Online Fraud Prediction Using ML

Project Description:

Fraud detection includes the entire process of institutions to identify fraudulent activities. These activities can be financial as well as other forms such as fraudulent credit card transactions, data theft, or cyberattacks.

Fraud detection is usually done with methods to predict abnormal behaviour, considering predetermined rule flows.
Fraud detection methods are also quite diverse in themselves.

Fraud detection products, which can be customized from the technology used to the workflows defined, to the industry in which they are used, are also developing rapidly to cope with the increasing number of fraud cases because of the rising digitalization.

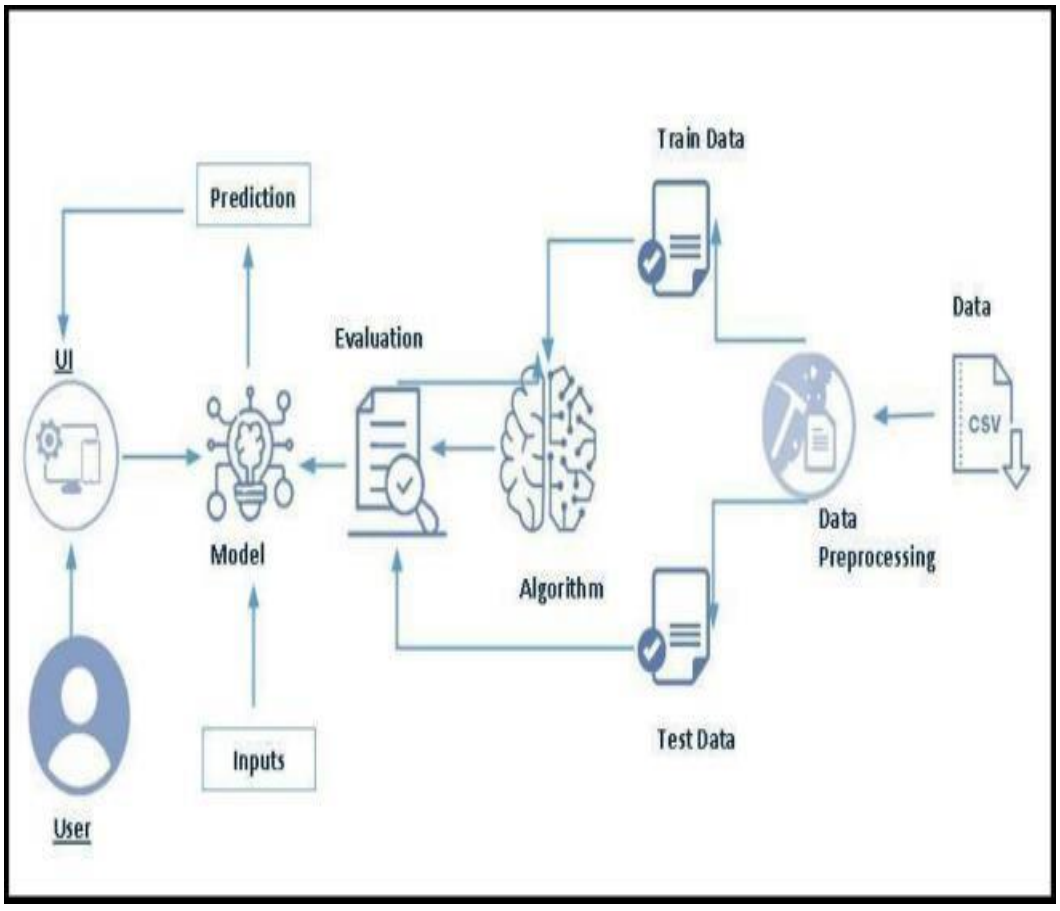
It is very important that the methods developed for fraud detection adapt to the current technology, detect the vulnerabilities before the fraudsters, and do not leave any open doors to the fraudsters.

Many scammers test predetermined patterns and exploit vulnerabilities.

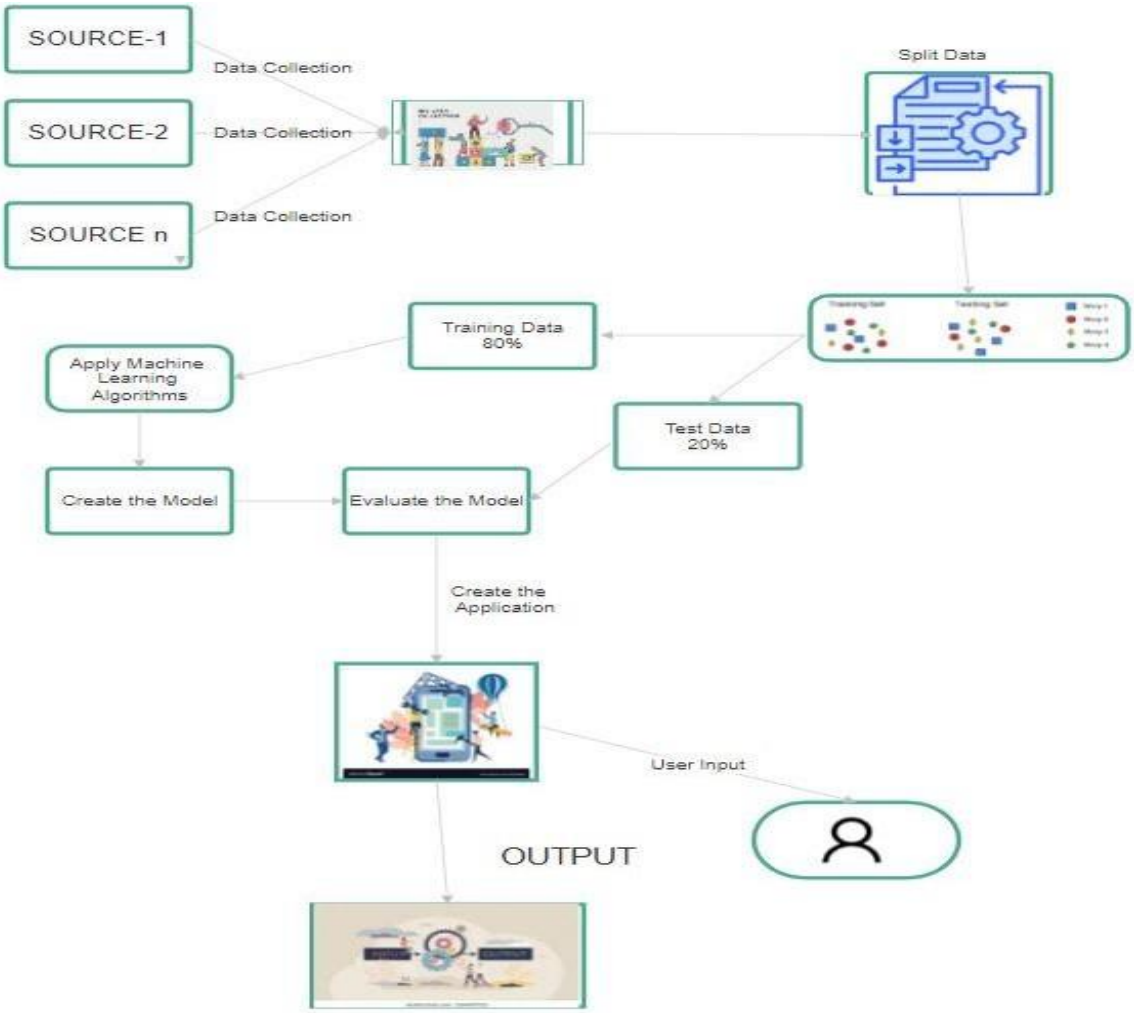
The opportunities provided by the developing technology day by day is the best friend of fraud detection, and it is also the best friend of the fraudsters.

We will be using classification algorithms such as Logistic Regression, KNN, Decision tree, Random forest, AdaBoost and GradientBoost. We will train and test the data with these algorithms. From this the best model is selected and saved in pkl format. We will also be deploying our model locally using Flask.

Technical Architecture:



Flow chart of model:



Pre requisites:

To complete this project, you will require the following softwares, concepts and packages

- Anaconda navigator:
  - By referring some Youtube videos we downloaded anaconda navigator

- Python packages:
  - Open anaconda prompt as administrator
  - Type "pip install numpy" and click enter.
  - Type "pip install pandas" and click enter.
  - Type "pip install scikit-learn" and click enter.
  - Type "pip install matplotlib" and click enter.
  - Type "pip install scipy" and click enter.
  - Type "pip install pickle-mixin" and click enter.
  - Type "pip install seaborn" and click enter.
  - Type "pip install Flask" and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- ML Concepts
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning:https://www.javatpoint.com/unsupervised-machine-learning
  - Regression and classification
    - Logistic regression:https://www.javatpoint.com/logistic-regression-in-machine-learning
    - Decision tree:https://www.javatpoint.com/machine-learning-decision-tree-classification
    - Random forest:https://www.javatpoint.com/machine-learning-random-forest-algorithm
    - KNN:https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

▪ AdaBoost:https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-begineers/

▪ Gradient Boost:https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-begineers/

▪ Evaluation metrics:https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evluation-metrics/

- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Know how to deal with imbalanced target variables.
- Have knowledge on pre-processing the data/transformation techniques and some visualisation concepts before building the model
- Learn how to build a machine learning model and tune it for better performance
- Know how to evaluate the model and deploy it using flask

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- The predictions made by the model is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data collection
  - Download the dataset
- Data pre-processing
  - Handling null values and removing unnecessary columns
- Visualising and analysing data
  - Univariate analysis
  - Bivariate analysis
  - Descriptive analysis
- Model building
  - Handling categorical values
  - Dividing data into train and test sets
  - Sampling the data
  - Dividing train data into train and validation sets
  - Comparing performance of various models
  - Feature Selection
  - Repeat process from dividing data into train and test sets

- Evaluating final model performance
- Save the final model

- Application Building
  - Building HTML pages
  - Build python code

Milestone 1: Data Collection

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used diabetic_data.csv. This data is downloaded from the following research paper:

Link:

https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008

Load the dataset using read_csv() function

```python
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

Python

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
```

Python

```python
data = pd.read_csv("C:/Users/Dell/Downloads/naren.csv")
data.head()
```

Python

Inside the read_csv() function, specify the path to your dataset.

```python
data.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

Next, we will have to see the information pertaining to each of the 50 columns. For that, we will be using info() function:

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype  
---  ------          -----  
 0   step            int64  
 1   type            object 
 2   amount          float64
 3   nameOrig        object 
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object 
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64  
 10  isFlaggedFraud  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

We can use the shape attribute of the data frame to know the shape of our dataset:

```
1 data.shape
```

```
(101766, 50)
```

From the above figure, we can say that our dataset has 101766 rows and 50 columns for describing about data

```
data.describe()
```

Pyth

```
data.isnull().sum()
```

Pyth

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

There is no null columns in dataset.

Milestone 2: Data Pre-processing

We need to pre-process the collected data before gaining insights and building our model. We need to clean the dataset properly in order to fetch good results. This activity includes handling null values and removing unnecessary columns.

Activity 1: Handling Null values and removing unnecessary columns

Though the dataset seems to be completely free of null values, it is not so. We observe from the head and tail of data that a number of fields are filled with '?'. These are nothing but null values. So in order to get the null values count of each column, replace all '?' in data with np.nan and then find the sum of null values.

We observe that 3 columns - weight, payer_code and medical_speciality contain a huge number of null values. So we need to drop these columns.

Also, we need to check for columns that have a very large number of unique values and cannot be bucketed. For this purpose, we will use the nunique() function.

From the above result, we can remove encounter_id and patient_nbr as they have a large amount of unique values. We can also remove examide and citoglipton as they have only 1

unique value and hence do not provide any information. So, let us drop al of these columns using drop() method.

Imbalance dataset.

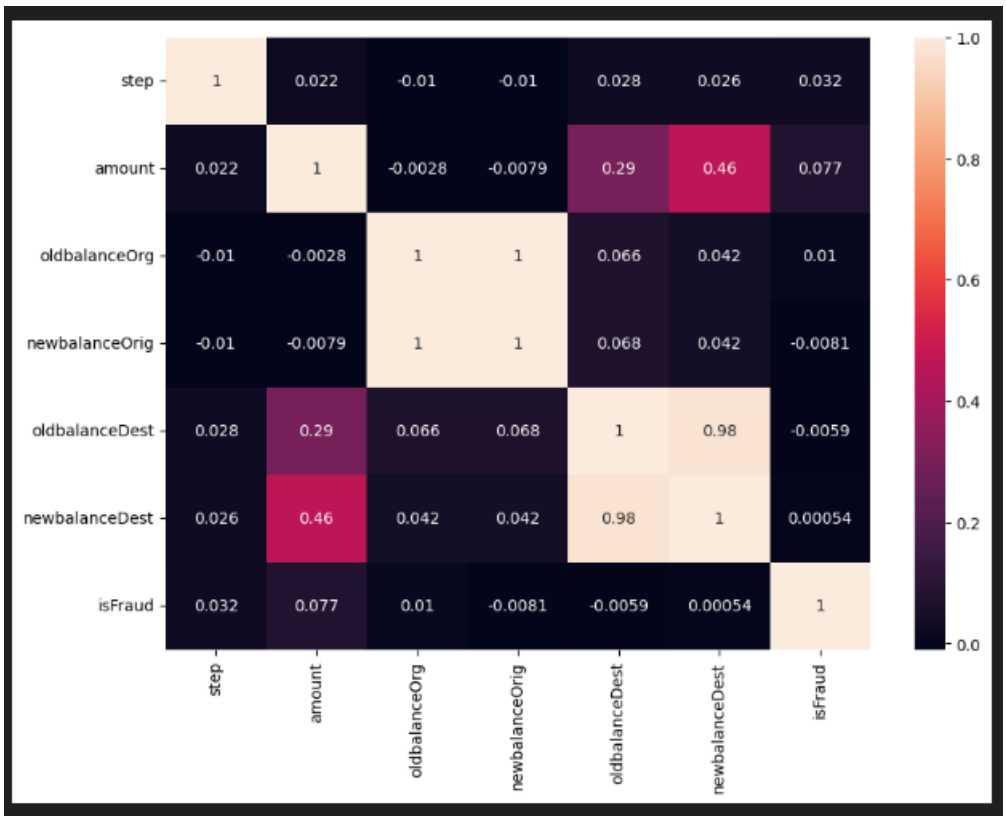
```python
data.drop(['isFlaggedFraud','nameOrig','nameDest'], axis = 1,
```

Python

```python
correlationdata = data.copy()
correlationdata.drop(['type'], axis = 1, inplace = True)

fig = plt.figure(figsize =(10, 7))
sns.heatmap(correlationdata.corr(), annot = True)
plt.show()
```

Python

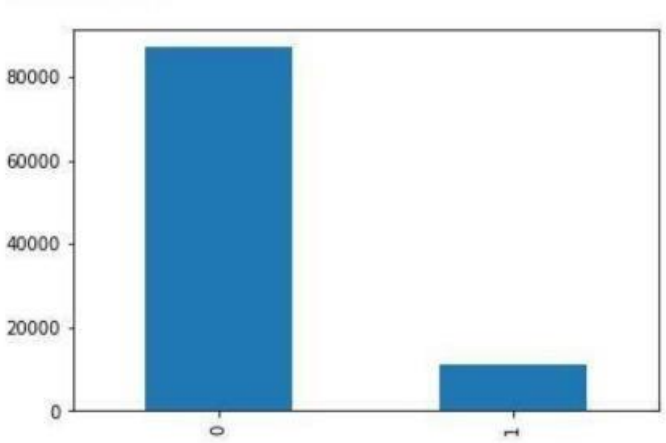


Milestone 3: Data analysis and visualisation

Activity 1: Univariate analysis

In simple words, univariate analysis is understanding the data with a single feature.

Let us first plot the values of our target column – readmitted

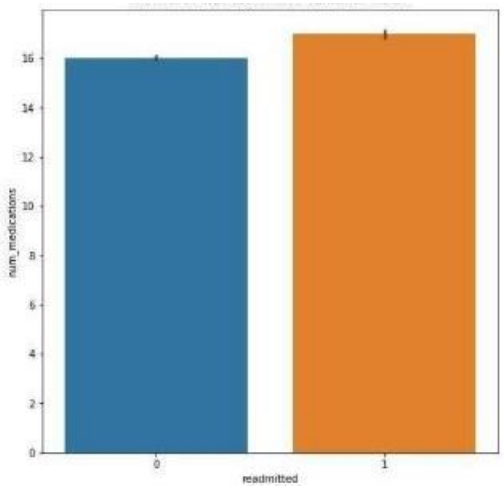


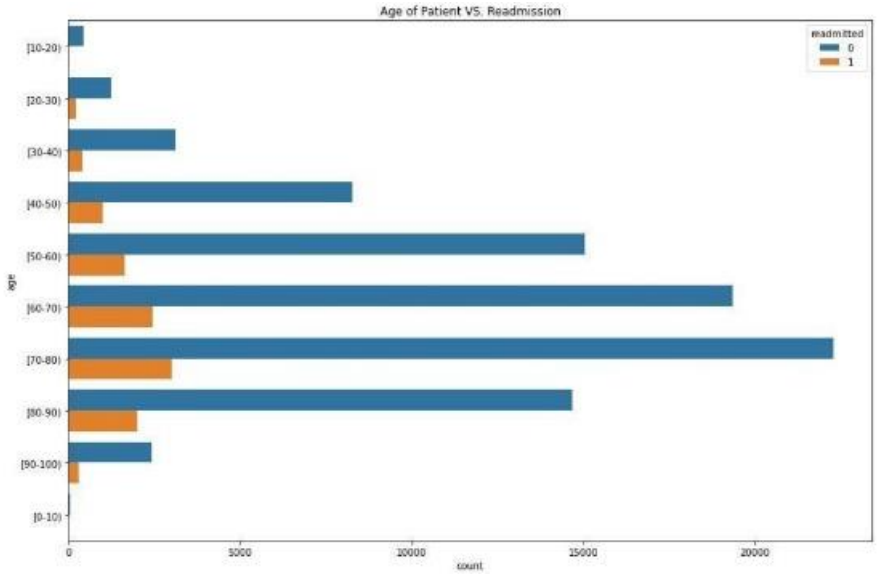
From the above figure, it is observed that the target is quite imbalanced. So, before proceeding with model building, we will be balancing the target data.

Activity 2: Bivariate analysis

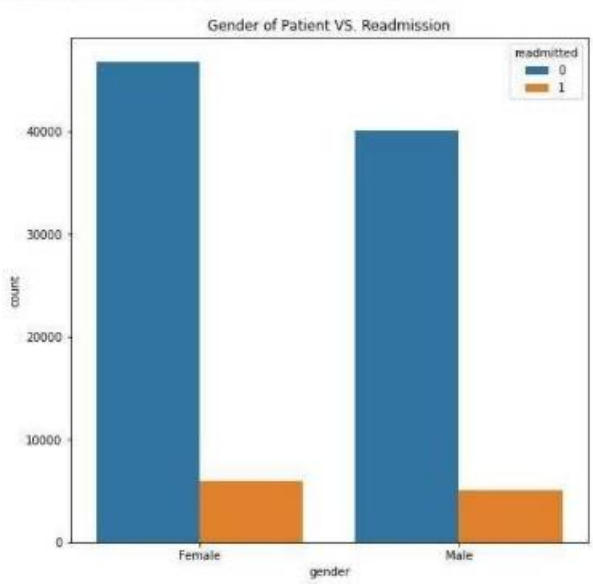
We use bivariate analysis to find the relation between two features. Here we are visualising the relationship of various features with respect to readmitted, which is our target variable.

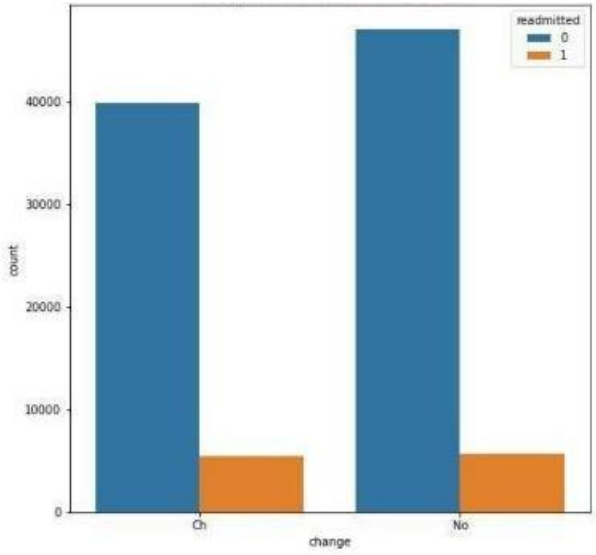
- Number of persons who are affected by fraud activity
- Age and readmitted





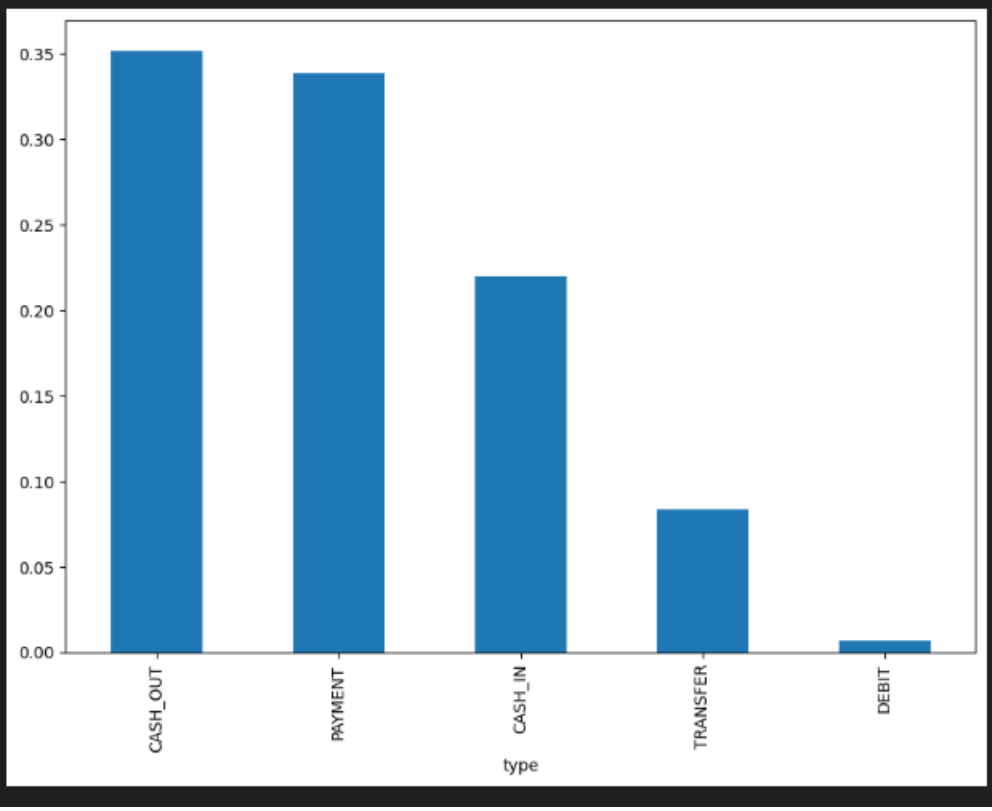
- Gender and readmitted
- Change of affected and not





```python
fig = plt.figure(figsize =(10, 7))
data['type'].value_counts(normalize=True).plot(kind='bar')
plt.show()
```

Python




```python
print("No Frauds Percentage:",data['isFraud'].value_counts()[
print("Frauds Percentage:",data['isFraud'].value_counts()[1]/
```

Python

```
No Frauds Percentage: 99.87091795518198
Frauds Percentage: 0.12908204481801522
```

There is no correlation between columns.

```python
dataf = pd.get_dummies(data = data,columns = ['type'], drop_f
dataf.head()
```

Python

```python
from sklearn.preprocessing import RobustScaler
rscaler = RobustScaler()
scaled_data = rscaler.fit_transform(dataf)
data_sc = pd.DataFrame(scaled_data, columns = dataf.columns)

data_sc.head()
```

Python

Model Training

```python
nonfraud = dataf[dataf['isFraud']==0]
fraud = dataf[dataf['isFraud']==1]
nonfraud = nonfraud.sample(n=8300, random_state = 1)

frauddata = pd.merge(fraud,nonfraud, how = "outer")
```

Python

```python
x = frauddata.drop('isFraud', axis = 1)
y = frauddata['isFraud']
```

Python

```python
print(x)
```

Python

```
     amount  oldbalanceOrg  newbalanceOrig  oldbalanceDest  \
     181.00          181.0            0.00            0.00
     181.00          181.0            0.00        21182.00
    2806.00         2806.0            0.00            0.00
    2806.00         2806.0            0.00        26202.00
   20128.00        20128.0            0.00            0.00
        ...            ...             ...             ...
  153038.88        27133.0       180171.88      1236380.74
  467640.16        19941.0       487581.16       292841.78
    1032.42            0.0            0.00            0.00
 1219321.28            0.0            0.00      5521898.00
  927540.58            0.0            0.00      1041286.50

alanceDest  type_CASH_OUT  type_DEBIT  type_PAYMENT  type_TRANSFER
      0.00          False       False         False           True
      0.00           True       False         False          False
      0.00          False       False         False           True
      0.00           True       False         False          False
      0.00          False       False         False           True
       ...            ...         ...           ...            ...
1083341.86          False       False         False          False
      0.00          False       False         False          False
      0.00          False       False          True          False
5741219.28          False       False         False           True
1968827.07          False       False         False           True

 x 10 columns]
```

```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


x_train,x_test,y_train,y_test = train_test_split(x,y,train_si
```

Python

TESTING:

Logistic Regression

```python
#LogisticRegression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train,y_train)

y_pred = logreg.predict(x_test)
```

Python

Decision Tree Classifier

```python
#DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)

y_pred_dt = dt.predict(x_test)
```

Python

Random Forest Classifier

```python
#RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(x_train,y_train)

y_pred_rf = rf.predict(x_test)
```

Python

Gradient Boosting

```python
#GradientBoosting
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier()
gb.fit(x_train,y_train)

y_pred_gb = gb.predict(x_test)
```

Python

Classification Reports and Evaluations

```
from sklearn.metrics import accuracy_score, classification_re
print("Logistic Regression classification report: \n\n"  ,cla
```

Python

```
Logistic Regression classification report:

              precision    recall  f1-score   support

           0       0.90      0.90      0.90      5779
           1       0.90      0.90      0.90      5781

    accuracy                           0.90     11560
   macro avg       0.90      0.90      0.90     11560
weighted avg       0.90      0.90      0.90     11560
```

```
print("Decision Tree classification report: \n\n"  ,classific
```

Python

```
Decision Tree classification report:

              precision    recall  f1-score   support

           0       0.98      0.98      0.98      5779
           1       0.98      0.98      0.98      5781

    accuracy                           0.98     11560
   macro avg       0.98      0.98      0.98     11560
weighted avg       0.98      0.98      0.98     11560
```

```
print("Random Forest classification report: \n\n"  ,classific
```

Python

```
Random Forest classification report:

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5779
           1       0.99      0.99      0.99      5781

    accuracy                           0.99     11560
   macro avg       0.99      0.99      0.99     11560
weighted avg       0.99      0.99      0.99     11560
```

```
print("Gradient Boosting classification report: \n\n"  ,class
```

Python

```
Gradient Boosting classification report:

              precision    recall  f1-score   support

           0       0.99      0.98      0.99      5779
           1       0.98      0.99      0.99      5781

    accuracy                           0.99     11560
   macro avg       0.99      0.99      0.99     11560
weighted avg       0.99      0.99      0.99     11560
```

```
print("Logistic Regression Accuracy Score:", accuracy_score(y
print("Decision Tree Accuracy Score: ", accuracy_score(y_test
print("Random Forest accuracy score: ", accuracy_score(y_test
print("Gradient Boosting accuracy score: ", accuracy_score(y_
```

26] Python

```
Logistic Regression Accuracy Score: 0.8992214532871973
Decision Tree Accuracy Score:  0.9834775086505191
Random Forest accuracy score:  0.9901384083044983
Gradient Boosting accuracy score:  0.9873702422145328
```

Conclusion

For the fraud prediction, Random Forest Classification model has the highest accuracy score.