In [1]:

```python
import pandas as pd
import numpy as np
df = pd.read_csv('fraudTest.csv')
```

In [2]:

```python
df
```

Out[2]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott	M
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams	F
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez	F
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05	Brian	Williams	M
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19	Nathan	Massey	M
...	...	...	...	...	...	...	...	...	...
555714	555714	2020-12-31 23:59:07	30560609640617	fraud_Reilly and Sons	health_fitness	43.77	Michael	Olson	M
555715	555715	2020-12-31 23:59:09	3556613125071656	fraud_Hoppe-Parisian	kids_pets	111.84	Jose	Vasquez	M
555716	555716	2020-12-31 23:59:15	6011724471098086	fraud_Rau-Robel	kids_pets	86.88	Ann	Lawson	F
555717	555717	2020-12-31 23:59:24	4079773899158	fraud_Breitenberg LLC	travel	7.99	Eric	Preston	M
555718	555718	2020-12-31 23:59:34	4170689372027579	fraud_Dare-Marvin	entertainment	38.13	Samuel	Frey	M

555719 rows × 23 columns

In [84]:

```python
df.columns
```

Out[84]:

```
Index(['amt', 'first', 'last', 'zip', 'city_pop', 'job', 'trans_num',
       'unix_time', 'is_fraud', 'distance', 'age', 'gender', 'merchant num',
       'merchant size', 'city size', 'month', 'dayzone', 'weekend',
       'num_cc_num', 'type', 'region'],
      dtype='object')
```

lat, long,merch_lat,merch_long

convert lat, long, merch_lat, merch_long to distance

new column: distance

drop column: lat, long, merch_lat, merch_long

drop: 'Unnamed: 0'

In [16]:

```python
#以后只能20w跑一次，跑到98w会炸
#df = pd.read_csv('fraudtrain_encodeddistanceonly.csv')
#df
```

Out[16]:

	Unnamed: 0	Unnamed: 0.1	trans_date_trans_time	cc_num	merchant	category	amt	first
0	0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer
1	1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie
2	2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward
3	3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy
4	4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler
...	...	...	...	...	...	...	...	...
1296670	1296670	1296670	2020-06-21 12:12:08	30263540414123	fraud_Reichel Inc	entertainment	15.56	Erik
1296671	1296671	1296671	2020-06-21 12:12:19	6011149206456997	fraud_Abernathy and Sons	food_dining	51.70	Jeffrey
1296672	1296672	1296672	2020-06-21 12:12:32	3514865930894695	fraud_Stiedemann Ltd	food_dining	105.93	Christopher
1296673	1296673	1296673	2020-06-21 12:13:36	2720012583106919	fraud_Reinger, Weissnat and Strosin	food_dining	74.90	Joseph
1296674	1296674	1296674	2020-06-21 12:13:37	4292902571056973207	fraud_Langosh, Wintheiser and Hyatt	food_dining	4.30	Jeffrey

1296675 rows × 25 columns

In [1]:

```python
#import geopy.distance
#for i in range(len(df['lat'])):
    #print(i)
    #coords_1 = (df['lat'][i], df['long'][i])
    #coords_2 = (df['merch_lat'][i], df['merch_long'][i])
    #df.loc[i,'distance'] = geopy.distance.geodesic(coords_1, coords_2).km
```

In [6]:

```python
#drop lat, long, merch_lat, merch_long
df.drop(['lat', 'long', 'merch_lat', 'merch_long'], axis=1, inplace=True)
df.to_csv('fraudtest_encodeddistanceonly.csv')
df.drop(['Unnamed: 0'], axis=1, inplace=True)
```

In [7]:

```python
#check if there is NaN
df['distance'].isnull().values.any()
```

Out[7]:

```
False
```

dob

encode date of birth to age

new cloumn: age

drop column: dob

In [28]:

```python
df = pd.read_csv('fraudtest_encodeddistanceonly.csv')
```

In [29]:

```python
from datetime import date
from datetime import datetime

#age function
def age(birthdate):
    today = date.today()
    birthdate = datetime.strptime(birthdate,'%Y-%m-%d')
    age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))
    return age

#apply
df["age"] = df["dob"].apply(age)

#drop dob
df.drop('dob', axis=1, inplace=True)
```

trans_date_trans_time

convert trans_date to month(1-12) and weekend(weekday=0,weekend=1)

convert trans_time to broad time of day categories(0:00:00-3:59:59 to before_dawn, 4:00:00-7:59:59 to early_morning,8:00:00-11:59:59 to morning,12:00:00-15:59:59 to afternoon,16:00:00-19:59:59 to evening ,20:00:00-23:59:59 to night)

(备选)convert trans_time to broad time of day categories(0am-2:59am to before_dawn,3am-5:59am to early_morning, 6am-8:59am to late_morning,9am-11:59am to mid_day, 12pm-2:59pm to afternoon,3pm-5:59pm to evening,6pm-8:59pm to early_night,9pm-11:59pm to late_night)

new columns: month, dayzone,weekend

drop columns: trans_date_trans_time

In [30]:

```python
#function
def month_func(transdate):
    transdate = datetime.strptime(transdate,'%Y-%m-%d %H:%M:%S')
    month = transdate.month
    return month

#apply to month
df["month"] = df['trans_date_trans_time'].apply(month_func)
```

In [31]:

```python
#0:00:00-3:59:59 to before_dawn,    4:00:00-7:59:59 to early_morning
#8:00:00-11:59:59 to morning,       12:00:00-15:59:59 to afternoon
#16:00:00-19:59:59 to evening       20:00:00-23:59:59 to night
#function
def dayzone_func(transtime):
    dayzone = ''
    transtime = datetime.strptime(transtime,'%Y-%m-%d %H:%M:%S')
    if transtime.hour>=0 and transtime.hour<=3:
        dayzone = 'before_dawn'
    if transtime.hour>=4 and transtime.hour<=7:
        dayzone = 'early_morning'
    if transtime.hour>=8 and transtime.hour<=11:
        dayzone = 'morning'
    if transtime.hour>=12 and transtime.hour<=15:
        dayzone = 'afternoon'
    if transtime.hour>=16 and transtime.hour<=19:
        dayzone = 'evening'
    if transtime.hour>=20 and transtime.hour<=23:
        dayzone = 'night'
    return dayzone

#apply to month
df["dayzone"] = df['trans_date_trans_time'].apply(dayzone_func)
```

In [32]:

```python
#weekday or weekend
def get_weekend(transdate):
    transdate = datetime.strptime(transdate,'%Y-%m-%d %H:%M:%S').weekday()
    if transdate <= 4:
        weekend = 0
    else:
        weekend = 1
    return weekend

#apply
df["weekend"] = df["trans_date_trans_time"].apply(get_weekend)
```

```
#drop trans_date_trans_time
df.drop('trans_date_trans_time', axis=1, inplace=True)
```

Gender

encode gender to 0/1 where 0 = female and 1 = male

In [33]:

```
#change name
df.rename(columns={"gender":"gen"},inplace = True)

#gender_ func function
def gender_func(item):
    gender = ""
    item = item
    if item == "F":
        gender = 0
    else:
        gender = 1
    return gender

#apply
df["gender"] = df["gen"].apply(gender_func)

#drop gen
df.drop('gen', axis=1, inplace=True)
```

City and City_pop

encode city to city size based on the four quartiles of the city population for each city

city size: small, medium, medium_large, large

new column: city size

drop column: city, street

In [34]:

```
fromtrain = pd.read_csv("fraudTrain.csv")
```

In [35]:

```
import collections
#connect city and city_pop
nums = len(df['city'])
city_pop = {}
for i in range(nums):
    city = df['city'][i]
    city_pop[city] = df['city_pop'][i]
city_pop

#check each city
a=list(city_pop.keys())
counter=collections.Counter(a)
#print(counter)

#4 quartiles
df.city_pop.quantile([0.25,0.5,0.75])
```

Out[35]:

```
0.25      741.0
0.50     2408.0
0.75    19685.0
Name: city_pop, dtype: float64
```

In [36]:

```
#change city_pop to city_size(small,medium,merdium_large,large)
def city_size(num):
    city_size = ""
    num = num
    if num < 743:
        city_size = "small"
    elif num >= 743 and num <2456:
        city_size = "medium"
    elif num >= 2456 and num <20328:
        city_size = "medium_large"
    else:
        city_size = "large"
    return city_size

#apply
df["city size"] = df["city_pop"].apply(city_size)

#drop city, street
df.drop('city', axis=1, inplace=True)
df.drop('street', axis=1, inplace=True)
```

Merchant

encode merchant to merchant size based on the four quartiles of the number of transactions for each merchant merchant size: small, medium, medium_large, large

new column: merchant num, merchant size

drop column: merchant

In [ ]:

In [37]:

```
#merchant size
m = fromtrain['merchant'].value_counts()
m

#merchant_size dictionary
merchant_size = dict(m)

#check each merchant size
a=list(merchant_size.values())
counter=collections.Counter(a)
#print(counter)

#apply size to df
df['merchant size'] = df.merchant.apply(lambda x:merchant_size[x])

#drop merchant
df.drop('merchant', axis=1, inplace=True)
```

In [38]:

```
#4 quartiles
print(m.quantile([0.25,0.5,0.75]))
#rename
df.rename(columns={"merchant size":"merchant num"},inplace = True)
df.head()
```

```
0.25    1592.0
0.50    1863.0
0.75    2345.0
Name: merchant, dtype: float64
```

Out[38]:

	Unnamed: 0	Unnamed: 0.1	cc_num	category	amt	first	last	state	zip	city_pop	...	unix_time	is_fraud
0	0	0	2291163933867244	personal_care	2.86	Jeff	Elliott	SC	29209	333497	...	1371816865	0
1	1	1	3573030041201292	personal_care	29.84	Joanne	Williams	UT	84002	302	...	1371816873	0
2	2	2	3598215285024754	health_fitness	41.28	Ashley	Lopez	NY	11710	34496	...	1371816893	0
3	3	3	3591919803438423	misc_pos	60.05	Brian	Williams	FL	32780	54767	...	1371816915	0
4	4	4	3526826139003047	travel	3.19	Nathan	Massey	MI	49632	1126	...	1371816917	0

5 rows × 22 columns

In [39]:

```
#convert merchant_num to merchant_size(small,medium,merdium_large,large)
def merchant_size(num):
    merchant_size = ""
    num = num
    if num < 1592:
        merchant_size = "small"
    elif num >= 1592 and num <1863:
        merchant_size = "medium"
    elif num >= 1863 and num <2345:
        merchant_size = "medium_large"
    else:
        merchant_size = "large"
    return merchant_size

df["merchant size"] = df["merchant num"].apply(merchant_size)
```

cc_num

convert cc_num to number of digits in cc_num

new column: num_cc_num

drop column: cc_num

In [40]:

```
#func
def get_num_digits(num):
    num = len(str(num))
    return num

#apply
df["num_cc_num"] = df["cc_num"].apply(get_num_digits)

#drop cc_num
df.drop('cc_num', axis=1, inplace=True)
```

category

convert category to type(net = online, pos = pos_machine, else = else)

new column: type

drop column: category

In [41]:

```
#check category type
df['category'].value_counts()

def get_type(cat):
    type = ''
    if 'net' in cat:
        type = 'online'
    elif 'pos' in cat:
        type = 'pos_machine'
    else:
        type = 'other'
    return type

#apply
df['type'] = df['category'].apply(get_type)

#drop category
df.drop('category', axis=1, inplace=True)
```

state

convert state to region(westcoast_north,westcoast_south, midwest_north,midwest_south,mideast_north,mideast_south,eastcoast_north, eastcoast_south)

new column: region

drop column: state

In [42]:

```
#map state to region
```

```
westcoast_north_states = ['WA','OR','AK']
westcoast_south_states = ['CA','HI']
midwest_north_states=['MT','ID','WY','ND','SD','NE']
midwest_south_states=['NV','UT','AZ','CO','NM','KS','OK','TX']
mideast_north_states=['MN','IA','WI','IL','MI','IN','OH','WV','PA','VT']
mideast_south_states=['MO','AR','LA','KY','TN','MS','AL']
eastcoast_north_states = ['NY','ME','NH','MA','RI','CT','NJ','DE','MD','VA']
eastcoast_south_states = ['NC','SC','GA','FL']

#func
def get_region(state):
    region = ''
    if state in westcoast_north_states:
        region = 'westcoast_north'
    elif state in westcoast_south_states:
        region = 'westcoast_south'
    elif state in midwest_north_states:
        region = 'midwest_north'
    elif state in midwest_south_states:
        region = 'midwest_south'
    elif state in mideast_north_states:
        region = 'mideast_north'
    elif state in mideast_south_states:
        region = 'mideast_south'
    elif state in eastcoast_north_states:
        region = 'eastcoast_north'
    else:
        region = 'eastcoast_south'
    return region

#apply
df['region'] = df['state'].apply(get_region)

#drop state
df.drop('state', axis=1, inplace=True)
```

In [43]:

```
#check NaN
df['region'].isna().any()
```

Out[43]:

```
False
```

In [83]:

```
#drop first, last,job,trans_num
#df.drop(['first', 'last','job','trans_num'], axis=1, inplace=True)
df
```

Out[83]:

	amt	first	last	zip	city_pop	job	trans_num	unix_time	is_fraud
0	4.97	Jennifer	Banks	28654	3495	Psychologist, counselling	0b242abb623afc578575680df30655b9	1325376018	0
1	107.23	Stephanie	Gill	99160	149	Special educational needs teacher	1f76529f8574734946361c461b024d99	1325376044	0
2	220.11	Edward	Sanchez	83252	4154	Nature conservation officer	a1a22d70485983eac12b5b88dad1cf95	1325376051	0
3	45.00	Jeremy	White	59632	1939	Patent attorney	6b849c168bdad6f867558c3793159a81	1325376076	0
4	41.96	Tyler	Garcia	24433	99	Dance movement psychotherapist	a41d7549acf90789359a9aa5346dcb46	1325376186	0
...	...	...	...	...	...	...	...	...	...
1296670	15.56	Erik	Patterson	84735	258	Geoscientist	440b587732da4dc1a6395aba5fb41669	1371816728	0
1296671	51.70	Jeffrey	White	21790	100	Production assistant, television	278000d2e0d2277d1de2f890067dcc0a	1371816739	0
1296672	105.93	Christopher	Castaneda	88325	899	Naval architect	483f52fe67fabef353d552c1e662974c	1371816752	0
1296673	74.90	Joseph	Murray	57756	1126	Volunteer coordinator	d667cdcbadaaed3da3f4020e83591c83	1371816816	0
1296674	4.30	Jeffrey	Smith	59871	218	Therapist, horticultural	8f7c8e4ab7f25875d753b422917c98c9	1371816817	0

1296675 rows × 21 columns

In [23]:

```
df.to_csv('fraudtest_encoded_w_first,last,job,trans_num.csv')
```

In [44]:

```
df2 = df
df2.drop(['first', 'last','job','trans_num'], axis=1, inplace=True)
df2.to_csv('fraudtest_encoded.csv')
```

One hot encoding

In [45]:

```
fraud_df = pd.read_csv('fraudtest_encoded.csv')
fraud_df
```

Out[45]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	amt	zip	city_pop	unix_time	is_fraud	distance	age	month	dayzone	weeker
0	0	0	0	2.86	29209	333497	1371816865	0	24.613746	54	6	afternoon	
1	1	1	1	29.84	84002	302	1371816873	0	104.834043	32	6	afternoon	
2	2	2	2	41.28	11710	34496	1371816893	0	59.204796	51	6	afternoon	
3	3	3	3	60.05	32780	54767	1371816915	0	27.615117	34	6	afternoon	
4	4	4	4	3.19	49632	1126	1371816917	0	104.423175	66	6	afternoon	
...	...	...	...	...	...	...	...	...	...	...	...	...	
555714	555714	555714	555714	43.77	63453	519	1388534347	0	77.032467	56	12	night	
555715	555715	555715	555715	111.84	77566	28739	1388534349	0	100.023736	22	12	night	
555716	555716	555716	555716	86.88	99323	3684	1388534355	0	80.887812	40	12	night	
555717	555717	555717	555717	7.99	83643	129	1388534364	0	53.060882	56	12	night	
555718	555718	555718	555718	38.13	73034	116001	1388534374	0	72.380990	28	12	night	

555719 rows × 20 columns

In [46]:

```
#merchant size
from sklearn.preprocessing import OneHotEncoder

#creating instance of one-hot-encoder
encoder = OneHotEncoder(handle_unknown='ignore')

#perform one-hot encoding on 'merchant size' column
encoder_merch_size = pd.DataFrame(encoder.fit_transform(fraud_df[['merchant size']]).toarray())
encoder_merch_size.columns = ['merch_size_large','merch_size_medium','merch_size_mediumlarge','merch_size_small

#merge one-hot encoded columns back with original DataFrame
merch_size_df = fraud_df.join(encoder_merch_size)
merch_size_df.drop('merchant size', axis=1, inplace=True)

merch_size_df
```

Out[46]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	amt	zip	city_pop	unix_time	is_fraud	distance	age	...	gender	city size
0	0	0	0	2.86	29209	333497	1371816865	0	24.613746	54	...	1	large
1	1	1	1	29.84	84002	302	1371816873	0	104.834043	32	...	0	small
2	2	2	2	41.28	11710	34496	1371816893	0	59.204796	51	...	0	large
3	3	3	3	60.05	32780	54767	1371816915	0	27.615117	34	...	1	large
4	4	4	4	3.19	49632	1126	1371816917	0	104.423175	66	...	1	medium
...	...	...	...	...	...	...	...	...	...	...	...	...	...
555714	555714	555714	555714	43.77	63453	519	1388534347	0	77.032467	56	...	1	small
555715	555715	555715	555715	111.84	77566	28739	1388534349	0	100.023736	22	...	1	large
555716	555716	555716	555716	86.88	99323	3684	1388534355	0	80.887812	40	...	0	medium_large
555717	555717	555717	555717	7.99	83643	129	1388534364	0	53.060882	56	...	1	small
555718	555718	555718	555718	38.13	73034	116001	1388534374	0	72.380990	28	...	1	large

555719 rows × 23 columns

In [47]:

```
#city size
#perform one-hot encoding on 'city size' column
encoder_city_size = pd.DataFrame(encoder.fit_transform(fraud_df[['city size']]).toarray())
encoder_city_size.columns = ['city_size_large','city_size_medium','city_size_mediumlarge','city_size_small']

#merge one-hot encoded columns back
city_size_df = merch_size_df.join(encoder_city_size)
merch_size_df.drop('city size', axis=1, inplace=True)

city_size_df
```

Out[47]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	amt	zip	city_pop	unix_time	is_fraud	distance	age	...	type	
0	0	0	0	2.86	29209	333497	1371816865	0	24.613746	54	...	other	eastcoa
1	1	1	1	29.84	84002	302	1371816873	0	104.834043	32	...	other	midwe
2	2	2	2	41.28	11710	34496	1371816893	0	59.204796	51	...	other	eastcoa
3	3	3	3	60.05	32780	54767	1371816915	0	27.615117	34	...	pos_machine	eastcoa
4	4	4	4	3.19	49632	1126	1371816917	0	104.423175	66	...	other	midea
...	...	...	...	...	...	...	...	...	...	...	...	...	
555714	555714	555714	555714	43.77	63453	519	1388534347	0	77.032467	56	...	other	midea
555715	555715	555715	555715	111.84	77566	28739	1388534349	0	100.023736	22	...	other	midwe
555716	555716	555716	555716	86.88	99323	3684	1388534355	0	80.887812	40	...	other	westcoa
555717	555717	555717	555717	7.99	83643	129	1388534364	0	53.060882	56	...	other	midwe
555718	555718	555718	555718	38.13	73034	116001	1388534374	0	72.380990	28	...	other	midwe

555719 rows × 27 columns

In [48]:

```
#dayzone
#perform one-hot encoding on 'dayzone' column
encoder_dayzone = pd.DataFrame(encoder.fit_transform(fraud_df[['dayzone']]).toarray())
encoder_dayzone.columns = ['afternoon','before_dawn','early_morning','evening','morning','night']

#merge one-hot encoded columns back
dayzone_df = city_size_df.join(encoder_dayzone)
dayzone_df.drop('dayzone', axis=1, inplace=True)

dayzone_df
```

Out[48]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	amt	zip	city_pop	unix_time	is_fraud	distance	age	...	city_size_large	city_
0	0	0	0	2.86	29209	333497	1371816865	0	24.613746	54	...	1.0	
1	1	1	1	29.84	84002	302	1371816873	0	104.834043	32	...	0.0	
2	2	2	2	41.28	11710	34496	1371816893	0	59.204796	51	...	1.0	
3	3	3	3	60.05	32780	54767	1371816915	0	27.615117	34	...	1.0	
4	4	4	4	3.19	49632	1126	1371816917	0	104.423175	66	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
555714	555714	555714	555714	43.77	63453	519	1388534347	0	77.032467	56	...	0.0	
555715	555715	555715	555715	111.84	77566	28739	1388534349	0	100.023736	22	...	1.0	
555716	555716	555716	555716	86.88	99323	3684	1388534355	0	80.887812	40	...	0.0	
555717	555717	555717	555717	7.99	83643	129	1388534364	0	53.060882	56	...	0.0	
555718	555718	555718	555718	38.13	73034	116001	1388534374	0	72.380990	28	...	1.0	

555719 rows × 32 columns

In [49]:

```
#type
```

```
#perform one-hot encoding on 'type' column
encoder_type = pd.DataFrame(encoder.fit_transform(fraud_df[['type']]).toarray())
encoder_type.columns = ['online','other','pos_machine']

#merge one-hot encoded columns back
type_df = dayzone_df.join(encoder_type)
dayzone_df.drop('type', axis=1, inplace=True)

type_df
```

Out[49]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	amt	zip	city_pop	unix_time	is_fraud	distance	age	...	city_size_small	after
0	0	0	0	2.86	29209	333497	1371816865	0	24.613746	54	...	0.0	
1	1	1	1	29.84	84002	302	1371816873	0	104.834043	32	...	1.0	
2	2	2	2	41.28	11710	34496	1371816893	0	59.204796	51	...	0.0	
3	3	3	3	60.05	32780	54767	1371816915	0	27.615117	34	...	0.0	
4	4	4	4	3.19	49632	1126	1371816917	0	104.423175	66	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
555714	555714	555714	555714	43.77	63453	519	1388534347	0	77.032467	56	...	1.0	
555715	555715	555715	555715	111.84	77566	28739	1388534349	0	100.023736	22	...	0.0	
555716	555716	555716	555716	86.88	99323	3684	1388534355	0	80.887812	40	...	0.0	
555717	555717	555717	555717	7.99	83643	129	1388534364	0	53.060882	56	...	1.0	
555718	555718	555718	555718	38.13	73034	116001	1388534374	0	72.380990	28	...	0.0	

555719 rows × 35 columns

In [50]:

```
#region
#perform one-hot encoding on 'region' column
encoder_region = pd.DataFrame(encoder.fit_transform(fraud_df[['region']]).toarray())
encoder_region.columns = ['eastcoast_north','eastcoast_south','mideast_north','mideast_south','midwest_north','

#merge one-hot encoded columns back
region_df = type_df.join(encoder_region)
dayzone_df.drop('region', axis=1, inplace=True)

region_df
```

Out[50]:

	Unnamed: 0	Unnamed: 0.1	Unnamed: 0.1.1	amt	zip	city_pop	unix_time	is_fraud	distance	age	...	other	pos_machine
0	0	0	0	2.86	29209	333497	1371816865	0	24.613746	54	...	1.0	0.0
1	1	1	1	29.84	84002	302	1371816873	0	104.834043	32	...	1.0	0.0
2	2	2	2	41.28	11710	34496	1371816893	0	59.204796	51	...	1.0	0.0
3	3	3	3	60.05	32780	54767	1371816915	0	27.615117	34	...	0.0	1.0
4	4	4	4	3.19	49632	1126	1371816917	0	104.423175	66	...	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
555714	555714	555714	555714	43.77	63453	519	1388534347	0	77.032467	56	...	1.0	0.0
555715	555715	555715	555715	111.84	77566	28739	1388534349	0	100.023736	22	...	1.0	0.0
555716	555716	555716	555716	86.88	99323	3684	1388534355	0	80.887812	40	...	1.0	0.0
555717	555717	555717	555717	7.99	83643	129	1388534364	0	53.060882	56	...	1.0	0.0
555718	555718	555718	555718	38.13	73034	116001	1388534374	0	72.380990	28	...	1.0	0.0

555719 rows × 43 columns

In [52]:

```
final_encoded_df = region_df
final_encoded_df.columns
final_encoded_df.drop(['city size','Unnamed: 0','Unnamed: 0.1','Unnamed: 0.1.1'], axis=1, inplace=True)
```

In [53]:

```
final_encoded_df.columns
```

Out[53]:

```
Index(['amt', 'zip', 'city_pop', 'unix_time', 'is_fraud', 'distance', 'age',
       'month', 'weekend', 'gender', 'merchant num', 'num_cc_num', 'type',
       'region', 'merch_size_large', 'merch_size_medium',
       'merch_size_mediumlarge', 'merch_size_small', 'city_size_large',
       'city_size_medium', 'city_size_mediumlarge', 'city_size_small',
       'afternoon', 'before_dawn', 'early_morning', 'evening', 'morning',
       'night', 'online', 'other', 'pos_machine', 'eastcoast_north',
       'eastcoast_south', 'mideast_north', 'mideast_south', 'midwest_north',
       'midwest_south', 'westcoast_north', 'westcoast_south'],
      dtype='object')
```

In [55]:

```
final_encoded_df.to_csv('fraudtest_final_encoded.csv')
```

In [ ]: