پیش پردازش داده ها

شروع هر نوع کار و عملیاتی در مرحله اول، دارای یک سری مقدمات و پیش‌نیازها است.

(Data Mining)نیز از این قانون مستثنی نبوده و نیازمند آماده‌سازی و پردازش‌های مقدماتی است. در علم داده‌کاوی، تمامی داده‌هایی که برای هدف مورد نظر استفاده خواهند شد، باید پیش از شروع پردازش با استفاده از روش‌هایی، آماده و تنظیم و یا به اصطلاح «پیش‌پردازش (Preprocess) «شوند. مرحله آماده‌سازی داده‌ها قبل از پردازش را، پیش‌پردازش (Preprocessing) می‌گویند. پیش‌پردازش نقشی اساسی در روند پردازش داده‌ها و نتایج حاصل از آن‌ها ایفا می‌کند. برای پیش پردازش داده ها مراحل و ابزارهای مختلفی وجود دارند. برخی از مهم‌ترین مواردی که طی فرایند پیش‌پردازش داده‌ها باید به آن‌ها پرداخته شود، در ادامه بیان شده‌اند.

پیش پردازش داده ها: داده های ناموجود

در برخی موارد، ممکن است بعضی از ویژگی‌های مربوط به یک یا چند نمونه، فاقد مقادیر معتبر باشند. این امر می‌تواند دلایل مختلفی داشته باشد، از جمله نویزی (Noise) بودن داده‌های ثبت شده، عدم ثبت و یا نامعتبر بودن مقدار آن. این داده‌ها را داده‌های ناموجود | Null Data | Missing Data) داده‌های گم‌شده(، می‌نامند.

برای پردازش مجموعه داده‌هایی که دچار چالش داده ناموجود هستند، باید راهکاری برای تعیین مقادیر داده‌های ناموجود یافت. روش‌های گوناگونی برای مدیریت داده‌های ناموجود وجود دارند که در ادامه به چند نمونه از آن‌ها اشاره شده است.

حذف نمونه :در این روش، تمام نمونه‌هایی (Sample) که دارای مقدار ویژگی ناموجود هستند، به کلی از مجموعه داده حذف شده و در روند پردازش مورد استفاده قرار نمی‌گیرند. این روش بیشتر در مواردی

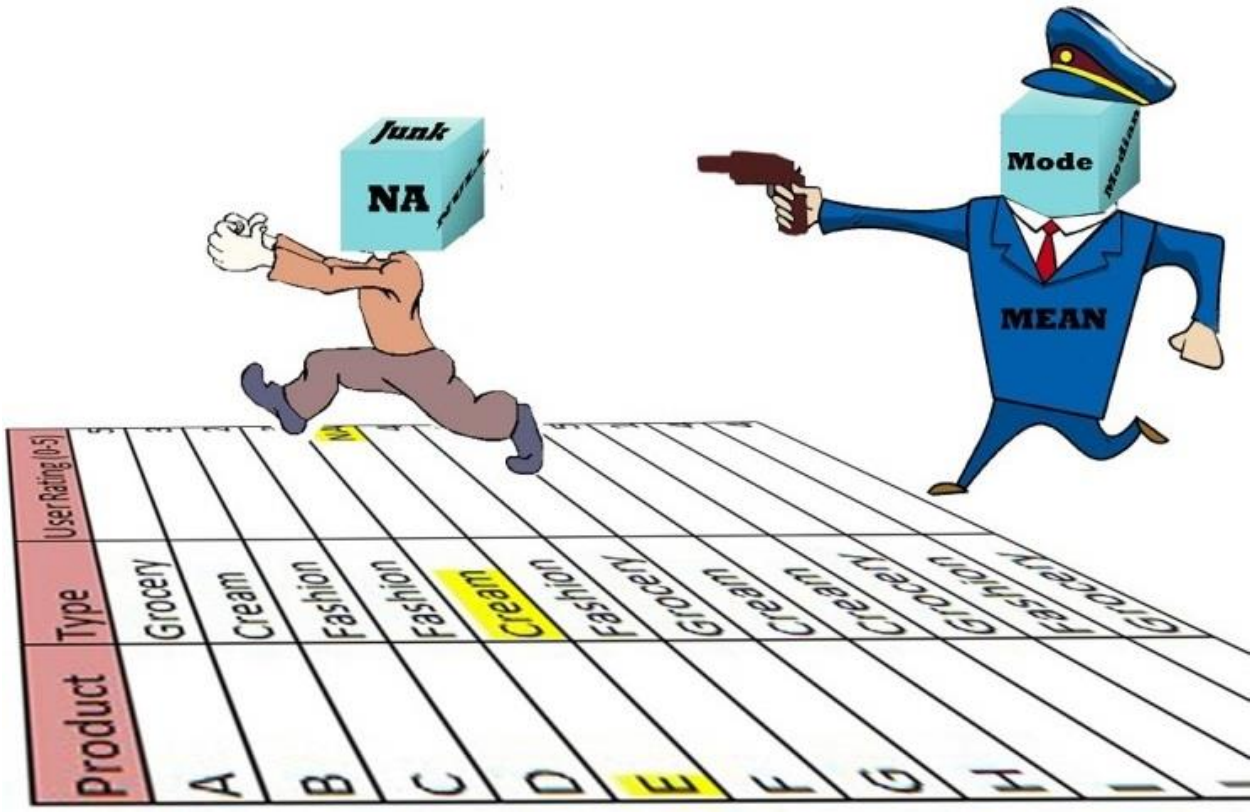
به کار می‌رود که تعداد نمونه‌های موجود، بسیار زیاد است و یا ویژگی جا افتاده، مربوط به برچسب دسته است. اما در حالت کلی، این روش به دلیل حذف داده‌های موجود، روش مناسبی نبوده و زیاد مورد استفاده قرار نمی‌گیرد.

پر کردن دستی :در این روش، مقادیری که هنگام آماده‌سازی مجموعه داده، جا افتاده و یا نادرست ثبت شده‌اند به صورت دستی پر می‌شوند. این روش زمانی امکان‌پذیر است که منبع جمع‌آوری داده‌ها در دسترس باشد؛ در عین حال، به دلیل زمان‌گیر بودن این راهکار، در مواقعی که سایز مجموعه داده بزرگ است و یا تعداد داده‌های مفقود زیاد هستند، مقرون به صرفه نبوده و عملا استفاده از این روش غیر ممکن است.

استفاده از یک مقدار ثابت :در این روش تمام مقادیر ناموجود در مجموعه داده، با یک مقدار ثابت و از پیش تعیین شده پر می‌شوند. این روش از دقت بالایی برخوردار نیست؛ به ویژه اگر تعداد داده‌های ناموجود زیاد باشد، ممکن است این مقدار ثابت جاگذاری شده، اطلاعات معنادار اشتباهی را به داده‌ها اضافه کند که در روند پردازش تاثیرگذار باشد. مثلا اگر تعداد زیادی از داده‌های مربوط به یک ویژگی

(Feature)، در مجموعه داده ناموجود باشند، ثبت یک مقدار ثابت برای تمامی نمونه‌هایی که مقدارشان موجود نیست، می‌تواند در روند یک پردازش، دسته‌بند را دچار اشتباه کند.

استفاده از مقدار میانگین ویژگی :در این روش برای هر ویژگی، مقدار میانگین، با استفاده از نمونه‌هایی که مقادیرشان معلوم است محاسبه شده و برای جاگذاری مقادیر مربوط به آن ویژگی مورد استفاده قرار می‌گیرد.



استفاده از مقدار میانگین ویژگی به تفکیک دسته :این روش مشابه روش قبلی است، با این تفاوت که مقدار میانگین، برای نمونه‌های هر دسته، به صورت جداگانه محاسبه می‌شود. میانگین‌های به دست

آمده برای هر دسته، جهت پرکردن داده‌های ناموجود نمونه‌های متعلق به همان دسته مورد استفاده قرار می‌گیرند. این روش در مقایسه با روش قبلی کارایی بیشتری دارد.

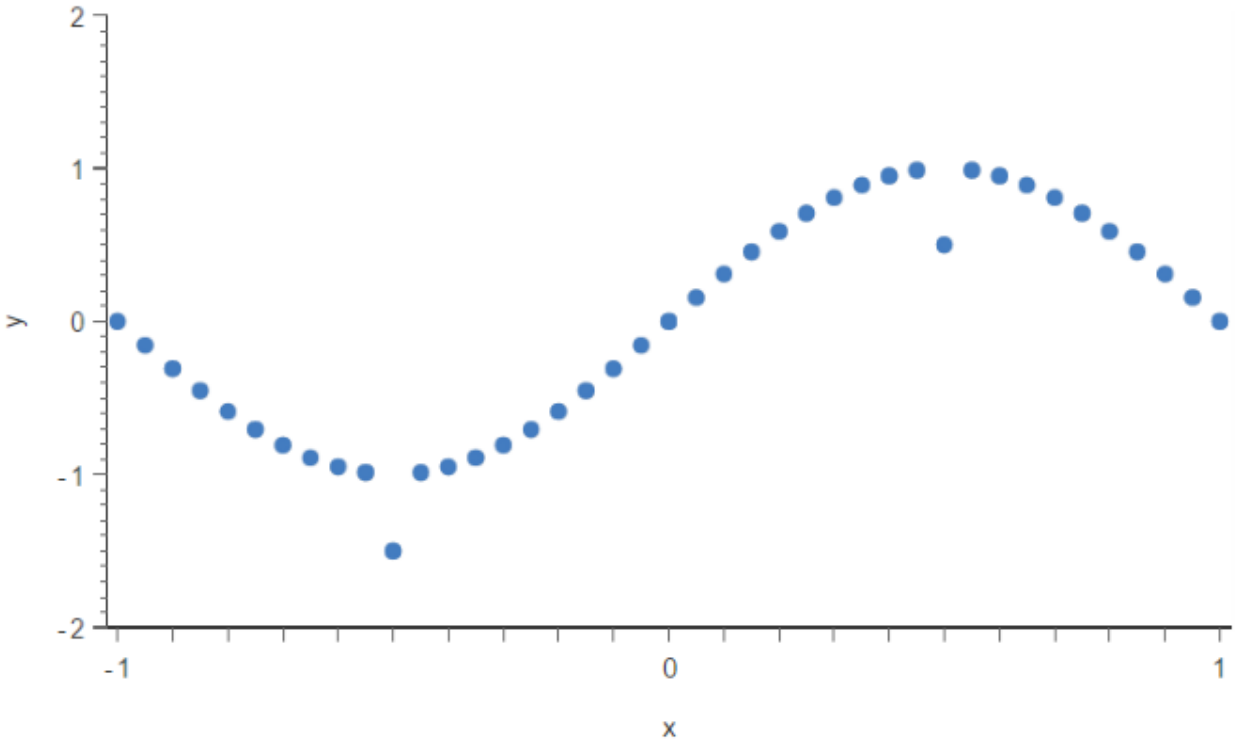
استفاده از محتمل‌ترین مقدار :اساس این روش فراوانی مقادیر موجود در هر ویژگی است. در این روش، فراوانی مقادیر ظاهر شده در هر ویژگی به دست می‌آیند و مقداری که بالاترین فراوانی را در بین مقادیر دارد به عنوان مقدار جایگزین انتخاب می‌شود. این مقدار منتخب، جایگزین داده‌های مفقود در آن ویژگی می‌شود.

طراحی پیش‌بینی کننده :بهترین و دقیق‌ترین روش برای پرکردن داده‌های ناموجود در یک مجموعه داده، طراحی یک پیش‌بینی کننده است. در این روش یک پیش‌بینی کننده، متناسب با ماهیت داده‌ها انتخاب و طراحی می‌شود، تا مقادیر داده‌های گمشده مربوط به یک ویژگی را بر اساس سایر ویژگی‌ها تخمین بزند.

پیش پردازش داده ها: تحلیل داده‌های پرت

در برخی موارد، ممکن است به دلایلی، مقادیری در مجموعه داده ظاهر شوند که تفاوت زیاد و غیر معمولی با سایر مقادیر موجود در مجموعه داشته باشند، این داده‌ها را داده‌های پرت می‌گویند. در واقع منظور از داده‌های پرت، داده‌ها یا نمونه‌هایی هستند که با رفتار کلی، یا مدل مجموعه کل داده‌ها شباهت نداشته و از آن تبعیت نمی‌کنند.

به عنوان مثال، در مجموعه داده مربوط به قد افراد مونث، که بازه قدی آن‌ها بین ۱۵۰ تا ۱۷۵ سانتی‌متر است، قد ۲۰۵ سانتی‌متر، یک داده پرت محسوب می‌شود. آنالیز و مدیریت داده‌های پرت یکی از مهمترین مراحل پیش‌پردازش داده‌ها محسوب می‌شود، زیرا این داده‌ها می‌توانند عملکرد الگوریتم‌ها و «دسته‌بندهای (Classifier) «مورد استفاده را دچار مشکل کنند. بنابراین بایستی تا حد ممکن، تاثیر داده‌های پرت را کاهش داد. روش‌های مختلفی برای شناسایی و مدیریت این داده‌ها وجود دارد که در ادامه به چند روش ساده اشاره شده است.



حذف :ساده‌ترین روش مدیریت داده‌های پرت، حذف آن‌ها است. برای این منظور ابتدا لازم است داده‌های پرت شناسایی شوند. روش متداولی که مورد استفاده قرار می‌گیرد، به این صورت است که

ابتدا بازه‌ای را که داده‌ها در آن قرار دارند مشخص می‌کنند، سپس تعداد معینی از داده‌های بالا و پایین این بازه را به عنوان داده پرت در نظر گرفته و آن‌ها را از مجموعه داده حذف می‌کنند.

جایگذاری با میانگین :یک روش دیگر که در مقایسه با روش قبلی دقت مناسب‌تری دارد، استفاده از مقدار میانگین داده‌ها است. در این روش، عمل مشخص کردن داده‌های پرت به روشی که در بالا ذکر شد، بر اساس بازه داده‌ها مشخص می‌شود. اما تفاوتی که با روش قبلی دارد این است که به جای حذف داده‌های پرت، مقدار میانگین مربوط به داده‌های غیر پرت را به دست آورده و آن را جایگزین داده‌های پرت می‌کند. با این کار تمامی داده‌ها در یک بازه مشخص و معقول قرار می‌گیرند. البته این روش، مشکل «سوگیری (Bias) «در داده‌ها را به دنبال دارد.

نرمال‌سازی داده‌ها

نرمال‌سازی داده‌ها از جمله مهمترین مراحل پیش‌پردازش در علم داده‌کاوی است. در توضیح اهمیت نرمال‌سازی، تصور کنید بازه مربوط به مقادیر دو ویژگی، تفاوت عمده‌ای نسبت به یکدیگر داشته باشند؛ به عنوان مثال فرض کنید بازه مربوط به یکی از ویژگی‌ها بازه [۱ , ۰] و بازه مربوط به ویژگی دیگری از همین مجموعه داده، [۵۰۰ , ۱] باشد، در چنین شرایطی واضح است که هنگام استفاده از معیارهایی که مبتنی بر فاصله هستند، ویژگی با بازه کوچکتر عملا تاثیر محسوسی در محاسبات نخواهد داشت.

با توجه به این توضیحات، برای دستیابی به نتایج دقیق‌تر، لازم است که بازه مربوط به ویژگی‌های مختلف، به نحوی با یکدیگر یکسان و یا نزدیک شوند. برای این منظور از روش‌های نرمال‌سازی استفاده می‌شود. روش‌های مختلفی برای نرمال‌سازی وجود دارد که در مقالات و تحقیقات مورد استفاده قرار می‌گیرد. در ادامه به چند نمونه از متداول‌ترین روش‌های نرمال‌سازی اشاره شده است.

نرمال‌سازی مین-ماکس(Min-Max)

در این روش ساده، هر مجموعه‌ای از داده‌ها به بازه‌ای دلخواه، که کمترین و بیشترین مقدار آن از قبل مشخص است نگاشت می‌شود. در این روش می‌توان هر بازه دلخواه را تنها با یک تبدیل ساده، به بازه‌ای جدید نگاشت کرد. فرض کنید قرار است ویژگیA ، از مجموعه داده که در بازه بین min_A تا max_Aقرار دارد، به بازه جدید new_Min تا new_Max نگاشت شود. برای این منظور، هر مقدار اولیه مانند v در بازه اولیه، طبق رابطه زیر به مقدار جدید v′ در بازه جدید تبدیل خواهد شد:

$$v'=(v-minA)newMax-newMinmaxA-minA+(newMin)v'=(v-min_A$$

$$)maxA-minAnewMax-newMin+(newMin)$$

نرمال‌سازی نمره زد(Z-Score)

یکی دیگر از پرکاربردترین و مهم‌ترین روش‌های نرمال‌سازی روش نمره زد (Z-Score) است. پایه این روش بر خلاف روش مین-ماکس، بر اساس میانگین و انحراف معیار داده‌ها است. ویژگی A را در نظر بگیرید، فرض کنیدm ، میانگین داده‌ها و σ نشان‌گر انحراف معیار آن‌ها است. با این مفروضات، مقادیر

جدید برای این مجموعه داده، بر اساس مقدار میانگین و انحراف معیار، با استفاده از رابطه زیر محاسبه می‌شوند:

$$v'=v-m\sigma v'=\sigma v-m$$

شایان توجه است که فارغ از بازه اولیه مجموعه داده مورد بررسی، مقدار میانگین داده‌های حاصل از نرمال‌سازی به روشZ-Score ، همواره برابر با صفر و انحراف معیار آن‌ها عدد یک خواهد بود.

مقیاس گذاری اعشاری(Decimal Scaling)

مقیاس گذاری اعشاری نوع دیگری از روش‌های نرمال‌سازی است که در واقع منطق آن تغییر نقطه اعشار مقادیر موجود در مجموعه داده است. در این روش، نرمال‌سازی با استفاده از رابطه زیر انجام می‌شود:

$$v'=v۱۰jv'=۱۰jv$$

که در آن، jبرابر با کوچکترین مقداری است که شرط زیر را فراهم کند:

max(|v'|)<۱

با اندکی دقت در روابط بالا، مشخص می‌شود که مقدارj ، کاملا بستگی به کران‌های مربوط به بازه مجموعه داده اولیه دارد. به عنوان مثال، اگر بازه مربوط به مقادیر یک ویژگی از مجموعه‌داده‌ها، [۹۷ , ۹۸۶-] باشد، با توجه به این که بزرگترین مقدار قدرمطلق این بازه، ۹۸۶ است، کوچکتری مقدار j که این

عدد را به اعشار کمتر از یک ببرد، ۳ است. بنابراین بر اساس رابطه‌ای که در بالا ارائه شد، با تقسیم تمام مقادیر اولیه به ۱۰۳، بازه داده‌های جدید، [۰/۹۷ , ۰/۹۸۶-] خواهد بود.

روش‌های ارائه شده در بالا، از مهمترین مراحل پیش‌پردازش داده‌ها هستند، که علاوه بر اینکه به عنوان مرحله‌ای مهم از مراحل داده‌کاوی و به منظور افزایش دقت نتایج، مورد استفاده قرار می‌گیرند، هر یک به تنهایی به عنوان مبحثی مهم، مورد توجه پژوهشگران هستند.

پیش پردازش داده یکی از مراحل حیاتی در فرایند مدل‌سازی داده‌ها است. در ادامه مراحل مختلف پیش پردازش داده را با توضیحات کامل ارائه می‌دهم:

۱. **جمع‌آوری داده‌ها:(Data Collection)**
   - **توضیح** :داده‌ها از منابع مختلف جمع‌آوری می‌شوند. این منابع می‌توانند شامل دیتابیس‌ها، فایل‌هایCSV ، APIها، وب‌سایت‌ها و غیره باشند.
۲. **وارد کردن داده‌ها:(Data Importing)**
   - **توضیح** :داده‌های جمع‌آوری شده به محیط پایتون وارد می‌شوند. این مرحله شامل خواندن داده‌ها از فایل‌هایCSV ، Excel، دیتابیس‌ها و غیره است.
   - **مثال پایتون:**

```
python
Copy code
import pandas as pd

df = pd.read_csv('data.csv')
```

۳. **بررسی داده‌ها:(Data Exploration)**
   - **توضیح** :بررسی اولیه داده‌ها برای فهمیدن ساختار، نوع داده‌ها، مقادیر گم شده و الگوهای کلی.
   - **مثال پایتون:**

```
python
Copy code
print(df.head())
```

```
print(df.info())
print(df.describe())
```

٤. **پاک‌سازی داده‌ها:(Data Cleaning)**
   - **توضیح** :این مرحله شامل حذف یا جایگزینی مقادیر گم شده، حذف داده‌های تکراری و تصحیح خطاهای موجود در داده‌ها است.
   - **مثال پایتون:**

```
python
Copy code
#حذف مقادیر گم شده
df.dropna(inplace=True)

#جایگزینی مقادیر گم شده با میانگین ستون
df.fillna(df.mean(), inplace=True)

#حذف داده های تکراری
df.drop_duplicates(inplace=True)
```

٥. **تبدیل داده‌ها:(Data Transformation)**
   - **توضیح** :شامل استانداردسازی یا نرمال‌سازی داده‌ها، تبدیل داده‌های متنی به داده‌های عددی و ایجاد ویژگی‌های جدید است.
   - **مثال پایتون:**

```
python
Copy code
#استانداردسازی داده ها
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)

#تبدیل داده های متنی به داده های عددی
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['category'] = le.fit_transform(df['category'])
```

٦. **تقسیم داده‌ها به مجموعه‌های آموزشی و تست:(Train-Test Split)**
   - **توضیح** :تقسیم داده‌ها به دو مجموعه آموزشی و تست برای ارزیابی مدل.

- **مثال پایتون:**

```
python
Copy code
from sklearn.model_selection import
train_test_split

X = df.drop('target', axis=۱)
y = df['target']

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=۰,۲,
random_state=٤۲)
```

۷. **انتخاب ویژگی‌ها:(Feature Selection)**
   - **توضیح** :انتخاب ویژگی‌های مهم و حذف ویژگی‌های غیرضروری یا کم اهمیت.
   - **مثال پایتون:**

```
python
Copy code
from sklearn.feature_selection import SelectKBest,
f_classif

selector = SelectKBest(score_func=f_classif, k=۱۰)
X_new = selector.fit_transform(X_train, y_train)
```

۸. **مهندسی ویژگی‌ها:(Feature Engineering)**
   - **توضیح** :ایجاد ویژگی‌های جدید از داده‌های موجود برای بهبود عملکرد مدل.
   - **مثال پایتون:**

```
python
Copy code
#ایجاد ویژگی جدید بر اساس ویژگی‌های موجود
df['new_feature'] = df['feature۱'] * df['feature۲']
```

با انجام این مراحل، داده‌ها آماده مدل‌سازی می‌شوند و می‌توانید به مرحله بعدی یعنی انتخاب و آموزش مدل بروید.

درمورد داده های گم شده یا nan ها

داده‌های گم شده (Missing Data) یا NaN (Not a Number) یکی از چالش‌های رایج در پیش پردازش داده‌ها است. نحوه برخورد با داده‌های گم شده می‌تواند تاثیر زیادی بر دقت و کارایی مدل‌های یادگیری ماشین داشته باشد. در ادامه، روش‌های مختلف برخورد با داده‌های گم شده را توضیح می‌دهم و کدهای مرتبط با هر روش را ارائه می‌کنم:

۱. **شناسایی داده‌های گم شده:**
   - **توضیح** :ابتدا باید داده‌های گم شده را شناسایی کنید تا بدانید که در کدام قسمت‌ها و به چه میزان داده‌های گم شده وجود دارد.
   - **مثال پایتون**:

```
python
Copy code
import pandas as pd

df = pd.read_csv('data.csv')

#شناسایی داده های گم شده
print(df.isnull().sum())
```

۲. **حذف داده‌های گم شده:**
   - **توضیح** :می‌توانید سطرها یا ستون‌هایی که داده‌های گم شده دارند را به طور کامل حذف کنید. این روش زمانی مفید است که تعداد داده‌های گم شده کم باشد.
   - **مثال پایتون**:

```
python
Copy code
#حذف سطرهایی که داده های گم شده دارند
df.dropna(inplace=True)

#حذف ستون‌هایی که داده های گم شده دارند
df.dropna(axis=۱, inplace=True)
```

۳. **جایگزینی داده‌های گم شده با مقادیر آماری:**
   - **توضیح** :می‌توانید داده‌های گم شده را با مقادیری مثل میانگین، میانه، یا مد جایگزین کنید. این روش به حفظ بیشتر داده‌ها کمک می‌کند.
   - **مثال پایتون**:

```
python
Copy code
#جایگزینی داده های گم شده با میانگین ستون
```

```
df.fillna(df.mean(), inplace=True)

#جایگزینی داده‌های گم شده با میانه ستون
df.fillna(df.median(), inplace=True)

#جایگزینی داده‌های گم شده با مد ستون
df.fillna(df.mode().iloc[0], inplace=True)
```

٤. **استفاده از مدل‌های پیش‌بینی برای جایگزینی داده‌های گم شده:**
   - **توضیح**: می‌توانید از مدل‌های یادگیری ماشین برای پیش‌بینی و جایگزینی داده‌های گم شده استفاده کنید. این روش می‌تواند دقیق‌تر باشد، اما پیچیدگی بیشتری دارد.
   - **مثال پایتون**:

```
python
Copy code
from sklearn.impute import SimpleImputer

#جایگزینی داده‌های گم شده با میانگین ستون با استفاده از SimpleImputer
imputer = SimpleImputer(strategy='mean')
df_imputed = imputer.fit_transform(df)

#تبدیل دوباره به DataFrame
df = pd.DataFrame(df_imputed, columns=df.columns)
```

٥. **استفاده از روش‌های پیشرفته‌تر جایگزینی داده‌های گم شده:**
   - **توضیح**: روش‌های پیشرفته‌تری مثل K-Nearest Neighbors (KNN) یا Multiple Imputation by Chained Equations (MICE) می‌توانند برای جایگزینی داده‌های گم شده استفاده شوند.
   - **مثال پایتون**:

```
python
Copy code
from sklearn.impute import KNNImputer

#جایگزینی داده‌های گم شده با استفاده از KNNImputer
imputer = KNNImputer(n_neighbors=5)
df_imputed = imputer.fit_transform(df)

#تبدیل دوباره به DataFrame
```

```
df = pd.DataFrame(df_imputed, columns=df.columns)
```

در نهایت، انتخاب روش مناسب برای برخورد با داده‌های گم شده بستگی به نوع داده‌ها، میزان داده‌های گم شده و کاربرد خاص پروژه شما دارد.

درمورد داده ای پرت

داده‌های پرت (Outliers) نقاط داده‌ای هستند که به طور قابل توجهی با سایر نقاط داده در مجموعه داده تفاوت دارند. این داده‌ها می‌توانند ناشی از خطاهای اندازه‌گیری، ورود داده‌های نادرست یا پدیده‌های واقعی باشند. شناسایی و مدیریت داده‌های پرت اهمیت زیادی در تحلیل داده‌ها و مدل‌سازی دارد، زیرا می‌توانند بر عملکرد مدل‌های یادگیری ماشین تاثیر منفی بگذارند. در ادامه به توضیح روش‌های شناسایی و مدیریت داده‌های پرت می‌پردازم و کدهای مرتبط با هر روش را ارائه می‌کنم:

شناسایی داده‌های پرت

۱. **استفاده از نمودارها:**
   - **توضیح** :می‌توان از نمودارهایی مانند جعبه‌ای (Box Plot) و پراکندگی (Scatter Plot) برای شناسایی داده‌های پرت استفاده کرد.
   - **مثال پایتون:**

```
python
Copy code
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

#نمودار جعبه‌ای برای شناسایی داده‌های پرت
sns.boxplot(x=df['column_name'])
plt.show()

#نمودار پراکندگی برای شناسایی داده‌های پرت
sns.scatterplot(x=df['column_x'], y=df['column_y'])
plt.show()
```

۲. **استفاده از شاخص‌های آماری:**
   - **توضیح** :می‌توان از شاخص‌هایی مانند ضریب انحراف استاندارد و چارک‌ها (Quartiles)برای شناسایی داده‌های پرت استفاده کرد.
   - **مثال پایتون:**

```
python
Copy code
#شناسایی داده‌های پرت با استفاده از چارک‌ها
Q۱ = df['column_name'].quantile(۰,۲۵)
Q۳ = df['column_name'].quantile(۰,۷۵)
IQR = Q۳ - Q۱

#تعیین محدوده داده‌های پرت
lower_bound = Q۱ - ۱,۵ * IQR
upper_bound = Q۳ + ۱,۵ * IQR

outliers = df[(df['column_name'] < lower_bound) |
(df['column_name'] > upper_bound)]
print(outliers)
```

۳. **استفاده از مدل‌های آماری و یادگیری ماشین:**
   - **توضیح**: می‌توان از مدل‌هایی مانند Isolation Forest و Local Outlier Factor برای شناسایی داده‌های پرت استفاده کرد.
   - **مثال پایتون**:

```
python
Copy code
from sklearn.ensemble import IsolationForest

#استفاده از Isolation Forest برای شناسایی داده‌های
پرت
iso_forest = IsolationForest(contamination=۰,۰۵)
outliers =
iso_forest.fit_predict(df[['column_name']])

df['outlier'] = outliers
print(df[df['outlier'] == -۱])
```

مدیریت داده‌های پرت

۱. **حذف داده‌های پرت:**
   - **توضیح**: ساده‌ترین روش برای مدیریت داده‌های پرت حذف آنها است.
   - **مثال پایتون**:

```
python
```

```
Copy code
#حذف داده های پرت
df_cleaned = df[(df['column_name'] >= lower_bound)
& (df['column_name'] <= upper_bound)]
```

۲. **جایگزینی داده‌های پرت:**
   - **توضیح** :می‌توان داده‌های پرت را با مقادیر آماری مانند میانگین یا میانه جایگزین کرد.
   - **مثال پایتون:**

```
python
Copy code
#جایگزینی داده های پرت با میانگین
df['column_name'] = df['column_name'].apply(lambda
x: df['column_name'].mean() if x < lower_bound or x
> upper_bound else x)
```

۳. **تبدیل داده‌های پرت:**
   - **توضیح** :می‌توان داده‌های پرت را به نحوی تبدیل کرد که تاثیر منفی کمتری بر مدل داشته باشند.
   - **مثال پایتون:**

```
python
Copy code
#محدود کردن داده های پرت به محدوده مشخص
df['column_name'] = df['column_name'].apply(lambda
x: lower_bound if x < lower_bound else (upper_bound
if x > upper_bound else x))
```

انتخاب روش مناسب برای مدیریت داده‌های پرت بستگی به نوع داده‌ها و کاربرد خاص پروژه شما دارد. در برخی موارد، داده‌های پرت ممکن است اطلاعات مهمی در مورد پدیده‌های نادر داشته باشند که نباید نادیده گرفته شوند.

درمورد داده های object

در پایتون، داده‌های نوع `object`معمولا به رشته‌ها (strings) اشاره دارند، اما ممکن است شامل انواع دیگر داده‌ها مانند دسته‌بندی‌ها (categories) نیز باشند. برای کار با داده‌های `object`، معمولا نیاز به تبدیل آنها به فرمت‌های عددی داریم تا بتوانیم از آنها در مدل‌های یادگیری ماشین استفاده کنیم. در ادامه، روش‌های مختلف پیش پردازش داده‌های نوع `object`را توضیح می‌دهم و کدهای مرتبط را ارائه می‌کنم:

۱. بررسی داده‌های نوع object

قبل از هر گونه پیش پردازش، بهتر است داده‌های نوع object را بررسی کنیم تا بفهمیم که چه نوع داده‌هایی در اختیار داریم.

```
python
Copy code
import pandas as pd

#خواندن داده ها
df = pd.read_csv('data.csv')

 #نمایش ستون‌های نوع object
print(df.select_dtypes(include=['object']).head())
```

۲. بررسی مقادیر منحصربفرد

بررسی مقادیر منحصربفرد در هر ستون نوع object می‌تواند به فهمیدن اینکه چگونه باید آنها را پردازش کنیم کمک کند.

```
python
Copy code
 #نمایش مقادیر منحصربفرد در هر ستون نوع object
for column in
df.select_dtypes(include=['object']).columns:
    print(f"{column}: {df[column].unique()}")
```

۳. تبدیل داده‌های دسته‌بندی شده (Categorical Data)

الف) استفاده از Label Encoding

Label Encoding برای تبدیل مقادیر دسته‌بندی شده به اعداد صحیح استفاده می‌شود.

```
python
Copy code
from sklearn.preprocessing import LabelEncoder

 #ایجاد یک نمونه از LabelEncoder
le = LabelEncoder()
```

```
#تبدیل ستون‌های دسته‌بندی شده به اعداد صحیح
df['category_encoded'] =
le.fit_transform(df['category'])
```

ب) استفاده از One-Hot Encoding

One-Hot Encodingبرای تبدیل مقادیر دسته‌بندی شده به بردارهای باینری استفاده می‌شود.

```
python
Copy code
 One-Hot Encodingاستفاده از pandas برای انجام#
df = pd.get_dummies(df, columns=['category'])
```

٤. تبدیل داده‌های متنی(Text Data)

الف) استفاده از TF-IDF

TF-IDFبرای تبدیل متن به مقادیر عددی استفاده می‌شود که فرکانس واژه‌ها و اهمیت آنها را در نظر می‌گیرد.

```
python
Copy code
from sklearn.feature_extraction.text import
TfidfVectorizer

 TfidfVectorizerایجاد یک نمونه از#
tfidf = TfidfVectorizer()

#تبدیل متن به مقادیر عددی
tfidf_matrix = tfidf.fit_transform(df['text_column'])
```

ب) استفاده از Word Embeddings

Word Embeddingsمانند Word۲Vec یا GloVe برای تبدیل واژه‌ها به بردارهای عددی با ابعاد ثابت استفاده می‌شوند.

```
python
Copy code
from gensim.models import Word۲Vec

#آموزش مدل Word۲Vec بر روی داده‌های متنی
```

```
sentences = df['text_column'].apply(lambda x:
x.split())
model = Word۲Vec(sentences, vector_size=۱۰۰, window=٥,
min_count=۱, workers=٤)

#نمایش بردارهای متناظر با واژه ها
word_vectors = model.wv
print(word_vectors['example'])
```

۵.مدیریت داده‌های گم شده در ستون‌های نوع object

در صورت وجود داده‌های گم شده در ستون‌های نوع object، می‌توانید آنها را حذف یا جایگزین کنید.

```
python
Copy code
#حذف سطرهایی که داده های گم شده دارند
df.dropna(subset=['object_column'], inplace=True)

#جایگزینی داده های گم شده با مقادیر پر استفاده
df['object_column'].fillna(df['object_column'].mode()[۰
], inplace=True)
```

۶.ترکیب داده‌های نوع object با داده‌های عددی

در نهایت، بعد از تبدیل داده‌های نوع object به فرمت‌های عددی، می‌توانید آنها را با داده‌های عددی ترکیب کنید و به مدل‌سازی بپردازید.

```
python
Copy code
#ترکیب داده های عددی و داده های تبدیل شده
X = pd.concat([df.select_dtypes(exclude=['object']),
pd.DataFrame(tfidf_matrix.toarray())], axis=۱)
```

با استفاده از این روش‌ها، می‌توانید داده‌های نوع object را به درستی پردازش کرده و از آنها در مدل‌های یادگیری ماشین استفاده کنید. انتخاب روش مناسب بستگی به نوع داده‌ها و کاربرد خاص پروژه شما دارد.

روش ساخت دامی

ساخت دامی (Dummy Encoding) یکی از روش‌های محبوب برای تبدیل داده‌های دسته‌بندی شده (categorical data)به داده‌های عددی است. در این روش، هر دسته به یک یا چند ستون باینری (۰ یا ۱) تبدیل می‌شود. این روش به ویژه زمانی مفید است که بخواهیم از داده‌های دسته‌بندی شده در مدل‌های یادگیری ماشین استفاده کنیم.

مراحل ساخت دامی

۱. **شناسایی داده‌های دسته‌بندی شده** :شناسایی ستون‌هایی که شامل داده‌های دسته‌بندی شده هستند.
۲. **استفاده از تابع** :**get_dummies**استفاده از تابع get_dummiesدر پایتون برای تبدیل داده‌های دسته‌بندی شده به دامی.
۳. **بررسی و استفاده از داده‌های تبدیل شده** :بررسی داده‌های تبدیل شده و استفاده از آنها در مدل‌های یادگیری ماشین.

مثال پایتون

فرض کنید داده‌ای به صورت زیر داریم:

```
python
Copy code
import pandas as pd

data = {
    'color': ['red', 'blue', 'green', 'blue', 'green',
'red'],
    'size': ['S', 'M', 'L', 'M', 'S', 'L'],
    'price': [۱۰, ۱۵, ۲۰, ۱۵, ۱۰, ۲۰]
}

df = pd.DataFrame(data)
print(df)
```

مرحله ۱ : شناسایی داده‌های دسته‌بندی شده

در این مثال، ستون‌های color و size داده‌های دسته‌بندی شده هستند.

مرحله ۲ : استفاده از تابع get_dummies

با استفاده از تابع get_dummiesاز کتابخانه pandas، ستون‌های دسته‌بندی شده را به دامی تبدیل می‌کنیم:

```
python
Copy code
#تبدیل داده های دسته‌بندی شده به دامی
df_dummies = pd.get_dummies(df, columns=['color',
'size'])
print(df_dummies)
```

خروجی به صورت زیر خواهد بود:

```
                                                plaintext
                                                Copy code
   price  color_blue  color_green  color_red  size_L
                                      size_M  size_S
0     10           0            0          1       0
                                      0       1
1     15           1            0          0       0
                                      1       0
2     20           0            1          0       1
                                      0       0
3     15           1            0          0       0
                                      1       0
4     10           0            1          0       0
                                      0       1
5     20           0            0          1       1
                                      0       0
```

مرحله ۳: بررسی و استفاده از داده‌های تبدیل شده

اکنون داده‌های دسته‌بندی شده به دامی تبدیل شده‌اند و می‌توانیم از این داده‌ها در مدل‌های یادگیری ماشین استفاده کنیم.

نکات مهم

- **حذف یک ستون دامی (Dummy Variable Trap)**: در برخی موارد، برای جلوگیری از هم‌خطی چندگانه(multicollinearity) ، یک ستون دامی را حذف می‌کنند. این کار را می‌توان با استفاده از پارامتر `drop_first=True` در تابع `get_dummies` انجام داد.

```
python
Copy code
df_dummies = pd.get_dummies(df, columns=['color',
'size'], drop_first=True)
```

```
print(df_dummies)
```

- **تعامل بین ویژگی‌ها (Interaction Terms):** گاهی اوقات می‌توان تعامل بین ویژگی‌های دامی و دیگر ویژگی‌ها را نیز مد نظر قرار داد. این کار می‌تواند به بهبود دقت مدل کمک کند.
- **ترکیب داده‌های دامی با داده‌های عددی:** پس از تبدیل داده‌های دسته‌بندی شده به دامی، می‌توانیم آنها را با دیگر داده‌های عددی ترکیب کرده و در مدل‌های یادگیری ماشین استفاده کنیم.

با استفاده از این روش‌ها، می‌توانید داده‌های دسته‌بندی شده خود را به درستی پردازش کرده و از آنها در مدل‌های یادگیری ماشین بهره‌مند شوید.