

CIND820_Capstone_Project_v03_SupervisedML

July 21, 2023

0.1 CIND820 - Capstone Project

1 Investigate Airline passenger satisfaction using Machine Learning Techniques

2 Supervised Machine Learning

3 Preparation:

```
[ ]: ! python -V
```

Python 3.10.6

```
[ ]: pip install pandas-profiling
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
    and should_run_async(code)
```

Collecting pandas-profiling

Downloading pandas_profiling-3.6.6-py2.py3-none-any.whl (324 kB)
324.4/324.4

kB 2.5 MB/s eta 0:00:00

Collecting ydata-profiling (from pandas-profiling)

Downloading ydata_profiling-4.3.2-py2.py3-none-any.whl (352 kB)
353.0/353.0 kB

16.0 MB/s eta 0:00:00

Requirement already satisfied: scipy<1.11,>=1.4.1 in

/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(1.10.1)

Requirement already satisfied: pandas!=1.4.0,<2.1,>1.1 in

/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(1.5.3)

Requirement already satisfied: matplotlib<4,>=3.2 in

/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)

```

(3.7.1)
Requirement already satisfied: pydantic<2,>=1.8.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(1.10.11)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(6.0)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(3.1.2)
Collecting visions[type_image_path]==0.7.5 (from ydata-profiling->pandas-
profiling)
  Downloading visions-0.7.5-py3-none-any.whl (102 kB)
                                102.7/102.7 kB
10.2 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.24,>=1.16.0 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(1.22.4)
Collecting htmlmin==0.1.12 (from ydata-profiling->pandas-profiling)
  Downloading htmlmin-0.1.12.tar.gz (19 kB)
  Preparing metadata (setup.py) ... done
Collecting phik<0.13,>=0.11.1 (from ydata-profiling->pandas-profiling)
  Downloading
phik-0.12.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (679 kB)
                                679.5/679.5 kB
16.1 MB/s eta 0:00:00
Requirement already satisfied: requests<3,>=2.24.0 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(2.27.1)
Requirement already satisfied: tqdm<5,>=4.48.2 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(4.65.0)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(0.12.2)
Collecting multimethod<2,>=1.4 (from ydata-profiling->pandas-profiling)
  Downloading multimethod-1.9.1-py3-none-any.whl (10 kB)
Requirement already satisfied: statsmodels<1,>=0.13.2 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling->pandas-profiling)
(0.13.5)
Collecting typeguard<3,>=2.13.2 (from ydata-profiling->pandas-profiling)
  Downloading typeguard-2.13.3-py3-none-any.whl (17 kB)
Collecting imagehash==4.3.1 (from ydata-profiling->pandas-profiling)
  Downloading ImageHash-4.3.1-py2.py3-none-any.whl (296 kB)
                                296.5/296.5 kB
27.0 MB/s eta 0:00:00
Collecting wordcloud>=1.9.1 (from ydata-profiling->pandas-profiling)
  Downloading

```

wordcloud-1.9.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (455 kB)

455.4/455.4 kB

16.2 MB/s eta 0:00:00

Collecting dacite>=1.8 (from ydata-profiling->pandas-profiling)

Downloading dacite-1.8.1-py3-none-any.whl (14 kB)

Requirement already satisfied: PyWavelets in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata-profiling->pandas-profiling) (1.4.1)

Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata-profiling->pandas-profiling) (8.4.0)

Requirement already satisfied: attrs>=19.3.0 in /usr/local/lib/python3.10/dist-packages (from visions[type_image_path]==0.7.5->ydata-profiling->pandas-profiling) (23.1.0)

Requirement already satisfied: networkx>=2.4 in /usr/local/lib/python3.10/dist-packages (from visions[type_image_path]==0.7.5->ydata-profiling->pandas-profiling) (3.1)

Collecting tangled-up-in-unicode>=0.0.4 (from visions[type_image_path]==0.7.5->ydata-profiling->pandas-profiling)

Downloading tangled_up_in_unicode-0.2.0-py3-none-any.whl (4.7 MB)

4.7/4.7 MB

64.2 MB/s eta 0:00:00

Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2<3.2,>=2.11.1->ydata-profiling->pandas-profiling) (2.1.3)

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling->pandas-profiling) (1.1.0)

Requirement already satisfied: cyclor>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling->pandas-profiling) (0.11.0)

Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling->pandas-profiling) (4.41.0)

Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling->pandas-profiling) (1.4.4)

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling->pandas-profiling) (23.1)

Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling->pandas-profiling) (3.1.0)

Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.2->ydata-profiling->pandas-profiling) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.4.0,<2.1,>1.1->ydata-profiling->pandas-profiling) (2022.7.1)

Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.10/dist-

```

packages (from phik<0.13,>=0.11.1->ydata-profiling->pandas-profiling) (1.3.1)
Requirement already satisfied: typing-extensions>=4.2.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic<2,>=1.8.1->ydata-
profiling->pandas-profiling) (4.7.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-
profiling->pandas-profiling) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-
profiling->pandas-profiling) (2023.5.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-
profiling->pandas-profiling) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.24.0->ydata-profiling->pandas-profiling) (3.4)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-
packages (from statsmodels<1,>=0.13.2->ydata-profiling->pandas-profiling)
(0.5.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from patsy>=0.5.2->statsmodels<1,>=0.13.2->ydata-profiling->pandas-profiling)
(1.16.0)
Building wheels for collected packages: htmlmin
  Building wheel for htmlmin (setup.py) ... done
  Created wheel for htmlmin: filename=htmlmin-0.1.12-py3-none-any.whl size=27081
sha256=0414dd75d29e240c9ef2c76161341379ffc1838e94ede99ccf3c60ac965e15fe
  Stored in directory: /root/.cache/pip/wheels/dd/91/29/a79cecb328d01739e64017b6
fb9a1ab9d8cb1853098ec5966d
Successfully built htmlmin
Installing collected packages: htmlmin, typeguard, tangled-up-in-unicode,
multimethod, dacite, imagehash, wordcloud, visions, phik, ydata-profiling,
pandas-profiling
  Attempting uninstall: wordcloud
    Found existing installation: wordcloud 1.8.2.2
    Uninstalling wordcloud-1.8.2.2:
      Successfully uninstalled wordcloud-1.8.2.2
Successfully installed dacite-1.8.1 htmlmin-0.1.12 imagehash-4.3.1
multimethod-1.9.1 pandas-profiling-3.6.6 phik-0.12.3 tangled-up-in-unicode-0.2.0
typeguard-2.13.3 visions-0.7.5 wordcloud-1.9.2 ydata-profiling-4.3.2

```

```
[ ]: # !pip install --upgrade scikit-learn
```

```
[ ]: pip install tabulate
```

```

Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-
packages (0.8.10)

```

```

[212]: # Importing required libraries

import pandas as pd

import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

from sklearn.model_selection import train_test_split

# from sklearn.preprocessing import LabelEncoder
from operator import itemgetter
from sklearn import preprocessing

from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
import sklearn.feature_selection as fs
import sklearn.datasets as datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import sklearn.metrics as metrics
import matplotlib.pyplot as plt

from sklearn.feature_selection import SelectKBest, f_classif
import sklearn.feature_selection as fs

import sklearn.datasets as datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import sklearn.metrics as metrics
import matplotlib.pyplot as plt

from imblearn.over_sampling import SMOTE
from collections import Counter
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report
from tabulate import tabulate
# from xgboost import XGBClassifier

import sklearn
import time
from resource import getrusage, RUSAGE_SELF
from sklearn.model_selection import RandomizedSearchCV

```

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
import xgboost
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, accuracy_score

import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

from mlxtend.preprocessing import TransactionEncoder

from pandas_profiling import ProfileReport

import warnings
warnings.filterwarnings("ignore")

```

Import csv file (the dataset and the data dictionary)

```

[213]: # Import the dataset
# Use first column "ID" as Index by using index_col=0

url = 'https://raw.githubusercontent.com/HitomiMo/CIND820_Capstone-Project/main/
↪airline_passenger_satisfaction.csv'
df1 = pd.read_csv(url, index_col=0)
df1.head()

```

```

[213]:
   Gender  Age Customer Type  Type of Travel  Class  Flight Distance \
ID
1    Male   48   First-time      Business  Business             821
2  Female   35    Returning      Business  Business             821
3    Male   41    Returning      Business  Business             853
4    Male   50    Returning      Business  Business            1905
5  Female   49    Returning      Business  Business            3470

   Departure Delay  Arrival Delay  Departure and Arrival Time Convenience \

```

ID				
1	2	5.0		3
2	26	39.0		2
3	0	0.0		4
4	0	0.0		2
5	0	1.0		3

	Ease of Online Booking	...	On-board Service	Seat Comfort	\
ID		...			
1	3	...	3	5	
2	2	...	5	4	
3	4	...	3	5	
4	2	...	5	5	
5	3	...	3	4	

	Leg Room Service	Cleanliness	Food and Drink	In-flight Service	\
ID					
1	2	5	5	5	
2	5	5	3	5	
3	3	5	5	3	
4	5	4	4	5	
5	4	5	4	3	

	In-flight Wifi Service	In-flight Entertainment	Baggage Handling	\
ID				
1	3	5	5	
2	2	5	5	
3	4	3	3	
4	2	5	5	
5	3	3	3	

	Satisfaction
ID	
1	Neutral or Dissatisfied
2	Satisfied
3	Satisfied
4	Satisfied
5	Satisfied

[5 rows x 23 columns]

```
[214]: # check the data
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129880 entries, 1 to 129880
Data columns (total 23 columns):
```

#	Column	Non-Null Count	Dtype
0	Gender	129880 non-null	object
1	Age	129880 non-null	int64
2	Customer Type	129880 non-null	object
3	Type of Travel	129880 non-null	object
4	Class	129880 non-null	object
5	Flight Distance	129880 non-null	int64
6	Departure Delay	129880 non-null	int64
7	Arrival Delay	129487 non-null	float64
8	Departure and Arrival Time Convenience	129880 non-null	int64
9	Ease of Online Booking	129880 non-null	int64
10	Check-in Service	129880 non-null	int64
11	Online Boarding	129880 non-null	int64
12	Gate Location	129880 non-null	int64
13	On-board Service	129880 non-null	int64
14	Seat Comfort	129880 non-null	int64
15	Leg Room Service	129880 non-null	int64
16	Cleanliness	129880 non-null	int64
17	Food and Drink	129880 non-null	int64
18	In-flight Service	129880 non-null	int64
19	In-flight Wifi Service	129880 non-null	int64
20	In-flight Entertainment	129880 non-null	int64
21	Baggage Handling	129880 non-null	int64
22	Satisfaction	129880 non-null	object

dtypes: float64(1), int64(17), object(5)

memory usage: 23.8+ MB

From “Departure and Arrival Time Convenience” to Baggage Handling” are categorical valuable. The values are satisfaction level from 1 (lowest) to 5 (highest) - 0 means “not applicable”. Therefore, we convert the data type of them from int64 to category.

```
[215]: num_cols = ["Age","Flight Distance","Departure Delay","Arrival Delay"]
cat_cols = list(set(df1.columns) - set(num_cols))
## cat_columns = ["Gender","Customer Type","Type of Travel","Class","Departure
↳and Arrival Time Convenience","Ease of Online Booking","Check-in
↳Service","Online Boarding","Gate Location","On-board Service","Seat
↳Comfort","Leg Room Service","Cleanliness","Food and Drink", "In-flight
↳Service","In-flight Wifi Service","In-flight Entertainment","Baggage
↳Handling","Satisfaction"]
display(len(cat_cols), len(cat_cols))
```

19

19


```
[216]: # check the data after change the data type
for clmn in cat_cols:
    df1[clmn] = df1[clmn].astype('category')
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129880 entries, 1 to 129880
Data columns (total 23 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Gender                                     129880 non-null  category
1   Age                                         129880 non-null  int64
2   Customer Type                             129880 non-null  category
3   Type of Travel                            129880 non-null  category
4   Class                                       129880 non-null  category
5   Flight Distance                           129880 non-null  int64
6   Departure Delay                           129880 non-null  int64
7   Arrival Delay                             129487 non-null  float64
8   Departure and Arrival Time Convenience    129880 non-null  category
9   Ease of Online Booking                    129880 non-null  category
10  Check-in Service                          129880 non-null  category
11  Online Boarding                           129880 non-null  category
12  Gate Location                             129880 non-null  category
13  On-board Service                          129880 non-null  category
14  Seat Comfort                              129880 non-null  category
15  Leg Room Service                          129880 non-null  category
16  Cleanliness                               129880 non-null  category
17  Food and Drink                            129880 non-null  category
18  In-flight Service                         129880 non-null  category
19  In-flight Wifi Service                    129880 non-null  category
20  In-flight Entertainment                   129880 non-null  category
21  Baggage Handling                          129880 non-null  category
22  Satisfaction                              129880 non-null  category
dtypes: category(19), float64(1), int64(3)
memory usage: 7.3 MB
```

```
[217]: # Import the dictionary

url2 = 'https://raw.githubusercontent.com/HitomiMo/CIND820_Capstone-Project/
      ↪main/data_dictionary.csv'
data_dictionary = pd.read_csv(url2, index_col=0)
data_dictionary
```

```
[217]: Description
Field
ID Unique passenger
identifier
```

Gender (Female/Male)	Gender of the passenger
Age passenger	Age of the passenger
Customer Type time/Returning)	Type of airline customer (First-
Type of Travel (Business/Personal)	Purpose of the flight
Class passenger...	Travel class in the airplane for the passenger...
Flight Distance in miles	Flight distance in miles
Departure Delay in minutes	Flight departure delay in minutes
Arrival Delay in minutes	Flight arrival delay in minutes
Departure and Arrival Time Convenience of the...	Satisfaction level with the convenience of the...
Ease of Online Booking booking exp...	Satisfaction level with the online booking exp...
Check-in Service service f...	Satisfaction level with the check-in service f...
Online Boarding boarding ex...	Satisfaction level with the online boarding ex...
Gate Location location in t...	Satisfaction level with the gate location in t...
On-board Service servic...	Satisfaction level with the on-boarding servic...
Seat Comfort the air...	Satisfaction level with the comfort of the air...
Leg Room Service the ai...	Satisfaction level with the leg room of the ai...
Cleanliness of the...	Satisfaction level with the cleanliness of the...
Food and Drink drinks on...	Satisfaction level with the food and drinks on...
In-flight Service service ...	Satisfaction level with the in-flight service ...
In-flight Wifi Service Wifi ser...	Satisfaction level with the in-flight Wifi ser...
In-flight Entertainment entertai...	Satisfaction level with the in-flight entertai...
Baggage Handling handling f...	Satisfaction level with the baggage handling f...
Satisfaction airline (S...	Overall satisfaction level with the airline (S...

4 Exploratory Data Analysis (EDA)

Install pandas-profiling

```
[ ]: prof = ProfileReport(df1)
     prof.to_file(output_file='output.html')
```

Summarize dataset: 0%| | 0/5 [00:00<?, ?it/s]

Generate report structure: 0%| | 0/1 [00:00<?, ?it/s]

Render HTML: 0%| | 0/1 [00:00<?, ?it/s]

Export report to file: 0%| | 0/1 [00:00<?, ?it/s]

```
[218]: # check the first 10 rows
       df1.head(10)
```

```
[218]:
```

	Gender	Age	Customer Type	Type of Travel	Class	Flight Distance	\
ID							
1	Male	48	First-time	Business	Business	821	
2	Female	35	Returning	Business	Business	821	
3	Male	41	Returning	Business	Business	853	
4	Male	50	Returning	Business	Business	1905	
5	Female	49	Returning	Business	Business	3470	
6	Male	43	Returning	Business	Business	3788	
7	Male	43	Returning	Business	Business	1963	
8	Female	60	Returning	Business	Business	853	
9	Male	50	Returning	Business	Business	2607	
10	Female	38	Returning	Business	Business	2822	

	Departure Delay	Arrival Delay	Departure and Arrival Time Convenience	\
ID				
1	2	5.0	3	
2	26	39.0	2	
3	0	0.0	4	
4	0	0.0	2	
5	0	1.0	3	
6	0	0.0	4	
7	0	0.0	3	
8	0	3.0	3	
9	0	0.0	1	
10	13	0.0	2	

	Ease of Online Booking	... On-board Service	Seat Comfort	Leg Room Service	\
ID		...			
1	3	...	3	5	2
2	2	...	5	4	5
3	4	...	3	5	3

4	2	...	5	5	5
5	3	...	3	4	4
6	4	...	4	4	4
7	3	...	5	5	5
8	4	...	3	4	4
9	1	...	4	3	4
10	5	...	5	4	5

	Cleanliness	Food and Drink	In-flight Service	In-flight Wifi Service	\
ID					
1	5	5	5	3	
2	5	3	5	2	
3	5	5	3	4	
4	4	4	5	2	
5	5	4	3	3	
6	3	3	4	4	
7	4	5	5	3	
8	4	4	3	4	
9	3	3	4	4	
10	4	2	5	2	

	In-flight Entertainment	Baggage Handling	Satisfaction
ID			
1	5	5	Neutral or Dissatisfied
2	5	5	Satisfied
3	3	3	Satisfied
4	5	5	Satisfied
5	3	3	Satisfied
6	4	4	Satisfied
7	5	5	Satisfied
8	3	3	Satisfied
9	4	4	Neutral or Dissatisfied
10	5	5	Satisfied

[10 rows x 23 columns]

```
[219]: df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129880 entries, 1 to 129880
Data columns (total 23 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Gender                                129880 non-null  category
1   Age                                   129880 non-null  int64
2   Customer Type                         129880 non-null  category
3   Type of Travel                        129880 non-null  category
```

4	Class	129880	non-null	category
5	Flight Distance	129880	non-null	int64
6	Departure Delay	129880	non-null	int64
7	Arrival Delay	129487	non-null	float64
8	Departure and Arrival Time Convenience	129880	non-null	category
9	Ease of Online Booking	129880	non-null	category
10	Check-in Service	129880	non-null	category
11	Online Boarding	129880	non-null	category
12	Gate Location	129880	non-null	category
13	On-board Service	129880	non-null	category
14	Seat Comfort	129880	non-null	category
15	Leg Room Service	129880	non-null	category
16	Cleanliness	129880	non-null	category
17	Food and Drink	129880	non-null	category
18	In-flight Service	129880	non-null	category
19	In-flight Wifi Service	129880	non-null	category
20	In-flight Entertainment	129880	non-null	category
21	Baggage Handling	129880	non-null	category
22	Satisfaction	129880	non-null	category

dtypes: category(19), float64(1), int64(3)

memory usage: 7.3 MB

Observations: * Number of variable: 23 * Number of entries: 129880

Check missing data

```
[220]: missing_values = pd.isnull(df1)
missing_values.head()
```

```
[220]:
```

	Gender	Age	Customer Type	Type of Travel	Class	Flight Distance	\
ID							
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
5	False	False	False	False	False	False	

	Departure Delay	Arrival Delay	Departure and Arrival Time Convenience	\
ID				
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	
5	False	False	False	

	Ease of Online Booking	...	On-board Service	Seat Comfort	\
ID					
1	False	...	False	False	

2	False	...	False	False
3	False	...	False	False
4	False	...	False	False
5	False	...	False	False

	Leg Room Service	Cleanliness	Food and Drink	In-flight Service \
ID				
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
5	False	False	False	False

	In-flight Wifi Service	In-flight Entertainment	Baggage Handling \
ID			
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
5	False	False	False

	Satisfaction
ID	
1	False
2	False
3	False
4	False
5	False

[5 rows x 23 columns]

```
[221]: df1.isnull().sum()
```

```
[221]: Gender          0
Age                0
Customer Type      0
Type of Travel     0
Class              0
Flight Distance    0
Departure Delay    0
Arrival Delay      393
Departure and Arrival Time Convenience  0
Ease of Online Booking  0
Check-in Service   0
Online Boarding    0
Gate Location      0
On-board Service   0
```

Seat Comfort	0
Leg Room Service	0
Cleanliness	0
Food and Drink	0
In-flight Service	0
In-flight Wifi Service	0
In-flight Entertainment	0
Baggage Handling	0
Satisfaction	0
dtype: int64	

Check description of the data

```
[222]: df1.describe()
```

```
[222]:
```

	Age	Flight Distance	Departure Delay	Arrival Delay
count	129880.000000	129880.000000	129880.000000	129487.000000
mean	39.427957	1190.316392	14.713713	15.091129
std	15.119360	997.452477	38.071126	38.465650
min	7.000000	31.000000	0.000000	0.000000
25%	27.000000	414.000000	0.000000	0.000000
50%	40.000000	844.000000	0.000000	0.000000
75%	51.000000	1744.000000	12.000000	13.000000
max	85.000000	4983.000000	1592.000000	1584.000000

5 Supervised Machine Learning

5.1 Data preparation for Supervised Machine Learning

Remove missing values

```
[223]: df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129880 entries, 1 to 129880
Data columns (total 23 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Gender                                129880 non-null  category
1   Age                                    129880 non-null  int64
2   Customer Type                         129880 non-null  category
3   Type of Travel                        129880 non-null  category
4   Class                                 129880 non-null  category
5   Flight Distance                       129880 non-null  int64
6   Departure Delay                       129880 non-null  int64
7   Arrival Delay                         129487 non-null  float64
8   Departure and Arrival Time Convenience 129880 non-null  category
9   Ease of Online Booking                 129880 non-null  category
```

```

10 Check-in Service          129880 non-null category
11 Online Boarding          129880 non-null category
12 Gate Location            129880 non-null category
13 On-board Service         129880 non-null category
14 Seat Comfort             129880 non-null category
15 Leg Room Service         129880 non-null category
16 Cleanliness              129880 non-null category
17 Food and Drink           129880 non-null category
18 In-flight Service        129880 non-null category
19 In-flight Wifi Service   129880 non-null category
20 In-flight Entertainment  129880 non-null category
21 Baggage Handling         129880 non-null category
22 Satisfaction            129880 non-null category
dtypes: category(19), float64(1), int64(3)
memory usage: 7.3 MB

```

```
[224]: df1['Arrival Delay'].isnull().sum()
```

```
[224]: 393
```

```

[225]: # Checking % of missing values
percent_missing = df1['Arrival Delay'].isnull().sum() * 100 / len(df1['Arrival_
↳ Delay'])
percent_missing

```

```
[225]: 0.3025870033877425
```

```

[226]: # Missing value ratio is only 0.3%. Therefore, I will remove missing values_
↳ before splitting dataset
df2 = df1.copy()
df2 = df2.dropna(how='any',axis=0)
df2.isnull().sum()

```

```

[226]: Gender          0
Age                  0
Customer Type       0
Type of Travel      0
Class               0
Flight Distance     0
Departure Delay     0
Arrival Delay       0
Departure and Arrival Time Convenience 0
Ease of Online Booking 0
Check-in Service    0
Online Boarding     0
Gate Location       0
On-board Service    0

```



```

Seat Comfort                                0
Leg Room Service                           0
Cleanliness                                0
Food and Drink                             0
In-flight Service                          0
In-flight Wifi Service                     0
In-flight Entertainment                    0
Baggage Handling                           0
Satisfaction                              0
dtype: int64

```

```

[227]: # check the # of entries after removing missing values
row_count = len(df2.index)
row_count

```

```

[227]: 129487

```

```

[228]: # Create on-hot-key for the categorical variables using the function pd.
↳ get_dummies()
y = df2['Satisfaction']
df3 = df2.drop(['Satisfaction'], axis=1)
if 'Satisfaction' in cat_cols:
    cat_cols.remove('Satisfaction')
df3 = pd.get_dummies(df3, columns=cat_cols, prefix = cat_cols)
X = df3.copy()
X.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 129487 entries, 1 to 129880
Data columns (total 96 columns):

```

#	Column	Non-Null Count	Dtype
0	Age	129487 non-null	int64
1	Flight Distance	129487 non-null	int64
2	Departure Delay	129487 non-null	int64
3	Arrival Delay	129487 non-null	float64
4	In-flight Service_0	129487 non-null	uint8
5	In-flight Service_1	129487 non-null	uint8
6	In-flight Service_2	129487 non-null	uint8
7	In-flight Service_3	129487 non-null	uint8
8	In-flight Service_4	129487 non-null	uint8
9	In-flight Service_5	129487 non-null	uint8
10	Gate Location_0	129487 non-null	uint8
11	Gate Location_1	129487 non-null	uint8
12	Gate Location_2	129487 non-null	uint8
13	Gate Location_3	129487 non-null	uint8
14	Gate Location_4	129487 non-null	uint8

15	Gate Location_5	129487	non-null	uint8
16	Ease of Online Booking_0	129487	non-null	uint8
17	Ease of Online Booking_1	129487	non-null	uint8
18	Ease of Online Booking_2	129487	non-null	uint8
19	Ease of Online Booking_3	129487	non-null	uint8
20	Ease of Online Booking_4	129487	non-null	uint8
21	Ease of Online Booking_5	129487	non-null	uint8
22	Online Boarding_0	129487	non-null	uint8
23	Online Boarding_1	129487	non-null	uint8
24	Online Boarding_2	129487	non-null	uint8
25	Online Boarding_3	129487	non-null	uint8
26	Online Boarding_4	129487	non-null	uint8
27	Online Boarding_5	129487	non-null	uint8
28	Class_Business	129487	non-null	uint8
29	Class_Economy	129487	non-null	uint8
30	Class_Economy Plus	129487	non-null	uint8
31	Leg Room Service_0	129487	non-null	uint8
32	Leg Room Service_1	129487	non-null	uint8
33	Leg Room Service_2	129487	non-null	uint8
34	Leg Room Service_3	129487	non-null	uint8
35	Leg Room Service_4	129487	non-null	uint8
36	Leg Room Service_5	129487	non-null	uint8
37	Cleanliness_0	129487	non-null	uint8
38	Cleanliness_1	129487	non-null	uint8
39	Cleanliness_2	129487	non-null	uint8
40	Cleanliness_3	129487	non-null	uint8
41	Cleanliness_4	129487	non-null	uint8
42	Cleanliness_5	129487	non-null	uint8
43	Type of Travel_Business	129487	non-null	uint8
44	Type of Travel_Personal	129487	non-null	uint8
45	In-flight Entertainment_0	129487	non-null	uint8
46	In-flight Entertainment_1	129487	non-null	uint8
47	In-flight Entertainment_2	129487	non-null	uint8
48	In-flight Entertainment_3	129487	non-null	uint8
49	In-flight Entertainment_4	129487	non-null	uint8
50	In-flight Entertainment_5	129487	non-null	uint8
51	On-board Service_0	129487	non-null	uint8
52	On-board Service_1	129487	non-null	uint8
53	On-board Service_2	129487	non-null	uint8
54	On-board Service_3	129487	non-null	uint8
55	On-board Service_4	129487	non-null	uint8
56	On-board Service_5	129487	non-null	uint8
57	Departure and Arrival Time Convenience_0	129487	non-null	uint8
58	Departure and Arrival Time Convenience_1	129487	non-null	uint8
59	Departure and Arrival Time Convenience_2	129487	non-null	uint8
60	Departure and Arrival Time Convenience_3	129487	non-null	uint8
61	Departure and Arrival Time Convenience_4	129487	non-null	uint8
62	Departure and Arrival Time Convenience_5	129487	non-null	uint8

63	Customer Type_First-time	129487	non-null	uint8
64	Customer Type_Returning	129487	non-null	uint8
65	Gender_Female	129487	non-null	uint8
66	Gender_Male	129487	non-null	uint8
67	Baggage Handling_1	129487	non-null	uint8
68	Baggage Handling_2	129487	non-null	uint8
69	Baggage Handling_3	129487	non-null	uint8
70	Baggage Handling_4	129487	non-null	uint8
71	Baggage Handling_5	129487	non-null	uint8
72	Seat Comfort_0	129487	non-null	uint8
73	Seat Comfort_1	129487	non-null	uint8
74	Seat Comfort_2	129487	non-null	uint8
75	Seat Comfort_3	129487	non-null	uint8
76	Seat Comfort_4	129487	non-null	uint8
77	Seat Comfort_5	129487	non-null	uint8
78	Food and Drink_0	129487	non-null	uint8
79	Food and Drink_1	129487	non-null	uint8
80	Food and Drink_2	129487	non-null	uint8
81	Food and Drink_3	129487	non-null	uint8
82	Food and Drink_4	129487	non-null	uint8
83	Food and Drink_5	129487	non-null	uint8
84	Check-in Service_0	129487	non-null	uint8
85	Check-in Service_1	129487	non-null	uint8
86	Check-in Service_2	129487	non-null	uint8
87	Check-in Service_3	129487	non-null	uint8
88	Check-in Service_4	129487	non-null	uint8
89	Check-in Service_5	129487	non-null	uint8
90	In-flight Wifi Service_0	129487	non-null	uint8
91	In-flight Wifi Service_1	129487	non-null	uint8
92	In-flight Wifi Service_2	129487	non-null	uint8
93	In-flight Wifi Service_3	129487	non-null	uint8
94	In-flight Wifi Service_4	129487	non-null	uint8
95	In-flight Wifi Service_5	129487	non-null	uint8

dtypes: float64(1), int64(3), uint8(92)

memory usage: 16.3 MB

Feature Selection

```
[229]: # Finding the best K (optimal # of features for feature selection.
# Use f_classif. This computes the ANOVA F-value for the provided sample.
# use RandomForestClassifier
f1_list = []
for k in range(1,len(df3.columns)):
    bk = fs.SelectKBest(fs.f_classif, k = k)
    bk.fit(X, y)
    x_trans = bk.transform(X)
    #print(type(x_trans))
    #print(x_trans)
```

```

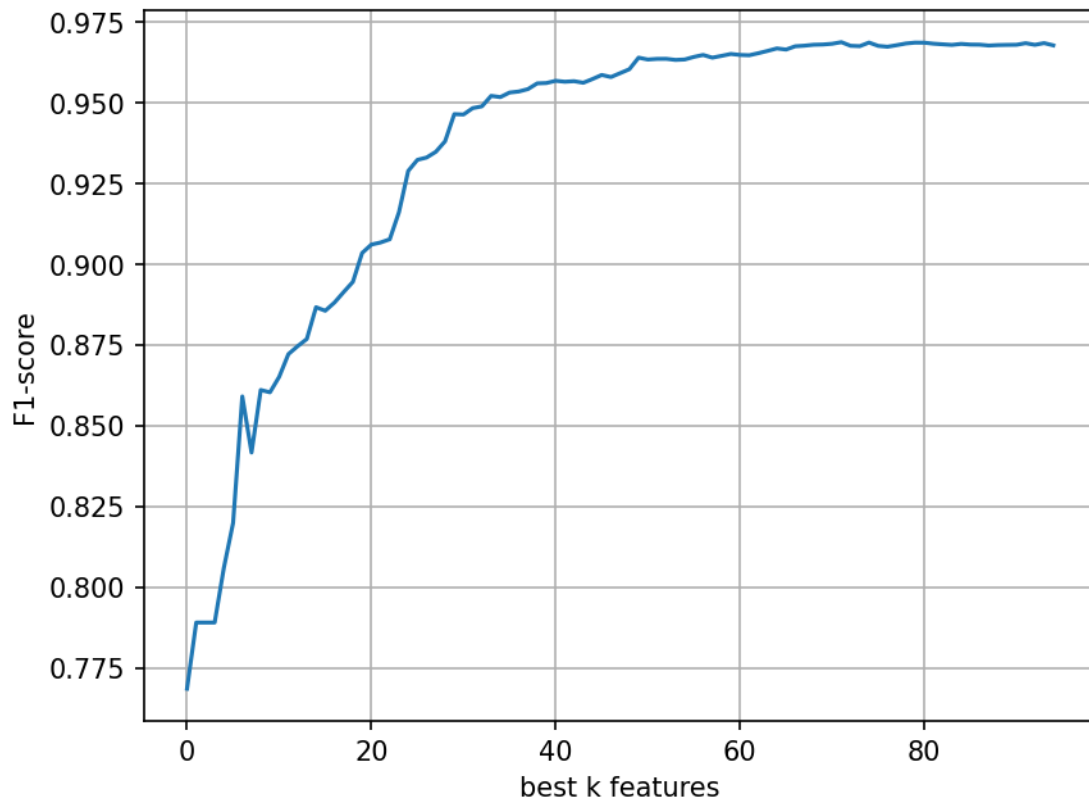
train_x, test_x, train_y, test_y = train_test_split(x_trans,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=42)

rf = RandomForestClassifier()
rf.fit(train_x, train_y)
y_pred = rf.predict(test_x)
#print(test_y)
y_pred01 = y_pred == "Neutral or Dissatisfied"
test_y01 = test_y == "Neutral or Dissatisfied"
#print(y_pred01)
y_pred02 = np.array(y_pred01, dtype=int)
test_y02 = np.array(test_y01, dtype=int)
# print(test_y, y_pred02)
f1 = metrics.f1_score(test_y02, y_pred02)
f1_list.append((f1, k))

fig, axe = plt.subplots(dpi = 150)

# print(type(axe))
f1_list01 = [f1_scr for f1_scr, nf in f1_list]
axe.plot(range(0, len(f1_list01)), f1_list01)
axe.set_xlabel("best k features")
axe.set_ylabel("F1-score")
plt.grid(True)
plt.show()

```



```
[231]: # Sort f1_list using f1_score as a key for sorting
# The best k would be the value of last element in the list [-1]
# Get the last element by using -1 as index
# display(f1_list)
best_k = sorted(f1_list, key=itemgetter(0))
display(best_k[-1][1])
k = best_k[-1][1]
# excluded = []
# for ndx, col in enumerate(f1_list):
#     if col >= 0.9:
#         selected_f1.append(ndx)
#     else:
#         excluded.append(ndx)
# selected_df = df3.iloc[:, selected_f1]
# display(selected_df)
```

72

```
[232]: # Feature selection, applying Select K Best to output the most important_
↪ features
```

```

# Currently the best k value is 72, which means that the Random Forest would
↳ run best on the best 95 variable out of 96
# from sklearn.feature_selection import SelectKBest, f_classif
# y has the column "Satisfaction" as the class column as assigned earlier

X = df3

selector = SelectKBest(f_classif, k = best_k[-1][1])
selector.fit(X, y)
X_new = selector.transform(X)

features = (X.columns[selector.get_support(indices=True)])
features

```

```

[232]: Index(['Age', 'Flight Distance', 'Departure Delay', 'Arrival Delay',
            'In-flight Service_1', 'In-flight Service_2', 'In-flight Service_3',
            'In-flight Service_4', 'In-flight Service_5', 'Gate Location_1',
            'Gate Location_3', 'Gate Location_5', 'Ease of Online Booking_0',
            'Ease of Online Booking_1', 'Ease of Online Booking_2',
            'Ease of Online Booking_3', 'Ease of Online Booking_4',
            'Ease of Online Booking_5', 'Online Boarding_1', 'Online Boarding_2',
            'Online Boarding_3', 'Online Boarding_4', 'Online Boarding_5',
            'Class_Business', 'Class_Economy', 'Class_Economy Plus',
            'Leg Room Service_1', 'Leg Room Service_2', 'Leg Room Service_3',
            'Leg Room Service_4', 'Leg Room Service_5', 'Cleanliness_1',
            'Cleanliness_2', 'Cleanliness_4', 'Cleanliness_5',
            'Type of Travel_Business', 'Type of Travel_Personal',
            'In-flight Entertainment_1', 'In-flight Entertainment_2',
            'In-flight Entertainment_3', 'In-flight Entertainment_4',
            'In-flight Entertainment_5', 'On-board Service_1', 'On-board Service_2',
            'On-board Service_3', 'On-board Service_4', 'On-board Service_5',
            'Departure and Arrival Time Convenience_4', 'Customer Type_First-time',
            'Customer Type_Returning', 'Baggage Handling_1', 'Baggage Handling_2',
            'Baggage Handling_3', 'Baggage Handling_4', 'Baggage Handling_5',
            'Seat Comfort_1', 'Seat Comfort_2', 'Seat Comfort_3', 'Seat Comfort_4',
            'Seat Comfort_5', 'Food and Drink_1', 'Food and Drink_4',
            'Food and Drink_5', 'Check-in Service_1', 'Check-in Service_2',
            'Check-in Service_5', 'In-flight Wifi Service_0',
            'In-flight Wifi Service_1', 'In-flight Wifi Service_2',
            'In-flight Wifi Service_3', 'In-flight Wifi Service_4',
            'In-flight Wifi Service_5'],
           dtype='object')

```

```

[233]: # Assign the best 85 selected feature to a new data frame df4
df4 = df3[list(features)]
display(df4)

```

ID	Age	Flight Distance	Departure Delay	Arrival Delay	\
1	48	821	2	5.0	
2	35	821	26	39.0	
3	41	853	0	0.0	
4	50	1905	0	0.0	
5	49	3470	0	1.0	
...	
129876	28	447	2	3.0	
129877	41	308	0	0.0	
129878	42	337	6	14.0	
129879	50	337	31	22.0	
129880	20	337	0	0.0	

ID	In-flight Service_1	In-flight Service_2	In-flight Service_3	\
1	0	0	0	
2	0	0	0	
3	0	0	1	
4	0	0	0	
5	0	0	1	
...	
129876	0	0	0	
129877	0	0	0	
129878	0	0	0	
129879	0	0	0	
129880	0	1	0	

ID	In-flight Service_4	In-flight Service_5	Gate Location_1	...	\
1	0	1	0	...	
2	0	1	0	...	
3	0	0	0	...	
4	0	1	0	...	
5	0	0	0	...	
...	
129876	0	1	0	...	
129877	1	0	0	...	
129878	1	0	1	...	
129879	1	0	1	...	
129880	0	0	0	...	

ID	Food and Drink_5	Check-in Service_1	Check-in Service_2	\
1	1	0	0	
2	0	0	0	
3	1	0	0	
4	0	0	0	

5	0	0	0
...
129876	0	0	0
129877	0	0	0
129878	0	0	0
129879	0	0	0
129880	0	0	0

	Check-in Service_5	In-flight Wifi Service_0 \
ID		
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
...
129876	0	0
129877	1	0
129878	0	0
129879	0	0
129880	0	0

	In-flight Wifi Service_1	In-flight Wifi Service_2 \
ID		
1	0	0
2	0	1
3	0	0
4	0	1
5	0	0
...
129876	0	0
129877	0	0
129878	0	1
129879	0	0
129880	0	0

	In-flight Wifi Service_3	In-flight Wifi Service_4 \
ID		
1	1	0
2	0	0
3	0	1
4	0	0
5	1	0
...
129876	0	1
129877	1	0
129878	0	0
129879	0	0

129880	1	0
--------	---	---

	In-flight Wifi Service_5
--	--------------------------

ID	
1	0
2	0
3	0
4	0
5	0
...	...
129876	0
129877	0
129878	0
129879	1
129880	0

[129487 rows x 72 columns]

Split the dataset

Split the dataset (df4) into Traing set and Test set after removing missing values

```
[234]: # split the data set into four pieces - X_train, X_test, y_train and y_test.
# randomly sampling without replacement about 80% into training set and 20%
↳ into test set.
# splitting dataset is 80% training set and 20% test set is most common ratio.
X = df4.copy()
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state= 0 ,
↳ test_size=0.20, shuffle=True)
```

```
[235]: # view first few rows of train set
X_train.head()
```

```
[235]:      Age  Flight Distance  Departure Delay  Arrival Delay \
```

ID				
41758	58	230	0	0.0
121068	25	861	0	0.0
43321	42	3508	0	0.0
76604	11	3338	0	0.0
61370	58	679	0	0.0

	In-flight Service_1	In-flight Service_2	In-flight Service_3	\
ID				
41758	0	0	0	
121068	0	0	0	
43321	0	0	0	
76604	0	0	0	
61370	0	0	0	

	In-flight Service_4	In-flight Service_5	Gate Location_1	...	\
ID					...
41758	1	0	1		...
121068	0	1	0		...
43321	1	0	0		...
76604	0	1	0		...
61370	0	1	0		...

	Food and Drink_5	Check-in Service_1	Check-in Service_2	\
ID				
41758	0	0	1	
121068	0	0	0	
43321	1	0	0	
76604	0	0	0	
61370	1	0	0	

	Check-in Service_5	In-flight Wifi Service_0	\
ID			
41758	0	0	
121068	1	0	
43321	0	0	
76604	0	0	
61370	0	0	

	In-flight Wifi Service_1	In-flight Wifi Service_2	\
ID			
41758	0	0	
121068	0	1	
43321	0	0	
76604	0	0	
61370	0	0	

	In-flight Wifi Service_3	In-flight Wifi Service_4	\
ID			
41758	0	1	
121068	0	0	
43321	1	0	
76604	0	0	
61370	1	0	

	In-flight Wifi Service_5
ID	
41758	0
121068	0
43321	0
76604	1

61370 0

[5 rows x 72 columns]

```
[236]: # count # of missing value in X_train set
X_train["Arrival Delay"].isnull().sum()
```

[236]: 0

```
[237]: # view first few rows of test set
y_train.head()
```

```
[237]: ID
41758    Neutral or Dissatisfied
121068           Satisfied
43321           Satisfied
76604           Satisfied
61370           Satisfied
Name: Satisfaction, dtype: category
Categories (2, object): ['Neutral or Dissatisfied', 'Satisfied']
```

```
[238]: # check the size of each set
print(X_train.shape, X_test.shape)
```

(103589, 72) (25898, 72)

Handling outliers of train set

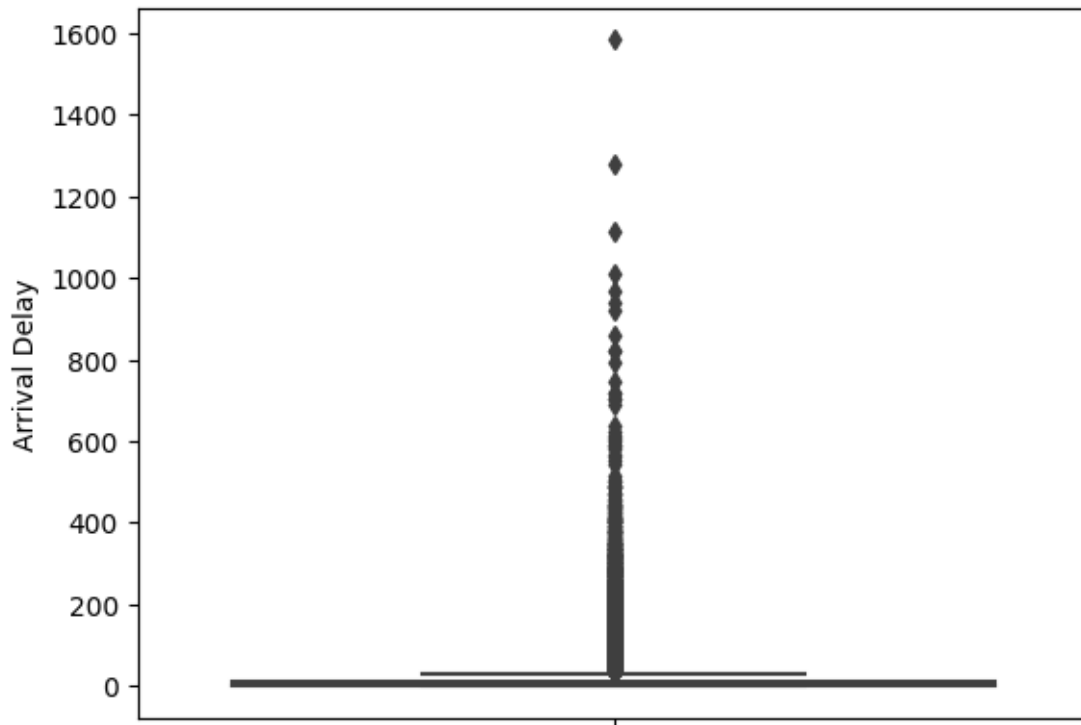
Use IQR (Inter Quartile Range) to finding the outliers and cap the outliers

- capping: to replace the outlier values with a maximum or minimum capped value

Arrival Delay

```
[239]: # before capping outliers
sns.boxplot( y="Arrival Delay", data = X_train)
```

[239]: <Axes: ylabel='Arrival Delay'>



```
[240]: # IQR
Q1 = np.percentile(X_train['Arrival Delay'], 25, method='midpoint')
Q3 = np.percentile(X_train['Arrival Delay'], 75, method='midpoint')
IQR = Q3 - Q1
print(IQR)
```

13.0

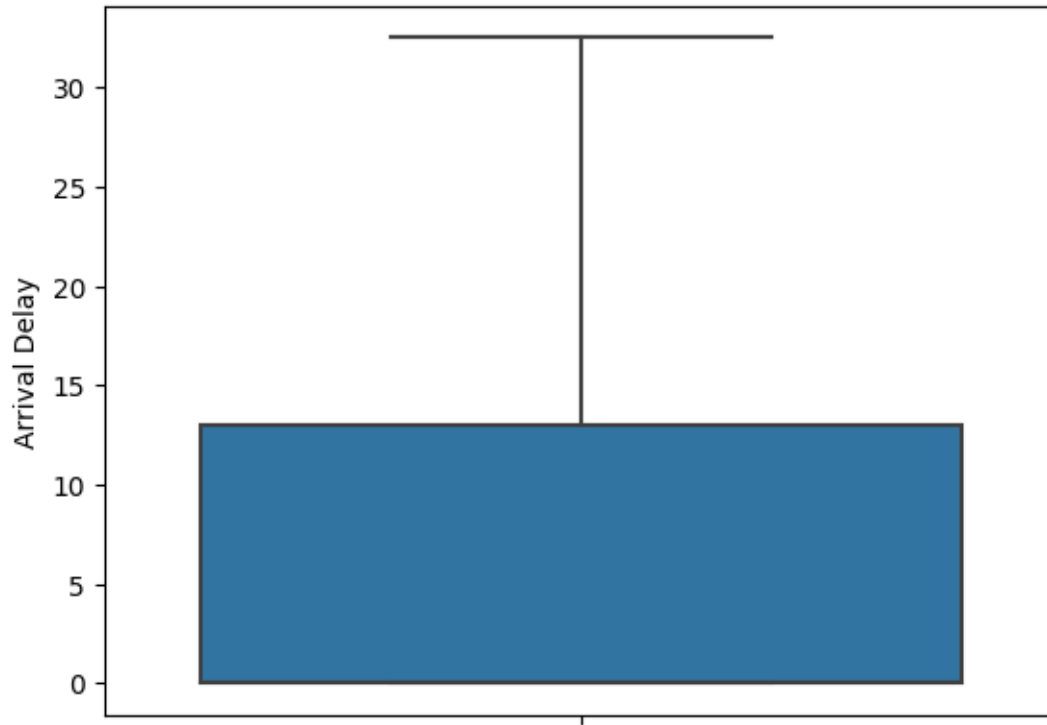
```
[241]: upper_bound = Q3 + 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR
print(upper_bound)
print(lower_bound)
```

32.5
-19.5

```
[242]: X_train["Arrival Delay"] = np.where(X_train["Arrival Delay"] > upper_bound,
↪upper_bound,
                                             np.where(X_train["Arrival Delay"] < lower_bound,
↪lower_bound,
                                             X_train["Arrival Delay"]))
```

```
[243]: # after capping outliers
sns.boxplot( y="Arrival Delay", data = X_train)
```

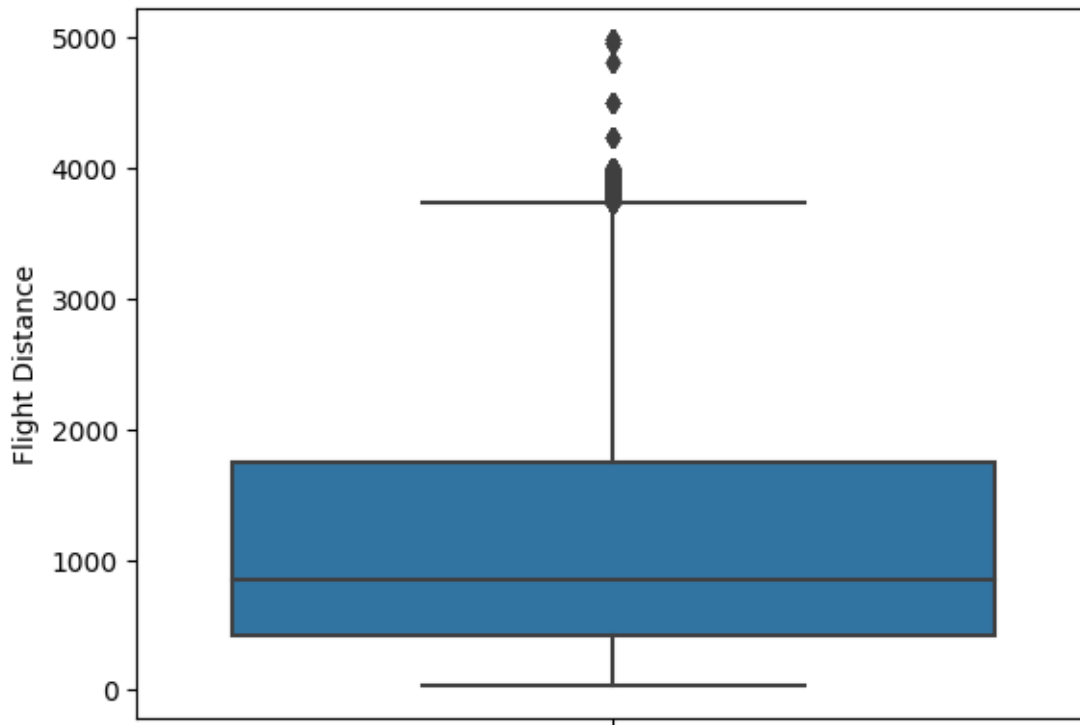
```
[243]: <Axes: ylabel='Arrival Delay'>
```



Flight Distance

```
[244]: # before capping outliers
sns.boxplot( y="Flight Distance", data = X_train)
```

```
[244]: <Axes: ylabel='Flight Distance'>
```



```
[245]: # IQR
Q1 = np.percentile(df1['Flight Distance'], 25, method='midpoint')
Q3 = np.percentile(df1['Flight Distance'], 75, method='midpoint')
IQR = Q3 - Q1
print(IQR)
```

1330.0

```
[246]: upper_bound = Q3 + 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR
print(upper_bound)
print(lower_bound)
```

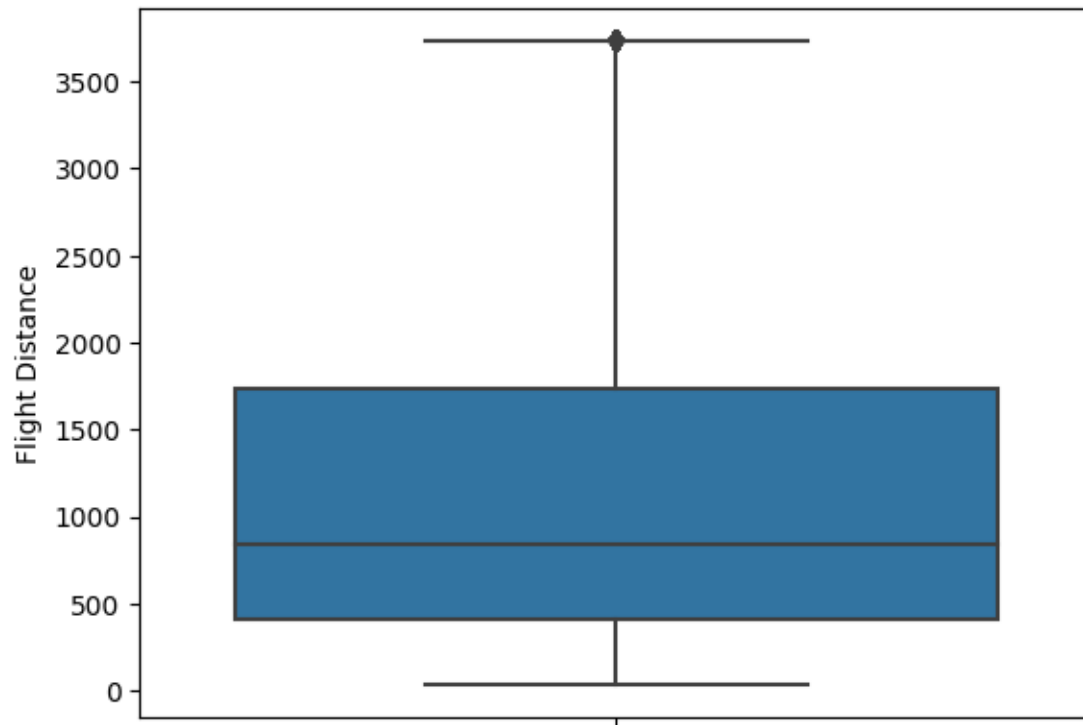
3739.0

-1581.0

```
[247]: X_train["Flight Distance"] = np.where(X_train["Flight Distance"] > upper_bound,
↪upper_bound,
                                             np.where(X_train["Flight Distance"] < lower_bound,
↪lower_bound,
                                             X_train["Flight Distance"]))
```

```
[248]: # after capping outliers
sns.boxplot( y="Flight Distance", data = X_train)
```

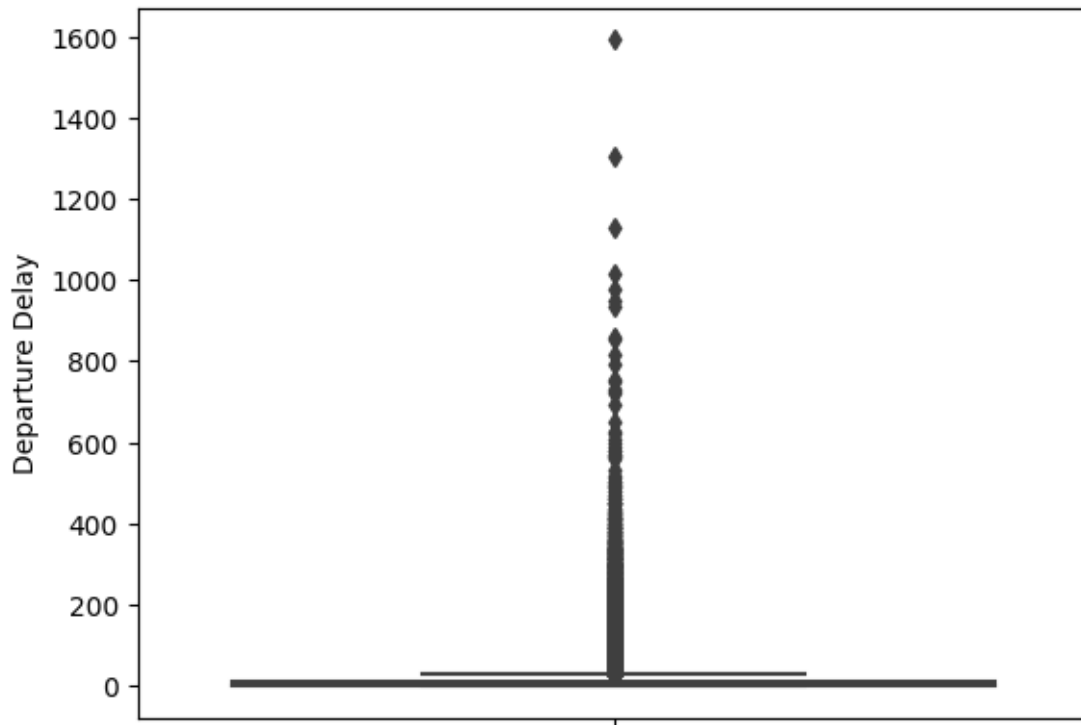
```
[248]: <Axes: ylabel='Flight Distance'>
```



Departure Delay

```
[249]: # before capping outliers
sns.boxplot( y="Departure Delay", data = X_train)
```

```
[249]: <Axes: ylabel='Departure Delay'>
```



```
[250]: # IQR
Q1 = np.percentile(df1['Departure Delay'], 25, method='midpoint')
Q3 = np.percentile(df1['Departure Delay'], 75, method='midpoint')
IQR = Q3 - Q1
print(IQR)
```

12.0

```
[251]: upper_bound = Q3 + 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR
print(upper_bound)
print(lower_bound)
```

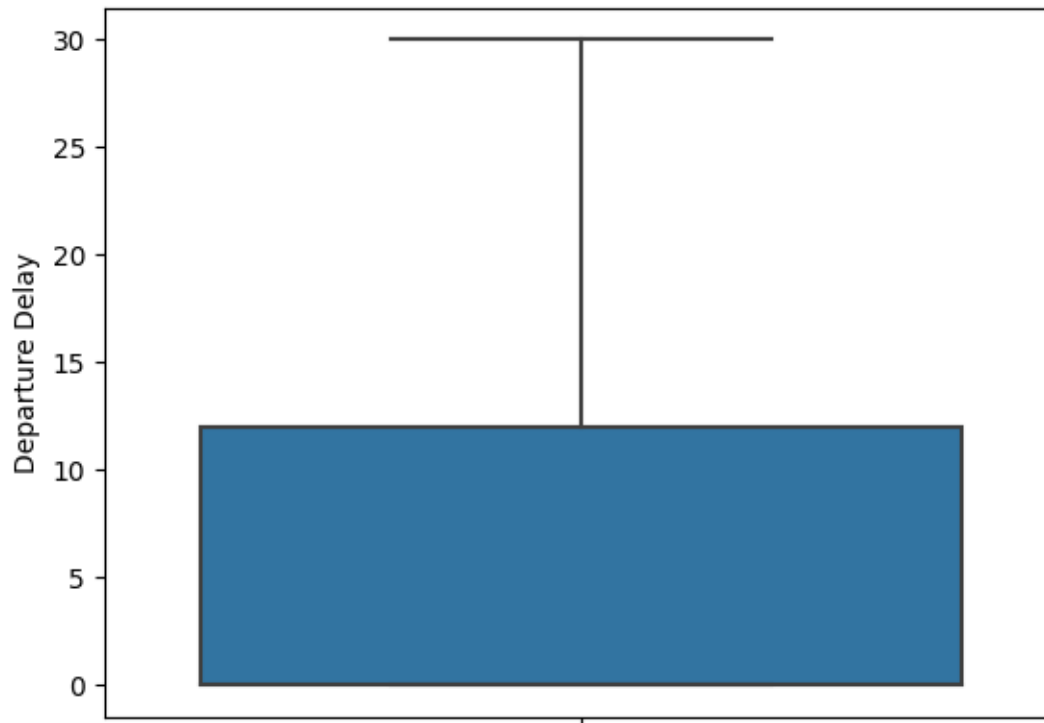
30.0
-18.0

```
[252]: X_train["Departure Delay"] = np.where(X_train["Departure Delay"] > upper_bound,
↪upper_bound,
                                             np.where(X_train["Departure Delay"] < lower_bound,
↪lower_bound,
                                             X_train["Departure Delay"]))
```



```
[253]: # after capping outliers
sns.boxplot( y="Departure Delay", data = X_train)
```

```
[253]: <Axes: ylabel='Departure Delay'>
```



Scaling

```
[254]: # encoded_X_train.info()
num_cols
```

```
[254]: ['Age', 'Flight Distance', 'Departure Delay', 'Arrival Delay']
```

```
[255]: display(X_train[num_cols])
```

	Age	Flight Distance	Departure Delay	Arrival Delay
ID				
41758	58	230.0	0.0	0.0
121068	25	861.0	0.0	0.0
43321	42	3508.0	0.0	0.0
76604	11	3338.0	0.0	0.0
61370	58	679.0	0.0	0.0
...
46070	26	3344.0	0.0	0.0
118313	23	639.0	2.0	2.0

42779	23	744.0	30.0	32.5
43736	34	622.0	30.0	32.5
68504	64	1303.0	18.0	6.0

[103589 rows x 4 columns]

```
[256]: # Pre-processing and scaling dataset for numerical variables
scaler = preprocessing.MinMaxScaler()
# scaler.fit(encoded_X_train[num_cols])
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
```

```
[257]: display(X_train)
```

	Age	Flight Distance	Departure Delay	Arrival Delay	\
ID					
41758	0.653846	0.053668	0.000000	0.000000	
121068	0.230769	0.223840	0.000000	0.000000	
43321	0.448718	0.937702	0.000000	0.000000	
76604	0.051282	0.891855	0.000000	0.000000	
61370	0.653846	0.174757	0.000000	0.000000	
...	
46070	0.243590	0.893474	0.000000	0.000000	
118313	0.205128	0.163970	0.066667	0.061538	
42779	0.205128	0.192287	1.000000	1.000000	
43736	0.346154	0.159385	1.000000	1.000000	
68504	0.730769	0.343042	0.600000	0.184615	

	In-flight Service_1	In-flight Service_2	In-flight Service_3	\
ID				
41758	0	0	0	
121068	0	0	0	
43321	0	0	0	
76604	0	0	0	
61370	0	0	0	
...	
46070	0	1	0	
118313	0	0	0	
42779	0	0	1	
43736	0	0	0	
68504	0	0	0	

	In-flight Service_4	In-flight Service_5	Gate Location_1	...	\
ID					
41758	1	0	1	...	
121068	0	1	0	...	
43321	1	0	0	...	
76604	0	1	0	...	
61370	0	1	0	...	

...
46070	0	0	0	...
118313	1	0	0	...
42779	0	0	0	...
43736	0	1	0	...
68504	0	1	1	...

ID	Food and Drink_5	Check-in Service_1	Check-in Service_2	\
41758	0	0	1	
121068	0	0	0	
43321	1	0	0	
76604	0	0	0	
61370	1	0	0	
...	
46070	0	0	0	
118313	1	1	0	
42779	0	0	0	
43736	0	0	0	
68504	0	0	0	

ID	Check-in Service_5	In-flight Wifi Service_0	\
41758	0	0	
121068	1	0	
43321	0	0	
76604	0	0	
61370	0	0	
...	
46070	0	0	
118313	0	0	
42779	0	0	
43736	1	0	
68504	0	0	

ID	In-flight Wifi Service_1	In-flight Wifi Service_2	\
41758	0	0	
121068	0	1	
43321	0	0	
76604	0	0	
61370	0	0	
...	
46070	0	0	
118313	0	0	
42779	0	0	
43736	0	1	
68504	0	0	

ID	In-flight Wifi Service_3	In-flight Wifi Service_4 \
41758	0	1
121068	0	0
43321	1	0
76604	0	0
61370	1	0
...
46070	1	0
118313	1	0
42779	0	1
43736	0	0
68504	0	1

ID	In-flight Wifi Service_5
41758	0
121068	0
43321	0
76604	1
61370	0
...	...
46070	0
118313	0
42779	0
43736	0
68504	0

[103589 rows x 72 columns]

(Absute, please review the cells above) Keep only selected 14 features in test set for build model in following process

```
[ ]: # selected_X_test = encoded_X_test[['Type of Travel', 'Class', 'Flight
↳Distance', 'Check-in Service',
#                               'Online Boarding', 'On-board Service',
↳'Seat Comfort',
#                               'Leg Room Service', 'Cleanliness', 'Food
↳and Drink',
#                               'In-flight Service', 'In-flight Wifi
↳Service',
#                               'In-flight Entertainment', 'Baggage
↳Handling']].copy()
```

```
[ ]: # selected_X_test.info()
```

```
[ ]: print(type(selected_X_test))
```

```
<class 'pandas.core.frame.DataFrame'>
```

Applying undersampling, oversampling and SMOTE to address the imbalance in the target class, “Satisfaction”.

Applying SMOTE

```
[258]: # from imblearn.over_sampling import SMOTE
# from collections import Counter
# define dataset
X_SMOTE = X_train.copy()
y_SMOTE = y_train.copy()
# summarize class distribution
counter = Counter(y_SMOTE)
print('Before SMOTE', (counter))
# transform the dataset
smotesample = SMOTE()
X_SMOTE, y_SMOTE = smotesample.fit_resample(X_SMOTE, y_SMOTE)
# summarize the new class distribution
counter = Counter(y_SMOTE)
print('After SMOTE', (counter))
```

Before SMOTE Counter({'Neutral or Dissatisfied': 58687, 'Satisfied': 44902})

After SMOTE Counter({'Neutral or Dissatisfied': 58687, 'Satisfied': 58687})

```
[259]: SMOTE_X_train = X_SMOTE.copy()
```

```
[260]: SMOTE_X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 117374 entries, 0 to 117373
```

```
Data columns (total 72 columns):
```

#	Column	Non-Null Count	Dtype
0	Age	117374 non-null	float64
1	Flight Distance	117374 non-null	float64
2	Departure Delay	117374 non-null	float64
3	Arrival Delay	117374 non-null	float64
4	In-flight Service_1	117374 non-null	uint8
5	In-flight Service_2	117374 non-null	uint8
6	In-flight Service_3	117374 non-null	uint8
7	In-flight Service_4	117374 non-null	uint8
8	In-flight Service_5	117374 non-null	uint8
9	Gate Location_1	117374 non-null	uint8
10	Gate Location_3	117374 non-null	uint8
11	Gate Location_5	117374 non-null	uint8
12	Ease of Online Booking_0	117374 non-null	uint8
13	Ease of Online Booking_1	117374 non-null	uint8
14	Ease of Online Booking_2	117374 non-null	uint8

15	Ease of Online Booking_3	117374	non-null	uint8
16	Ease of Online Booking_4	117374	non-null	uint8
17	Ease of Online Booking_5	117374	non-null	uint8
18	Online Boarding_1	117374	non-null	uint8
19	Online Boarding_2	117374	non-null	uint8
20	Online Boarding_3	117374	non-null	uint8
21	Online Boarding_4	117374	non-null	uint8
22	Online Boarding_5	117374	non-null	uint8
23	Class_Business	117374	non-null	uint8
24	Class_Economy	117374	non-null	uint8
25	Class_Economy Plus	117374	non-null	uint8
26	Leg Room Service_1	117374	non-null	uint8
27	Leg Room Service_2	117374	non-null	uint8
28	Leg Room Service_3	117374	non-null	uint8
29	Leg Room Service_4	117374	non-null	uint8
30	Leg Room Service_5	117374	non-null	uint8
31	Cleanliness_1	117374	non-null	uint8
32	Cleanliness_2	117374	non-null	uint8
33	Cleanliness_4	117374	non-null	uint8
34	Cleanliness_5	117374	non-null	uint8
35	Type of Travel_Business	117374	non-null	uint8
36	Type of Travel_Personal	117374	non-null	uint8
37	In-flight Entertainment_1	117374	non-null	uint8
38	In-flight Entertainment_2	117374	non-null	uint8
39	In-flight Entertainment_3	117374	non-null	uint8
40	In-flight Entertainment_4	117374	non-null	uint8
41	In-flight Entertainment_5	117374	non-null	uint8
42	On-board Service_1	117374	non-null	uint8
43	On-board Service_2	117374	non-null	uint8
44	On-board Service_3	117374	non-null	uint8
45	On-board Service_4	117374	non-null	uint8
46	On-board Service_5	117374	non-null	uint8
47	Departure and Arrival Time Convenience_4	117374	non-null	uint8
48	Customer Type_First-time	117374	non-null	uint8
49	Customer Type_Returning	117374	non-null	uint8
50	Baggage Handling_1	117374	non-null	uint8
51	Baggage Handling_2	117374	non-null	uint8
52	Baggage Handling_3	117374	non-null	uint8
53	Baggage Handling_4	117374	non-null	uint8
54	Baggage Handling_5	117374	non-null	uint8
55	Seat Comfort_1	117374	non-null	uint8
56	Seat Comfort_2	117374	non-null	uint8
57	Seat Comfort_3	117374	non-null	uint8
58	Seat Comfort_4	117374	non-null	uint8
59	Seat Comfort_5	117374	non-null	uint8
60	Food and Drink_1	117374	non-null	uint8
61	Food and Drink_4	117374	non-null	uint8
62	Food and Drink_5	117374	non-null	uint8

```

63 Check-in Service_1          117374 non-null  uint8
64 Check-in Service_2          117374 non-null  uint8
65 Check-in Service_5          117374 non-null  uint8
66 In-flight Wifi Service_0    117374 non-null  uint8
67 In-flight Wifi Service_1    117374 non-null  uint8
68 In-flight Wifi Service_2    117374 non-null  uint8
69 In-flight Wifi Service_3    117374 non-null  uint8
70 In-flight Wifi Service_4    117374 non-null  uint8
71 In-flight Wifi Service_5    117374 non-null  uint8
dtypes: float64(4), uint8(68)
memory usage: 11.2 MB

```

```

[ ]: # SMOTE_X_train = pd.DataFrame(X_SMOTE)
     # SMOTE_X_train.columns = features.columns

```

```

[261]: SMOTE_y_train = y_SMOTE.copy()
       SMOTE_y_train.value_counts()

```

```

[261]: Neutral or Dissatisfied    58687
       Satisfied                  58687
       Name: Satisfaction, dtype: int64

```

Applying undersampling

```

[262]: # from imblearn.under_sampling import RandomUnderSampler
       # define dataset
       X_under = X_train
       y_under = y_train
       # summarize class distribution
       print('Before UnderSampling', (Counter(y_under)))
       # define undersample strategy
       undersample = RandomUnderSampler(sampling_strategy='majority')
       # fit and apply the transform
       X_under, y_under = undersample.fit_resample(X_under, y_under)
       # summarize class distribution
       print('After UnderSampling', (Counter(y_under)))

```

```

Before UnderSampling Counter({'Neutral or Dissatisfied': 58687, 'Satisfied':
44902})

```

```

After UnderSampling Counter({'Neutral or Dissatisfied': 44902, 'Satisfied':
44902})

```

```

[263]: undersampling_X_train = X_under.copy()
       undersampling_X_train.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89804 entries, 0 to 89803
Data columns (total 72 columns):

```

#	Column	Non-Null Count	Dtype
----	-----	-----	-----
0	Age	89804 non-null	float64
1	Flight Distance	89804 non-null	float64
2	Departure Delay	89804 non-null	float64
3	Arrival Delay	89804 non-null	float64
4	In-flight Service_1	89804 non-null	uint8
5	In-flight Service_2	89804 non-null	uint8
6	In-flight Service_3	89804 non-null	uint8
7	In-flight Service_4	89804 non-null	uint8
8	In-flight Service_5	89804 non-null	uint8
9	Gate Location_1	89804 non-null	uint8
10	Gate Location_3	89804 non-null	uint8
11	Gate Location_5	89804 non-null	uint8
12	Ease of Online Booking_0	89804 non-null	uint8
13	Ease of Online Booking_1	89804 non-null	uint8
14	Ease of Online Booking_2	89804 non-null	uint8
15	Ease of Online Booking_3	89804 non-null	uint8
16	Ease of Online Booking_4	89804 non-null	uint8
17	Ease of Online Booking_5	89804 non-null	uint8
18	Online Boarding_1	89804 non-null	uint8
19	Online Boarding_2	89804 non-null	uint8
20	Online Boarding_3	89804 non-null	uint8
21	Online Boarding_4	89804 non-null	uint8
22	Online Boarding_5	89804 non-null	uint8
23	Class_Business	89804 non-null	uint8
24	Class_Economy	89804 non-null	uint8
25	Class_Economy Plus	89804 non-null	uint8
26	Leg Room Service_1	89804 non-null	uint8
27	Leg Room Service_2	89804 non-null	uint8
28	Leg Room Service_3	89804 non-null	uint8
29	Leg Room Service_4	89804 non-null	uint8
30	Leg Room Service_5	89804 non-null	uint8
31	Cleanliness_1	89804 non-null	uint8
32	Cleanliness_2	89804 non-null	uint8
33	Cleanliness_4	89804 non-null	uint8
34	Cleanliness_5	89804 non-null	uint8
35	Type of Travel_Business	89804 non-null	uint8
36	Type of Travel_Personal	89804 non-null	uint8
37	In-flight Entertainment_1	89804 non-null	uint8
38	In-flight Entertainment_2	89804 non-null	uint8
39	In-flight Entertainment_3	89804 non-null	uint8
40	In-flight Entertainment_4	89804 non-null	uint8
41	In-flight Entertainment_5	89804 non-null	uint8
42	On-board Service_1	89804 non-null	uint8
43	On-board Service_2	89804 non-null	uint8
44	On-board Service_3	89804 non-null	uint8
45	On-board Service_4	89804 non-null	uint8


```

46 On-board Service_5                89804 non-null uint8
47 Departure and Arrival Time Convenience_4 89804 non-null uint8
48 Customer Type_First-time          89804 non-null uint8
49 Customer Type_Returning            89804 non-null uint8
50 Baggage Handling_1                 89804 non-null uint8
51 Baggage Handling_2                 89804 non-null uint8
52 Baggage Handling_3                 89804 non-null uint8
53 Baggage Handling_4                 89804 non-null uint8
54 Baggage Handling_5                 89804 non-null uint8
55 Seat Comfort_1                     89804 non-null uint8
56 Seat Comfort_2                     89804 non-null uint8
57 Seat Comfort_3                     89804 non-null uint8
58 Seat Comfort_4                     89804 non-null uint8
59 Seat Comfort_5                     89804 non-null uint8
60 Food and Drink_1                   89804 non-null uint8
61 Food and Drink_4                   89804 non-null uint8
62 Food and Drink_5                   89804 non-null uint8
63 Check-in Service_1                 89804 non-null uint8
64 Check-in Service_2                 89804 non-null uint8
65 Check-in Service_5                 89804 non-null uint8
66 In-flight Wifi Service_0           89804 non-null uint8
67 In-flight Wifi Service_1           89804 non-null uint8
68 In-flight Wifi Service_2           89804 non-null uint8
69 In-flight Wifi Service_3           89804 non-null uint8
70 In-flight Wifi Service_4           89804 non-null uint8
71 In-flight Wifi Service_5           89804 non-null uint8
dtypes: float64(4), uint8(68)
memory usage: 8.6 MB

```

```
[ ]: # undersampling_X_train = pd.DataFrame(X_under)
      # undersampling_X_train.columns = features.columns
```

```
[ ]: # undersampling_X_train.info()
```

```
[264]: undersampling_y_train = y_under.copy()
undersampling_y_train.value_counts()
```

```
[264]: Neutral or Dissatisfied    44902
Satisfied                        44902
Name: Satisfaction, dtype: int64
```

Applying oversampling

```
[265]: # from imblearn.over_sampling import RandomOverSampler
      # define dataset
X_over = X_train
# y_over = y_train
y_over = y_train
```

```

# summarize class distribution
print('Before OverSampling', (Counter(y_over)))
# define oversampling strategy
oversample = RandomOverSampler(sampling_strategy='minority')
# fit and apply the transform
X_over, y_over = oversample.fit_resample(X_over, y_over)
# summarize class distribution
print('After OverSampling', (Counter(y_over)))

```

Before OverSampling Counter({'Neutral or Dissatisfied': 58687, 'Satisfied': 44902})

After OverSampling Counter({'Neutral or Dissatisfied': 58687, 'Satisfied': 58687})

```

[266]: oversampling_X_train = X_over.copy()
oversampling_X_train.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117374 entries, 0 to 117373
Data columns (total 72 columns):

```

#	Column	Non-Null Count	Dtype
0	Age	117374 non-null	float64
1	Flight Distance	117374 non-null	float64
2	Departure Delay	117374 non-null	float64
3	Arrival Delay	117374 non-null	float64
4	In-flight Service_1	117374 non-null	uint8
5	In-flight Service_2	117374 non-null	uint8
6	In-flight Service_3	117374 non-null	uint8
7	In-flight Service_4	117374 non-null	uint8
8	In-flight Service_5	117374 non-null	uint8
9	Gate Location_1	117374 non-null	uint8
10	Gate Location_3	117374 non-null	uint8
11	Gate Location_5	117374 non-null	uint8
12	Ease of Online Booking_0	117374 non-null	uint8
13	Ease of Online Booking_1	117374 non-null	uint8
14	Ease of Online Booking_2	117374 non-null	uint8
15	Ease of Online Booking_3	117374 non-null	uint8
16	Ease of Online Booking_4	117374 non-null	uint8
17	Ease of Online Booking_5	117374 non-null	uint8
18	Online Boarding_1	117374 non-null	uint8
19	Online Boarding_2	117374 non-null	uint8
20	Online Boarding_3	117374 non-null	uint8
21	Online Boarding_4	117374 non-null	uint8
22	Online Boarding_5	117374 non-null	uint8
23	Class_Business	117374 non-null	uint8
24	Class_Economy	117374 non-null	uint8
25	Class_Economy Plus	117374 non-null	uint8

26	Leg Room Service_1	117374	non-null	uint8
27	Leg Room Service_2	117374	non-null	uint8
28	Leg Room Service_3	117374	non-null	uint8
29	Leg Room Service_4	117374	non-null	uint8
30	Leg Room Service_5	117374	non-null	uint8
31	Cleanliness_1	117374	non-null	uint8
32	Cleanliness_2	117374	non-null	uint8
33	Cleanliness_4	117374	non-null	uint8
34	Cleanliness_5	117374	non-null	uint8
35	Type of Travel_Business	117374	non-null	uint8
36	Type of Travel_Personal	117374	non-null	uint8
37	In-flight Entertainment_1	117374	non-null	uint8
38	In-flight Entertainment_2	117374	non-null	uint8
39	In-flight Entertainment_3	117374	non-null	uint8
40	In-flight Entertainment_4	117374	non-null	uint8
41	In-flight Entertainment_5	117374	non-null	uint8
42	On-board Service_1	117374	non-null	uint8
43	On-board Service_2	117374	non-null	uint8
44	On-board Service_3	117374	non-null	uint8
45	On-board Service_4	117374	non-null	uint8
46	On-board Service_5	117374	non-null	uint8
47	Departure and Arrival Time Convenience_4	117374	non-null	uint8
48	Customer Type_First-time	117374	non-null	uint8
49	Customer Type_Returning	117374	non-null	uint8
50	Baggage Handling_1	117374	non-null	uint8
51	Baggage Handling_2	117374	non-null	uint8
52	Baggage Handling_3	117374	non-null	uint8
53	Baggage Handling_4	117374	non-null	uint8
54	Baggage Handling_5	117374	non-null	uint8
55	Seat Comfort_1	117374	non-null	uint8
56	Seat Comfort_2	117374	non-null	uint8
57	Seat Comfort_3	117374	non-null	uint8
58	Seat Comfort_4	117374	non-null	uint8
59	Seat Comfort_5	117374	non-null	uint8
60	Food and Drink_1	117374	non-null	uint8
61	Food and Drink_4	117374	non-null	uint8
62	Food and Drink_5	117374	non-null	uint8
63	Check-in Service_1	117374	non-null	uint8
64	Check-in Service_2	117374	non-null	uint8
65	Check-in Service_5	117374	non-null	uint8
66	In-flight Wifi Service_0	117374	non-null	uint8
67	In-flight Wifi Service_1	117374	non-null	uint8
68	In-flight Wifi Service_2	117374	non-null	uint8
69	In-flight Wifi Service_3	117374	non-null	uint8
70	In-flight Wifi Service_4	117374	non-null	uint8
71	In-flight Wifi Service_5	117374	non-null	uint8

dtypes: float64(4), uint8(68)

memory usage: 11.2 MB

```
[ ]: # oversampling_X_train = pd.DataFrame(X_over)
# oversampling_X_train.columns = features.columns
```

```
[ ]: # oversampling_X_train.info()
```

```
[267]: oversampling_y_train = y_over.copy()
oversampling_y_train.value_counts()
```

```
[267]: Neutral or Dissatisfied    58687
Satisfied                    58687
Name: Satisfaction, dtype: int64
```

6 Building Model

Apply Random Forest, k-Nearest Neighbours, and Gradient Boosting (Extreme Gradient Boosting (XGBoost))

Random Forest: SMOTE

```
[268]: RF = RandomForestClassifier(max_features= 14, max_depth=7) # set parameter_
↳randomly
```

```
[269]: RF.fit(SMOTE_X_train, SMOTE_y_train)
RF.score(SMOTE_X_train, SMOTE_y_train)
```

```
[269]: 0.9352241552643686
```

Random Forest: undersampling

```
[270]: RF.fit(undersampling_X_train, undersampling_y_train)
RF.score(undersampling_X_train, undersampling_y_train)
```

```
[270]: 0.935136519531424
```

Random Forest: oversampling

```
[271]: RF.fit(oversampling_X_train, oversampling_y_train)
RF.score(oversampling_X_train, oversampling_y_train)
```

```
[271]: 0.9337672738425886
```

Comparing the scores

```
[272]: score_RF = [["SMOTE", 0.935], ["undersampling", 0.935], ["oversampling", 0.933]]

# define header name
cols_names = ["Method - RF", "Score"]
```

```
# display table
print(tabulate(score_RF, headers=cols_names, tablefmt="fancy_grid"))
```

Method - RF	Score
SMOTE	0.935
undersampling	0.935
oversampling	0.933

Observations: Got all the same score.

Checking the important features using Random Forest: SMOTE

```
[ ]: # RF_SMOTE_X_train = SMOTE_X_train.copy()
# RF_SMOTE_y_train = SMOTE_y_train.copy()

# def f_importances(coef, names, top=-1):
#     imp = coef
#     imp, names = zip(*sorted(list(zip(imp, names))))

#     if top == -1:
#         top = len(names)

#     plt.barh(range(top), imp[::-1][0:top], align='center', color = 'LightBlue')
#     plt.yticks(range(top), names[::-1][0:top])
#     plt.title('feature importances for Random Forest: SMOTE')
#     plt.show()

# features_names = RF_SMOTE_X_train.columns

# rf = RandomForestClassifier(n_estimators=14 , max_depth=7 ,
#     min_samples_split=25 , max_features=4, random_state=0)
# rf.fit(RF_SMOTE_X_train , RF_SMOTE_y_train)
# f_importances(abs(rf.feature_importances_), features_names, top=7)
```

```
[273]: # fit the model using the training data
# RF = RandomForestClassifier(max_features= 14, max_depth=7)
RF = RandomForestClassifier(max_features= best_k[-1][1], max_depth=7)
RF.fit(SMOTE_X_train , SMOTE_y_train)

# use model to make predictions on test data
y_pred_RF = (RF.predict(X_test))
y_pred_RF
```

```
[273]: array(['Satisfied', 'Neutral or Dissatisfied', 'Satisfied', ...,  
        'Satisfied', 'Neutral or Dissatisfied', 'Satisfied'], dtype=object)
```

```
[274]: # Printing actuals and predictions  
# print('Actuals:\t\t', list(encoded_y_test))  
print('Actuals:\t\t', list(y_test))  
print('Predictions with RF model:\t', list(y_pred_RF))
```

```
Actuals:          ['Satisfied', 'Neutral or Dissatisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied',  
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or  
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or  
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied',  
Neutral or  
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied',  
'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied',  
'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',  
'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied',  
'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied',
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied',
'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied']
```

```
[275]: # confusion matrix of RF
# cnf_matrix_RF = metrics.confusion_matrix(encoded_y_test, y_pred_RF)
cnf_matrix_RF = metrics.confusion_matrix(y_test, y_pred_RF)
cnf_matrix_RF
```

```
[275]: array([[13460, 1078],
       [ 944, 10416]])
```

```
[276]: # classification report - RF
# print(classification_report(encoded_y_test, y_pred_RF))
print(classification_report(y_test, y_pred_RF))
```

	precision	recall	f1-score	support
Neutral or Dissatisfied	0.93	0.93	0.93	14538
Satisfied	0.91	0.92	0.91	11360
accuracy			0.92	25898
macro avg	0.92	0.92	0.92	25898
weighted avg	0.92	0.92	0.92	25898

KNN: SMOTE

```
[277]: KNN = KNeighborsClassifier(n_neighbors=5) # 5 is default
```



```
[278]: KNN.fit(SMOTE_X_train, SMOTE_y_train)
KNN.score(SMOTE_X_train, SMOTE_y_train)
```

```
[278]: 0.965920902414504
```

KNN: undersampling

```
[279]: KNN.fit(undersampling_X_train, undersampling_y_train)
KNN.score(undersampling_X_train, undersampling_y_train)
```

```
[279]: 0.9586878090062804
```

KNN: oversampling

```
[280]: KNN.fit(oversampling_X_train, oversampling_y_train)
KNN.score(oversampling_X_train, oversampling_y_train)
```

```
[280]: 0.9620273655153612
```

Comparing the scores

```
[281]: score_KNN = [{"SMOTE", 0.965}, {"undersampling", 0.958}, {"oversampling", 0.
↪962}]

# define header name
cols_names = ["Method - KNN", "Score"]

# display table
print(tabulate(score_KNN, headers=cols_names, tablefmt="fancy_grid"))
```

Method - KNN	Score
SMOTE	0.965
undersampling	0.958
oversampling	0.962

Observations: KNN: There is no significant difference.

```
[282]: # fit the model using the training data
KNN = KNeighborsClassifier(n_neighbors=5)
KNN.fit(SMOTE_X_train , SMOTE_y_train)

# use model to make predictions on test data
y_pred_KNN = KNN.predict(X_test)
y_pred_KNN
```

```
[282]: array(['Neutral or Dissatisfied', 'Neutral or Dissatisfied',
        'Neutral or Dissatisfied', ..., 'Satisfied',
        'Neutral or Dissatisfied', 'Satisfied'], dtype=object)
```

```
[283]: # Printing actuals and predictions
# print('Actuals:\t\t', list(encoded_y_test))
print('Actuals:\t\t', list(y_test))
print('Predictions with RF model:\t', list(y_pred_KNN))
```

```
Actuals:          ['Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Satisfied',
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]


```

Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied']

```

```

[284]: # confusion matrix of KNN
# cnf_matrix_RF = metrics.confusion_matrix(encoded_y_test, y_pred_RF)
cnf_matrix_KNN = metrics.confusion_matrix(y_test, y_pred_KNN)
cnf_matrix_KNN

```

```

[284]: array([[13454, 1084],
              [ 8383, 2977]])

```

```

[285]: # classification report - RF
# print(classification_report(encoded_y_test, y_pred_RF))
print(classification_report(y_test, y_pred_KNN))

```

	precision	recall	f1-score	support
Neutral or Dissatisfied	0.62	0.93	0.74	14538
Satisfied	0.73	0.26	0.39	11360

accuracy			0.63	25898
macro avg	0.67	0.59	0.56	25898
weighted avg	0.67	0.63	0.58	25898

XGBoost: SMOTE

```
[287]: XGB = GradientBoostingClassifier(max_depth=3) # 3 is default
```

```
[288]: XGB.fit(SMOTE_X_train, SMOTE_y_train)
XGB.score(SMOTE_X_train, SMOTE_y_train)
```

```
[288]: 0.9428152742515378
```

XGBoost: undersampling

```
[289]: XGB.fit(undersampling_X_train, undersampling_y_train)
XGB.score(undersampling_X_train, undersampling_y_train)
```

```
[289]: 0.9436105295977908
```

XGBoost: oversampling

```
[290]: XGB.fit(oversampling_X_train, oversampling_y_train)
XGB.score(oversampling_X_train, oversampling_y_train)
```

```
[290]: 0.9436587319167788
```

Comparing the scores

```
[291]: score_XGBoost = [{"SMOTE", 0.942}, {"undersampling", 0.943}, {"oversampling", 0.
↪943}]

# define header name
cols_names = ["Method - XGBoost", "Score"]

# display table
print(tabulate(score_XGBoost, headers=cols_names, tablefmt="fancy_grid"))
```

Method - XGBoost	Score
SMOTE	0.942
undersampling	0.943
oversampling	0.943

Observations: XGBoost: there is no significant difference.

```
[321]: # convert y class to nemarical class, 0 and 1. 0 is 'Neutral or Dissatisfied'
      ↪and 1 is 'Satisfied'.
      y0y = y != 'Neutral or Dissatisfied'
```

```
[322]: y0y
```

```
[322]: ID
      1      False
      2       True
      3       True
      4       True
      5       True
      ...
    129876  False
    129877  False
    129878  False
    129879   True
    129880  False
      Name: Satisfaction, Length: 129487, dtype: bool
```

```
[323]: type(y0y)
```

```
[323]: pandas.core.series.Series
```

```
[324]: y0y.values
```

```
[324]: array([False,  True,  True, ..., False,  True, False])
```

```
[325]: y0y = y0y.replace({True:1, False:0})
      display(y0y)
```

```
ID
1      0
2      1
3      1
4      1
5      1
...
129876  0
129877  0
129878  0
129879  1
129880  0
      Name: Satisfaction, Length: 129487, dtype: int64
```

```
[326]: y0y = SMOTE_y_train != 'Neutral or Dissatisfied'
y0y = y0y.replace({True:1, False:0})
display(y0y)
```

```
0      0
1      1
2      1
3      1
4      1
..
117369  1
117370  1
117371  1
117372  1
117373  1
Name: Satisfaction, Length: 117374, dtype: int64
```

```
[327]: XGB_SMOTE_y_train = y0y
```

```
[328]: y0y_test = y_test != 'Neutral or Dissatisfied'
y0y_test = y0y_test.replace({True:1, False:0})
display(y0y_test)
```

```
ID
43209    1
125239    0
28497     0
124204    1
39419     1
..
104897    1
122462    1
34291     1
105846    0
123640    1
Name: Satisfaction, Length: 25898, dtype: int64
```

```
[329]: XGB_y_test = y0y_test
display(XGB_y_test)
```

```
ID
43209    1
125239    0
28497     0
124204    1
39419     1
..
104897    1
```

```

122462    1
34291     1
105846    0
123640    1
Name: Satisfaction, Length: 25898, dtype: int64

```

```

[330]: # fit the model using the training data
XGB = XGBClassifier(max_depth = 6)
XGB.fit(SMOTE_X_train, XGB_SMOTE_y_train)

# use model to make predictions on test data
y_pred_XGB = XGB.predict(X_test)
y_pred_XGB

```

```

[330]: array([1, 0, 0, ..., 1, 0, 1])

```

```

[331]: # Printing actuals and predictions
# print('Actuals:\t\t', list(encoded_y_test))
print('Actuals:\t\t', list(y_test))
print('Predictions with RF model:\t', list(y_pred_XGB))

```

```

Actuals:          ['Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied',
'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied',
'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied',
'Satisfied', 'Neutral or Dissatisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Satisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Neutral or
Dissatisfied', 'Satisfied', 'Neutral or Dissatisfied', 'Satisfied', 'Satisfied',

```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1,
 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0,
 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0,
 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
 1, 0, 0, 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, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1,
 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0,
 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1,
 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1,
 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1,
 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1,
 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1,
 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0,
 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0,
 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,
 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1,
 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,

0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1,
 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0,
 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1,
 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0,
 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,
 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1,
 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0,
 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0,
 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,

0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1,
 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,
 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0,
 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1,
 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0,
 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0,
 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,
 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1,
 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1,
1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0,
1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1,
0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1,
1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0,
0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,

0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1,
 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1,
 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0,
 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0,
 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0,
 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,

1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1,
1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1,
0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,

1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1,
0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1,
0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0,

0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0,
 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1,
 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0,
 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1,
 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0,
 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,

1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1,
 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0,
 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0,
 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0,
 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0,
 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1,
 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1,
 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1,
 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,

0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0,
 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0,
 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1,
 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1,
 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
 1, 0, 1, 0, 1, 1,
 1, 0, 1, 0, 1, 1,
 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1,
 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,
 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1,
 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1,
 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1,

0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0,
 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,

0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0,
 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0,
 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0,
 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0,
 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1,
 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1,
 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,

[illegible]

0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0,
1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0,
0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0,
0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,

0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0,
0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1,
0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1,
1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1,
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0,

1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1,
0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1,
1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1,
1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1,
0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,
0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1,
0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1,
1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,

0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1,
 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1,
 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1,
 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0,
 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1,
 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,


```

0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1,
0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1]

```

```

[332]: # confusion matrix of KNN
# cnf_matrix_RF = metrics.confusion_matrix(encoded_y_test, y_pred_RF)
cnf_matrix_XGB = metrics.confusion_matrix(XGB_y_test, y_pred_XGB)
cnf_matrix_XGB

```

```
[332]: array([[14359, 179],
              [ 1183, 10177]])
```

```
[333]: # classification report - RF
# print(classification_report(encoded_y_test, y_pred_RF))
print(classification_report(XGB_y_test, y_pred_XGB))
```

	precision	recall	f1-score	support
0	0.92	0.99	0.95	14538
1	0.98	0.90	0.94	11360
accuracy			0.95	25898
macro avg	0.95	0.94	0.95	25898
weighted avg	0.95	0.95	0.95	25898

Based on score, this project selected SMOTE dataset and find the optimal parameters for given models

```
[314]: # Model activation and result plot function

def get_model_metrics(model, SMOTE_X_train, X_test, SMOTE_y_train, y_test):
    '''
    Model activation function, takes in model as a parameter and returns
    metrics as specified.

    Inputs:
        model, SMOTE_X_train, selected_X_test, SMOTE_y_train, encoded_y_test
    Output:
        Model output metrics, confusion matrix, ROC AUC curve
    '''

    # Mark of current time when model began running
    t0 = time.time()

    # Fit the model on the training data and run predictions on test data
    model.fit(SMOTE_X_train, SMOTE_y_train)
    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)[:,-1]
    ConfusionMatrix = confusion_matrix(y_test, y_pred, labels = model.classes_)
    # Obtain training accuracy as a comparative metric using Sklearn's metrics
    train_score = model.score(SMOTE_X_train, SMOTE_y_train)
    # Obtain testing accuracy as a comparative metric using Sklearn's metrics
    accuracy = accuracy_score(y_test, y_pred)
```

```

# Obtain precision from predictions using Sklearn's metrics package
precision = precision_score(y_test, y_pred)
# Obtain recall from predictions using Sklearn's metrics package
recall = recall_score(y_test, y_pred)
# Obtain f1 from predictions using Sklearn's metrics package
f1 = f1_score(y_test, y_pred)
# Obtain ROC score from predictions using Sklearn's metrics package
roc = roc_auc_score(y_test, y_pred_proba)
# Obtain the time taken used to run the model, by subtracting the start
↪time from the current time
time_taken = time.time() - t0
# Obtain the resources consumed in running the model
memory_used = int(getrusage(RUSAGE_SELF).ru_maxrss / 1024)

# Outputting the metrics of the model performance
print("Accuracy on Training = {}".format(train_score))
print("Accuracy on Test = {} • Precision = {}".format(accuracy, precision))
print("Recall = {} • ROC Area under Curve = {}".format(recall, roc))
print("F1 = {} • ROC Area under Curve = {}".format(f1, roc))
print("Time taken = {} seconds • Memory consumed = {} Bytes".
↪format(time_taken, memory_used))

# Plotting the confusion matrix of the model's predictive capabilities
dis = ConfusionMatrixDisplay(confusion_matrix = ConfusionMatrix,
↪display_labels= model.classes_)
# dis = ConfusionMatrixDisplay(confusion_matrix = ConfusionMatrix,
↪display_labels= ['Dissatisfied', 'Satisfied'])
# Plotting the ROC AUC curve of the model
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
# create ROC curve
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
dis.plot(values_format='')
plt.show()

return model, train_score, accuracy, precision, recall, f1, roc,
↪time_taken, memory_used

```

Parameters for Random Forest

```
[307]: RandomForestClassifier().get_params()
```

```
[307]: {'bootstrap': True,
       'ccp_alpha': 0.0,
       'class_weight': None,
```

```

'criterion': 'gini',
'max_depth': None,
'max_features': 'sqrt',
'max_leaf_nodes': None,
'max_samples': None,
'min_impurity_decrease': 0.0,
'min_samples_leaf': 1,
'min_samples_split': 2,
'min_weight_fraction_leaf': 0.0,
'n_estimators': 100,
'n_jobs': None,
'oob_score': False,
'random_state': None,
'verbose': 0,
'warm_start': False}

```

Performing GridSearchCV to obtain optimal model parameters

```
[ ]: from sklearn.metrics import make_scorer, accuracy_score
```

```

[308]: %%time
clf = RandomForestClassifier()

params = { 'max_depth': [5, 10, 15, 20],
           'max_leaf_nodes': [10, 20, 30, 40],
           'min_samples_split': [1, 2, 3, 4]}

# Type of scoring used to compare parameter combinations
acc_scorer = make_scorer(accuracy_score)

rscv = GridSearchCV(estimator = clf,
                    param_grid = params,
                    scoring = acc_scorer,
                    cv = None,
                    n_jobs = -1,
                    verbose = 1)
rscv.fit(SMOTE_X_train, SMOTE_y_train)
rscv.predict(X_test)

# Parameter object to be passed through to function activation
params = rscv.best_params_

print("Best parameters:", params)

```

Fitting 5 folds for each of 64 candidates, totalling 320 fits

Best parameters: {'max_depth': 10, 'max_leaf_nodes': 40, 'min_samples_split': 2}

CPU times: user 22.6 s, sys: 2.66 s, total: 25.3 s

Wall time: 21min 32s

```
[334]: SMOTE_y_train
```

```
[334]: 0      Neutral or Dissatisfied
      1      Satisfied
      2      Satisfied
      3      Satisfied
      4      Satisfied
      ...
      117369      Satisfied
      117370      Satisfied
      117371      Satisfied
      117372      Satisfied
      117373      Satisfied
      Name: Satisfaction, Length: 117374, dtype: category
      Categories (2, object): ['Neutral or Dissatisfied', 'Satisfied']
```

```
[337]: SMOTE_y_train = y0y.copy()
      display(SMOTE_y_train)
```

```
0      0
1      1
2      1
3      1
4      1
      ..
117369      1
117370      1
117371      1
117372      1
117373      1
      Name: Satisfaction, Length: 117374, dtype: int64
```

```
[338]: y_test = y0y_test.copy()
```

```
[339]: y0y_test
```

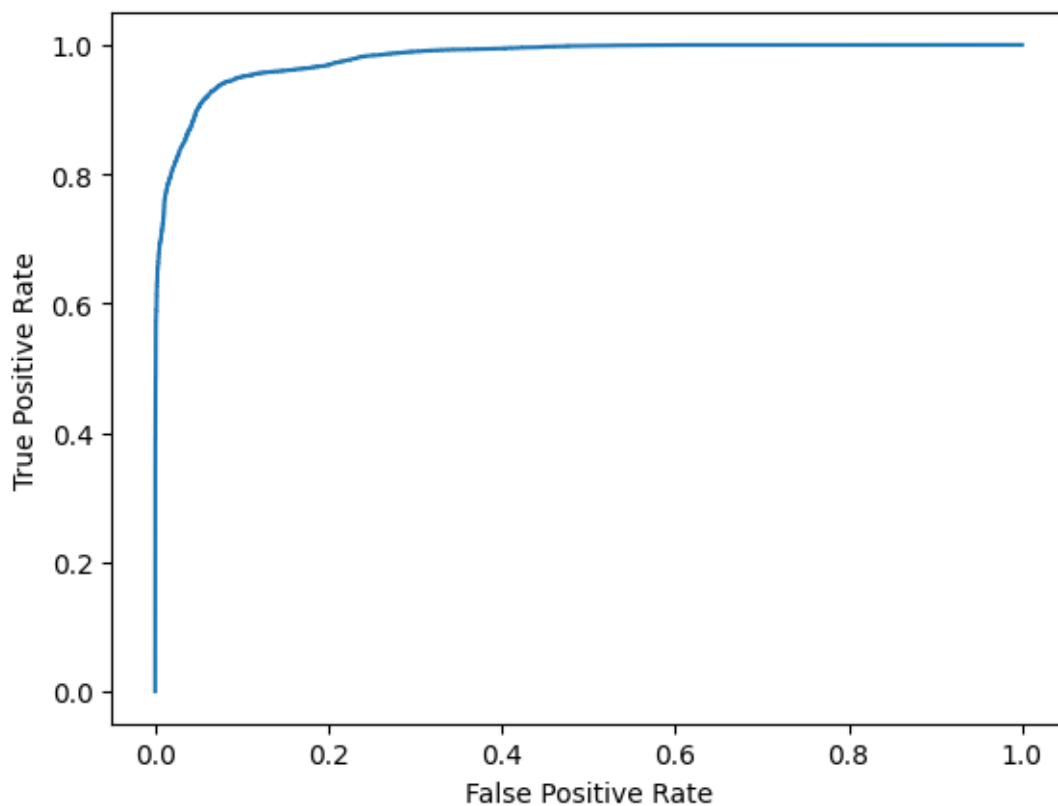
```
[339]: ID
      43209      1
      125239      0
      28497      0
      124204      1
      39419      1
      ..
      104897      1
      122462      1
      34291      1
      105846      0
```

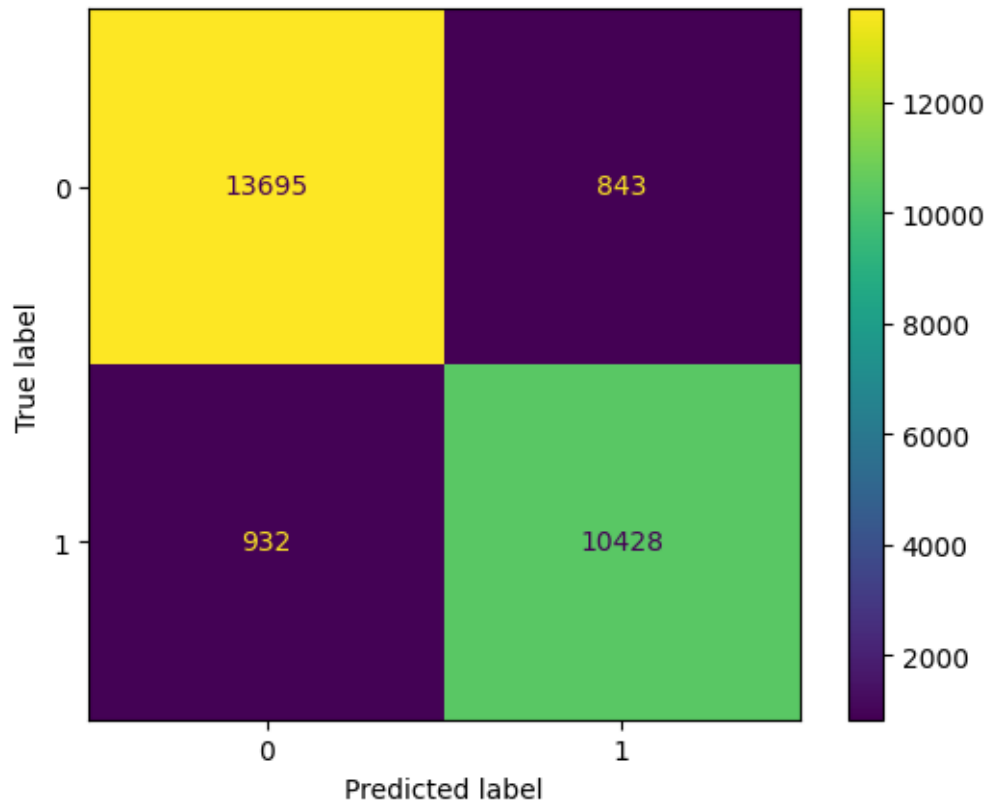
123640 1
Name: Satisfaction, Length: 25898, dtype: int64

Running model pipeline and obtaining performance metrics - Random Forest

```
[340]: model_rf = RandomForestClassifier(**params)
model_rf, train_rf, accuracy_rf, precision_rf, recall_rf, f1_rf, roc_rf, tt_rf, \
    ↳ mu_rf = get_model_metrics(model_rf, SMOTE_X_train, X_test, SMOTE_y_train, \
    ↳ y_test)
```

Accuracy on Training = 0.9334946410619047
Accuracy on Test = 0.9314618889489535 • Precision = 0.9252062816076657
Recall = 0.9179577464788733 • ROC Area under Curve = 0.9798255700456695
F1 = 0.9215677610357474 • ROC Area under Curve = 0.9798255700456695
Time taken = 11.879323720932007 seconds • Memory consumed = 2569 Bytes





```
[335]: SMOTE_y_train
```

```
[335]: 0      Neutral or Dissatisfied
      1      Satisfied
      2      Satisfied
      3      Satisfied
      4      Satisfied
      ...
      117369      Satisfied
      117370      Satisfied
      117371      Satisfied
      117372      Satisfied
      117373      Satisfied
      Name: Satisfaction, Length: 117374, dtype: category
      Categories (2, object): ['Neutral or Dissatisfied', 'Satisfied']
```

```
[320]: y_test
```

```
[320]: ID
      43209      Satisfied
      125239      Neutral or Dissatisfied
```

```

28497    Neutral or Dissatisfied
124204                Satisfied
39419                Satisfied
...
104897                Satisfied
122462                Satisfied
34291                Satisfied
105846    Neutral or Dissatisfied
123640                Satisfied
Name: Satisfaction, Length: 25898, dtype: category
Categories (2, object): ['Neutral or Dissatisfied', 'Satisfied']

```

Parameters for KNN

```
[341]: KNeighborsClassifier().get_params()
```

```

[341]: {'algorithm': 'auto',
       'leaf_size': 30,
       'metric': 'minkowski',
       'metric_params': None,
       'n_jobs': None,
       'n_neighbors': 5,
       'p': 2,
       'weights': 'uniform'}

```

Performing GridSearchCV to obtain optimal model parameters

```

[343]: %%time
clf = KNeighborsClassifier()

params = { 'n_neighbors': [5, 10, 15, 25]}

# Type of scoring used to compare parameter combinations
acc_scorer = make_scorer(accuracy_score)

rscv = GridSearchCV(estimator = clf,
                    param_grid = params,
                    scoring = acc_scorer,
                    cv = None,
                    n_jobs = -1,
                    verbose = 1)
rscv.fit(SMOTE_X_train, SMOTE_y_train)
rscv.predict(X_test)

# Parameter object to be passed through to function activation
params = rscv.best_params_

print("Best parameters:", params)

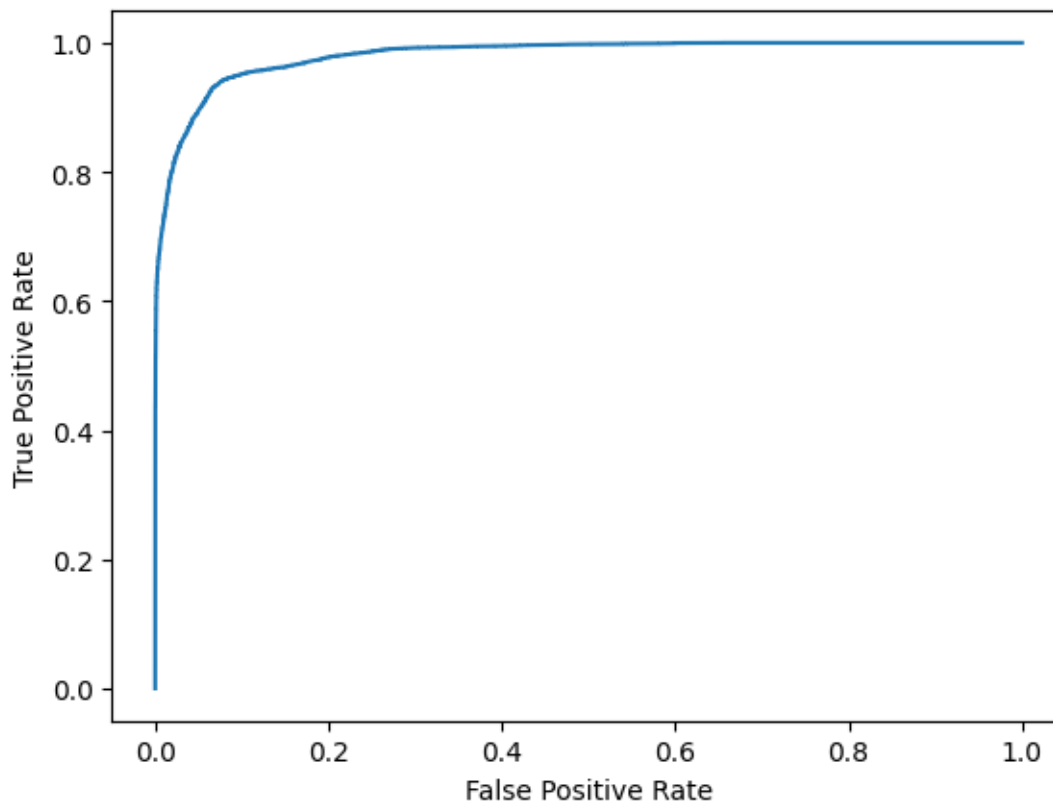
```

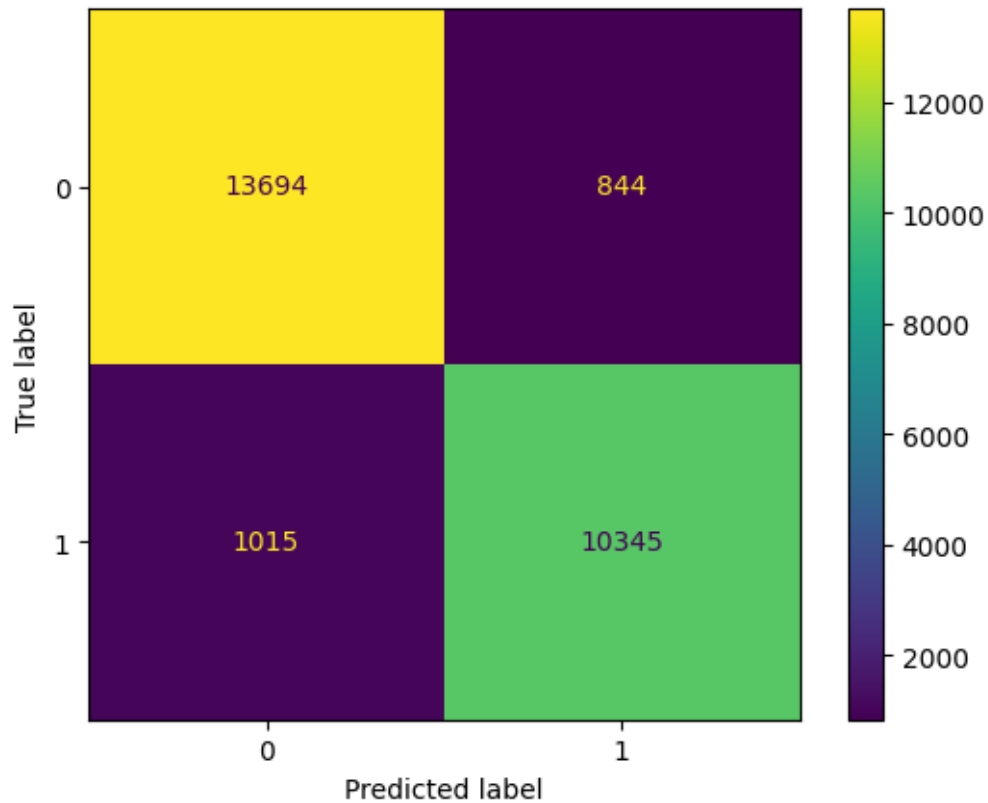

Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best parameters: {'n_neighbors': 5}
CPU times: user 45.6 s, sys: 922 ms, total: 46.5 s
Wall time: 6min 50s

Running model pipeline and obtaining performance metrics - KNN

```
[344]: model_KNN = KNeighborsClassifier(**params)
model_KNN, train_KNN, accuracy_KNN, precision_KNN, recall_KNN, f1_KNN, roc_KNN, \
    tt_KNN, mu_KNN = get_model_metrics(model_rf, SMOTE_X_train, X_test, \
    SMOTE_y_train, y_test)
```

Accuracy on Training = 0.9307257143830832
Accuracy on Test = 0.9282183952428759 • Precision = 0.9245687729019573
Recall = 0.9106514084507042 • ROC Area under Curve = 0.9805256234753409
F1 = 0.9175573196150606 • ROC Area under Curve = 0.9805256234753409
Time taken = 12.043384075164795 seconds • Memory consumed = 2569 Bytes





Parameters for XGBoost

```
[345]: XGBClassifier().get_params()
```

```
[345]: {'objective': 'binary:logistic',  
       'use_label_encoder': None,  
       'base_score': None,  
       'booster': None,  
       'callbacks': None,  
       'colsample_bylevel': None,  
       'colsample_bynode': None,  
       'colsample_bytree': None,  
       'early_stopping_rounds': None,  
       'enable_categorical': False,  
       'eval_metric': None,  
       'feature_types': None,  
       'gamma': None,  
       'gpu_id': None,  
       'grow_policy': None,  
       'importance_type': None,  
       'interaction_constraints': None,
```

```

'learning_rate': None,
'max_bin': None,
'max_cat_threshold': None,
'max_cat_to_onehot': None,
'max_delta_step': None,
'max_depth': None,
'max_leaves': None,
'min_child_weight': None,
'missing': nan,
'monotone_constraints': None,
'n_estimators': 100,
'n_jobs': None,
'num_parallel_tree': None,
'predictor': None,
'random_state': None,
'reg_alpha': None,
'reg_lambda': None,
'sampling_method': None,
'scale_pos_weight': None,
'subsample': None,
'tree_method': None,
'validate_parameters': None,
'verbosity': None}

```

Performing GridSearchCV to obtain optimal model parameters

```

[346]: %%time
clf = XGBClassifier()

params = { 'max_depth': [3, 5, 6],
           'learning_rate': [0.01, 0.1],
           'n_estimators': [100, 500]}

# Type of scoring used to compare parameter combinations
acc_scorer = make_scorer(accuracy_score)

rscv = GridSearchCV(estimator = clf,
                    param_grid = params,
                    scoring = acc_scorer,
                    cv = None,
                    n_jobs = -1,
                    verbose = 1)
rscv.fit(SMOTE_X_train, SMOTE_y_train)
rscv.predict(X_test)

# Parameter object to be passed through to function activation

```

```
params = rscv.best_params_  
  
print("Best parameters:", params)
```

Fitting 5 folds for each of 12 candidates, totalling 60 fits

Best parameters: {'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 500}

CPU times: user 7min 25s, sys: 12.4 s, total: 7min 38s

Wall time: 1h 25min 24s

Running model pipeline and obtaining performance metrics - XGBoost

```
[347]: model_XGB = XGBClassifier(**params)  
model_XGB, train_XGB, accuracy_XGB, precision_XGB, recall_XGB, f1_XGB, roc_XGB,   
    ↪ tt_XGB, mu_XGB = get_model_metrics(model_rf, SMOTE_X_train, X_test,   
    ↪ SMOTE_y_train, y_test)
```

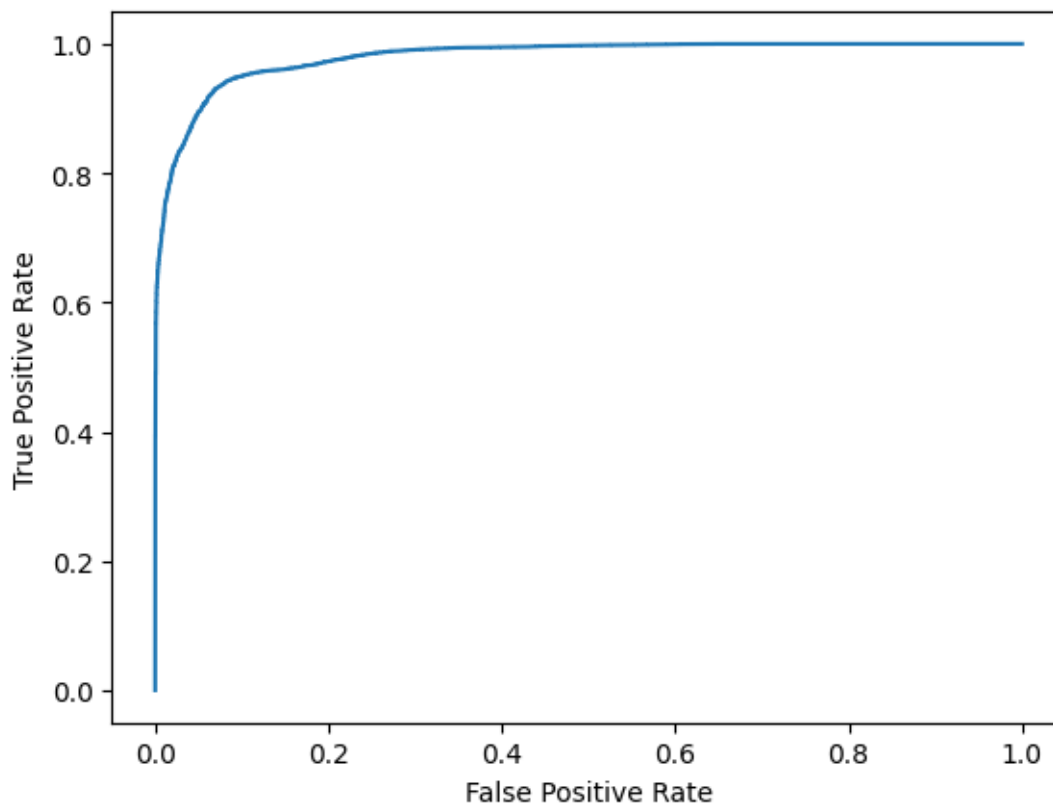
Accuracy on Training = 0.930768313255065

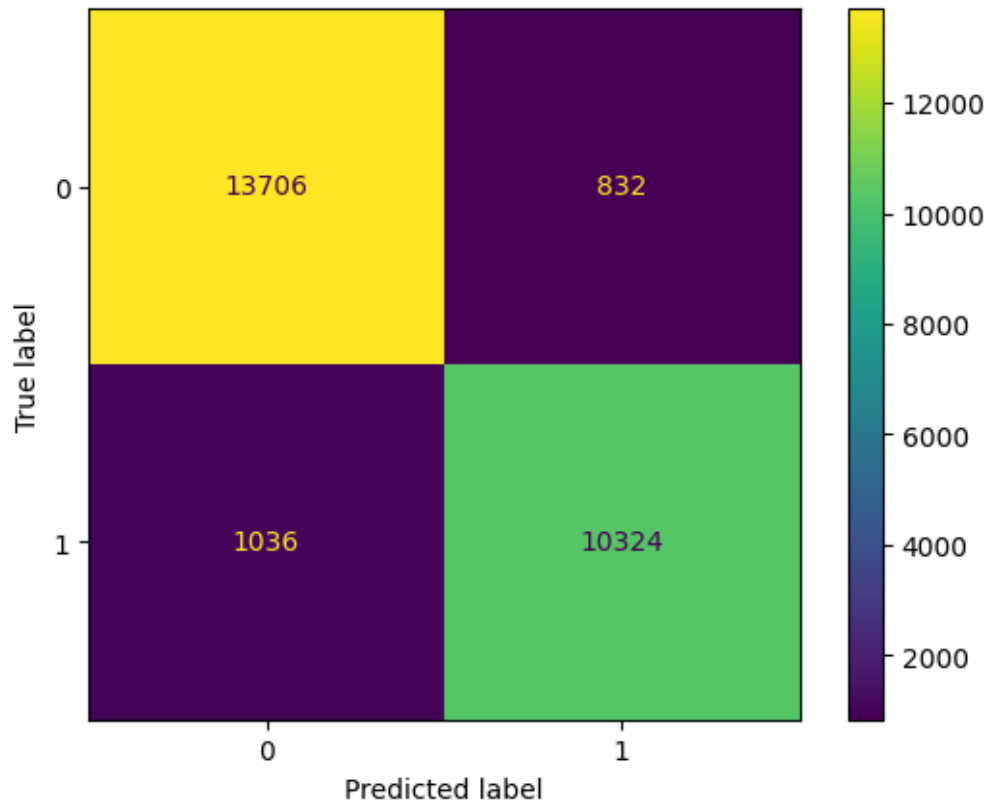
Accuracy on Test = 0.9278708780600818 • Precision = 0.9254212979562567

Recall = 0.9088028169014084 • ROC Area under Curve = 0.9797234154687375

F1 = 0.9170367738497069 • ROC Area under Curve = 0.9797234154687375

Time taken = 13.496250629425049 seconds • Memory consumed = 2569 Bytes





Comparing output performance of the model pipelines

```
[348]: # Collecting model data
training_scores = [train_rf, train_KNN, train_XGB]
accuracy = [accuracy_rf, accuracy_KNN, accuracy_XGB]
precision = [precision_rf, precision_KNN, precision_XGB]
recall = [recall_rf, recall_KNN, recall_XGB]
f1_scores = [f1_rf, f1_KNN, f1_XGB]
roc_scores = [roc_rf, roc_KNN, roc_XGB]
time_scores = [tt_rf, tt_KNN, tt_XGB]
memory_scores = [mu_rf, mu_KNN, mu_XGB]

model_data = {'Model': ['Random Forest', 'KNN', 'XGBoost'],
              'Accuracy on Training' : training_scores,
              'Accuracy on Test' : accuracy,
              'Precision' : precision,
              'Recall' : recall,
              'F1' : f1_scores,
              'ROC AUC Score' : roc_scores,
              'Time Elapsed (seconds)' : time_scores,
              'Memory Consumed (bytes)': memory_scores}
```

```
model_data = pd.DataFrame(model_data)
model_data
```

```
[348]:
```

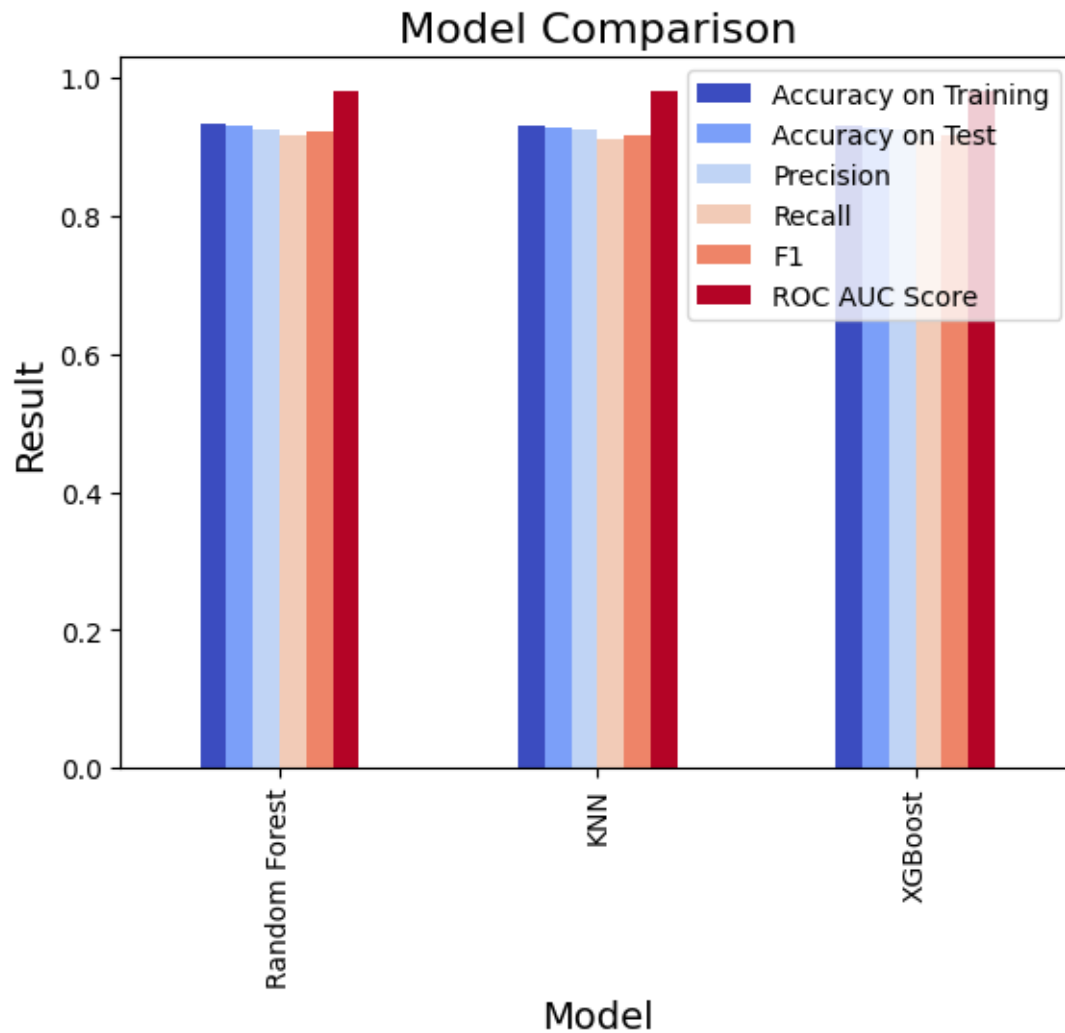
	Model	Accuracy on Training	Accuracy on Test	Precision	Recall	\
0	Random Forest	0.933495	0.931462	0.925206	0.917958	
1	KNN	0.930726	0.928218	0.924569	0.910651	
2	XGBoost	0.930768	0.927871	0.925421	0.908803	

	F1	ROC AUC Score	Time Elapsed (seconds)	Memory Consumed (bytes)
0	0.921568	0.979826	11.879324	2569
1	0.917557	0.980526	12.043384	2569
2	0.917037	0.979723	13.496251	2569

```
[349]: # Plotting each model's performance scores vs time elapsed
# plt.rcParams["figure.figsize"] = (15,10)

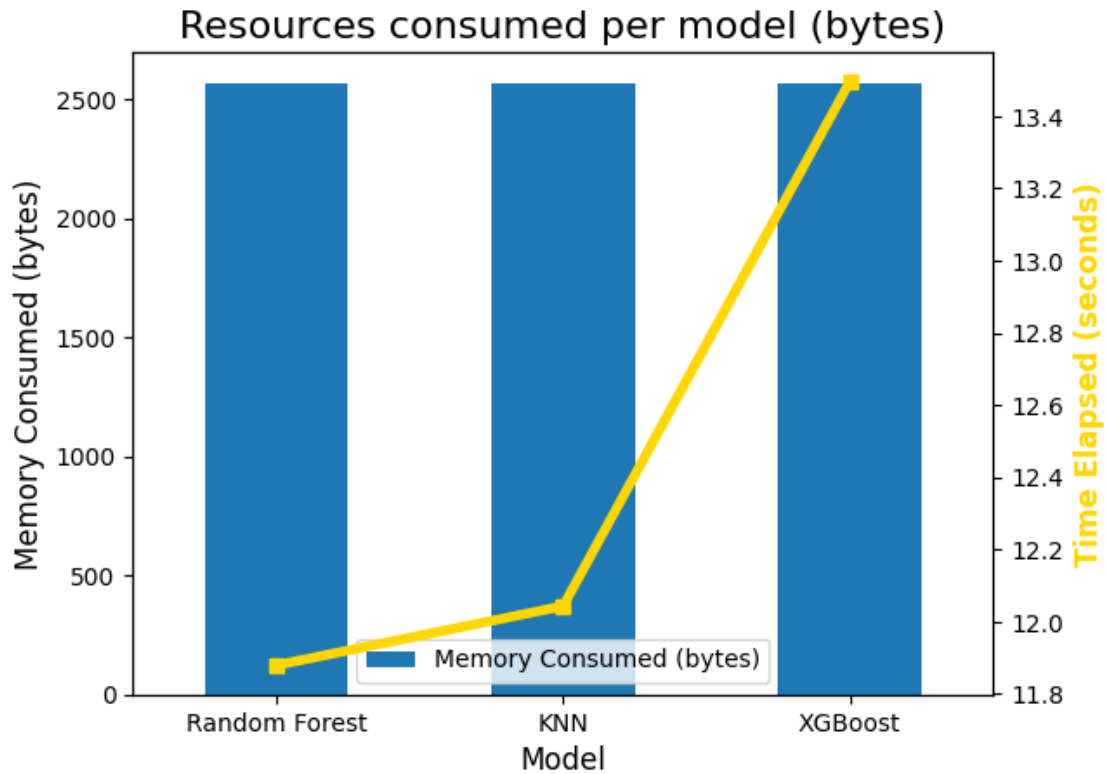
ax1 = model_data.plot.bar(x = 'Model', y = ["Accuracy on Training", "Accuracy_
↪on Test", "Precision", "Recall",
                                         "F1", "ROC AUC Score"],
                           cmap = 'coolwarm')
ax1.legend()

ax1.set_title("Model Comparison", fontsize = 16)
ax1.set_xlabel('Model', fontsize = 14)
ax1.set_ylabel('Result', fontsize = 14, color = 'Black');
```



```
[350]: # Plotting each model's memory consumption
ax1 = model_data.plot.bar(x = 'Model', y = 'Memory Consumed (bytes)')

ax1.set_title("Resources consumed per model (bytes)", fontsize = 16)
ax2 = model_data['Time Elapsed (seconds)'].plot(secondary_y = True, color = 'Gold',
        linewidth = 4, marker = 's')
ax1.set_xlabel('Model', fontsize = 12)
ax2.set_ylabel('Time Elapsed (seconds)', fontsize = 12, color = 'Gold',
        fontweight = 'bold')
ax1.set_ylabel('Memory Consumed (bytes)', fontsize = 12, color = 'Black');
```



Feature importance of finalist pipeline - Random Forest

Check top 15 features

```
[352]: RF_SMOTE_X_train = SMOTE_X_train.copy()
RF_SMOTE_y_train = SMOTE_y_train.copy()

def f_importances(coef, names, top=-1):
    imp = coef
    imp, names = zip(*sorted(list(zip(imp, names))))

    if top == -1:
        top = len(names)

    plt.barh(range(top), imp[::-1][0:top], align='center', color = 'LightBlue')
    plt.yticks(range(top), names[::-1][0:top])
    plt.title('feature importances for Random Forest: SMOTE')
    plt.show()

features_names = RF_SMOTE_X_train.columns

rf = RandomForestClassifier(max_depth=10 , max_leaf_nodes=40,
    ↪min_samples_split=2, random_state=0)
```



```
rf.fit(RF_SMOTE_X_train , RF_SMOTE_y_train)
f_importances(abs(rf.feature_importances_), features_names, top=15)
```



Observations: The top feature is Class_Business and Online Boarding and Type of Travel are follow.