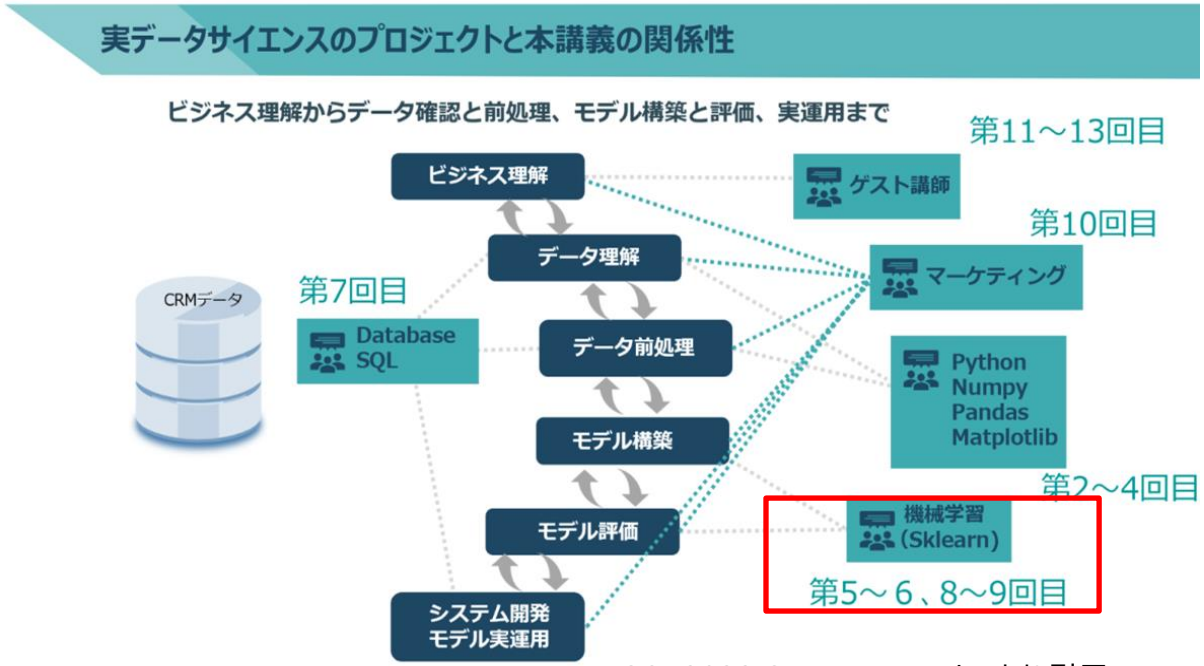
GCI 2024 Summer
Week9 特徴量エンジニアリング

松尾・岩澤研究室
Matsuo-Iwasawa Lab UTokyo

作成者・講師：石田将貴

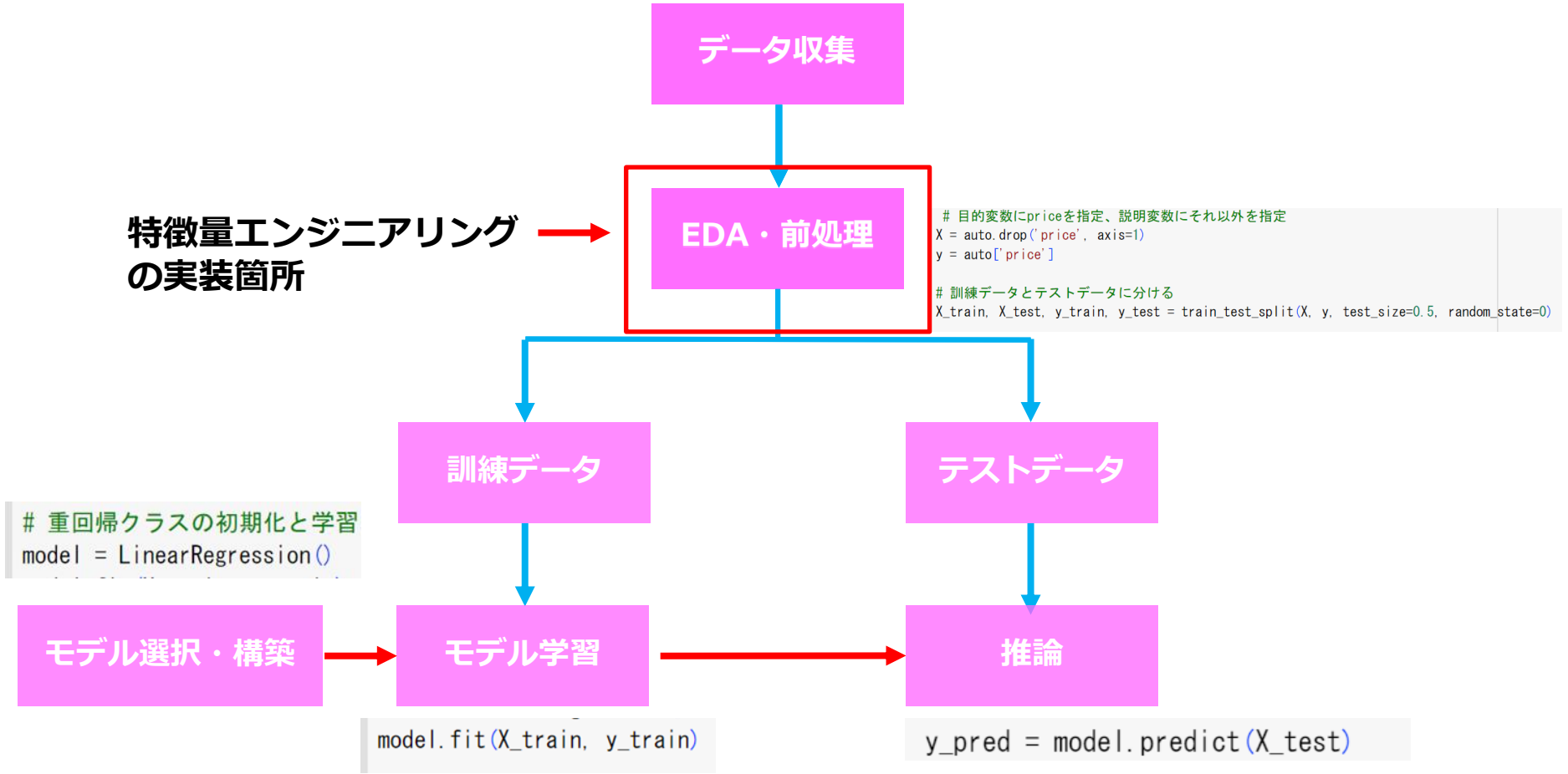
位置づけ

今日はモデル構築部分に位置づけられる特徴量エンジニアリングを扱います



GCI 2023 Summer week1より引用

教師あり学習の流れ(week5引用)



今週の目標

- 特徴量エンジニアリングの概要を理解できる
- 代表的な特徴量エンジニアリングの手法を行うことができる
  〇数値変数の扱い
  〇カテゴリカル変数の扱い
  〇時系列データの扱い

特徴量エンジニアリングとは

特徴量エンジニアリング：
生データからモデルが扱いやすい形の特徴量(説明変数)を生成すること

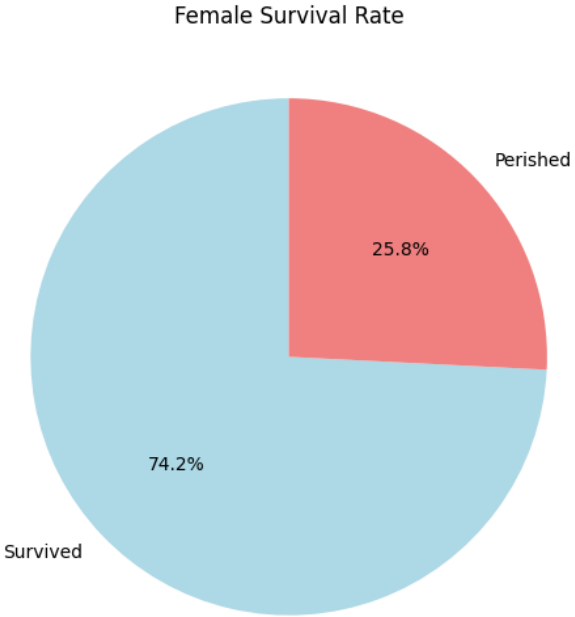
※この手法が必ず効く、というものはないので試行錯誤が必要

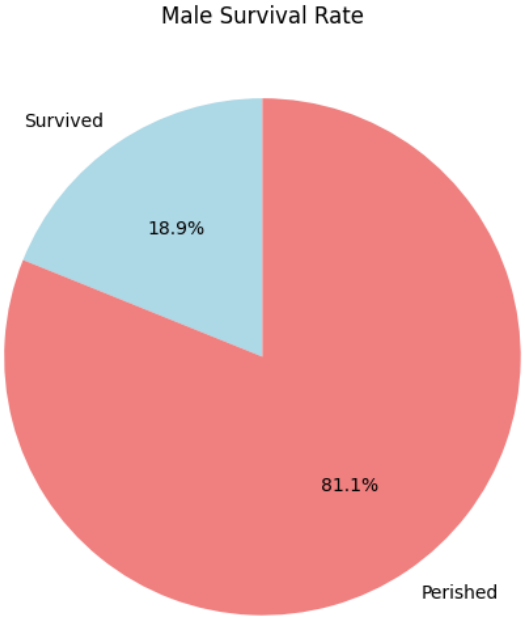
特徴量エンジニアリングの意義
- モデルのパフォーマンス向上
- モデルの解釈性の向上
- データの理解を深める
- 過学習の防止

など

特徴量エンジニアリングの有効性:Titanicコンペの例

Titanicコンペの場合、性別が生死判定に大きく寄与していそう
この特徴量をどう使うかで精度に影響するか？





特徴量エンジニアリングの有効性:Titanicコンペの例

beginner：カテゴリカル変数である性別を削除している
(そのままでは使えないから)

beginner.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ 最終編集: 6月21日

コメント 共有

+ コード + テキスト 接続

```
[ ] print('訓練データのデータ数は{}、変数は{}種類です。'.format(df.shape[0], df.shape[1]))
    print('テストデータのデータ数は{}、変数は{}種類です'.format(df_test.shape[0], df_test.shape[1]))
```

訓練データのデータ数は891、変数は12種類です。
テストデータのデータ数は418、変数は11種類です

訓練データの初めの10データを見てみましょう。

```
df.head(10)
```

	PassengerId	Perished	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	1	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	1	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	1	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	1	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	0	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	0	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

次に**カテゴリカルデータ**を機械学習モデルで扱えるよう処理します。カテゴリカルデータには、**Name**, **Sex**, **Ticket**, **Embarked**がありました。　ここでも、カテゴリカルデータである変数を削除してしまいましょう。カテゴリカルデータのより良い取り扱い方については、**professional.ipynb**をご参照ください。

```
[ ] category_list = ['Name', 'Sex', 'Ticket']

    df.drop(category_list, axis=1, inplace=True)
    df_test.drop(category_list, axis=1, inplace=True)
```

特徴量エンジニアリングの有効性:Titanicコンペの例

professional :
性別を0,1に変換してモデルで扱えるようにしている

次にSex（性別）です。このような二値のカテゴリカル変数は、一方を0、もう一方を1とすることで数値化することができます。ここでは男性を0、女性を1としておきます。

```
df.replace({'Sex': {'male': 0, 'female': 1}}, inplace=True)
df_test.replace({'Sex': {'male': 0, 'female': 1}}, inplace=True)

df.head()
```

	PassengerId	Perished	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	1	3	0	22.0	1	0	7.2500	S
1	2	0	1	1	38.0	1	0	71.2833	C
2	3	0	3	1	26.0	0	0	7.9250	S
3	4	0	1	1	35.0	1	0	53.1000	S
4	5	1	3	0	35.0	0	0	8.0500	S

特徴量エンジニアリングの有効性:Titanicコンペの例

この性別という特徴量を使うかどうかの差分で、精度の違いを検証

モデル：ランダムフォレスト

```
df_pro=df.copy()
df_test_pro=df_test.copy()
```

```
#欠損値のあるカラムをどちらも同様に落とす
missing_list = ['Age', 'Fare', 'Cabin', 'Embarked']
df.drop(missing_list, axis=1, inplace=True)
df_test.drop(missing_list, axis=1, inplace=True)

df_pro.drop(missing_list, axis=1, inplace=True)
df_test_pro.drop(missing_list, axis=1, inplace=True)

#対照群は性別含めて落とす
category_list = ['Name', 'Sex', 'Ticket']
df.drop(category_list, axis=1, inplace=True)
df_test.drop(category_list, axis=1, inplace=True)

#実験群は性別は残す
category_list_pro = ['Name', 'Ticket']
df_pro.drop(category_list_pro, axis=1, inplace=True)
df_test_pro.drop(category_list_pro, axis=1, inplace=True)
```

```
df_pro.replace({'Sex': {'male': 0, 'female': 1}}, inplace=True)
df_test_pro.replace({'Sex': {'male': 0, 'female': 1}}, inplace=True)

df_pro.head()
```

特徴量エンジニアリングの有効性:Titanicコンペの例

性別を特徴量に加えただけで正解率が大きく向上した

右：学習コード部分
左：正解率比較部分の拡大

```
rfc.fit(X_train, y_train)
print('Train Score: {}'.format(round(rfc.score(X_train, y_train), 3)))
print(' Test Score: {}'.format(round(rfc.score(X_valid, y_valid), 3)))
```

```
Train Score: 0.717
 Test Score: 0.709
```

beginnerのもの(性別の特徴量を削除)

```
rfc.fit(X_train_pro, y_train_pro)
print('Train Score pro: {}'.format(round(rfc.score(X_train_pro, y_train_pro), 3)))
print(' Test Score pro: {}'.format(round(rfc.score(X_valid_pro, y_valid_pro), 3)))
```

```
Train Score pro: 0.823
 Test Score pro: 0.784
```

性別を特徴量として利用したもの

```
X = df.iloc[:, 2:].values
y = df.iloc[:, 1].values

X_test = df_test.iloc[:, 1:].values

X_pro = df_pro.iloc[:, 2:].values
y_pro = df_pro.iloc[:, 1].values

X_test_pro = df_test_pro.iloc[:, 1:].values
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.3, random_state=42)
X_train_pro, X_valid_pro, y_train_pro, y_valid_pro = train_test_split(X_pro, y_pro, test_size=0.3, random_state=42)
```

```
rfc = RandomForestClassifier(max_depth=10, min_samples_leaf=1, n_estimators=100, n_jobs=-1, random_state=42)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=10, n_jobs=-1, random_state=42)
```

```
rfc.fit(X_train, y_train)
print('Train Score: {}'.format(round(rfc.score(X_train, y_train), 3)))
print(' Test Score: {}'.format(round(rfc.score(X_valid, y_valid), 3)))
```

```
Train Score: 0.717
 Test Score: 0.709
```

```
rfc.fit(X_train_pro, y_train_pro)
print('Train Score pro: {}'.format(round(rfc.score(X_train_pro, y_train_pro), 3)))
print(' Test Score pro: {}'.format(round(rfc.score(X_valid_pro, y_valid_pro), 3)))
```

```
Train Score pro: 0.823
 Test Score pro: 0.784
```

今日学ぶ概略

特徴量変換　異なるスケールの特徴量を統一し、精度を向上させたりする

特徴量生成　モデルがより効果的に学習できるような特徴量を作成する

特徴量抽出　元のデータの情報を圧縮して少数の重要な特徴量にまとめる

特徴量選択　モデルの性能向上や過学習の防止を目的として、不要な特徴量を除去する

分類法は書籍や人によってまちまちなので、ざっくりと中身を把握してください

今日やGCIで既に学んでいる特徴量エンジニアリング

特徴量変換

- 数値データのスケーリング 9.2.1 (week5 5.3.4)
- カテゴリー変数のエンコーディング 9.3
- 時系列データのエンコーディング 9.4.2

特徴量生成

- 欠損値の補完 week3 3.5.1
- 統計量の使用(groupbyなど) week3 3.4.9
- 交差項の作成 9.2.3
- ドメイン知識などの活用 (9.3.5), スライド
- 時系列データにおけるラグ特徴量 9.4.1

特徴量抽出

- 次元削減9.5.1

特徴量選択

- 特徴選択 9.5.2

GCIで学ぶ以外にも様々な特徴量エンジニアリングの手法があります

今日やGCIで既に学んでいる特徴量エンジニアリング

特徴量変換

- **数値データのスケーリング 9.2.1 (week5 5.3.4)**
- カテゴリー変数のエンコーディング 9.3
- 時系列データのエンコーディング 9.4.2

特徴量生成

- **欠損値の補完 week3 3.5.1**
- 統計量の使用(groupbyなど) week3 3.4.9
- **交差項の作成 9.2.3**
- ドメイン知識などの活用 (9.3.5), スライド
- 時系列データにおけるラグ特徴量 9.4.1

特徴量抽出

- 次元削減9.5.1

特徴量選択

- 特徴選択 9.5.2

欠損値の扱い(具体手法はweek3確認)

基本的に欠損値のままではモデルは扱えない。
欠損値自体に意味がある可能性があり、適切に処理する必要がある。

欠損値の扱いには以下のような手法がある
- 代表値で埋める
- 欠損値とわかるような値(例えば-999など)で埋める
- 予測する
- 欠損値を用いて新たな特徴量を生成する

など

```
# trainの欠損値を確認
train.isnull().sum()
```

```
NAME_CONTRACT_TYPE          0
AMT_INCOME_TOTAL            0
EXT_SOURCE_2              369
OWN_CAR_AGE            112992
ORGANIZATION_TYPE           0
TARGET                      0
dtype: int64
```

コンペ2のtutorial.ipynbより引用

変数の型について

大きく分けると数値変数とカテゴリカル変数がある(ここは最低限抑える)

	種類	説明	例
量的データ(数値変数)	比例尺度	量的な差と絶対ゼロ点を有し、比較が可能な尺度	体重(50kg, 75kg)、距離(5km, 10km)
	間隔尺度	量的な差はあるが絶対ゼロ点を持たない尺度(0に意味がない)	温度（摂氏20度, 摂氏30度）、年（西暦2020年, 西暦2023年）
質的データ(カテゴリカル変数)	順序尺度	順序関係はあるが、間隔が一定でない尺度	成績評価(A, B, C)、映画の評価(★, ★★, ★★★)
	名義尺度	単に分類するための尺度で、数量的な意味は持たない尺度	血液型（A型, O型）、国籍（日本, 米国）

数値変換について

標準化・正規化(week5なども参照)

スケーリング：異なる特徴量間のスケールを均一にすること

スケールが大きく異なる特徴量を同時に扱うと、スケールが大きい特徴量が結果に過度に影響を及ぼす可能性がある

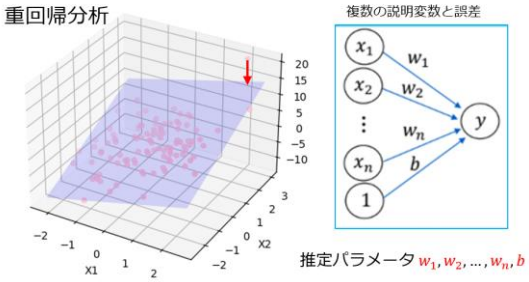
- 標準化(Standardization)：平均が0・標準偏差が1になるように変換

$$x_{std} = \frac{x - \mu_x}{\sigma_x}$$

- 正規化(Min-Max normalization)：最小値を0、最大値を1になるように変換

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

例えば重回帰の場合、回帰係数に影響する



最小二乗法で残差を最小化するような**平面**を求める

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + b$$

notebookへ

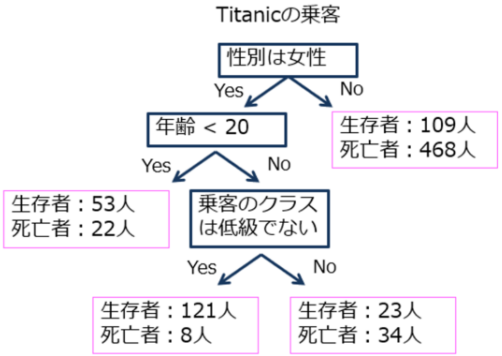
なぜ決定木系のモデルはスケーリングが効かないか？

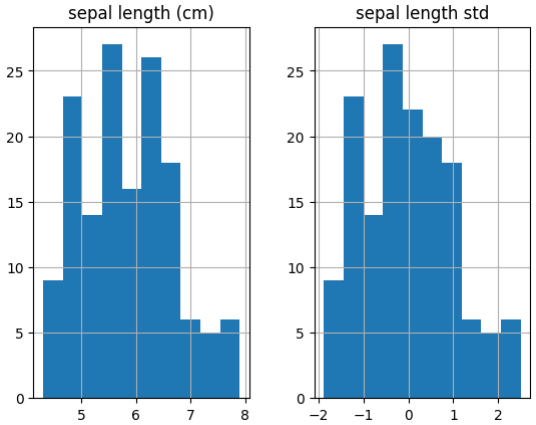
決定木は数字の大小関係しか見ていないから。
(スケーリングでは大小関係自体は変わらない)

決定木

- モデル概要：ノード・枝・葉からなる木構造でデータの分類を行う
- 用途：回帰・分類
- 分類の仕方がわかりやすく説明性が高い
- 過学習になりやすい

Titanicでの分類(例)
① 生存・死亡を分ける上で重要な特徴量は性別である
② 女の子は生き残りやすい
③ 大人の女性の場合、乗客のクラスが高いと生き残りやすい
などの情報が視覚的に分かる





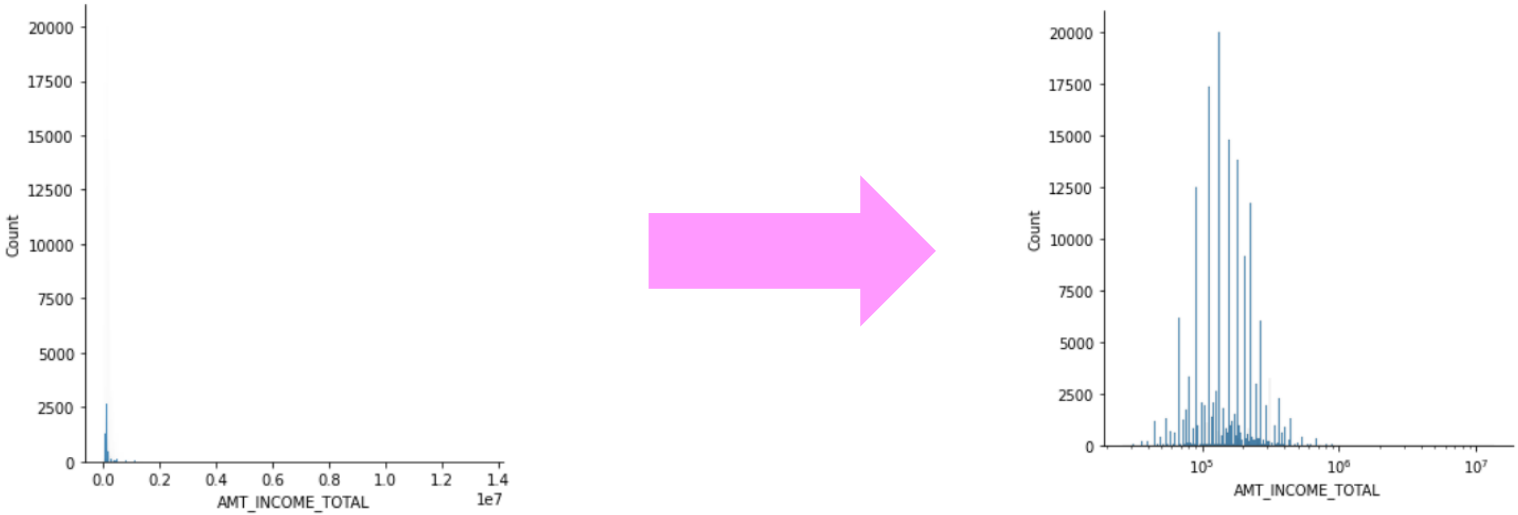
GCI 2023 Summer week5より引用

対数変換

対数変換：データセット内の各値に対して対数関数を適用すること
歪んだデータ分布をより正規分布に近づけることができる

一部のモデルは正規分布を仮定しているため、正規分布に近いようにすると精度が向上する可能性がある

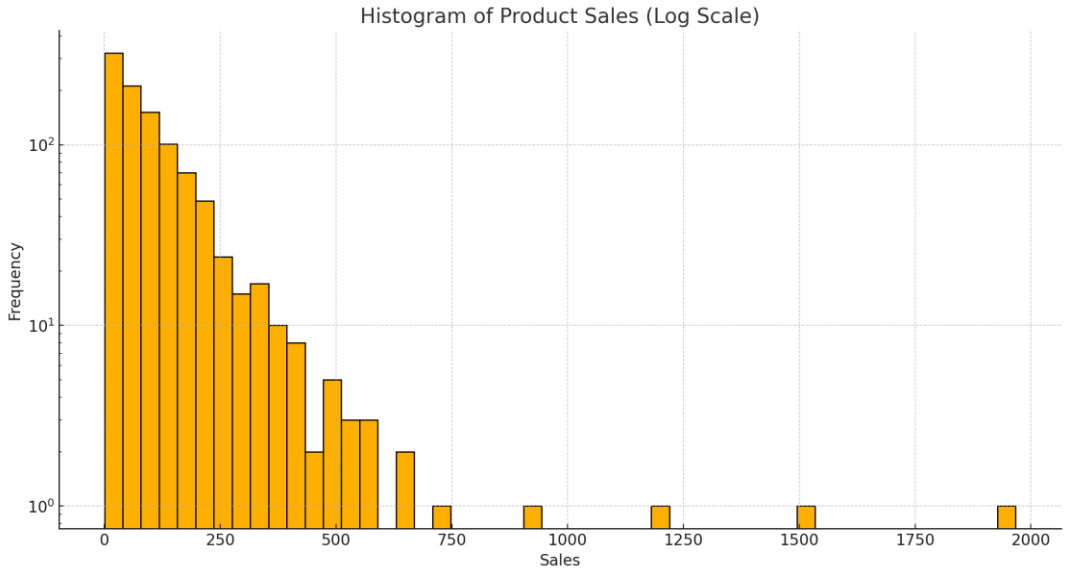
- 非常に広い範囲の値を持つデータや右に歪んだ分布(ロングテール)のデータなどに有効



ロングテールなデータ

ロングテールなデータ：少数のアイテムが多数の売上やアクセスを占める一方で、大多数のアイテムが少数の売上やアクセスを持つデータ

例：Eコマースサイトの商品売上など



交差項の作成

交差項：二つ以上の特徴量の積として表される新しい特徴量
scikit-learnのPolynomialFeaturesにて作成可能

$$(x_1, x_2, x_3) \rightarrow (x_1 x_2, x_2 x_3, x_3 x_1)$$

```
# ライブラリの読み込み
from sklearn.preprocessing import PolynomialFeatures

# 交差項作成器の作成
pf = PolynomialFeatures(degree=2, include_bias=False, interaction_only=True)

# 交差項の作成
X_pf = pf.fit_transform(X)
```

- 生成される特徴量の数が多くなる可能性があるため、メモリ使用量や計算時間、モデルの過学習に注意

notebookへ

カテゴリカル変換のエンコーディング

今日やGCIで既に学んでいる特徴量エンジニアリング

特徴量変換

- 数値データのスケーリング 9.2.1 (week5 5.3.4)
- **カテゴリー変数のエンコーディング 9.3**
- 時系列データのエンコーディング 9.4.2

特徴量生成

- 欠損値の補完 week3 3.5.1
- 統計量の使用(groupbyなど) week3 3.4.9
- 交差項の作成 9.2.3
- **ドメイン知識などの活用 (9.3.5), スライド**
- 時系列データにおけるラグ特徴量 9.4.1

特徴量抽出

- 次元削減9.5.1

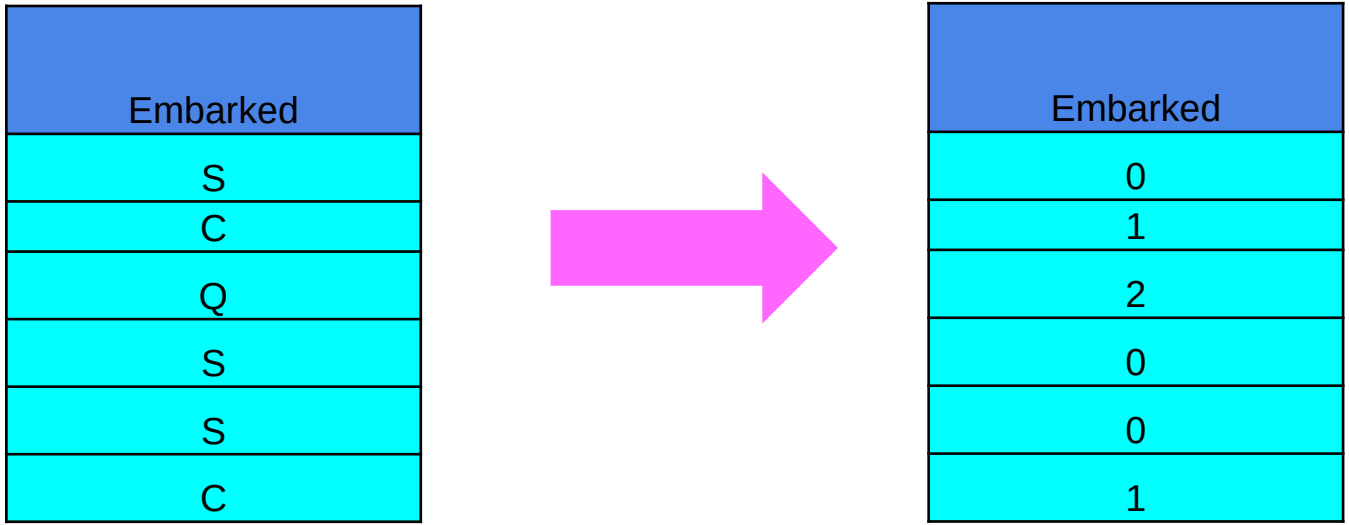
特徴量選択

- 特徴選択 9.5.2

Label Encoding

各カテゴリに整数を割り当てるエンコーディング方法

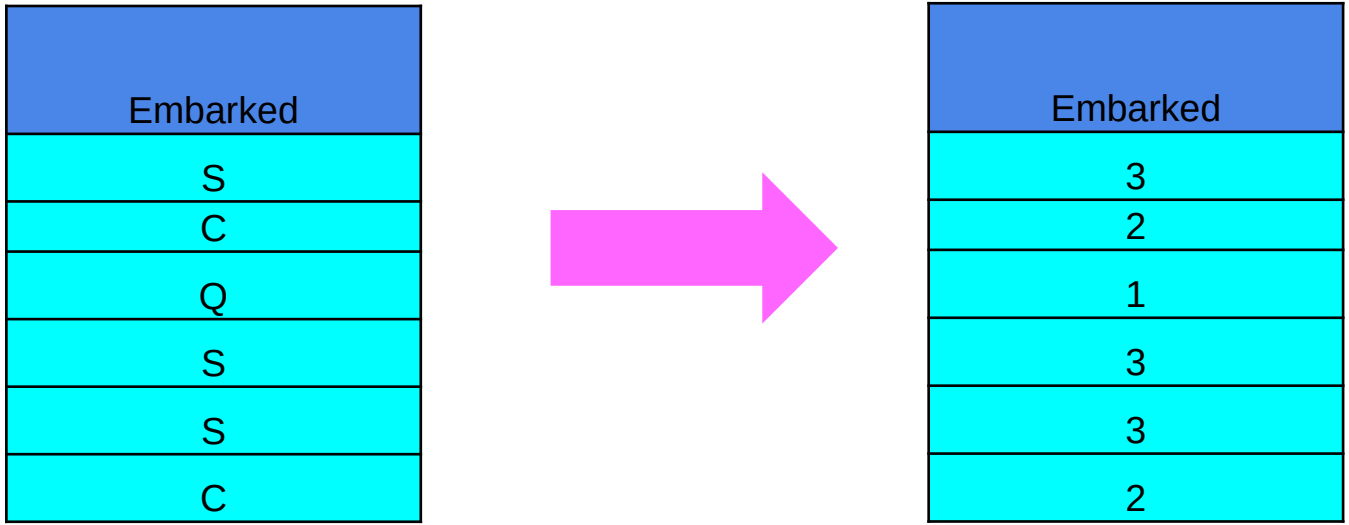
- メリット: カテゴリカル変数をモデルが扱いやすい数値形式に変換できる
- デメリット: 整数の大小関係がデータにない意味を導入し、モデルが誤解する可能性がある



Count Encoding

各カテゴリの出現頻度に基づいて数値を割り当てるエンコーディング方法

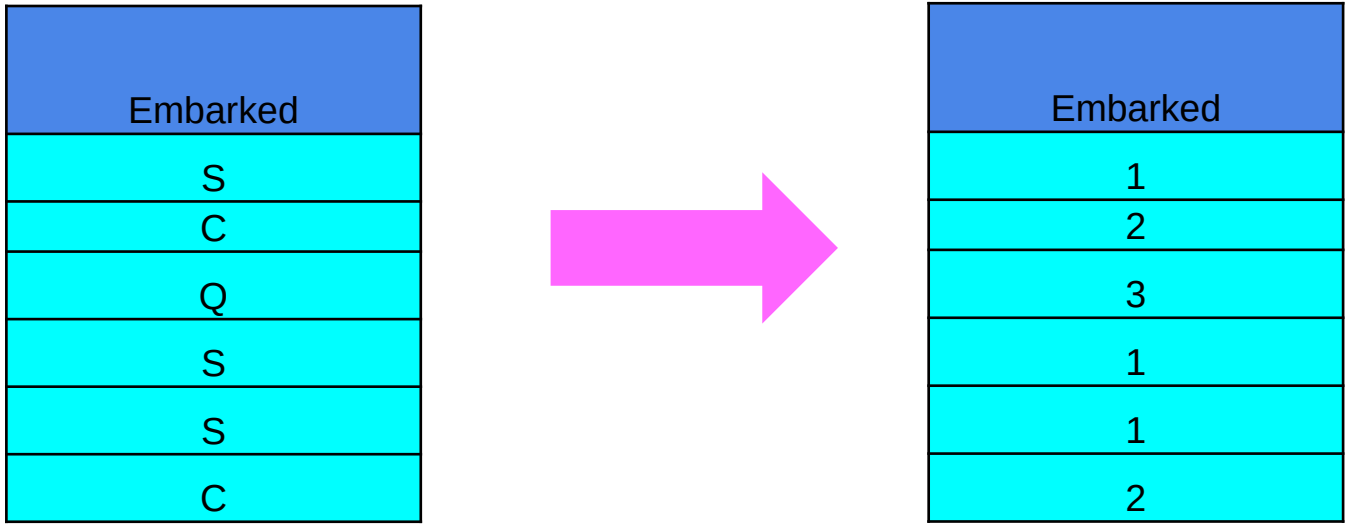
- メリット: 頻度情報を活用してカテゴリの重要性をモデルに伝えることができる
- デメリット: 異なるカテゴリが同じ出現回数を持つ場合、情報の損失が生じる



Label-Count Encoding

カテゴリをラベルエンコーディングした後、それらの出現頻度に基づいて順位付けするエンコーディング方法

- メリット: ラベルと頻度の情報を組み合わせて、より豊かな表現が可能
- デメリット: 頻度が同じカテゴリ間での区別が難しく、情報の曖昧さが生じる(回避は可能)

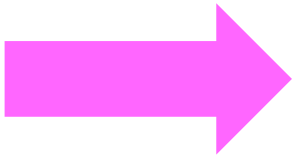


One-Hot Encoding

各カテゴリを独立した列として表現し、該当するカテゴリのみに1を割り当てるエンコーディング方法

- メリット: カテゴリ間の数値的な関係を排除し、モデルに明確なカテゴリ情報を提供する
- デメリット: 多くのカテゴリを持つ変数では次元の爆発を引き起こし、メモリ消費が大きくなる

Embarked
S
C
Q
S
S
C



S	C	Q
1	0	0
0	1	0
0	0	1
1	0	0
1	0	0
0	1	0

notebookへ

時間変数の扱いについて

今日やGCIで既に学んでいる特徴量エンジニアリング

特徴量変換

- 数値データのスケーリング 9.2.1 (week5 5.3.4)
- カテゴリー変数のエンコーディング 9.3
- **時系列データのエンコーディング 9.4.2**

特徴量生成

- 欠損値の補完 week3 3.5.1
- 統計量の使用(groupbyなど) week3 3.4.9
- 交差項の作成 9.2.3
- ドメイン知識などの活用 (9.3.5), スライド
- **時系列データにおけるラグ特徴量 9.4.1**

特徴量抽出

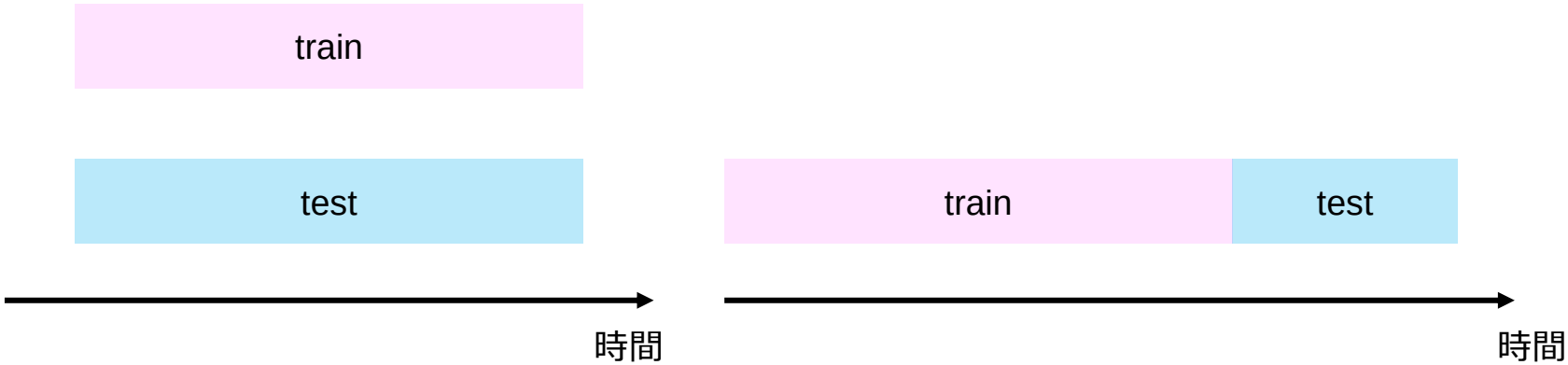
- 次元削減9.5.1

特徴量選択

- 特徴選択 9.5.2

時系列情報の扱い方

基本的に時系列データを用いる場合、未来予測の場合が多い(右)
その場合、学習データに未来のデータが入らないように気を付ける必要がある(リークになる)



時系列データの処理

モデルが入力に使いやすい形

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
25560	25561	2012	12	1	0	41.0	-16	-8.0	1035.0	NW	1.79	0	0
25561	25562	2012	12	1	1	46.0	-17	-7.0	1035.0	cv	0.89	0	0
25562	25563	2012	12	1	2	37.0	-15	-7.0	1036.0	cv	1.34	0	0
25563	25564	2012	12	1	3	48.0	-15	-9.0	1035.0	NE	0.89	0	0
25564	25565	2012	12	1	4	43.0	-14	-9.0	1035.0	NE	1.78	0	0

9.4.1

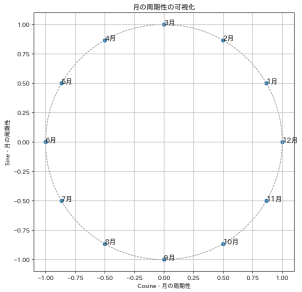
特徴量を作ったり可視化に向く形(時系列変化を追いやすい)

	pm2.5
2012-12-01 00:00:00	41.0
2012-12-01 01:00:00	46.0
2012-12-01 02:00:00	37.0
2012-12-01 03:00:00	48.0
2012-12-01 04:00:00	43.0

特徴量生成

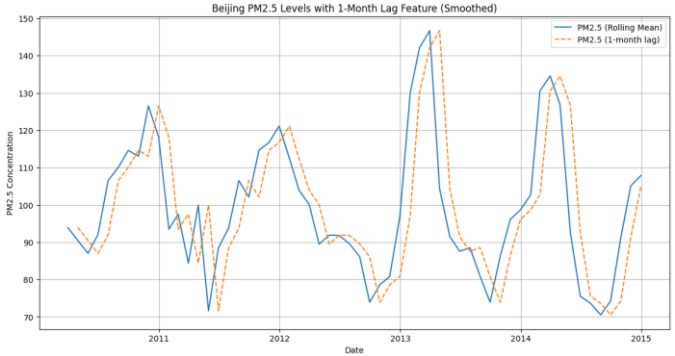
	pm2.5
2012-12-01	NaN
2012-12-02	2.630252
2012-12-03	0.127796
2012-12-04	0.975000
2012-12-05	0.743590

周期性を考慮したエンコーディング



9.4.2

可視化(week4)



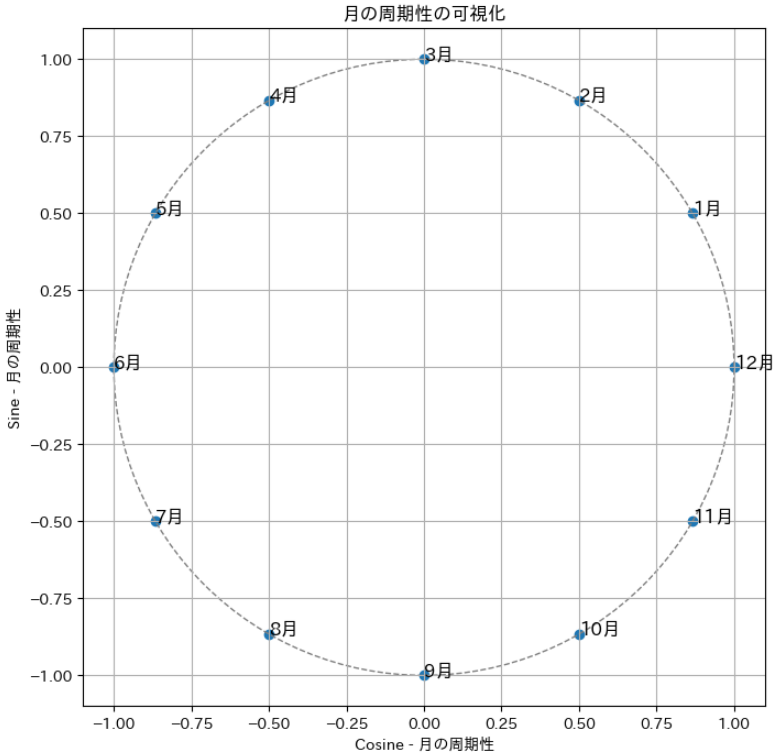
notebookへ

時間変数のエンコーディング　周期性を考慮するケース

月や日のような周期的な要素は、単に数値として扱うよりもその周期性を反映させた方がモデルにとって有意義な情報になりえる

→例として三角関数の利用

- 周期性の保存: 時間の周期性を適切にモデルに反映させることができる
- 距離の維持: このエンコーディングにより、例えば1月と12月のような時間的に近いが数値的には離れた期間も適切に近いと認識できる
- 特徴量の拡張: 元の1次元の時間変数を2次元に拡張することで、モデルが時間データのパターンをより複雑に捉えることができる



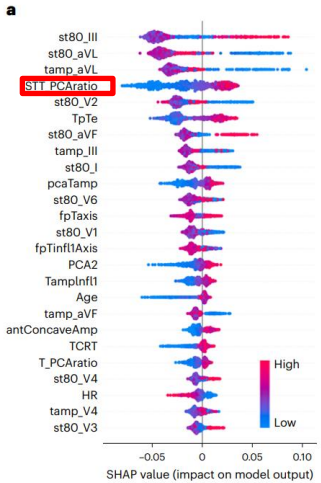
notebookへ

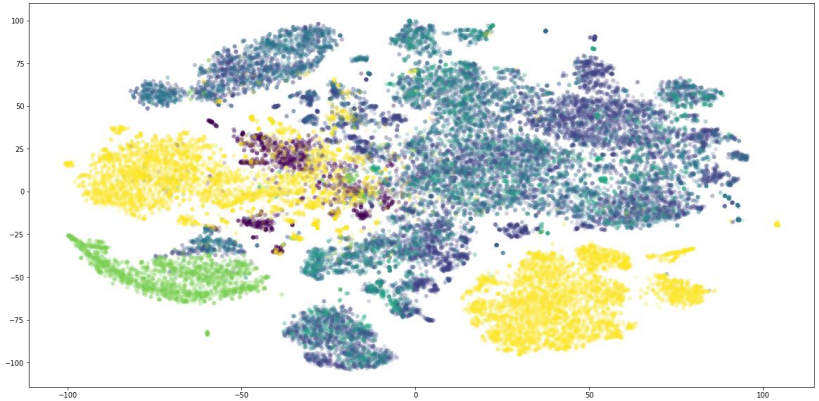
講義資料にない手法の紹介

次元削減

講義では特徴量を減らす手法としてPCAを使ったが、PCAなど、次元削減した情報を特徴量とすることで精度向上する場合もある

- 心電図の波形データから特徴量生成してPCAの特徴量を追加した論文
  (Machine learning for ECG diagnosis and risk stratification of occlusion myocardial infarction)
- KaggleのOtto Group Product Classification Challengeコンペ
  (https://www.kaggle.com/code/zmey56/using-t-sne-and-pca-for-otto-group-product)



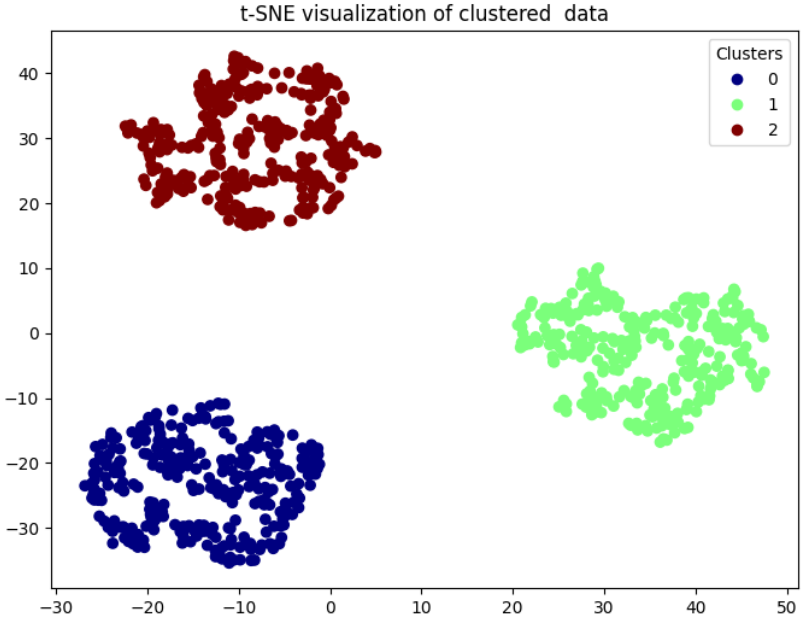


https://www.nature.com/articles/s41591-023-02396-3より引用

https://www.kaggle.com/code/zmey56/using-t-sne-and-pca-for-otto-group-productより引用

クラスタリング

k-meansなどでクラスタリングしたクラスター番号を特徴量にすることや、クラスターの中心からの距離を特徴量に使うなどの手法がある



- KaggleのAllstate Claims Severityコンペ
  (https://www.kaggle.com/competitions/allstate-claims-severity/discussion/26427より引用)

特徴量の組み合わせ

交差項で見たように、特徴量同士の組み合わせによる新しい特徴量の作成が有効なケースがある

- 意味のある特徴量を作りデータの解釈性を上げることができる場合もある

例：

〇年間返済額と借入金額から返済負担率を計算(年間返済額 / 借入金額)
〇総人口と地域の面積からその地域の人口密度を計算
〇Home Credit Default Riskコンペ1位の手法より

credit_goods_price_ratio: AMT_CREDIT / AMT_GOODS_PRICE

last_active_DAYS_CREDIT: From the active loans in bureau, the most recent DAYS CREDIT value, grouped by SK_ID_CURR.

credit_downpayment: AMT_GOOD_PRICE - AMT_CREDIT

https://www.kaggle.com/competitions/home-credit-default-risk/discussion/64821より引用

ドメイン知識を活かす

課題のドメイン知識を活かすことで、精度向上や解釈性の向上につながるケースもある

例：

- LTV＝平均顧客単価×収益率×購買頻度×継続期間
- 前述のMachine learning for ECG diagnosis and risk stratification of occlusion myocardial infarctionでは心電図波形に関して医師のドメイン知識組み込んで特徴量生成を行っている
- KaggleのRSNA STR Pulmonary Embolism Detection(画像コンペ)では学習データを眺めて肺にフォーカスする必要に気づき、それを実行している(https://www.kaggle.com/competitions/rsna-str-pulmonary-embolism-detection/discussion/194145)
- KaggleのPLAsTiCC Astronomical Classification(天体の多クラス分類コンペ)では天文学のドメイン知識が有効だったらしい

参考資料：講義から学びを深めたい方にもおすすめです

- Kaggleで勝つデータ分析の技術　門脇 大輔 (著), 阪田 隆司 (著), 保坂 桂佑 (著), 平松 雄司 (著)