機械学習エンジニアコース
Sprint

－ 機械学習フロー －

DIVE INTO CODE

今回のモチベーション

1. クロスバリデーション
   信頼性の高いモデルを手に入れたい!

2. グリッドサーチ
   ハイパーパラメータを探しに行こう。

3. 高い汎化性能のモデル
   観測されたデータに適合するのではなく、未知のデータをうまく予測したい

クロスバリデーション

Kaggle iterative manners

アドバイザーYifan Xieの反復的手法 https://www.kaggle.com/yifanxie

クロスバリデーション(cross-validation)への言及が多い↓

- 最初にクロスバリデーションの型を作成し、クロスバリデーションとリーダーボードスコアの関係をみる

- 厳密なクロスバリデーションを作るために実験を開始する。これには、クロスバリデーションの方法論、実装、データ取得メカニズムが含まれる。- 特徴量の効果とモデルのアーキテクチャを決定するためにクロスバリデーション手法を用いる

– Understand problem definition
– Understand the evaluation metric in the context of the problem.
– Understand how public and private leaderboard are split
– Initial Data exploration, establish initial understand the nature of dataset
– Build first batch of models, create first batch of submissions
– Create first cross-validation scheme, and start evaluating relationship between cross-validation and leaderboard score
– Start designing experiment for rigorous cross-validation, this shall include both the methodology of cross-validation, the actual implementation, a data capturing mechanism for experiment data – In-depth data manipulation for feature engineering,
– For deep learning dominated approaches, design and evaluate model architecture

– use established cross-validation method to decide the effectiveness of features and model architecture
– Apply ensemble methods – this can range from simple weigh averaging to stacking
– post-process of prediction from machine learning model. this is optional depends on specific problem and evaluation metric.
The following shall be strongly encouraged throughout the competition:
– Pay close attention to what others are sharing on forum and competition specific kernels
– Review and check past competition top solutions – especially the ones with similar problems and similar evaluation metrics

クロスバリデーション

Why Cross validation ?
なぜクロスバリデーション(交差検証)をするの?

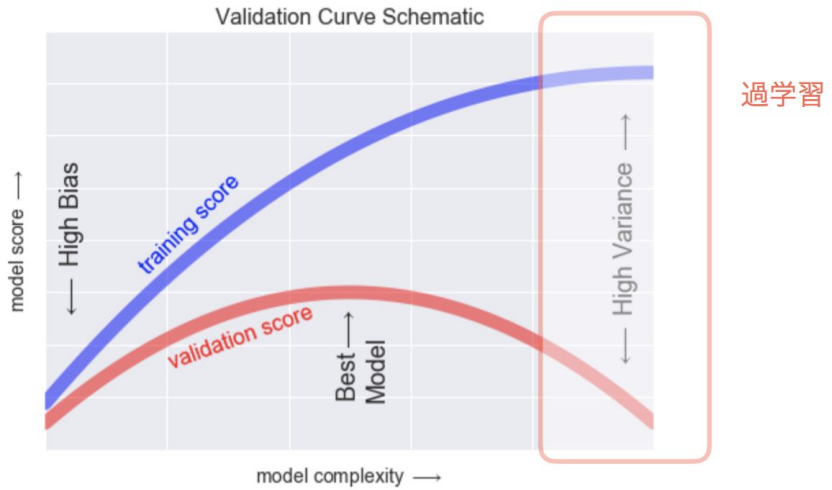
新しい特徴量を手に入れた!
↓
Fit して Pred しよ!
↓
Submission した!
↓
PublicLBのスコア上がった
↓
ヤッター!!
↓
ちょっと待って!
↓
一部のデータに過学習した結果かもよ?

クロスバリデーション

Why Cross validation ?
なぜクロスバリデーション(交差検証)をするの?

機械学習としての狙いは、モデルを観測された手元のデータに適合させるのではなく、未知のデータに対し予測性能をあげるため。
https://jakevdp.github.io/PythonDataScienceHandbook/05.03-hyperparameters-and-model-validation.html

Kaggleにおいては、Public LBのスコアが信頼できるスコアかどうか見積もるために、クロスバリデーションを利用する。
※Public LBの順位は、基本的にテストデータの一部で評価した結果なので、残りのデータで評価されたとき(Private LB)、順位が下がってしまう恐れがある。どんなデータであれ安定した評価結果が得られるようにクロスバリデーションを利用して汎化性能を検証する。



クロスバリデーション

What does cross validation do?

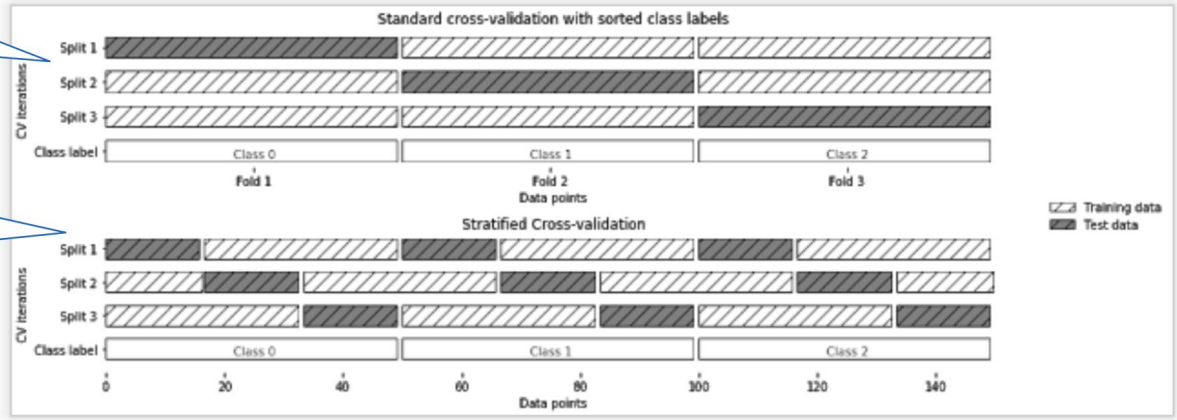
クロスバリデーションは何をしているの?

https://machinelearningmastery.com/k-fold-cross-validation/

1. データセットをランダムにシャッフルします。
2. データセットをk個のグループに分割する
3. 分割する度に以下を実行(k回):
4. 一つのグループをホールドアウトまたはデータセットとする
5. 残りのグループをトレーニングデータセットとする
6. モデルをトレーニングセットにフィットさせ、テストセットで評価する
7. 評価スコアを保持してモデルを破棄する
8. モデル評価スコアを平均してモデルの性能を要約する

通常のKFold

Stratified
(層化) KFold



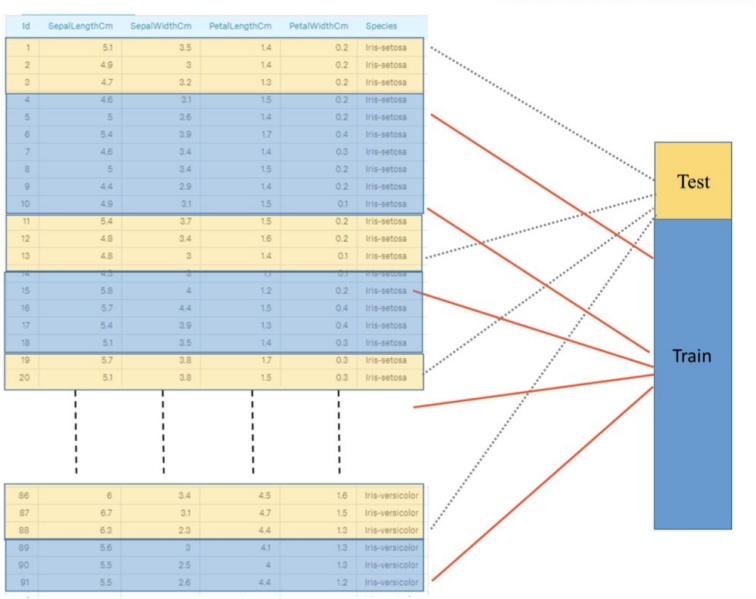
クロスバリデーション

How big you should make K for k-fold cross validation?

Kはいくつがいいの?

https://stats.stackexchange.com/questions/157689/how-big-to-make-k-for-cross-validation

1. k=1: 十分に大きいデータセットならば、Holdout法(train_test_splitと同じこと)を用いる。
2. k=4 or 5 or 10: 一般的に分析コンペでは、4、5、10をよく見かける。
3. k=n: nはデータセットのレコード数そのもの。レコード数が極めて少ないときの対処。
   アプローチは、leave-one-out法と呼ばれる。



クロスバリデーション

Does k-fold cross validation always imply k uniformly sized subsets?
分割されたサブセットの数はいつも均等なの?
https://stats.stackexchange.com/questions/134266/does-k-fold-cross-validation-always-imply-k-uniformly-sized-subsets

なるべく等分になるように分割されるようになっている。
101個のデータセットについて10-fold cross-validationを指定した場合、ひとつのfoldを11個として、自動的に調整される。

クロスバリデーション

良いCross validationとは?

いかに良いValidation(検証)データが作れるかにかかっている。
Validationデータは、テストデータの分布に似ていることが望ましい。

shuffle, random_stateといったパラメータも設定してみよう

```
from sklearn.model_selection import KFold, StratifiedKFold
```

sklearn.model_selection クラスをimportして、cross validationしてみよう!
※ StratifiedKFoldは、サンプルの各クラスの割合を保ったまま分割する手法

*たまにsklearn.cross_validationのKFoldやStratifiedKFold を使っている記事がありますが、それは古い表記法なので注意

http://segafreder.hatenablog.com/entry/2016/10/18/163925

グリッドサーチ

How to Tune Algorithm Parameters ?
パラメータチューニングはどんな手法がある?
Grid Searchが最も時間がかかる。

Grid Search
グリッドで指定されたアルゴリズムパラメータの各組み合わせ
に対して、モデルを系統的に構築し、評価する

Random Search
一定数の反復のためにランダム分布(ランダムな事象を表現
する確率分布、すなわち一様分布)からパラメータをサンプリングする

Bayesian Optimization
参考サイト
https://github.com/fmfn/BayesianOptimization

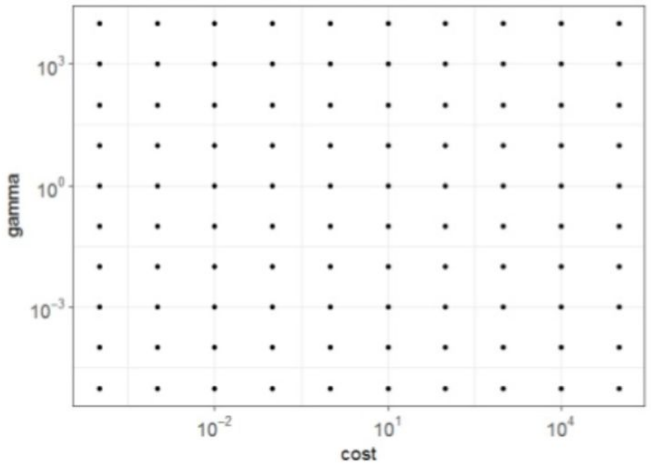
グリッドサーチ

Grid Search はなにをしてるの?

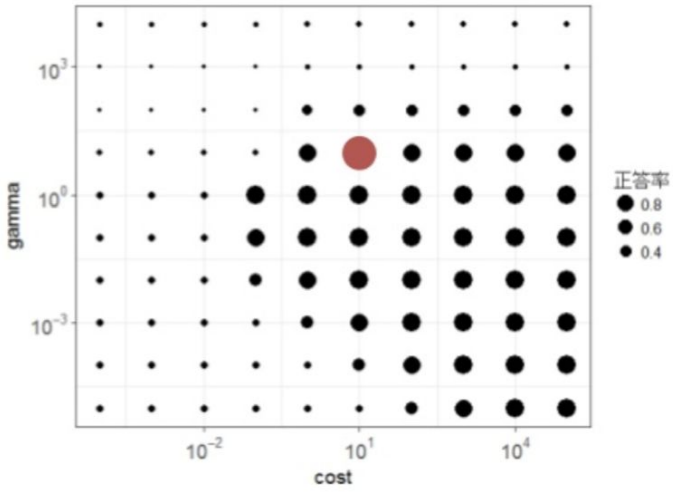
ハイパーパラメータの取りうる値によって格子をつくり、網羅的に(すべての格子点の中から)モデルの評価のよい組み合わせを求める。

例:SVM
Gamma = (10^-5, 10^4, … , 10^2, 10^3) # 小さいほど単純な決定境界になる
Cost = (10^-5, 10^4, … , 10^2, 10^3) # Cと表記される。小さいほど誤分類を許容する

- 探索範囲を格子状(グリッド)にする



- すべての格子点で正答率を求める



20

グリッドサーチ

Grid Search (Sklearn)では
探索した結果はどんな指標で評価されるの?

回帰問題:　決定係数(R^2)
分類問題:　正答率

グリッドサーチ

Model	Parameters to optimize	Good range of values
Linear Regression	• fit_intercept • normalize	• True / False • True / False
Ridge	• alpha • Fit_intercept • Normalize	• 0.01, 0.1, 1.0, 10, 100 • True/False • True/False
k-neighbors	• N_neighbors • p	• 2, 4, 8, 16 …. • 2, 3
SVM	• C • Gamma • class_weight	• 0.001, 0.01…..10...100...1000 • 'Auto', RS* • 'Balanced' , None
Logistic Regression	• Penalty • C	• L1 or l2 • 0.001, 0.01…..10...100
Naive Bayes (all variations)	NONE	NONE
Lasso	• Alpha • Normalize	• 0.1, 1.0, 10 • True/False
Random Forest	• N_estimators • Max_depth • Min_samples_split • Min_samples_leaf • Max features	• 120, 300, 500, 800, 1200 • 5, 8, 15, 25, 30, None • 1, 2, 5, 10, 15, 100 • 1, 2, 5, 10 • Log2, sqrt, None
Xgboost	• Eta • Gamma • Max_depth • Min_child_weight • Subsample • Colsample_bytree • Lambda • alpha	• 0.01,0.015, 0.025, 0.05, 0.1 • 0.05-0.1,0.3,0.5,0.7,0.9,1.0 • 3, 5, 7, 9, 12, 15, 17, 25 • 1, 3, 5, 7 • 0.6, 0.7, 0.8, 0.9, 1.0 • 0.6, 0.7, 0.8, 0.9, 1.0 • 0.01-0.1, 1.0 , RS* • 0, 0.1, 0.5, 1.0 RS*

ハイパーパラメータの探索範囲の事例
これはKaggleブログに掲載されていた一つの事例。データセットによって最適なパラメータ探索範囲は変わるので、必ずしも常に この通り行ってうまくいくとは限らない。
また、探索対象のハイパーパラメータが多すぎたり、探索範囲を広げすぎると、膨大な時間がかかるので、まずは少ない選択肢から始めよう。

scikit-learnの表記事例:

```
clf = GridSearchCV(estimator=RandomForestClassifier(),
param_grid=parameter_candidates,
                   cv=5,
                   refit=True,
                   error_score=0,
                   n_jobs=-1)
clf.fit(data_X, data_y)
cv_result = pd.DataFrame(clf.cv_results_)
cv_result # cv_results_ 属性で各パラメータごとの学習結果を確認できる
```

https://scikit-learn.org/stable/modules/grid_search.html

Submission

検証が終わったら、全トレーニングデータで訓練し、そのモデルでテストデータの予測を作ります。検証でトレーニングに用いたモデルインスタンスはこの最後の推定用モデルには用いません(標準的には)。

予測が出たら、ここを見てSubmission Fileを作ってみよう。

https://www.kaggle.com/dansbecker/submitting-from-a-kernel

↑こちらのHouse Priceのkernelから、Submission Fileの準備の仕方を学ぼう。カラム名はHouse Price用になっているので注意！

機械学習フロー
完