```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
import plotly.express as px
%matplotlib inline

sns.set_style("darkgrid")
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 150)
```

```
import os
```

```
os.listdir()
```

```
['.config',
 'target.csv',
 'train.csv',
 'Fraud.csv',
 'fraud_pred.joblib',
 'sample_data']
```

```
df=pd.read_csv('/content/Fraud.csv')
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDes
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0
...	...	...	...	...	...	...	...	.
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65	C435674507	484329.3
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64	M668364942	0.0
1048572	95	PAYMENT	14140.05	C2037964975	20545.00	6404.95	M1355182933	0.0
1048573	95	PAYMENT	10020.05	C1633237354	90605.00	80584.95	M1964992463	0.0
1048574	95	PAYMENT	11450.03	C1264356443	80584.95	69134.92	M677577406	0.0

1048575 rows × 11 columns

```
df.describe()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	2.696617e+01	1.586670e+05	8.740095e+05	8.938089e+05	9.781600e+05	1.114198e+06	1.089097e-03
std	1.562325e+01	2.649409e+05	2.971751e+06	3.008271e+06	2.296780e+06	2.416593e+06	3.298351e-02
min	1.000000e+00	1.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.500000e+01	1.214907e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.000000e+01	7.634333e+04	1.600200e+04	0.000000e+00	1.263772e+05	2.182604e+05	0.000000e+00
75%	3.900000e+01	2.137619e+05	1.366420e+05	1.746000e+05	9.159235e+05	1.149808e+06	0.000000e+00
max	9.500000e+01	1.000000e+07	3.890000e+07	3.890000e+07	4.210000e+07	4.220000e+07	1.000000e+00

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   step            1048575 non-null  int64
 1   type            1048575 non-null  object
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64
 10  isFlaggedFraud  1048575 non-null  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 88.0+ MB
```

```
df.isna().sum()
```

```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
```

```
isFraud           0
isFlaggedFraud    0
dtype: int64
```

```
df=df.dropna()
```

```
df
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDes
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M1979787155	0.0
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M2044282225	0.0
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C553264065	0.0
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C38997010	21182.0
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M1230701703	0.0
...	...	...	...	...	...	...	...	..
1048570	95	CASH_OUT	132557.35	C1179511630	479803.00	347245.65	C435674507	484329.3
1048571	95	PAYMENT	9917.36	C1956161225	90545.00	80627.64	M668364942	0.0
1048572	95	PAYMENT	14140.05	C2037964975	20545.00	6404.95	M1355182933	0.0
1048573	95	PAYMENT	10020.05	C1633237354	90605.00	80584.95	M1964992463	0.0
1048574	95	PAYMENT	11450.03	C1264356443	80584.95	69134.92	M677577406	0.0

1048575 rows × 11 columns

```
df.isna().sum()
```

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

```
df.nunique()
```

```
step                   95
type                    5
amount            1009606
nameOrig          1048317
oldbalanceOrg      391033
newbalanceOrig     440792
```

```
nameDest          449635
oldbalanceDest    590110
newbalanceDest    437054
isFraud                2
isFlaggedFraud         1
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column          Non-Null Count    Dtype  
---  ------          --------------    -----  
 0   step            1048575 non-null  int64  
 1   type            1048575 non-null  object 
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object 
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object 
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64  
 10  isFlaggedFraud  1048575 non-null  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 96.0+ MB
```

```
target=df.isFraud
```

```
df=df.drop(columns='isFraud')
```

```
df=df.drop(columns=['nameOrig','nameDest'])
```

```
df
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFlaggedFr
0	1	PAYMENT	9839.64	170136.00	160296.36	0.00	0.00	
1	1	PAYMENT	1864.28	21249.00	19384.72	0.00	0.00	
2	1	TRANSFER	181.00	181.00	0.00	0.00	0.00	
3	1	CASH_OUT	181.00	181.00	0.00	21182.00	0.00	
4	1	PAYMENT	11668.14	41554.00	29885.86	0.00	0.00	
...	...	...	...	...	...	...	...	

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFlaggedFr
1048570	95	CASH_OUT	132557.35	479803.00	347245.65	484329.37	616886.72	
1048571	95	PAYMENT	9917.36	90545.00	80627.64	0.00	0.00	
1048572	95	PAYMENT	14140.05	20545.00	6404.95	0.00	0.00	
1048573	95	PAYMENT	10020.05	90605.00	80584.95	0.00	0.00	
1048574	95	PAYMENT	11450.03	80584.95	69134.92	0.00	0.00	

1048575 rows × 8 columns

```
df.info()
```

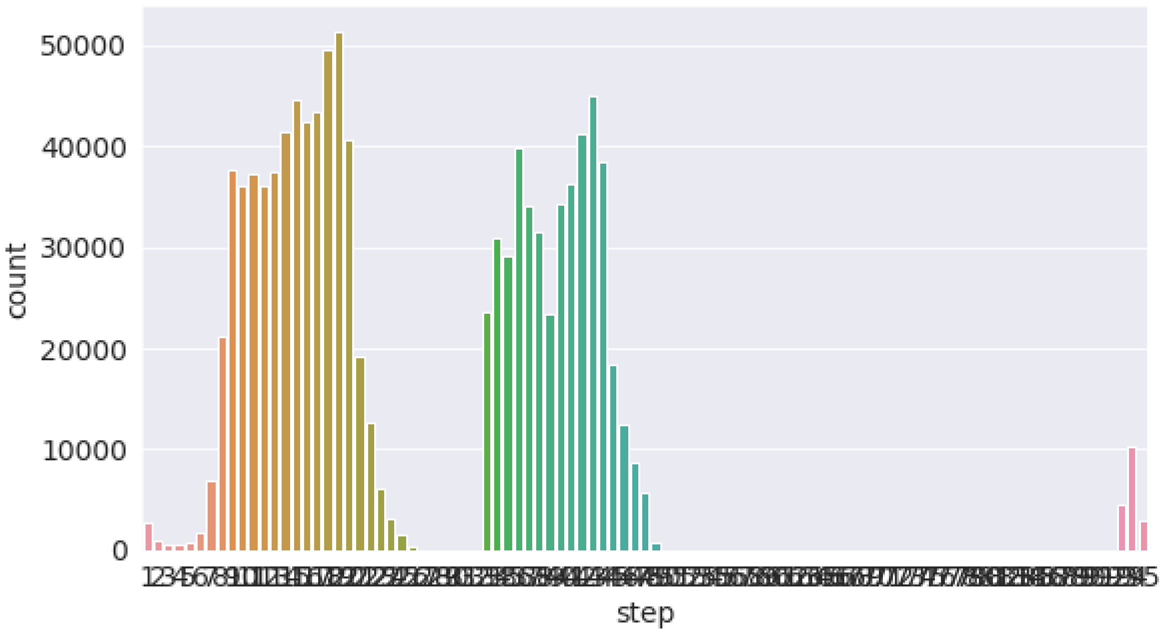
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column          Non-Null Count    Dtype  
---  ------          --------------    -----  
 0   step            1048575 non-null  int64  
 1   type            1048575 non-null  object 
 2   amount          1048575 non-null  float64
 3   oldbalanceOrg   1048575 non-null  float64
 4   newbalanceOrig  1048575 non-null  float64
 5   oldbalanceDest  1048575 non-null  float64
 6   newbalanceDest  1048575 non-null  float64
 7   isFlaggedFraud  1048575 non-null  int64  
dtypes: float64(5), int64(2), object(1)
memory usage: 72.0+ MB
```

```
sns.countplot(df.step)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.


<matplotlib.axes._subplots.AxesSubplot at 0x7fbc28a63310>
```



```
sns.barplot(df,x='type',y=target,color='step')
```

```
px.scatter(df,x='type',y='amount',color=target)
```

```
px.scatter(df,x='amount',y='newbalanceDest',color=target)
```

```
px.scatter(df,x='amount',y='oldbalanceDest',color=target)
```

```
px.scatter(df,x='amount',y='newbalanceOrig',color=target)
```

```
px.scatter(df,x='amount',y='oldbalanceOrg',color=target)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-17-4cc18c4f80c8> in <module>()
----> 1 px.scatter(df,x='amount',y='oldbalanceOrg',color=target)

/usr/local/lib/python3.7/dist-packages/IPython/core/displayhook.py in __call__(self,
result)
    244             self.start_displayhook()
    245             self.write_output_prompt()
--> 246             format_dict, md_dict = self.compute_format_data(result)
    247             self.update_user_ns(result)
    248             self.fill_exec_result(result)

/usr/local/lib/python3.7/dist-packages/IPython/core/displayhook.py in
compute_format_data(self, result)
    148
    149         """
--> 150         return self.shell.display_formatter.format(result)
```

```
    151 
    152     # This can be set to True by the write_output_prompt method in a subclass

/usr/local/lib/python3.7/dist-packages/IPython/core/formatters.py in format(self, obj, include, exclude)
    146         md_dict = {}
    147 
--> 148         if self.ipython_display_formatter(obj):
    149             # object handled itself, don't proceed
    150             return {}, {}

<decorator-gen-4> in __call__(self, obj)

/usr/local/lib/python3.7/dist-packages/IPython/core/formatters.py in catch_format_error(method, self, *args, **kwargs)
    215     """show traceback on failed format call"""
    216     try:
--> 217         r = method(self, *args, **kwargs)
    218     except NotImplementedError:
    219         # don't warn on NotImplementedErrors

/usr/local/lib/python3.7/dist-packages/IPython/core/formatters.py in __call__(self, obj)
    913             method = get_real_method(obj, self.print_method)
    914             if method is not None:
--> 915                 method()
    916                 return True
    917 

/usr/local/lib/python3.7/dist-packages/plotly/basedatatypes.py in _ipython_display_(self)
    842 
    843         if pio.renderers.render_on_display and pio.renderers.default:
--> 844             pio.show(self)
    845         else:
    846             print(repr(self))

/usr/local/lib/python3.7/dist-packages/plotly/io/_renderers.py in show(fig, renderer, validate, **kwargs)
    387 
    388     # Mimetype renderers
--> 389     bundle = renderers._build_mime_bundle(fig_dict, renderers_string=renderer, **kwargs)
    390     if bundle:
    391         if not ipython_display:

/usr/local/lib/python3.7/dist-packages/plotly/io/_renderers.py in _build_mime_bundle(self, fig_dict, renderers_string, **kwargs)
    295                             setattr(renderer, k, v)
    296 
```

```
--> 297                 bundle.update(renderer.to_mimebundle(fig_dict))
    298 
    299         return bundle

/usr/local/lib/python3.7/dist-packages/plotly/io/_base_renderers.py in
to_mimebundle(self, fig_dict)
    389             default_width="100%",
    390             default_height=525,
--> 391             validate=False,
    392         )
    393 

/usr/local/lib/python3.7/dist-packages/plotly/io/_html.py in to_html(fig, config,
auto_play, include_plotlyjs, include_mathjax, post_script, full_html, animation_opts,
default_width, default_height, validate, div_id)
    144 
    145     # ## Serialize figure ##
--> 146     jdata = to_json_plotly(fig_dict.get("data", []))
    147     jlayout = to_json_plotly(fig_dict.get("layout", {}))
    148 

/usr/local/lib/python3.7/dist-packages/plotly/io/_json.py in
to_json_plotly(plotly_object, pretty, engine)
    122         from _plotly_utils.utils import PlotlyJSONEncoder
    123 
--> 124         return json.dumps(plotly_object, cls=PlotlyJSONEncoder, **opts)
    125     elif engine == "orjson":
    126         JsonConfig.validate_orjson()

/usr/lib/python3.7/json/__init__.py in dumps(obj, skipkeys, ensure_ascii,
check_circular, allow_nan, cls, indent, separators, default, sort_keys, **kw)
    236         check_circular=check_circular, allow_nan=allow_nan, indent=indent,
    237         separators=separators, default=default, sort_keys=sort_keys,
--> 238         **kw).encode(obj)
    239 
    240 

/usr/local/lib/python3.7/dist-packages/_plotly_utils/utils.py in encode(self, o)
     57         """
     58         # this will raise errors in a normal-expected way
---> 59         encoded_o = super(PlotlyJSONEncoder, self).encode(o)
     60         # Brute force guessing whether NaN or Infinity values are in the string
     61         # We catch false positive cases (e.g. strings such as titles, labels
etc.)

/usr/lib/python3.7/json/encoder.py in encode(self, o)
    197         # exceptions aren't as detailed.  The list call should be roughly
    198         # equivalent to the PySequence_Fast that ''.join() would do.
--> 199         chunks = self.iterencode(o, _one_shot=True)
```

```
    200         if not isinstance(chunks, (list, tuple)):
    201             chunks = list(chunks)

/usr/lib/python3.7/json/encoder.py in iterencode(self, o, _one_shot)
    255                 self.key_separator, self.item_separator, self.sort_keys,
    256                 self.skipkeys, _one_shot)
--> 257         return _iterencode(o, 0)
    258 
    259 def _make_iterencode(markers, _default, _encoder, _indent, _floatstr,

KeyboardInterrupt:
```

```
px.scatter(df,x='amount',y='type',color=target)
```

```
numeric_cols = df.select_dtypes(include=np.number).columns.tolist()
categorical_cols = df.select_dtypes('object').columns.tolist()
```

```
numeric_cols
```

```
categorical_cols
```

```
sns.pairplot(df,hue='type')
```

```
df.corr()
```

```
df.isFlaggedFraud.nunique()
```

```
1
```

```
df=df.drop(columns='isFlaggedFraud')
```

```
df.corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
step	1.000000	-0.025996	-0.006780	-0.007180	-0.002251	-0.019503
amount	-0.025996	1.000000	0.004864	-0.001133	0.215558	0.311936
oldbalanceOrg	-0.006780	0.004864	1.000000	0.999047	0.093305	0.064049
newbalanceOrig	-0.007180	-0.001133	0.999047	1.000000	0.095182	0.063725
oldbalanceDest	-0.002251	0.215558	0.093305	0.095182	1.000000	0.978403
newbalanceDest	-0.019503	0.311936	0.064049	0.063725	0.978403	1.000000

```
target.isna().sum()
```

```
0
```

```
target.nunique()
```

```
2
```

```
from sklearn.impute import SimpleImputer
```

```
imputer=SimpleImputer(strategy='mean')
```

```
imputer.fit(df[numeric_cols])
```

```
SimpleImputer()
```

```
df[numeric_cols]=imputer.transform(df[numeric_cols])
```

```

```

```
from sklearn.preprocessing import OneHotEncoder
```

```
encoder=OneHotEncoder(sparse=False, handle_unknown='ignore').fit(df[categorical_cols])
```

```
encoded_cols=list(encoder.get_feature_names(categorical_cols))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:

Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and
will be removed in 1.2. Please use get_feature_names_out instead.
```

```

```

```
df[encoded_cols]=encoder.transform(df[categorical_cols])
```

```
df
```

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_
0	1.0	PAYMENT	9839.64	170136.00	160296.36	0.00	0.00	(
1	1.0	PAYMENT	1864.28	21249.00	19384.72	0.00	0.00	(
2	1.0	TRANSFER	181.00	181.00	0.00	0.00	0.00	(
3	1.0	CASH_OUT	181.00	181.00	0.00	21182.00	0.00	(
4	1.0	PAYMENT	11668.14	41554.00	29885.86	0.00	0.00	(
...	...	...	...	...	...	...	...	
1048570	95.0	CASH_OUT	132557.35	479803.00	347245.65	484329.37	616886.72	(

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_
1048571	95.0	PAYMENT	9917.36	90545.00	80627.64	0.00	0.00	(
1048572	95.0	PAYMENT	14140.05	20545.00	6404.95	0.00	0.00	(
1048573	95.0	PAYMENT	10020.05	90605.00	80584.95	0.00	0.00	(
1048574	95.0	PAYMENT	11450.03	80584.95	69134.92	0.00	0.00	(

1048575 rows × 12 columns

```
df=df.drop(columns='type')
```

```
df
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_IN	type_CAS
0	1.0	9839.64	170136.00	160296.36	0.00	0.00	0.0	
1	1.0	1864.28	21249.00	19384.72	0.00	0.00	0.0	
2	1.0	181.00	181.00	0.00	0.00	0.00	0.0	
3	1.0	181.00	181.00	0.00	21182.00	0.00	0.0	
4	1.0	11668.14	41554.00	29885.86	0.00	0.00	0.0	
...	...	...	...	...	...	...	...	
1048570	95.0	132557.35	479803.00	347245.65	484329.37	616886.72	0.0	
1048571	95.0	9917.36	90545.00	80627.64	0.00	0.00	0.0	
1048572	95.0	14140.05	20545.00	6404.95	0.00	0.00	0.0	
1048573	95.0	10020.05	90605.00	80584.95	0.00	0.00	0.0	
1048574	95.0	11450.03	80584.95	69134.92	0.00	0.00	0.0	

1048575 rows × 11 columns

```
df.to_csv('train.csv', index=None)
```

```
target.to_csv('target.csv', index=None)
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler=MinMaxScaler().fit(df[numeric_cols])
```

```
df.describe().loc[['min', 'max']]
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_IN	type_CASH_
min	1.0	0.1	0.0	0.0	0.0	0.0	0.0	
max	95.0	10000000.0	38900000.0	38900000.0	42100000.0	42200000.0	1.0	

```
df[numeric_cols]=scaler.transform(df[numeric_cols])
```

```
df
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_IN	type_CAS
0	0.0	0.000984	0.004374	0.004121	0.000000	0.000000	0.0	
1	0.0	0.000186	0.000546	0.000498	0.000000	0.000000	0.0	
2	0.0	0.000018	0.000005	0.000000	0.000000	0.000000	0.0	
3	0.0	0.000018	0.000005	0.000000	0.000503	0.000000	0.0	
4	0.0	0.001167	0.001068	0.000768	0.000000	0.000000	0.0	
...	...	...	...	...	...	...	...	
1048570	1.0	0.013256	0.012334	0.008927	0.011504	0.014618	0.0	
1048571	1.0	0.000992	0.002328	0.002073	0.000000	0.000000	0.0	
1048572	1.0	0.001414	0.000528	0.000165	0.000000	0.000000	0.0	
1048573	1.0	0.001002	0.002329	0.002072	0.000000	0.000000	0.0	
1048574	1.0	0.001145	0.002072	0.001777	0.000000	0.000000	0.0	

1048575 rows × 11 columns

```
df=pd.DataFrame(df)
```

```
df
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_IN	type_CAS
0	0.0	0.000984	0.004374	0.004121	0.000000	0.000000	0.0	
1	0.0	0.000186	0.000546	0.000498	0.000000	0.000000	0.0	
2	0.0	0.000018	0.000005	0.000000	0.000000	0.000000	0.0	
3	0.0	0.000018	0.000005	0.000000	0.000503	0.000000	0.0	
4	0.0	0.001167	0.001068	0.000768	0.000000	0.000000	0.0	
...	...	...	...	...	...	...	...	
1048570	1.0	0.013256	0.012334	0.008927	0.011504	0.014618	0.0	
1048571	1.0	0.000992	0.002328	0.002073	0.000000	0.000000	0.0	
1048572	1.0	0.001414	0.000528	0.000165	0.000000	0.000000	0.0	
1048573	1.0	0.001002	0.002329	0.002072	0.000000	0.000000	0.0	
1048574	1.0	0.001145	0.002072	0.001777	0.000000	0.000000	0.0	

1048575 rows × 11 columns

```
from sklearn.model_selection import train_test_split
```

```
train_df,val_df,train_target,val_target = train_test_split(df,target, test_size=0.2, ra
```

```
train_df
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_IN	type_C
408561	0.180851	0.026837	0.000000	0.000000	0.007647	0.009356	0.0	
70143	0.085106	0.000383	0.000171	0.000073	0.000000	0.000000	0.0	
708782	0.382979	0.043902	0.012414	0.001128	0.011658	0.022034	0.0	
572694	0.244681	0.001354	0.000024	0.000000	0.000000	0.000000	0.0	
774181	0.404255	0.001456	0.000000	0.000000	0.000000	0.000000	0.0	
...	...	...	...	...	...	...	...	
259178	0.138298	0.000608	0.000000	0.000000	0.000000	0.000000	0.0	
365838	0.170213	0.003268	0.000712	0.000000	0.000000	0.000000	0.0	
131932	0.106383	0.024626	0.000638	0.000000	0.006028	0.013160	0.0	
671155	0.372340	0.051297	0.000520	0.000000	0.000520	0.012675	0.0	
121958	0.106383	0.001006	0.000000	0.000000	0.000000	0.000000	0.0	

838860 rows × 11 columns

```
val_df
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	type_CASH_IN	type_C
781974	0.404255	0.057475	0.002727	0.000000	0.000000	0.013620	0.0	
937737	0.446809	0.002112	0.269923	0.269923	0.063877	0.063225	1.0	
907828	0.446809	0.002655	0.000529	0.001211	0.000000	0.000000	1.0	
784628	0.404255	0.000782	0.000000	0.000000	0.000000	0.000000	0.0	
662460	0.372340	0.031291	0.000000	0.000000	0.031138	0.038479	0.0	
...	...	...	...	...	...	...	...	
673443	0.372340	0.023446	0.000458	0.000000	0.000336	0.005891	0.0	
656736	0.361702	0.020696	0.241984	0.247305	0.083394	0.078292	1.0	
858501	0.425532	0.005826	0.000000	0.000000	0.018027	0.019365	0.0	
617079	0.351064	0.000928	0.000000	0.000000	0.000000	0.000000	0.0	
487559	0.191489	0.019070	0.000000	0.000000	0.076996	0.096311	0.0	

209715 rows × 11 columns

```
train_target
```

```
408561    0
70143     0
708782    0
572694    0
774181    0
         ..
259178    0
365838    0
131932    0
671155    0
```

```
121958    0
Name: isFraud, Length: 838860, dtype: int64
```

```
val_target
```

```
781974    0
937737    0
907828    0
784628    0
662460    0
         ..
673443    0
656736    0
858501    0
617079    0
487559    0
Name: isFraud, Length: 209715, dtype: int64
```

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(solver='liblinear')
```

```
model.fit(train_df,train_target)
```

```
LogisticRegression(solver='liblinear')
```

```
model.score(train_df,train_target)
```

```
0.9990069856710297
```

```
model.score(val_df,val_target)
```

```
0.9990177145173211
```

```
print(model.coef_.tolist())
```

```
[[4.50727814896816, 11.173462702038702, 9.991109197153355, -2.7004587618657347,
-6.291399268844971, -8.964961920674169, -5.129353430848819, 0.6000565407037542,
-1.3056909929013092, -4.32542831211062, 1.58213497861857]]
```

```
print(model.intercept_)
```

```
[-8.57828122]
```

```
train_probs = model.predict_proba(train_df)
train_probs
```

```
array([[9.99084573e-01, 9.15426746e-04],
       [9.99996326e-01, 3.67367572e-06],
       [9.97298906e-01, 2.70109438e-03],
       ...,
       [9.99372472e-01, 6.27527729e-04],
       [9.97095324e-01, 2.90467557e-03],
       [9.99995935e-01, 4.06547358e-06]])
```

```
val_probs = model.predict_proba(val_df)
val_probs
```

```
array([[9.90307984e-01, 9.69201612e-03],
       [9.99976790e-01, 2.32096184e-05],
       [9.99991386e-01, 8.61440348e-06],
       ...,
       [9.95033629e-01, 4.96637072e-03],
       [9.99987763e-01, 1.22374362e-05],
       [9.99738782e-01, 2.61218311e-04]])
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
confusion_matrix(model.predict(val_df), val_target, normalize='true')
```

```
array([[9.9901763e-01, 9.8236980e-04],
       [0.0000000e+00, 1.0000000e+00]])
```

```
! pip install xgboost --upgrade --quiet
from xgboost import XGBClassifier
```

```
     |████████████████████████████████| 173.6 MB 7.0 kB/s
```

```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=200,max_depth=10,learning_
```

```
model.fit(train_df, train_target)
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].


[09:31:22] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default
evaluation metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.3, max_delta_step=0,
              max_depth=10, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=200, n_jobs=-1,
              num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
model.score(train_df, train_target)
```

```
1.0
```

```
model.score(val_df,val_target)
```

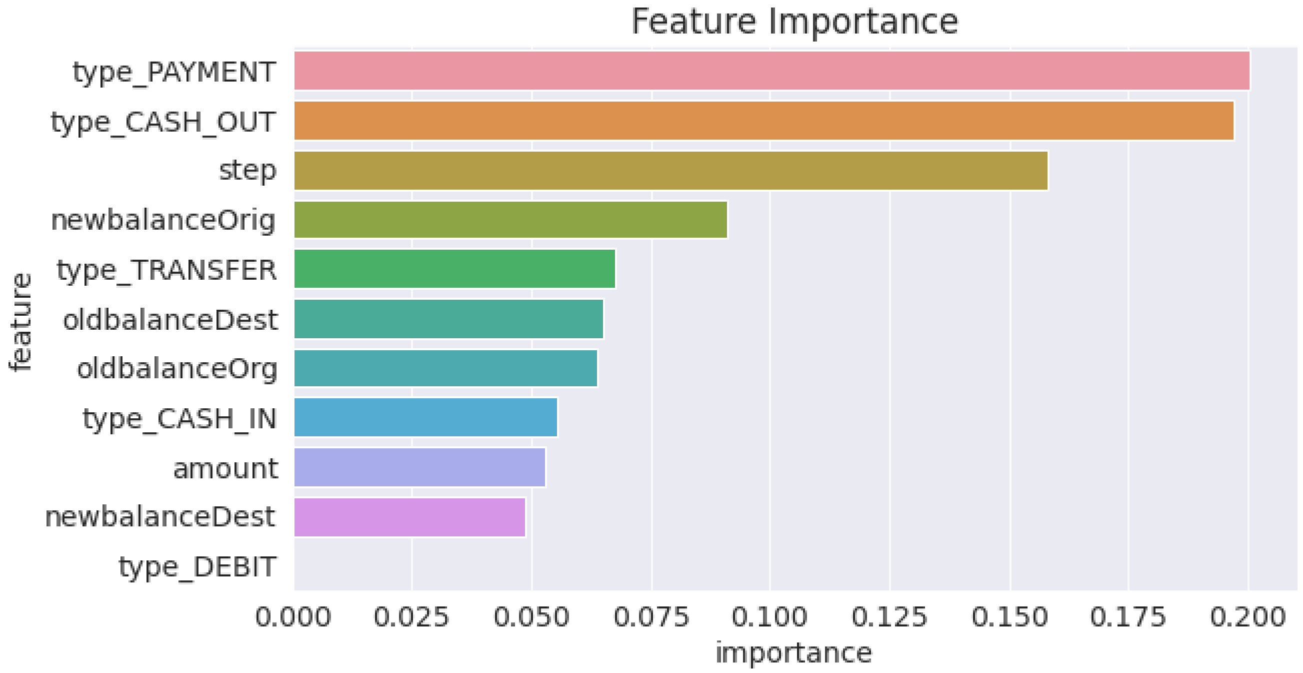
```
0.9998426435877262
```

```
importance_df = pd.DataFrame({
    'feature': numeric_cols+encoded_cols,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```
importance_df
```

	feature	importance
9	type_PAYMENT	0.200476
7	type_CASH_OUT	0.197210
0	step	0.158193
3	newbalanceOrig	0.091097
10	type_TRANSFER	0.067636
4	oldbalanceDest	0.064950
2	oldbalanceOrg	0.063605
6	type_CASH_IN	0.055213
1	amount	0.052757
5	newbalanceDest	0.048863
8	type_DEBIT	0.000000

```
plt.title('Feature Importance')
sns.barplot(data=importance_df, x='importance', y='feature');
```



```
model.feature_importances_
```

```
array([0.15819307, 0.05275699, 0.06360476, 0.09109744, 0.06495025,
       0.0488626 , 0.05521316, 0.19720972, 0.        , 0.2004758 ,
       0.06763624], dtype=float32)
```

```
import joblib
```

```
fraud_pred = {
    'model': model,
    'imputer': imputer,
    'scaler': scaler,
    'encoder': encoder,
    'input_cols': numeric_cols+encoded_cols,
    'target_col': target,
    'numeric_cols': numeric_cols,
    'categorical_cols': categorical_cols,
    'encoded_cols': encoded_cols
}
```

```
joblib.dump(fraud_pred, 'fraud_pred.joblib')
```

```
['fraud_pred.joblib']
```