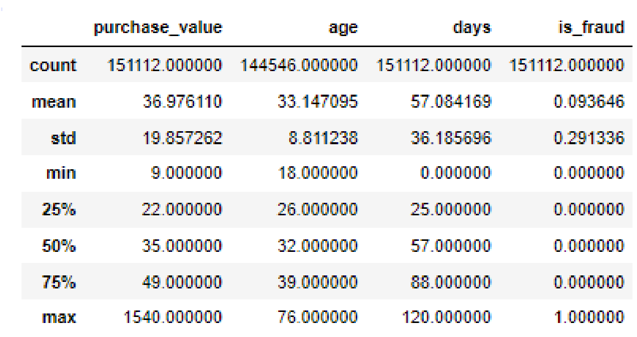
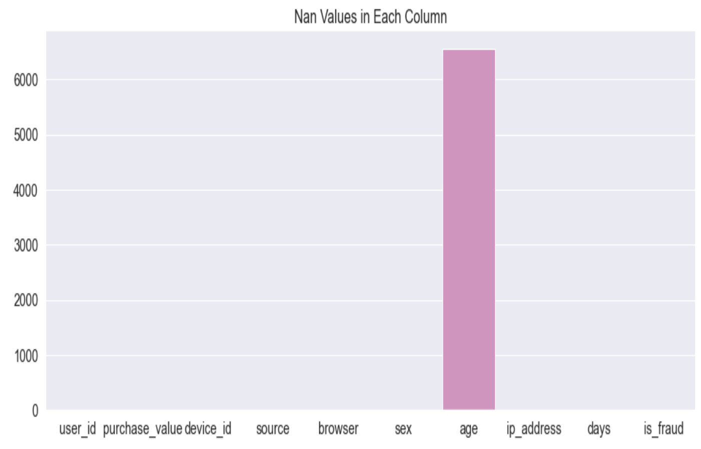
ANLYSIS OF FRAUDALENT TRANSICATIONS

The Fraud data consists of ten columns which are user id, purchase value, device id (from which purchase occurred), Source, Browser, sex (m or f), age, Ip-address, days, is fraud (label). The data set consists of 15112 entries. Extensive analysis was conducted and hidden patterns were seen also there was some useful insights which are mentioned below.

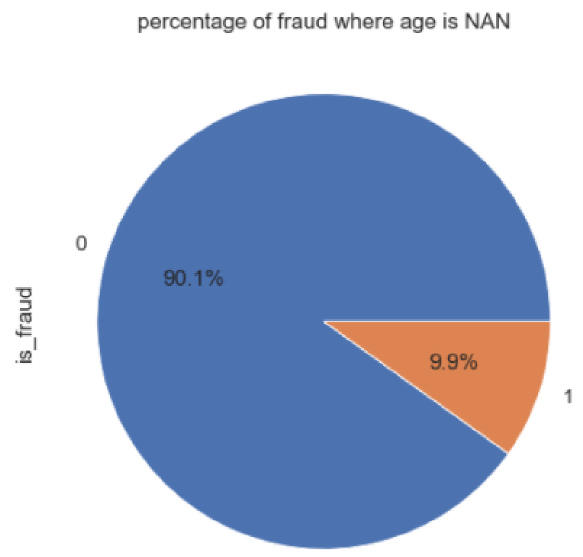


	purchase_value	age	days	is_fraud
count	151112.000000	144546.000000	151112.000000	151112.000000
mean	36.976110	33.147095	57.084169	0.093646
std	19.857262	8.811238	36.185696	0.291336
min	9.000000	18.000000	0.000000	0.000000
25%	22.000000	26.000000	25.000000	0.000000
50%	35.000000	32.000000	57.000000	0.000000
75%	49.000000	39.000000	88.000000	0.000000
max	1540.000000	76.000000	120.000000	1.000000

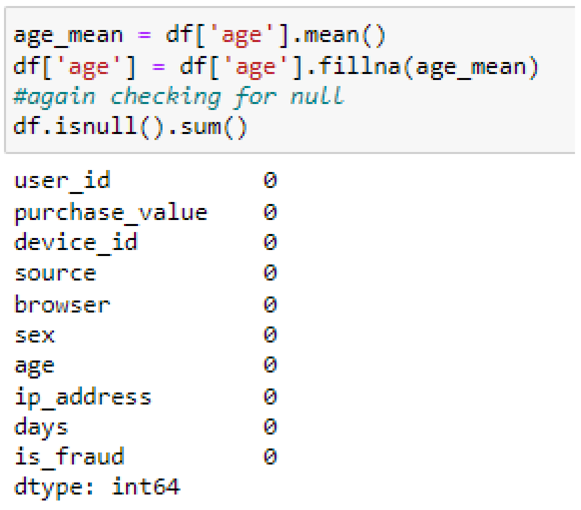
Firstly describing the data to look at the qualitative statistics.



Then, looking at the null values in the data set. In order to make changes accordingly. It can be seen that the age column has some Nan values.



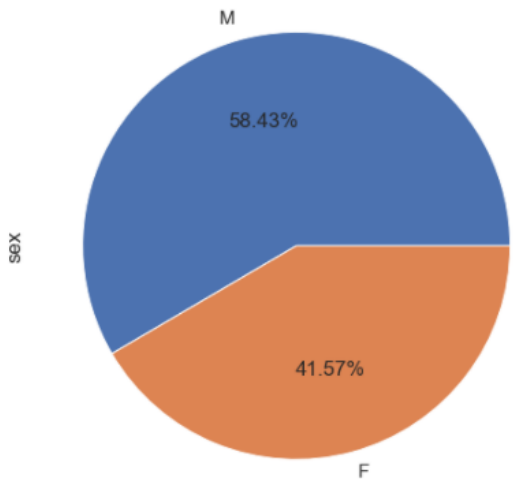
As there are some fraudulent transactions where age is nan so replacing them with the mean in case of analysis with respect to the age.



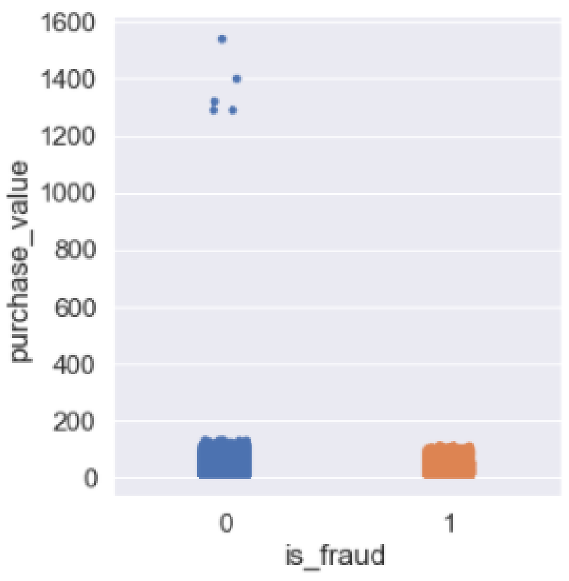
```
age_mean = df['age'].mean()
df['age'] = df['age'].fillna(age_mean)
#again checking for null
df.isnull().sum()
```

```
user_id           0
purchase_value    0
device_id         0
source            0
browser           0
sex               0
age               0
ip_address        0
days              0
is_fraud          0
dtype: int64
```

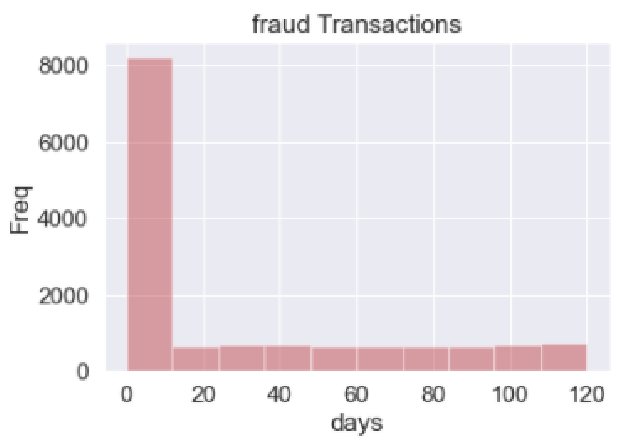
From the above Graph it Can be seen that the Nan values have been replaced and, now there are no more Nan values left.



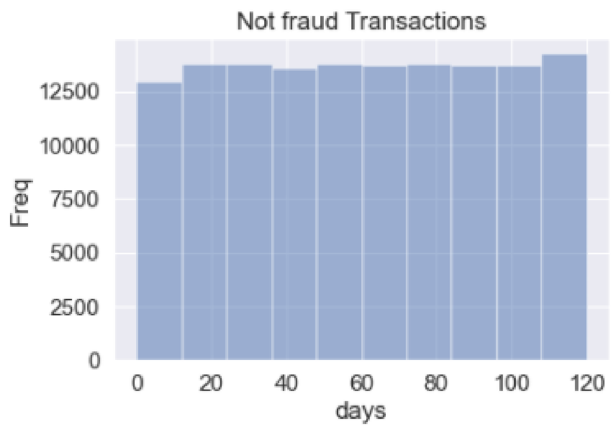
In the above Graph, Loking at the distribution of male and female to understand the division of gender in the given data.



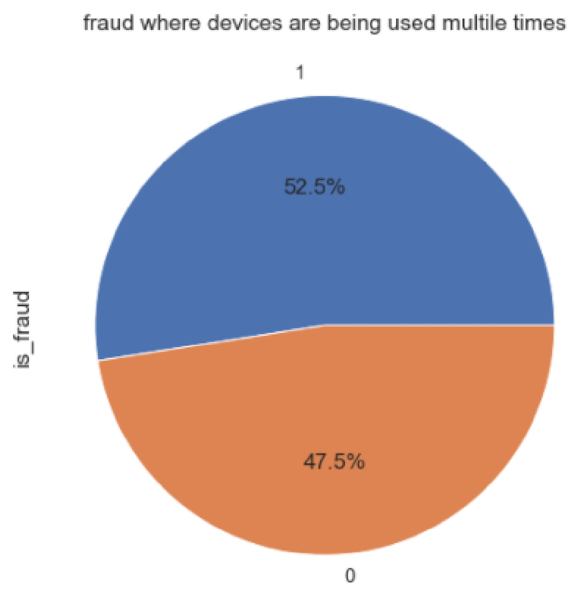
From the above graph it can be seen that there some outliers in the purchase value column where there is no fraud but generally the trend is almost same.



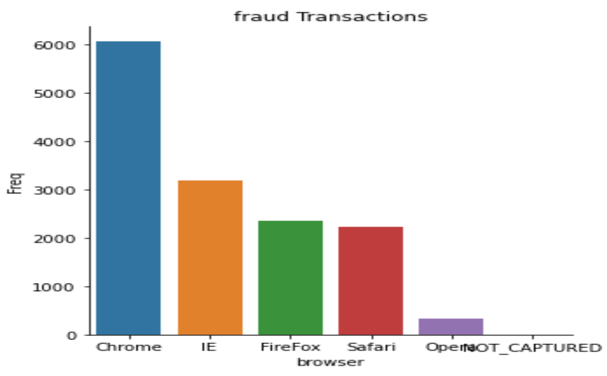
A significant observation was made, as it can be seen clearly that the purchase made on the same day as signup day has more percentage of occurrences in fraudulent data.

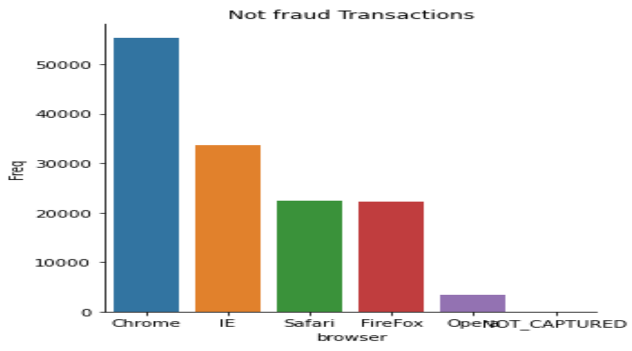


On the other hand in case of Not Fraud Transactions it gives a normal curve.



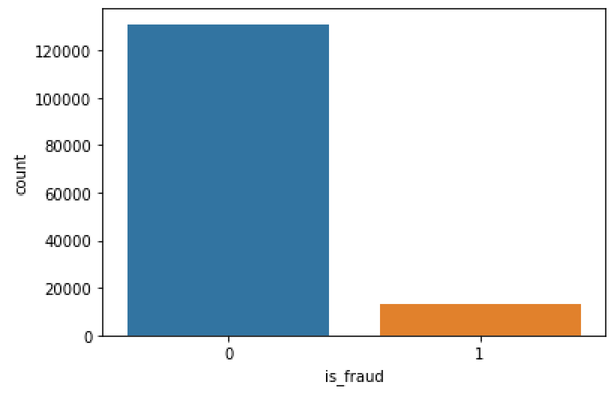
Another Important observation was made that the people who tend to make fraudulent transactions keep on doing that multiple times. That is from the same device. About 52 percent of the people used the same device for fraud transactions.



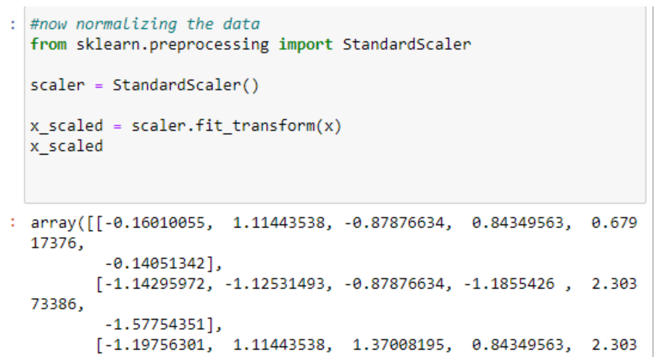


Browsers used seemed to be similar for both fraud and non-fraud transactions there doesn’t seemed to be any problem here. This can be clearly seen from the above graph that the percent difference between the two graphs is approximately zero.

Building Model



I have separated the Y label from the data which will serve as the predictive feature. I have also visualized it to see the distribution of fraudulent and non-fraudulent transactions in the data.



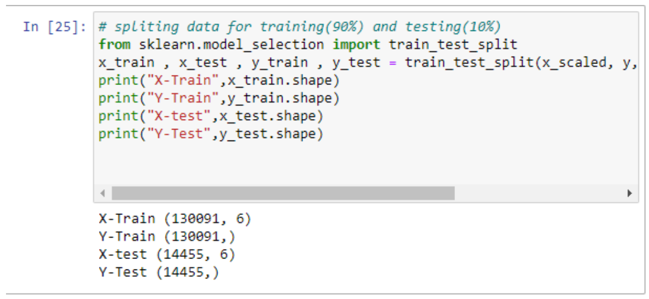
```
#now normalizing the data
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x_scaled = scaler.fit_transform(x)
x_scaled
```

```
array([[-0.16010055,  1.11443538, -0.87876634,  0.84349563,  0.679
17376,
        -0.14051342],
       [-1.14295972, -1.12531493, -0.87876634, -1.1855426 ,  2.303
73386,
        -1.57754351],
       [-1.19756301,  1.11443538,  1.37008195,  0.84349563,  2.303
```

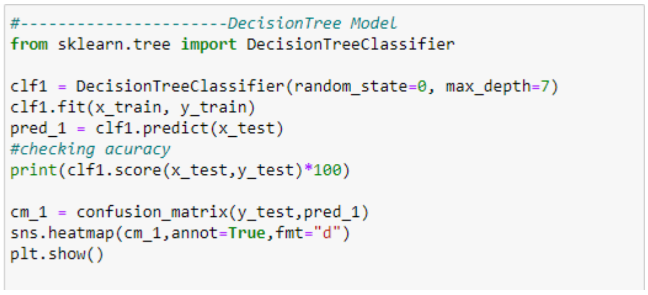
I have also normalized the Data in order to avoid any sort of biasness while training a model on the given data.



```
In [25]: # spliting data for training(90%) and testing(10%)
         from sklearn.model_selection import train_test_split
         x_train , x_test , y_train , y_test = train_test_split(x_scaled, y,
         print("X-Train",x_train.shape)
         print("Y-Train",y_train.shape)
         print("X-test",x_test.shape)
         print("Y-Test",y_test.shape)

         X-Train (130091, 6)
         Y-Train (130091,)
         X-test (14455, 6)
         Y-Test (14455,)
```

In the next step I have divided the data in to training and testing so that afterwards I can calculate the accuracy of the model.



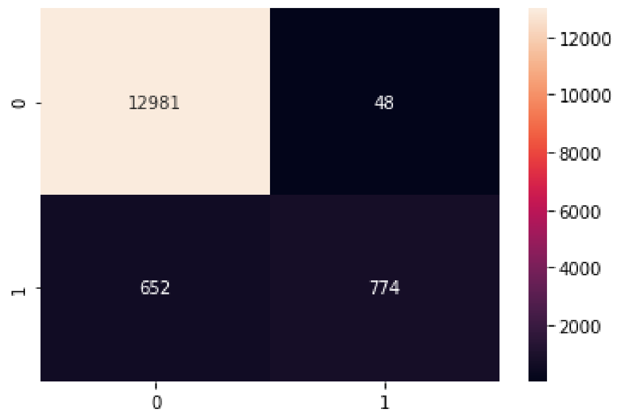
```
#---------------------DecisionTree Model
from sklearn.tree import DecisionTreeClassifier

clf1 = DecisionTreeClassifier(random_state=0, max_depth=7)
clf1.fit(x_train, y_train)
pred_1 = clf1.predict(x_test)
#checking acuracy
print(clf1.score(x_test,y_test)*100)

cm_1 = confusion_matrix(y_test,pred_1)
sns.heatmap(cm_1,annot=True,fmt="d")
plt.show()
```

I have used Decision Tree Model and have trained the data on to it. The accuracy given by the Decision Tree model is as follow.

```
95.15738498789347
```



From the above Graph it Can be Seen that our Model performed very well after Preprocessing the Data and it correctly predicted 13755 and failed to predict only 700 values as can be seen from the Graph.