مقدمات

خواندن از فایل CSV

یکی از فرمت‌های پرکاربرد در استفاده از داده‌های جدولی، **CSV (Comma Separated Value)** است. در این فرمت، هر نمونه در یک خط جدید آمده است و ستون‌های هر سطر با علامت کاما (,) از یکدیگر جدا شده‌اند. نمونه‌ی زیر بخشی از یک فایل `csv` است.

```
Race,Population (2020),Percentage
Total population,5706494,100%
White or European American,4423146,77.5%
Black or African American,398434,7.0%
Native American,68641,1.2%
Asian American,299190,5.2%
Native Hawaiian and Other Pacific Islander,2918,0.1%
Some other race,168444,3.0%
Two or more races,345721,6.1%
```

کتابخانه‌ی پانداز برای خواندن فایل‌هایی با پسوند `csv` ، تابع `read_csv` را معرفی کرده است.

این تابع با دریافت آدرس فایل (آدرس محلی یا *URL*) به عنوان ورودی و خواندن آن، یک شیء از کلاس دیتافریم پانداز را خروجی می‌دهد.

فرض کنید فایل `csv` که بالاتر بخشی از محتوای آن را نشان دادیم در فایلی به نام `population.csv` وجود داشته باشد. برای خواندن محتوای فایل `population.csv` می‌توانیم از دستور زیر استفاده کنیم.

```
df = pd.read_csv('./population.csv')
df
```

	Race	Population (2020)	Percentage
0	Total population	5706494	100%
1	White or European American	4423146	77.5%
2	Black or African American	398434	7.0%
3	Native American	68641	1.2%
4	Asian American	299190	5.2%
5	Native Hawaiian and Other Pacific Islander	2918	0.1%
6	Some other race	168444	3.0%
7	Two or more races	345721	6.1%

پس از اجرای کد بالا، `df` یک دیتافریم خواهد بود. برخی از آرگومان‌های مهم این تابع در جدول زیر آورده شده است:

آرگومان	توضیحات
`sep`	رشته‌ی جداکننده‌ای که در فایل استفاده شده است. به‌طور پیش‌فرض کاما (,)
`header`	این آرگومان مقادیر عددی می‌پذیرد و مشخص می‌کند که چه سطرهایی از فایل باید به‌عنوان نام ستون‌ها در نظر گرفته شوند. به‌طور پیش‌فرض مقدار `0` دارد و نام ستون‌ها از سطر اول فایل استنتاج می‌شوند.
`index_col`	این آرگومان ستونی که باید به‌عنوان نمایه در نظر گرفته شود را مشخص می‌کند و نام یا شماره‌ی ستون را می‌پذیرد.
`na_values`	این آرگومان رشته‌هایی که باید به‌عنوان `NA/NaN` در نظر گرفته شوند را ورودی می‌گیرد.

آرگومان	توضیحات
`encoding`	این آرگومان رمزگذاری فایل را مشخص می‌کند (رمزگذاری‌های استاندارد پایتون).
`parse_dates`	این آرگومان مقدار بولی یا لیستی از نام‌ها یا شماره ستون‌ها می‌گیرد و مشخص می‌کند که آن ستون‌ها باید به‌عنوان نوع تاریخ شناخته شوند. اگر مقدار `True` داشته باشد نمایه‌های دیتافریم را به‌عنوان تاریخ در نظر می‌گیرد.

▼ دیتافریم چیست؟

در دنیای واقعی معمولاً **مجموعه‌داده‌ها (Datasets)** به‌صورت جدول هستند و نباید با آن‌ها تک‌بعدی رفتار کرد؛ به همین دلیل، معمولاً در پانداز، از **ساختار دیتافریم (DataFrame)** استفاده می‌کنیم. این ساختار بسیار شبیه به جدول‌های مجموعه‌داده‌ها طراحی شده است.

برای ساخت یک دیتافریم در پانداز می‌توان از یک دیکشنری برای مقداردهی اولیه‌ی آن استفاده کرد. به مثال زیر توجه کنید:

```
import pandas as pd
import numpy as np

cars_dictionary = {
    "model" :[ "Peykan", "BMW X6", "Peugeot 206", "Tiba", "Nissan Junior", "Pe
    "Color" :["Red", "White", "Black", "White", "Blue", "Red", "Silver"],
    "HP" :[48, 335, 105, 87, 93, 80, 48],
    "Weight (KG)": [990, 2320, 1025, 1027, 1540, 960, 850],
    "price" :[33, 2400, 105, 120, 230, 68, 38]
}

df = pd.DataFrame(cars_dictionary)

df
```

	model	Color	HP	Weight (KG)	price
0	Peykan	Red	48	990	33
1	BMW X6	White	335	2320	2400
2	Peugeot 206	Black	105	1025	105
3	Tiba	White	87	1027	120
4	Nissan Junior	Blue	93	1540	230
5	Peykan Vanet	Red	80	960	68
6	renault 5	Silver	48	850	38

در سلول بالا، از دیکشنری `cars_dictionary` برای ساخت دیتافریم استفاده کردیم که این مقادیر، همان‌طور که از اسم آن پیداست، اطلاعات ۷ ماشین در یک نمایشگاه اتومبیل را شامل می‌شود. اما همان‌طور که می‌بینید، کلیدهای این دیکشنری، **به عنوان اسم سرستون‌ها قرار داده شده‌اند**.

انواع ویژگی‌های دسته‌ای

ویژگی‌های دسته‌ای به دو گروه اصلی زیر تقسیم می‌شوند:

۱. **اسمی (Nominal)**: متشکل از دو یا چند دسته که هیچ ترتیبی بین مقادیر آن‌ها وجود ندارد. برای مثال، «جنسیت» یک متغیر اسمی است.
۲. **ترتیبی (Ordinal)**: ترتیب خاصی بین مقادیر گروه‌ها وجود دارند. برای مثال، گروه‌های سنی (جوان، میان‌سال و پیر) یک متغیر ترتیبی است.

روش‌های تبدیل داده‌های دسته‌ای

می‌خواهیم ستون‌های دسته‌ای مجموعه‌داده‌ی خانه‌های شهر پکن را به اعداد تبدیل کنیم. ابتدا به سراغ متغیرهای اسمی می‌رویم.

پیش از هر چیز نگاهی به مقادیر ستون `renovationCondition` بیندازیم:

```
df['renovationCondition'].value_counts()
```

```
other         118740
hardcover     117438
Simplicity     77251
rough           5390
Name: renovationCondition, dtype: int64
```

ساده‌ترین راه برای تبدیل ستون‌های اسمی به عددی، استفاده از یک دیکشنری برای تبدیل هر دسته به یک عدد است. در این روش، دسته‌ها با استفاده از یک دیکشنری به اعداد `0` تا `n-1` نگاشت می‌شوند. (`n` تعداد دسته‌ها است). با کد زیر می‌توانیم ستون `renovationCondition` را به عدد تبدیل کنیم:

```
mapping = {'other': 0,
           'hardcover' : 1,
           'Simplicity' : 2,
           'rough' : 3}

df['Mapping'] = df['renovationCondition'].map(mapping)

df['Mapping'].value_counts()
```

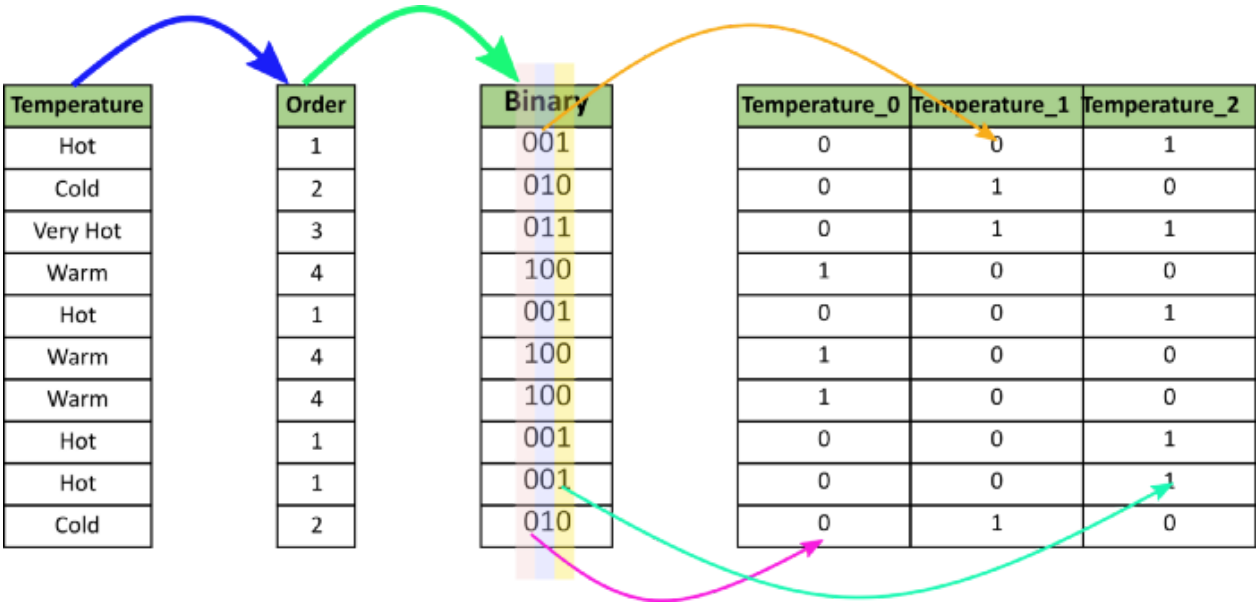
```
0    118740
1    117438
2     77251
3      5390
Name: Mapping, dtype: int64
```

این روش تبدیل ویژگی‌های دسته‌ای، **کدگذاری برچسبی (Label Encoding)** نامیده می‌شود. به کمک کلاس `LabelEncoder` در کتابخانه‌ی `scikit-learn` نیز می‌توانیم همین کار را به شکل ساده‌تر انجام دهیم:

```python
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

df['LabelEncoding'] = label_encoder.fit_transform(df['renovationCondition'])

df['LabelEncoding'].value_counts()
```

```
2    118740
1    117438
0     77251
3      5390
Name: LabelEncoding, dtype: int64
```

یک راه‌حل برای رفع این مشکل، باینری کردن اعداد تولید شده و استفاده از تکنیک **کدگذاری باینری (Binary Encoding)** است. تبدیل اعداد به فرم باینری، کاری ساده است که در شکل زیر قابل مشاهده است:



Temperature
Hot
Cold
Very Hot
Warm
Hot
Warm
Warm
Hot
Hot
Cold

Order
1
2
3
4
1
4
4
1
1
2

Binary
001
010
011
100
001
100
100
001
001
010

Temperature_0	Temperature_1	Temperature_2
0	0	1
0	1	0
0	1	1
1	0	0
0	0	1
1	0	0
1	0	0
0	0	1
0	0	1
0	1	0

اما در عمل به جای نمایش اعداد به فرم مبنای دو، آنها را به فرم **One-Hot** ذخیره می‌کنند. در این روش، در هر سطر فقط یکی از ستون‌های مرتبط با یک ویژگی دسته‌ای مقدار `1` دارد و بقیه ستون‌ها `0` هستند.

بیایید، ستون `renovationCondition` را بصورت One-Hot کدگذاری کنیم. برای این کار می‌توانیم از تابع `get_dummies` کتابخانه‌ی `pandas` کمک بگیریم.

```
dummies = pd.get_dummies(df['renovationCondition'])
dummies.head()
```

```
   Simplicity  hardcover  other  rough

0           1          0      0      0
1           0          1      0      0
2           1          0      0      0
3           0          0      1      0
4           0          0      0      1
```

اگر بخواهیم حاصل این کدگذاری را به دیتافریم اصلی خود اضافه کنیم می‌توانیم از کد زیر کمک بگیریم:

```
encoded_df = pd.concat([df, dummies], axis=1)
```

برای ویژگی‌های ترتیبی، به جای استفاده از روش کدگذاری One-Hot، می‌توان از اعداد به شکل معناداری نیز استفاده کرد. برای مثال در ستون «دسته قیمت»، می‌توان از ۱ برای نمایش دسته `cheap` ، از ۲ برای نمایش دسته `average` ، از ۳ برای نمایش دسته `expensive` و از ۴ برای نمایش دسته `very expensive` استفاده کرد.

تقسیم‌بندی مجموعه‌داده

به کمک کتابخانه‌ی `scikit-learn` می‌توانید به راحتی مجموعه‌داده‌های خود را تقسیم‌بندی کرده و از تکنیک‌های رایجی نظیر بُر زدن یا طبقه‌بندی نیز استفاده کنید.

فرض کنید که ویژگی‌های نمونه‌ها را در متغیر `X` و خروجی‌ها (مثل برچسب‌ها) را در متغیر `y` ذخیره کرده باشید. آن‌گاه کافیست از تابع `train_test_split` استفاده کنید تا این مجموعه را با درصد دلخواه شما به دو بخش تقسیم کند. آرگومان `test_size` را می‌توانید معادل درصد نمونه‌های آزمون (`float`) یا تعداد آن‌ها (`int`) تنظیم کنید. همچنین اگر قصد بُر زدن مجموعه را دارید می‌توانید آرگومان `shuffle` را `True` کنید.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle
```

توجه داشته باشید که با کد بالا، مجموعه‌داده به دو بخش مجموعه‌های آموزش و آزمون تقسیم شد.

مصورسازی داده

برای استفاده از کتابخانه‌ی مت‌پلات، رابط (interface) اصلی آن که `pyplot` نام دارد را به شکل زیر `import` می‌کنیم.

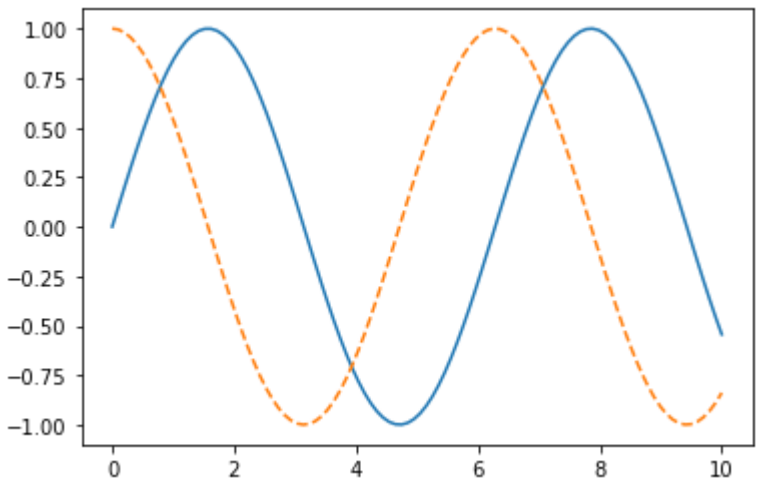
```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(1400)
```

همچنین در صورت اجرای برنامه در نت‌بوک ژوپیتر دستور `%matplotlib inline` را در ادامه قرار می‌دهیم تا خروجی نمودارهایی که رسم می‌کنیم به شکل عکس داخل نت‌بوک رسم شود (در اسکریپت پایتون می‌توانیم از تابع `plt.show` برای نمایش خروجی پس از رسم شکل و در پایتون تعاملی از `%matplotlib` در ابتدای اجرا برای تنظیم نمایش استفاده کنیم).

```
%matplotlib inline
```

حال با اجرای کد زیر، اولین نمودارمان را رسم می‌کنیم.

```
x = np.linspace(0, 10, 100)
plt.plot(x, np.sin(x), linestyle='solid')
plt.plot(x, np.cos(x), linestyle='dashed');
```



همان‌طور که در شکل حاصل می‌بینید، مقدار دو تابع مثلثاتی سینوس و کسینوس را برای مقادیر بین ۰ تا ۱۰ در یک صفحه‌ی مختصات رسم کرده‌ایم. در ادامه با نحوه‌ی ساخت شکل‌هایی از این دست آشنا می‌شویم.

رسم نمودار میله‌ای با رابط پانداز

رابط پانداز برای ساده‌تر کردن رسم نمودارها طراحی شده است و به کمک آن مستقیماً می‌توانید بر روی دیتافریم، توابع رسم مورد نیاز خود را صدا بزنید.

▼ تبدیل مجموعه‌داده شهر تبریز به دیتافریم

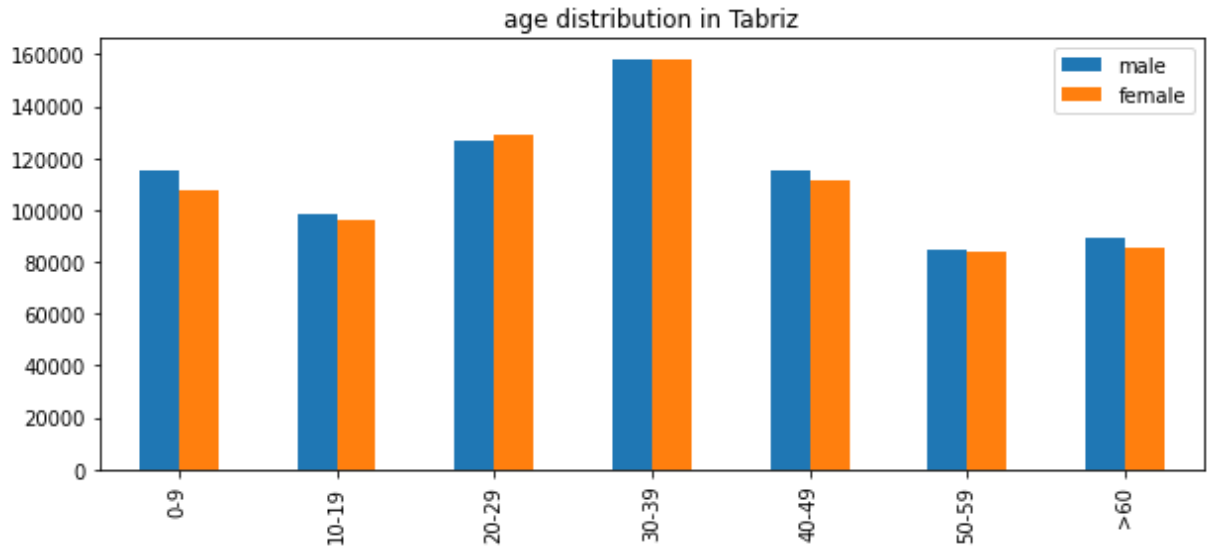
ابتدا داده‌های شهر تبریز را به یک دیکشنری تبدیل می‌کنیم؛ سپس با این دیکشنری، یک دیتافریم می‌سازیم:

```
import pandas as pd

ages = ['0-9','10-19','20-29','30-39','40-49','50-59','>60']
data = {'male':[115165, 98484, 126469, 157612, 115085, 84919, 88927],
        'female':[107770,  96027, 129288, 158169, 111618,  83929,  85231]}
df = pd.DataFrame(data, index = ages)
df
```

	male	female
0-9	115165	107770
10-19	98484	96027
20-29	126469	129288
30-39	157612	158169
40-49	115085	111618
50-59	84919	83929
>60	88927	85231

در این قسمت، مجموعه‌داده‌ی جمعیتی ما در قالب دیتافریم ذخیره شده است و می‌خواهیم نمودار میله‌ای را به تفکیک جنسیت رسم کنیم.

```
fig , ax = plt.subplots(figsize = (10,4))
df.plot(ax = ax, kind = 'bar')
ax.set_title("age distribution in Tabriz");
```



این نمودار، **نمودار میله‌ای گروهی (Grouped Bar Plot)** نام دارد که در آن میله‌ها از صفر شروع شده‌اند.

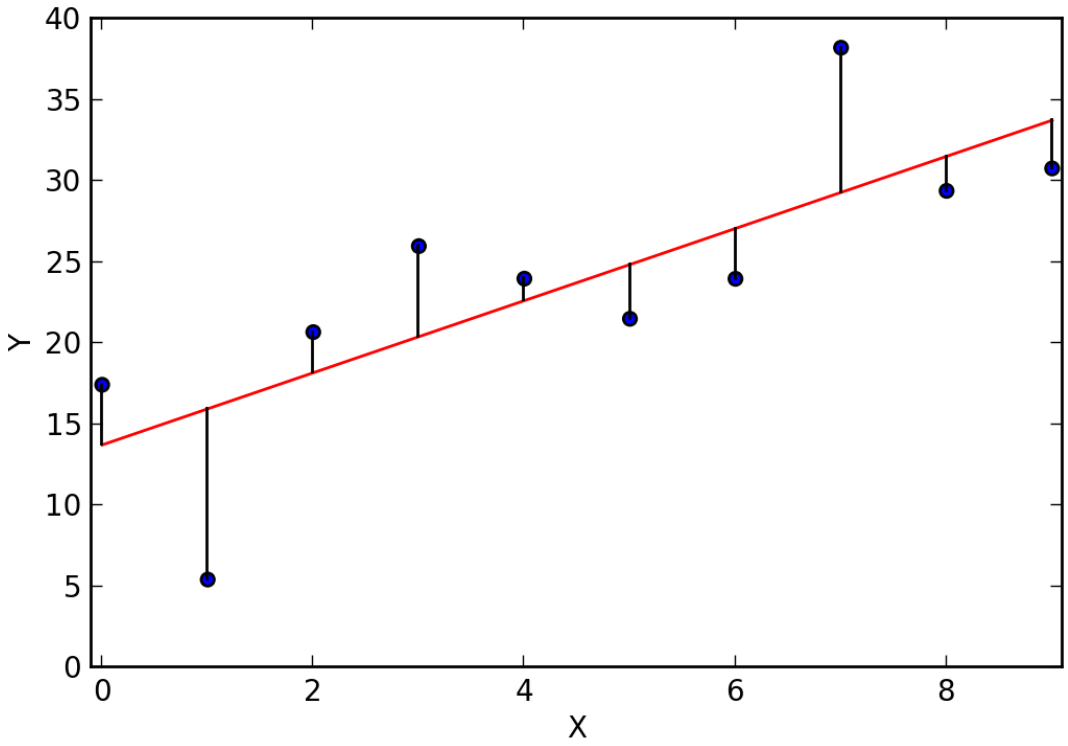
ارزیابی

معیار Mean Absolute Error (MAE)

در معیار میانگین خطای مطلق، ابتدا قدر مطلق اختلاف تمام مقادیر پیش‌بینی‌شده با مقادیر واقعی حساب شده و پس از جمع کردن تمام مقادیر خطا، با تقسیم بر تعداد مشاهدات، میانگین گرفته می‌شود. برای فهم بهتر به فرمول زیر توجه فرمایید:

فرمول ۱
$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid$

در این فرمول n تعداد داده‌ها یا مشاهدات، yi مقدار داده واقعی i ام و ŷi مقدار داده پیش‌بینی شده i ام می‌باشد.



همانطور که مشاهده می‌شود، فاصله‌ی بین هر نقطه داده واقعی با داده پیش‌بینی شده حساب شده و در آخر تمام این فواصل با هم جمع و تقسیم بر تعداد نقاط داده (در این مثال ۱۰) می‌شود.

مثال این معیار با استفاده از کتابخانه `scikit-learn` در زیر آورده شده است:

```
from sklearn.metrics import mean_absolute_error
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
mean_absolute_error(y_true, y_pred)
```

```
0.5
```

معیار Mean Squared Error (MSE)

معیار MSE شباهت بسیار زیادی به معیار MAE دارد. اما به جای قدرمطلق گرفتن فاصله بین دو نمونه، مجذور (توان دوم) فاصله بین دو نمونه را اندازه‌گیری می‌کند. فرمول این معیار در زیر آورده شده است.

فرمول ۲

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

در این فرمول n تعداد داده‌ها یا مشاهدات، yi مقدار داده واقعی i ام و ŷi مقدار داده پیش‌بینی شده i ام می‌باشد.

معیار MAE نسبتاً به داده‌های پرت مقاوم است؛ به این معنی که هنگام وجود داده‌های پرت، این معیار نسبت به MSE مقدار خطای کمتری را نشان می‌دهد. زیرا در صورت وجود داده‌ی پرت، به دلیل به توان رسیدن میزان اختلاف مقدار واقعی و پیش‌بینی شده، معیار MSE مقدار خطای زیادتری را به ما نشان می‌دهد.

در مثال زیر با استفاده از `scikit-learn` به محاسبه این معیار پرداخته‌ایم:

```
from sklearn.metrics import mean_squared_error
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
mean_squared_error(y_true, y_pred)
```

```
0.375
```

معیار Root Mean Squared Error (RMSE)

با جذر گرفتن از MSE، معیار RMSE به‌دست می‌آید. یکی از مزیت‌های استفاده از معیار RMSE این است که واحد اندازه‌گیری داده‌ها با این معیار یکسان می‌باشد. به طور مثال اگر واحد اندازه‌گیری داده‌ها متر

باشد، پس به‌دست آوردن معیار MSE، به متر مربع تبدیل شده و با گرفتن جذر، دوباره به متر تبدیل می‌شود. در نتیجه فرمول این معیار به شکل زیر خواهد بود:

فرمول ۳

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_{pred,i} - y_i)^2}{n}}$$

معیار R Squared (R2)

این معیار، عملکرد مدل ما را با عددی بین ۰ و ۱ نشان می‌دهد و با معیارهای پیشین تقریباً متفاوت است. به فرمول زیر توجه کنید:

فرمول ۴

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$$

این معیار نیز در `scikit-learn` قابل دسترسی است.

```
from sklearn.metrics import r2_score
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
r2_score(y_true, y_pred)
```

```
0.9486081370449679
```

برگرفته از کوئرا کالج‌های ماشین لرنینگ ۱ و ماشین لرنینگ ۲