Capital One - Data Science Challenge

Business problem: The prevalence of fraudulent transactions remains a significant challenge for banks, prompting them to seek strategies for reducing the frequency of fraudulent activities across customers' accounts. Utilizing historical transaction data from specific accounts, the objective is to create a supervised machine learning model capable of predicting future transactions and effectively distinguishing between valid and fraudulent ones.

Question 1: Load

In [1]:
```
# Load the necessary libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, ConfusionMatrixDisplay
import scikitplot as skplt

# Load dataset
txn = pd.read_json("transactions.txt", lines=True)
```

In [2]:
```
# Get an overview of the dataset
txn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 786363 entries, 0 to 786362
Data columns (total 29 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   accountNumber             786363 non-null  int64  
 1   customerId                786363 non-null  int64  
 2   creditLimit               786363 non-null  int64  
 3   availableMoney            786363 non-null  float64
 4   transactionDateTime       786363 non-null  object 
 5   transactionAmount         786363 non-null  float64
 6   merchantName              786363 non-null  object 
 7   acqCountry                786363 non-null  object 
 8   merchantCountryCode       786363 non-null  object 
 9   posEntryMode              786363 non-null  object 
 10  posConditionCode          786363 non-null  object 
 11  merchantCategoryCode      786363 non-null  object 
 12  currentExpDate            786363 non-null  object 
 13  accountOpenDate           786363 non-null  object 
 14  dateOfLastAddressChange   786363 non-null  object 
 15  cardCVV                   786363 non-null  int64  
 16  enteredCVV                786363 non-null  int64  
 17  cardLast4Digits           786363 non-null  int64  
 18  transactionType           786363 non-null  object 
 19  echoBuffer                786363 non-null  object 
 20  currentBalance            786363 non-null  float64
 21  merchantCity              786363 non-null  object 
 22  merchantState             786363 non-null  object 
 23  merchantZip               786363 non-null  object 
 24  cardPresent               786363 non-null  bool   
 25  posOnPremises             786363 non-null  object 
 26  recurringAuthInd          786363 non-null  object 
 27  expirationDateKeyInMatch  786363 non-null  bool   
 28  isFraud                   786363 non-null  bool   
dtypes: bool(3), float64(3), int64(6), object(17)
memory usage: 158.2+ MB
```

The dataset has 786,363 records and each record has 29 fields. There are 3 fields with boolean data type, 3 fields with float data type, 6 fields with integer data type and 17 fields that are of type object. At first there are no null values recorded in the dataset but will investigate further. All fields are in the right data type except for the transactionDateTime which is in the object datatype but should be in the datetime data type.

In [3]:
```python
# Get the statistical summary of the numerical features in the dataset
txn.describe()
```

Out[3]:

	accountNumber	customerId	creditLimit	availableMoney	transactionAmount	cardCVV	enteredCVV	cardL
count	7.863630e+05	7.863630e+05	786363.000000	786363.000000	786363.000000	786363.000000	786363.000000	786:
mean	5.372326e+08	5.372326e+08	10759.464459	6250.725369	136.985791	544.467338	544.183857	47
std	2.554211e+08	2.554211e+08	11636.174890	8880.783989	147.725569	261.524220	261.551254	29
min	1.000881e+08	1.000881e+08	250.000000	-1005.630000	0.000000	100.000000	0.000000	
25%	3.301333e+08	3.301333e+08	5000.000000	1077.420000	33.650000	310.000000	310.000000	2
50%	5.074561e+08	5.074561e+08	7500.000000	3184.860000	87.900000	535.000000	535.000000	47
75%	7.676200e+08	7.676200e+08	15000.000000	7500.000000	191.480000	785.000000	785.000000	73
max	9.993896e+08	9.993896e+08	50000.000000	50000.000000	2011.540000	998.000000	998.000000	99

```
Statistical summary of the numerical fields gotten where the count, minimum and maximum values,
mean,
25%, 50%, 75% and standard deviation was displayed.

The minimum value recorded is:
creditLimit is $250
availableMoney is $ -1005.63
transactionAmount is $0
cardCVV is 100
enteredCVV is 0
cardLast4Digits is 0
currentBalance is $0

The maximum value recorded in:
creditLimit is $50,000
availableMoney is $50,000
transactionAmount is $2011.54
cardCVV is 998
enteredCVV is 998
cardLast4Digits is 9998
currentBalance is $47,498.81
```

In [4]:
```python
# Visually inspect the dataset
txn.head()
```

Out[4]:

	accountNumber	customerId	creditLimit	availableMoney	transactionDateTime	transactionAmount	merchantName	acqCoun
0	737265056	737265056	5000	5000.0	2016-08-13T14:27:32	98.55	Uber	
1	737265056	737265056	5000	5000.0	2016-10-11T05:05:54	74.51	AMC #191138	
2	737265056	737265056	5000	5000.0	2016-11-08T09:18:39	7.47	Play Store	
3	737265056	737265056	5000	5000.0	2016-12-10T02:14:50	7.47	Play Store	
4	830329091	830329091	5000	5000.0	2016-03-24T21:04:46	71.18	Tim Hortons #947751	

5 rows × 29 columns

The visual inspection of the dataset shows that some fields such as echoBuffer have no value. It is important to investigate further.

In [5]:
```
# Get number of unique values in each field
for column in txn.columns:
    print(column, ": ", txn[column].nunique())
```

```
accountNumber :  5000
customerId :  5000
creditLimit :  10
availableMoney :  521915
transactionDateTime :  776637
transactionAmount :  66038
merchantName :  2490
acqCountry :  5
merchantCountryCode :  5
posEntryMode :  6
posConditionCode :  4
merchantCategoryCode :  19
currentExpDate :  165
accountOpenDate :  1820
dateOfLastAddressChange :  2184
cardCVV :  899
enteredCVV :  976
cardLast4Digits :  5245
transactionType :  4
echoBuffer :  1
currentBalance :  487318
merchantCity :  1
merchantState :  1
merchantZip :  1
cardPresent :  2
posOnPremises :  1
recurringAuthInd :  1
expirationDateKeyInMatch :  2
isFraud :  2
```

In [6]:
```
# Get the unique values for fields with few unique values
for column in txn.columns:
    if (txn[column].nunique()) < 11:
        print("The unique values in ", column, "is ", txn[column].unique())
    else:
        print("Number of unique values in ", column, "is ", txn[column].nunique())
```

```
Number of unique values in  accountNumber is  5000
Number of unique values in  customerId is  5000
The unique values in  creditLimit is  [ 5000  2500 50000 15000 10000   250   500  1000  7500 20000]
Number of unique values in  availableMoney is  521915
Number of unique values in  transactionDateTime is  776637
Number of unique values in  transactionAmount is  66038
Number of unique values in  merchantName is  2490
The unique values in  acqCountry is  ['US' '' 'CAN' 'MEX' 'PR']
The unique values in  merchantCountryCode is  ['US' 'CAN' '' 'PR' 'MEX']
The unique values in  posEntryMode is  ['02' '09' '05' '80' '90' '']
The unique values in  posConditionCode is  ['01' '08' '99' '']
Number of unique values in  merchantCategoryCode is  19
Number of unique values in  currentExpDate is  165
Number of unique values in  accountOpenDate is  1820
Number of unique values in  dateOfLastAddressChange is  2184
Number of unique values in  cardCVV is  899
Number of unique values in  enteredCVV is  976
Number of unique values in  cardLast4Digits is  5245
The unique values in  transactionType is  ['PURCHASE' 'ADDRESS_VERIFICATION' 'REVERSAL' '']
The unique values in  echoBuffer is  ['']
Number of unique values in  currentBalance is  487318
The unique values in  merchantCity is  ['']
The unique values in  merchantState is  ['']
The unique values in  merchantZip is  ['']
The unique values in  cardPresent is  [False  True]
The unique values in  posOnPremises is  ['']
The unique values in  recurringAuthInd is  ['']
The unique values in  expirationDateKeyInMatch is  [False  True]
The unique values in  isFraud is  [False  True]
```

The target fields of interest echoBuffer, merchantCity, merchantState, merchantZip, posOnPremises and recurringAuthInd shows there is one unique value.

In [7]:
```
# Get a list of field of interest
foi = []
for column in txn.columns:
    if (txn[column].nunique()) == 1:
        foi.append(column)

print(foi)
```

```
['echoBuffer', 'merchantCity', 'merchantState', 'merchantZip', 'posOnPremises', 'recurringAuthInd']
```

In [8]:
```
# Get the unique values in fields of interest
for column in foi:
    print(column," : ", txn[column].unique())
```

```
echoBuffer  :  ['']
merchantCity  :  ['']
merchantState  :  ['']
merchantZip  :  ['']
posOnPremises  :  ['']
recurringAuthInd  :  ['']
```

The fields contains no values but was flagged as non null because the values in them are whitespaces, this shows that the target fields are in fact null. The fields will be deleted since they contain no value.

In [9]:
```
# Drop the empty fields
txn.drop(columns = ["echoBuffer", "merchantCity", "merchantState", "merchantZip", "posOnPremises",
                    "recurringAuthInd"], inplace = True)
```

In [10]:
```
# Check for empty string in other fields
txn.isin([""]).any(axis = 0)
```

Out[10]:
```
accountNumber             False
customerId                False
creditLimit               False
availableMoney            False
transactionDateTime       False
transactionAmount         False
merchantName              False
acqCountry                 True
merchantCountryCode        True
posEntryMode               True
posConditionCode           True
merchantCategoryCode      False
currentExpDate            False
accountOpenDate           False
dateOfLastAddressChange   False
cardCVV                   False
enteredCVV                False
cardLast4Digits           False
transactionType            True
currentBalance            False
cardPresent               False
expirationDateKeyInMatch  False
isFraud                   False
dtype: bool
```

There are 5 fields that have empty string in them, those fields are of object data type hence they define the entry and not primarily for calculation. The empty spaces will be replaced with "notSpecified".

In [11]:
```
# Make a list of the fields that have empty string
col = txn == ""
emptyCol = txn.columns[col.any()]
cols = emptyCol.tolist()

cols
```

Out[11]:
```
['acqCountry',
 'merchantCountryCode',
 'posEntryMode',
 'posConditionCode',
 'transactionType']
```

In [12]:
```
# Check the unique values in the fields
for col in cols:
    print(col, ":", txn[col].unique())
```

```
acqCountry : ['US' '' 'CAN' 'MEX' 'PR']
merchantCountryCode : ['US' 'CAN' '' 'PR' 'MEX']
posEntryMode : ['02' '09' '05' '80' '90' '']
posConditionCode : ['01' '08' '99' '']
transactionType : ['PURCHASE' 'ADDRESS_VERIFICATION' 'REVERSAL' '']
```

In [13]:
```
# Replace empy string with "Not specified"
for col in cols:
    txn[col].replace("", "notSpecified", inplace=True)
```

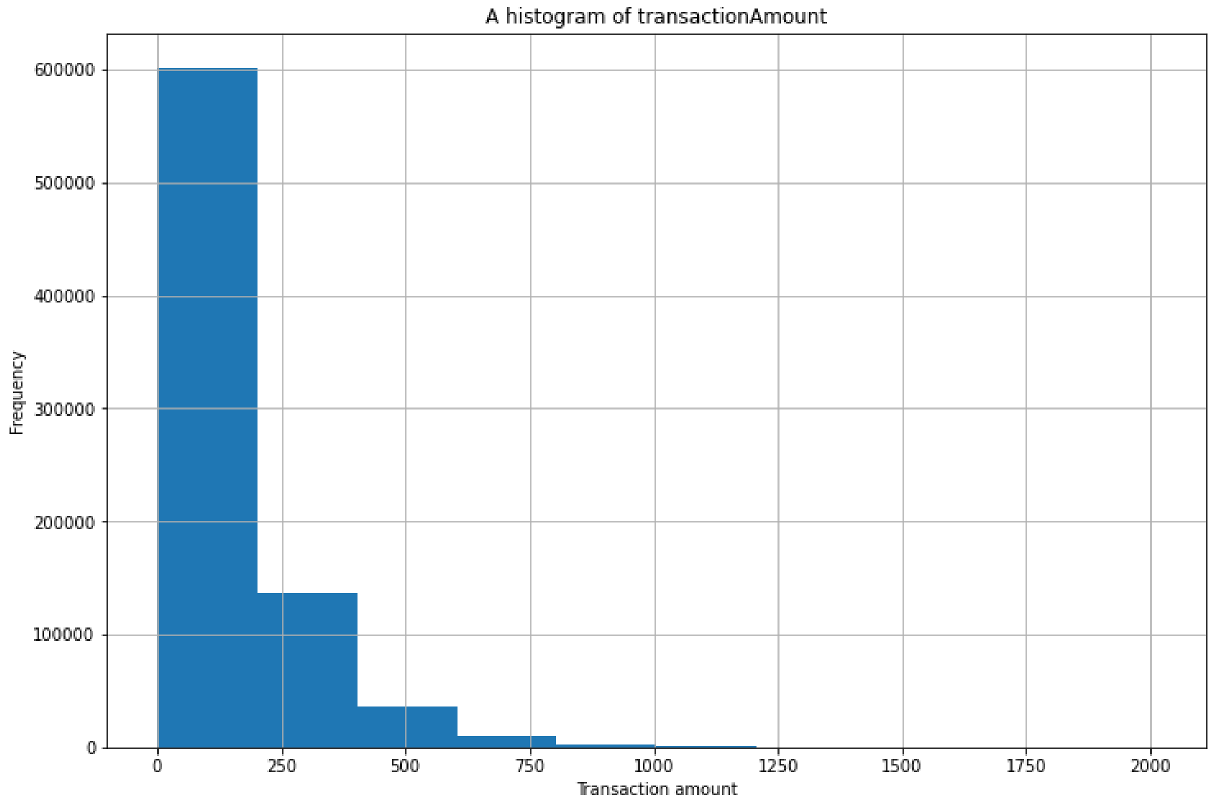
In [14]:
```
txn.isin([""]).any(axis = 0)
```

Out[14]:
```
accountNumber               False
customerId                  False
creditLimit                 False
availableMoney              False
transactionDateTime         False
transactionAmount           False
merchantName                False
acqCountry                  False
merchantCountryCode         False
posEntryMode                False
posConditionCode            False
merchantCategoryCode        False
currentExpDate              False
accountOpenDate             False
dateOfLastAddressChange     False
cardCVV                     False
enteredCVV                  False
cardLast4Digits             False
transactionType             False
currentBalance              False
cardPresent                 False
expirationDateKeyInMatch    False
isFraud                     False
dtype: bool
```

There are no more empty strings in the dataset.The null fields were dropped while the fields with some null values was imputed with notSpecified because the missing values are less than 2% of the dataset.

Question 2: Plot

In [15]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
txn["transactionAmount"].hist(grid=True)
plt.title("A histogram of transactionAmount")
plt.xlabel("Transaction amount")
plt.ylabel("Frequency");
```




```
The distribution is right skewed and shows that majority of the transaction amount is between $0
and $250
```

Question 3: Data Wrangling - Duplicate Transactions

In [16]:
```
#  Check for duplicated entry
txn[txn.duplicated()].sum().sum()
```

Out[16]: 0.0

There is no duplicated entry in the dataset

In [17]:
```
# Check the types of transaction in the dataset
txn["transactionType"].unique()
```

Out[17]:
```
array(['PURCHASE', 'ADDRESS_VERIFICATION', 'REVERSAL', 'notSpecified'],
      dtype=object)
```

In [18]:
```
# Get the sum of each transactions transaction type
for transaction in txn["transactionType"].unique():
    charge = txn[txn["transactionType"] == transaction].transactionAmount.sum()
    print("Sum of transactionAmount for", transaction, " : $", charge)
```

```
Sum of transactionAmount for PURCHASE  : $ 104790305.25
Sum of transactionAmount for ADDRESS_VERIFICATION  : $ 0.0
Sum of transactionAmount for REVERSAL  : $ 2821792.5
Sum of transactionAmount for notSpecified  : $ 108459.78
```

The types of transaction recorded in this dataset are:

1. Purchase: This is a purchase made on a specific account, this type of transaction has an actual transaction amount associated to it.
2. ADDRESS_VERIFICATION: This is a type of transaction that is used to verify an account's residential address. There is no actual charge on this type of transaction.
3. REVERSAL: This is a type of transaction that is considered unsuccessful and the transaction amount is refunded to the account.
4. Not_specified: This type of transaction is not labelled although there are charges on the transaction.

The dataset contains multi-swipe transactions where the customer is accidentally charged more than once, and also reversed transaction the transaction is considered unsuccessful and the transaction amount is refunded to the account. These transactions are considered duplicate transactions.

To accurately filter duplicate transaction, fields that accurately describe the transaction will be included, such fields are:

1. accountNumber
2. dateOfTransaction
3. hourOfTransaction
4. transactionAmount
5. merchantName

Some of the fields mentioned above are not in the dataset hence will be created.

Create dateOfTransaction and hourOfTransaction fields

In [19]:
```
# Convert transactionDateTime field from object to datetime
txn["transactionDateTime"] = pd.to_datetime(txn["transactionDateTime"])
```

In [20]:
```
# Create dateOfTransaction field
txn["dateOfTransaction"] = txn["transactionDateTime"].dt.date
```

In [21]:
```
# Create hourOfTransaction field
txn["hourOfTransaction"] = txn["transactionDateTime"].dt.hour
```

In [22]:
```
# Get the number of Reversed transaction
rev = txn[txn["transactionType"] == "REVERSAL"]
len(rev)
```

Out[22]:
```
20303
```

In [23]:
```
# Get the total transaction amount for reversed transaction
rev["transactionAmount"].sum()
```

Out[23]:
```
2821792.5
```

The number of reversed transactions in the dataset is 20,303 which accounts for $2,821,792.5 in total transaction amount.

In [24]:
```python
# Rearrange the fields
txn = txn[['accountNumber', 'customerId', 'creditLimit', 'availableMoney','transactionDateTime', 'date
       'hourOfTransaction', 'transactionAmount', 'merchantName', 'acqCountry', 'merchantCountryCode',
       'posConditionCode', 'merchantCategoryCode', 'currentExpDate', 'accountOpenDate', 'dateOfLastAdd
       'enteredCVV','cardLast4Digits','transactionType', 'currentBalance', 'cardPresent', 'expirationD

txn.head()
```

Out[24]:

	accountNumber	customerId	creditLimit	availableMoney	transactionDateTime	dateOfTransaction	hourOfTransaction	transa
0	737265056	737265056	5000	5000.0	2016-08-13 14:27:32	2016-08-13	14	
1	737265056	737265056	5000	5000.0	2016-10-11 05:05:54	2016-10-11	5	
2	737265056	737265056	5000	5000.0	2016-11-08 09:18:39	2016-11-08	9	
3	737265056	737265056	5000	5000.0	2016-12-10 02:14:50	2016-12-10	2	
4	830329091	830329091	5000	5000.0	2016-03-24 21:04:46	2016-03-24	21	

5 rows × 25 columns

In [25]:
```python
# filter duplicate transaction in the dataset
dup = txn[txn.duplicated(subset = ["accountNumber", "dateOfTransaction", "hourOfTransaction", "transa
                                   "merchantName"], keep = "first")]

duplicate = dup[dup["transactionType"] != "REVERSAL"]
```

In [26]:
```python
duplicate["transactionType"].unique()
```

Out[26]: `array(['PURCHASE', 'ADDRESS_VERIFICATION', 'notSpecified'], dtype=object)`

In [27]:
```python
# Get the number of duplicate transactions
len(duplicate)
```

Out[27]: 7485

In [28]:
```python
# Get the sum of transactions for duplicate transactions
duplicate["transactionAmount"].sum()
```

Out[28]: 1076660.25

The number of duplicate transactions in the dataset is 7485 which accounts for $1,076,660.25 in total transaction amount.

```
Question 4: Model
```

Building a predictive model for the dataset, some fields needs to be modified so that it can be processed by the model.

In [29]:
```
# Frequency encoding for fields that are of object data type with many unique values


merch_cat = txn["merchantName"].value_counts(normalize = True)
txn["merchantNameCat"] = txn["merchantName"].map(merch_cat.to_dict())

dOTran_cat = txn["dateOfTransaction"].value_counts(normalize = True)
txn["dateOfTransactionCat"] = txn["dateOfTransaction"].map(dOTran_cat.to_dict())

merch_categ = txn["merchantCategoryCode"].value_counts(normalize = True)
txn["merchantCategoryCodeCat"] = txn["merchantCategoryCode"].map(merch_categ.to_dict())

hour_Cat = txn["hourOfTransaction"].value_counts(normalize = True)
txn["hourOfTransactionCat"] = txn["hourOfTransaction"].map(hour_Cat.to_dict())

cardCVV_cat = txn["cardCVV"].value_counts(normalize = True)
txn["cardCVVCat"] = txn["cardCVV"].map(cardCVV_cat.to_dict())

enteredCVV_cat = txn["enteredCVV"].value_counts(normalize = True)
txn["enteredCVVCat"] = txn["enteredCVV"].map(enteredCVV_cat.to_dict())

pEM_cat = txn["posEntryMode"].value_counts(normalize = True)
txn["posEntryModeCat"] = txn["posEntryMode"].map(pEM_cat.to_dict())

pCC_cat = txn["posConditionCode"].value_counts(normalize = True)
txn["posConditionCodeCat"] = txn["posConditionCode"].map(pCC_cat.to_dict())

# txnType_cat = txn["transactionType"].value_counts(normalize = True)
# txn["transactionTypeCat"] = txn["transactionType"].map(txnType_cat.to_dict())
```

In [30]:
```
txn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 786363 entries, 0 to 786362
Data columns (total 33 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   accountNumber             786363 non-null  int64
 1   customerId                786363 non-null  int64
 2   creditLimit               786363 non-null  int64
 3   availableMoney            786363 non-null  float64
 4   transactionDateTime       786363 non-null  datetime64[ns]
 5   dateOfTransaction         786363 non-null  object
 6   hourOfTransaction         786363 non-null  int64
 7   transactionAmount         786363 non-null  float64
 8   merchantName              786363 non-null  object
 9   acqCountry                786363 non-null  object
 10  merchantCountryCode       786363 non-null  object
 11  posEntryMode              786363 non-null  object
 12  posConditionCode          786363 non-null  object
 13  merchantCategoryCode      786363 non-null  object
 14  currentExpDate            786363 non-null  object
 15  accountOpenDate           786363 non-null  object
 16  dateOfLastAddressChange   786363 non-null  object
 17  cardCVV                   786363 non-null  int64
 18  enteredCVV                786363 non-null  int64
 19  cardLast4Digits           786363 non-null  int64
 20  transactionType           786363 non-null  object
 21  currentBalance            786363 non-null  float64
 22  cardPresent               786363 non-null  bool
 23  expirationDateKeyInMatch  786363 non-null  bool
 24  isFraud                   786363 non-null  bool
 25  merchantNameCat           786363 non-null  float64
 26  dateOfTransactionCat      786363 non-null  float64
 27  merchantCategoryCodeCat   786363 non-null  float64
 28  hourOfTransactionCat      786363 non-null  float64
 29  cardCVVCat                786363 non-null  float64
 30  enteredCVVCat             786363 non-null  float64
 31  posEntryModeCat           786363 non-null  float64
 32  posConditionCodeCat       786363 non-null  float64
dtypes: bool(3), datetime64[ns](1), float64(11), int64(7), object(11)
memory usage: 182.2+ MB
```

In [31]:
```
# Create a dummy variable for fields that are of object data type with few unique values. One of each
# be dropped to avoid dummy variable trap
txn = pd.get_dummies(txn, columns = ["cardPresent", "expirationDateKeyInMatch", "merchantCountryCode",
                                     "transactionType"])
```

In [32]:
```
txn.drop(columns = ["merchantName", "dateOfTransaction", "merchantCategoryCode", "hourOfTransaction",
                   "merchantCountryCode_CAN", "acqCountry_CAN", "expirationDateKeyInMatch_True", "car
                    "transactionType_ADDRESS_VERIFICATION", "posConditionCode", "posEntryMode"], inpl
```

In [33]:
```
txn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 786363 entries, 0 to 786362
Data columns (total 33 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   accountNumber                     786363 non-null  int64
 1   customerId                        786363 non-null  int64
 2   creditLimit                       786363 non-null  int64
 3   availableMoney                    786363 non-null  float64
 4   transactionDateTime               786363 non-null  datetime64[ns]
 5   transactionAmount                 786363 non-null  float64
 6   currentExpDate                    786363 non-null  object
 7   accountOpenDate                   786363 non-null  object
 8   dateOfLastAddressChange           786363 non-null  object
 9   cardLast4Digits                   786363 non-null  int64
 10  currentBalance                    786363 non-null  float64
 11  isFraud                           786363 non-null  bool
 12  merchantNameCat                   786363 non-null  float64
 13  dateOfTransactionCat              786363 non-null  float64
 14  merchantCategoryCodeCat           786363 non-null  float64
 15  hourOfTransactionCat              786363 non-null  float64
 16  cardCVVCat                        786363 non-null  float64
 17  enteredCVVCat                     786363 non-null  float64
 18  posEntryModeCat                   786363 non-null  float64
 19  posConditionCodeCat               786363 non-null  float64
 20  cardPresent_False                 786363 non-null  uint8
 21  expirationDateKeyInMatch_False    786363 non-null  uint8
 22  merchantCountryCode_MEX           786363 non-null  uint8
 23  merchantCountryCode_PR            786363 non-null  uint8
 24  merchantCountryCode_US            786363 non-null  uint8
 25  merchantCountryCode_notSpecified  786363 non-null  uint8
 26  acqCountry_MEX                    786363 non-null  uint8
 27  acqCountry_PR                     786363 non-null  uint8
 28  acqCountry_US                     786363 non-null  uint8
 29  acqCountry_notSpecified           786363 non-null  uint8
 30  transactionType_PURCHASE          786363 non-null  uint8
 31  transactionType_REVERSAL          786363 non-null  uint8
 32  transactionType_notSpecified      786363 non-null  uint8
dtypes: bool(1), datetime64[ns](1), float64(11), int64(4), object(3), uint8(13)
memory usage: 124.5+ MB
```

In [34]:
```
# Plot a correlation plot of fields in the dataset
import seaborn as sns
plt.figure(figsize = (15, 10))
sns.heatmap(txn.corr(), cmap= 'BrBG');
```

There are fields that are correlated with each other, fields such as customerid & accountNumber, availableMoney & creditLimit, cardCVVCat and enteredCVVCat, acqCountry & merchantCountryCode. One field among each pair will be dropped to avoid multicollinearity.

In [35]:
```
txn.corr()
```

Out[35]:

	accountNumber	customerId	creditLimit	availableMoney	transactionAmount	cardLast4Digi
accountNumber	1.000000	1.000000	0.140673	0.066345	-0.001364	0.0385
customerId	1.000000	1.000000	0.140673	0.066345	-0.001364	0.0385
creditLimit	0.140673	0.140673	1.000000	0.834977	0.005581	0.1256
availableMoney	0.066345	0.066345	0.834977	1.000000	-0.010070	0.0738
transactionAmount	-0.001364	-0.001364	0.005581	-0.010070	1.000000	-0.0015
cardLast4Digits	0.038517	0.038517	0.125611	0.073879	-0.001513	1.0000
currentBalance	0.162248	0.162248	0.653652	0.129332	0.023905	0.1247
isFraud	-0.004011	-0.004011	0.003108	-0.001538	0.075651	0.0008
merchantNameCat	-0.008245	-0.008245	-0.028009	-0.023868	0.031180	-0.0120
dateOfTransactionCat	0.000296	0.000296	-0.000678	-0.039622	-0.027010	0.0022
merchantCategoryCodeCat	-0.053177	-0.053177	-0.032427	-0.032999	0.068485	0.0239
hourOfTransactionCat	-0.001285	-0.001285	-0.000813	-0.002531	0.000091	0.0026
cardCVVCat	-0.111101	-0.111101	-0.031592	-0.090810	0.020032	-0.2771
enteredCVVCat	-0.110185	-0.110185	-0.031131	-0.090154	0.019848	-0.2755
posEntryModeCat	-0.000777	-0.000777	-0.001453	-0.002393	-0.001881	-0.0002
posConditionCodeCat	0.000218	0.000218	-0.000284	-0.000131	-0.001931	-0.0010
cardPresent_False	-0.006099	-0.006099	0.003929	-0.002508	-0.037361	-0.0407
expirationDateKeyInMatch_False	0.001057	0.001057	-0.002391	-0.004067	-0.001751	-0.0020
merchantCountryCode_MEX	-0.001602	-0.001602	-0.002087	-0.002543	0.000596	0.0009
merchantCountryCode_PR	-0.003287	-0.003287	-0.001143	-0.002102	-0.000850	0.0009
merchantCountryCode_US	0.004269	0.004269	0.001936	0.002362	-0.000277	0.0003
merchantCountryCode_notSpecified	-0.002711	-0.002711	-0.000926	-0.001104	0.000339	0.0012
acqCountry_MEX	-0.001601	-0.001601	-0.001773	-0.002024	0.000378	0.0006
acqCountry_PR	-0.003164	-0.003164	-0.001038	-0.001865	-0.000654	0.0008
acqCountry_US	0.003120	0.003120	0.000803	0.000995	-0.000462	-0.0006
acqCountry_notSpecified	-0.000637	-0.000637	0.000548	0.000483	0.000565	0.0022
transactionType_PURCHASE	0.002780	0.002780	-0.001177	0.004615	0.104713	0.0008
transactionType_REVERSAL	-0.000868	-0.000868	0.001827	-0.003377	0.002202	-0.0006
transactionType_notSpecified	-0.001576	-0.001576	0.000220	0.000119	0.003713	0.0004

29 rows × 29 columns

In [36]:
```
X = txn.drop(columns = ["accountNumber", "customerId", "creditLimit", "transactionDateTime","cardCVVCa
                        "accountOpenDate","dateOfLastAddressChange", "acqCountry_MEX", "acqCountry_PR'
                        "acqCountry_notSpecified", "isFraud"])
```

In [37]:
```
txn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 786363 entries, 0 to 786362
Data columns (total 33 columns):
 #   Column                           Non-Null Count   Dtype
---  ------                           --------------   -----
 0   accountNumber                    786363 non-null  int64
 1   customerId                       786363 non-null  int64
 2   creditLimit                      786363 non-null  int64
 3   availableMoney                   786363 non-null  float64
 4   transactionDateTime              786363 non-null  datetime64[ns]
 5   transactionAmount                786363 non-null  float64
 6   currentExpDate                   786363 non-null  object
 7   accountOpenDate                  786363 non-null  object
 8   dateOfLastAddressChange          786363 non-null  object
 9   cardLast4Digits                  786363 non-null  int64
 10  currentBalance                   786363 non-null  float64
 11  isFraud                          786363 non-null  bool
 12  merchantNameCat                  786363 non-null  float64
 13  dateOfTransactionCat             786363 non-null  float64
 14  merchantCategoryCodeCat          786363 non-null  float64
 15  hourOfTransactionCat             786363 non-null  float64
 16  cardCVVCat                       786363 non-null  float64
 17  enteredCVVCat                    786363 non-null  float64
 18  posEntryModeCat                  786363 non-null  float64
 19  posConditionCodeCat              786363 non-null  float64
 20  cardPresent_False                786363 non-null  uint8
 21  expirationDateKeyInMatch_False   786363 non-null  uint8
 22  merchantCountryCode_MEX          786363 non-null  uint8
 23  merchantCountryCode_PR           786363 non-null  uint8
 24  merchantCountryCode_US           786363 non-null  uint8
 25  merchantCountryCode_notSpecified 786363 non-null  uint8
 26  acqCountry_MEX                   786363 non-null  uint8
 27  acqCountry_PR                    786363 non-null  uint8
 28  acqCountry_US                    786363 non-null  uint8
 29  acqCountry_notSpecified          786363 non-null  uint8
 30  transactionType_PURCHASE         786363 non-null  uint8
 31  transactionType_REVERSAL         786363 non-null  uint8
 32  transactionType_notSpecified     786363 non-null  uint8
dtypes: bool(1), datetime64[ns](1), float64(11), int64(4), object(3), uint8(13)
memory usage: 124.5+ MB
```

In [38]:
```
y = txn["isFraud"]
```

In [39]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, random_state = 42)
```

In [40]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 629090 entries, 257074 to 121958
Data columns (total 20 columns):
 #   Column                           Non-Null Count   Dtype  
---  ------                           --------------   -----  
 0   availableMoney                   629090 non-null  float64
 1   transactionAmount                629090 non-null  float64
 2   cardLast4Digits                  629090 non-null  int64  
 3   currentBalance                   629090 non-null  float64
 4   merchantNameCat                  629090 non-null  float64
 5   dateOfTransactionCat             629090 non-null  float64
 6   merchantCategoryCodeCat          629090 non-null  float64
 7   hourOfTransactionCat             629090 non-null  float64
 8   enteredCVVCat                    629090 non-null  float64
 9   posEntryModeCat                  629090 non-null  float64
 10  posConditionCodeCat              629090 non-null  float64
 11  cardPresent_False                629090 non-null  uint8  
 12  expirationDateKeyInMatch_False   629090 non-null  uint8  
 13  merchantCountryCode_MEX          629090 non-null  uint8  
 14  merchantCountryCode_PR           629090 non-null  uint8  
 15  merchantCountryCode_US           629090 non-null  uint8  
 16  merchantCountryCode_notSpecified 629090 non-null  uint8  
 17  transactionType_PURCHASE         629090 non-null  uint8  
 18  transactionType_REVERSAL         629090 non-null  uint8  
 19  transactionType_notSpecified     629090 non-null  uint8  
dtypes: float64(10), int64(1), uint8(9)
memory usage: 63.0 MB
```

In [41]:
```
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
```

Out[41]:
```
DecisionTreeClassifier()
```

In [42]:
```
y_pred = model.predict(X_test)
```

In [43]:
```
# Evaluate the overall performance of the model
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

Out[43]:
```
0.9687867593293191
```

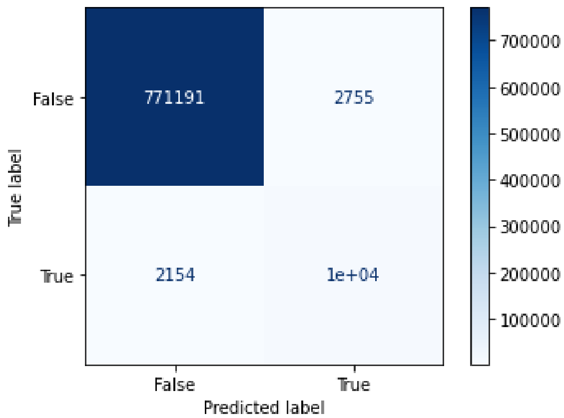
In [44]:
```
# Evaluate how well the model did in predicting genuine transactions
recall = recall_score(y_test, y_pred,pos_label=False)
recall
```

Out[44]:
```
0.982200542705776
```

The fields used for the model were selected using correlation plot – ensuring that there is no correlation between them. The predictor variable for the model were 20 after ensuring that the fields with object data types were converted to categorical data type (frequency encoding or one-hot encoding). The predictor variable used for the model were 20 while the predicted variable was the isFraud field.

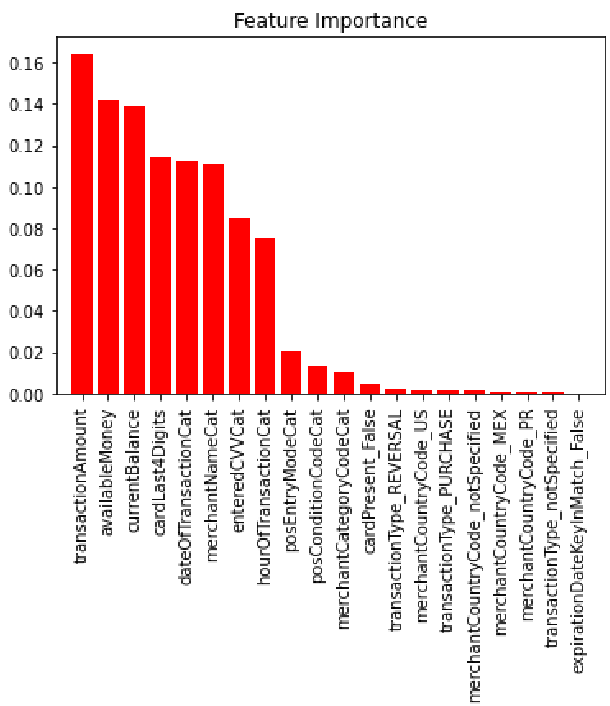
The model was built with Sci-kit learn’s DecisionTree algorithm because the target outcome is a binary outcome in object data type. The 20% testing set was predicted and the overall accuracy 97% while the recall rate was 98% which indicated that the model performed well.

In [45]:
```
# Plot the confusion matrix of the model
ConfusionMatrixDisplay.from_estimator(model, X, y, cmap = "Blues");
```




```
The confusion matrix for the model to see the model’s performance.
```

In [46]:
```
# Visualize the important features used by the model
skplt.estimators.plot_feature_importances(model, feature_names=list(X.columns),x_tick_rotation=90);
```




```
The variable importance plot for the model shows that transactionAmount, availableMoney and
currentBalance are the most important fields in predicting whether a transaction is fraudulent or
not.
```

```
More time:
If I had more time, I would have tuned the model using different parameters and I will also build
models using Random Forest, Logistic Regression and Linear Regression to see how the model will
improve. I would have met the domain expert in fraud detection to understand more about the fields
in the dataset.

Attempted methods:
I attempted to use One-hot-encoding for all categorical variables but it was not ideal because it
made the fields in the dataset become massive. I attempted to get duplicate transactions using
transactionAmount only but the result was undesired
```

because there are multiple normal transactions with the same transactionAmount.