```
import pandas as pd
```

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

```
df.shape
```

```
(120490, 23)
```

```
df.head(3)
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	l
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	...	36.07
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	48.88
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	42.18

3 rows × 23 columns

```
df.columns
```

```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

```
# 1. Dropping the unwanted columns
df.drop(columns = ['Unnamed: 0','trans_date_trans_time','cc_num','merchant','first','last', 'street','city','zip','lat','long','dob','t
```

```
df.head()
```

	category	amt	gender	state	city_pop	job	is_fraud
0	misc_net	4.97	F	NC	3495	Psychologist, counselling	0.0
1	grocery_pos	107.23	F	WA	149	Special educational needs teacher	0.0
2	entertainment	220.11	M	ID	4154	Nature conservation officer	0.0
3	gas_transport	45.00	M	MT	1939	Patent attorney	0.0

```
# 2. Checking Null values
df.isna().sum()
```

```
category    0
amt         0
gender      0
state       0
city_pop    0
job         0
is_fraud    1
dtype: int64
```

```
df.dropna(inplace = True)
```

```
df.isna().sum()
```

```
category    0
amt         0
gender      0
state       0
city_pop    0
job         0
```

```
is_fraud    0
dtype: int64
```

```
# 3. Converting categorical values to numerical
df['category'].unique()
```

```
array(['misc_net', 'grocery_pos', 'entertainment', 'gas_transport',
       'misc_pos', 'grocery_net', 'shopping_net', 'shopping_pos',
       'food_dining', 'personal_care', 'health_fitness', 'travel',
       'kids_pets', 'home'], dtype=object)
```

```
category_df=pd.get_dummies(df['category']).astype(int)
```

```
category_df.head()
```

	entertainment	food_dining	gas_transport	grocery_net	grocery_pos	health_fitnes
0	0	0	0	0	0	
1	0	0	0	0	1	
2	1	0	0	0	0	
3	0	0	1	0	0	
4	0	0	0	0	0	

```
df.drop(columns= ['category'], inplace = True)
```

Double-click (or enter) to edit

```
df.head()
```

	amt	gender	state	city_pop	job	is_fraud
0	4.97	F	NC	3495	Psychologist, counselling	0.0
1	107.23	F	WA	149	Special educational needs teacher	0.0
2	220.11	M	ID	4154	Nature conservation officer	0.0
3	45.00	M	MT	1939	Patent attorney	0.0
4	41.96	M	VA	99	Dance movement psychotherapist	0.0

```
df.reset_index(drop=True, inplace=True)
```

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

```
df.head()
```

	amt	gender	state	city_pop	job	is_fraud	entertainment	food_dinir
0	4.97	F	NC	3495	Psychologist, counselling	0.0	0	
1	107.23	F	WA	149	Special educational needs teacher	0.0	0	
2	220.11	M	ID	4154	Nature conservation officer	0.0	1	
3	45.00	M	MT	1939	Patent attorney	0.0	0	
4	41.96	M	VA	99	Dance movement psychotherapist	0.0	0	

```
df['gender'].unique()
```

```
array(['F', 'M'], dtype=object)
```

```
df['gender'].replace({'F' : 1, 'M' : 0}, inplace = True)
```

```
df['gender'].unique()
```

```
array([1, 0])
```

```
df.head()
```

	amt	gender	state	city_pop	job	is_fraud	entertainment	food_dini
0	4.97	1	NC	3495	Psychologist, counselling	0.0	0	
1	107.23	1	WA	149	Special educational needs teacher	0.0	0	
2	220.11	0	ID	4154	Nature conservation officer	0.0	1	
3	45.00	0	MT	1939	Patent attorney	0.0	0	
4	41.96	0	VA	99	Dance movement psychotherapist	0.0	0	

```
df['state'].unique()
```

```
array(['NC', 'WA', 'ID', 'MT', 'VA', 'PA', 'KS', 'TN', 'IA', 'WV', 'FL',
       'CA', 'NM', 'NJ', 'OK', 'IN', 'MA', 'TX', 'WI', 'MI', 'WY', 'HI',
       'NE', 'OR', 'LA', 'DC', 'KY', 'NY', 'MS', 'UT', 'AL', 'AR', 'MD',
       'GA', 'ME', 'AZ', 'MN', 'OH', 'CO', 'VT', 'MO', 'SC', 'NV', 'IL',
       'NH', 'SD', 'AK', 'ND', 'CT', 'RI'], dtype=object)
```

```
state_df=pd.get_dummies(df['state']).astype(int)
```

```
state_df.head()
```

	AK	AL	AR	AZ	CA	CO	CT	DC	FL	GA	...	SD	TN	TX	UT	VA	VT	WA	WI	WV	WY
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0

5 rows × 50 columns

```
df.reset_index(drop=True, inplace=True)
```

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

```
df.drop(columns = ['state'], inplace = True)
```

```
df.head()
```

	amt	gender	city_pop	job	is_fraud	entertainment	food_dining	gas_
0	4.97	1	3495	Psychologist, counselling	0.0	0	0	
1	107.23	1	149	Special educational needs teacher	0.0	0	0	
2	220.11	0	4154	Nature conservation officer	0.0	1	0	
3	45.00	0	1939	Patent attorney	0.0	0	0	
4	41.96	0	99	Dance movement psychotherapist	0.0	0	0	

5 rows × 69 columns

```
df['job'].unique().shape
```

```
(479,)
```

```
job_df=pd.get_dummies(df['job']).astype(int)
```

```
job_df.head()
```

	Academic librarian	Accountant, chartered certified	Accountant, chartered public finance	Accounting technician	Acupuncturist	Administrator	Adm
0	0	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	

5 rows × 479 columns

```
df.drop(columns = ['job'], inplace = True)
```

```
df.reset_index(drop=True, inplace = True)
```

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

```
df.head()
```

	amt	gender	city_pop	is_fraud	entertainment	food_dining	gas_transport	gro
0	4.97	1	3495	0.0	0	0	0	
1	107.23	1	149	0.0	0	0	0	
2	220.11	0	4154	0.0	1	0	0	
3	45.00	0	1939	0.0	0	0	1	
4	41.96	0	99	0.0	0	0	0	

5 rows × 547 columns

```
df.shape
```

```
(120489, 547)
```

```
# 4. Dividing the dataset
X = df.drop(columns = ['is_fraud'])
y = df['is_fraud']
```

```
X.head()
```

	amt	gender	city_pop	entertainment	food_dining	gas_transport	grocery_net
0	4.97	1	3495	0	0	0	0
1	107.23	1	149	0	0	0	0
2	220.11	0	4154	1	0	0	0
3	45.00	0	1939	0	0	1	0
4	41.96	0	99	0	0	0	0

5 rows × 546 columns

```
y.value_counts()
```

```
is_fraud
0.0    119341
1.0      1148
Name: count, dtype: int64
```

```
# 5. Handling the imbalanced dataset using SMOTE technique
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='minority')
X,y =smote.fit_resample(X,y)
```

```
y.value_counts()
```

```
is_fraud
0.0    119341
1.0    119341
Name: count, dtype: int64
```

```
X.shape
```

```
(238682, 546)
```

```
# 6. Data Normalization
X.describe()
```

	amt	gender	city_pop	entertainment	food_dining	gas_t
count	238682.000000	238682.000000	2.386820e+05	238682.000000	238682.000000	2386
mean	290.462502	0.516872	9.989651e+04	0.038545	0.037925	
std	365.503926	0.499716	3.174680e+05	0.192508	0.191015	
min	1.000000	0.000000	2.300000e+01	0.000000	0.000000	
25%	20.962240	0.000000	7.180000e+02	0.000000	0.000000	
50%	88.520000	1.000000	2.607000e+03	0.000000	0.000000	
75%	397.956913	1.000000	2.702000e+04	0.000000	0.000000	
max	12788.070000	1.000000	2.906700e+06	1.000000	1.000000	

8 rows × 546 columns

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X[:] = scaler.fit_transform(X)
```

```
X.describe()
```

	amt	gender	city_pop	entertainment	food_dining	gas_
count	238682.000000	238682.000000	238682.000000	238682.000000	238682.000000	2386
mean	0.022637	0.516872	0.034360	0.038545	0.037925	
std	0.028584	0.499716	0.109220	0.192508	0.191015	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.001561	0.000000	0.000239	0.000000	0.000000	
50%	0.006844	1.000000	0.000889	0.000000	0.000000	
75%	0.031044	1.000000	0.009288	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 546 columns

```
X.head()
```

	amt	gender	city_pop	entertainment	food_dining	gas_transport	grocery_net
0	0.000310	1	0.001194	0	0	0	0
1	0.008308	1	0.000043	0	0	0	0
2	0.017135	0	0.001421	1	0	0	0
3	0.003441	0	0.000659	0	0	1	0
4	0.003203	0	0.000026	0	0	0	0

5 rows × 546 columns

```
# 7. Train - Test Split
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, random_state=42, stratify=y)
```

```
X_train.shape, X_test.shape
```

```
((190945, 546), (47737, 546))
```

```
y_train.value_counts()
```

```
is_fraud
1.0    95473
0.0    95472
Name: count, dtype: int64
```

```
y_test.value_counts()
```

```
is_fraud
0.0    23869
1.0    23868
Name: count, dtype: int64
```

```
# 8. Building the Artificial Neural Network
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
    # input layer + hidden layer 1
    keras.layers.Dense(300, input_shape =(546,), activation = 'relu'),
    # hidden layer 2
    keras.layers.Dense(150, activation = 'relu'),
    # output layer
    keras.layers.Dense(1, activation = 'sigmoid')
])

model.compile(optimizer = 'adam',
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
model.fit(X_train, y_train, epochs = 10, batch_size=200)
```

```
Epoch 1/10
955/955 [==============================] - 6s 3ms/step - loss: 0.0633 - accuracy: 0.9773
Epoch 2/10
955/955 [==============================] - 4s 4ms/step - loss: 0.0267 - accuracy: 0.9917
Epoch 3/10
955/955 [==============================] - 4s 4ms/step - loss: 0.0229 - accuracy: 0.9926
Epoch 4/10
955/955 [==============================] - 3s 3ms/step - loss: 0.0192 - accuracy: 0.9939
Epoch 5/10
955/955 [==============================] - 3s 3ms/step - loss: 0.0168 - accuracy: 0.9946
Epoch 6/10
955/955 [==============================] - 4s 4ms/step - loss: 0.0145 - accuracy: 0.9952
Epoch 7/10
955/955 [==============================] - 4s 4ms/step - loss: 0.0137 - accuracy: 0.9955
Epoch 8/10
955/955 [==============================] - 3s 3ms/step - loss: 0.0124 - accuracy: 0.9959
Epoch 9/10
955/955 [==============================] - 3s 3ms/step - loss: 0.0121 - accuracy: 0.9960
Epoch 10/10
955/955 [==============================] - 4s 4ms/step - loss: 0.0117 - accuracy: 0.9961
<keras.src.callbacks.History at 0x7f0bd7fe50c0>
```

```
# 9. Model Evaluation
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Model Accuracy : {accuracy * 100}')
```

```
1492/1492 [==============================] - 4s 2ms/step - loss: 0.0154 - accuracy: 0.9957
Model Accuracy : 99.57475066184998
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 300)               164100

 dense_1 (Dense)             (None, 150)               45150

 dense_2 (Dense)             (None, 1)                 151

=================================================================
Total params: 209401 (817.97 KB)
Trainable params: 209401 (817.97 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# 10. To make predictions
pred = model.predict(X_test)
```

```
1492/1492 [==============================] - 4s 2ms/step
```

```
pred[:5]
```

```
array([[1.2696146e-13],
       [9.9999988e-01],
       [1.0000000e+00],
       [9.9999976e-01],
       [1.2975411e-08]], dtype=float32)
```

```
binary_pred = ((pred > 0.5)).astype(int)
```

```
binary_pred[:5]
```

```
array([[0],
       [1],
       [1],
       [1],
       [0]])
```

```
y_test[:5]
```

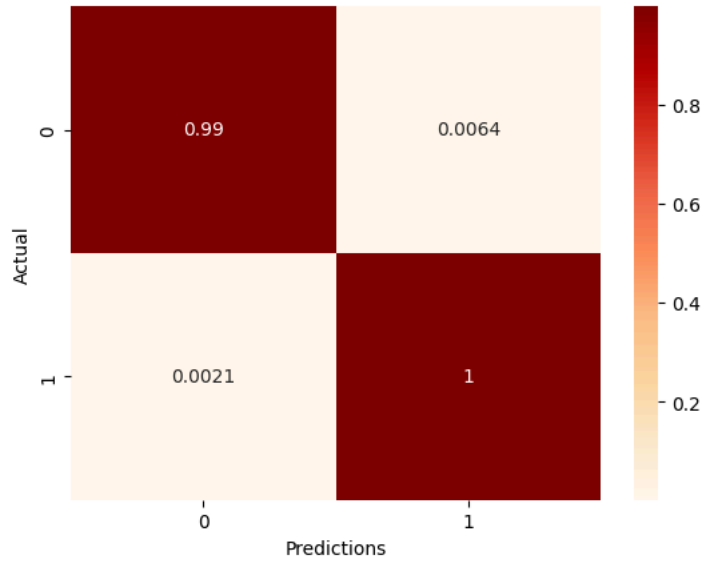
```
36487     0.0
197951    1.0
123820    1.0
177083    1.0
45819     0.0
Name: is_fraud, dtype: float64
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, binary_pred))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.99      1.00     23869
         1.0       0.99      1.00      1.00     23868

    accuracy                           1.00     47737
   macro avg       1.00      1.00      1.00     47737
weighted avg       1.00      1.00      1.00     47737
```

```
import seaborn as sns
import matplotlib.pyplot as plt

cf = confusion_matrix(y_test, binary_pred, normalize = 'true')
sns.heatmap(cf, annot = True, cmap = 'OrRd')
plt.xlabel('Predictions')
plt.ylabel('Actual')
```

```
Text(50.722222222222214, 0.5, 'Actual')
```



Start coding or generate with AI.