In [52]:
```
import numpy as np
import pandas as pd
```

In [53]:
```
data=pd.read_csv('houses_to_rent.csv')
data.head()
```

Out[53]:

	Unnamed: 0	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa	rent amount	property tax	fire insurance	total
0	0	1	240	3	3	4	-	acept	furnished	R$0	R$8,000	R$1,000	R$121	R$9,121
1	1	0	64	2	1	1	10	acept	not furnished	R$540	R$820	R$122	R$11	R$1,493
2	2	1	443	5	5	4	3	acept	furnished	R$4,172	R$7,000	R$1,417	R$89	R$12,680
3	3	1	73	2	2	1	12	acept	not furnished	R$700	R$1,250	R$150	R$16	R$2,116
4	4	1	19	1	1	0	-	not acept	not furnished	R$0	R$1,200	R$41	R$16	R$1,257

In [54]:
```
data.shape
```

Out[54]:
```
(6080, 14)
```

In [55]:
```
data.drop(['Unnamed: 0'],axis=1,inplace=True)
```

In [56]:
```
data.head()
```

Out[56]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa	rent amount	property tax	fire insurance	total
0	1	240	3	3	4	-	acept	furnished	R$0	R$8,000	R$1,000	R$121	R$9,121
1	0	64	2	1	1	10	acept	not furnished	R$540	R$820	R$122	R$11	R$1,493
2	1	443	5	5	4	3	acept	furnished	R$4,172	R$7,000	R$1,417	R$89	R$12,680
3	1	73	2	2	1	12	acept	not furnished	R$700	R$1,250	R$150	R$16	R$2,116
4	1	19	1	1	0	-	not acept	not furnished	R$0	R$1,200	R$41	R$16	R$1,257

In [57]:
```
data['floor'].replace(to_replace='-',value=0, inplace=True)
```

In [58]:
```
data['animal'].replace(to_replace='not acept',value=0,inplace=True)
data['animal'].replace(to_replace='acept',value=1,inplace=True)
```

In [59]:
```
data.head()
```

Out[59]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa	rent amount	property tax	fire insurance	total
0	1	240	3	3	4	0	1	furnished	R$0	R$8,000	R$1,000	R$121	R$9,121
1	0	64	2	1	1	10	1	not furnished	R$540	R$820	R$122	R$11	R$1,493
2	1	443	5	5	4	3	1	furnished	R$4,172	R$7,000	R$1,417	R$89	R$12,680
3	1	73	2	2	1	12	1	not furnished	R$700	R$1,250	R$150	R$16	R$2,116
4	1	19	1	1	0	0	0	not furnished	R$0	R$1,200	R$41	R$16	R$1,257

In [60]:
```
data['furniture'].replace(to_replace='furnished',value=1,inplace=True)
data['furniture'].replace(to_replace='not furnished',value=0,inplace=True)
```

In [61]:
```
data.head()
```

Out[61]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa	rent amount	property tax	fire insurance	total
0	1	240	3	3	4	0	1	1	R$0	R$8,000	R$1,000	R$121	R$9,121
1	0	64	2	1	1	10	1	0	R$540	R$820	R$122	R$11	R$1,493
2	1	443	5	5	4	3	1	1	R$4,172	R$7,000	R$1,417	R$89	R$12,680
3	1	73	2	2	1	12	1	0	R$700	R$1,250	R$150	R$16	R$2,116
4	1	19	1	1	0	0	0	0	R$0	R$1,200	R$41	R$16	R$1,257

In [62]:
```
data.columns
```

Out[62]:
```
Index(['city', 'area', 'rooms', 'bathroom', 'parking spaces', 'floor',
       'animal', 'furniture', 'hoa', 'rent amount', 'property tax',
       'fire insurance', 'total'],
      dtype='object')
```

In [63]:
```
for col in['hoa', 'rent amount', 'property tax','fire insurance', 'total']:
    data[col].replace(to_replace='R\$', value='',regex=True,inplace=True)
    data[col].replace(to_replace=',', value='',regex=True,inplace=True)
```

In [64]:
```
data['hoa'].replace(to_replace='Sem info',value=0, inplace=True)
```

In [68]:
```
data['hoa'].replace(to_replace='Incluso',value=0, inplace=True)
data['property tax'].replace(to_replace='Incluso',value=0, inplace=True)
```

In [69]:
```
data.head()
```

Out[69]:

	city	area	rooms	bathroom	parking spaces	floor	animal	furniture	hoa	rent amount	property tax	fire insurance	total
0	1	240	3	3	4	0	1	1	0	8000	1000	121	9121
1	0	64	2	1	1	10	1	0	540	820	122	11	1493
2	1	443	5	5	4	3	1	1	4172	7000	1417	89	12680
3	1	73	2	2	1	12	1	0	700	1250	150	16	2116
4	1	19	1	1	0	0	0	0	0	1200	41	16	1257

In [70]:
```
data.isin(['Incluso']).any()
```

Out[70]:
```
city              False
area              False
rooms             False
bathroom          False
parking spaces    False
floor             False
animal            False
furniture         False
hoa               False
rent amount       False
property tax      False
fire insurance    False
total             False
dtype: bool
```

In [71]:
```
data=data.astype(dtype=np.int64)
```

In [72]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6080 entries, 0 to 6079
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   city            6080 non-null   int64
 1   area            6080 non-null   int64
 2   rooms           6080 non-null   int64
 3   bathroom        6080 non-null   int64
 4   parking spaces  6080 non-null   int64
 5   floor           6080 non-null   int64
 6   animal          6080 non-null   int64
 7   furniture       6080 non-null   int64
 8   hoa             6080 non-null   int64
 9   rent amount     6080 non-null   int64
 10  property tax    6080 non-null   int64
 11  fire insurance  6080 non-null   int64
 12  total           6080 non-null   int64
dtypes: int64(13)
memory usage: 617.6 KB
```

In [73]:
```
data=data.sample(frac=1).reset_index(drop=True)
```

In [74]:
```
y=data['city']
X=data.drop('city',axis=1)
```

In [75]:
```
y
```

Out[75]:
```
0       1
1       1
2       0
3       1
4       1
       ..
6075    1
6076    1
6077    1
6078    1
6079    1
Name: city, Length: 6080, dtype: int64
```

In [77]:
```
from sklearn.model_selection import train_test_split
```

In [78]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [80]:
```
scaler=MinMaxScaler()
scaler.fit(X)
X=scaler.transform(X)
```

In [83]:
```
pd.DataFrame(X)
```

Out[83]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.001382	0.111111	0.000000	0.000000	0.030303	1.0	0.0	0.002227	0.017497	0.000082	0.019288	0.002892
1	0.009188	0.333333	0.444444	0.416667	0.050505	1.0	0.0	0.012850	0.102737	0.003443	0.090504	0.022826
2	0.001830	0.000000	0.111111	0.083333	0.040404	0.0	0.0	0.003382	0.011978	0.000202	0.014837	0.003024
3	0.006302	0.333333	0.222222	0.166667	0.000000	1.0	0.0	0.000000	0.102737	0.000126	0.108309	0.011993
4	0.001748	0.111111	0.000000	0.083333	0.080808	1.0	1.0	0.003273	0.063481	0.000183	0.057864	0.009190
...	...	...	...	...	...	...	...	...	...	...	...	...
6075	0.003578	0.111111	0.111111	0.000000	0.060606	1.0	0.0	0.003000	0.046658	0.000000	0.043027	0.006806
6076	0.006627	0.666667	0.111111	0.000000	0.000000	0.0	0.0	0.000000	0.098250	0.000000	0.103858	0.011324
6077	0.002074	0.111111	0.000000	0.083333	0.040404	1.0	0.0	0.002086	0.032077	0.000000	0.031157	0.004497
6078	0.006546	0.222222	0.111111	0.083333	0.131313	1.0	0.0	0.005455	0.046658	0.000819	0.043027	0.009064
6079	0.005489	0.222222	0.000000	0.166667	0.000000	1.0	1.0	0.000409	0.152086	0.000650	0.157270	0.018753

6080 rows × 12 columns

In [84]:
```
X_test,X_train,y_test,y_train=train_test_split(X,y,train_size=0.80)
```

In [89]:
```
X_test.shape
```

Out[89]:
```
(4864, 12)
```

In [90]:
```
X_train.shape
```

Out[90]:
```
(1216, 12)
```

In [92]:
```
y_train.shape
```

Out[92]:
```
(1216,)
```

In [94]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
```

In [98]:
```
log_model=LogisticRegression(penalty='l2',verbose=1)
svm_model=SVC(kernel='rbf',verbose=1)
nn_model=MLPClassifier(hidden_layer_sizes=(16,16),activation='relu',solver='adam',verbose=1)
```

In [99]:
```
log_model.fit(X_train,y_train)
svm_model.fit(X_train,y_train)
nn_model.fit(X_train,y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s finished
```

```
[LibSVM]Iteration 1, loss = 0.70206135
Iteration 2, loss = 0.68692448
Iteration 3, loss = 0.67283647
Iteration 4, loss = 0.65890005
Iteration 5, loss = 0.64375108
Iteration 6, loss = 0.62764148
Iteration 7, loss = 0.60923192
Iteration 8, loss = 0.58990230
Iteration 9, loss = 0.57001932
Iteration 10, loss = 0.54985807
Iteration 11, loss = 0.52939075
Iteration 12, loss = 0.50915115
Iteration 13, loss = 0.48976290
Iteration 14, loss = 0.47117678
Iteration 15, loss = 0.45386596
Iteration 16, loss = 0.43792509
Iteration 17, loss = 0.42359499
Iteration 18, loss = 0.41092882
Iteration 19, loss = 0.39960334
Iteration 20, loss = 0.39099798
Iteration 21, loss = 0.38326676
Iteration 22, loss = 0.37845207
Iteration 23, loss = 0.37472490
Iteration 24, loss = 0.37265848
Iteration 25, loss = 0.37118285
Iteration 26, loss = 0.37013109
Iteration 27, loss = 0.36947078
Iteration 28, loss = 0.36926531
Iteration 29, loss = 0.36898904
Iteration 30, loss = 0.36863414
Iteration 31, loss = 0.36843021
Iteration 32, loss = 0.36821959
Iteration 33, loss = 0.36800917
Iteration 34, loss = 0.36769590
Iteration 35, loss = 0.36738708
Iteration 36, loss = 0.36719751
Iteration 37, loss = 0.36701960
Iteration 38, loss = 0.36682005
Iteration 39, loss = 0.36668040
Iteration 40, loss = 0.36646622
Iteration 41, loss = 0.36632278
Iteration 42, loss = 0.36618904
Iteration 43, loss = 0.36599646
Iteration 44, loss = 0.36589967
```

```
Iteration 45, loss = 0.36559498
Iteration 46, loss = 0.36540163
Iteration 47, loss = 0.36525960
Iteration 48, loss = 0.36513485
Iteration 49, loss = 0.36490002
Iteration 50, loss = 0.36468353
Iteration 51, loss = 0.36449078
Iteration 52, loss = 0.36426514
Iteration 53, loss = 0.36410611
Iteration 54, loss = 0.36399663
Iteration 55, loss = 0.36377608
Iteration 56, loss = 0.36356023
Iteration 57, loss = 0.36340558
Iteration 58, loss = 0.36316506
Iteration 59, loss = 0.36294305
Iteration 60, loss = 0.36273383
Iteration 61, loss = 0.36252547
Iteration 62, loss = 0.36260386
Iteration 63, loss = 0.36223512
Iteration 64, loss = 0.36202245
Iteration 65, loss = 0.36175381
Iteration 66, loss = 0.36158148
Iteration 67, loss = 0.36138727
Iteration 68, loss = 0.36118811
Iteration 69, loss = 0.36098580
Iteration 70, loss = 0.36094681
Iteration 71, loss = 0.36083806
Iteration 72, loss = 0.36054109
Iteration 73, loss = 0.36045931
Iteration 74, loss = 0.36035218
Iteration 75, loss = 0.36024461
Iteration 76, loss = 0.36007583
Iteration 77, loss = 0.35964263
Iteration 78, loss = 0.35961024
Iteration 79, loss = 0.35935210
Iteration 80, loss = 0.35917842
Iteration 81, loss = 0.35899826
Iteration 82, loss = 0.35880473
Iteration 83, loss = 0.35850580
Iteration 84, loss = 0.35834979
Iteration 85, loss = 0.35814593
Iteration 86, loss = 0.35794289
Iteration 87, loss = 0.35771371
Iteration 88, loss = 0.35741624
```

```
Iteration 89, loss = 0.35742827
Iteration 90, loss = 0.35714784
Iteration 91, loss = 0.35699382
Iteration 92, loss = 0.35703781
Iteration 93, loss = 0.35709509
Iteration 94, loss = 0.35653734
Iteration 95, loss = 0.35604283
Iteration 96, loss = 0.35574258
Iteration 97, loss = 0.35569235
Iteration 98, loss = 0.35570515
Iteration 99, loss = 0.35553909
Iteration 100, loss = 0.35526965
Iteration 101, loss = 0.35467329
Iteration 102, loss = 0.35446829
Iteration 103, loss = 0.35422026
Iteration 104, loss = 0.35410095
Iteration 105, loss = 0.35405050
Iteration 106, loss = 0.35388478
Iteration 107, loss = 0.35355570
Iteration 108, loss = 0.35340903
Iteration 109, loss = 0.35308002
Iteration 110, loss = 0.35272418
Iteration 111, loss = 0.35259823
Iteration 112, loss = 0.35247881
Iteration 113, loss = 0.35224324
Iteration 114, loss = 0.35202666
Iteration 115, loss = 0.35190381
Iteration 116, loss = 0.35179206
Iteration 117, loss = 0.35140499
Iteration 118, loss = 0.35129046
Iteration 119, loss = 0.35097551
Iteration 120, loss = 0.35082498
Iteration 121, loss = 0.35071594
Iteration 122, loss = 0.35041898
Iteration 123, loss = 0.35022801
Iteration 124, loss = 0.35007140
Iteration 125, loss = 0.35004819
Iteration 126, loss = 0.34976374
Iteration 127, loss = 0.34970014
Iteration 128, loss = 0.34957772
Iteration 129, loss = 0.34950045
Iteration 130, loss = 0.34905408
Iteration 131, loss = 0.34909236
Iteration 132, loss = 0.34898743
```

```
Iteration 133, loss = 0.34860771
Iteration 134, loss = 0.34833447
Iteration 135, loss = 0.34832048
Iteration 136, loss = 0.34842123
Iteration 137, loss = 0.34835736
Iteration 138, loss = 0.34783202
Iteration 139, loss = 0.34752799
Iteration 140, loss = 0.34738472
Iteration 141, loss = 0.34726870
Iteration 142, loss = 0.34718380
Iteration 143, loss = 0.34750189
Iteration 144, loss = 0.34702063
Iteration 145, loss = 0.34672967
Iteration 146, loss = 0.34656423
Iteration 147, loss = 0.34638701
Iteration 148, loss = 0.34624755
Iteration 149, loss = 0.34601214
Iteration 150, loss = 0.34586308
Iteration 151, loss = 0.34576231
Iteration 152, loss = 0.34564012
Iteration 153, loss = 0.34543972
Iteration 154, loss = 0.34524041
Iteration 155, loss = 0.34530585
Iteration 156, loss = 0.34509811
Iteration 157, loss = 0.34482405
Iteration 158, loss = 0.34462399
Iteration 159, loss = 0.34449573
Iteration 160, loss = 0.34432319
Iteration 161, loss = 0.34414265
Iteration 162, loss = 0.34397877
Iteration 163, loss = 0.34373181
Iteration 164, loss = 0.34351085
Iteration 165, loss = 0.34352307
Iteration 166, loss = 0.34345761
Iteration 167, loss = 0.34333355
Iteration 168, loss = 0.34319382
Iteration 169, loss = 0.34284129
Iteration 170, loss = 0.34271400
Iteration 171, loss = 0.34255299
Iteration 172, loss = 0.34242252
Iteration 173, loss = 0.34225467
Iteration 174, loss = 0.34210768
Iteration 175, loss = 0.34218046
Iteration 176, loss = 0.34191472
```

```
Iteration 177, loss = 0.34164745
Iteration 178, loss = 0.34151767
Iteration 179, loss = 0.34144800
Iteration 180, loss = 0.34129762
Iteration 181, loss = 0.34113273
Iteration 182, loss = 0.34096976
Iteration 183, loss = 0.34068768
Iteration 184, loss = 0.34061423
Iteration 185, loss = 0.34083512
Iteration 186, loss = 0.34073365
Iteration 187, loss = 0.34035697
Iteration 188, loss = 0.34017671
Iteration 189, loss = 0.33990808
Iteration 190, loss = 0.33975504
Iteration 191, loss = 0.33962758
Iteration 192, loss = 0.33949153
Iteration 193, loss = 0.33941799
Iteration 194, loss = 0.33910519
Iteration 195, loss = 0.33896308
Iteration 196, loss = 0.33894293
Iteration 197, loss = 0.33882665
Iteration 198, loss = 0.33864729
Iteration 199, loss = 0.33870924
Iteration 200, loss = 0.33835526
D:\anaconda\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[99]:
```
MLPClassifier(hidden_layer_sizes=(16, 16), verbose=1)
```

In [101…
```
print(log_model.score(X_test,y_test))
print(svm_model.score(X_test,y_test))
print(nn_model.score(X_test,y_test))
```

```
0.8610197368421053
0.8610197368421053
0.8610197368421053
```

In [103…
```
data[data.columns[0]].sum()/data.shape[0]
```

Out[103]:
```
0.8633223684210526
```

In [104…
```
from sklearn.metrics import f1_score
```

In [110…
```
log_pred=log_model.predict(X_test)
svm_pred=svm_model.predict(X_test)
nn_pred=nn_model.predict(X_test)
```

In [111…
```
print(log_pred)
```

```
[1 1 1 ... 1 1 1]
```

In [112…
```
print(f1_score(log_pred,y_test))
print(f1_score(svm_pred,y_test))
print(f1_score(nn_pred,y_test))
```

```
0.9253203711886876
0.9253203711886876
0.9253203711886876
```

In [ ]:

In [ ]: