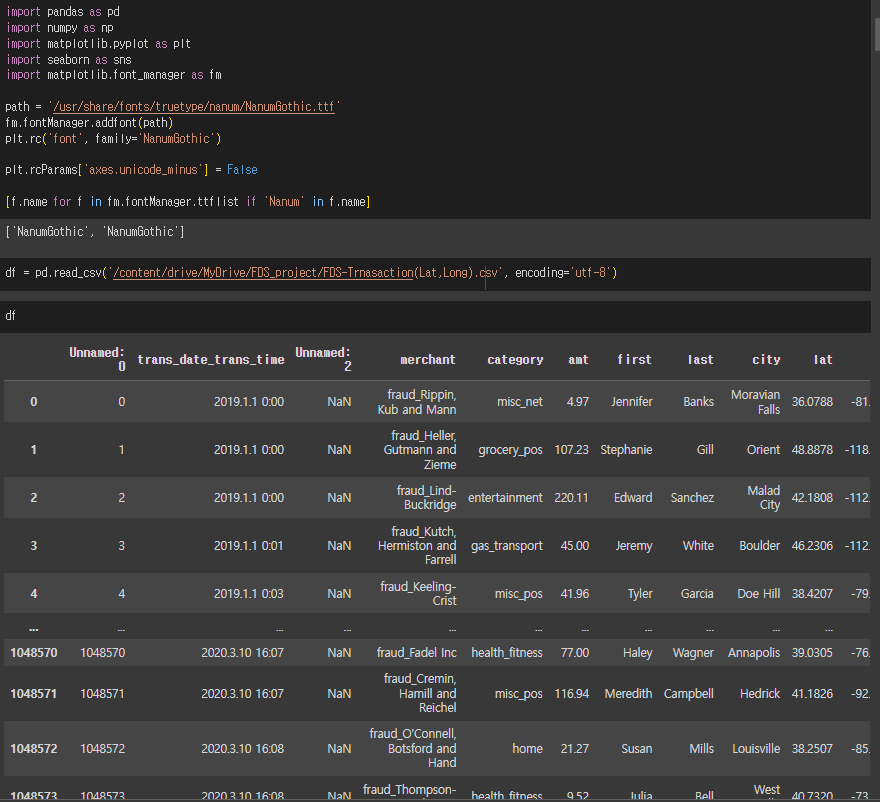
<코드 설명서 - 데이터 처리 및 분석>

| FDS 데이터 처리 및 분석 설명서 | |
| --- | --- |
| 작성자 | 김동훈 |
| 작성 일자 |  |
| 팀명 | Sniffers |

| 순서 | 작업 내용 | 페이지 |
| --- | --- | --- |
| 1 | 데이터 및 필요 라이브러리 로드 | 2 ~ 3 |
| 2 | 날짜 및 시간 데이터 처리 | 4 ~ 5 |
| 3 | 데이터 탐색적 분석 (EDA) | 6 |
| 데이터 구조 및 개요 파악 | 6 ~ 8 |
| 데이터 분포 및 특성 파악 | 9 ~ 12 |
| 4 | 데이터 전처리 | 13 |
| 거리 계산 | 13 ~ 14 |
| 결측치 처리 | 15 ~ 16 |
| EDA 추가 분석 | 17 ~ 22 |
| 이상치 처리 | 23 ~ 28 |
| 데이터 정규화 및 스케일링 | 29 ~ 30 |
| 파생 변수 생성 | 31 ~ 33 |
| 5 | 데이터 분할 및 샘플링 | 34 ~ 35 |
| 6 | 데이터 변환 및 인코딩 | 36 ~ 38 |
| 7 | 최종 데이터셋 준비 및 확인 | 39 ~ 41 |

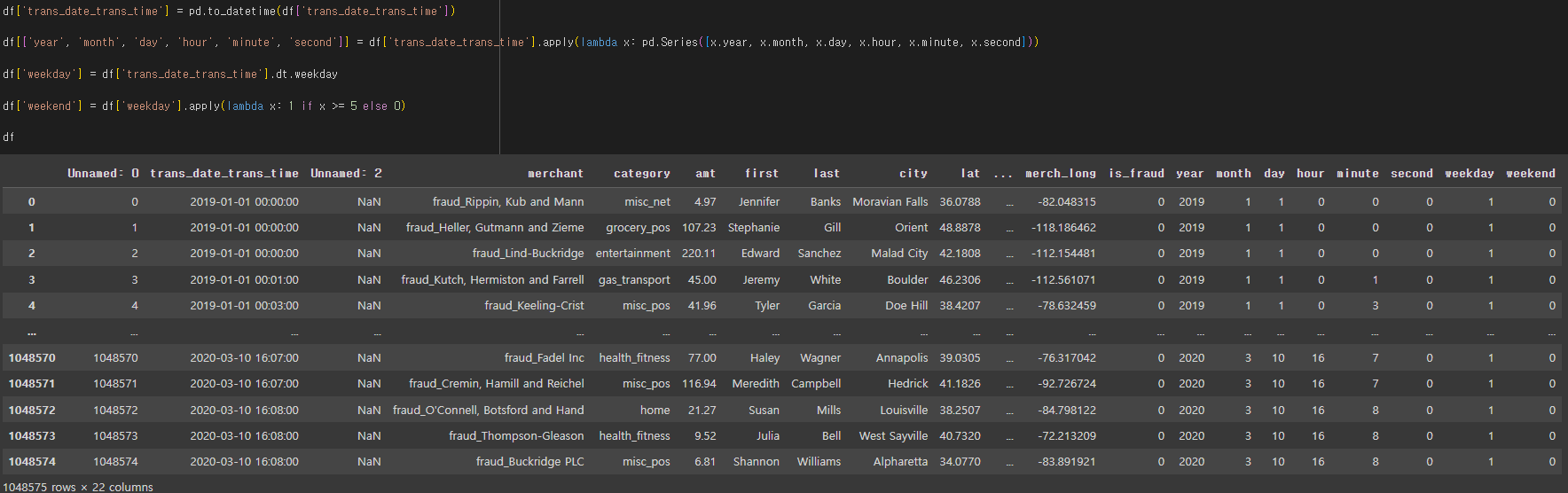
# 

# <1. 데이터 및 필요 라이브러리 로드>



| 기능 | |
| --- | --- |
| 데이터 분석에 필요한 데이터를 불러오고, Python 라이브러리를 임포트 한다. Matplotlib에서 한글 폰트를 사용하기 위한 설정을 포함한다. | |
| 코드 설명 | |
| import pandas as pd | Python 라이브러리인 pandas를 pd라는 약어로 불러들인다. |
| import numpy as np | Python 라이브러리인 numpy를 np라는 약어로 불러들인다. |
| import matplotlib.pyplot as plt | Python 라이브러리인 matplotlib의 pyplot을 plt라는 약어로 불러들인다. |
| import seaborn as sns | Python 라이브러리인 seaborn를 sns라는 약어로 불러들인다. |
| import matplotlib.font\_manager as fm | Python 라이브러리인 matplotlib의 font\_manager를 fm이라는 약어로 불러들인다. |
| path = ‘/usr/share/fonts/truetype/nanum/NanumGothic.ttf’ | 한글 사용을 위해 나눔 고딕 글꼴 파일의 경로를 정의 한다. |
| fm.fontManager.addfont(path) | matplotlib 라이브러리 글꼴 관리자에 나눔 고딕 글꼴을 추가한다. |
| plt.rc(‘font’, family=’NanumGothic’) | matplotlib 플롯의 기본 글꼴을 나눔 고딕으로 설정한다. |
| plt.rcParams[‘axes.unicode\_minus’] = False | 플롯에서 한국어 문자와 기호가 제대로 표시되도록 설정한다. |
| [f.name for f in fm.fontManager.ttflist if ‘Nanum’ in f.name] | 글꼴 관리자에서 이름이 ‘Nanum’이 포함된 글꼴을 사용할 수 있는지 확인한다. |
| 입력 | |
| /content/drive/MyDrive/FDS\_project/FDS-Transaction(Lat,Long).csv | 데이터 파일 경로 (Google Colab 드라이브 경로) |
| 출력 | |
| df | pandas 데이터프레임 객체 |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| FileNotFoundError | 데이터 파일 경로가 잘못 되었거나, 파일이 존재하지 않을 경우 발생한다. |
| 알고리즘 | |
| 필요한 라이브러리 (pandas, numpy, matplotlib.pyplot, seaborn, matplotlib.font\_manager)를 임포트 한다. | |
| Google Colab 환경에서 한글 폰트 (NanumGothic)를 설치하고 matplotlib에 설정한다. | |
| pd.read\_csv() 함수를 사용하여 지정된 경로의 CSV 파일을 읽고 pandas 데이터 프레임 객체로 저장한다. | |
| 추가 설명 | |
| 한글 폰트 설정은 matplitlib에서 한글이 깨져 보이는 문제를 방지하기 위해 필요하다. | |

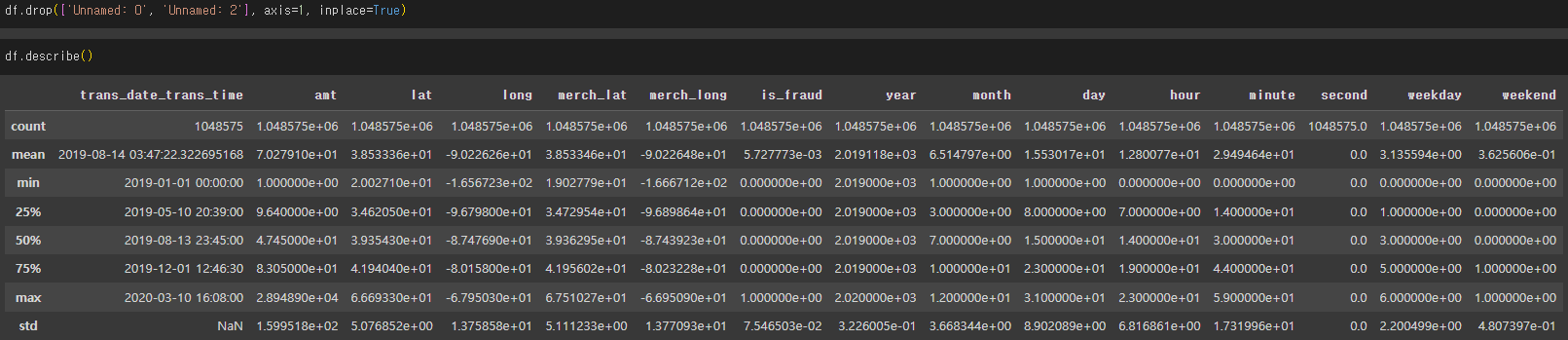
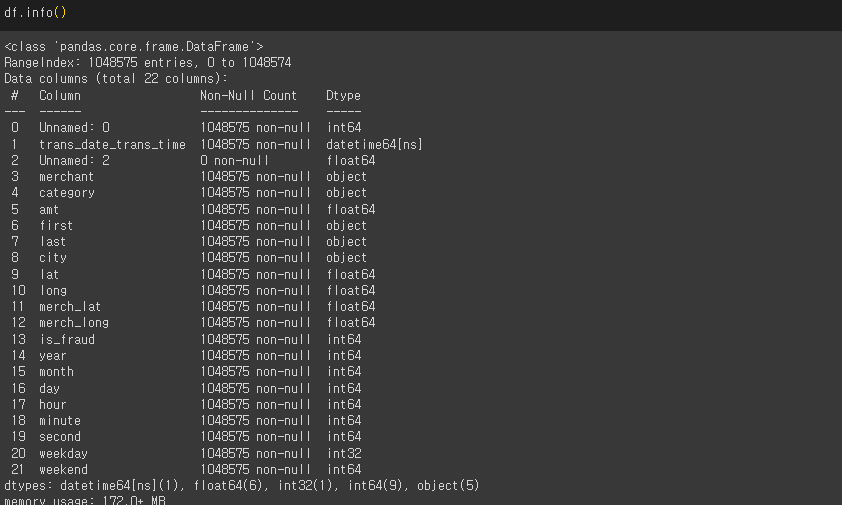
# <2. 날짜 및 시간 데이터 처리>



| 기능 | |
| --- | --- |
| ‘trans\_date\_trans\_time’ 컬럼의 날짜 및 시간 데이터를 처리하여 분석에 활용하기 쉽도록 변환하고, 연도, 월, 일, 시간, 분, 초, 요일, 주말 여부 등의 정보를 추출한다. | |
| 코드 설명 | |
| df[‘trans\_date\_trans\_time’] = pd.to\_datetime(df[‘trans\_date\_trans\_time’] | 거래 날짜 및 시간(trans\_date\_trans\_time) 컬럼을 datetime 객체로 변환한다. |
| df[['year', 'month', 'day', 'hour', 'minute', 'second']] = df['trans\_date\_trans\_time'].apply(lambda x: pd.Series([x.year, x.month, x.day, x.hour, x.minute, x.second])) | apply 함수와 lambda 표현식을 사용하여 datetime 객체에서 연도, 월, 일, 시간, 분, 초 정보를 추출하여 새로운 컬럼에 저장한다. |
| df['weekday'] = df['trans\_date\_trans\_time'].dt.weekday | dt.weekday 속성을 사용하여 datetime 객체에서 요일 정보를 추출한다.  (월요일: 0 ~ 일요일: 6) |
| df['weekend'] = df['weekday'].apply(lambda x: 1 if x >= 5 else 0) | apply 함수와 lambda 표현식을 사용하여 요일 정보를 기반으로 주말 여부를 나타내는 새로운 컬럼을 생성한다.  (주말: 1, 평일: 0) |
| 입력 | |
| df | ‘trans\_date\_trans\_time’ 컬럼을 포함하는 pandas 데이터프레임 객체 |
| 출력 | |
| df | 연도, 월, 일, 시간, 분, 초, 요일, 주말 여부 컬럼이 추가된 pandas 데이터프레임 객체 |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| ValueError | ‘trans\_date\_trans\_time’ 컬럼의 값이 날짜/시간 형식으로 변환될 수 없는 경우 발생 |
| 알고리즘 | |
| pd.to\_datetime() 함수를 사용하여 ‘trans\_date\_trans\_time’ 컬럼의 값을 datetime 객체로 변환한다. | |
| datetime 객체에서 연도, 월, 일, 시간, 분, 초 정보를 추출하고 새로운 컬럼에 저장한다. | |
| dt.weekday 속성을 사용하여 datetime 객체에서 요일 정보를 추출한다. | |
| 요일 정보를 기반으로 주말 여부를 나타내는 새로운 컬럼을 생성한다. | |
| 추가 설명 | |
| apply 함수와 lambda 표현식은 데이터 프레임의 각 행에 특정 함수를 적용하는데 사용된다. | |

# <3. 데이터 탐색적 분석 (EDA)>

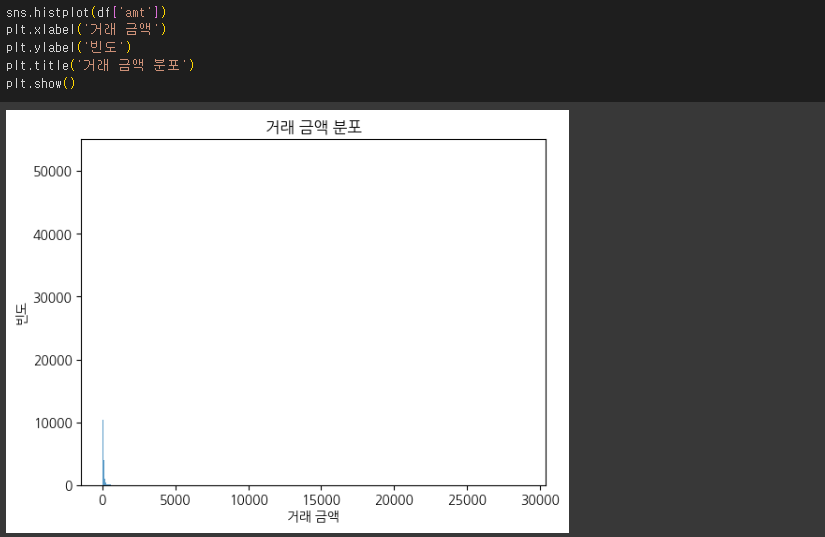
<3.1. 데이터 구조 및 개요 파악>



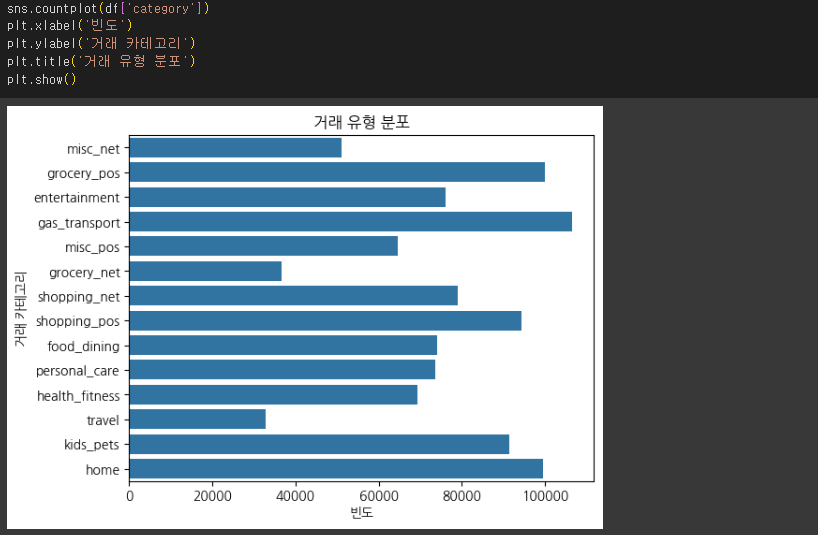
| 기능 | |
| --- | --- |
| 로드된 데이터프레임의 구조와 개요를 파악한다. 데이터의 크기, 컬럼 정보, 데이터 타입, 결측치 등을 확인한다. | |
| 코드 설명 | |
| df.info() | 데이터프레임의 각 컬럼에 대한 정보를 출력한다. |
| df.drop([‘Unnamed: 0, ‘Unnamed: 2’], axis = 1, inplace = True) | 불필요한 컬럼인 ‘Unnamed: 0’와 ‘Unnamed: 2’ 컬럼을 제거한다. |
| df.describe() | 숫자형 컬럼에 대한 기술 통계량을 출력한다. 평균, 표준 편차, 최솟값, 최댓값, 사분위수 등을 확인할 수 있다. |
| df.shape | 데이터프레임의 행과 열의 개수를 출력한다. |
| df.dtypes | 각 컬럼의 데이터 타입을 출력한다. |
| df.count() | 각 컬럼의 Non-Null 값의 개수를 출력한다. |
| df.index | 데이터프레임의 인덱스 정보를 출력한다. |
| df.columns | 데이터프레임의 컬럼 이름을 출력한다. |
| 입력 | |
| df | pandas 데이터프레임 객체 (분석할 데이터) |
| 출력 | |
| 각 함수의 실행 결과 (데이터프레임 정보, 기술 통계량, 크기, 데이터 타입, Non-Null 값 개수, 인덱스 정보, 컬럼 이름) | |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| 없음 | |
| 알고리즘 | |
| info(), describe(), shape, dtypes, count(), index, columns 함수를 호출하여 데이터프레임의 구조 및 개요 정보를 확인한다. | |
| 추가 설명 | |
| describe() 함수는 숫자형 데이터에 대한 정보만 제공한다. | |

| 컬럼 설명 | |
| --- | --- |
| trans\_date\_trans\_time | 거래 날짜 및 시간 정보 |
| merchant | 거래가 발생한 가맹점 정보 |
| category | 거래가 발생한 품목 또는 서비스의 카테고리 정보 |
| amt | 거래 금액 정보 |
| first | 고객의 이름 정보 |
| last | 고객의 성 정보 |
| city | 거래가 발생한 도시 정보 |
| lat | 고객의 위도 정보 |
| long | 고객의 경도 정보 |
| merch\_lat | 가맹점의 위도 정보 |
| merch\_long | 가맹점의 경도 정보 |
| is\_fraud | 거래가 사기 거래인지 여부 |

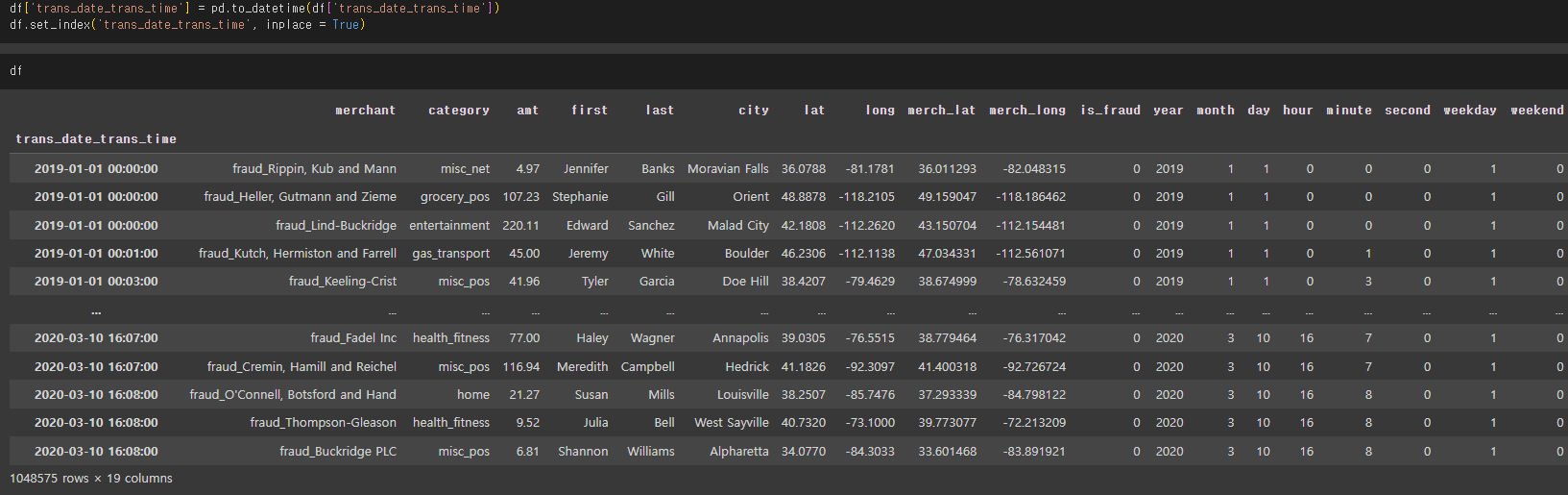
<3.2. 데이터 분포 및 특성 파악>



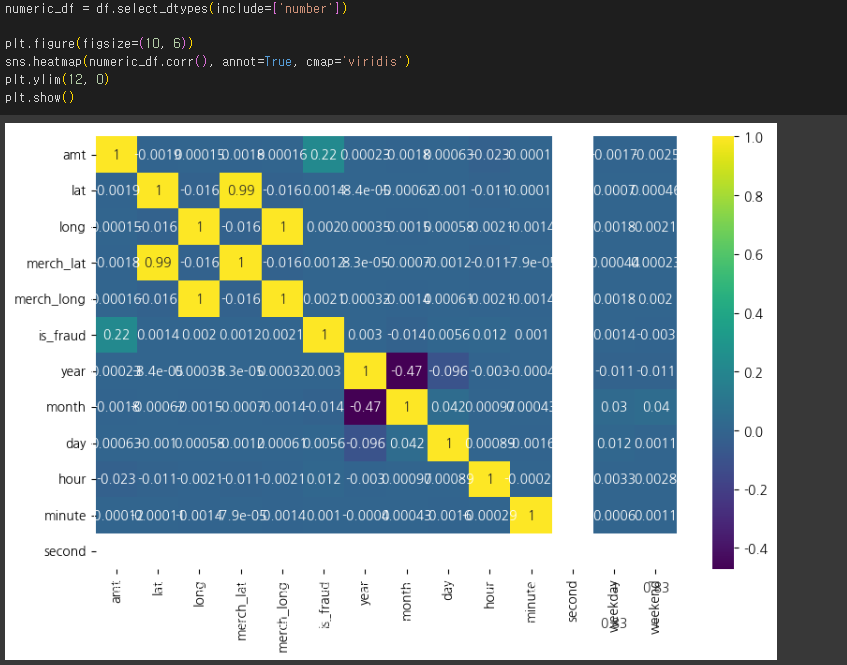
수치형 데이터 탐색: 히스토그램



범주형 데이터 탐색: 거래 카테고리



시계열 데이터 탐색: 거래 날짜 및 시간

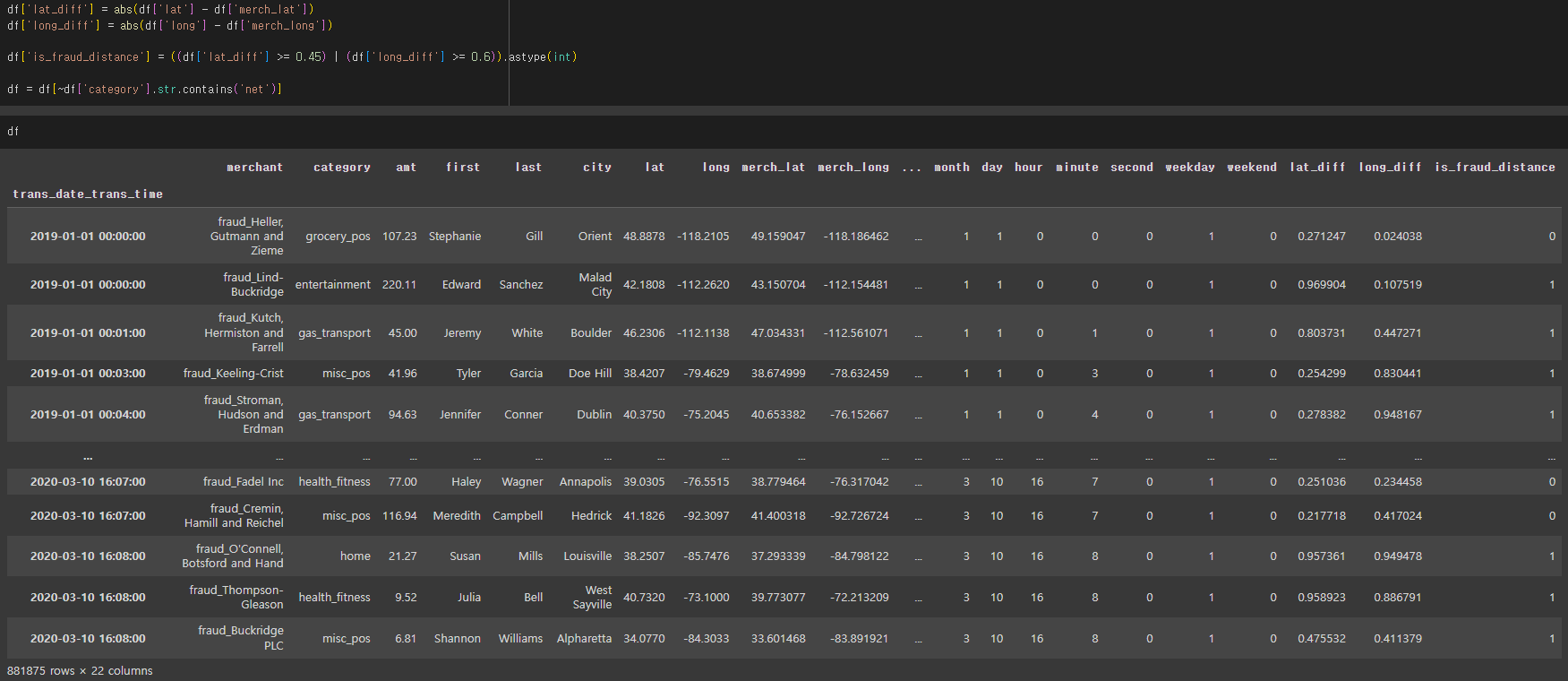


상관 분석: 숫자형 컬럼

| 기능 | |
| --- | --- |
| 데이터의 분포 및 특성을 파악하기 위해 다양한 시각화 기법 (히스토그램, 카운트 플롯, 시계열 그래프, 히트맵)을 사용하고, 변수 간의 상관 관계를 분석한다. | |
| 코드 설명 | |
| sns.histplot(df[‘amt’]) | 거래 금액(amt)의 분포를 히스토그램으로 시각화하여 데이터의 분포와 이상치를 확인한다.  히스토그램을 통해 거래 금액이 어떤 범위에 주로 분포하는지, 극단적인 값이 있는지를 파악할 수 있다. |
| sns.countplot(df[‘category’]) | 거래 카테고리(category)의 각 범주별 개수를 카운트 플롯으로 시각화하여 어떤 카테고리의 거래가 많고, 적은지 파악한다. |
| df[‘trans\_date\_trans\_time’] = pd.to\_datetime(df[‘trans\_date\_trans\_time’]) | 거래 날짜 및 시간(trans\_date\_trans\_time) 컬럼을 datetime 객체로 변환하여 시계열 데이터로 처리한다. |
| df.set\_index(‘trans\_date\_trans\_time’, inplace = True) | 거래 날짜 및 시간(trans\_date\_trans\_time) 컬럼을 인덱스로 설정하여 시간 기반 분석을 용이하게 한다. |
| numeric\_df = df.select\_dtypes(include = [‘number’]) | 숫자형 컬럼만 선택하여 상관 분석에 사용한다. |
| sns.heatmap(numeric\_df.corr(), annot = True, cmap = ‘viridis’) | 숫자형 변수 간의 상관 관계를 히트맵으로 시각화하여 변수 간의 관계를 파악한다.  annot = True 옵션을 사용하여 히트맵에 상관 계수를 표시한다. |
| 입력 | |
| df | pandas 데이터프레임 객체 (분석할 데이터) |
| 출력 | |
| 히스토그램, 카운트 플롯, 시계열 그래프, 히트맵 (데이터 시각화 결과) | |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| TypeError | 숫자형 데이터가 아닌 컬럼에 histplot 함수를 사용하는 경우 발생 |
| valueError | resample 함수에 잘못된 시간 단위를 지정하는 경우 발생 |
| 알고리즘 | |
| pd.to\_datetime(), set\_index(), resample() 함수를 사용하여 시계열 데이터를 분석하고 시각화 한다. | |
| select\_dtypes(), corr(), sns.heatmap() 함수를 사용하여 변수 간의 상관 관계를 분석하고 히트맵으로 시각화 한다. | |
| 추가 설명 | |
| 히스토그램은 데이터의 분포를 파악하는데 유용하며, 이상치를 확인하는 데에도 사용할 수 있다. | |
| 카운트 플롯은 범주형 데이터의 각 범주별 개수를 비교하는데 유용하다. | |
| 시계열 그래프는 시간에 따른 데이터의 변화를 파악하는데 유용하다. | |
| 히트맵은 변수 간의 상관 관계를 시각적으로 파악하는 데 유용하다. | |

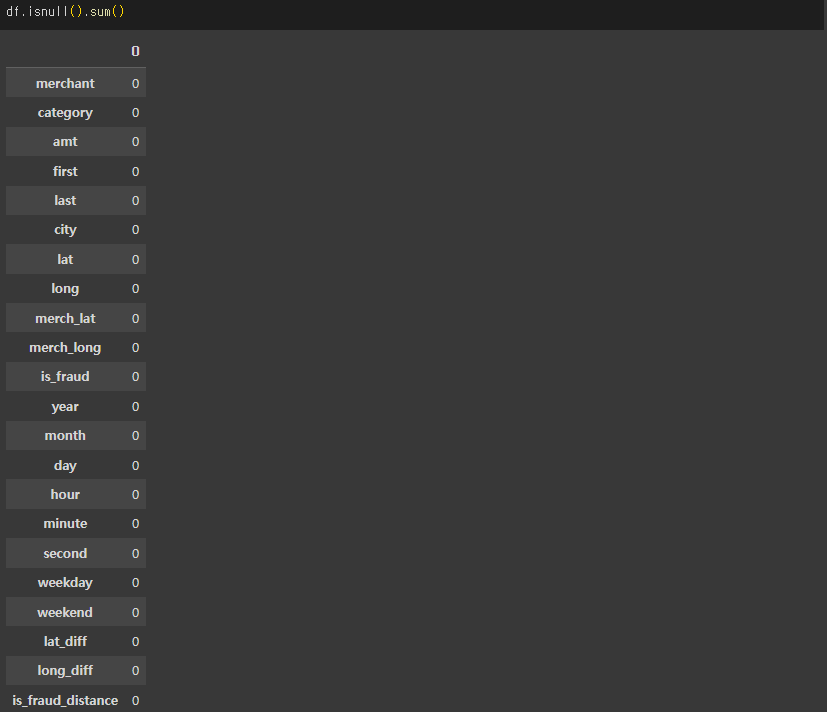
# <4. 데이터 전처리>

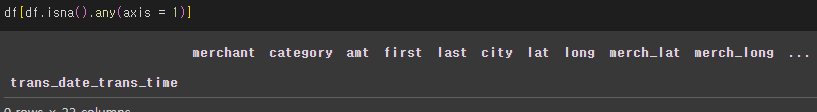
<4.1. 거리 계산>



| 기능 | |
| --- | --- |
| 고객의 위치(위도, 경도)와 가맹점의 위치(위도, 경도) 간의 거리 차이를 계산하여 새로운 변수 (lat\_diff, long\_diff)를 생성한다. 이를 통해 고객과 가맹점 간의 거리가 비정상적으로 먼 이상 거래를 탐지한다. | |
| 코드 설명 | |
| df[‘lat\_diff’] = abs(df[‘lat’] - df[‘merch\_lat’]) | 고객의 위도(lat)와 개맹점의 위도(merch\_lat) 차이의 절댓값을 계산하여 새로운 컬럼 ‘lat\_diff’에 저장한다. |
| df[‘long\_diff’] = abs(df[‘long’] - df[‘merch\_long’]) | 고객의 경도(long)와 가맹점의 경도(merch\_long) 차이의 절댓값을 계산하여 새로운 컬럼 ‘long\_idff’에 저장한다. |
| df[‘is\_fraud\_distance’] = ((df[‘lat\_diff’] >= 0.45) | (df[‘long\_diff’] >= 0.6)).astype(int) | 경도 차이가 0.45 이상이거나, 위도 차이가 0.6 이상이면 이상 거래로 판단하고 새로운 컬럼 ‘is\_fraud\_distance’에 1을 저장한다. |
| df = df[~df[‘category’].str.contains(‘net’)] | category 컬럼에 net이 포함된 거래 (인터넷 거래)를 제거한다. |
| 입력 | |
| df | 고객 및 가맹점의 위도/경도 정보(lat, long, merch\_lat, merch\_long), 거래 카테고리 정보(category)를 포함하는 pandas 데이터프레임 객체 |
| 출력 | |
| df | 위도/경도 차이(lat\_diff, long\_diff), 이상 거래 여부(is\_fraud\_distance) 컬럼이 추가되고 인터넷 거래가 제외된 pandas 데이터 프레임 객체 |
| 매개변수 | |
| lat\_diff\_threshold | 위도 차이 임계값 (기본값: 0.45) |
| long\_diff\_threshold | 경도 차이 임계값 (기본값: 0.6) |
| 예외 처리 | |
| KeyError | 입력 데이터프레임에 필요한 컬럼(lat, long, merch\_lat, mercho\_long, category)이 없는 경우 발생 |
| 알고리즘 | |
| 고객과 가맹점 간의 위도 차이와 경도 차이를 계산한다. | |
| 계싼된 위도/경도 차이가 지정된 임계값을 초과하는 경우 이상 거래로 판단한다. | |
| 인터넷 거래는 거리 기반 이상 거래 탐지에서 제외한다. | |
| 추가 설명 | |
| 인터넷 거래는 물리적인 거리 개념이 적용되지 않으므로 제외한다. | |
| Haversine 공식과 같은 복잡한 거리 계산 공식을 사용하지 않고, 단순히 위도/경도 차이를 이용하여 이상 거래를 탐지하는 간단한 방법을 사용한다. | |
| 보다 정확한 거리 계산을 위해서는 Haversine 공식이나 Manhattan Distance, Euclidean Distance, Spherical Law of Cosines, Vincenty’s Formulae를 사용할 수 있다. | |

<4.2. 결측치 처리>

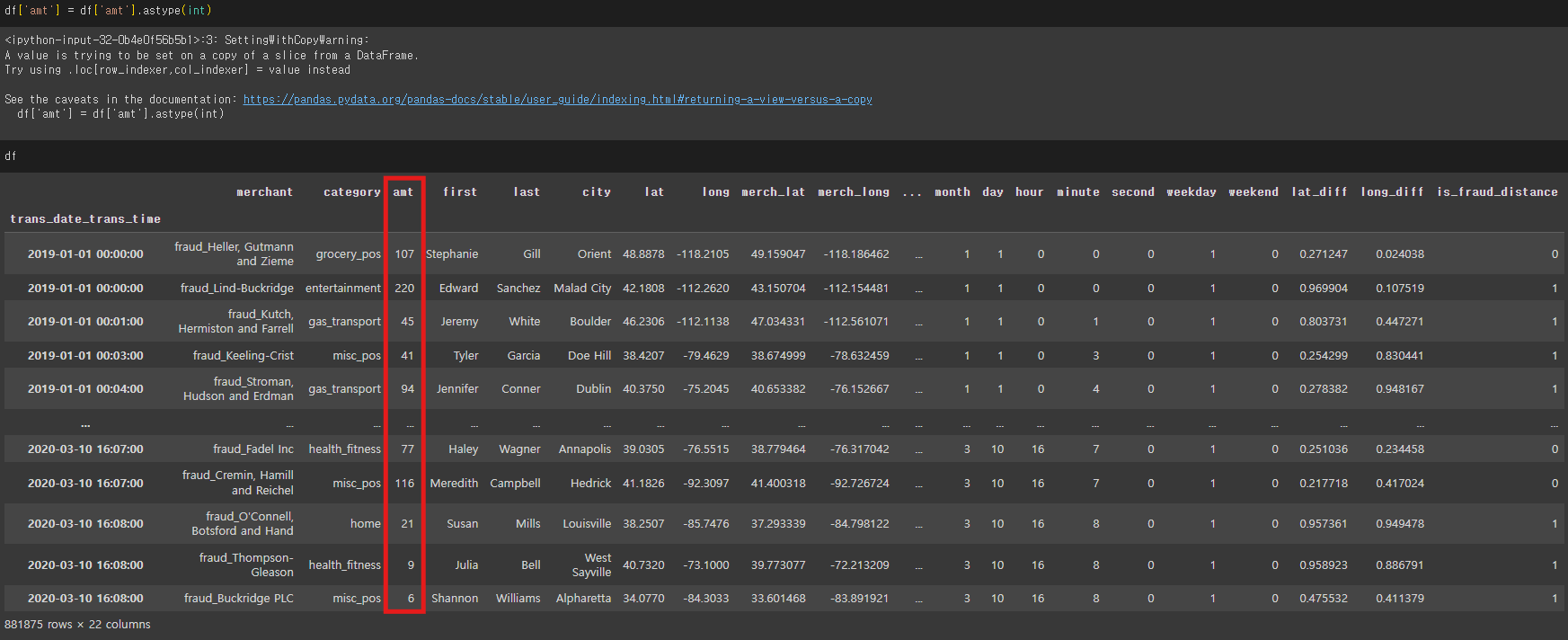




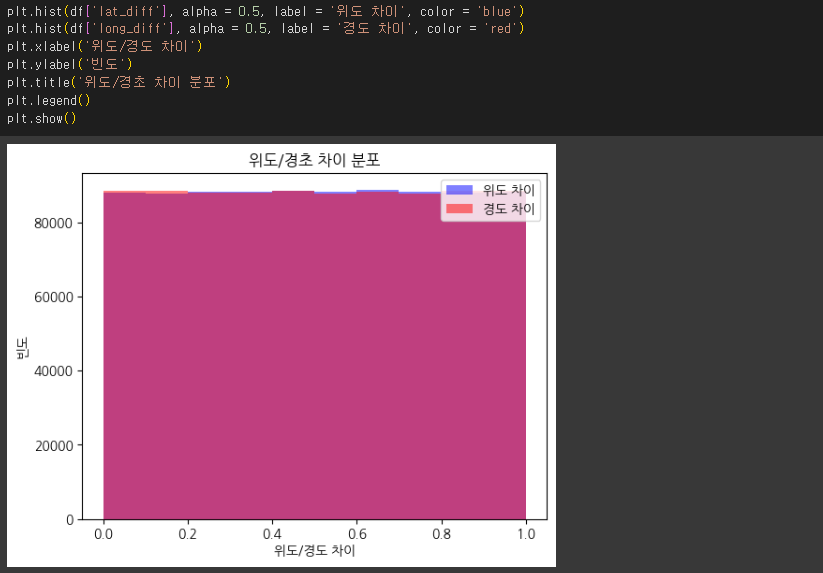
컬럼별 결측치 개수 확인

| 기능 | |
| --- | --- |
| 데이터에 존재하는 결측치를 식별하고, 적절한 방법을 사용하여 처리한다. 이를 통해 데이터 분석 및 모델링의 정확성을 높일 수 있다. | |
| 코드 설명 | |
| df.isnull().sum() | 각 컬럼별로 결측치(NaN) 개수를 계산하고 출력한다. |
| df[df.isna().any(axis = 1)] | 데이터 프레임에서 결측치가 하나라도 있는 행을 출력한다. |
| 입력 | |
| df | 결측치를 포함할 수 있는 pandas 데이터프레임 객체 |
| 출력 | |
| df | 결측치가 처리된 pandas 데이터프레임 객체 |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| 없음 | |
| 알고리즘 | |
| isnull().sum() 함수를 사용하여 각 컬럼별 결측치 개수를 확인한다. | |
| 결측치가 있는 경우, dropna() 함수를 사용하여 결측치가 있는 행을 제거한다. | |
| 필요에 따라 다른 결측치 처리 방법(예: 평균값으로 대체)을 적용할 수 있다. | |
| 추가 설명 | |
| 결측치 처리는 데이터 분석 및 모델링의 정확성을 높이는 데 중요한 단계이다. | |
| 결측치를 제거하는 방법은 데이터 손실을 초래할 수 있으므로, 데이터의 특성을 고려하여 신중하게 결정해야 한다. | |
| 결측치를 다른 값으로 대체하는 방법 (평균, 중앙값, 최빈값 등)은 데이터 손실을 줄일 수 있지만, 데이터의 분포를 왜곡할 수 있다. | |

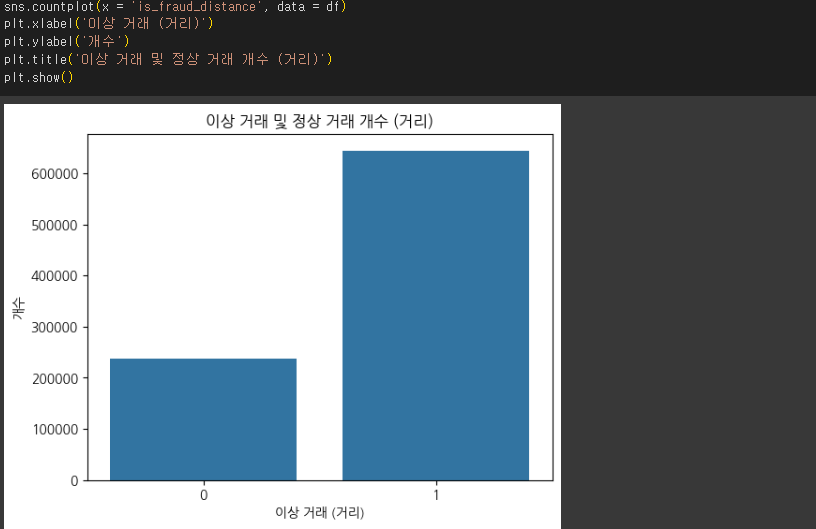
<4.3. EDA 추가 분석>



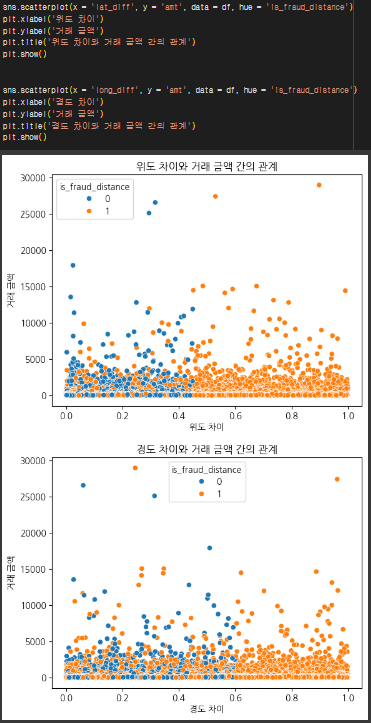
거래 금액 데이터 정수형 변환



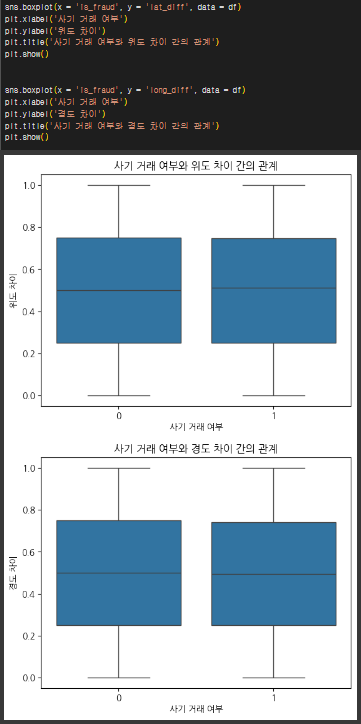
거리 분포 시각화



이상 거래 탐지 겨로가 시각화 (위도/경도 기반)



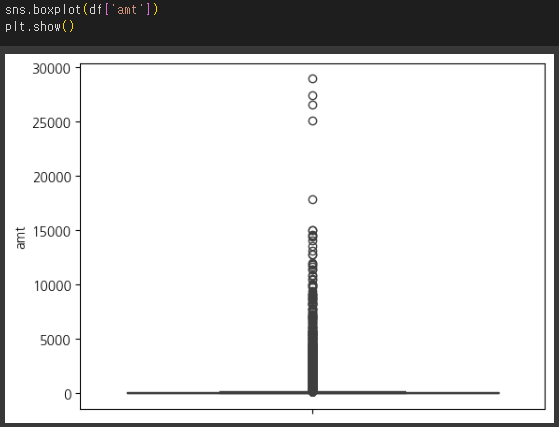
다른 변수와의 관계 분석 (scatterplot: 거리 - 거래 금액)



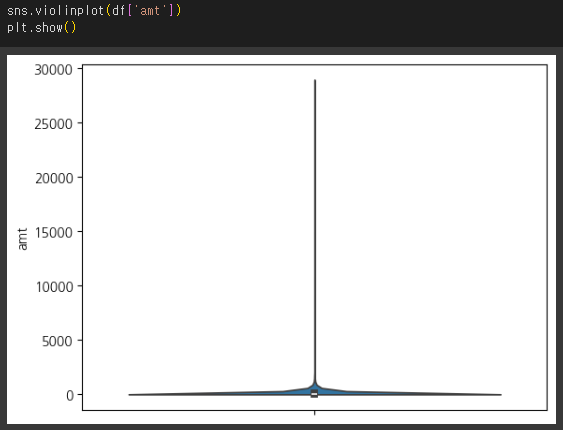
다른 변수와의 관계 순석 (boxplot: 거리 - 사기 거래 여부)

| 기능 | |
| --- | --- |
| 거리 변수 (lat\_diff, long\_diff)를 추가적으로 분석하여 이상 거래 탐지 성능을 향상시키고, 다른 변수들과의 관계를 파악한다. | |
| 코드 설명 | |
| plt.hist(df[‘at\_diff’], alpha = 0.5, label = ‘위도 차이’, color = ‘blue’) | ‘lat\_diff’ 컬럼 (위도 차이)의 히스토그램을 그린다. alpha는 투명도를 설정한다. |
| plt.hist(df[‘long\_diff’], alpha = 0.5, label = ‘경도 차이’, color = ‘blue’) | ‘long\_diff’ 컬럼 (경도 차이)의 히스토그램을 그린다. |
| plt.xlabel() | x축 레이블을 설정한다. |
| plt.ylabel() | y축 레이블을 설정한다. |
| plt.title() | 그래프의 제목을 설정한다. |
| plt.legend() | 범례를 표시한다. |
| plt.show() | 그래프를 출력한다. |
| sns.countplot(x = ‘is\_fraud\_distance’, data = df) | ‘is\_fraud\_distance’ 컬럼 (거리 기반 이상 거래 여부)을 기반으로 카운트 플롯을 그린다. |
| sns.scatterplot(x = ‘lat\_diff’, y = ‘amt’, data - df, hue = ‘is\_fraud\_distance’) | ‘lat\_diff’ (위도 차이)와 ‘amt’ (거래금액) 간의 관계를 scatterplot으로 시각화한다. hue는 ‘is\_fraud\_distance’를 기준으로 색상을 다르게 표시한다. |
| sns.scatterplot(x = ‘long\_diff’, y = ‘amt’, data = df, hue = ‘is\_fraud\_distance’) | ‘long\_diff’ (경도 차이)와 ‘amt’ (거래 금액) 간의 관계를 scatterplot으로 시각화 한다. |
| sns.boxplot(x = ‘is\_fraud’, y = ‘lat\_diff’, data = df) | ‘is\_fraud’ (사기 거래 여부)와 ‘lat\_diff’ (위도 차이) 간의 관계를 boxplot으로 시각화한다. |
| 입력 | |
| df | 위도/경도 차이 (‘lat\_diff’, ‘long\_diff’), 거리 기반 이상 거래 여부 (‘is\_fraud\_distance’), 거래 금액 (‘amt’), 사기 거래 여부 (‘is\_fraud’) 컬럼을 포함하는 pandas 데이터 프레임 객체 |
| 출력 | |
| 히스토그램, 카운트 플롯, scatterplot, boxplot (데이터 시각화 결과) | |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| KeyError | 입력 데이터프레임에 필요한 컬럼이 없는 경우 발생 |
| 알고리즘 | |
| ‘plt.hist()’ 함수를 사용하여 거리 변수(‘lat\_diff’, ‘long\_diff’)의 분포를 히스토그램으로 시각화한다. | |
| ‘sns.countplot()’ 함수를 사용하여 이상 거래 탐지 결과 (‘is\_fraud\_distance’)를 카운트 플롯으로 시각화한다. | |
| ‘sns.scatterplot()’ 함수를 사용하여 거리 변수와 거래 금액 (‘amt’) 간의 관계를 산점도로 시각화한다. | |
| ‘sns.boxplot()’ 함수를 사용하여 거리 변수와 사기 거래 여부 (‘is\_fraud’) 간의 관계를 boxplot으로 시각화한다. | |
| 추가 설명 | |
| 거리 변수의 분포를 시각화하여 이상치를 탐지하고, 적절한 임계값을 설정하는 데 활용할 수 있다. | |
| 거리 변수와 다른 변수들 간의 관계를 분석하여 사기 거래 탐지 모델의 성능을 향상시키는 데 도움이 되는 정보를 얻을 수 있다. | |
| 시각화 결과를 통해 거리 변수가 사기 거래와 관련이 있는지, 어떤 패턴을 보이는지 등을 파악할 수 있다. | |
| EDA 추가 분석을 통해 얻은 정보는 이상 거래 탐지 모델의 개발 및 개선에 활용될 수 있으며, 데이터에 대한 심층적인 이해를 제공한다. | |

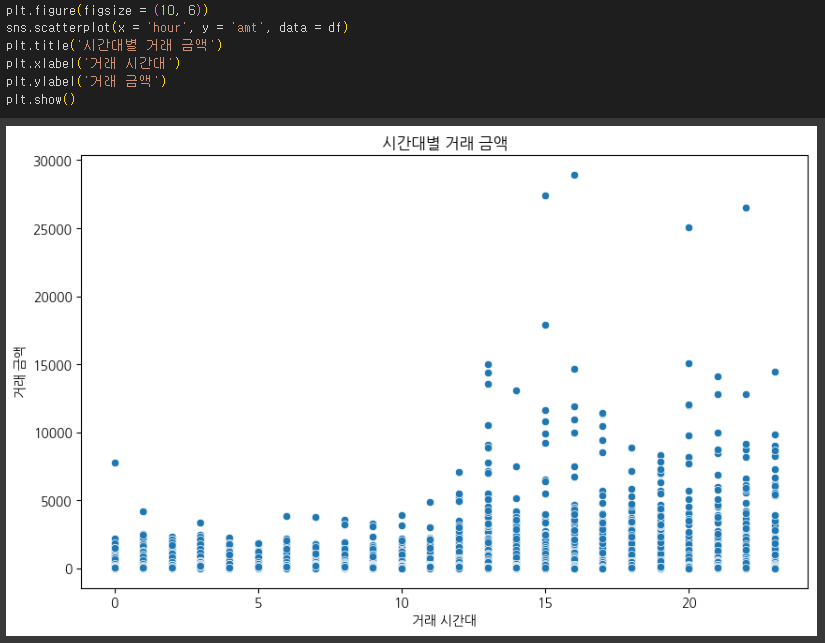
<4.4. 이상치 처리>



이상치 시각화 (boxplot)



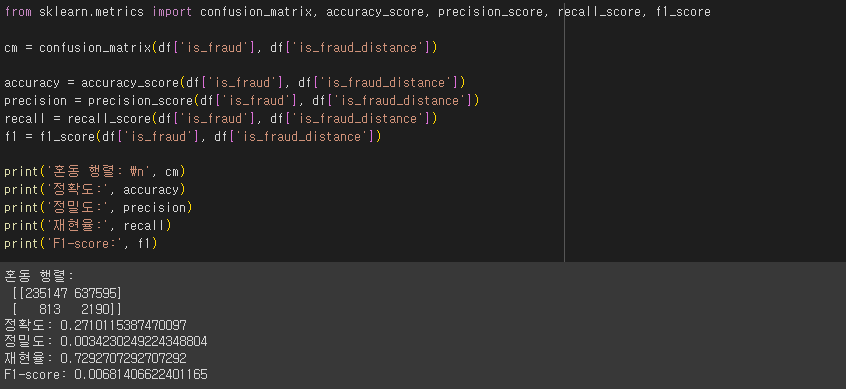
이상치 시각화 (violinplot)



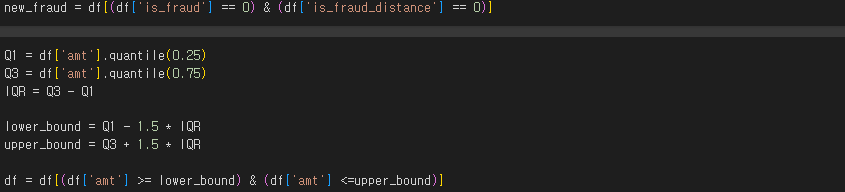
이상치 시각화 (scatterplot: 거래 시간대 - 거래 금액)



Z-score를 이용한 이상치 처리



혼동 행렬, 정확도, 정밀도, 재현율, F1-score



새로운 이상치 후보 추출 및 IQR 계산

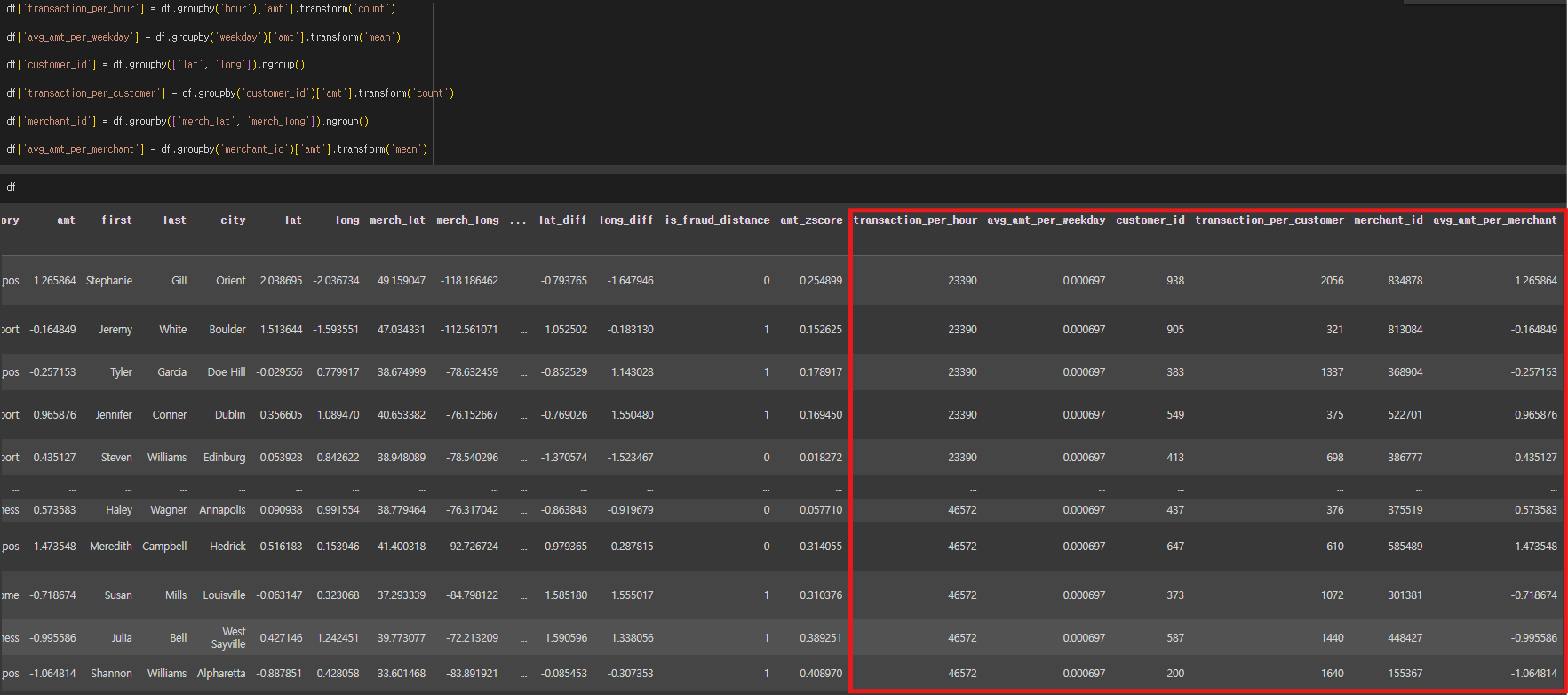
| 기능 | |
| --- | --- |
| 데이터에 존재하는 이상치를 시각화하고, Z-score 방법을 사용하여 처리한다. 이를 통해 데이터 분석 및 모델링의 정확성을 높이고, 이상치로 인한 왜곡을 방지한다. | |
| 코드 설명 | |
|  |  |
|  |  |
|  |  |
| 입력 | |
| df | 거래 금액(amt), 거래 시간대(hour) 컬럼을 포함하는 pandas 데이터프레임 객체 |
| 출력 | |
| df | 이상치가 처리된 pandas 데이터프레임 객체 |
| 매개변수 | |
| threshold | Z-score 임계값 (기본값: 3) |
| 예외 처리 | |
| KeyError | 입력 데이터프레임에 필요한 컬럼이 없는 경우 발생 |
| 알고리즘 | |
| sns.boxplot(), sns.violinplot(), sns.scatterplot() 함수를 사용하여 이상치를 시각화한다. | |
| Z-score를 계산하고 임계값을 초과하는 데이터를 이상치로 판단하여 제거한다. | |
| IQR을 계산하고 IQR을 사용하여 이상치 범위를 설정하여 이상치를 제거한다. | |
| 추가 설명 | |
| 이상치는 데이터 분석 및 모델링 결과에 큰 영향을 미칠 수 있으므로, 적절하게 처리하는 것이 중요하다. | |
| Z-score 방법은 데이터의 평균과 표준 편차를 기반으로 이상치를 탐지하는 방법이다. | |
| 새로운 이상치 후보 추출은 기존의 사기 거래 라벨링에서 놓쳤을 가능성이 있는 새로운 이상치 후보이다. 추출된 거래들의 특징을 분석하여 사기 거래 여부를 판단하고, 필요에 따라 추가적인 조사를 수행한다. | |
| IQR 방법은 데이터의 사분위수를 기반으로 이상치를 탐지하는 방법이다. | |
| 이상치 처리 방법은 데이터의 특성과 분석 목적에 따라 적절히 선택해야 한다. | |

| 출력 결과 및 설명 | | |
| --- | --- | --- |
| 혼동 행렬 | [[235147 637595]  [ 813 2190]] | 이진 분류 모델의 성능을 평가하는 데 사용되는 표.  실제 값과 모델이 예측한 값을 비교하여 True Positive(TP, 정답), False Positive(FP, 1종 오류), True Negative(TN, 정답), False Negative(FN, 2종 오류) 네가지 경우로 나누어 표시한다. |
| 정확도 | 0.2710115387470097 | 전체 예측 중 정확하게 예측한 비율.  (TP + TN) / (TP + FP + FN + TN) |
| 정밀도 | 0.0034230249224348804 | 모델이 Positive로 예측한 것 중 실제로 Positive인 비율.  TP / (TP + FP)  Positive 예측의 정확성을 나타낸다. |
| 재현율 | 0.7292707292707292 | 실제 Positive 중 모델이 Positive로 예측한 비율  TP / (TP + FN)  Positive 케이스를 얼마나 잘 찾아내는지 나타낸다. |
| F1-score | 0.00681406622401165 | 정밀도와 재현율의 조화 평균.  2 \* (정밀도 + 재현율) / (정밀도 + 재현율) |
| 비고 | | |
| 해당 결과는 모델 학습을 통한 결과값이 아닌, 새롭게 생성한 위도/경도 거리 기반 이상 거래 컬럼 (is\_fraud\_distance)와 이상 거래 여부 컬럼 (is\_fraud)을 비교하여 산출된 결과이다. | | |

<4.5. 데이터 정규화 및 스케일링>

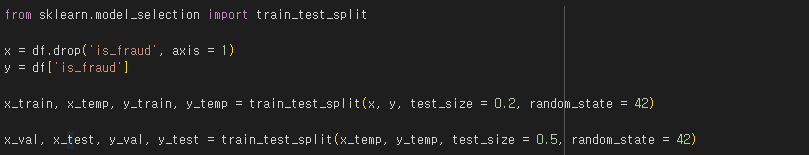
| 기능 | |
| --- | --- |
| 수치형 변수의 값을 일정한 범위로 조정한다. 이를 통해 변수간의 스케일 차이로 인한 문제를 해결하고, 모델의 성능을 향상시킨다. | |
| 코드 설명 | |
| from sklearn.preprocessing import StandardScaler | scikit-learn 라이브러리의 preprocessing 모듈에서 StandardScaler 클래스를 임포트한다. |
| num\_cols = [‘amt’, ‘lat’, ‘long’, ‘lat\_diff’, ‘long\_diff’, ‘year’, ‘month’, ‘day’, ‘hour’, ‘minute’, ‘second’] | 정규화할 수치형 변수들을 리스트로 저장한다. |
| scaler = StandardScaler() | StandardScaler 객체를 생성한다. |
| df[num\_col] = scaler.fit\_transform(df[num\_cols]) | fit\_transform() 메서드를 사용하여 num\_cols에 저장된 변수들을 표준화한다.  표준화는 각 변수의 평균을 0, 표준 편차를 1로 변환하는 것이다. |
| 입력 | |
| df | 정규화할 수치형 변수(num\_cols)를 포함하는 pandas 데이터프레임 객체 |
| 출력 | |
| df | 수치형 변수가 표준화된 pandas 데이터프레임 객체 |
| 매개변수 | |
| num\_cols | 정규화할 수치형 변수 목록 |
| 예외 처리 | |
| KeyError | 입력 데이터프레임에 필요한 컬럼이 존재하지 않을 때 발생. |
| ValueError | num\_cols에 숫자형 변수가 아닌 변수가 포함된 경우 발생 |
| 알고리즘 | |
| StandardScaler 객체를 생성한다. | |
| fit\_transform() 메서드를 사용하여 지정된 수치형 변수들을 표준화 한다. | |
| 추가 설명 | |
| 데이터 정규화는 변수의 스케일을 조정하여 변수 간의 영향력 차이를 줄이는 데 사용된다. | |
| 표준화는 데이터를 평균 0, 표준 편차 1을 갖도록 변환하는 일반적인 정규화 방법이다. | |
| scikit-learn 라이브러리는 StandardScaler 외에도 MinMaxScaler, RobustScaler 등 다양한 스케일링 방법을 제공한다. | |
| 데이터 정규화는 특히 거리 기반 알고리즘(K-Means, KNN 등)이나 경사 하강법을 사용하는 알고리즘 (선형 회귀, 로지스틱 회귀 등)에서 중요하다. | |

<4.6. 파생 변수 생성>



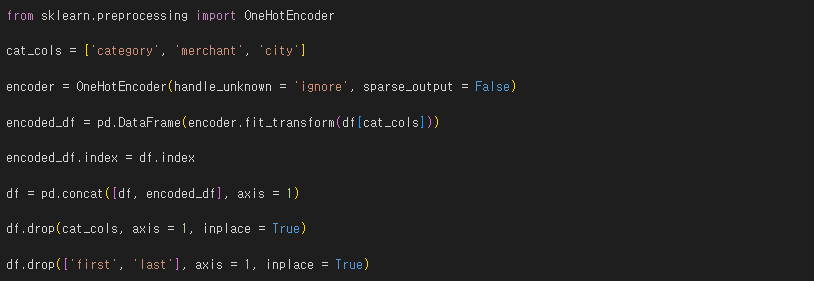
| 기능 | |
| --- | --- |
| 기존 변수를 활용하여 새로운 변수(파생 변수)를 생성한다. 파생 변수는 데이터 분석 및 모델링에 유용한 정보를 제공할 수 있다. | |
| 코드 설명 | |
| df[‘transaction\_per\_hour’] = df.groupby(‘hour’)[‘amt’].transform(‘count’) | 시간대별(hour)로 그룹화하여 거래 금액(amt) 컬럼의 개수를 세고, 각 행에 해당 시간대의 거래 횟수를 저장하는 새로운 컬럼 ‘transaction\_per\_hour’를 생성한다. |
| df[‘avg\_amt\_per\_weekday’] = df.groupby(‘weekday’)[‘amt’].transform(‘mean’) | 요일별(weekday)로 그룹화하여 거래 금액(amt) 컬럼의 평균을 계산하고, 각 행에 해당 요일의 평균 거래 금액을 저장하는 새로운 컬럼 ‘avg\_amt\_per\_weekday’를 생성한다. |
| df[‘customer\_id’] = df.groupby([‘lat’, ‘long’]).ngroup() | 위도(lat)와 경도(long)를 기준으로 그룹화하여 각 그룹에 고유한 ID를 부여하고, 새로운 컬럼 ‘customer\_id’에 저장한다. |
| df[‘transaction\_per\_customer’] = df.groupby(‘customer\_id’)[‘amt’].transform(‘count’) | ‘customer\_id’별로 그룹화하여 거래 금액(amt) 컬럼의 개수를 세고, 각 행에 해당 고객의 거래 횟수를 저장하는 새로운 컬럼 ‘transaction\_per\_customer’를 생성한다. |
| df[‘merchant\_id’] = df.groupby([‘merch\_lat’, ‘merch\_long’]).ngroup() | 가맹점 위도(merch\_lat)와 가맹점 경도(merch\_long)를 기준으로 그룹화하여 각 그룹에 고유한 ID를 부여하고, 새로운 컬럼 ‘merchant\_id’에 저장한다. |
| df[‘avg\_amt\_per\_merchant’] = df.groupby(‘merchant\_id’)[‘amt’].transform(‘mean’) | ‘merchant\_id’별로 그룹화하여 거래 금액(amt) 컬럼의 평균을 계산하고, 각 행에 해당 가맹점의 평균 거래 금액을 저장하는 새로운 컬럼 ‘avg\_amt\_per\_merchant’를 생성한다. |
| 입력 | |
| df | 시간대(hour), 요일(weekday), 위도/경도(lat,long), 가맹점 위도/경도(merch\_lat, merch\_long), 거래 금액(amt) 컬럼을 포함하는 pandas 데이터프레임 객체 |
| 출력 | |
| df | 시간대별 거래 횟수(transaction\_per\_hour), 요일별 거래 금액 평균(avg\_amt\_per\_weekday), 고객 ID(customer\_id), 고객별 거래 횟수(transaction\_per\_customer), 가맹점 ID(merchant\_id), 가맹점별 거래 금액 평균(avg\_amt\_per\_merchant) 컬럼이 추가된 pandas 데이터프레임 객체 |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| KeyError | 입력 데이터프레임에 필요한 컬럼이 없는 경우 발생 |
| 알고리즘 | |
| groupby() 함수를 사용하여 특정 변수를 기준으로 데이터를 그룹화한다. | |
| transform() 함수를 사용하여 각 그룹에 대한 집계 함수(예: count, mean)를 적용하고, 그 결과를 원본 데이터프레임에 새로운 컬럼으로 추가한다. | |
| ngroup() 함수를 사용하여 각 그룹에 고유한 ID를 부여한다. | |
| 추가 설명 | |
| 시간대별 거래 횟수는 시간대별로 거래가 얼마나 발생했는지 나타낸다. 특정 시간대에 사기 거래가 집중될 수 있으므로 파생변수로 추가한다. | |
| 요일별 거래 금액 평균은 요일별로 거래 금액이 얼마나 차이나는지 나타낸다. 특정 요일에 사기 거래 패터이 다를 수 있으므로 파생 변수로 추가한다. | |
| 고객별 거래 횟수는 각 고객이 얼마나 많은 거래를 했는지 나타낸다. 개인 정보 보호를 위해 고객 정보(first, last)는 사용하지 않고, 임의의 고객 ID를 생성한다. | |
| 가맹점별 거래 금액 평균은 특정 가맹점에서 사기 거래가 빈번하게 발생하는지 파악하기 위해 파생 변수로 추가한다. | |
| 파생 변수를 생성할 떄는 도메인 지식을 활용하여 분석 목적에 맞는 변수를 선택하는 것이 중요하다. | |
| 너무 많은 파생 변수를 생성하면 모델의 복잡성이 증가하고 과적합 문제가 발생할 수 있으므로, 적절한 수의 변수를 선택하는 것이 중요하다. | |

# <5. 데이터 분할 및 샘플링>

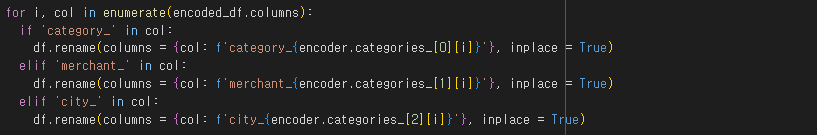


| 기능 | |
| --- | --- |
| 머신러닝 모델 학습 및 평가를 위해 데이터를 훈련, 검증, 테스트 세트로 분할한다. 이를 통해 모델의 일반화 성능을 평가하고 과적합을 방지할 수 있다. | |
| 코드 설명 | |
| from sklearn.model\_selection import train\_test\_split | scikit-learn 라이브러리의 model\_selection 모듈에서 train\_test\_split 함수를 임포트한다. |
| x = df.drop(‘is\_fraud’, axis = 1) | 데이터프레임에서 타겟 변수 ‘is\_fraud’를 제외한 나머지 변수를 특징 변수 x에 저장한다. |
| y = df[‘is\_fraud’] | 타겟 변수 ‘is\_fraud’를 y에 저장한다. |
| x\_train, x\_temp, y\_train, y\_temp = train\_test\_split(x, y, test\_size = 0.2, random\_state = 42) | train\_test\_split 함수를 사용하여 데이터를 8:2 비율로 학습 데이터와 나머지 데이터로 분할한다. random\_state는 난수 생성 시드를 설정하여 결과를 재현 가능하도록 한다. |
| x\_val, x\_test, y\_val, y\_test = train\_test\_split(x\_temp, y\_temp, test\_size = 0.5, random\_state = 42) | train\_test\_split 함수를 사용하여 나머지 데이터를 5:5 비율로 검증 데이터와 테스트 데이터로 분할한다. |
| 입력 | |
| df | 특징 변수와 타겟 변수(is\_fraud)를 포함하는 pandas 데이터프레임 객체 |
| 출력 | |
| x\_train | 훈련 데이터의 특징 변수 |
| y\_train | 훈련 데이터의 타겟 변수 |
| x\_val | 검증 데이터의 특징 변수 |
| y\_val | 검증 데이터의 타겟 변수 |
| x\_test | 테스트 데이터의 특징 변수 |
| y\_test | 테스트 데이터의 타겟 변수 |
| 매개변수 | |
| test\_size | 테스트 세트의 비율 (기본값: 0.25) |
| random\_state | 난수 생성 시드 |
| 예외 처리 | |
| KeyError | 입력 데이터프레임에 타겟 변수(is\_fraud)가 없는 경우 발생 |
| 알고리즘 | |
| train\_test\_split() 함수를 사용하여 데이터를 훈련 세트와 테스트 세트로 분할한다. | |
| 테스트 세트를 다시 검증 세트와 테스트 세트로 분할한다. | |
| 추가 설명 | |
| 데이터 분할은 머신러닝 모델의 일반화 성능을 평가하는 데 필수적인 단계이다. | |
| 훈련 세트는 모델 학습에 사용되고, 검증 세트는 모델의 하이퍼파라미터 튜닝에 사용되며, 테스트 세트는 최종 모델 성능 평가에 사용된다. | |
| random\_state를 설정함ㄴ 데이터 분할 결과를 재현할 수 있다. | |
| 데이터의 크기와 특성에 따라 분할 비율을 조정할 수 있다. | |

# <6. 데이터 변환 및 인코딩>







One-Hot Encoding으로 생성된 변수 이름 변경



One-Hot Encoding으로 생성된 변수 타입 변경

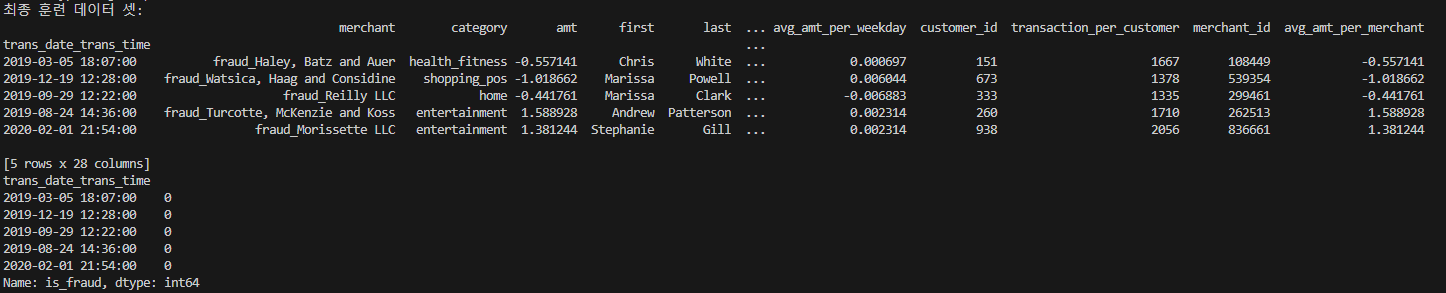
| 기능 | |
| --- | --- |
| 머신러닝 모델이 범주형 데이터를 이해하고 처리할 수 있도록 범주형 변수를 수치형으로 변환한다. | |
| 코드 설명 | |
| from sklearn.preprocessing import OneHotEncoder | scikit-learn 라이브러리의 preprocessing모듈에서 OneHotEncoder함수를 임포트한다. |
| encoder = OneHotEncoder(handle\_unknown = ‘ignore’, sparse\_output = False) | OneHotEncoder 객체를 생성한다. handle\_unknown = ‘ignore’은 학습 데이터에 없는새로운 값이 나타날 경우 무시하고, sparse\_output = False는 밀집 행렬을 반환하다. |
| encoded\_df = pd.DataFrame(encoder.fit\_transform(df[cat\_cols])) | One-Hot Encoding을 적용한다.  fit\_transform은 데이터에 One-Hot Encoding을 적용하고 변환된 데이터를 반환한다. |
| encoded\_df.index = df.index | 인덱스를 재설정 한다. |
| df = pd.concat([df, encoded\_df], axis = 1) | concat 함수를 사용하여 두 개의 데이터프레임(df, encoded\_df)를 병합한다. |
| df.drop(cat\_cols, axis = 1, inplace = True) | 데이터프레임에서 ‘cat\_cols’ 컬럼을 제거하고 원본 데이터프레임을 수정한다. |
| df.drop([‘first’, ‘last’], axis = 1, inplace = True) | 데이터프레임에서 ‘first’와 ‘last’ 컬럼을 제거한다. |
| 입력 | |
| df | 인코딩할 범주형 변수(cat\_cols)를 포함하는 pandas 데이터프레임 객체 |
| 출력 | |
| df | 범주형 변수가 one-hot 인코딩된 pandas 데이터프레임 객체 |
| 매개변수 | |
| cat\_cols | 인코딩할 범주형 변수 목록 |
| 예외 처리 | |
| KeyError | 입력 데이터프레임에 cat\_cols에 지정된 변수가 없는 경우 발생. |
| 알고리즘 | |
| OneHotEncoder 객체를 생성한다. handle\_unknown = ‘ignore’은 학습 데이터에 없는 새로운 범주 값이 나타나도 오류를 발생시키지 않고 무시한다. sparse\_output = False는 희소 행렬 대신 밀집 배열 형태로 출력한다. | |
| fit\_transform() 메서드를 사용하여 지정된 범주형 변수(cat\_cols)들을 one-hot 인코딩한다. 각 범주형 변수의 고유한 값들을 식별하고, 해당 값에 대해 새로운 이진 변수를 생성한다. | |
| 추가 설명 | |
| One-Hot Encoding 과정에서 범주형 변수의 고유한 값들이 각각 새로운 컬럼으로 생성되면서 컬럼 개수가 늘어났다. 데이터가 해당 범주에 속하는 값이 없는 경우, 0으로 채워지고, 해당 컬럼은 삭제 조치 한다. | |
| One-Hot 인코딩은 범주형 변수를 0과 1로 이루어진 여러개의 새로운 변수로 변환하는 방법이다. 각 변수는 해당 값에 해당하면 1, 아니면 0을 갖는다. | |
| 머신러닝 모델은 일반적으로 숫자형 데이터만 처리할 수 있으므로, 범주형 변수를 인코딩하는 것이 중요하다. | |
| One-Hot 인코딩은 범주형 변수 간의 순서나 크기를 고려하지 않고 각 값을 독립적인 변수로 처리한다. | |

# <7. 최종 데이터셋 준비 및 확인>

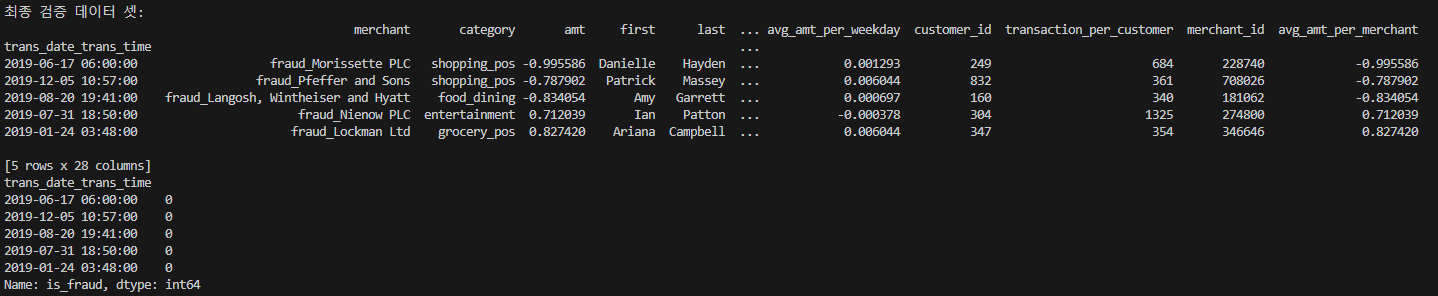




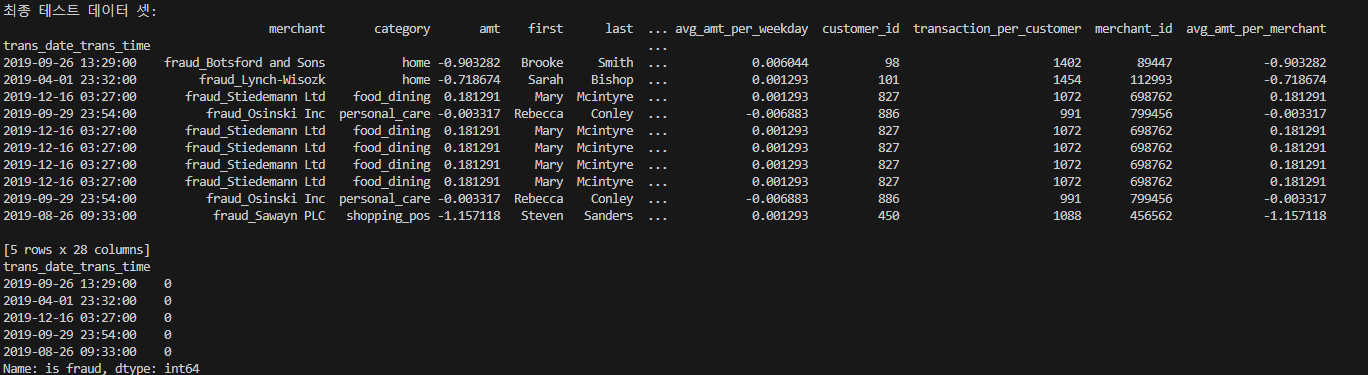




최종 훈련 데이터셋



최종 검증 데이터셋



최종 테스트 데이터 셋

| 기능 | |
| --- | --- |
| 머신러닝 모델 학습에 사용할 최종 데이터셋을 준비하고, 데이터의 형태와 특징을 확인하고 CSV 파일로 저장한다. | |
| 코드 설명 | |
| print(df.info()) | 최종 데이터프레임의 정보를 출력한다. (컬럼, 데이터 타입, Non-Null 값 개수 등) |
| print(df.head()) | 최종 데이터프레임의 처음 5개 행을 출력한다. |
| x\_train.to\_csv(‘x\_train.csv’, index = False) | 훈련 데이터의 특징 변수(x\_train)를 ‘x\_train.csv’ 파일로 저장한다.  index = False 옵션은 인덱스를 저장하지 않는다. |
| y\_train.to\_csv(‘y\_train.csv’, index = False) | 훈련 데이터의 타겟 변수 (y\_train)를 'y\_train.csv' 파일로 저장합니다. |
| x\_val.to\_csv(‘x\_val.csv’, index = False) | # 검증 데이터의 특징 변수 (x\_val)를 'x\_val.csv' 파일로 저장합니다. |
| y\_val.to\_csv(‘y\_val.csv’, index = False) | # 검증 데이터의 타겟 변수 (y\_val)를 'y\_val.csv' 파일로 저장합니다. |
| x\_test.to\_csv(‘x\_test.csv’, index = False) | 테스트 데이터의 특징 변수 (x\_test)를 'x\_test.csv' 파일로 저장합니다. |
| y\_test.to\_csv(‘y\_test.csv’, index = False) | 테스트 데이터의 타겟 변수 (y\_test)를 'y\_test.csv' 파일로 저장합니다. |
| 입력 | |
| df | 모든 전처리 과정을 거친 pandas 데이터프레임 객체 |
| 출력 | |
| 데이터프레임 정보 (컬럼, 데이터 타입, Non-Null 값 개수 등) | |
| 데이터프레임의 처음 5개 행 | |
| 최종 데이터셋을 CSV 파일로 저장 | |
| 매개변수 | |
| 없음 | |
| 예외 처리 | |
| 없음 | |
| 알고리즘 | |
| info() 함수를 사용하여 데이터프레임의 정보를 출력한다. | |
| head() 함수를 사용하여 데이터프레임의 처음 5개 행을 출력한다. | |
| to\_csv() 함수를 사용하여 데이터프레임을 CSV 파일로 저장한다. | |
| 추가 설명 | |
| One-Hot Encoding 과정에서 범주형 변수의 고유한 값들이 각각 새로운 컬럼으로 생성되면서 컬럼 개수가 늘어났다. 데이터가 해당 범주에 속하는 값이 없는 경우, 0으로 채워지고, 해당 컬럼은 삭제 조치 한다. | |
| 최종 데이터셋 준비 단계에서는 데이터의 형태와 특징을 최종적으로 확인하고, 머신러닝 모델 학습에 필요한 적합한 형태로 데이터를 가공한다. | |
| 필요에 따라 추가적인 데이터 정제 작업(중복 값 제거, 데이터 타입 변환 등)을 수행할 수 있다. | |