MECHANISMS OF ACTION PREDICTION

1

PRIVATE SCORE :
総合順位：261位/4373チーム

ついに、
銅メダル獲得！！

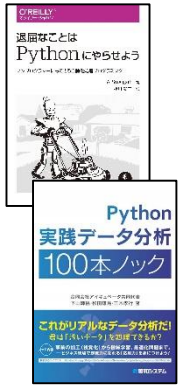
2020/12/03 中向

1. 背景

勉強してきたことを実践するために、Kaggleに挑戦してみました。
ついに銅メダルを取得することができました！

勉強内容

2019/10 業務効率化のためPython勉強開始



2020/2 機械学習の勉強開始

2020/8 Deep Learningの勉強開始

2020/9 実践のため、Kaggle 画像コンペに挑戦

2020/10 実践のため、Kaggle 言語コンペに挑戦

2020/12 実践のため、Kaggle 数値データコンペに挑戦

kaggle 統計家、データ分析家が分析手法を競い合う場所

1. テーマ

今回は206クラスの多ラベル分類問題への挑戦です。

多ラベル分類問題（206クラス）
遺伝子発現データ、細胞生存能力データ等から、
206種類のMoA(※)の内、どれに該当するかを予測する

※ MoA(Mechanisms of Action)：作用機序
薬が治療効果を及ぼす仕組みのこと。⇒新薬の開発にも使われる

学習データ
23,814個のサンプル×875個の特徴量

学習ラベル
23814個のサンプル×206ラベル

非学習ラベル
23814個のサンプル×402ラベル

Test Dataを予測した時のLoglossで競う

テストデータ
3,982個のサンプル×875個の特徴量

2. データの確認

特徴量同士の相関が高いところと、不均衡なデータとなっているのが今回の難しいポイント。

学習データ

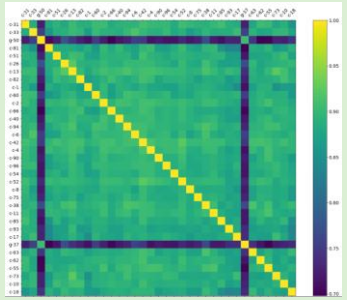
23,814個のサンプル×875個の特徴量

学習ラベル

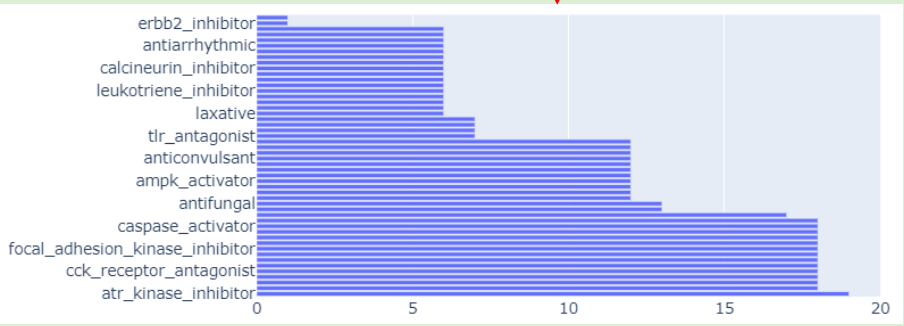
23,814個のサンプル×206ラベル

相関の高い特徴量が多数存在する

相関係数0.9以上を有する特徴量は35個ある



かなり不均衡なデータ

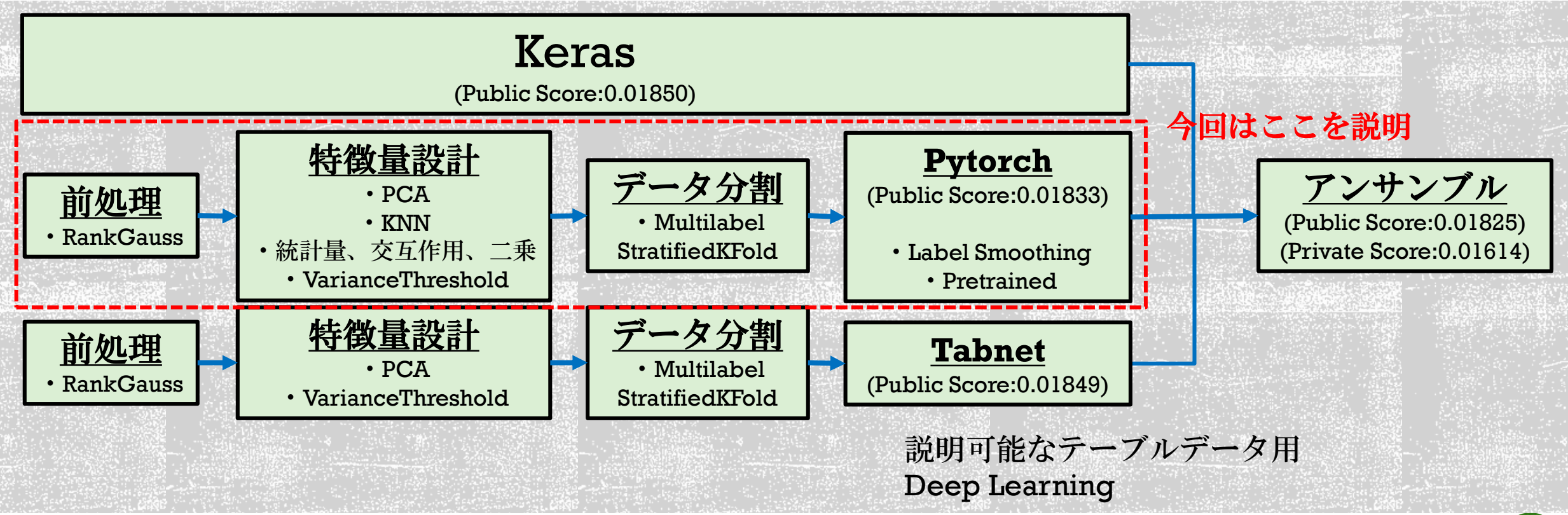


陽性サンプル数

23,814サンプル中、
陽性サンプルが20個以下の目的変数もある

3.全体像

まず、今回の取り組みの全体像を示します。時間の都合上
今回は Pytorchの部分に絞って、内容共有します。



https://www.kaggle.com/vikazrajpurohit/3-model-training-and-inference

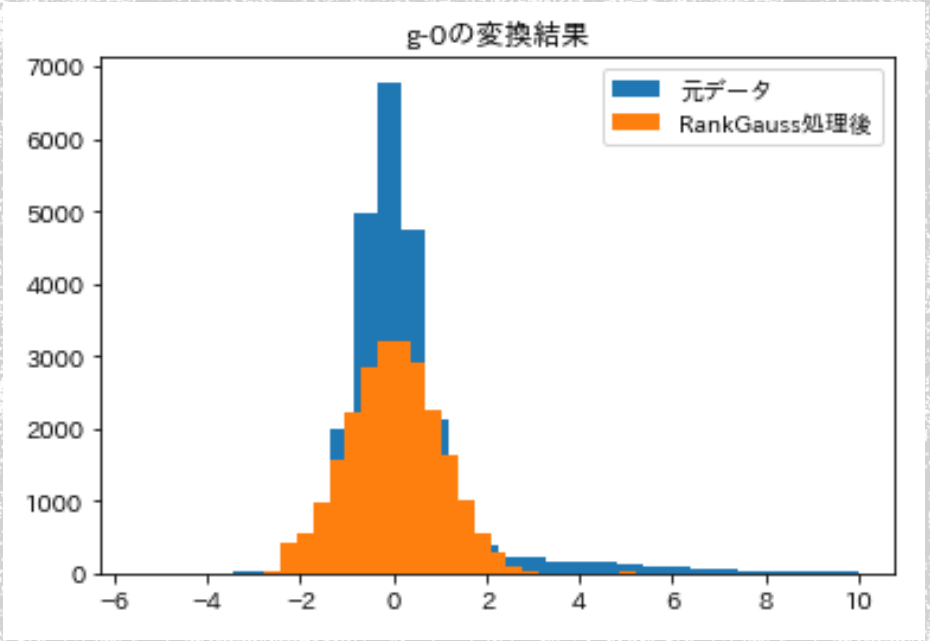
4. PYTORCH 前処理(RANK GAUSS)

前処理にRang Gaussという処理を行いました。

RankGauss (QuantileTransformer)

順位を維持したまま、正規分布に変換する手法

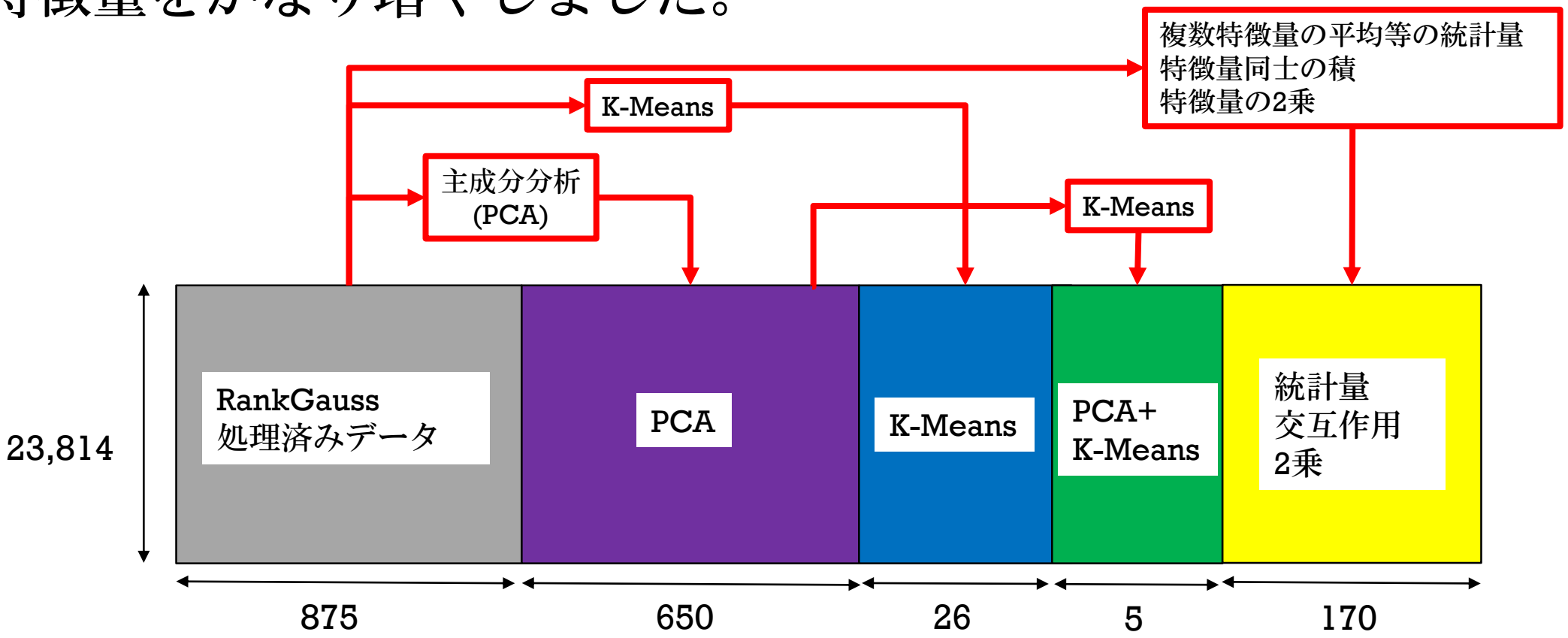
ニューラルネットワークを使用する際には、
Min-Max ScalerやStandardScalerよりも
優れた性能を発揮する




```
# RankGauss - transform to Gauss
qt = QuantileTransformer(n_quantiles=100,random_state=42,output_distribution='normal')
data = pd.concat([pd.DataFrame(train_features[GENES+CELLS]), pd.DataFrame(test_features[GENES+CELLS])])
data2 = qt.fit_transform(data[GENES+CELLS])
train_features[GENES+CELLS] = pd.DataFrame(data2[:train_features.shape[0]])
test_features[GENES+CELLS] = pd.DataFrame(data2[-test_features.shape[0]:])
```

4. PYTORCH 特徴量作成

PCAとK-Means、また統計量を使うことで、もともと875個だった特徴量をかなり増やしました。

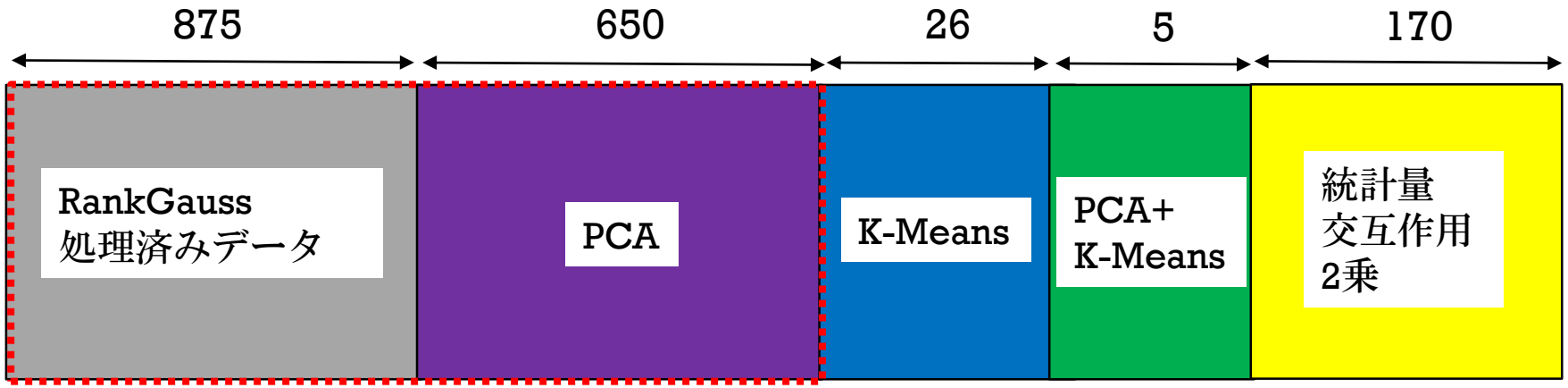


一般的に、PCAの結果を既存の特徴量に追加することは望ましくない
今回はスコアが向上することが観測されたため、大多数が実施していた

https://www.kaggle.com/c/lish-moa/discussion/183166

4. PYTORCH 特徴量選択

分散を使うことで、1239個まで特徴量を絞りました。



分散が0.85以下の特徴量を削除
(487特徴量削除)

```
#分散が0.85以下の変数を削除する
var_thresh = VarianceThreshold(0.85)
data = train_features.append(test_features)
data_transformed = var_thresh.fit_transform(data.iloc[:, 4:])
```

学習データ

23,814個のサンプル×1,239個の特徴量

RankGauss処理しているので、大多数の特徴量は分散1になるはずなので、あまり特徴量は減っていません。

https://www.kaggle.com/c/lish-moa/discussion/194973

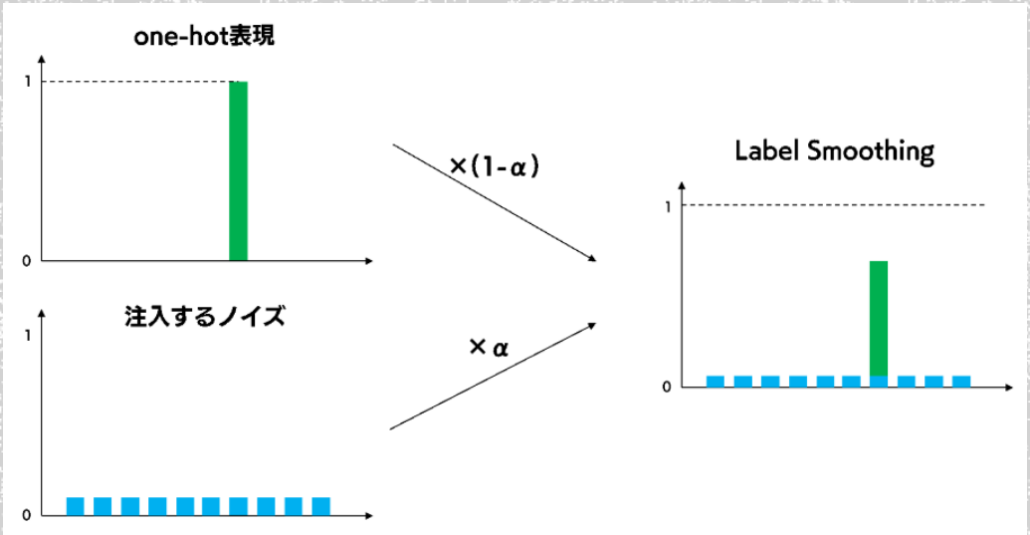
4. PYTORCH LABEL SMOOTHING

過学習対策に、Label Somoothingを行いました。

Label Smoothing

過学習対策として、ノイズを入れて学習させる

One Hot表現	0	1	0	0	0
Label Smoothing	0.0005	0.9995	0.0005	0.0005	0.0005




```
@staticmethod
def _smooth(targets:torch.Tensor, n_labels:int, smoothing=0.0):
    assert 0 <= smoothing < 1
    with torch.no_grad():
        targets = targets * (1.0 - smoothing) + 0.5 * smoothing #正解ラベルの他にノイズとして0.5の一様分布を注入
    return targets
```


https://www.softbanktech.co.jp/special/blog/cloud_blog/2019/0073/

4. PYTORCH CROSS VALIDATION

データが不均衡のため、StratifiedKFoldを使って、CrossValidationをしました。

StratifiedKFold

各陽性サンプルが同じ数ずつ分割できるように、TrainとValidationデータに分割する
多ラベルの場合はMultilabelStratifiedKFoldを使用する。

```
skf = MultilabelStratifiedKFold(n_splits=NFOLDS, shuffle=True, random_state=seed_id)
```

今回はサンプルではなくて、Drug_idをもとにKfoldに分割しているので、
同じDrug_idが分割されないようになっている。

kfold	protein_tyrosine_kinase_inhibitor	sig_id
0	0	3134
0	1	2
1	0	3132
1	1	3
2	0	3132
2	1	3
3	0	3133
3	1	3
4	0	3132
4	1	3
5	0	3132
5	1	3
6	0	3134
6	1	2

Kfold=0-6の7グループに分割

Protein_tyrosine_kinase_inhibiterの例

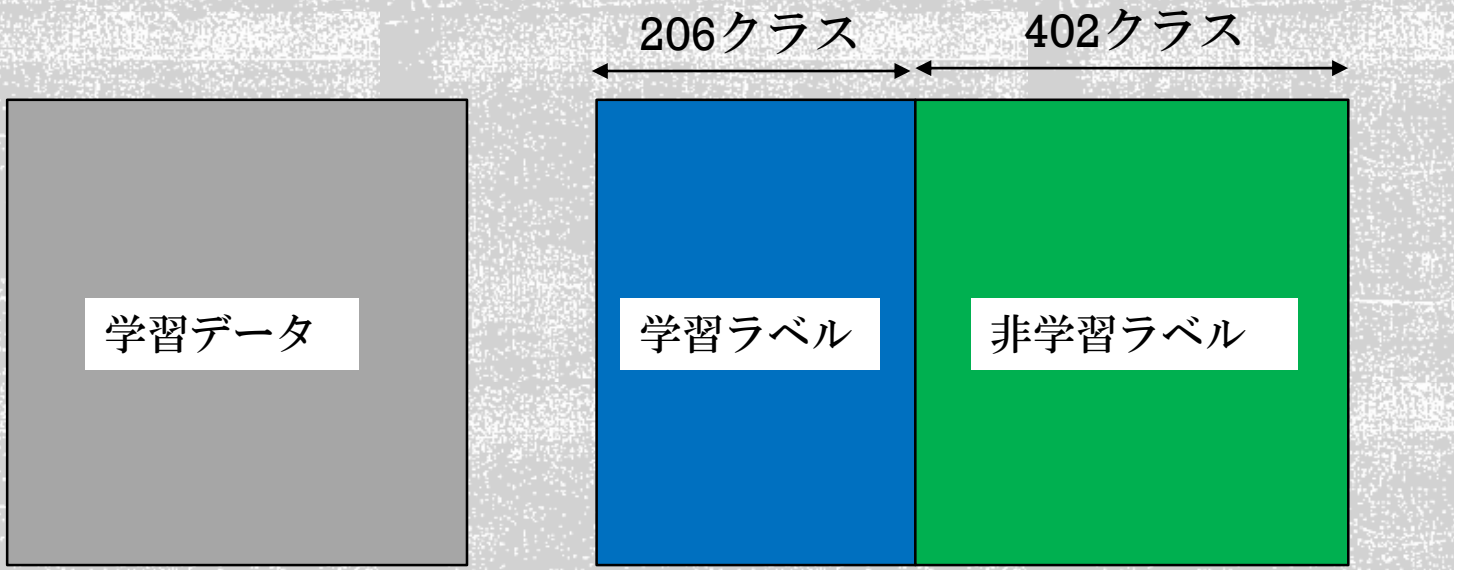
グループ3 陰性サンプル 3133個、陽性サンプル3個

グループ5 陰性サンプル 3132個、陽性サンプル3個

陽性サンプルの数が近い数になるようにグループ分けされる

4. PYTORCH 学習 STEP-1

学習は2STEPに分かれています。



学習

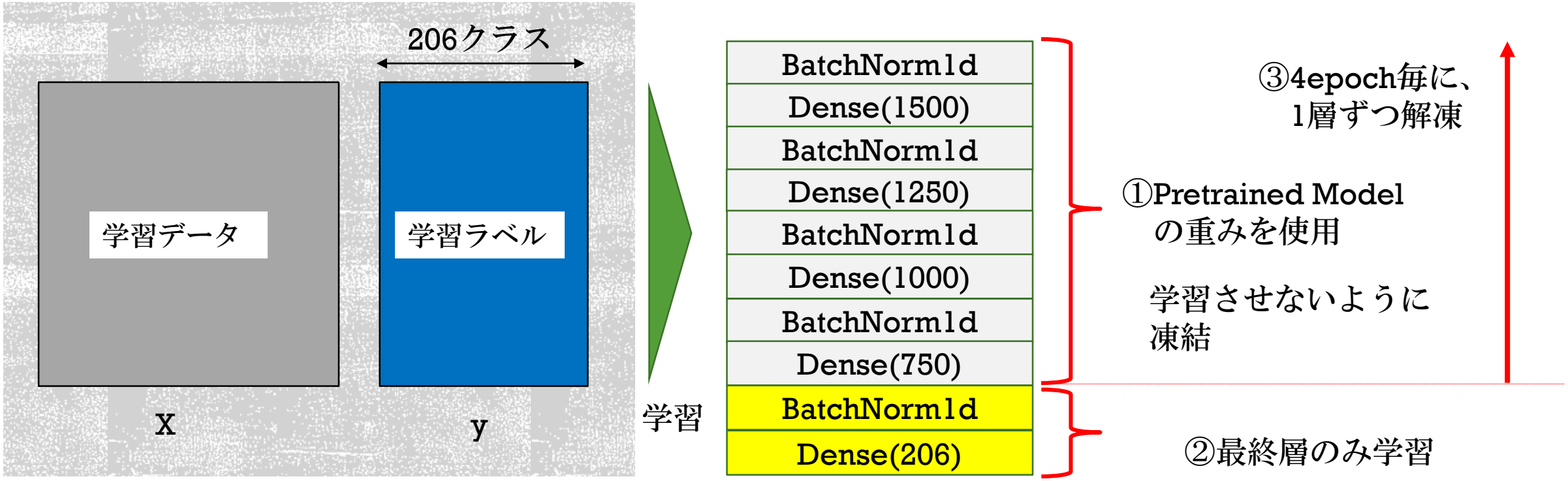
Model

BatchNorm1d
Dense(1500)
BatchNorm1d
Dense(1250)
BatchNorm1d
Dense(1000)
BatchNorm1d
Dense(750)
BatchNorm1d
Dense(608)

23Epoch回して、一番Validationが良かったものを
Pretrained Modelとして保存

4. PYTORCH 学習 STEP-2

2STEP目の学習は以下のようになります。



23Epoch回して、徐々に後ろから解凍しながら学習させていく
以上のステップを乱数シードを変えながら複数回計算し、平均値を出力

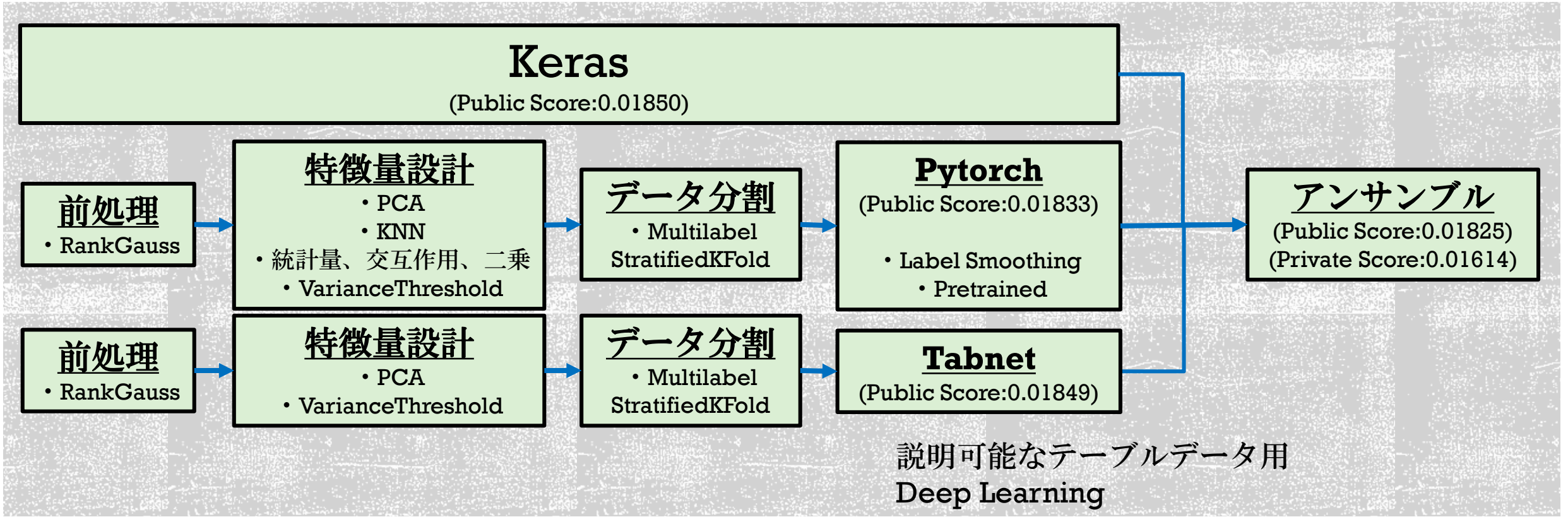
5. 他に試したこと

他にもいろいろ試してはみたが、効果はありませんでした

- 不均衡データだったので、SMOTEによるオーバーサンプリングを適用
- モデルの構造、Seedの変更
- 特徴量選択、PCA、K-Meansの閾値変更
- 活性化関数を Mishに変えてみる

6.纏め

ここまでの内容 と Keras＋Tabnetによる判定結果をアンサンブルすることで精度が向上しました。



https://www.kaggle.com/vikazrajpurohit/3-model-training-and-inference