In [1]:
```
#import libraries

import json
import pandas as pd
```

In [2]:
```
#open user_profiles
with open("user_profiles.json","r") as read_file:
    data = [json.loads(line) for line in read_file]

#Normalize data
pd.json_normalize(data)
```

Out[2]:

	fraudulent	orders	paymentMethods	transactions	customer.customerEmai
0	False	[{'orderId': 'vjbdvd', 'orderAmount': 18, 'ord...	[{'paymentMethodId': 'wt07xm68b', 'paymentMeth...	[{'transactionId': 'a9lcj51r', 'orderId': 'vjb...	josephhoward@yahoo.cor
1	True	[{'orderId': 'nlghpa', 'orderAmount': 45, 'ord...	[{'paymentMethodId': 'y3xp697jx', 'paymentMeth...	[{'transactionId': '5mi94sfw', 'orderId': 'nlg...	evansjeffery@yahoo.cor
2	False	[{'orderId': 'yk34y2', 'orderAmount': 33, 'ord...	[{'paymentMethodId': '8pneoi03z', 'paymentMeth...	[{'transactionId': 'q3lyvbza', 'orderId': 'yk3...	andersonwilliam@yahoo.cor
3	False	[{'orderId': 'fbz9ep', 'orderAmount': 34, 'ord...	[{'paymentMethodId': 'pdxjdwui4', 'paymentMeth...	[{'transactionId': 'vx4cjc27', 'orderId': 'fbz...	rubenjuarez@yahoo.cor
4	True	[{'orderId': '56h7iw', 'orderAmount': 71, 'ord...	[{'paymentMethodId': 'w1i1zq3rg', 'paymentMeth...	[{'transactionId': 'q8j3dgni', 'orderId': '56h...	uchen@malone.cor
...	...	...	...	...	.
163	True	[{'orderId': '6086rd', 'orderAmount': 31, 'ord...	[{'paymentMethodId': 'x644o186l', 'paymentMeth...	[{'transactionId': 'ryxa1mzn', 'orderId': '608...	mitchellvickie@brewer jones.cor
164	True	[{'orderId': '57yhdo', 'orderAmount': 30, 'ord...	[{'paymentMethodId': 'wsy5c41nf', 'paymentMeth...	[{'transactionId': '293g6767', 'orderId': '57y...	sbrown@hughes.bi
165	True	[{'orderId': 'vcyuse', 'orderAmount': 35, 'ord...	[{'paymentMethodId': 'j3foo2l16', 'paymentMeth...	[{'transactionId': 'qio8vvq7', 'orderId': 'vcy...	johnlowery@gmail.cor
166	True	[{'orderId': '9s9ne9', 'orderAmount': 49, 'ord...	[{'paymentMethodId': 'b3jl38c9g', 'paymentMeth...	[{'transactionId': '1uusmhj9', 'orderId': '9s9...	ethompson@jackson sanders.cor
167	False	[{'orderId': 'ow1cih', 'orderAmount': 23, 'ord...	[{'paymentMethodId': 'f0xnu5om9', 'paymentMeth...	[{'transactionId': '6incxb5x', 'orderId': 'ow1...	dawn05@tucker-brown.cor

168 rows × 9 columns

In [3]:
```
orders = pd.json_normalize(data, "orders", ["fraudulent",["customer","customerEm
orders
```

Out[3]:

	orderId	orderAmount	orderState	orderShippingAddress	fraudulent	customer.customer
0	vjbdvd	18	pending	5493 Jones Islands\nBrownside, CA 51896	False	josephhoward@yaho
1	yp6x27	26	fulfilled	5493 Jones Islands\nBrownside, CA 51896	False	josephhoward@yaho
2	nlghpa	45	fulfilled	898 Henry Ports\nNew Keithview, CA 95893-2497	True	evansjeffery@yaho
3	uw0eeb	23	fulfilled	356 Elizabeth Inlet Suite 120\nPort Joshuabury...	True	evansjeffery@yaho
4	bn44oh	43	fulfilled	5093 Bryan Forks\nJoshuaton, FM 01565-9801	True	evansjeffery@yaho
...	...	...	...	...	...	
473	ow1cih	23	fulfilled	130 Kimberly Junctions\nEricmouth, KS 59756-2919	False	dawn05@t brow
474	ak5a9n	32	fulfilled	750 Sarah Stream\nAndersonfurt, WI 68970	False	dawn05@t brow
475	1bubxa	25	fulfilled	130 Kimberly Junctions\nEricmouth, KS 59756-2919	False	dawn05@t brow
476	p0gdbf	19	fulfilled	130 Kimberly Junctions\nEricmouth, KS 59756-2919	False	dawn05@t brow
477	5lmvrj	27	fulfilled	539 Branch Shore\nLake Roy, IA 11884	False	dawn05@t brow

478 rows × 10 columns

In [4]:
```
transactions = pd.json_normalize(data, "transactions")
transactions
```

Out[4]:

	transactionId	orderId	paymentMethodId	transactionAmount	transactionFailed
0	a9lcj51r	vjbdvd	wt07xm68b	18	False
1	y4wcv03i	yp6x27	wt07xm68b	26	False
2	5mi94sfw	nlghpa	41ug157xz	45	False
3	br8ba1nu	uw0eeb	41ug157xz	23	False
4	a33145ss	bn44oh	y3xp697jx	43	True
...	...	...	...	...	...
618	7yilsi1o	1bubxa	7yen1m1q8	25	True
619	wmh52bns	1bubxa	3zmkegkb8	25	True
620	c82k5bcv	1bubxa	7yen1m1q8	25	False
621	xdw33hfp	p0gdbf	3zmkegkb8	19	False
622	wgmajf82	5lmvrj	f0xnu5om9	27	False

623 rows × 5 columns

In [5]:
```
paymentMethods = pd.json_normalize(data, "paymentMethods")
paymentMethods
```

Out[5]:

	paymentMethodId	paymentMethodRegistrationFailure	paymentMethodType	paymentMet
0	wt07xm68b	True	card	
1	y3xp697jx	True	bitcoin	
2	6krszxc05	False	card	
3	5z1szj2he	False	card	Diner
4	m52tx8e1s	False	card	
...	...	...	...	
337	1tmkeoxbm	False	card	Ame
338	f0xnu5om9	False	card	
339	3zmkegkb8	False	paypal	
340	7yen1m1q8	False	card	
341	rh7676yct	True	apple pay	

342 rows × 5 columns

In [6]:
```
#merge paymentMethods & transactions on paymentMethodId
merged_data= transactions.merge(paymentMethods, on=["paymentMethodId"])
merged_data
```

Out[6]:

	transactionId	orderId	paymentMethodId	transactionAmount	transactionFailed	payment
0	a9lcj51r	vjbdvd	wt07xm68b	18	False	
1	y4wcv03i	yp6x27	wt07xm68b	26	False	
2	5mi94sfw	nlghpa	41ug157xz	45	False	
3	br8ba1nu	uw0eeb	41ug157xz	23	False	
4	a33145ss	bn44oh	y3xp697jx	43	True	
...	...	...	...	...	...	
618	xdw33hfp	p0gdbf	3zmkegkb8	19	False	
619	wpd29k0c	ak5a9n	7yen1m1q8	32	False	
620	7yilsi1o	1bubxa	7yen1m1q8	25	True	
621	c82k5bcv	1bubxa	7yen1m1q8	25	False	
622	wgmajf82	5lmvrj	f0xnu5om9	27	False	

623 rows × 9 columns

In [7]:
```
users= merged_data.merge(orders, on=["orderId"])
users
```

Out[7]:

	transactionId	orderId	paymentMethodId	transactionAmount	transactionFailed	payment
0	a9lcj51r	vjbdvd	wt07xm68b	18	False	
1	y4wcv03i	yp6x27	wt07xm68b	26	False	
2	5mi94sfw	nlghpa	41ug157xz	45	False	
3	br8ba1nu	uw0eeb	41ug157xz	23	False	
4	a33145ss	bn44oh	y3xp697jx	43	True	
...	...	...	...	...	...	
618	7yilsi1o	1bubxa	7yen1m1q8	25	True	
619	c82k5bcv	1bubxa	7yen1m1q8	25	False	
620	xdw33hfp	p0gdbf	3zmkegkb8	19	False	
621	wpd29k0c	ak5a9n	7yen1m1q8	32	False	
622	wgmajf82	5lmvrj	f0xnu5om9	27	False	

623 rows × 18 columns

In [8]:
```
#Renaming columns
users = users.rename(columns={"customer.customerEmail":"customerEmail","customer
```

In [9]:
```
from pandas_profiling import ProfileReport
```

In [10]:
```
profile_train = ProfileReport(users, title="Pandas Profiling Report")
profile_train.to_widgets()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render widgets:   0%|          | 0/1 [00:00<?, ?it/s]
```

In [11]:
```
users.dtypes
```

In [12]:
```
users.describe()
```

In [13]:
```
users['fraudulent'] = users['fraudulent'].astype('bool')
```

In [14]:
```
users.transactionFailed = users.transactionFailed.replace({True: 1, False: 0})
users.paymentMethodRegistrationFailure = users.paymentMethodRegistrationFailure.
users.fraudulent = users.fraudulent.replace({True: 1, False: 0})
```

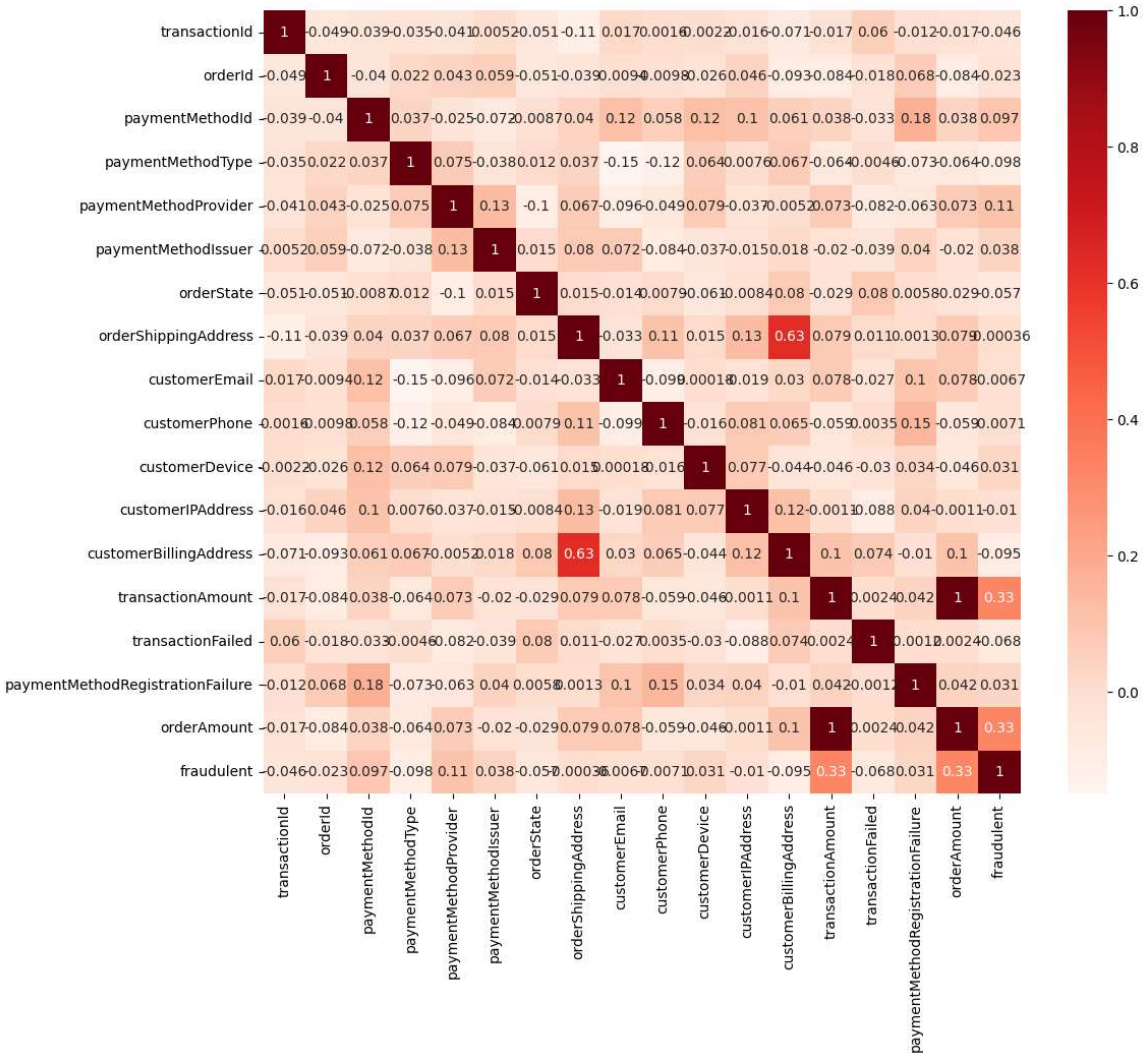
In [15]:
```
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
```

In [16]:
```
#Numerisation

obj_users = users.select_dtypes(include=['object']).copy()
int_users = users.select_dtypes(include=['int64']).copy()

le = preprocessing.LabelEncoder()

obj_users=obj_users.astype(str).apply(le.fit_transform)
```

In [17]:
```
users_df = pd.concat([obj_users,int_users], axis = 1)
users_df.head()
```

In [18]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
%matplotlib inline
```

In [19]:
```
#Using Pearson Correlation
plt.figure(figsize=(12,10))
cor = users_df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [20]:
```
#check if the data is imbalanced
fraud = users_df['fraudulent'].value_counts()
fraud
```

Out[20]:
```
0    366
1    257
Name: fraudulent, dtype: int64
```

In [21]:
```
# Print the ratio of fraud cases
ratio_cases = fraud/len(users_df.index)
print(f'Ratio of fraudulent cases: {ratio_cases[1]}\nRatio of non-fraudulent cas
```

```
Ratio of fraudulent cases: 0.41252006420545745
Ratio of non-fraudulent cases: 0.5874799357945425
```

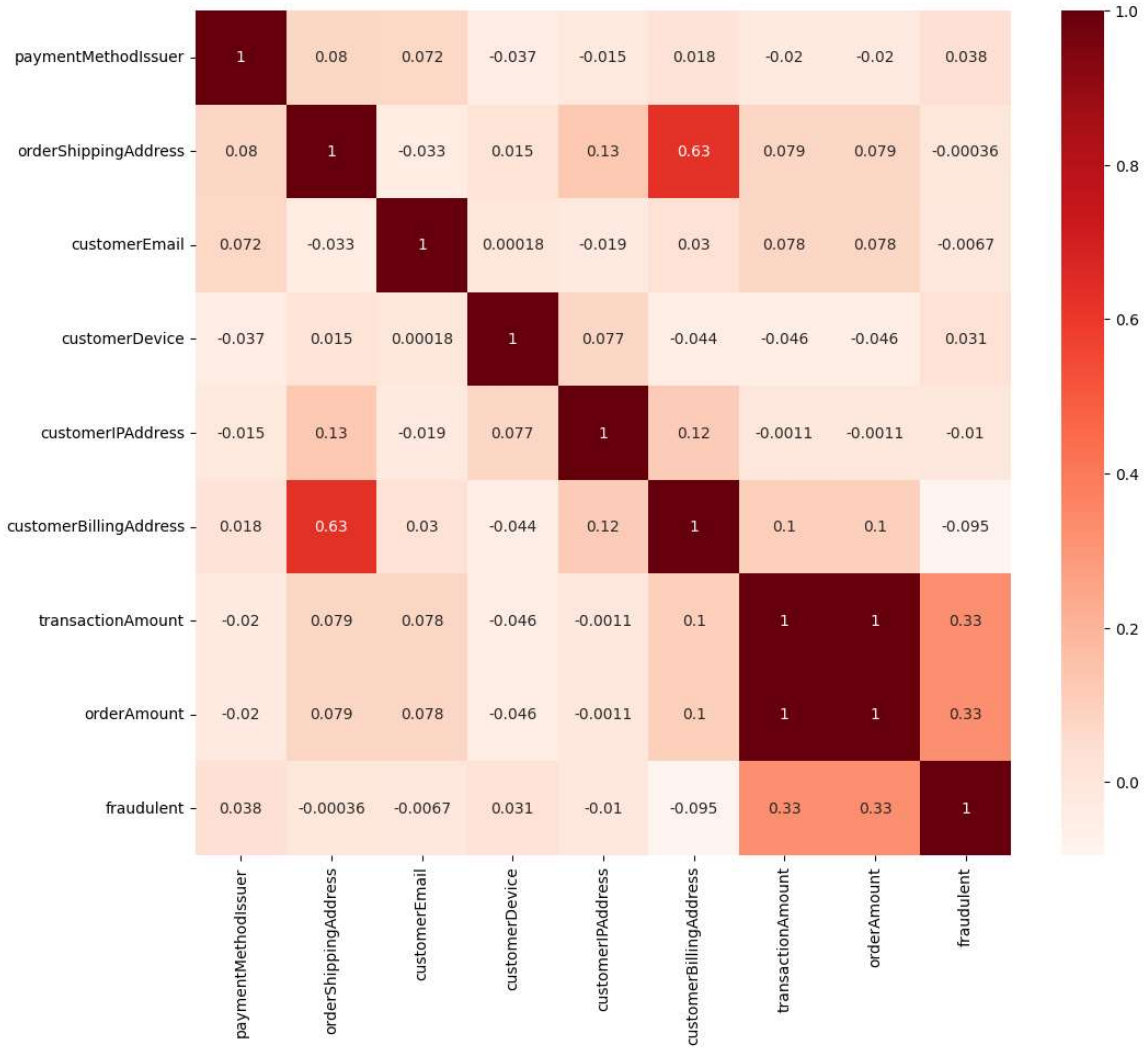
In [23]:
```
users_df = users_df.drop(columns=[
      'transactionId', 'orderId', 'paymentMethodId','customerPhone','paymentMeth

users_df.shape
```

Out[23]:
```
(623, 9)
```

In [24]:
```
users_df.shape
```

Out[24]:
```
(623, 9)
```

In [25]:
```
plt.figure(figsize=(12,10))
cor = users_df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



In [26]:
```
from sklearn.model_selection import train_test_split
X = users_df.iloc[:,0:8].values
y = users_df.iloc [:,8].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [27]:
```
#function multiple ML models
def models(X_train,y_train):
    #Logistic Regression
    from sklearn.linear_model import LogisticRegression
    log = LogisticRegression(random_state = 0)
    log.fit(X_train, y_train)

    #USE KNeighbors
    from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p=2)
    knn.fit(X_train, y_train)

    #Use SVC (linear kernel)
    from sklearn.svm import SVC
    svc_lin = SVC(kernel='linear', random_state = 0)
    svc_lin.fit(X_train, y_train)
```

```
    #USE SVC (RBF Kernel)
    from sklearn.svm import SVC
    svc_rbf = SVC(kernel='rbf', random_state=0)
    svc_rbf.fit(X_train, y_train)

    #USE GaussionNB
    from sklearn.naive_bayes import GaussianNB
    gauss = GaussianNB()
    gauss.fit(X_train, y_train)


    #USE Decision Tree
    from sklearn.tree import DecisionTreeClassifier
    tree = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
    tree.fit(X_train, y_train)

    #USE RandomForestClassifier
    from sklearn.ensemble import RandomForestClassifier
    forest = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', ra
    forest.fit(X_train, y_train)

    #print the training accuracy
    print('[0]Logistic Regression Training Accuracy:', log.score(X_train, y_trai
    print('[1]KNeighbors Training Accuracy:', knn.score(X_train, y_train))
    print('[2]SVC linear kernel Training Accuracy:', svc_lin.score(X_train, y_tr
    print('[3]SVC RBF Kernel Training Accuracy:', svc_rbf.score(X_train, y_trair
    print('[4]GaussionNB Training Accuracy:', gauss.score(X_train, y_train))
    print('[5]Decision Tree Training Accuracy:', tree.score(X_train, y_train))
    print('[6]RandomForestClassifier Training Accuracy:', forest.score(X_train,

    return log, knn, svc_lin, svc_rbf, gauss, tree, forest
```

In [28]:
```
model = models(X_train, y_train)
```

```
[0]Logistic Regression Training Accuracy: 0.6967871485943775
[1]KNeighbors Training Accuracy: 0.8955823293172691
[2]SVC linear kernel Training Accuracy: 0.7088353413654619
[3]SVC RBF Kernel Training Accuracy: 0.7771084337349398
[4]GaussionNB Training Accuracy: 0.7349397590361446
[5]Decision Tree Training Accuracy: 1.0
[6]RandomForestClassifier Training Accuracy: 1.0
```

In [30]:
```
#confusion matrix and accuracy for the models on test data
from sklearn.metrics import confusion_matrix

for i in range( len(model)):
    cm = confusion_matrix(y_test, model[i].predict(X_test))

    #extract TN, FP, FN, TP
    TN, FP, FN, TP = confusion_matrix(y_test, model[i].predict(X_test)).ravel()

    test_score = (TP + TN) / (TP + TN + FN + FP)

    print(cm)
    print('Model[{}] Testing Accuracy = "{}"' .format(i, test_score))
    print()
```

```
[[55 18]
 [23 29]]
Model[0] Testing Accuracy = "0.672"

[[64  9]
 [19 33]]
Model[1] Testing Accuracy = "0.776"

[[59 14]
 [26 26]]
Model[2] Testing Accuracy = "0.68"

[[66  7]
 [24 28]]
Model[3] Testing Accuracy = "0.752"

[[65  8]
 [26 26]]
Model[4] Testing Accuracy = "0.728"

[[69  4]
 [ 9 43]]
Model[5] Testing Accuracy = "0.896"

[[70  3]
 [ 4 48]]
Model[6] Testing Accuracy = "0.944"
```

In [31]:
```
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, confusion_matrix, classification_rep
```

In [32]:
```
pred = model[6].predict(X_test)

print('ROCAUC score:',roc_auc_score(y_test, pred))
print('Accuracy score:',accuracy_score(y_test, pred))
print('F1 score:',f1_score(y_test, pred))
print('Confusion matrix:' ,confusion_matrix(y_test,pred))
print('Log Loss:' ,log_loss(y_test,pred))
print('Recall:' ,recall_score(y_test,pred))
print(classification_report(y_test,pred))
```

```
ROCAUC score: 0.9409905163329821
Accuracy score: 0.944
F1 score: 0.9320388349514563
Confusion matrix: [[70  3]
 [ 4 48]]
Log Loss: 2.0184445897905605
Recall: 0.9230769230769231
              precision    recall  f1-score   support

           0       0.95      0.96      0.95        73
           1       0.94      0.92      0.93        52

    accuracy                           0.94       125
   macro avg       0.94      0.94      0.94       125
weighted avg       0.94      0.94      0.94       125
```

In [33]:
```
#get feature importance
forest = model[6]
importances = pd.DataFrame({'feature':users_df.iloc[:,0:8].columns, 'importance'
importances = importances.sort_values('importance', ascending=False).set_index('
importances
```

Out[33]:

feature	importance
orderAmount	0.183
customerIPAddress	0.142
paymentMethodIssuer	0.131
customerDevice	0.130
customerBillingAddress	0.127
customerEmail	0.112
transactionAmount	0.101
orderShippingAddress	0.075

In [34]:
```
#print prediction of random forest classifier

print(pred)

print()



#print the actual values
print(y_test)
```

```
[1 1 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 1 0 1
 1 1 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1
 0 0 1 0 1 1 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1
 0 1 0 0 1 1 1 0 0 0 0 0 0 0]

[1 1 0 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 1
 1 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 1 1 1
 0 0 1 0 1 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 1
 0 1 0 0 1 1 1 0 0 0 0 0 1 0 0]
```

In [ ]:
```

```