第3回データ分析コンペティション ： PUBG Finish Placement Prediction

方針

1. 特徴量の作成
2. RFEによる特徴量削減
3. スタッキング

参考

https://www.kaggle.com/deffro/eda-is-fun (https://www.kaggle.com/deffro/eda-is-fun)
https://www.kaggle.com/kamalchhirang/5th-place-solution-0-0184-score
(https://www.kaggle.com/kamalchhirang/5th-place-solution-0-0184-score)
https://www.kaggle.com/ceshine/a-simple-post-processing-trick-lb-0237-0204
(https://www.kaggle.com/ceshine/a-simple-post-processing-trick-lb-0237-0204)
https://github.com/ghmagazine/kagglebook/blob/master/ch07/ch07-01-stacking.py
(https://github.com/ghmagazine/kagglebook/blob/master/ch07/ch07-01-stacking.py)

GCI2019-Winter Sekikawa318

In [1]:

```
%%html
<style>
   img {
      float:left;
      width: 1300px;
      height: 450px;
      border: 2px solid #000;
   }
   h1{color: #00008b; padding: 0.25em 0.5em; border-left: solid 5px #00008b;}
   h2{color: #2323b1; padding: 0.25em 0.5em; border-left: solid 5px #2323b1;}
   h3{color: #4e4edc; padding: 0.5em; border-bottom: solid 3px #4e4edc;}
   h4{color: #00008b; padding: 0.5em; border-left: solid 3px #00008b;}
</style>
```

扱うモジュールのインポート

In [1]:

```
import warnings
warnings.simplefilter('ignore')
import numpy as np
import pandas as pd

# lightgbmをインストールしておいてください
import lightgbm as lgb
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split, GridSearchCV, GroupKFold
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import RFE

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.svm import SVR, LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Ridge

# ! pip3 install eli5
from eli5.sklearn import PermutationImportance

%matplotlib inline
pd.set_option('display.max_columns', 1000)
```

```
Using TensorFlow backend.
```

データのインポート

このファイルを保存したディレクトリに"input"ディレクトリを作成し，その中に"train.csv"と"test.csv"のcsvファイルを保存しておいてください.

In [2]:

```
test_df = pd.read_csv("./input/test.csv")
train_df = pd.read_csv("./input/train.csv")
train_test_df = pd.concat([train_df, test_df])
```

1. 特徴量作成

matchTypeを数値に変更する

In [3]:

```
train_test_df['matchType'].value_counts()
```

Out[3]:

```
squad-fpp        3439
duo-fpp          2479
solo-fpp          966
squad             828
duo               657
solo              398
crashfpp           97
normal-squad-fpp    64
Name: matchType, dtype: int64
```

In [4]:

```
def standardize_matchType(df):
    df['team'] = [1 if i>50 else 2 if (i>25 & i<=50) else 4 for i in df['numGroups']]
    df['team'][df['matchType'] == 'solo'] = 1
    df['team'][df['matchType'] == 'solo-fpp'] = 1
    df['team'][df['matchType'] == 'duo'] = 2
    df['team'][df['matchType'] == 'duo-fpp'] = 2
    df['team'][df['matchType'] == 'squad'] = 4
    df['team'][df['matchType'] == 'squad-fpp'] = 4
    df['team'][df['matchType'] == 'normal-squad-fpp'] = 4
    return df
```

In [5]:

```
train_test_df = standardize_matchType(train_test_df)
train_test_df = train_test_df.drop(["matchType"], axis=1)
```

In [6]:

```
train_test_df["team"].value_counts()
```

Out[6]:

```
4    4331
2    3233
1    1364
Name: team, dtype: int64
```

基本的な特徴量を追加する

自チームの人数やマッチに参加した総人数など
チームメンバーが5人以上いる場合は,中央値で置き換え
それでも5以上なら4で置き換え

In [7]:

```
def basic_features(df):
    df['playersJoined'] = df.groupby('matchId')['matchId'].transform('count')
    df['teamMembers'] = df.groupby(['matchId', 'groupId'])['groupId'].transform('count')
    # チームメンバーが5人以上の人たちがいる -> 中央値で置き換え -> それでも5以上なら4で埋める
    temp = df.groupby("matchId")["teamMembers"].transform("median")
    df.loc[df["teamMembers"]>4, "teamMembers"] = temp[df["teamMembers"]>4]
    df.loc[df["teamMembers"]>4, "teamMembers"] = 4
    df['opponents'] = df['playersJoined'] - df['teamMembers']
    return df
```

In [8]:

```
train_test_df = basic_features(train_test_df)
```

In [9]:

```
# 確認用
# temp_columns = ["playersJoined", "teamMembers", "opponents"]
# train_test_df[temp_columns].head()
```

特徴量を追加する

合計移動距離やヘッドショットキルレートなど

In [10]:

```
def add_features(df):
    df['totalDistance'] = df['rideDistance'] + df['walkDistance'] + df['swimDistance']
    df['kills_assists'] = df['kills'] + df['assists']
    df['headshot_rate'] = df['headshotKills'] / (df['kills']+0.01)
    df['killStreaks_rate'] = df["killStreaks"] / (df["kills"]+0.01)
    df['pointsSum'] = df['killPoints'] + df['rankPoints'] + df['winPoints']
    df['heals_boosts'] = df['heals'] + df['boosts']
    df['kills_assists_per_heal_boost'] = df['kills_assists'] / (df['heals_boosts']+0.01)
    df['damageDealt_per_heal_boost'] = df['damageDealt'] / (df['heals_boosts'] + 1)
    df['kills_assists_per_heal_boost'] = df['kills_assists'] / (df['heals_boosts'] + 1)
    df['killPerc'] = df.groupby('matchId')['kills'].rank(pct=True).values
    df['killPlacePerc'] = df.groupby('matchId')['killPlace'].rank(pct=True).values
    df['heals_boostsPerc'] = df.groupby('matchId')['heals_boosts'].rank(pct=True).values
    return df
```

In [11]:

```
train_test_df = add_features(train_test_df)
```

In [12]:

```
# 確認用
# temp_columns = [
#     "totalDistance", "kills_assists", "headshot_rate", "killStreaks_rate", "pointsSum", "heals_boosts", "kills_assists_per_heal_boost",
#     "damageDealt_per_heal_boost", "kills_assists_per_heal_boost", "killPerc", "killPlacePerc", "heals_boostsPerc"
# ]
# train_test_df[temp_columns].head()
```

特徴量を統一化する

対戦相手が99人いる中の3人をkillした場合と，対戦相手が50人いる中の3人をkillした場合では重みが違うはず

In [13]:

```
def norm_features(df):
    df['teamKills'] = df['teamKills'] / df['teamMembers']
    df['killPlace'] = df['killPlace'] / (df['maxPlace'])
    df['roadKills'] = df['roadKills'] / (df['rideDistance'] + 0.01) / df["opponents"]
    df['maxPlace'] = train_test_df['maxPlace'] / train_test_df['numGroups']
    df['kills_Dis'] = df['kills'] / df["opponents"] /(df['totalDistance']+0.01)
    df['kills'] = df['kills'] / (df['opponents']) / df["matchDuration"]
    df["headshotKills"] = df["headshotKills"] / df["opponents"]
    df["killStreaks"] = df["killStreaks"]  / df["opponents"]

    df["boosts_Dis"] = df["boosts"] / (df["totalDistance"]+0.01)
    df['boosts'] = df['boosts'] / (df['matchDuration'])
    df['heals'] = df['heals'] / (df['matchDuration'])
    df["DBNOs_Dis"] = df["DBNOs"] / (df["totalDistance"]+0.01)
    df['DBNOs'] = df['DBNOs'] / (df['opponents']) / df["matchDuration"]
    df['damageDealt'] = df['damageDealt'] / df["opponents"] / df["matchDuration"]
    df['revives'] = df['revives'] / (df['numGroups']+0.01)
    df["weaponsAcquired_Dis"] = df["weaponsAcquired"] / (df["totalDistance"]+0.01)
    df["weaponsAcquired"] = df["weaponsAcquired"] / df["matchDuration"]
    df["heals_boosts_Dis"] = df["heals_boosts"] / (df["totalDistance"]+0.01)
    df["heals_boosts"] = df["heals_boosts"] / df["matchDuration"]
    df["kills_assists_Dis"] = df["kills_assists"] / df["opponents"] / (df["totalDistance"]+0.01)
    df["kills_assists"] = df["kills_assists"] / df["opponents"] / df["matchDuration"]

    df['rideDistance'] = df['rideDistance'] / (df['matchDuration'])
    df['swimDistance'] = df['swimDistance'] / (df['matchDuration'])
    df['walkDistance'] = df['walkDistance'] / (df['matchDuration'])
    df["totalDistance"] = df["totalDistance"] / (df['matchDuration'])
    return df
```

In [14]:

```
train_test_df = norm_features(train_test_df)
```

In [15]:

```
# 確認用
# temp_columns = [
#     "teamKills", "killPlace", "roadKills", "maxPlace", "kills_Dis", "kills", "headshotKills", "killStreaks", "boosts_Dis", "boosts", "heals", "DBNOs_Dis",
#     "DBNOs", "damageDealt", "revives", "weaponsAcquired_Dis", "weaponsAcquired", "heals_boosts_Dis", "heals_boosts", "kills_assists_Dis",
#     "kills_assists", "rideDistance", "swimDistance", "walkDistance", "totalDistance"
# ]
# train_test_df[temp_columns].head()
```

同じマッチの同じチームにおける特徴量作成

同じチームの合計キル数など

In [16]:

```
cols_to_drop = ["Id", "groupId", "matchId", "winPlacePerc", "matchType", "opponents", "playersJoined", "team", "missing_groups_percent", "maxPlace", "missingMembers","matchDuration", "numGroups", "teamMembers"]
features = [col for col in train_test_df.columns if col not in cols_to_drop]
def by_team(df):
    # std
    agg0 = df.groupby(['matchId','groupId'])[features].std()
    agg0 = agg0.replace([np.inf, np.NINF,np.nan], 0)
    df.merge(agg0, suffixes=['', '_std'], how='left', on=['matchId', 'groupId'])
    # std_rank
    agg0 = agg0.groupby(['matchId'])[features].rank(pct=True)
    df = df.merge(agg0, suffixes=['', '_std_rank'], how='left', on=['matchId', 'groupId'])
    # mean
    agg1 = df.groupby(['matchId', 'groupId'])[features].mean()
    df = df.merge(agg1, suffixes=['', '_mean'], how='left', on=['matchId', 'groupId'])
    # mean_rank
    agg1 = agg1.groupby('matchId')[features].rank(pct=True)
    df = df.merge(agg1, suffixes=['', '_mean_rank'], how='left', on=['matchId', 'groupId'])
    # max
    agg2 = df.groupby(['matchId', 'groupId'])[features].max()
    df = df.merge(agg2, suffixes=['', '_max'], how='left', on=['matchId', 'groupId'])
    # max_rank
    agg2 = agg2.groupby("matchId")[features].rank(pct=True)
    df = df.merge(agg2, suffixes = ["", "_max_rank"], how="left", on=["matchId", 'groupId'])
    # min
    agg3 = df.groupby(['matchId', 'groupId'])[features].min()
    df = df.merge(agg3, suffixes=['', '_min'], how='left', on=['matchId', 'groupId'])
    # min_rank
    agg3 = agg3.groupby("matchId")[features].rank(pct=True)
    df = df.merge(agg3, suffixes=['', '_min_rank'], how='left', on=['matchId', 'groupId'])
    # sum
    agg4 = df.groupby(['matchId', 'groupId'])[features].sum()
    df = df.merge(agg4, suffixes=['', '_sum'], how='left', on=['matchId', 'groupId'])
    return df
```

In [17]:

```
train_test_df = by_team(train_test_df)
```

同じマッチにおける特徴量作成

In [18]:

```
def by_match(df):
    # std
    agg0 = df.groupby('matchId')[features].std()
    agg0 = agg0.replace([np.inf, np.NINF,np.nan], 0)
    df.merge(agg0, suffixes=['', '_match_std'], how='left', on='matchId')
    # mean
    agg1 = df.groupby('matchId')[features].mean()
    df = df.merge(agg1, suffixes=['', '_match_mean'], how='left', on='matchId')
    # max
    agg2 = df.groupby('matchId')[features].max()
    df = df.merge(agg2, suffixes=['', '_match_max'], how='left', on='matchId')
    # min
    agg3 = df.groupby("matchId")[features].min()
    df = df.merge(agg3, suffixes=["", "_match_min"], how="left", on="matchId")
    return df
```

In [19]:

```
# コンペ中は下記のように集合を活用してリストの更新をしていたため,columnsの順番が変わってしまい,モデルのスコアにバラツキが出てしまっていました.
# features = list(set(features) - set(["killPerc", "killPlacePerc", "heals_boostsPerc"]))

features.remove("killPerc")
features.remove("killPlacePerc")
features.remove("heals_boostsPerc")
train_test_df = by_match(train_test_df)
```

必要のない特徴量の削除

In [20]:

```
train_test_df = train_test_df.drop(["Id", "groupId", "matchId"], axis=1)
```

出来上がったデータフレームの確認

In [21]:

```
print("columns: ", len(train_test_df.columns))
print("isna: ", 1794 - train_test_df.isna().sum().sum())
train_test_df.describe()
```

columns: 455
isna: 0

Out[21]:

	DBNOs	assists	boosts	damageDealt	headshotKills	
count	8928.000000	8928.000000	8928.000000	8928.000000	8928.000000	8928.0
mean	0.000005	0.236335	0.000703	0.000914	0.002343	0.0
std	0.000008	0.594336	0.001088	0.001211	0.006188	0.0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	0.000000	0.000000	0.000000	0.000565	0.000000	0.0
75%	0.000008	0.000000	0.001098	0.001288	0.000000	0.0
max	0.000091	6.000000	0.008000	0.024545	0.093750	0.0

2. データやモデルの準備

データの分割，標準化など

In [22]:

```
# データの分割
train_df = train_test_df[train_test_df["winPlacePerc"].notnull()]
test_df = train_test_df[train_test_df["winPlacePerc"].isnull()]

X_train = train_df.copy()
y_train = X_train["winPlacePerc"]
X_train = X_train.drop("winPlacePerc", axis=1)
X_test = test_df.copy()
X_test = X_test.drop("winPlacePerc", axis=1)
X_train1, X_valid, y_train1, y_valid = train_test_split(X_train, y_train, random_state=42)
```

In [23]:

```
# 何もしてない状態でどの程度のスコアになっているか確認
rfr = RandomForestRegressor(n_estimators=500, n_jobs=-1,random_state=42)
rfr.fit(X_train1, y_train1)
print("MAE: ", MAE(rfr.predict(X_valid), y_valid))
```

MAE: 0.02391583694869215

gbm, rfr, svr用のデータを用意する

In [24]:

```
X_train_gbm = X_train.copy()
X_test_gbm = X_test.copy()
X_train1_gbm = X_train1.copy()
X_valid_gbm = X_valid.copy()

X_train_rfr = X_train.copy()
X_test_rfr = X_test.copy()
X_train1_rfr = X_train1.copy()
X_valid_rfr = X_valid.copy()

# スケールが違いすぎると影響を受けてしまうので標準化
SS = StandardScaler().fit(pd.concat([X_train, X_test]))
X_train_svr = pd.DataFrame(SS.transform(X_train))
X_test_svr = pd.DataFrame(SS.transform(X_test))
X_train1_svr = pd.DataFrame(SS.transform(X_train1))
X_valid_svr = pd.DataFrame(SS.transform(X_valid))
```

特徴量の数をscikit-learnのRFEを使って減らす

特徴量の数を50個,20個,5個まで減らしたデータセットを作成する．(5時間以上かかります)
同じ流れを追いたい人以外は，一つ下のセルは飛ばす事を推奨します(実行してもこれ以降のセルで結果が上書きされます).

In [26]:

```
# 時間がかなりかかります.
# 流れを追いたい人のみ実行することを推奨します(実行してもこれ以降のセルで結果が上書きされます).
# gbm
perm_gbm = PermutationImportance(lgb.LGBMRegressor(n_estimators=80, random_state=42,n_jobs=-1),random_state=42,cv=None,scoring="neg_mean_absolute_error",n_iter=5)
select_gbm = RFE(perm_gbm, n_features_to_select=200,verbose=1,step=3).fit(X_train_gbm, y_train)
temp_gbm_200 = select_gbm.get_support()
X_train_gbm_200 = X_train_gbm.loc[:, temp_gbm_200]
X_test_gbm_200 = X_test_gbm.loc[:, temp_gbm_200]
X_train1_gbm_200 = X_train1_gbm.loc[:, temp_gbm_200]
X_valid_gbm_200 = X_valid_gbm.loc[:, temp_gbm_200]
select_gbm = RFE(perm_gbm, n_features_to_select=50,verbose=1,step=1).fit(X_train_gbm_200, y_train)
temp_gbm_50 = select_gbm.get_support()
X_train_gbm_50 = X_train_gbm_200.loc[:, temp_gbm_50]
X_test_gbm_50 = X_test_gbm_200.loc[:, temp_gbm_50]
X_train1_gbm_50 = X_train1_gbm_200.loc[:, temp_gbm_50]
X_valid_gbm_50 = X_valid_gbm_200.loc[:, temp_gbm_50]
select_gbm = RFE(perm_gbm, n_features_to_select=20,verbose=1,step=1).fit(X_train_gbm_50, y_train)
temp_gbm_20 = select_gbm.get_support()
X_train_gbm_20 = X_train_gbm_50.loc[:, temp_gbm_20]
X_test_gbm_20 = X_test_gbm_50.loc[:, temp_gbm_20]
X_train1_gbm_20 = X_train1_gbm_50.loc[:, temp_gbm_20]
X_valid_gbm_20 = X_valid_gbm_50.loc[:, temp_gbm_20]
select_gbm = RFE(perm_gbm, n_features_to_select=5,verbose=1,step=1).fit(X_train_gbm_20, y_train)
temp_gbm_5 = select_gbm.get_support()
X_train_gbm_5 = X_train_gbm_20.loc[:, temp_gbm_5]
X_test_gbm_5 = X_test_gbm_20.loc[:, temp_gbm_5]
X_train1_gbm_5 = X_train1_gbm_20.loc[:, temp_gbm_5]
X_valid_gbm_5 = X_valid_gbm_20.loc[:, temp_gbm_5]
# rfr
perm_rfr = PermutationImportance(RandomForestRegressor(n_estimators=50, random_state=42,n_jobs=-1),random_state=42,cv=None,scoring="neg_mean_absolute_error",n_iter=3)
select_rfr = RFE(perm_rfr, n_features_to_select=200, verbose=1,step=5).fit(X_train_rfr, y_train)
temp_rfr_200 = select_rfr.get_support()
X_train_rfr_200 = X_train_rfr.loc[:, temp_rfr_200]
X_test_rfr_200 = X_test_rfr.loc[:, temp_rfr_200]
X_train1_rfr_200 = X_train1_rfr.loc[:, temp_rfr_200]
X_valid_rfr_200 = X_valid_rfr.loc[:, temp_rfr_200]
select_rfr = RFE(perm_rfr, n_features_to_select=50, verbose=1,step=1).fit(X_train_rfr_200, y_train)
temp_rfr_50 = select_rfr.get_support()
X_train_rfr_50 = X_train_rfr_200.loc[:, temp_rfr_50]
X_test_rfr_50 = X_test_rfr_200.loc[:, temp_rfr_50]
X_train1_rfr_50 = X_train1_rfr_200.loc[:, temp_rfr_50]
X_valid_rfr_50 = X_valid_rfr_200.loc[:, temp_rfr_50]
select_rfr = RFE(perm_rfr, n_features_to_select=20, verbose=1,step=1).fit(X_train_rfr_50, y_train)
temp_rfr_20 = select_rfr.get_support()
X_train_rfr_20 = X_train_rfr_50.loc[:, temp_rfr_20]
```

```
X_test_rfr_20 = X_test_rfr_50.loc[:, temp_rfr_20]
X_train1_rfr_20 = X_train1_rfr_50.loc[:, temp_rfr_20]
X_valid_rfr_20 = X_valid_rfr_50.loc[:, temp_rfr_20]
select_rfr = RFE(perm_rfr, n_features_to_select=5, verbose=1,step=1).fit(X_train_rfr_2
0, y_train)
temp_rfr_5 = select_rfr.get_support()
X_train_rfr_5 = X_train_rfr_20.loc[:, temp_rfr_5]
X_test_rfr_5 = X_test_rfr_20.loc[:, temp_rfr_5]
X_train1_rfr_5 = X_train1_rfr_20.loc[:, temp_rfr_5]
X_valid_rfr_5 = X_valid_rfr_20.loc[:, temp_rfr_5]
# svm
perm_svr = PermutationImportance(LinearSVR(random_state=42),random_state=42,cv=
None,scoring="neg_mean_absolute_error",n_iter=5)
select_svr = RFE(perm_svr, n_features_to_select=200, verbose=1,step=3).fit(X_train_s
vr, y_train)
temp_svr_200 = select_svr.get_support()
X_train_svr_200 = pd.DataFrame(X_train_svr).loc[:, temp_svr_200]
X_test_svr_200 = pd.DataFrame(X_test_svr).loc[:, temp_svr_200]
X_train1_svr_200 = pd.DataFrame(X_train1_svr).loc[:, temp_svr_200]
X_valid_svr_200 = pd.DataFrame(X_valid_svr).loc[:, temp_svr_200]
select_svr = RFE(perm_svr, n_features_to_select=50, verbose=1,step=1).fit(X_train_sv
r_200, y_train)
temp_svr_50 = select_svr.get_support()
X_train_svr_50 = pd.DataFrame(X_train_svr_200).loc[:, temp_svr_50]
X_test_svr_50 = pd.DataFrame(X_test_svr_200).loc[:, temp_svr_50]
X_train1_svr_50 = pd.DataFrame(X_train1_svr_200).loc[:, temp_svr_50]
X_valid_svr_50 = pd.DataFrame(X_valid_svr_200).loc[:, temp_svr_50]
select_svr = RFE(perm_svr, n_features_to_select=20, verbose=1,step=1).fit(X_train_sv
r_50, y_train)
temp_svr_20 = select_svr.get_support()
X_train_svr_20 = pd.DataFrame(X_train_svr_50).loc[:, temp_svr_20]
X_test_svr_20 = pd.DataFrame(X_test_svr_50).loc[:, temp_svr_20]
X_train1_svr_20 = pd.DataFrame(X_train1_svr_50).loc[:, temp_svr_20]
X_valid_svr_20 = pd.DataFrame(X_valid_svr_50).loc[:, temp_svr_20]
select_svr = RFE(perm_svr, n_features_to_select=5, verbose=1,step=1).fit(X_train_svr
_20, y_train)
temp_svr_5 = select_svr.get_support()
X_train_svr_5 = pd.DataFrame(X_train_svr_20).loc[:, temp_svr_5]
X_test_svr_5 = pd.DataFrame(X_test_svr_20).loc[:, temp_svr_5]
X_train1_svr_5 = pd.DataFrame(X_train1_svr_20).loc[:, temp_svr_5]
X_valid_svr_5 = pd.DataFrame(X_valid_svr_20).loc[:, temp_svr_5]
```

Fitting estimator with 454 features.
Fitting estimator with 451 features.
Fitting estimator with 448 features.
Fitting estimator with 445 features.
Fitting estimator with 442 features.
Fitting estimator with 439 features.
Fitting estimator with 436 features.
Fitting estimator with 433 features.
Fitting estimator with 430 features.
Fitting estimator with 427 features.
Fitting estimator with 424 features.
Fitting estimator with 421 features.
Fitting estimator with 418 features.
Fitting estimator with 415 features.
Fitting estimator with 412 features.
Fitting estimator with 409 features.
Fitting estimator with 406 features.
Fitting estimator with 403 features.
Fitting estimator with 400 features.
Fitting estimator with 397 features.
Fitting estimator with 394 features.
Fitting estimator with 391 features.
Fitting estimator with 388 features.
Fitting estimator with 385 features.
Fitting estimator with 382 features.
Fitting estimator with 379 features.
Fitting estimator with 376 features.
Fitting estimator with 373 features.
Fitting estimator with 370 features.
Fitting estimator with 367 features.
Fitting estimator with 364 features.
Fitting estimator with 361 features.
Fitting estimator with 358 features.
Fitting estimator with 355 features.
Fitting estimator with 352 features.
Fitting estimator with 349 features.
Fitting estimator with 346 features.
Fitting estimator with 343 features.
Fitting estimator with 340 features.
Fitting estimator with 337 features.
Fitting estimator with 334 features.
Fitting estimator with 331 features.
Fitting estimator with 328 features.
Fitting estimator with 325 features.
Fitting estimator with 322 features.
Fitting estimator with 319 features.
Fitting estimator with 316 features.
Fitting estimator with 313 features.
Fitting estimator with 310 features.
Fitting estimator with 307 features.
Fitting estimator with 304 features.
Fitting estimator with 301 features.
Fitting estimator with 298 features.
Fitting estimator with 295 features.
Fitting estimator with 292 features.
Fitting estimator with 289 features.
Fitting estimator with 286 features.
Fitting estimator with 283 features.
Fitting estimator with 280 features.

Fitting estimator with 277 features.
Fitting estimator with 274 features.
Fitting estimator with 271 features.
Fitting estimator with 268 features.
Fitting estimator with 265 features.
Fitting estimator with 262 features.
Fitting estimator with 259 features.
Fitting estimator with 256 features.
Fitting estimator with 253 features.
Fitting estimator with 250 features.
Fitting estimator with 247 features.
Fitting estimator with 244 features.
Fitting estimator with 241 features.
Fitting estimator with 238 features.
Fitting estimator with 235 features.
Fitting estimator with 232 features.
Fitting estimator with 229 features.
Fitting estimator with 226 features.
Fitting estimator with 223 features.
Fitting estimator with 220 features.
Fitting estimator with 217 features.
Fitting estimator with 214 features.
Fitting estimator with 211 features.
Fitting estimator with 208 features.
Fitting estimator with 205 features.
Fitting estimator with 202 features.
Fitting estimator with 200 features.
Fitting estimator with 199 features.
Fitting estimator with 198 features.
Fitting estimator with 197 features.
Fitting estimator with 196 features.
Fitting estimator with 195 features.
Fitting estimator with 194 features.
Fitting estimator with 193 features.
Fitting estimator with 192 features.
Fitting estimator with 191 features.
Fitting estimator with 190 features.
Fitting estimator with 189 features.
Fitting estimator with 188 features.
Fitting estimator with 187 features.
Fitting estimator with 186 features.
Fitting estimator with 185 features.
Fitting estimator with 184 features.
Fitting estimator with 183 features.
Fitting estimator with 182 features.
Fitting estimator with 181 features.
Fitting estimator with 180 features.
Fitting estimator with 179 features.
Fitting estimator with 178 features.
Fitting estimator with 177 features.
Fitting estimator with 176 features.
Fitting estimator with 175 features.
Fitting estimator with 174 features.
Fitting estimator with 173 features.
Fitting estimator with 172 features.
Fitting estimator with 171 features.
Fitting estimator with 170 features.
Fitting estimator with 169 features.
Fitting estimator with 168 features.

Fitting estimator with 167 features.
Fitting estimator with 166 features.
Fitting estimator with 165 features.
Fitting estimator with 164 features.
Fitting estimator with 163 features.
Fitting estimator with 162 features.
Fitting estimator with 161 features.
Fitting estimator with 160 features.
Fitting estimator with 159 features.
Fitting estimator with 158 features.
Fitting estimator with 157 features.
Fitting estimator with 156 features.
Fitting estimator with 155 features.
Fitting estimator with 154 features.
Fitting estimator with 153 features.
Fitting estimator with 152 features.
Fitting estimator with 151 features.
Fitting estimator with 150 features.
Fitting estimator with 149 features.
Fitting estimator with 148 features.
Fitting estimator with 147 features.
Fitting estimator with 146 features.
Fitting estimator with 145 features.
Fitting estimator with 144 features.
Fitting estimator with 143 features.
Fitting estimator with 142 features.
Fitting estimator with 141 features.
Fitting estimator with 140 features.
Fitting estimator with 139 features.
Fitting estimator with 138 features.
Fitting estimator with 137 features.
Fitting estimator with 136 features.
Fitting estimator with 135 features.
Fitting estimator with 134 features.
Fitting estimator with 133 features.
Fitting estimator with 132 features.
Fitting estimator with 131 features.
Fitting estimator with 130 features.
Fitting estimator with 129 features.
Fitting estimator with 128 features.
Fitting estimator with 127 features.
Fitting estimator with 126 features.
Fitting estimator with 125 features.
Fitting estimator with 124 features.
Fitting estimator with 123 features.
Fitting estimator with 122 features.
Fitting estimator with 121 features.
Fitting estimator with 120 features.
Fitting estimator with 119 features.
Fitting estimator with 118 features.
Fitting estimator with 117 features.
Fitting estimator with 116 features.
Fitting estimator with 115 features.
Fitting estimator with 114 features.
Fitting estimator with 113 features.
Fitting estimator with 112 features.
Fitting estimator with 111 features.
Fitting estimator with 110 features.
Fitting estimator with 109 features.

Fitting estimator with 108 features.
Fitting estimator with 107 features.
Fitting estimator with 106 features.
Fitting estimator with 105 features.
Fitting estimator with 104 features.
Fitting estimator with 103 features.
Fitting estimator with 102 features.
Fitting estimator with 101 features.
Fitting estimator with 100 features.
Fitting estimator with 99 features.
Fitting estimator with 98 features.
Fitting estimator with 97 features.
Fitting estimator with 96 features.
Fitting estimator with 95 features.
Fitting estimator with 94 features.
Fitting estimator with 93 features.
Fitting estimator with 92 features.
Fitting estimator with 91 features.
Fitting estimator with 90 features.
Fitting estimator with 89 features.
Fitting estimator with 88 features.
Fitting estimator with 87 features.
Fitting estimator with 86 features.
Fitting estimator with 85 features.
Fitting estimator with 84 features.
Fitting estimator with 83 features.
Fitting estimator with 82 features.
Fitting estimator with 81 features.
Fitting estimator with 80 features.
Fitting estimator with 79 features.
Fitting estimator with 78 features.
Fitting estimator with 77 features.
Fitting estimator with 76 features.
Fitting estimator with 75 features.
Fitting estimator with 74 features.
Fitting estimator with 73 features.
Fitting estimator with 72 features.
Fitting estimator with 71 features.
Fitting estimator with 70 features.
Fitting estimator with 69 features.
Fitting estimator with 68 features.
Fitting estimator with 67 features.
Fitting estimator with 66 features.
Fitting estimator with 65 features.
Fitting estimator with 64 features.
Fitting estimator with 63 features.
Fitting estimator with 62 features.
Fitting estimator with 61 features.
Fitting estimator with 60 features.
Fitting estimator with 59 features.
Fitting estimator with 58 features.
Fitting estimator with 57 features.
Fitting estimator with 56 features.
Fitting estimator with 55 features.
Fitting estimator with 54 features.
Fitting estimator with 53 features.
Fitting estimator with 52 features.
Fitting estimator with 51 features.
Fitting estimator with 50 features.

Fitting estimator with 49 features.
Fitting estimator with 48 features.
Fitting estimator with 47 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 454 features.
Fitting estimator with 449 features.
Fitting estimator with 444 features.
Fitting estimator with 439 features.
Fitting estimator with 434 features.
Fitting estimator with 429 features.
Fitting estimator with 424 features.
Fitting estimator with 419 features.
Fitting estimator with 414 features.
Fitting estimator with 409 features.
Fitting estimator with 404 features.
Fitting estimator with 399 features.
Fitting estimator with 394 features.
Fitting estimator with 389 features.
Fitting estimator with 384 features.

Fitting estimator with 379 features.
Fitting estimator with 374 features.
Fitting estimator with 369 features.
Fitting estimator with 364 features.
Fitting estimator with 359 features.
Fitting estimator with 354 features.
Fitting estimator with 349 features.
Fitting estimator with 344 features.
Fitting estimator with 339 features.
Fitting estimator with 334 features.
Fitting estimator with 329 features.
Fitting estimator with 324 features.
Fitting estimator with 319 features.
Fitting estimator with 314 features.
Fitting estimator with 309 features.
Fitting estimator with 304 features.
Fitting estimator with 299 features.
Fitting estimator with 294 features.
Fitting estimator with 289 features.
Fitting estimator with 284 features.
Fitting estimator with 279 features.
Fitting estimator with 274 features.
Fitting estimator with 269 features.
Fitting estimator with 264 features.
Fitting estimator with 259 features.
Fitting estimator with 254 features.
Fitting estimator with 249 features.
Fitting estimator with 244 features.
Fitting estimator with 239 features.
Fitting estimator with 234 features.
Fitting estimator with 229 features.
Fitting estimator with 224 features.
Fitting estimator with 219 features.
Fitting estimator with 214 features.
Fitting estimator with 209 features.
Fitting estimator with 204 features.
Fitting estimator with 200 features.
Fitting estimator with 199 features.
Fitting estimator with 198 features.
Fitting estimator with 197 features.
Fitting estimator with 196 features.
Fitting estimator with 195 features.
Fitting estimator with 194 features.
Fitting estimator with 193 features.
Fitting estimator with 192 features.
Fitting estimator with 191 features.
Fitting estimator with 190 features.
Fitting estimator with 189 features.
Fitting estimator with 188 features.
Fitting estimator with 187 features.
Fitting estimator with 186 features.
Fitting estimator with 185 features.
Fitting estimator with 184 features.
Fitting estimator with 183 features.
Fitting estimator with 182 features.
Fitting estimator with 181 features.
Fitting estimator with 180 features.
Fitting estimator with 179 features.
Fitting estimator with 178 features.

Fitting estimator with 177 features.
Fitting estimator with 176 features.
Fitting estimator with 175 features.
Fitting estimator with 174 features.
Fitting estimator with 173 features.
Fitting estimator with 172 features.
Fitting estimator with 171 features.
Fitting estimator with 170 features.
Fitting estimator with 169 features.
Fitting estimator with 168 features.
Fitting estimator with 167 features.
Fitting estimator with 166 features.
Fitting estimator with 165 features.
Fitting estimator with 164 features.
Fitting estimator with 163 features.
Fitting estimator with 162 features.
Fitting estimator with 161 features.
Fitting estimator with 160 features.
Fitting estimator with 159 features.
Fitting estimator with 158 features.
Fitting estimator with 157 features.
Fitting estimator with 156 features.
Fitting estimator with 155 features.
Fitting estimator with 154 features.
Fitting estimator with 153 features.
Fitting estimator with 152 features.
Fitting estimator with 151 features.
Fitting estimator with 150 features.
Fitting estimator with 149 features.
Fitting estimator with 148 features.
Fitting estimator with 147 features.
Fitting estimator with 146 features.
Fitting estimator with 145 features.
Fitting estimator with 144 features.
Fitting estimator with 143 features.
Fitting estimator with 142 features.
Fitting estimator with 141 features.
Fitting estimator with 140 features.
Fitting estimator with 139 features.
Fitting estimator with 138 features.
Fitting estimator with 137 features.
Fitting estimator with 136 features.
Fitting estimator with 135 features.
Fitting estimator with 134 features.
Fitting estimator with 133 features.
Fitting estimator with 132 features.
Fitting estimator with 131 features.
Fitting estimator with 130 features.
Fitting estimator with 129 features.
Fitting estimator with 128 features.
Fitting estimator with 127 features.
Fitting estimator with 126 features.
Fitting estimator with 125 features.
Fitting estimator with 124 features.
Fitting estimator with 123 features.
Fitting estimator with 122 features.
Fitting estimator with 121 features.
Fitting estimator with 120 features.
Fitting estimator with 119 features.

Fitting estimator with 118 features.
Fitting estimator with 117 features.
Fitting estimator with 116 features.
Fitting estimator with 115 features.
Fitting estimator with 114 features.
Fitting estimator with 113 features.
Fitting estimator with 112 features.
Fitting estimator with 111 features.
Fitting estimator with 110 features.
Fitting estimator with 109 features.
Fitting estimator with 108 features.
Fitting estimator with 107 features.
Fitting estimator with 106 features.
Fitting estimator with 105 features.
Fitting estimator with 104 features.
Fitting estimator with 103 features.
Fitting estimator with 102 features.
Fitting estimator with 101 features.
Fitting estimator with 100 features.
Fitting estimator with 99 features.
Fitting estimator with 98 features.
Fitting estimator with 97 features.
Fitting estimator with 96 features.
Fitting estimator with 95 features.
Fitting estimator with 94 features.
Fitting estimator with 93 features.
Fitting estimator with 92 features.
Fitting estimator with 91 features.
Fitting estimator with 90 features.
Fitting estimator with 89 features.
Fitting estimator with 88 features.
Fitting estimator with 87 features.
Fitting estimator with 86 features.
Fitting estimator with 85 features.
Fitting estimator with 84 features.
Fitting estimator with 83 features.
Fitting estimator with 82 features.
Fitting estimator with 81 features.
Fitting estimator with 80 features.
Fitting estimator with 79 features.
Fitting estimator with 78 features.
Fitting estimator with 77 features.
Fitting estimator with 76 features.
Fitting estimator with 75 features.
Fitting estimator with 74 features.
Fitting estimator with 73 features.
Fitting estimator with 72 features.
Fitting estimator with 71 features.
Fitting estimator with 70 features.
Fitting estimator with 69 features.
Fitting estimator with 68 features.
Fitting estimator with 67 features.
Fitting estimator with 66 features.
Fitting estimator with 65 features.
Fitting estimator with 64 features.
Fitting estimator with 63 features.
Fitting estimator with 62 features.
Fitting estimator with 61 features.
Fitting estimator with 60 features.

Fitting estimator with 59 features.
Fitting estimator with 58 features.
Fitting estimator with 57 features.
Fitting estimator with 56 features.
Fitting estimator with 55 features.
Fitting estimator with 54 features.
Fitting estimator with 53 features.
Fitting estimator with 52 features.
Fitting estimator with 51 features.
Fitting estimator with 50 features.
Fitting estimator with 49 features.
Fitting estimator with 48 features.
Fitting estimator with 47 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.
Fitting estimator with 454 features.
Fitting estimator with 451 features.
Fitting estimator with 448 features.
Fitting estimator with 445 features.
Fitting estimator with 442 features.

Fitting estimator with 439 features.
Fitting estimator with 436 features.
Fitting estimator with 433 features.
Fitting estimator with 430 features.
Fitting estimator with 427 features.
Fitting estimator with 424 features.
Fitting estimator with 421 features.
Fitting estimator with 418 features.
Fitting estimator with 415 features.
Fitting estimator with 412 features.
Fitting estimator with 409 features.
Fitting estimator with 406 features.
Fitting estimator with 403 features.
Fitting estimator with 400 features.
Fitting estimator with 397 features.
Fitting estimator with 394 features.
Fitting estimator with 391 features.
Fitting estimator with 388 features.
Fitting estimator with 385 features.
Fitting estimator with 382 features.
Fitting estimator with 379 features.
Fitting estimator with 376 features.
Fitting estimator with 373 features.
Fitting estimator with 370 features.
Fitting estimator with 367 features.
Fitting estimator with 364 features.
Fitting estimator with 361 features.
Fitting estimator with 358 features.
Fitting estimator with 355 features.
Fitting estimator with 352 features.
Fitting estimator with 349 features.
Fitting estimator with 346 features.
Fitting estimator with 343 features.
Fitting estimator with 340 features.
Fitting estimator with 337 features.
Fitting estimator with 334 features.
Fitting estimator with 331 features.
Fitting estimator with 328 features.
Fitting estimator with 325 features.
Fitting estimator with 322 features.
Fitting estimator with 319 features.
Fitting estimator with 316 features.
Fitting estimator with 313 features.
Fitting estimator with 310 features.
Fitting estimator with 307 features.
Fitting estimator with 304 features.
Fitting estimator with 301 features.
Fitting estimator with 298 features.
Fitting estimator with 295 features.
Fitting estimator with 292 features.
Fitting estimator with 289 features.
Fitting estimator with 286 features.
Fitting estimator with 283 features.
Fitting estimator with 280 features.
Fitting estimator with 277 features.
Fitting estimator with 274 features.
Fitting estimator with 271 features.
Fitting estimator with 268 features.
Fitting estimator with 265 features.

Fitting estimator with 262 features.
Fitting estimator with 259 features.
Fitting estimator with 256 features.
Fitting estimator with 253 features.
Fitting estimator with 250 features.
Fitting estimator with 247 features.
Fitting estimator with 244 features.
Fitting estimator with 241 features.
Fitting estimator with 238 features.
Fitting estimator with 235 features.
Fitting estimator with 232 features.
Fitting estimator with 229 features.
Fitting estimator with 226 features.
Fitting estimator with 223 features.
Fitting estimator with 220 features.
Fitting estimator with 217 features.
Fitting estimator with 214 features.
Fitting estimator with 211 features.
Fitting estimator with 208 features.
Fitting estimator with 205 features.
Fitting estimator with 202 features.
Fitting estimator with 200 features.
Fitting estimator with 199 features.
Fitting estimator with 198 features.
Fitting estimator with 197 features.
Fitting estimator with 196 features.
Fitting estimator with 195 features.
Fitting estimator with 194 features.
Fitting estimator with 193 features.
Fitting estimator with 192 features.
Fitting estimator with 191 features.
Fitting estimator with 190 features.
Fitting estimator with 189 features.
Fitting estimator with 188 features.
Fitting estimator with 187 features.
Fitting estimator with 186 features.
Fitting estimator with 185 features.
Fitting estimator with 184 features.
Fitting estimator with 183 features.
Fitting estimator with 182 features.
Fitting estimator with 181 features.
Fitting estimator with 180 features.
Fitting estimator with 179 features.
Fitting estimator with 178 features.
Fitting estimator with 177 features.
Fitting estimator with 176 features.
Fitting estimator with 175 features.
Fitting estimator with 174 features.
Fitting estimator with 173 features.
Fitting estimator with 172 features.
Fitting estimator with 171 features.
Fitting estimator with 170 features.
Fitting estimator with 169 features.
Fitting estimator with 168 features.
Fitting estimator with 167 features.
Fitting estimator with 166 features.
Fitting estimator with 165 features.
Fitting estimator with 164 features.
Fitting estimator with 163 features.

Fitting estimator with 162 features.
Fitting estimator with 161 features.
Fitting estimator with 160 features.
Fitting estimator with 159 features.
Fitting estimator with 158 features.
Fitting estimator with 157 features.
Fitting estimator with 156 features.
Fitting estimator with 155 features.
Fitting estimator with 154 features.
Fitting estimator with 153 features.
Fitting estimator with 152 features.
Fitting estimator with 151 features.
Fitting estimator with 150 features.
Fitting estimator with 149 features.
Fitting estimator with 148 features.
Fitting estimator with 147 features.
Fitting estimator with 146 features.
Fitting estimator with 145 features.
Fitting estimator with 144 features.
Fitting estimator with 143 features.
Fitting estimator with 142 features.
Fitting estimator with 141 features.
Fitting estimator with 140 features.
Fitting estimator with 139 features.
Fitting estimator with 138 features.
Fitting estimator with 137 features.
Fitting estimator with 136 features.
Fitting estimator with 135 features.
Fitting estimator with 134 features.
Fitting estimator with 133 features.
Fitting estimator with 132 features.
Fitting estimator with 131 features.
Fitting estimator with 130 features.
Fitting estimator with 129 features.
Fitting estimator with 128 features.
Fitting estimator with 127 features.
Fitting estimator with 126 features.
Fitting estimator with 125 features.
Fitting estimator with 124 features.
Fitting estimator with 123 features.
Fitting estimator with 122 features.
Fitting estimator with 121 features.
Fitting estimator with 120 features.
Fitting estimator with 119 features.
Fitting estimator with 118 features.
Fitting estimator with 117 features.
Fitting estimator with 116 features.
Fitting estimator with 115 features.
Fitting estimator with 114 features.
Fitting estimator with 113 features.
Fitting estimator with 112 features.
Fitting estimator with 111 features.
Fitting estimator with 110 features.
Fitting estimator with 109 features.
Fitting estimator with 108 features.
Fitting estimator with 107 features.
Fitting estimator with 106 features.
Fitting estimator with 105 features.
Fitting estimator with 104 features.

Fitting estimator with 103 features.
Fitting estimator with 102 features.
Fitting estimator with 101 features.
Fitting estimator with 100 features.
Fitting estimator with 99 features.
Fitting estimator with 98 features.
Fitting estimator with 97 features.
Fitting estimator with 96 features.
Fitting estimator with 95 features.
Fitting estimator with 94 features.
Fitting estimator with 93 features.
Fitting estimator with 92 features.
Fitting estimator with 91 features.
Fitting estimator with 90 features.
Fitting estimator with 89 features.
Fitting estimator with 88 features.
Fitting estimator with 87 features.
Fitting estimator with 86 features.
Fitting estimator with 85 features.
Fitting estimator with 84 features.
Fitting estimator with 83 features.
Fitting estimator with 82 features.
Fitting estimator with 81 features.
Fitting estimator with 80 features.
Fitting estimator with 79 features.
Fitting estimator with 78 features.
Fitting estimator with 77 features.
Fitting estimator with 76 features.
Fitting estimator with 75 features.
Fitting estimator with 74 features.
Fitting estimator with 73 features.
Fitting estimator with 72 features.
Fitting estimator with 71 features.
Fitting estimator with 70 features.
Fitting estimator with 69 features.
Fitting estimator with 68 features.
Fitting estimator with 67 features.
Fitting estimator with 66 features.
Fitting estimator with 65 features.
Fitting estimator with 64 features.
Fitting estimator with 63 features.
Fitting estimator with 62 features.
Fitting estimator with 61 features.
Fitting estimator with 60 features.
Fitting estimator with 59 features.
Fitting estimator with 58 features.
Fitting estimator with 57 features.
Fitting estimator with 56 features.
Fitting estimator with 55 features.
Fitting estimator with 54 features.
Fitting estimator with 53 features.
Fitting estimator with 52 features.
Fitting estimator with 51 features.
Fitting estimator with 50 features.
Fitting estimator with 49 features.
Fitting estimator with 48 features.
Fitting estimator with 47 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.

Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
Fitting estimator with 15 features.
Fitting estimator with 14 features.
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
Fitting estimator with 6 features.

In [25]:

```
temp_gbm_50 = [
    'maxPlace', 'heals_boosts_std_rank', 'kills_mean', 'walkDistance_mean',
    'weaponsAcquired_mean', 'totalDistance_mean', 'killPerc_mean',
    'killPlacePerc_mean', 'kills_Dis_mean', 'boosts_mean_rank',
    'walkDistance_mean_rank', 'weaponsAcquired_mean_rank',
    'winPoints_mean_rank', 'totalDistance_mean_rank',
    'killStreaks_rate_mean_rank', 'heals_boosts_mean_rank',
    'weaponsAcquired_Dis_mean_rank', 'boosts_max', 'killPlace_max',
    'walkDistance_max', 'totalDistance_max', 'killPerc_max',
    'killPlacePerc_max', 'heals_boostsPerc_max', 'boosts_max_rank',
    'killPlace_max_rank', 'killPoints_max_rank', 'rideDistance_max_rank',
    'walkDistance_max_rank', 'totalDistance_max_rank', 'killPlace_min',
    'totalDistance_min', 'killPerc_min', 'killPlacePerc_min',
    'kills_Dis_min', 'headshotKills_min_rank', 'killPlace_min_rank',
    'killStreaks_min_rank', 'longestKill_min_rank', 'rankPoints_min_rank',
    'totalDistance_min_rank', 'kills_assists_min_rank',
    'killStreaks_rate_min_rank', 'killPerc_min_rank', 'killPlacePerc_sum',
    'heals_boosts_match_mean', 'damageDealt_match_mean',
    'longestKill_match_mean', 'boosts_match_mean', 'kills_Dis_match_max']
X_train_gbm_50 = X_train_gbm.loc[:, temp_gbm_50]
X_test_gbm_50 = X_test_gbm.loc[:, temp_gbm_50]
X_train1_gbm_50 = X_train1_gbm.loc[:, temp_gbm_50]
X_valid_gbm_50 = X_valid_gbm.loc[:, temp_gbm_50]
```

In [26]:

```
temp_gbm_20 = [
    'maxPlace', 'weaponsAcquired_mean', 'totalDistance_mean',
    'killPerc_mean', 'killPlacePerc_mean', 'totalDistance_mean_rank',
    'killPlace_max', 'walkDistance_max', 'killPerc_max',
    'killPlacePerc_max', 'boosts_max_rank', 'killPlace_max_rank',
    'walkDistance_max_rank', 'totalDistance_max_rank', 'killPerc_min',
    'killPlacePerc_min', 'kills_Dis_min', 'killStreaks_min_rank',
    'killPerc_min_rank', 'boosts_match_mean']
X_train_gbm_20 = X_train_gbm_50.loc[:, temp_gbm_20]
X_test_gbm_20 = X_test_gbm_50.loc[:, temp_gbm_20]
X_train1_gbm_20 = X_train1_gbm_50.loc[:, temp_gbm_20]
X_valid_gbm_20 = X_valid_gbm_50.loc[:, temp_gbm_20]
```

In [27]:

```
temp_gbm_5 = [
    'totalDistance_mean_rank', 'killPlacePerc_max', 'killPlace_max_rank',
    'walkDistance_max_rank', 'killPerc_min']
X_train_gbm_5 = X_train_gbm_20.loc[:, temp_gbm_5]
X_test_gbm_5 = X_test_gbm_20.loc[:, temp_gbm_5]
X_train1_gbm_5 = X_train1_gbm_20.loc[:, temp_gbm_5]
X_valid_gbm_5 = X_valid_gbm_20.loc[:, temp_gbm_5]
```

In [28]:

```
temp_rfr_50 = [
    'walkDistance', 'pointsSum', 'killPlacePerc', 'weaponsAcquired_Dis',
    'damageDealt_std_rank', 'heals_boosts_std_rank', 'damageDealt_mean',
    'killPlace_mean', 'walkDistance_mean', 'weaponsAcquired_mean',
    'totalDistance_mean', 'killStreaks_rate_mean', 'boosts_mean_rank',
    'swimDistance_mean_rank', 'walkDistance_mean_rank',
    'totalDistance_mean_rank', 'killStreaks_rate_mean_rank',
    'killPlacePerc_mean_rank', 'boosts_max', 'killPlace_max',
    'walkDistance_max', 'totalDistance_max', 'killPlacePerc_max',
    'heals_boostsPerc_max', 'boosts_max_rank', 'killPlace_max_rank',
    'walkDistance_max_rank', 'weaponsAcquired_max_rank',
    'totalDistance_max_rank', 'killPlacePerc_max_rank',
    'DBNOs_Dis_max_rank', 'weaponsAcquired_Dis_max_rank', 'killPlace_min',
    'weaponsAcquired_min', 'totalDistance_min', 'killPerc_min',
    'killPlacePerc_min', 'headshotKills_min_rank', 'heals_min_rank',
    'longestKill_min_rank', 'walkDistance_min_rank',
    'totalDistance_min_rank', 'killStreaks_rate_min_rank',
    'kills_assists_per_heal_boost_min_rank', 'killPlacePerc_min_rank',
    'kills_Dis_min_rank', 'boosts_sum', 'totalDistance_sum',
    'damageDealt_per_heal_boost_sum', 'kills_Dis_match_max']
X_train_rfr_50 = X_train_rfr.loc[:, temp_rfr_50]
X_test_rfr_50 = X_test_rfr.loc[:, temp_rfr_50]
X_train1_rfr_50 = X_train1_rfr.loc[:, temp_rfr_50]
X_valid_rfr_50 = X_valid_rfr.loc[:, temp_rfr_50]
```

In [29]:

```
temp_rfr_20 = [
    'weaponsAcquired_mean', 'totalDistance_mean', 'walkDistance_mean_rank',
    'totalDistance_mean_rank', 'killPlace_max', 'walkDistance_max',
    'killPlacePerc_max', 'boosts_max_rank', 'killPlace_max_rank',
    'walkDistance_max_rank', 'totalDistance_max_rank',
    'killPlacePerc_max_rank', 'killPlace_min', 'killPerc_min',
    'killPlacePerc_min', 'totalDistance_min_rank',
    'killStreaks_rate_min_rank', 'kills_Dis_min_rank', 'boosts_sum',
    'totalDistance_sum']
X_train_rfr_20 = X_train_rfr_50.loc[:, temp_rfr_20]
X_test_rfr_20 = X_test_rfr_50.loc[:, temp_rfr_20]
X_train1_rfr_20 = X_train1_rfr_50.loc[:, temp_rfr_20]
X_valid_rfr_20 = X_valid_rfr_50.loc[:, temp_rfr_20]
```

In [30]:

```
temp_rfr_5 = [
    'totalDistance_mean_rank', 'killPlacePerc_max', 'killPlace_max_rank',
    'walkDistance_max_rank', 'killPlacePerc_max_rank']
X_train_rfr_5 = X_train_rfr_20.loc[:, temp_rfr_5]
X_test_rfr_5 = X_test_rfr_20.loc[:, temp_rfr_5]
X_train1_rfr_5 = X_train1_rfr_20.loc[:, temp_rfr_5]
X_valid_rfr_5 = X_valid_rfr_20.loc[:, temp_rfr_5]
```

In [31]:

```
temp_svr_50 = np.array(
   [12,  16,  21,  28,  79,  84,  92, 101, 103, 105, 113, 127, 134,
    136, 151, 169, 189, 191, 193, 203, 210, 212, 215, 226, 227, 231,
    259, 264, 265, 277, 279, 282, 296, 297, 302, 303, 305, 322, 329,
    331, 333, 336, 343, 345, 346, 408, 444, 430, 426, 439])
X_train_svr_50 = pd.DataFrame(X_train_svr).loc[:, temp_svr_50]
X_test_svr_50 = pd.DataFrame(X_test_svr).loc[:, temp_svr_50]
X_train1_svr_50 = pd.DataFrame(X_train1_svr).loc[:, temp_svr_50]
X_valid_svr_50 = pd.DataFrame(X_valid_svr).loc[:, temp_svr_50]
```

In [32]:

```
temp_svr_20 = np.array(
   [16,  21,  28, 101, 127, 136, 151, 189, 203, 212, 227, 259, 282,
    297, 302, 305, 444, 430, 426, 439])
X_train_svr_20 = pd.DataFrame(X_train_svr_50).loc[:, temp_svr_20]
X_test_svr_20 = pd.DataFrame(X_test_svr_50).loc[:, temp_svr_20]
X_train1_svr_20 = pd.DataFrame(X_train1_svr_50).loc[:, temp_svr_20]
X_valid_svr_20 = pd.DataFrame(X_valid_svr_50).loc[:, temp_svr_20]
```

In [33]:

```
temp_svr_5 = np.array([203, 227, 282, 302, 305])
X_train_svr_5 = pd.DataFrame(X_train_svr_20).loc[:, temp_svr_5]
X_test_svr_5 = pd.DataFrame(X_test_svr_20).loc[:, temp_svr_5]
X_train1_svr_5 = pd.DataFrame(X_train1_svr_20).loc[:, temp_svr_5]
X_valid_svr_5 = pd.DataFrame(X_valid_svr_20).loc[:, temp_svr_5]
```

出来上がったデータフレームに対してスケール変換をかける

svrとMLPで使うときのためにスケール変換をしておく

In [34]:

```
# スケール変換
SS = StandardScaler().fit(pd.concat([X_train_gbm_50, X_test_gbm_50]))
X_train_gbm_50_scaled = pd.DataFrame(SS.transform(X_train_gbm_50))
X_test_gbm_50_scaled = pd.DataFrame(SS.transform(X_test_gbm_50))
SS = StandardScaler().fit(pd.concat([X_train_rfr_50, X_test_rfr_50]))
X_train_rfr_50_scaled = pd.DataFrame(SS.transform(X_train_rfr_50))
X_test_rfr_50_scaled = pd.DataFrame(SS.transform(X_test_rfr_50))

SS = StandardScaler().fit(pd.concat([X_train_gbm_20, X_test_gbm_20]))
X_train_gbm_20_scaled = pd.DataFrame(SS.transform(X_train_gbm_20))
X_test_gbm_20_scaled = pd.DataFrame(SS.transform(X_test_gbm_20))
SS = StandardScaler().fit(pd.concat([X_train_rfr_20, X_test_rfr_20]))
X_train_rfr_20_scaled = pd.DataFrame(SS.transform(X_train_rfr_20))
X_test_rfr_20_scaled = pd.DataFrame(SS.transform(X_test_rfr_20))
```

解析に使うモデルの準備

In [35]:

```
# 1層目
gbm_top50_1 = lgb.LGBMRegressor(n_jobs=-1, random_state=42, n_estimators=5000,
max_depth=10, num_leaves=30, learning_rate=0.005, min_data_in_leaf=30)
rfr_top50_1 = RandomForestRegressor(n_jobs=-1, random_state=42, max_depth=30, m
ax_features=25, min_samples_leaf=2, min_samples_split=2, n_estimators=10000)
gbm_top20_1 = lgb.LGBMRegressor(n_jobs=-1, random_state=42, n_estimators=5000,
max_depth=10, num_leaves=30, learning_rate=0.005, min_data_in_leaf=30)
rfr_top20_1 = RandomForestRegressor(n_jobs=-1, random_state=42, max_depth=30, m
ax_features=10, min_samples_leaf=2, min_samples_split=2, n_estimators=10000)
svr_1 = SVR(kernel="rbf", C=10, gamma=0.001)
MLP_svr_1 = MLPRegressor(random_state=42, alpha=0.1, hidden_layer_sizes=(50, 50, 5
0, 50, 50, 50, 50, 50,))
MLP_gbm_1 = MLPRegressor(random_state=42, alpha=0.05, hidden_layer_sizes=(20, 2
0, 20, 20, 20, 20, 20, 20,))
MLP_rfr_1 = MLPRegressor(random_state=42, alpha= 0.05, hidden_layer_sizes=(20, 20,
20, 20, 20, 20, 20, 20,))
ridge_1 = Ridge(alpha=10, random_state=42)
knn_svr_1 = KNeighborsRegressor(n_neighbors=30, n_jobs=-1, weights="distance")
knn_gbm_1 = KNeighborsRegressor(n_neighbors=30, n_jobs=-1, weights="distance")
knn_rfr_1 = KNeighborsRegressor(n_neighbors=30, n_jobs=-1, weights="distance")
# 2層目
gbm_2 = lgb.LGBMRegressor(n_jobs=-1, random_state=42, learning_rate=0.005, max_d
epth=2, n_estimators=5000,num_leaves=20, min_data_in_leaf=50)
rfr_2 = RandomForestRegressor(n_jobs=-1, random_state=42, max_depth=6, max_featu
res=8, min_samples_leaf=4, min_samples_split=2, n_estimators=10000)
# 3層目
gbm_3 = lgb.LGBMRegressor(n_jobs=-1, random_state=42, learning_rate=0.005, max_d
epth=2, n_estimators=5000, num_leaves=20, min_data_in_leaf=50)
```

3. モデル作成(スタッキング)

1時間くらいかかります

In [36]:

```
# 後で使うデータを事前に持っきておく
test_temp = pd.read_csv("./input/test.csv")
test_maxPlace = test_temp["maxPlace"]
test_matchId = test_temp["matchId"]
train_temp = pd.read_csv("./input/train.csv")
train_maxPlace = train_temp["maxPlace"]
train_matchId = train_temp["matchId"]
```

In [37]:

```
# 参考: kaggleで勝つデータ分析の技術
# https://github.com/ghmagazine/kagglebook/blob/master/ch07/ch07-01-stacking.py
def predict_cv(model, train_x, train_y, test_x):
    preds = []
    preds_test = []
    va_idxes = []
    kf = GroupKFold(n_splits=5)

    # クロスバリデーションで学習・予測を行い,予測値とインデックスを保存する
    for i, (tr_idx, va_idx) in enumerate(kf.split(train_x, groups=train_matchId)):
        tr_x, va_x = train_x.iloc[tr_idx], train_x.iloc[va_idx]
        tr_y, va_y = train_y.iloc[tr_idx], train_y.iloc[va_idx]
        model.fit(tr_x, tr_y)
        pred = model.predict(va_x)
        preds.append(pred)
        pred_test = model.predict(test_x)
        preds_test.append(pred_test)
        va_idxes.append(va_idx)

    # バリデーションデータに対する予測値を連結し,その後元の順序に並べ直す
    va_idxes = np.concatenate(va_idxes)
    preds = np.concatenate(preds, axis=0)
    order = np.argsort(va_idxes)
    pred_train = preds[order]

    # テストデータに対する予測値の平均をとる
    preds_test = np.mean(preds_test, axis=0)

    return pred_train, preds_test
```

In [38]:

```
# 1層目
print("-----Start-----\n")
pred1_train_a, pred1_test_a = predict_cv(gbm_top50_1, X_train_gbm_50, y_train, X_test_gbm_50)
pred1_train_b, pred1_test_b = predict_cv(rfr_top50_1, X_train_rfr_50, y_train, X_test_rfr_50)
pred1_train_c, pred1_test_c = predict_cv(gbm_top20_1, X_train_gbm_20, y_train, X_test_gbm_20)
pred1_train_d, pred1_test_d = predict_cv(rfr_top20_1, X_train_rfr_20, y_train, X_test_rfr_20)

pred1_train_e, pred1_test_e = predict_cv(svr_1, X_train_svr_50, y_train, X_test_svr_50)
pred1_train_f, pred1_test_f = predict_cv(ridge_1, X_train_svr_50, y_train, X_test_svr_50)

pred1_train_g, pred1_test_g = predict_cv(MLP_svr_1, X_train_svr_50, y_train, X_test_svr_50)
pred1_train_h, pred1_test_h = predict_cv(MLP_gbm_1, X_train_gbm_20_scaled, y_train, X_test_gbm_20_scaled)
pred1_train_i, pred1_test_i = predict_cv(MLP_rfr_1, X_train_rfr_20_scaled, y_train, X_test_rfr_20_scaled)

pred1_train_j, pred1_test_j = predict_cv(knn_svr_1, X_train_svr_5, y_train, X_test_svr_5)
pred1_train_k, pred1_test_k = predict_cv(knn_rfr_1, X_train_rfr_5, y_train, X_test_rfr_5)
pred1_train_l, pred1_test_l = predict_cv(knn_gbm_1, X_train_gbm_5, y_train, X_test_gbm_5)

# 2層目
# -----特徴量作成
train_mean_feature1 = (pred1_train_a+pred1_train_b+pred1_train_c+pred1_train_d+pred1_train_e+pred1_train_f+pred1_train_g+pred1_train_h+pred1_train_i+pred1_train_j+pred1_train_k+pred1_train_l)/12
test_mean_feature1 = (pred1_test_a+pred1_test_b+pred1_test_c+pred1_test_d+pred1_test_e+pred1_test_f+pred1_test_g+pred1_test_h+pred1_test_i+pred1_test_j+pred1_test_k+pred1_test_l)/12
train_mean_feature2 = (pred1_train_g+pred1_train_h+pred1_train_i+pred1_train_c+pred1_train_a)/5
test_mean_feature2 = (pred1_test_g+pred1_test_h+pred1_test_i+pred1_test_c+pred1_test_a)/5
train_dif_feature1 = (pred1_train_c+pred1_train_a)/2 - (pred1_train_j+pred1_train_k+pred1_train_l)/3
test_dif_feature1 = (pred1_test_c+pred1_test_a)/2 - (pred1_test_j+pred1_test_k+pred1_test_l)/3
train_dif_feature2 = (pred1_train_c+pred1_train_a)/2 - (pred1_train_g+pred1_train_h+pred1_train_i)/3
test_dif_feature2 = (pred1_test_c+pred1_test_a)/2 - (pred1_test_g+pred1_test_h+pred1_test_i)/3
train_dif_feature3 = (pred1_train_g+pred1_train_h+pred1_train_i)/3 - (pred1_train_j+pred1_train_k+pred1_train_l)/3
test_dif_feature3 = (pred1_test_g+pred1_test_h+pred1_test_i)/3 - (pred1_test_j+pred1_test_k+pred1_test_l)/3
# -----

X_train2 = pd.DataFrame({
```

```
    "1a": pred1_train_a,"1b": pred1_train_b,"1c": pred1_train_c,"1d": pred1_train_d,"1
e": pred1_train_e, "1f": pred1_train_f,"1g": pred1_train_g,"1h": pred1_train_h,"1i": pre
d1_train_i,"1j": pred1_train_j,"1k": pred1_train_k,"1l": pred1_train_l,
    "1mean_a": train_mean_feature1,"1mean_b": train_mean_feature2,"1dif_a": train_dif
_feature1,"1dif_b": train_dif_feature2,"1dif_c": train_dif_feature3,
    })
X_test2 = pd.DataFrame({
    "1a": pred1_test_a,"1b": pred1_test_b,"1c": pred1_test_c,"1d": pred1_test_d,"1e":
pred1_test_e,"1f": pred1_test_f,"1g": pred1_test_g,"1h": pred1_test_h,"1i": pred1_te
st_i,"1j": pred1_test_j,"1k": pred1_test_k,"1l": pred1_test_l,
    "1mean_a": test_mean_feature1, "1mean_b": test_mean_feature2,"1dif_a": test_dif
_feature1,"1dif_b": test_dif_feature2,"1dif_c": test_dif_feature3
    })
print("----- 1層目 -----")
print(" valid mean MAE: ",  MAE(train_mean_feature1, y_train))
print(" valid gbm MAE: ",  MAE((pred1_train_c+pred1_train_a)/2, y_train))
print(" valid rfr MAE: ",  MAE((pred1_train_b+pred1_train_d)/2, y_train))
print(" valid knn MAE: ",  MAE((pred1_train_j+pred1_train_k+pred1_train_l)/3, y_train))
print(" valid MLP MAE: ",  MAE((pred1_train_g+pred1_train_h+pred1_train_i)/3, y_train
))
print(" valid svr MAE: ",  MAE(pred1_train_e, y_train))
print(" valid ridge MAE: ",  MAE(pred1_train_f, y_train))
print(" valid (MLP+gbm) MAE: ",  MAE((pred1_train_g+pred1_train_h+pred1_train_i+pre
d1_train_c+pred1_train_a)/5, y_train))
display(X_train2.head(2))
display(X_test2.head(2))
X_train2_gbm = X_train_gbm_5.reset_index().join(X_train2)
X_test2_gbm = X_test_gbm_5.reset_index().join(X_test2)
X_train2_rfr = X_train_rfr_5.reset_index().join(X_train2)
X_test2_rfr = X_test_rfr_5.reset_index().join(X_test2)
X_train2_svr = X_train_svr_5.reset_index().join(X_train2)
X_test2_svr = X_test_svr_5.reset_index().join(X_test2)

pred2_train_a, pred2_test_a = predict_cv(gbm_2, X_train2_gbm, y_train, X_test2_gbm
)
pred2_train_b, pred2_test_b = predict_cv(rfr_2, X_train2_rfr, y_train, X_test2_rfr)

# 3層目
# -----特徴量作成
train_mean_feature2 = (pred2_train_a+pred2_train_b)/2
test_mean_feature2 = (pred2_test_a+pred2_test_b)/2
train_add_feature2 = pred2_train_a + pred2_train_b
test_add_feature2 = pred2_test_a + pred2_test_b
train_dif_feature2 = pred2_train_a - pred2_train_b
test_dif_feature2 = pred2_test_a - pred2_test_b
train_mul_feature2 = pred2_train_a * pred2_train_b
test_mul_feature2 = pred2_test_a * pred2_test_b
# -----

X_train3 = pd.DataFrame({
    "2a": pred2_train_a, "2b": pred2_train_b,
    "2mean": train_mean_feature2, "2add": train_add_feature2, "2dif": train_dif_feature2
, "2mul": train_mul_feature2
})
X_test3 = pd.DataFrame({
    "2a": pred2_test_a, "2b": pred2_test_b,
    "2mean": test_mean_feature2, "2add": test_add_feature2, "2dif": test_dif_feature2,
"2mul": test_mul_feature2
})
```

```
print("----- 2層目 -----")
print(" valid mean MAE: ",  MAE(train_mean_feature2, y_train))
print(" valid gbm MAE: ",  MAE(pred2_train_a, y_train))
print(" valid rfr MAE: ",  MAE(pred2_train_b, y_train))
display(X_train3.head(2))
display(X_test3.head(2))

pred3_train, pred3_test = predict_cv(gbm_3, X_train3, y_train, X_test3)
print("----- 3層目 -----")
print(" valid MAE: ", MAE(pred3_train, y_train))
print("\n-----Finish-----")
```

-----Start-----

----- 1層目 -----
valid mean MAE: 0.03326791574209489
valid gbm MAE: 0.03288062560860551
valid rfr MAE: 0.035324572542379404
valid knn MAE: 0.03780631129283767
valid MLP MAE: 0.03376210994443785
valid svr MAE: 0.0424968182985502
valid ridge MAE: 0.04125669022189193
valid (MLP+gbm) MAE: 0.03208460923920183

	1a	1b	1c	1d	1e	1f	1g	1h
0	0.180967	0.182083	0.171778	0.173768	0.156772	0.184706	0.162710	0.161109
1	0.613862	0.639644	0.568659	0.649119	0.680503	0.680148	0.648308	0.655884

	1a	1b	1c	1d	1e	1f	1g	1h
0	0.284520	0.291739	0.290020	0.296366	0.309242	0.314528	0.294155	0.298074
1	0.224696	0.224105	0.228659	0.219196	0.246234	0.228668	0.223144	0.210961

----- 2層目 -----
valid mean MAE: 0.03158739394099304
valid gbm MAE: 0.032495806199072254
valid rfr MAE: 0.031403379232057306

	2a	2b	2mean	2add	2dif	2mul
0	0.163170	0.171954	0.167562	0.335124	-0.008783	0.028058
1	0.600898	0.639799	0.620348	1.240697	-0.038901	0.384454

	2a	2b	2mean	2add	2dif	2mul
0	0.291024	0.292918	0.291971	0.583942	-0.001893	0.085246
1	0.218697	0.219747	0.219222	0.438443	-0.001050	0.048058

----- 3層目 -----
valid MAE: 0.03288962479481427

-----Finish-----

一番スコアの良い2層目のrfr (pred2_train_b)を採用

提出用のデータの後処理

目的変数は離散値なので,モデルの予測値が離散値に近い値を予測していたならばその離散値に合わせる

目的変数 = (maxPlace - 最終順位) / (maxPlace - 1)
幅 = 1/(maxPlace-1)

In [39]:

```
pred_train = pred2_train_b.copy()
cnt=0
for i, pre in enumerate(pred2_train_b):
    gap = 1.0 / (train_maxPlace[i] - 1)
    winPlacePerc = round(pre / gap) * gap
    if abs(winPlacePerc - pre) < (gap/8):
        cnt+=1
        pred_train[i] = winPlacePerc
print("valid MAE: ", MAE(pred_train, y_train))
print("cnt: ", cnt)
```

valid MAE: 0.03139289003410715
cnt: 2265

In [40]:

```
pred = pred2_test_b.copy()
cnt=0
for i, pre in enumerate(pred2_test_b):
    gap = 1.0 / (test_maxPlace[i] - 1)
    winPlacePerc = round(pre / gap) * gap
    if abs(winPlacePerc - pre) < (gap/8):
        cnt+=1
        pred[i] = winPlacePerc
print("cnt: ", cnt)
```

cnt: 658

提出用ファイルの作成

このファイルを保存しているディレクトリに， "submission.csv"を保存しておいてください

In [41]:

```
submission = pd.read_csv("./submission.csv")
submission["winPlacePerc"] = pred
submission.to_csv("submission.csv", index=False)
```

ご覧いただきありがとうございました

以上で提出ファイルの再現は終わりです.
下記の内容は雑記なので，提出ファイルに関係はありません.

再現性の確保について

random-stateを固定しているからといって必ずしも同じスコアになるわけではない.
columnsの順番が入れ替わっているだけでスコアが変動する場合がある(今回使ったモデルの中だとMLP).
気づけなくてかなり苦労しました.

In [42]:

```
# 行の並び順を変える
X_train_gbm_20_sorted = X_train_gbm_20.sort_index(axis=1)

display(X_train_gbm_20.head())
display(X_train_gbm_20_sorted.head())

grid_params = {"alpha": [0.05], "hidden_layer_sizes": [(20, 20, 20, 20, 20, 20, 20, 20
,)]}
MLP = MLPRegressor(random_state=42)

generator_train = GroupKFold(n_splits=5).split(X_train_rfr, y_train, train_matchId)
notSort_columns = GridSearchCV(MLP, grid_params, cv=generator_train, n_jobs=-1, verb
ose=1, scoring="neg_mean_absolute_error")
generator_train = GroupKFold(n_splits=5).split(X_train_rfr, y_train, train_matchId)
Sort_columns = GridSearchCV(MLP, grid_params, cv=generator_train, n_jobs=-1, verbose
=1, scoring="neg_mean_absolute_error")

notSort_columns.fit(X_train_gbm_20, y_train)
Sort_columns.fit(X_train_gbm_20_sorted, y_train)
print()
```

	maxPlace	weaponsAcquired_mean	totalDistance_mean	killPerc_mean	killPlacel
0	1.000000	0.002477	0.150743	0.291667	
1	1.021277	0.002886	0.943759	0.901042	
2	1.020833	0.002837	0.601348	0.876316	
3	1.107143	0.003132	1.721997	0.523990	
4	1.000000	0.001434	0.105125	0.482500	

	boosts_match_mean	boosts_max_rank	killPerc_max	killPerc_mean	killPerc_mir
0	0.000686	0.520833	0.291667	0.291667	0.291667
1	0.000639	0.712766	0.958333	0.901042	0.843750
2	0.000754	0.697917	0.926316	0.876316	0.826316
3	0.000774	0.589286	0.853535	0.523990	0.277778
4	0.000731	0.446429	0.680000	0.482500	0.285000

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent worke
rs.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    7.9s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent worke
rs.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    6.7s finished
```

In [43]:

```
print("notSort score: ", notSort_columns.best_score_)
print("Sort score: ", Sort_columns.best_score_)
```

notSort score: -0.04152552433666905
Sort score: -0.037874443922474396

KFold vs groupKFold (cvスコアとLeader Boardスコアの乖離)

KFoldで分けた場合とgroupKFoldで分けた場合のスコアの差を見てみる.
(単純な分割(KFold)と，同じmatchIDを持つデータがtrainとtestに混在しないようにした分割(groupKFold)の比較)

参考: https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html#sphx-glr-auto-examples-model-selection-plot-cv-indices-py (https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html#sphx-glr-auto-examples-model-selection-plot-cv-indices-py)

In [44]:

```
from sklearn.model_selection import KFold
generator_train = GroupKFold(n_splits=5).split(X_train_rfr, y_train, train_matchId)

grid_params = {"max_depth": [6,10,50], "max_features": [7,8,9,10,11], "min_samples_
split": [2,3,4], "min_samples_leaf": [2,3,4]}
rfr = RandomForestRegressor(n_estimators=100, random_state=42, n_jobs=-1)

# groupKFold
rfr_groupKFold = GridSearchCV(rfr, grid_params, cv=generator_train, n_jobs=-1, verbose
=1, scoring="neg_mean_absolute_error")
rfr_groupKFold.fit(X_train2_rfr, y_train)

# KFold
rfr_KFold = GridSearchCV(rfr, grid_params, cv=KFold(n_splits=5), n_jobs=-1, verbose=1,
scoring="neg_mean_absolute_error")
rfr_KFold.fit(X_train2_rfr, y_train)
print()
```

```
Fitting 5 folds for each of 135 candidates, totalling 675 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent worke
rs.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   17.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done 675 out of 675 | elapsed:  7.2min finished

Fitting 5 folds for each of 135 candidates, totalling 675 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent worke
rs.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   19.1s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:  4.3min
[Parallel(n_jobs=-1)]: Done 675 out of 675 | elapsed:  7.5min finished
```

In [45]:

```
print("groupKFold params: ", rfr_groupKFold.best_params_)
print("gropuKFold score: ", -1*rfr_groupKFold.best_score_)
print()
print("KFold params: ", rfr_KFold.best_params_)
print("KFold score: ", -1*rfr_KFold.best_score_)
```

```
groupKFold params:  {'max_depth': 6, 'max_features': 8, 'min_samples_lea
f': 4, 'min_samples_split': 2}
gropuKFold score:  0.0313624214572135

KFold params:  {'max_depth': 50, 'max_features': 10, 'min_samples_leaf':
2, 'min_samples_split': 2}
KFold score:  0.02074197664930879
```