応用プログラミング3

第11回　機械学習
フレームワークによる
機械学習（2）

専修大学ネットワーク情報学部
田中健太

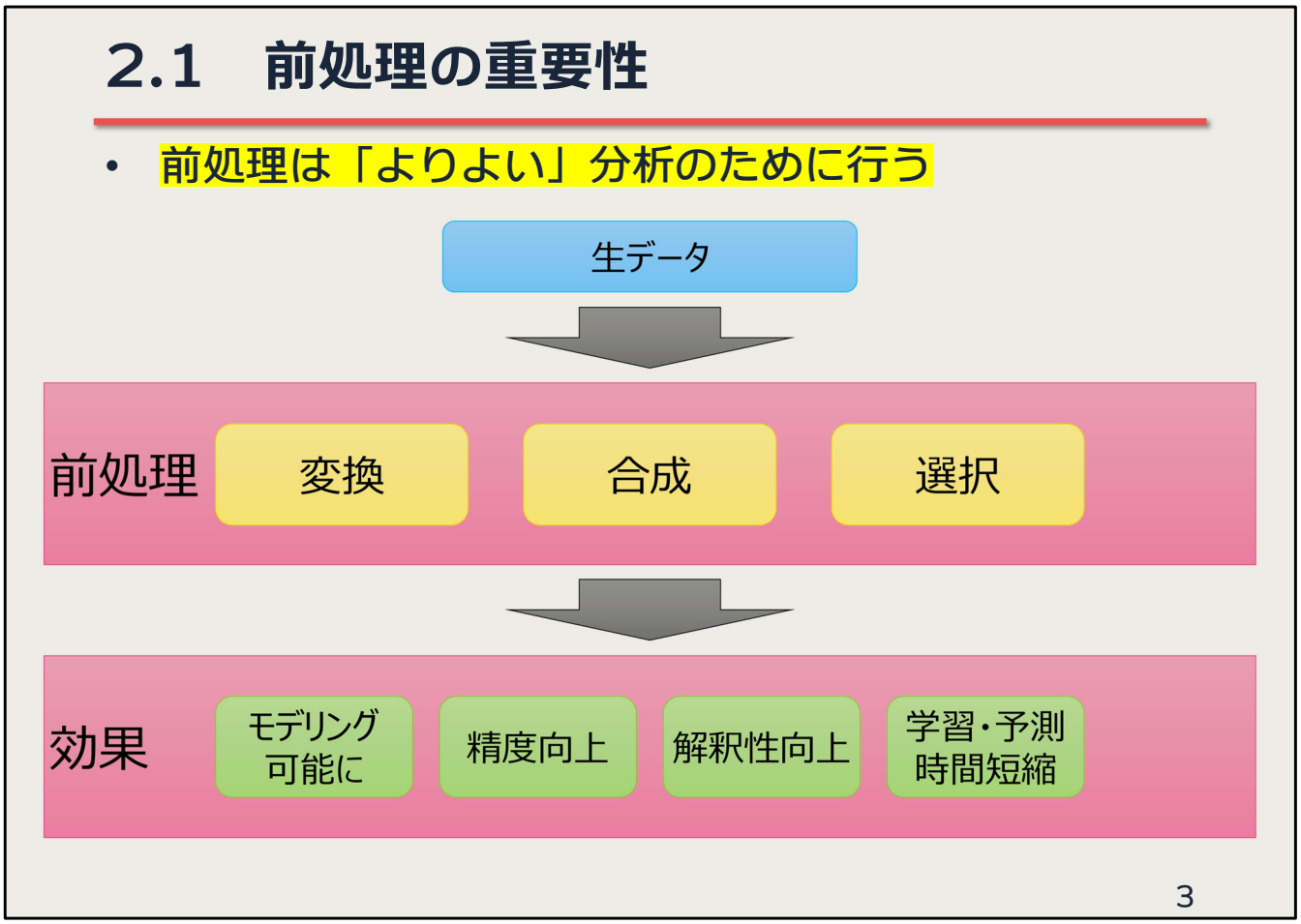
1．課題の振り返り

1

2. 機械学習と前処理

2

今回の講義では、機械学習における前処理を中心に取り上げます。モデルの解釈性あるいは予測精度を高めるために、前処理は重要な役割を果たします。今回も機械学習フレームワークtidymodelsを使用する前提で、さまざまな前処理の方法を紹介します。



前処理 (Data preprocessin, Data wrangling) には、大きく3つのアプローチがあります。

- 変換: 正規化、標準化、対数変換、べき乗変換、ダミー変数化など
- 合成: 多項式特徴量、交互作用特徴量など
- 選択: サンプリング、条件抽出など

いずれのアプローチも、目的は「よりよい」分析のためです。よりよい分析とは、モデルの解釈性を高める、予測精度を高める、実行速度が速くなる、あるいはそもそもモデルが作成可能になる、といったことです。収集したばかりの生データ (raw data) はそのままでは扱いづらく、仮にモデリングできたとしても、よい結果が得られないことがほとんどです。そのため、多くの場合、データ分析には前処理の工程が必要になります。

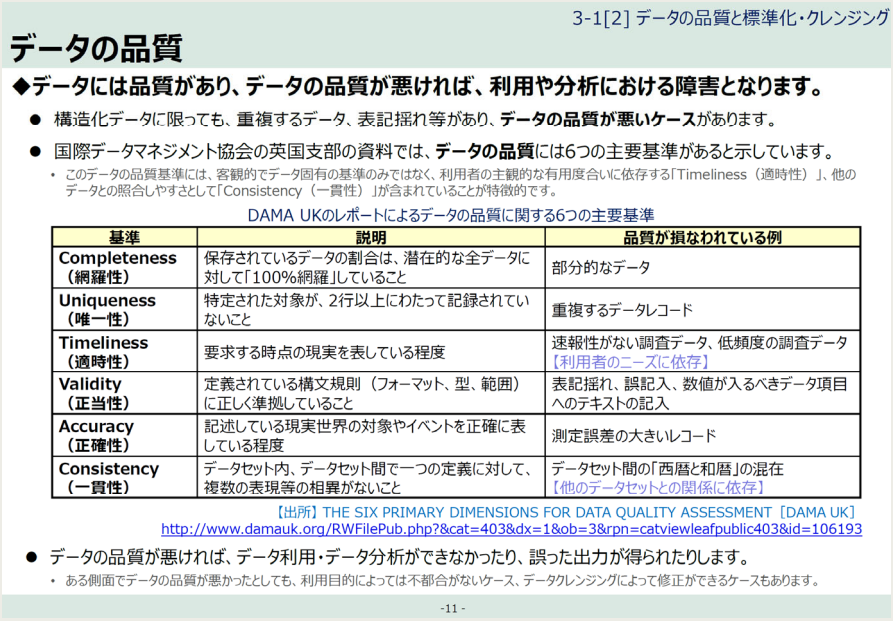
よく、データ分析かいわいでは、「データ分析は前処理が8割、9割、95%…」といったように語られます。プログラミングスキルやアルゴリズムについての知識も必要ですが、それらと同等以上に、前処理についてのスキルも必要とされます。研究やビジネスにおいて、データ分析を仕事にしようと考えている方は、前処理スキルを磨いておくと、現場で求められる人材となることができるでしょう。

以下は、データサイエンティスト協会が公開している、前処理スキルを高めるための学習コンテンツです。R, Python, SQLに対応しています。Docker環境やGoogle Colabなどで実行できます。

- 参考: GitHub - The-Japan-DataScientist-Society/100knocks-preprocess: データサイエンス100本ノック（構造化データ加工編） https://github.com/The-Japan-DataScientist-Society/100knocks-preprocess

2.2　データの品質

- 前処理はデータ品質を高めるために行う



3-1[2] データの品質と標準化・クレンジング

データの品質

◆データには品質があり、データの品質が悪ければ、利用や分析における障害となります。

- 構造化データに限っても、重複するデータ、表記揺れ等があり、**データの品質が悪いケース**があります。
- 国際データマネジメント協会の英国支部の資料では、**データの品質**には6つの主要基準があると示しています。
  - このデータの品質基準には、客観的でデータ固有の基準のみではなく、利用者の主観的な有用度合いに依存する「Timeliness（適時性）」、他のデータとの照合しやすさとして「Consistency（一貫性）」が含まれていることが特徴的です。

DAMA UKのレポートによるデータの品質に関する6つの主要基準

基準	説明	品質が損なわれている例
Completeness（網羅性）	保存されているデータの割合は、潜在的な全データに対して「100%網羅」していること	部分的なデータ
Uniqueness（唯一性）	特定された対象が、2行以上にわたって記録されていないこと	重複するデータレコード
Timeliness（適時性）	要求する時点の現実を表している程度	速報性がない調査データ、低頻度の調査データ【利用者のニーズに依存】
Validity（正当性）	定義されている構文規則（フォーマット、型、範囲）に正しく準拠していること	表記揺れ、誤記入、数値が入るべきデータ項目へのテキストの記入
Accuracy（正確性）	記述している現実世界の対象やイベントを正確に表している程度	測定誤差の大きいレコード
Consistency（一貫性）	データセット内、データセット間で一つの定義に対して、複数の表現等の相異がないこと	データセット間の「西暦と和暦」の混在【他のデータセットとの関係に依存】

【出所】THE SIX PRIMARY DIMENSIONS FOR DATA QUALITY ASSESSMENT［DAMA UK］
http://www.damauk.org/RWFilePub.php?&cat=403&dx=1&ob=3&rpn=catviewleafpublic403&id=106193

- データの品質が悪ければ、データ利用・データ分析ができなかったり、誤った出力が得られたりします。
  - ある側面でデータの品質が悪かったとしても、利用目的によっては不都合がないケース、データクレンジングによって修正ができるケースもあります。

-11 -

出典：総務省 ICTスキル総合習得プログラム コース3（データ分析） 3-2 データのクレンジングと可視化

4

前のページで、前処理はよりよいデータ分析のため、と述べましたが、別の表現をすると「データ品質を高めるため」とも言えます。データ品質についての明確な定義はありません（というか、ビジネスではいろいろな場面でやたらと品質、品質と言いますが、そもそも品質という概念を定義できていないことが多いです）が、大まかには「活用しやすいデータであるか」という観点で評価されるものだと思います。

ここでは、総務省が公開している「ICTスキル総合習得プログラム」（ https://warp.da.ndl.go.jp/info:ndljp/pid/12475307/www.soumu.go.jp/ict_skill ）の中でデータ品質について論じているページを示しています。この中で、国際データマネジメント協会（DAMA）が示す、データ品質を評価する基準が6つ挙げられています。収集時点で意識しないと、後から対処することが難しい項目が多いですが、このような観点を意識して手元のデータを観察し、必要に応じて適切な前処理を行うことで、データ品質を高めることができます。

なお、データマネジメントは、分析者が個人で留意するものではなく、研究プロジェクトあるいは企業全体で意識すべき取り組みで、近年その重要性に注目が集まっています。

- 参考: データマネジメントの知識体系"DMBOK"とは？どう役立つのか？
  https://jp.drinet.co.jp/blog/about_dmbok

2.3　典型的な前処理

手法	目的	組み込み関数	tidymodels	その他
正規化	データを［0, 1］の範囲に変換する	なし	step_range()	scales::rescale()
標準化	データを平均0、標準偏差1の範囲に変換する	scale()	step_normalize()	
対数変換	データの対数を取る	log(), log10()	step_log()	scales::log_trans()
べき乗変換	データ x を x^n に変換する	なし	step_YeoJohnson()	scales::boxcox_trans()
無相関化	データ間の相関を0にする	prcomp()	step_pca()	psych::principal()
欠損値の除去	欠損値を含む行を削除する	na_omit()	step_naomit()	tidyr::drop_na()
外れ値の処理	極端に大きい・小さい値を検出・処理する	なし	step_spatialsign()	Rlof::lof()
ダミー変数化	カテゴリーデータを数値化する	model.matrix()	step_dummy()	fastDummies::dummy_cols()
ビニング	数値データをカテゴリー化する	cut()	step_discretize()	infotheo::discretize()

5

ここでは、さまざまなデータに対して適用されることの多い、典型的な前処理の手法を列挙しています。詳しくは次のページ以降で紹介していきますが、いずれにしても、よくある前処理は、すべてRで簡単にできる、ということを知っておいてください。

2.4　正規化・標準化

- 線形モデルやクラスタリングでは、説明変数間で桁を揃える必要がある
- 正規化は［0, 1］の範囲に数値を変換する

$$x'_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- 標準化は平均0、標準偏差1の範囲に変換する

$$z_i = \frac{x_i - \mu}{\sigma}$$

- tidymodelsでは、step_range() 関数やstep_normalize() 関数で実行できる
- 数値以外のデータが含まれる場合はstep_normalize(all_numeric_predictors()) とする

6

はじめに、正規化・標準化について取り上げます。これらは、特に線形（直線か曲線かによらず）モデルと、クラスタリングにおいて効果的です。線形モデルにおいては、各特徴量にかかる係数（重み）を、目的変数に及ぼす影響の強さとして考えることができますが、目的変数や他の特徴量との間で数値の桁が大きく異なっていると、「影響が大きいから大きな係数なのか、単に桁が小さいから（調整のために）大きな係数なのか」が判断しにくく、モデルの解釈性が低下します。また、クラスタリングにおいても、データ間の距離（類似度）を計算する際に、大きな桁の特徴量に引っ張られて、小さな桁の特徴量における差異が見えにくくなり、適切なグループ分けができないことがあります。

そのため、特徴量どうしで大きく桁が異なるようなデータでは、それぞれの数値を一定の範囲に揃える正規化や標準化が必要です。一方で、決定木やランダムフォレストなどでは、「以上、以下」の分岐をするだけなので、桁を揃える必要はありません。

正規化は、一般に最小値0、最大値1となるように変換します。一方、標準化は平均0、標準偏差1になるように変換します。計算自体は、それぞれシンプルです。Rでは標準のscale() 関数でも正規化、標準化ができます。tidymodelsの文脈では、recipesパッケージのstep_range()（正規化）関数やstep_normalize()（標準化）関数を使います。いずれも、特にオプションの指定は必要ないでしょう。

2.4　正規化・標準化

```
library(tidymodels)

iris_split <- initial_split(iris, prop = 0.8)

iris_train <- training(iris_split)
iris_test <- testing(iris_split)

iris_rec <- recipe(Species ~ ., data = iris_train) %>%
    step_normalize(all_numeric_predictors()) %>%
    prep()

iris_train2 <- bake(iris_rec, new_data = NULL)
head(iris_train2)
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1        0.649      -0.601         1.03        1.33  virginica
## 2       -0.416      -1.77          0.137       0.140 versicolor
```

7

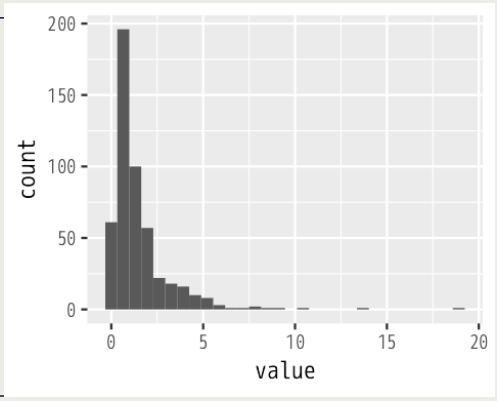
なお、レシピとして登録した処理を、実際にデータに適用するには、まず prep() 関数を実行します。この段階ではじめて、必要なパラメーター、例えば正規化であれば最小値と最大値が計算されます。

そして、前処理を適用したデータフレームを生成するには、bake() 関数を使います。レシピをもとに「焼き上げ」ます。以前のrecipesパッケージでは、学習用データには bake()、検証用あるいは予測用のデータには juice() と（意味不明に）関数が分かれていましたが、現在ではどちらも bake() 関数を使います。juice() の機能は、bake(..., new_data = NULL) とするようになりました。

2.5　対数変換

- データが対数正規分布に従いそうな場合、対数変換する
- 単純な変換には log() または log10() 関数を使う
- tidymodelsでは、step_log() 関数を使う
- 数値以外のデータが含まれる場合は step_log(all_numeric_predictors()) とする

```
# 意図的に対数正規分布に沿うデータを生成
df <- data.frame(id = 1:500,
   value = exp(rnorm(500)))

ggplot(df, aes(x = value)) +
   geom_histogram()
```



8

次に、対数変換を取り上げます。これは、データの対数 log を取って変換する操作です。特徴量の分布を観察したときに、0以上の値で右に裾野の長い、対数正規分布に従いそうな形状である場合に、対数を取ると、正規分布に近い形状に変換できます。多くのアルゴリズムで、データが正規分布していることは望ましい条件なので、モデルの当てはまりがよくなります。

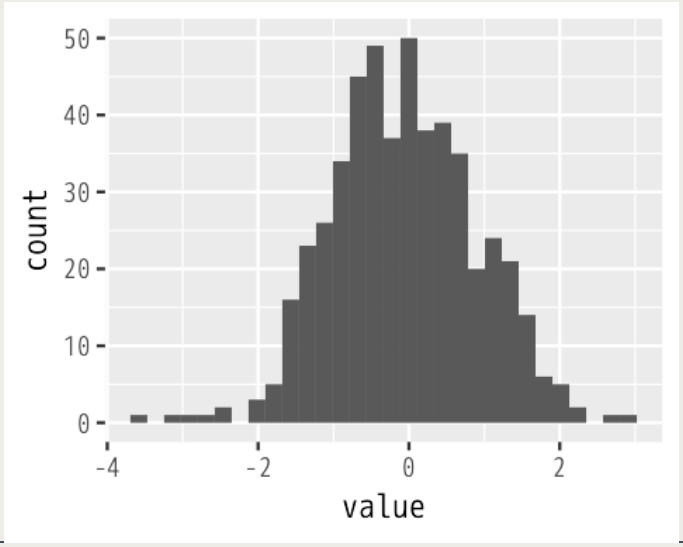
処理としては、対象となる列に組み込みの log() または log10() 関数を適用するだけでもよいですが、tidymodelsでは step_log() 関数が使えます。常用対数（$\log_{10}$）を使いたい場合は、base = 10 とオプションを指定します。

また、データフレームにfactor型など、数値以外の列が含まれる場合、デフォルトではそれらの列にも変換を適用しようとしてエラーになります。そのため、「数値列だけ」を意味する all_numeric_predictors() 関数を引数に与えます。他にも、factor型だけを選択する all_factor_predictors() などがあります。

- 参考: Role Selection — has_role • recipes
  https://recipes.tidymodels.org/reference/has_role.html

2.5　対数変換

```
df_rec <- recipe(~ ., data = df) %>%
    step_log(value) %>%
    prep()
df2 <- bake(df_rec, new_data = NULL)
ggplot(df2, aes(x = value)) + geom_histogram()
```

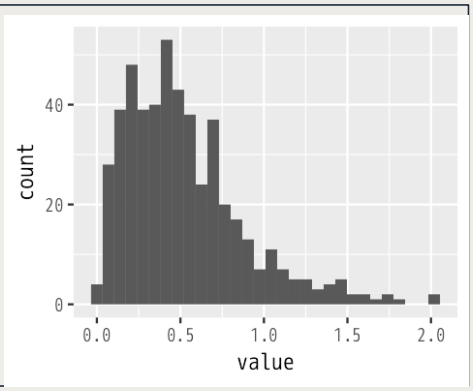


9

2.6　べき乗変換

- データxをx^{λ}に変換することをべき乗変換と言う
- データが正規分布になるようなλを探索する
- Yeo-Johnson変換（step_YeoJohnson()）が一般的
- Box-Cox変換（scales::boxcox_trans()）は基本的に正の値にのみ対応

```
df <- data.frame(id = 1:500,
   value = rgamma(500,shape=2,rate=4))

ggplot(df, aes(x = value)) +
   geom_histogram()
```



10

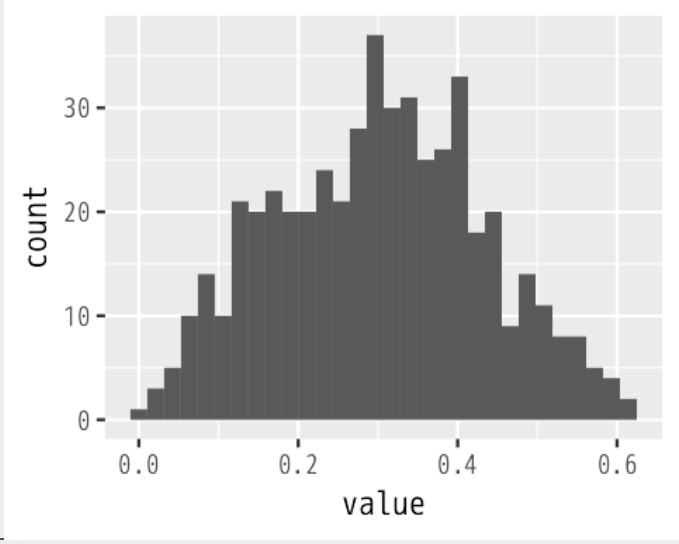
続いて、べき乗変換を取り上げます。これは、データ x について、λ 乗 (x^{λ}) すると正規分布に近づくような、適切な λ を探す処理です。計算過程は（いつものごとく）触れませんが、一般に、Yeo-Johnson変換と呼ばれる処理が使われます。他に、Box-Cox変換もよく書籍やWeb記事などで取り上げられますが、こちらは、(基本的には) 0より大きい正の値にしか対応していません。

tidymodelsでは、`step_YeoJohnson()` 関数が利用できます。`limits` オプションで、λ の探索範囲を指定できます。デフォルトは `limits = c(-5, 5)` です。大抵その範囲内で見つかると思いますが、どうしてもうまくいかない場合に、範囲を変更してみてください。

- 参考1: Yeo-Johnson Transformation — step_YeoJohnson • recipes
  https://recipes.tidymodels.org/reference/step_YeoJohnson.html
- 参考2: Box-Cox & modulus transformations — boxcox_trans • scales
  https://scales.r-lib.org/reference/boxcox_trans.html

2.6　べき乗変換

```
df_rec <- recipe(~ ., data = df) %>%
    step_YeoJohnson(value, limits = c(-10, 10)) %>%
    prep()
df2 <- bake(df_rec, new_data = NULL)
ggplot(df2, aes(x = value)) + geom_histogram()
```



11

2.7　無相関化

- 線形モデルでは、説明変数間に相関がないと仮定することが多い
- 主成分分析でデータを相関のない任意の主成分に変換できる
- ただし解釈が難しくなるので、研究では使う機会は少ない？
- 組み込みの `prcomp()` 関数やtidymodelsの `step_pca()` 関数で実行できる

12

線形（直線、曲線問わず）モデルにおける仮定として、「説明変数（特徴量）は互いに無相関である」というものがあります。特徴量どうしの相関が強いと、多重共線性（Multicollinearity）が生じ、係数の解釈が難しくなります。

- 参考1: 多重共線性　multicollinearity - 一般社団法人 日本理学療法学会連合 https://www.jspt.or.jp/ebpt_glossary/multicollinearity.html
- 参考2: 統計学と機械学習におけるマルチコ（多重共線性）に対する考えの相違 | キヨシの命題 https://yolo-kiyoshi.com/2019/05/27/post-1160/

多重共線性はモデルの予測精度には大きく影響しないと考えられているため、予測を重視する場合にはあまり問題になりませんが、モデルの解釈を重視する場合は、特徴量間の相関をなくす前処理が求められます。ひとつの方法として、相関係数をもとに片方の列を削除することがあります。また、ここで取り上げるのは、主成分分析を用いて、特徴量を、互いに相関がない（r = 0）いくつかの主成分として合成する方法です。例によって、主成分分析の理論については触れませんが、以下のURLを参考にしてください。

- 参考3: 主成分分析とは | 市場調査ならインテージ https://www.intage.co.jp/glossary/401/
- 参考4: Rで主成分分析 | キャスレーコンサルティング株式会社 https://www.casleyconsulting.co.jp/blog/engineer/119/

2.7　無相関化

```
iris_rec <- recipe(Species ~ ., data = iris_train) %>%
    step_pca(all_numeric_predictors(), threshold = .8,
         options = list(center = TRUE, scale. = TRUE)) %>%
    prep()
iris_train2 <- bake(iris_rec, new_data = NULL)
head(iris_train2)
##   Species       PC1     PC2
##   <fct>       <dbl>   <dbl>
## 1 virginica  -1.85   0.149
## 2 versicolor -0.478  1.76
```

```
cor(iris_train2[, 2:3]) # 相関がゼロになる
##             PC1         PC2
## PC1  1.000e+00 -5.929e-16
## PC2 -5.929e-16  1.000e+00
```

13

tidymodelsの文脈では、step_pca() 関数を適用できます。num_comp オプションに主成分の数を指定できます。主成分分析も、数値データにしか適用できないので、factor型などが混在している場合、all_numeric_predictors() 関数で対象を明示します。また、デフォルトでは標準化、中心化を行わない設定になっているので、他のレシピでそれらを実行していない場合は、options = list(center = TRUE, scale. = TRUE) と指定します。
scale.（スケール、ドット）に注意してください。

2.8　欠損値の処理

- 欠損値は「値が記録されていない」こと
- Rでは NA として扱われる
- すでに特別な NA となっているので、検出は容易（is.na() 関数など）
- 欠損値を含む行を削除するには、step_naomit() 関数などを使う
- 平均など統計量で補間するには、step_impute_*() 関数を使う

14

次に、欠損値の取り扱いについて紹介します。多くのアルゴリズムは、欠損値（Rでは NA）を含んだデータを処理できません。そのため、欠損値を取り除くか、「それらしい」値で置換する必要があります。欠損値の除去は、組み込み関数でも na.omit() や complete.cases() で可能です。tidymodelsの文脈では、step_naomit() 関数で実行できます。

欠損値を「それらしい」値で置換する方法はさまざまに存在します。単純なのは、平均や中央値などの統計量を用いる方法です。また、近年では、ブートストラップ法を用いて何度も計算を繰り返し、妥当な値を求める多重代入法が使われることが増えています。tidymodelsでは、これらの方法が step_impute_*() 関数として実装されています。特性を理解して、適切な手法を選択してください。

- 参考1: 欠損値があるデータの分析 | Sunny side up! https://norimune.net/1811
- 参考2: Rで実践！欠損データ分析入門【1】 | NHN テコラス Tech Blog https://techblog.nhn-techorus.com/archives/6309
- 参考3: R のモダンな NA 処理まとめ - Qiita https://qiita.com/five-dots/items/361a42baf1e94edf5846
- 参考4: Function reference • recipes - Imputation https://recipes.tidymodels.org/reference/index.html#step-functions-imputation

2.8　欠損値の処理

```
df_rec1 <- recipe(タイプ ~ ., data = df) %>%
    step_naomit(all_predictors(), all_outcomes()) %>% prep()
df1 <- bake(df_rec1, new_data = NULL)
summary(df1)
```

```
## アルコール分     日本酒度          エキス分        総酸             アミノ酸度
##  Min.   :12.7   Min.   :-6.35   Min.   :2.13   Min.   :0.585   Min.   :0.197
##  1st Qu.:14.9   1st Qu.: 1.48   1st Qu.:4.13   1st Qu.:1.060   1st Qu.:1.070  ...
```

```
df_rec2 <- recipe(タイプ ~ ., data = df) %>%
    step_impute_mean(all_numeric_predictors()) %>%
# all_nominal_predictors() は質的変数、all_outcomes() は目的変数を指す
   step_impute_mode(all_nominal_predictors(), all_outcomes()) %>%
    prep()
df2 <- bake(df_rec2, new_data = NULL)
summary(df2)
```

```
##   アルコール分      日本酒度          エキス分         総酸          アミノ酸度
##  Min.   :12.7   Min.   :-6.82   Min.   :2.13   Min.   :0.585   Min.   :0.133
##  1st Qu.:14.9   1st Qu.: 1.48   1st Qu.:4.13   1st Qu.:1.062   1st Qu.:1.070...
```

15

なお、多重代入法についてはtidymodelsの関数ではサポートされていません。これが定番だ、というパッケージもまだない状況ですが、missForestパッケージやmlimパッケージが比較的容易に利用できそうです。

- 参考5: GitHub - stekhoven/missForest https://github.com/stekhoven/missForest
- 参考6: GitHub - haghish/mlim https://github.com/haghish/mlim

2.9　外れ値の処理

- 外れ値（Outlier）は他のデータと比べて極端に大きいまたは小さい値
- どれくらいから「極端に」かは場合による
- 目視で決めることもできるが、近傍点との距離を計算し、統計的に求めることもできる
- シンプルな方法がLocal Outlier Factor（LOF）
- あるいは、外れ値が発生しにくいようにデータを変換する

16

続いて、外れ値の検出について取り上げます。外れ値 (Outlier) は、「他のデータと比べると極端に大きい / 小さい値」と言えます。どこから極端に大きい / 小さいのかについては、基準はありません。シンプルな対処としては、ヒストグラムを描き、分布の形状から「ここから先は外れ値」と決めて、取り除くまたは統計量と置換する方法があります。他に、統計的な手法を用いて、外れ値の検出を行うこともできます。いくつかの手法がありますが、基本的には周囲のデータとの "距離" をもとに判定します。ここでは、Local outlier factor法を実行するためのRlofパッケージを使う例を示しています。`lof()` 関数の `k` オプションに、近傍いくつのデータを使って外れ値判定をするかを指定します。`k` の値についても基準はありませんが、筆者の経験的には、データ数の10%前後の値を指定するとよいように思います。また、関数の実行結果は、それぞれのデータが他のデータとどれくらい離れているかを示す値です。これも、どれくらい大きければ外れ値とみなすかは決まっていないため、外れ値として検出してほしいデータだけが検出されるちょうどよい値を決めます (本末転倒ですが)。

- 参考1: Local Outlier Factor (LOF) による外れ値検知についてまとめた - コンサルでデータサイエンティスト https://hktech.hatenablog.com/entry/2018/09/04/002034

なお、まだCRANに登録されていない、開発段階ですが、tidymodelsの文脈で外れ値検出ができるtidy.outliersパッケージがあるようです。今後、この機能がtidymodelsに取り込まれたり、パッケージがCRANに登録されて広く使われるようになるかもしれません。

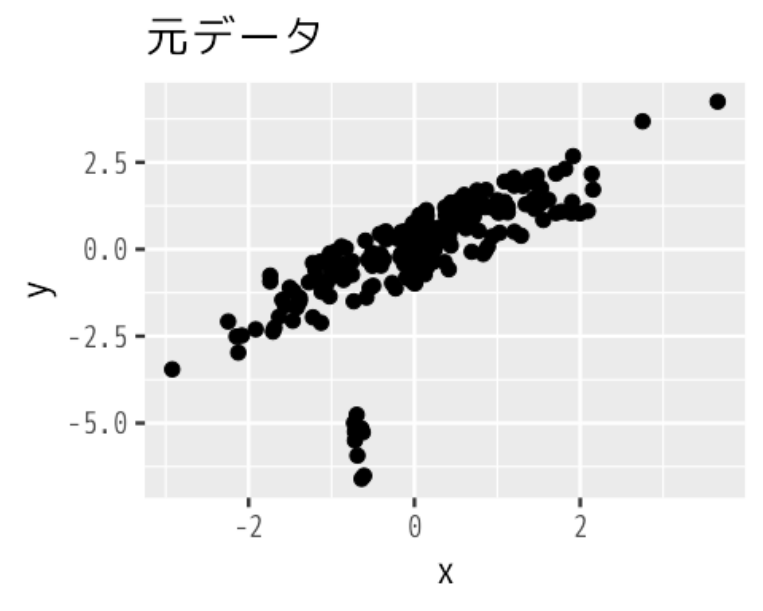
- 参考2: GitHub - brunocarlin/tidy.outliers: Outliers handling in tidymodels https://github.com/brunocarlin/tidy.outliers/

2.9　外れ値の処理

```
library(Rlof)
# データを生成
set.seed(334)
x <- sort(rnorm(200))
df <- data.frame(x = x, y = x + runif(100, min = -1, max = 1))
df[51:60, "y"] <- df[51:60, "y"] - 5 # 意図的に外れ値を生成

# 近傍100点を使ったLOF
df$lof_res <- lof(df, k = 20)
df$lof_res

##   [1] 2.2477 1.3268 1.4696 1.7060 1.4383 1.3064 1.2315 1.1434...

summary(df$lof_res)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.955   0.989   1.039   1.153   1.144   3.830

# LOF scoreが1.5を超える値に外れ値フラグをつける
df$outlier <- ifelse(df$lof_res >= 1.5, TRUE, FALSE)
```

2.9　外れ値の処理

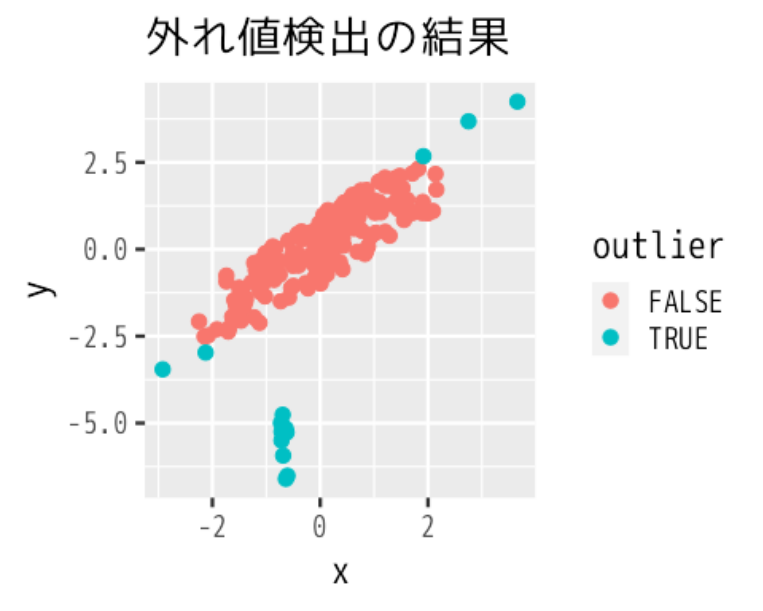
```
ggplot(df, aes(x = x, y = y)) +
  geom_point(size = 3) + labs(title = "元データ")
```



18

2.9　外れ値の処理

```
# 外れ値フラグがついたデータを除いたヒストグラムを描画
ggplot(df, aes(x = x, y = y, colour = outlier)) +
  geom_point(size = 3) + labs(title = "外れ値検出の結果")
```



19

2.10　ダミー変数化

- カテゴリーデータを線形モデルに組み込むために、「〇〇であるかどうか」の 1/0 に変換する
- 統計学の文脈ではダミー変数化、機械学習の文脈では One Hot Encodingと呼ばれる
- 最後のカテゴリーを残すかどうかも文脈によって異なる
- 組み込みの `model.matrix()` 関数やrecipesパッケージの `step_dummy()` 関数で変換する

20

ここでは、ダミー変数化について取り上げます。ダミー変数化は、質的データ（性別、居住都道府県など）を線形モデルに組み込む際に用いられる手法です。1列に格納された質的データを、ラベルごとに列方向に展開（「性別」列を「男性」「女性」列に展開など）し、その条件に当てはまるかを1 / 0で表現します。このようにすることで、質的データのラベルごとに異なる係数を求めることができます。なお、展開する列の数はラベル数 - 1になります（「男性」列の値が1なら男性、0なら女性、「北海道」から「鹿児島県」までが0なら沖縄、など最後のラベルが一意に定まるため）。

ダミー変数化は、組み込みの `model.matrix()` 関数でも可能です。引数に目的変数と特徴量の関係をあらわす式を与えます。この時、特徴量について `0+` とすると、最後のラベルも残してダミー変数化します（. はすべての列をあらわす）。tidymodelsの文脈では、`step_dummy()` 関数が利用できます。`onehot = TRUE / FALSE` オプションを `TRUE` とすると、最後のラベルを残します。

- 参考1: model.matrix() でダミー変数化と相互作用項 - Qiita
  https://qiita.com/daifuku_mochi2/items/b132dd5049feee6db7d0
- 参考2: Create traditional dummy variables — step_dummy • recipes
  https://recipes.tidymodels.org/reference/step_dummy.html

2.10　ダミー変数化

```
model_mat <- model.matrix(as.formula(タイプ ~ 0+.), data = df)
head(model_mat, 5)
```

```
...
##   容器紙パック 容器リターナブル瓶 容器茶色瓶 容器青色瓶 容器緑色瓶
## 1           0                 0          1          0          0
## 2           1                 0          0          0          0
...
```

```
df_rec <- recipe(タイプ ~ ., data = df) %>%
    step_dummy(all_nominal_predictors(), one_hot = TRUE) %>%
    prep()
df1 <- bake(df_rec, new_data = NULL)
head(df1, 5)
```

```
##   容器_紙パック 容器_リターナブル瓶 容器_茶色瓶 容器_青色瓶 容器_緑色瓶
##           <dbl>               <dbl>       <dbl>       <dbl>       <dbl>
## 1             0                   0           1           0           0
## 2             1                   0           0           0           0
...
```

21

2.11　ビニング

- 離散化とも呼ばれ、定量データ（連続値）を区間を区切ってラベルに変換する手法
- 線形モデルでは、離散化した区間ごとに異なる係数を設定でき、モデルの柔軟性が高まる
- 組み込みの `cut()` 関数やrecipesパッケージの `step_cut()` 関数、`step_discretize()` 関数で実行できる

22

次に、ビニングという処理を紹介します。これは離散化 (Discretization) とも呼ばれ、連続値をある基準に基づいて区間に分割する操作です。例えば、19歳、23歳といった年齢を10代、20代と分割するようなことが、典型的なビニングです。これも、線形モデルで効果を発揮する手法です。線形モデルで、連続値をそのままひとつの特徴量として組み込むと、その値が大きくても小さくても、同じひとつの係数しかかかりません。一方で、離散化して個別に特徴量として組み込めば、この区間には大き目の係数、この区間には小さめの係数、といった柔軟なモデリングができます。

離散化は、組み込みの `cut()` 関数で行えますし、tidymodelsにおいては、`step_cut()` 関数や `step_discretize()` 関数が利用できます。ふたつの `step_*()` 関数の違いは、`step_cut()` は区間数 `breaks` オプションを指定して等間隔に分割し、`step_discretize()` は `num_breaks` オプションで指定した区間内に、おおよそ同じ数のデータが含まれるように分割する、というものです。人間には等間隔に分割するほうがわかりやすいですが、機械学習アルゴリズムには、各区間に同じくらいのデータ数が存在するほうが、過学習などが減ってよいようです。

決定木やランダムフォレストなどのツリー系アルゴリズムでは、内部で同じことを行っている（境界値で分岐する）のですが、人間が経験をもとにあらかじめ適切に分割したほうが、結果の解釈性がよくなることもあります。

- 参考1: Cut a numeric variable into a factor — step_cut • recipes
  https://recipes.tidymodels.org/reference/step_cut.html
- 参考2: Discretize Numeric Variables — step_discretize • recipes
  https://recipes.tidymodels.org/reference/step_discretize.html

2.11　ビニング

```
iris_rec1 <- recipe(Species ~ ., data = iris_train) %>%
    step_cut(all_numeric_predictors(), breaks = 4) %>%
    prep()

iris_train1 <- bake(iris_rec1, new_data = NULL)
head(iris_train1)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <fct>        <fct>       <fct>        <fct>       <fct>
## 1 [4,7.9]      [2,4]       (4,6.9]      [0.1,4]     virginica
## 2 [4,7.9]      [2,4]       [1,4]        [0.1,4]     versicolor
...
```

23

2.11　ビニング

```
iris_rec2 <- recipe(Species ~ ., data = iris_train) %>%
    step_discretize(all_numeric_predictors(),
         num_breaks = 4, min_unique = 3) %>%
    prep()

iris_train2 <- bake(iris_rec2, new_data = NULL)
head(iris_train2)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##   <fct>        <fct>       <fct>        <fct>       <fct>
## 1 bin2         bin2        bin3         bin3        versicolor
## 2 bin1         bin3        bin1         bin1        setosa
## 3 bin4         bin2        bin4         bin4        virginica
...
```

2.12　アンダーサンプリング・オーバーサンプリング

- 分類モデルにおいては、ラベルごとのサンプルサイズが揃っていることが望ましい
- アンダーサンプリングは、少ないほうのラベルに合わせて他のラベルのサンプルを減らす
- オーバーサンプリングは、多いほうのラベルに合わせて他のラベルのサンプルを増やす
- tidymodelsではthemisパッケージの各種関数で実行できる

```
library(themis)
table(df[["タイプ"]])
##
##   一般酒   吟醸酒   純米酒 本醸造酒
##      647      136      125       90
```

25

続いて、アンダーサンプリング・オーバーサンプリングについて紹介します。この手法は、大まかには「各ラベルごとのサンプルサイズを揃えるためにデータを増やしたり減らしたりする」というものです。分類モデルを作成する際、目的変数のラベル（種類）の度数が偏っていると、学習結果が歪んでしまいます。そのため、学習用のデータでは、ラベルごとの度数が揃っていることが望ましいです。これを実現するため、アンダーサンプリングでは少ないラベルに合わせて、多いラベルのデータを削除します。一方、オーバーサンプリングでは、多いラベルに合わせて少ないラベルのデータを「水増し」します。実際には存在しないデータを、既存のデータの特性（分布など）に合わせて生成します。

tidymodelsでは、関連するthemisパッケージを組み合わせることで実行できます。`step_downsample()` 関数や `step_upsample()` 関数などが利用できます。それぞれ、`under_ratio`（`step_downsample()`）と `over_ratio`（`step_upsample()`）という引数に、「少ない（また多い）ラベルに対する他のラベルの割合」を0から1の値で指定します。デフォルトは1で、これは少ない（または多い）ラベルと他のラベルが同数になるようにします。実際には、まったく同じ数である必要はないので、0.7から0.8くらいでよいかもしれません。また、より高度なアルゴリズムを使う方法もありますので、ドキュメントを参照してください。

- 参考: Extra Recipes Steps for Dealing with Unbalanced Data • themis
  https://themis.tidymodels.org/

なお、アンダーサンプリングは、せっかく集めたデータを減らしてしまうため、オーバーサンプリングは存在しないデータを生成するため、いずれもモデルの精度が下がってしまう可能性もあるため、適用には注意が必要です。

2.12 アンダーサンプリング・オーバーサンプリング

```
df_rec1 <- recipe(タイプ ~ ., data = df) %>%
    step_downsample(タイプ) %>%
    prep()
df1 <- bake(df_rec1, new_data = NULL)
table(df1[["タイプ"]])
##
##   一般酒   吟醸酒   純米酒 本醸造酒
##       90       90       90       90
```

```
df_rec2 <- recipe(タイプ ~ ., data = df) %>%
    step_upsample(タイプ) %>%
    prep()
df2 <- bake(df_rec2, new_data = NULL)
table(df2[["タイプ"]])
##
##   一般酒   吟醸酒   純米酒 本醸造酒
##      647      647      647      647
```

26

2.13　交互作用特徴量・多項式特徴量の合成

- 特徴量（説明変数）どうしを掛け合わせたものを交互作用特徴量と言う
- 特徴量をn乗したものを多項式特徴量と言う
- 機械学習的アプローチでは、「効くかもしれない」操作を積極的に行う
- `step_interact()` 関数や `step_poly()` 関数で実行できる

27

次に、交互作用特徴量・多項式特徴量を合成する方法を紹介します。交互作用特徴量は、特徴量どうしを掛け合わせた値です。多項式特徴量は、それぞれの特徴量の値を2乗、3乗した値です。いずれも、統計学的な観点で見れば、「身長と血圧を掛け合わせた値は何なのか」「購入数量の3乗に何の意味があるのか」と解釈は困難ですが、機械学習的なアプローチでは「なぜか精度向上に効果がある」ことがあるため、積極的に取り入れられています。

それぞれの処理自体は、単純な掛け算なので、組み込み関数でも実現できますが、tidymodelsの文脈では交互作用特徴量は `step_interact()` 関数、多項式特徴量は `step_poly()` 関数で実行できます。`step_interact()` 関数では、組み合わせる特徴量を `terms` 引数に式の形で指定します。`step_poly()` 関数では、対象となる特徴量と、`degree` 引数で次数を指定します。`degree = 3` と指定すると、2乗、3乗した特徴量が生成されます。

- 参考1: Create Interaction Variables — step_interact • recipes
  https://recipes.tidymodels.org/reference/step_interact.html
- 参考2: Orthogonal Polynomial Basis Functions — step_poly • recipes
  https://recipes.tidymodels.org/reference/step_poly.html

2.13　交互作用特徴量・多項式特徴量の合成

```
df_rec1 <- recipe(タイプ ~ ., data = df) %>%
    step_interact(terms = ~
all_numeric_predictors():all_numeric_predictors()) %>%
    prep()

df1 <- bake(df_rec1, new_data = NULL)

head(df1)
```

```
##   容器     タイプ アルコール分_x_日本酒度 アルコール分_x_エキス分
##   <fct>    <fct>                  <dbl>                 <dbl>
## 1 茶色瓶   一般酒                 121.                   73.6
## 2 紙パック 一般酒                  -1.15                 75.4
##   アルコール分_x_総酸 アルコール分_x_アミノ酸度 アルコール分_x_甘辛度
##                 <dbl>                  <dbl>                <dbl>
## 1                14.2                   22.9                -8.27
## 2                21.8                   15.4                -1.54
...
```

28

2.13　交互作用特徴量・多項式特徴量の合成

```
df_rec2 <- recipe(タイプ ~ ., data = df) %>%
    step_poly(all_numeric_predictors(), degree = 3) %>%
    prep()

df2 <- bake(df_rec2, new_data = NULL)

head(df2)
```

```
##   容器     タイプ アルコール分_poly_1 アルコール分_poly_2 アルコール分_poly_3
##   <fct>    <fct>               <dbl>               <dbl>               <dbl>
## 1 茶色瓶   一般酒             0.0128             -0.0188             -0.0132
## 2 紙パック 一般酒             0.0260             -0.00863            -0.0230
##   日本酒度_poly_1 日本酒度_poly_2 日本酒度_poly_3 エキス分_poly_1
##             <dbl>           <dbl>           <dbl>           <dbl>
## 1        0.0394          0.0122         -0.0229          0.00947
## 2       -0.0356          0.00541         0.0281          0.0105
...
```

29

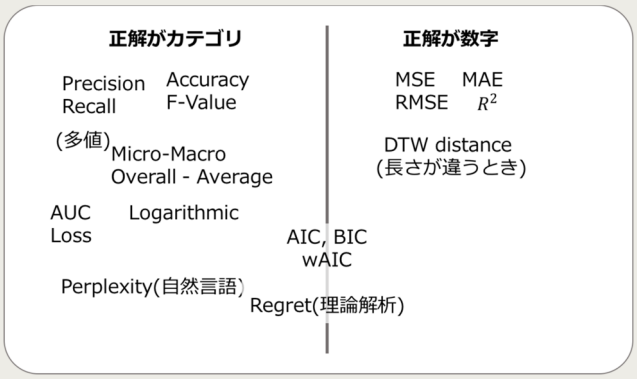
3. 機械学習とチューニング

30

ここからは、機械学習において最適なハイパーパラメーターを決めるためのチューニングについて取り上げます。といっても、あまり本格的な内容には踏み込みませんが、モデルの精度を上げるための入り口として捉えてください。

3.1　機械学習における評価指標

- 回帰モデルの評価指標としてR^2（決定係数）やRMSEが用いられる
- 分類モデルの評価指標として正解率、ROC曲線、AUCなどが用いられる
- 回帰、分類いずれにおいても使われる指標としてAICなどがある



出典: 機械学習で使う指標総まとめ(教師あり学習編) - プロクラシスト

31

すでに何度か示していますが、統計モデリングや機械学習には、さまざまな評価指標があります。「モデルの精度を上げる」とは、言い換えればこれらの指標をよくすることです。ハイパーパラメーターの異なる複数のモデルを作成し、何らかの評価指標を基準に、最も精度がよいと判断されるモデルを選択します。ハイパーパラメーターの値を変えて同じ処理を繰り返すだけなので、for文などを使って自動化ができますし、(Rに限らず) 多くの機械学習ライブラリにはそれをもっとシンプルに実行するための関数が用意されています。

- 参考: 機械学習で使う指標総まとめ(教師あり学習編) - プロクラシスト
  https://www.procrasist.com/entry/ml-metrics

3.2　チューニング対象のパラメーター

- 精度を高めるために、各アルゴリズムのハイパーパラメーターをチューニングする
- 線形回帰（GLM）：正則化の種類と強さ
    - L1正則化：標準化した説明変数から定数λを引き、目的変数と関係の少ない説明変数の係数をゼロに近づける
    - L2正則化：標準化した説明変数の係数の2乗和を最小化し、個々の係数をできるだけ小さくする
    - ElasticNet：L1正則化とL2正則化を両方適用する →両者のバランスをチューニングする
- 決定木：モデル全体の複雑さ、分岐の深さなど
- ランダムフォレスト：決定木の本数、1本の木に与える説明変数の数、分岐の深さなど
- クラスタリング（k-means法）：クラスタの数

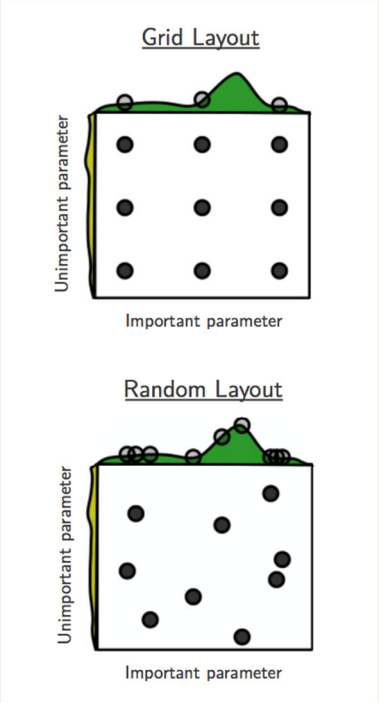
32

ここでは、アルゴリズムごとに、チューニング対象となるハイパーパラメーターを列挙しています。

なお、モデルのチューニングは、やりすぎると過学習してしまうリスクが高まります。学習用のデータにより適合させようという作業がチューニングですので、過剰なチューニングは、学習用データにしか通用しないモデルになる可能性をはらんでいます。そのため、随時検証用データで評価しながら、過学習と高精度のバランスを取ることが重要です。

3.3　グリッドサーチ

- チューニング対象のパラメーターを組み合わせて全通り試行する
- 組み合わせが多いと計算時間が長くなる
- 最適なパラメーターがうまく見つからない場合もある
- ランダムサーチやベイズ最適化などが使用されることもある



33

チューニングを（半）自動的に行う、もっともシンプルな方法がグリッドサーチです。チューニング対象となるハイパーパラメーターごとに、数値の範囲などを決めて、それらのすべての組み合わせについて、ひとつずつモデル作成から評価指標の算出までを行います。そして、もっとも評価指標がよい値となったモデルを選択します。パラメーターの組み合わせを格子状に配置するイメージから、グリッドサーチと呼ばれます。

グリッドサーチはシンプルな方法ですが、組み合わせ数が多いと、計算時間が飛躍的に増大します。また、パラメーターの決め方しだいでは、本来もっとも精度のよい範囲に当たらない可能性もあります。そのような課題を克服するため、ランダムサーチやベイズ最適化といったアプローチが開発されています。

- 参考: 第3回　機械学習のためのベイズ最適化入門 | Tech Book Zone Manatee
  https://book.mynavi.jp/manatee/detail/id=59393

3.4 tidymodelsにおけるグリッドサーチ

```
df_split <- initial_split(df, prop = 0.8)

df_train <- training(df_split)
df_test <- testing(df_split)

folds <- vfold_cv(df_train, v = 10)

metrics <- metric_set(accuracy)

df_rec <- recipe(タイプ ~ ., data = df_train) %>%
    step_normalize(all_numeric_predictors()) %>%
    prep()

dt_model <- decision_tree(mode = "classification",
   cost_complexity = tune(), min_n = tune()) %>%
    set_engine("rpart")
```

34

ここではtidymodelsにおいて、グリッドサーチを行う例を示しています。tidymodelsでは、tuneパッケージの tune_grid() 関数で、先に指定した指標を最適化するグリッドサーチができます。プログラム例では、まず vfold_cv() 関数で、k = 10の交差検証を行うように設定しています。そして、metric_set() 関数で最適化の対象となる指標（正解率）を指定しています。次に、モデルを定義する際に、チューニング対象となるハイパーパラメーターについて、tune() 関数で指定します。ここでは、複雑さのパラメーター cost_complexity と、最終的な分岐に最低何件のデータがあるように木を構成するか min_n を対象とします。

そして、tune_grid() 関数にワークフローを指定し、grid オプションに格子の数を指定します。ここでは、チューニング対象のパラメーターの範囲については、デフォルトでは自動的に決まります。指定したい場合は、dialsパッケージの各種関数を使い、先にグリッドを定義して、前記の grid オプションに、作成したグリッドを指定します。最終的に得られた結果を、ワークフローに finalize_workflow() 関数で適用することで、最適なハイパーパラメーターを指定したモデルが利用できるようになります。

なお、前述のように、グリッドサーチの計算量は多くなりがちなので、PCなどマルチコアが利用できる環境では並列処理も行われますが、ドキュメントがあまり整理されておらず、どのように記述するのが正解（妥当）なのかがはっきりしません。

- 参考1: Tidy Tuning Tools • tune https://tune.tidymodels.org/index.html
- 参考2: 【tidymodels講座7】{tune}, {dials}ハイパラチューニング - データサイエンスの道標 https://datasciencemore.com/tidymodels-tune-dials/
- 参考3: Tools for Creating Tuning Parameter Values • dials https://dials.tidymodels.org/

3.4 tidymodelsにおけるグリッドサーチ

```
df_flow <- workflow() %>%
    add_model(dt_model) %>%
    add_recipe(df_rec)

df_tune_flow <- tune_grid(df_flow, resamples = folds, metrics =
metrics, grid = 10)
show_best(df_tune_flow)
```

```
##   cost_complexity min_n .metric  .estimator  mean     n std_err               .config
##             <dbl> <int> <chr>    <chr>      <dbl> <int>   <dbl>                 <chr>
## 1   0.000000933      22 accuracy multiclass 0.950    10 0.00647  Preprocessor1_Model06
## 2   0.0000750        29 accuracy multiclass 0.949    10 0.00780  Preprocessor1_Model07
## 3   0.0000000151     19 accuracy multiclass 0.949    10 0.00780  Preprocessor1_Model08
## 4   0.00000000139    17 accuracy multiclass 0.946    10 0.0089   Preprocessor1_Model05
## 5   0.00000466       11 accuracy multiclass 0.946    10 0.00916  Preprocessor1_Model01
```

```
df_flow <- df_flow %>%
   finalize_workflow(parameters = select_best(df_tune_flow))
```

35

4. まとめ

4.1　今日の内容

- 機械学習と前処理
  - 正規化・標準化
  - 対数変換・
べき乗変換
  - 無相関化
  - 欠損値の処理
  - 外れ値の処理
  - ダミー変数化
  - ビニング
  - アンダーサンプリング・
オーバーサンプリング
  - 交互作用特徴量・
多項式特徴量の合成
- 機械学習とチューニング
  - グリッドサーチ

37

4.2　次回までの課題

- Posit Cloudプロジェクト内の
  `12_ml_framework_exercise.R` について、
  指示に従ってプログラムを作成してください
- 編集したファイルは、ファイル一覧でチェックを
  入れ、[more] メニューから [Export] を選択し、
  [Download] ボタンを押してダウンロードして
  ください
- ダウンロードしたファイルを、Classroomの
  課題ページから提出してください
- **提出期限: 2023-12-11 23:59**

38