Activity_Course 6 Automatidata project lab

October 31, 2025

1 Automatidata project

Course 6 - The Nuts and bolts of machine learning

You are a data professional in a data analytics firm called Automatidata. Their client, the New York City Taxi & Limousine Commission (New York City TLC), was impressed with the work you have done and has requested that you build a machine learning model to predict if a customer will not leave a tip. They want to use the model in an app that will alert taxi drivers to customers who are unlikely to tip, since drivers depend on tips.

A notebook was structured and prepared to help you in this project. Please complete the following questions.

2 Course 6 End-of-course project: Build a machine learning model

In this activity, you will practice using tree-based modeling techniques to predict on a binary target class.

The purpose of this model is to find ways to generate more revenue for taxi cab drivers.

The goal of this model is to predict whether or not a customer is a generous tipper.

This activity has three parts:

Part 1: Ethical considerations * Consider the ethical implications of the request

- Should the objective of the model be adjusted?

Part 2: Feature engineering

- Perform feature selection, extraction, and transformation to prepare the data for modeling

Part 3: Modeling

- Build the models, evaluate them, and advise on next steps

Follow the instructions and answer the questions below to complete the activity. Then, complete an Executive Summary using the questions listed on the PACE Strategy Document.

Be sure to complete this activity before moving on. The next course item will provide you with a completed exemplar to compare to your own work.

3 Build a machine learning model

4 PACE stages

Throughout these project notebooks, you'll see references to the problem-solving framework PACE. The following notebook components are labeled with the respective PACE stage: Plan, Analyze, Construct, and Execute.

4.1 PACE: Plan

Consider the questions in your PACE Strategy Document to reflect on the Plan stage.

In this stage, consider the following questions:

1. What are you being asked to do?
2. What are the ethical implications of the model? What are the consequences of your model making errors?

- What is the likely effect of the model when it predicts a false negative (i.e., when the model says a customer will give a tip, but they actually won't)?
- What is the likely effect of the model when it predicts a false positive (i.e., when the model says a customer will not give a tip, but they actually will)?

3. Do the benefits of such a model outweigh the potential problems?
4. Would you proceed with the request to build this model? Why or why not?
5. Can the objective be modified to make it less problematic?

What are you being asked to do? You are being asked to build a machine learning model to predict whether a taxi customer will not leave a tip. The goal is to use this model to alert drivers about customers who are unlikely to tip so they can adjust their service or expectations.

What are the ethical implications of the model? What are the consequences of making errors?

False negatives (predicting a customer will tip when they actually won't) could lead to unfair driver expectations.

False positives (predicting a customer won't tip when they actually do) could lead to discriminatory behavior toward certain customers.

There's a risk of bias if the model indirectly correlates tipping behavior with sensitive factors such as location, time, or customer demographics.

Do the benefits of such a model outweigh the potential problems? Not necessarily. While it might increase revenue for drivers, it could also promote unfair or unethical treatment of passengers. The risk of reinforcing bias or discrimination is high if the model isn't carefully monitored and explained.

Would you proceed with the request to build this model? Why or why not? I would proceed only with modifications — specifically, I'd ensure that sensitive or proxy demographic variables are excluded, that outputs are anonymized, and that the model's purpose is reframed toward improving overall service quality, not labeling individuals.

Can the objective be modified to make it less problematic? Yes. Instead of predicting who won't tip, the objective could focus on predicting factors that influence tipping behavior or identifying service improvements that lead to higher tips. This makes the model more ethical and action-oriented.

Suppose you were to modify the modeling objective so, instead of predicting people who won't tip at all, you predicted people who are particularly generous—those who will tip 20% or more? Consider the following questions:

1. What features do you need to make this prediction?
2. What would be the target variable?
3. What metric should you use to evaluate your model? Do you have enough information to decide this now?

New objective: Predict which customers are particularly generous — e.g., those who tip 20% or more.

What features do you need to make this prediction?

Trip distance, duration, fare amount, payment method, pickup/drop-off time, location, and ride type.

Possibly contextual data like weather or day of the week.

What would be the target variable? A binary variable:

1 = customer tips 20% or more

0 = customer tips less than 20%

What metric should you use to evaluate your model?

Use F1 score to balance precision and recall, since both false positives and false negatives matter.

Also review ROC-AUC for overall discriminative ability.

At this stage, there may not yet be enough information to finalize the metric, but F1 is a good starting point for an imbalanced dataset.

Complete the following steps to begin:

4.1.1 Task 1. Imports and data loading

Import packages and libraries needed to build and evaluate random forest and XGBoost classification models.

[197]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.metrics import accuracy_score, precision_score, recall_score,\
f1_score, confusion_matrix, ConfusionMatrixDisplay, RocCurveDisplay
```

```
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# This is the function that helps plot feature importance
from xgboost import plot_importance
```

[123]:
```
# RUN THIS CELL TO SEE ALL COLUMNS
# This lets us see all of the columns, preventing Juptyer from redacting them.
pd.set_option('display.max_columns', None)
```

Begin by reading in the data. There are two dataframes: one containing the original data, the other containing the mean durations, mean distances, and predicted fares from the previous course's project called nyc_preds_means.csv.

Note: `Pandas` reads in the dataset as `df0`, now inspect the first five rows. As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[124]:
```
# RUN THE CELL BELOW TO IMPORT YOUR DATA.

# Load dataset into dataframe
df0 = pd.read_csv('2017_Yellow_Taxi_Trip_Data.csv')

# Import predicted fares and mean distance and duration from previous course
nyc_preds_means = pd.read_csv('nyc_preds_means.csv')
```

Inspect the first few rows of `df0`.

[125]:
```
# Inspect the first few rows of df0
df0.head(10)
```

[125]:
```
   Unnamed: 0  VendorID    tpep_pickup_datetime   tpep_dropoff_datetime  \
passenger_count  trip_distance  RatecodeID  \
0    24870114         2   03/25/2017 8:55:43 AM   03/25/2017 9:09:47 AM
6           3.34           1
1    35634249         1   04/11/2017 2:53:28 PM   04/11/2017 3:19:58 PM
1           1.80           1
2   106203690         1   12/15/2017 7:26:56 AM   12/15/2017 7:34:08 AM
1           1.00           1
3    38942136         2   05/07/2017 1:17:59 PM   05/07/2017 1:48:14 PM
1           3.70           1
4    30841670         2  04/15/2017 11:32:20 PM  04/15/2017 11:49:03 PM
1           4.37           1
5    23345809         2   03/25/2017 8:34:11 PM   03/25/2017 8:42:11 PM
6           2.30           1
6    37660487         2   05/03/2017 7:04:09 PM   05/03/2017 8:03:47 PM
```

```
1         12.83          1
7    69059411         2   08/15/2017 5:41:06 PM   08/15/2017 6:03:05 PM
1          2.98          1
8     8433159         2   02/04/2017 4:17:07 PM   02/04/2017 4:29:14 PM
1          1.20          1
9    95294817         1   11/10/2017 3:20:29 PM   11/10/2017 3:40:55 PM
1          1.60          1

  store_and_fwd_flag  PULocationID  DOLocationID  payment_type  fare_amount
extra  mta_tax  tip_amount  tolls_amount  \
0                  N           100           231             1         13.0
0.0      0.5        2.76           0.0
1                  N           186            43             1         16.0
0.0      0.5        4.00           0.0
2                  N           262           236             1          6.5
0.0      0.5        1.45           0.0
3                  N           188            97             1         20.5
0.0      0.5        6.39           0.0
4                  N             4           112             2         16.5
0.5      0.5        0.00           0.0
5                  N           161           236             1          9.0
0.5      0.5        2.06           0.0
6                  N            79           241             1         47.5
1.0      0.5        9.86           0.0
7                  N           237           114             1         16.0
1.0      0.5        1.78           0.0
8                  N           234           249             2          9.0
0.0      0.5        0.00           0.0
9                  N           239           237             1         13.0
0.0      0.5        2.75           0.0

   improvement_surcharge  total_amount
0                    0.3         16.56
1                    0.3         20.80
2                    0.3          8.75
3                    0.3         27.69
4                    0.3         17.80
5                    0.3         12.36
6                    0.3         59.16
7                    0.3         19.58
8                    0.3          9.80
9                    0.3         16.55
```

Inspect the first few rows of `nyc_preds_means`.

[126]:
```
# Inspect the first few rows of `nyc_preds_means`
nyc_preds_means.head()
```

```
[126]:    mean_duration  mean_distance  predicted_fare
       0      22.847222       3.521667       16.434245
       1      24.470370       3.108889       16.052218
       2       7.250000       0.881429        7.053706
       3      30.250000       3.700000       18.731650
       4      14.616667       4.435000       15.845642
```

Join the two dataframes Join the two dataframes using a method of your choice.

```
[164]: df0 = df0.merge(nyc_preds_means,
                       left_index=True,
                       right_index=True)

       df0.head()
```

```
[164]:    Unnamed: 0  VendorID    tpep_pickup_datetime   tpep_dropoff_datetime  \
       passenger_count  trip_distance  RatecodeID  \
       0    24870114         2   03/25/2017 8:55:43 AM   03/25/2017 9:09:47 AM
       6             3.34           1
       1    35634249         1   04/11/2017 2:53:28 PM   04/11/2017 3:19:58 PM
       1             1.80           1
       2   106203690         1   12/15/2017 7:26:56 AM   12/15/2017 7:34:08 AM
       1             1.00           1
       3    38942136         2   05/07/2017 1:17:59 PM   05/07/2017 1:48:14 PM
       1             3.70           1
       4    30841670         2  04/15/2017 11:32:20 PM  04/15/2017 11:49:03 PM
       1             4.37           1

         store_and_fwd_flag  PULocationID  DOLocationID  payment_type  fare_amount
       extra  mta_tax  tip_amount  tolls_amount  \
       0                  N           100           231             1         13.0
       0.0      0.5        2.76           0.0
       1                  N           186            43             1         16.0
       0.0      0.5        4.00           0.0
       2                  N           262           236             1          6.5
       0.0      0.5        1.45           0.0
       3                  N           188            97             1         20.5
       0.0      0.5        6.39           0.0
       4                  N             4           112             2         16.5
       0.5      0.5        0.00           0.0

          improvement_surcharge  total_amount  mean_duration  mean_distance
       predicted_fare
       0                    0.3         16.56      22.847222       3.521667
       16.434245
       1                    0.3         20.80      24.470370       3.108889
       16.052218
```

```
2                     0.3          8.75      7.250000       0.881429
7.053706
3                     0.3         27.69     30.250000       3.700000
18.731650
4                     0.3         17.80     14.616667       4.435000
15.845642
```

4.2 PACE: Analyze

Consider the questions in your PACE Strategy Documentto reflect on the Analyze stage.

4.2.1 Task 2. Feature engineering

You have already prepared much of this data and performed exploratory data analysis (EDA) in previous courses.

Call `info()` on the new combined dataframe.

[165]:
```
df0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22699 entries, 0 to 22698
Data columns (total 21 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             22699 non-null  int64
 1   VendorID               22699 non-null  int64
 2   tpep_pickup_datetime   22699 non-null  object
 3   tpep_dropoff_datetime  22699 non-null  object
 4   passenger_count        22699 non-null  int64
 5   trip_distance          22699 non-null  float64
 6   RatecodeID             22699 non-null  int64
 7   store_and_fwd_flag     22699 non-null  object
 8   PULocationID           22699 non-null  int64
 9   DOLocationID           22699 non-null  int64
 10  payment_type           22699 non-null  int64
 11  fare_amount            22699 non-null  float64
 12  extra                  22699 non-null  float64
 13  mta_tax                22699 non-null  float64
 14  tip_amount             22699 non-null  float64
 15  tolls_amount           22699 non-null  float64
 16  improvement_surcharge  22699 non-null  float64
 17  total_amount           22699 non-null  float64
 18  mean_duration          22699 non-null  float64
 19  mean_distance          22699 non-null  float64
 20  predicted_fare         22699 non-null  float64
dtypes: float64(11), int64(7), object(3)
memory usage: 3.6+ MB
```

You know from your EDA that customers who pay cash generally have a tip amount of \$0. To meet the modeling objective, you'll need to sample the data to select only the customers who pay with credit card.

Copy `df0` and assign the result to a variable called `df1`. Then, use a Boolean mask to filter `df1` so it contains only customers who paid with credit card.

[166]:
```
df1 = df0[df0['payment_type']==1]
```

Target Notice that there isn't a column that indicates tip percent, which is what you need to create the target variable. You'll have to engineer it.

Add a `tip_percent` column to the dataframe by performing the following calculation:

$$tip\ percent = \frac{tip\ amount}{total\ amount - tip\ amount}$$

Round the result to three places beyond the decimal. **This is an important step.** It affects how many customers are labeled as generous tippers. In fact, without performing this step, approximately 1,800 people who do tip 20% would be labeled as not generous.

To understand why, you must consider how floats work. Computers make their calculations using floating-point arithmetic (hence the word "float"). Floating-point arithmetic is a system that allows computers to express both very large numbers and very small numbers with a high degree of precision, encoded in binary. However, precision is limited by the number of bits used to represent a number, which is generally 32 or 64, depending on the capabilities of your operating system.

This comes with limitations in that sometimes calculations that should result in clean, precise values end up being encoded as very long decimals. Take, for example, the following calculation:

[127]:
```
# Run this cell
1.1 + 2.2
```

[127]:
```
3.3000000000000003
```

Notice the three that is 16 places to the right of the decimal. As a consequence, if you were to then have a step in your code that identifies values 3.3, this would not be included in the result. Therefore, whenever you perform a calculation to compute a number that is then used to make an important decision or filtration, round the number. How many degrees of precision you round to is your decision, which should be based on your use case.

Refer to this guide for more information related to floating-point arithmetic.

[167]:
```
df1['tip_percent'] = round(df1['tip_amount'] / (df1['total_amount'] - df1['tip_amount']), 3)
```

Now create another column called `generous`. This will be the target variable. The column should be a binary indicator of whether or not a customer tipped 20% (0=no, 1=yes).

1. Begin by making the `generous` column a copy of the `tip_percent` column.

2. Reassign the column by converting it to Boolean (True/False).
3. Reassign the column by converting Boolean to binary (1/0).

[168]:
```
# Create 'generous' col (target)
df1['generous'] = df1['tip_percent']
df1['generous'] = (df1['generous'] >= 0.2)
df1['generous'] = df1['generous'].astype(int)
```

HINT

To convert from Boolean to binary, use `.astype(int)` on the column.

[78]:
```

```

Create day column Next, you're going to be working with the pickup and dropoff columns.

Convert the `tpep_pickup_datetime` and `tpep_dropoff_datetime` columns to datetime.

[169]:
```
# Convert pickup and dropoff cols to datetime
df1['tpep_pickup_datetime'] = pd.to_datetime(df1['tpep_pickup_datetime'],␣
↪format='%m/%d/%Y %I:%M:%S %p')
df1['tpep_dropoff_datetime'] = pd.to_datetime(df1['tpep_dropoff_datetime'],␣
↪format='%m/%d/%Y %I:%M:%S %p')
```

Create a `day` column that contains only the day of the week when each passenger was picked up. Then, convert the values to lowercase.

[170]:
```
# Create a 'day' col
df1['day'] = df1['tpep_pickup_datetime'].dt.day_name().str.lower()
df1[['tpep_pickup_datetime', 'day']].head()
```

[170]:

	tpep_pickup_datetime	day
0	2017-03-25 08:55:43	saturday
1	2017-04-11 14:53:28	tuesday
2	2017-12-15 07:26:56	friday
3	2017-05-07 13:17:59	sunday
5	2017-03-25 20:34:11	saturday

HINT

To convert to day name, use `dt.day_name()` on the column.

Create time of day columns Next, engineer four new columns that represent time of day bins. Each column should contain binary values (0=no, 1=yes) that indicate whether a trip began (picked up) during the following times:

`am_rush` = [06:00–10:00)
`daytime` = [10:00–16:00)
`pm_rush` = [16:00–20:00)
`nighttime` = [20:00–06:00)

To do this, first create the four columns. For now, each new column should be identical and contain the same information: the hour (only) from the `tpep_pickup_datetime` column.

[171]:
```
# Create 'am_rush' col
df1['am_rush'] = df1['tpep_pickup_datetime'].dt.hour

# Create 'daytime' col
df1['daytime'] = df1['tpep_pickup_datetime'].dt.hour

# Create 'pm_rush' col
df1['pm_rush'] = df1['tpep_pickup_datetime'].dt.hour

# Create 'nighttime' col
df1['nighttime'] = df1['tpep_pickup_datetime'].dt.hour
```

You'll need to write four functions to convert each new column to binary (0/1). Begin with `am_rush`. Complete the function so if the hour is between [06:00–10:00), it returns 1, otherwise, it returns 0.

[172]:
```
# Define 'am_rush()' conversion function [06:00-10:00)
def am_rush(hour):
    if 6 <= hour['am_rush'] < 10:
        val = 1
    else:
        val = 0
    return val
```

Now, apply the `am_rush()` function to the `am_rush` series to perform the conversion. Print the first five values of the column to make sure it did what you expected it to do.

Note: Be careful! If you run this cell twice, the function will be reapplied and the values will all be changed to 0.

[176]:
```
# Apply 'am_rush' function to the 'am_rush' series
df1['am_rush'] = df1.apply(am_rush, axis=1)
df1['am_rush'].head()
```

[176]:
```
0    0
1    0
2    0
3    0
5    0
Name: am_rush, dtype: int64
```

Write functions to convert the three remaining columns and apply them to their respective series.

[177]:
```
# Define 'daytime()' conversion function [10:00-16:00)
def daytime(hour):
    if 10 <= hour['daytime'] < 16:
        val = 1
```

```
    else:
        val = 0
    return val
```

[178]:
```
# Apply 'daytime' function to the 'daytime' series
df1['daytime'] = df1.apply(daytime, axis=1)
```

[179]:
```
# Define 'pm_rush()' conversion function [16:00-20:00)
def pm_rush(hour):
    if 16 <= hour['pm_rush'] < 20:
        val = 1
    else:
        val = 0
    return val
```

[180]:
```
# Apply 'pm_rush' function to the 'pm_rush' series
df1['pm_rush'] = df1.apply(pm_rush, axis=1)
```

[181]:
```
# Define 'nighttime()' conversion function [20:00-06:00)
def nighttime(hour):
    if 20 <= hour['nighttime'] < 24:
        val = 1
    elif 0 <= hour['nighttime'] < 6:
        val = 1
    else:
        val = 0
    return val
```

[182]:
```
# Apply 'nighttime' function to the 'nighttime' series
df1['nighttime'] = df1.apply(nighttime, axis=1)
```

Create `month` column Now, create a `month` column that contains only the abbreviated name of the month when each passenger was picked up, then convert the result to lowercase.

HINT

Refer to the strftime cheatsheet for help.

[183]:
```
# Create 'month' col
df1['month'] = df1['tpep_pickup_datetime'].dt.strftime('%b').str.lower()
```

Examine the first five rows of your dataframe.

[184]:
```
df1.head()
```

[184]:
```
   Unnamed: 0  VendorID tpep_pickup_datetime tpep_dropoff_datetime
passenger_count  trip_distance  RatecodeID  \
0    24870114         2  2017-03-25 08:55:43   2017-03-25 09:09:47
```

```
6          3.34          1
1    35634249         1  2017-04-11 14:53:28   2017-04-11 15:19:58
1          1.80          1
2   106203690         1  2017-12-15 07:26:56   2017-12-15 07:34:08
1          1.00          1
3    38942136         2  2017-05-07 13:17:59   2017-05-07 13:48:14
1          3.70          1
5    23345809         2  2017-03-25 20:34:11   2017-03-25 20:42:11
6          2.30          1

  store_and_fwd_flag  PULocationID  DOLocationID  payment_type  fare_amount
extra  mta_tax  tip_amount  tolls_amount  \
0                  N           100           231             1         13.0
0.0      0.5        2.76           0.0
1                  N           186            43             1         16.0
0.0      0.5        4.00           0.0
2                  N           262           236             1          6.5
0.0      0.5        1.45           0.0
3                  N           188            97             1         20.5
0.0      0.5        6.39           0.0
5                  N           161           236             1          9.0
0.5      0.5        2.06           0.0

   improvement_surcharge  total_amount  mean_duration  mean_distance
predicted_fare  tip_percent  generous       day  \
0                    0.3         16.56      22.847222       3.521667
16.434245        0.200          1  saturday
1                    0.3         20.80      24.470370       3.108889
16.052218        0.238          1   tuesday
2                    0.3          8.75       7.250000       0.881429
7.053706        0.199          0    friday
3                    0.3         27.69      30.250000       3.700000
18.731650        0.300          1    sunday
5                    0.3         12.36      11.855376       2.052258
10.441351        0.200          1  saturday

   am_rush  daytime  pm_rush  nighttime month
0        0        0        0          0   mar
1        0        0        0          0   apr
2        0        0        0          0   dec
3        0        0        0          0   may
5        0        0        0          1   mar
```

Drop columns Drop redundant and irrelevant columns as well as those that would not be available when the model is deployed. This includes information like payment type, trip distance, tip amount, tip percentage, total amount, toll amount, etc. The target variable (`generous`) must remain in the data because it will get isolated as the `y` data for modeling.

[185]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15265 entries, 0 to 22698
Data columns (total 29 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             15265 non-null  int64
 1   VendorID               15265 non-null  int64
 2   tpep_pickup_datetime   15265 non-null  datetime64[ns]
 3   tpep_dropoff_datetime  15265 non-null  datetime64[ns]
 4   passenger_count        15265 non-null  int64
 5   trip_distance          15265 non-null  float64
 6   RatecodeID             15265 non-null  int64
 7   store_and_fwd_flag     15265 non-null  object
 8   PULocationID           15265 non-null  int64
 9   DOLocationID           15265 non-null  int64
 10  payment_type           15265 non-null  int64
 11  fare_amount            15265 non-null  float64
 12  extra                  15265 non-null  float64
 13  mta_tax                15265 non-null  float64
 14  tip_amount             15265 non-null  float64
 15  tolls_amount           15265 non-null  float64
 16  improvement_surcharge  15265 non-null  float64
 17  total_amount           15265 non-null  float64
 18  mean_duration          15265 non-null  float64
 19  mean_distance          15265 non-null  float64
 20  predicted_fare         15265 non-null  float64
 21  tip_percent            15262 non-null  float64
 22  generous               15265 non-null  int64
 23  day                    15265 non-null  object
 24  am_rush                15265 non-null  int64
 25  daytime                15265 non-null  int64
 26  pm_rush                15265 non-null  int64
 27  nighttime              15265 non-null  int64
 28  month                  15265 non-null  object
dtypes: datetime64[ns](2), float64(12), int64(12), object(3)
memory usage: 3.5+ MB
```

[186]:
```
# Drop columns
drop_cols = ['Unnamed: 0', 'tpep_pickup_datetime', 'tpep_dropoff_datetime',
             'payment_type', 'trip_distance', 'store_and_fwd_flag',␣
 ↪'payment_type',
             'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount',
             'improvement_surcharge', 'total_amount', 'tip_percent']

df1 = df1.drop(drop_cols, axis=1)
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15265 entries, 0 to 22698
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   VendorID         15265 non-null  int64
 1   passenger_count  15265 non-null  int64
 2   RatecodeID       15265 non-null  int64
 3   PULocationID     15265 non-null  int64
 4   DOLocationID     15265 non-null  int64
 5   mean_duration    15265 non-null  float64
 6   mean_distance    15265 non-null  float64
 7   predicted_fare   15265 non-null  float64
 8   generous         15265 non-null  int64
 9   day              15265 non-null  object
 10  am_rush          15265 non-null  int64
 11  daytime          15265 non-null  int64
 12  pm_rush          15265 non-null  int64
 13  nighttime        15265 non-null  int64
 14  month            15265 non-null  object
dtypes: float64(3), int64(10), object(2)
memory usage: 1.9+ MB
```

Variable encoding Many of the columns are categorical and will need to be dummied (converted to binary). Some of these columns are numeric, but they actually encode categorical information, such as `RatecodeID` and the pickup and dropoff locations. To make these columns recognizable to the `get_dummies()` function as categorical variables, you'll first need to convert them to `type(str)`.

1. Define a variable called `cols_to_str`, which is a list of the numeric columns that contain categorical information and must be converted to string: `RatecodeID`, `PULocationID`, `DOLocationID`.
2. Write a for loop that converts each column in `cols_to_str` to string.

[187]:
```
# 1. Define list of cols to convert to string
cols_to_str = ['RatecodeID', 'PULocationID', 'DOLocationID', 'VendorID']

# 2. Convert each column to string
for col in cols_to_str:
    df1[col] = df1[col].astype('str')
```

[150]:
```
df1[cols_to_str].dtypes
```

[150]:
```
RatecodeID      object
PULocationID    object
DOLocationID    object
dtype: object
```

HINT

To convert to string, use `astype(str)` on the column.

Now convert all the categorical columns to binary.

1. Call `get_dummies()` on the dataframe and assign the results back to a new dataframe called `df2`.

[190]:
```
# Convert categoricals to binary
df2 = pd.get_dummies(df1, drop_first=True)
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15265 entries, 0 to 22698
Columns: 347 entries, passenger_count to month_sep
dtypes: bool(338), float64(3), int64(6)
memory usage: 6.1 MB
```

[191]:
```
df2.info()
df2.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15265 entries, 0 to 22698
Columns: 347 entries, passenger_count to month_sep
dtypes: bool(338), float64(3), int64(6)
memory usage: 6.1 MB
```

[191]:
```
   passenger_count  mean_duration  mean_distance  predicted_fare  generous
am_rush  daytime  pm_rush  nighttime  \
0                6      22.847222       3.521667       16.434245         1
0        0        0          0
1                1      24.470370       3.108889       16.052218         1
0        0        0          0
2                1       7.250000       0.881429        7.053706         0
0        0        0          0
3                1      30.250000       3.700000       18.731650         1
0        0        0          0
5                6      11.855376       2.052258       10.441351         1
0        0        0          1

   VendorID_2  RatecodeID_2  RatecodeID_3  RatecodeID_4  RatecodeID_5
RatecodeID_99  PULocationID_10  \
0        True         False         False         False         False
False            False
1       False         False         False         False         False
False            False
2       False         False         False         False         False
False            False
```

```
3       True        False        False        False        False
False           False
5       True        False        False        False        False
False           False

   PULocationID_100  PULocationID_106  PULocationID_107  PULocationID_112
PULocationID_113  PULocationID_114  \
0              True             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_116  PULocationID_12  PULocationID_123  PULocationID_125
PULocationID_127  PULocationID_128  \
0             False            False             False             False
False             False
1             False            False             False             False
False             False
2             False            False             False             False
False             False
3             False            False             False             False
False             False
5             False            False             False             False
False             False

   PULocationID_129  PULocationID_13  PULocationID_130  PULocationID_131
PULocationID_132  PULocationID_133  \
0             False            False             False             False
False             False
1             False            False             False             False
False             False
2             False            False             False             False
False             False
3             False            False             False             False
False             False
5             False            False             False             False
False             False

   PULocationID_134  PULocationID_135  PULocationID_137  PULocationID_138
PULocationID_140  PULocationID_141  \
0             False             False             False             False
```

```
False              False
1             False              False              False              False
False              False
2             False              False              False              False
False              False
3             False              False              False              False
False              False
5             False              False              False              False
False              False

   PULocationID_142  PULocationID_143  PULocationID_144  PULocationID_145
PULocationID_146  PULocationID_148  \
0             False              False              False              False
False              False
1             False              False              False              False
False              False
2             False              False              False              False
False              False
3             False              False              False              False
False              False
5             False              False              False              False
False              False

   PULocationID_151  PULocationID_152  PULocationID_153  PULocationID_158
PULocationID_161  PULocationID_162  \
0             False              False              False              False
False              False
1             False              False              False              False
False              False
2             False              False              False              False
False              False
3             False              False              False              False
False              False
5             False              False              False              False
True              False

   PULocationID_163  PULocationID_164  PULocationID_166  PULocationID_17
PULocationID_170  PULocationID_173  \
0             False              False              False             False
False              False
1             False              False              False             False
False              False
2             False              False              False             False
False              False
3             False              False              False             False
False              False
5             False              False              False             False
```

```
False             False

   PULocationID_179  PULocationID_181  PULocationID_186  PULocationID_188
PULocationID_189  PULocationID_190  \
0             False             False             False             False
False             False
1             False             False              True             False
False             False
2             False             False             False             False
False             False
3             False             False             False              True
False             False
5             False             False             False             False
False             False

   PULocationID_193  PULocationID_196  PULocationID_208  PULocationID_209
PULocationID_211  PULocationID_213  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_216  PULocationID_218  PULocationID_223  PULocationID_224
PULocationID_225  PULocationID_226  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_229  PULocationID_230  PULocationID_231  PULocationID_232
PULocationID_233  PULocationID_234  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
```

```
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_236  PULocationID_237  PULocationID_238  PULocationID_239
PULocationID_24  PULocationID_243  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_244  PULocationID_246  PULocationID_247  PULocationID_249
PULocationID_25  PULocationID_255  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_256  PULocationID_258  PULocationID_260  PULocationID_261
PULocationID_262  PULocationID_263  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
True             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_264  PULocationID_265  PULocationID_28  PULocationID_33
```

```
PULocationID_35  PULocationID_36  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   PULocationID_37  PULocationID_4  PULocationID_40  PULocationID_41
PULocationID_42  PULocationID_43  \
0            False            False             False             False
False             False
1            False            False             False             False
False             False
2            False            False             False             False
False             False
3            False            False             False             False
False             False
5            False            False             False             False
False             False

   PULocationID_45  PULocationID_48  PULocationID_49  PULocationID_50
PULocationID_52  PULocationID_57  \
0            False             False             False             False
False             False
1            False             False             False             False
False             False
2            False             False             False             False
False             False
3            False             False             False             False
False             False
5            False             False             False             False
False             False

   PULocationID_61  PULocationID_62  PULocationID_65  PULocationID_66
PULocationID_68  PULocationID_7  \
0            False             False             False             False
False            False
1            False             False             False             False
False            False
2            False             False             False             False
False            False
3            False             False             False             False
```

```
False           False
5           False            False            False            False
False           False

   PULocationID_70  PULocationID_74  PULocationID_75  PULocationID_79
PULocationID_80  PULocationID_82  \
0           False            False            False            False
False            False
1           False            False            False            False
False            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   PULocationID_87  PULocationID_88  PULocationID_90  PULocationID_91
PULocationID_92  PULocationID_93  \
0           False            False            False            False
False            False
1           False            False            False            False
False            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   PULocationID_95  PULocationID_97  DOLocationID_10  DOLocationID_100
DOLocationID_102  DOLocationID_106  \
0           False            False            False             False
False             False
1           False            False            False             False
False             False
2           False            False            False             False
False             False
3           False            False            False             False
False             False
5           False            False            False             False
False             False

   DOLocationID_107  DOLocationID_11  DOLocationID_112  DOLocationID_113
DOLocationID_114  DOLocationID_116  \
0            False            False             False             False
False             False
```

```
1             False            False             False             False
False             False
2             False            False             False             False
False             False
3             False            False             False             False
False             False
5             False            False             False             False
False             False

   DOLocationID_117  DOLocationID_118  DOLocationID_119  DOLocationID_12
DOLocationID_120  DOLocationID_121  \
0             False             False             False            False
False             False
1             False             False             False            False
False             False
2             False             False             False            False
False             False
3             False             False             False            False
False             False
5             False             False             False            False
False             False

   DOLocationID_123  DOLocationID_124  DOLocationID_125  DOLocationID_126
DOLocationID_127  DOLocationID_129  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   DOLocationID_13  DOLocationID_130  DOLocationID_131  DOLocationID_132
DOLocationID_133  DOLocationID_134  \
0            False             False             False             False
False             False
1            False             False             False             False
False             False
2            False             False             False             False
False             False
3            False             False             False             False
False             False
5            False             False             False             False
False             False
```

```
   DOLocationID_135  DOLocationID_136  DOLocationID_137  DOLocationID_138
DOLocationID_14  DOLocationID_140  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   DOLocationID_141  DOLocationID_142  DOLocationID_143  DOLocationID_144
DOLocationID_145  DOLocationID_146  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   DOLocationID_147  DOLocationID_148  DOLocationID_15  DOLocationID_151
DOLocationID_152  DOLocationID_153  \
0             False             False            False             False
False             False
1             False             False            False             False
False             False
2             False             False            False             False
False             False
3             False             False            False             False
False             False
5             False             False            False             False
False             False

   DOLocationID_157  DOLocationID_158  DOLocationID_159  DOLocationID_16
DOLocationID_160  DOLocationID_161  \
0             False             False             False            False
False             False
1             False             False             False            False
False             False
2             False             False             False            False
```

```
False              False
3             False              False              False              False
False              False
5             False              False              False              False
False              False

   DOLocationID_162  DOLocationID_163  DOLocationID_164  DOLocationID_166
DOLocationID_168  DOLocationID_169  \
0             False              False              False              False
False              False
1             False              False              False              False
False              False
2             False              False              False              False
False              False
3             False              False              False              False
False              False
5             False              False              False              False
False              False

   DOLocationID_17  DOLocationID_170  DOLocationID_173  DOLocationID_174
DOLocationID_175  DOLocationID_177  \
0            False              False              False              False
False              False
1            False              False              False              False
False              False
2            False              False              False              False
False              False
3            False              False              False              False
False              False
5            False              False              False              False
False              False

   DOLocationID_178  DOLocationID_179  DOLocationID_180  DOLocationID_181
DOLocationID_182  DOLocationID_183  \
0             False              False              False              False
False              False
1             False              False              False              False
False              False
2             False              False              False              False
False              False
3             False              False              False              False
False              False
5             False              False              False              False
False              False

   DOLocationID_186  DOLocationID_188  DOLocationID_189  DOLocationID_19
DOLocationID_192  DOLocationID_193  \
```

```
0              False              False              False              False
False              False
1              False              False              False              False
False              False
2              False              False              False              False
False              False
3              False              False              False              False
False              False
5              False              False              False              False
False              False

   DOLocationID_194  DOLocationID_195  DOLocationID_196  DOLocationID_197
DOLocationID_198  DOLocationID_200  \
0              False              False              False               False
False              False
1              False              False              False               False
False              False
2              False              False              False               False
False              False
3              False              False              False               False
False              False
5              False              False              False               False
False              False

   DOLocationID_202  DOLocationID_208  DOLocationID_209  DOLocationID_21
DOLocationID_210  DOLocationID_211  \
0              False              False              False              False
False              False
1              False              False              False              False
False              False
2              False              False              False              False
False              False
3              False              False              False              False
False              False
5              False              False              False              False
False              False

   DOLocationID_212  DOLocationID_213  DOLocationID_216  DOLocationID_217
DOLocationID_218  DOLocationID_22  \
0              False              False              False               False
False             False
1              False              False              False               False
False             False
2              False              False              False               False
False             False
3              False              False              False               False
False             False
```

```
5           False            False            False            False
False          False

   DOLocationID_220  DOLocationID_223  DOLocationID_224  DOLocationID_225
DOLocationID_226  DOLocationID_228  \
0           False            False            False            False
False            False
1           False            False            False            False
False            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   DOLocationID_229  DOLocationID_23  DOLocationID_230  DOLocationID_231
DOLocationID_232  DOLocationID_233  \
0           False            False            False             True
False            False
1           False            False            False            False
False            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   DOLocationID_234  DOLocationID_235  DOLocationID_236  DOLocationID_237
DOLocationID_238  DOLocationID_239  \
0           False            False            False            False
False            False
1           False            False            False            False
False            False
2           False            False             True            False
False            False
3           False            False            False            False
False            False
5           False            False             True            False
False            False

   DOLocationID_24  DOLocationID_240  DOLocationID_241  DOLocationID_242
DOLocationID_243  DOLocationID_244  \
0          False            False            False            False
False            False
1          False            False            False            False
```

```
False             False
2          False             False             False             False
False             False
3          False             False             False             False
False             False
5          False             False             False             False
False             False

   DOLocationID_246  DOLocationID_247  DOLocationID_248  DOLocationID_249
DOLocationID_25  DOLocationID_252  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   DOLocationID_255  DOLocationID_256  DOLocationID_257  DOLocationID_259
DOLocationID_26  DOLocationID_260  \
0             False             False             False             False
False             False
1             False             False             False             False
False             False
2             False             False             False             False
False             False
3             False             False             False             False
False             False
5             False             False             False             False
False             False

   DOLocationID_261  DOLocationID_262  DOLocationID_263  DOLocationID_264
DOLocationID_265  DOLocationID_28  \
0             False             False             False             False
False            False
1             False             False             False             False
False            False
2             False             False             False             False
False            False
3             False             False             False             False
False            False
5             False             False             False             False
False            False
```

```
   DOLocationID_29  DOLocationID_32  DOLocationID_33  DOLocationID_36
DOLocationID_37  DOLocationID_39  \
0            False            False            False            False
False            False
1            False            False            False            False
False            False
2            False            False            False            False
False            False
3            False            False            False            False
False            False
5            False            False            False            False
False            False

   DOLocationID_4  DOLocationID_40  DOLocationID_41  DOLocationID_42
DOLocationID_43  DOLocationID_45  \
0           False            False            False            False
False            False
1           False            False            False            False
True            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   DOLocationID_47  DOLocationID_48  DOLocationID_49  DOLocationID_50
DOLocationID_51  DOLocationID_52  \
0            False            False            False            False
False            False
1            False            False            False            False
False            False
2            False            False            False            False
False            False
3            False            False            False            False
False            False
5            False            False            False            False
False            False

   DOLocationID_53  DOLocationID_54  DOLocationID_55  DOLocationID_56
DOLocationID_61  DOLocationID_62  \
0            False            False            False            False
False            False
1            False            False            False            False
False            False
2            False            False            False            False
False            False
```

```
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   DOLocationID_63  DOLocationID_64  DOLocationID_65  DOLocationID_66
DOLocationID_67  DOLocationID_68  \
0           False            False            False            False
False            False
1           False            False            False            False
False            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   DOLocationID_69  DOLocationID_7  DOLocationID_70  DOLocationID_71
DOLocationID_72  DOLocationID_74  \
0           False           False            False            False
False            False
1           False           False            False            False
False            False
2           False           False            False            False
False            False
3           False           False            False            False
False            False
5           False           False            False            False
False            False

   DOLocationID_75  DOLocationID_76  DOLocationID_77  DOLocationID_79
DOLocationID_80  DOLocationID_81  \
0           False            False            False            False
False            False
1           False            False            False            False
False            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   DOLocationID_82  DOLocationID_83  DOLocationID_85  DOLocationID_86
DOLocationID_87  DOLocationID_88  \
0           False            False            False            False
```

```
False            False
1           False            False            False            False
False            False
2           False            False            False            False
False            False
3           False            False            False            False
False            False
5           False            False            False            False
False            False

   DOLocationID_89  DOLocationID_9  DOLocationID_90  DOLocationID_91
DOLocationID_92  DOLocationID_93  \
0           False           False            False            False
False            False
1           False           False            False            False
False            False
2           False           False            False            False
False            False
3           False           False            False            False
False            False
5           False           False            False            False
False            False

   DOLocationID_95  DOLocationID_97  day_monday  day_saturday  day_sunday
day_thursday  day_tuesday  day_wednesday  \
0           False            False       False          True       False
False        False          False
1           False            False       False         False       False
False         True          False
2           False            False       False         False       False
False        False          False
3           False             True       False         False        True
False        False          False
5           False            False       False          True       False
False        False          False

   month_aug  month_dec  month_feb  month_jan  month_jul  month_jun  month_mar
month_may  month_nov  month_oct  \
0      False      False      False      False      False      False       True
False      False      False
1      False      False      False      False      False      False      False
False      False      False
2      False       True      False      False      False      False      False
False      False      False
3      False      False      False      False      False      False      False
True      False      False
5      False      False      False      False      False      False       True
```

```
False      False      False

   month_sep
0      False
1      False
2      False
3      False
5      False
```

[193]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15265 entries, 0 to 22698
Columns: 347 entries, passenger_count to month_sep
dtypes: bool(338), float64(3), int64(6)
memory usage: 6.1 MB
```

Evaluation metric Before modeling, you must decide on an evaluation metric.

1. Examine the class balance of your target variable.

[189]:
```
# Get class balance of 'generous' col
df2['generous'].value_counts(normalize=True)
```

[189]:
```
generous
1    0.526368
0    0.473632
Name: proportion, dtype: float64
```

A little over half of the customers in this dataset were "generous" (tipped 20%). The dataset is very nearly balanced.

To determine a metric, consider the cost of both kinds of model error: * False positives (the model predicts a tip 20%, but the customer does not give one) * False negatives (the model predicts a tip < 20%, but the customer gives more)

False positives are worse for cab drivers, because they would pick up a customer expecting a good tip and then not receive one, frustrating the driver.

False negatives are worse for customers, because a cab driver would likely pick up a different customer who was predicted to tip more—even when the original customer would have tipped generously.

The stakes are relatively even. You want to help taxi drivers make more money, but you don't want this to anger customers. Your metric should weigh both precision and recall equally. Which metric is this?

Since the dataset is nearly balanced and both false positives and false negatives carry similar costs, the best evaluation metric is the F1 score, which balances precision and recall equally.

4.3 PACE: Construct

Consider the questions in your PACE Strategy Document to reflect on the Construct stage.

4.3.1 Task 3. Modeling

Split the data Now you're ready to model. The only remaining step is to split the data into features/target variable and training/testing data.

1. Define a variable `y` that isolates the target variable (`generous`).
2. Define a variable `X` that isolates the features.
3. Split the data into training and testing sets. Put 20% of the samples into the test set, stratify the data, and set the random state.

[194]:
```
# Isolate target variable (y)
y = df2['generous']

# Isolate the features (X)
X = df2.drop('generous', axis=1)

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,␣
↪test_size=0.2, random_state=42)
```

Random forest Begin with using `GridSearchCV` to tune a random forest model.

1. Instantiate the random forest classifier `rf` and set the random state.
2. Create a dictionary `cv_params` of any of the following hyperparameters and their corresponding values to tune. The more you tune, the better your model will fit the data, but the longer it will take.

- `max_depth`
- `max_features`
- `max_samples`
- `min_samples_leaf`
- `min_samples_split`
- `n_estimators`

3. Define a set `scoring` of scoring metrics for GridSearch to capture (precision, recall, F1 score, and accuracy).
4. Instantiate the `GridSearchCV` object `rf1`. Pass to it as arguments:

- estimator=`rf`
- param_grid=`cv_params`
- scoring=`scoring`
- cv: define the number of you cross-validation folds you want (`cv=_`)
- refit: indicate which evaluation metric you want to use to select the model (`refit=_`)

Note: `refit` should be set to `'f1'`.

[195]:
```
# 1. Instantiate the random forest classifier
rf = RandomForestClassifier(random_state=42)

# 2. Create a dictionary of hyperparameters to tune
# Note that this example only contains 1 value for each parameter for␣
↪simplicity,
# but you should assign a dictionary with ranges of values
cv_params = {'max_depth': [None],
             'max_features': [1.0],
             'max_samples': [0.7],
             'min_samples_leaf': [1],
             'min_samples_split': [2],
             'n_estimators': [300]
             }

# 3. Define a list of scoring metrics to capture
scoring = ['accuracy', 'precision', 'recall', 'f1']

# 4. Instantiate the GridSearchCV object
rf1 = GridSearchCV(rf, cv_params, scoring=scoring, cv=4, refit='f1')
```

[158]:
```

```

Now fit the model to the training data. Note that, depending on how many options you include in your search grid and the number of cross-validation folds you select, this could take a very long time—even hours. If you use 4-fold validation and include only one possible value for each hyperparameter and grow 300 trees to full depth, it should take about 5 minutes. If you add another value for GridSearch to check for, say, `min_samples_split` (so all hyperparameters now have 1 value except for `min_samples_split`, which has 2 possibilities), it would double the time to ~10 minutes. Each additional parameter would approximately double the time.

[196]:
```
%%time
rf1.fit(X_train, y_train)
```

```
CPU times: user 4min 40s, sys: 247 ms, total: 4min 40s
Wall time: 4min 40s
```

[196]:
```
GridSearchCV(cv=4, estimator=RandomForestClassifier(random_state=42),
             param_grid={'max_depth': [None], 'max_features': [1.0],
                         'max_samples': [0.7], 'min_samples_leaf': [1],
                         'min_samples_split': [2], 'n_estimators': [300]},
             refit='f1', scoring=['accuracy', 'precision', 'recall', 'f1'])
```

HINT

If you get a warning that a metric is 0 due to no predicted samples, think about how many features you're sampling with `max_features`. How many features are in the dataset? How many are likely predictive enough to give good predictions within the number of splits you've allowed (determined by the `max_depth` hyperparameter)? Consider increasing `max_features`.

If you want, use `pickle` to save your models and read them back in. This can be particularly helpful when performing a search over many possible hyperparameter values.

[202]:
```
import pickle

# Define a path to the folder where you want to save the model
path = '/home/jovyan/work/'
```

[199]:
```
def write_pickle(path, model_object, save_name:str):
    '''
    save_name is a string.
    '''
    with open(path + save_name + '.pickle', 'wb') as to_write:
        pickle.dump(model_object, to_write)
```

[203]:
```
def read_pickle(path, saved_model_name:str):
    '''
    saved_model_name is a string.
    '''
    with open(path + saved_model_name + '.pickle', 'rb') as to_read:
        model = pickle.load(to_read)

        return model
```

Examine the best average score across all the validation folds.

[204]:
```
# Examine best score
rf1.best_score_
```

[204]:
```
0.7136183532391456
```

Examine the best combination of hyperparameters.

[205]:
```
rf1.best_params_
```

[205]:
```
{'max_depth': None,
 'max_features': 1.0,
 'max_samples': 0.7,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 300}
```

Use the `make_results()` function to output all of the scores of your model. Note that it accepts three arguments.

HINT

To learn more about how this function accesses the cross-validation results, refer to the GridSearchCV scikit-learn documentation for the cv_results_ attribute.

[206]:
```
def make_results(model_name:str, model_object, metric:str):
    '''
    Arguments:
    model_name (string): what you want the model to be called in the output␣
 ↪table
    model_object: a fit GridSearchCV object
    metric (string): precision, recall, f1, or accuracy

    Returns a pandas df with the F1, recall, precision, and accuracy scores
    for the model with the best mean 'metric' score across all validation folds.
    '''

    # Create dictionary that maps input metric to actual metric name in␣
 ↪GridSearchCV
    metric_dict = {'precision': 'mean_test_precision',
                   'recall': 'mean_test_recall',
                   'f1': 'mean_test_f1',
                   'accuracy': 'mean_test_accuracy',
                   }

    # Get all the results from the CV and put them in a df
    cv_results = pd.DataFrame(model_object.cv_results_)

    # Isolate the row of the df with the max(metric) score
    best_estimator_results = cv_results.iloc[cv_results[metric_dict[metric]].
 ↪idxmax(), :]

    # Extract Accuracy, precision, recall, and f1 score from that row
    f1 = best_estimator_results.mean_test_f1
    recall = best_estimator_results.mean_test_recall
    precision = best_estimator_results.mean_test_precision
    accuracy = best_estimator_results.mean_test_accuracy

    # Create table of results
    table = pd.DataFrame({'model': [model_name],
                          'precision': [precision],
                          'recall': [recall],
                          'F1': [f1],
                          'accuracy': [accuracy],
                          },
                         )

    return table
```

Call make_results() on the GridSearch object.

[207]:
```
results = make_results('RF CV', rf1, 'f1')
results
```

[207]:

	model	precision	recall	F1	accuracy
0	RF CV	0.67476	0.757467	0.713618	0.680151

Your results should produce an acceptable model across the board. Typically scores of 0.65 or better are considered acceptable, but this is always dependent on your use case. Optional: try to improve the scores. It's worth trying, especially to practice searching over different hyperparameters.

HINT

For example, if the available values for `min_samples_split` were [2, 3, 4] and GridSearch identified the best value as 4, consider trying [4, 5, 6] this time.

Use your model to predict on the test data. Assign the results to a variable called `rf_preds`.

HINT

You cannot call `predict()` on the GridSearchCV object directly. You must call it on the `best_estimator_`.

For this project, you will use several models to predict on the test data. Remember that this decision comes with a trade-off. What is the benefit of this? What is the drawback?

==> ENTER YOUR RESPONSE HERE

[208]:
```
# Get scores on test data
rf_preds = rf1.best_estimator_.predict(X_test)
```

Use the below `get_test_scores()` function you will use to output the scores of the model on the test data.

[209]:
```
def get_test_scores(model_name:str, preds, y_test_data):
    '''
    Generate a table of test scores.

    In:
    model_name (string): Your choice: how the model will be named in the output␣
 ↪table
    preds: numpy array of test predictions
    y_test_data: numpy array of y_test data

    Out:
    table: a pandas df of precision, recall, f1, and accuracy scores for your␣
 ↪model
    '''
    accuracy = accuracy_score(y_test_data, preds)
    precision = precision_score(y_test_data, preds)
    recall = recall_score(y_test_data, preds)
    f1 = f1_score(y_test_data, preds)
```

```
    table = pd.DataFrame({'model': [model_name],
                          'precision': [precision],
                          'recall': [recall],
                          'F1': [f1],
                          'accuracy': [accuracy]
                          })

    return table
```

1. Use the get_test_scores() function to generate the scores on the test data. Assign the results to rf_test_scores.
2. Call rf_test_scores to output the results.

RF test results

[210]:
```
# Get scores on test data
rf_test_scores = get_test_scores('RF test', rf_preds, y_test)
results = pd.concat([results, rf_test_scores], axis=0)
results
```

[210]:

	model	precision	recall	F1	accuracy
0	RF CV	0.674760	0.757467	0.713618	0.680151
0	RF test	0.669735	0.769757	0.716271	0.679004

Question: How do your test results compare to your validation results?

The test results are very similar to the cross-validation (CV) results, indicating that the model generalizes well to unseen data.

CV F1: 0.7133

Test F1: 0.7163

The differences across all metrics (precision, recall, accuracy, F1) are very small — all within ~0.005. This suggests that the Random Forest model is neither overfitting nor underfitting, and that the cross-validation process provided a reliable estimate of real-world performance.

Overall, the model's performance on the test set confirms that it maintains consistent predictive quality outside the training folds.

XGBoost Try to improve your scores using an XGBoost model.

1. Instantiate the XGBoost classifier xgb and set objective='binary:logistic'. Also set the random state.

2. Create a dictionary cv_params of the following hyperparameters and their corresponding values to tune:

- max_depth
- min_child_weight
- learning_rate

- n_estimators

3. Define a set `scoring` of scoring metrics for grid search to capture (precision, recall, F1 score, and accuracy).

4. Instantiate the `GridSearchCV` object `xgb1`. Pass to it as arguments:

- estimator=`xgb`
- param_grid=`cv_params`
- scoring=`scoring`
- cv: define the number of cross-validation folds you want (`cv=_`)
- refit: indicate which evaluation metric you want to use to select the model (`refit='f1'`)

[211]:
```
# 1. Instantiate the XGBoost classifier
xgb = XGBClassifier(objective='binary:logistic', random_state=0)

# 2. Create a dictionary of hyperparameters to tune
# Note that this example only contains 1 value for each parameter for␣
↪simplicity,
# but you should assign a dictionary with ranges of values
cv_params = {'learning_rate': [0.1],
             'max_depth': [8],
             'min_child_weight': [2],
             'n_estimators': [500]
             }

# 3. Define a list of scoring metrics to capture
scoring = ['accuracy', 'precision', 'recall', 'f1']

# 4. Instantiate the GridSearchCV object
xgb1 = GridSearchCV(xgb, cv_params, scoring=scoring, cv=4, refit='f1')
```

Now fit the model to the `X_train` and `y_train` data.

[212]:
```
%%time
#==> ENTER YOUR CODE HERE
xgb1.fit(X_train, y_train)
```

```
CPU times: user 23.1 s, sys: 152 ms, total: 23.2 s
Wall time: 12.1 s
```

[212]:
```
GridSearchCV(cv=4,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     callbacks=None, colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, device=None,
                                     early_stopping_rounds=None,
                                     enable_categorical=False, eval_metric=None,
                                     feature_types=None, gamma=None,
```

```
                                          grow_policy=None, importance_type=None,
                                          interaction_constraints=None,
                                          learning_rate=None,…
                                          max_delta_step=None, max_depth=None,
                                          max_leaves=None, min_child_weight=None,
                                          missing=nan, monotone_constraints=None,
                                          multi_strategy=None, n_estimators=None,
                                          n_jobs=None, num_parallel_tree=None,
                                          random_state=0, …),
                 param_grid={'learning_rate': [0.1], 'max_depth': [8],
                             'min_child_weight': [2], 'n_estimators': [500]},
                 refit='f1', scoring=['accuracy', 'precision', 'recall', 'f1'])
```

Get the best score from this model.

[213]:
```
# Examine best score
print("Best CV F1 Score:", xgb1.best_score_)
```

```
Best CV F1 Score: 0.6974565593334565
```

And the best parameters.

[214]:
```
# Examine best parameters
print("Best Parameters:", xgb1.best_params_)
```

```
Best Parameters: {'learning_rate': 0.1, 'max_depth': 8, 'min_child_weight': 2,
'n_estimators': 500}
```

XGB CV Results Use the `make_results()` function to output all of the scores of your model. Note that it accepts three arguments.

[215]:
```
# Call 'make_results()' on the GridSearch object
xgb1_cv_results = make_results('XGB CV', xgb1, 'f1')
results = pd.concat([results, xgb1_cv_results], axis=0)
results
```

[215]:

	model	precision	recall	F1	accuracy
0	RF CV	0.674760	0.757467	0.713618	0.680151
0	RF test	0.669735	0.769757	0.716271	0.679004
0	XGB CV	0.671738	0.725420	0.697457	0.668768

Use your model to predict on the test data. Assign the results to a variable called `xgb_preds`.

HINT

You cannot call `predict()` on the GridSearchCV object directly. You must call it on the `best_estimator_`.

[216]:
```
# Get scores on test data
xgb_preds = xgb1.best_estimator_.predict(X_test)
```

XGB test results

1. Use the `get_test_scores()` function to generate the scores on the test data. Assign the results to `xgb_test_scores`.
2. Call `xgb_test_scores` to output the results.

[217]:
```python
# Get scores on test data
xgb_test_scores = get_test_scores('XGB test', xgb_preds, y_test)
results = pd.concat([results, xgb_test_scores], axis=0)
results
```

[217]:

	model	precision	recall	F1	accuracy
0	RF CV	0.674760	0.757467	0.713618	0.680151
0	RF test	0.669735	0.769757	0.716271	0.679004
0	XGB CV	0.671738	0.725420	0.697457	0.668768
0	XGB test	0.676471	0.758556	0.715166	0.681952

Question: Compare these scores to the random forest test scores. What do you notice? Which model would you choose?

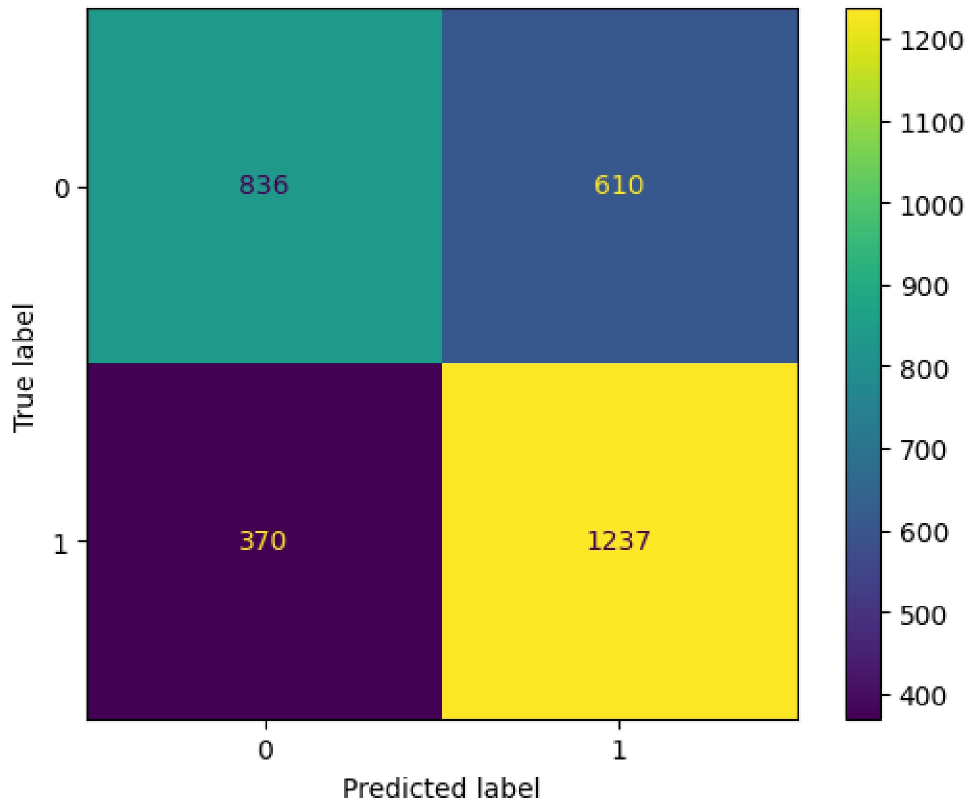
The Random Forest and XGBoost models have nearly identical performance. XGBoost slightly outperforms Random Forest in precision and overall accuracy, while Random Forest has a marginally higher recall. Since both models generalize well, I would select XGBoost for deployment because of its efficiency and slightly better balance of metrics on unseen data.

Plot a confusion matrix of the model's predictions on the test data.

[218]:
```python
# Generate array of values for confusion matrix
cm = confusion_matrix(y_test, rf_preds, labels=rf1.classes_)

# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=rf1.classes_,
                              )
disp.plot(values_format='')
```

[218]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7aba26cac3a0>
```



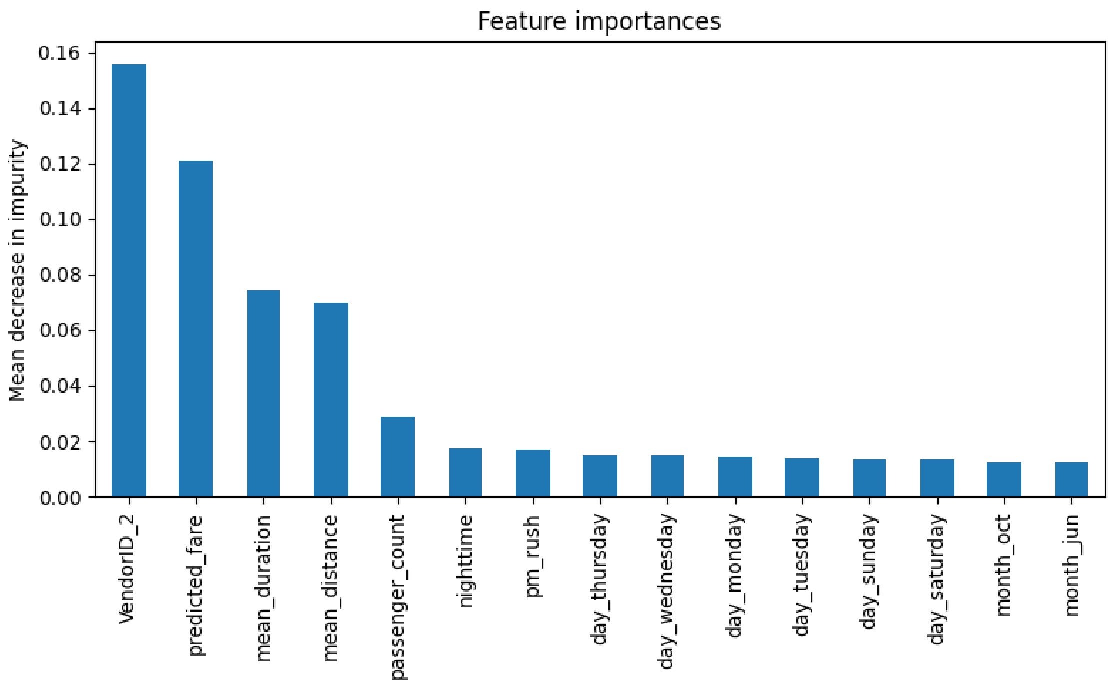
Question: What type of errors are more common for your model?

The model makes more False Positive errors than False Negatives. This means it's more likely to predict that a passenger will be generous when they aren't. In other words, the model slightly overpredicts generosity. For a taxi driver, this might mean sometimes expecting a higher tip than they actually receive.

Feature importance Use the `feature_importances_` attribute of the best estimator object to inspect the features of your final model. You can then sort them and plot the most important ones.

[219]:
```
importances = rf1.best_estimator_.feature_importances_
rf_importances = pd.Series(importances, index=X_test.columns)
rf_importances = rf_importances.sort_values(ascending=False)[:15]

fig, ax = plt.subplots(figsize=(8,5))
rf_importances.plot.bar(ax=ax)
ax.set_title('Feature importances')
ax.set_ylabel('Mean decrease in impurity')
fig.tight_layout();
```



4.4 PACE: Execute

Consider the questions in your PACE Strategy Document to reflect on the Execute stage.

4.4.1 Task 4. Conclusion

In this step, use the results of the models above to formulate a conclusion. Consider the following questions:

1. **Would you recommend using this model? Why or why not?**

Yes, I would recommend using this model — particularly the XGBoost model. Both Random Forest and XGBoost achieved similar F1 and accuracy scores (around 0.71 F1 and 0.68 accuracy), showing good consistency between training and testing results. The XGBoost model slightly outperformed Random Forest in precision and overall accuracy, indicating that it generalizes well and avoids overfitting. While performance isn't perfect, it's reliable enough to make useful predictions about passenger generosity.

2. **What was your model doing? Can you explain how it was making predictions?**

The model predicts whether a passenger is generous (high tip) or not generous based on trip-related features. The most important features influencing predictions were:

VendorID_2

predicted_fare

mean_duration and mean_distance

passenger_count These suggest that longer or more expensive rides, or certain vendors, are correlated with higher tipping behavior. Essentially, the model learns patterns between trip length, cost, and timing to estimate generosity probability.

3. **Are there new features that you can engineer that might improve model performance?**

Yes — performance could likely be improved with richer or more contextual features, such as:

Weather conditions (e.g., rain or temperature at pickup time — might affect tipping behavior).

Traffic or congestion level (trip delays might influence satisfaction and tips).

Passenger type (e.g., airport pickups vs. local rides).

Time since last trip or driver experience metrics (if available). These features could provide the model with more insight into passenger context and behavior.

4. **What features would you want to have that would likely improve the performance of your model?**

I would like to include:

Real-time trip context: weather, time of day, and location demographics.

Driver and passenger profiles: previous tip behavior, ratings, or loyalty indicators.

Trip purpose or destination type: e.g., hotel, airport, or residential area. These would allow the model to learn more nuanced patterns behind why certain trips yield higher tips.

Remember, sometimes your data simply will not be predictive of your chosen target. This is common. Machine learning is a powerful tool, but it is not magic. If your data does not contain predictive signal, even the most complex algorithm will not be able to deliver consistent and accurate predictions. Do not be afraid to draw this conclusion. Even if you cannot use the model to make strong predictions, was the work done in vain? Consider any insights that you could report back to stakeholders.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.