

Dataの前処理

Titanic - Machine Learning from Disaster | Kaggle



第1章: データ前処理の概要

データ前処理とは

今あるデータを一度分析が正しくできるように整形する作業

前処理の目的

データの一貫性確保

データの品質向上

解析準備

パフォーマンスの向上

分析における前処理の位置づけ

効果的な前処理は、分析結果の信頼性を高め、ビジネスインサイトの発見に大きく貢献。

第2章: pandasの導入

pandasとは

データ分析に特化したライブラリ。表計算の分析に使われる。

pandasの基本操作

- CSVファイルやExcelデータの読み込み
- 行数や列名からのデータの取得
- データの特徴や形状あるいは一部の表示
- データの要約や統計量の表示
- データの変形やピボット
- データのフィルタリングや切り出し(スライス)
- 値やインデックスごとの並べ替え
- 列や表の集計



第3章: タイタニック データセットの前処理

基本的なデータクリーニング

重複値の削除、不要な列の削除、基本的な欠損値の処理。

データの探索と評価

データセットの構造を理解、問題点や欠損値の特定。

データsetの読み込み

タイタニックデータ(CVS file)を読み込む。



第4章: タイタニックデータの詳細な前処理

-欠損値の扱い

平均値・中央値補完: 連続変数に対して外れ値が少ない場合に有効。

最頻値補完: カテゴリカル変数に対して。

固定値補完: 特定のカテゴリまたは識別子を表す場合。

ホールドアウト法: 時系列データやインデックスが順序を持つ場合に有効。

```
# Age列の欠損値を中央値で補完
titanic_df['Age'].fillna(titanic_df['Age'].median(), inplace=True)
```

```
# Embarked列の欠損値を最頻値で補完
titanic_df['Embarked'].fillna(titanic_df['Embarked'].mode().iloc[0], inplace=True)
```

```
# Cabin列の欠損値を 'Unknown' で補完
titanic_df['Cabin'].fillna('Unknown', inplace=True)
```

```
# 振り返り: 欠損値の数を確認
print(titanic_df.isnull().sum())
```

第4章: タイタニックデータの詳細な前処理

-エンコーディングの方法

ラベルエンコーディング

- 方法: カテゴリ変数を数値に変換します。例えば、"赤"、"青"、"緑"を 0, 1, 2 のように数値に変換します。
- メリット: シンプルで計算コストが低い。データの次元数を増やしません。
- デメリット: 数値の大小関係に意味がない場合、モデルが誤った解釈をしてしまう可能性があります。（例: "赤" < "青" と解釈される可能性）
- 適用例: 順序関係があるカテゴリデータ（例: 低・中・高、S・M・L）に適しています。

ワンホットエンコーディング

- 方法: カテゴリ変数の各値に対応するダミー変数を作成し、該当するダミー変数のみを1、それ以外を0とします。
- メリット: 数値の大小関係に影響されません。
- デメリット: カテゴリ変数の値が多い場合、データの次元数が大幅に増加する可能性があり、計算コストが高くなる可能性があります。
- 適用例: 順序関係がないカテゴリデータ（例: 色、国）に適しています。

第4章: タイタニックデータの詳細な前処理-

新たな特徴量の作成 - 既存のデータから新しい情報を引き出す方法

- 家族の人数

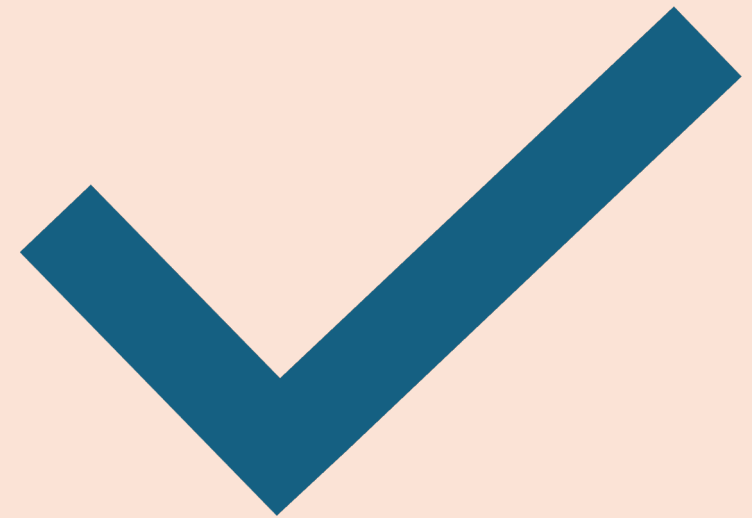
'SibSp' (兄弟姉妹/配偶者の数) と
'Parch' (両親/子供の数) を足して、家族の
人数を計算


```
titanic_df['FamilySize'] =  
titanic_df['SibSp'] + titanic_df['Parch']  
+ 1 # +1 は本人を含む
```

```
print(titanic_df[['FamilySize', 'SibSp',  
'Parch']].head())
```

- 船室の有無

- 乗船港のカテゴリー数





第4章: タイタニックデータの 詳細な前処理-補完の前後の データ確認

- 補完処理を行った後は、必ずデータの確認を行い、適切に欠損値が処理されたかを確認します。

```
# 振り返り: 欠損値の数を確認
print(titanic_df.isnull().sum())
# 補完後の統計情報を表示
print(titanic_df.describe())
# データが適切に補完されたことを確認
print(titanic_df.info())
```


第5章： 質問

Q1. プラグインとは？（モジュールのインストールの際）

プラグインは、ソフトウェア本体の機能を拡張するための追加プログラムです ✨

スマホに好きなアプリを入れて機能を追加していくようなイメージ 📱

プラグインを使う際の注意点

- 信頼できる提供元から入手する
- 最新版を維持する
- 互換性を確認する
- 不要なプラグインは削除する （どうやって操作するの？）

Q2. Pullリクエストとメインブランチにマージする。

どこの段階で他の人と共有できるのか？

Q3. ワンホットエンコーディング方法:

カテゴリ変数の各値に対応するダミー変数を作成し、該当するダミー変数のみを1、それ以外を0とします？

Q4. GitHubにターミナルを使ってプッシュできなかった。

git add:

<filename>: ステージングエリアに追加するファイルを指定。

∴ 現在のディレクトリからすべての変更をステージングエリアに追加。

git commit:

-m "Your commit message": コミットメッセージを指定してコミットを作成。

git push:

origin <branch-name>: リモートリポジトリの origin と指定したブランチ <branch-name> にコミットをプッシュ。

Pandas キーcode集

データ構造

- **Series:** 1次元のラベル付き配列。
 - `pd.Series()`, `index`, `values`
- **DataFrame:** 2次元のラベル付き表形式データ。Excelシートのような構造。
 - `pd.DataFrame()`, `columns`, `index`
- **Panel:** 3次元以上のデータ構造。
 - `pd.Panel()`, `items`, `major_axis`, `minor_axis`
- **Series:** 1次元のラベル付き配列。データ分析の基本単位。

データ入出力

- **読み込み:** 様々な形式のデータを読み込む。
 - `pd.read_csv()`, `pd.read_excel()`, `pd.read_sql()`, `pd.read_json()`, `header`, `sep`
- **書き出し:** 様々な形式でデータを保存
 - `df.to_csv()`, `df.to_excel()`, `df.to_sql()`, `df.to_json()`, `index=False`

データ操作

- **選択:** 行、列、特定の要素を選択
 - `loc`, `iloc`, `[], query()`, `isin()`
- **フィルタリング:** 条件に合致するデータの抽出
 - `[], query()`, `isin()`, `str.contains()`, `~`
- **ソート:** 特定の列の値に基づいてデータを並べ替え
 - `sort_values()`, `ascending`

データ操作

- **結合:** 複数のDataFrameを結合
 - `merge()`, `concat()`, `join()`, `how`, `on`
- **グループ化:** 特定の列の値でデータをグループ分けし、集計や分析を行う
 - `groupby()`, `agg()`, `transform()`, `apply()`
- **欠損値処理:**
 - `isnull()`, `notnull()`, `dropna()`, `fillna()`, `method`
- **重複処理:**
 - `duplicated()`, `drop_duplicates()`

データ集計・統計

- **記述統計量:** データの全体像を把握
 - `describe()`, `mean()`, `median()`, `std()`, `min()`, `max()`, `quantile()`
- **相関:** 変数間の関係性を把握
 - `corr()`, `method`
- **クロス集計表:** カテゴリデータの集計
 - `crosstab()`, `margins`

その他

- **データの可視化:** matplotlibと連携してデータのグラフ化が可能
 - `plot`, `hist`, `scatter`
- **日付データ処理:** 日付データの変換や抽出
 - `pd.to_datetime()`, `dt.year`, `dt.month`, `dt.day`, `dt.weekday`
- **文字列処理:** 文字列の検索、置換、分割など
 - `str.contains()`, `str.replace()`, `str.split()`, `str.upper()`, `str.lower()`

参考文献

インターネットアカデミー Pythonのscikit-learnの「前処理」とは <https://python-study.org/archives/216>

前処理説明：<https://drive.google.com/file/d/1-0-6SI--TMolrj0Eh9VLUSX78x01uvkV/view?usp=sharing>

機械学習チュートリアル① - はじめに ～ pandas入門：
<https://zenn.dev/nishimoto/articles/5c215b00edbb5b>

Pandas入門：[https://shinonome.io/PythonTutorialForDSCourse/ja-edited/11 Introduction to Pandas.html](https://shinonome.io/PythonTutorialForDSCourse/ja-edited/11%20Introduction%20to%20Pandas.html)

Pythonプログラミング入門 | 7-1. pandasライブラリ <https://utokyo-ipp.github.io/7/7-1.html>