

```
In [1]: #!pip install numpy pandas matplotlib scikit-learn seaborn
```

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

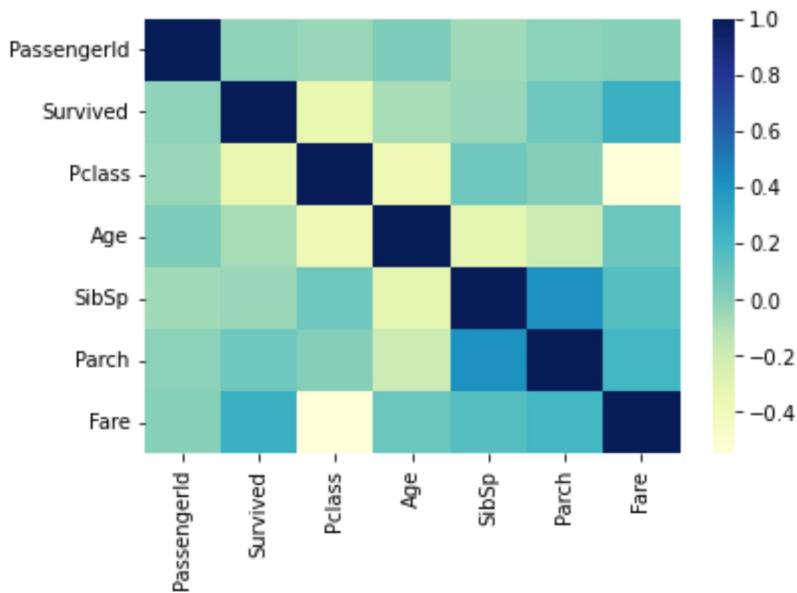
```
In [3]: titanic_data = pd.read_csv('data/train.csv')
```

```
In [4]: #titanic_data
```

```
In [5]: #titanic_data.describe()
```

```
In [6]: import seaborn as sns

sns.heatmap(titanic_data.corr(), cmap="YlGnBu")
plt.show()
```



```
In [7]: from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2)
for train_indices, test_indices in split.split(titanic_data, titanic_data[['Survived']])
    strat_train_set = titanic_data.loc[train_indices]
    strat_test_set = titanic_data.loc[test_indices]
```

```
In [8]: strat_test_set
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
289	290	1	3	Connolly, Miss. Kate	female	22.0	0	0	370373	7.7500	¶
68	69	1	3	Andersson, Miss. Erna Alexandra	female	17.0	4	2	3101281	7.9250	¶
153	154	0	3	van Billiard, Mr. Austin Blyler	male	40.5	0	2	A/5. 851	14.5000	¶
568	569	0	3	Doharr, Mr. Tannous	male	NaN	0	0	2686	7.2292	¶
858	859	1	3	Baclini, Mrs. Solomon (Latifa Qurban)	female	24.0	0	3	2666	19.2583	¶
...
480	481	0	3	Goodwin, Master. Harold Victor	male	9.0	5	2	CA 2144	46.9000	¶
595	596	0	3	Van Impe, Mr. Jean Baptiste	male	36.0	1	1	345773	24.1500	¶
490	491	0	3	Hagland, Mr. Konrad Mathias Reiersen	male	NaN	1	0	65304	19.9667	¶
527	528	0	1	Farthing, Mr. John	male	NaN	0	0	PC 17483	221.7792	
282	283	0	3	de Pelsmaeker, Mr. Alfons	male	16.0	0	0	345778	9.5000	¶

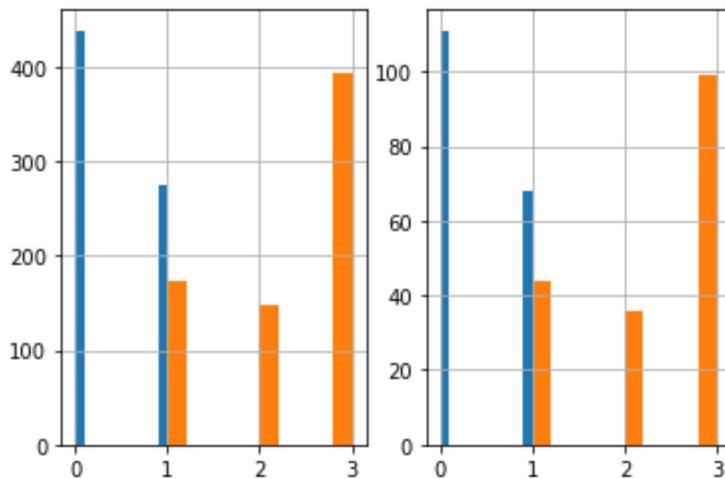
179 rows × 12 columns

In [9]:

```
plt.subplot(1,2,1)
strat_train_set['Survived'].hist() #blue
strat_train_set['Pclass'].hist() #orange

plt.subplot(1,2,2)
strat_test_set['Survived'].hist() #blue
strat_test_set['Pclass'].hist() #orange

plt.show()
```



```
In [10]: strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 7 to 110
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   PassengerId 712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Name         712 non-null    object  
 4   Sex          712 non-null    object  
 5   Age          574 non-null    float64 
 6   SibSp        712 non-null    int64  
 7   Parch        712 non-null    int64  
 8   Ticket       712 non-null    object  
 9   Fare          712 non-null    float64 
 10  Cabin        159 non-null    object  
 11  Embarked     712 non-null    object  
dtypes: float64(2), int64(5), object(5)
memory usage: 72.3+ KB
```

```
In [11]: from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.impute import SimpleImputer
```

```
class AgeImputer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        imputer = SimpleImputer(strategy="mean")
        X['Age'] = imputer.fit_transform(X[['Age']])
        return X
```

```
In [12]: from sklearn.preprocessing import OneHotEncoder

class FeatureEncoder(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        encoder = OneHotEncoder()
        matrix = encoder.fit_transform(X[['Embarked']]).toarray()

        column_names = ["C", "S", "Q", "N"]

        for i in range(len(matrix.T)):
            X[column_names[i]] = matrix.T[i]

        matrix = encoder.fit_transform(X[['Sex']]).toarray()

        column_names = ["Female", "Male"]

        for i in range(len(matrix.T)):
            X[column_names[i]] = matrix.T[i]

        return X
```

```
In [13]: class FeatureDropper(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X.drop(["Embarked", "Name", "Ticket", "Cabin", "Sex", "N"], axis=1,
```

```
In [14]: from sklearn.pipeline import Pipeline

pipeline = Pipeline([('ageimputer', AgeImputer()),
                     ('featureencoder', FeatureEncoder()),
                     ('featuredropper', FeatureDropper())])
```

```
In [15]: strat_train_set = pipeline.fit_transform(strat_train_set)
```

```
In [16]: strat_train_set
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
7	8	0	3	2.000000	3	1	21.0750	0.0	0.0	1.0	0.0	1.0
851	852	0	3	74.000000	0	0	7.7750	0.0	0.0	1.0	0.0	1.0
45	46	0	3	29.764233	0	0	8.0500	0.0	0.0	1.0	0.0	1.0
498	499	0	1	25.000000	1	2	151.5500	0.0	0.0	1.0	1.0	0.0
730	731	1	1	29.000000	0	0	211.3375	0.0	0.0	1.0	1.0	0.0
...
415	416	0	3	29.764233	0	0	8.0500	0.0	0.0	1.0	1.0	0.0
732	733	0	2	29.764233	0	0	0.0000	0.0	0.0	1.0	0.0	1.0
346	347	1	2	40.000000	0	0	13.0000	0.0	0.0	1.0	1.0	0.0
150	151	0	2	51.000000	0	0	12.5250	0.0	0.0	1.0	0.0	1.0
110	111	0	1	47.000000	0	0	52.0000	0.0	0.0	1.0	0.0	1.0

712 rows × 12 columns

In [17]: `strat_train_set.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 7 to 110
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   PassengerId 712 non-null    int64  
 1   Survived     712 non-null    int64  
 2   Pclass       712 non-null    int64  
 3   Age          712 non-null    float64 
 4   SibSp        712 non-null    int64  
 5   Parch        712 non-null    int64  
 6   Fare          712 non-null    float64 
 7   C             712 non-null    float64 
 8   S             712 non-null    float64 
 9   Q             712 non-null    float64 
 10  Female        712 non-null    float64 
 11  Male          712 non-null    float64 
dtypes: float64(7), int64(5)
memory usage: 72.3 KB
```

In [18]: `strat_train_set`

Out[18]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
7	8	0	3	2.000000	3	1	21.0750	0.0	0.0	1.0	0.0	1.0
851	852	0	3	74.000000	0	0	7.7750	0.0	0.0	1.0	0.0	1.0
45	46	0	3	29.764233	0	0	8.0500	0.0	0.0	1.0	0.0	1.0
498	499	0	1	25.000000	1	2	151.5500	0.0	0.0	1.0	1.0	0.0
730	731	1	1	29.000000	0	0	211.3375	0.0	0.0	1.0	1.0	0.0
...
415	416	0	3	29.764233	0	0	8.0500	0.0	0.0	1.0	1.0	0.0
732	733	0	2	29.764233	0	0	0.0000	0.0	0.0	1.0	0.0	1.0
346	347	1	2	40.000000	0	0	13.0000	0.0	0.0	1.0	1.0	0.0
150	151	0	2	51.000000	0	0	12.5250	0.0	0.0	1.0	0.0	1.0
110	111	0	1	47.000000	0	0	52.0000	0.0	0.0	1.0	0.0	1.0

712 rows × 12 columns

In [19]:

```
from sklearn.preprocessing import StandardScaler

X = strat_train_set.drop(['Survived'], axis=1)
y = strat_train_set['Survived']

scaler = StandardScaler()
X_data = scaler.fit_transform(X)
y_data = y.to_numpy()
```

In [20]:

```
#X_data
```

In [21]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

clf = RandomForestClassifier()

param_grid = [
    {"n_estimators": [10, 100, 200, 500], "max_depth": [None, 5, 10], "min_samples_"}
]

grid_search = GridSearchCV(clf, param_grid, cv=3, scoring="accuracy", return_train_
grid_search.fit(X_data, y_data)
```

Out[21]:

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
            param_grid=[{'max_depth': [None, 5, 10],
                         'min_samples_split': [2, 3, 4],
                         'n_estimators': [10, 100, 200, 500]}],
            return_train_score=True, scoring='accuracy')
```

In [22]:

```
final_clf = grid_search.best_estimator_
```

In [23]:

```
final_clf
```

```
Out[23]: RandomForestClassifier(max_depth=10, min_samples_split=4)
```

```
In [24]: strat_test_set = pipeline.fit_transform(strat_test_set)
```

```
In [25]: #strat_test_set
```

```
In [26]: X_test = strat_test_set.drop(['Survived'], axis=1)
y_test = strat_test_set['Survived']

scaler = StandardScaler()
X_data_test = scaler.fit_transform(X_test)
y_data_test = y_test.to_numpy()
```

```
In [27]: final_clf.score(X_data_test, y_data_test)
```

```
Out[27]: 0.8212290502793296
```

```
In [28]: final_data = pipeline.fit_transform(titanic_data)
```

```
In [29]: final_data
```

```
Out[29]:   PassengerId  Survived  Pclass      Age  SibSp  Parch     Fare      C      S      Q  Female  Male
          0            1        0    3  22.000000      1       0    7.2500  0.0  0.0  1.0      0.0  1.0
          1            2        1    1  38.000000      1       0   71.2833  1.0  0.0  0.0      1.0  0.0
          2            3        1    3  26.000000      0       0    7.9250  0.0  0.0  1.0      1.0  0.0
          3            4        1    1  35.000000      1       0   53.1000  0.0  0.0  1.0      1.0  0.0
          4            5        0    3  35.000000      0       0    8.0500  0.0  0.0  1.0      0.0  1.0
...
          886           887       0    2  27.000000      0       0   13.0000  0.0  0.0  1.0      0.0  1.0
          887           888       1    1  19.000000      0       0   30.0000  0.0  0.0  1.0      1.0  0.0
          888           889       0    3  29.699118      1       2   23.4500  0.0  0.0  1.0      1.0  0.0
          889           890       1    1  26.000000      0       0   30.0000  1.0  0.0  0.0      0.0  1.0
          890           891       0    3  32.000000      0       0    7.7500  0.0  1.0  0.0      0.0  1.0
```

891 rows × 12 columns

```
In [30]: X_final = final_data.drop(['Survived'], axis=1)
y_final = final_data['Survived']

scaler = StandardScaler()
X_data_final = scaler.fit_transform(X_final)
y_data_final = y_final.to_numpy()
```

```
In [31]: production_clf = RandomForestClassifier()

param_grid = [
    {"n_estimators": [10, 100, 200, 500], "max_depth": [None, 5, 10], "min_samples_"
]

grid_search = GridSearchCV(production_clf, param_grid, cv=3, scoring="accuracy", re
grid_search.fit(X_data_final, y_data_final)
```

```
Out[31]: GridSearchCV(cv=3, estimator=RandomForestClassifier(),
    param_grid=[{'max_depth': [None, 5, 10],
                 'min_samples_split': [2, 3, 4],
                 'n_estimators': [10, 100, 200, 500]}],
    return_train_score=True, scoring='accuracy')
```

```
In [32]: prod_final_clf = grid_search.best_estimator_
```

```
In [33]: prod_final_clf
```

```
Out[33]: RandomForestClassifier(max_depth=5, n_estimators=200)
```

```
In [34]: titanic_test_data = pd.read_csv("data/test.csv")
```

```
In [35]: titanic_test_data
```

Out[35]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	
...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	

418 rows × 11 columns

In [36]: `final_test_data = pipeline.fit_transform(titanic_test_data)`

In [37]: `final_test_data`

Out[37]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	C	S	Q	Female	Male
0	892	3	34.50000	0	0	7.8292	0.0	1.0	0.0	0.0	1.0
1	893	3	47.00000	1	0	7.0000	0.0	0.0	1.0	1.0	0.0
2	894	2	62.00000	0	0	9.6875	0.0	1.0	0.0	0.0	1.0
3	895	3	27.00000	0	0	8.6625	0.0	0.0	1.0	0.0	1.0
4	896	3	22.00000	1	1	12.2875	0.0	0.0	1.0	1.0	0.0
...
413	1305	3	30.27259	0	0	8.0500	0.0	0.0	1.0	0.0	1.0
414	1306	1	39.00000	0	0	108.9000	1.0	0.0	0.0	1.0	0.0
415	1307	3	38.50000	0	0	7.2500	0.0	0.0	1.0	0.0	1.0
416	1308	3	30.27259	0	0	8.0500	0.0	0.0	1.0	0.0	1.0
417	1309	3	30.27259	1	1	22.3583	1.0	0.0	0.0	0.0	1.0

418 rows × 11 columns

In [38]: `final_test_data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   PassengerId  418 non-null   int64  
 1   Pclass        418 non-null   int64  
 2   Age           418 non-null   float64 
 3   SibSp         418 non-null   int64  
 4   Parch         418 non-null   int64  
 5   Fare          417 non-null   float64 
 6   C              418 non-null   float64 
 7   S              418 non-null   float64 
 8   Q              418 non-null   float64 
 9   Female        418 non-null   float64 
 10  Male          418 non-null   float64 
dtypes: float64(7), int64(4)
memory usage: 36.0 KB
```

In [39]: `X_final_test = final_test_data
X_final_test = X_final_test.fillna(method="ffill")`

```
scaler = StandardScaler()  
X_data_final_test = scaler.fit_transform(X_final_test)
```

In [40]: `predictions = prod_final_clf.predict(X_data_final_test)`

In [41]: `predictions`

```
In [42]: final_df = pd.DataFrame(titanic_test_data['PassengerId'])
final_df['Survived'] = predictions
final_df.to_csv("Data/predictions.csv", index=False)
```

In [43]: final_df

Out[43]:

PassengerId	Survived
1	0
2	1
3	1
4	0
5	1
6	0
7	1
8	1
9	0
10	1
11	1
12	0
13	1
14	1
15	0
16	1
17	1
18	0
19	1
20	1
21	0
22	1
23	1
24	0
25	1
26	1
27	0
28	1
29	1
30	0
31	1
32	1
33	0
34	1
35	1
36	0
37	1
38	1
39	0
40	1
41	1
42	0
43	1
44	1
45	0
46	1
47	1
48	0
49	1
50	1
51	0
52	1
53	1
54	0
55	1
56	1
57	0
58	1
59	1
60	0
61	1
62	1
63	0
64	1
65	1
66	0
67	1
68	1
69	0
70	1
71	1
72	0
73	1
74	1
75	0
76	1
77	1
78	0
79	1
80	1
81	0
82	1
83	1
84	0
85	1
86	1
87	0
88	1
89	1
90	0
91	1
92	1
93	0
94	1
95	1
96	0
97	1
98	1
99	0
100	1
101	1
102	0
103	1
104	1
105	0
106	1
107	1
108	0
109	1
110	1
111	0
112	1
113	1
114	0
115	1
116	1
117	0
118	1
119	1
120	0
121	1
122	1
123	0
124	1
125	1
126	0
127	1
128	1
129	0
130	1
131	1
132	0
133	1
134	1
135	0
136	1
137	1
138	0
139	1
140	1
141	0
142	1
143	1
144	0
145	1
146	1
147	0
148	1
149	1
150	0
151	1
152	1
153	0
154	1
155	1
156	0
157	1
158	1
159	0
160	1
161	1
162	0
163	1
164	1
165	0
166	1
167	1
168	0
169	1
170	1
171	0
172	1
173	1
174	0
175	1
176	1
177	0
178	1
179	1
180	0
181	1
182	1
183	0
184	1
185	1
186	0
187	1
188	1
189	0
190	1
191	1
192	0
193	1
194	1
195	0
196	1
197	1
198	0
199	1
200	1
201	0
202	1
203	1
204	0
205	1
206	1
207	0
208	1
209	1
210	0
211	1
212	1
213	0
214	1
215	1
216	0
217	1
218	1
219	0
220	1
221	1
222	0
223	1
224	1
225	0
226	1
227	1
228	0
229	1
230	1
231	0
232	1
233	1
234	0
235	1
236	1
237	0
238	1
239	1
240	0
241	1
242	1
243	0
244	1
245	1
246	0
247	1
248	1
249	0
250	1
251	1
252	0
253	1
254	1
255	0
256	1
257	1
258	0
259	1
260	1
261	0
262	1
263	1
264	0
265	1
266	1
267	0
268	1
269	1
270	0
271	1
272	1
273	0
274	1
275	1
276	0
277	1
278	1
279	0
280	1
281	1
282	0
283	1
284	1
285	0
286	1
287	1
288	0
289	1
290	1
291	0
292	1
293	1
294	0
295	1
296	1
297	0
298	1
299	1
300	0
301	1
302	1
303	0
304	1
305	1
306	0
307	1
308	1
309	0
310	1
311	1
312	0
313	1
314	1
315	0
316	1
317	1
318	0
319	1
320	1
321	0
322	1
323	1
324	0
325	1
326	1
327	0
328	1
329	1
330	0
331	1
332	1
333	0
334	1
335	1
336	0
337	1
338	1
339	0
340	1
341	1
342	0
343	1
344	1
345	0
346	1
347	1
348	0
349	1
350	1
351	0
352	1
353	1
354	0
355	1
356	1
357	0
358	1
359	1
360	0
361	1
362	1
363	0
364	1
365	1
366	0
367	1
368	1
369	0
370	1
371	1
372	0
373	1
374	1
375	0
376	1
377	1
378	0
379	1
380	1
381	0
382	1
383	1
384	0
385	1
386	1
387	0
388	1
389	1
390	0
391	1
392	1
393	0
394	1
395	1
396	0
397	1
398	1
399	0
400	1
401	1
402	0
403	1
404	1
405	0
406	1
407	1
408	0
409	1
410	1
411	0
412	1
413	1
414	0
415	1
416	1
417	0
418	1
419	1
420	0
421	1
422	1
423	0
424	1
425	1
426	0
427	1
428	1
429	0
430	1
431	1
432	0
433	1
434	1
435	0
436	1
437	1
438	0
439	1
440	1
441	0
442	1
443	1
444	0
445	1
446	1
447	0
448	1
449	1
450	0
451	1
452	1
453	0
454	1
455	1
456	0
457	1
458	1
459	0
460	1
461	1
462	0
463	1
464	1
465	0
466	1
467	1
468	0
469	1
470	1
471	0
472	1
473	1
474	0
475	1
476	1
477	0
478	1
479	1
480	0
481	1
482	1
483	0
484	1
485	1
486	0
487	1
488	1
489	0
490	1
491	1
492	0
493	1
494	1
495	0
496	1
497	1
498	0
499	1
500	1
501	0
502	1
503	1
504	0
505	1
506	1
507	0
508	1
509	1
510	0
511	1
512	1
513	0
514	1
515	1
516	0
517	1
518	1
519	0
520	1
521	1
522	0
523	1
524	1
525	0
526	1
527	1
528	0
529	1
530	1
531	0
532	1
533	1
534	0
535	1
536	1
537	0
538	1
539	1
540	0
541	1
542	1
543	0
544	1
545	1
546	0
547	1
548	1
549	0
550	1
551	1
552	0
553	1
554	1
555	0
556	1
557	1
558	0
559	1
560	1
561	0
562	1
563	1
564	0
565	1
566	1
567	0
568	1
569	1
570	0
571	1
572	1
573	0
574	1
575	1
576	0
577	1
578	1
579	0
580	1
581	1
582	0
583	1
584	1
585	0
586	1
587	1
588	0
589	1
590	1
591	0
592	1
593	1
594	0
595	1
596	1
597	0
598	1
599	1
600	0
601	1
602	1
603	0
604	1
605	1
606	0
607	1
608	1
609	0
610	1
611	1
612	0
613	1
614	1
615	0
616	1
617	1
618	0
619	1
620	1
621	0
622	1
623	1
624	0
625	1
626	1
627	0
628	1
629	1
630	0
631	1
632	1
633	0
634	1
635	1
636	0
637	1
638	1
639	0
640	1
641	1
642	0
643	1
644	1
645	0
646	1
647	1
648	0
649	1
650	1
651	0
652	1
653	1
654	0
655	1
656	1
657	0
658	1
659	1
660	0
661	1
662	1
663	0
664	1
665	1
666	0
667	1
668	1
669	0
670	1
671	1
672	0
673	1
674	1
675	0
676	1
677	1
678	0
679	1
680	1
681	0
682	1
683	1
684	0
685	1
686	1
687	0
688	1
689	1
690	0
691	

0	892	0
1	893	0
2	894	0
3	895	0
4	896	1
...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns