Генерация случайной матрицы

Сгенерируйте матрицу, состоящую из 1000 строк и 50 столбцов, элементы которой являются случайными из нормального распределения N(1,100).

Функция для генерации чисел из нормального распределения: np.random.normal

Параметры:

loc: среднее нормального распределения (в нашем случае 1) scale: стандартное отклонение нормального распределения (в нашем случае 10) size: размер матрицы (в нашем случае (1000, 50))

```
import numpy as np
X = np.random.normal(loc=1, scale=10, size=(1000, 50))
print(X)
```

```
[[ -6.2201196   14.0860253   -5.12600592 ...  11.46746267   2.59147128
   -0.40065279]
 [-13.4470602   11.78637203  -7.32912667 ...   1.85287912  -3.75254696
    0.36407027]
 [ 12.18647942   1.13704169   0.64549411 ... -14.87864602  -4.57867335
   -8.534461  ]
 ...
 [ -7.24130195  -8.05822435  -8.64126897 ... -12.68801129 -16.06391148
   -8.61964963]
 [ -1.57616808  -5.16962521  -2.77977339 ... -11.81904461  22.98684149
    3.30636145]
 [ -0.29310373  -1.89251516   7.67350662 ...   4.56252394  -2.79200688
   14.11669986]]
```

Нормировка матрицы

Произведите нормировку матрицы из предыдущего задания: вычтите из каждого столбца его среднее значение, а затем поделите на его стандартное отклонение.

Функция для вычисления среднего: np.mean

Функция для вычисления стандартного отклонения: np.std

Первый параметр — матрица, для которой производятся вычисления. Также полезным будет параметр axis, который указывает, по какому измерению вычисляются среднее и стандартное отклонение (если axis=0, то по столбцам, если axis=1, то по строкам; если его не указывать, то данные величины будут вычислены по всей матрице).

```
m = np.mean(X, axis=0)
std = np.std(X, axis=0)
```

```
X_norm = ((X - m)  / std)
print (X_norm)
```

```
[[-0.73796023  1.28492662 -0.60979934 ...  1.10225486  0.22910964
  -0.11723012]
 [-1.46796827  1.06101582 -0.83362246 ...  0.1073095  -0.39245503
  -0.040182  ]
 [ 1.12132797  0.02412033 -0.02345147 ... -1.62411788 -0.473396
  -0.93673552]
 ...
 [-0.84111194 -0.87119699 -0.96692781 ... -1.39742456 -1.59867938
  -0.94531853]
 [-0.26886519 -0.58994218 -0.37143696 ... -1.30750134  2.22737655
   0.2562626 ]
 [-0.13926024 -0.27085914  0.69055012 ...  0.38771151 -0.29834468
   1.34543638]]
```

Операции над элементами матрицы

Выведите для заданной матрицы номера строк, сумма элементов в которых превосходит 10.

Функция для подсчета суммы: np.sum

Аргументы аналогичны функциям np.mean и np.std.

К матрицам можно применять логические операции, которые будут применяться поэлементно. Соответственно, результатом такой операции будет матрица такого же размера, в ячейках которой будет записано либо True, либо False. Индексы элементов со значением True можно получить с помощью функции np.nonzero.

```
Z = np.array([[4, 5, 0],
              [1, 9, 3],
              [5, 1, 1],
              [3, 3, 3],
              [9, 9, 9],
              [4, 7, 1]])
r = np.sum(Z, axis=1)
print(r)
print (np.nonzero(r > 10))
```

```
[ 9 13  7  9 27 12]
(array([1, 4, 5]),)
```

Объединение матриц

Сгенерируйте две единичные матрицы (т.е. с единицами на диагонали) размера 3x3. Соедините две матрицы в одну размера 6x3.

Функция для генерации единичной матрицы: np.eye

Аргумент: число строк (или, что эквивалентно, столбцов).

Функция для вертикальной стыковки матриц: np.vstack((A, B))

```
A = np.eye(3)
B = np.eye(3)
print (A)
print (B)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
AB = np.vstack((A, B))
print (AB)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Напишите программный код или сгенерируйте его с помощью искусственного интеллект

Данное задание направлено на знакомство с инструментарием, который пригодится в дальнейших практических заданиях.

Вы научитесь:

- работать с данными используя язык Python и пакет Pandas
- делать предобработку данных
- находить простые закономерности в данных

Введение

Сейчас Python является одним из наиболее распространенных языков программирования. Одним из его преимуществ является большое количество пакетов, решающих самые разные задачи. В нашем курсе мы рекомендуем использовать библиотеки Pandas, NumPy и SciPy, которые существенно упрощают чтение, хранение и обработку данных. В дальнейших работах вы также познакомитесь с пакетом Scikit-Learn, в котором реализованы многие алгоритмы машинного обучения.

Начало работы

Для того, чтобы начать работать с данными, необходимо сначала загрузить их из файла. В данном задании мы будем работать с данными в формате CSV, предназначенном для хранения табличных данных: столбцы разделяются запятой, первая строка содержит имена столбцов.

Пример загрузки данных в Pandas:

```
import pandas
data = pandas.read_csv('train.csv', index_col='PassengerId')
```

Данные будут загружены в виде DataFrame, с помощью которого можно удобно работать с ними. В данном случае параметр

index_col='PassengerId' означает, что колонка PassengerId задает нумерацию строк данного датафрейма.

Для того, чтобы посмотреть что представляют из себя данные, можно воспользоваться несколькими способами:

- более привычным с точки зрения Python (если индекс указывается только один, то производится выбор строк):

```
data[:10]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8

- или же воспользоваться методом датафрейма:

```
data.head(10)
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8

Далее: New interactive sheet

Один из способов доступа к столбцам датафрейма — использовать квадратные скобки и название столбца:

```
data['Sex']
```

PassengerId	Sex
1	male
2	female
3	female
4	female
5	male
...	...
887	male
888	female
889	female
890	male
891	male

891 rows × 1 columns

dtype: object

Для подсчета некоторых статистик (количества, среднее, максимум, минимум) можно также использовать методы датафрейма:

```
data['Sex'].value_counts()
```

Sex	count
male	577
female	314

dtype: int64

1.Какое количество мужчин и женщин ехало на корабле?

```
gender_group = data.groupby('Sex').size()
print(gender_group)
```

```
Sex
female    314
male      577
dtype: int64
```

2.Какой части пассажиров удалось выжить? Посчитайте долю выживших пассажиров.

```
survival_stats = data['Survived'].value_counts()
print(f"\nДетальная статистика:")
print(survival_stats)
print(f"Выжили: {survival_stats.get(1, 0)} пассажиров")
print(f"Погибли: {survival_stats.get(0, 0)} пассажиров")
```

```
Детальная статистика:
Survived
0    549
1    342
Name: count, dtype: int64
Выжили: 342 пассажиров
Погибли: 549 пассажиров
```

3.Какую долю пассажиры первого класса составляли среди всех пассажиров?

```
first_class_count = (data['Pclass'] == 1).sum()
total_passengers = len(data)
first_class_rate = first_class_count / total_passengers
class_distribution = data['Pclass'].value_counts().sort_index()
print(f"\nРаспределение по классам:")
for pclass, count in class_distribution.items():
    rate = count / total_passengers
    print(f"Класс {pclass}: {count} пассажиров ({rate:.2%})")
```

```
Распределение по классам:
Класс 1: 216 пассажиров (24.24%)
Класс 2: 184 пассажиров (20.65%)
Класс 3: 491 пассажиров (55.11%)
```

4. Какого возраста были пассажиры? Посчитайте среднее и медиану возраста пассажиров.

```
mean_age = data['Age'].mean()
print(f"Средний возраст пассажиров: {mean_age:.2f} лет")
```

```
Средний возраст пассажиров: 29.70 лет
```

```
median_age = data['Age'].median()
print(f"Медианный возраст пассажиров: {median_age:.2f} лет")
```

```
Медианный возраст пассажиров: 28.00 лет
```

5. Коррелируют ли число братьев/сестер/супругов с числом родителей/детей?

```
correlation = data['SibSp'].corr(data['Parch'])
print(f"Корреляция Пирсона между SibSp и Parch: {correlation:.4f}")
```

```
Корреляция Пирсона между SibSp и Parch: 0.4148
```

6. Какое самое популярное женское имя на корабле? Извлеките из полного имени пассажира (колонка Name) его личное имя (First Name). Это задание — типичный пример того, с чем сталкивается специалист по анализу данных. Данные очень разнородные и шумные, но из них требуется извлечь необходимую информацию. Попробуйте вручную разобрать несколько значений столбца Name и выработать правило для извлечения имен, а также разделения их на женские и мужские.

```
women = data[data['Sex'] == 'female'].copy()
# Список для хранения извлеченных имен
female_names = []
for name in women['Name']:
    first_name = None
    # Проверяем наличие скобок - приоритетный способ извлечения имени
    if '(' in name and ')' in name:
        start_index = name.find('(') + 1
        end_index = name.find(')')
        name_in_brackets = name[start_index:end_index]
        # Извлекаем первое слово из содержимого скобок
        parts = name_in_brackets.split()
        if parts:
            first_name = parts[0]
    # Если скобок нет или не удалось извлечь имя из скобок
    if first_name is None:
        # Разделяем имя по запятой и точке
        parts = name.split(',')
        if len(parts) > 1:
            after_comma = parts[1].strip()
            # Список женских титулов
            titles = ['Mrs.', 'Miss.', 'Ms.', 'Lady.', 'Mlle.', 'Mme.', 'Countes
            for title in titles:
                if title in after_comma:
                    title_index = after_comma.find(title)
                    after_title = after_comma[title_index + len(title):].strip()

                    if after_title:
                        name_parts = after_title.split()
                        if name_parts:
                            first_name = name_parts[0]
```

Данное задание основано на материалах лекций по логическим методам и направлено на знакомство с решающими деревьями (Decision Trees).

Вы научитесь:

- обучать решающие деревья
- находить наиболее важные для них признаки

Введение

Решающие деревья относятся к классу логических методов. Их основная идея состоит в объединении определенного количества простых решающих правил, благодаря чему итоговый алгоритм является интерпретируемым. Как следует из названия, решающее дерево представляет собой бинарное дерево, в котором каждой вершине сопоставлено некоторое правило вида "j-й признак имеет значение меньше b". В листьях этого дерева записаны числа-предсказания. Чтобы получить ответ, нужно стартовать из корня и делать переходы либо в левое, либо в правое поддерево в зависимости от того, выполняется правило из текущей вершины или нет.

Одна из особенностей решающих деревьев заключается в том, что они позволяют получать важности всех используемых признаков. Важность признака можно оценить на основе того, как сильно улучшился критерий качества благодаря использованию этого признака в вершинах дерева.

Данные

В этом задании мы вновь рассмотрим данные о пассажирах Титаника. Будем решать на них задачу классификации, в которой по различным характеристикам пассажиров требуется предсказать, кто из них выжил после крушения корабля.

Реализация в Scikit-Learn

В библиотеке scikit-learn решающие деревья реализованы в классах sklearn.tree.DecisionTreeClassifier (для классификации) и sklearn.tree.DecisionTreeRegressor (для регрессии). Обучение модели производится с помощью функции fit.

Пример использования:

В этом задании вам также потребуется находить важность признаков. Это можно сделать, имея уже обученный классификатор:

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
X = np.array([[-10, 0], [-5, 0], [10, 0], [10, 0]])
y = np.array([0, 1, 2, 3])
clf = DecisionTreeClassifier()
clf.fit(X, y)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
importances = clf.feature_importances_
importances
```

```
array([1., 0.])
```

Переменная importances будет содержать массив "важностей" признаков. Индекс в этом массиве соответствует индексу признака в данных.

Стоит обратить внимание, что данные могут содержать пропуски. Pandas хранит такие значения как nan (not a number). Для того, чтобы проверить, является ли число nan'ом, можно воспользоваться функцией np.isnan.

Инструкция по выполнению

1. Загрузите выборку из файла train.csv с помощью пакета Pandas.
2. Оставьте в выборке четыре признака: класс пассажира (Pclass), цену билета (Fare), возраст пассажира (Age) и его пол (Sex).
3. Обратите внимание, что признак Sex имеет строковые значения.

4. Выделите целевую переменную — она записана в столбце Survived.
5. В данных есть пропущенные значения — например, для некоторых пассажиров неизвестен их возраст. Такие записи при чтении их в pandas принимают значение nan. Найдите все объекты, у которых есть пропущенные признаки, и удалите их из выборки.
6. Обучите решающее дерево с параметром random_state=241 и остальными параметрами по умолчанию (речь идет о параметрах конструктора DecisionTreeClassifier).
7. Вычислите важности признаков и найдите два признака с наибольшей важностью. Их названия будут ответами для данной задачи (в качестве ответа укажите названия признаков через запятую или пробел, порядок не важен).

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

data = pd.read_csv('train.csv')
features = ['Pclass', 'Fare', 'Age', 'Sex']
X = data[features]
y = data['Survived']
X = pd.get_dummies(X, columns=['Sex'], drop_first=True)
X_cleaned = X.dropna()
y_cleaned = y[X_cleaned.index]
model = DecisionTreeClassifier(random_state=241)
model.fit(X_cleaned, y_cleaned)
importances = pd.DataFrame({
    'feature': X_cleaned.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
top_features = importances.head(2)['feature'].tolist()
print('Самые важные признаки:', ', '.join(top_features))
```

```
Самые важные признаки: Fare, Sex_male
```

Напишите программный код или сгенерируйте его с помощью искусственного интеллекта.