Titanic Survival Prediction

In this model we will predict the survival of passengers.

Importing required libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing training data

In [2]:
```
train=pd.read_csv('train.csv')
train.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Details about data

Survived : 0: No, 1: Yes

Pclass : Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)

Name : Name of the passenger

Sex : Male or Female

Age : Age of the passenger

SibSp : Number of Siblings/Spouses Aboard

Parch : Number of Parents/Children Aboard

Ticket : Ticket Number

Fare : Passenger Fare

Cabin : Cabin

Embarked : Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Check for null values

In [3]:
```
train.isnull()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

In [4]:
```
train.isnull().sum()
```

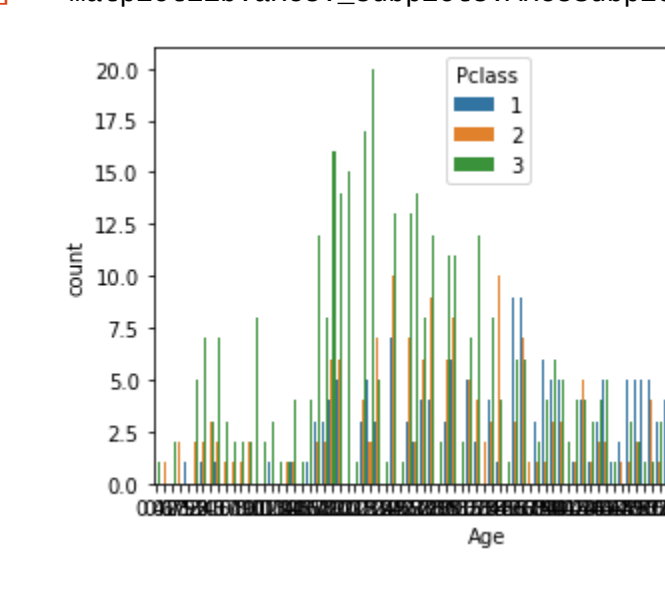
Out[4]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

We can see that columns Age, Cabin and Embarked have some null values.

Filling null values in age

In [5]:
```
sns.countplot(x='Age',hue='Pclass',data=train)
```

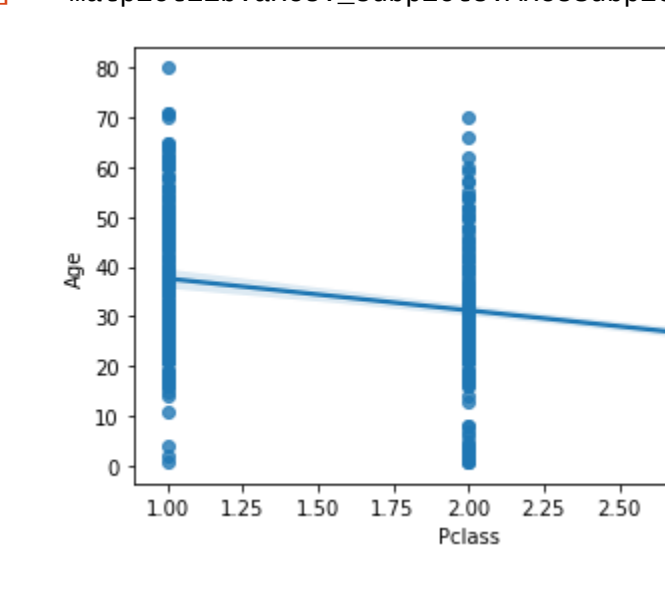
Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22bb8b64708>
```



We can see that age group can be generalized according to the passenger class.Therefore, we will now find the average age according to passenger class.

In [6]:
```
sns.regplot(x='Pclass',y='Age',data=train)
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22bb966ffc8>
```



Average age group in first class was around 39-40,in second class was around 32-35 and in third class was around 28-29.

In [7]:
```
def fill_age(cols):
    Class=cols[0]
    Age=cols[1]

    if pd.isnull(Age):
        if Class==1:
            return 39
        elif Class==2:
            return 32
        else:
            return 28
    return Age
```

In [8]:
```
train['Age']=train[['Pclass','Age']].apply(fill_age,axis=1)
```

Checking for null values.

In [9]:
```
train.isnull().sum()
```

Out[9]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

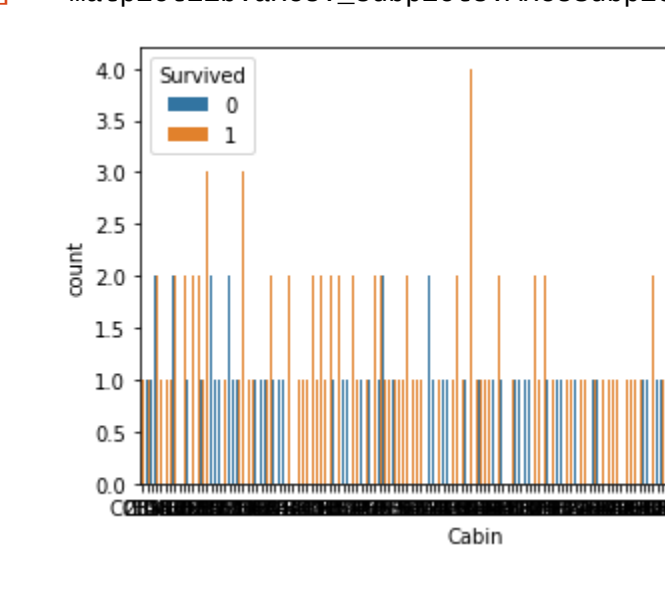
We can see that all the rows in age column have been filled.

Analysing cabin column for null values

Checking the correlation between Cabin and Survived.

In [10]:
```
sns.countplot(x='Cabin',hue='Survived',data=train)
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22bb97babc8>
```



Cabin does not provide any valuable information so we will drop cabin.

In [11]:
```
train.drop('Cabin',axis=1,inplace=True)
```

In [12]:
```
train.isnull().sum()
```

Out[12]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
dtype: int64
```

All the null values all removed.

Analysing relation between different features and the target variable('Survived')

In [13]:
```
train.head()
```

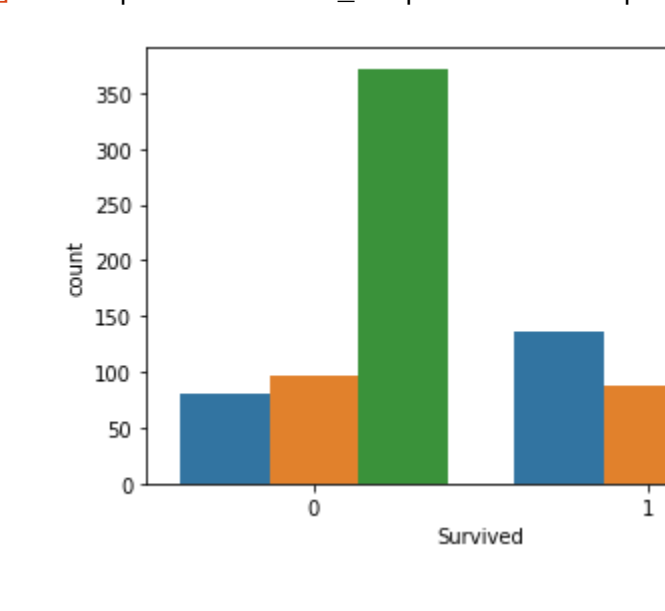
Out[13]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S

Relation between passenger's class and survival rate.

In [14]:
```
sns.countplot(x='Survived',hue='Pclass',data=train)
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22bb9c499c8>
```

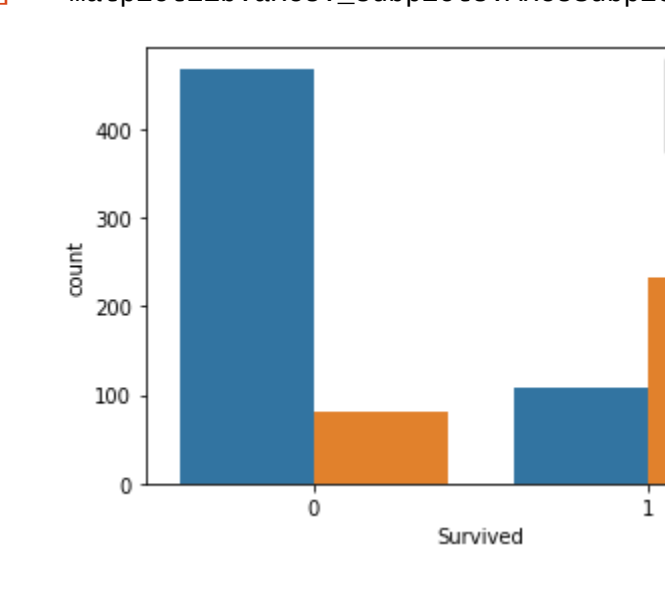


We can see that a large proportion of passengers who didn't survive belonged to third class.

Relation between passenger's sex and survival rate.

In [15]:
```
sns.countplot(x='Survived',hue='Sex',data=train)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22bb9add948>
```

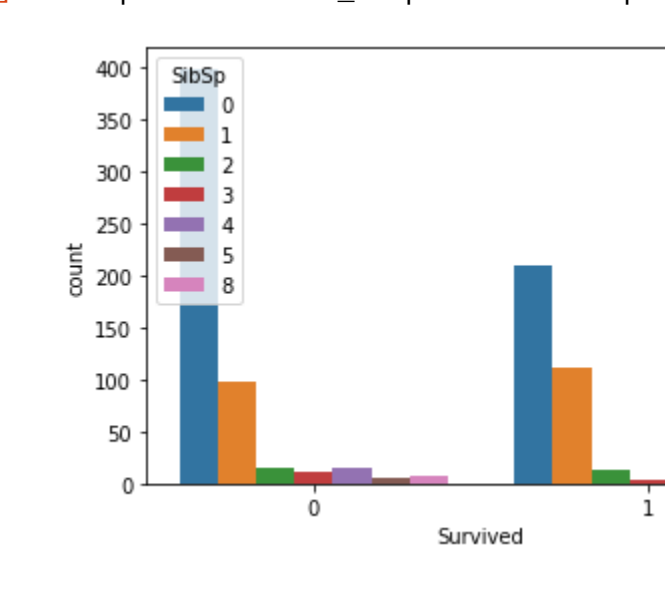


Ratio of females is more than ratio of boys.

Relation between siblings or married couples and survival rate.

In [16]:
```
sns.countplot(x='Survived',hue='SibSp',data=train)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22bb9d482c8>
```

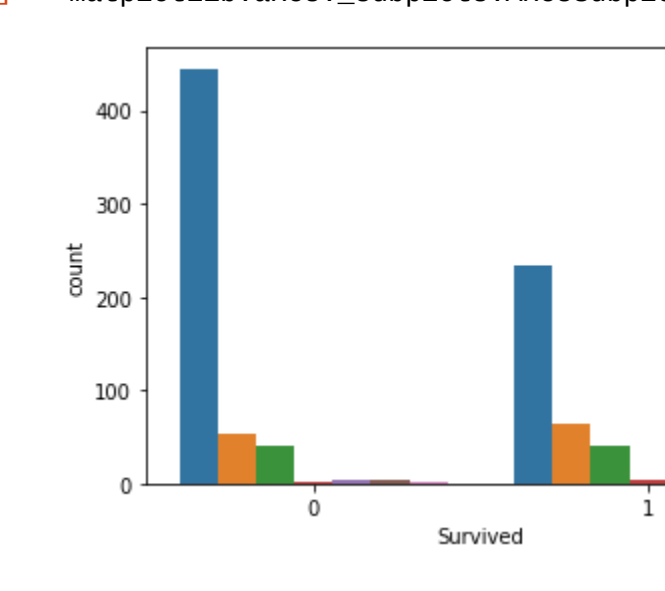


We are not able to draw any conclusion from this graph.

Relation between passengers who had their parents or children on board and survival rate.

In [17]:
```
sns.countplot(x='Survived',hue='Parch',data=train)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22bb9de5108>
```



We are not able to draw any conclusion from this graph.

Converting categorical data into indicator data

In [18]:
```
sex=pd.get_dummies(train['Sex'],drop_first=True)
embarked=pd.get_dummies(train['Embarked'],drop_first=True)
```

Concatinating this data into our train dataframe.

In [19]:
```
train=pd.concat([train,sex,embarked],axis=1)
train.head()
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	male	Q	S
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	S	1	0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C	0	0	0
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	S	0	0	1
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	S	0	0	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	S	1	0	1

Separating our required information

We do not require name and ticket number of our passengers as they do not provide any useful information that could predict their survival.Also, we need to remove Sex and Embarked column as we have already converted them into useful data.

In [20]:
```
train.drop(['Name','Sex','Ticket','Embarked'],axis=1,inplace=True)
```

In [21]:
```
train.head()
```

Out[21]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	1	0	3	22.0	1	0	7.2500	1	0	1
1	2	1	1	38.0	1	0	71.2833	0	0	0
2	3	1	3	26.0	0	0	7.9250	0	0	1
3	4	1	1	35.0	1	0	53.1000	0	0	1
4	5	0	3	35.0	0	0	8.0500	1	0	1

Building our model

In [22]:
```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier()
```

Separating our features and target variable.

In [23]:
```
x=train.drop('Survived',axis=1)
y=train['Survived']
```

Splitting into training and testing data.

In [24]:
```
classifier.fit(x,y)
```

Out[24]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Training our model

Dividing our training data into training and testing set.

In [25]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [26]:
```
classifier.fit(x_train,y_train)
```

Out[26]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Making predictions.

In [27]:
```
test_predictions=classifier.predict(x_test)
```

Finding accuracy

Importing accuracy_score from metrics library.

In [28]:
```
from sklearn.metrics import accuracy_score
```

In [29]:
```
accuracy=accuracy_score(y_test,test_predictions)
accuracy
```

Out[29]:
```
0.770949720670391
```

Importing testing data

In [30]:
```
test=pd.read_csv('test.csv')
```

In [31]:
```
test.head()
```

Out[31]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Checking for missing values in our testing data

In [32]:
```
test.isnull().sum()
```

Out[32]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

Modifying our testing data

Filling null values in age column.

In [33]:
```
test['Age']=test[['Pclass','Age']].apply(fill_age,axis=1)
```

In [34]:
```
test.isnull().sum()
```

Out[34]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

Dropping cabin column.

In [35]:
```
test.drop('Cabin',axis=1,inplace=True)
```

In [36]:
```
test.isnull().sum()
```

Out[36]:
```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           1
Embarked       0
dtype: int64
```

Converting categorical data into indicator data.

In [37]:
```
test_sex=pd.get_dummies(test['Sex'],drop_first=True)
test_embarked=pd.get_dummies(test['Embarked'],drop_first=True)
```

In [38]:
```
test=pd.concat([test,test_sex,test_embarked],axis=1)
test.head()
```

Out[38]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	male	Q	S
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	Q	1	1	0
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	S	0	0	1
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	Q	1	1	0
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	S	1	0	1
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	S	0	0	1

Selecting our required fields.

In [39]:
```
test.drop(['Name','Sex','Ticket','Embarked'],axis=1,inplace=True)
```

In [40]:
```
test.head()
```

Out[40]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	male	Q	S
0	892	3	34.5	0	0	7.8292	1	1	0
1	893	3	47.0	1	0	7.0000	0	0	1
2	894	2	62.0	0	0	9.6875	1	1	0
3	895	3	27.0	0	0	8.6625	1	0	1
4	896	3	22.0	1	1	12.2875	0	0	1

In [41]:
```
test.isnull().sum()
```

Out[41]:
```
PassengerId    0
Pclass         0
Age            0
SibSp          0
Parch          0
Fare           1
male           0
Q              0
S              0
dtype: int64
```

We can see that we have one null value in our fare column.So,we can replace it with 0.

In [42]:
```
def fill_fare(cols):
    fare=cols[0]

    if pd.isnull(fare):
        return 0

    return fare
```

In [43]:
```
test['Fare']=test[['Fare']].apply(fill_fare,axis=1)
```

Checking our data again.

In [44]:
```
test.isnull().sum()
```

Out[44]:
```
PassengerId    0
Pclass         0
Age            0
SibSp          0
Parch          0
Fare           0
male           0
Q              0
S              0
dtype: int64
```

Our data is now ready as we do not have any more null values left.

Making predictions on our testing data

In [45]:
```
predictions=classifier.predict(test)
```

In [46]:
```
predictions
```

Out[46]:
```
array([0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
```