<코드 설명서 - 데이터 처리 및 분석>

| 순서 | 작업 내용 |
| --- | --- |
| 1 | 데이터 및 필요 라이브러리 로드 |
| 2 | 데이터 탐색적 분석 (EDA) |
| 데이터 구조 및 개요 파악 |
| 데이터 분포 및 특성 파악 |
| 3 | 데이터 전처리 |
| 결측치 처리 |
| 이상치 처리 |
| 데이터 정규화 및 스케일링 |
| 범주형 변수 처리 |
| 날짜 및 시간 데이터 처리 |
| 파생 변수 생성 |
| 4 | 데이터 분할 및 샘플링 |
| 5 | 데이터 변환 및 인코딩 |
| 6 | 최종 데이터셋 준비 및 확인 |

<데이터 로드>





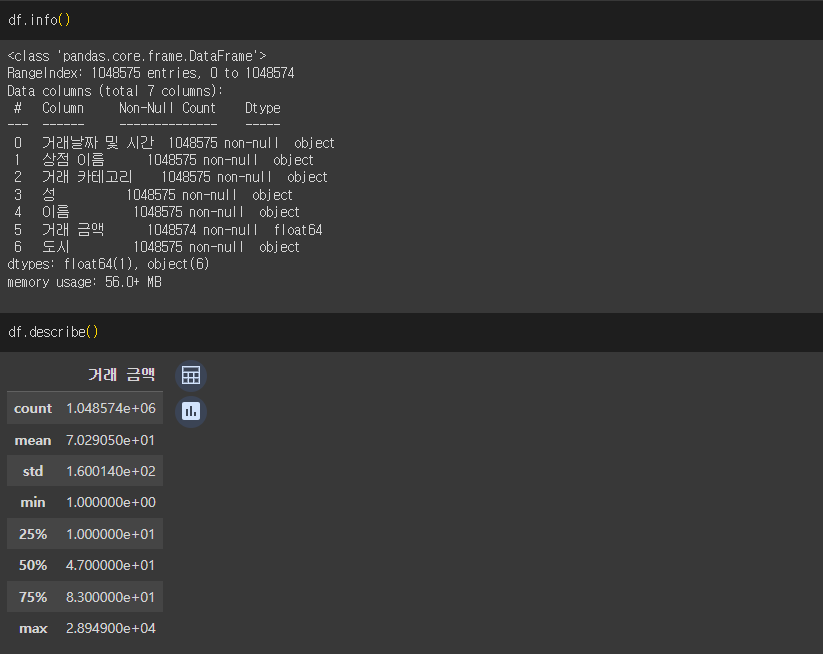
작업 설명: 준비된 데이터셋 ‘filtered\_data+Fraud.csv’를 데이터 프레임 형태로 읽어들인다.

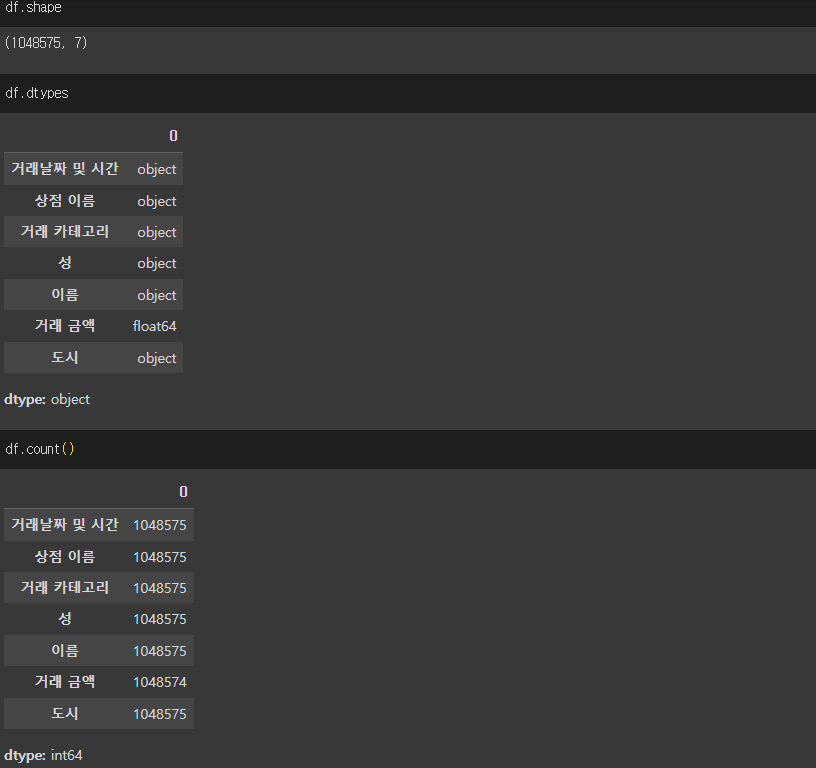
사용 함수 설명

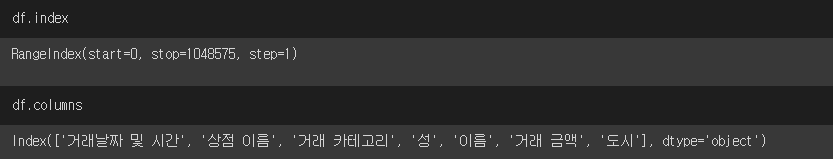
| import pandas as pd | Python의 라이브러리인 pandas를 pd라는 약어로 불러들임. |
| --- | --- |
| import numpy as np | Python의 라이브러리인 numpy를 np라는 약어로 불러들임. |
| import matplotlib.pyplot as plt | Python의 라이브러리인 matplotlib의 pyplot을 plt라는 약어로 불러들임. |
| import seaborn as sns | Python의 라이브러리인 seaborn를 sns라는 약어로 불러들임. |
| import matplotlib.font\_manager as fm | Python의 라이브러리인 matplotlib의 font\_manager를 fm라는 약어로 불러들임.  데이터 시각화 과정에서 데이터셋 내의 한글 문자가 깨지는 것을 방지하기 위해 사용한다. |
| read\_csv | Python의 라이브러리인 pandas에서 제공하는 함수로 csv파일을 로딩할 때 사용. |
| encoding = ‘cp949’ | 데이터셋 내용 중 utf-8으로 지원되지 않는 문자(한글)를 인코딩 하기 위해 사용. |

<데이터 탐색적 분석(EDA)>

* 데이터 구조 및 개요 파악





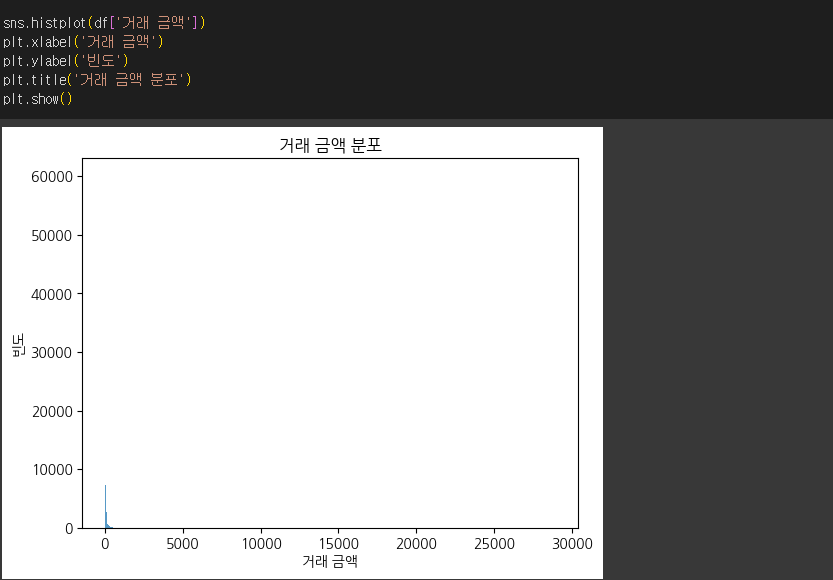


작업 설명: 준비된 데이터셋의 기본 구조 및 내용을 확인한다.

사용 함수 설명

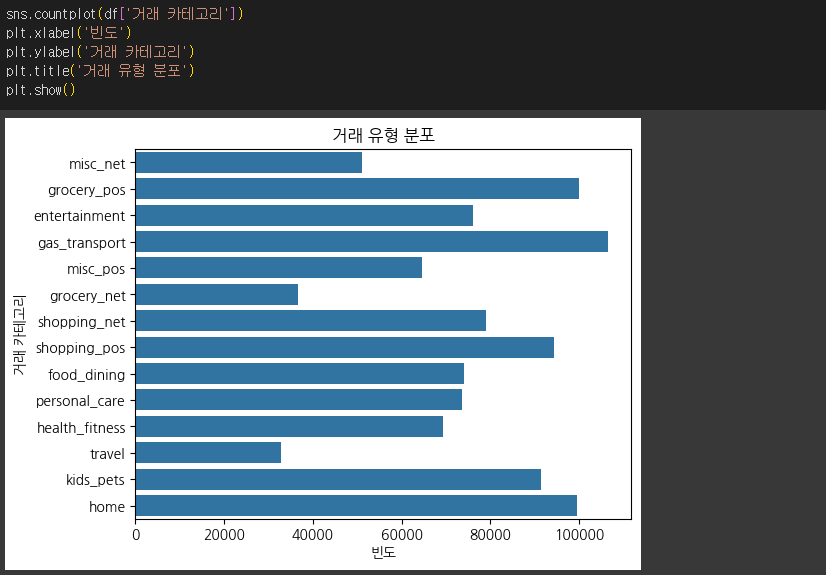
| df.info() | 데이터프레임의 간략한 요약을 출력한다. |
| --- | --- |
| df.describe() | 데이터 프레임의 기술 통계(중심 경향, 분산, 분포 모양 등)를 생성한다. |
| df.shape | 데이터 프레임의 차원을 나타내는 튜플을 반환한다. |
| df.dtypes | 데이터 프레임의 각 열의 데이터 유형을 반환한다. |
| df.count() | 데이터 프레임의 각 열이나 행에 대해 결측치를 제외한 셀의 수를 반환한다. |
| df.index | 데이터 프레임의 인덱스 (행 레이블)을 반환한다. |
| df.columns | 데이터 프레임의 열 레이블을 반환한다. |

* 데이터 분포 및 특성 파악



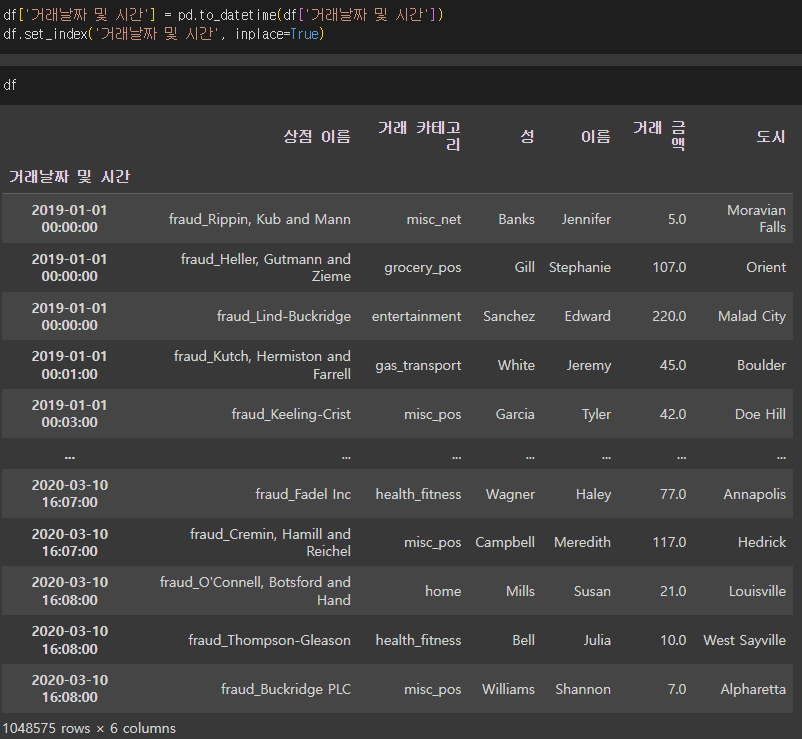
수치형 데이터 탐색: 히스토그램

작업 설명: 데이터셋에서 ‘거래 금액’만을 기준으로 하여 전체 거래 금액 빈도를 시각화한다.



범주형 데이터 탐색

작업 설명: 데이터셋에서 ‘거래 카테고리’를 기준으로 하여 전체 거래 유형을 시각화한다.



시계열 데이터 탐색

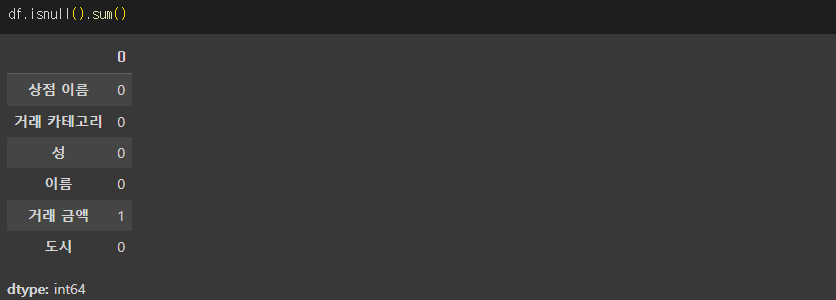
작업 설명: 데이터셋에서 ‘거래 날짜 및 시간’을 기준으로 하여 전체 데이터를 시간별 오름 차순으로 테이블화 한다.

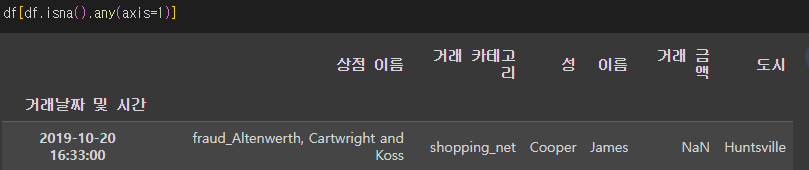
사용 함수 설명

| sns.histplot | Python의 라이브러리인 seaborn을 이용하여 히스토그램을 생성한다. |
| --- | --- |
| sns.countplot | Python의 라이브러리인 seaborn을 이용하여 카테고리 별 차트를 생성한다. |
| pd.to\_datetime() | 인수를 pandas의 datetime 객체로 변환한다. |
| df.set\_index() | 데이터 프레임의 인덱스를 설정한다. |

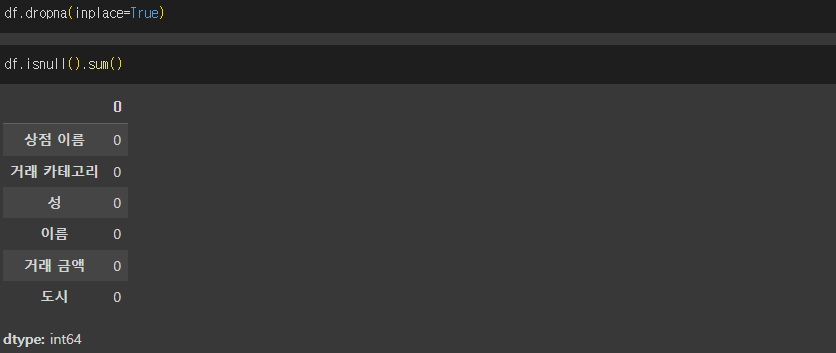
<데이터 전처리>

* 결측치 처리





결측치 확인



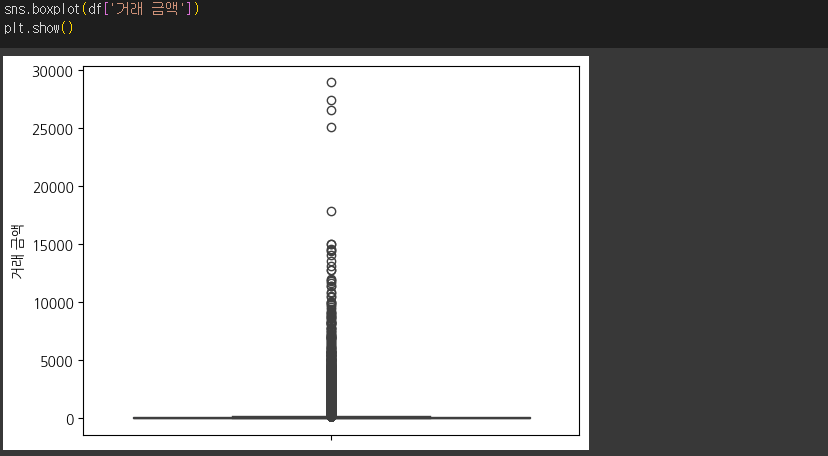
결측치 제거

작업 설명: 데이터셋 내의 결측치를 확인하고 처리한다.

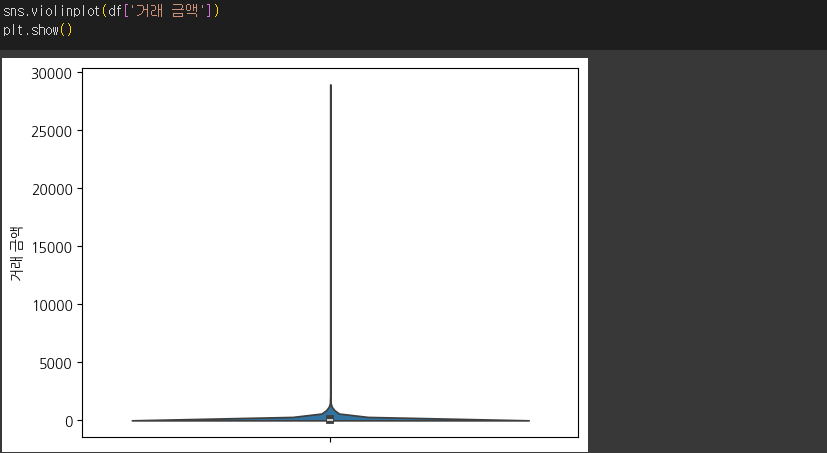
사용 함수 설명

| df.isnull().sum() | 데이터 프레임 내의 결측치를 확인하고 그 개수를 열 레이블 별로 출력한다. |
| --- | --- |
| df[df.isna().any(axis=1)] | 열 레이블을 기준으로 결측치가 확인되면 해당 행을 출력한다. |

* 이상치 처리

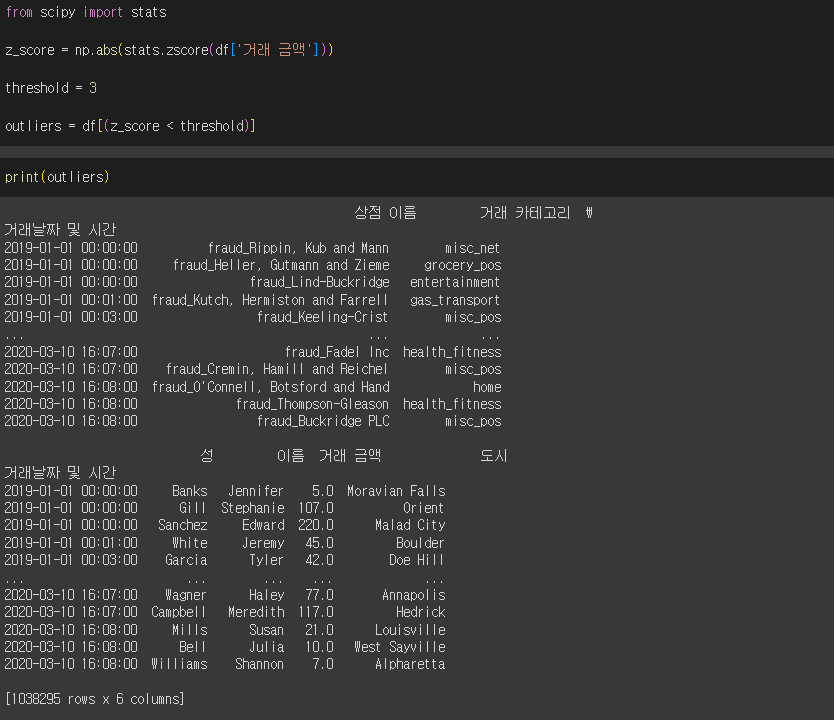


이상치 시각화 (boxplot)



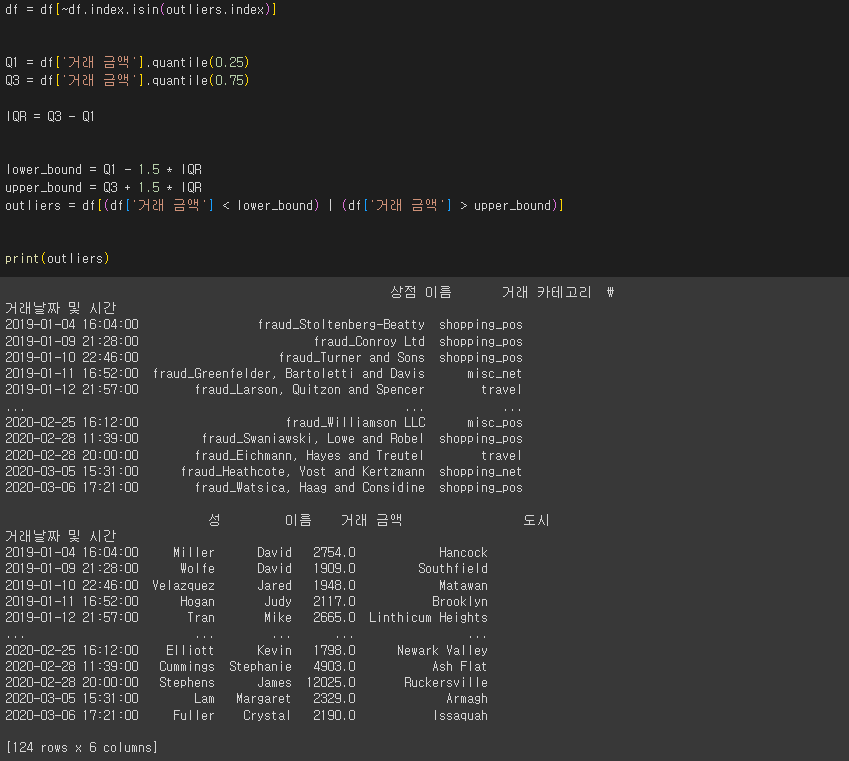
이상치 시각화 (violinplot)

작업 설명: 데이터셋 내의 이상치를 확인하고 시각화 한다.



Z-Score

작업 설명: 데이터 프레임의 ‘거래 내역’ 열에 대한 z-score를 계산하고 절대값을 취한다. 기준값(임계값)을 3으로 설정하고 z-score가 기준값 보다 작은 데이터를 이상치로 간주하여 outliers 변수에 저장한다.



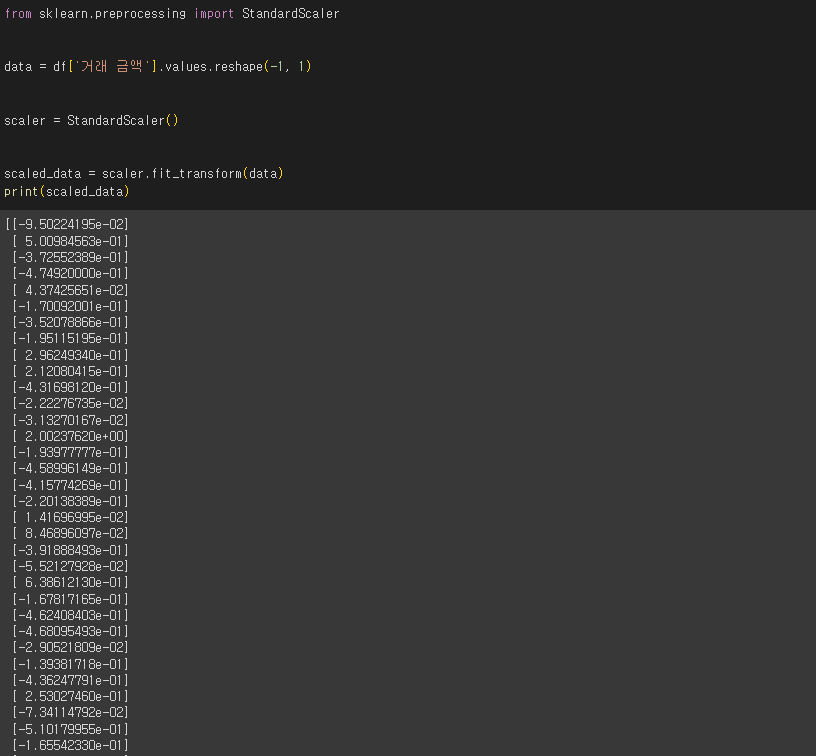
IQR

작업 설명: 데이터셋에서 z-score를 통해 발견 된 이상치 값들(outliers)을 제거하고 ‘거래 금액’의 1사분위수와 3사분위수를 계산하여 IQR을 계산한다. IQR을 사용하여 이상치의 상한 및 하한 경계를 계산하고 경계 밖에 있는 데이터들을 이상치로 식별하여 outliers 변수에 저장한다.

사용 함수 설명

| sns.boxplot() | Python의 라이브러리인 seaborn을 이용하여 박스플롯을 생성한다. |
| --- | --- |
| sns.violinplot() | Python의 라이브러리인 seaborn을 이용하여 바이올린플롯을 생성한다. |
| np.abs(stats.zscore()) | Python의 라이브러리인 scipy의 stats.zscore를 이용하여 Z-Score를 계산하고, Numpy의 abs를 이용하여 해당 결과값에 절대값을 취한다. |
| df[~df.index.isin(outliers.index)] | outliers 데이터프레임에서 이상값의 인덱스를 반환한 후, df 데이터프레임의 인덱스가 이상값의 인덱스에 있는지 확인하여 해당 값을 필터링하여 데이터프레임에 저장한다. |
| df[‘거래 금액’].quantile(0.25) | ‘거래 금액’ 열의 1사분위수(25번째 백분위수)를 계산한다. |
| df[‘거래 금액’].quantile(0.75) | ‘거래 금액’ 열의 3사분위수(75번째 백분위수)를 계산한다. |
| IQR = Q3 - Q1 | 3사분위수와 1사분위수의 차이인 사분위수 범위를 계산한다. |
| Q1 - 1.5 \* IQR | 이상값을 식별하기 위한 하한을 계산한다. |
| Q3 + 1.5 \* IQR | 이상값을 식별하기 위한 상한을 계산한다. |
| df[(df[‘거래 금액’] < lower\_bound) | (df[‘거래 금액’] > upper\_bound)] | 하한보다 낮거나 상한보다 높은 데이터 포인트를 식별하기 위해 데이터프레임을 필터링 한다. 필터링 된 데이터는 이상값으로 간주 된다. |

* 데이터 정규화 및 스케일링



데이터 정규화 및 스케일링에서 데이터 표준화 표준화 작업

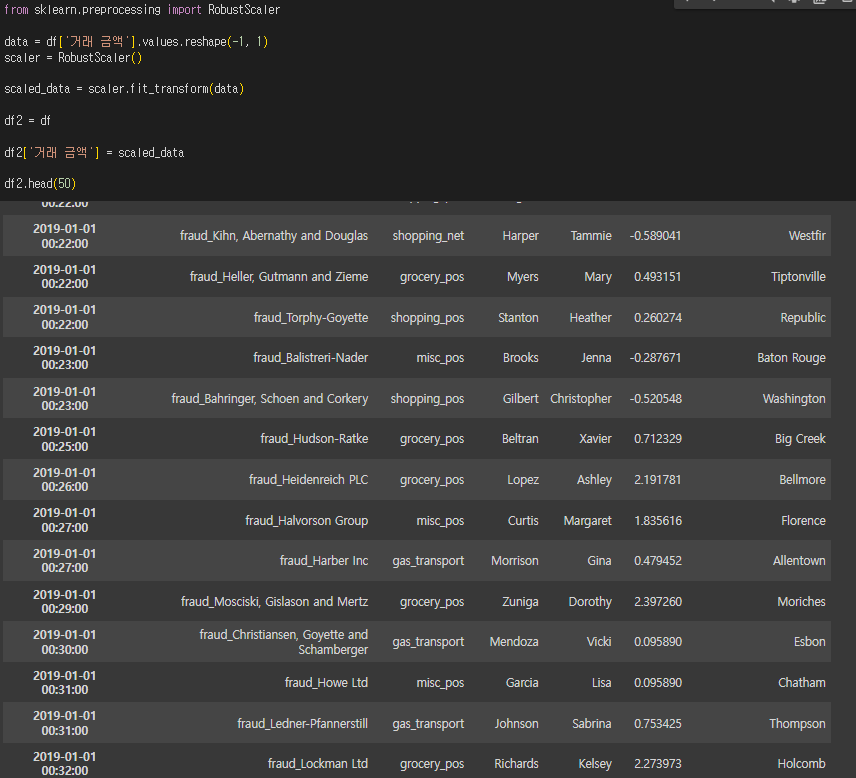
작업 설명: 특징들이 동일한 스케일을 갖도록 하여 모델이 특정 특징에 대해 지나치게 편향되지 않도록 한다.

경사하강법과 같은 최적화 알고리즘이 더 빠르게 수렴할 수 있게 한다.

사용 함수 설명

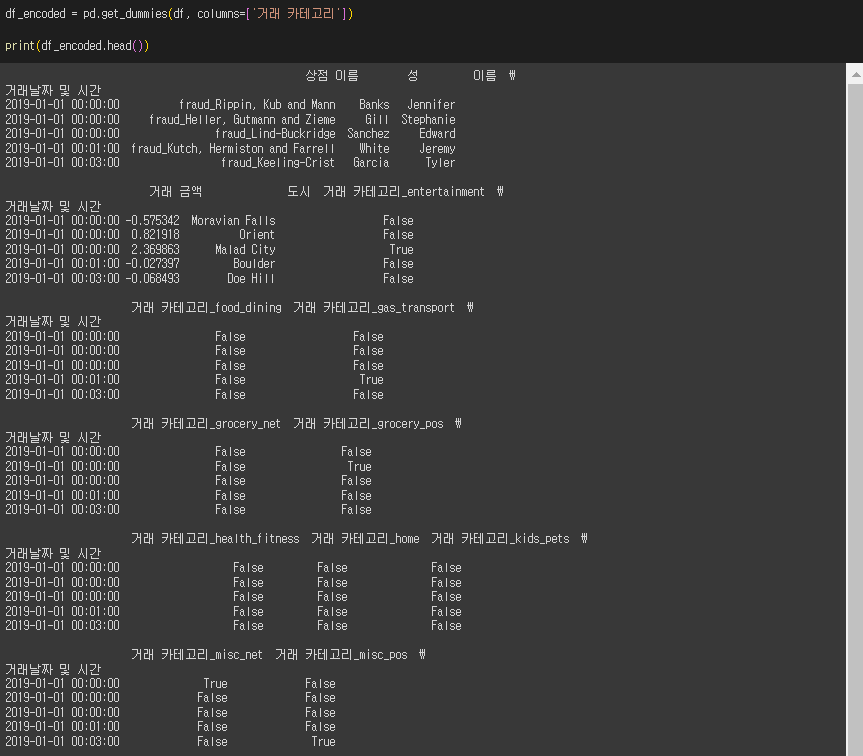
| df[‘거래 금액’].values.reshape(-1, 1) | 데이터프레임의 열 값들을 Numpy 배열로 추출한 후 reshape함수를 이용하여 배열 형태를 변경한다. reshape 함수의 인자로 받는 -1은 열의 개수를 기반으로 행의 개수를 자동으로 계산하고, 1은 변경될 배열의 열의 개수를 설정한다. |
| --- | --- |
| StandardScaler() | 스케일러를 데이터에 맞추고 평균을 뺀 다음 표준 편차로 나누어 반환한다.  엄밀한 표현식은 다음과 같다.  (df - mean) / std\_dev  이는 각 데이터 포인트에서 평균을 빼고, 표준 편차로 나누어 표준화를 수행 함을 의미한다. |
| fit\_transform(data) | scikit-learn에서 데이터 스케일링에 사용되는 메서드로, fit()과 transform()의 두 단계를 결합한 함수이다.  fit() 단계에서 스케일링에 필요한 매개변수를 계산하고, transform() 단계에서 계산된 매개변수를 사용하여 데이터에 스케일링을 적용한다. |

* 데이터 정규화 (RobustScaler)



작업 설명: 이상치가 사기 거래를 나타내는 유용한 정보일 수 있으므로, 제거하지 않고 스케일링을 통해 모델 학습에 활용한다.

* 범주형 변수 처리



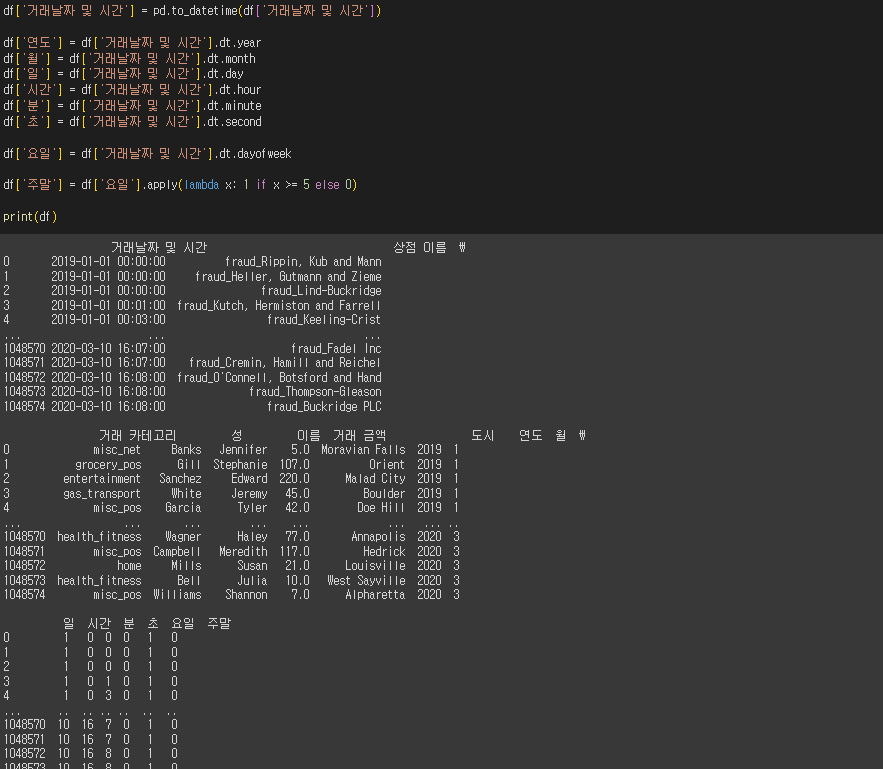
원-핫 인코딩

작업 설명: 범주형 변수인 ‘거래 카테고리’를 숫자형으로 변환하기 위해 ‘원-핫 인코딩을 시행한다.

사용 함수 설명

| pd.get\_dummies(df, columns=[‘거래 카테고리’]) | Pandas 라이브러리의 get\_dummies 함수를 사용하여 ‘거래 카테고리’ 열에 원-핫 인코딩을 적용한다. |
| --- | --- |

* 날짜 및 시간 데이터 처리



작업 설명: 날짜 및 시간에 따른 특징을 추출하여 모델에 더 많은 정보를 제공하기 위해 ‘거래날짜 및 시간’ 컬럼을 datetime 객체로 변환한다.

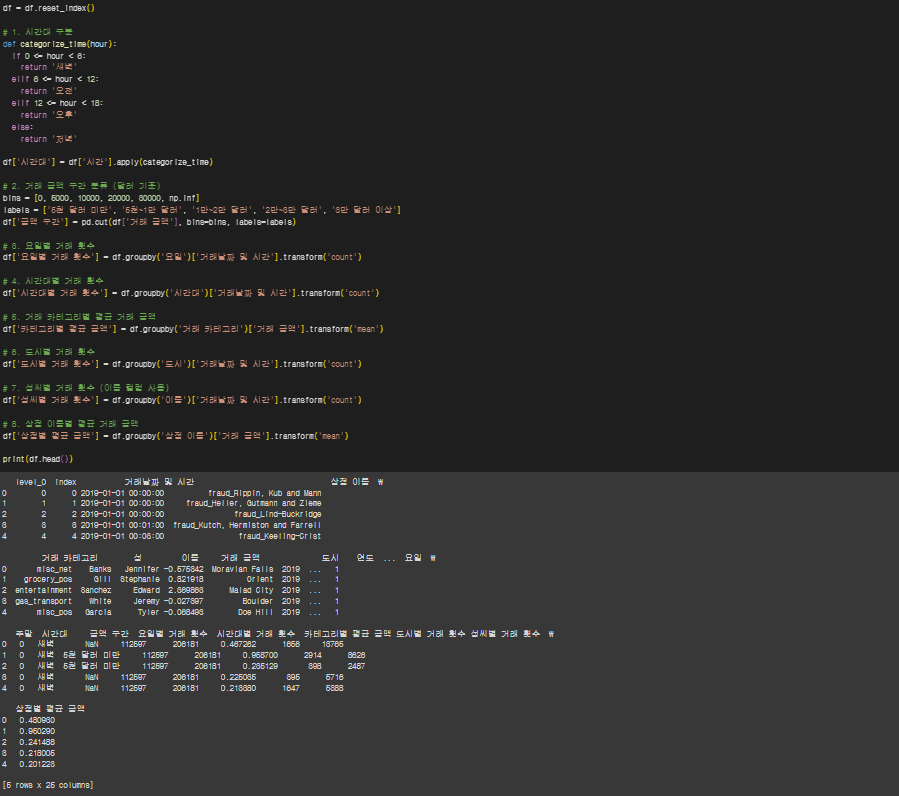
사용 함수 설명

| pd.to\_datetime() | 문자열 또는 숫자 형식의 데이터를 datetime 객체로 변환한다. |
| --- | --- |

비고: 결측치 및 이상치 처리 과정에서 df.set\_index(‘거래날짜 및 시간’, inplace=True) 부분에서 해당 열이 삭제되어 에러가 발생 함.

해결 방안으로, 날짜 및 시간 데이터 처리를 먼저 수행한 후 결측치 및 이상치 처리 등 다른 작업을 수행하거나, set\_index 이후에 다시 컬럼을 생성하는 방법이 있으나, 후자의 방안 시도시, 이전에 수행했던 결측치 및 이상치 처리 과정에서 변경된 데이터가 초기화될 수 있어 전자의 방안으로 수행 함.

* 파생 변수 생성

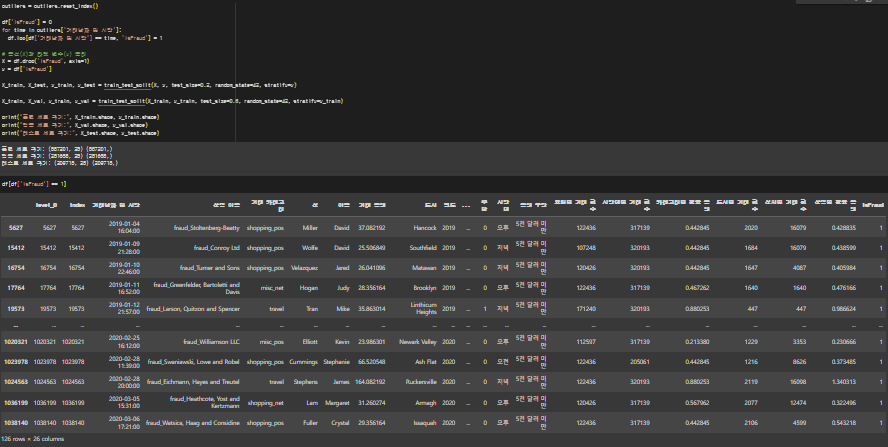


작업 설명: 파생 변수를 생성하기 위한 작업을 수행한다. 생성한 파생 변수로는 ‘시간대 구분’, ‘거래 금액 구간 분류’, ‘요일별 거래 횟수’, ‘시간대별 거래 횟수’, ‘거래 카테고리별 평균 거래 금액’, ‘도시별 거래 횟수’, ‘성씨별 거래 횟수’, ‘상점 이름별 평균 거래 금액’ 이다.

사용 함수 설명

| df.reset\_index() | 데이터프레임의 인덱스를 재설정 한다. 기존 인덱스는 열로 변환되고, 새로운 정수 인덱스가 생성된다. |
| --- | --- |
| df.apply() | 데이터프레임의 각 행 또는 열에 함수를 적용한다. |
| pd.cut() | 연속형 데이터를 범주형 데이터로 변환한다. |
| df.groupby() | 데이터프레임을 특정 열의 값을 기준으로 그룹화 한다. |
| df.groupby().transform() | 그룹화된 데이터에 함수를 적용하고, 원본 데이터프레임과 동일한 형태의 결과를 반환한다. |

<데이터 분할 및 샘플링>



작업 설명: 데이터 셋에서 추출한 이상치(평균 거래 금액 대비 더 높은 거래 금액)를 이상 거래 탐지의 기준으로 하여 훈련 세트, 검증 세트, 테스트 세트를 분할하여 생성한다.

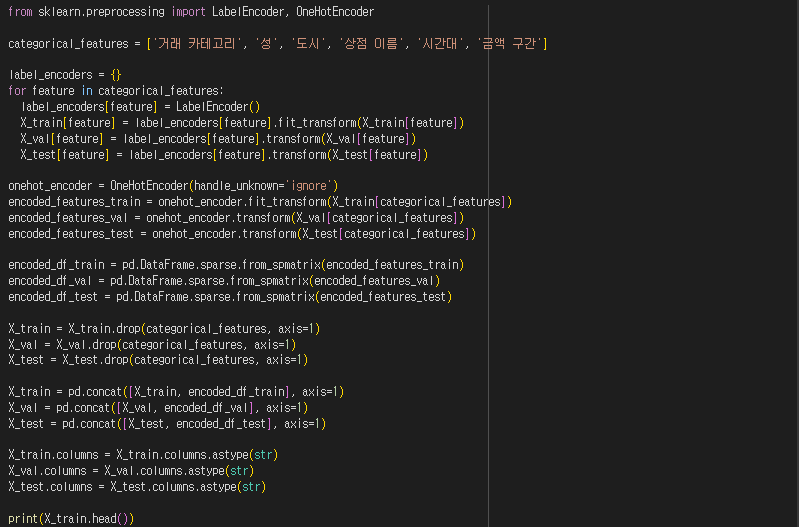
사용 함수 설명

| outliers = outliers.reset\_index() | outliers 데이터프레임의 인덱스를 컬럼으로 재설정 한다. |
| --- | --- |
| df[‘isFraud’] = 0  for time in outliers[‘거래날짜 및 시간’]:  df.loc[df[‘거래날짜 및 시간’] == time, isFraud’] = 1 | isFraud 열을 생성하고 모든 값을 0으로 설정한 후, outliers 데이터프레임에 있는 값들 중 ‘거래날짜 및 시간’ 값을 기반으로 원본 데이터프레임(df)에서 isFraud 열의 값을 1로 설정한다. |
| X = df.drop(‘isFraud’, axis = 1)  y = df[‘isFraud’] | 특성(X)과 타겟 변수(y)를 분리한다.  isFraud 열을 제외한 모든 열을 X에 저장하고, isFraud 열을 y에 저장한다. |
| train\_test\_split(X, y, test\_size, random\_state, stratify) | train\_test\_split 함수를 사용하여 데이터를 훈련 세트와 테스트 세트로 분할한다.  test\_size 매개변수는 테스트 세트의 크기를, random\_state 매개변수는 재현성을 위해 설정하고, stratify 매개변수는 타겟 변수의 클래스 분포를 유지한다. |

비고:

* 훈련 세트 : 테스트 세트 = 80 : 20
* 훈련 세트 : 검증 세트 = 73 : 30

<데이터 변환 및 인코딩>

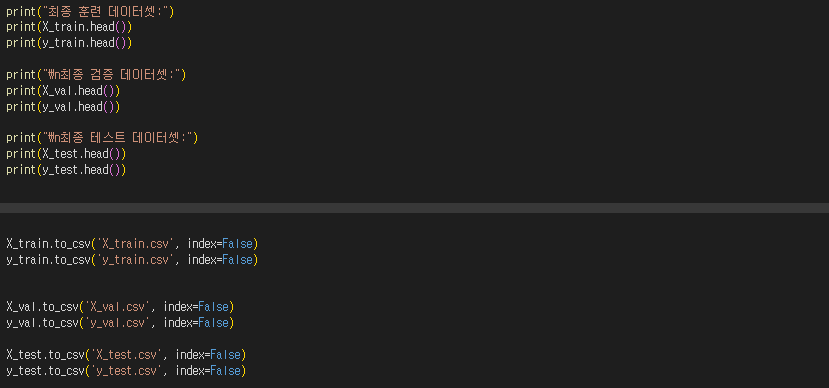


작업 설명: 범주형 데이터를 수치형 데이터로 변환한다.

사용 함수 설명

| LabelEncoder() | 범주형를 숫자로 변환한다.  fit\_transform 메서드를 사용하여 훈련 데이터에 인코더를 적용하고, transform 메서드를 사용하여 검증 및 테스트 데이터에 적용한다. |
| --- | --- |
| OneHotEncoder() | 숫자로 변환된 범주형 특성을 one-hot 인코딩 한다.  handle\_unknown=’ignore’ 옵션은 테스트 데이터에 훈련 데이터에 없는 범주가 있을 경우 무시하도록 설정한다. |
| sparese.from\_spmatrix() | scipy 라이브러리의 희소 행렬(sparse matrix)을 pandas의 데이터프레임으로 변환한다. |
| pd.concat() | 여러 개의 데이터프레임 객체를 연결하여 하나의 객체로 합친다. |

<최종 데이터셋 준비 및 확인>



작업 설명: 최종적으로 처리 된 데이터 셋을 확인하고 저장한다.

비고:

* 훈련 데이터셋 저장: ‘X\_train.csv’, ‘y\_train.csv’
* 검증 데이터셋 저장: ‘X\_val.csv’, ‘y\_val.csv’
* 테스트 데이터셋 저장: ‘X\_test.csv’, ‘y\_test.csv’