CSE519 HW2 Template

September 26, 2019

1 Homework 2 - IEEE Fraud Detection

For all parts below, answer all parts as shown in the Google document for Homework 2. Be sure to include both code that justifies your answer as well as text to answer the questions. We also ask that code be commented to make it easier to follow.

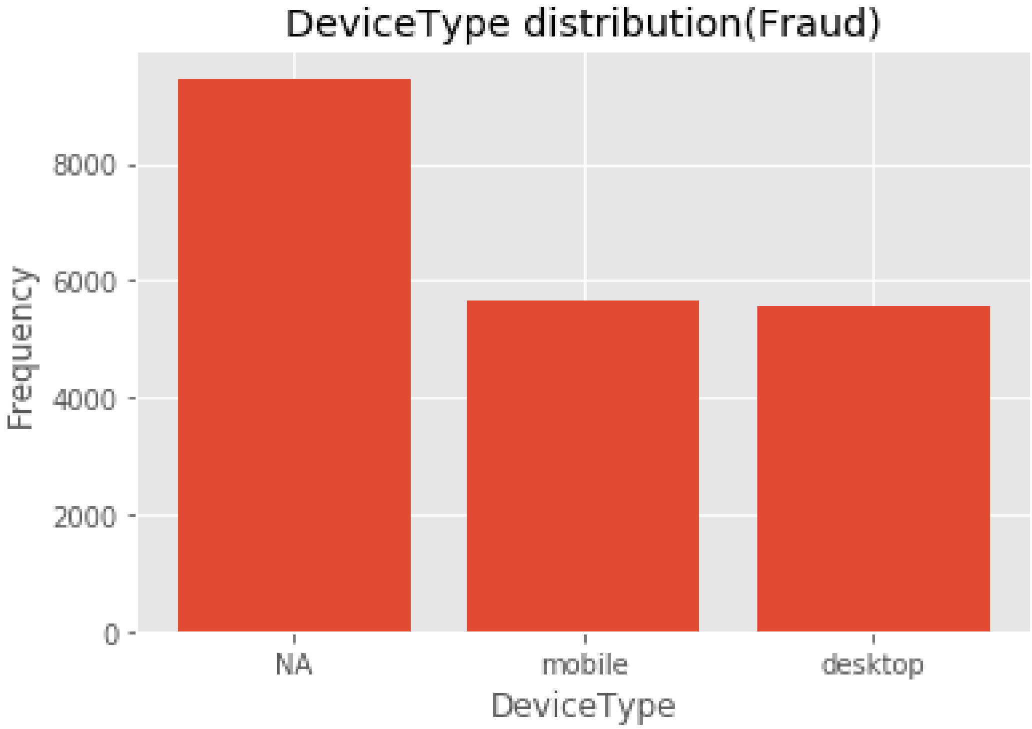
```
In [63]: import pandas as pd
         df = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\train_identity
         df1 = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\train_transac
         df_transaction = df1[['TransactionID','isFraud','TransactionDT','TransactionAmt','Produ
                                      ,'R_emaildomain','addr1','addr2','dist1','dist2']]
         df_identity = df[['TransactionID','DeviceType','DeviceInfo']]
         df_merged = pd.merge(df_identity, df_transaction,on='TransactionID',how='outer')
         df_fraudulent = df_merged[df_merged.isFraud==1]
         df_nonfraudulent = df_merged[df_merged.isFraud==0]
```

1.1 Part 1 - Fraudulent vs Non-Fraudulent Transaction

```
In [24]: import matplotlib.pyplot as plt
         df_fraudulent['DeviceType'] = df_fraudulent['DeviceType'].fillna('NA')
         fig, ax = plt.subplots()
         data_fraud = df_fraudulent['DeviceType'].value_counts()
         points_fraud = data_fraud.index
         frequency_fraud = data_fraud.values
         ax.bar(points_fraud,frequency_fraud)
         ax.set_title('DeviceType distribution(Fraud)')
         ax.set_xlabel('DeviceType')
         ax.set_ylabel('Frequency')
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
```

```
Out[24]: Text(0, 0.5, 'Frequency')
```

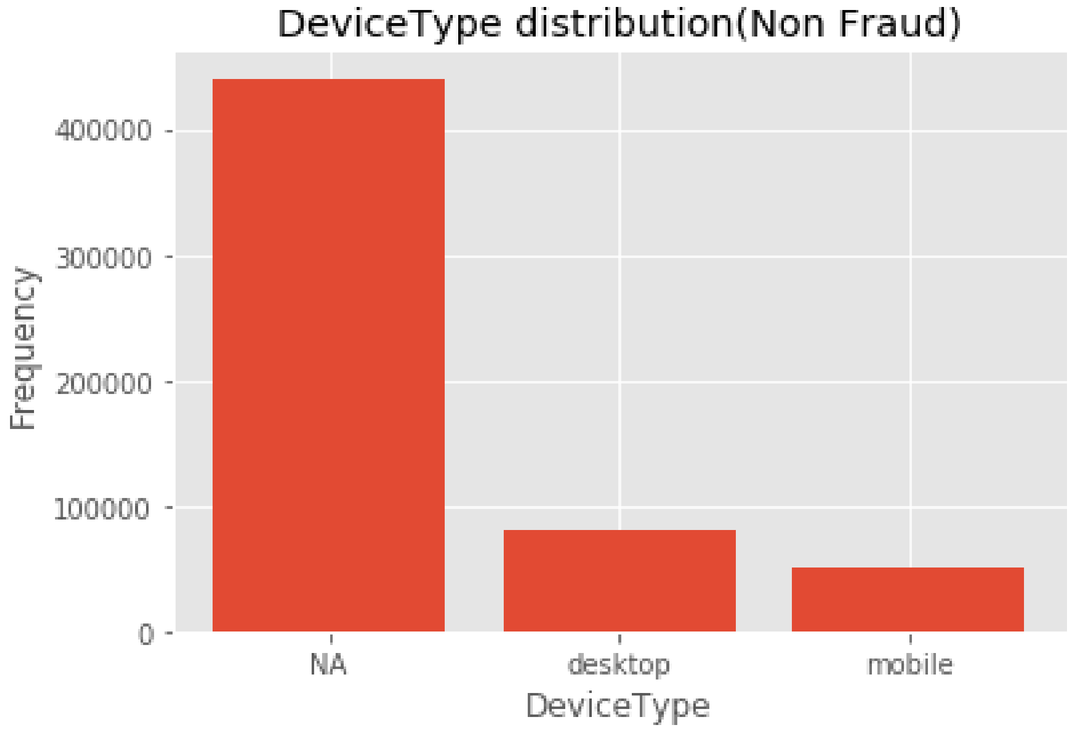


Mobile has higher frauds than desktop as per frequency

```
In [25]: import matplotlib.pyplot as plt
         df_nonfraudulent['DeviceType'] = df_nonfraudulent['DeviceType'].fillna('NA')
         fig, ax = plt.subplots()
         data_nonfraud = df_nonfraudulent['DeviceType'].value_counts()
         points_nonfraud = data_nonfraud.index
         frequency_nonfraud = data_nonfraud.values
         ax.bar(points_nonfraud,frequency_nonfraud)
         ax.set_title('DeviceType distribution(Non Fraud)')
         ax.set_xlabel('DeviceType')
         ax.set_ylabel('Frequency')
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
```

```
Out[25]: Text(0, 0.5, 'Frequency')
```

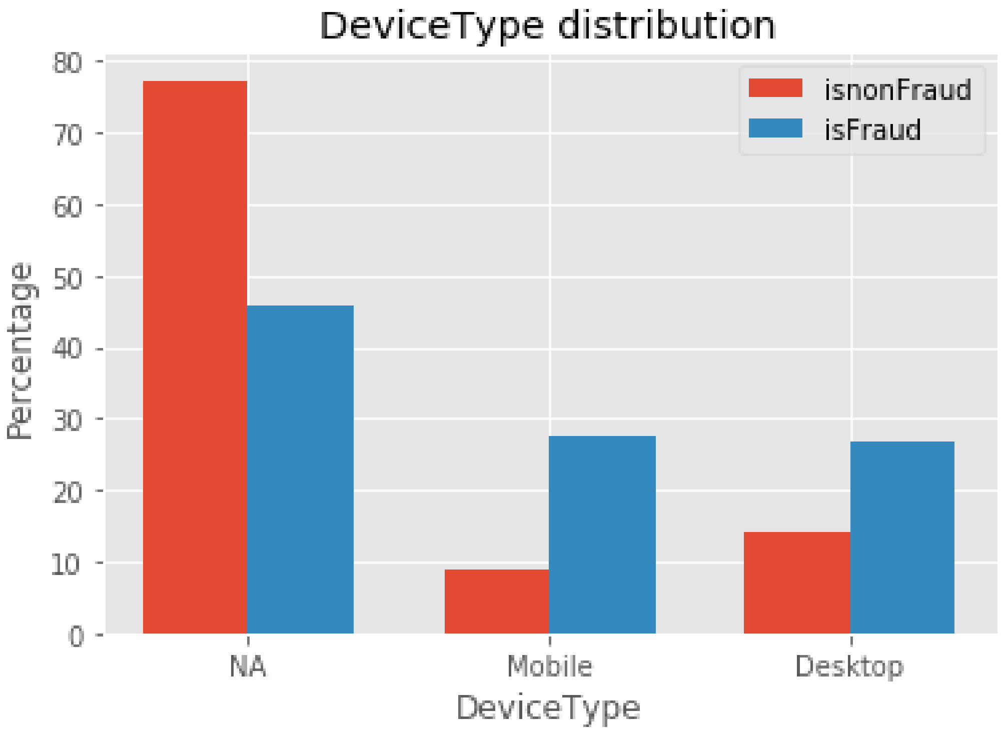


Desktop has numbers that desktop in non fraud case

```
In [26]: import numpy as np #drawn from ref http://benalexkeen.com/bar-charts-in-matplotlib/
         plt.style.use('ggplot')

         ind = np.arange(3)
         df_nonfraudulent['DeviceType'] = df_nonfraudulent['DeviceType'].fillna('NA')
         data_nonfraud = df_nonfraudulent['DeviceType'].value_counts()
         points_nonfraud = data_nonfraud.index
         percentage_nonfraud = (data_nonfraud.values/np.sum(data_nonfraud.values))*100
         temp = percentage_nonfraud[1]
         percentage_nonfraud[1] = percentage_nonfraud[2]
         percentage_nonfraud[2] = temp

         width = 0.35

         plt.bar(ind,percentage_nonfraud,width,label='isnonFraud')

         df_fraudulent['DeviceType'] = df_fraudulent['DeviceType'].fillna('NA')
         data_fraud = df_fraudulent['DeviceType'].value_counts()
         points_fraud = data_fraud.index
         percentage_fraud = (data_fraud.values/np.sum(data_fraud.values))*100
         plt.bar(ind+width,percentage_fraud,width,label='isFraud')
```

```
        plt.title('DeviceType distribution')
        plt.xlabel('DeviceType')
        plt.ylabel('Percentage')
        plt.xticks(ind + width / 2, ('NA', 'Mobile', 'Desktop'))
        plt.legend(loc='best')
        plt.show()
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  """
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
```

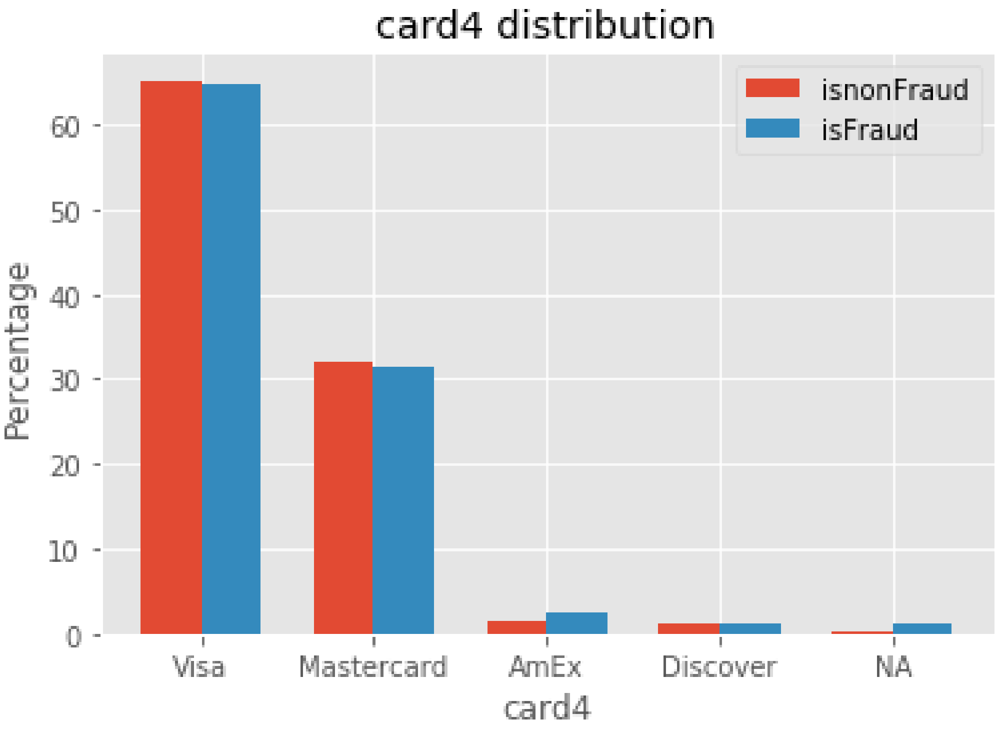


The above is the percentage plot for Desktop, mobile and NA type.

```
In [49]: ind = np.arange(5)
         df_nonfraudulent['card4'] = df_nonfraudulent['card4'].fillna('NA')
         data_nonfraud = df_nonfraudulent['card4'].value_counts()
```

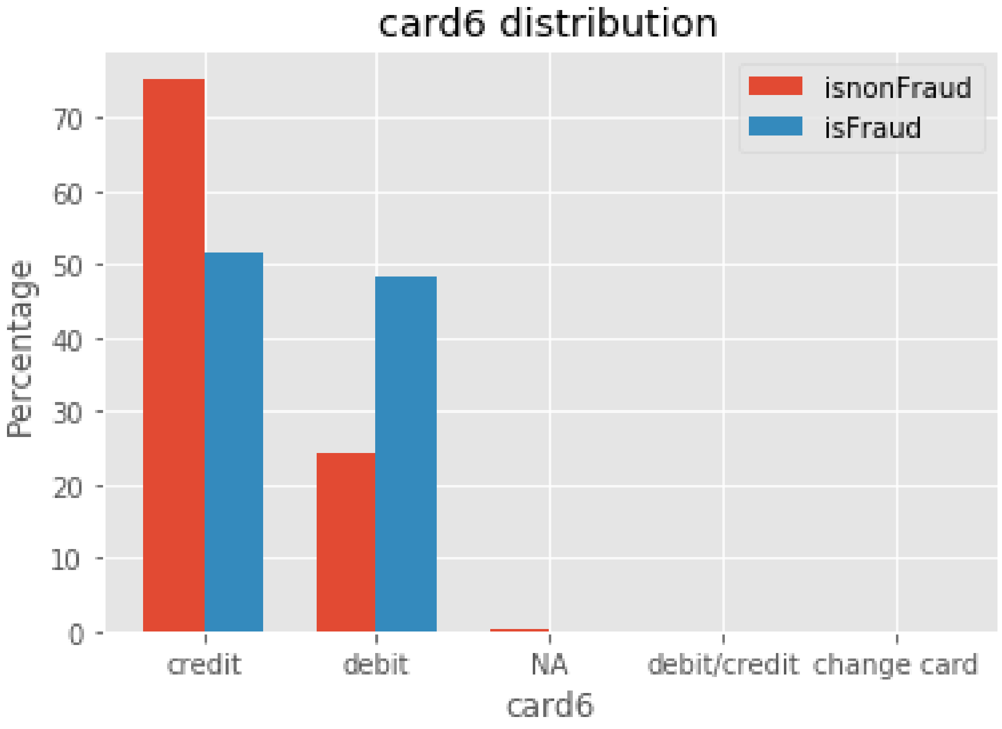
```
points_nonfraud = data_nonfraud.index
percentage_nonfraud = (data_nonfraud.values/np.sum(data_nonfraud.values))*100
width = 0.35

plt.bar(ind,percentage_nonfraud,width,label='isnonFraud')

df_fraudulent['card4'] = df_fraudulent['card4'].fillna('NA')
data_fraud = df_fraudulent['card4'].value_counts()
points_fraud = data_fraud.index

percentage_fraud = (data_fraud.values/np.sum(data_fraud.values))*100
temp = percentage_fraud[3]
percentage_fraud[4] = percentage_fraud[3]
percentage_fraud[3] = temp


plt.bar(ind+width,percentage_fraud,width,label='isFraud')

plt.title('card4 distribution')
plt.xlabel('card4')
plt.ylabel('Percentage')
plt.xticks(ind + width/2, ('Visa', 'Mastercard', 'AmEx','Discover','NA'))
plt.legend(loc='best')
plt.show()
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#

D:\Anaconda\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  # Remove the CWD from sys.path while we load stuff.
```



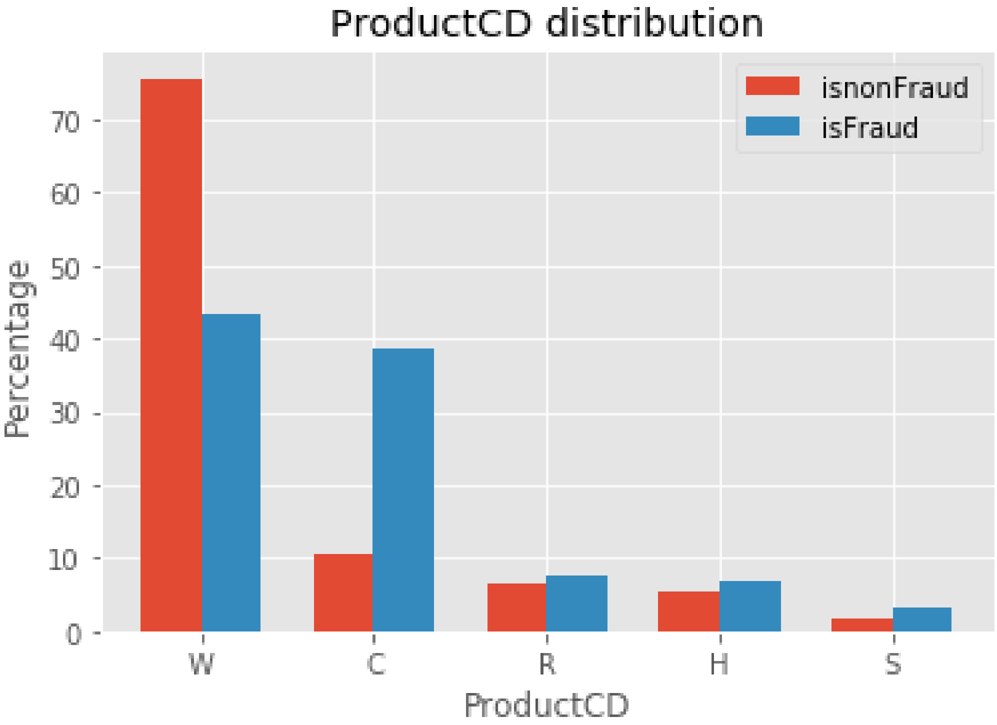
```
In [48]: ind = np.arange(5)
         df_nonfraudulent['card6'] = df_nonfraudulent['card6'].fillna('NA')
         data_nonfraud = df_nonfraudulent['card6'].value_counts()
         points_nonfraud = data_nonfraud.index
         percentage_nonfraud = (data_nonfraud.values/np.sum(data_nonfraud.values))*100
         width = 0.35

         plt.bar(ind,percentage_nonfraud,width,label='isnonFraud')

         df_nonfraudulent['card6'] = df_nonfraudulent['card6'].fillna('NA')
         data_fraud = df_fraudulent['card6'].value_counts()
         points_fraud = data_fraud.index
         percentage_fraud = (data_fraud.values/np.sum(data_fraud.values))*100

         percentage_fraud = np.append(percentage_fraud,[0.0,0.0,0.0])

         plt.bar(ind+width,percentage_fraud,width,label='isFraud')

         plt.title('card6 distribution')
         plt.xlabel('card6')
         plt.ylabel('Percentage')
         plt.xticks(ind + width/2, ('credit', 'debit', 'NA','debit/credit','change card'))
         plt.legend(loc='best')
         plt.show()
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#

D:\Anaconda\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  # Remove the CWD from sys.path while we load stuff.
```



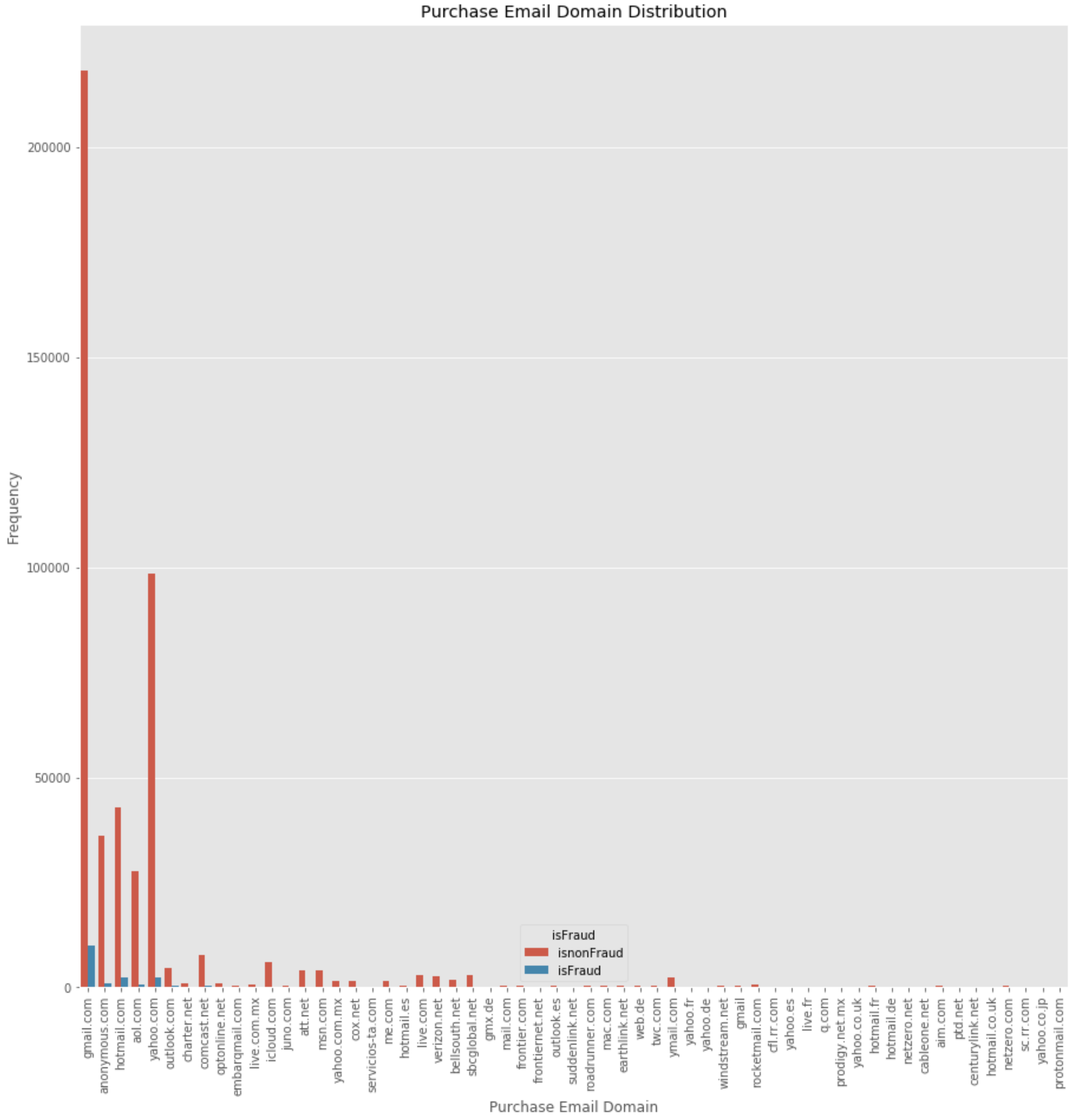
```
In [53]: ind = np.arange(5)

         data_nonfraud = df_nonfraudulent['ProductCD'].value_counts()
         points_nonfraud = data_nonfraud.index
         percentage_nonfraud = (data_nonfraud.values/np.sum(data_nonfraud.values))*100
         width = 0.35

         plt.bar(ind,percentage_nonfraud,width,label='isnonFraud')

         data_fraud = df_fraudulent['ProductCD'].value_counts()
```

```
points_fraud = data_fraud.index
percentage_fraud = (data_fraud.values/np.sum(data_fraud.values))*100
temp = percentage_nonfraud[2]
percentage_nonfraud[2] = percentage_nonfraud[3]
percentage_nonfraud[3] = temp

plt.bar(ind+width,percentage_fraud,width,label='isFraud')

plt.title('ProductCD distribution')
plt.xlabel('ProductCD')
plt.ylabel('Percentage')
plt.xticks(ind + width/2, ('W','C', 'R', 'H', 'S'))
plt.legend(loc='best')
plt.show()
```

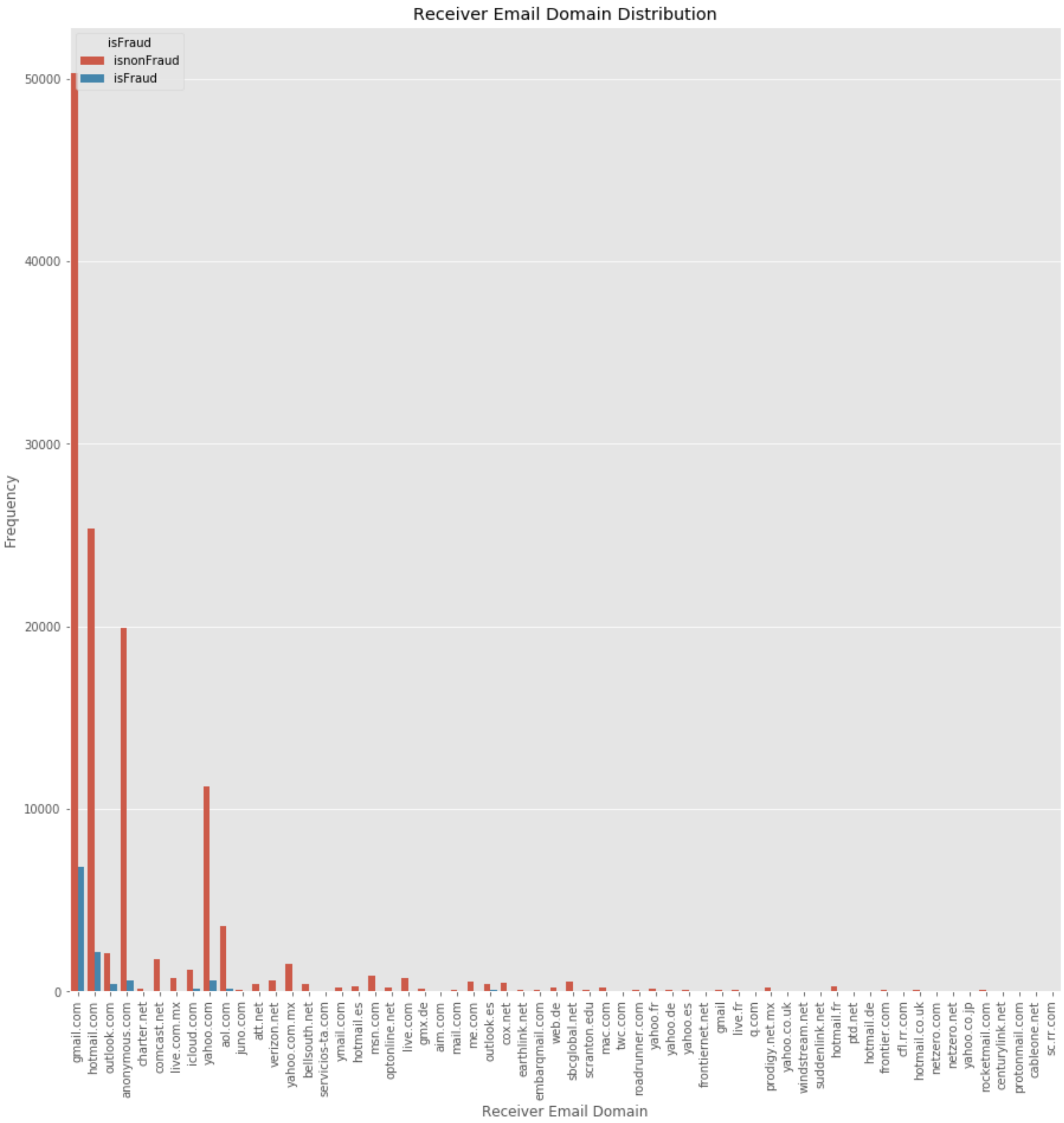


```
In [73]: import seaborn as sns
         import matplotlib.pyplot as plt
         fig = plt.figure(figsize=(15,15))
         plot = sns.countplot(x='P_emaildomain',data=df_merged,hue='isFraud')
         plot.set_xticklabels(plot.get_xticklabels(),rotation=90)
         plot.set_title('Purchase Email Domain Distribution')
         plot.set_xlabel('Purchase Email Domain')
         plot.set_ylabel('Frequency')
         plot.legend(title='isFraud',labels=['isnonFraud','isFraud'])
```

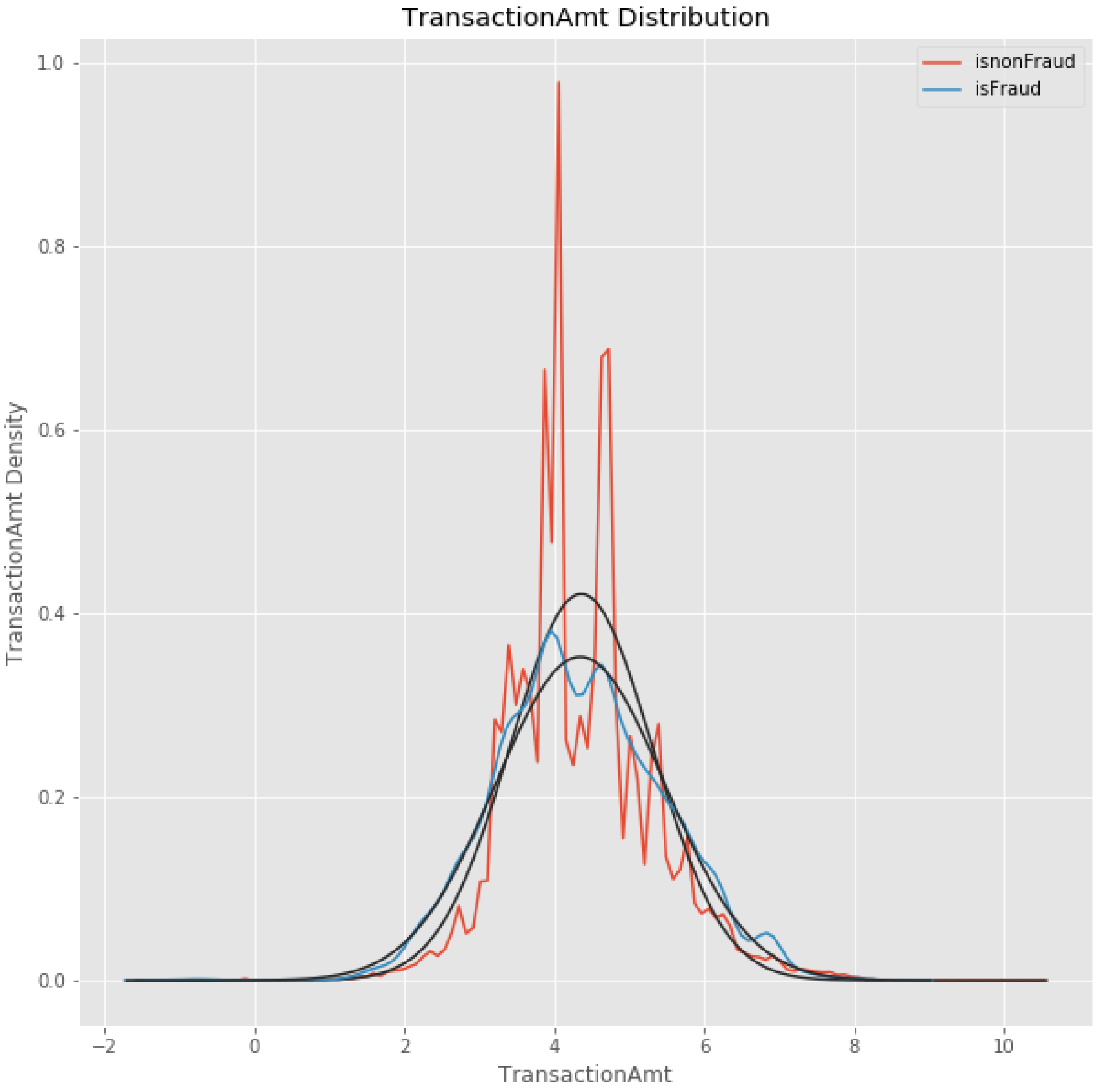
```
Out[73]: <matplotlib.legend.Legend at 0x213911e1550>
```



```
In [74]: fig = plt.figure(figsize=(15,15))
         plot = sns.countplot(x='R_emaildomain',data=df_merged,hue='isFraud')
         plot.set_xticklabels(plot.get_xticklabels(),rotation=90)
         plot.set_title('Receiver Email Domain Distribution')
         plot.set_xlabel('Receiver Email Domain')
         plot.set_ylabel('Frequency')
         plot.legend(title='isFraud',labels=['isnonFraud','isFraud'])

Out[74]: <matplotlib.legend.Legend at 0x213910c9f60>
```

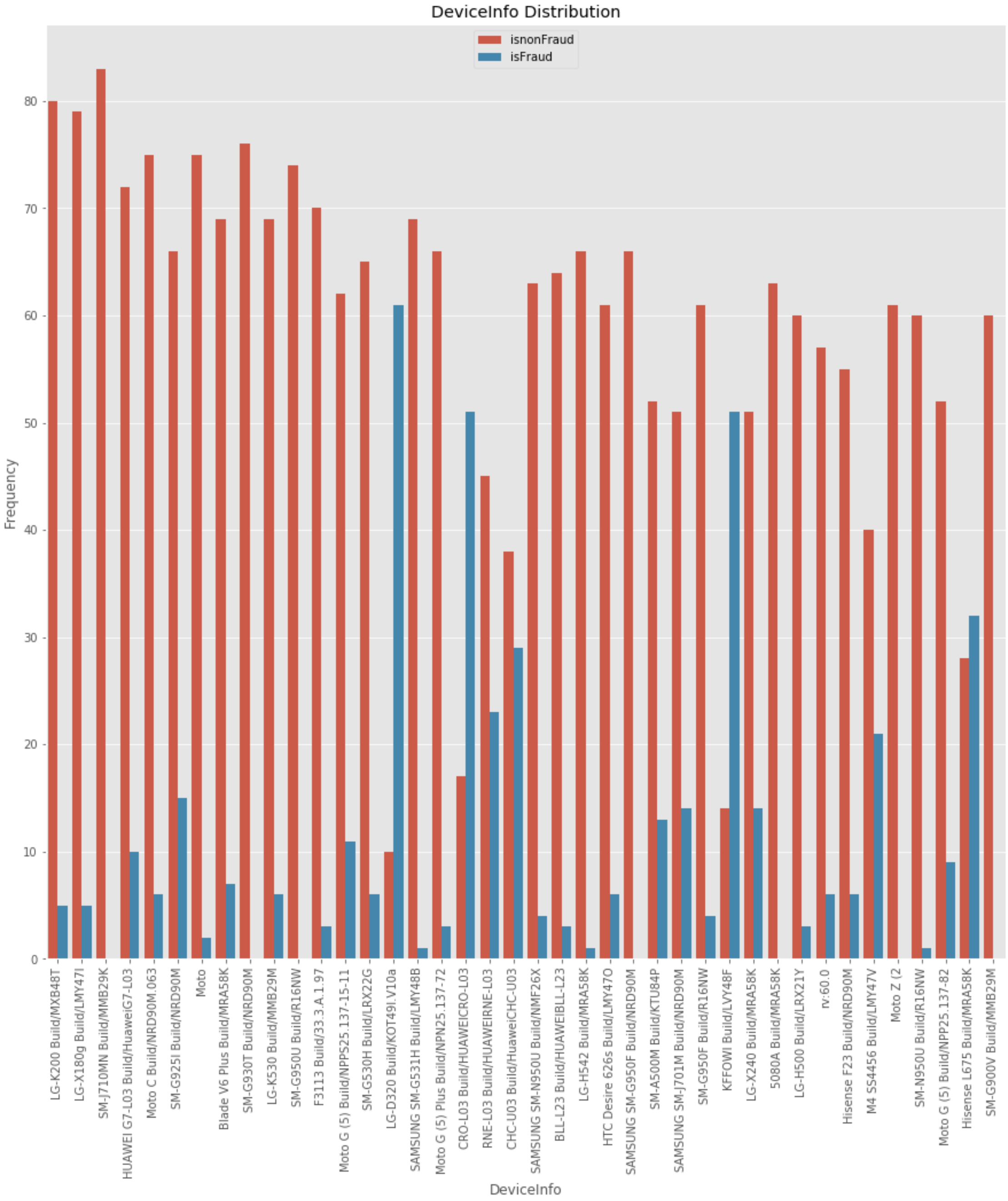


```
In [56]: from scipy.stats import norm
         fig, ax = plt.subplots(figsize=(10,10))
         sns.set_color_codes()
         ax = sns.distplot(np.log(df_nonfraudulent['TransactionAmt']),fit=norm, rug=False, hist=
         ax = sns.distplot(np.log(df_fraudulent['TransactionAmt']),fit=norm, rug=False, hist=Fal
         ax.set_title('TransactionAmt Distribution')
         ax.set_ylabel('TransactionAmt Density')
         ax.set_xlabel('TransactionAmt')

Out[56]: Text(0.5, 0, 'TransactionAmt')
```

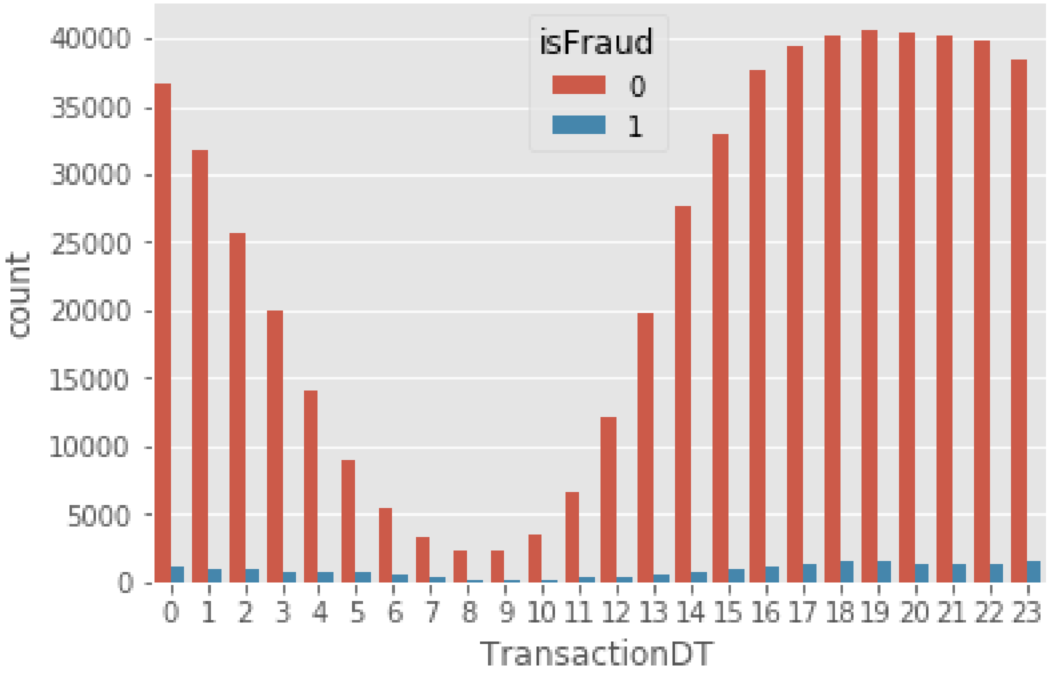


```
In [57]: import matplotlib.pyplot as plt
         import seaborn as sns
         fig = plt.figure(figsize=(15,15))
         plot = sns.countplot(x='DeviceInfo',data=df_merged,order = df_merged['DeviceInfo'].valu
         plot.set_xticklabels(plot.get_xticklabels(),rotation=90)
         plot.set_xlabel('DeviceInfo')
         plot.set_ylabel('Frequency')
         plot.set_title('DeviceInfo Distribution')
         plot.legend(labels=['isnonFraud','isFraud'])
```

```
Out[57]: <matplotlib.legend.Legend at 0x21367719518>
```



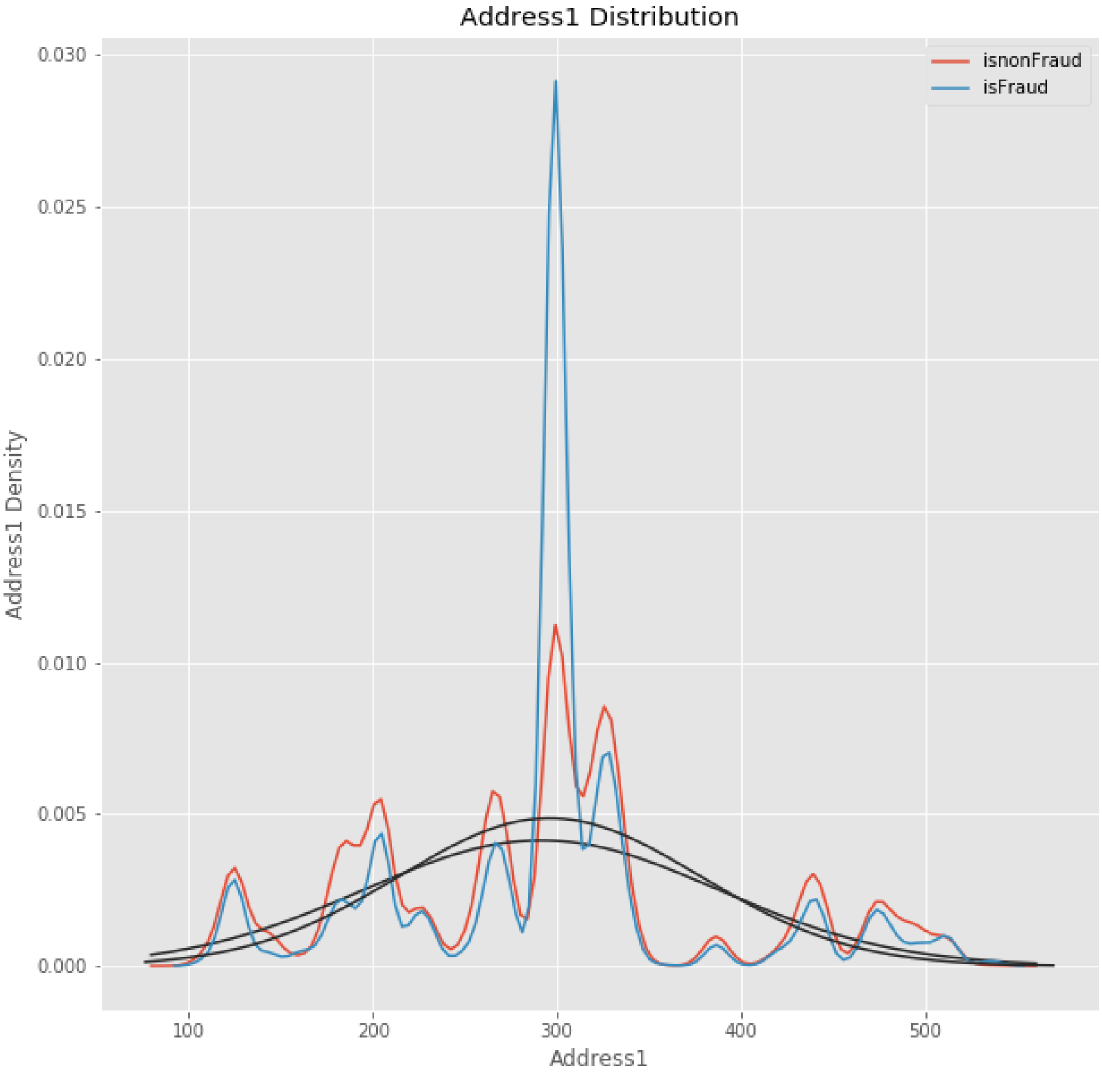
```
In [72]: import matplotlib.pyplot as plt
         import seaborn as sns
         df_merged['TransactionDT'] = (df_merged['TransactionDT']/3600)%24
         df_merged['TransactionDT'] = df_merged['TransactionDT'].astype(int)
         sns.countplot(x='TransactionDT',data=df_merged,hue='isFraud')

Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x213911caa90>
```



```
In [79]: fig, ax = plt.subplots(figsize=(10,10))
         sns.set_color_codes()
         df_nonfraudulent['addr1']=df_nonfraudulent['addr1'].fillna(np.nanmedian(df_nonfraudulen
         df_fraudulent['addr1']=df_fraudulent['addr1'].fillna(np.nanmedian(df_fraudulent['addr1'
         ax = sns.distplot(df_nonfraudulent['addr1'],fit=norm, rug=False, hist=False, label='isn
         ax = sns.distplot(df_fraudulent['addr1'],fit=norm, rug=False, hist=False, label='isFrau
         ax.set_title('Address1 Distribution')
         ax.set_ylabel('Address1 Density')
         ax.set_xlabel('Address1')
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  This is separate from the ipykernel package so we can avoid doing imports until
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  after removing the cwd from sys.path.
```

```
Out[79]: Text(0.5, 0, 'Address1')
```

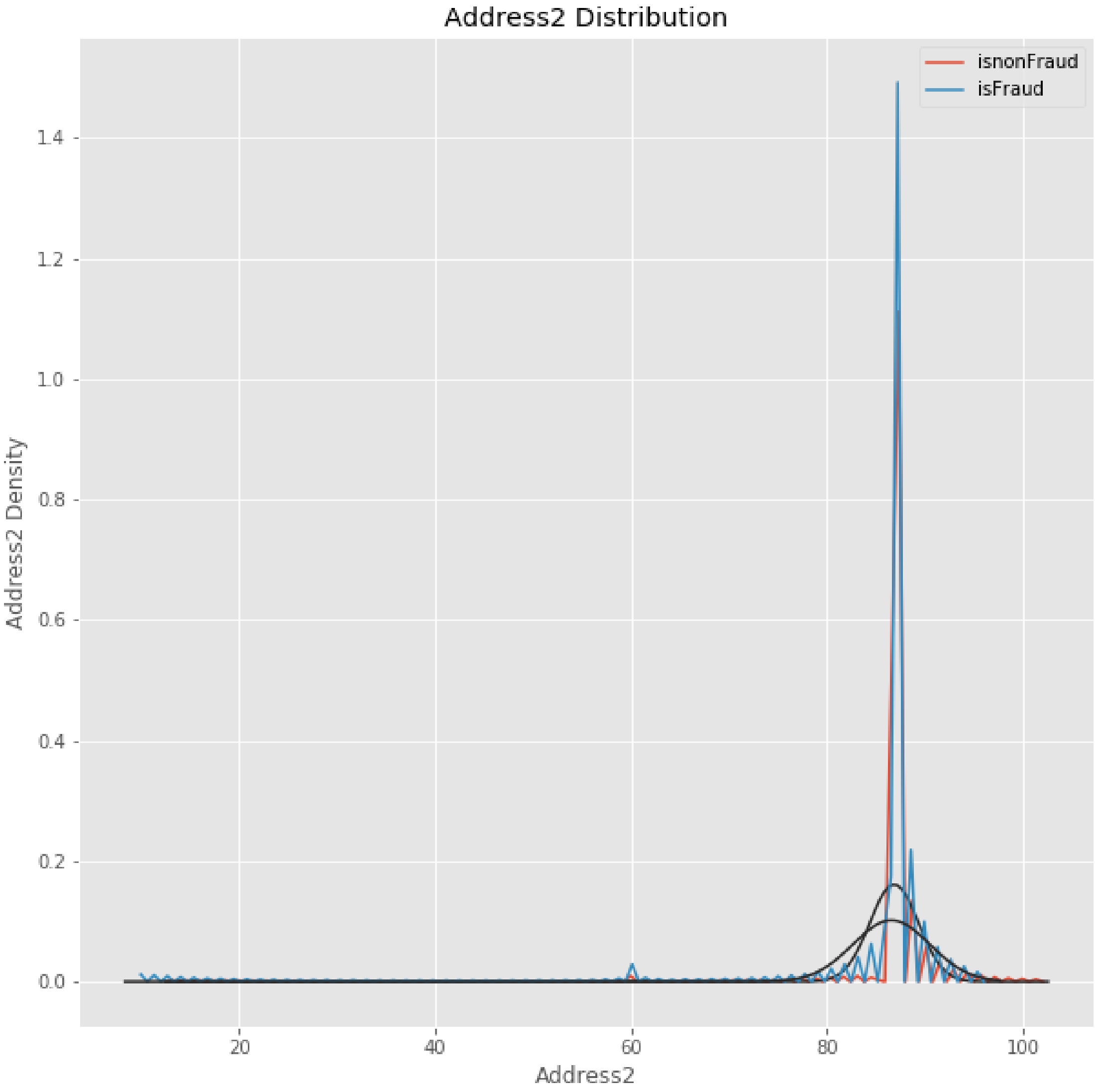


```
In [118]: fig, ax = plt.subplots(figsize=(10,10))
          df_nonfraudulent['addr2']=df_nonfraudulent['addr2'].fillna(np.nanmedian(df_nonfraudule
          df_fraudulent['addr2']=df_fraudulent['addr2'].fillna(np.nanmedian(df_fraudulent['addr2
          ax = sns.distplot(df_nonfraudulent['addr2'],fit=norm, rug=False, hist=False, label='is
          ax = sns.distplot(df_fraudulent['addr2'],fit=norm, rug=False, hist=False, label='isFra
          ax.set_title('Address2 Distribution')
          ax.set_ylabel('Address2 Density')
          ax.set_xlabel('Address2')
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
```

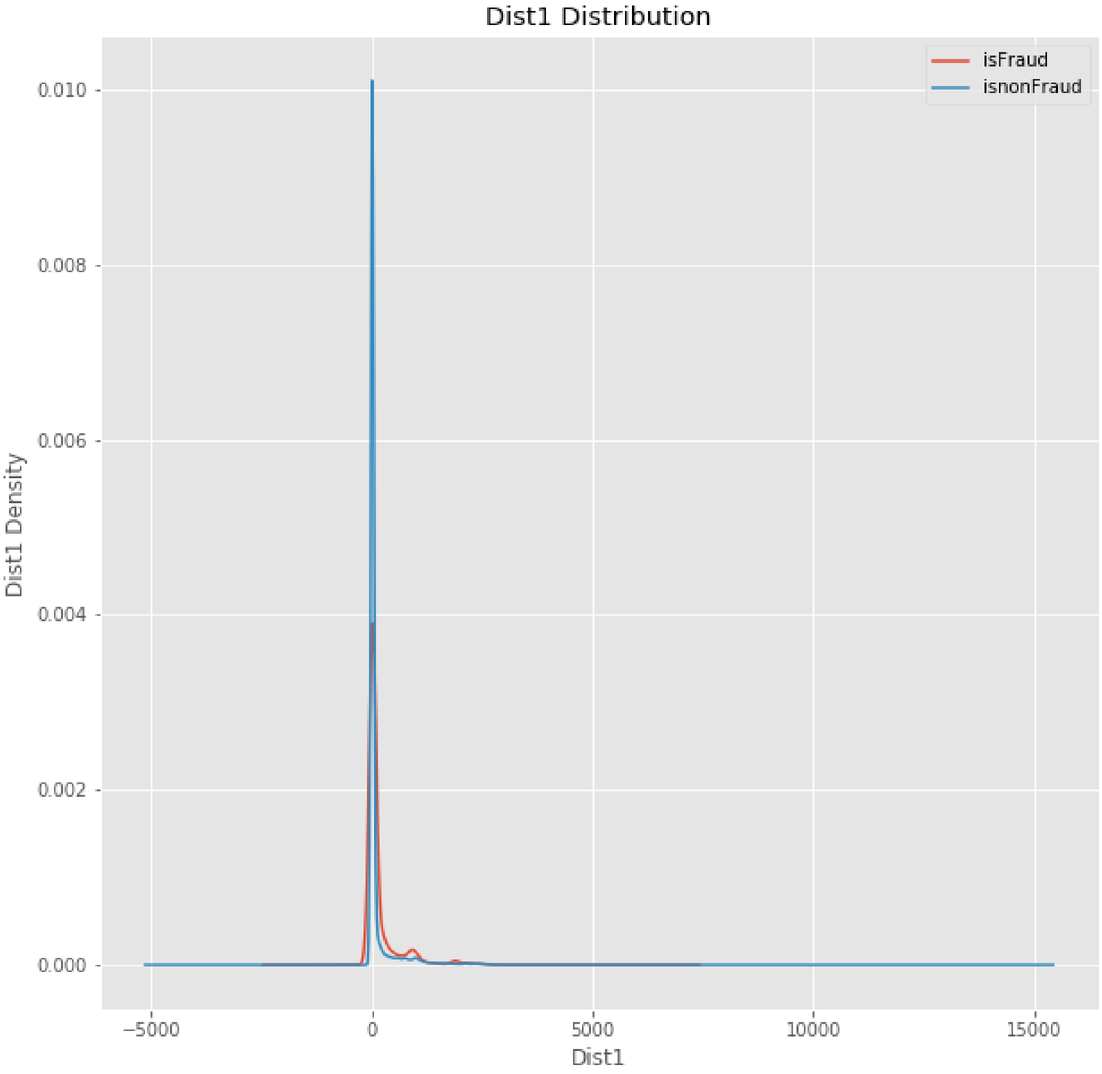
```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
Out[118]: Text(0.5, 0, 'Address2')
```



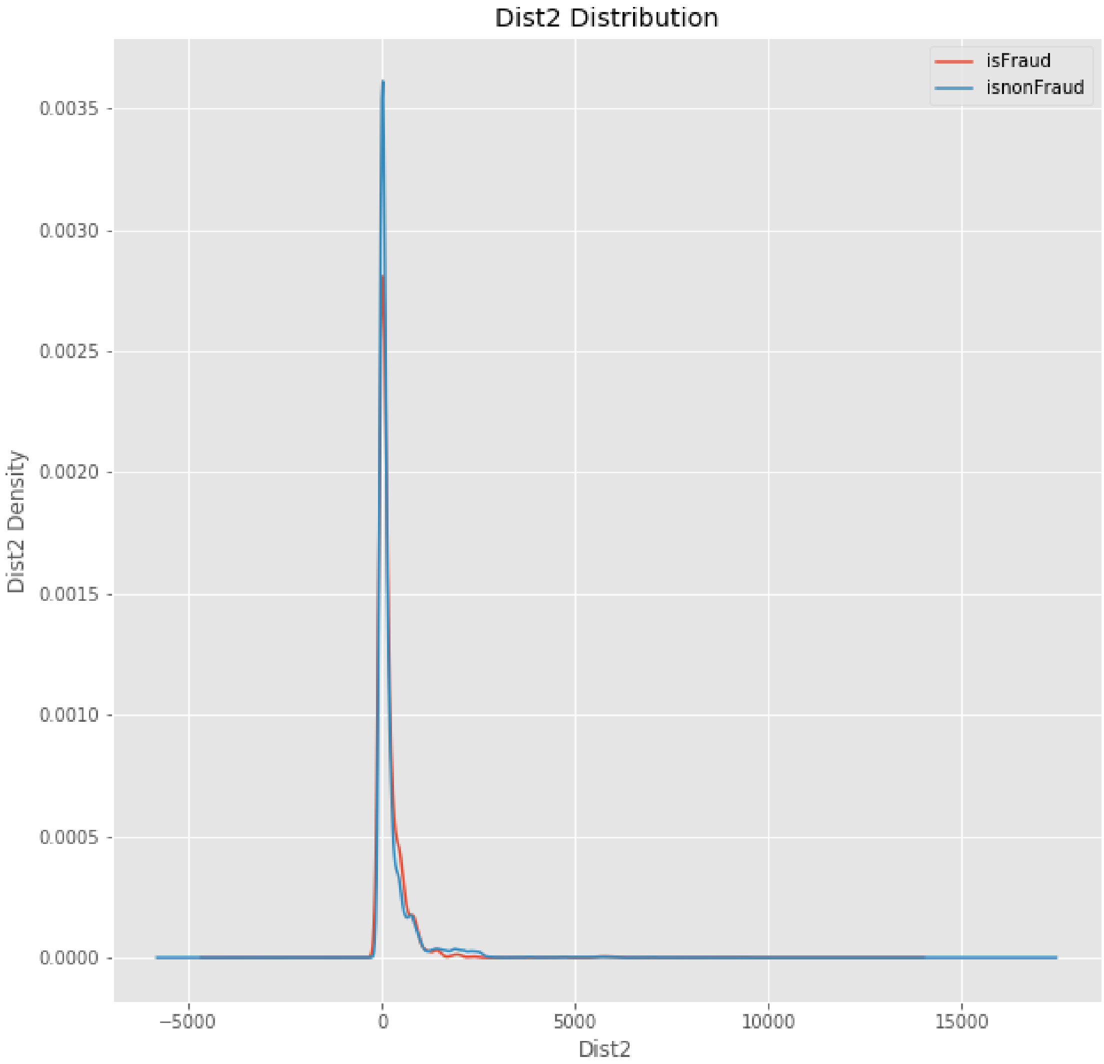
```
In [139]: fig, ax = plt.subplots(figsize=(10,10))
          #df_nonfraudulent['dist1']=df_nonfraudulent['dist1'].fillna(np.nanmedian(df_nonfraudul
          #df_fraudulent['dist1']=df_fraudulent['dist1'].fillna(np.nanmedian(df_fraudulent['dist
          #ax = sns.distplot(df_nonfraudulent['dist1'], rug=False, hist=False, label='isnonFraud
          #ax = sns.distplot(df_fraudulent['dist1'], rug=False, hist=False, label='isFraud')
          ax = df_fraudulent['dist1'].plot.kde()
          ax = df_nonfraudulent['dist1'].plot.kde()
          ax.set_title('Dist1 Distribution')
          ax.set_ylabel('Dist1 Density')
          ax.set_xlabel('Dist1')
          ax.legend(['isFraud','isnonFraud'])
```

```
Out[139]: <matplotlib.legend.Legend at 0x214089f0d30>
```



```
In [131]: fig, ax = plt.subplots(figsize=(10,10))
         #df_nonfraudulent['dist2']=df_nonfraudulent['dist2'].fillna(np.nanmedian(df_nonfraudul
         #df_fraudulent['dist2']=df_fraudulent['dist2'].fillna(np.nanmedian(df_fraudulent['dist
         #ax = sns.distplot(df_nonfraudulent['dist2'], rug=False, hist=False, label='isnonFraud
         #ax = sns.distplot(df_fraudulent['dist2'], rug=False, hist=False, label='isFraud')
         ax = df_fraudulent['dist2'].plot.kde()
         ax = df_nonfraudulent['dist2'].plot.kde()
         ax.set_title('Dist2 Distribution')
         ax.set_ylabel('Dist2 Density')
         ax.set_xlabel('Dist2')
         ax.legend(['isFraud','isnonFraud'])
```

```
Out[131]: <matplotlib.legend.Legend at 0x21408c2b6a0>
```

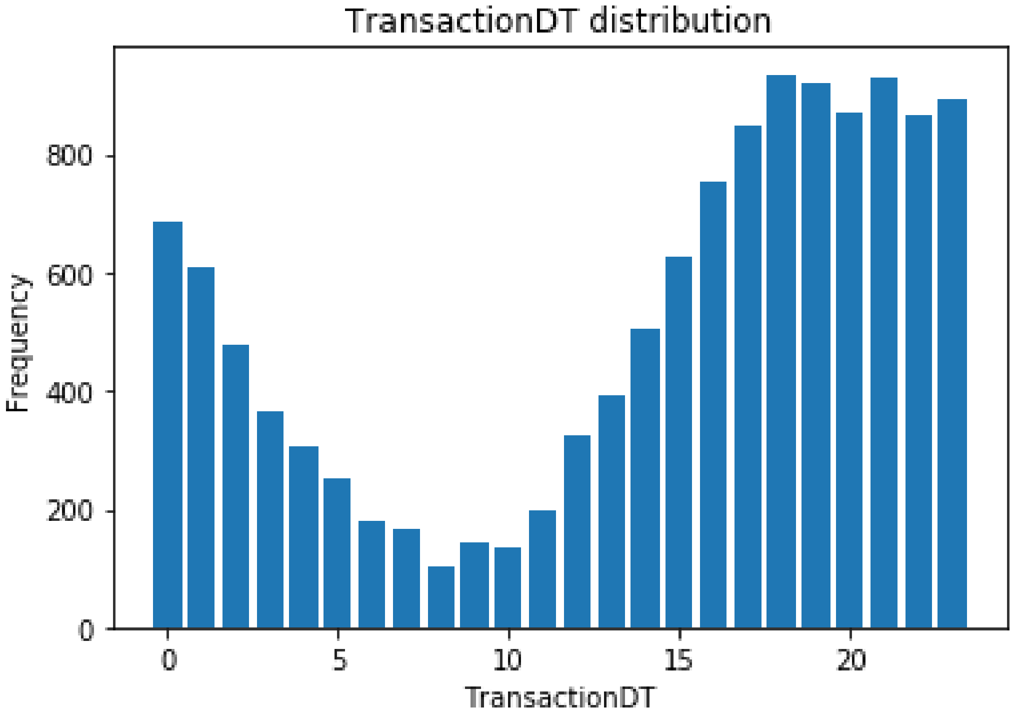


1.2 Part 2 - Transaction Frequency

```
In [7]: import matplotlib.pyplot as plt
        import numpy as np
        fig, ax = plt.subplots()
        df_fraudulent['TransactionDT'] = (df_fraudulent['TransactionDT']/3600)%24
        df_fraudulent['TransactionDT'] = df_fraudulent['TransactionDT'].astype(int)
        dfaddr2 = df_fraudulent[df_fraudulent['addr2']==87]#most frequent country code
        data_merged = dfaddr2['TransactionDT'].value_counts()
        points_merged = data_merged.index
        frequency_merged = data_merged.values
        ax.bar(points_merged,frequency_merged)
        ax.set_title('TransactionDT distribution')
        ax.set_xlabel('TransactionDT')
        ax.set_ylabel('Frequency')
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  after removing the cwd from sys.path.
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  """
```

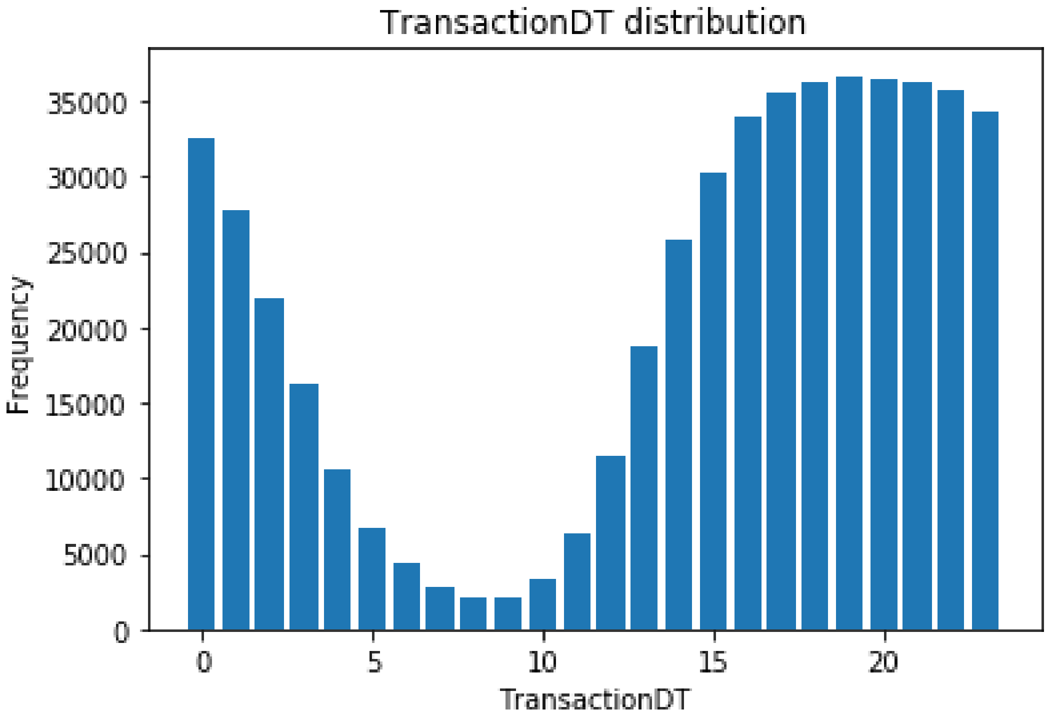
```
Out[7]: Text(0, 0.5, 'Frequency')
```



```
In [8]: fig, ax = plt.subplots()
        df_nonfraudulent['TransactionDT'] = (df_nonfraudulent['TransactionDT']/3600)%24
        df_nonfraudulent['TransactionDT'] = df_nonfraudulent['TransactionDT'].astype(int)
        dfaddr2 = df_nonfraudulent[df_nonfraudulent['addr2']==87]#most frequent country code
        data_merged = dfaddr2['TransactionDT'].value_counts()
        points_merged = data_merged.index
        frequency_merged = data_merged.values
        ax.bar(points_merged,frequency_merged)
        ax.set_title('TransactionDT distribution')
        ax.set_xlabel('TransactionDT')
        ax.set_ylabel('Frequency')
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#

D:\Anaconda\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
Out[8]: Text(0, 0.5, 'Frequency')
```



1. From the first plot as we can see, in the time between 5-10, the relative frequencies of fraud is higher in length compared to the second plot which is non fraud one
2. And since, during sleeping hours, the number of transactions would be lesser compared to waking hours, one can safely guess that the sleeping hours are possible between 4-12(10pm to 7am)

1.3 Part 3 - Product Code

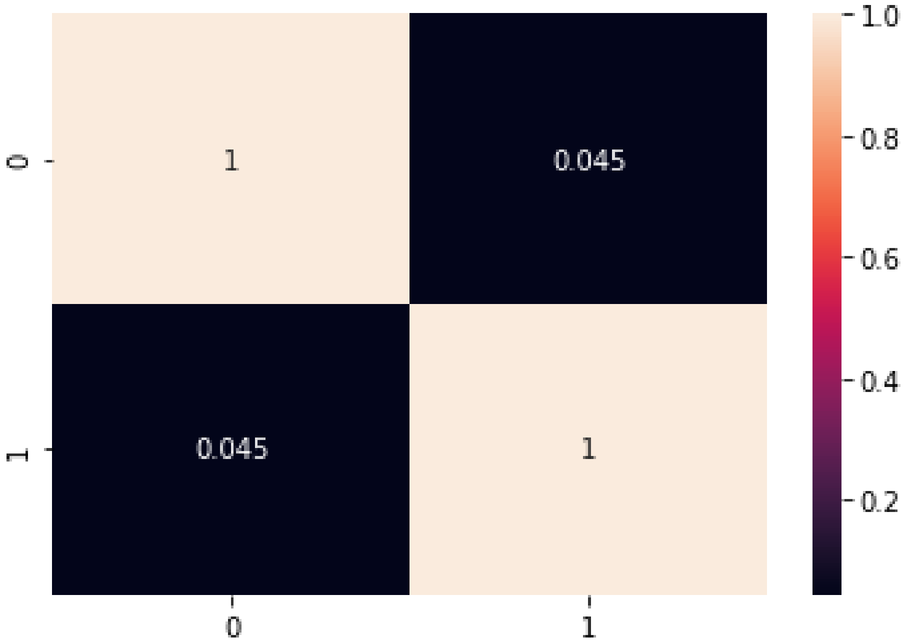
```
In [10]: import numpy as np
         c = np.mean(df_merged[df_merged['ProductCD']=='C']['TransactionAmt'])#42.8
         h = np.mean(df_merged[df_merged['ProductCD']=='H']['TransactionAmt'])#73.2
         r = np.mean(df_merged[df_merged['ProductCD']=='R']['TransactionAmt'])#168.3
         s = np.mean(df_merged[df_merged['ProductCD']=='S']['TransactionAmt'])#60.3
         w = np.mean(df_merged[df_merged['ProductCD']=='W']['TransactionAmt'])#153.2
```

On calculating various means for each item in each case, one can observe that the mean is greatest for product type R on a whole and therefore it probably is the costliest

1.4 Part 4 - Correlation Coefficient

```
In [17]: import seaborn as sns
         c = np.corrcoef((df_merged['TransactionDT']/3600)%24,df_merged['TransactionAmt'])
         sns.heatmap(c, annot=True)
```

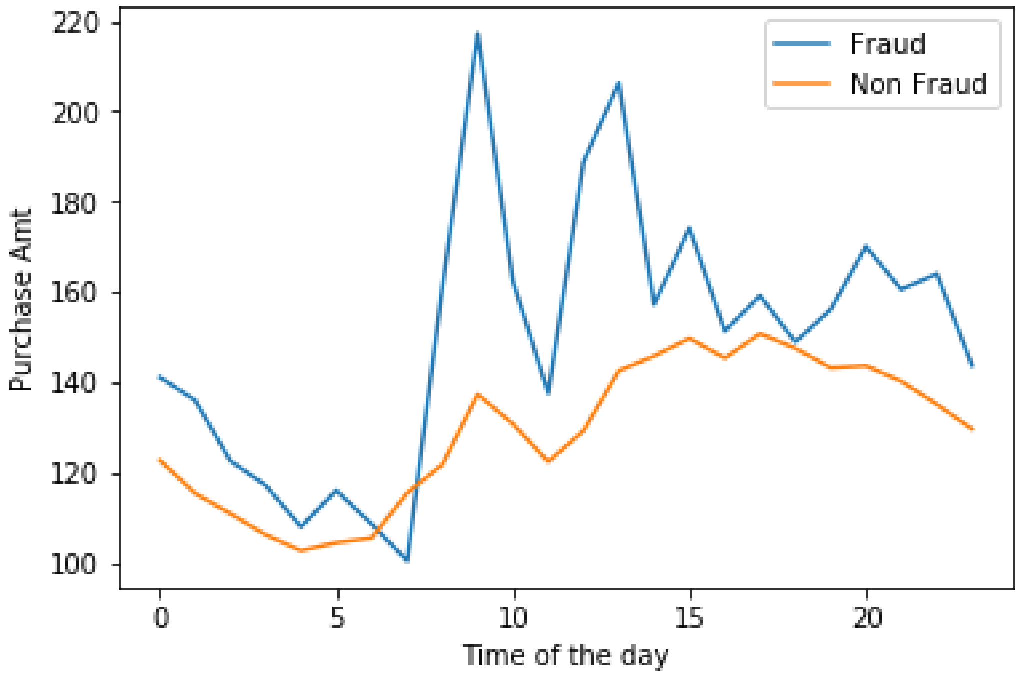
```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x25f3524f3c8>
```



```
In [64]: import matplotlib.pyplot as plt
         import pandas as pd
         import numpy as np

         TransactionDT_hour = pd.to_datetime(df_fraudulent['TransactionDT'],unit='s').dt.hour
         TransactionDT_hour_non = pd.to_datetime(df_nonfraudulent['TransactionDT'],unit='s').dt.

         hour_Amt = pd.DataFrame({'Transaction_hour':TransactionDT_hour,'TransactionAmt':df_frau
         hour_Amt_non = pd.DataFrame({'Transaction_hour':TransactionDT_hour_non,'TransactionAmt'

         list = []
         list1 = []
         for i in range(0,24):
             list.append(np.mean(hour_Amt[hour_Amt['Transaction_hour']==i]['TransactionAmt']))
             list1.append(np.mean(hour_Amt_non[hour_Amt_non['Transaction_hour']==i]['Transaction
```

```
In [65]: plt.plot(range(0,24),list)
         plt.plot(range(0,24),list1)
         plt.xlabel('Time of the day')
         plt.ylabel('Purchase Amt')
         plt.legend(['Fraud','Non Fraud'])
```

```
Out[65]: <matplotlib.legend.Legend at 0x25f2edce2e8>
```



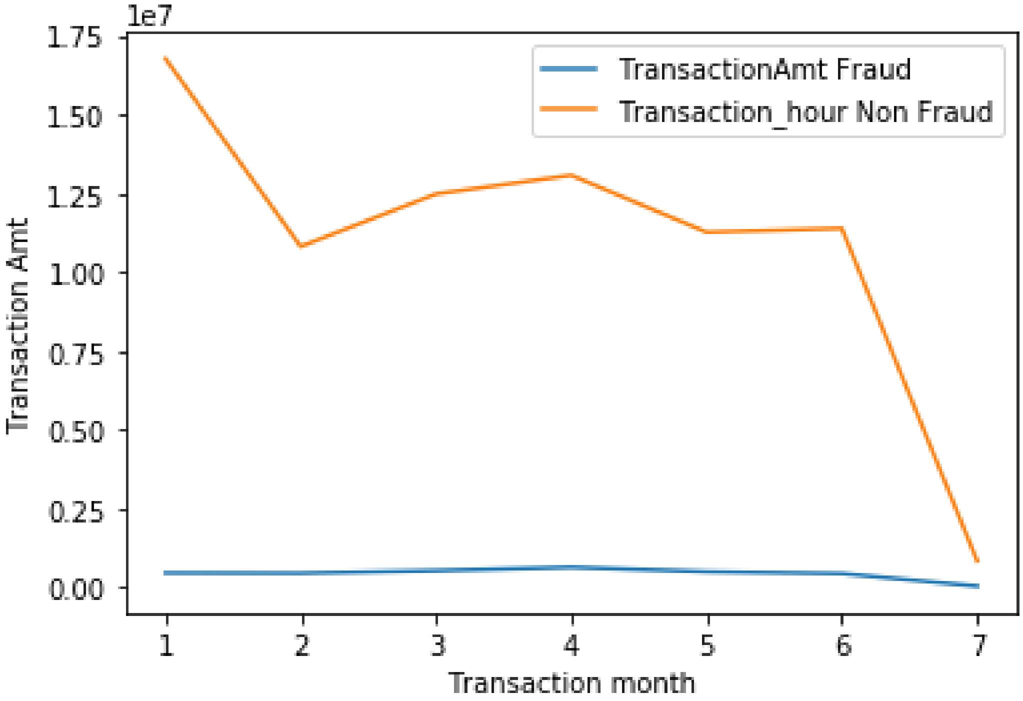
The correlation coefficient is 0.0445

1.5 Part 5 - Interesting Plot

```
In [52]: import matplotlib.pyplot as plt
         import pandas as pd

In [53]: TransactionDT_month = pd.to_datetime(df_fraudulent['TransactionDT'],unit='s').dt.month
         TransactionDT_month_non = pd.to_datetime(df_nonfraudulent['TransactionDT'],unit='s').dt

In [54]: TransactionDT_day = pd.to_datetime(df_fraudulent['TransactionDT'],unit='s').dt.day
         TransactionDT_day_non = pd.to_datetime(df_nonfraudulent['TransactionDT'],unit='s').dt.d

In [56]: month_Amt = pd.DataFrame({'Transaction_month':TransactionDT_month,'TransactionAmt':df_f
         month_Amt_non = pd.DataFrame({'Transaction_month':TransactionDT_month_non,'TransactionA

In [57]: import numpy as np
         list = []
         list1 = []
         for i in range(1,8):
             list.append(np.sum(month_Amt[month_Amt['Transaction_month']==i]['TransactionAmt']))
             list1.append(np.sum(month_Amt_non[month_Amt_non['Transaction_month']==i]['Transacti

In [58]: import matplotlib.pyplot as plt
         plt.plot(range(1,8),list)
         plt.plot(range(1,8),list1)
```

```
        plt.xlabel('Transaction month')
        plt.ylabel('Transaction Amt')
        plt.legend(['TransactionAmt Fraud','Transaction_hour Non Fraud'])
```

```
Out[58]: <matplotlib.legend.Legend at 0x25f2fa5bba8>
```

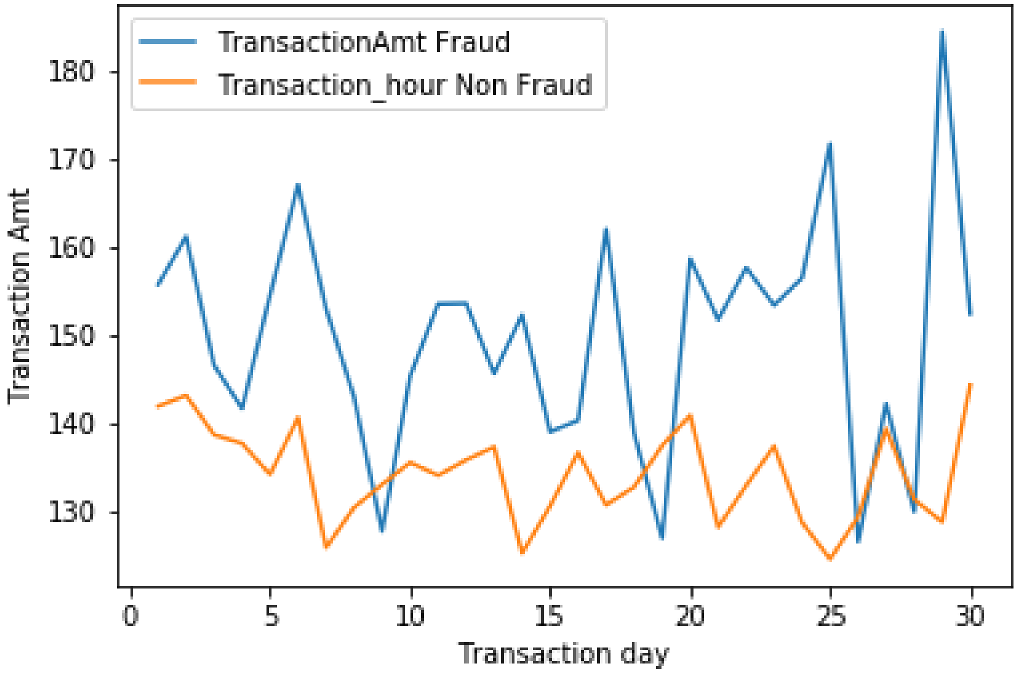


```
In [59]: day_Amt = pd.DataFrame({'Transaction_day':TransactionDT_day,'TransactionAmt':df_fraudul
         day_Amt_non = pd.DataFrame({'Transaction_day':TransactionDT_day_non,'TransactionAmt':df
```

```
In [60]: import numpy as np
         list = []
         list1 = []
         for i in range(1,31):
             list.append(np.mean(day_Amt[day_Amt['Transaction_day']==i]['TransactionAmt']))
             list1.append(np.mean(day_Amt_non[day_Amt_non['Transaction_day']==i]['TransactionAmt
```

```
In [61]: plt.plot(range(1,31),list)
         plt.plot(range(1,31),list1)
         plt.xlabel('Transaction day')
         plt.ylabel('Transaction Amt')
         plt.legend(['TransactionAmt Fraud','Transaction_hour Non Fraud'])
```

```
Out[61]: <matplotlib.legend.Legend at 0x25f311e2a90>
```



1. The month plot doesn't seem to maintain much of correlation between fraud and non-fraud but the day plot against the T-amount seems to almost maintain good correlation between them.

1.6 Part 6 - Prediction Model

```
In [6]: #Cleaning Training data and storing in a csv after conversion of categorical features to

        import pandas as pd
        import numpy as np
        from sklearn.preprocessing import LabelEncoder
        from sklearn.feature_extraction import FeatureHasher
        from sklearn.linear_model import LinearRegression
        from sklearn import metrics
        df_train = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\train_ide
        df1_train = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\train_tr
        df_merged_train = pd.merge(df_train, df1_train,on='TransactionID',how='outer')
        df_merged_train['DeviceType'] = df_merged_train['DeviceType'].fillna('X')
        df_merged_train['DeviceInfo'] = df_merged_train['DeviceInfo'].fillna('Y')
        df_merged_train['card4'] = df_merged_train['card4'].fillna('Z')
        df_merged_train['card6'] = df_merged_train['card6'].fillna('A')
        df_merged_train['card1'] = df_merged_train['card1'].fillna('X')
        df_merged_train['card2'] = df_merged_train['card2'].fillna('X')
        df_merged_train['card3'] = df_merged_train['card3'].fillna('X')
        df_merged_train['card5'] = df_merged_train['card5'].fillna('X')
```

```
df_merged_train['P_emaildomain'] = df_merged_train['P_emaildomain'].fillna('B')
df_merged_train['R_emaildomain'] = df_merged_train['R_emaildomain'].fillna('C')
df_merged_train['addr1'] = df_merged_train['addr1'].fillna('D')
df_merged_train['addr2'] = df_merged_train['addr2'].fillna(87)
df_merged_train['dist1'] = df_merged_train['dist1'].fillna('E')
df_merged_train['dist2'] = df_merged_train['dist2'].fillna('G')
medianaddr1 = np.median(df_merged_train[df_merged_train['addr1']!='D']['addr1'])
df_merged_train['addr1'] = np.where(df_merged_train['addr1']=='D',medianaddr1, df_merged
medianaddr2 = np.median(df_merged_train[df_merged_train['dist1']!='E']['dist1'])
df_merged_train['dist1'] = np.where(df_merged_train['dist1']=='E',medianaddr2, df_merged
mediandist3 = np.median(df_merged_train[df_merged_train['dist2']!='G']['dist2'])
df_merged_train['dist2'] = np.where(df_merged_train['dist2']=='G',mediandist3, df_merged
medianaddr1 = np.median(df_merged_train[df_merged_train['card1']!='X']['card1'])
df_merged_train['card1'] = np.where(df_merged_train['card1']=='X',medianaddr1, df_merged
medianaddr1 = np.median(df_merged_train[df_merged_train['card3']!='X']['card3'])
df_merged_train['card3'] = np.where(df_merged_train['card3']=='X',medianaddr1, df_merged
medianaddr1 = np.median(df_merged_train[df_merged_train['card5']!='X']['card5'])
df_merged_train['card5'] = np.where(df_merged_train['card5']=='X',medianaddr1, df_merged
medianaddr1 = np.median(df_merged_train[df_merged_train['card2']!='X']['card2'])
df_merged_train['card2'] = np.where(df_merged_train['card2']=='X',medianaddr1, df_merged
for x in range(1,15):
    df_merged_train['C'+str(x)] = df_merged_train['C'+str(x)].fillna('X')
    median = np.median(df_merged_train['C'+str(x)][df_merged_train['C'+str(x)]!='X'])
    df_merged_train['C'+str(x)] = np.where(df_merged_train['C'+str(x)]=='X',median, df_m
    df_merged_train['C'+str(x)] = df_merged_train['C'+str(x)].values.astype(float)
for x in range(1,16):
    df_merged_train['D'+str(x)] = df_merged_train['D'+str(x)].fillna('X')
    median = np.median(df_merged_train['D'+str(x)][df_merged_train['D'+str(x)]!='X'])
    df_merged_train['D'+str(x)] = np.where(df_merged_train['D'+str(x)]=='X',median, df_m
    df_merged_train['D'+str(x)] = df_merged_train['D'+str(x)].values.astype(float)
for x in range(1,340):
    df_merged_train['V'+str(x)] = df_merged_train['V'+str(x)].fillna('X')
    median = np.median(df_merged_train['V'+str(x)][df_merged_train['V'+str(x)]!='X'])
    df_merged_train['V'+str(x)] = np.where(df_merged_train['V'+str(x)]=='X',median, df_m
    df_merged_train['V'+str(x)] = df_merged_train['V'+str(x)].values.astype(float)
for x in range(1,10):
    df_merged_train['M'+str(x)] = df_merged_train['M'+str(x)].fillna('X')
    MType_test = LabelEncoder()
    MType_labels_test = MType_test.fit_transform(df_merged_train['M'+str(x)])#using exis
    MType_mappings_test = {index: label for index, label in
                          enumerate(MType_test.classes_)}
    df_merged_train['M'+str(x)+'labels'] = MType_labels_test
    df_merged_train['M'+str(x)+'labels'] = df_merged_train['M'+str(x)+'labels'].values.a
deType_train = LabelEncoder()
deType_labels_train = deType_train.fit_transform(df_merged_train['DeviceType'])
deType_mappings_train = {index: label for index, label in
                  enumerate(deType_train.classes_)}
df_merged_train['DeTypeLabels'] = deType_labels_train
```

```
prodType_train = LabelEncoder()
prodType_labels_train = prodType_train.fit_transform(df_merged_train['ProductCD'])
prodType_mappings_train = {index: label for index, label in
                  enumerate(prodType_train.classes_)}
df_merged_train['ProductCDLabels'] = prodType_labels_train
card4Type_train = LabelEncoder()
card4Type_labels_train = card4Type_train.fit_transform(df_merged_train['card4'])
card4Type_mappings_train = {index: label for index, label in
                  enumerate(card4Type_train.classes_)}
df_merged_train['card4Labels'] = card4Type_labels_train
card6Type_train = LabelEncoder()
card6Type_labels_train = card6Type_train.fit_transform(df_merged_train['card6'])
card6Type_mappings_train = {index: label for index, label in
                  enumerate(card6Type_train.classes_)}
df_merged_train['card6Labels'] = card6Type_labels_train
fh_devInfo_train = FeatureHasher(n_features=6, input_type='string')
hashed_features_devInfo_train = fh_devInfo_train.fit_transform(df_merged_train['DeviceIn
hashed_features_devInfo_train = hashed_features_devInfo_train.toarray()
df_merged_train = pd.concat([df_merged_train, pd.DataFrame(hashed_features_devInfo_train
                                                             'DeInfo5'])], axis=1)
fh_pemail_train = FeatureHasher(n_features=6, input_type='string')
hashed_features_pemail_train = fh_pemail_train.fit_transform(df_merged_train['P_emaildom
hashed_features_pemail_train = hashed_features_pemail_train.toarray()
df_merged_train = pd.concat([df_merged_train, pd.DataFrame(hashed_features_pemail_train,
fh_remail_train = FeatureHasher(n_features=6, input_type='string')
hashed_features_remail_train = fh_remail_train.fit_transform(df_merged_train['R_emaildom
hashed_features_remail_train = hashed_features_remail_train.toarray()
df_merged_train = pd.concat([df_merged_train, pd.DataFrame(hashed_features_remail_train,
        columns=['R_emaildomain0', 'R_emaildomain1', 'R_emaildomain2','R_emaildomain3','
for x in range(1,10):
    df_merged_train['id_0'+str(x)] = df_merged_train['id_0'+str(x)].fillna('X')
    median = np.median(df_merged_train['id_0'+str(x)][df_merged_train['id_0'+str(x)]!='X
    df_merged_train['id_0'+str(x)] = np.where(df_merged_train['id_0'+str(x)]=='X',median
    df_merged_train['id_0'+str(x)] = df_merged_train['id_0'+str(x)].values.astype(float)
df_merged_train['id_10'] = df_merged_train['id_10'].fillna('X')
median = np.median(df_merged_train['id_10'][df_merged_train['id_10']!='X'])
df_merged_train['id_10'] = np.where(df_merged_train['id_10']=='X',median, df_merged_trai
df_merged_train['id_10'] = df_merged_train['id_10'].values.astype(float)
df_merged_train['id_11'] = df_merged_train['id_11'].fillna('X')
median = np.median(df_merged_train['id_11'][df_merged_train['id_11']!='X'])
df_merged_train['id_11'] = np.where(df_merged_train['id_11']=='X',median, df_merged_trai
df_merged_train['id_11'] = df_merged_train['id_11'].values.astype(float)
df_merged_train['id_12'] = df_merged_train['id_12'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_train['id_12'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_train['id_12'] = MType_labels_test
```

```
df_merged_train['id_12'] = df_merged_train['id_12'].values.astype(float)
df_merged_train['id_13'] = df_merged_train['id_13'].fillna('X')
median = np.median(df_merged_train['id_13'][df_merged_train['id_13']!='X'])
df_merged_train['id_13'] = np.where(df_merged_train['id_13']=='X',median, df_merged_trai
df_merged_train['id_13'] = df_merged_train['id_13'].values.astype(float)
df_merged_train['id_14'] = df_merged_train['id_14'].fillna('X')
median = np.median(df_merged_train['id_14'][df_merged_train['id_14']!='X'])
df_merged_train['id_14'] = np.where(df_merged_train['id_14']=='X',median, df_merged_trai
df_merged_train['id_14'] = df_merged_train['id_14'].values.astype(float)
df_merged_train['id_15'] = df_merged_train['id_15'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_train['id_15'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_train['id_15labels'] = MType_labels_test
df_merged_train['id_15labels'] = df_merged_train['id_15labels'].values.astype(float)
df_merged_train['id_16'] = df_merged_train['id_16'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_train['id_16'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_train['id_16labels'] = MType_labels_test
df_merged_train['id_16labels'] = df_merged_train['id_16labels'].values.astype(float)
for x in range(17,23):
    df_merged_train['id_'+str(x)] = df_merged_train['id_'+str(x)].fillna('X')
    median = np.median(df_merged_train['id_'+str(x)][df_merged_train['id_'+str(x)]!='X']
    df_merged_train['id_'+str(x)] = np.where(df_merged_train['id_'+str(x)]=='X',median,
    df_merged_train['id_'+str(x)] = df_merged_train['id_'+str(x)].values.astype(float)
df_merged_train['id_23'] = df_merged_train['id_23'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_train['id_23'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_train['id_23labels'] = MType_labels_test
df_merged_train['id_23labels'] = df_merged_train['id_23labels'].values.astype(float)
df_merged_train['id_24'] = df_merged_train['id_24'].fillna('X')
median = np.median(df_merged_train['id_24'][df_merged_train['id_24']!='X'])
df_merged_train['id_24'] = np.where(df_merged_train['id_24']=='X',median, df_merged_trai
df_merged_train['id_24'] = df_merged_train['id_24'].values.astype(float)
df_merged_train['id_25'] = df_merged_train['id_25'].fillna('X')
median = np.median(df_merged_train['id_25'][df_merged_train['id_25']!='X'])
df_merged_train['id_25'] = np.where(df_merged_train['id_25']=='X',median, df_merged_trai
df_merged_train['id_25'] = df_merged_train['id_25'].values.astype(float)
df_merged_train['id_26'] = df_merged_train['id_26'].fillna('X')
median = np.median(df_merged_train['id_26'][df_merged_train['id_26']!='X'])
df_merged_train['id_26'] = np.where(df_merged_train['id_26']=='X',median, df_merged_trai
df_merged_train['id_26'] = df_merged_train['id_26'].values.astype(float)
df_merged_train['id_27'] = df_merged_train['id_27'].fillna('X')
```

```
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_train['id_27'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_train['id_27labels'] = MType_labels_test
df_merged_train['id_27labels'] = df_merged_train['id_27labels'].values.astype(float)

df_merged_train['id_28'] = df_merged_train['id_28'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_train['id_28'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_train['id_28labels'] = MType_labels_test
df_merged_train['id_28labels'] = df_merged_train['id_28labels'].values.astype(float)

df_merged_train['id_29'] = df_merged_train['id_29'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_train['id_29'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_train['id_29labels'] = MType_labels_test
df_merged_train['id_29labels'] = df_merged_train['id_29labels'].values.astype(float)
df_merged_train['id_30'] = df_merged_train['id_30'].fillna('X')
fh_id_train = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_train = fh_id_train.fit_transform(df_merged_train['id_30'])
hashed_features_id_train = hashed_features_id_train.toarray()
df_merged_train = pd.concat([df_merged_train, pd.DataFrame(hashed_features_id_train, col
df_merged_train['id_31'] = df_merged_train['id_31'].fillna('X')
fh_id_train = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_train = fh_id_train.fit_transform(df_merged_train['id_31'])
hashed_features_id_train = hashed_features_id_train.toarray()
df_merged_train = pd.concat([df_merged_train, pd.DataFrame(hashed_features_id_train, col
df_merged_train['id_32'] = df_merged_train['id_32'].fillna('X')
median = np.median(df_merged_train['id_32'][df_merged_train['id_32']!='X'])
df_merged_train['id_32'] = np.where(df_merged_train['id_32']=='X',median, df_merged_trai
df_merged_train['id_32'] = df_merged_train['id_32'].values.astype(float)

df_merged_train['id_33'] = df_merged_train['id_33'].fillna('X')
fh_id_train = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_train = fh_id_train.fit_transform(df_merged_train['id_33'])
hashed_features_id_train = hashed_features_id_train.toarray()
df_merged_train = pd.concat([df_merged_train, pd.DataFrame(hashed_features_id_train, col

df_merged_train['id_34'] = df_merged_train['id_34'].fillna('X')
fh_id_train = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_train = fh_id_train.fit_transform(df_merged_train['id_34'])
hashed_features_id_train = hashed_features_id_train.toarray()
df_merged_train = pd.concat([df_merged_train, pd.DataFrame(hashed_features_id_train, col
for x in range(35,39):
    df_merged_train['id_'+str(x)] = df_merged_train['id_'+str(x)].fillna('X')
    MType_test = LabelEncoder()
    MType_labels_test = MType_test.fit_transform(df_merged_train['id_'+str(x)])#using ea
    MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
    df_merged_train['id_'+str(x)+'labels'] = MType_labels_test
    df_merged_train['id_'+str(x)+'labels'] = df_merged_train['id_'+str(x)+'labels'].valu
df_merged_train = df_merged_train.drop(['id_15','id_16','id_23','id_27','id_28','id_29',
df_merged_train = df_merged_train.drop(['id_30','id_31','id_33','id_34'],axis=1)
df_merged_train = df_merged_train.drop(['DeviceType','ProductCD','card4','card6'],axis=1
df_merged_train = df_merged_train.drop(['DeviceInfo','P_emaildomain','R_emaildomain'],ax
df_merged_train = df_merged_train.drop(['M1','M2','M3','M4','M5','M6','M7','M8','M9'],ax
hour_test = pd.to_datetime(df_merged_train['TransactionDT'], unit='s').dt.hour.values.as
minute_test = pd.to_datetime(df_merged_train['TransactionDT'], unit='s').dt.minute.value
mindt_test = np.min(hour_test+minute_test)
maxdt_test = np.max(hour_test+minute_test)
df_merged_train['TransactionDT']=hour_test+minute_test/maxdt_test
minAmt_test = np.min(df_merged_train['TransactionAmt'])
maxAmt_test = np.max(df_merged_train['TransactionAmt'])
df_merged_train['TransactionAmt']=(df_merged_train['TransactionAmt']-minAmt_test)/(maxAm
df_merged_train['TransactionAmt'] = df_merged_train['TransactionAmt'].values.astype(floa
df_merged_train['addr1'] = df_merged_train['addr1'].values.astype(float)
df_merged_train['dist1'] = df_merged_train['dist1'].values.astype(float)
df_merged_train['dist2'] = df_merged_train['dist2'].values.astype(float)
df_merged_train.to_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\df_merged
```

```
In [ ]: #Cleaning Testing data and storing in a csv after conversion of categorical features to

        import pandas as pd
        import numpy as np
        from sklearn.preprocessing import LabelEncoder
        from sklearn.feature_extraction import FeatureHasher
        from sklearn.linear_model import LinearRegression
        from sklearn import metrics
        df_test = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\test_ident
        df1_test = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\test_tran
        df_merged_test = pd.merge(df_test, df1_test,on='TransactionID',how='outer')
        df_merged_test['DeviceType'] = df_merged_test['DeviceType'].fillna('X')
        df_merged_test['DeviceInfo'] = df_merged_test['DeviceInfo'].fillna('Y')
        df_merged_test['card4'] = df_merged_test['card4'].fillna('Z')
        df_merged_test['card6'] = df_merged_test['card6'].fillna('A')
        df_merged_test['card1'] = df_merged_test['card1'].fillna('X')
        df_merged_test['card2'] = df_merged_test['card2'].fillna('X')
        df_merged_test['card3'] = df_merged_test['card3'].fillna('X')
        df_merged_test['card5'] = df_merged_test['card5'].fillna('X')
        df_merged_test['P_emaildomain'] = df_merged_test['P_emaildomain'].fillna('B')
        df_merged_test['R_emaildomain'] = df_merged_test['R_emaildomain'].fillna('C')
```

```
df_merged_test['addr1'] = df_merged_test['addr1'].fillna('D')
df_merged_test['addr2'] = df_merged_test['addr2'].fillna(87)
df_merged_test['dist1'] = df_merged_test['dist1'].fillna('E')
df_merged_test['dist2'] = df_merged_test['dist2'].fillna('G')
medianaddr1 = np.median(df_merged_test[df_merged_test['addr1']!='D']['addr1'])
df_merged_test['addr1'] = np.where(df_merged_test['addr1']=='D',medianaddr1, df_merged_t
medianaddr2 = np.median(df_merged_test[df_merged_test['dist1']!='E']['dist1'])
df_merged_test['dist1'] = np.where(df_merged_test['dist1']=='E',medianaddr2, df_merged_t
mediandist3 = np.median(df_merged_test[df_merged_test['dist2']!='G']['dist2'])
df_merged_test['dist2'] = np.where(df_merged_test['dist2']=='G',mediandist3, df_merged_t
medianaddr1 = np.median(df_merged_test[df_merged_test['card1']!='X']['card1'])
df_merged_test['card1'] = np.where(df_merged_test['card1']=='X',medianaddr1, df_merged_t
medianaddr1 = np.median(df_merged_test[df_merged_test['card3']!='X']['card3'])
df_merged_test['card3'] = np.where(df_merged_test['card3']=='X',medianaddr1, df_merged_t
medianaddr1 = np.median(df_merged_test[df_merged_test['card5']!='X']['card5'])
df_merged_test['card5'] = np.where(df_merged_test['card5']=='X',medianaddr1, df_merged_t
medianaddr1 = np.median(df_merged_test[df_merged_test['card2']!='X']['card2'])
df_merged_test['card2'] = np.where(df_merged_test['card2']=='X',medianaddr1, df_merged_t
for x in range(1,15):
    df_merged_test['C'+str(x)] = df_merged_test['C'+str(x)].fillna('X')
    median = np.median(df_merged_test['C'+str(x)][df_merged_test['C'+str(x)]!='X'])
    df_merged_test['C'+str(x)] = np.where(df_merged_test['C'+str(x)]=='X',median, df_mer
    df_merged_test['C'+str(x)] = df_merged_test['C'+str(x)].values.astype(float)
for x in range(1,16):
    df_merged_test['D'+str(x)] = df_merged_test['D'+str(x)].fillna('X')
    median = np.median(df_merged_test['D'+str(x)][df_merged_test['D'+str(x)]!='X'])
    df_merged_test['D'+str(x)] = np.where(df_merged_test['D'+str(x)]=='X',median, df_mer
    df_merged_test['D'+str(x)] = df_merged_test['D'+str(x)].values.astype(float)
for x in range(1,340):
    df_merged_test['V'+str(x)] = df_merged_test['V'+str(x)].fillna('X')
    median = np.median(df_merged_test['V'+str(x)][df_merged_test['V'+str(x)]!='X'])
    df_merged_test['V'+str(x)] = np.where(df_merged_test['V'+str(x)]=='X',median, df_mer
    df_merged_test['V'+str(x)] = df_merged_test['V'+str(x)].values.astype(float)
for x in range(1,10):
    df_merged_test['M'+str(x)] = df_merged_test['M'+str(x)].fillna('X')
    MType_test = LabelEncoder()
    MType_labels_test = MType_test.fit_transform(df_merged_test['M'+str(x)])#using exist
    MType_mappings_test = {index: label for index, label in
                  enumerate(MType_test.classes_)}
    df_merged_test['M'+str(x)+'labels'] = MType_labels_test
    df_merged_test['M'+str(x)+'labels'] = df_merged_test['M'+str(x)+'labels'].values.ast
deType_test = LabelEncoder()
deType_labels_test = deType_test.fit_transform(df_merged_test['DeviceType'])
deType_mappings_test = {index: label for index, label in
                  enumerate(deType_test.classes_)}
df_merged_test['DeTypeLabels'] = deType_labels_test
prodType_test = LabelEncoder()
prodType_labels_test = prodType_test.fit_transform(df_merged_test['ProductCD'])
```

```
prodType_mappings_test = {index: label for index, label in
                  enumerate(prodType_test.classes_)}
df_merged_test['ProductCDLabels'] = prodType_labels_test
card4Type_test = LabelEncoder()
card4Type_labels_test = card4Type_test.fit_transform(df_merged_test['card4'])
card4Type_mappings_test = {index: label for index, label in
                  enumerate(card4Type_test.classes_)}
df_merged_test['card4Labels'] = card4Type_labels_test
card6Type_test = LabelEncoder()
card6Type_labels_test = card6Type_test.fit_transform(df_merged_test['card6'])
card6Type_mappings_test = {index: label for index, label in
                  enumerate(card6Type_test.classes_)}
df_merged_test['card6Labels'] = card6Type_labels_test
fh_devInfo_test = FeatureHasher(n_features=6, input_type='string')
hashed_features_devInfo_test = fh_devInfo_test.fit_transform(df_merged_test['DeviceInfo'
hashed_features_devInfo_test = hashed_features_devInfo_test.toarray()
df_merged_test = pd.concat([df_merged_test, pd.DataFrame(hashed_features_devInfo_test, c
                                                             'DeInfo5'])], axis=1)
fh_pemail_test = FeatureHasher(n_features=6, input_type='string')
hashed_features_pemail_test = fh_pemail_test.fit_transform(df_merged_test['P_emaildomain
hashed_features_pemail_test = hashed_features_pemail_test.toarray()
df_merged_test = pd.concat([df_merged_test, pd.DataFrame(hashed_features_pemail_test, co
fh_remail_test = FeatureHasher(n_features=6, input_type='string')
hashed_features_remail_test = fh_remail_test.fit_transform(df_merged_test['R_emaildomain
hashed_features_remail_test = hashed_features_remail_test.toarray()
df_merged_test = pd.concat([df_merged_test, pd.DataFrame(hashed_features_remail_test,
        columns=['R_emaildomain0', 'R_emaildomain1', 'R_emaildomain2','R_emaildomain3','
for x in range(1,10):
    df_merged_test['id_0'+str(x)] = df_merged_test['id_0'+str(x)].fillna('X')
    median = np.median(df_merged_test['id_0'+str(x)][df_merged_test['id_0'+str(x)]!='X']
    df_merged_test['id_0'+str(x)] = np.where(df_merged_test['id_0'+str(x)]=='X',median,
    df_merged_test['id_0'+str(x)] = df_merged_test['id_0'+str(x)].values.astype(float)
df_merged_test['id_10'] = df_merged_test['id_10'].fillna('X')
median = np.median(df_merged_test['id_10'][df_merged_test['id_10']!='X'])
df_merged_test['id_10'] = np.where(df_merged_test['id_10']=='X',median, df_merged_test['
df_merged_test['id_10'] = df_merged_test['id_10'].values.astype(float)
df_merged_test['id_11'] = df_merged_test['id_11'].fillna('X')
median = np.median(df_merged_test['id_11'][df_merged_test['id_11']!='X'])
df_merged_test['id_11'] = np.where(df_merged_test['id_11']=='X',median, df_merged_test['
df_merged_test['id_11'] = df_merged_test['id_11'].values.astype(float)
df_merged_test['id_12'] = df_merged_test['id_12'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_test['id_12'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_test['id_12'] = MType_labels_test
df_merged_test['id_12'] = df_merged_test['id_12'].values.astype(float)
df_merged_test['id_13'] = df_merged_test['id_13'].fillna('X')
```

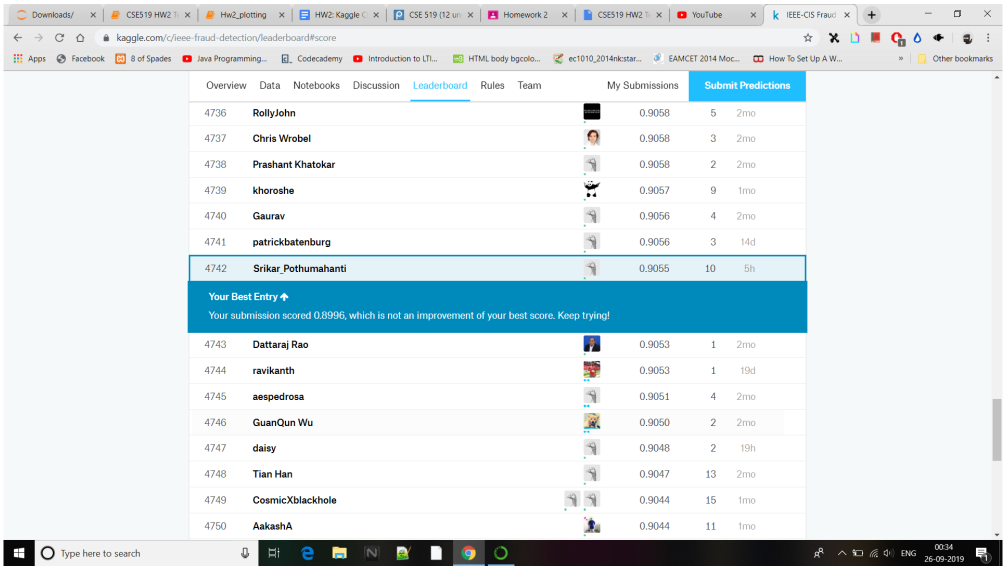
```
median = np.median(df_merged_test['id_13'][df_merged_test['id_13']!='X'])
df_merged_test['id_13'] = np.where(df_merged_test['id_13']=='X',median, df_merged_test['
df_merged_test['id_13'] = df_merged_test['id_13'].values.astype(float)
df_merged_test['id_14'] = df_merged_test['id_14'].fillna('X')
median = np.median(df_merged_test['id_14'][df_merged_test['id_14']!='X'])
df_merged_test['id_14'] = np.where(df_merged_test['id_14']=='X',median, df_merged_test['
df_merged_test['id_14'] = df_merged_test['id_14'].values.astype(float)
df_merged_test['id_15'] = df_merged_test['id_15'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_test['id_15'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_test['id_15labels'] = MType_labels_test
df_merged_test['id_15labels'] = df_merged_test['id_15labels'].values.astype(float)
df_merged_test['id_16'] = df_merged_test['id_16'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_test['id_16'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_test['id_16labels'] = MType_labels_test
df_merged_test['id_16labels'] = df_merged_test['id_16labels'].values.astype(float)
for x in range(17,23):
    df_merged_test['id_'+str(x)] = df_merged_test['id_'+str(x)].fillna('X')
    median = np.median(df_merged_test['id_'+str(x)][df_merged_test['id_'+str(x)]!='X'])
    df_merged_test['id_'+str(x)] = np.where(df_merged_test['id_'+str(x)]=='X',median, df
    df_merged_test['id_'+str(x)] = df_merged_test['id_'+str(x)].values.astype(float)
df_merged_test['id_23'] = df_merged_test['id_23'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_test['id_23'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_test['id_23labels'] = MType_labels_test
df_merged_test['id_23labels'] = df_merged_test['id_23labels'].values.astype(float)
df_merged_test['id_24'] = df_merged_test['id_24'].fillna('X')
median = np.median(df_merged_test['id_24'][df_merged_test['id_24']!='X'])
df_merged_test['id_24'] = np.where(df_merged_test['id_24']=='X',median, df_merged_test['
df_merged_test['id_24'] = df_merged_test['id_24'].values.astype(float)
df_merged_test['id_25'] = df_merged_test['id_25'].fillna('X')
median = np.median(df_merged_test['id_25'][df_merged_test['id_25']!='X'])
df_merged_test['id_25'] = np.where(df_merged_test['id_25']=='X',median, df_merged_test['
df_merged_test['id_25'] = df_merged_test['id_25'].values.astype(float)
df_merged_test['id_26'] = df_merged_test['id_26'].fillna('X')
median = np.median(df_merged_test['id_26'][df_merged_test['id_26']!='X'])
df_merged_test['id_26'] = np.where(df_merged_test['id_26']=='X',median, df_merged_test['
df_merged_test['id_26'] = df_merged_test['id_26'].values.astype(float)
df_merged_test['id_27'] = df_merged_test['id_27'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_test['id_27'])#using existing
```

```
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_test['id_27labels'] = MType_labels_test
df_merged_test['id_27labels'] = df_merged_test['id_27labels'].values.astype(float)

df_merged_test['id_28'] = df_merged_test['id_28'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_test['id_28'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_test['id_28labels'] = MType_labels_test
df_merged_test['id_28labels'] = df_merged_test['id_28labels'].values.astype(float)

df_merged_test['id_29'] = df_merged_test['id_29'].fillna('X')
MType_test = LabelEncoder()
MType_labels_test = MType_test.fit_transform(df_merged_test['id_29'])#using existing
MType_mappings_test = {index: label for index, label in
                      enumerate(MType_test.classes_)}
df_merged_test['id_29labels'] = MType_labels_test
df_merged_test['id_29labels'] = df_merged_test['id_29labels'].values.astype(float)
df_merged_test['id_30'] = df_merged_test['id_30'].fillna('X')
fh_id_test = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_test = fh_id_test.fit_transform(df_merged_test['id_30'])
hashed_features_id_test = hashed_features_id_test.toarray()
df_merged_test = pd.concat([df_merged_test, pd.DataFrame(hashed_features_id_test, column
df_merged_test['id_31'] = df_merged_test['id_31'].fillna('X')
fh_id_test = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_test = fh_id_test.fit_transform(df_merged_test['id_31'])
hashed_features_id_test = hashed_features_id_test.toarray()
df_merged_test = pd.concat([df_merged_test, pd.DataFrame(hashed_features_id_test, column
df_merged_test['id_32'] = df_merged_test['id_32'].fillna('X')
median = np.median(df_merged_test['id_32'][df_merged_test['id_32']!='X'])
df_merged_test['id_32'] = np.where(df_merged_test['id_32']=='X',median, df_merged_test['
df_merged_test['id_32'] = df_merged_test['id_32'].values.astype(float)

df_merged_test['id_33'] = df_merged_test['id_33'].fillna('X')
fh_id_test = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_test = fh_id_test.fit_transform(df_merged_test['id_33'])
hashed_features_id_test = hashed_features_id_test.toarray()
df_merged_test = pd.concat([df_merged_test, pd.DataFrame(hashed_features_id_test, column

df_merged_test['id_34'] = df_merged_test['id_34'].fillna('X')
fh_id_test = FeatureHasher(n_features=3, input_type='string')
hashed_features_id_test = fh_id_test.fit_transform(df_merged_test['id_34'])
hashed_features_id_test = hashed_features_id_test.toarray()
df_merged_test = pd.concat([df_merged_test, pd.DataFrame(hashed_features_id_test, column
for x in range(35,39):
    df_merged_test['id_'+str(x)] = df_merged_test['id_'+str(x)].fillna('X')
```

```
        MType_test = LabelEncoder()
        MType_labels_test = MType_test.fit_transform(df_merged_test['id_'+str(x)])#using exi
        MType_mappings_test = {index: label for index, label in
                         enumerate(MType_test.classes_)}
        df_merged_test['id_'+str(x)+'labels'] = MType_labels_test
        df_merged_test['id_'+str(x)+'labels'] = df_merged_test['id_'+str(x)+'labels'].values
    df_merged_test = df_merged_test.drop(['id_15','id_16','id_23','id_27','id_28','id_29','i
    df_merged_test = df_merged_test.drop(['id_30','id_31','id_33','id_34'],axis=1)
    df_merged_test = df_merged_test.drop(['DeviceType','ProductCD','card4','card6'],axis=1)
    df_merged_test = df_merged_test.drop(['DeviceInfo','P_emaildomain','R_emaildomain'],axis
    df_merged_test = df_merged_test.drop(['M1','M2','M3','M4','M5','M6','M7','M8','M9'],axis
    hour_test = pd.to_datetime(df_merged_test['TransactionDT'], unit='s').dt.hour.values.ast
    minute_test = pd.to_datetime(df_merged_test['TransactionDT'], unit='s').dt.minute.values
    mindt_test = np.min(hour_test+minute_test)
    maxdt_test = np.max(hour_test+minute_test)
    df_merged_test['TransactionDT']=hour_test+minute_test/maxdt_test
    minAmt_test = np.min(df_merged_test['TransactionAmt'])
    maxAmt_test = np.max(df_merged_test['TransactionAmt'])
    df_merged_test['TransactionAmt']=(df_merged_test['TransactionAmt']-minAmt_test)/(maxAmt_
    df_merged_test['TransactionAmt'] = df_merged_test['TransactionAmt'].values.astype(float)
    df_merged_test['addr1'] = df_merged_test['addr1'].values.astype(float)
    df_merged_test['dist1'] = df_merged_test['dist1'].values.astype(float)
    df_merged_test['dist2'] = df_merged_test['dist2'].values.astype(float)
    df_merged_test.to_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\df_merged_
```

```
In [ ]: #Applying models

        import xgboost as xgb
        import pandas as pd
        import numpy as np
        from sklearn import metrics
        df_merged_train = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\df
        #from correlation drop 'V280', 'V332', 'V308',  'V97', 'V323', 'V324', 'V168',  'V96', '
        # 'V295', 'V318', 'V133', 'V213', 'V102', 'V134', 'V128', 'V333', 'V212',
        # 'V179', 'V178', 'V127', 'V306', 'V279', 'V316', 'V211', 'V293', 'V132',
        # 'V101', 'V177', 'V202', 'V322',  'V95', 'V167', 'V143', 'V126', 'V331',
        # 'V164', 'V317', 'V329', 'V204', 'V307', 'V105', 'V298', 'V294', 'id_34_2'

        #second attempt
        #'C2','C11','V330','V57','id_36labels','C1','V106','C10','C8','V217','V232','V219',
        #'V182','C6','V218','V16','V231','V203','C4','V233','V33','V299','V155','V148','C12',
        #'V156','V73','V296','C7','V149','V151','V275','V273','id_28labels','V31','V79',
        #'id_15labels','V160','V145','V144','C14','V153','V58','V51','V154','V150','id_29labels'
        #'id_12','V50','V34','V159','V334','V21','V183','V70','V17','V157','V32','V206','V196',
        #'V42','V254','V266','V18','V248','V91','V269','V72','id_16labels','V22','V69','V158',
        #'id_35labels','V71','V92','V30','V193','V29','V180','V190','V90','V63','V302','V59',
        #'V271','V74','id_34_1','M2labels','V236','id_37labels','V328','V278','DeTypeLabels',
        #'id_38labels','V187','M7labels','V270','M8labels','V104','V221','V84','V249','V321',
        #'id_32','V163','V259','M9labels','V336','V152','id_34_0','V60','V52','V191','V93',
        #'V272','V137','V192','V186','V253','V237','V339','V304','V64'

        y_train = df_merged_train['isFraud']
        df1_train = df_merged_train.drop('isFraud', axis = 1)
        model = LinearRegression().fit(df1_train, y_train)
        y_pred = model.predict(df1_train)
        RMSE = np.sqrt(metrics.mean_squared_error(y_train, y_pred))
        y_pred_test = model.predict(df_test)

        df1_train = df_merged_train.drop(['isFraud', 'V280', 'V332', 'V308',  'V97', 'V323', 'V3
                             'V295', 'V318', 'V133', 'V213', 'V102', 'V134', 'V128', 'V333', 'V2
                             'V179', 'V178', 'V127', 'V306', 'V279', 'V316', 'V211', 'V293', 'V1
                             'V101', 'V177', 'V202', 'V322',  'V95', 'V167', 'V143', 'V126', 'V3
                             'V164', 'V317', 'V329', 'V204', 'V307', 'V105', 'V298', 'V294', 'id
        df1_train['card2'] = df1_train['card2'].values.astype(float)
        df1_train['card3'] = df1_train['card3'].values.astype(float)
        df1_train['card5'] = df1_train['card5'].values.astype(float)
        xgb_model = xgb.XGBRegressor(colsample_bytree=0.4,
                         gamma=0,
                         learning_rate=0.07,
                         max_depth=10,
                         min_child_weight=1.5,
                         n_estimators=100,
                         reg_alpha=0.75,
                         reg_lambda=0.45,
                         subsample=0.6,
                         seed=42)
        xgb_model.fit(df1_train,y_train)
        df_merged_test = pd.read_csv("C:\\Users\\srika\\OneDrive\\Desktop\\Kaggle_challenge\\df_
        df_merged_test['card2'] = df_merged_test['card2'].values.astype(float)
        df_merged_test['card3'] = df_merged_test['card3'].values.astype(float)
        df_merged_test['card5'] = df_merged_test['card5'].values.astype(float)
        df1_test = df_merged_test.drop(['V280', 'V332', 'V308',  'V97', 'V323', 'V324', 'V168',
                             'V295', 'V318', 'V133', 'V213', 'V102', 'V134', 'V128', 'V333', 'V2
                             'V179', 'V178', 'V127', 'V306', 'V279', 'V316', 'V211', 'V293', 'V1
                             'V101', 'V177', 'V202', 'V322',  'V95', 'V167', 'V143', 'V126', 'V3
                             'V164', 'V317', 'V329', 'V204', 'V307', 'V105', 'V298', 'V294', 'id
        y_pred_xgb_test = xgb_model.predict(df_merged_test)
```

For feature extraction, applied label encoding and feature hashing from reference https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63 1. With the given 14 features, the baseline model obtained a score of 0.7931 2. After applying random forest and testing out various depths and estimators, the score slightly increased upto 0.8066 3. Then on making use of 456 features post feature transformations, the score increased with the baseline(linear model) to 0.8584 4. Using random forest, the score increased upto 0.8731 5. Eventually applying XGB model, increased score to 0.9051 6. By using correlation and on removal of half features having a correlation of >0.95, the score increased a



4736	RollyJohn	0.9058	5	2mo
4737	Chris Wrobel	0.9058	3	2mo
4738	Prashant Khatokar	0.9058	2	2mo
4739	khoroshe	0.9057	9	1mo
4740	Gaurav	0.9056	4	2mo
4741	patrickbatenburg	0.9056	3	14d
4742	Srikar_Pothumahanti	0.9055	10	5h

Your Best Entry ↑

Your submission scored 0.8996, which is not an improvement of your best score. Keep trying!

4743	Dattaraj Rao	0.9053	1	2mo
4744	ravikanth	0.9053	1	19d
4745	aespedrosa	0.9051	4	2mo
4746	GuanQun Wu	0.9050	2	2mo
4747	daisy	0.9048	2	19h
4748	Tian Han	0.9047	13	2mo
4749	CosmicXblackhole	0.9044	15	1mo
4750	AakashA	0.9044	11	1mo

alt text

maximum of 0.9055 7. On further reducing the features with high correlation, the model yielded slightly lesser score.

1.7 Part 7 - Final Result

Report the rank, score, number of entries, for your highest rank. Include a snapshot of your best score on the leaderboard as confirmation. Be sure to provide a link to your Kaggle profile. Make sure to include a screenshot of your ranking. Make sure your profile includes your face and affiliation with SBU.

Kaggle Link: https://www.kaggle.com/srikarpothumahanti
Highest Rank: 4742
Score: 0.9055
Number of entries: 10