

Import

```
In [1]: import pandas as pd
```

Charger les données

```
In [2]: df = pd.read_csv('CC_FRAUD.csv')
```

Examiner la structure et le contenu des données

```
In [3]: df.columns
```

```
Out[3]: Index(['DOMAIN', 'STATE', 'ZIPCODE', 'TIME1', 'TIME2', 'VIS1', 'VIS2', 'XRN1',
              'XRN2', 'XRN3', 'XRN4', 'XRN5', 'VAR1', 'VAR2', 'VAR3', 'VAR4', 'VAR5',
              'TRN_AMT', 'TOTAL_TRN_AMT', 'TRN_TYPE'],
              dtype='object')
```

```
In [4]: df.shape
```

```
Out[4]: (94682, 20)
```

```
In [5]: df.describe(include='all')
```

Out[5]:

	DOMAIN	STATE	ZIPCODE	TIME1	TIME2	VIS1	VIS2	
count	94682	94682	94682.000000	94682.000000	94682.000000	94682.000000	94682.000000	94682
unique	9809	53	NaN	NaN	NaN	NaN	NaN	
top	TMA.COM	KR	NaN	NaN	NaN	NaN	NaN	
freq	16451	18676	NaN	NaN	NaN	NaN	NaN	
mean	NaN	NaN	454.379470	13.864726	13.875858	0.113306	0.018367	
std	NaN	NaN	228.279524	5.263233	5.258338	0.316968	0.134274	
min	NaN	NaN	101.000000	0.000000	0.000000	0.000000	0.000000	
25%	NaN	NaN	166.000000	10.000000	11.000000	0.000000	0.000000	
50%	NaN	NaN	600.000000	14.000000	14.000000	0.000000	0.000000	
75%	NaN	NaN	655.000000	18.000000	18.000000	0.000000	0.000000	
max	NaN	NaN	694.000000	23.000000	23.000000	1.000000	1.000000	

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 94682 entries, 0 to 94681
Data columns (total 20 columns):
#   Column          Non-Null Count  Dtype
---  -
#   Column          Non-Null Count  Dtype
```

```

0    DOMAIN      94682 non-null object
1    STATE      94682 non-null object
2    ZIPCODE    94682 non-null int64
3    TIME1      94682 non-null int64
4    TIME2      94682 non-null int64
5    VIS1       94682 non-null int64
6    VIS2       94682 non-null int64
7    XRN1       94682 non-null int64
8    XRN2       94682 non-null int64
9    XRN3       94682 non-null int64
10   XRN4       94682 non-null int64
11   XRN5       94682 non-null int64
12   VAR1       94682 non-null int64
13   VAR2       94682 non-null int64
14   VAR3       94682 non-null float64
15   VAR4       94682 non-null int64
16   VAR5       94682 non-null int64
17   TRN_AMT    94682 non-null float64
18   TOTAL_TRN_AMT 94682 non-null float64
19   TRN_TYPE   94682 non-null object
dtypes: float64(3), int64(14), object(3)
memory usage: 14.4+ MB

```

```
In [7]: df.head(10)
```

Out[7]:

	DOMAIN	STATE	ZIPCODE	TIME1	TIME2	VIS1	VIS2	XRN1	XRN2	XRN3	XRN4
0	CDRZLKAJIVQHCN.COM	AO	675	12	12	1	0	0	1	1	0
1	NEKSXUK.NET	KK	680	18	18	1	0	0	0	0	0
2	XOSOP.COM	UO	432	3	3	1	0	0	1	1	0
3	TMA.COM	KR	119	23	23	0	0	1	0	0	0
4	VUHZRN.B.COM	PO	614	9	9	0	0	0	1	0	0
5	CIWEVXGWRG.ORG	ROI	386	11	11	0	0	0	1	1	0
6	KZOGEIFBAVSI.NET	LM	127	20	20	0	0	1	0	0	0
7	TMA.COM	AR	649	12	12	0	0	1	1	1	0
8	VUHZRN.B.COM	BO	308	13	13	0	0	0	1	1	0
9	EAYROLLTBU.COM	PO	614	6	6	0	0	1	0	0	0

Enlever les colonnes catégoriales

```
In [8]: data = df.drop(['DOMAIN', 'STATE'], axis=1)
```

Encoder la sortie

```
In [9]: from sklearn.preprocessing import LabelEncoder
data['TRN_TYPE']=LabelEncoder().fit_transform(data['TRN_TYPE'].astype(str))
```

```
In [10]: X = data.drop(['TRN_TYPE'], axis = 1)
```

```
In [11]: X
```

	ZIPCODE	TIME1	TIME2	VIS1	VIS2	XRN1	XRN2	XRN3	XRN4	XRN5	VAR1	VAR2	VAR3
0	675	12	12	1	0	0	1	1	0	1	2	1	16.680
1	680	18	18	1	0	0	0	0	0	1	3	0	37.880
2	432	3	3	1	0	0	1	1	0	1	3	1	-9.080
3	119	23	23	0	0	1	0	0	0	3	0	0	-6.392
4	614	9	9	0	0	0	1	0	0	1	3	0	42.512
...
94677	685	11	11	0	0	0	1	1	0	1	3	0	8.112
94678	108	16	16	0	0	1	0	0	1	1	4	0	11.248
94679	601	18	18	0	0	1	1	1	0	1	2	0	27.824
94680	398	23	23	0	0	0	0	0	0	1	3	0	31.904
94681	655	11	11	0	0	0	0	0	0	1	2	0	17.608

```
In [12]: Y = data['TRN_TYPE']
```

In [13]: Y

```
Out[13]: 0      1
          1      1
          2      1
          3      1
          4      1
          ..
          94677   1
          94678   1
          94679   1
          94680   1
          94681   1
          Name: TRN TYPE, Length: 94682, dtype: int32
```

Division des données entraînement / test

```
In [14]: from sklearn.model_selection import train_test_split
```

```
In [15]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random state = 0)
```

```
In [16]: X_train
```

Out[16]:

[illegible]

21243	124	14	14	0	0	1	1	1	0	1	0	0	-12.000
45891	685	7	7	1	0	1	0	0	0	1	2	0	39.688
42613	600	11	11	0	0	1	1	0	0	1	3	1	-10.040
43567	685	16	16	0	0	0	1	1	0	2	4	0	-51.200
68268	644	10	10	0	0	0	1	1	0	1	3	1	18.952

71011 rows × 17 columns

Normaliser

```
In [17]: from sklearn.preprocessing import StandardScaler
```

```
In [18]: sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test) # NB fit_transform, not best practice
```

```
In [19]: X_train
```

```
Out[19]: array([[ 0.88156473,  0.21765958,  0.21568689, ...,  3.13506673,
                  0.93058991,  0.9307182 ],
                [ 0.93415987,  0.59641435,  0.59478991, ..., -0.56590637,
                  0.01859421,  0.01933601],
                [ 0.84650131,  0.0282822 ,  0.02613537, ..., -0.56590637,
                  0.93058991,  0.9307182 ],
                ...,
                [ 0.6405037 , -0.53984995, -0.54251917, ..., -0.56590637,
                  1.71230052,  1.71190294],
                [ 1.01305257,  0.40703696,  0.4052384 , ..., -0.56590637,
                  0.93058991,  0.9307182 ],
                [ 0.83335253, -0.72922734, -0.73207068, ..., -0.56590637,
                  0.93058991,  0.9307182 ]])
```

Entraînement

```
In [20]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [21]: KNN = KNeighborsClassifier()
```

```
In [22]: # training
KNN.fit(X_train, Y_train)
```

```
Out[22]: KNeighborsClassifier()
```

```
In [23]: pred_knn = KNN.predict(X_test)
```

```
In [24]: from sklearn.metrics import accuracy_score, confusion_matrix
```

```
In [25]: accuracy_knn = accuracy_score(Y_test, pred_knn)
print(f"KNN accuracy: {accuracy_knn:.7%}")
```

KNN accuracy: 97.7229521%

```
In [26]: confusion_matrix(Y_test, pred_knn)
```

```
Out[26]: array([[    4,    537],  
               [    2, 23128]], dtype=int64)
```