
3.842161063004646e-05.
 - 0s - loss: 0.0203 - mae: 0.1067 - val_loss: 0.0263 - val_mae: 0.1185
Epoch 109/200

Epoch 00109: LearningRateScheduler setting learning rate to
3.726896231114507e-05.
 - 0s - loss: 0.0203 - mae: 0.1073 - val_loss: 0.0261 - val_mae: 0.1181
```

```
Epoch 110/200

Epoch 00110: LearningRateScheduler setting learning rate to
3.615089344181072e-05.
 - 0s - loss: 0.0201 - mae: 0.1065 - val_loss: 0.0260 - val_mae: 0.1178
Epoch 111/200

Epoch 00111: LearningRateScheduler setting learning rate to
3.5066366638556394e-05.
 - 0s - loss: 0.0203 - mae: 0.1070 - val_loss: 0.0258 - val_mae: 0.1174
Epoch 112/200

Epoch 00112: LearningRateScheduler setting learning rate to
3.40143756393997e-05.
 - 0s - loss: 0.0200 - mae: 0.1065 - val_loss: 0.0256 - val_mae: 0.1170
Epoch 113/200

Epoch 00113: LearningRateScheduler setting learning rate to
3.299394437021771e-05.
 - 0s - loss: 0.0202 - mae: 0.1068 - val_loss: 0.0255 - val_mae: 0.1167
Epoch 114/200

Epoch 00114: LearningRateScheduler setting learning rate to
3.200412603911118e-05.
 - 0s - loss: 0.0201 - mae: 0.1063 - val_loss: 0.0253 - val_mae: 0.1163
Epoch 115/200

Epoch 00115: LearningRateScheduler setting learning rate to
3.1044002257937845e-05.
 - 0s - loss: 0.0199 - mae: 0.1065 - val_loss: 0.0252 - val_mae: 0.1160
Epoch 116/200

Epoch 00116: LearningRateScheduler setting learning rate to
3.0112682190199705e-05.
 - 0s - loss: 0.0200 - mae: 0.1068 - val_loss: 0.0250 - val_mae: 0.1157
Epoch 117/200

Epoch 00117: LearningRateScheduler setting learning rate to
2.9209301724493712e-05.
 - 0s - loss: 0.0202 - mae: 0.1065 - val_loss: 0.0249 - val_mae: 0.1154
Epoch 118/200

Epoch 00118: LearningRateScheduler setting learning rate to
2.83330226727589e-05.
 - 0s - loss: 0.0201 - mae: 0.1069 - val_loss: 0.0247 - val_mae: 0.1150
Epoch 119/200

Epoch 00119: LearningRateScheduler setting learning rate to
```

```
2.7483031992576134e-05.
 - 0s - loss: 0.0199 - mae: 0.1062 - val_loss: 0.0246 - val_mae: 0.1148
Epoch 120/200

Epoch 00120: LearningRateScheduler setting learning rate to
2.6658541032798848e-05.
 - 0s - loss: 0.0201 - mae: 0.1070 - val_loss: 0.0245 - val_mae: 0.1145
Epoch 121/200

Epoch 00121: LearningRateScheduler setting learning rate to
2.5858784801814883e-05.
 - 0s - loss: 0.0198 - mae: 0.1056 - val_loss: 0.0244 - val_mae: 0.1142
Epoch 122/200

Epoch 00122: LearningRateScheduler setting learning rate to
2.5083021257760435e-05.
 - 0s - loss: 0.0202 - mae: 0.1069 - val_loss: 0.0242 - val_mae: 0.1139
Epoch 123/200

Epoch 00123: LearningRateScheduler setting learning rate to
2.433053062002762e-05.
 - 0s - loss: 0.0200 - mae: 0.1065 - val_loss: 0.0241 - val_mae: 0.1136
Epoch 124/200

Epoch 00124: LearningRateScheduler setting learning rate to
2.360061470142679e-05.
 - 0s - loss: 0.0201 - mae: 0.1063 - val_loss: 0.0240 - val_mae: 0.1133
Epoch 125/200

Epoch 00125: LearningRateScheduler setting learning rate to
2.2892596260383987e-05.
 - 0s - loss: 0.0200 - mae: 0.1066 - val_loss: 0.0239 - val_mae: 0.1131
Epoch 126/200

Epoch 00126: LearningRateScheduler setting learning rate to
2.220581837257247e-05.
 - 0s - loss: 0.0201 - mae: 0.1064 - val_loss: 0.0238 - val_mae: 0.1128
Epoch 127/200

Epoch 00127: LearningRateScheduler setting learning rate to
2.1539643821395293e-05.
 - 0s - loss: 0.0200 - mae: 0.1063 - val_loss: 0.0237 - val_mae: 0.1126
Epoch 128/200

Epoch 00128: LearningRateScheduler setting learning rate to
2.0893454506753432e-05.
 - 0s - loss: 0.0199 - mae: 0.1065 - val_loss: 0.0236 - val_mae: 0.1124
Epoch 129/200
```

```
Epoch 00129: LearningRateScheduler setting learning rate to
2.0266650871550828e-05.
 - 0s - loss: 0.0203 - mae: 0.1071 - val_loss: 0.0234 - val_mae: 0.1121
Epoch 130/200

Epoch 00130: LearningRateScheduler setting learning rate to
1.9658651345404306e-05.
 - 0s - loss: 0.0199 - mae: 0.1059 - val_loss: 0.0233 - val_mae: 0.1119
Epoch 131/200

Epoch 00131: LearningRateScheduler setting learning rate to
1.906889180504218e-05.
 - 0s - loss: 0.0201 - mae: 0.1061 - val_loss: 0.0233 - val_mae: 0.1117
Epoch 132/200

Epoch 00132: LearningRateScheduler setting learning rate to
1.849682505089091e-05.
 - 0s - loss: 0.0200 - mae: 0.1064 - val_loss: 0.0232 - val_mae: 0.1115
Epoch 133/200

Epoch 00133: LearningRateScheduler setting learning rate to
1.7941920299364185e-05.
 - 0s - loss: 0.0200 - mae: 0.1062 - val_loss: 0.0231 - val_mae: 0.1113
Epoch 134/200

Epoch 00134: LearningRateScheduler setting learning rate to
1.7403662690383253e-05.
 - 0s - loss: 0.0203 - mae: 0.1070 - val_loss: 0.0230 - val_mae: 0.1111
Epoch 135/200

Epoch 00135: LearningRateScheduler setting learning rate to
1.6881552809671757e-05.
 - 0s - loss: 0.0201 - mae: 0.1066 - val_loss: 0.0229 - val_mae: 0.1109
Epoch 136/200

Epoch 00136: LearningRateScheduler setting learning rate to
1.6375106225381605e-05.
 - 0s - loss: 0.0201 - mae: 0.1061 - val_loss: 0.0228 - val_mae: 0.1107
Epoch 137/200

Epoch 00137: LearningRateScheduler setting learning rate to
1.5883853038620156e-05.
 - 0s - loss: 0.0200 - mae: 0.1063 - val_loss: 0.0227 - val_mae: 0.1105
Epoch 138/200

Epoch 00138: LearningRateScheduler setting learning rate to
1.540733744746155e-05.
```

```
 - 0s - loss: 0.0201 - mae: 0.1064 - val_loss: 0.0226 - val_mae: 0.1103
Epoch 139/200

Epoch 00139: LearningRateScheduler setting learning rate to
1.4945117324037704e-05.
 - 0s - loss: 0.0200 - mae: 0.1068 - val_loss: 0.0226 - val_mae: 0.1101
Epoch 140/200

Epoch 00140: LearningRateScheduler setting learning rate to
1.4496763804316573e-05.
 - 0s - loss: 0.0202 - mae: 0.1066 - val_loss: 0.0225 - val_mae: 0.1100
Epoch 141/200

Epoch 00141: LearningRateScheduler setting learning rate to
1.4061860890187074e-05.
 - 0s - loss: 0.0202 - mae: 0.1064 - val_loss: 0.0224 - val_mae: 0.1098
Epoch 142/200

Epoch 00142: LearningRateScheduler setting learning rate to
1.3640005063481462e-05.
 - 0s - loss: 0.0203 - mae: 0.1070 - val_loss: 0.0223 - val_mae: 0.1097
Epoch 143/200

Epoch 00143: LearningRateScheduler setting learning rate to
1.3230804911577018e-05.
 - 0s - loss: 0.0200 - mae: 0.1062 - val_loss: 0.0223 - val_mae: 0.1095
Epoch 144/200

Epoch 00144: LearningRateScheduler setting learning rate to
1.2833880764229706e-05.
 - 0s - loss: 0.0199 - mae: 0.1063 - val_loss: 0.0222 - val_mae: 0.1094
Epoch 145/200

Epoch 00145: LearningRateScheduler setting learning rate to
1.2448864341302816e-05.
 - 0s - loss: 0.0198 - mae: 0.1060 - val_loss: 0.0221 - val_mae: 0.1092
Epoch 146/200

Epoch 00146: LearningRateScheduler setting learning rate to
1.2075398411063731e-05.
 - 0s - loss: 0.0201 - mae: 0.1065 - val_loss: 0.0220 - val_mae: 0.1091
Epoch 147/200

Epoch 00147: LearningRateScheduler setting learning rate to
1.1713136458731819e-05.
 - 0s - loss: 0.0201 - mae: 0.1064 - val_loss: 0.0220 - val_mae: 0.1089
Epoch 148/200
```

```
Epoch 00148: LearningRateScheduler setting learning rate to
1.1361742364969863e-05.
 - 0s - loss: 0.0201 - mae: 0.1066 - val_loss: 0.0219 - val_mae: 0.1088
Epoch 149/200

Epoch 00149: LearningRateScheduler setting learning rate to
1.1020890094020768e-05.
 - 0s - loss: 0.0200 - mae: 0.1063 - val_loss: 0.0219 - val_mae: 0.1087
Epoch 150/200

Epoch 00150: LearningRateScheduler setting learning rate to
1.0690263391200144e-05.
 - 0s - loss: 0.0199 - mae: 0.1058 - val_loss: 0.0218 - val_mae: 0.1085
Epoch 151/200

Epoch 00151: LearningRateScheduler setting learning rate to
1.0369555489464138e-05.
 - 0s - loss: 0.0199 - mae: 0.1062 - val_loss: 0.0217 - val_mae: 0.1084
Epoch 152/200

Epoch 00152: LearningRateScheduler setting learning rate to
1.0058468824780214e-05.
 - 0s - loss: 0.0199 - mae: 0.1058 - val_loss: 0.0217 - val_mae: 0.1083
Epoch 153/200

Epoch 00153: LearningRateScheduler setting learning rate to
9.756714760036808e-06.
 - 0s - loss: 0.0201 - mae: 0.1066 - val_loss: 0.0216 - val_mae: 0.1082
Epoch 154/200

Epoch 00154: LearningRateScheduler setting learning rate to
9.464013317235704e-06.
 - 0s - loss: 0.0204 - mae: 0.1071 - val_loss: 0.0216 - val_mae: 0.1080
Epoch 155/200

Epoch 00155: LearningRateScheduler setting learning rate to
9.180092917718633e-06.
 - 0s - loss: 0.0199 - mae: 0.1062 - val_loss: 0.0215 - val_mae: 0.1079
Epoch 156/200

Epoch 00156: LearningRateScheduler setting learning rate to
8.904690130187072e-06.
 - 0s - loss: 0.0200 - mae: 0.1061 - val_loss: 0.0215 - val_mae: 0.1078
Epoch 157/200

Epoch 00157: LearningRateScheduler setting learning rate to
8.63754942628146e-06.
 - 0s - loss: 0.0202 - mae: 0.1066 - val_loss: 0.0214 - val_mae: 0.1077
```

```
Epoch 158/200

Epoch 00158: LearningRateScheduler setting learning rate to
8.378422943493016e-06.
 - 0s - loss: 0.0199 - mae: 0.1064 - val_loss: 0.0214 - val_mae: 0.1076
Epoch 159/200

Epoch 00159: LearningRateScheduler setting learning rate to
8.127070255188227e-06.
 - 0s - loss: 0.0197 - mae: 0.1059 - val_loss: 0.0213 - val_mae: 0.1075
Epoch 160/200

Epoch 00160: LearningRateScheduler setting learning rate to
7.883258147532578e-06.
 - 0s - loss: 0.0200 - mae: 0.1062 - val_loss: 0.0213 - val_mae: 0.1074
Epoch 161/200

Epoch 00161: LearningRateScheduler setting learning rate to
7.646760403106601e-06.
 - 0s - loss: 0.0199 - mae: 0.1062 - val_loss: 0.0212 - val_mae: 0.1073
Epoch 162/200

Epoch 00162: LearningRateScheduler setting learning rate to
7.417357591013403e-06.
 - 0s - loss: 0.0199 - mae: 0.1063 - val_loss: 0.0212 - val_mae: 0.1072
Epoch 163/200

Epoch 00163: LearningRateScheduler setting learning rate to 7.194836863283e-06.
 - 0s - loss: 0.0199 - mae: 0.1061 - val_loss: 0.0211 - val_mae: 0.1071
Epoch 164/200

Epoch 00164: LearningRateScheduler setting learning rate to
6.978991757384511e-06.
 - 0s - loss: 0.0198 - mae: 0.1057 - val_loss: 0.0211 - val_mae: 0.1070
Epoch 165/200

Epoch 00165: LearningRateScheduler setting learning rate to
6.769622004662975e-06.
 - 0s - loss: 0.0200 - mae: 0.1063 - val_loss: 0.0210 - val_mae: 0.1070
Epoch 166/200

Epoch 00166: LearningRateScheduler setting learning rate to
6.566533344523086e-06.
 - 0s - loss: 0.0198 - mae: 0.1059 - val_loss: 0.0210 - val_mae: 0.1069
Epoch 167/200

Epoch 00167: LearningRateScheduler setting learning rate to
6.369537344187393e-06.
```

```
 - 0s - loss: 0.0199 - mae: 0.1060 - val_loss: 0.0210 - val_mae: 0.1068
Epoch 168/200

Epoch 00168: LearningRateScheduler setting learning rate to
6.178451223861771e-06.
 - 0s - loss: 0.0206 - mae: 0.1078 - val_loss: 0.0209 - val_mae: 0.1067
Epoch 169/200

Epoch 00169: LearningRateScheduler setting learning rate to
5.9930976871459175e-06.
 - 0s - loss: 0.0200 - mae: 0.1058 - val_loss: 0.0209 - val_mae: 0.1066
Epoch 170/200

Epoch 00170: LearningRateScheduler setting learning rate to
5.81330475653154e-06.
 - 0s - loss: 0.0198 - mae: 0.1054 - val_loss: 0.0209 - val_mae: 0.1066
Epoch 171/200

Epoch 00171: LearningRateScheduler setting learning rate to
5.638905613835593e-06.
 - 0s - loss: 0.0200 - mae: 0.1067 - val_loss: 0.0208 - val_mae: 0.1065
Epoch 172/200

Epoch 00172: LearningRateScheduler setting learning rate to
5.4697384454205255e-06.
 - 0s - loss: 0.0201 - mae: 0.1063 - val_loss: 0.0208 - val_mae: 0.1064
Epoch 173/200

Epoch 00173: LearningRateScheduler setting learning rate to
5.30564629205791e-06.
 - 0s - loss: 0.0202 - mae: 0.1065 - val_loss: 0.0207 - val_mae: 0.1064
Epoch 174/200

Epoch 00174: LearningRateScheduler setting learning rate to
5.146476903296172e-06.
 - 0s - loss: 0.0198 - mae: 0.1058 - val_loss: 0.0207 - val_mae: 0.1063
Epoch 175/200

Epoch 00175: LearningRateScheduler setting learning rate to
4.992082596197287e-06.
 - 0s - loss: 0.0202 - mae: 0.1068 - val_loss: 0.0207 - val_mae: 0.1062
Epoch 176/200

Epoch 00176: LearningRateScheduler setting learning rate to
4.842320118311368e-06.
 - 0s - loss: 0.0199 - mae: 0.1058 - val_loss: 0.0206 - val_mae: 0.1062
Epoch 177/200
```

```
Epoch 00177: LearningRateScheduler setting learning rate to
4.6970505147620266e-06.
 - 0s - loss: 0.0199 - mae: 0.1063 - val_loss: 0.0206 - val_mae: 0.1061
Epoch 178/200

Epoch 00178: LearningRateScheduler setting learning rate to
4.556138999319166e-06.
 - 0s - loss: 0.0198 - mae: 0.1058 - val_loss: 0.0206 - val_mae: 0.1060
Epoch 179/200

Epoch 00179: LearningRateScheduler setting learning rate to
4.419454829339591e-06.
 - 0s - loss: 0.0197 - mae: 0.1053 - val_loss: 0.0206 - val_mae: 0.1060
Epoch 180/200

Epoch 00180: LearningRateScheduler setting learning rate to
4.286871184459403e-06.
 - 0s - loss: 0.0198 - mae: 0.1057 - val_loss: 0.0205 - val_mae: 0.1059
Epoch 181/200

Epoch 00181: LearningRateScheduler setting learning rate to
4.1582650489256206e-06.
 - 0s - loss: 0.0201 - mae: 0.1066 - val_loss: 0.0205 - val_mae: 0.1059
Epoch 182/200

Epoch 00182: LearningRateScheduler setting learning rate to
4.033517097457852e-06.
 - 0s - loss: 0.0200 - mae: 0.1064 - val_loss: 0.0205 - val_mae: 0.1058
Epoch 183/200

Epoch 00183: LearningRateScheduler setting learning rate to
3.912511584534117e-06.
 - 0s - loss: 0.0197 - mae: 0.1059 - val_loss: 0.0204 - val_mae: 0.1058
Epoch 184/200

Epoch 00184: LearningRateScheduler setting learning rate to
3.795136236998093e-06.
 - 0s - loss: 0.0202 - mae: 0.1066 - val_loss: 0.0204 - val_mae: 0.1057
Epoch 185/200

Epoch 00185: LearningRateScheduler setting learning rate to
3.68128214988815e-06.
 - 0s - loss: 0.0201 - mae: 0.1067 - val_loss: 0.0204 - val_mae: 0.1056
Epoch 186/200

Epoch 00186: LearningRateScheduler setting learning rate to
3.5708436853915056e-06.
 - 0s - loss: 0.0197 - mae: 0.1058 - val_loss: 0.0204 - val_mae: 0.1056
```

```
Epoch 187/200

Epoch 00187: LearningRateScheduler setting learning rate to
3.46371837482976e-06.
 - 0s - loss: 0.0196 - mae: 0.1054 - val_loss: 0.0203 - val_mae: 0.1056
Epoch 188/200

Epoch 00188: LearningRateScheduler setting learning rate to
3.3598068235848672e-06.
 - 0s - loss: 0.0202 - mae: 0.1073 - val_loss: 0.0203 - val_mae: 0.1055
Epoch 189/200

Epoch 00189: LearningRateScheduler setting learning rate to
3.2590126188773214e-06.
 - 0s - loss: 0.0197 - mae: 0.1058 - val_loss: 0.0203 - val_mae: 0.1055
Epoch 190/200

Epoch 00190: LearningRateScheduler setting learning rate to
3.1612422403110015e-06.
 - 0s - loss: 0.0198 - mae: 0.1056 - val_loss: 0.0203 - val_mae: 0.1054
Epoch 191/200

Epoch 00191: LearningRateScheduler setting learning rate to
3.066404973101671e-06.
 - 0s - loss: 0.0199 - mae: 0.1065 - val_loss: 0.0202 - val_mae: 0.1054
Epoch 192/200

Epoch 00192: LearningRateScheduler setting learning rate to
2.974412823908621e-06.
 - 0s - loss: 0.0198 - mae: 0.1055 - val_loss: 0.0202 - val_mae: 0.1053
Epoch 193/200

Epoch 00193: LearningRateScheduler setting learning rate to
2.8851804391913623e-06.
 - 0s - loss: 0.0201 - mae: 0.1066 - val_loss: 0.0202 - val_mae: 0.1053
Epoch 194/200

Epoch 00194: LearningRateScheduler setting learning rate to
2.7986250260156215e-06.
 - 0s - loss: 0.0200 - mae: 0.1062 - val_loss: 0.0202 - val_mae: 0.1053
Epoch 195/200

Epoch 00195: LearningRateScheduler setting learning rate to
2.7146662752351526e-06.
 - 0s - loss: 0.0200 - mae: 0.1064 - val_loss: 0.0202 - val_mae: 0.1052
Epoch 196/200

Epoch 00196: LearningRateScheduler setting learning rate to
```

```
2.633226286978098e-06.
 - 0s - loss: 0.0201 - mae: 0.1065 - val_loss: 0.0201 - val_mae: 0.1052
Epoch 197/200

Epoch 00197: LearningRateScheduler setting learning rate to
2.554229498368755e-06.
 - 0s - loss: 0.0202 - mae: 0.1071 - val_loss: 0.0201 - val_mae: 0.1052
Epoch 198/200

Epoch 00198: LearningRateScheduler setting learning rate to
2.477602613417692e-06.
 - 0s - loss: 0.0200 - mae: 0.1062 - val_loss: 0.0201 - val_mae: 0.1051
Epoch 199/200

Epoch 00199: LearningRateScheduler setting learning rate to
2.4032745350151615e-06.
 - 0s - loss: 0.0197 - mae: 0.1057 - val_loss: 0.0201 - val_mae: 0.1051
Epoch 200/200

Epoch 00200: LearningRateScheduler setting learning rate to
2.3311762989647066e-06.
 - 0s - loss: 0.0199 - mae: 0.1064 - val_loss: 0.0201 - val_mae: 0.1051
```

[37]:
```
predictions = model.predict(X_test)

print(X_test.shape)
print(predictions.shape)
```

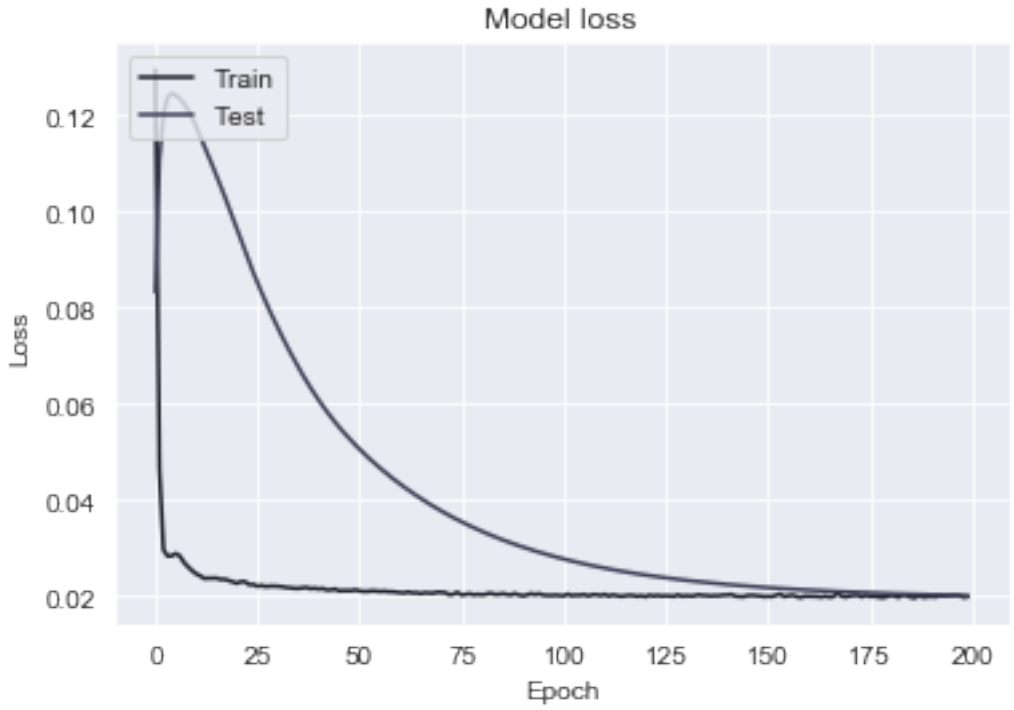
```
(7489, 146)
(7489, 1)
7489/7489 [==============================] - 0s 33us/step
0.0
0.0
```

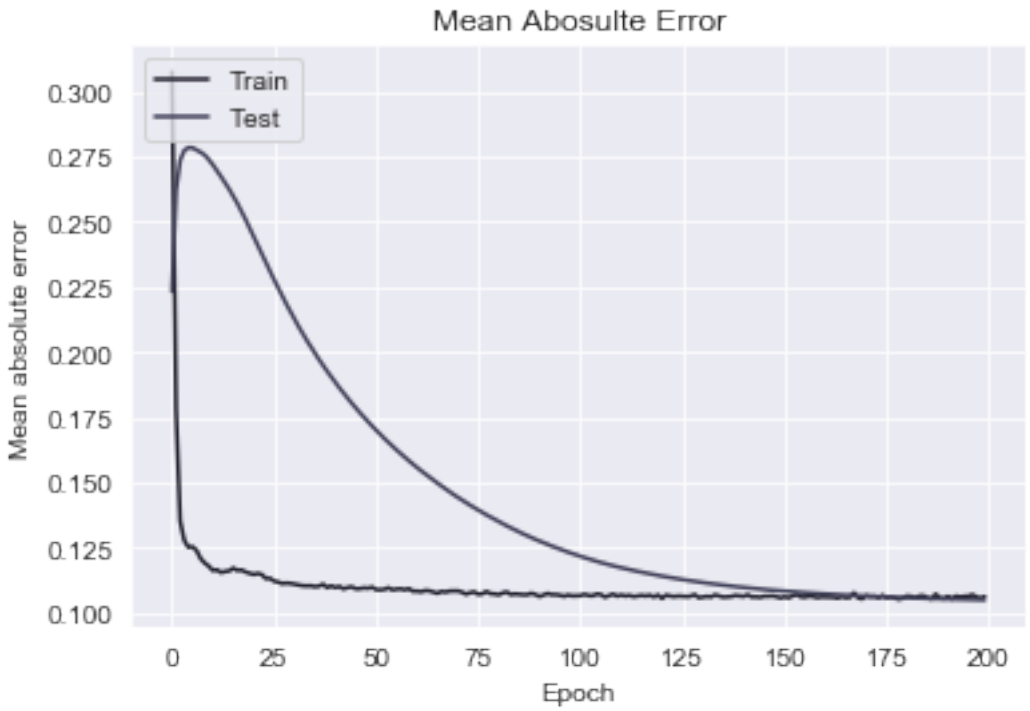
[42]:
```
type(history)
print(history.history.keys())
```

```
dict_keys(['val_loss', 'val_mae', 'loss', 'mae', 'lr'])
```

[43]:
```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

```
# Plot training & validation mae values
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('Mean Abosulte Error')
plt.ylabel('Mean absolute error')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```





5.1 alignment

[45]:
```
X_test = pd.read_csv('test_V2.csv')
X_test = X_test.groupby(['matchId','groupId','matchType']).first().reset_index()
X_test =␣
 ↪X_test[['matchId','groupId','matchType','numGroups','maxPlace','kills','killPlace']]

group = X_test_grp.groupby(['matchId'])
X_test_grp['winPlacePerc'] = predictions
X_test_grp['_rank.winPlacePerc'] = group['winPlacePerc'].rank(method='min')
X_test = pd.merge(X_test, X_test_grp)
```

[46]:
```
fullgroup = (X_test['numGroups'] == X_test['maxPlace'])

# full group (201366) --> calculate from rank
subset = X_test.loc[fullgroup]
X_test.loc[fullgroup, 'winPlacePerc'] = (subset['_rank.winPlacePerc'].values -␣
 ↪1) / (subset['maxPlace'].values - 1)

# not full group (684872) --> align with maxPlace
subset = X_test.loc[~fullgroup]
gap = 1.0 / (subset['maxPlace'].values - 1)
```

```
new_perc = np.around(subset['winPlacePerc'].values / gap) * gap  # half&up
X_test.loc[~fullgroup, 'winPlacePerc'] = new_perc

X_test['winPlacePerc'] = X_test['winPlacePerc'].clip(lower=0,upper=1)
```

[47]:
```
from tqdm import tqdm

# credit to https://www.kaggle.com/nagroda100/pubg-submission-postprocessor/code
print("Checking for anomalies in the winPlacePerc - players with same number of␣
 ↪kills should have scores in order of killPlace")

do_correct = True
iteration_number = 1

while do_correct & (iteration_number <= 1000):
    X_test.sort_values(ascending=False,␣
 ↪by=["matchId","kills","killPlace","winPlacePerc","groupId"], inplace=True)
    X_test["winPlacePerc_diff"] = X_test["winPlacePerc"].diff()
    X_test["kills_diff"] = X_test["kills"].diff()
    X_test["prev_matchId"] = X_test["matchId"].shift(1)
    X_test["prev_groupId"] = X_test["groupId"].shift(1)
    X_test["prev_winPlacePerc"] = X_test["winPlacePerc"].shift(1)

    df_sub2 = X_test[(X_test["winPlacePerc_diff"] < 0)
                     & (X_test["kills_diff"] == 0)
                     & (X_test["matchId"] == X_test["prev_matchId"])]
    anomalies_count = len(df_sub2)

    print("Iteration " + str(iteration_number) + " Anomalies count: " +␣
 ↪str(anomalies_count))

    changed_groups = list()

    if anomalies_count > 0:
        print()
        print("Looking for pairs to change...")

        df_sub2["new_winPlacePerc"] = df_sub2["winPlacePerc"]

        df_sub3 = pd.DataFrame()

        for i in tqdm(range(1, min(15001, max(anomalies_count, 2))),
                      desc="Identifying unique groups", mininterval=10):
            row = df_sub2.iloc[i - 1]
            id_prev = str(row["prev_matchId"]) + "!" + str(row["prev_groupId"])
            id_cur = str(row["matchId"]) + "!" + str(row["groupId"])
            if (not id_prev in changed_groups) & (not id_cur in changed_groups):
```

```
                changed_groups.append(id_prev)
                changed_groups.append(id_cur)
                df_sub3 = df_sub3.append({"matchId": row["matchId"], "groupId":␣
↪row["prev_groupId"],
                                          "new_winPlacePerc":␣
↪row["winPlacePerc"]},
                                         sort=False, ignore_index=True)
                df_sub3 = df_sub3.append({"matchId": row["matchId"], "groupId":␣
↪row["groupId"],
                                          "new_winPlacePerc":␣
↪row["prev_winPlacePerc"]},
                                         sort=False, ignore_index=True)

        df_sub3.drop_duplicates(inplace=True)
        X_test = X_test.merge(df_sub3, on=["matchId", "groupId"], how="left")
        notna = X_test["new_winPlacePerc"].notna()
        X_test.loc[notna, "winPlacePerc"] = X_test.
↪loc[notna]["new_winPlacePerc"]
        X_test.drop(labels="new_winPlacePerc", axis=1, inplace=True)
        del df_sub2
        del df_sub3
        df_sub2 = None
        df_sub3 = None
        gc.collect()
    else:
        do_correct = False

    iteration_number = iteration_number + 1

if do_correct:
    print("Limit of iterations reached...")

print("Finished correcting winPlacePerc")
```

```
/Users/vipulgaur/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:30: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
Identifying unique groups:   0%|          | 0/105 [00:00<?, ?it/s]

Checking for anomalies in the winPlacePerc - players with same number of kills
should have scores in order of killPlace
Iteration 1 Anomalies count: 106
```

```
Looking for pairs to change…

Identifying unique groups: 100%|      | 105/105 [00:00<00:00, 268.31it/s]
Identifying unique groups: 100%|      | 4/4 [00:00<00:00, 189.66it/s]
Identifying unique groups:   0%|          | 0/1 [00:00<?, ?it/s]

Iteration 2 Anomalies count: 5

Looking for pairs to change…
Iteration 3 Anomalies count: 1

Looking for pairs to change…

Identifying unique groups: 100%|      | 1/1 [00:00<00:00, 182.02it/s]

Iteration 4 Anomalies count: 0
Finished correcting winPlacePerc
```

[48]:
```
# edge cases
X_test.loc[X_test['maxPlace'] == 0, 'winPlacePerc'] = 0
X_test.loc[X_test['maxPlace'] == 1, 'winPlacePerc'] = 1  # nothing
X_test.loc[(X_test['maxPlace'] > 1) & (X_test['numGroups'] == 1),␣
 ↪'winPlacePerc'] = 0
X_test['winPlacePerc'].describe()
```

[48]:
```
count   7,489.000
mean        0.386
std         0.430
min         0.000
25%         0.000
50%         0.085
75%         0.897
max         1.000
Name: winPlacePerc, dtype: float64
```

[ ]:
```

```