learn-ml

October 29, 2019

[59]:
```
'''
cURL is a computer software project providing a library and command-line tool␣
 ↪for transferring data using various protocols.
Bash is a command line interpreter that typically runs in a text window where␣
 ↪user can interpret commands to carry out various actions.

curl is a Bash command. You can execute Bash commands in a Jupyter notebook by␣
 ↪prefixing them with an exclamation mark.
This command downloads a CSV file from Azure blob storage and saves it using the␣
 ↪name flightdata.csv.
'''

!curl https://topcs.blob.core.windows.net/public/FlightData.csv -o flightdata.csv
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1552k  100 1552k    0     0   933k      0  0:00:01  0:00:01 --:--:--  933k
```

1 STEP 1: FETCH THE DATA

[60]:
```
'''
pandas is an open source, BSD-licensed library providing high-performance,␣
 ↪easy-to-use data
structures and data analysis tools for the Python programming language.


A DataFrame is a two-dimensional labeled data structure. The columns in a␣
 ↪DataFrame can be of different types, just like columns in a spreadsheet or␣
 ↪database table.
It is the most commonly used object in Pandas. In this exercise, you will␣
 ↪examine the DataFrame  and the data inside it  more closely.

The DataFrame that you created contains on-time arrival information for a major␣
 ↪U.S. airline. It has more than
11,000 rows and 26 columns. (The output says "5 rows" because DataFrame's head␣
 ↪function only returns the first five rows.)
```

```
'''Each row represents one flight and contains information such as the origin, the destination, the scheduled departure time,
and whether the flight arrived on time or late. We'll look at the data more closely a bit later in this module.
'''

import pandas as pd

df = pd.read_csv('flightdata.csv')
df.head()
```

[60]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	UNIQUE_CARRIER	TAIL_NUM	FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	...	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	ARR_DEL15	CANCELLED	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	DISTANCE	Unnamed: 25
0	2016	1	1	1	5	DL	N836DN	1399	10397	ATL	...	2143	2102.0	-41.0	0.0	0.0	0.0	338.0	295.0	2182.0	NaN
1	2016	1	1	1	5	DL	N964DN	1476	11433	DTW	...	1435	1439.0	4.0	0.0	0.0	0.0	110.0	115.0	528.0	NaN
2	2016	1	1	1	5	DL	N813DN	1597	10397	ATL	...	1215	1142.0	-33.0	0.0	0.0	0.0	335.0	300.0	2182.0	NaN
3	2016	1	1	1	5	DL	N587NW	1768	14747	SEA	...	1335	1345.0	10.0	0.0	0.0	0.0	196.0	205.0	1399.0	NaN
4	2016	1	1	1	5	DL	N836DN	1823	14747	SEA	...	607	615.0	8.0	0.0	0.0	0.0	247.0	259.0	1927.0	NaN

5 rows × 26 columns

[61]:
```
df.shape
```

[61]:
```
(11231, 26)
```

[62]:
```
'''
To know if shape is not a function , what is it ?
'''
help(pd.DataFrame.shape)
```

```
Help on property:

    Return a tuple representing the dimensionality of the DataFrame.

    See Also
    --------
    ndarray.shape

    Examples
    --------
    >>> df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4]})
    >>> df.shape
    (2, 2)

    >>> df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4],
    ...                    'col3': [5, 6]})
    >>> df.shape
    (2, 3)
```

[63]:
```
'''
To see what the head function does?
'''
help(pd.DataFrame.head)
```

```
Help on function head in module pandas.core.generic:

head(self, n=5)
    Return the first `n` rows.

    This function returns the first `n` rows for the object based
    on position. It is useful for quickly testing if your object
    has the right type of data in it.

    Parameters
    ----------
    n : int, default 5
        Number of rows to select.

    Returns
    -------
    obj_head : type of caller
        The first `n` rows of the caller object.

    See Also
    --------
    pandas.DataFrame.tail: Returns the last `n` rows.
```

```
Examples
--------
>>> df = pd.DataFrame({'animal':['alligator', 'bee', 'falcon', 'lion',
...                    'monkey', 'parrot', 'shark', 'whale', 'zebra']})
>>> df
      animal
0  alligator
1        bee
2     falcon
3       lion
4     monkey
5     parrot
6      shark
7      whale
8      zebra

Viewing the first 5 lines

>>> df.head()
      animal
0  alligator
1        bee
2     falcon
3       lion
4     monkey

Viewing the first `n` lines (three in this case)

>>> df.head(3)
      animal
0  alligator
1        bee
2     falcon
```

2 LET US STUDY EACH COLUMN IN DETAIL USING THE PRESENTATION

3 STEP 2: CLEAN AND PREPARE DATA

[64]:
```
'''

To eliminate missing values, either by deleting the rows or columns containing␣
 ↪them or replacing them with meaningful values.
```

```
Eliminate extraneous columns

Selecting the "feature" columns that are relevant to the outcome you are trying␣
 ↪to predict while filtering out
columns that do not affect the outcome, could bias

Binning or quantization of the data

Convert columns containing categorical data to discrete columns containing␣
 ↪indicator values

Confirm that the output is "True," which indicates that there is at least one␣
 ↪missing value somewhere in the dataset.


'''

df.isnull().values.any() #checking if there is any null values
```

[64]:
```
True
```

[65]:
```
#df.isnull()  # gives a whole table which gives true or false for which each␣
 ↪column of the dataframe
#df.isnull().values #a 2D list of true or false
#df.isnull().values.any() #check if any null values are present
```

[66]:
```
df.isnull().values.any()
```

[66]:
```
True
```

[67]:
```
df.isnull().sum() #The next step is to find out where the missing values are.
```

[67]:
```
YEAR                   0
QUARTER                0
MONTH                  0
DAY_OF_MONTH           0
DAY_OF_WEEK            0
UNIQUE_CARRIER         0
TAIL_NUM               0
FL_NUM                 0
ORIGIN_AIRPORT_ID      0
ORIGIN                 0
DEST_AIRPORT_ID        0
DEST                   0
CRS_DEP_TIME           0
DEP_TIME             107
DEP_DELAY            107
DEP_DEL15            107
CRS_ARR_TIME           0
ARR_TIME             115
```

```
ARR_DELAY                188
ARR_DEL15                188
CANCELLED                  0
DIVERTED                   0
CRS_ELAPSED_TIME           0
ACTUAL_ELAPSED_TIME      188
DISTANCE                   0
Unnamed: 25            11231
dtype: int64
```

[68]:
```
'''
Curiously, the 26th column ("Unnamed: 25") contains 11,231 missing values, which␣
 ↪equals the number of rows in the dataset.
This column was mistakenly created because the CSV file that you imported
contains a comma at the end of each line. To eliminate that column, add the␣
 ↪following code to the notebook and execute it:'''
#axis : {0 or index, 1 or columns}, default 0
#Whether to drop labels from the index (0 or index) or columns (1 or columns).

df = df.drop('Unnamed: 25', axis=1)
df.isnull().sum()
```

[68]:
```
YEAR                     0
QUARTER                  0
MONTH                    0
DAY_OF_MONTH             0
DAY_OF_WEEK              0
UNIQUE_CARRIER           0
TAIL_NUM                 0
FL_NUM                   0
ORIGIN_AIRPORT_ID        0
ORIGIN                   0
DEST_AIRPORT_ID          0
DEST                     0
CRS_DEP_TIME             0
DEP_TIME               107
DEP_DELAY              107
DEP_DEL15              107
CRS_ARR_TIME             0
ARR_TIME               115
ARR_DELAY              188
ARR_DEL15              188
CANCELLED                0
DIVERTED                 0
CRS_ELAPSED_TIME         0
ACTUAL_ELAPSED_TIME    188
DISTANCE                 0
dtype: int64
```

[69]:
```
help(pd.DataFrame.drop)
```

```
Help on function drop in module pandas.core.frame:

drop(self, labels=None, axis=0, index=None, columns=None, level=None,
inplace=False, errors='raise')
    Drop specified labels from rows or columns.

    Remove rows or columns by specifying label names and corresponding
    axis, or by specifying directly index or column names. When using a
    multi-index, labels on different levels can be removed by specifying
    the level.

    Parameters
    ----------
    labels : single label or list-like
        Index or column labels to drop.
    axis : {0 or 'index', 1 or 'columns'}, default 0
        Whether to drop labels from the index (0 or 'index') or
        columns (1 or 'columns').
    index, columns : single label or list-like
        Alternative to specifying axis (``labels, axis=1``
        is equivalent to ``columns=labels``).

        .. versionadded:: 0.21.0
    level : int or level name, optional
        For MultiIndex, level from which the labels will be removed.
    inplace : bool, default False
        If True, do operation inplace and return None.
    errors : {'ignore', 'raise'}, default 'raise'
        If 'ignore', suppress error and only existing labels are
        dropped.

    Returns
    -------
    dropped : pandas.DataFrame

    See Also
    --------
    DataFrame.loc : Label-location based indexer for selection by label.
    DataFrame.dropna : Return DataFrame with labels on given axis omitted
        where (all or any) data are missing
    DataFrame.drop_duplicates : Return DataFrame with duplicate rows
        removed, optionally only considering certain columns
    Series.drop : Return Series with specified index labels removed.

    Raises
```

```
------
KeyError
    If none of the labels are found in the selected axis

Examples
--------
>>> df = pd.DataFrame(np.arange(12).reshape(3,4),
...                   columns=['A', 'B', 'C', 'D'])
>>> df
   A  B   C   D
0  0  1   2   3
1  4  5   6   7
2  8  9  10  11

Drop columns

>>> df.drop(['B', 'C'], axis=1)
   A   D
0  0   3
1  4   7
2  8  11

>>> df.drop(columns=['B', 'C'])
   A   D
0  0   3
1  4   7
2  8  11

Drop a row by index

>>> df.drop([0, 1])
   A  B   C   D
2  8  9  10  11

Drop columns and/or rows of MultiIndex DataFrame

>>> midx = pd.MultiIndex(levels=[['lama', 'cow', 'falcon'],
...                              ['speed', 'weight', 'length']],
...                      labels=[[0, 0, 0, 1, 1, 1, 2, 2, 2],
...                              [0, 1, 2, 0, 1, 2, 0, 1, 2]])
>>> df = pd.DataFrame(index=midx, columns=['big', 'small'],
...                   data=[[45, 30], [200, 100], [1.5, 1], [30, 20],
...                         [250, 150], [1.5, 0.8], [320, 250],
...                         [1, 0.8], [0.3,0.2]])
>>> df
                big     small
lama    speed   45.0    30.0
        weight  200.0   100.0
```

```
        length  1.5     1.0
cow     speed   30.0    20.0
        weight  250.0   150.0
        length  1.5     0.8
falcon  speed   320.0   250.0
        weight  1.0     0.8
        length  0.3     0.2

>>> df.drop(index='cow', columns='small')
                big
lama    speed   45.0
        weight  200.0
        length  1.5
falcon  speed   320.0
        weight  1.0
        length  0.3

>>> df.drop(index='length', level=1)
                big     small
lama    speed   45.0    30.0
        weight  200.0   100.0
cow     speed   30.0    20.0
        weight  250.0   150.0
falcon  speed   320.0   250.0
        weight  1.0     0.8
```

[70]:
```
'''
The DataFrame still contains a lot of missing values, but some of them aren't␣
 ↪useful because the columns
containing them are not relevant to the model that you are building. The goal of␣
 ↪that model is to predict whether
a flight you are considering booking is likely to arrive on time. If you know␣
 ↪that the flight is likely to be late,
you might choose to book another flight.

The next step, therefore, is to filter the dataset to eliminate columns that␣
 ↪aren't relevant to a predictive model.
For example, the aircraft's tail number probably has little bearing on whether a␣
 ↪flight will arrive on time, and at the
time you book a ticket, you have no way of knowing whether a flight will be␣
 ↪cancelled, diverted, or delayed.
By contrast, the scheduled departure time could have a lot to do with on-time␣
 ↪arrivals.

'''
```

```
df = df[["MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "ORIGIN", "DEST",␣
 ↪"CRS_DEP_TIME", "ARR_DEL15"]]
df.isnull().sum()
```

[70]:
```
MONTH             0
DAY_OF_MONTH      0
DAY_OF_WEEK       0
ORIGIN            0
DEST              0
CRS_DEP_TIME      0
ARR_DEL15       188
dtype: int64
```

[71]:
```
'''
The only column that now contains missing values is the ARR_DEL15 column,
which uses 0s to identify flights that arrived on time and 1s for flights that␣
 ↪didn't.
The reason these rows are missing ARR_DEL15 values is that they all correspond␣
 ↪to flights that were canceled or diverted.
'''

df[df.isnull().values.any(axis=1)].head()
```

[71]:

	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_DEP_TIME	ARR_DEL15
177	1	9	6	MSP	SEA	701	NaN
179	1	10	7	MSP	DTW	1348	NaN
184	1	10	7	MSP	DTW	625	NaN
210	1	10	7	DTW	MSP	1200	NaN
478	1	22	5	SEA	JFK	2305	NaN

[72]:
```
'''

You could call dropna on the DataFrame to remove these rows. But since a flight␣
 ↪that is canceled
or diverted to another airport could be considered "late," let's use the fillna␣
 ↪method to replace the missing values with 1s.

Use the following code to replace missing values in the ARR_DEL15 column with 1s␣
 ↪and display rows 177 through 184:
'''

df = df.fillna({'ARR_DEL15': 1})
df.iloc[177:185]
```

[72]:

	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_DEP_TIME	ARR_DEL15
177	1	9	6	MSP	SEA	701	1.0
178	1	9	6	DTW	JFK	1527	0.0
179	1	10	7	MSP	DTW	1348	1.0
180	1	10	7	DTW	MSP	1540	0.0

```
181      1            10            7    JFK   ATL          1325        0.0
182      1            10            7    JFK   ATL           610        0.0
183      1            10            7    JFK   SEA          1615        0.0
184      1            10            7    MSP   DTW           625        1.0
```

[73]:
```
'''
Intuitively, it makes sense, because it probably doesn't matter much whether a␣
 ↪flight leaves at 10:30 a.m. or 10:40 a.m.
It matters a great deal whether it leaves at 10:30 a.m. or 5:30 p.m.

Binning/Quantization :

Equal width (or distance) binning : The simplest binning approach is to␣
 ↪partition the range of the variable into k equal-width intervals.
The interval width is simply the range [A, B] of the variable divided by k,
w = (B-A) / k
A=0000 , B=2359 so after dividing by 100 all values will be between 00 and 23.
'''

#:iterrows(): Iterate over the rows of a DataFrame as (index, Series) pairs.␣
 ↪This converts the rows to
#Series objects, which can change the dtypes and has some performance␣
 ↪implications.

import math
for index, row in df.iterrows():
    df.loc[index, 'CRS_DEP_TIME'] = math.floor(row['CRS_DEP_TIME'] / 100) #index␣
 ↪means row index, row contains all columns with values
    df.head()
```

[74]:
```
'''
To convert a categorical variable into a dummy or indicator DataFrame, for␣
 ↪example a column in a
DataFrame (a Series) which has k distinct values, can derive a DataFrame␣
 ↪containing k columns of 1s and
0s using get_dummies():

'''

dd= pd.DataFrame({'city':␣
 ↪('delhi','chennai','bengaluru','delhi','delhi','bengaluru'), 'guest_no':␣
 ↪range(6)})
```

[75]:
```
dd
```

[75]:
```
      city  guest_no
0    delhi         0
1  chennai         1
```

```
2  bengaluru         2
3      delhi         3
4      delhi         4
5  bengaluru         5
```

[76]:
```
dd=pd.get_dummies(dd,columns=['city'])
```

[77]:
```
dd
```

[77]:
```
   guest_no  city_bengaluru  city_chennai  city_delhi
0         0               0             0           1
1         1               0             1           0
2         2               1             0           0
3         3               0             0           1
4         4               0             0           1
5         5               1             0           0
```

[78]:
```
df = pd.get_dummies(df, columns=['ORIGIN', 'DEST'])
df.head()
```

[78]:
```
   MONTH  DAY_OF_MONTH  DAY_OF_WEEK  CRS_DEP_TIME  ARR_DEL15  ORIGIN_ATL  \
0      1             1            5            19        0.0           1
1      1             1            5            13        0.0           0
2      1             1            5             9        0.0           1
3      1             1            5             8        0.0           0
4      1             1            5            23        0.0           0

   ORIGIN_DTW  ORIGIN_JFK  ORIGIN_MSP  ORIGIN_SEA  DEST_ATL  DEST_DTW  \
0           0           0           0           0         0         0
1           1           0           0           0         0         0
2           0           0           0           0         0         0
3           0           0           0           1         0         0
4           0           0           0           1         0         1

   DEST_JFK  DEST_MSP  DEST_SEA
0         0         0         1
1         0         1         0
2         0         0         1
3         0         1         0
4         0         0         0
```

4 STEP 3: Build Machine Learning Model

[79]:
```
'''
https://medium.com/@contactsunny/
 ↪how-to-split-your-dataset-to-train-and-test-datasets-using-scikit-learn-e7cf6eb5e0d

The first statement imports scikit-learn's train_test_split helper function.
```

```
The second line uses the function to split the DataFrame into a training set␣
 ↪containing 80% of the original data,
and a test set containing the remaining 20%. The random_state parameter seeds␣
 ↪the random-number generator used to do the splitting,
while the first and second parameters are DataFrames containing the feature␣
 ↪columns and the label column.

train_test_split returns four DataFrames.
'''

from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(df.drop('ARR_DEL15',␣
 ↪axis=1), df['ARR_DEL15'], test_size=0.2, random_state=42)
```

[80]:
```
'''
Random forest classifier
https://www.datacamp.com/community/tutorials/random-forests-classifier-python
Scikit-learn includes a variety of classes for implementing common machine␣
 ↪learning models. One of them is RandomForestClassifier,
which fits multiple decision trees to the data and uses averaging to boost the␣
 ↪overall accuracy and limit overfitting.

The output shows the parameters used in the classifier, including n_estimators,␣
 ↪which specifies the number of trees in
each decision-tree forest, and max_depth, which specifies the maximum depth of␣
 ↪the decision trees.
The values shown are the defaults, but you can override any of them when␣
 ↪creating the RandomForestClassifier object.
'''

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=13)
model.fit(train_x, train_y)
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-
packages/sklearn/ensemble/forest.py:246: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

[80]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=13, verbose=0, warm_start=False)
```

[107]:
```
'''Now call the predict method to test the model using the values in test_x,
followed by the score method to determine the mean accuracy of the model:
We will go over each line in detail.'''
predicted = model.predict(test_x)
model.score(test_x, test_y)
```

[107]:
```
0.8602581219403649
```

[110]:
```
'''
This piece of code is not to be written or executed.
It is to show the what values has our model classified the flight into , 1 for late and 0 for on time.
'''

import sys
import numpy
numpy.set_printoptions(threshold=sys.maxsize)
model.predict(test_x) # this was just to check the
```

[110]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 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., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 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., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 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., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 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., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 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., 1., 0., 0., 0., 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., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 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.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 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., 0., 1., 1., 0., 0., 0., 0., 0.,
0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 0., 0.,
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., 0., 0.,
0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 1., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,

0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 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., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 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., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 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., 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., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 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., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 0., 0.,
0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 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., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,

```
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 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., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 1., 1., 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., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0.])
```

[72]:
```
help(RandomForestClassifier.score)
```

```
Help on function score in module sklearn.base:

score(self, X, y, sample_weight=None)
    Returns the mean accuracy on the given test data and labels.

    In multi-label classification, this is the subset accuracy
    which is a harsh metric since you require for each sample that
    each label set be correctly predicted.

    Parameters
    ----------
    X : array-like, shape = (n_samples, n_features)
        Test samples.

    y : array-like, shape = (n_samples) or (n_samples, n_outputs)
        True labels for X.

    sample_weight : array-like, shape = [n_samples], optional
        Sample weights.

    Returns
    -------
    score : float
        Mean accuracy of self.predict(X) wrt. y.
```

[119]:
```
'''
prediction probabilities for the test set.
These probabilities are estimates for each of the classes, or answers, the␣
↪model can predict.
For example, [0.88199435, 0.11800565] means that there's an 89% chance that a␣
↪flight will arrive on time (ARR_DEL15 = 0)
and a 12% chance that it won't (ARR_DEL15 = 1). The sum of the two␣
↪probabilities adds up to 100%.
```

```
'''
from sklearn.metrics import roc_auc_score
probabilities = model.predict_proba(test_x)
```

[120]:
```
probabilities #it prints the probability for each flight in the test set
```

[120]:
```
array([[0.8, 0.2],
       [0.9, 0.1],
       [1. , 0. ],
       [0.5, 0.5],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [0.8, 0.2],
       [0.9, 0.1],
       [0.8, 0.2],
       [1. , 0. ],
       [0.6, 0.4],
       [0.6, 0.4],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.9, 0.1],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [0.7, 0.3],
       [0.8, 0.2],
       [0.5, 0.5],
       [0.9, 0.1],
       [0.8, 0.2],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.5, 0.5],
       [0.9, 0.1],
       [0.8, 0.2],
       [0.9, 0.1],
       [0.8, 0.2],
       [0.9, 0.1],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
```

```
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[0.8, 0.2],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.4, 0.6],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
```

```
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[0.8, 0.2],
[0.9, 0.1],
[0.3, 0.7],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.5, 0.5],
```

```
[0.9, 0.1],
[0.4, 0.6],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.4, 0.6],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.8, 0.2],
[0.6, 0.4],
[0.9, 0.1],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[0.8, 0.2],
[1. , 0. ],
[0.5, 0.5],
```

```
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.4, 0.6],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
```

```
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.7, 0.3],
[0.7, 0.3],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[0.3, 0.7],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.4, 0.6],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
```

```
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.7, 0.3],
[0.5, 0.5],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[0.5, 0.5],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.6, 0.4],
[0.7, 0.3],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.6, 0.4],
[0.8, 0.2],
[0.5, 0.5],
[0.7, 0.3],
```

```
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.7, 0.3],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.3, 0.7],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
```

```
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.4, 0.6],
[0.9, 0.1],
[0.1, 0.9],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
```

```
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.5, 0.5],
[0.8, 0.2],
[0.8, 0.2],
[0.3, 0.7],
[0.9, 0.1],
[0.8, 0.2],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
```

```
[0.4, 0.6],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.4, 0.6],
[0.9, 0.1],
[0.6, 0.4],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
```

```
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.5, 0.5],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[0.4, 0.6],
[0.5, 0.5],
[0.9, 0.1],
[0.3, 0.7],
[0.9, 0.1],
[0.6, 0.4],
[1. , 0. ],
[0.7, 0.3],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.5, 0.5],
[0.7, 0.3],
[0.9, 0.1],
```

```
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.3, 0.7],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.6, 0.4],
[1. , 0. ],
[0.8, 0.2],
[0.7, 0.3],
[0.7, 0.3],
[0.5, 0.5],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.4, 0.6],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[0.9, 0.1],
```

```
[1. , 0. ],
[0.1, 0.9],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.3, 0.7],
[0.3, 0.7],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.3, 0.7],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.7, 0.3],
[0.3, 0.7],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
```

```
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.4, 0.6],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.5, 0.5],
[0.8, 0.2],
[1. , 0. ],
[0.4, 0.6],
[0.7, 0.3],
[0.7, 0.3],
[0.8, 0.2],
[0.5, 0.5],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
```

```
[0.6, 0.4],
[1. , 0. ],
[0.8, 0.2],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.4, 0.6],
[1. , 0. ],
[0.7, 0.3],
[0.2, 0.8],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[0.8, 0.2],
[0.7, 0.3],
[0.2, 0.8],
[1. , 0. ],
[0.8, 0.2],
[0.6, 0.4],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.4, 0.6],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
```

```
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.3, 0.7],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.3, 0.7],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[0.6, 0.4],
[1. , 0. ],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
```

```
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.2, 0.8],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.8, 0.2],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.3, 0.7],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
```

```
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.3, 0.7],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.3, 0.7],
[0.2, 0.8],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.4, 0.6],
[0.5, 0.5],
[0.5, 0.5],
[0.6, 0.4],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
```

```
[0.8, 0.2],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.2, 0.8],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.4, 0.6],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
```

```
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.5, 0.5],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
```

```
[0.9, 0.1],
[0.6, 0.4],
[0.2, 0.8],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[0.9, 0.1],
[0.2, 0.8],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
```

```
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.4, 0.6],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[0.6, 0.4],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.5, 0.5],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[0.7, 0.3],
[1. , 0. ],
[0.7, 0.3],
```

```
[0.5, 0.5],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0. , 1. ],
[0.9, 0.1],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.2, 0.8],
[0.7, 0.3],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.2, 0.8],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[0.3, 0.7],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
```

```
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.3, 0.7],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[0.8, 0.2],
[0.4, 0.6],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.4, 0.6],
[0.9, 0.1],
[0.3, 0.7],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
```

```
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.2, 0.8],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.6, 0.4],
[0.5, 0.5],
[0.6, 0.4],
[0.4, 0.6],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
```

```
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.3, 0.7],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.3, 0.7],
[0.9, 0.1],
[0.8, 0.2],
[0.7, 0.3],
[0.6, 0.4],
[0.6, 0.4],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.5, 0.5],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
```

```
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.5, 0.5],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[0.8, 0.2],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[0.5, 0.5],
[1. , 0. ],
[0.9, 0.1],
```

```
[1. , 0. ],
[0.9, 0.1],
[0.6, 0.4],
[0.7, 0.3],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.6, 0.4],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.4, 0.6],
[0.9, 0.1],
```

```
[0.5, 0.5],
[1. , 0. ],
[1. , 0. ],
[0.4, 0.6],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.1, 0.9],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.5, 0.5],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
```

```
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[1. , 0. ],
[0.3, 0.7],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.6, 0.4],
[0.6, 0.4],
[0.7, 0.3],
[0.6, 0.4],
[0.8, 0.2],
[0.6, 0.4],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.3, 0.7],
[0.9, 0.1],
[0.5, 0.5],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.3, 0.7],
[0.9, 0.1],
[0.2, 0.8],
[1. , 0. ],
```

```
[0.5, 0.5],
[0.9, 0.1],
[0.7, 0.3],
[0.7, 0.3],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.3, 0.7],
[0.7, 0.3],
[0.8, 0.2],
[0.8, 0.2],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
```

```
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.8, 0.2],
[0.4, 0.6],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.2, 0.8],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.3, 0.7],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.5, 0.5],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.4, 0.6],
[1. , 0. ],
```

```
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.3, 0.7],
[0.8, 0.2],
[0.4, 0.6],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.6, 0.4],
[0.5, 0.5],
[0.9, 0.1],
[0.3, 0.7],
[0.8, 0.2],
[0.9, 0.1],
```

```
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.6, 0.4],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.5, 0.5],
[0.5, 0.5],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.6, 0.4],
[0.9, 0.1],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
```

```
[0.8, 0.2],
[0.6, 0.4],
[0.5, 0.5],
[0.7, 0.3],
[0.9, 0.1],
[0.8, 0.2],
[0.5, 0.5],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.4, 0.6],
[0.8, 0.2],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.6, 0.4],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
```

```
[0.9, 0.1],
[0.7, 0.3],
[0.2, 0.8],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.3, 0.7],
[0.7, 0.3],
[0.8, 0.2],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.4, 0.6],
[0.5, 0.5],
[0.2, 0.8],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
```

```
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[1. , 0. ],
[0.5, 0.5],
[0.7, 0.3],
[1. , 0. ],
[0.8, 0.2],
[0. , 1. ],
[0.6, 0.4],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[0.4, 0.6],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.5, 0.5],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.4, 0.6],
```

```
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[0.5, 0.5],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
```

```
[0.7, 0.3],
[0.6, 0.4],
[1. , 0. ],
[0.2, 0.8],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.7, 0.3],
[0.4, 0.6],
[0.7, 0.3],
[0.5, 0.5],
[0.8, 0.2],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[0.6, 0.4],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
```

```
[0.8, 0.2],
[0.5, 0.5],
[0.6, 0.4],
[1. , 0. ],
[0.2, 0.8],
[0.9, 0.1],
[0.3, 0.7],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.5, 0.5],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.4, 0.6],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
```

```
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[0.8, 0.2],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.5, 0.5],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[0.2, 0.8],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
```

```
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.3, 0.7],
[0.7, 0.3],
[0.9, 0.1],
[0.2, 0.8],
[0.6, 0.4],
[0.9, 0.1],
[0.6, 0.4],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[0.9, 0.1],
[0.9, 0.1],
[0.5, 0.5],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
```

```
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.5, 0.5],
[0.9, 0.1],
[0.7, 0.3],
```

```
[0.8, 0.2],
[0.8, 0.2],
[0.4, 0.6],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.6, 0.4],
[0.6, 0.4],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
[0.3, 0.7],
[0.9, 0.1],
[0.6, 0.4],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.6, 0.4],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.7, 0.3],
[1. , 0. ],
[0.5, 0.5],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[0.8, 0.2],
```

```
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.7, 0.3],
[1. , 0. ],
[0.8, 0.2],
[0.9, 0.1],
[0.4, 0.6],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.9, 0.1],
[0.9, 0.1],
```

```
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.5, 0.5],
[0.8, 0.2],
[1. , 0. ],
[0.4, 0.6],
[0.8, 0.2],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.9, 0.1],
[0.4, 0.6],
[1. , 0. ],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[0.4, 0.6],
[0.4, 0.6],
[0.7, 0.3],
[0.9, 0.1],
[0.9, 0.1],
[0.8, 0.2],
[0.8, 0.2],
[0.8, 0.2],
[1. , 0. ],
[1. , 0. ],
[0.7, 0.3],
[0.9, 0.1],
[1. , 0. ],
[1. , 0. ],
[0.9, 0.1],
[0.8, 0.2],
[1. , 0. ],
[0.3, 0.7],
[0.9, 0.1],
[0.9, 0.1],
[1. , 0. ],
[0.7, 0.3],
```

```
       [0.9, 0.1],
       [1. , 0. ],
       [0.9, 0.1],
       [0.9, 0.1],
       [1. , 0. ],
       [1. , 0. ],
       [0.8, 0.2],
       [0.4, 0.6],
       [1. , 0. ],
       [0.9, 0.1],
       [0.9, 0.1],
       [0.9, 0.1],
       [0.8, 0.2],
       [0.9, 0.1],
       [1. , 0. ],
       [0.8, 0.2],
       [0.7, 0.3],
       [1. , 0. ],
       [0.5, 0.5],
       [0.8, 0.2],
       [1. , 0. ],
       [0.5, 0.5],
       [0.9, 0.1],
       [1. , 0. ],
       [1. , 0. ],
       [0.5, 0.5],
       [0.9, 0.1],
       [1. , 0. ],
       [0.5, 0.5],
       [0.9, 0.1],
       [0.9, 0.1],
       [1. , 0. ],
       [0.6, 0.4],
       [1. , 0. ],
       [0.8, 0.2],
       [0.7, 0.3],
       [0.9, 0.1],
       [0.8, 0.2],
       [0.9, 0.1],
       [0.9, 0.1],
       [0.9, 0.1],
       [0.7, 0.3],
       [1. , 0. ]])
```

[74]:
```
'''
SKIP
Image on precision and recall,confusion matrix
```

```
https://medium.com/x8-the-ai-community/
 ↪understanding-ml-evaluation-metrics-precision-recall-2b3fb915b666'''


'''The first row in the output represents flights that were on time. The first␣
 ↪column in that row shows how many
flights were correctly predicted to be on time, while the second column reveals␣
 ↪how many flights were predicted as delayed
but weren't. From this, the model appears to be adept at predicting that a␣
 ↪flight will be on time.

Generating a confusion matrix

Generating a confusion matrix

But look at the second row, which represents flights that were delayed. The␣
 ↪first column shows how
many delayed flights were incorrectly predicted to be on time. The second column␣
 ↪shows how many flights were
correctly predicted to be delayed. Clearly, the model isn't nearly as adept at␣
 ↪predicting that a flight will be delayed
as it is at predicting that a flight will arrive on time. What you want in a␣
 ↪confusion matrix is large numbers in the
upper-left and lower-right corners, and small numbers (preferably zeros) in the␣
 ↪upper-right and lower-left corners.'''

from sklearn.metrics import confusion_matrix
confusion_matrix(test_y, predicted)
```

[74]:
```
array([[1809,  127],
       [ 240,   71]])
```

5 Step 5:Visualizing and output of the model

[139]:
```
'''
The first statement is one of several magic commands supported by the Python␣
 ↪kernel that you selected when you created the notebook.
It enables Jupyter to render Matplotlib output in a notebook without making␣
 ↪repeated calls to show. And it
must appear before any references to Matplotlib itself. The final statement␣
 ↪configures Seaborn to enhance the output from Matplotlib.
'''
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.set()
```

[81]:
```
'''
https://www.programiz.com/python-programming/datetime/strptime
The function isoweekday() returns an integer value  corresponding to the day of␣
 ↪the week.

This function takes as input a date and time, an origin airport code, and a␣
 ↪destination airport code,
and returns a value between 0.0 and 1.0 indicating the probability that the␣
 ↪flight will arrive at its destination on time.
It uses the machine-learning model you built in the previous lab to compute the␣
 ↪probability. And to call the model,
it passes a DataFrame containing the input values to predict_proba. The␣
 ↪structure of the DataFrame exactly matches the
structure of the DataFrame we used earlier.

'''

def predict_delay(departure_date_time, origin, destination):
    from datetime import datetime

    try:
        departure_date_time_parsed = datetime.strptime(departure_date_time, '%d/
 ↪%m/%Y %H:%M:%S')
    except ValueError as e:
        return 'Error parsing date/time - {}'.format(e)

    month = departure_date_time_parsed.month
    day = departure_date_time_parsed.day
    day_of_week = departure_date_time_parsed.isoweekday()
    hour = departure_date_time_parsed.hour

    origin = origin.upper()
    destination = destination.upper()

    input = [{'MONTH': month,
              'DAY': day,
              'DAY_OF_WEEK': day_of_week,
              'CRS_DEP_TIME': hour,
              'ORIGIN_ATL': 1 if origin == 'ATL' else 0,
              'ORIGIN_DTW': 1 if origin == 'DTW' else 0,
              'ORIGIN_JFK': 1 if origin == 'JFK' else 0,
              'ORIGIN_MSP': 1 if origin == 'MSP' else 0,
              'ORIGIN_SEA': 1 if origin == 'SEA' else 0,
              'DEST_ATL': 1 if destination == 'ATL' else 0,
```

```
            'DEST_DTW': 1 if destination == 'DTW' else 0,
            'DEST_JFK': 1 if destination == 'JFK' else 0,
            'DEST_MSP': 1 if destination == 'MSP' else 0,
            'DEST_SEA': 1 if destination == 'SEA' else 0 }]

    if(model.predict(pd.DataFrame(input))==0):
        print("The flight will be on time")
    else:
        print("The flight will be delayed")

    print("The probability of the flight being on time is")
    return model.predict_proba(pd.DataFrame(input))[0][0]
```

[141]:
```
predict_delay('2/10/2018 10:00:00', 'ATL', 'SEA')
```

```
The flight will be on time
The probability of the flight being on time is
```

[141]: 1.0

[140]:
```
predict_delay('1/10/2018 21:45:00', 'JFK', 'ATL')
```

```
The flight will be on time
The probability of the flight being on time is
```

[140]: 0.6

[138]:
```
predict_delay('2/10/2018 21:45:00', 'JFK', 'ATL')
```

```
The flight will be on time
The probability of the flight being on time is
```

[138]: 0.8

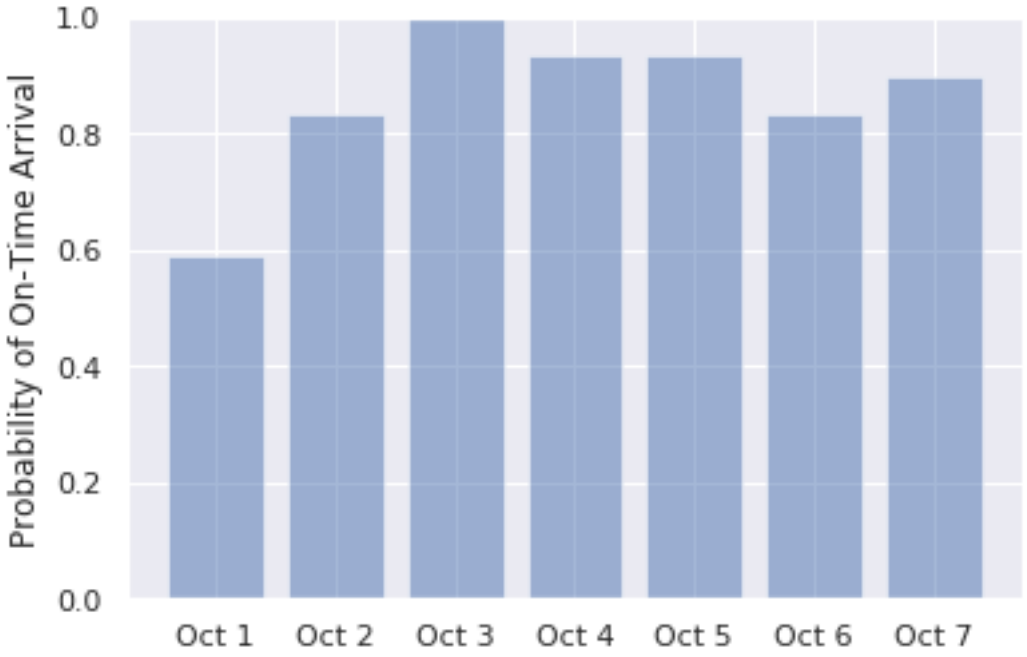
[87]:
```
'''
 plot the probability of on-time arrivals for an evening flight from JFK to ATL␣
 ↪over a range of days:
'''
import numpy as np

labels = ('Oct 1', 'Oct 2', 'Oct 3', 'Oct 4', 'Oct 5', 'Oct 6', 'Oct 7')
values = (predict_delay('1/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('2/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('3/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('4/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('5/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('6/10/2018 21:45:00', 'JFK', 'ATL'),
          predict_delay('7/10/2018 21:45:00', 'JFK', 'ATL'))
```

```
alabels = np.arange(len(labels)) #x co-ordinates of the bar plot

plt.bar(alabels, values, align='center', alpha=0.5)
plt.xticks(alabels, labels)
plt.ylabel('Probability of On-Time Arrival')
plt.ylim((0.0, 1.0))
```

[87]: (0.0, 1.0)



[ ]:
```
labels
```

[ ]:
```
len(labels)
```

[ ]:
```
np.arange(len(labels))
```

[88]:
```
help(plt.bar)
```

```
Help on function bar in module matplotlib.pyplot:

bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)
    Make a bar plot.

    The bars are positioned at *x* with the given *align*\ment. Their
    dimensions are given by *width* and *height*. The vertical baseline
    is *bottom* (default 0).

    Each of *x*, *height*, *width*, and *bottom* may either be a scalar
```

```
applying to all bars, or it may be a sequence of length N providing a
separate value for each bar.

Parameters
----------
x : sequence of scalars
    The x coordinates of the bars. See also *align* for the
    alignment of the bars to the coordinates.

height : scalar or sequence of scalars
    The height(s) of the bars.

width : scalar or array-like, optional
    The width(s) of the bars (default: 0.8).

bottom : scalar or array-like, optional
    The y coordinate(s) of the bars bases (default: 0).

align : {'center', 'edge'}, optional, default: 'center'
    Alignment of the bars to the *x* coordinates:

    - 'center': Center the base on the *x* positions.
    - 'edge': Align the left edges of the bars with the *x* positions.

    To align the bars on the right edge pass a negative *width* and
    ``align='edge'``.

Returns
-------
container : `.BarContainer`
    Container with all the bars and optionally errorbars.

Other Parameters
----------------
color : scalar or array-like, optional
    The colors of the bar faces.

edgecolor : scalar or array-like, optional
    The colors of the bar edges.

linewidth : scalar or array-like, optional
    Width of the bar edge(s). If 0, don't draw edges.

tick_label : string or array-like, optional
    The tick labels of the bars.
    Default: None (Use default numeric labels.)

xerr, yerr : scalar or array-like of shape(N,) or shape(2,N), optional
```

```
    If not *None*, add horizontal / vertical errorbars to the bar tips.
    The values are +/- sizes relative to the data:

    - scalar: symmetric +/- values for all bars
    - shape(N,): symmetric +/- values for each bar
    - shape(2,N): Separate - and + values for each bar. First row
        contains the lower errors, the second row contains the
        upper errors.
    - *None*: No errorbar. (Default)

    See :doc:`/gallery/statistics/errorbar_features`
    for an example on the usage of ``xerr`` and ``yerr``.

ecolor : scalar or array-like, optional, default: 'black'
    The line color of the errorbars.

capsize : scalar, optional
   The length of the error bar caps in points.
   Default: None, which will take the value from
   :rc:`errorbar.capsize`.

error_kw : dict, optional
    Dictionary of kwargs to be passed to the `~.Axes.errorbar`
    method. Values of *ecolor* or *capsize* defined here take
    precedence over the independent kwargs.

log : bool, optional, default: False
    If *True*, set the y-axis to be log scale.

orientation : {'vertical',  'horizontal'}, optional
    *This is for internal use only.* Please use `barh` for
    horizontal bar plots. Default: 'vertical'.

See also
--------
barh: Plot a horizontal bar plot.

Notes
-----
The optional arguments *color*, *edgecolor*, *linewidth*,
*xerr*, and *yerr* can be either scalars or sequences of
length equal to the number of bars.  This enables you to use
bar as the basis for stacked bar charts, or candlestick plots.
Detail: *xerr* and *yerr* are passed directly to
:meth:`errorbar`, so they can also have shape 2xN for
independent specification of lower and upper errors.

Other optional kwargs:
```

```
      agg_filter: a filter function, which takes a (m, n, 3) float array and a
dpi value, and returns a (m, n, 3) array
      alpha: float or None
      animated: bool
      antialiased: unknown
      capstyle: {'butt', 'round', 'projecting'}
      clip_box: `.Bbox`
      clip_on: bool
      clip_path: [(`~matplotlib.path.Path`, `.Transform`) | `.Patch` | None]
      color: color
      contains: callable
      edgecolor: color or None or 'auto'
      facecolor: color or None
      figure: `.Figure`
      fill: bool
      gid: str
      hatch: {'/', '\\', '|', '-', '+', 'x', 'o', 'O', '.', '*'}
      in_layout: bool
      joinstyle: {'miter', 'round', 'bevel'}
      label: object
      linestyle: {'-', '--', '-.', ':', '', (offset, on-off-seq), ...}
      linewidth: float or None for default
      path_effects: `.AbstractPathEffect`
      picker: None or bool or float or callable
      rasterized: bool or None
      sketch_params: (scale: float, length: float, randomness: float)
      snap: bool or None
      transform: `.Transform`
      url: str
      visible: bool
      zorder: float

    .. note::
        In addition to the above described arguments, this function can take a
        **data** keyword argument. If such a **data** argument is given, the
        following arguments are replaced by **data[<arg>]**:

        * All arguments with the following names: 'bottom', 'color', 'ecolor',
'edgecolor', 'height', 'left', 'linewidth', 'tick_label', 'width', 'x', 'xerr',
'y', 'yerr'.
        * All positional arguments.

        Objects passed as **data** must support item access (``data[<arg>]``)
and
        membership test (``<arg> in data``).
```

[144]:
```
help(plt.xticks)
```

```
Help on function xticks in module matplotlib.pyplot:

xticks(ticks=None, labels=None, **kwargs)
    Get or set the current tick locations and labels of the x-axis.

    Call signatures::

        locs, labels = xticks()           # Get locations and labels

        xticks(ticks, [labels], **kwargs)  # Set locations and labels

    Parameters
    ----------
    ticks : array_like
        A list of positions at which ticks should be placed. You can pass an
        empty list to disable xticks.

    labels : array_like, optional
        A list of explicit labels to place at the given *locs*.

    **kwargs
        :class:`.Text` properties can be used to control the appearance of
        the labels.

    Returns
    -------
    locs
        An array of label locations.
    labels
        A list of `.Text` objects.

    Notes
    -----
    Calling this function with no arguments (e.g. ``xticks()``) is the pyplot
    equivalent of calling `~.Axes.get_xticks` and `~.Axes.get_xticklabels` on
    the current axes.
    Calling this function with arguments is the pyplot equivalent of calling
    `~.Axes.set_xticks` and `~.Axes.set_xticklabels` on the current axes.

    Examples
    --------
    Get the current locations and labels:

        >>> locs, labels = xticks()
```

```
Set label locations:

    >>> xticks(np.arange(0, 1, step=0.2))

Set text labels:

    >>> xticks(np.arange(5), ('Tom', 'Dick', 'Harry', 'Sally', 'Sue'))

Set text labels and properties:

    >>> xticks(np.arange(12), calendar.month_name[1:13], rotation=20)

Disable xticks:

    >>> xticks([])
```

[ ]: