DSC680_Project1Final_LincolnBrown

December 22, 2024

0.1 Import the libraries

[190]:
```
import category_encoders as ce
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers.legacy import Adam
import time
```

0.2 Import the Dataset

[2]:
```
cc_f = "creditcard_dataset/credit_card_transactions-ibm_v2.csv"
user_f = "creditcard_dataset/sd254_users.csv"
cc_df = pd.read_csv(cc_f)
user_df = pd.read_csv(user_f)
```

[3]:
```
cc_df
```

[3]:
```
          User  Card  Year  Month  Day   Time   Amount           Use Chip  \
0            0     0  2002      9    1  06:21  $134.09  Swipe Transaction
1            0     0  2002      9    1  06:42   $38.48  Swipe Transaction
2            0     0  2002      9    2  06:22  $120.34  Swipe Transaction
3            0     0  2002      9    2  17:45  $128.95  Swipe Transaction
```

```
4            0     0  2002      9    3  06:23  $104.71  Swipe Transaction
…           …    …    …     …   …     …      …                  …
24386895  1999     1  2020      2   27  22:23  $-54.00   Chip Transaction
24386896  1999     1  2020      2   27  22:24   $54.00   Chip Transaction
24386897  1999     1  2020      2   28  07:43   $59.15   Chip Transaction
24386898  1999     1  2020      2   28  20:10   $43.12   Chip Transaction
24386899  1999     1  2020      2   28  23:10   $45.13   Chip Transaction

                Merchant Name  Merchant City Merchant State      Zip   MCC  \
0         3527213246127876953       La Verne             CA  91750.0  5300
1        -727612092139916043  Monterey Park             CA  91754.0  5411
2        -727612092139916043  Monterey Park             CA  91754.0  5411
3         3414527459579106770  Monterey Park             CA  91754.0  5651
4         5817218446178736267       La Verne             CA  91750.0  5912
…                         …              …              …        …     …
24386895 -5162038175624867091      Merrimack             NH   3054.0  5541
24386896 -5162038175624867091      Merrimack             NH   3054.0  5541
24386897  2500998799892805156      Merrimack             NH   3054.0  4121
24386898  2500998799892805156      Merrimack             NH   3054.0  4121
24386899  4751695835751691036      Merrimack             NH   3054.0  5814

         Errors? Is Fraud?
0            NaN        No
1            NaN        No
2            NaN        No
3            NaN        No
4            NaN        No
…             …        …
24386895     NaN        No
24386896     NaN        No
24386897     NaN        No
24386898     NaN        No
24386899     NaN        No

[24386900 rows x 15 columns]
```

[4]:
```
user_df
```

[4]:
```
                 Person  Current Age  Retirement Age  Birth Year  Birth Month  \
0        Hazel Robinson           53              66        1966           11
1            Sasha Sadr           53              68        1966           12
2            Saanvi Lee           81              67        1938           11
3         Everlee Clark           63              63        1957            1
4         Kyle Peterson           43              70        1976            9
…                   …          …              …         …           …
1995       Jose Faraday           32              70        1987            7
1996  Ximena Richardson           62              65        1957           11
```

```
1997    Annika Russell          47            67        1973            1
1998       Juelz Roman          66            60        1954            2
1999      Kenia Harris          21            60        1998           11

      Gender                   Address  Apartment          City State  \
0     Female             462 Rose Lane        NaN      La Verne    CA
1     Female    3606 Federal Boulevard        NaN   Little Neck    NY
2     Female           766 Third Drive        NaN   West Covina    CA
3     Female          3 Madison Street        NaN      New York    NY
4       Male  9620 Valley Stream Drive        NaN  San Francisco   CA
…        …                         …          …             …     …
1995    Male        6577 Lexington Lane        9.0      Freeport    NY
1996  Female               2 Elm Drive      955.0  Independence    KY
1997  Female        276 Fifth Boulevard        NaN     Elizabeth    NJ
1998    Male       259 Valley Boulevard        NaN     Camp Hill    PA
1999  Female     472 Ocean View Street        NaN     Merrimack    NH

      Zipcode  Latitude  Longitude Per Capita Income - Zipcode  \
0       91750     34.15    -117.76                      $29278
1       11363     40.76     -73.74                      $37891
2       91792     34.02    -117.89                      $22681
3       10069     40.71     -73.99                     $163145
4       94117     37.76    -122.44                      $53797
…         …         …          …                           …
1995    11520     40.65     -73.58                      $23550
1996    41051     38.95     -84.54                      $24218
1997     7201     40.66     -74.19                      $15175
1998    17011     40.24     -76.92                      $25336
1999     3054     42.86     -71.48                      $32325

     Yearly Income - Person Total Debt  FICO Score  Num Credit Cards
0                    $59696    $127613         787                 5
1                    $77254    $191349         701                 5
2                    $33483       $196         698                 5
3                   $249925    $202328         722                 4
4                   $109687    $183855         675                 1
…                        …          …          …                 …
1995                 $48010     $87837         703                 3
1996                 $49378    $104480         740                 4
1997                 $30942     $71066         779                 3
1998                 $54654     $27241         618                 1
1999                 $65909    $181261         673                 2

[2000 rows x 18 columns]
```

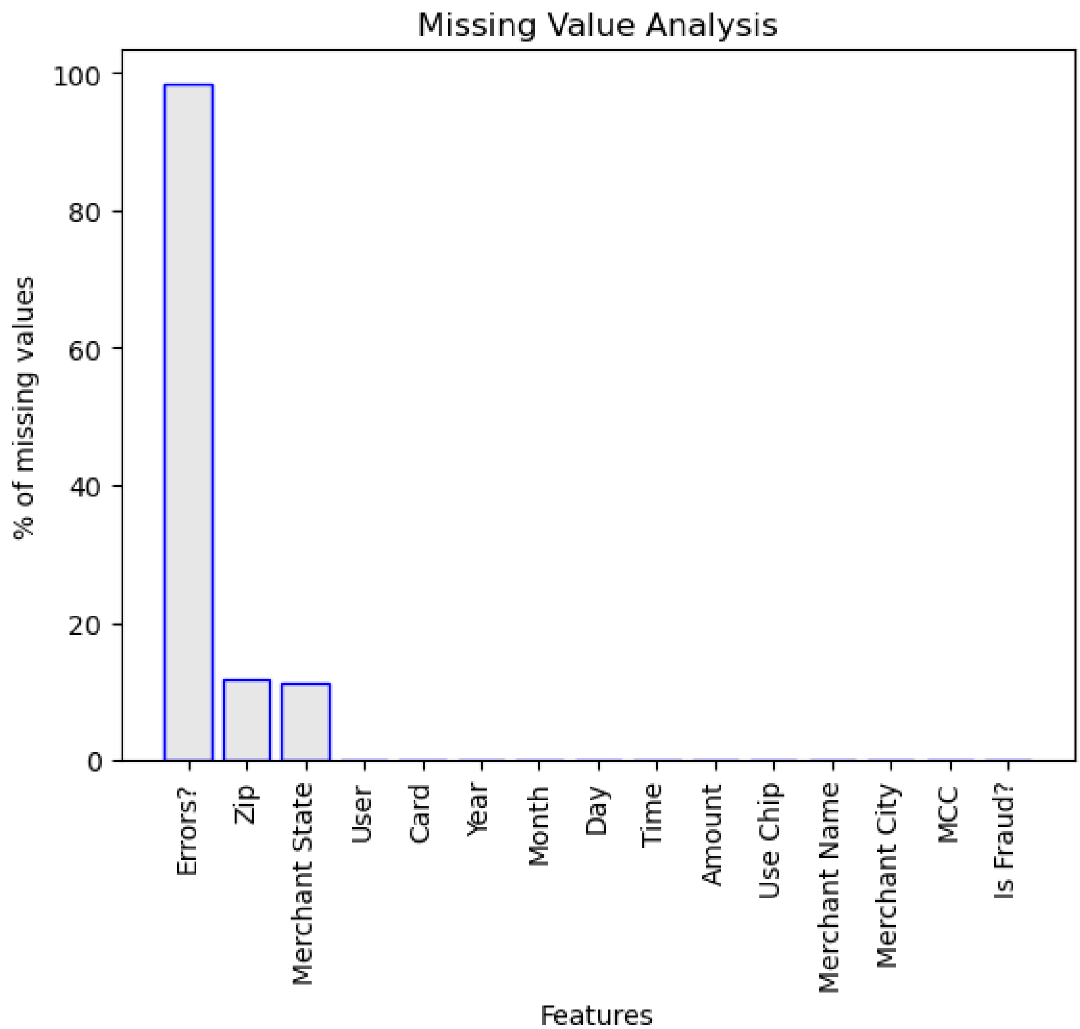
0.3 Check/Clean for missing Values

[5]:
```
cc_df.loc[cc_df['Merchant State'].isna()]
```

[5]:
```
          User  Card  Year  Month  Day   Time   Amount            Use Chip  \
11           0     0  2002      9    5  20:41   $53.91  Online Transaction
24           0     0  2002      9    9  20:02  $144.90  Online Transaction
85           0     0  2002      9   30  06:21  $127.32  Online Transaction
99           0     0  2002     10    6  06:14  $139.39  Online Transaction
106          0     0  2002     10    9  08:16   $53.09  Online Transaction
…          …   …    …    …  …    …      …                  …
24386877  1999     1  2020      2   24  20:04   $55.79  Online Transaction
24386879  1999     1  2020      2   25  07:06   $43.08  Online Transaction
24386880  1999     1  2020      2   25  07:34   $43.76  Online Transaction
24386884  1999     1  2020      2   26  07:43   $45.18  Online Transaction
24386889  1999     1  2020      2   27  07:47   $47.18  Online Transaction

                Merchant Name Merchant City Merchant State  Zip   MCC Errors?  \
11       -9092677072201095172        ONLINE            NaN  NaN  4900     NaN
24       -8338381919281017248        ONLINE            NaN  NaN  4899     NaN
85       -7421093378627544099        ONLINE            NaN  NaN  5311     NaN
99       -7421093378627544099        ONLINE            NaN  NaN  5311     NaN
106      -4956618006720593695        ONLINE            NaN  NaN  5193     NaN
…                       …            …             …   …   …      …
24386877 -6160036380778658394        ONLINE            NaN  NaN  4121     NaN
24386879 -6160036380778658394        ONLINE            NaN  NaN  4121     NaN
24386880 -6160036380778658394        ONLINE            NaN  NaN  4121     NaN
24386884 -6160036380778658394        ONLINE            NaN  NaN  4121     NaN
24386889 -5841929396161652653        ONLINE            NaN  NaN  4121     NaN

         Is Fraud?
11              No
24              No
85              No
99              No
106             No
…             …
24386877        No
24386879        No
24386880        No
24386884        No
24386889        No

[2720821 rows x 15 columns]
```

It looks like all of the missing values for Merchant State are due to Online purchases, we will resolve this by labeling these missing values as ONLINE, similar to what is already found in Merchant City

[6]:
```
percent_missing=(cc_df.isnull().sum()*100/cc_df.shape[0]).
 ↪sort_values(ascending=True)
plt.title("Missing Value Analysis")
plt.xlabel("Features")
plt.ylabel("% of missing values")
plt.bar(percent_missing.sort_values(ascending=False).index,percent_missing.
 ↪sort_values(ascending=False),color=(0.1, 0.1, 0.1, 0.1),edgecolor='blue')
plt.xticks(rotation=90)
plt.show()
```



[7]:
```
cc_df
```

[7]:
```
            User  Card  Year  Month  Day   Time   Amount           Use Chip  \
0              0     0  2002      9    1  06:21  $134.09  Swipe Transaction
```

```
1            0     0  2002      9    1  06:42   $38.48  Swipe Transaction
2            0     0  2002      9    2  06:22  $120.34  Swipe Transaction
3            0     0  2002      9    2  17:45  $128.95  Swipe Transaction
4            0     0  2002      9    3  06:23  $104.71  Swipe Transaction
…          …   …    …     …  …     …     …              …
24386895  1999     1  2020      2   27  22:23  $-54.00   Chip Transaction
24386896  1999     1  2020      2   27  22:24   $54.00   Chip Transaction
24386897  1999     1  2020      2   28  07:43   $59.15   Chip Transaction
24386898  1999     1  2020      2   28  20:10   $43.12   Chip Transaction
24386899  1999     1  2020      2   28  23:10   $45.13   Chip Transaction

                Merchant Name  Merchant City Merchant State      Zip   MCC  \
0         3527213246127876953      La Verne             CA  91750.0  5300
1         -727612092139916043  Monterey Park            CA  91754.0  5411
2         -727612092139916043  Monterey Park            CA  91754.0  5411
3         3414527459579106770  Monterey Park            CA  91754.0  5651
4         5817218446178736267      La Verne             CA  91750.0  5912
…                           …            …           …       …     …
24386895 -5162038175624867091     Merrimack             NH   3054.0  5541
24386896 -5162038175624867091     Merrimack             NH   3054.0  5541
24386897  2500998799892805156     Merrimack             NH   3054.0  4121
24386898  2500998799892805156     Merrimack             NH   3054.0  4121
24386899  4751695835751691036     Merrimack             NH   3054.0  5814

         Errors? Is Fraud?
0            NaN        No
1            NaN        No
2            NaN        No
3            NaN        No
4            NaN        No
…          …       …
24386895     NaN        No
24386896     NaN        No
24386897     NaN        No
24386898     NaN        No
24386899     NaN        No

[24386900 rows x 15 columns]
```

[8]:
```python
cc_df['Amount'] = cc_df['Amount'].str.replace("\\$", "", regex=True)
cc_df['Amount'] = pd.to_numeric(cc_df['Amount'], errors='coerce')
print(cc_df['Amount'].head(10))
```

```
0    134.09
1     38.48
2    120.34
3    128.95
4    104.71
```

```
5     86.19
6     93.84
7    123.50
8     61.72
9     57.10
Name: Amount, dtype: float64
```

[9]:
```
cc_df['Errors?'].unique()
```

[9]:
```
array([nan, 'Technical Glitch', 'Insufficient Balance', 'Bad PIN',
       'Bad PIN,Insufficient Balance', 'Bad Expiration',
       'Bad PIN,Technical Glitch', 'Bad Card Number', 'Bad CVV',
       'Bad Zipcode', 'Insufficient Balance,Technical Glitch',
       'Bad Card Number,Insufficient Balance', 'Bad Card Number,Bad CVV',
       'Bad CVV,Insufficient Balance', 'Bad Card Number,Bad Expiration',
       'Bad Expiration,Bad CVV', 'Bad Expiration,Insufficient Balance',
       'Bad Expiration,Technical Glitch',
       'Bad Card Number,Bad Expiration,Technical Glitch',
       'Bad CVV,Technical Glitch', 'Bad Card Number,Technical Glitch',
       'Bad Zipcode,Insufficient Balance', 'Bad Zipcode,Technical Glitch',
       'Bad Card Number,Bad Expiration,Insufficient Balance'],
      dtype=object)
```

0.4 Clean the dataset

[10]:
```
cc_df['Zip'] = cc_df['Zip'].astype(str)
cc_df.loc[cc_df['Merchant City'] == 'ONLINE', ['Merchant State', 'Zip']] =␣
 ↪'ONLINE'
```

[11]:
```
cc_df.loc[cc_df['Zip'].isna(), 'Zip'] = 'Foreign'
```

[12]:
```
cc_df.isna().any()
```

[12]:
```
User              False
Card              False
Year              False
Month             False
Day               False
Time              False
Amount            False
Use Chip          False
Merchant Name     False
Merchant City     False
Merchant State    False
Zip               False
MCC               False
Errors?            True
Is Fraud?         False
```

```
dtype: bool
```

[13]:
```python
# Get a unique list of "States"
states = cc_df['Merchant State'].unique()
```

[14]:
```python
# Check the US States abbreviations
us_states = [state for state in states if len(str(state)) == 2]
```

[15]:
```python
us_states
```

[15]:
```
['CA',
 'NE',
 'IL',
 'MO',
 'IA',
 'TX',
 'NJ',
 'NV',
 'NY',
 'AZ',
 'UT',
 'FL',
 'MI',
 'WA',
 'OH',
 'NM',
 'SC',
 'AK',
 'PA',
 'VA',
 'HI',
 'CT',
 'MA',
 'MN',
 'CO',
 'GA',
 'AR',
 'OR',
 'WI',
 'NC',
 'WV',
 'ME',
 'NH',
 'VT',
 'MD',
 'AL',
 'KY',
```

```
 'TN',
 'MS',
 'KS',
 'ND',
 'DC',
 'MT',
 'OK',
 'WY',
 'ID',
 'RI',
 'IN',
 'LA',
 'DE',
 'SD',
 'AA']
```

[16]:
```
len(us_states)
```

[16]:
```
52
```

It looks like there are 52 state abbreviations, including DC for District of Columbia and AA for Armed Forces of America

Let's map the days of the month to day of the week and separate the time column into hour and minute

[17]:
```
# Get the day of the month

cc_df['Date'] =  pd.to_datetime(cc_df[['Year', 'Month', 'Day']])

# Extract day of the week and map it to its name
days = {0:'Mon', 1:'Tue', 2:'Wed', 3:'Thu', 4:'Fri', 5:'Sat', 6:'Sun'}
cc_df['Day of Week'] = cc_df['Date'].dt.dayofweek.map(days)
```

[18]:
```
# Get the hour and minute from Time
cc_df['Hour'] = pd.to_numeric(cc_df['Time'].str [0:2])
cc_df['Minute'] = pd.to_numeric(cc_df['Time'].str [3:5])
```

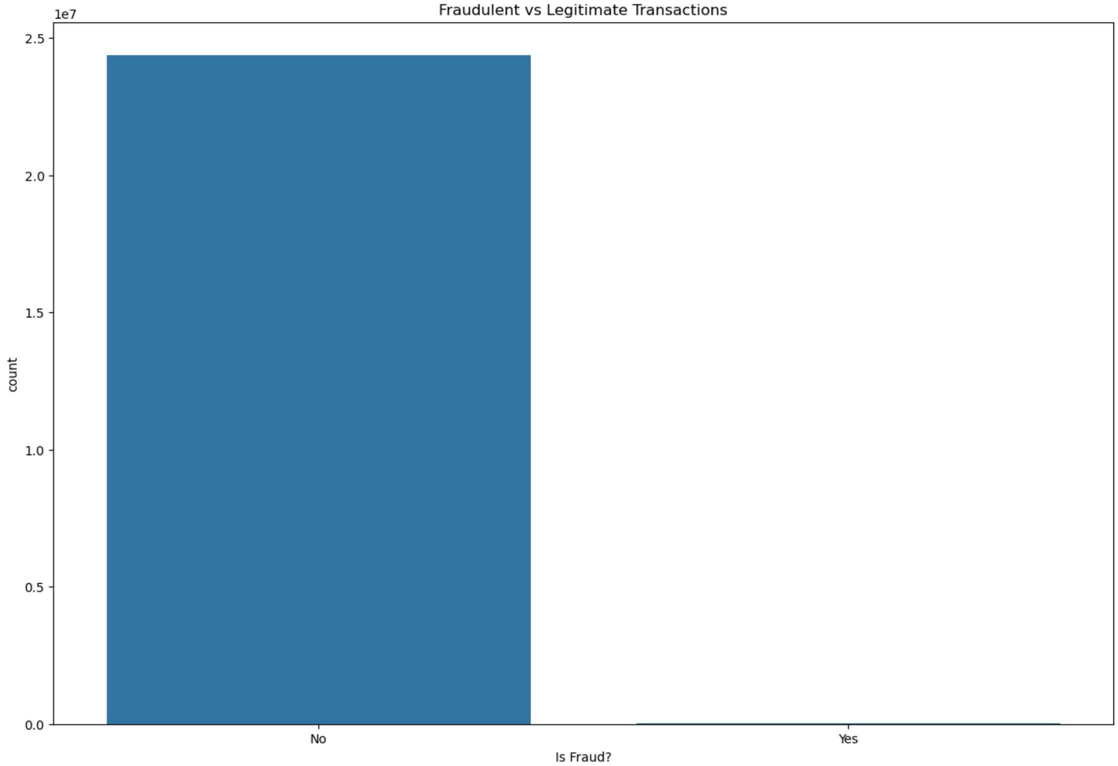
[19]:
```
cc_df
```

[19]:
```
          User  Card  Year  Month  Day   Time  Amount           Use Chip  \
0            0     0  2002      9    1  06:21  134.09  Swipe Transaction
1            0     0  2002      9    1  06:42   38.48  Swipe Transaction
2            0     0  2002      9    2  06:22  120.34  Swipe Transaction
3            0     0  2002      9    2  17:45  128.95  Swipe Transaction
4            0     0  2002      9    3  06:23  104.71  Swipe Transaction
…          …     …     …      …  …      …       …                    …
24386895  1999     1  2020      2   27  22:23  -54.00   Chip Transaction
```

```
24386896  1999     1  2020       2   27  22:24   54.00   Chip Transaction
24386897  1999     1  2020       2   28  07:43   59.15   Chip Transaction
24386898  1999     1  2020       2   28  20:10   43.12   Chip Transaction
24386899  1999     1  2020       2   28  23:10   45.13   Chip Transaction

                Merchant Name  Merchant City Merchant State      Zip   MCC  \
0          3527213246127876953      La Verne             CA  91750.0  5300
1         -727612092139916043  Monterey Park             CA  91754.0  5411
2         -727612092139916043  Monterey Park             CA  91754.0  5411
3          3414527459579106770  Monterey Park             CA  91754.0  5651
4          5817218446178736267      La Verne             CA  91750.0  5912
…                          …            …             …        …    …
24386895 -5162038175624867091      Merrimack             NH   3054.0  5541
24386896 -5162038175624867091      Merrimack             NH   3054.0  5541
24386897  2500998799892805156      Merrimack             NH   3054.0  4121
24386898  2500998799892805156      Merrimack             NH   3054.0  4121
24386899  4751695835751691036      Merrimack             NH   3054.0  5814

         Errors? Is Fraud?       Date Day of Week  Hour  Minute
0            NaN        No 2002-09-01         Sun     6      21
1            NaN        No 2002-09-01         Sun     6      42
2            NaN        No 2002-09-02         Mon     6      22
3            NaN        No 2002-09-02         Mon    17      45
4            NaN        No 2002-09-03         Tue     6      23
…          …       …          …       …    …      …
24386895     NaN        No 2020-02-27         Thu    22      23
24386896     NaN        No 2020-02-27         Thu    22      24
24386897     NaN        No 2020-02-28         Fri     7      43
24386898     NaN        No 2020-02-28         Fri    20      10
24386899     NaN        No 2020-02-28         Fri    23      10

[24386900 rows x 19 columns]
```

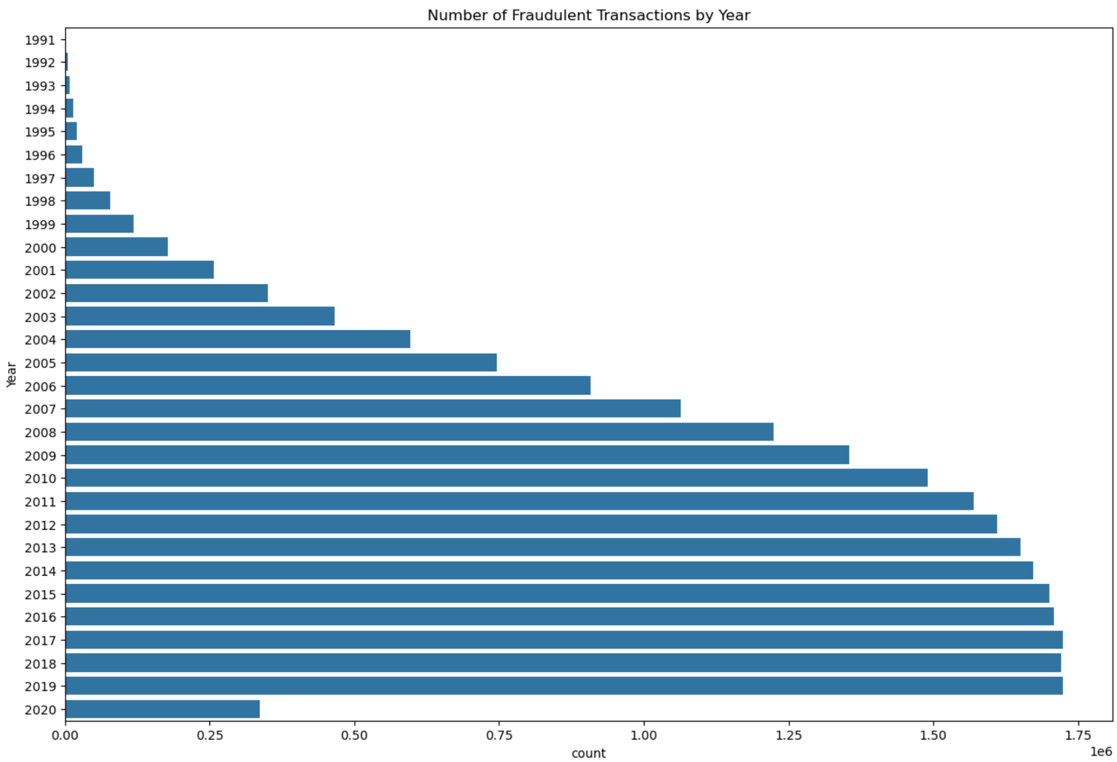
[20]:

```python
# Fraudulent vs legitimate transactions
plt.figure(figsize=(15, 10))
sns.countplot(data=cc_df, x='Is Fraud?')
plt.title('Fraudulent vs Legitimate Transactions')
plt.show()
```



As we can see, this dataset is heavily imbalanced, with the number of legitimate transactions far exceeding fraudulent.

[21]:
```
plt.figure(figsize=(15, 10))
sns.countplot(data=cc_df, y='Year')
plt.title('Number of Fraudulent Transactions by Year')
plt.show()
```



It looks like the number of fraudulent transactions increases year over year. I want to confirm that we do not have full yearly data for 2020.

[22]:
```
cc_df['Date'].max()
```

[22]:
```
Timestamp('2020-02-28 00:00:00')
```

The latest date we have for 2020 is February 28th, indicating that we do not have a full year's worth of data for 2020.

0.5 Fraudulent Transactions Analysis

Let's take a look at the amounts of the fraudulent transactions.

[23]:
```
cc_df['Is Fraud?'] = cc_df['Is Fraud?'].map({'No': 0, 'Yes': 1})
cc_df
```

[23]:
```
          User  Card  Year  Month  Day   Time  Amount           Use Chip  \
0            0     0  2002      9    1  06:21  134.09  Swipe Transaction
1            0     0  2002      9    1  06:42   38.48  Swipe Transaction
2            0     0  2002      9    2  06:22  120.34  Swipe Transaction
3            0     0  2002      9    2  17:45  128.95  Swipe Transaction
4            0     0  2002      9    3  06:23  104.71  Swipe Transaction
…            …     …     …      …    …      …       …                  …
```

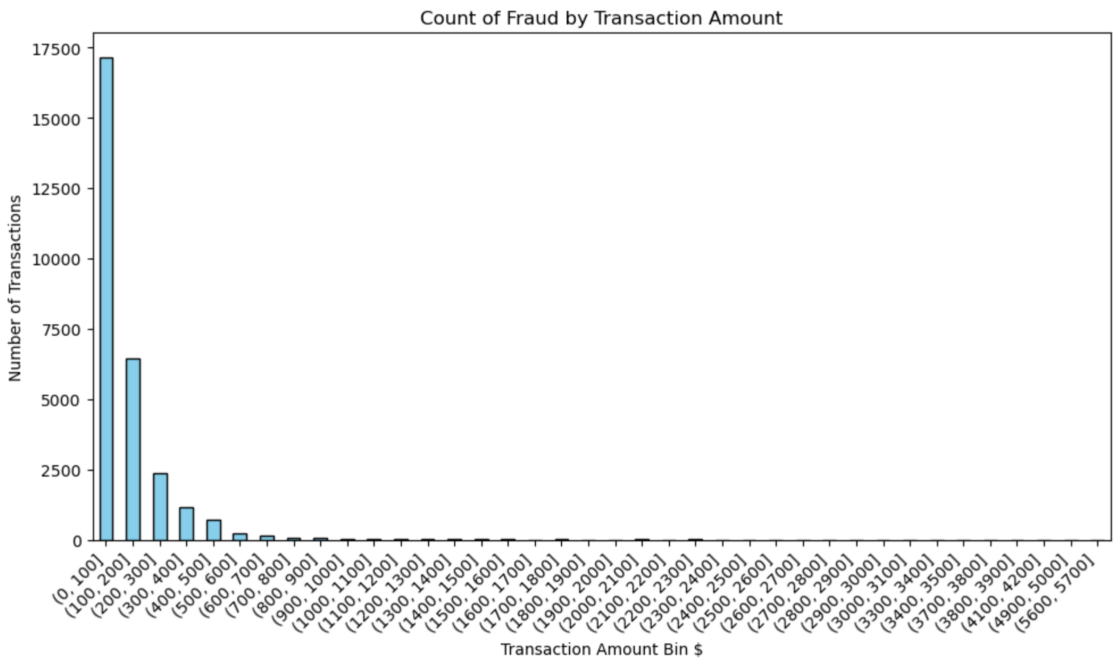
```
24386895  1999     1  2020       2   27  22:23  -54.00   Chip Transaction
24386896  1999     1  2020       2   27  22:24   54.00   Chip Transaction
24386897  1999     1  2020       2   28  07:43   59.15   Chip Transaction
24386898  1999     1  2020       2   28  20:10   43.12   Chip Transaction
24386899  1999     1  2020       2   28  23:10   45.13   Chip Transaction

                Merchant Name  Merchant City Merchant State      Zip   MCC  \
0         3527213246127876953       La Verne             CA  91750.0  5300
1         -727612092139916043  Monterey Park             CA  91754.0  5411
2         -727612092139916043  Monterey Park             CA  91754.0  5411
3         3414527459579106770  Monterey Park             CA  91754.0  5651
4         5817218446178736267       La Verne             CA  91750.0  5912
…                         …            …            …      …    …
24386895 -5162038175624867091      Merrimack             NH   3054.0  5541
24386896 -5162038175624867091      Merrimack             NH   3054.0  5541
24386897  2500998799892805156      Merrimack             NH   3054.0  4121
24386898  2500998799892805156      Merrimack             NH   3054.0  4121
24386899  4751695835751691036      Merrimack             NH   3054.0  5814

         Errors?  Is Fraud?       Date Day of Week  Hour  Minute
0            NaN          0 2002-09-01         Sun     6      21
1            NaN          0 2002-09-01         Sun     6      42
2            NaN          0 2002-09-02         Mon     6      22
3            NaN          0 2002-09-02         Mon    17      45
4            NaN          0 2002-09-03         Tue     6      23
…            …          …        …          …    …     …
24386895     NaN          0 2020-02-27         Thu    22      23
24386896     NaN          0 2020-02-27         Thu    22      24
24386897     NaN          0 2020-02-28         Fri     7      43
24386898     NaN          0 2020-02-28         Fri    20      10
24386899     NaN          0 2020-02-28         Fri    23      10

[24386900 rows x 19 columns]
```

[24]:
```python
# Create a fraud_df that has all of the fraudulent transactions
fraud_df = cc_df.loc[cc_df['Is Fraud?'] == 1].copy()
# Set the bins for transaction amounts that will be used in graphing
bins = bin_edges = range(0, int(fraud_df['Amount'].max()) + 100, 100)
fraud_df['Amount Bin'] = pd.cut(fraud_df['Amount'], bins=bin_edges)
```

[25]:
```python
# Get the number of fraud in each bin
fraud_count = fraud_df.groupby('Amount Bin', observed=True).size()
```

[26]:
```python
# Plot the figure
plt.figure(figsize=(10,6))
fraud_count.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Count of Fraud by Transaction Amount')
```

```
plt.xlabel('Transaction Amount Bin $')
plt.ylabel('Number of Transactions')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

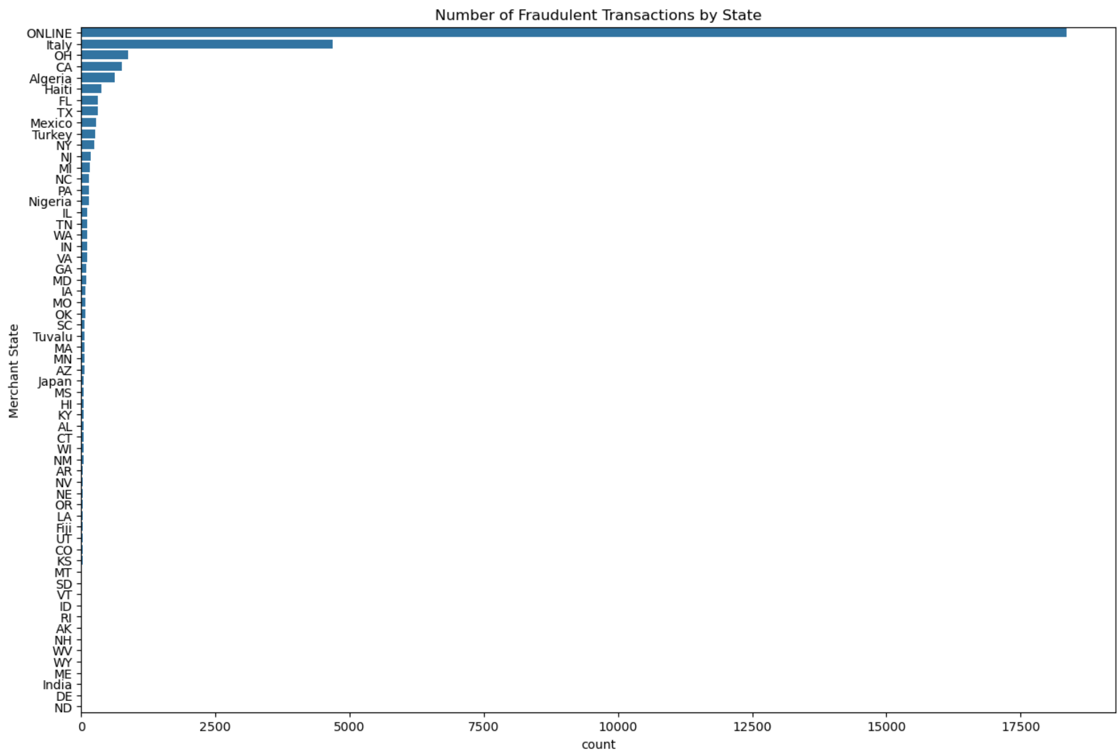


From the above graph, we can see that the vast majority of fraudulent transactions are less than \$100. Let's take a look at locations next, we will start with State (which is also Country). Additionally, we have modified the Merchant State column to include Online information.

[27]:
```
fraud_df['Merchant State'].unique()
```

[27]:
```
array(['ONLINE', 'CA', 'NY', 'Italy', 'Haiti', 'Algeria', 'OH', 'MI',
       'OK', 'TX', 'NE', 'CO', 'AZ', 'IN', 'Nigeria', 'Mexico', 'Tuvalu',
       'TN', 'FL', 'MA', 'KS', 'GA', 'NJ', 'SC', 'WA', 'UT', 'MD', 'PA',
       'AL', 'Turkey', 'IL', 'Japan', 'NC', 'HI', 'IA', 'DE', 'ID', 'MN',
       'VA', 'MS', 'KY', 'WI', 'CT', 'VT', 'AR', 'NV', 'Fiji', 'NM', 'LA',
       'OR', 'MO', 'AK', 'SD', 'ND', 'ME', 'WV', 'NH', 'India', 'WY',
       'MT', 'RI'], dtype=object)
```

[28]:
```
# Number of fraudulent transactions by State/Country
plt.figure(figsize=(15, 10))
sns.countplot(data=fraud_df, y='Merchant State', order=fraud_df['Merchant␣
↪State'].value_counts().index)
plt.title('Number of Fraudulent Transactions by State')
plt.show()
```



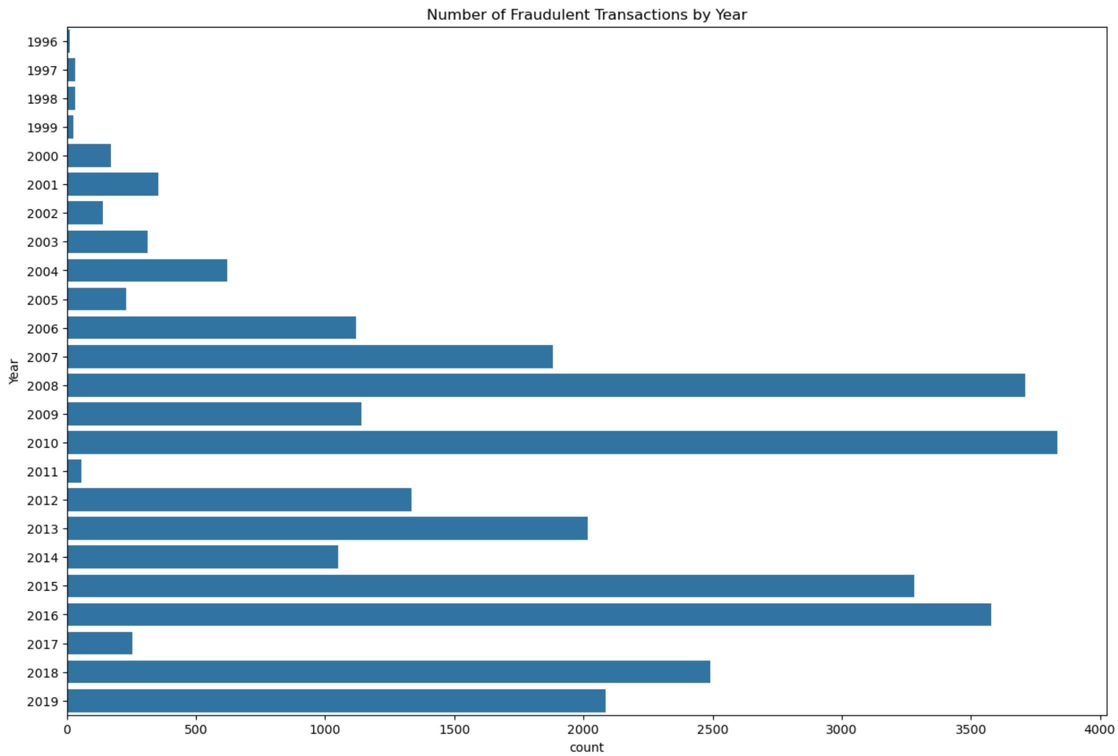
As we can see from the graph above, the majority of fraudulent transactions were made online. In second place we have Italy, and in third Ohio.

[29]:
```
fraud_df['Year'].unique()
```

[29]:
```
array([2015, 2016, 2008, 2019, 2010, 2006, 2018, 2013, 2017, 2001, 2014,
       2009, 2007, 2005, 2000, 2012, 2002, 1999, 2004, 2003, 2011, 1997,
       1998, 1996])
```

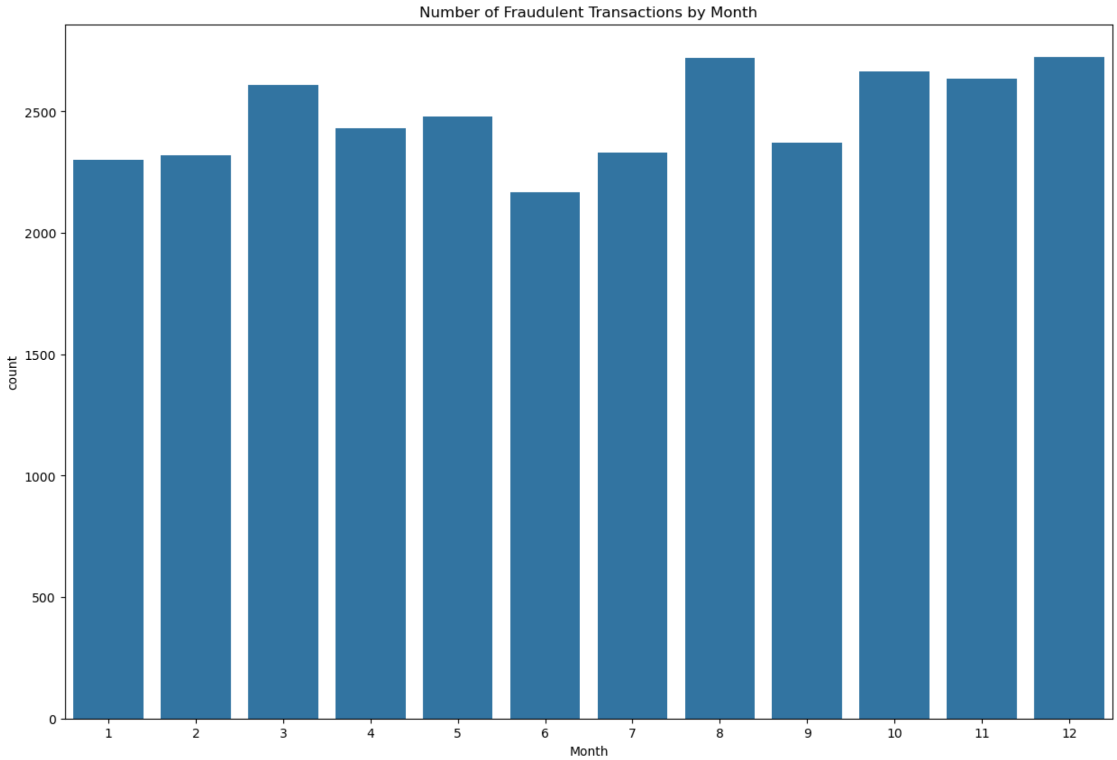
0.6 Date/Time Fraud Analysis

[30]:
```
plt.figure(figsize=(15, 10))
sns.countplot(data=fraud_df, y='Year')
plt.title('Number of Fraudulent Transactions by Year')
plt.show()
```



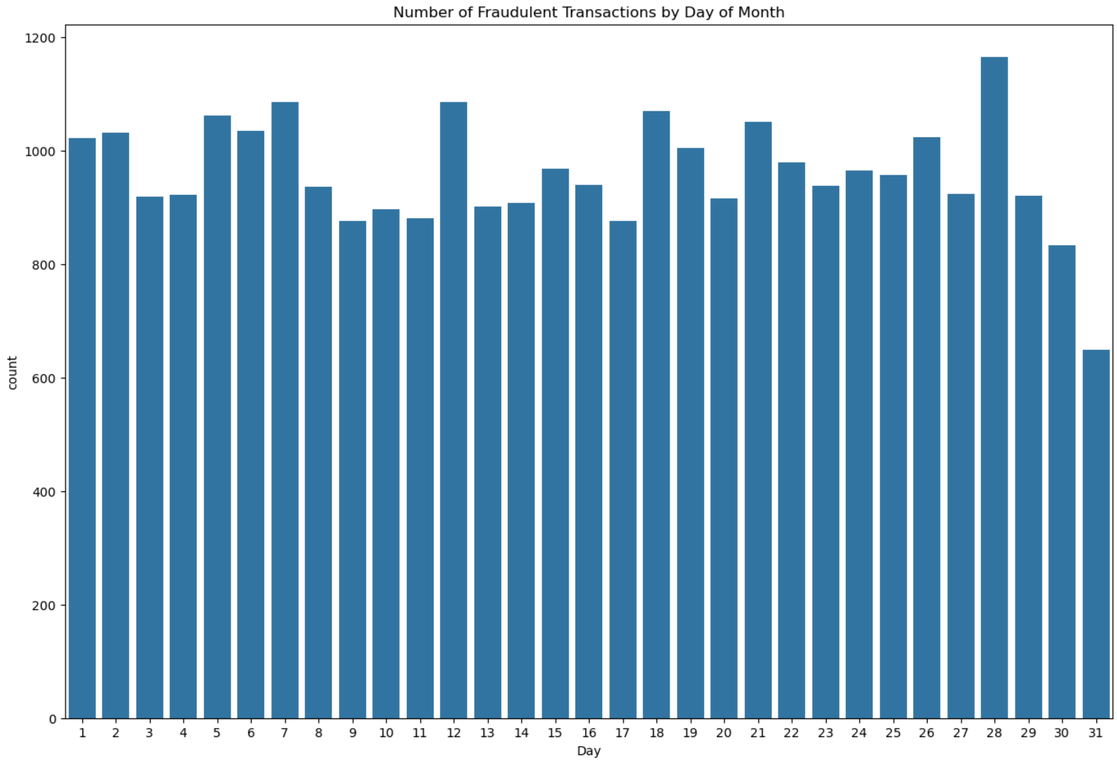
We can see that the majority of fraudulent transactions occured in 2010, 2008, and 2016.

[31]:
```
plt.figure(figsize=(15, 10))
sns.countplot(data=fraud_df, x='Month')
plt.title('Number of Fraudulent Transactions by Month')
plt.show()
```



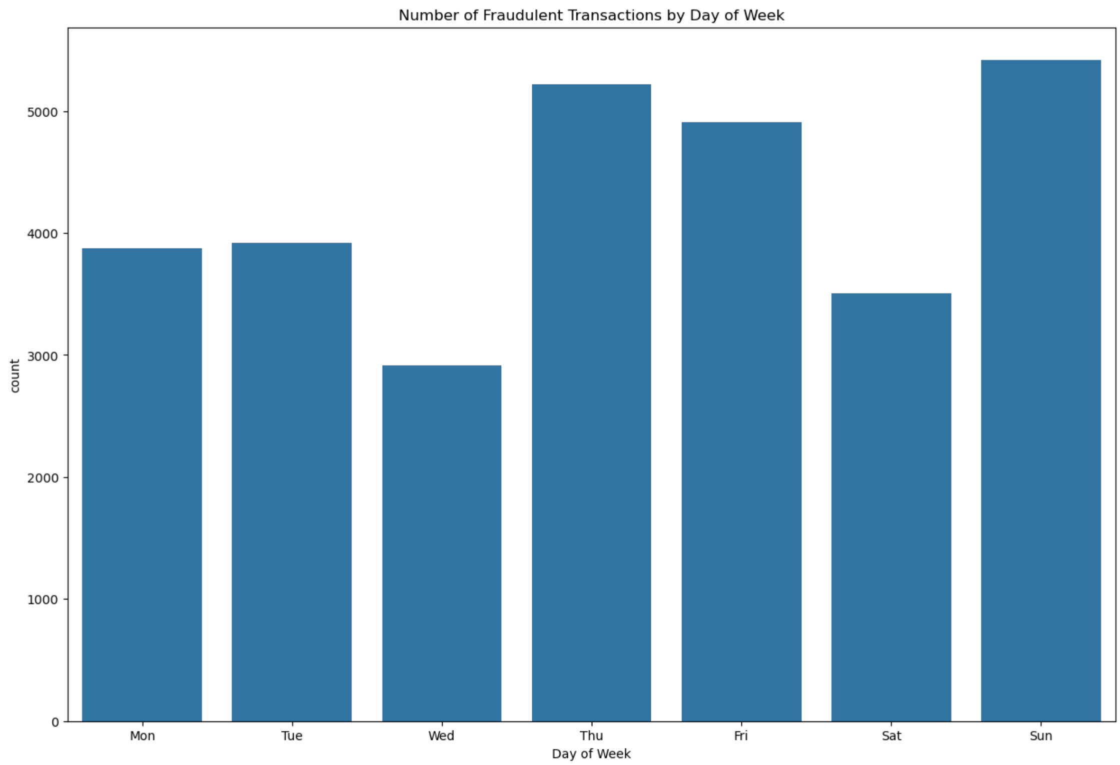
Here, we can see that the end of the year has slightly higher rates of fraudulent transactions, but it is not a very clear trend.

[32]:
```
# Graph for Day of the Month
plt.figure(figsize=(15, 10))
sns.countplot(data=fraud_df, x='Day')
plt.title('Number of Fraudulent Transactions by Day of Month')
plt.show()
```



We can see that while the 28th has the highest rate, there are not clearly defined patterns of increased fraud associated with a particular day of the month.

[33]:
```
plt.figure(figsize=(15, 10))
sns.countplot(data=fraud_df, x='Day of Week', order=['Mon', 'Tue', 'Wed',␣
  ↪'Thu', 'Fri', 'Sat', 'Sun'])
plt.title('Number of Fraudulent Transactions by Day of Week')
plt.show()
```

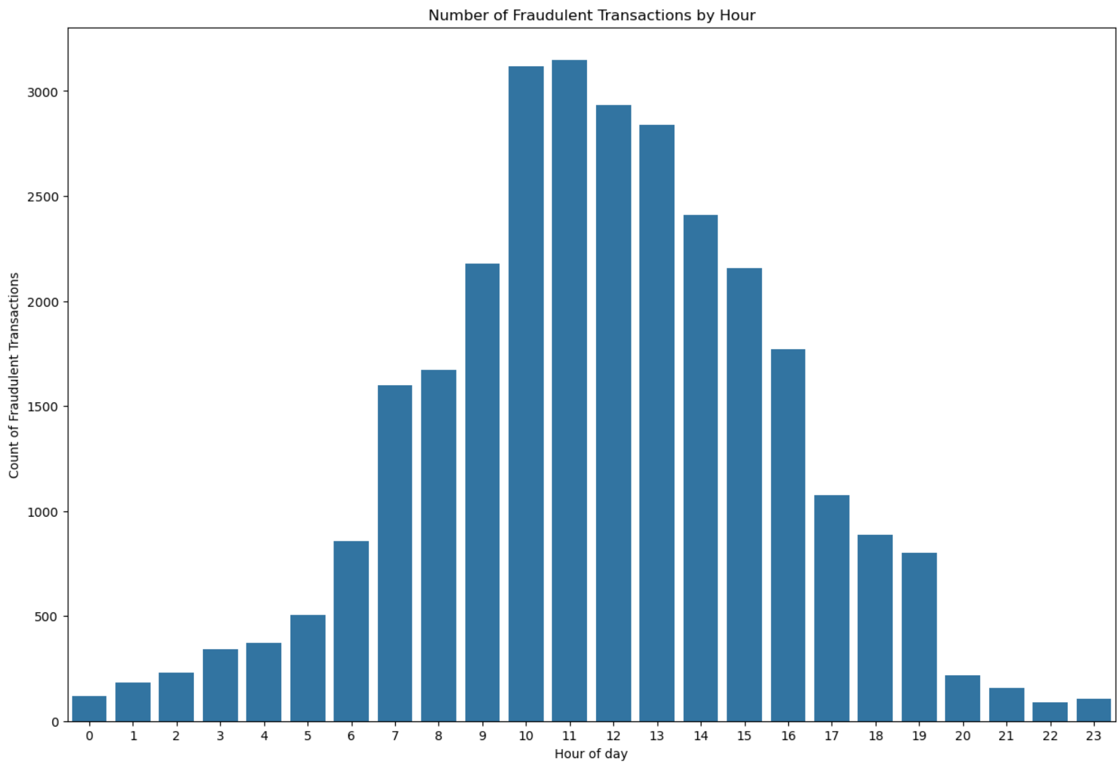


From the above graph, we can see that Sunday, Thursday, and Friday all see elevated rates of fraud.

[34]:
```
len(fraud_df['Hour'].unique())
```

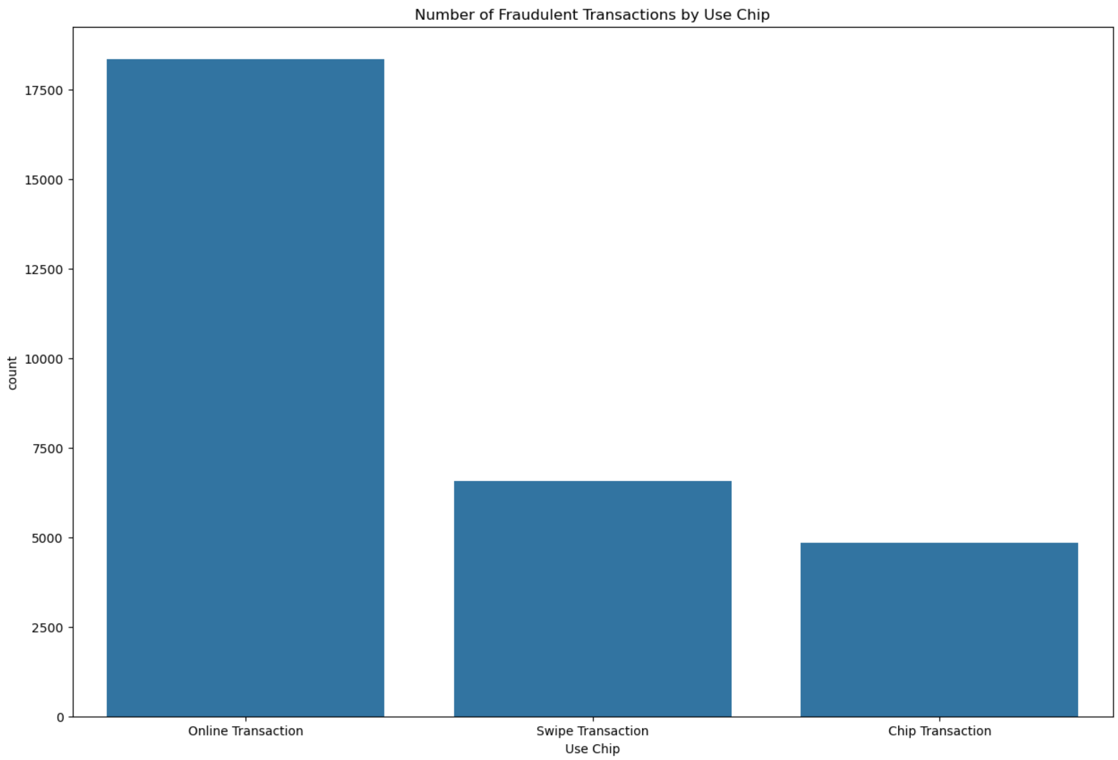
[34]: 24

[35]:
```
# Let's look at fraudulent activity based on the hours of the day
plt.figure(figsize=(15, 10))
sns.countplot(data=fraud_df, x='Hour', order=range(0, 24))  # 0 to 12 inclusive
plt.title('Number of Fraudulent Transactions by Hour')
plt.xlabel('Hour of day')
plt.ylabel('Count of Fraudulent Transactions')
plt.xticks(range(0, 24), labels=range(0, 24))
plt.show()
```

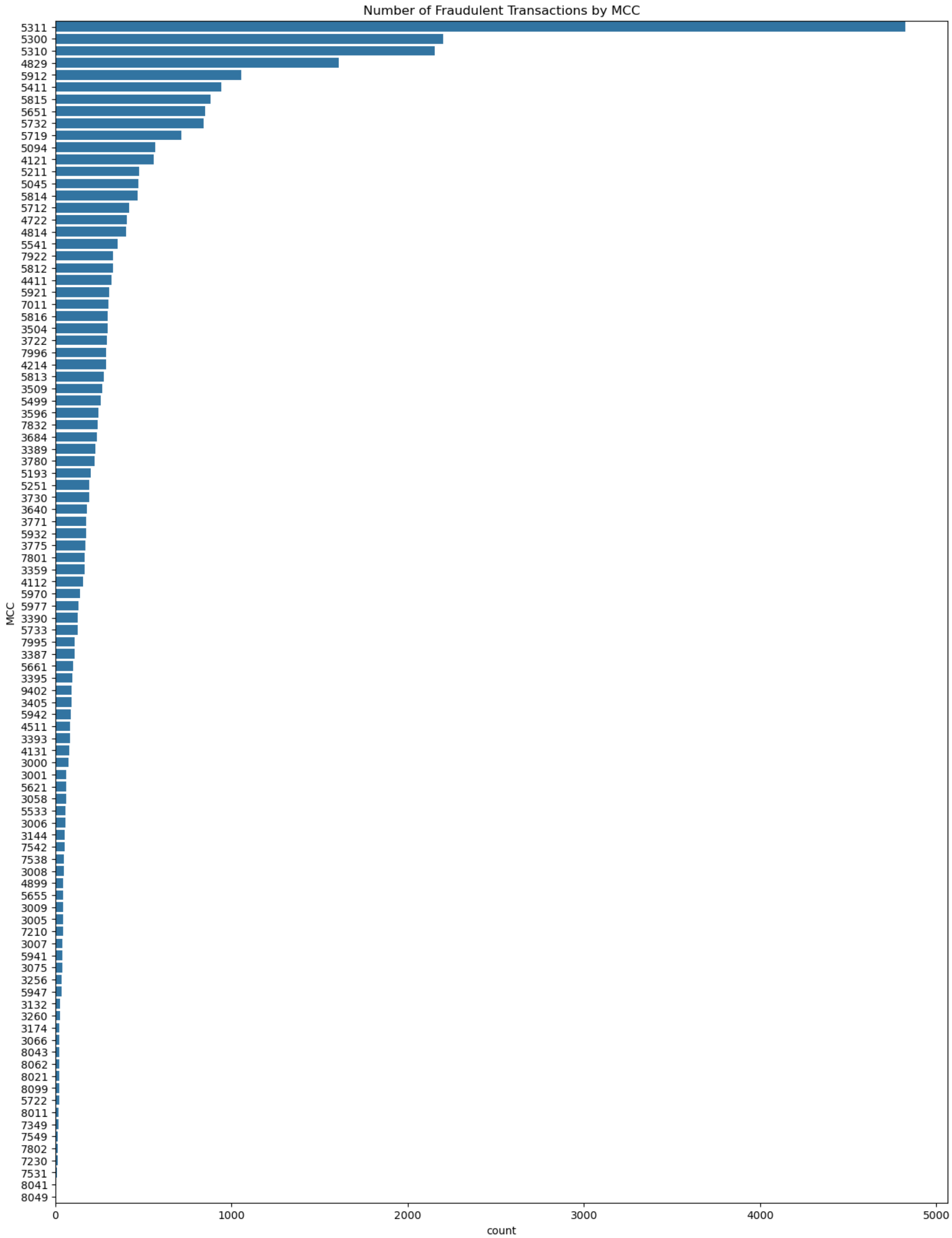


From the graph above, we can see that most of fraudulent transactions occur between 10-12 in this dataset.

[36]:
```
# Graph of use chip
plt.figure(figsize=(15, 10))
sns.countplot(data=fraud_df, x='Use Chip', order=fraud_df['Use Chip'].
 ↪value_counts().index)
plt.title('Number of Fraudulent Transactions by Use Chip')
plt.show()
```



[37]:
```
plt.figure(figsize=(15, 20))
sns.countplot(data=fraud_df, y='MCC', order=fraud_df['MCC'].value_counts().
    ↪index)
plt.title('Number of Fraudulent Transactions by MCC')
plt.show()
```



0.7 Feature Selection

Looking at the above analysis, the following features are good candidates to include in our models:

Year

Amount

State

Day of Week

Hour

Use Chip

MCC

[87]:
```
# Build the LSTM Model
def build_lstm_model(input_shape):
    model = Sequential()
    # Add the first LSTM layer
    model.add(LSTM(200, activation='tanh', input_shape=input_shape,␣
 ↪return_sequences=True))
    model.add(Dropout(0.3)) # Dropout to avoid overfitting
    # Add the second LSTM layer
    model.add(LSTM(200, activation='tanh', return_sequences=False))
    # Dense layers
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.2))
    # Output layer
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(), loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model
```

0.8 Online Transactions Model

Since we are dealing with a heavily imbalanced dataset, let's investigate whether we can improve our model's balance by focusing exclusively on online transactions where most of the fraud occurs.

[120]:
```
cc_df
```

[120]:
```
          User  Card  Year  Month  Day   Time  Amount           Use Chip  \
0            0     0  2002      9    1  06:21  134.09  Swipe Transaction
1            0     0  2002      9    1  06:42   38.48  Swipe Transaction
2            0     0  2002      9    2  06:22  120.34  Swipe Transaction
3            0     0  2002      9    2  17:45  128.95  Swipe Transaction
4            0     0  2002      9    3  06:23  104.71  Swipe Transaction
…          …    …     …    … …     …    …                 …
24386895  1999     1  2020      2   27  22:23  -54.00   Chip Transaction
24386896  1999     1  2020      2   27  22:24   54.00   Chip Transaction
24386897  1999     1  2020      2   28  07:43   59.15   Chip Transaction
24386898  1999     1  2020      2   28  20:10   43.12   Chip Transaction
24386899  1999     1  2020      2   28  23:10   45.13   Chip Transaction
```

```
                Merchant Name  Merchant City Merchant State      Zip   MCC  \
0         3527213246127876953       La Verne             CA  91750.0  5300
1         -727612092139916043  Monterey Park             CA  91754.0  5411
2         -727612092139916043  Monterey Park             CA  91754.0  5411
3         3414527459579106770  Monterey Park             CA  91754.0  5651
4         5817218446178736267       La Verne             CA  91750.0  5912
…                       …            …            …      …    …
24386895 -5162038175624867091      Merrimack             NH   3054.0  5541
24386896 -5162038175624867091      Merrimack             NH   3054.0  5541
24386897  2500998799892805156      Merrimack             NH   3054.0  4121
24386898  2500998799892805156      Merrimack             NH   3054.0  4121
24386899  4751695835751691036      Merrimack             NH   3054.0  5814

         Errors?  Is Fraud?        Date Day of Week  Hour  Minute
0            NaN          0 2002-09-01         Sun     6      21
1            NaN          0 2002-09-01         Sun     6      42
2            NaN          0 2002-09-02         Mon     6      22
3            NaN          0 2002-09-02         Mon    17      45
4            NaN          0 2002-09-03         Tue     6      23
…          …         …          …          …    …      …
24386895     NaN          0 2020-02-27         Thu    22      23
24386896     NaN          0 2020-02-27         Thu    22      24
24386897     NaN          0 2020-02-28         Fri     7      43
24386898     NaN          0 2020-02-28         Fri    20      10
24386899     NaN          0 2020-02-28         Fri    23      10

[24386900 rows x 19 columns]
```

[121]:
```
online_model_cols = ['User', 'Year', 'Day of Week', 'Hour', 'Amount', 'Use Chip', 'MCC', 'Is Fraud?']
online_df = cc_df.loc[:,online_model_cols]
online_df = online_df.loc[cc_df['Merchant State'] == 'ONLINE']
```

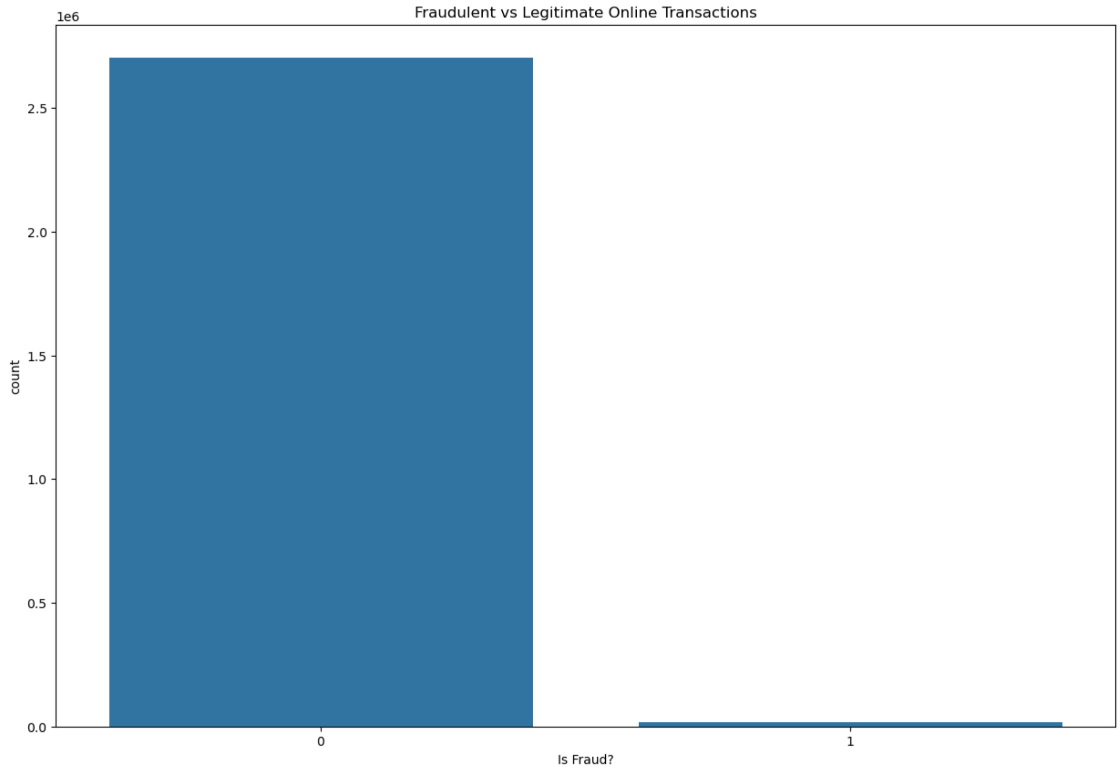
[122]:
```
print(f"There are {cc_df.shape[0]} records in the cc_df and {online_df.shape[0]} records in the online_df.")
print(f"We have eliminated {cc_df.shape[0] - online_df.shape[0]} records by using the online_df.")
```

```
There are 24386900 records in the cc_df and 2720821 records in the online_df.
We have eliminated 21666079 records by using the online_df.
```

[123]:
```
online_df.shape
```

[123]:
```
(2720821, 8)
```

[124]:
```
# Fraudulent vs legitimate transactions
plt.figure(figsize=(15, 10))
sns.countplot(data=online_df, x='Is Fraud?')
plt.title('Fraudulent vs Legitimate Online Transactions')
plt.show()
```



0.9 Online Model Create - Random Forest

[180]:
```
online_rf_df = online_df.copy()
online_rf_df
```

[180]:
```
          User  Year Day of Week  Hour  Amount           Use Chip   MCC  \
11           0  2002         Thu    20   53.91  Online Transaction  4900
24           0  2002         Mon    20  144.90  Online Transaction  4899
85           0  2002         Mon     6  127.32  Online Transaction  5311
99           0  2002         Sun     6  139.39  Online Transaction  5311
106          0  2002         Wed     8   53.09  Online Transaction  5193
…          …     …            …     …      …                   …     …
24386877  1999  2020         Mon    20   55.79  Online Transaction  4121
24386879  1999  2020         Tue     7   43.08  Online Transaction  4121
24386880  1999  2020         Tue     7   43.76  Online Transaction  4121
24386884  1999  2020         Wed     7   45.18  Online Transaction  4121
```

```
24386889  1999  2020         Thu     7   47.18  Online Transaction  4121

          Is Fraud?
11               0
24               0
85               0
99               0
106              0
…             …
24386877         0
24386879         0
24386880         0
24386884         0
24386889         0

[2720821 rows x 8 columns]
```

[181]:
```
# Target encode the user features
user_target_mean = online_rf_df.groupby('User')['Is Fraud?'].mean()
online_rf_df['User_encoded'] = online_rf_df['User'].map(user_target_mean)
online_rf_df = online_rf_df.drop(columns=['User'])
# Convert columns to the appropriate dtype
online_rf_df['MCC'] = online_rf_df['MCC'].astype('category')

# Encode the Object columns accordingly
online_rf_df = pd.get_dummies(online_rf_df, columns=['Day of Week', 'Use␣
↪Chip'], prefix="Day")

# Create a binary encoder
rf_binary_encoder = ce.BinaryEncoder(cols=['MCC'])
online_rf_df = rf_binary_encoder.fit_transform(online_rf_df)

# Sine-Cosine Encoding for Hour and Year
online_rf_df['Hour_Sin'] = np.sin(2 * np.pi * online_rf_df['Hour'] / 24)
online_rf_df['Hour_Cos'] = np.cos(2 * np.pi * online_rf_df['Hour'] /24)
online_rf_df.drop(columns='Hour', inplace=True)

rf_target_column = "Is Fraud?"

online_rf_df
```

[181]:
```
          Year  Amount  MCC_0  MCC_1  MCC_2  MCC_3  MCC_4  MCC_5  MCC_6  \
11        2002   53.91      0      0      0      0      0      0      1
24        2002  144.90      0      0      0      0      0      1      0
85        2002  127.32      0      0      0      0      0      1      1
99        2002  139.39      0      0      0      0      0      1      1
106       2002   53.09      0      0      0      0      1      0      0
```

```
…          …     …     …      …      …      …      …      …      …
24386877  2020   55.79      0      0      1      0      1      0      0
24386879  2020   43.08      0      0      1      0      1      0      0
24386880  2020   43.76      0      0      1      0      1      0      0
24386884  2020   45.18      0      0      1      0      1      0      0
24386889  2020   47.18      0      0      1      0      1      0      0

          Is Fraud?  …  Day_Mon  Day_Sat  Day_Sun  Day_Thu  Day_Tue  \
11                0  …    False    False    False     True    False
24                0  …     True    False    False    False    False
85                0  …     True    False    False    False    False
99                0  …    False    False     True    False    False
106               0  …    False    False    False    False    False
…               …  …      …      …      …      …      …
24386877          0  …     True    False    False    False    False
24386879          0  …    False    False    False    False     True
24386880          0  …    False    False    False    False     True
24386884          0  …    False    False    False    False    False
24386889          0  …    False    False    False     True    False

          Day_Wed  Day_Chip Transaction  Day_Online Transaction  Hour_Sin  \
11          False                 False                    True -0.866025
24          False                 False                    True -0.866025
85          False                 False                    True  1.000000
99          False                 False                    True  1.000000
106          True                 False                    True  0.866025
…           …                   …                      …       …
24386877    False                 False                    True -0.866025
24386879    False                 False                    True  0.965926
24386880    False                 False                    True  0.965926
24386884     True                 False                    True  0.965926
24386889    False                 False                    True  0.965926

              Hour_Cos
11        5.000000e-01
24        5.000000e-01
85        6.123234e-17
99        6.123234e-17
106      -5.000000e-01
…                 …
24386877  5.000000e-01
24386879 -2.588190e-01
24386880 -2.588190e-01
24386884 -2.588190e-01
24386889 -2.588190e-01

[2720821 rows x 22 columns]
```

[182]:
```python
# Split the features from the target variable
rf_X = online_rf_df.drop(columns=[rf_target_column])
rf_y = online_rf_df[rf_target_column]

# Train Test Split
rf_X_train, rf_X_test, rf_y_train, rf_y_test = train_test_split(rf_X, rf_y,␣
↪test_size=0.2, random_state=42, stratify=rf_y)
```

[183]:
```python
# Create RF Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42,␣
↪class_weight='balanced', n_jobs=-1)

# Fit the model
rf_model.fit(rf_X_train, rf_y_train)
```

[183]:
```
RandomForestClassifier(class_weight='balanced', n_jobs=-1, random_state=42)
```

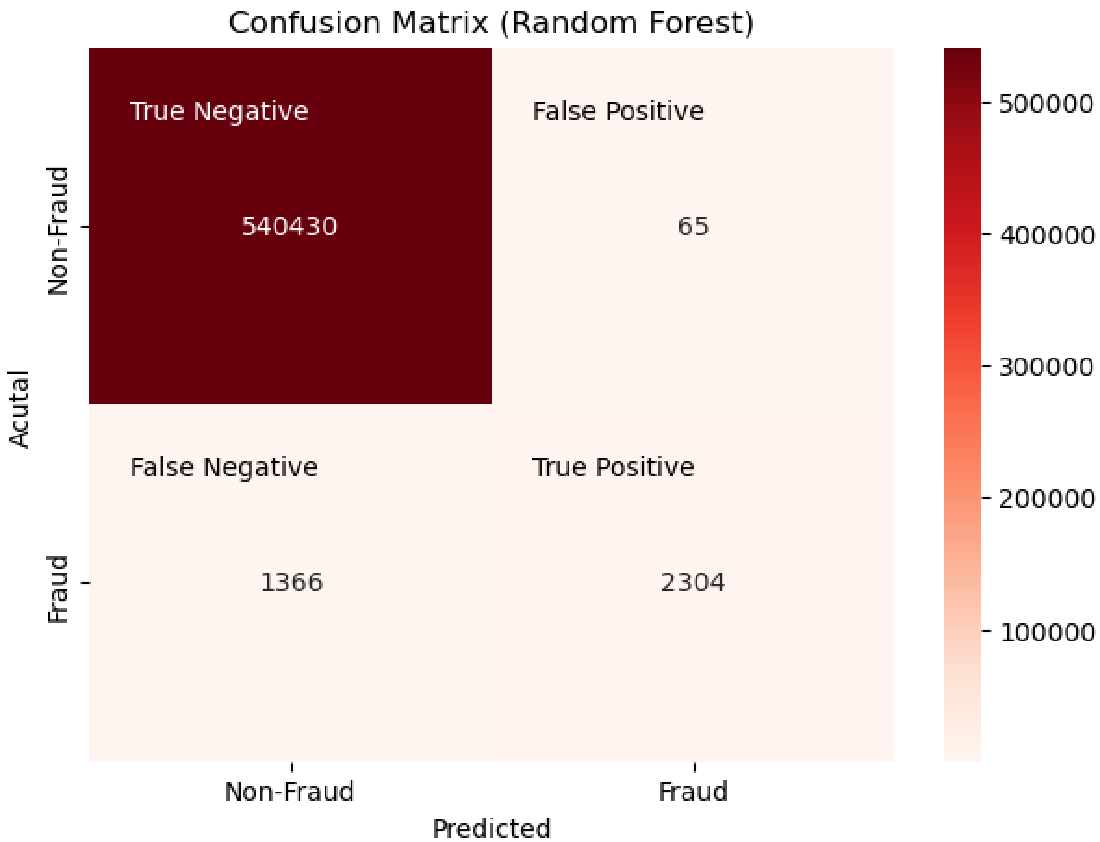
[184]:
```python
# Evaulate the model
rf_y_pred = rf_model.predict(rf_X_test)
rf_y_pred_proba = rf_model.predict_proba(rf_X_test)[:, 1]
```

[191]:
```python
rf_cm = confusion_matrix(rf_y_test, rf_y_pred)
print(f"Classification Report:\n{classification_report(rf_y_test, rf_y_pred)}")
print(f"Confusion Matrix:\n{rf_cm}")
print(f"ROC AUC Score:\n {roc_auc_score(rf_y_test, rf_y_pred_proba)}")
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    540495
           1       0.97      0.63      0.76      3670

    accuracy                           1.00    544165
   macro avg       0.99      0.81      0.88    544165
weighted avg       1.00      1.00      1.00    544165

Confusion Matrix:
[[540430     65]
 [  1366   2304]]
ROC AUC Score:
 0.9812301913275431
```

[196]:
```python
# Plot the confusion matrix
plt.figure(figsize=(7,5))
annotations = [f'TN: {rf_cm[0,0]}', f'FP: {online_cm[0,1]}'], [f'FN:␣
↪{rf_cm[1,0]}', f'TP: {rf_cm[1,1]}']
hm = sns.heatmap(rf_cm, annot=True, fmt='d', cmap='Reds',␣
↪xticklabels=['Non-Fraud', 'Fraud'], yticklabels=['Non-Fraud', 'Fraud'])
```

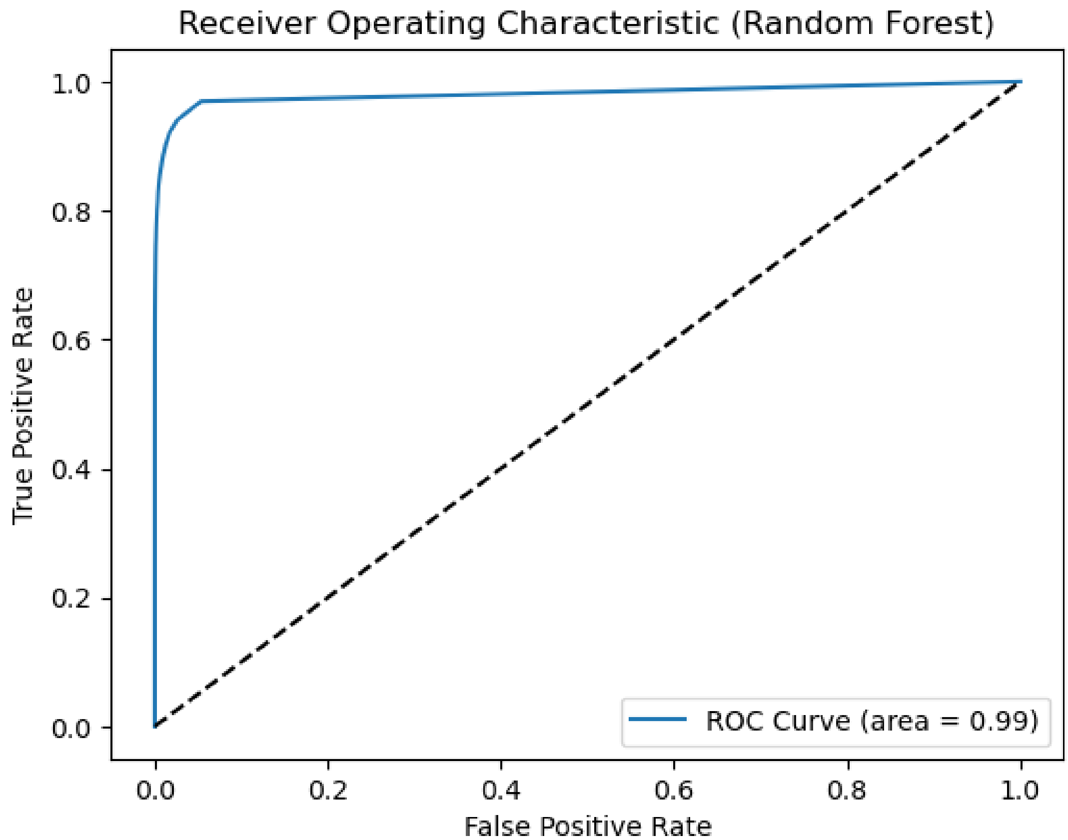
```
plt.xlabel('Predicted')
plt.ylabel('Acutal')
plt.title('Confusion Matrix (Random Forest)')
hm.text(0.1, 0.2, 'True Negative', color='white')
hm.text(1.1, 0.2, 'False Positive')
hm.text(0.1, 1.2, 'False Negative')
hm.text(1.1, 1.2, 'True Positive')
plt.show()
```



[192]:
```
# Get the ROC Curve
rf_fpr, rf_tpr, rf_thresholds = roc_curve(rf_y_test, rf_y_pred_proba)
rf_roc_auc = auc(rf_fpr, rf_tpr)
```

[195]:
```
# Plot the ROC curve
plt.figure()
plt.plot(rf_fpr, rf_tpr, label=f'ROC Curve (area = {online_roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Random guess line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (Random Forest)')
```

```
plt.legend(loc='lower right')
plt.show()
```



0.10 Online Model Creation - LSTM

[198]:
```
# Create the dataset to be used by the models
online_model_df = online_df.copy()
```

0.11 Encode the data

[126]:
```
# Target encode the user features
user_target_mean = online_model_df.groupby('User')['Is Fraud?'].mean()
online_model_df['User_encoded'] = online_model_df['User'].map(user_target_mean)
online_model_df = online_model_df.drop(columns=['User'])
# Convert columns to the appropriate dtype
online_model_df['MCC'] = online_model_df['MCC'].astype('category')

# Encode the Object columns accordingly
online_model_df = pd.get_dummies(online_model_df, columns=['Day of Week', 'Use␣
↪Chip'], prefix="Day")
```

```python
# Create a binary encoder
online_binary_encoder = ce.BinaryEncoder(cols=['MCC'])
online_model_df = online_binary_encoder.fit_transform(online_model_df)

# Sine-Cosine Encoding for Hour and Year
online_model_df['Hour_Sin'] = np.sin(2 * np.pi * online_model_df['Hour'] / 24)
online_model_df['Hour_Cos'] = np.cos(2 * np.pi * online_model_df['Hour'] /24)
online_model_df.drop(columns='Hour', inplace=True)

online_target_column = "Is Fraud?"

online_model_df
```

[126]:

```
          Year  Amount  MCC_0  MCC_1  MCC_2  MCC_3  MCC_4  MCC_5  MCC_6  \
11        2002   53.91      0      0      0      0      0      0      1
24        2002  144.90      0      0      0      0      0      1      0
85        2002  127.32      0      0      0      0      0      1      1
99        2002  139.39      0      0      0      0      0      1      1
106       2002   53.09      0      0      0      0      1      0      0
…        …     …     …     …     …     …     …     …     …     …
24386877  2020   55.79      0      0      1      0      1      0      0
24386879  2020   43.08      0      0      1      0      1      0      0
24386880  2020   43.76      0      0      1      0      1      0      0
24386884  2020   45.18      0      0      1      0      1      0      0
24386889  2020   47.18      0      0      1      0      1      0      0

          Is Fraud?  …  Day_Mon  Day_Sat  Day_Sun  Day_Thu  Day_Tue  \
11                0  …    False    False    False     True    False
24                0  …     True    False    False    False    False
85                0  …     True    False    False    False    False
99                0  …    False    False     True    False    False
106               0  …    False    False    False    False    False
…              …   …      …      …      …      …      …
24386877          0  …     True    False    False    False    False
24386879          0  …    False    False    False    False     True
24386880          0  …    False    False    False    False     True
24386884          0  …    False    False    False    False    False
24386889          0  …    False    False    False     True    False

          Day_Wed  Day_Chip Transaction  Day_Online Transaction  Hour_Sin  \
11          False                 False                    True -0.866025
24          False                 False                    True -0.866025
85          False                 False                    True  1.000000
99          False                 False                    True  1.000000
106          True                 False                    True  0.866025
…           …                   …                      …       …
```

```
24386877    False                False                    True -0.866025
24386879    False                False                    True  0.965926
24386880    False                False                    True  0.965926
24386884     True                False                    True  0.965926
24386889    False                False                    True  0.965926

             Hour_Cos
11        5.000000e-01
24        5.000000e-01
85        6.123234e-17
99        6.123234e-17
106      -5.000000e-01
…                 …
24386877  5.000000e-01
24386879 -2.588190e-01
24386880 -2.588190e-01
24386884 -2.588190e-01
24386889 -2.588190e-01

[2720821 rows x 22 columns]
```

[127]:

```python
# Use compute_class_weights to handle class imbalance
online_y_train = online_df['Is Fraud?']

# Compute class weights
class_weights = compute_class_weight(class_weight='balanced', classes=np.
 ↪unique(online_y_train), y=online_y_train)

# Get Class Weights
class_weights_dict = {i: weight for i, weight in enumerate(class_weights)}
class_weights_dict
```

[127]:

```
{0: 0.5033948547848044, 1: 74.14085236252657}
```

[128]:

```python
# Split the features from the target variable
online_X = online_model_df.drop(columns=['Is Fraud?']) # Features
online_y = online_model_df['Is Fraud?'] # Target

# Create the train and  test sets
online_X_train, online_X_test, online_y_train, online_y_test =␣
 ↪train_test_split(online_X, online_y, test_size=0.2, random_state=42)

# Scale the data for the Random Forest Classifier and LSTM
online_scaler = StandardScaler()
online_X_train_scaled = online_scaler.fit_transform(online_X_train)
online_X_test_scaled = online_scaler.transform(online_X_test)
```

[129]:
```
# Make sure that my GPU is available
print(f"Num GPUs available: {len(tf.config.list_physical_devices('GPU'))}")

# Reshape the data for LSTM
online_X_train_lstm = np.reshape(online_X_train_scaled, (online_X_train_scaled.
 ↪shape[0], 1, online_X_train_scaled.shape[1]))
online_X_test_lstm = np.reshape(online_X_test_scaled, (online_X_test_scaled.
 ↪shape[0], 1, online_X_test_scaled.shape[1]))
```

```
Num GPUs available: 1
```

[130]:
```
# Build and train the model
online_input_shape = (online_X_train_lstm.shape[1], online_X_train_lstm.
 ↪shape[2])
online_lstm_model = build_lstm_model(online_input_shape)
```

[131]:
```
online_lstm_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 1, 200)            177600

 dropout_2 (Dropout)         (None, 1, 200)            0

 lstm_3 (LSTM)               (None, 200)               320800

 dense_2 (Dense)             (None, 32)                6432

 dropout_3 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 504865 (1.93 MB)
Trainable params: 504865 (1.93 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[132]:
```
# Time the training process
online_start_time = time.time()
online_history = online_lstm_model.fit(online_X_train_lstm, online_y_train,␣
 ↪epochs=10, batch_size=64, validation_split=0.2,␣
 ↪class_weight=class_weights_dict)
online_end_time = time.time()

print(f"Training time: {online_end_time - online_start_time:.2f} seconds")
```

```
Epoch 1/10
27209/27209 [==============================] - 308s 11ms/step - loss: 0.2595 -
accuracy: 0.8695 - val_loss: 0.1754 - val_accuracy: 0.9113
Epoch 2/10
27209/27209 [==============================] - 314s 12ms/step - loss: 0.1802 -
accuracy: 0.9113 - val_loss: 0.1458 - val_accuracy: 0.9260
Epoch 3/10
27209/27209 [==============================] - 309s 11ms/step - loss: 0.1726 -
accuracy: 0.9169 - val_loss: 0.1390 - val_accuracy: 0.9332
Epoch 4/10
27209/27209 [==============================] - 302s 11ms/step - loss: 0.1695 -
accuracy: 0.9169 - val_loss: 0.1596 - val_accuracy: 0.9135
Epoch 5/10
27209/27209 [==============================] - 309s 11ms/step - loss: 0.1681 -
accuracy: 0.9176 - val_loss: 0.1545 - val_accuracy: 0.9224
Epoch 6/10
27209/27209 [==============================] - 306s 11ms/step - loss: 0.1666 -
accuracy: 0.9180 - val_loss: 0.1568 - val_accuracy: 0.9269
Epoch 7/10
27209/27209 [==============================] - 296s 11ms/step - loss: 0.1799 -
accuracy: 0.9174 - val_loss: 0.1485 - val_accuracy: 0.9274
Epoch 8/10
27209/27209 [==============================] - 288s 11ms/step - loss: 0.1904 -
accuracy: 0.9166 - val_loss: 0.1340 - val_accuracy: 0.9292
Epoch 9/10
27209/27209 [==============================] - 281s 10ms/step - loss: 0.1973 -
accuracy: 0.9160 - val_loss: 0.1347 - val_accuracy: 0.9296
Epoch 10/10
27209/27209 [==============================] - 283s 10ms/step - loss: 0.2345 -
accuracy: 0.9128 - val_loss: 0.2338 - val_accuracy: 0.8977
Training time: 2997.02 seconds
```

[133]:
```
online_loss, online_accuracy = online_lstm_model.evaluate(online_X_test_lstm,
    online_y_test, batch_size=64, verbose=1)
```

```
8503/8503 [==============================] - 39s 5ms/step - loss: 0.2332 -
accuracy: 0.8981
```

[134]:
```
print(f"Test loss: {online_loss:.4f}")
print(f"Test Accuracy: {online_accuracy:.4f}")
```

```
Test loss: 0.2332
Test Accuracy: 0.8981
```

[135]:
```
online_lstm_pred = online_lstm_model.predict(online_X_test_lstm)
```

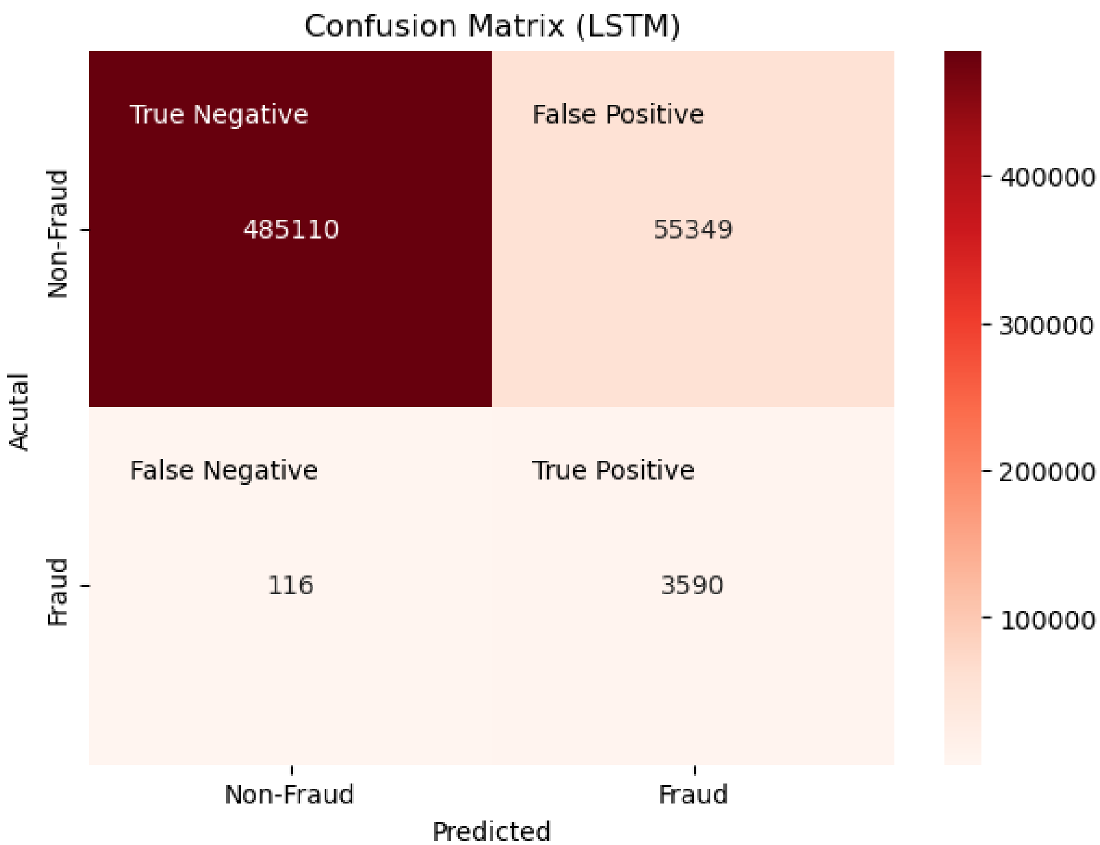
```
17006/17006 [==============================] - 46s 3ms/step
```

[136]:
```python
# Convert probabilities to binary predictions (0 or 1)
threshold = 0.5
online_binary_predictions = (online_lstm_pred > threshold).astype(int)
```

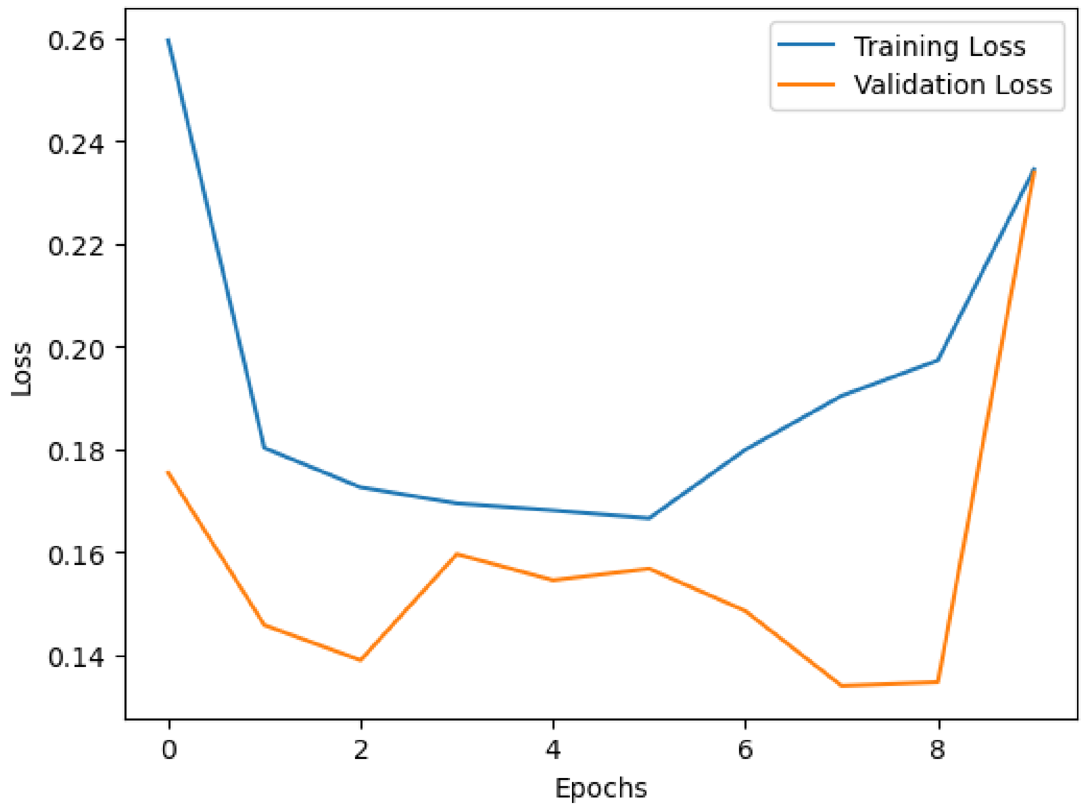
[137]:
```python
# Classification Report
print(classification_report(online_y_test, online_binary_predictions))

# Confusion Matrix
print(f"Confusion Matrix:\n{confusion_matrix(online_y_test,␣
↪online_binary_predictions)}")
```

```
              precision    recall  f1-score   support

           0       1.00      0.90      0.95    540459
           1       0.06      0.97      0.11      3706

    accuracy                           0.90    544165
   macro avg       0.53      0.93      0.53    544165
weighted avg       0.99      0.90      0.94    544165

Confusion Matrix:
[[485110  55349]
 [   116   3590]]
```

[197]:
```python
online_cm = confusion_matrix(online_y_test, online_binary_predictions)
# Plot the confusion matrix
plt.figure(figsize=(7,5))
annotations = [f'TN: {online_cm[0,0]}', f'FP: {online_cm[0,1]}'], [f'FN:␣
↪{online_cm[1,0]}', f'TP: {online_cm[1,1]}']
hm = sns.heatmap(online_cm, annot=True, fmt='d', cmap='Reds',␣
↪xticklabels=['Non-Fraud', 'Fraud'], yticklabels=['Non-Fraud', 'Fraud'])
plt.xlabel('Predicted')
plt.ylabel('Acutal')
plt.title('Confusion Matrix (LSTM)')
hm.text(0.1, 0.2, 'True Negative', color='white')
hm.text(1.1, 0.2, 'False Positive')
hm.text(0.1, 1.2, 'False Negative')
hm.text(1.1, 1.2, 'True Positive')
plt.show()
```

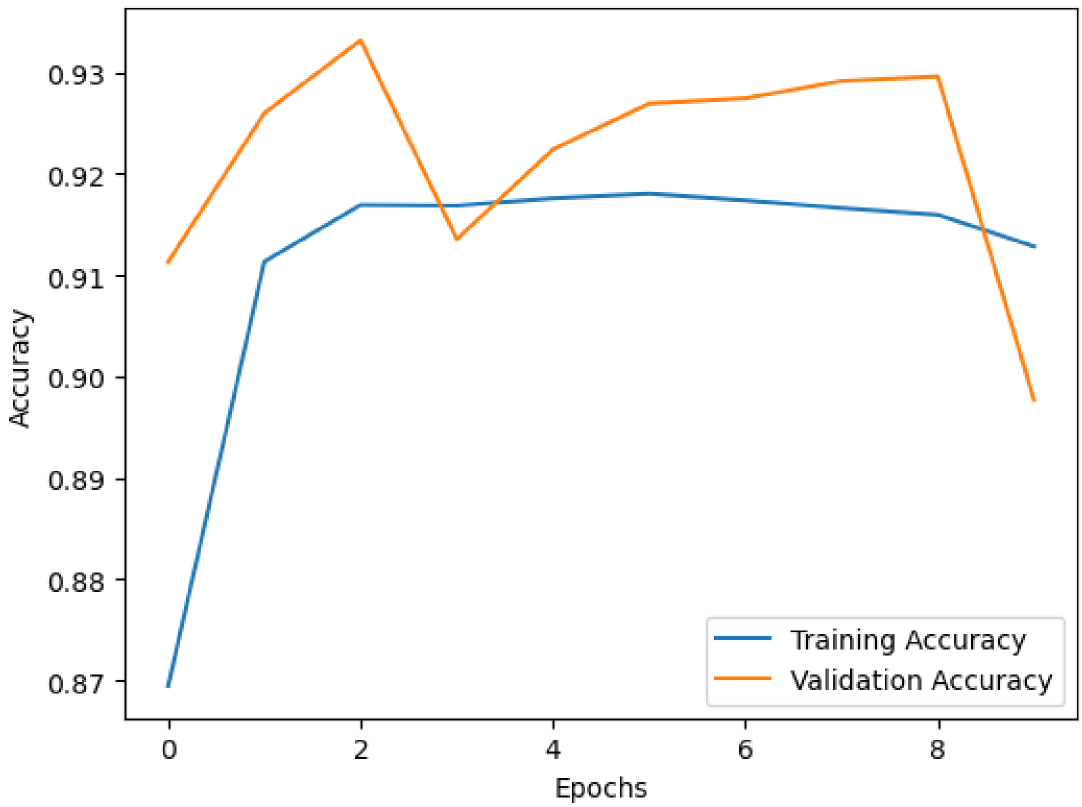


[138]:
```python
import matplotlib.pyplot as plt

# Plot loss
plt.plot(online_history.history['loss'], label='Training Loss')
plt.plot(online_history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

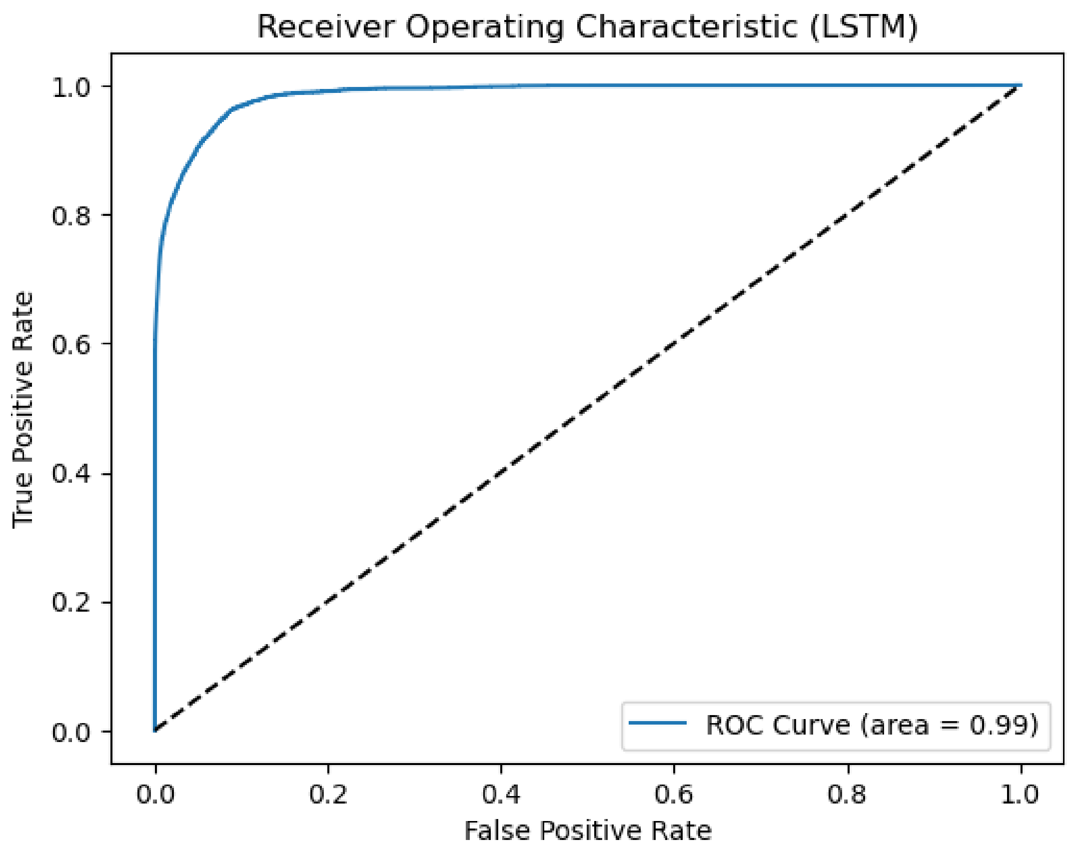


[139]:

```
# Plot accuracy
plt.plot(online_history.history['accuracy'], label='Training Accuracy')
plt.plot(online_history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



[194]:
```
# Get the ROC curve
online_fpr, online_tpr, online_thresholds = roc_curve(online_y_test,␣
 ↪online_lstm_pred)
online_roc_auc = auc(online_fpr, online_tpr)

# Plot the ROC curve
plt.figure()
plt.plot(online_fpr, online_tpr, label=f'ROC Curve (area = {online_roc_auc:.
 ↪2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Random guess line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (LSTM)')
plt.legend(loc='lower right')
plt.show()
```



[141]:
```
# Save the model
online_lstm_model.save('online_lstm_fraud_model2_user.keras')

# Loading the model
# from tensorflow.keras.models import load_model
# model = load_model('lstm_fraud_model.h5')
```

0.12 Experimental: Random Undersampling

[42]:
```
cc_df
```

[42]:
```
          User  Card  Year  Month  Day   Time  Amount            Use Chip  \
0            0     0  2002      9    1  06:21  134.09  Swipe Transaction
1            0     0  2002      9    1  06:42   38.48  Swipe Transaction
2            0     0  2002      9    2  06:22  120.34  Swipe Transaction
3            0     0  2002      9    2  17:45  128.95  Swipe Transaction
4            0     0  2002      9    3  06:23  104.71  Swipe Transaction
…          …     …     …     …    …     …       …                   …
24386895  1999     1  2020      2   27  22:23  -54.00   Chip Transaction
24386896  1999     1  2020      2   27  22:24   54.00   Chip Transaction
```

```
24386897  1999     1  2020       2   28  07:43   59.15   Chip Transaction
24386898  1999     1  2020       2   28  20:10   43.12   Chip Transaction
24386899  1999     1  2020       2   28  23:10   45.13   Chip Transaction

                Merchant Name  Merchant City Merchant State      Zip   MCC  \
0         3527213246127876953      La Verne             CA  91750.0  5300
1         -727612092139916043  Monterey Park            CA  91754.0  5411
2         -727612092139916043  Monterey Park            CA  91754.0  5411
3         3414527459579106770  Monterey Park            CA  91754.0  5651
4         5817218446178736267      La Verne             CA  91750.0  5912
…                         …            …              …      …    …
24386895 -5162038175624867091     Merrimack             NH   3054.0  5541
24386896 -5162038175624867091     Merrimack             NH   3054.0  5541
24386897  2500998799892805156     Merrimack             NH   3054.0  4121
24386898  2500998799892805156     Merrimack             NH   3054.0  4121
24386899  4751695835751691036     Merrimack             NH   3054.0  5814

         Errors? Is Fraud?       Date Day of Week  Hour  Minute
0            NaN          0 2002-09-01         Sun     6      21
1            NaN          0 2002-09-01         Sun     6      42
2            NaN          0 2002-09-02         Mon     6      22
3            NaN          0 2002-09-02         Mon    17      45
4            NaN          0 2002-09-03         Tue     6      23
…            …          …        …           …   …     …
24386895     NaN          0 2020-02-27         Thu    22      23
24386896     NaN          0 2020-02-27         Thu    22      24
24386897     NaN          0 2020-02-28         Fri     7      43
24386898     NaN          0 2020-02-28         Fri    20      10
24386899     NaN          0 2020-02-28         Fri    23      10

[24386900 rows x 19 columns]
```

[43]:
```
user_df
```

[43]:
```
                Person  Current Age  Retirement Age  Birth Year  Birth Month  \
0       Hazel Robinson           53              66        1966           11
1           Sasha Sadr           53              68        1966           12
2           Saanvi Lee           81              67        1938           11
3        Everlee Clark           63              63        1957            1
4        Kyle Peterson           43              70        1976            9
…                  …          …             …         …          …
1995      Jose Faraday           32              70        1987            7
1996  Ximena Richardson           62              65        1957           11
1997    Annika Russell           47              67        1973            1
1998       Juelz Roman           66              60        1954            2
1999      Kenia Harris           21              60        1998           11
```

```
      Gender                Address  Apartment           City State  \
0     Female          462 Rose Lane        NaN       La Verne    CA
1     Female   3606 Federal Boulevard      NaN    Little Neck    NY
2     Female        766 Third Drive        NaN    West Covina    CA
3     Female       3 Madison Street        NaN       New York    NY
4       Male  9620 Valley Stream Drive     NaN  San Francisco    CA
…        …                      …          …              …     …
1995    Male      6577 Lexington Lane        9.0       Freeport    NY
1996  Female              2 Elm Drive      955.0   Independence    KY
1997  Female        276 Fifth Boulevard      NaN      Elizabeth    NJ
1998    Male      259 Valley Boulevard       NaN      Camp Hill    PA
1999  Female       472 Ocean View Street     NaN      Merrimack    NH

      Zipcode  Latitude  Longitude Per Capita Income - Zipcode  \
0       91750     34.15    -117.76                      $29278
1       11363     40.76     -73.74                      $37891
2       91792     34.02    -117.89                      $22681
3       10069     40.71     -73.99                     $163145
4       94117     37.76    -122.44                      $53797
…         …         …         …                          …
1995    11520     40.65     -73.58                      $23550
1996    41051     38.95     -84.54                      $24218
1997     7201     40.66     -74.19                      $15175
1998    17011     40.24     -76.92                      $25336
1999     3054     42.86     -71.48                      $32325

     Yearly Income - Person Total Debt  FICO Score  Num Credit Cards
0                    $59696    $127613         787                 5
1                    $77254    $191349         701                 5
2                    $33483       $196         698                 5
3                   $249925    $202328         722                 4
4                   $109687    $183855         675                 1
…                        …          …          …                 …
1995                 $48010     $87837         703                 3
1996                 $49378    $104480         740                 4
1997                 $30942     $71066         779                 3
1998                 $54654     $27241         618                 1
1999                 $65909    $181261         673                 2

[2000 rows x 18 columns]
```

[172]:
```
select_user_cols = ['Current Age', 'Gender', 'State', 'Yearly Income - Person',
'FICO Score', 'Num Credit Cards']
```

[173]:
```
user_df.columns
```

```
[173]: Index(['Person', 'Current Age', 'Retirement Age', 'Birth Year', 'Birth Month',
              'Gender', 'Address', 'Apartment', 'City', 'State', 'Zipcode',
              'Latitude', 'Longitude', 'Per Capita Income - Zipcode',
              'Yearly Income - Person', 'Total Debt', 'FICO Score',
              'Num Credit Cards'],
             dtype='object')
```

```
[175]: user_df.loc[:,select_user_cols]
```

```
[175]:       Current Age  Gender State Yearly Income - Person  FICO Score  \
       0              53  Female    CA                 $59696         787
       1              53  Female    NY                 $77254         701
       2              81  Female    CA                 $33483         698
       3              63  Female    NY                $249925         722
       4              43    Male    CA                $109687         675
       …             …      …    …                    …          …
       1995           32    Male    NY                 $48010         703
       1996           62  Female    KY                 $49378         740
       1997           47  Female    NJ                 $30942         779
       1998           66    Male    PA                 $54654         618
       1999           21  Female    NH                 $65909         673

             Num Credit Cards
       0                    5
       1                    5
       2                    5
       3                    4
       4                    1
       …                 …
       1995                 3
       1996                 4
       1997                 3
       1998                 1
       1999                 2

       [2000 rows x 6 columns]
```

```
[178]: filter_user = user_df.copy()
       filter_user = user_df.loc[:, select_user_cols]
```

```
[179]: filter_user
```

```
[179]:       Current Age  Gender State Yearly Income - Person  FICO Score  \
       0              53  Female    CA                 $59696         787
       1              53  Female    NY                 $77254         701
       2              81  Female    CA                 $33483         698
       3              63  Female    NY                $249925         722
```

```
4               43    Male    CA                  $109687         675
…            …     …   …                    …         …
1995            32    Male    NY                   $48010         703
1996            62  Female    KY                   $49378         740
1997            47  Female    NJ                   $30942         779
1998            66    Male    PA                   $54654         618
1999            21  Female    NH                   $65909         673

      Num Credit Cards
0                    5
1                    5
2                    5
3                    4
4                    1
…                 …
1995                 3
1996                 4
1997                 3
1998                 1
1999                 2

[2000 rows x 6 columns]
```