DS Lv.1-5

データサイエンス講座

Level 01-5

第５章：統計学と確率論の基礎

mjin.space

第５章：統計学と確率論の基礎

統計学と確率論

重要性とメリット

統計学と確率論は、データの特徴や傾向を正確に理解し、そこから有意義な結論を導き出すための基盤となる学問です。データサイエンティストにとっては、仮説を検証し、モデルの信頼性を評価し、予測の精度を向上させるために欠かせないスキルであり、データに基づいた意思決定を科学的かつ客観的に行う力を提供します。結果、ビジネスや社会課題の解決において、データの価値を最大化することが可能になります。

統計学と確立４つの分野

DS Lv.1-5

データサイエンティストがデータを正しく理解し、分析を通じて有意義な結果を導き出すための基礎４分野。
それぞれの役割と手法を理解し、状況に応じて適切に使い分けることが重要

記述統計

データの中心傾向（平均、中央値、モード）やばらつき（分散、標準偏差）を計算し、ヒストグラムや箱ひげ図で視覚化することで、全体像を把握する手法。

実例: 月ごとの平均売上やばらつきを把握して売上傾向を分析。

推測統計

サンプルデータをもとに母集団の特性を推測し、t検定やカイ二乗検定で仮説を検証する手法。

実例: 新商品の購入意向を調査し、異なるグループ間での効果を比較。

確率論

ランダムな事象の発生確率を計算し、正規分布や期待値を用いて不確実性を評価する理論。

実例: A/Bテストの結果を基に広告施策の成功確率を予測。

相関分析と回帰分析

変数間の関連性を数値で評価し（相関分析）、単回帰や重回帰モデルで未来の予測を行う手法。

実例: 広告費と売上の関係を分析し、広告の最適な予算を算出。

記述統計の基礎

DS Lv.1-5

平均、中央値、モード

- **平均:** データの合計をデータの個数で割った値で、データの中心傾向を示す代表値
  - *外れ値の影響を受けやすいが、全体の分布の中心を把握するのに適している。*
- **中央値:** データを昇順または降順に並べたときの中央の値
  - *外れ値の影響を受けにくく、偏った分布での中心傾向を示すのに便利*
- **モード（最頻値）:** データ内で最も頻繁に出現する値
  - *カテゴリデータや離散データの分析に適しており、複数存在する場合もある*

5-01.py

```
import pandas as pd
from scipy import stats

# サンプルデータ作成
data = {"Values": [10, 20, 20, 30, 40, 50]}
df = pd.DataFrame(data)

# 平均、中央値、モードの計算
mean = df["Values"].mean()
median = df["Values"].median()
mode = stats.mode(df["Values"])[0][0]

print(f"平均: {mean}, 中央値: {median}, モード: {mode}")
```

記述統計の基礎

分散と標準偏差

- **分散:** データが平均からどれだけ離れているかのばらつきを示す指標（値の平方和の平均）
  - *分布の全体的な広がりを定量的に評価するが、単位が元データと異なる（平方単位）*
- **標準偏差:** 分散の平方根をとった値で、元データと同じ単位でばらつきを表現
  - *データのばらつきを直感的に把握しやすく、正規分布では平均から±1標準偏差内に約68%のデータが収まる*

5-02.py

```
import pandas as pd

# サンプルデータ作成
data = {"Values": [10, 20, 20, 30, 40, 50]}
df = pd.DataFrame(data)

# 分散と標準偏差の計算
variance = df["Values"].var()
std_dev = df["Values"].std()

print(f"分散: {variance}, 標準偏差: {std_dev}")
```

記述統計の基礎

箱ひげ図の読み方

データの分布を視覚的に示し、中央値、四分位範囲（IQR）、外れ値を一目で把握可能。

- 中央の線は中央値を示し、箱の上下は第一四分位数（Q1）と第三四分位数（Q3）
- 箱の長さ（IQR）は分布のばらつきを示し、ひげは通常、Q1 - 1.5×IQRおよびQ3 + 1.5×IQRで計算される
- 箱の外にプロットされる点は外れ値

5-03.py

```
import pandas as pd
import matplotlib.pyplot as plt

# サンプルデータ作成
data = {"Values": [10, 20, 20, 30, 40, 50, 70, 100]}
df = pd.DataFrame(data)

# 箱ひげ図の作成
plt.boxplot(df["Values"])
plt.title("箱ひげ図")
plt.ylabel("値")
plt.show()
```

記述統計の基礎

DS Lv.1-5

分布（正規分布、歪度、尖度）

- 正規分布: 平均を中心とした左右対称の分布で、多くの自然現象に当てはまる
  - *平均、中央値、モードが一致し、分布の形状が釣鐘型になる*
- 歪度: 分布の非対称性を表す指標。正の歪度は右に尾を引く、負の歪度は左に尾を引く分布を意味する
  - *分布のバランスを評価し、偏ったデータの補正が必要か判断する際に使用*
- 尖度: 分布の鋭さ（ピークの高さ）を表す指標。高い尖度は鋭いピークを示し、低い尖度は平坦な分布を示す
  - *分布の形状が正規分布からどれだけ異なるかを評価する*

5-04.py

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 正規分布の生成
data = np.random.normal(loc=50, scale=10, size=1000)

# ヒストグラムとカーネル密度推定
sns.histplot(data, kde=True, bins=30, color="blue")
plt.title("正規分布")
plt.xlabel("値")
plt.ylabel("頻度")
plt.show()
```

記述統計の基礎

その他

- **範囲（Range）**：最大値と最小値の差で、データのばらつきの簡易的な指標
  - *全体の広がりを把握できるが、外れ値の影響を受けやすい*
- **四分位範囲（IQR）**：データを4等分した範囲（Q3 - Q1）で、中央値を中心としたばらつきを示す
  - *外れ値の影響を受けにくい指標として、分布の広がりを評価する際に有効*
- **ヒストグラム**: データの分布を視覚化する棒グラフ。データがどの範囲に集中しているかを把握可能
  - *分布の形状（正規分布、偏った分布など）を直感的に理解できる*

5-05.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# サンプルデータ作成
data = {"Values": [10, 20, 20, 30, 40, 50, 70, 100]}
df = pd.DataFrame(data)

# 1. 範囲（Range）
range_value = df["Values"].max() - df["Values"].min()
print(f"範囲 (Range): {range_value}")

# 2. 四分位範囲（IQR）
q1 = df["Values"].quantile(0.25)
q3 = df["Values"].quantile(0.75)
iqr = q3 - q1
print(f"四分位範囲 (IQR): {iqr}")

# 3. ヒストグラム
plt.hist(df["Values"], bins=5, color="blue", edgecolor="black", alpha=0.7)
plt.title("ヒストグラム")
plt.xlabel("値")
plt.ylabel("頻度")
plt.show()
```

推測統計の基礎

DS Lv.1-5

母集団と標本

- **母集団:** 調査対象となる全てのデータや個体の集合で、調査結果の対象範囲となる
  - *母集団全体の特性を直接調査することは非現実的な場合が多いため、標本を用いる*
- **標本:** 母集団から抽出した一部のデータで、母集団の特性を推測するための基礎となる
  - *標本はランダムに抽出され、母集団を代表するようにすることが重要*

5-06.py

```
import numpy as np
import random

# 母集団データ作成
population = np.random.randint(1, 101, 1000)  # 1〜100のランダムな値で構成された母集団

# 標本抽出
sample = random.sample(list(population), 100)

print(f"母集団平均: {np.mean(population)}, 標本平均: {np.mean(sample)}")
```

推測統計の基礎

仮説検定

仮説が正しいかどうかをデータに基づいて検証する手法。

- **帰無仮説（H0）**: 検証したい仮説の否定（例: 差がない、効果がない）
- **対立仮説（H1）**: 検証したい主張（例: 差がある、効果がある）
- 検定結果によって、帰無仮説を棄却（または採択）する

5-07.py

```
import numpy as np
from scipy.stats import ttest_ind

# 仮説設定
# H0: グループAとグループBの平均に差はない
# H1: グループAとグループBの平均に差がある

# サンプルデータ作成
np.random.seed(42)
group_a = np.random.normal(50, 5, 30)  # グループAのデータ
group_b = np.random.normal(52, 5, 30)  # グループBのデータ

# t検定を実行
t_stat, p_value = ttest_ind(group_a, group_b)

# 結果の出力
print("t値:", t_stat)
print("p値:", p_value)
・・・
```

推測統計の基礎

t検定とカイ二乗検定

- **t検定:** 2つの平均値の差が偶然か有意かを判定する手法
  - 対応のあるt検定: 同じ対象の前後比較（例: ダイエット前後の体重変化）
  - 対応のないt検定: 異なる2つのグループ間の比較（例: 男性と女性の平均スコア）
- **カイ二乗検定:** カテゴリデータの分布が期待値と一致するかを評価する手法
  - 独立性の検定: 2つのカテゴリ変数間の関連性を検証（例: 性別と購入意欲）
  - 適合度の検定: 観測データが理論的分布にどれだけ適合しているかを評価

5-08.py

```
import numpy as np
from scipy.stats import ttest_ind

# グループAとグループBのサンプルデータ
group_a = np.random.normal(50, 5, 30)
group_b = np.random.normal(52, 5, 30)

# t検定
t_stat, p_value = ttest_ind(group_a, group_b)

print(f"t値: {t_stat}, p値: {p_value}")
if p_value < 0.05:
    print("統計的に有意な差があります。")
else:
    print("統計的に有意な差はありません。")

import numpy as np
from scipy.stats import chi2_contingency

# カイ二乗検定用データ
data = np.array([[50, 30], [20, 40]])  # 行：カテゴリ、列：観測データ
chi2, p, dof, expected = chi2_contingency(data)
・・・
```

推測統計の基礎

p値、信頼区間、効果量

- **p値:** 仮説が偶然成立する確率を表す値（通常、0.05以下で有意とされる）
  - *小さいp値は帰無仮説を棄却する根拠となり、有意な結果を示す*
- **信頼区間:** 母集団の真の値が含まれる範囲を推定（例: 平均値の95%信頼区間）
  - *幅が狭いほど推定の精度が高く、サンプルサイズに依存する*
- **効果量:** 実際の差の大きさを定量的に評価する指標（例: Cohen's d）
  - *p値が有意でも効果量が小さい場合、実用的な意味が乏しいことがある*

5-09.py

```
import numpy as np
from scipy.stats import norm

# サンプルデータ
sample = np.random.normal(50, 5, 100)

# 平均と標準誤差
mean = np.mean(sample)
std_err = np.std(sample, ddof=1) / np.sqrt(len(sample))

# 95%信頼区間の計算
ci = norm.interval(0.95, loc=mean, scale=std_err)

print(f"平均: {mean}")
print(f"95%信頼区間: {ci}")
```

推測統計の基礎

その他

- **標準誤差（Standard Error）:** 標本平均が母集団平均からどの程度離れているかの指標
  - *標本サイズが大きいほど標準誤差は小さくなる*
- **Z検定:** 標準正規分布に基づく仮説検定で、大規模なサンプルに適用される
  - *t検定に似ているが、標本数が多い場合に使用される*
- **分散分析（ANOVA）:** 3つ以上のグループ間の平均値の差を検証する手法
  - *多群比較のため、t検定を繰り返す代わりに用いられる*
- **ベイズ推定:** 事前確率と新しいデータを組み合わせて結果を更新する手法
  - *不確実性を含むデータに対して柔軟に適用可能*

5-10.py

```
import numpy as np

# サンプルデータ
sample = np.random.normal(50, 5, 100)

# 標本平均と標準誤差の計算
sample_mean = np.mean(sample)
sample_std = np.std(sample, ddof=1)
standard_error = sample_std / np.sqrt(len(sample))

print(f"標本平均: {sample_mean}")
print(f"標準誤差: {standard_error}")
```

確率論の基礎

DS Lv.1-5

確率

ある事象が発生する可能性を数値で表したもの（0〜1の範囲）。

- 0は「絶対に起こらない」、1は「絶対に起こる」を意味する
- 確率の合計は必ず1になる
- 条件付き確率や独立事象を理解することで、複雑な状況の分析が可能になる

5-11.py

```
import random

# サイコロを振るシミュレーション
outcomes = [1, 2, 3, 4, 5, 6]
probability = 1 / len(outcomes)

print(f"サイコロの各面が出る確率: {probability}")

# 特定の目（例えば4）が出る確率
event = 4
event_probability = outcomes.count(event) / len(outcomes)
print(f"特定の目 ({event}) が出る確率: {event_probability}")
```

確率論の基礎

確率分布

ランダムな変数が取り得る値と、その値が発生する確率を記述したもの。

- **正規分布:** 平均を中心とした左右対称の分布（釣鐘型）。多くの自然現象に適用可能
- **二項分布:** 2つの結果（成功/失敗）を持つ試行の確率分布（例: コインの表裏）
- **ポアソン分布:** 一定期間内の事象発生回数を表す（例: サーバへのアクセス頻度）
- 確率分布を用いて、不確実性を数値的に表現する

5-12.py

```
import numpy as np
import matplotlib.pyplot as plt

# 1. 正規分布
normal_data = np.random.normal(loc=50, scale=10, size=1000)  # 平均50, 標準偏差10
plt.hist(normal_data, bins=30, color="blue", alpha=0.7, edgecolor="black")
plt.title("正規分布")
plt.xlabel("値")
plt.ylabel("頻度")
plt.show()

# 2. 二項分布
binomial_data = np.random.binomial(n=10, p=0.5, size=1000)  # 試行回数10, 成功確率0.5
plt.hist(binomial_data, bins=10, color="green", alpha=0.7, edgecolor="black")
plt.title("二項分布")
plt.xlabel("成功回数")
plt.ylabel("頻度")
plt.show()

. . .
```

確率論の基礎

DS Lv.1-5

期待値と分散

- **期待値:** 確率分布における平均値で、ランダムな変数の長期的な平均的振る舞いを示す
  - *確率の重み付き平均で計算され、中心的傾向を評価するのに重要*
- **分散:** データが期待値からどれだけ離れているかを示す指標
  - *分散が小さいほど、データが期待値周辺に集中している*

5-13.py

```
import numpy as np

# サイコロを振る場合の期待値
outcomes = [1, 2, 3, 4, 5, 6]
probabilities = [1/6] * 6  # 各目が出る確率

# 期待値の計算
expected_value = sum(o * p for o, p in zip(outcomes, probabilities))
print(f"サイコロを振ったときの期待値: {expected_value}")
```

確率論の基礎

ベイズの定理

新しい情報をもとに、既存の確率（事前確率）を更新する手法。

- 条件付き確率を用いて、特定の事象が起こる確率を動的に評価
- 不確実性が高いデータや動的に変化する状況に対応できる
- 実例: 医療診断で検査結果を基に病気の発生確率を更新する

5-14.py

```
# ベイズの定理の例
# 事前確率
p_a = 0.01  # 病気にかかっている確率
p_not_a = 1 - p_a  # 病気にかかっていない確率

# 検査の正確さ
p_positive_given_a = 0.99  # 病気の人が陽性になる確率
p_positive_given_not_a = 0.05  # 健康な人が陽性になる確率

# 全体で陽性になる確率
p_positive = p_positive_given_a * p_a + p_positive_given_not_a * p_not_a

# ベイズの定理: P(A | Positive) = P(Positive | A) * P(A) / P(Positive)
p_a_given_positive = (p_positive_given_a * p_a) / p_positive

print(f"検査が陽性だった場合に病気である確率:
{p_a_given_positive:.4f}")
```

確率論の基礎

その他

- **独立事象:** 2つの事象が互いに影響を与えない場合、それらは独立であるとされる
  - *独立事象の確率はそれぞれの確率の積で計算できる*
- **条件付き確率:** ある事象が起こった条件下で別の事象が起こる確率
  - *条件付き確率を用いることで、データの依存関係を明確にできる*
- **大数の法則:** 試行回数が増えると、平均値が期待値に近づくという法則
  - *長期的に安定した結果を得るための基盤となる*
- **中心極限定理:** 十分な数の独立したランダム変数の平均は正規分布に近づく
  - *多くの統計手法が正規分布を前提にしている理由*

5-15.py

```
import numpy as np
import random
from collections import Counter

# 1. 独立事象
# 例: コインを2回投げた場合、それぞれの結果は独立
prob_coin1 = 0.5  # コイン1が表になる確率
prob_coin2 = 0.5  # コイン2が表になる確率
independent_probability = prob_coin1 * prob_coin2
print(f"コイン2枚が両方表になる確率: {independent_probability}")

# 2. 条件付き確率
# 例: 簡易的な条件付き確率計算
p_a_and_b = 0.2  # AとBが同時に発生する確率
p_b = 0.5  # Bが発生する確率
conditional_probability = p_a_and_b / p_b
print(f"条件付き確率 P(A | B): {conditional_probability}")

. . .
```

相関・回帰の基礎

DS Lv.1-5

相関分析

2つの変数間の関係性を数値で示し、その強さや方向を評価する手法。

- **ピアソン相関係数:** 変数間の線形関係を測定（-1から1の範囲）
  - 正の値: 一方が増えるともう一方も増加
  - 負の値: 一方が増えるともう一方が減少
  - 0: 関連性なし
- **スピアマン相関係数:** 順位（ランク）に基づく相関で、非線形関係にも対応可能
- 線形関係にはピアソン、非線形や順位データにはスピアマンを適用

5-16.py

```
import numpy as np
import pandas as pd
from scipy.stats import pearsonr, spearmanr

# サンプルデータ作成
np.random.seed(42)
data = {
    "Variable_X": np.random.randint(1, 100, 50),  # 変数X
    "Variable_Y": np.random.randint(1, 100, 50)  # 変数Y
}
df = pd.DataFrame(data)

# 1. ピアソン相関係数
pearson_corr, pearson_p = pearsonr(df["Variable_X"], df["Variable_Y"])
print(f"ピアソン相関係数: {pearson_corr:.2f}, p値: {pearson_p:.4f}")
if pearson_p < 0.05:
    print("ピアソン相関係数は統計的に有意です。")
else:
    print("ピアソン相関係数は統計的に有意ではありません。")
・・・
```

相関・回帰の基礎

単回帰分析

1つの説明変数を使って目的変数を予測するためのモデル。

- 直線（y = mx + b）の形で関係を表現
- 傾き（m）は説明変数の変化が目的変数に与える影響を示す
- 切片（b）は説明変数が0のときの目的変数の値
- 1つの要因が結果にどのように影響を与えるか理解する
- 実例: 広告費（説明変数）が売上（目的変数）に与える影響を分析

5-17.py

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# サンプルデータ作成
np.random.seed(42)
X = np.random.rand(50) * 10  # 説明変数
y = 2.5 * X + np.random.randn(50) * 2  # 目的変数（傾き2.5 + ノイズ）

# データをDataFrameに格納
df = pd.DataFrame({"X": X, "y": y})

# モデルの作成と学習
model = LinearRegression()
model.fit(df[["X"]], df["y"])

# モデルのパラメータ
print(f"回帰直線の傾き: {model.coef_[0]:.2f}")
print(f"回帰直線の切片: {model.intercept_:.2f}")

・・・
```

相関・回帰の基礎

DS Lv.1-5

重回帰分析

複数の説明変数を用いて目的変数を予測するモデル。

- 目的変数と複数の説明変数の関係を直線の方程式で表現（y = b0 + b1x1 + b2x2 + ...）
- 各説明変数の係数（b）は、その変数が目的変数に与える独立した影響を示す
- 複数の要因が絡む現象を解析し、予測モデルを構築する
- 実例: 商品価格、広告費、顧客満足度（説明変数）が売上（目的変数）に与える影響を分析

5-18.py

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# サンプルデータ作成
np.random.seed(42)
X1 = np.random.rand(50) * 10  # 説明変数1
X2 = np.random.rand(50) * 5   # 説明変数2
y = 3 * X1 + 1.5 * X2 + np.random.randn(50) * 2  # 目的変数（傾き + ノイズ）

# データをDataFrameに格納
df = pd.DataFrame({"X1": X1, "X2": X2, "y": y})

# モデルの作成と学習
model = LinearRegression()
model.fit(df[["X1", "X2"]], df["y"])

. . .
```

相関・回帰の基礎

その他

- **決定係数（R^2）**：モデルがデータのどれだけの変動を説明しているかを示す指標（0〜1の範囲）
  - *高いほどモデルの説明力が強いが、過剰適合の可能性にも注意が必要*
- **標準誤差:** モデルの予測値と実際の値のズレを測る指標
  - *標準誤差が小さいほど予測が正確*
- **多重共線性:** 重回帰分析において、説明変数間で高い相関がある場合に発生する問題
  - *モデルの安定性を損なうため、相関の高い変数を除外するなどの対処が必要*
- **残差分析:** モデルの予測誤差（残差）を分析し、モデルの適合度や仮定の妥当性を評価
  - *残差のランダム性を確認することで、モデルが適切か判断*

5-19.py

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# サンプルデータ作成
np.random.seed(42)
X1 = np.random.rand(50) * 10  # 説明変数1
X2 = X1 + np.random.rand(50)  # 説明変数2（高い相関を持つ）
y = 3 * X1 + 1.5 * X2 + np.random.randn(50) * 2  # 目的変数

# データをDataFrameに格納
df = pd.DataFrame({"X1": X1, "X2": X2, "y": y})

# モデルの作成と学習
model = LinearRegression()
model.fit(df[["X1", "X2"]], df["y"])

# 1. 決定係数（R²）
r2 = r2_score(df["y"], model.predict(df[["X1", "X2"]]))
・・・
```

演習：映画レビュー評価

学習のポイント

1. **記述統計の基礎理解:** 平均、中央値、モード、分散、標準偏差、範囲、四分位範囲（IQR）など、基本的な統計指標を計算し、データの中心傾向やばらつきを理解します。
2. **データの視覚化スキルの習得:** ヒストグラムや箱ひげ図を作成し、データの分布や外れ値を視覚的に把握します。
3. **歪度と尖度の活用:** データの形状（左右対称性やピークの鋭さ）を定量的に評価し、分布の特性を解釈します。
4. **データからの洞察発見:** 統計的指標と可視化結果をもとに、データの傾向や特徴を発見し、具体的な意味を導き出します。

動作の流れ

1. 架空の映画レビュー評価データを作成し、DataFrameに格納します。
2. 平均、中央値、モード、分散、標準偏差、範囲、四分位範囲、歪度、尖度を計算してデータの特性を数値化します。
3. ヒストグラムを用いて評価スコアの分布を視覚化し、箱ひげ図で外れ値や分布の広がりを確認します。
4. 記述統計と可視化結果から、映画の評価傾向を分析し、「高評価か」「低評価か」「平均的か」などの洞察を導きます。
5. データの特徴をもとに、評価分布の対称性やピークの鋭さについての解釈を行い、最終的な結論を導きます。

5-20.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import skew, kurtosis

# 1. サンプルデータ作成（架空の映画レビュー評価データ）
np.random.seed(42)
movie_ratings = np.random.randint(1, 11, 100)  # 1〜10の範囲で100人の評価

# データをDataFrameに格納
df = pd.DataFrame({"MovieRatings": movie_ratings})

# 2. 記述統計を計算
mean_rating = df["MovieRatings"].mean()
median_rating = df["MovieRatings"].median()
mode_rating = df["MovieRatings"].mode().iloc[0]
variance_rating = df["MovieRatings"].var()
std_dev_rating = df["MovieRatings"].std()
range_rating = df["MovieRatings"].max() - df["MovieRatings"].min()

. . . .
```

演習：カフェチェーンの売上

学習のポイント

1. **t検定の実践:** 2つのグループ間（都市Aと都市B）の平均の違いが統計的に有意かどうかを検証する方法を学びます。
2. **カイ二乗検定の実践:** カテゴリデータ（コーヒーと紅茶の注文数）を使って、2つの都市間で注文傾向に違いがあるかを分析します。
3. **信頼区間の理解:** 都市Aの売上平均について、母集団平均の範囲を95%の信頼で推定する方法を学びます。
4. **データ可視化スキルの強化:** ヒストグラムと棒グラフを作成して、売上や注文分布を視覚的に把握します。

動作の流れ

1. 架空の都市Aと都市Bの1ヶ月間の売上データと、コーヒーと紅茶の注文数を作成します。
2. 2つの都市の売上平均に有意な差があるかどうかを検定します。
3. 都市Aと都市Bで注文種類（コーヒー、紅茶）の分布に違いがあるかを検定します。
4. 都市Aの売上平均について、95%の信頼区間を計算して母集団平均を推定します。
5. 売上のヒストグラムを作成して都市間の売上分布を比較します。
6. 注文種類ごとの棒グラフを作成して注文傾向を視覚化します。

5-21.py

```
import numpy as np
import pandas as pd
from scipy.stats import ttest_ind, chi2_contingency, norm
import matplotlib.pyplot as plt

# 1. サンプルデータ作成（架空のカフェチェーン売上データ）
np.random.seed(42)
city_a_sales = np.random.normal(500, 50, 30)  # 都市Aの1ヶ月の売上（ドル）
city_b_sales = np.random.normal(520, 55, 30)  # 都市Bの1ヶ月の売上（ドル）

# カテゴリデータ（例えば、ドリンクの注文種類）
orders = np.array([
    [50, 30],  # 都市A: コーヒー、紅茶
    [40, 60]   # 都市B: コーヒー、紅茶
])

# 2. t検定（売上の比較）
t_stat, p_value = ttest_ind(city_a_sales, city_b_sales)
print("=== t検定: 売上の比較 ===")
 ・ ・ ・ ・
```

演習：バスケのフリースロー

学習のポイント

1. **独立事象の理解:** 2人の選手がそれぞれ独立してフリースローを成功させる確率を計算し、独立事象の概念を学びます。
2. **条件付き確率の実践:** 選手1が成功した場合に選手2が成功する確率を仮定し、条件付き確率の応用を学びます。
3. **大数の法則のシミュレーション:** 選手1のフリースロー成功率を多回数試行し、理論値に近づく過程を観察しながら大数の法則を理解します。
4. **中心極限定理の理解:** フリースロー成功率のサンプル平均を多数回計算し、その分布が正規分布に近づく様子を視覚的に学びます。

動作の流れ

1. 選手1（成功率80%）と選手2（成功率65%）の確率を設定します。
2. 選手1と選手2が両方成功する確率を計算して出力します。
3. 選手1が成功した場合に選手2も成功する確率を計算し、条件付き確率の応用例を示します。
4. 選手1のフリースローを10,000回試行し、長期的な成功率が理論値に近づくことを確認します。
5. 選手1のフリースロー成功率を30回試行した平均を1,000回繰り返し、その分布が正規分布に近づく様子をヒストグラムで可視化します。

5-22.py

```
import numpy as np
import random

# 1. サンプルデータ作成（架空のバスケットボール選手のフリースロー成功確率）
player_1_success_rate = 0.8  # 選手1のフリースロー成功率（80%）
player_2_success_rate = 0.65  # 選手2のフリースロー成功率（65%）

# 2. 独立事象の確率
# 選手1と選手2が両方成功する確率
independent_probability = player_1_success_rate * player_2_success_rate
print(f"選手1と選手2が両方フリースローを成功させる確率: {independent_probability:.2f}")

# 3. 条件付き確率
# 選手1がフリースローに成功した場合、選手2も成功する確率を仮定
conditional_probability = 0.7  # 選手1が成功したときに選手2が成功する確率
print(f"条件付き確率: 選手1が成功した場合に選手2が成功する確率: {conditional_probability:.2f}")
・・・
```

演習：バスケのフリースロー

学習のポイント

1. **独立事象の理解:** 2人の選手がそれぞれ独立してフリースローを成功させる確率を計算し、独立事象の概念を学びます。
2. **条件付き確率の実践:** 選手1が成功した場合に選手2が成功する確率を仮定し、条件付き確率の応用を学びます。
3. **大数の法則のシミュレーション:** 選手1のフリースロー成功率を多回数試行し、理論値に近づく過程を観察しながら大数の法則を理解します。
4. **中心極限定理の理解:** フリースロー成功率のサンプル平均を多数回計算し、その分布が正規分布に近づく様子を視覚的に学びます。

動作の流れ

1. 選手1（成功率80%）と選手2（成功率65%）の確率を設定します。
2. 選手1と選手2が両方成功する確率を計算して出力します。
3. 選手1が成功した場合に選手2も成功する確率を計算し、条件付き確率の応用例を示します。
4. 選手1のフリースローを10,000回試行し、長期的な成功率が理論値に近づくことを確認します。
5. 選手1のフリースロー成功率を30回試行した平均を1,000回繰り返し、その分布が正規分布に近づく様子をヒストグラムで可視化します。

5-22.py

```
import numpy as np
import random

# 1. サンプルデータ作成（架空のバスケットボール選手のフリースロー成功確率）
player_1_success_rate = 0.8  # 選手1のフリースロー成功率（80%）
player_2_success_rate = 0.65  # 選手2のフリースロー成功率（65%）

# 2. 独立事象の確率
# 選手1と選手2が両方成功する確率
independent_probability = player_1_success_rate * player_2_success_rate
print(f"選手1と選手2が両方フリースローを成功させる確率: {independent_probability:.2f}")

# 3. 条件付き確率
# 選手1がフリースローに成功した場合、選手2も成功する確率を仮定
conditional_probability = 0.7  # 選手1が成功したときに選手2が成功する確率
print(f"条件付き確率: 選手1が成功した場合に選手2が成功する確率: {conditional_probability:.2f}")
・・・
```

演習：フィットネス分析

学習のポイント

1. **ピアソン相関係数の理解:** 運動時間と消費カロリーの間の線形関係を定量的に測定し、相関の強さと方向を学びます。
2. **スピアマン相関係数の応用:** 運動時間と歩数の間の順位相関を分析し、非線形関係やランクベースの相関の特性を学びます。
3. **単回帰分析の実践:** 運動時間を説明変数、消費カロリーを目的変数として回帰モデルを構築し、回帰直線を求めます。
4. **統計指標の解釈力の向上:** 相関係数、p値、回帰係数を解釈し、データからの洞察を得るスキルを磨きます。

動作の流れ

1. 架空のフィットネスデータとして、運動時間、消費カロリー、歩数を含むデータセットを作成します。
2. 運動時間と消費カロリーの間の相関係数とp値を計算し、線形関係の有無を確認します。
3. 運動時間と歩数の間の順位相関係数とp値を計算し、非線形な関係を検証します。
4. 運動時間を使って消費カロリーを予測する単回帰モデルを構築し、回帰直線を計算します。
5. 統計的検定結果と可視化をもとに、運動時間が消費カロリーや歩数にどのように影響しているかを解釈します。

5-23.py

```
import numpy as np
import pandas as pd
from scipy.stats import pearsonr, spearmanr
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# 1. サンプルデータ作成（架空のフィットネスデータ）
np.random.seed(42)
hours_of_exercise = np.random.uniform(1, 10, 50)  # 週あたりの運動時間（1〜10時間）
calories_burned = 200 * hours_of_exercise + np.random.normal(0, 50, 50)  # 消費カロリー（ノイズ付き）
steps_taken = np.random.uniform(5000, 20000, 50)  # 歩数（非線形な関係）

# データをDataFrameに格納
df = pd.DataFrame({
    "ExerciseHours": hours_of_exercise,
    "CaloriesBurned": calories_burned,
    "StepsTaken": steps_taken
})
・・・
```

DS Lv.1-5

お疲れ様でした

info@mjin.space