```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/card_transdata.csv")
```

```
data.head()
```

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase_pric
0	57.877857	0.311140	1.94594
1	10.829943	0.175592	1.29421
2	5.091079	0.805153	0.42771
3	2.247564	5.600044	0.36266
4	44.190936	0.566486	2.22276

```
print(data.shape)
```

```
(1000000, 8)
```

```
onlycredit = data[(data['used_chip'] == 1.0)]
print(onlycredit.shape)
```

```
(350399, 8)
```

```
onlyonline = onlycredit[(onlycredit['online_order'] == 1.0)]
print(onlyonline.shape)
```

```
(227903, 8)
```

```
onlyonline.isnull().any().any()
```

```
False
```

```
X = onlyonline.drop(['fraud'], axis=1)
Y = onlyonline['fraud']

print(X, Y)
```

```
        distance_from_home  distance_from_last_transaction  \
3                 2.247564                        5.600044
4                44.190936                        0.566486
10               14.263530                        0.158758
11               13.592368                        0.240540
15              179.665148                        0.120920
...                    ...                             ...
999982            3.805818                        0.685528
999987           12.539374                        1.773940
999990           20.334489                       11.437333
999997            2.914857                        1.472687
999999           58.108125                        0.318110

        ratio_to_median_purchase_price  repeat_retailer  used_chip  \
3                             0.362663              1.0        1.0
4                             2.222767              1.0        1.0
10                            1.136102              1.0        1.0
11                            1.370330              1.0        1.0
15                            0.535640              1.0        1.0
...                                ...              ...        ...
999982                        0.336647              1.0        1.0
999987                        0.792166              1.0        1.0
999990                        0.699527              1.0        1.0
999997                        0.218075              1.0        1.0
999999                        0.386920              1.0        1.0

        used_pin_number  online_order
3                   0.0           1.0
4                   0.0           1.0
10                  0.0           1.0
11                  0.0           1.0
15                  1.0           1.0
...                 ...           ...
999982              0.0           1.0
999987              0.0           1.0
999990              0.0           1.0
999997              0.0           1.0
999999              0.0           1.0

[227903 rows x 7 columns] 3         0.0
4         0.0
10        0.0
11        0.0
15        0.0
         ...
999982    0.0
999987    0.0
999990    0.0
999997    0.0
999999    0.0
Name: fraud, Length: 227903, dtype: float64
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state = 2,
```

```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, Y_train)
pred = lr.predict(X_test)
acc = accuracy_score(Y_test, pred)

f'Acurácia:{acc * 100:.2f}'
```

'Acurácia:99.24'

```
only_real = onlyonline.fraud
only_total = onlyonline.drop(['fraud'], axis=1)
only_total
```

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase
3	2.247564	5.600044	0.
4	44.190936	0.566486	2.
10	14.263530	0.158758	1.
11	13.592368	0.240540	1.
15	179.665148	0.120920	0.
...	...	...	
999982	3.805818	0.685528	0.
999987	12.539374	1.773940	0.
999990	20.334489	11.437333	0.
999997	2.914857	1.472687	0.
999999	58.108125	0.318110	0.

227903 rows × 7 columns

```
pred = lr.predict(only_total)

only_val = pd.DataFrame({'real':only_real, 'previsao':pred})
only_val.head(n=30)
```

	real	previsao
3	0.0	0.0
4	0.0	0.0
10	0.0	0.0
11	0.0	0.0
15	0.0	0.0
28	0.0	0.0
30	0.0	0.0
31	0.0	0.0
35	1.0	1.0
39	0.0	0.0
42	0.0	0.0
43	0.0	0.0
51	0.0	0.0
52	0.0	0.0
55	0.0	0.0
67	0.0	0.0
72	0.0	0.0
77	0.0	0.0
78	0.0	0.0
79	0.0	0.0
81	0.0	0.0
91	0.0	0.0
95	0.0	0.0
98	0.0	0.0

```
only_val.previsao.value_counts()
```

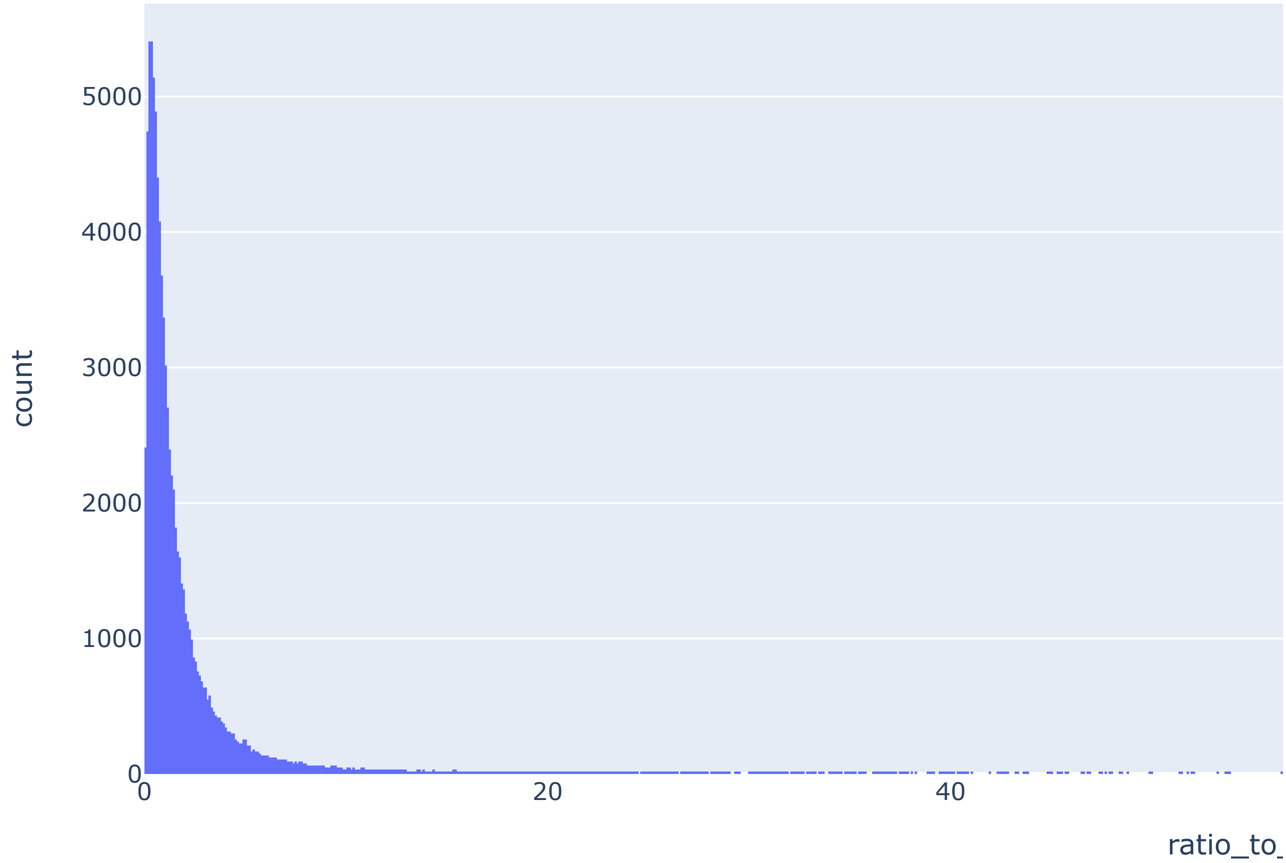
```
0.0    145650
1.0     14093
Name: previsao, dtype: int64
```

```
only_val.real.value_counts()
```

```
0.0    144670
1.0     15073
Name: real, dtype: int64
```

```
import plotly.express as px
```

```
px.histogram(onlyonline, x = 'ratio_to_median_purchase_price')
```




```
data['credit_and_online'] = np.where (
    (data['used_chip'] == 1.0) & (data['used_pin_number'] == 1.0) & (data['online_order']
    'yes',
    'no'
)
```

```
data
```

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase
0	57.877857	0.311140	1.
1	10.829943	0.175592	1.
2	5.091079	0.805153	0.

```
data2 = data[data.credit_and_online != 'no']
data2
```

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase
15	179.665148	0.120920	0.
51	43.281314	3.367793	0.
55	24.268906	0.136521	1.
98	6.136181	2.579574	1.
138	5.169928	0.534060	1.
...	...	...	
700792	54.018855	0.215318	0.
700843	11.077239	3.175977	2.
700848	3.687145	9.964012	1.
700962	5.914416	0.008577	0.
701063	3.000823	0.148435	0.

15909 rows × 9 columns

```
data2 = data2.drop([('used_chip')], axis=1)
data2
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-32-ff96b3bf7ca0> in <module>
----> 1 data2 = data2.drop([('used_chip')], axis=1)
      2 data2
```

4 frames

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexes/base.py in drop(self, labe
   6015         if mask.any():
   6016             if errors != "ignore":
-> 6017                 raise KeyError(f"{labels[mask]} not found in axis")
   6018             indexer = indexer[~mask]
   6019         return self.delete(indexer)

KeyError: "['used_chip'] not found in axis"
```

SEARCH STACK OVERFLOW

```
data2 = data2.drop([('used_pin_number')], axis=1)
data2
```

	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase
15	179.665148	0.120920	0.
51	43.281314	3.367793	0.
55	24.268906	0.136521	1.
98	6.136181	2.579574	1.
138	5.169928	0.534060	1.
...	...	...	
700792	54.018855	0.215318	0.
700843	11.077239	3.175977	2.
700848	3.687145	9.964012	1.
700962	5.914416	0.008577	0.
701063	3.000823	0.148435	0.

15909 rows × 7 columns

```
data2 = data2.drop([('online_order')], axis=1)
data2
```

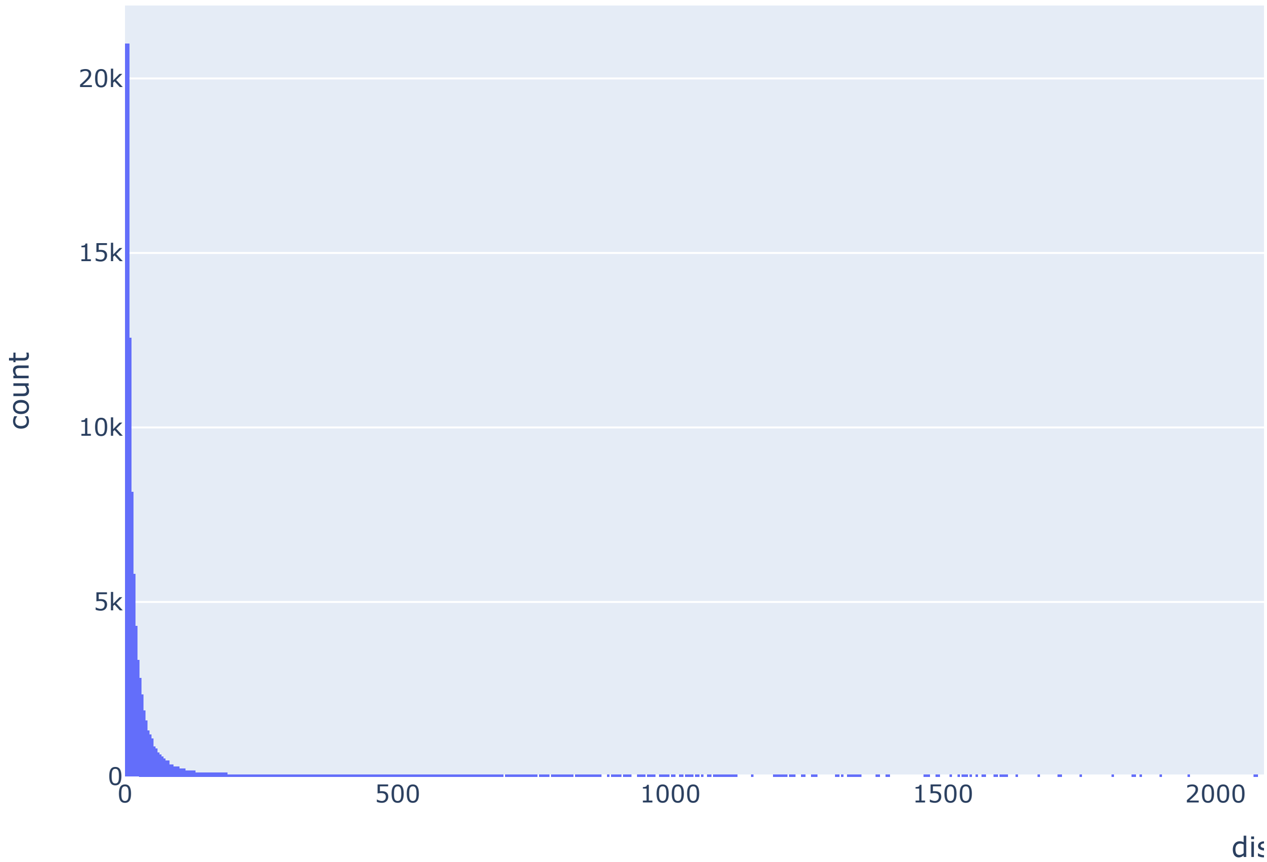
	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase
15	179.665148	0.120920	0.
51	43.281314	3.367793	0.
55	24.268906	0.136521	1.
98	6.136181	2.579574	1.
138	5.169928	0.534060	1.
...	...	...	
700792	54.018855	0.215318	0.
700843	11.077239	3.175977	2.
700848	3.687145	9.964012	1.
700962	5.914416	0.008577	0.
701063	3.000823	0.148435	0.

15909 rows × 6 columns

```
print(data2.shape)
```

```
(15909, 6)
```

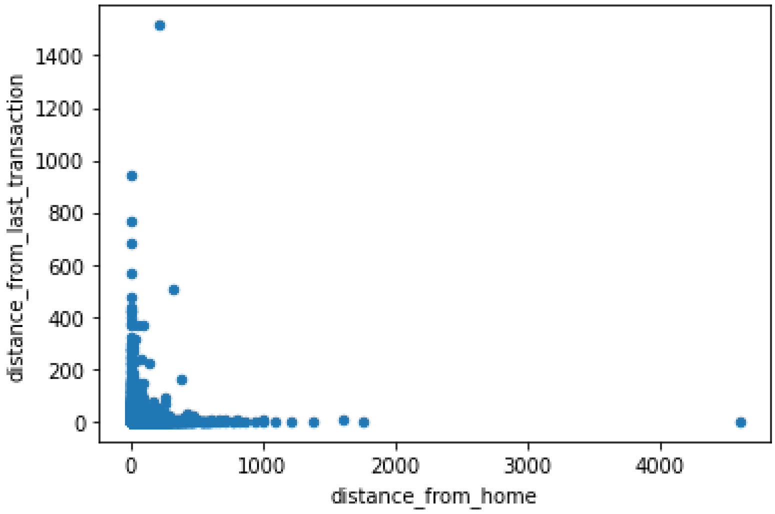
```
import plotly.express as px
px.histogram(onlyonline, x = 'distance_from_home')
```




```
import plotly.express as px
px.histogram(onlyonline, x = 'distance_from_last_transaction')
```


```
import matplotlib.pyplot as plt
data2.plot.scatter('distance_from_home', 'distance_from_last_transaction')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5a1cedb650>
```




```
from sklearn.tree import DecisionTreeClassifier
```

```
arvore = DecisionTreeClassifier(criterion='entropy')
arvore.fit(X_train, Y_train)
```

```
DecisionTreeClassifier(criterion='entropy')
```

```
arvore.feature_importances_
```

```
array([0.00739904, 0.02765845, 0.84669187, 0.00851158, 0.        ,
       0.10973907, 0.        ])
```

```
from sklearn import tree
figura, eixos = plt.subplots(nrows = 1, ncols = 1, figsize = (10,10))
tree.plot_tree(arvore)
```

```
[Text(0.4230769230769231, 0.9166666666666666, 'X[2] <= 4.0\nentropy = 0.452\nsamples
[165043, 17279]'),
 Text(0.23076923076923078, 0.75, 'X[1] <= 50.184\nentropy = 0.023\nsamples = 163503\n
365]'),
 Text(0.15384615384615385, 0.5833333333333334, 'entropy = 0.0\nsamples = 160978\nvalu
0]'),
 Text(0.3076923076923077, 0.5833333333333334, 'X[3] <= 0.5\nentropy = 0.596\nsamples
[2160, 365]'),
 Text(0.15384615384615385, 0.4166666666666667, 'X[5] <= 0.5\nentropy = 0.536\nsamples
[35, 251]'),
 Text(0.07692307692307693, 0.25, 'entropy = 0.0\nsamples = 251\nvalue = [0, 251]'),
 Text(0.23076923076923078, 0.25, 'entropy = 0.0\nsamples = 35\nvalue = [35, 0]'),
 Text(0.46153846153846156, 0.4166666666666667, 'X[0] <= 100.258\nentropy = 0.29\nsamp
= [2125, 114]'),
 Text(0.38461538461538464, 0.25, 'entropy = 0.0\nsamples = 2114\nvalue = [2114, 0]'),
 Text(0.5384615384615384, 0.25, 'X[5] <= 0.5\nentropy = 0.43\nsamples = 125\nvalue =
 Text(0.46153846153846156, 0.08333333333333333, 'entropy = 0.0\nsamples = 114\nvalue
 Text(0.6153846153846154, 0.08333333333333333, 'entropy = 0.0\nsamples = 11\nvalue =
 Text(0.6153846153846154, 0.75, 'X[5] <= 0.5\nentropy = 0.473\nsamples = 18819\nvalue
16914]'),
 Text(0.5384615384615384, 0.5833333333333334, 'entropy = 0.0\nsamples = 16908\nvalue
 Text(0.6923076923076923, 0.5833333333333334, 'X[1] <= 53.569\nentropy = 0.031\nsampl
= [1905, 6]'),
 Text(0.6153846153846154, 0.4166666666666667, 'entropy = 0.0\nsamples = 1887\nvalue =
 Text(0.7692307692307693, 0.4166666666666667, 'X[0] <= 1.785\nentropy = 0.811\nsample
[18, 6]'),
 Text(0.6923076923076923, 0.25, 'entropy = 0.0\nsamples = 5\nvalue = [0, 5]'),
 Text(0.8461538461538461, 0.25, 'X[1] <= 54.719\nentropy = 0.297\nsamples = 19\nvalue
 Text(0.7692307692307693, 0.08333333333333333, 'entropy = 0.0\nsamples = 1\nvalue = [
 Text(0.9230769230769231, 0.08333333333333333, 'entropy = 0.0\nsamples = 18\nvalue =
```