In this problem we would data prep, data clean and perform EDA using python pandas and perform prediction using R.

In [1]:
```
#Load libraries
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

It is impractical to perform feature engineering on the 394 individual column variables of the 'train_transaction.csv' dataset. However, 'train_identity.csv' contains 41 column variables and with the removal of columns with significant missing values, we end up with a 'train_identity' dataframe with 31 columns which can be individually analyzed. In the 'train_transaction' dataframe, we treat character or object data types as factors

In [2]:
```
train_identity = pd.read_csv("../BANA 780/Problem 4/train_identity.csv")
train_transaction = pd.read_csv("../BANA 780/Problem 4/train_transaction.csv")
```

In [ ]:
```
#train_identity.info()
```

In [ ]:
```
#train_transaction.info()
```

In [3]:
```
train_transaction['isFraud'].value_counts()
```

Out[3]:
```
0    569877
1     20663
Name: isFraud, dtype: int64
```

About 3.6% of the transactions in the train_transaction dataset are fraudulent.

In [4]:
```
train_dataset = pd.merge(train_identity, train_transaction, on='TransactionID'
, how='inner')
```

In [5]:
```
train_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144233 entries, 0 to 144232
Columns: 434 entries, TransactionID to V339
dtypes: float64(399), int64(4), object(31)
memory usage: 478.7+ MB
```

There are 144233 rows in the merged dataframe. Using the info function output of the train_identity dataframe, we start by dropping columns with less than 100,000 row entries as a first step. This stil represents less than half of the total rows in the dataframe.

In [6]:
```
train_dataset = train_dataset[[column for column in train_dataset if train_dat
aset[column].count() / len(train_dataset) >= (100000/144233)]]
```

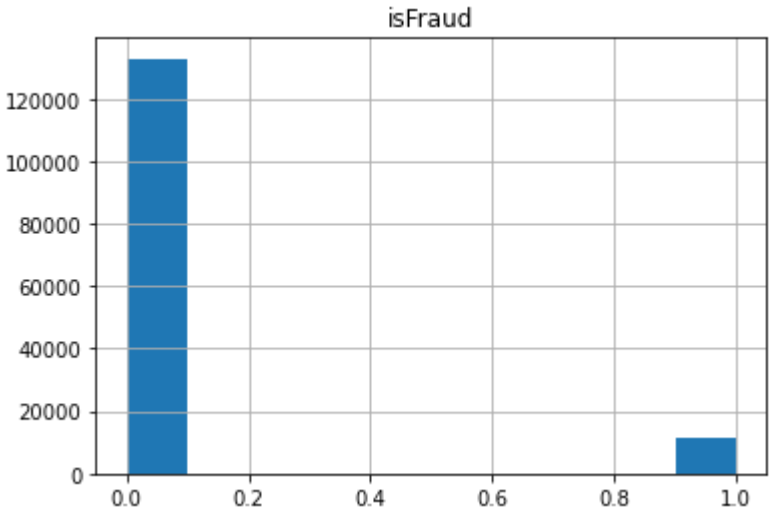
In [7]:
```
train_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144233 entries, 0 to 144232
Columns: 247 entries, TransactionID to V321
dtypes: float64(226), int64(4), object(17)
memory usage: 272.9+ MB
```

Include plots for EDA

In [8]:
```
train_dataset.hist('isFraud')
```

Out[8]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001776585AFA0
>]],
      dtype=object)
```



In [9]:
```
train_dataset['isFraud'].value_counts()
```

Out[9]:
```
0    132915
1     11318
Name: isFraud, dtype: int64
```

In [ ]:
```
#train_dataset_copy = train_dataset.copy()
```

In [ ]:
```
#train_dataset_copy.select_dtypes(include = ['object'])
#train_dataset_copy.select_dtypes(include = ['int64'])
```

Next we drop missing values for the categorical variables identified as the object data type (dtype). The intention is that they serve as factor levels for our model.

In [ ]:
```
#train_dataset_copy = train_dataset_copy.dropna(subset=['id_12', 'id_15', 'id_16', 'id_28', 'id_29', 'id_31', 'id_35', 'id_36', 'id_37', 'id_38', 'DeviceType', 'DeviceInfo', 'ProductCD', 'card4', 'card6', 'P_emaildomain', 'R_emaildomain'])
```

In [ ]:
```
#train_dataset_copy.info()
```

In [10]:
```
another_train = train_dataset.copy()
another_train = another_train.dropna()
another_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 75988 entries, 2 to 144230
Columns: 247 entries, TransactionID to V321
dtypes: float64(226), int64(4), object(17)
memory usage: 143.8+ MB
```

In [32]:
```
#another_train['id_01'].value_counts()
```

In [ ]:
```
#another_train['id_20'].value_counts().head(10) #id_01, id_02, id_19, id_20, id_31, card1, card2, card5
```

In [11]:
```
another_train['id_01'] = np.where(another_train.id_01 == -5.0, 1, 0)
####another_train['id_01'] = np.where(another_train.id_01 == -5.0, -5, 0)
another_train['id_05'] = np.where(another_train.id_05 == 0.0, 1, 0)
another_train['id_06'] = np.where(another_train.id_06 == 0.0, 1, 0)
#####another_train['id_11'] = np.where(another_train.id_11 == 100.000000, 1,
 0)
another_train['id_13'] = another_train['id_13'].apply(lambda x: 2 if x == 52.0
else 1 if x == 49.0 else 0)
another_train['id_17'] = another_train['id_17'].apply(lambda x: 2 if x == 166.
0 else 1 if x == 225.0 else 0)
#####another_train['id_19'] = np.where(another_train.id_19 == 266.0, 0, 1)
another_train['id_20'] = np.where(another_train.id_20 == 507.0, 1, 0)
another_train['id_31'] = np.where(another_train.id_31 == "chrome 63.0", 1, 0)
another_train['card3'] = another_train['card3'].apply(lambda x: 2 if x == 150.
0 else 1 if x == 185.0 else 0)
another_train['card5'] = another_train['card5'].apply(lambda x: 2 if x == 226.
0 else 1 if x == 224.0 else 0)
another_train['DeviceInfo'] = another_train['DeviceInfo'].apply(lambda x: 2 if
x == "Windows" else 1 if x == "iOS Device" else 0)
another_train['P_emaildomain'] = another_train['P_emaildomain'].apply(lambda x
: 3 if x == "gmail.com" else 2 if x == "hotmail.com" else 1 if x == "anonymou
s.com" else 0)
another_train['R_emaildomain'] = another_train['R_emaildomain'].apply(lambda x
: 3 if x == "gmail.com" else 2 if x == "anonymous.com" else 1 if x == "hotmai
l.com" else 0)

another_train['C1'] = another_train['C1'].apply(lambda x: 1 if x == 1.0 else 2
if x == 2.0 else 0)
another_train['C2'] = np.where(another_train.C2 == 1.0, 1, 0)
another_train['C4'] = np.where(another_train.C4 == 1.0, 1, 0)
another_train['C6'] = np.where(another_train.C6 == 1.0, 1, 0)
another_train['C7'] = another_train['C7'].apply(lambda x: 1 if x == 0.0 else 2
if x == 1.0 else 0)
another_train['C8'] = np.where(another_train.C8 == 1.0, 1, 0)
another_train['C10'] = np.where(another_train.C10 == 1.0, 1, 0)
another_train['C11'] = np.where(another_train.C11 == 1.0, 1, 0)
another_train['C12'] = another_train['C12'].apply(lambda x: 1 if x == 0.0 else
2 if x == 1.0 else 0)
another_train['C13'] = another_train['C13'].apply(lambda x: 2 if x == 1.0 else
1 if x == 0.0 else 0)
another_train['C14'] = another_train['C14'].apply(lambda x: 2 if x == 1.0 else
1 if x == 0.0 else 0)
another_train['D1'] = np.where(another_train.D1 == 0.0, 1, 0)
```

We manually create two to three level factors for some of the variables using variable values that are significant. The rest of the values in most cases are grouped together as factor level 0.

In [26]:
```
another_train.to_csv('../BANA 780/Problem 4/_train_data.csv', index=False)
```

In [12]:
```
test_identity = pd.read_csv("../BANA 780/Problem 4/test_identity_copy.csv")
test_transaction = pd.read_csv("../BANA 780/Problem 4/test_transaction.csv")
```

In [13]:
```
test_dataset = pd.merge(test_identity, test_transaction, on='TransactionID', how='inner')
```

In [14]:
```
test_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 141907 entries, 0 to 141906
Columns: 433 entries, TransactionID to V339
dtypes: float64(398), int64(4), object(31)
memory usage: 469.9+ MB
```

In [15]:
```
drop_fraud = another_train.copy()
drop_fraud = drop_fraud.drop(['isFraud'], axis = 1)
drop_fraud.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 75988 entries, 2 to 144230
Columns: 246 entries, TransactionID to V321
dtypes: float64(206), int32(12), int64(15), object(13)
memory usage: 139.7+ MB
```

In [16]:
```
test_dataset_copy = test_dataset.copy()
```

In [17]:
```
test_dataset_copy = test_dataset_copy[list(drop_fraud.columns)]
```

In [18]:
```
test_dataset_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 141907 entries, 0 to 141906
Columns: 246 entries, TransactionID to V321
dtypes: float64(225), int64(4), object(17)
memory usage: 267.4+ MB
```

In [19]:
```
test_dataset_final = test_dataset_copy.copy()
test_dataset_final = test_dataset_final.dropna()
test_dataset_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 75043 entries, 5 to 141906
Columns: 246 entries, TransactionID to V321
dtypes: float64(225), int64(4), object(17)
memory usage: 141.4+ MB
```

In [20]:
```
test_dataset_final['id_01'] = np.where(test_dataset_final.id_01 == -5.0, 1, 0)
####another_train['id_01'] = np.where(another_train.id_01 == -5.0, -5, 0)
test_dataset_final['id_05'] = np.where(test_dataset_final.id_05 == 0.0, 1, 0)
test_dataset_final['id_06'] = np.where(test_dataset_final.id_06 == 0.0, 1, 0)
#####another_train['id_11'] = np.where(another_train.id_11 == 100.000000, 1,
 0)
test_dataset_final['id_13'] = test_dataset_final['id_13'].apply(lambda x: 2 if
x == 52.0 else 1 if x == 49.0 else 0)
test_dataset_final['id_17'] = test_dataset_final['id_17'].apply(lambda x: 2 if
x == 166.0 else 1 if x == 225.0 else 0)
#####another_train['id_19'] = np.where(another_train.id_19 == 266.0, 0, 1)
test_dataset_final['id_20'] = np.where(test_dataset_final.id_20 == 507.0, 1, 0
)
test_dataset_final['id_31'] = np.where(test_dataset_final.id_31 == "chrome 63.
0", 1, 0)
test_dataset_final['card3'] = test_dataset_final['card3'].apply(lambda x: 2 if
x == 150.0 else 1 if x == 185.0 else 0)
test_dataset_final['card5'] = test_dataset_final['card5'].apply(lambda x: 2 if
x == 226.0 else 1 if x == 224.0 else 0)
test_dataset_final['DeviceInfo'] = test_dataset_final['DeviceInfo'].apply(lamb
da x: 2 if x == "Windows" else 1 if x == "iOS Device" else 0)
test_dataset_final['P_emaildomain'] = test_dataset_final['P_emaildomain'].appl
y(lambda x: 3 if x == "gmail.com" else 2 if x == "hotmail.com" else 1 if x ==
"anonymous.com" else 0)
test_dataset_final['R_emaildomain'] = test_dataset_final['R_emaildomain'].appl
y(lambda x: 3 if x == "gmail.com" else 2 if x == "anonymous.com" else 1 if x =
= "hotmail.com" else 0)

test_dataset_final['C1'] = test_dataset_final['C1'].apply(lambda x: 1 if x ==
1.0 else 2 if x == 2.0 else 0)
test_dataset_final['C2'] = np.where(test_dataset_final.C2 == 1.0, 1, 0)
test_dataset_final['C4'] = np.where(test_dataset_final.C4 == 1.0, 1, 0)
test_dataset_final['C6'] = np.where(test_dataset_final.C6 == 1.0, 1, 0)
test_dataset_final['C7'] = test_dataset_final['C7'].apply(lambda x: 1 if x ==
0.0 else 2 if x == 1.0 else 0)
test_dataset_final['C8'] = np.where(test_dataset_final.C8 == 1.0, 1, 0)
test_dataset_final['C10'] = np.where(test_dataset_final.C10 == 1.0, 1, 0)
test_dataset_final['C11'] = np.where(test_dataset_final.C11 == 1.0, 1, 0)
test_dataset_final['C12'] = test_dataset_final['C12'].apply(lambda x: 1 if x =
= 0.0 else 2 if x == 1.0 else 0)
test_dataset_final['C13'] = test_dataset_final['C13'].apply(lambda x: 2 if x =
= 1.0 else 1 if x == 0.0 else 0)
test_dataset_final['C14'] = test_dataset_final['C14'].apply(lambda x: 2 if x =
= 1.0 else 1 if x == 0.0 else 0)
test_dataset_final['D1'] = np.where(test_dataset_final.D1 == 0.0, 1, 0)
```

In [28]:
```
test_dataset_final.to_csv('../BANA 780/Problem 4/check_testdata.csv', index=Fa
lse)
```