Machine Learning in the Clouds (no, not that Cloud)

A MSDS 696 Final Capstone Project

by

Jeremy Beard

This Jupyter Notebook was made to take in a database of flights and their respective metadata and prices, scraped from the web via an API. This scraping of flight data was performed in a different file, flight-price-api.py. Once the database is created, machine learning model comparative analytics was used in order to determine the best model for this specific dataset and use case.

The markdown documentation in this Jupyter Notebook serves as a general commentary on what is happening programmatically in the notebook. For a broader overview of the project and different elements involved, please see the GitHub repository link.

The GitHub repository link is: https://github.com/jeremyabeard5/MSDS696

Let's begin!

In [1]:
```
################################################
# This python script is designed to take the flight data from
# the sqlite3 database or .csv created in flight-price-api.py and create a model to predict the price
# of flights for any given day, airline and route.

# Thanks!
# Jeremy Beard
################################################
```

First Steps

Of course, let's import packages first

In [2]:

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import os
import time
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

from sklearn import metrics
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score

from pycaret.regression import *
#from pycaret.time_series import *

from datetime import datetime
```

Set some constant values now.

In [3]:

```
dpi = 300
data_dir = "data"
flights_dir = "flight-price-api"
file_path = os.path.join(data_dir, flights_dir)
```

The GetValues function below is a simple function that spits out the values for a specific field in the 'df' dataframe.

In [4]:

```
def GetValues(str):
    print(f"Value counts for {str} column:")
    print(df[str].value_counts())
    print()
```

Connect to Database, Parse Dates

The main theme of my project is focused on comparative analytics to find the best model. Beyond comparing the performance of models, I also wanted to compare specific different formats of datetime.

Through some research, I really wanted to get the date handling correct and wasn't sure which format would perform best. I researched online and included a few of the most helpful articles below and throughout the notebook. What I finally decided on were the following:

Parse datetime field into:

- Day, Month, Year, and Day of Week as integers
- Ordinal datetime (int version of day/month/year, doesn't account for hour)
- time.mktime() datetime (float version of hour/day/month/year)

Beyond that, there can be a lot more options to consider. Stuff like:

- Creating a boolean "is_holiday" field or similar
- Similar booleans for "is_weekday", "is_weekend", etc.

And more! But these were not explored in this notebook

In [5]:
```
# Connect to database
conn = sqlite3.connect(os.path.join(data_dir, flights_dir, "flights.db"))

# Create dataframe from database
df = pd.read_sql_query("SELECT * FROM flights", conn)
```

In [6]:
```
# Create various columns from the datetimes!
df['deptDateTime'] = pd.to_datetime(df['deptDateTime'])
df['arrvDateTime'] = pd.to_datetime(df['arrvDateTime'])
df['deptDayOfWeek'] = df['deptDateTime'].dt.day_of_week
df['arrvDayOfWeek'] = df['arrvDateTime'].dt.day_of_week
df['deptMonth'] = df['deptDateTime'].dt.month
df['deptDate'] = df['deptDateTime'].dt.date
df['deptDate'] = pd.to_datetime(df['deptDate'])
df['arrvDate'] = df['arrvDateTime'].dt.date
df['arrvDate'] = pd.to_datetime(df['arrvDate'])
df['deptDayOfYear'] = df['deptDateTime'].dt.dayofyear
df['deptHour'] = df['deptDateTime'].dt.hour
df['deptDayOfMonth'] = df['deptDateTime'].dt.day
```

Throughout this whole project, I wanted to make sure I got the date-handling correct.

Using references:

- https://stackoverflow.com/questions/40217369/python-linear-regression-predict-by-date

In [7]:
```
print(df['deptDateTime'])
```

```
0       2023-11-20 20:46:00
```

```
1       2023-11-20 09:40:00
2       2023-11-20 05:55:00
3       2023-11-20 17:50:00
4       2023-11-20 09:42:00
               ...
44946   2024-10-08 12:35:00
44947   2024-10-09 16:45:00
44948   2024-10-09 09:10:00
44949   2024-10-10 16:45:00
44950   2024-10-10 09:10:00
Name: deptDateTime, Length: 44951, dtype: datetime64[ns]
```

In [8]:
```
print(df['deptDateTime'][0])
```

```
2023-11-20 20:46:00
```

In [9]:
```
print(type(df['deptDateTime'][0]))
```

```
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

Ordinal Datetime and time.mktime() Datetime Exploration

Should we use toordinal()?

In [10]:
```
def getordinal(x):
    return x.toordinal()

def getmktime(x):
    return time.mktime(x.timetuple())
```

In [11]:
```
print(df['deptDateTime'][0].toordinal())
df['deptDateTimeOrd'] = df['deptDateTime'].apply(getordinal)
```

```
738844
```

In [12]:
```
for d8 in range(15):
    print(df['deptDateTime'][d8].toordinal())
```

```
738844
738844
738844
738844
738844
738844
738844
738844
```

```
738844
738844
738845
738845
738845
738845
```

Looks like it doesn't account for the time of day, so we'd need to have that as another field.

In [13]:
```
print(type(df['deptDateTime'][0].toordinal()))
```

```
<class 'int'>
```

Or should we use time.mktime()? time.mktime accounts for the Hour/Minute/Second

In [14]:
```
df['deptDateTimeMktime'] = df['deptDateTime'].apply(getmktime)
```

In [15]:
```
print(time.mktime(df['deptDateTime'][0].timetuple()))
```

```
1700538360.0
```

In [16]:
```
for d8 in range(15):
    print(time.mktime(df['deptDateTime'][d8].timetuple()))
```

```
1700538360.0
1700498400.0
1700484900.0
1700527800.0
1700498520.0
1700481600.0
1700490300.0
1700485200.0
1700523360.0
1700549940.0
1700622060.0
1700614200.0
1700584920.0
1700568000.0
1700571600.0
```

In [17]:
```
print(type(time.mktime(df['deptDateTime'][0].timetuple())))
```

```
<class 'float'>
```

And should we include day of week as well as a field? Hmm...

Intro Exploration

Let's get the info, head, value_counts, all the preliminary exploratory items!

In [18]:
```
print("OG Dataframe info:")
print(df.info())
```

```
OG Dataframe info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44951 entries, 0 to 44950
Data columns (total 26 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   destCode            44951 non-null  object
 1   destName            44951 non-null  object
 2   orgCode             44951 non-null  object
 3   orgName             44951 non-null  object
 4   deptDateTime        44951 non-null  datetime64[ns]
 5   arrvDateTime        44951 non-null  datetime64[ns]
 6   price               44951 non-null  float64
 7   currency            44951 non-null  object
 8   marktName           44951 non-null  object
 9   marktCode           44951 non-null  object
 10  optName             44951 non-null  object
 11  optCode             44951 non-null  object
 12  flightNum           44951 non-null  object
 13  duration            44951 non-null  float64
 14  numConex            44951 non-null  int64
 15  url                 44951 non-null  object
 16  deptDayOfWeek       44951 non-null  int64
 17  arrvDayOfWeek       44951 non-null  int64
 18  deptMonth           44951 non-null  int64
 19  deptDate            44951 non-null  datetime64[ns]
 20  arrvDate            44951 non-null  datetime64[ns]
 21  deptDayOfYear       44951 non-null  int64
 22  deptHour            44951 non-null  int64
 23  deptDayOfMonth      44951 non-null  int64
 24  deptDateTimeOrd     44951 non-null  int64
 25  deptDateTimeMktime  44951 non-null  float64
dtypes: datetime64[ns](4), float64(3), int64(8), object(11)
memory usage: 8.9+ MB
None
```

In [19]:
```
df.head()
```

Out[19]:

	destCode	destName	orgCode	orgName	deptDateTime	arrvDateTime	price	currency	marktName	marktCode	...	deptDayOfWeek	arrvDayOfWeek

	destCode	destName	orgCode	orgName	deptDateTime	arrvDateTime	price	currency	marktName	marktCode	...	deptDayOfWeek	arrvDayOfWeek
0	CVG	Cincinnati	DEN	Denver	2023-11-20 20:46:00	2023-11-21 01:29:00	55.29	USD	Frontier Airlines	F9	...	0	1
1	CVG	Cincinnati	DEN	Denver	2023-11-20 09:40:00	2023-11-20 14:15:00	95.16	USD	Allegiant Air	G4	...	0	0
2	CVG	Cincinnati	DEN	Denver	2023-11-20 05:55:00	2023-11-20 10:38:00	96.45	USD	Frontier Airlines	F9	...	0	0
3	CVG	Cincinnati	DEN	Denver	2023-11-20 17:50:00	2023-11-20 22:37:00	138.59	USD	United Airlines	UA	...	0	0
4	CVG	Cincinnati	DEN	Denver	2023-11-20 09:42:00	2023-11-20 14:29:00	186.79	USD	United Airlines	UA	...	0	0

In [20]:
```
# We should have NO null values
df.isnull().sum()
```

Out[20]:
```
destCode              0
destName              0
orgCode               0
orgName               0
deptDateTime          0
arrvDateTime          0
price                 0
currency              0
marktName             0
marktCode             0
optName               0
optCode               0
flightNum             0
duration              0
numConex              0
url                   0
deptDayOfWeek         0
arrvDayOfWeek         0
deptMonth             0
deptDate              0
arrvDate              0
deptDayOfYear         0
deptHour              0
deptDayOfMonth        0
deptDateTimeOrd       0
deptDateTimeMktime    0
dtype: int64
```

Everything looking pretty clean! I guess whoever scraped the data did it in a pretty good way (heh heh...)

Now you'll see the use of the 'GetValues' function mentioned above, basically just a consolidated and clean 'value_counts'

Marketing Airline Name

In [21]:
```
# Get value counts for a lot of columns (next few cells)
GetValues('marktName')
```

```
Value counts for marktName column:
United Airlines      28139
American Airlines    12149
Frontier Airlines     2436
Sun Country            656
Air Canada             521
Spirit Airlines        455
Allegiant Air          141
Aeromexico             117
Flair Airlines          53
Alaska Airlines         51
Volaris                 50
WestJet                 48
Lynx Air                47
Key Lime Air            35
Silver Airways          12
Breeze Airways          12
Porter Airlines          7
Canada Jetlines          7
Air Transat              6
Lufthansa                4
LATAM                    3
Air France               1
Icelandair               1
Name: marktName, dtype: int64
```

Marketing Airline Code

In [22]:
```
GetValues('marktCode')
```

```
Value counts for marktCode column:
UA    28139
AA    12149
F9     2436
SY      656
AC      521
NK      455
G4      141
AM      117
F8       53
```

```
AS       51
Y4       50
WS       48
Y9       47
KG       35
3M       12
MX       12
PD        7
AU        7
TS        6
LH        4
LA        3
AF        1
FI        1
Name: marktCode, dtype: int64
```

Operating Airline Name

In [23]:

```
GetValues('optName')
```

```
Value counts for optName column:
United Airlines                      28139
American Airlines                    10474
Frontier Airlines                     2436
Skywest Airlines As American Eagle    1254
Sun Country                            656
Air Canada                             521
Spirit Airlines                        455
Envoy Air As American Eagle            279
Allegiant Air                          141
Aeromexico                             117
Air Wisconsin As American Eagle        105
Psa Airlines As American Eagle          71
Flair Airlines                          53
Volaris                                 50
WestJet                                 48
Lynx Air                                47
Key Lime Air                            35
Republic Airways As American Eagle      17
Silver Airways                          12
Breeze Airways                          12
Canada Jetlines                          7
Porter Airlines                          7
Air Transat                              6
Lufthansa                                3
LATAM                                    3
Discover                                 1
Icelandair                               1
```

```
Air France                          1
Name: optName, dtype: int64
```

Operating Airline Code

In [24]:

```
GetValues('optCode')
```

```
Value counts for optCode column:
UA    28139
AA    12200
F9     2436
SY      656
AC      521
NK      455
G4      141
AM      117
F8       53
Y4       50
WS       48
Y9       47
KG       35
3M       12
MX       12
PD        7
AU        7
TS        6
LA        3
LH        3
AF        1
4Y        1
FI        1
Name: optCode, dtype: int64
```

Origin Location Name

In [25]:

```
GetValues('orgName')
```

```
Value counts for orgName column:
Denver           22337
New York          3260
Los Angeles       3260
Chicago           3260
San Francisco     3258
Miami             3238
Cincinnati        3224
Orlando           3114
Name: orgName, dtype: int64
```

Origin Location Code

In [26]:
```
GetValues('orgCode')
```

```
Value counts for orgCode column:
DEN    22337
NYC     3260
LAX     3260
CHI     3260
SFO     3258
MIA     3238
CVG     3224
ORL     3114
Name: orgCode, dtype: int64
```

Destination Location Name

In [27]:
```
GetValues('destName')
```

```
Value counts for destName column:
Denver           22614
Chicago           3260
Los Angeles       3260
New York          3260
San Francisco     3259
Cincinnati        3242
Miami             3238
Orlando           2818
Name: destName, dtype: int64
```

Destination Location Code

In [28]:
```
GetValues('destCode')
```

```
Value counts for destCode column:
DEN    22614
CHI     3260
LAX     3260
NYC     3260
SFO     3259
CVG     3242
MIA     3238
ORL     2818
Name: destCode, dtype: int64
```

Departure Month

In [29]:
```
GetValues('deptMonth')
```

```
Value counts for deptMonth column:
12    4337
3     4319
1     4304
7     4272
5     4251
8     4248
4     4157
6     4128
9     4083
2     4037
11    1540
10    1275
Name: deptMonth, dtype: int64
```

Departure Hour

In [30]:
```
GetValues('deptHour')
```

```
Value counts for deptHour column:
17    3543
8     3425
6     3249
16    3154
7     3136
15    2949
12    2926
5     2910
14    2541
9     2389
10    2331
11    2321
13    2258
18    1983
19    1875
20    1416
21    1315
23     538
0      530
22     159
1        3
Name: deptHour, dtype: int64
```

Departure Day of Month

In [31]:
```
GetValues('deptDayOfMonth')
```

```
Value counts for deptDayOfMonth column:
21    1533
25    1531
2     1529
4     1525
22    1523
28    1523
20    1522
6     1522
23    1521
1     1521
27    1519
7     1518
5     1518
29    1517
26    1516
24    1516
3     1509
8     1506
9     1486
10    1470
19    1391
16    1387
18    1385
17    1383
11    1381
12    1379
14    1378
30    1377
13    1371
15    1368
31     826
Name: deptDayOfMonth, dtype: int64
```

Currency

If there is only USD, we can definitely remove this feature

In [32]:
```
GetValues('currency')
```

```
Value counts for currency column:
USD    44951
```

```
Name: currency, dtype: int64
```

In [33]:

```
# I saw that the only difference between MARKETING airline and OPERATING airline resided with American Airlines
# So I will drop the optName and optCode columns
# Drop unnecessary columns. We don't care about the codes, about the operating name
df = df.drop(columns=['destCode', 'orgCode', 'optCode', 'marktCode', 'optName'])

# If the currency value counts length is 1 (only USD), then drop the column
if len(df['currency'].value_counts()) == 1:
    print()
    print("Only USD observed in currency column. Dropping column.")
    df = df.drop(columns=['currency'])
    print()
```

```
Only USD observed in currency column. Dropping column.
```

Let's see the data again now...

In [34]:

```
print("Dataframe head:")
print(df.head())
```

```
Dataframe head:
     destName orgName        deptDateTime        arrvDateTime   price  \
0  Cincinnati  Denver 2023-11-20 20:46:00 2023-11-21 01:29:00   55.29
1  Cincinnati  Denver 2023-11-20 09:40:00 2023-11-20 14:15:00   95.16
2  Cincinnati  Denver 2023-11-20 05:55:00 2023-11-20 10:38:00   96.45
3  Cincinnati  Denver 2023-11-20 17:50:00 2023-11-20 22:37:00  138.59
4  Cincinnati  Denver 2023-11-20 09:42:00 2023-11-20 14:29:00  186.79

           marktName flightNum  duration  numConex  \
0  Frontier Airlines    F94128     163.0         1
1      Allegiant Air     G4406     155.0         1
2  Frontier Airlines    F93286     163.0         1
3    United Airlines    UA1830     167.0         1
4    United Airlines    UA1978     167.0         1

                                                 url  deptDayOfWeek  \
0  https://flights-us.gotogate.com/air/DENCVG20NO...              0
1  https://flights-us.gotogate.com/air/DENCVG20NO...              0
2  https://flights-us.gotogate.com/air/DENCVG20NO...              0
3  https://flights-us.gotogate.com/air/DENCVG20NO...              0
4  https://flights-us.gotogate.com/air/DENCVG20NO...              0

   arrvDayOfWeek  deptMonth   deptDate   arrvDate  deptDayOfYear  deptHour  \
0              1         11 2023-11-20 2023-11-21            324        20
1              0         11 2023-11-20 2023-11-20            324         9
2              0         11 2023-11-20 2023-11-20            324         5
```

```
3              0         11 2023-11-20 2023-11-20            324        17
4              0         11 2023-11-20 2023-11-20            324         9

   deptDayOfMonth  deptDateTimeOrd  deptDateTimeMktime
0              20           738844        1.700538e+09
1              20           738844        1.700498e+09
2              20           738844        1.700485e+09
3              20           738844        1.700528e+09
4              20           738844        1.700499e+09
```

In [35]:

```
print("Dataframe tail:")
print(df.tail())
```

```
Dataframe tail:
      destName     orgName        deptDateTime        arrvDateTime   price  \
44946   Denver  Cincinnati 2024-10-08 12:35:00 2024-10-08 17:41:00  598.20
44947   Denver  Cincinnati 2024-10-09 16:45:00 2024-10-09 17:59:00  170.90
44948   Denver  Cincinnati 2024-10-09 09:10:00 2024-10-09 10:15:00  190.80
44949   Denver  Cincinnati 2024-10-10 16:45:00 2024-10-10 17:59:00  247.70
44950   Denver  Cincinnati 2024-10-10 09:10:00 2024-10-10 10:15:00  251.03

               marktName flightNum  duration  numConex  \
44946  American Airlines    AA1568     426.0         3
44947    United Airlines     UA479     194.0         1
44948    United Airlines     UA497     185.0         1
44949    United Airlines     UA479     194.0         1
44950    United Airlines     UA497     185.0         1

                                                    url  deptDayOfWeek  \
44946  https://flights-us.gotogate.com/air/CVGDEN08OC...              1
44947  https://flights-us.gotogate.com/air/CVGDEN09OC...              2
44948  https://flights-us.gotogate.com/air/CVGDEN09OC...              2
44949  https://flights-us.gotogate.com/air/CVGDEN10OC...              3
44950  https://flights-us.gotogate.com/air/CVGDEN10OC...              3

       arrvDayOfWeek  deptMonth   deptDate   arrvDate  deptDayOfYear  \
44946              1         10 2024-10-08 2024-10-08            282
44947              2         10 2024-10-09 2024-10-09            283
44948              2         10 2024-10-09 2024-10-09            283
44949              3         10 2024-10-10 2024-10-10            284
44950              3         10 2024-10-10 2024-10-10            284

       deptHour  deptDayOfMonth  deptDateTimeOrd  deptDateTimeMktime
44946        12               8           739167        1.728412e+09
44947        16               9           739168        1.728514e+09
44948         9               9           739168        1.728487e+09
44949        16              10           739169        1.728600e+09
44950         9              10           739169        1.728573e+09
```

In [36]:

```
print("Dataframe describe:")
print(df.describe())
```

```
Dataframe describe:
              price      duration      numConex  deptDayOfWeek  arrvDayOfWeek  \
count  44951.000000  44951.000000  44951.000000   44951.000000   44951.000000
mean     208.218587    265.057685      1.457787       2.989500       2.972837
std      107.934430    217.025827      0.878563       2.000328       2.000288
min       24.810000    131.000000      1.000000       0.000000       0.000000
25%      138.150000    157.000000      1.000000       1.000000       1.000000
50%      190.970000    217.000000      1.000000       3.000000       3.000000
75%      243.195000    276.000000      1.000000       5.000000       5.000000
max     3057.870000   2147.000000      7.000000       6.000000       6.000000

          deptMonth  deptDayOfYear      deptHour  deptDayOfMonth  \
count  44951.000000   44951.000000  44951.000000    44951.000000
mean       6.014438     168.136949     12.276412       15.769950
std        3.351740     102.513778      4.988419        8.912191
min        1.000000       1.000000      0.000000        1.000000
25%        3.000000      81.000000      8.000000        8.000000
50%        6.000000     163.000000     12.000000       16.000000
75%        8.000000     244.000000     16.000000       24.000000
max       12.000000     365.000000     23.000000       31.000000

       deptDateTimeOrd  deptDateTimeMktime
count     44951.000000        4.495100e+04
mean     739005.415986        1.714453e+09
std          93.901422        8.111833e+06
min      738844.000000        1.700465e+09
25%      738924.000000        1.707426e+09
50%      739005.000000        1.714411e+09
75%      739087.000000        1.721478e+09
max      739169.000000        1.728625e+09
```

In [37]:

```
print("Dataframe shape:")
print(df.shape)
```

```
Dataframe shape:
(44951, 20)
```

In [38]:

```
print("Dataframe columns:")
print(df.columns)
```

```
Dataframe columns:
Index(['destName', 'orgName', 'deptDateTime', 'arrvDateTime', 'price',
       'marktName', 'flightNum', 'duration', 'numConex', 'url',
       'deptDayOfWeek', 'arrvDayOfWeek', 'deptMonth', 'deptDate', 'arrvDate',
```

```
       'deptDayOfYear', 'deptHour', 'deptDayOfMonth', 'deptDateTimeOrd',
       'deptDateTimeMktime'],
```

In [39]:

```
print("Few Columns of Interest: ")
print(df.loc[:, ['orgName', 'destName', 'deptDateTime', 'price']])
```

```
Few Columns of Interest:
          orgName    destName        deptDateTime   price
0          Denver  Cincinnati 2023-11-20 20:46:00   55.29
1          Denver  Cincinnati 2023-11-20 09:40:00   95.16
2          Denver  Cincinnati 2023-11-20 05:55:00   96.45
3          Denver  Cincinnati 2023-11-20 17:50:00  138.59
4          Denver  Cincinnati 2023-11-20 09:42:00  186.79
...           ...         ...                 ...     ...
44946  Cincinnati      Denver 2024-10-08 12:35:00  598.20
44947  Cincinnati      Denver 2024-10-09 16:45:00  170.90
44948  Cincinnati      Denver 2024-10-09 09:10:00  190.80
44949  Cincinnati      Denver 2024-10-10 16:45:00  247.70
44950  Cincinnati      Denver 2024-10-10 09:10:00  251.03

[44951 rows x 4 columns]
```

In [40]:

```
print("NEW Dataframe info:")
print(df.info())
```

```
NEW Dataframe info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44951 entries, 0 to 44950
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   destName            44951 non-null  object
 1   orgName             44951 non-null  object
 2   deptDateTime        44951 non-null  datetime64[ns]
 3   arrvDateTime        44951 non-null  datetime64[ns]
 4   price               44951 non-null  float64
 5   marktName           44951 non-null  object
 6   flightNum           44951 non-null  object
 7   duration            44951 non-null  float64
 8   numConex            44951 non-null  int64
 9   url                 44951 non-null  object
 10  deptDayOfWeek       44951 non-null  int64
 11  arrvDayOfWeek       44951 non-null  int64
 12  deptMonth           44951 non-null  int64
 13  deptDate            44951 non-null  datetime64[ns]
 14  arrvDate            44951 non-null  datetime64[ns]
 15  deptDayOfYear       44951 non-null  int64
 16  deptHour            44951 non-null  int64
```

```
 17  deptDayOfMonth      44951 non-null  int64
 18  deptDateTimeOrd     44951 non-null  int64
 19  deptDateTimeMktime  44951 non-null  float64
dtypes: datetime64[ns](4), float64(3), int64(8), object(5)
memory usage: 6.9+ MB
```

Initial Plot Creation

In [41]:

```
plt.style.use('dark_background')
# https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html
```

In [42]:

```
print("Creating first plot: Price vs. Date")

# Set the font size for all text on the plot
plt.rc('font', size=100)

# Set the font size for the title
#plt.rc('axes', titlesize=15)

# Set the font size for the x and y labels
#plt.rc('axes', labelsize=15)

# Set the font size for the x and y tick labels
#plt.rc('xtick', labelsize=15)
#plt.rc('ytick', labelsize=15)

fig0, ax0 = plt.subplots(figsize=(12,6))
df.plot.scatter(x='deptDateTime', y='price', c='cyan', ax=ax0, alpha=0.5, s=25, linewidths=2)
ax0.set(xlabel='datetime', ylabel='price', title='price by datetime')
plt.tight_layout()
plt.grid(False)
plt.show()
```

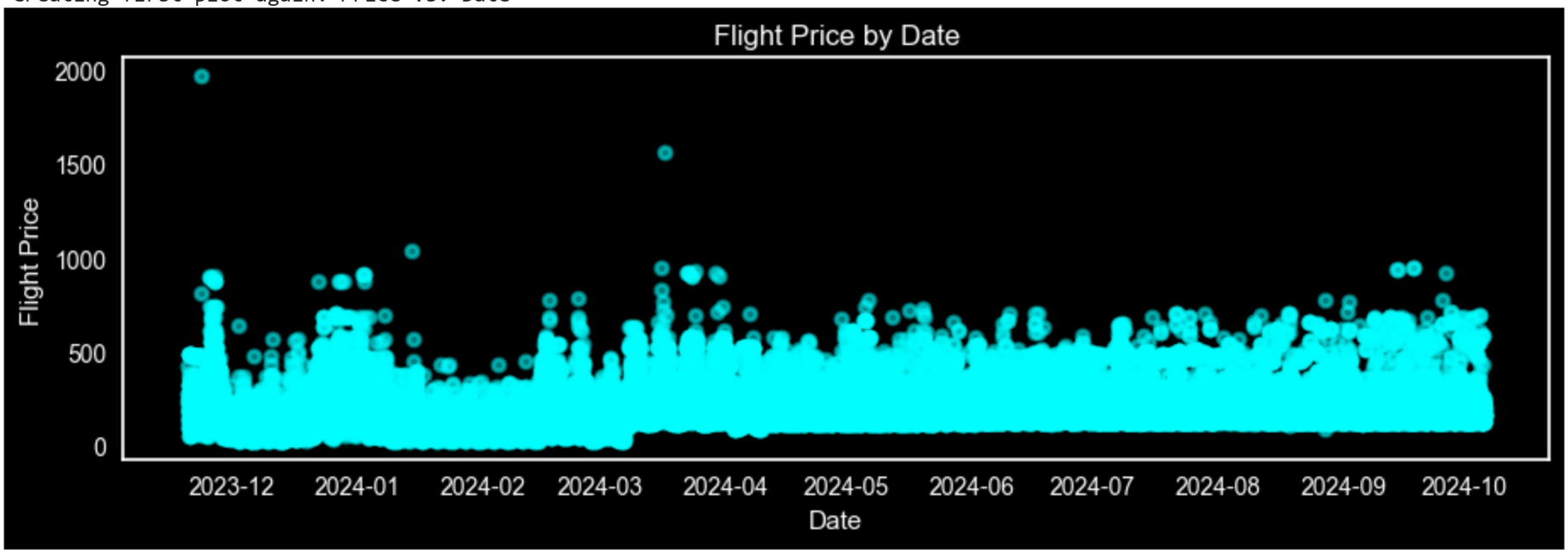
```
Creating first plot: Price vs. Date
```

Why are the October 2024 prices so high? Let's just remove October 2024, maybe that's too far out

In [43]:
```
# Create mask
mask = (df['deptDateTime'].dt.month == 10) & (df['deptDateTime'].dt.year == 2024) & (df['deptDateTime'].dt.day > 5)
# Use the mask to drop the rows
df = df[~mask]
```

In [44]:
```
print("Creating first plot again: Price vs. Date")
fig0, ax0 = plt.subplots(figsize=(12,6))
plt.rc('font', size=100)
df.plot.scatter(x='deptDateTime', y='price', c='cyan', ax=ax0, alpha=0.5, s=25, linewidths=2)
ax0.set(xlabel='Date', ylabel='Flight Price', title='Flight Price by Date')
plt.tight_layout()
plt.grid(False)
#plt.show()
plt.savefig('output/00-price-by-datetime-scatter.png', dpi=dpi)
```

```
Creating first plot again: Price vs. Date
```



Let's now explore the data by charting avg. price vs. day of week

In [45]:
```
# Find mean price by dayOfWeek
df_dow = df.groupby('deptDayOfWeek')
mean_dow = df_dow.mean()
mean_dow = mean_dow.reset_index()
```

In [46]:
```
mean_dow.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   deptDayOfWeek       7 non-null      int64  
 1   price               7 non-null      float64
 2   duration            7 non-null      float64
 3   numConex            7 non-null      float64
 4   arrvDayOfWeek       7 non-null      float64
 5   deptMonth           7 non-null      float64
 6   deptDayOfYear       7 non-null      float64
 7   deptHour            7 non-null      float64
 8   deptDayOfMonth      7 non-null      float64
 9   deptDateTimeOrd     7 non-null      float64
 10  deptDateTimeMktime  7 non-null      float64
dtypes: float64(10), int64(1)
memory usage: 744.0 bytes
```

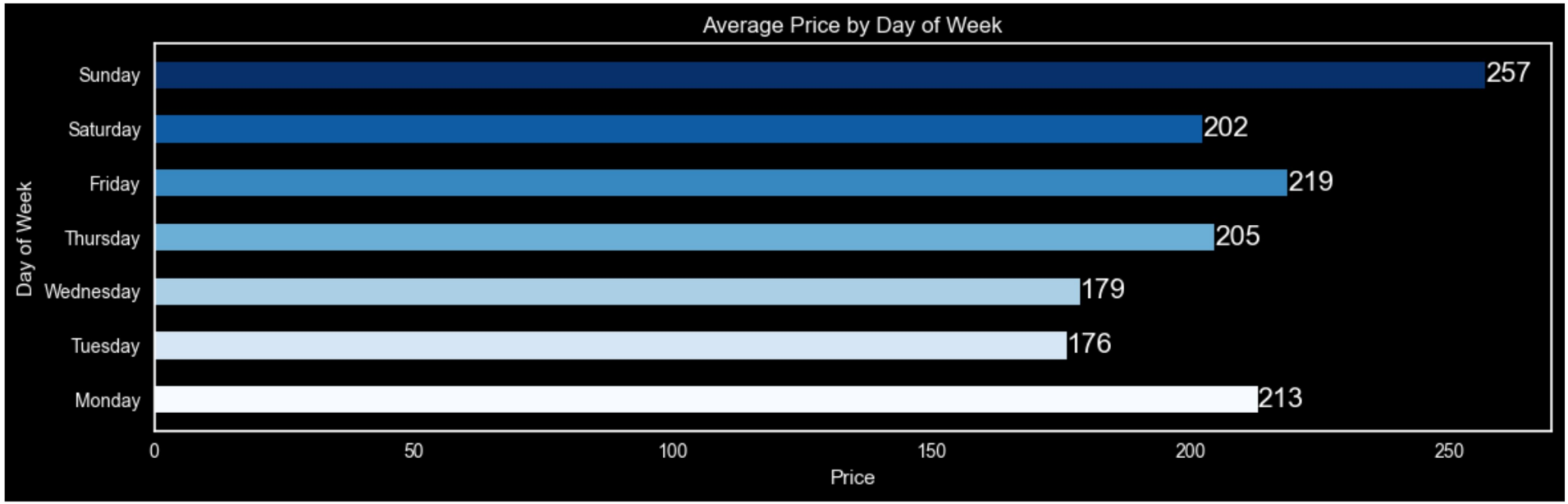
In [47]:
```
# Monday=0, Sunday=6
# https://pandas.pydata.org/docs/reference/api/pandas.DatetimeIndex.dayofweek.html
mean_dow.head(10)
```

Out[47]:

	deptDayOfWeek	price	duration	numConex	arrvDayOfWeek	deptMonth	deptDayOfYear	deptHour	deptDayOfMonth	deptDateTimeOrd	deptD
0	0	212.924779	256.592184	1.438386	0.061456	5.876458	163.674441	12.587457	15.497321	739000.360542	
1	1	176.138508	259.597088	1.465580	1.055389	5.916759	165.218547	12.295616	15.826555	739001.813736	
2	2	178.614870	261.592020	1.461126	2.057089	5.948589	166.533512	12.287809	16.180098	739003.181517	
3	3	204.576618	271.651090	1.472009	3.066959	5.970989	166.892426	12.043751	15.831425	739003.813392	
4	4	218.672087	265.117297	1.451623	4.065391	6.018504	168.000941	12.405363	15.534107	739004.979144	
5	5	202.238069	257.950473	1.445426	5.055521	6.032808	168.764353	12.270505	15.851262	739005.404732	
6	6	256.776815	285.350566	1.484761	5.519866	5.969363	167.397958	12.034147	16.410882	739003.475028	

In [48]:

```
fig1, ax1 = plt.subplots(figsize=(12,4))
plt.rc('font', size=15)
mean_dow_tmp = mean_dow.drop(columns=['duration', 'numConex', 'arrvDayOfWeek', 'deptMonth'])
daysOfWeek = ['','Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# Create a colormap
comap = plt.cm.get_cmap('Blues', len(mean_dow_tmp['deptDayOfWeek']))

# Assign colors to bars
colors = comap(np.arange(len(mean_dow_tmp['deptDayOfWeek'])))

bars = ax1.barh(mean_dow_tmp['deptDayOfWeek'], mean_dow_tmp['price'], color=colors, height=0.5)
ax1.set(ylabel='Day of Week', xlabel='Price', title='Average Price by Day of Week')
ax1.set_yticklabels(daysOfWeek)
ax1.bar_label(bars, fmt='{:,.0f}', fontsize=15) # https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.bar_label.html
plt.tight_layout()
plt.grid(False)
#plt.show()
plt.savefig('output/01-avg-price-by-dow.png', dpi=dpi)
```



Let's now explore the data by charting avg. price vs. month

In [49]:

```
# Find mean price by month
df_mon = df.groupby('deptMonth')
mean_mon = df_mon.mean()
mean_mon = mean_mon.reset_index()
mean_mon.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   deptMonth           12 non-null     int64
 1   price               12 non-null     float64
 2   duration            12 non-null     float64
 3   numConex            12 non-null     float64
 4   deptDayOfWeek       12 non-null     float64
 5   arrvDayOfWeek       12 non-null     float64
 6   deptDayOfYear       12 non-null     float64
 7   deptHour            12 non-null     float64
 8   deptDayOfMonth      12 non-null     float64
 9   deptDateTimeOrd     12 non-null     float64
 10  deptDateTimeMktime  12 non-null     float64
dtypes: float64(10), int64(1)
memory usage: 1.2 KB
```

In [50]:

```
# 1=January, 2=February, etc.
# https://pandas.pydata.org/docs/reference/api/pandas.Series.dt.month.html
mean_mon.head(12)
```

Out[50]:

	deptMonth	price	duration	numConex	deptDayOfWeek	arrvDayOfWeek	deptDayOfYear	deptHour	deptDayOfMonth	deptDateTimeOrd	deptI
0	1	143.680895	257.714219	1.379647	2.813662	2.810409	15.969331	11.876162	15.969331	738900.969331	
1	2	151.246061	240.919742	1.345306	3.006440	2.992569	46.028982	12.144166	15.028982	738931.028982	
2	3	233.228226	237.239176	1.295439	3.192406	3.158138	76.004399	12.523038	16.004399	738961.004399	
3	4	207.390678	268.108732	1.510945	2.841713	2.810440	106.453933	12.422661	15.453933	738991.453933	
4	5	216.352733	269.085156	1.544343	3.007763	2.984945	136.999530	12.330981	15.999530	739021.999530	
5	6	249.855082	279.130329	1.543605	3.186289	3.151405	167.521318	12.349079	15.521318	739052.521318	
6	7	230.331498	270.625000	1.536985	2.802903	2.793305	198.002107	12.267088	16.002107	739083.002107	
7	8	227.684463	271.605461	1.534369	3.107580	3.100282	228.967750	12.271422	15.967750	739113.967750	
8	9	226.023595	260.462895	1.518246	3.008327	2.998530	259.502327	12.264266	15.502327	739144.502327	
9	10	215.398234	269.205499	1.538350	2.985528	3.037627	276.985528	12.247467	2.985528	739161.985528	
10	11	241.538747	296.972727	1.448052	2.454545	2.476623	329.000000	12.298701	25.000000	738849.000000	
11	12	171.298949	285.991238	1.385520	3.194374	3.175928	350.003689	12.299055	16.003689	738870.003689	

In [51]:

```
fig2, ax2 = plt.subplots(figsize=(12,6))
plt.rc('font', size=100)
mean_mon_tmp = mean_mon.drop(columns=['duration', 'numConex', 'deptDayOfWeek', 'arrvDayOfWeek'])
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']

# Create a colormap
comap2 = plt.cm.get_cmap('Blues', len(mean_mon_tmp['deptMonth']))

# Assign colors to bars
colors2 = comap2(np.arange(len(mean_mon_tmp['deptMonth'])))

bars = ax2.barh(mean_mon_tmp['deptMonth'], mean_mon_tmp['price'], color=colors2, height=0.5)
ax2.set(ylabel='Month', xlabel='Price', title='Average Price by Month')
ax2.set_yticks(mean_mon_tmp['deptMonth'])
ax2.set_yticklabels(months)
ax2.bar_label(bars, fmt='{:,.0f}', fontsize=10)
plt.tight_layout()
plt.grid(False)
#plt.show()
plt.savefig('output/02-avg-price-by-month.png', dpi=dpi)
```

Let's now explore the data by charting average flight price vs. day of year

In [52]:

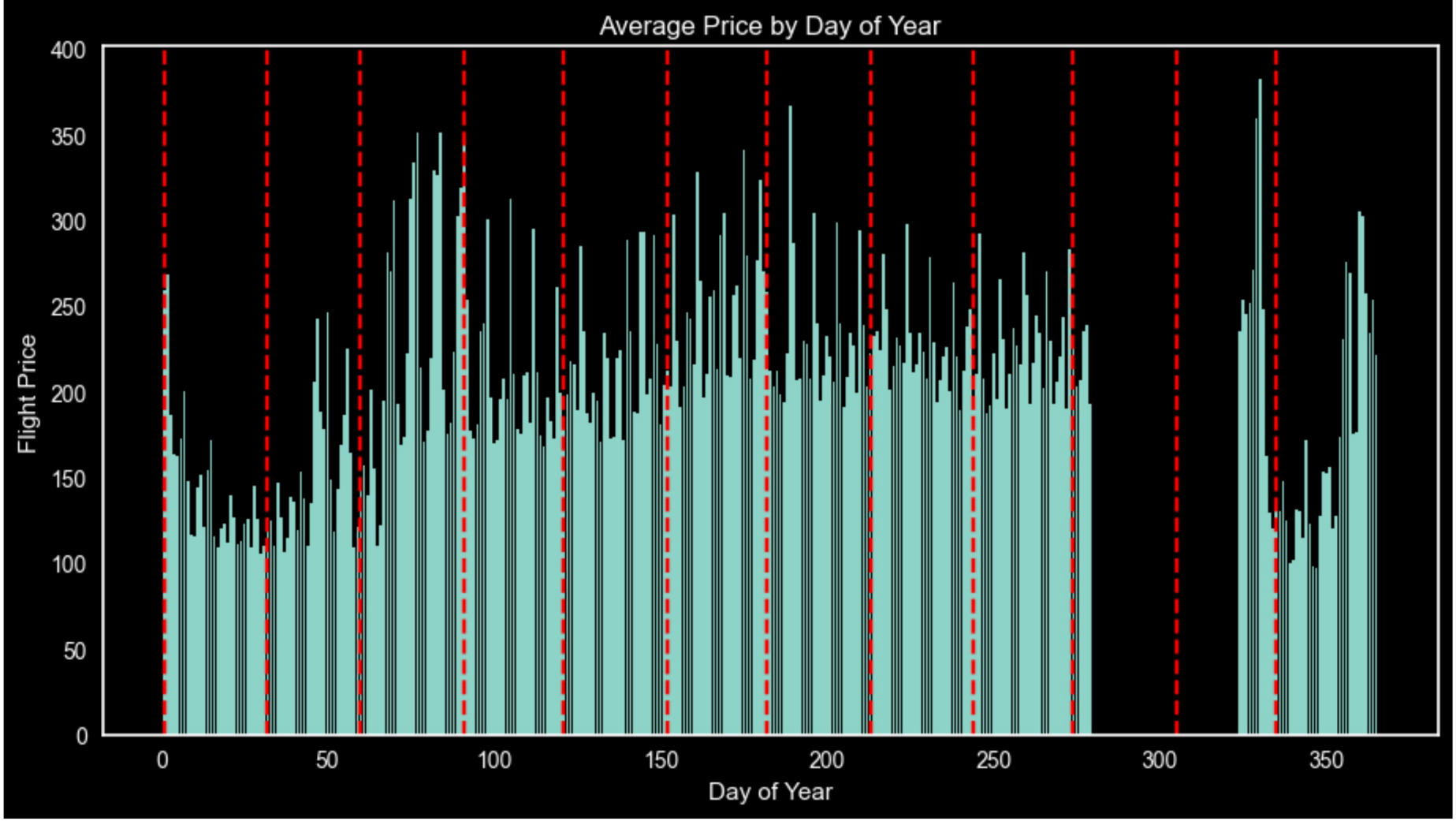
```
df_doy = df.groupby('deptDayOfYear')['price'].mean()
df_doy = df_doy.reset_index()
df_doy.head(10)
```

Out[52]:

	deptDayOfYear	price
0	1	259.394786
1	2	267.975071
2	3	186.024357
3	4	162.957786
4	5	162.622071
5	6	172.204143
6	7	199.856857
7	8	147.448143
8	9	116.072230
9	10	115.745522

In [53]:

```
fig3, ax3 = plt.subplots(figsize=(12,8))
plt.rc('font', size=100)
bars = ax3.bar(df_doy['deptDayOfYear'], df_doy['price'])
ax3.set(ylabel='Flight Price', xlabel='Day of Year', title='Average Price by Day of Year')

# Add vertical lines for the months ( https://stackoverflow.com/questions/71419004/how-to-plot-vertical-lines-at-specific-dates-in-m
month_days = [1, 32, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335]
for day in month_days:
   ax3.axvline(x=day, color='red', linestyle='--')

plt.tight_layout()
plt.grid(False)
#plt.show()
plt.savefig('output/03-avg-price-by-doy.png', dpi=dpi)
```



Calendar Plot Creation

I'd like to create the same chart above, but in calendar heatmap form beacuse those look really nice!

https://medium.com/analytics-vidhya/calendar-heatmaps-a-perfect-way-to-display-your-time-series-quantitative-data-ad36bf81a3ed

https://calplot.readthedocs.io/en/latest/index.html

https://matplotlib.org/stable/users/explain/colors/colormaps.html

Let's begin!

In [54]:

```
df_cal = df.copy()
df_cal.head()
```

Out[54]:

	destName	orgName	deptDateTime	arrvDateTime	price	marktName	flightNum	duration	numConex	url	deptDayOfWeek	arrvDayOf
0	Cincinnati	Denver	2023-11-20 20:46:00	2023-11-21 01:29:00	55.29	Frontier Airlines	F94128	163.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
1	Cincinnati	Denver	2023-11-20 09:40:00	2023-11-20 14:15:00	95.16	Allegiant Air	G4406	155.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
2	Cincinnati	Denver	2023-11-20 05:55:00	2023-11-20 10:38:00	96.45	Frontier Airlines	F93286	163.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
3	Cincinnati	Denver	2023-11-20 17:50:00	2023-11-20 22:37:00	138.59	United Airlines	UA1830	167.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
4	Cincinnati	Denver	2023-11-20 09:42:00	2023-11-20 14:29:00	186.79	United Airlines	UA1978	167.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	

In [55]:
```
df_cal = df_cal[['deptDate', 'price']]
df_cal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   deptDate  44367 non-null  datetime64[ns]
 1   price     44367 non-null  float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 1.0 MB
```

In [56]:
```
df_cal['deptDate'] = pd.to_datetime(df_cal['deptDate'])
```

In [57]:
```
df_cal = df_cal.groupby('deptDate').mean()
df_cal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 321 entries, 2023-11-20 to 2024-10-05
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   price   321 non-null    float64
dtypes: float64(1)
```

In [58]:
```
df_cal.head(10)
```

Out[58]:

deptDate	price
2023-11-20	234.874857
2023-11-21	253.916000
2023-11-22	245.188571
2023-11-23	251.482571
2023-11-24	270.612571
2023-11-25	358.938071
2023-11-26	382.092929
2023-11-27	247.859071
2023-11-28	162.324714
2023-11-29	129.147786

In [59]:
```
df_cal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 321 entries, 2023-11-20 to 2024-10-05
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   price   321 non-null    float64
dtypes: float64(1)
memory usage: 5.0 KB
```

In [60]:
```
import calplot
print(type(df_cal.index))
# The index should be a DatetimeIndex for calplot
```

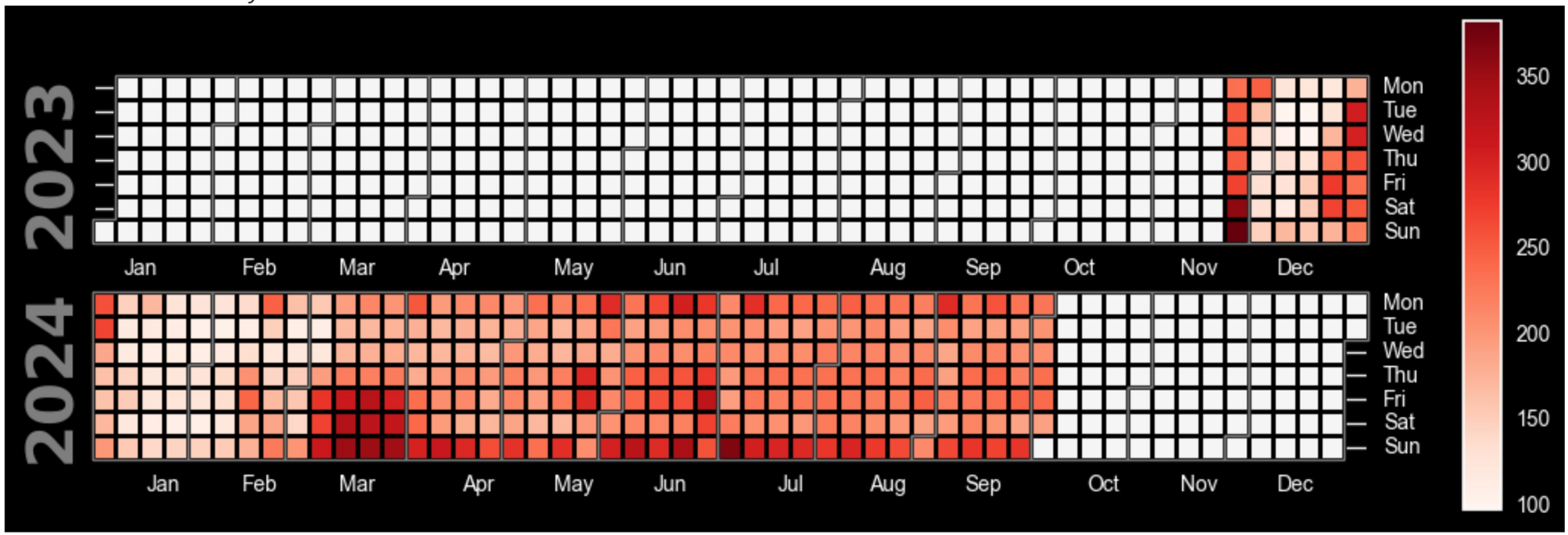
```
<class 'pandas.core.indexes.datetimes.DatetimeIndex'>
```

In [61]:
```
calplot.calplot(df_cal['price'], cmap='Reds', colorbar=True, textformat=None) # textformat='{:.0f}')
```

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```

Out[61]:
```
(<Figure size 1250x340 with 3 Axes>,
 array([<Axes: ylabel='2023'>, <Axes: ylabel='2024'>], dtype=object))
```

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



What about the calendar plot using the july package?

https://medium.com/analytics-vidhya/calendar-heatmaps-a-perfect-way-to-display-your-time-series-quantitative-data-ad36bf81a3ed

https://pypi.org/project/july/

In [62]:
```
!pip install july
```

```
Requirement already satisfied: july in c:\programdata\anaconda3\lib\site-packages (0.1.3)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (from july) (3.4.3)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from july) (1.20.3)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->july) (8.4.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->july) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->july) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->july) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->july) (2.8.2)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib->july) (1.16.0)
```

In [63]:

```
import july
df_jul = df.copy()
df_jul = df_jul[['deptDate', 'price']]
df_jul.head(15)
```

Out[63]:

	deptDate	price
0	2023-11-20	55.29
1	2023-11-20	95.16
2	2023-11-20	96.45
3	2023-11-20	138.59
4	2023-11-20	186.79
5	2023-11-20	141.53
6	2023-11-20	141.53
7	2023-11-20	141.53
8	2023-11-20	192.26
9	2023-11-20	195.61
10	2023-11-21	116.52
11	2023-11-21	208.64
12	2023-11-21	255.12
13	2023-11-21	141.45
14	2023-11-21	192.26

In [64]:

```
df_jul.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   deptDate  44367 non-null  datetime64[ns]
 1   price     44367 non-null  float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 1.0 MB
```

In [65]:

```
df_jul['deptDate'] = pd.to_datetime(df_jul['deptDate'])
df_jul = df_jul.groupby('deptDate').mean()
df_jul = df_jul.reset_index()
df_jul.head(15)
```

Out[65]:

	deptDate	price
0	2023-11-20	234.874857
1	2023-11-21	253.916000
2	2023-11-22	245.188571
3	2023-11-23	251.482571
4	2023-11-24	270.612571
5	2023-11-25	358.938071
6	2023-11-26	382.092929
7	2023-11-27	247.859071
8	2023-11-28	162.324714
9	2023-11-29	129.147786
10	2023-11-30	120.489071
11	2023-12-01	130.848357
12	2023-12-02	130.471429
13	2023-12-03	147.656857
14	2023-12-04	124.340071

In [66]:

```
df_jul.tail(15)
```

Out[66]:

	deptDate	price
306	2024-09-21	202.416058
307	2024-09-22	269.919778
308	2024-09-23	229.740821
309	2024-09-24	192.987778
310	2024-09-25	205.891765

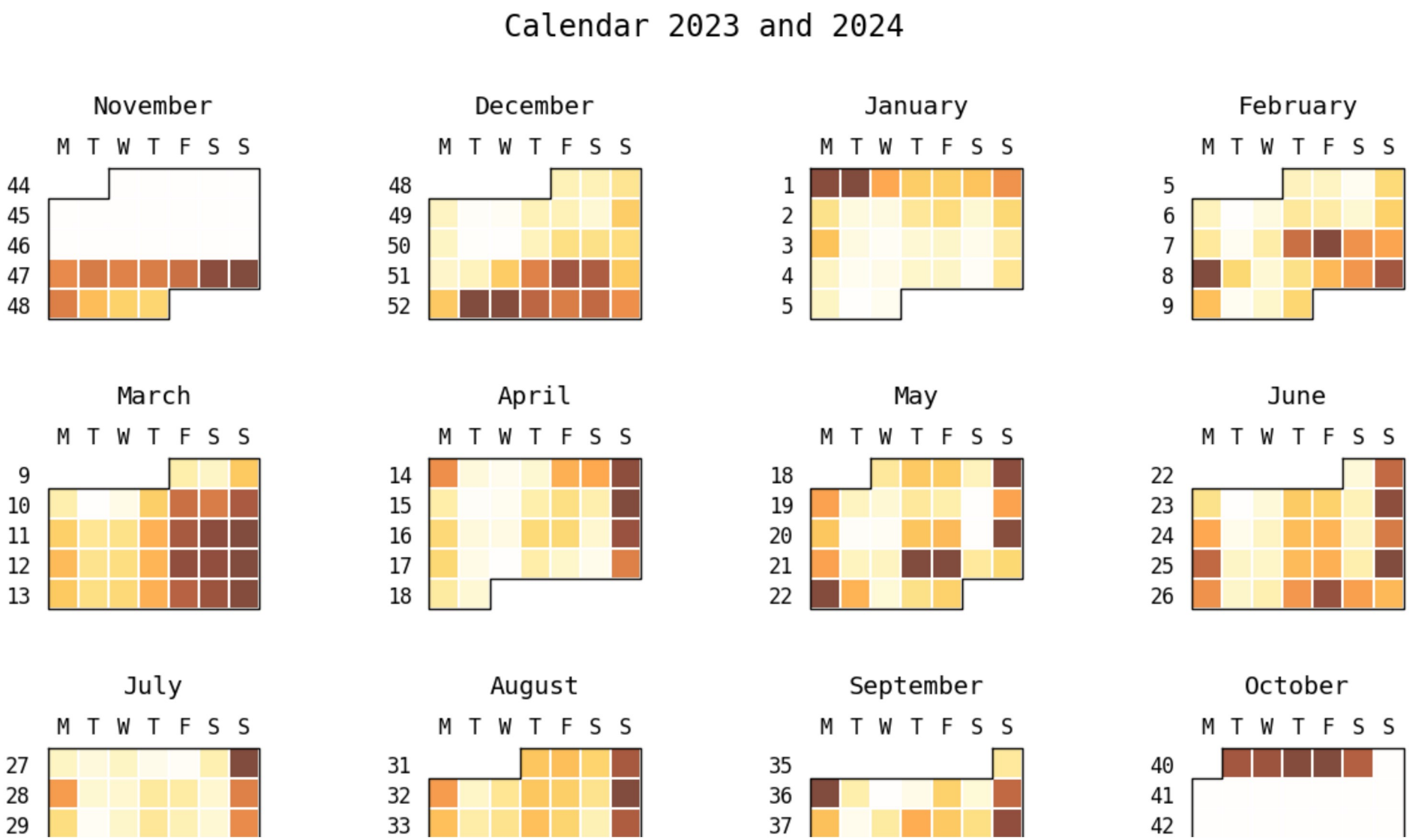
	deptDate	price
311	2024-09-26	220.678788
312	2024-09-27	243.742899
313	2024-09-28	190.180827
314	2024-09-29	282.844599
315	2024-09-30	228.955985
316	2024-10-01	202.868214
317	2024-10-02	206.637464
318	2024-10-03	235.399214
319	2024-10-04	238.909565

In [67]:
```
df_jul.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 321 entries, 0 to 320
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   deptDate  321 non-null    datetime64[ns]
 1   price     321 non-null    float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 5.1 KB
```

In [68]:
```
plt.style.use('default')
july.calendar_plot(df_jul['deptDate'], df_jul['price'], cmap = 'golden')
```

Out[68]:
```
array([[<Axes: title={'center': 'November'}>,
        <Axes: title={'center': 'December'}>,
        <Axes: title={'center': 'January'}>,
        <Axes: title={'center': 'February'}>],
       [<Axes: title={'center': 'March'}>,
        <Axes: title={'center': 'April'}>,
        <Axes: title={'center': 'May'}>,
        <Axes: title={'center': 'June'}>],
       [<Axes: title={'center': 'July'}>,
        <Axes: title={'center': 'August'}>,
        <Axes: title={'center': 'September'}>,
        <Axes: title={'center': 'October'}>]], dtype=object)
```



In [ ]:

More Plot Creation

Let's now explore the data by charting average flight price vs. hour of day

In [69]:

```
# Find mean price by hour
df_hr = df.groupby('deptHour')
mean_hr = df_hr.mean()
mean_hr = mean_hr.reset_index()
mean_hr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   deptHour            21 non-null     int64
 1   price               21 non-null     float64
 2   duration            21 non-null     float64
 3   numConex            21 non-null     float64
 4   deptDayOfWeek       21 non-null     float64
 5   arrvDayOfWeek       21 non-null     float64
 6   deptMonth           21 non-null     float64
 7   deptDayOfYear       21 non-null     float64
 8   deptDayOfMonth      21 non-null     float64
 9   deptDateTimeOrd     21 non-null     float64
 10  deptDateTimeMktime  21 non-null     float64
dtypes: float64(10), int64(1)
memory usage: 1.9 KB
```

In [70]:

```
mean_hr.head(24)
```

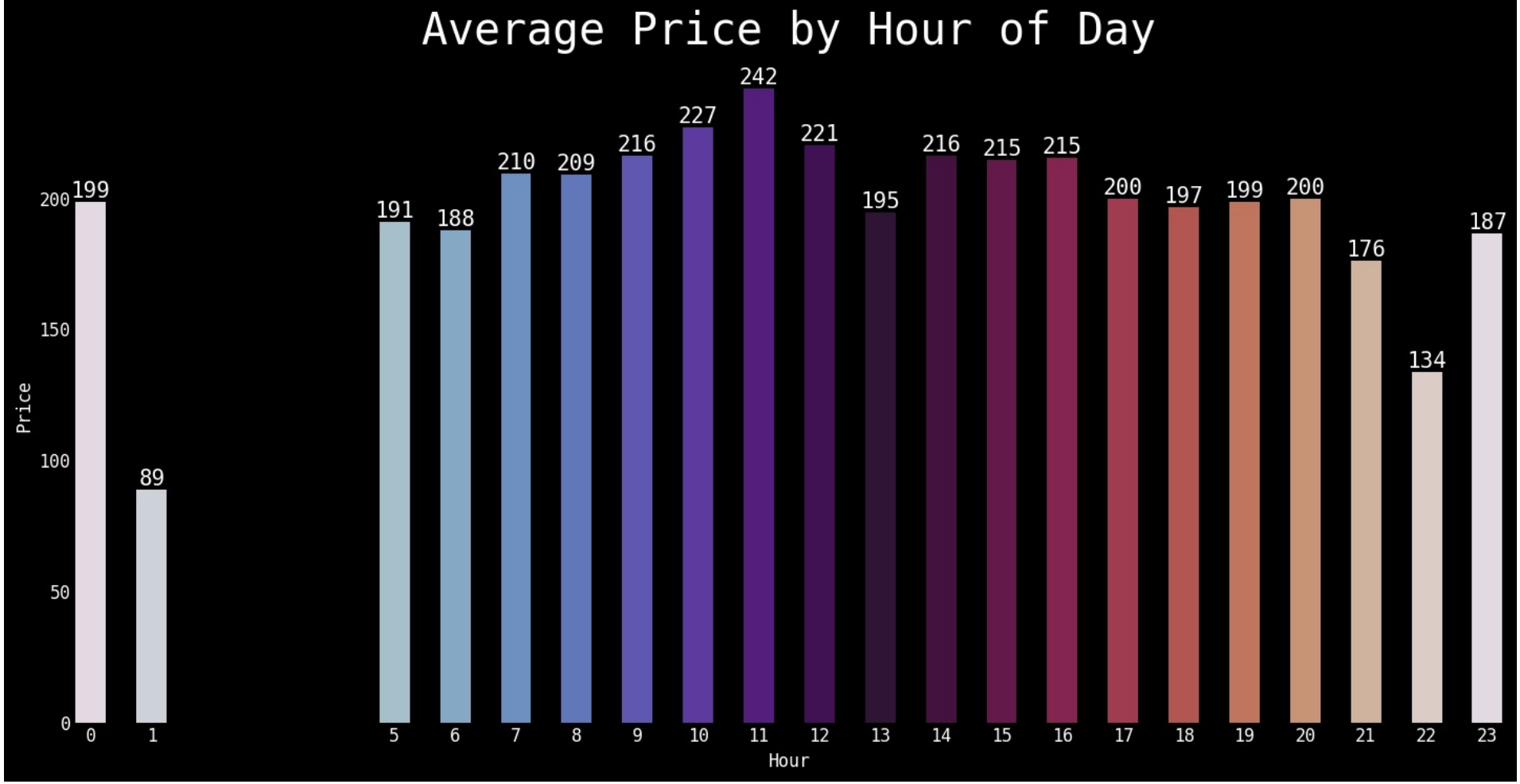
Out[70]:

	deptHour	price	duration	numConex	deptDayOfWeek	arrvDayOfWeek	deptMonth	deptDayOfYear	deptDayOfMonth	deptDateTimeOrd	deptI
0	0	198.537173	225.138520	1.030361	2.963947	2.963947	5.851992	163.481973	16.119545	738968.833017	
1	1	88.750000	225.000000	1.000000	4.000000	4.000000	2.333333	54.000000	13.333333	738939.000000	
2	5	190.912104	253.836174	1.834087	3.074435	3.072696	6.055304	169.376348	15.772174	739008.291130	
3	6	187.704626	247.673737	1.466001	3.019339	3.017779	5.901747	164.856519	15.913911	739000.104492	
4	7	209.693897	266.735380	1.401939	3.116963	3.113409	5.906947	165.029402	15.933764	738996.959935	
5	8	209.051216	243.268307	1.458939	3.116513	3.108509	6.137267	171.941891	15.847317	739005.541061	
6	9	216.392712	235.181239	1.230051	3.070883	3.043718	5.553905	154.486418	16.122666	738998.741511	
7	10	227.043197	330.741685	1.941685	2.945572	2.958963	5.902376	164.593952	15.606911	738991.414687	
8	11	242.124816	288.542832	1.291084	3.107955	3.098776	6.034965	168.941871	15.956731	739010.231206	
9	12	220.527629	285.922811	1.668051	2.913811	2.908619	5.776393	160.902042	15.757009	739013.811353	
10	13	194.674211	256.595878	1.271505	3.057796	2.965054	5.981183	167.250896	15.919803	738996.977599	
11	14	216.344031	284.420737	1.550040	2.871898	2.887110	5.935148	165.940753	15.954363	739013.680945	
12	15	214.837089	261.303467	1.380364	2.844147	2.844834	6.277377	176.085822	15.743563	738995.177824	
13	16	215.389525	271.091784	1.646983	2.897304	2.911425	5.867779	163.614249	15.716303	739015.112965	
14	17	200.045394	219.533123	1.180098	2.957557	2.960138	5.886435	164.388586	15.907083	739009.402925	

	deptHour	price	duration	numConex	deptDayOfWeek	arrvDayOfWeek	deptMonth	deptDayOfYear	deptDayOfMonth	deptDateTimeOrd	deptI
15	18	196.867610	305.011282	1.433846	2.965641	2.929231	6.368205	178.815897	15.718974	739022.823590	
16	19	198.746705	272.585498	1.358225	2.951840	2.843615	5.768939	160.774892	15.866883	739001.137446	
17	20	199.757312	283.546942	1.452347	3.065434	3.001422	6.129445	171.916785	16.121622	738982.151494	
18	21	176.251302	224.207242	1.077042	2.970724	2.976888	5.710324	159.023883	15.911402	739008.030046	
19	22	133.705786	532.937107	2.018868	3.207547	2.798742	8.150943	234.251572	17.100629	738889.691824	

In [71]:

```
plt.style.use('dark_background')
fig4, ax4 = plt.subplots(figsize=(15,8))
plt.rc('font', size=25)
mean_hr_tmp = mean_hr.drop(columns=['duration', 'numConex', 'deptDayOfWeek', 'arrvDayOfWeek', 'deptMonth', 'deptDayOfYear'])

# Create a colormap
comap4 = plt.cm.get_cmap('twilight', len(mean_hr_tmp['deptHour']))

# Assign colors to bars
colors4 = comap4(np.arange(len(mean_hr_tmp['deptHour'])))

bars = ax4.bar(mean_hr_tmp['deptHour'], mean_hr_tmp['price'], color=colors4, width=0.5)
ax4.set(xlabel='Hour', ylabel='Price', title='Average Price by Hour of Day')
ax4.set_xticks(mean_hr_tmp['deptHour'])
ax4.bar_label(bars, fmt='{:,.0f}', fontsize=15)
plt.tight_layout()
ax4.spines['top'].set_visible(False)
ax4.spines['right'].set_visible(False)
ax4.spines['bottom'].set_visible(False)
ax4.spines['left'].set_visible(False)
#plt.show()
plt.savefig('output/04-avg-price-by-hour.png', dpi=dpi)
```



Machine Learning Time!!!

Now let's start with the machine learning models

Let's see, what features and what targets do we want...

From a few articles:

- https://datascience.stackexchange.com/questions/2368/machine-learning-features-engineering-from-date-time-data
- https://datascience.stackexchange.com/questions/112357/feature-engineering-for-datetime-column

It seems like popular features are dayOfWeek, dayOfMonth, dayOfYear, hour, and month. This is what we will use.

Features

- deptDayOfWeek

- deptDayOfMonth
- deptDayOfYear
- deptHour
- deptMonth
- marktName
- orgName
- destName

Targets

- price

ML Model 1: Parsed Datetime Fields

In [72]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   destName            44367 non-null  object
 1   orgName             44367 non-null  object
 2   deptDateTime        44367 non-null  datetime64[ns]
 3   arrvDateTime        44367 non-null  datetime64[ns]
 4   price               44367 non-null  float64
 5   marktName           44367 non-null  object
 6   flightNum           44367 non-null  object
 7   duration            44367 non-null  float64
 8   numConex            44367 non-null  int64
 9   url                 44367 non-null  object
 10  deptDayOfWeek       44367 non-null  int64
 11  arrvDayOfWeek       44367 non-null  int64
 12  deptMonth           44367 non-null  int64
 13  deptDate            44367 non-null  datetime64[ns]
 14  arrvDate            44367 non-null  datetime64[ns]
 15  deptDayOfYear       44367 non-null  int64
 16  deptHour            44367 non-null  int64
 17  deptDayOfMonth      44367 non-null  int64
 18  deptDateTimeOrd     44367 non-null  int64
 19  deptDateTimeMktime  44367 non-null  float64
dtypes: datetime64[ns](4), float64(3), int64(8), object(5)
memory usage: 7.1+ MB
```

In [73]:
```
df.head(15)
```

Out[73]:

	destName	orgName	deptDateTime	arrvDateTime	price	marktName	flightNum	duration	numConex	url	deptDayOfWeek	arrvDay(
0	Cincinnati	Denver	2023-11-20 20:46:00	2023-11-21 01:29:00	55.29	Frontier Airlines	F94128	163.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
1	Cincinnati	Denver	2023-11-20 09:40:00	2023-11-20 14:15:00	95.16	Allegiant Air	G4406	155.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
2	Cincinnati	Denver	2023-11-20 05:55:00	2023-11-20 10:38:00	96.45	Frontier Airlines	F93286	163.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
3	Cincinnati	Denver	2023-11-20 17:50:00	2023-11-20 22:37:00	138.59	United Airlines	UA1830	167.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
4	Cincinnati	Denver	2023-11-20 09:42:00	2023-11-20 14:29:00	186.79	United Airlines	UA1978	167.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
5	Cincinnati	Denver	2023-11-20 05:00:00	2023-11-20 12:27:00	141.53	American Airlines	AA2612	327.0	3	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
6	Cincinnati	Denver	2023-11-20 07:25:00	2023-11-20 15:54:00	141.53	American Airlines	AA852	389.0	3	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
7	Cincinnati	Denver	2023-11-20 06:00:00	2023-11-20 15:54:00	141.53	American Airlines	AA1067	474.0	3	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
8	Cincinnati	Denver	2023-11-20 16:36:00	2023-11-20 23:59:00	192.26	American Airlines	AA1832	323.0	3	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
9	Cincinnati	Denver	2023-11-20 23:59:00	2023-11-21 12:29:00	195.61	Spirit Airlines	NK1320	630.0	3	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
10	Cincinnati	Denver	2023-11-21 20:01:00	2023-11-22 00:44:00	116.52	Frontier Airlines	F94128	163.0	1	https://flights-us.gotogate.com/air/DENCVG21NO...	1	
11	Cincinnati	Denver	2023-11-21 17:50:00	2023-11-21 22:37:00	208.64	United Airlines	UA1830	167.0	1	https://flights-us.gotogate.com/air/DENCVG21NO...	1	
12	Cincinnati	Denver	2023-11-21 09:42:00	2023-11-21 14:29:00	255.12	United Airlines	UA1978	167.0	1	https://flights-us.gotogate.com/air/DENCVG21NO...	1	

	destName	orgName	deptDateTime	arrvDateTime	price	marktName	flightNum	duration	numConex	url	deptDayOfWeek	arrvDay(
13	Cincinnati	Denver	2023-11-21 05:00:00	2023-11-21 12:27:00	141.45	American Airlines	AA2612	327.0	3	https://flights-us.gotogate.com/air/DENCVG21NO...	1	
14	Cincinnati	Denver	2023-11-21	2023-11-21	192.26	American	AA1067	474.0	3	https://flights-us.gotogate.com	1	

Feature Creation

In [74]:
```
features = df[['deptDayOfWeek', 'deptDayOfMonth', 'deptDayOfYear', 'deptHour', 'deptMonth', 'marktName', 'orgName', 'destName']]
print(features)
```

```
       deptDayOfWeek  deptDayOfMonth  deptDayOfYear  deptHour  deptMonth  \
0                  0              20            324        20         11
1                  0              20            324         9         11
2                  0              20            324         5         11
3                  0              20            324        17         11
4                  0              20            324         9         11
...              ...             ...            ...       ...        ...
44912              5               5            279         7         10
44913              5               5            279        12         10
44914              5               5            279        12         10
44915              5               5            279        20         10
44916              5               5            279        16         10

               marktName     orgName    destName
0      Frontier Airlines      Denver  Cincinnati
1          Allegiant Air      Denver  Cincinnati
2      Frontier Airlines      Denver  Cincinnati
3        United Airlines      Denver  Cincinnati
4        United Airlines      Denver  Cincinnati
...                  ...         ...         ...
44912  American Airlines  Cincinnati      Denver
44913  American Airlines  Cincinnati      Denver
44914  American Airlines  Cincinnati      Denver
44915  American Airlines  Cincinnati      Denver
44916  American Airlines  Cincinnati      Denver

[44367 rows x 8 columns]
```

Target Creation

In [75]:
```
targets = df[['price']]
print(targets)
```

```
       price
0      55.29
```

```
1       95.16
2       96.45
3      138.59
4      186.79
...       ...
44912  192.20
44913  192.20
44914  192.20
44915  192.20
44916  192.20

[44367 rows x 1 columns]
```

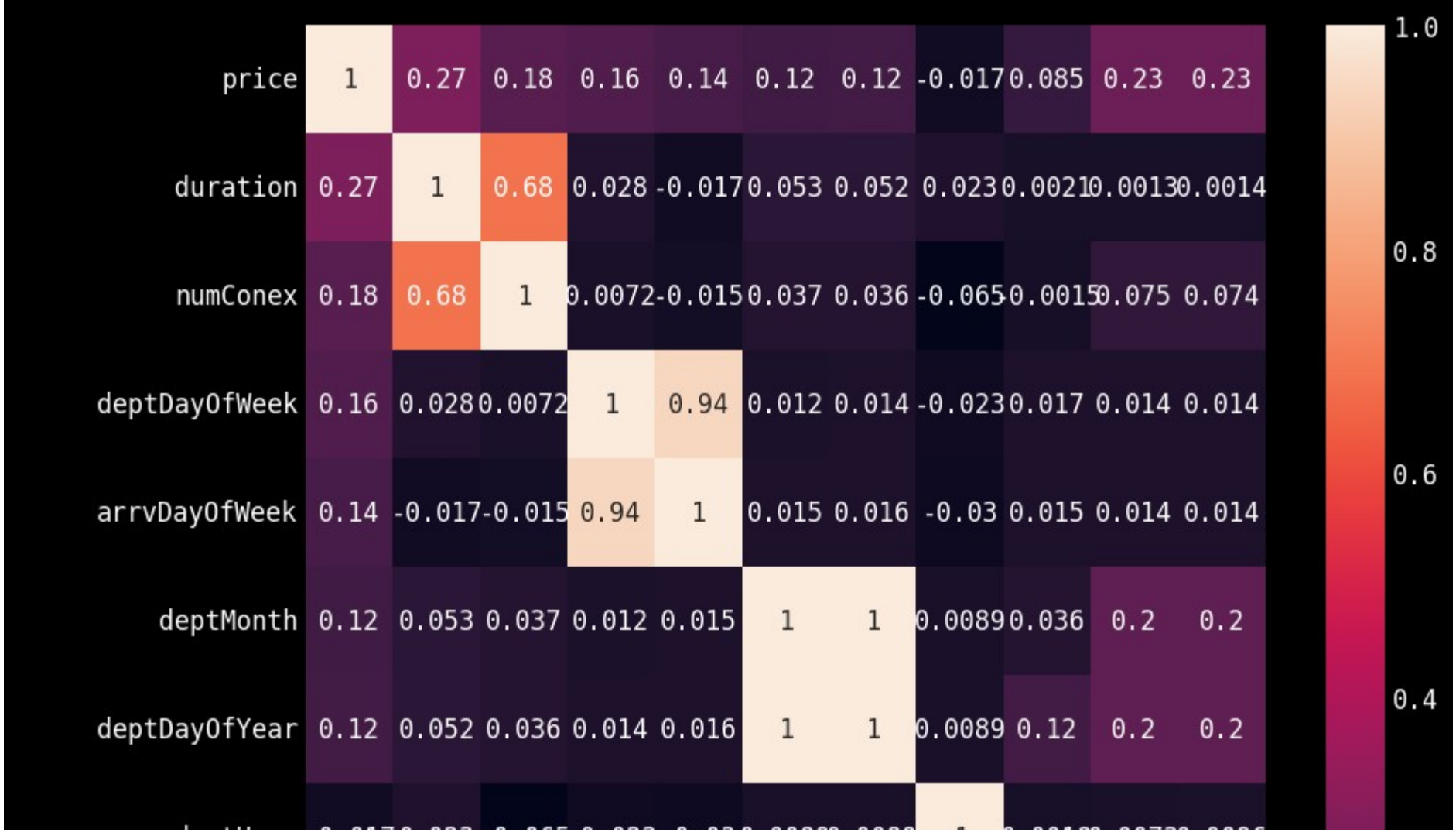
Seaborn Heatmap Exploration

In [76]:

```
#plot the correlations between features and targets
plt.rc('font', size=12)
f = plt.figure(figsize=(10,10))
sns.heatmap(df.corr(), annot=True)
```

Out[76]:

```
<Axes: >
```



All of the above heatmap definitely makes sense:

- duration strongly positively correlated to numConex
- arrival day strongly corresponds with departure day

Dummy Variable Creation

Now I need to create some dummy variables.

I had some thought too about whether month, dayOfWeek, etc. were numerical or categorical....

Monday=0, Tuesday=1, etc... but Tuesday is not bigger than Monday.

https://www.reddit.com/r/learnpython/comments/chunas/correlation_with_day_of_week/

https://mikulskibartosz.name/time-in-machine-learning

This might be a looooooooot of dummy variables...

In [77]:
```
# Get the dummies and store it in a variable
#dummies = pd.get_dummies(features['marktName'])
features_dum = pd.get_dummies(features, columns=['marktName', 'orgName', 'destName', 'deptDayOfWeek'])
features_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 47 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDayOfMonth               44367 non-null  int64
 1   deptDayOfYear                44367 non-null  int64
 2   deptHour                     44367 non-null  int64
 3   deptMonth                    44367 non-null  int64
 4   marktName_Aeromexico         44367 non-null  uint8
 5   marktName_Air Canada         44367 non-null  uint8
 6   marktName_Air Transat        44367 non-null  uint8
 7   marktName_Alaska Airlines    44367 non-null  uint8
 8   marktName_Allegiant Air      44367 non-null  uint8
 9   marktName_American Airlines  44367 non-null  uint8
 10  marktName_Breeze Airways     44367 non-null  uint8
 11  marktName_Canada Jetlines    44367 non-null  uint8
 12  marktName_Flair Airlines     44367 non-null  uint8
 13  marktName_Frontier Airlines  44367 non-null  uint8
 14  marktName_Key Lime Air       44367 non-null  uint8
 15  marktName_LATAM              44367 non-null  uint8
 16  marktName_Lynx Air           44367 non-null  uint8
 17  marktName_Porter Airlines    44367 non-null  uint8
 18  marktName_Silver Airways     44367 non-null  uint8
 19  marktName_Spirit Airlines    44367 non-null  uint8
 20  marktName_Sun Country        44367 non-null  uint8
 21  marktName_United Airlines    44367 non-null  uint8
 22  marktName_Volaris            44367 non-null  uint8
 23  marktName_WestJet            44367 non-null  uint8
 24  orgName_Chicago              44367 non-null  uint8
 25  orgName_Cincinnati           44367 non-null  uint8
 26  orgName_Denver               44367 non-null  uint8
 27  orgName_Los Angeles          44367 non-null  uint8
 28  orgName_Miami                44367 non-null  uint8
 29  orgName_New York             44367 non-null  uint8
 30  orgName_Orlando              44367 non-null  uint8
 31  orgName_San Francisco        44367 non-null  uint8
 32  destName_Chicago             44367 non-null  uint8
 33  destName_Cincinnati          44367 non-null  uint8
```

```
 34  destName_Denver              44367 non-null  uint8
 35  destName_Los Angeles         44367 non-null  uint8
 36  destName_Miami               44367 non-null  uint8
 37  destName_New York            44367 non-null  uint8
 38  destName_Orlando             44367 non-null  uint8
 39  destName_San Francisco       44367 non-null  uint8
 40  deptDayOfWeek_0              44367 non-null  uint8
 41  deptDayOfWeek_1              44367 non-null  uint8
 42  deptDayOfWeek_2              44367 non-null  uint8
 43  deptDayOfWeek_3              44367 non-null  uint8
 44  deptDayOfWeek_4              44367 non-null  uint8
 45  deptDayOfWeek_5              44367 non-null  uint8
 46  deptDayOfWeek_6              44367 non-null  uint8
dtypes: int64(4), uint8(43)
```

In [78]:

```
#split data into train and test sets
x_train, x_test, y_train, y_test = train_test_split(features_dum, targets, test_size=0.22, random_state=23)
print("x_train")
print(x_train)
print("x_test")
print(x_test)
print("y_train")
print(y_train)
print("y_test")
print(y_test)
#default size is 25% test size
```

```
x_train
       deptDayOfMonth  deptDayOfYear  deptHour  deptMonth  \
34054              17            169        15          6
2418               18            200        14          7
11475               9            130         7          5
42628              18             49         9          2
27043              17            108        17          4
...               ...            ...       ...        ...
9836               27            331        15         11
11322              24            115        14          4
26886               1             92        12          4
9338               29            242         7          8
42105              27            361         9         12

       marktName_Aeromexico  marktName_Air Canada  marktName_Air Transat  \
34054                     0                     0                      0
2418                      0                     0                      0
11475                     0                     0                      0
42628                     0                     0                      0
27043                     0                     0                      0
...                     ...                   ...                    ...
9836                      0                     0                      0
```

```
11322                     0                     0                     0
26886                     0                     0                     0
9338                      0                     0                     0
42105                     0                     0                     0

       marktName_Alaska Airlines  marktName_Allegiant Air  \
34054                          0                        0
2418                           0                        0
11475                          0                        0
42628                          0                        0
27043                          0                        0
...                          ...                      ...
9836                           0                        0
11322                          0                        0
26886                          0                        0
9338                           0                        0
42105                          0                        0

       marktName_American Airlines  ...  destName_New York  destName_Orlando  \
34054                            0  ...                  0                 0
2418                             1  ...                  0                 0
11475                            0  ...                  1                 0
42628                            0  ...                  0                 0
27043                            0  ...                  0                 0
...                            ...  ...                ...               ...
9836                             0  ...                  1                 0
11322                            0  ...                  1                 0
26886                            0  ...                  0                 0
9338                             0  ...                  0                 0
42105                            1  ...                  0                 0

       destName_San Francisco  deptDayOfWeek_0  deptDayOfWeek_1  \
34054                       0                1                0
2418                        0                0                0
11475                       0                0                0
42628                       0                0                0
27043                       0                0                0
...                       ...              ...              ...
9836                        0                1                0
11322                       0                0                0
26886                       0                1                0
9338                        0                0                0
42105                       0                0                0

       deptDayOfWeek_2  deptDayOfWeek_3  deptDayOfWeek_4  deptDayOfWeek_5  \
34054                0                0                0                0
2418                 0                1                0                0
11475                0                1                0                0
42628                0                0                0                0
27043                1                0                0                0
```

```
...                ...              ...              ...              ...
9836                 0                0                0                0
11322                1                0                0                0
26886                0                0                0                0
9338                 0                1                0                0
42105                1                0                0                0

       deptDayOfWeek_6
34054                0
2418                 0
11475                0
42628                1
27043                0
...                ...
9836                 0
11322                0
26886                0
9338                 0
42105                0

[34606 rows x 47 columns]
x_test
       deptDayOfMonth  deptDayOfYear  deptHour  deptMonth  \
16381              30            334         0         11
23903              24            115         8          4
27446              30            151         8          5
20403              29             89        10          3
16495              11            345         0         12
...               ...            ...       ...        ...
3303               26            330        14         11
1968                3            155        16          6
39820               3             94         8          4
6841               23            357        13         12
10324              15             15         7          1

       marktName_Aeromexico  marktName_Air Canada  marktName_Air Transat  \
16381                     0                     0                      0
23903                     0                     0                      0
27446                     0                     0                      0
20403                     0                     0                      0
16495                     0                     0                      0
...                     ...                   ...                    ...
3303                      0                     0                      0
1968                      0                     0                      0
39820                     0                     0                      0
6841                      0                     0                      0
10324                     0                     0                      0

       marktName_Alaska Airlines  marktName_Allegiant Air  \
16381                          0                        0
```

```
23903                          0                    0
27446                          0                    0
20403                          0                    0
16495                          0                    0
...                          ...                  ...
3303                           0                    0
1968                           0                    0
39820                          0                    0
6841                           0                    0
10324                          0                    0

       marktName_American Airlines  ...  destName_New York  destName_Orlando  \
16381                             0  ...                  0                 1
23903                             1  ...                  0                 0
27446                             0  ...                  0                 0
20403                             0  ...                  0                 0
16495                             0  ...                  0                 1
...                             ...  ...                ...               ...
3303                              1  ...                  0                 0
1968                              1  ...                  0                 0
39820                             1  ...                  0                 0
6841                              0  ...                  0                 0
10324                             0  ...                  1                 0

       destName_San Francisco  deptDayOfWeek_0  deptDayOfWeek_1  \
16381                       0                0                0
23903                       0                0                0
27446                       0                0                0
20403                       0                0                0
16495                       0                1                0
...                       ...              ...              ...
3303                        0                0                0
1968                        0                1                0
39820                       0                0                0
6841                        0                0                0
10324                       0                1                0

       deptDayOfWeek_2  deptDayOfWeek_3  deptDayOfWeek_4  deptDayOfWeek_5  \
16381                0                1                0                0
23903                1                0                0                0
27446                0                1                0                0
20403                0                0                1                0
16495                0                0                0                0
...                ...              ...              ...              ...
3303                 0                0                0                0
1968                 0                0                0                0
39820                1                0                0                0
6841                 0                0                0                1
10324                0                0                0                0
```

```
       deptDayOfWeek_6
16381                0
23903                0
27446                0
20403                0
16495                0
...                ...
3303                 1
1968                 0
39820                0
6841                 0
10324                0

[9761 rows x 47 columns]
y_train
        price
34054  342.20
2418   227.34
11475  136.06
42628  117.11
27043  229.22
...       ...
9836   344.93
11322  136.06
26886  277.20
9338   125.76
42105  512.62

[34606 rows x 1 columns]
y_test
        price
16381   96.26
23903  172.71
27446  229.22
20403  290.09
16495   91.23
...       ...
3303   343.44
1968   298.37
39820  145.20
6841   219.71
10324  275.78
```

In [79]:

```
#use at least one ML model to fit to the training data
#create the model first, then fit it to training data
lr_model = LinearRegression().fit(x_train, y_train)
#lr_model = lr_model.fit(x_train, y_train)
```

In [80]:
```
# https://realpython.com/linear-regression-in-python/
y_predictions = lr_model.predict(x_test)
```

In [81]:
```
# Calculate mean squared error
mse = mean_squared_error(y_test, y_predictions)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 6129.733066683015
```

In [82]:
```
# Calculate R-squared
r2 = r2_score(y_test, y_predictions)
print("R-squared:", r2)
```

```
R-squared: 0.3866250808600583
```

In [83]:
```
r_sq = lr_model.score(x_train, y_train)
print(f'R-squared again: {r_sq}')
```

```
R-squared again: 0.39676132785277063
```

Mean Absolute Error (MAE) will turn out to be the common measure between all of the ML models compared. This MAE value should be noted for later!

In [84]:
```
mae = mean_absolute_error(y_test, y_predictions)

print('Mean Absolute Error:', mae)
```

```
Mean Absolute Error: 54.67673555693131
```

In [ ]:
```

```

Pycaret Model Creation

So after creating our features_dum and targets for our test LinearRegression model, we want to move onto pycaret.

Before we do that though, I'm going to concatenate our features_dum and targets (well, target) into one dataframe, df_dum

In [85]:
```
df_dum = pd.concat([features_dum, targets], axis='columns')
```

In [86]:
```
df_dum.columns
```

Out[86]:
```
Index(['deptDayOfMonth', 'deptDayOfYear', 'deptHour', 'deptMonth',
       'marktName_Aeromexico', 'marktName_Air Canada', 'marktName_Air Transat',
       'marktName_Alaska Airlines', 'marktName_Allegiant Air',
       'marktName_American Airlines', 'marktName_Breeze Airways',
       'marktName_Canada Jetlines', 'marktName_Flair Airlines',
       'marktName_Frontier Airlines', 'marktName_Key Lime Air',
       'marktName_LATAM', 'marktName_Lynx Air', 'marktName_Porter Airlines',
       'marktName_Silver Airways', 'marktName_Spirit Airlines',
       'marktName_Sun Country', 'marktName_United Airlines',
       'marktName_Volaris', 'marktName_WestJet', 'orgName_Chicago',
       'orgName_Cincinnati', 'orgName_Denver', 'orgName_Los Angeles',
       'orgName_Miami', 'orgName_New York', 'orgName_Orlando',
       'orgName_San Francisco', 'destName_Chicago', 'destName_Cincinnati',
       'destName_Denver', 'destName_Los Angeles', 'destName_Miami',
       'destName_New York', 'destName_Orlando', 'destName_San Francisco',
       'deptDayOfWeek_0', 'deptDayOfWeek_1', 'deptDayOfWeek_2',
       'deptDayOfWeek_3', 'deptDayOfWeek_4', 'deptDayOfWeek_5',
       'deptDayOfWeek_6', 'price'],
      dtype='object')
```

In [87]:
```
df_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 48 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDayOfMonth               44367 non-null  int64
 1   deptDayOfYear                44367 non-null  int64
 2   deptHour                     44367 non-null  int64
 3   deptMonth                    44367 non-null  int64
 4   marktName_Aeromexico         44367 non-null  uint8
 5   marktName_Air Canada         44367 non-null  uint8
 6   marktName_Air Transat        44367 non-null  uint8
 7   marktName_Alaska Airlines    44367 non-null  uint8
 8   marktName_Allegiant Air      44367 non-null  uint8
 9   marktName_American Airlines  44367 non-null  uint8
 10  marktName_Breeze Airways     44367 non-null  uint8
 11  marktName_Canada Jetlines    44367 non-null  uint8
 12  marktName_Flair Airlines     44367 non-null  uint8
 13  marktName_Frontier Airlines  44367 non-null  uint8
 14  marktName_Key Lime Air       44367 non-null  uint8
 15  marktName_LATAM              44367 non-null  uint8
 16  marktName_Lynx Air           44367 non-null  uint8
 17  marktName_Porter Airlines    44367 non-null  uint8
 18  marktName_Silver Airways     44367 non-null  uint8
 19  marktName_Spirit Airlines    44367 non-null  uint8
 20  marktName_Sun Country        44367 non-null  uint8
 21  marktName_United Airlines    44367 non-null  uint8
```

```
 22  marktName_Volaris            44367 non-null  uint8
 23  marktName_WestJet            44367 non-null  uint8
 24  orgName_Chicago              44367 non-null  uint8
 25  orgName_Cincinnati           44367 non-null  uint8
 26  orgName_Denver               44367 non-null  uint8
 27  orgName_Los Angeles          44367 non-null  uint8
 28  orgName_Miami                44367 non-null  uint8
 29  orgName_New York             44367 non-null  uint8
 30  orgName_Orlando              44367 non-null  uint8
 31  orgName_San Francisco        44367 non-null  uint8
 32  destName_Chicago             44367 non-null  uint8
 33  destName_Cincinnati          44367 non-null  uint8
 34  destName_Denver              44367 non-null  uint8
 35  destName_Los Angeles         44367 non-null  uint8
 36  destName_Miami               44367 non-null  uint8
 37  destName_New York            44367 non-null  uint8
 38  destName_Orlando             44367 non-null  uint8
 39  destName_San Francisco       44367 non-null  uint8
 40  deptDayOfWeek_0              44367 non-null  uint8
 41  deptDayOfWeek_1              44367 non-null  uint8
 42  deptDayOfWeek_2              44367 non-null  uint8
 43  deptDayOfWeek_3              44367 non-null  uint8
 44  deptDayOfWeek_4              44367 non-null  uint8
 45  deptDayOfWeek_5              44367 non-null  uint8
 46  deptDayOfWeek_6              44367 non-null  uint8
 47  price                        44367 non-null  float64
dtypes: float64(1), int64(4), uint8(43)
```

In [88]:

```
df_dum.head(10)
```

Out[88]:

	deptDayOfMonth	deptDayOfYear	deptHour	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Alleg
0	20	324	20	11	0	0	0	0	
1	20	324	9	11	0	0	0	0	
2	20	324	5	11	0	0	0	0	
3	20	324	17	11	0	0	0	0	
4	20	324	9	11	0	0	0	0	
5	20	324	5	11	0	0	0	0	
6	20	324	7	11	0	0	0	0	
7	20	324	6	11	0	0	0	0	
8	20	324	16	11	0	0	0	0	

	deptDayOfMonth	deptDayOfYear	deptHour	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Alleg
9	20	324	23	11	0	0	0	0	

Pycaret Model 1: Parsed Time, KFold Generator

Now I want to use pycarat to compare ml models

https://www.pycaret.org/tutorials/html/REG101.html

https://pycaret.readthedocs.io/en/latest/api/regression.html

I don't really understand the default 'fold' parameter of 10 but it seems to be related to computational power. I will stick with the default of 10.

"Reducing the number of folds will improve the training time."

In [89]:
```
reg1 = setup(data = df_dum, target = 'price', session_id=23)
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 48)
4	Transformed data shape	(44367, 48)
5	Transformed train set shape	(31056, 48)
6	Transformed test set shape	(13311, 48)
7	Numeric features	47
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	KFold
13	Fold Number	10
14	CPU Jobs	-1

	Description	Value
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	reg-default-name

In [90]:
```
best = compare_models(n_select=4, exclude="auto_arima")
```

	Model	MAE	MSE	RMSE	R2	RMSLE
rf	Random Forest Regressor	22.3922	1738.2994	41.5714	0.8287	0.1681
et	Extra Trees Regressor	22.2305	1872.4788	43.1010	0.8158	0.1758
lightgbm	Light Gradient Boosting Machine	32.3108	2519.4096	50.1319	0.7515	0.2108
dt	Decision Tree Regressor	25.6773	2982.4714	54.4355	0.7057	0.2140
gbr	Gradient Boosting Regressor	45.2813	4465.6951	66.7875	0.5591	0.2847
knn	K Neighbors Regressor	49.8493	5658.2161	75.1728	0.4418	0.3229
lr	Linear Regression	53.9990	6122.7636	78.2216	0.3951	0.3613
br	Bayesian Ridge	53.9890	6122.9232	78.2227	0.3951	0.3612
ridge	Ridge Regression	53.9794	6124.6803	78.2341	0.3949	0.3611
lasso	Lasso Regression	58.4987	7137.1279	84.4442	0.2954	0.3856

	Model	MAE	MSE	RMSE	R2	RMSLE
llar	Lasso Least Angle Regression	58.4987	7137.1289	84.4442	0.2954	0.3856
huber	Huber Regressor	57.5096	7886.4837	88.7611	0.2216	0.3812
en	Elastic Net	68.6523	9333.2001	96.5660	0.0788	0.4502
omp	Orthogonal Matching Pursuit	68.9916	9498.1155	97.4155	0.0625	0.4559
dummy	Dummy Regressor	72.5853	10133.5062	100.6225	-0.0003	0.4723
ada	AdaBoost Regressor	99.5011	14183.9968	117.4617	-0.3918	0.5758
	Passive					

The pycaret Linear Regression results track with our former Linear Regression that we performed ourselves!

Looks like Random Forest is the best model according to Mean Squared Error, Root Mean Squared Error, and R^2 value.

Let's go with that. I also want to create a Decision Tree and K-Nearest Neighbor model to compare.

Random Forest

In [91]:
```
model_rf = create_model('rf')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	21.4681	1478.6607	38.4534	0.8430	0.1630	0.1066
1	22.8740	1704.0184	41.2798	0.8400	0.1699	0.1110
2	22.4565	1572.5232	39.6551	0.8431	0.1685	0.1093
3	22.6916	1691.9252	41.1330	0.8136	0.1710	0.1142
4	22.8040	1880.2239	43.3615	0.8124	0.1723	0.1086
5	21.3979	1492.2430	38.6296	0.8475	0.1654	0.1078

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
6	22.5436	2387.7590	48.8647	0.7842	0.1679	0.1077
7	23.3817	2044.7517	45.2189	0.8069	0.1720	0.1139
8	21.8471	1623.6780	40.2949	0.8371	0.1612	0.1069

In [92]:
```
evaluate_model(model_rf)
```

Let's run a simple test to see if we can get a price prediction from a datapoint input.

In [93]:
```
new_data = pd.DataFrame({'deptDayOfWeek': [0], 'deptDayOfMonth': [27], 'deptDayOfYear': ['2023-11-27'],
                         'deptHour': [10], 'deptMonth': [11], 'marktName_Aeromexico': [0], 'marktName_Air Canada': [0],
                         'marktName_Air Transat': [0], 'marktName_Alaska Airlines': [0], 'marktName_Allegiant Air': [0],
                         'marktName_American Airlines': [0], 'marktName_Breeze Airways': [0], 'marktName_Canada Jetlines': [0],
                         'marktName_Flair Airlines': [0], 'marktName_Frontier Airlines': [0], 'marktName_Key Lime Air': [0],
                         'marktName_LATAM': [0], 'marktName_Lynx Air': [0], 'marktName_Porter Airlines': [0],
                         'marktName_Silver Airways': [0], 'marktName_Spirit Airlines': [0], 'marktName_Sun Country': [0],
                         'marktName_United Airlines': [1], 'marktName_Volaris': [0], 'marktName_WestJet': [0],
                         'orgName_Chicago': [0], 'orgName_Cincinnati': [0], 'orgName_Denver': [1], 'orgName_Los Angeles': [0],
                         'orgName_Miami': [0], 'orgName_New York': [0], 'orgName_Orlando': [0], 'orgName_San Francisco': [0],
                         'destName_Chicago': [0], 'destName_Cincinnati': [1], 'destName_Denver': [0], 'destName_Los Angeles': [0],
                         'destName_Miami': [0], 'destName_New York': [0], 'destName_Orlando': [0], 'destName_San Francisco': [0],
                         'deptDayOfWeek_0': [1], 'deptDayOfWeek_1': [0], 'deptDayOfWeek_2': [0], 'deptDayOfWeek_3': [0],
                         'deptDayOfWeek_4': [0], 'deptDayOfWeek_5': [0], 'deptDayOfWeek_6': [0]})

new_data['deptDayOfYear'] = pd.to_datetime(new_data['deptDayOfYear'])
new_data['deptDayOfYear'] = new_data['deptDayOfYear'].dt.dayofyear

predictions = predict_model(model_rf, data = new_data)
```

In [94]:
```
print(predictions['prediction_label'])
```

```
0    264.464202
Name: prediction_label, dtype: float64
```

In [95]:
```
predictions = predict_model(model_rf, data = x_test)
```

In [96]:
```
preds = predictions['prediction_label']
preds.head()
```

Out[96]:
```
16381     87.653002
23903    179.715905
27446    222.719013
20403    427.968006
16495     78.578001
Name: prediction_label, dtype: float64
```

In [97]:
```
print(f'R2 Score: {r2_score(y_test, preds)}')
```

```
R2 Score: 0.8440041624094571
```

In [ ]:
```

```

Extra Tree Regressor

In [98]:
```
model_et = create_model('et')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	21.4089	1578.5004	39.7303	0.8324	0.1700	0.1071
1	22.6758	1848.9770	42.9997	0.8264	0.1765	0.1105
2	22.3096	1705.5651	41.2985	0.8299	0.1752	0.1085
3	22.1988	1575.0341	39.6867	0.8265	0.1754	0.1123
4	23.3304	2091.3530	45.7313	0.7913	0.1859	0.1138
5	21.3780	1589.1883	39.8646	0.8376	0.1724	0.1087
6	22.2317	2533.8371	50.3372	0.7711	0.1736	0.1060
7	23.3222	2446.0959	49.4580	0.7690	0.1824	0.1149
8	21.4539	1763.6041	41.9953	0.8231	0.1690	0.1067
9	21.9958	1592.6332	39.9078	0.8512	0.1776	0.1119
Mean	22.2305	1872.4788	43.1010	0.8158	0.1758	0.1100
Std	0.6831	344.6835	3.8453	0.0270	0.0050	0.0030

In [99]:
```
evaluate_model(model_et)
```

K-Nearest Neighbor

In [100…
```
model_knn = create_model('knn')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	49.0095	5200.4512	72.1142	0.4479	0.3176	0.2602
1	51.8882	6225.9072	78.9044	0.4156	0.3306	0.2693
2	50.2068	5517.0532	74.2769	0.4497	0.3205	0.2630
3	49.0377	5051.8066	71.0761	0.4434	0.3223	0.2665
4	49.4602	5488.2041	74.0824	0.4523	0.3152	0.2544
5	49.1159	5444.3452	73.7858	0.4438	0.3335	0.2740
6	48.9467	6167.6343	78.5343	0.4427	0.3122	0.2533
7	50.7686	6007.2153	77.5062	0.4327	0.3296	0.2673
8	48.5615	5498.1953	74.1498	0.4484	0.3139	0.2540
9	51.3817	6111.9751	78.1791	0.4290	0.3362	0.2747
Mean	49.8377	5671.2788	75.2609	0.4405	0.3232	0.2637
Std	1.0956	400.6533	2.6595	0.0109	0.0083	0.0076

In [101…
```
evaluate_model(model_knn)
```

Light Gradient Boosting Machine

In [102…
```
model_lgbm = create_model('lightgbm')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	32.0688	2341.1349	48.3853	0.7514	0.2098	0.1639
1	32.5116	2546.6116	50.4640	0.7610	0.2084	0.1629
2	32.2599	2322.8398	48.1958	0.7683	0.2077	0.1636
3	32.4749	2256.9601	47.5075	0.7513	0.2140	0.1673
4	32.8542	2758.7849	52.5241	0.7247	0.2157	0.1639
5	31.3629	2277.7269	47.7255	0.7673	0.2103	0.1641
6	32.1217	3110.0358	55.7677	0.7190	0.2042	0.1593
7	33.0853	2707.3990	52.0327	0.7443	0.2152	0.1675
8	32.1866	2461.3481	49.6120	0.7531	0.2063	0.1645
9	32.1824	2411.2548	49.1045	0.7747	0.2169	0.1674

In [103…

```
evaluate_model(model_lgbm)
```

In [ ]:

Pycaret Model 2: Parsed + Ordinate Time, KFold Generator

I will follow the same process with the next pycaret comparison. We'll create features and targets, compare the different pycaret models, and observe the Mean Absolute Error of each of the models. This time, we'll use the ordinate time function that we explored earlier.

Let's see how it does!

In [104…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   destName            44367 non-null  object
 1   orgName             44367 non-null  object
 2   deptDateTime        44367 non-null  datetime64[ns]
 3   arrvDateTime        44367 non-null  datetime64[ns]
```

```
 4   price                44367 non-null  float64
 5   marktName            44367 non-null  object
 6   flightNum            44367 non-null  object
 7   duration             44367 non-null  float64
 8   numConex             44367 non-null  int64
 9   url                  44367 non-null  object
 10  deptDayOfWeek        44367 non-null  int64
 11  arrvDayOfWeek        44367 non-null  int64
 12  deptMonth            44367 non-null  int64
 13  deptDate             44367 non-null  datetime64[ns]
 14  arrvDate             44367 non-null  datetime64[ns]
 15  deptDayOfYear        44367 non-null  int64
 16  deptHour             44367 non-null  int64
 17  deptDayOfMonth       44367 non-null  int64
 18  deptDateTimeOrd      44367 non-null  int64
 19  deptDateTimeMktime   44367 non-null  float64
dtypes: datetime64[ns](4), float64(3), int64(8), object(5)
memory usage: 8.1+ MB
```

In [105…

```
features2 = df[['deptDateTimeOrd', 'deptDayOfWeek', 'deptHour', 'deptMonth', 'marktName', 'orgName', 'destName']]
targets2 = df[['price']]
```

In [106…

```
features2_dum = pd.get_dummies(features2, columns=['marktName', 'orgName', 'destName', 'deptDayOfWeek'])
features2_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 46 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDateTimeOrd              44367 non-null  int64
 1   deptHour                     44367 non-null  int64
 2   deptMonth                    44367 non-null  int64
 3   marktName_Aeromexico         44367 non-null  uint8
 4   marktName_Air Canada         44367 non-null  uint8
 5   marktName_Air Transat        44367 non-null  uint8
 6   marktName_Alaska Airlines    44367 non-null  uint8
 7   marktName_Allegiant Air      44367 non-null  uint8
 8   marktName_American Airlines  44367 non-null  uint8
 9   marktName_Breeze Airways     44367 non-null  uint8
 10  marktName_Canada Jetlines    44367 non-null  uint8
 11  marktName_Flair Airlines     44367 non-null  uint8
 12  marktName_Frontier Airlines  44367 non-null  uint8
 13  marktName_Key Lime Air       44367 non-null  uint8
 14  marktName_LATAM              44367 non-null  uint8
 15  marktName_Lynx Air           44367 non-null  uint8
 16  marktName_Porter Airlines    44367 non-null  uint8
 17  marktName_Silver Airways     44367 non-null  uint8
```

```
 18  marktName_Spirit Airlines    44367 non-null  uint8
 19  marktName_Sun Country        44367 non-null  uint8
 20  marktName_United Airlines    44367 non-null  uint8
 21  marktName_Volaris            44367 non-null  uint8
 22  marktName_WestJet            44367 non-null  uint8
 23  orgName_Chicago              44367 non-null  uint8
 24  orgName_Cincinnati           44367 non-null  uint8
 25  orgName_Denver               44367 non-null  uint8
 26  orgName_Los Angeles          44367 non-null  uint8
 27  orgName_Miami                44367 non-null  uint8
 28  orgName_New York             44367 non-null  uint8
 29  orgName_Orlando              44367 non-null  uint8
 30  orgName_San Francisco        44367 non-null  uint8
 31  destName_Chicago             44367 non-null  uint8
 32  destName_Cincinnati          44367 non-null  uint8
 33  destName_Denver              44367 non-null  uint8
 34  destName_Los Angeles         44367 non-null  uint8
 35  destName_Miami               44367 non-null  uint8
 36  destName_New York            44367 non-null  uint8
 37  destName_Orlando             44367 non-null  uint8
 38  destName_San Francisco       44367 non-null  uint8
 39  deptDayOfWeek_0              44367 non-null  uint8
 40  deptDayOfWeek_1              44367 non-null  uint8
 41  deptDayOfWeek_2              44367 non-null  uint8
 42  deptDayOfWeek_3              44367 non-null  uint8
 43  deptDayOfWeek_4              44367 non-null  uint8
 44  deptDayOfWeek_5              44367 non-null  uint8
 45  deptDayOfWeek_6              44367 non-null  uint8
dtypes: int64(3), uint8(43)
```

In [107…

```
df_dum2 = pd.concat([features2_dum, targets2], axis='columns')
```

In [108…

```
x_train2, x_test2, y_train2, y_test2 = train_test_split(features2_dum, targets2, test_size=0.22, random_state=23)
```

In [109…

```
reg2 = setup(data = df_dum2, target = 'price', session_id=23)
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 47)
4	Transformed data shape	(44367, 47)

	Description	Value
5	Transformed train set shape	(31056, 47)
6	Transformed test set shape	(13311, 47)
7	Numeric features	46
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	KFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	reg-default-name

In [110…

```
best2 = compare_models(n_select=4)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
rf	Random Forest Regressor	22.0532	1717.3270	41.3090	0.8309	0.1659	0.1076	4.6150
et	Extra Trees Regressor	22.2147	1858.2160	42.9357	0.8172	0.1758	0.1095	4.9450
lightgbm	Light Gradient Boosting Machine	32.2642	2553.3294	50.4808	0.7480	0.2124	0.1646	0.2350
dt	Decision Tree Regressor	25.3544	2852.1946	53.3291	0.7184	0.2118	0.1194	0.1470
gbr	Gradient Boosting Regressor	45.2910	4463.4364	66.7732	0.5593	0.2837	0.2374	1.3040
knn	K Neighbors Regressor	47.4499	5246.6422	72.3906	0.4823	0.3085	0.2495	0.2850
lr	Linear Regression	53.8426	6130.1808	78.2659	0.3944	0.3546	0.2978	0.0820
ridge	Ridge Regression	53.8520	6130.5599	78.2685	0.3944	0.3546	0.2979	0.0540
br	Bayesian Ridge	53.8478	6130.3442	78.2670	0.3944	0.3546	0.2979	0.1020
lasso	Lasso Regression	58.1187	6978.8266	83.5009	0.3111	0.3743	0.3267	0.0610

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
llar	Lasso Least Angle Regression	58.1187	6978.8256	83.5009	0.3111	0.3743	0.3267	0.0570
en	Elastic Net	67.5013	8970.3864	94.6703	0.1146	0.4303	0.3942	0.0580
omp	Orthogonal Matching Pursuit	67.3252	9100.5236	95.3548	0.1017	0.4348	0.3949	0.0600
dummy	Dummy Regressor	72.5853	10133.5062	100.6225	-0.0003	0.4723	0.4483	0.0510
huber	Huber Regressor	70.1636	10387.5637	101.8747	-0.0253	0.4613	0.4020	0.0980
ada	AdaBoost Regressor	123.9946	19998.1895	140.6140	-0.9807	0.6649	0.8806	1.1440

In [111…

```
model_rf2 = create_model('rf')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	21.1546	1414.0188	37.6034	0.8499	0.1610	0.1047
1	22.2905	1615.3289	40.1912	0.8484	0.1673	0.1079
2	22.4011	1572.6652	39.6568	0.8431	0.1669	0.1075
3	22.2372	1601.5141	40.0189	0.8235	0.1671	0.1114
4	22.4817	1902.4926	43.6176	0.8101	0.1703	0.1064
5	21.4428	1535.8340	39.1897	0.8431	0.1649	0.1068
6	22.2789	2384.6629	48.8330	0.7845	0.1658	0.1054
7	22.8589	2045.1897	45.2238	0.8069	0.1694	0.1105
8	21.3360	1582.6828	39.7829	0.8412	0.1583	0.1046
9	22.0507	1518.8811	38.9728	0.8581	0.1675	0.1106
Mean	22.0532	1717.3270	41.3090	0.8309	0.1659	0.1076
Std	0.5281	285.3605	3.3003	0.0224	0.0035	0.0024

In [112…

```
evaluate_model(model_rf2)
```

In [113…

```
x_test2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9761 entries, 16381 to 10324
Data columns (total 46 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDateTimeOrd              9761 non-null   int64
 1   deptHour                     9761 non-null   int64
 2   deptMonth                    9761 non-null   int64
 3   marktName_Aeromexico         9761 non-null   uint8
 4   marktName_Air Canada         9761 non-null   uint8
 5   marktName_Air Transat        9761 non-null   uint8
 6   marktName_Alaska Airlines    9761 non-null   uint8
 7   marktName_Allegiant Air      9761 non-null   uint8
 8   marktName_American Airlines  9761 non-null   uint8
 9   marktName_Breeze Airways     9761 non-null   uint8
 10  marktName_Canada Jetlines    9761 non-null   uint8
 11  marktName_Flair Airlines     9761 non-null   uint8
 12  marktName_Frontier Airlines  9761 non-null   uint8
 13  marktName_Key Lime Air       9761 non-null   uint8
 14  marktName_LATAM              9761 non-null   uint8
 15  marktName_Lynx Air           9761 non-null   uint8
 16  marktName_Porter Airlines    9761 non-null   uint8
 17  marktName_Silver Airways     9761 non-null   uint8
 18  marktName_Spirit Airlines    9761 non-null   uint8
 19  marktName_Sun Country        9761 non-null   uint8
 20  marktName_United Airlines    9761 non-null   uint8
 21  marktName_Volaris            9761 non-null   uint8
 22  marktName_WestJet            9761 non-null   uint8
 23  orgName_Chicago              9761 non-null   uint8
 24  orgName_Cincinnati           9761 non-null   uint8
 25  orgName_Denver               9761 non-null   uint8
 26  orgName_Los Angeles          9761 non-null   uint8
 27  orgName_Miami                9761 non-null   uint8
 28  orgName_New York             9761 non-null   uint8
 29  orgName_Orlando              9761 non-null   uint8
 30  orgName_San Francisco        9761 non-null   uint8
 31  destName_Chicago             9761 non-null   uint8
 32  destName_Cincinnati          9761 non-null   uint8
 33  destName_Denver              9761 non-null   uint8
 34  destName_Los Angeles         9761 non-null   uint8
 35  destName_Miami               9761 non-null   uint8
 36  destName_New York            9761 non-null   uint8
 37  destName_Orlando             9761 non-null   uint8
 38  destName_San Francisco       9761 non-null   uint8
 39  deptDayOfWeek_0              9761 non-null   uint8
 40  deptDayOfWeek_1              9761 non-null   uint8
```

```
 41  deptDayOfWeek_2              9761 non-null   uint8
 42  deptDayOfWeek_3              9761 non-null   uint8
 43  deptDayOfWeek_4              9761 non-null   uint8
 44  deptDayOfWeek_5              9761 non-null   uint8
 45  deptDayOfWeek_6              9761 non-null   uint8
dtypes: int64(3), uint8(43)
```

In [114…

```
x_test2.head()
```

Out[114…

	deptDateTimeOrd	deptHour	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Allegiant Air	markt
16381	738854	0	11	0	0	0	0	0	
23903	739000	8	4	0	0	0	0	0	
27446	739036	8	5	0	0	0	0	0	
20403	738974	10	3	0	0	0	0	0	
16495	738865	0	12	0	0	0	0	0	

5 rows × 46 columns

In [115…

```
predictions2 = predict_model(model_rf2, data = x_test2)
```

In [116…

```
predictions2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9761 entries, 16381 to 10324
Data columns (total 47 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDateTimeOrd              9761 non-null   int32
 1   deptHour                     9761 non-null   int8
 2   deptMonth                    9761 non-null   int8
 3   marktName_Aeromexico         9761 non-null   uint8
 4   marktName_Air Canada         9761 non-null   uint8
 5   marktName_Air Transat        9761 non-null   uint8
 6   marktName_Alaska Airlines    9761 non-null   uint8
 7   marktName_Allegiant Air      9761 non-null   uint8
 8   marktName_American Airlines  9761 non-null   uint8
 9   marktName_Breeze Airways     9761 non-null   uint8
 10  marktName_Canada Jetlines    9761 non-null   uint8
 11  marktName_Flair Airlines     9761 non-null   uint8
 12  marktName_Frontier Airlines  9761 non-null   uint8
```

```
 13  marktName_Key Lime Air       9761 non-null   uint8
 14  marktName_LATAM              9761 non-null   uint8
 15  marktName_Lynx Air           9761 non-null   uint8
 16  marktName_Porter Airlines    9761 non-null   uint8
 17  marktName_Silver Airways     9761 non-null   uint8
 18  marktName_Spirit Airlines    9761 non-null   uint8
 19  marktName_Sun Country        9761 non-null   uint8
 20  marktName_United Airlines    9761 non-null   uint8
 21  marktName_Volaris            9761 non-null   uint8
 22  marktName_WestJet            9761 non-null   uint8
 23  orgName_Chicago              9761 non-null   uint8
 24  orgName_Cincinnati           9761 non-null   uint8
 25  orgName_Denver               9761 non-null   uint8
 26  orgName_Los Angeles          9761 non-null   uint8
 27  orgName_Miami                9761 non-null   uint8
 28  orgName_New York             9761 non-null   uint8
 29  orgName_Orlando              9761 non-null   uint8
 30  orgName_San Francisco        9761 non-null   uint8
 31  destName_Chicago             9761 non-null   uint8
 32  destName_Cincinnati          9761 non-null   uint8
 33  destName_Denver              9761 non-null   uint8
 34  destName_Los Angeles         9761 non-null   uint8
 35  destName_Miami               9761 non-null   uint8
 36  destName_New York            9761 non-null   uint8
 37  destName_Orlando             9761 non-null   uint8
 38  destName_San Francisco       9761 non-null   uint8
 39  deptDayOfWeek_0              9761 non-null   uint8
 40  deptDayOfWeek_1              9761 non-null   uint8
 41  deptDayOfWeek_2              9761 non-null   uint8
 42  deptDayOfWeek_3              9761 non-null   uint8
 43  deptDayOfWeek_4              9761 non-null   uint8
 44  deptDayOfWeek_5              9761 non-null   uint8
 45  deptDayOfWeek_6              9761 non-null   uint8
 46  prediction_label             9761 non-null   float64
dtypes: float64(1), int32(1), int8(2), uint8(43)
memory usage: 619.6 KB
```

In [117…

```
predictions2.head()
```

Out[117…

	deptDateTimeOrd	deptHour	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Allegiant Air	markt
16381	738854	0	11	0	0	0	0	0	
23903	739000	8	4	0	0	0	0	0	
27446	739036	8	5	0	0	0	0	0	
20403	738974	10	3	0	0	0	0	0	

	deptDateTimeOrd	deptHour	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Allegiant Air	markt
16495	738065	0	12	0	0	0	0	0	

In [118…
```
y_test2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9761 entries, 16381 to 10324
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   price   9761 non-null   float64
dtypes: float64(1)
memory usage: 152.5 KB
```

In [119…
```
y_test2.head()
```

Out[119…

	price
16381	96.26
23903	172.71
27446	229.22
20403	290.09
16495	91.23

In [120…
```
preds2 = predictions2['prediction_label']
preds2.head()
```

Out[120…
```
16381     80.297802
23903    173.063239
27446    219.882828
20403    441.724607
16495     78.581000
Name: prediction_label, dtype: float64
```

In [121…
```
print(f'R2 Score: {r2_score(y_test2, preds2)}')
```

```
R2 Score: 0.8432711378301447
```

Let's do a simple test to see if we can create a flight price prediction from an input.

In [122…

```
new_data2 = pd.DataFrame({'deptDateTimeOrd': ['2024-11-27'],
                          'deptHour': [10], 'deptMonth': [11], 'marktName_Aeromexico': [0], 'marktName_Air Canada': [0],
                          'marktName_Air Transat': [0], 'marktName_Alaska Airlines': [0], 'marktName_Allegiant Air': [0],
                          'marktName_American Airlines': [0], 'marktName_Breeze Airways': [0], 'marktName_Canada Jetlines': [0],
                          'marktName_Flair Airlines': [0], 'marktName_Frontier Airlines': [0], 'marktName_Key Lime Air': [0],
                          'marktName_LATAM': [0], 'marktName_Lynx Air': [0], 'marktName_Porter Airlines': [0],
                          'marktName_Silver Airways': [0], 'marktName_Spirit Airlines': [0], 'marktName_Sun Country': [0],
                          'marktName_United Airlines': [1], 'marktName_Volaris': [0], 'marktName_WestJet': [0],
                          'orgName_Chicago': [0], 'orgName_Cincinnati': [0], 'orgName_Denver': [1], 'orgName_Los Angeles': [0],
                          'orgName_Miami': [0], 'orgName_New York': [0], 'orgName_Orlando': [0], 'orgName_San Francisco': [0],
                          'destName_Chicago': [0], 'destName_Cincinnati': [1], 'destName_Denver': [0], 'destName_Los Angeles': [0],
                          'destName_Miami': [0], 'destName_New York': [0], 'destName_Orlando': [0], 'destName_San Francisco': [0],
                          'deptDayOfWeek_0': [1], 'deptDayOfWeek_1': [0], 'deptDayOfWeek_2': [0], 'deptDayOfWeek_3': [0],
                          'deptDayOfWeek_4': [0], 'deptDayOfWeek_5': [0], 'deptDayOfWeek_6': [0]})

new_data2['deptDateTimeOrd'] = pd.to_datetime(new_data2['deptDateTimeOrd'])
new_data2['deptDateTimeOrd'] = new_data2['deptDateTimeOrd'].apply(getordinal)


prediction2 = predict_model(model_rf2, data = new_data2)
print(new_data2.iloc[0])
```

```
deptDateTimeOrd               739217
deptHour                          10
deptMonth                         11
marktName_Aeromexico               0
marktName_Air Canada               0
marktName_Air Transat              0
marktName_Alaska Airlines          0
marktName_Allegiant Air            0
marktName_American Airlines        0
marktName_Breeze Airways           0
marktName_Canada Jetlines          0
marktName_Flair Airlines           0
marktName_Frontier Airlines        0
marktName_Key Lime Air             0
marktName_LATAM                    0
marktName_Lynx Air                 0
marktName_Porter Airlines          0
marktName_Silver Airways           0
marktName_Spirit Airlines          0
marktName_Sun Country              0
marktName_United Airlines          1
marktName_Volaris                  0
marktName_WestJet                  0
orgName_Chicago                    0
orgName_Cincinnati                 0
orgName_Denver                     1
```

```
orgName_Los Angeles                0
orgName_Miami                      0
orgName_New York                   0
orgName_Orlando                    0
orgName_San Francisco              0
destName_Chicago                   0
destName_Cincinnati                1
destName_Denver                    0
destName_Los Angeles               0
destName_Miami                     0
destName_New York                  0
destName_Orlando                   0
destName_San Francisco             0
deptDayOfWeek_0                    1
deptDayOfWeek_1                    0
deptDayOfWeek_2                    0
deptDayOfWeek_3                    0
deptDayOfWeek_4                    0
deptDayOfWeek_5                    0
deptDayOfWeek_6                    0
Name: 0, dtype: int64
```

In [123…

```
print(prediction2['prediction_label'])
```

```
0    277.763306
Name: prediction_label, dtype: float64
```

In [124…

```
def make2digit(num):
    if len(num) < 2:
        return '0' + num
    return num
```

In [125…

```
df_input = pd.DataFrame({'deptDateTimeOrd': [''],
                         'deptHour': [0], 'deptMonth': ['1'], 'marktName_Aeromexico': [0], 'marktName_Air Canada': [0],
                         'marktName_Air Transat': [0], 'marktName_Alaska Airlines': [0], 'marktName_Allegiant Air': [0],
                         'marktName_American Airlines': [0], 'marktName_Breeze Airways': [0], 'marktName_Canada Jetlines': [0],
                         'marktName_Flair Airlines': [0], 'marktName_Frontier Airlines': [0], 'marktName_Key Lime Air': [0],
                         'marktName_LATAM': [0], 'marktName_Lynx Air': [0], 'marktName_Porter Airlines': [0],
                         'marktName_Silver Airways': [0], 'marktName_Spirit Airlines': [0], 'marktName_Sun Country': [0],
                         'marktName_United Airlines': [0], 'marktName_Volaris': [0], 'marktName_WestJet': [0],
                         'orgName_Chicago': [0], 'orgName_Cincinnati': [0], 'orgName_Denver': [1], 'orgName_Los Angeles': [0],
                         'orgName_Miami': [0], 'orgName_New York': [0], 'orgName_Orlando': [0], 'orgName_San Francisco': [0],
                         'destName_Chicago': [0], 'destName_Cincinnati': [0], 'destName_Denver': [0], 'destName_Los Angeles': [0],
                         'destName_Miami': [0], 'destName_New York': [0], 'destName_Orlando': [0], 'destName_San Francisco': [0],
                         'deptDayOfWeek_0': [0], 'deptDayOfWeek_1': [0], 'deptDayOfWeek_2': [0], 'deptDayOfWeek_3': [0],
                         'deptDayOfWeek_4': [0], 'deptDayOfWeek_5': [0], 'deptDayOfWeek_6': [0]})
```

The following below is a simple prompt that requests a query from the user, and responds with the estimated price for that flight.

In [126…

```
year = input("Enter the year (####): ")
month = input("Enter the month (1-12): ")
day = input("Enter the day (1-31): ")
airline = input("Enter the airline: ")
origin = input("Enter the origin: ")
destination = input("Enter the destination: ")
hour_of_day = int(input("Enter the hour of day (0-23): "))
```

```
Enter the year (####): 2024
Enter the month (1-12): 3
Enter the day (1-31): 15
Enter the airline: American Airlines
Enter the origin: Denver
Enter the destination: Orlando
Enter the hour of day (0-23): 11
```

Before I spit out the result prediction, I wanted to confirm a few details to make sure the model got the inputs correct.

Obviously there is no error-handling yet. If this was for a business I would definitely include some error-handling.

In [127…

```
input_datetime = year + '-' + make2digit(month) + '-' + make2digit(day)
input_datetime = pd.to_datetime(input_datetime)
input_ord_datetime = getordinal(input_datetime)
input_day_of_week = input_datetime.day_of_week
print(f"Selected {input_datetime}, Day of Week is {input_day_of_week}")
```

```
Selected 2024-03-15 00:00:00, Day of Week is 4
```

In [128…

```
print(type(input_day_of_week))
```

```
<class 'int'>
```

In [129…

```
airline = 'marktName_' + airline
print(airline)
```

```
marktName_American Airlines
```

In [130…

```
origin = 'orgName_' + origin
print(origin)
```

```
orgName_Denver
```

In [131…

```
destination = 'destName_' + destination
print(destination)
```

```
destName_Orlando
```

In [132…

```
input_day_of_week = 'deptDayOfWeek_' + str(input_day_of_week)
print(input_day_of_week)
```

```
deptDayOfWeek_4
```

The code block below is a BEAR. It basically takes the user's input and creates a datapoint from it to be submitted into the prediction mode. A LOOOOOOOT of boolean checks were used...

In [133…

```
df_input = pd.DataFrame({'deptDateTimeOrd': [input_ord_datetime],
                         'deptHour': [hour_of_day], 'deptMonth': [make2digit(month)],
                         'marktName_Aeromexico': [int('marktName_Aeromexico'==airline)],
                         'marktName_Air Canada': [int('marktName_Air Canada'==airline)],
                         'marktName_Air Transat': [int('marktName_Air Transat'==airline)],
                         'marktName_Alaska Airlines': [int('marktName_Alaska Airlines'==airline)],
                         'marktName_Allegiant Air': [int('marktName_Allegiant Air'==airline)],
                         'marktName_American Airlines': [int('marktName_American Airlines'==airline)],
                         'marktName_Breeze Airways': [int('marktName_Breeze Airways'==airline)],
                         'marktName_Canada Jetlines': [int('marktName_Canada Jetlines'==airline)],
                         'marktName_Flair Airlines': [int('marktName_Flair Airlines'==airline)],
                         'marktName_Frontier Airlines': [int('marktName_Frontier Airlines'==airline)],
                         'marktName_Key Lime Air': [int('marktName_Key Lime Air'==airline)],
                         'marktName_LATAM': [int('marktName_LATAM'==airline)],
                         'marktName_Lynx Air': [int('marktName_Lynx Air'==airline)],
                         'marktName_Porter Airlines': [int('marktName_Porter Airlines'==airline)],
                         'marktName_Silver Airways': [int('marktName_Silver Airways'==airline)],
                         'marktName_Spirit Airlines': [int('marktName_Spirit Airlines'==airline)],
                         'marktName_Sun Country': [int('marktName_Sun Country'==airline)],
                         'marktName_United Airlines': [int('marktName_United Airlines'==airline)],
                         'marktName_Volaris': [int('marktName_Volaris'==airline)],
                         'marktName_WestJet': [int('marktName_WestJet'==airline)],
                         'orgName_Chicago': [int('orgName_Chicago'==origin)],
                         'orgName_Cincinnati': [int('orgName_Cincinnati'==origin)],
                         'orgName_Denver': [int('orgName_Denver'==origin)],
                         'orgName_Los Angeles': [int('orgName_Los Angeles'==origin)],
                         'orgName_Miami': [int('orgName_Miami'==origin)],
                         'orgName_New York': [int('orgName_New York'==origin)],
                         'orgName_Orlando': [int('orgName_Orlando'==origin)],
                         'orgName_San Francisco': [int('orgName_San Francisco'==origin)],
                         'destName_Chicago': [int('destName_Chicago'==destination)],
                         'destName_Cincinnati': [int('destName_Cincinnati'==destination)],
                         'destName_Denver': [int('destName_Denver'==destination)],
                         'destName_Los Angeles': [int('destName_Los Angeles'==destination)],
                         'destName_Miami': [int('destName_Miami'==destination)],
                         'destName_New York': [int('destName_New York'==destination)],
                         'destName_Orlando': [int('destName_Orlando'==destination)],
                         'destName_San Francisco': [int('destName_San Francisco'==destination)],
                         'deptDayOfWeek_0': [int('deptDayOfWeek_0'==input_day_of_week)],
                         'deptDayOfWeek_1': [int('deptDayOfWeek_1'==input_day_of_week)],
                         'deptDayOfWeek_2': [int('deptDayOfWeek_2'==input_day_of_week)],
                         'deptDayOfWeek_3': [int('deptDayOfWeek_3'==input_day_of_week)],
                         'deptDayOfWeek_4': [int('deptDayOfWeek_4'==input_day_of_week)],
                         'deptDayOfWeek_5': [int('deptDayOfWeek_5'==input_day_of_week)],
                         'deptDayOfWeek_6': [int('deptDayOfWeek_6'==input_day_of_week)]})

print(df_input.iloc[0])
```

```
deptDateTimeOrd                738960
deptHour                           11
deptMonth                          03
marktName_Aeromexico                0
marktName_Air Canada                0
marktName_Air Transat               0
marktName_Alaska Airlines           0
marktName_Allegiant Air             0
marktName_American Airlines         1
marktName_Breeze Airways            0
marktName_Canada Jetlines           0
marktName_Flair Airlines            0
marktName_Frontier Airlines         0
marktName_Key Lime Air              0
marktName_LATAM                     0
marktName_Lynx Air                  0
marktName_Porter Airlines           0
marktName_Silver Airways            0
marktName_Spirit Airlines           0
marktName_Sun Country               0
marktName_United Airlines           0
marktName_Volaris                   0
marktName_WestJet                   0
orgName_Chicago                     0
orgName_Cincinnati                  0
orgName_Denver                      1
orgName_Los Angeles                 0
orgName_Miami                       0
orgName_New York                    0
orgName_Orlando                     0
orgName_San Francisco               0
destName_Chicago                    0
destName_Cincinnati                 0
destName_Denver                     0
destName_Los Angeles                0
destName_Miami                      0
destName_New York                   0
destName_Orlando                    1
destName_San Francisco              0
deptDayOfWeek_0                     0
deptDayOfWeek_1                     0
deptDayOfWeek_2                     0
deptDayOfWeek_3                     0
deptDayOfWeek_4                     1
deptDayOfWeek_5                     0
deptDayOfWeek_6                     0
Name: 0, dtype: object
```

In [134…
```
input_prediction = predict_model(model_rf2, data = df_input)
```

Aaaaaand the resulting prediction!

In [135…
```
print(input_prediction['prediction_label'])
```

```
0    325.616601
Name: prediction_label, dtype: float64
```

In [ ]:
```
```

Pycaret Model 3: Mktime Time, KFold Generator

Time for pycaret model 3! Still sticking with the KFold Generator of pycaret, but moving on with a different datetime format.

Now we'll use the time.mktime() datetime format, which takes into account the hour of day. Let's continue.

After this, we'll move on with the same 3 date comparisons and pycaret comparison but with a different generator instead of KFolds.

In [136…
```
features3 = df[['deptDateTimeMktime', 'deptDayOfWeek', 'deptMonth', 'marktName', 'orgName', 'destName']]
targets3 = df[['price']]
```

In [137…
```
features3_dum = pd.get_dummies(features3, columns=['marktName', 'orgName', 'destName', 'deptDayOfWeek'])
features3_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 45 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDateTimeMktime           44367 non-null  float64
 1   deptMonth                    44367 non-null  int64
 2   marktName_Aeromexico         44367 non-null  uint8
 3   marktName_Air Canada         44367 non-null  uint8
 4   marktName_Air Transat        44367 non-null  uint8
 5   marktName_Alaska Airlines    44367 non-null  uint8
 6   marktName_Allegiant Air      44367 non-null  uint8
 7   marktName_American Airlines  44367 non-null  uint8
 8   marktName_Breeze Airways     44367 non-null  uint8
 9   marktName_Canada Jetlines    44367 non-null  uint8
 10  marktName_Flair Airlines     44367 non-null  uint8
 11  marktName_Frontier Airlines  44367 non-null  uint8
```

```
 12  marktName_Key Lime Air       44367 non-null  uint8
 13  marktName_LATAM              44367 non-null  uint8
 14  marktName_Lynx Air           44367 non-null  uint8
 15  marktName_Porter Airlines    44367 non-null  uint8
 16  marktName_Silver Airways     44367 non-null  uint8
 17  marktName_Spirit Airlines    44367 non-null  uint8
 18  marktName_Sun Country        44367 non-null  uint8
 19  marktName_United Airlines    44367 non-null  uint8
 20  marktName_Volaris            44367 non-null  uint8
 21  marktName_WestJet            44367 non-null  uint8
 22  orgName_Chicago              44367 non-null  uint8
 23  orgName_Cincinnati           44367 non-null  uint8
 24  orgName_Denver               44367 non-null  uint8
 25  orgName_Los Angeles          44367 non-null  uint8
 26  orgName_Miami                44367 non-null  uint8
 27  orgName_New York             44367 non-null  uint8
 28  orgName_Orlando              44367 non-null  uint8
 29  orgName_San Francisco        44367 non-null  uint8
 30  destName_Chicago             44367 non-null  uint8
 31  destName_Cincinnati          44367 non-null  uint8
 32  destName_Denver              44367 non-null  uint8
 33  destName_Los Angeles         44367 non-null  uint8
 34  destName_Miami               44367 non-null  uint8
 35  destName_New York            44367 non-null  uint8
 36  destName_Orlando             44367 non-null  uint8
 37  destName_San Francisco       44367 non-null  uint8
 38  deptDayOfWeek_0              44367 non-null  uint8
 39  deptDayOfWeek_1              44367 non-null  uint8
 40  deptDayOfWeek_2              44367 non-null  uint8
 41  deptDayOfWeek_3              44367 non-null  uint8
 42  deptDayOfWeek_4              44367 non-null  uint8
 43  deptDayOfWeek_5              44367 non-null  uint8
 44  deptDayOfWeek_6              44367 non-null  uint8
dtypes: float64(1), int64(1), uint8(43)
memory usage: 3.8 MB
```

In [138…

```
df_dum3 = pd.concat([features3_dum, targets3], axis='columns')
```

In [139…

```
x_train3, x_test3, y_train3, y_test3 = train_test_split(features3_dum, targets3, test_size=0.22, random_state=23)
```

In [140…

```
reg3 = setup(data = df_dum3, target = 'price', session_id=23)
```

	Description	Value
0	Session id	23
1	Target	price

	Description	Value
2	Target type	Regression
3	Original data shape	(44367, 46)
4	Transformed data shape	(44367, 46)
5	Transformed train set shape	(31056, 46)
6	Transformed test set shape	(13311, 46)
7	Numeric features	45
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	KFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	reg-default-name

In [141…

```
best3 = compare_models(n_select=4)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
rf	Random Forest Regressor	27.4715	2334.3253	48.1705	0.7702	0.2005	0.1359	4.4140
et	Extra Trees Regressor	27.2594	2723.4477	51.9749	0.7320	0.2148	0.1354	4.1190
lightgbm	Light Gradient Boosting Machine	35.1553	2887.8201	53.6906	0.7148	0.2264	0.1780	0.2300
dt	Decision Tree Regressor	31.0585	3564.6958	59.4945	0.6488	0.2447	0.1502	0.1180
gbr	Gradient Boosting Regressor	45.6520	4498.9321	67.0402	0.5558	0.2851	0.2389	1.5720
ridge	Ridge Regression	53.8542	6130.6340	78.2690	0.3944	0.3546	0.2979	0.0460
lasso	Lasso Regression	58.1272	6978.4756	83.4990	0.3111	0.3743	0.3268	0.1050
llar	Lasso Least Angle Regression	58.1272	6978.4756	83.4990	0.3111	0.3743	0.3268	0.0470

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
knn	K Neighbors Regressor	59.5802	7644.7999	87.4036	0.2450	0.3809	0.3222	0.2570
br	Bayesian Ridge	63.7200	8185.8572	90.4386	0.1919	0.4149	0.3750	0.0610
en	Elastic Net	67.5333	8974.5053	94.6926	0.1141	0.4304	0.3945	0.1460
omp	Orthogonal Matching Pursuit	68.7096	9311.4683	96.4267	0.0813	0.4393	0.4023	0.0440
lr	Linear Regression	70.0103	9537.0643	97.6175	0.0585	0.4447	0.4102	0.0550
dummy	Dummy Regressor	72.5853	10133.5062	100.6225	-0.0003	0.4723	0.4483	0.0430
huber	Huber Regressor	69.9112	10372.6277	101.8013	-0.0238	0.4596	0.3983	0.1210
ada	AdaBoost Regressor	132.9298	22775.3865	148.8880	-1.2548	0.6823	0.9267	0.9310
par	Passive Aggressive Regressor	107.4685	23212.1547	135.1897	-1.2950	0.5962	0.6468	0.0790

In [ ]:

Pycaret Model 4: Full Datetime, TimeSeries Generator

Now let's do Time Series! I will perform the same 3 model comparison using pycaret, as well as adding in 2 new datetime formats. This time the comparisons will be made with pycaret's time-series parameter.

The formats that I will add in are the full datetime field which includes Year/Month/Day as well as Hour and Minute. I will also add a comparison of the datetime with just Year/Month/Day, and provide Hour as a separate feature.

What does our data look like again?

In [142…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   destName            44367 non-null  object        
 1   orgName             44367 non-null  object        
 2   deptDateTime        44367 non-null  datetime64[ns]
 3   arrvDateTime        44367 non-null  datetime64[ns]
 4   price               44367 non-null  float64       
 5   marktName           44367 non-null  object        
 6   flightNum           44367 non-null  object        
 7   duration            44367 non-null  float64       
```

```
 8   numConex            44367 non-null  int64
 9   url                 44367 non-null  object
 10  deptDayOfWeek       44367 non-null  int64
 11  arrvDayOfWeek       44367 non-null  int64
 12  deptMonth           44367 non-null  int64
 13  deptDate            44367 non-null  datetime64[ns]
 14  arrvDate            44367 non-null  datetime64[ns]
 15  deptDayOfYear       44367 non-null  int64
 16  deptHour            44367 non-null  int64
 17  deptDayOfMonth      44367 non-null  int64
 18  deptDateTimeOrd     44367 non-null  int64
 19  deptDateTimeMktime  44367 non-null  float64
dtypes: datetime64[ns](4), float64(3), int64(8), object(5)
```

In [143…

```
df.head()
```

Out[143…

	destName	orgName	deptDateTime	arrvDateTime	price	marktName	flightNum	duration	numConex	url	deptDayOfWeek	arrvDayOf
0	Cincinnati	Denver	2023-11-20 20:46:00	2023-11-21 01:29:00	55.29	Frontier Airlines	F94128	163.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
1	Cincinnati	Denver	2023-11-20 09:40:00	2023-11-20 14:15:00	95.16	Allegiant Air	G4406	155.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
2	Cincinnati	Denver	2023-11-20 05:55:00	2023-11-20 10:38:00	96.45	Frontier Airlines	F93286	163.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
3	Cincinnati	Denver	2023-11-20 17:50:00	2023-11-20 22:37:00	138.59	United Airlines	UA1830	167.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	
4	Cincinnati	Denver	2023-11-20 09:42:00	2023-11-20 14:29:00	186.79	United Airlines	UA1978	167.0	1	https://flights-us.gotogate.com/air/DENCVG20NO...	0	

For this time series, we want the following:

Features:

- destName
- orgName
- marktName

- deptDateTime
- deptDayOfWeek

Targets:

- price

I think that's it for now

In [144…

```
time_features = df[['deptDayOfWeek', 'deptDateTime', 'marktName', 'orgName', 'destName']]
time_targets = df[['price']]
```

In [145…

```
time_features_dum = pd.get_dummies(time_features, columns=['deptDayOfWeek', 'marktName', 'orgName', 'destName'])
time_features_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 44 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDateTime                 44367 non-null  datetime64[ns]
 1   deptDayOfWeek_0              44367 non-null  uint8
 2   deptDayOfWeek_1              44367 non-null  uint8
 3   deptDayOfWeek_2              44367 non-null  uint8
 4   deptDayOfWeek_3              44367 non-null  uint8
 5   deptDayOfWeek_4              44367 non-null  uint8
 6   deptDayOfWeek_5              44367 non-null  uint8
 7   deptDayOfWeek_6              44367 non-null  uint8
 8   marktName_Aeromexico         44367 non-null  uint8
 9   marktName_Air Canada         44367 non-null  uint8
 10  marktName_Air Transat        44367 non-null  uint8
 11  marktName_Alaska Airlines    44367 non-null  uint8
 12  marktName_Allegiant Air      44367 non-null  uint8
 13  marktName_American Airlines  44367 non-null  uint8
 14  marktName_Breeze Airways     44367 non-null  uint8
 15  marktName_Canada Jetlines    44367 non-null  uint8
 16  marktName_Flair Airlines     44367 non-null  uint8
 17  marktName_Frontier Airlines  44367 non-null  uint8
 18  marktName_Key Lime Air       44367 non-null  uint8
 19  marktName_LATAM              44367 non-null  uint8
 20  marktName_Lynx Air           44367 non-null  uint8
 21  marktName_Porter Airlines    44367 non-null  uint8
 22  marktName_Silver Airways     44367 non-null  uint8
 23  marktName_Spirit Airlines    44367 non-null  uint8
 24  marktName_Sun Country        44367 non-null  uint8
 25  marktName_United Airlines    44367 non-null  uint8
```

```
 26  marktName_Volaris            44367 non-null  uint8
 27  marktName_WestJet            44367 non-null  uint8
 28  orgName_Chicago              44367 non-null  uint8
 29  orgName_Cincinnati           44367 non-null  uint8
 30  orgName_Denver               44367 non-null  uint8
 31  orgName_Los Angeles          44367 non-null  uint8
 32  orgName_Miami                44367 non-null  uint8
 33  orgName_New York             44367 non-null  uint8
 34  orgName_Orlando              44367 non-null  uint8
 35  orgName_San Francisco        44367 non-null  uint8
 36  destName_Chicago             44367 non-null  uint8
 37  destName_Cincinnati          44367 non-null  uint8
 38  destName_Denver              44367 non-null  uint8
 39  destName_Los Angeles         44367 non-null  uint8
 40  destName_Miami               44367 non-null  uint8
 41  destName_New York            44367 non-null  uint8
 42  destName_Orlando             44367 non-null  uint8
 43  destName_San Francisco       44367 non-null  uint8
dtypes: datetime64[ns](1), uint8(43)
```

In [146…

```
x_train, x_test, y_train, y_test = train_test_split(time_features_dum, time_targets, test_size=0.22, shuffle=False, random_state=23)
```

In [147…

```
df_time_dum = pd.concat([time_features_dum, time_targets], axis='columns')
df_test = pd.concat([x_test, y_test], axis='columns')
df_train = pd.concat([x_train, y_train], axis='columns')
```

In [148…

```
df_time_dum.head()
```

Out[148…

	deptDateTime	deptDayOfWeek_0	deptDayOfWeek_1	deptDayOfWeek_2	deptDayOfWeek_3	deptDayOfWeek_4	deptDayOfWeek_5	deptDayOfWeek_6	ma
0	2023-11-20 20:46:00	1	0	0	0	0	0	0	
1	2023-11-20 09:40:00	1	0	0	0	0	0	0	
2	2023-11-20 05:55:00	1	0	0	0	0	0	0	
3	2023-11-20 17:50:00	1	0	0	0	0	0	0	
4	2023-11-20 09:42:00	1	0	0	0	0	0	0	

5 rows × 45 columns

In [149…

```
df_time_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 45 columns):
 #   Column                       Non-Null Count  Dtype         
---  ------                       --------------  -----         
 0   deptDateTime                 44367 non-null  datetime64[ns]
 1   deptDayOfWeek_0              44367 non-null  uint8         
 2   deptDayOfWeek_1              44367 non-null  uint8         
 3   deptDayOfWeek_2              44367 non-null  uint8         
 4   deptDayOfWeek_3              44367 non-null  uint8         
 5   deptDayOfWeek_4              44367 non-null  uint8         
 6   deptDayOfWeek_5              44367 non-null  uint8         
 7   deptDayOfWeek_6              44367 non-null  uint8         
 8   marktName_Aeromexico         44367 non-null  uint8         
 9   marktName_Air Canada         44367 non-null  uint8         
 10  marktName_Air Transat        44367 non-null  uint8         
 11  marktName_Alaska Airlines    44367 non-null  uint8         
 12  marktName_Allegiant Air      44367 non-null  uint8         
 13  marktName_American Airlines  44367 non-null  uint8         
 14  marktName_Breeze Airways     44367 non-null  uint8         
 15  marktName_Canada Jetlines    44367 non-null  uint8         
 16  marktName_Flair Airlines     44367 non-null  uint8         
 17  marktName_Frontier Airlines  44367 non-null  uint8         
 18  marktName_Key Lime Air       44367 non-null  uint8         
 19  marktName_LATAM              44367 non-null  uint8         
 20  marktName_Lynx Air           44367 non-null  uint8         
 21  marktName_Porter Airlines    44367 non-null  uint8         
 22  marktName_Silver Airways     44367 non-null  uint8         
 23  marktName_Spirit Airlines    44367 non-null  uint8         
 24  marktName_Sun Country        44367 non-null  uint8         
 25  marktName_United Airlines    44367 non-null  uint8         
 26  marktName_Volaris            44367 non-null  uint8         
 27  marktName_WestJet            44367 non-null  uint8         
 28  orgName_Chicago              44367 non-null  uint8         
 29  orgName_Cincinnati           44367 non-null  uint8         
 30  orgName_Denver               44367 non-null  uint8         
 31  orgName_Los Angeles          44367 non-null  uint8         
 32  orgName_Miami                44367 non-null  uint8         
 33  orgName_New York             44367 non-null  uint8         
 34  orgName_Orlando              44367 non-null  uint8         
 35  orgName_San Francisco        44367 non-null  uint8         
 36  destName_Chicago             44367 non-null  uint8         
 37  destName_Cincinnati          44367 non-null  uint8         
 38  destName_Denver              44367 non-null  uint8         
 39  destName_Los Angeles         44367 non-null  uint8         
 40  destName_Miami               44367 non-null  uint8         
```

```
 41  destName_New York            44367 non-null  uint8
 42  destName_Orlando             44367 non-null  uint8
 43  destName_San Francisco       44367 non-null  uint8
 44  price                        44367 non-null  float64
dtypes: datetime64[ns](1), float64(1), uint8(43)
memory usage: 3.8 MB
```

In [150…

```
s = setup(data = df_train, test_data = df_test, target = 'price', fold_strategy = 'timeseries', fold = 10, transform_target = True,
#s = setup(data = df_time_dum, target = 'price', fold_strategy = 'timeseries', fold = 10, session_id = 23, data_split_shuffle=False,
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 45)
4	Transformed data shape	(44367, 47)
5	Transformed train set shape	(34606, 47)
6	Transformed test set shape	(9761, 47)
7	Numeric features	43
8	Date features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Transform target	True
14	Transform target method	yeo-johnson
15	Fold Generator	TimeSeriesSplit
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	reg-default-name
21	USI	f353

In [151…

```
best = compare_models()
```

	Model	MAE
rf	Random Forest Regressor	55.7333
gbr	Gradient Boosting Regressor	55.9773
et	Extra Trees Regressor	56.3187
lightgbm	Light Gradient Boosting Machine	55.7450
dt	Decision Tree Regressor	59.6344
ridge	Ridge Regression	62.3258
br	Bayesian Ridge	62.5475
knn	K Neighbors Regressor	61.5365
huber	Huber Regressor	70.5166
omp	Orthogonal Matching Pursuit	71.4797
ada	AdaBoost Regressor	70.5767
en	Elastic Net	76.2189
lasso	Lasso Regression	76.3882

	Model	MAE
llar	Lasso Least Angle Regression	76.3882
dummy	Dummy Regressor	76.4115
par	Passive Aggressive Regressor	88.0796
lr	Linear Regression	3497273.5542

In [152…

```
prediction_holdout = predict_model(best);
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Random Forest Regressor	52.2042	4549.1522	67.4474	0.2025	0.3474	0.3478

In [153…

```
df.reset_index(inplace=True)
```

In [154…

```
df_time_dum.head()
```

Out[154…

	deptDateTime	deptDayOfWeek_0	deptDayOfWeek_1	deptDayOfWeek_2	deptDayOfWeek_3	deptDayOfWeek_4	deptDayOfWeek_5	deptDayOfWeek_6	ma
0	2023-11-20 20:46:00	1	0	0	0	0	0	0	
1	2023-11-20 09:40:00	1	0	0	0	0	0	0	
2	2023-11-20 05:55:00	1	0	0	0	0	0	0	
3	2023-11-20 17:50:00	1	0	0	0	0	0	0	
4	2023-11-20 09:42:00	1	0	0	0	0	0	0	

5 rows × 45 columns

In [155…

```
#s = setup(data = df_time_dum, target = 'price', fold_strategy = 'timeseries', fold = 10, session_id = 23, data_split_shuffle=False,
# s = setup(data = df_train, target = 'price', fold_strategy = 'timeseries', fold = 10, transform_target = True, session_id = 123)
#s = setup(data = df_train, target = 'price', fold = 10, transform_target = True, session_id = 123)
```

In [156…

```
#best = compare_models()
```

Pycaret Model 5: DateTime Format #2, TimeSeries Generator

Now we'll use just the 'date' part of the departure datetime and include the hour as a separate field. Still using the timeseries generator of pycaret!

In [157…

```
df.head()
```

Out[157…

	index	destName	orgName	deptDateTime	arrvDateTime	price	marktName	flightNum	duration	numConex	...	deptDayOfWeek	arrvDayOfWeek	d
0	0	Cincinnati	Denver	2023-11-20 20:46:00	2023-11-21 01:29:00	55.29	Frontier Airlines	F94128	163.0	1	...	0	1	
1	1	Cincinnati	Denver	2023-11-20 09:40:00	2023-11-20 14:15:00	95.16	Allegiant Air	G4406	155.0	1	...	0	0	
2	2	Cincinnati	Denver	2023-11-20 05:55:00	2023-11-20 10:38:00	96.45	Frontier Airlines	F93286	163.0	1	...	0	0	
3	3	Cincinnati	Denver	2023-11-20 17:50:00	2023-11-20 22:37:00	138.59	United Airlines	UA1830	167.0	1	...	0	0	
4	4	Cincinnati	Denver	2023-11-20 09:42:00	2023-11-20 14:29:00	186.79	United Airlines	UA1978	167.0	1	...	0	0	

5 rows × 21 columns

In [158…

```
time2_features = df[['deptDayOfWeek', 'deptDate', 'deptHour', 'marktName', 'orgName', 'destName']]
time2_targets = df[['price']]
```

In [159…

```
time2_features_dum = pd.get_dummies(time2_features, columns=['deptDayOfWeek', 'marktName', 'orgName', 'destName'])
time2_features_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44367 entries, 0 to 44366
Data columns (total 45 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   deptDate                    44367 non-null  datetime64[ns]
```

```
 1   deptHour                    44367 non-null  int64
 2   deptDayOfWeek_0             44367 non-null  uint8
 3   deptDayOfWeek_1             44367 non-null  uint8
 4   deptDayOfWeek_2             44367 non-null  uint8
 5   deptDayOfWeek_3             44367 non-null  uint8
 6   deptDayOfWeek_4             44367 non-null  uint8
 7   deptDayOfWeek_5             44367 non-null  uint8
 8   deptDayOfWeek_6             44367 non-null  uint8
 9   marktName_Aeromexico        44367 non-null  uint8
 10  marktName_Air Canada        44367 non-null  uint8
 11  marktName_Air Transat       44367 non-null  uint8
 12  marktName_Alaska Airlines   44367 non-null  uint8
 13  marktName_Allegiant Air     44367 non-null  uint8
 14  marktName_American Airlines 44367 non-null  uint8
 15  marktName_Breeze Airways    44367 non-null  uint8
 16  marktName_Canada Jetlines   44367 non-null  uint8
 17  marktName_Flair Airlines    44367 non-null  uint8
 18  marktName_Frontier Airlines 44367 non-null  uint8
 19  marktName_Key Lime Air      44367 non-null  uint8
 20  marktName_LATAM             44367 non-null  uint8
 21  marktName_Lynx Air          44367 non-null  uint8
 22  marktName_Porter Airlines   44367 non-null  uint8
 23  marktName_Silver Airways    44367 non-null  uint8
 24  marktName_Spirit Airlines   44367 non-null  uint8
 25  marktName_Sun Country       44367 non-null  uint8
 26  marktName_United Airlines   44367 non-null  uint8
 27  marktName_Volaris           44367 non-null  uint8
 28  marktName_WestJet           44367 non-null  uint8
 29  orgName_Chicago             44367 non-null  uint8
 30  orgName_Cincinnati          44367 non-null  uint8
 31  orgName_Denver              44367 non-null  uint8
 32  orgName_Los Angeles         44367 non-null  uint8
 33  orgName_Miami               44367 non-null  uint8
 34  orgName_New York            44367 non-null  uint8
 35  orgName_Orlando             44367 non-null  uint8
 36  orgName_San Francisco       44367 non-null  uint8
 37  destName_Chicago            44367 non-null  uint8
 38  destName_Cincinnati         44367 non-null  uint8
 39  destName_Denver             44367 non-null  uint8
 40  destName_Los Angeles        44367 non-null  uint8
 41  destName_Miami              44367 non-null  uint8
 42  destName_New York           44367 non-null  uint8
 43  destName_Orlando            44367 non-null  uint8
 44  destName_San Francisco      44367 non-null  uint8
dtypes: datetime64[ns](1), int64(1), uint8(43)
memory usage: 2.5 MB
```

In [160…

```
x_train, x_test, y_train, y_test = train_test_split(time2_features_dum, time2_targets, test_size=0.22, shuffle=False, random_state=
```

In [161…

```
df_time2_dum = pd.concat([time2_features_dum, time2_targets], axis='columns')
df_test = pd.concat([x_test, y_test], axis='columns')
df_train = pd.concat([x_train, y_train], axis='columns')
```

In [162…

```
df_time2_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44367 entries, 0 to 44366
Data columns (total 46 columns):
 #   Column                       Non-Null Count  Dtype         
---  ------                       --------------  -----         
 0   deptDate                     44367 non-null  datetime64[ns]
 1   deptHour                     44367 non-null  int64         
 2   deptDayOfWeek_0              44367 non-null  uint8         
 3   deptDayOfWeek_1              44367 non-null  uint8         
 4   deptDayOfWeek_2              44367 non-null  uint8         
 5   deptDayOfWeek_3              44367 non-null  uint8         
 6   deptDayOfWeek_4              44367 non-null  uint8         
 7   deptDayOfWeek_5              44367 non-null  uint8         
 8   deptDayOfWeek_6              44367 non-null  uint8         
 9   marktName_Aeromexico         44367 non-null  uint8         
 10  marktName_Air Canada         44367 non-null  uint8         
 11  marktName_Air Transat        44367 non-null  uint8         
 12  marktName_Alaska Airlines    44367 non-null  uint8         
 13  marktName_Allegiant Air      44367 non-null  uint8         
 14  marktName_American Airlines  44367 non-null  uint8         
 15  marktName_Breeze Airways     44367 non-null  uint8         
 16  marktName_Canada Jetlines    44367 non-null  uint8         
 17  marktName_Flair Airlines     44367 non-null  uint8         
 18  marktName_Frontier Airlines  44367 non-null  uint8         
 19  marktName_Key Lime Air       44367 non-null  uint8         
 20  marktName_LATAM              44367 non-null  uint8         
 21  marktName_Lynx Air           44367 non-null  uint8         
 22  marktName_Porter Airlines    44367 non-null  uint8         
 23  marktName_Silver Airways     44367 non-null  uint8         
 24  marktName_Spirit Airlines    44367 non-null  uint8         
 25  marktName_Sun Country        44367 non-null  uint8         
 26  marktName_United Airlines    44367 non-null  uint8         
 27  marktName_Volaris            44367 non-null  uint8         
 28  marktName_WestJet            44367 non-null  uint8         
 29  orgName_Chicago              44367 non-null  uint8         
 30  orgName_Cincinnati           44367 non-null  uint8         
 31  orgName_Denver               44367 non-null  uint8         
 32  orgName_Los Angeles          44367 non-null  uint8         
 33  orgName_Miami                44367 non-null  uint8         
 34  orgName_New York             44367 non-null  uint8         
 35  orgName_Orlando              44367 non-null  uint8         
```

```
 36  orgName_San Francisco       44367 non-null  uint8
 37  destName_Chicago            44367 non-null  uint8
 38  destName_Cincinnati         44367 non-null  uint8
 39  destName_Denver             44367 non-null  uint8
 40  destName_Los Angeles        44367 non-null  uint8
 41  destName_Miami              44367 non-null  uint8
 42  destName_New York           44367 non-null  uint8
 43  destName_Orlando            44367 non-null  uint8
 44  destName_San Francisco      44367 non-null  uint8
 45  price                       44367 non-null  float64
dtypes: datetime64[ns](1), float64(1), int64(1), uint8(43)
```

In [163…

```
s = setup(data = df_train, test_data = df_test, target = 'price', fold_strategy = 'timeseries', fold = 10, transform_target = True,
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 46)
4	Transformed data shape	(44367, 48)
5	Transformed train set shape	(34606, 48)
6	Transformed test set shape	(9761, 48)
7	Numeric features	44
8	Date features	1
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Transform target	True
14	Transform target method	yeo-johnson
15	Fold Generator	TimeSeriesSplit
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False

	Description	Value
19	Log Experiment	False
20	Experiment Name	reg-default-name

In [164…

```
best = compare_models()
```

	Model	MAE
rf	Random Forest Regressor	55.8753
et	Extra Trees Regressor	56.7102
gbr	Gradient Boosting Regressor	56.9842
lightgbm	Light Gradient Boosting Machine	55.8252
ridge	Ridge Regression	62.3959
lr	Linear Regression	62.5843
br	Bayesian Ridge	62.6950
knn	K Neighbors Regressor	63.7537
huber	Huber Regressor	70.9816
omp	Orthogonal Matching Pursuit	72.4710
dt	Decision Tree Regressor	64.8410

	Model	MAE
ada	AdaBoost Regressor	71.7446
en	Elastic Net	76.2189
lasso	Lasso Regression	76.3882
llar	Lasso Least Angle Regression	76.3882
dummy	Dummy Regressor	76.4115
par	Passive Aggressive Regressor	89.8085

Pycaret Model 6: Parsed Datetime, TimeSeries Generator

After the 2 new datetime formats submitted to pycaret above, we'll now go back to the standard comparisons we used with the KFolds generator. We'll start with the parsed datetime which has separate fields for Year, Month, Day of Month, Hour, and more.

In [165…

```
df_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 48 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   deptDayOfMonth               44367 non-null  int64
 1   deptDayOfYear                44367 non-null  int64
 2   deptHour                     44367 non-null  int64
 3   deptMonth                    44367 non-null  int64
 4   marktName_Aeromexico         44367 non-null  uint8
 5   marktName_Air Canada         44367 non-null  uint8
 6   marktName_Air Transat        44367 non-null  uint8
 7   marktName_Alaska Airlines    44367 non-null  uint8
 8   marktName_Allegiant Air      44367 non-null  uint8
 9   marktName_American Airlines  44367 non-null  uint8
 10  marktName_Breeze Airways     44367 non-null  uint8
 11  marktName_Canada Jetlines    44367 non-null  uint8
 12  marktName_Flair Airlines     44367 non-null  uint8
 13  marktName_Frontier Airlines  44367 non-null  uint8
 14  marktName_Key Lime Air       44367 non-null  uint8
```

```
 15  marktName_LATAM              44367 non-null  uint8
 16  marktName_Lynx Air           44367 non-null  uint8
 17  marktName_Porter Airlines    44367 non-null  uint8
 18  marktName_Silver Airways     44367 non-null  uint8
 19  marktName_Spirit Airlines    44367 non-null  uint8
 20  marktName_Sun Country        44367 non-null  uint8
 21  marktName_United Airlines    44367 non-null  uint8
 22  marktName_Volaris            44367 non-null  uint8
 23  marktName_WestJet            44367 non-null  uint8
 24  orgName_Chicago              44367 non-null  uint8
 25  orgName_Cincinnati           44367 non-null  uint8
 26  orgName_Denver               44367 non-null  uint8
 27  orgName_Los Angeles          44367 non-null  uint8
 28  orgName_Miami                44367 non-null  uint8
 29  orgName_New York             44367 non-null  uint8
 30  orgName_Orlando              44367 non-null  uint8
 31  orgName_San Francisco        44367 non-null  uint8
 32  destName_Chicago             44367 non-null  uint8
 33  destName_Cincinnati          44367 non-null  uint8
 34  destName_Denver              44367 non-null  uint8
 35  destName_Los Angeles         44367 non-null  uint8
 36  destName_Miami               44367 non-null  uint8
 37  destName_New York            44367 non-null  uint8
 38  destName_Orlando             44367 non-null  uint8
 39  destName_San Francisco       44367 non-null  uint8
 40  deptDayOfWeek_0              44367 non-null  uint8
 41  deptDayOfWeek_1              44367 non-null  uint8
 42  deptDayOfWeek_2              44367 non-null  uint8
 43  deptDayOfWeek_3              44367 non-null  uint8
 44  deptDayOfWeek_4              44367 non-null  uint8
 45  deptDayOfWeek_5              44367 non-null  uint8
 46  deptDayOfWeek_6              44367 non-null  uint8
 47  price                        44367 non-null  float64
dtypes: float64(1), int64(4), uint8(43)
memory usage: 4.9 MB
```

In [166…

```
df_dum.head()
```

Out[166…

	deptDayOfMonth	deptDayOfYear	deptHour	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Alleg
0	20	324	20	11	0	0	0	0	
1	20	324	9	11	0	0	0	0	
2	20	324	5	11	0	0	0	0	
3	20	324	17	11	0	0	0	0	
4	20	324	9	11	0	0	0	0	

5 rows × 48 columns

In [167…

```
x_train, x_test, y_train, y_test = train_test_split(features_dum, targets, test_size=0.22, random_state=23)
df_train = pd.concat([x_train, y_train], axis='columns')
df_test = pd.concat([x_test, y_test], axis='columns')
```

In [168…

```
s = setup(data = df_train, test_data = df_test, target = 'price', fold_strategy = 'timeseries', fold = 10, transform_target = True,
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 48)
4	Transformed data shape	(44367, 48)
5	Transformed train set shape	(34606, 48)
6	Transformed test set shape	(9761, 48)
7	Numeric features	47
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Transform target	True
13	Transform target method	yeo-johnson
14	Fold Generator	TimeSeriesSplit
15	Fold Number	10
16	CPU Jobs	-1
17	Use GPU	False
18	Log Experiment	False
19	Experiment Name	reg-default-name
20	USI	0d9a

In [169…

```
best = compare_models()
```

	Model	MAE	MSE
rf	Random Forest Regressor	26.5741	2297.6356
et	Extra Trees Regressor	25.9583	2304.5764
lightgbm	Light Gradient Boosting Machine	33.0856	2962.2133
dt	Decision Tree Regressor	32.2413	4075.8848
gbr	Gradient Boosting Regressor	43.4367	4662.3930
lr	Linear Regression	52.8133	6321.0233
br	Bayesian Ridge	52.8024	6323.4155
ridge	Ridge Regression	52.8029	6326.7638
knn	K Neighbors Regressor	51.6084	6330.3771
huber	Huber Regressor	58.3140	8547.2933
ada	AdaBoost Regressor	73.8903	8708.6776
omp	Orthogonal Matching Pursuit	66.9577	9795.9759
en	Elastic Net	68.9775	10268.462

	Model	MAE	MSE
lasso	Lasso Regression	69.0206	10275.124
llar	Lasso Least Angle Regression	69.0206	10275.124
dummy	Dummy Regressor	70.2034	10432.699
par	Passive Aggressive	124.0402	34387.131

In [ ]:

Pycaret Model 7: Ordinate Datetime, TimeSeries Generator

Moving now onto the ordinate datetime! As a reminder this is an integer value which takes into account Year/Month/Day. I'll include hour as a separate feature.

In [170…

```
df_dum2.head()
```

Out[170…

	deptDateTimeOrd	deptHour	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Allegiant Air	marktNam
0	738844	20	11	0	0	0	0	0	
1	738844	9	11	0	0	0	0	1	
2	738844	5	11	0	0	0	0	0	
3	738844	17	11	0	0	0	0	0	
4	738844	9	11	0	0	0	0	0	

5 rows × 47 columns

In [171…

```
df_dum2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 47 columns):
 #   Column                      Non-Null Count  Dtype
```

```
---  ------                        --------------  -----
 0   deptDateTimeOrd               44367 non-null  int64
 1   deptHour                      44367 non-null  int64
 2   deptMonth                     44367 non-null  int64
 3   marktName_Aeromexico          44367 non-null  uint8
 4   marktName_Air Canada          44367 non-null  uint8
 5   marktName_Air Transat         44367 non-null  uint8
 6   marktName_Alaska Airlines     44367 non-null  uint8
 7   marktName_Allegiant Air       44367 non-null  uint8
 8   marktName_American Airlines   44367 non-null  uint8
 9   marktName_Breeze Airways      44367 non-null  uint8
 10  marktName_Canada Jetlines     44367 non-null  uint8
 11  marktName_Flair Airlines      44367 non-null  uint8
 12  marktName_Frontier Airlines   44367 non-null  uint8
 13  marktName_Key Lime Air        44367 non-null  uint8
 14  marktName_LATAM               44367 non-null  uint8
 15  marktName_Lynx Air            44367 non-null  uint8
 16  marktName_Porter Airlines     44367 non-null  uint8
 17  marktName_Silver Airways      44367 non-null  uint8
 18  marktName_Spirit Airlines     44367 non-null  uint8
 19  marktName_Sun Country         44367 non-null  uint8
 20  marktName_United Airlines     44367 non-null  uint8
 21  marktName_Volaris             44367 non-null  uint8
 22  marktName_WestJet             44367 non-null  uint8
 23  orgName_Chicago               44367 non-null  uint8
 24  orgName_Cincinnati            44367 non-null  uint8
 25  orgName_Denver                44367 non-null  uint8
 26  orgName_Los Angeles           44367 non-null  uint8
 27  orgName_Miami                 44367 non-null  uint8
 28  orgName_New York              44367 non-null  uint8
 29  orgName_Orlando               44367 non-null  uint8
 30  orgName_San Francisco         44367 non-null  uint8
 31  destName_Chicago              44367 non-null  uint8
 32  destName_Cincinnati           44367 non-null  uint8
 33  destName_Denver               44367 non-null  uint8
 34  destName_Los Angeles          44367 non-null  uint8
 35  destName_Miami                44367 non-null  uint8
 36  destName_New York             44367 non-null  uint8
 37  destName_Orlando              44367 non-null  uint8
 38  destName_San Francisco        44367 non-null  uint8
 39  deptDayOfWeek_0               44367 non-null  uint8
 40  deptDayOfWeek_1               44367 non-null  uint8
 41  deptDayOfWeek_2               44367 non-null  uint8
 42  deptDayOfWeek_3               44367 non-null  uint8
 43  deptDayOfWeek_4               44367 non-null  uint8
 44  deptDayOfWeek_5               44367 non-null  uint8
 45  deptDayOfWeek_6               44367 non-null  uint8
 46  price                         44367 non-null  float64
dtypes: float64(1), int64(3), uint8(43)
memory usage: 4.5 MB
```

In [172…

```
x_train2, x_test2, y_train2, y_test2 = train_test_split(features2_dum, targets2, test_size=0.22, random_state=23)
```

In [173…

```
df_train2 = pd.concat([x_train2, y_train2], axis='columns')
df_test2 = pd.concat([x_test2, y_test2], axis='columns')
```

In [174…

```
s = setup(data = df_train2, test_data = df_test2, target = 'price', fold_strategy = 'timeseries', fold = 10, transform_target = True
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 47)
4	Transformed data shape	(44367, 47)
5	Transformed train set shape	(34606, 47)
6	Transformed test set shape	(9761, 47)
7	Numeric features	46
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Transform target	True
13	Transform target method	yeo-johnson
14	Fold Generator	TimeSeriesSplit
15	Fold Number	10
16	CPU Jobs	-1
17	Use GPU	False
18	Log Experiment	False
19	Experiment Name	reg-default-name
20	USI	efd0

In [175…

```
best = compare_models()
```

	Model	MAE	MSE
rf	Random Forest Regressor	25.9359	2208
et	Extra Trees Regressor	25.6388	2298
lightgbm	Light Gradient Boosting Machine	32.6982	2907
dt	Decision Tree Regressor	31.2763	4008
gbr	Gradient Boosting Regressor	43.3481	4650
knn	K Neighbors Regressor	50.3086	6103
lr	Linear Regression	52.3964	6365
ridge	Ridge Regression	52.4116	6365
br	Bayesian Ridge	52.4198	6366
ada	AdaBoost Regressor	75.6697	8970
omp	Orthogonal Matching Pursuit	64.7209	9378
lasso	Lasso Regression	67.0633	9861
llar	Lasso Least Angle Regression	67.0633	9861

	Model	MAE	MSE
en	Elastic Net	67.0833	9863
dummy	Dummy Regressor	70.2034	1043
huber	Huber Regressor	70.1925	1044
par	Passive Aggressive Regressor	77.4290	1275
	Least		

In [ ]:

Pycaret Model 8: Mktime Time, TimeSeries Generator

And now for our final model, we move onto the time.mktime() datetime handling! Let's see how this performs when combined with the time series generator of pycaret.

In [176...
```
df_dum3.head()
```

Out[176...

	deptDateTimeMktime	deptMonth	marktName_Aeromexico	marktName_Air Canada	marktName_Air Transat	marktName_Alaska Airlines	marktName_Allegiant Air	marktName_Ameri Airli
0	1.700538e+09	11	0	0	0	0	0	
1	1.700498e+09	11	0	0	0	0	1	
2	1.700485e+09	11	0	0	0	0	0	
3	1.700528e+09	11	0	0	0	0	0	
4	1.700499e+09	11	0	0	0	0	0	

5 rows × 46 columns

In [177...
```
df_dum3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44367 entries, 0 to 44916
Data columns (total 46 columns):
 #   Column                      Non-Null Count  Dtype
```

```
---  ------                        --------------  -----
 0   deptDateTimeMktime            44367 non-null  float64
 1   deptMonth                     44367 non-null  int64
 2   marktName_Aeromexico          44367 non-null  uint8
 3   marktName_Air Canada          44367 non-null  uint8
 4   marktName_Air Transat         44367 non-null  uint8
 5   marktName_Alaska Airlines     44367 non-null  uint8
 6   marktName_Allegiant Air       44367 non-null  uint8
 7   marktName_American Airlines   44367 non-null  uint8
 8   marktName_Breeze Airways      44367 non-null  uint8
 9   marktName_Canada Jetlines     44367 non-null  uint8
 10  marktName_Flair Airlines      44367 non-null  uint8
 11  marktName_Frontier Airlines   44367 non-null  uint8
 12  marktName_Key Lime Air        44367 non-null  uint8
 13  marktName_LATAM               44367 non-null  uint8
 14  marktName_Lynx Air            44367 non-null  uint8
 15  marktName_Porter Airlines     44367 non-null  uint8
 16  marktName_Silver Airways      44367 non-null  uint8
 17  marktName_Spirit Airlines     44367 non-null  uint8
 18  marktName_Sun Country         44367 non-null  uint8
 19  marktName_United Airlines     44367 non-null  uint8
 20  marktName_Volaris             44367 non-null  uint8
 21  marktName_WestJet             44367 non-null  uint8
 22  orgName_Chicago               44367 non-null  uint8
 23  orgName_Cincinnati            44367 non-null  uint8
 24  orgName_Denver                44367 non-null  uint8
 25  orgName_Los Angeles           44367 non-null  uint8
 26  orgName_Miami                 44367 non-null  uint8
 27  orgName_New York              44367 non-null  uint8
 28  orgName_Orlando               44367 non-null  uint8
 29  orgName_San Francisco         44367 non-null  uint8
 30  destName_Chicago              44367 non-null  uint8
 31  destName_Cincinnati           44367 non-null  uint8
 32  destName_Denver               44367 non-null  uint8
 33  destName_Los Angeles          44367 non-null  uint8
 34  destName_Miami                44367 non-null  uint8
 35  destName_New York             44367 non-null  uint8
 36  destName_Orlando              44367 non-null  uint8
 37  destName_San Francisco        44367 non-null  uint8
 38  deptDayOfWeek_0               44367 non-null  uint8
 39  deptDayOfWeek_1               44367 non-null  uint8
 40  deptDayOfWeek_2               44367 non-null  uint8
 41  deptDayOfWeek_3               44367 non-null  uint8
 42  deptDayOfWeek_4               44367 non-null  uint8
 43  deptDayOfWeek_5               44367 non-null  uint8
 44  deptDayOfWeek_6               44367 non-null  uint8
 45  price                         44367 non-null  float64
dtypes: float64(2), int64(1), uint8(43)
memory usage: 4.2 MB
```

In [178…

```
x_train3, x_test3, y_train3, y_test3 = train_test_split(features3_dum, targets3, test_size=0.22, random_state=23)
```

In [179…

```
df_train3 = pd.concat([x_train3, y_train3], axis='columns')
df_test3 = pd.concat([x_test3, y_test3], axis='columns')
```

In [180…

```
s = setup(data = df_train3, test_data = df_test3, target = 'price', fold_strategy = 'timeseries', fold = 10, transform_target = True
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 46)
4	Transformed data shape	(44367, 46)
5	Transformed train set shape	(34606, 46)
6	Transformed test set shape	(9761, 46)
7	Numeric features	45
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Transform target	True
13	Transform target method	yeo-johnson
14	Fold Generator	TimeSeriesSplit
15	Fold Number	10
16	CPU Jobs	-1
17	Use GPU	False
18	Log Experiment	False
19	Experiment Name	reg-default-name
20	USI	6dfa

In [181…

```
best = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
rf	Random Forest Regressor	30.0686	2663.1727	51.3834	0.7357	0.2116	0.1445	3.4920
et	Extra Trees Regressor	30.3032	3083.8612	55.2927	0.6937	0.2289	0.1480	2.9250
lightgbm	Light Gradient Boosting Machine	35.3163	3207.9125	56.5720	0.6823	0.2250	0.1666	0.2210
dt	Decision Tree Regressor	34.9353	4207.4018	64.4693	0.5816	0.2652	0.1701	0.1470
gbr	Gradient Boosting Regressor	43.7667	4691.7474	68.4512	0.5358	0.2710	0.2086	1.3250
ridge	Ridge Regression	52.4087	6365.4888	79.7426	0.3698	0.3424	0.2655	0.0480
knn	K Neighbors Regressor	58.0642	7597.4642	87.1264	0.2478	0.3773	0.3046	0.2920
br	Bayesian Ridge	61.0699	8581.9587	92.5869	0.1511	0.3966	0.3200	0.0570
ada	AdaBoost Regressor	73.9840	8766.5050	93.4956	0.1346	0.4266	0.4374	0.8740
omp	Orthogonal Matching Pursuit	66.7436	9791.7725	98.9048	0.0311	0.4323	0.3571	0.0510
lr	Linear Regression	66.9579	9839.9131	99.1399	0.0267	0.4329	0.3578	0.0570
lasso	Lasso Regression	67.1085	9867.6707	99.2771	0.0241	0.4345	0.3593	0.1160
en	Elastic Net	67.1085	9867.6713	99.2771	0.0241	0.4345	0.3593	0.1470
llar	Lasso Least Angle Regression	67.1085	9867.6692	99.2771	0.0241	0.4345	0.3593	0.0540
huber	Huber Regressor	69.2390	10303.2210	101.4461	-0.0190	0.4549	0.3888	0.1080
dummy	Dummy Regressor	70.2034	10432.6995	102.0838	-0.0318	0.4606	0.3981	0.0550
par	Passive Aggressive Regressor	76.6258	12561.0521	111.7891	-0.2424	0.4996	0.3865	0.0710
lar	Least Angle Regression	58.1189	29655.0146	122.6384	-2.2145	0.3785	0.2884	0.0530

In [ ]:

```
```

Model Summary

We're now done with all of the pycaret model comparison we wanted to run! We ran a number of different ML models to compare, using the great functionality of pycaret. Within those comparison, we also compared a number of different datetime format to explore how timeseries can be handled within ML models. The results are below!

In [182…

```
models = ['Sklearn Linear Regression', 'Parsed Date, KFolds gen', 'Ordinal Date, KFolds gen', 'mkTime Date, KFolds gen', 'fullDateT
MAEs = [54.68, 22.23, 22.05, 27.26, 55.73, 55.83, 25.96, 25.64, 30.07]

data = list(zip(models, MAEs))

df_errors = pd.DataFrame(data, columns=['model', 'MAE'])

print(df_errors)
```

```
                        model    MAE
0     Sklearn Linear Regression  54.68
1       Parsed Date, KFolds gen  22.23
2      Ordinal Date, KFolds gen  22.05
3       mkTime Date, KFolds gen  27.26
4  fullDateTime, Timeseries gen  55.73
5      dateTime, Timeseries gen  55.83
6   Parsed Date, Timeseries gen  25.96
7  Ordinal Date, Timeseries gen  25.64
8   mkTime Date, Timeseries gen  30.07
```

In [183…

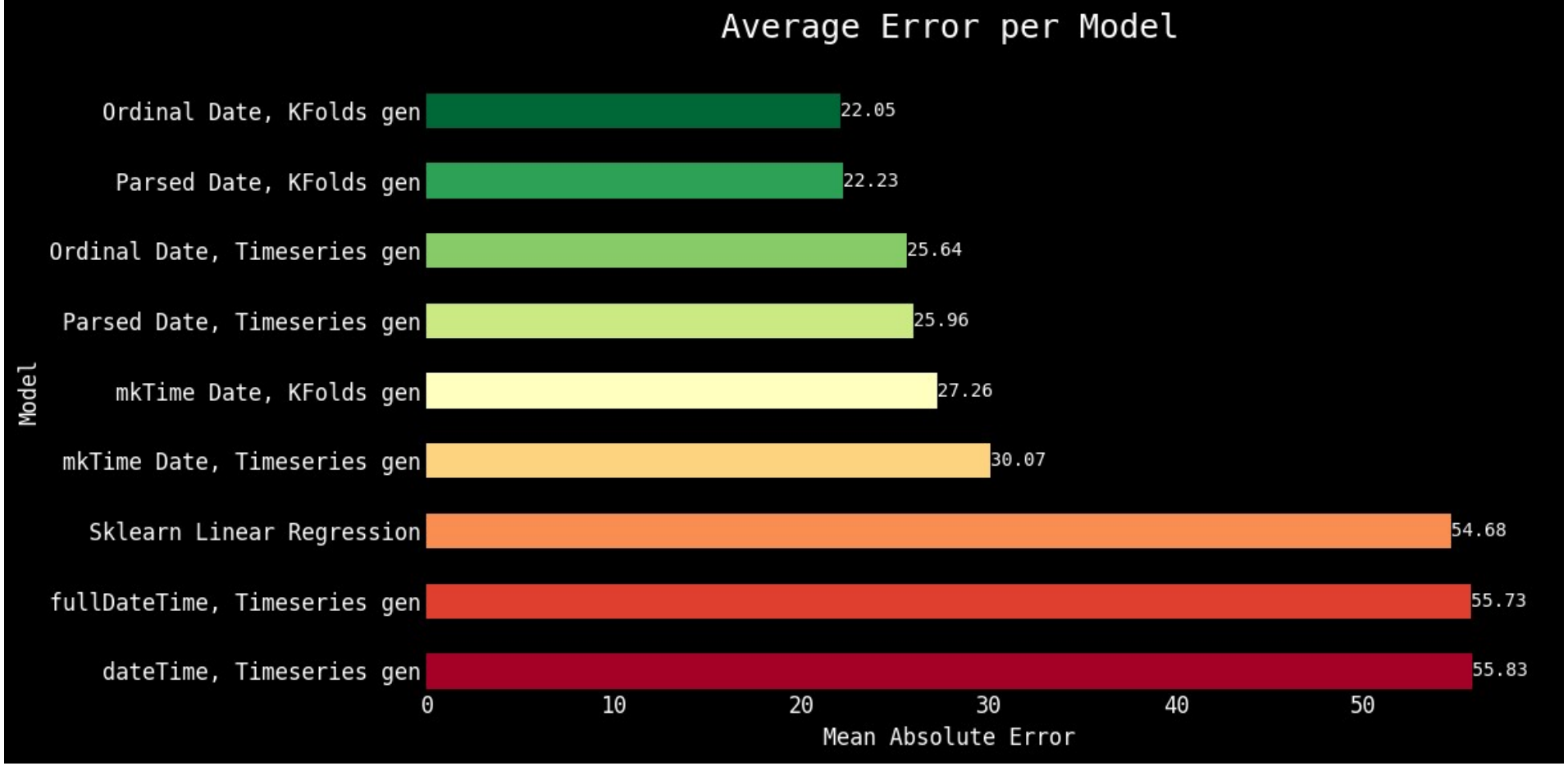
```
df_errors_sorted = df_errors.sort_values(by='MAE', ascending=False)
print(df_errors_sorted)
```

```
                        model    MAE
5      dateTime, Timeseries gen  55.83
4  fullDateTime, Timeseries gen  55.73
0     Sklearn Linear Regression  54.68
8   mkTime Date, Timeseries gen  30.07
3       mkTime Date, KFolds gen  27.26
6   Parsed Date, Timeseries gen  25.96
7  Ordinal Date, Timeseries gen  25.64
1       Parsed Date, KFolds gen  22.23
2      Ordinal Date, KFolds gen  22.05
```

Aaaaaaaand behold our results!!! in both table form and in bar chart form.

In [184…

```
fig5, ax5 = plt.subplots(figsize=(12,6))
plt.rc('font', size=15)
plt.style.use('dark_background')

# Create a colormap
comap3 = plt.cm.get_cmap('RdYlGn', len(df_errors_sorted['MAE']))

# Assign colors to bars
colors3 = comap3(np.arange(len(df_errors_sorted['MAE'])))

bars = ax5.barh(df_errors_sorted['model'], df_errors_sorted['MAE'], color=colors3, height=0.5)

ax5.set(ylabel='Model', xlabel='Mean Absolute Error', title='Average Error per Model')
ax5.bar_label(bars, fmt='{:,.2f}  ', fontsize=10)

plt.tight_layout()
ax5.spines['top'].set_visible(False)
ax5.spines['right'].set_visible(False)
ax5.spines['bottom'].set_visible(False)
ax5.spines['left'].set_visible(False)
plt.grid(False)
#plt.show()
plt.savefig('output/05-error-by-model.png', dpi=dpi)
```



Model #	Date Handling	Pycaret Generator	Top Model	MAE
1	Parsed Datetime	N/A	Linear Regression	54.68
2	Parsed Datetime	KFold	Extra Trees	22.23
3	Ordinate + Hour	KFold	Random Forest	22.05
4	mktime	KFold	Extra Trees	27.26
5	deptDatetime	Timeseries	Random Forest	55.73
6	deptDate + Hour	Timeseries	LightGBM	55.83
7	Parsed Datetime	Timeseries	Extra Trees Regressor	25.96
8	Parsed Datetime	Timeseries	Extra Trees Regressor	25.64
9	Parsed Datetime	Timeseries	Random Forest	30.07

Finally, let's perform one last pycaret run. This time we'll be using the exact data we used for the top-performing ML model, the Ordinate datetime + Hour pycaret Random Forest model!

Final Model: Pycaret Model #2

In [185…

```
reg2 = setup(data = df_dum2, target = 'price', session_id=23)
```

	Description	Value
0	Session id	23
1	Target	price
2	Target type	Regression
3	Original data shape	(44367, 47)
4	Transformed data shape	(44367, 47)
5	Transformed train set shape	(31056, 47)
6	Transformed test set shape	(13311, 47)
7	Numeric features	46
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	KFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	reg-default-name
18	USI	feb9

In [186…

```
best2 = compare_models(n_select=4, exclude="auto_arima")
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
rf	Random Forest Regressor	22.0532	1717.3270	41.3090	0.8309	0.1659	0.1076	4.7410

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	22.2147	1858.2160	42.9357	0.8172	0.1758	0.1095	5.2680
lightgbm	Light Gradient Boosting Machine	32.2642	2553.3294	50.4808	0.7480	0.2124	0.1646	0.2560
dt	Decision Tree Regressor	25.3544	2852.1946	53.3291	0.7184	0.2118	0.1194	0.1490
gbr	Gradient Boosting Regressor	45.2910	4463.4364	66.7732	0.5593	0.2837	0.2374	1.3170
knn	K Neighbors Regressor	47.4499	5246.6422	72.3906	0.4823	0.3085	0.2495	0.2880
lr	Linear Regression	53.8426	6130.1808	78.2659	0.3944	0.3546	0.2978	0.0880
ridge	Ridge Regression	53.8520	6130.5599	78.2685	0.3944	0.3546	0.2979	0.0670
br	Bayesian Ridge	53.8478	6130.3442	78.2670	0.3944	0.3546	0.2979	0.1090
lasso	Lasso Regression	58.1187	6978.8266	83.5009	0.3111	0.3743	0.3267	0.0680
llar	Lasso Least Angle Regression	58.1187	6978.8256	83.5009	0.3111	0.3743	0.3267	0.0650
en	Elastic Net	67.5013	8970.3864	94.6703	0.1146	0.4303	0.3942	0.0620
omp	Orthogonal Matching Pursuit	67.3252	9100.5236	95.3548	0.1017	0.4348	0.3949	0.0620
dummy	Dummy Regressor	72.5853	10133.5062	100.6225	-0.0003	0.4723	0.4483	0.0560
huber	Huber Regressor	70.1636	10387.5637	101.8747	-0.0253	0.4613	0.4020	0.1010
ada	AdaBoost Regressor	123.9946	19998.1895	140.6140	-0.9807	0.6649	0.8806	1.1030
par	Passive Aggressive Regressor	107.8459	22260.9209	135.5217	-1.2094	0.5090	0.6505	0.0910

In [187…

```
model_rf2 = create_model('rf')
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	21.1546	1414.0188	37.6034	0.8499	0.1610	0.1047
1	22.2905	1615.3289	40.1912	0.8484	0.1673	0.1079
2	22.4011	1572.6652	39.6568	0.8431	0.1669	0.1075
3	22.2372	1601.5141	40.0189	0.8235	0.1671	0.1114
4	22.4817	1902.4926	43.6176	0.8101	0.1703	0.1064
5	21.4428	1535.8340	39.1897	0.8431	0.1649	0.1068

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
6	22.2789	2384.6629	48.8330	0.7845	0.1658	0.1054
7	22.8589	2045.1897	45.2238	0.8069	0.1694	0.1105
8	21.3360	1582.6828	39.7829	0.8412	0.1583	0.1046
9	22.0507	1518.8811	38.9728	0.8581	0.1675	0.1106

In [188…

```
def PredictPrice(mod):
    df_input = pd.DataFrame({'deptDateTimeOrd': [''],
                             'deptHour': [0], 'deptMonth': ['1'], 'marktName_Aeromexico': [0], 'marktName_Air Canada': [0],
                             'marktName_Air Transat': [0], 'marktName_Alaska Airlines': [0], 'marktName_Allegiant Air': [0],
                             'marktName_American Airlines': [0], 'marktName_Breeze Airways': [0], 'marktName_Canada Jetlines': [0],
                             'marktName_Flair Airlines': [0], 'marktName_Frontier Airlines': [0], 'marktName_Key Lime Air': [0],
                             'marktName_LATAM': [0], 'marktName_Lynx Air': [0], 'marktName_Porter Airlines': [0],
                             'marktName_Silver Airways': [0], 'marktName_Spirit Airlines': [0], 'marktName_Sun Country': [0],
                             'marktName_United Airlines': [0], 'marktName_Volaris': [0], 'marktName_WestJet': [0],
                             'orgName_Chicago': [0], 'orgName_Cincinnati': [0], 'orgName_Denver': [1], 'orgName_Los Angeles': [0],
                             'orgName_Miami': [0], 'orgName_New York': [0], 'orgName_Orlando': [0], 'orgName_San Francisco': [0],
                             'destName_Chicago': [0], 'destName_Cincinnati': [0], 'destName_Denver': [0], 'destName_Los Angeles': [0],
                             'destName_Miami': [0], 'destName_New York': [0], 'destName_Orlando': [0], 'destName_San Francisco': [0],
                             'deptDayOfWeek_0': [0], 'deptDayOfWeek_1': [0], 'deptDayOfWeek_2': [0], 'deptDayOfWeek_3': [0],
                             'deptDayOfWeek_4': [0], 'deptDayOfWeek_5': [0], 'deptDayOfWeek_6': [0]})

    year = input("Enter the year (####): ")
    month = input("Enter the month (1-12): ")
    day = input("Enter the day (1-31): ")
    airline = input("Enter the airline: ")
    origin = input("Enter the origin: ")
    destination = input("Enter the destination: ")
    hour_of_day = int(input("Enter the hour of day (0-23): "))

    input_datetime = year + '-' + make2digit(month) + '-' + make2digit(day)
    input_datetime = pd.to_datetime(input_datetime)
    input_ord_datetime = getordinal(input_datetime)
    input_day_of_week = input_datetime.day_of_week
    print(f"Selected {input_datetime}, Day of Week is {input_day_of_week}")

    airline = 'marktName_' + airline
    origin = 'orgName_' + origin
    destination = 'destName_' + destination
    input_day_of_week = 'deptDayOfWeek_' + str(input_day_of_week)

    df_input = pd.DataFrame({'deptDateTimeOrd': [input_ord_datetime],
                             'deptHour': [hour_of_day], 'deptMonth': [make2digit(month)],
                             'marktName_Aeromexico': [int('marktName_Aeromexico'==airline)],
                             'marktName_Air Canada': [int('marktName_Air Canada'==airline)],
                             'marktName_Air Transat': [int('marktName_Air Transat'==airline)],
                             'marktName_Alaska Airlines': [int('marktName_Alaska Airlines'==airline)],
                             'marktName_Allegiant Air': [int('marktName_Allegiant Air'==airline)],
                             'marktName_American Airlines': [int('marktName_American Airlines'==airline)],
                             'marktName_Breeze Airways': [int('marktName_Breeze Airways'==airline)],
                             'marktName_Canada Jetlines': [int('marktName_Canada Jetlines'==airline)],
                             'marktName_Flair Airlines': [int('marktName_Flair Airlines'==airline)],
                             'marktName_Frontier Airlines': [int('marktName_Frontier Airlines'==airline)],
                             'marktName_Key Lime Air': [int('marktName_Key Lime Air'==airline)],
                             'marktName_LATAM': [int('marktName_LATAM'==airline)],
```

```
                            'marktName_Lynx Air': [int('marktName_Lynx Air'==airline)],
                            'marktName_Porter Airlines': [int('marktName_Porter Airlines'==airline)],
                            'marktName_Silver Airways': [int('marktName_Silver Airways'==airline)],
                            'marktName_Spirit Airlines': [int('marktName_Spirit Airlines'==airline)],
                            'marktName_Sun Country': [int('marktName_Sun Country'==airline)],
                            'marktName_United Airlines': [int('marktName_United Airlines'==airline)],
                            'marktName_Volaris': [int('marktName_Volaris'==airline)],
                            'marktName_WestJet': [int('marktName_WestJet'==airline)],
                            'orgName_Chicago': [int('orgName_Chicago'==origin)],
                            'orgName_Cincinnati': [int('orgName_Cincinnati'==origin)],
                            'orgName_Denver': [int('orgName_Denver'==origin)],
                            'orgName_Los Angeles': [int('orgName_Los Angeles'==origin)],
                            'orgName_Miami': [int('orgName_Miami'==origin)],
                            'orgName_New York': [int('orgName_New York'==origin)],
                            'orgName_Orlando': [int('orgName_Orlando'==origin)],
                            'orgName_San Francisco': [int('orgName_San Francisco'==origin)],
                            'destName_Chicago': [int('destName_Chicago'==destination)],
                            'destName_Cincinnati': [int('destName_Cincinnati'==destination)],
                            'destName_Denver': [int('destName_Denver'==destination)],
                            'destName_Los Angeles': [int('destName_Los Angeles'==destination)],
                            'destName_Miami': [int('destName_Miami'==destination)],
                            'destName_New York': [int('destName_New York'==destination)],
                            'destName_Orlando': [int('destName_Orlando'==destination)],
                            'destName_San Francisco': [int('destName_San Francisco'==destination)],
                            'deptDayOfWeek_0': [int('deptDayOfWeek_0'==input_day_of_week)],
                            'deptDayOfWeek_1': [int('deptDayOfWeek_1'==input_day_of_week)],
                            'deptDayOfWeek_2': [int('deptDayOfWeek_2'==input_day_of_week)],
                            'deptDayOfWeek_3': [int('deptDayOfWeek_3'==input_day_of_week)],
                            'deptDayOfWeek_4': [int('deptDayOfWeek_4'==input_day_of_week)],
                            'deptDayOfWeek_5': [int('deptDayOfWeek_5'==input_day_of_week)],
                            'deptDayOfWeek_6': [int('deptDayOfWeek_6'==input_day_of_week)]})

    input_prediction = predict_model(model_rf2, data = df_input)
    return input_prediction['prediction_label']
```

In [193…

```
result = PredictPrice(model_rf2)
print(result)
```

```
Enter the year (####): 2025
Enter the month (1-12): 1
Enter the day (1-31): 14
Enter the airline: Frontier Airlines
Enter the origin: Denver
Enter the destination: Cincinnati
Enter the hour of day (0-23): 7
Selected 2025-01-14 00:00:00, Day of Week is 1

0    217.715834
Name: prediction_label, dtype: float64
```

In [190…
```
print(f'R2 Score: {r2_score(y_test2, preds2)}')
```

```
R2 Score: 0.8432711378301447
```

Export a csv as a souvenir :)

In [191…
```
df.to_csv('flights_df.csv')
```

Close the original database file! No changes were made but I'll commit anyway, ha.

In [192…
```
conn.commit()
conn.close()
```

Final Thoughts

Before going into my final thoughts, I wanted to again direct the reader to my GitHub link to view my full documentation of the project.

https://github.com/jeremyabeard5/MSDS696

This was a really fun project and gave me a lot of ideas for what I could do to enhance this project in the future. Following my MSDS 692 project, I wanted to really give an example of a real use case for ML and the applications it could have.

Additionally, you'll see in my GitHub link but this project gave me an opportunity to get more in-depth with Tableau and the possiblilities that software has. I still consider there a lot more to be explored with the software but I had a lot of fun putting multiple Tableau dashboards together.

Thanks!

Jeremy

In [ ]:
```
```