ما سه نوع خطای رایج در داده ها داریم :

1. کمبود مقادیر
2. مقادیر بد
3. داده های تکراری

1- **\*\*کمبود مقادیر:\*\***

این خطا به معنی وجود خانه‌هایی در داده ها که مقداری در آن ها وجود ندارد است. برای مثال، اگر در یک جدول اطلاعاتی پزشکی، بخش وزن بیمار را در نظر بگیرید، اگر برای یک بیمار وزنی وارد نشده باشد، در این صورت یک خطا کمبود مقدار وجود دارد.

2- **\*\*مقادیر بد:\*\***

این خطا به معنی وجود مقادیر نامعقول یا غیرمنطقی است. برای مثال، اگر در یک جدول اطلاعاتی پزشکی، بخش قد بیمار را در نظر بگیرید، اگر یک بیمار قد ۱۰۰۰ سانتی‌متر را داشته باشد، در این صورت یک خطا مقادیر بد وجود دارد.

3- **\*\*داده های تکراری:\*\***

این خطا به معنی وجود داده های تکراری در داده های شماست. برای مثال، اگر در یک جدول اطلاعاتی پزشکی، بخش شماره تلفن بیمار را در نظر بگیرید، اگر دو ردیف با شماره تلفن یکسان وجود داشته باشد، در این صورت یک خطا داده های تکراری وجود دارد.

***خب ! امیدواریم کد هارا خوانده باشید و تمرین کرده باشید***

# بیاید راجب ML Schema صحبت کنیم

# اصلا همچین چیزی چی هست چه کاربردی داره و چه کار انجام میده اول یم نگاهی به منابع بکنید

# <https://ml-schema.github.io/documentation/ML%20Schema.html>

# <https://bryteflow.com/what-is-schema-on-read-and-how-does-it-differ-from-schema-on-write/>

و بیاید نظر چت جی پی تی رو هم بپرسیم:

ML Schema یک "طرح" یا "ساختار" است که به ما امکان می دهد که اطلاعات مربوط به یادگیری ماشین و داده کاوی را به طور منظم و استاندارد نمایش دهیم. این شامل اطلاعات در مورد الگوریتم های یادگیری ماشین، مجموعه داده ها و آزمایش هایی که انجام داده ایم، است.

برای مثال، فرض کنید شما یک الگوریتم یادگیری ماشین را برای تشخیص چهره استفاده کرده اید. با استفاده از ML Schema، شما می توانید جزئیات الگوریتم، پارامترهایی که استفاده کرده اید، مجموعه داده ای که برای آموزش الگوریتم استفاده کرده اید، و نتایج آزمایشات خود را به طور منظم و استاندارد ثبت کنید. این به شما کمک می کند تا کار خود را به دیگران به صورت قابل فهم منتقل کنید، و همچنین به دیگران اجازه می دهد تا کار شما را بررسی، تحلیل، و در صورت نیاز تکرار کنند.به طور خلاصه، ML Schema یک ابزار برای سازماندهی و استانداردسازی اطلاعات در حوزه یادگیری ماشین است.

استفاده از "Schema on Read" در پردازش داده ها به دلیل برخی مزایایی که دارد، مورد توجه قرار گرفته است. برخی از این مزایا عبارتند از:

۱. سهولت در تغییرات: در "Schema on Read"، داده ها بدون نیاز به تعریف ساختار قبلی، می توانند به سیستم اضافه شوند. این به کاربران اجازه می دهد تا با تغییرات در داده ها روبرو شوند و بدون نیاز به تغییر در ساختار دیتابیس، اطلاعات جدید را به دیتابیس اضافه کنند.

۲. انعطاف پذیری: این روش به کاربران اجازه می دهد که داده های خود را با استفاده از چندین فرمت و محتوا به سیستم اضافه کنند. همچنین، با استفاده از "Schema on Read"، کاربران می توانند به راحتی با داده های جدید کار کنند و آن ها را به سیستم خود اضافه کنند.

۳. عملکرد بهتر: در "Schema on Read"، داده ها به صورت فشرده در حافظه ذخیره می شوند، بنابراین پردازش آن ها سریع تر است و در نتیجه عملکرد بهتری دارد.

بطور خلاصه، استفاده از "Schema on Read" به کاربران اجازه می دهد تا با داده های جدید و نامعلوم کار کنند و بدون نیاز به تعریف ساختار قبلی، آن ها را به سیستم خود اضافه کنند.

برای استفاده از "Schema on Read"، می توان از ابزارهای مختلفی استفاده کرد. برخی از این ابزارها عبارتند از:

۱. Hadoop: یک سامانه توزیع شده برای پردازش داده ها که می تواند با استفاده از "Schema on Read" به عنوان پایگاه داده استفاده شود.

۲. Hive: یک ابزار پردازش داده های توزیع شده مبتنی بر Hadoop که از SQL برای پرس و جوی داده ها استفاده می کند.

۳. Spark: یک سامانه پردازش داده های توزیع شده که با استفاده از "Schema on Read"، به عنوان یک پایگاه داده مورد استفاده قرار می گیرد.

۴. NoSQL دیتابیس ها: بسیاری از NoSQL دیتابیس ها، مانند MongoDB و Cassandra، از "Schema on Read" پشتیبانی می کنند و به کاربران اجازه می دهند تا داده های خود را بدون نیاز به تعریف ساختار قبلی، به دیتابیس اضافه کنند.

۵. ELT (Extract, Load, Transform) Tools: این ابزارها به کاربران اجازه می دهند تا داده های خود را با استفاده از "Schema on Read"، به سیستم خود اضافه کنند و سپس با استفاده از فرآیند ETL، آن ها را تبدیل کنند و در نهایت به فرمت مورد نظر خود تبدیل کنند.

با جستجوی عبارت "آموزش Schema on Read" در موتورهای جستجو، می توانید به راحتی منابع آموزشی مختلفی درباره این موضوع پیدا کنید. این منابع شامل ویدئوهای آموزشی، مقالات، کتاب‌ها، و نرم‌افزارهای آموزشی مختلفی هستند.

برای مثال، در وب‌سایت Confluent، یک توضیح کامل درباره Schema on Read و نحوه استفاده از آن در پردازش داده ها قرار داده شده است. همچنین، شرکت Dell نیز در وب‌سایت خود، یک مقاله درباره تفاوت Schema on Read و Schema on Write و کاربردهای آن‌ها منتشر کرده است. همچنین، در وب‌سایت BryteFlow، یک مقاله درباره تفاوت Schema on Read و Schema on Write و کاربردهای آن‌ها قرار داده شده است.

در کل، برای یادگیری بیشتر درباره Schema on Read، می توانید به راحتی با جستجو در اینترنت به منابع آموزشی مختلف دسترسی پیدا کنید.

**بسیار خوب، بیایید فرض کنیم که ما یک الگوریتم یادگیری ماشین برای پیش بینی قیمت خانه استفاده کرده ایم.**

در این مثال، ما از یک الگوریتم رگرسیون خطی استفاده می کنیم و مجموعه داده های Boston Housing را به عنوان مجموعه داده آموزش استفاده می کنیم.

مراحل کار به شرح زیر است:

1. \*\*آماده سازی داده ها\*\*: ابتدا داده ها را برای آموزش الگوریتم آماده می کنیم. این مرحله ممکن است شامل پاکسازی داده ها، تبدیل داده های غیر عددی به عددی، و یا مقیاس بندی ویژگی ها باشد.

```python

from sklearn.datasets import load\_boston

from sklearn.model\_selection import train\_test\_split

boston = load\_boston()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(boston.data, boston.target, test\_size=0.2, random\_state=42)

2. \*\*آموزش الگوریتم\*\*: سپس الگوریتم رگرسیون خطی را بر روی داده های آموزش آموزش می دهیم.

```python

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(X\_train, y\_train)

3. \*\*ارزیابی الگوریتم\*\*: در نهایت، مدل را بر روی داده های تست ارزیابی می کنیم.

```python

from sklearn.metrics import mean\_squared\_error

predictions = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, predictions)

print(f"Mean Squared Error: {mse}")

**حال، با استفاده از ML Schema، ما می توانیم این اطلاعات را به صورت ساختارمند نگه داری کنیم. برای این کار، ما یک فایل JSON می سازیم که شامل جزئیات الگوریتم، پارامترهای استفاده شده، مجموعه داده، و نتایج آزمایش است.**

```json

{

"algorithm": {

"name": "Linear Regression",

"parameters": {

"fit\_intercept": true,

"normalize": false

}

},

"dataset": {

"name": "Boston Housing",

"features": ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"],

"target": "MEDV"

},

"experiment": {

"mse": mse

}

}

```لطفا توجه داشته باشید که این فقط یک مثال ساده است و در عمل، ML Schema می تواند بسیار پیچیده تر و جامع تر باشد.

در کار با داده‌های گمشده یا Missing Data در کتابخانه پانداس، چندین روش وجود دارد. این داده‌ها می‌توانند زمانی رخ دهند که اطلاعاتی برای یک یا چند مورد یا یک واحد کلی فراهم نشده باشد. داده‌های گمشده می‌توانند به عنوان مقادیر NA (Not Available) در pandas معرفی شوند[1].

\*\*مثال\*\*:

فرض کنید یک DataFrame به شرح زیر دارید:

```python

import pandas as pd

import numpy as np

data = {

'A': [1, 2, np.nan, 4, 5],

'B': [np.nan, 2, 3, 4, 5],

'C': [1, 2, 3, np.nan, 5],

'D': [1, 2, 3, 4, 5]

}

df = pd.DataFrame(data)

```

در کار با داده‌های گمشده یا Missing Data در کتابخانه پانداس، چندین روش وجود دارد. این داده‌ها می‌توانند زمانی رخ دهند که اطلاعاتی برای یک یا چند مورد یا یک واحد کلی فراهم نشده باشد. داده‌های گمشده می‌توانند به عنوان مقادیر NA (Not Available) در pandas معرفی شوند[1].

\*\*مثال\*\*:

فرض کنید یک DataFrame به شرح زیر دارید:

```python

import pandas as pd

import numpy as np

data = {

'A': [1, 2, np.nan, 4, 5],

'B': [np.nan, 2, 3, 4, 5],

'C': [1, 2, 3, np.nan, 5],

'D': [1, 2, 3, 4, 5]

}

df = pd.DataFrame(data)```

برای شناسایی داده‌های گمشده، می‌توانید از تابع `isna()` استفاده کنید:

```python

df.isna()

این کد برای هر ستون DataFrame، مقادیر True/False برمی‌گرداند که نشان‌دهنده وجود داده گمشده در آن ستون است[2].

برای مطالعه بیشتر در زمینه کار با داده‌های گمشده در pandas، منابع زیر را پیشنهاد می‌کنم:

1. "Working with Missing Data in Pandas - GeeksforGeeks"[1]

2. "Pandas Handling Missing Values (With Examples) - Programiz"[2]

3. "Working with missing data — pandas 2.1.1 documentation"[3]

4. "Python: How to Handle Missing Data in Pandas DataFrame - Stack Abuse"[4]

دستور `schema` در پانداس (Pandas) واژگانی است که در مورد داده‌هایی که در دیتافریم (DataFrame) قرار دارند، اطلاعاتی را فراهم می‌کند. این دستور، اطلاعاتی را مانند نوع داده (data type)، نام ستون‌ها (column names) و تعداد سطرها (number of rows) در دیتافریم به شما نشان می‌دهد.

برای نمایش اطلاعات `schema` یک دیتافریم، می‌توانید از تابع `info()` استفاده کنید. به عنوان مثال:

```python

import pandas as pd

# ساخت دیتافریم تست

df = pd.DataFrame({'col1': [1, 2, 3], 'col2': ['a', 'b', 'c']})

# نمایش اطلاعات schema

df.info()

خروجی:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 3 entries, 0 to 2

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 col1 3 non-null int64

1 col2 3 non-null object

dtypes: int64(1), object(1)

memory usage: 176.0+ bytes

در این خروجی، اطلاعات `schema` دیتافریم شامل نوع داده هر ستون (`Dtype`)، تعداد سطرها (`RangeIndex`) و تعداد مقادیر غیر خالی (`Non-Null Count`) است. همچنین، نام ستون‌ها و تعداد کل ستون‌ها (`total columns`) نیز در خروجی ذکر شده‌اند.