preprocess_merge_dataset

May 2, 2024

CIS 662: INTRO TO MACHINE LEARNING AND ALGORITHMS

Semester Project

Group 13: - Niranjan Balasubramani - Preet Karia - Spoorthi Jayaprakash Malgund - Parth Pramod Kulkarni

[1]:
```
# Generic inputs for most ML tasks
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn import tree
import graphviz
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb

pd.options.display.float_format = '{:,.2f}'.format

# setup interactive notebook mode
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from IPython.display import display, HTML
```

```
/var/folders/tl/lnf2sv191t77b2f4hhxqlcx80000gn/T/ipykernel_98584/831096046.py:2:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better
```

```
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

0.1 Merge all flight dataset together:

- Data collected from the following source: https://www.transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=gvzr.
- Duration: January 2022 to December 2023

[2]:
```
bts_files = [
    "../dataset/bts_data/filtered_jan_to_june_2022_bts.csv",
    "../dataset/bts_data/filtered_july_to_december_2022_bts.csv",
    "../dataset/bts_data/filtered_jan_to_june_2023_bts.csv",
    "../dataset/bts_data/filtered_july_to_december_2023_bts.csv"
]
```

[3]:
```
# fetch data
flight_data_set = pd.concat([pd.read_csv(f) for f in bts_files])
flight_data_set.head()
```

[3]:

	MONTH	DAY_OF_WEEK	FL_DATE	MKT_UNIQUE_CARRIER	OP_UNIQUE_CARRIER	\
0	1	1	1/3/22 0:00	AA	MQ	
1	1	1	1/3/22 0:00	AA	MQ	
2	1	1	1/3/22 0:00	AA	MQ	
3	1	1	1/3/22 0:00	B6	B6	
4	1	1	1/3/22 0:00	B6	B6	

	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	CRS_ARR_TIME	…	ARR_DELAY	\
0	4134	ORD	SYR	1025	1316	…	-19.00	
1	4253	ORD	SYR	1725	2012	…	12.00	
2	4316	ORD	SYR	1528	1815	…	-2.00	
3	116	JFK	SYR	829	950	…	NaN	
4	2516	JFK	SYR	2145	2259	…	39.00	

	CANCELLED	DIVERTED	DUP	DISTANCE	WEATHER_DELAY	NAS_DELAY	\
0	0	0	N	607	NaN	NaN	
1	0	0	N	607	NaN	NaN	
2	0	0	N	607	NaN	NaN	
3	1	0	N	209	NaN	NaN	
4	0	0	N	209	0.00	0.00	

	LATE_AIRCRAFT_DELAY	DIV_AIRPORT_LANDINGS	CRS_ELAPSED_TIME
0	NaN	0	NaN
1	NaN	0	NaN

```
2                  NaN                  0                NaN
3                  NaN                  0                NaN
4                 0.00                  0                NaN

[5 rows x 21 columns]
```

[4]:
```
flight_data_set.shape
```

[4]:
```
(7475, 21)
```

[5]:
```
flight_data_set.columns
```

[5]:
```
Index(['MONTH', 'DAY_OF_WEEK', 'FL_DATE', 'MKT_UNIQUE_CARRIER',
       'OP_UNIQUE_CARRIER', 'OP_CARRIER_FL_NUM', 'ORIGIN', 'DEST',
       'CRS_DEP_TIME', 'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY', 'CANCELLED',
       'DIVERTED', 'DUP', 'DISTANCE', 'WEATHER_DELAY', 'NAS_DELAY',
       'LATE_AIRCRAFT_DELAY', 'DIV_AIRPORT_LANDINGS', 'CRS_ELAPSED_TIME'],
      dtype='object')
```

[6]:
```
flight_data_set_filtered = flight_data_set[(flight_data_set['ORIGIN'].
 ↪isin(['ORD', 'JFK', 'MCO']))
                                            & (flight_data_set['DEST'] ==␣
 ↪'SYR')
                                                &␣
 ↪(flight_data_set['MKT_UNIQUE_CARRIER'].isin(['AA','B6','DL','UA','WN']))]

print(set(flight_data_set_filtered['ORIGIN']))
print(set(flight_data_set_filtered['DEST']))
print(set(flight_data_set_filtered['MKT_UNIQUE_CARRIER']))
```

```
{'MCO', 'JFK', 'ORD'}
{'SYR'}
{'B6', 'DL', 'WN', 'AA', 'UA'}
```

[7]:
```
flight_data_set_filtered.shape
```

[7]:
```
(7475, 21)
```

[8]:
```
flight_data_set_filtered.to_csv('../dataset/bts_data/
 ↪filtered_jan2022_to_december2023_bts.csv', index=False)
```

0.2 Merge all weather dataset together:

- Data collected from the following source: https://www.visualcrossing.com/weather-data.
- Duration: January 2022 to December 2023 hourly data.

[9]:
```
weather_data_files = [
    "../dataset/weather_data/chicago_jan_2022_to_december_2023_hourly.csv",
```

```
    "../dataset/weather_data/syracuse_jan_2022_to_december_2023_hourly.csv",
    "../dataset/weather_data/ny_jan_2022_to_december_2023_hourly.csv",
    "../dataset/weather_data/orlando_jan_2022_to_december_2023_hourly.csv",
]
```

[10]:
```
merged_weather = pd.concat([pd.read_csv(f) for f in weather_data_files])
merged_weather.head()
```

[10]:

	name	datetime	temp	feelslike	dew	humidity	precip	\
0	Chicago	2022-01-01T00:00:00	40.90	33.40	36.50	84.14	0.00	
1	Chicago	2022-01-01T01:00:00	39.50	30.80	36.00	86.96	0.00	
2	Chicago	2022-01-01T02:00:00	38.30	31.40	34.70	86.89	0.00	
3	Chicago	2022-01-01T03:00:00	38.50	30.80	34.40	85.10	0.00	
4	Chicago	2022-01-01T04:00:00	38.30	29.90	33.50	82.72	0.00	

	precipprob	preciptype	snow	…	sealevelpressure	cloudcover	visibility	\
0	0	NaN	0.00	…	1,006.70	100.00	5.40	
1	0	NaN	0.00	…	1,007.10	100.00	3.30	
2	0	NaN	0.00	…	1,007.90	100.00	3.30	
3	0	NaN	0.00	…	1,008.50	100.00	5.60	
4	0	NaN	0.00	…	1,008.40	100.00	8.40	

	solarradiation	solarenergy	uvindex	severerisk	conditions	icon	\
0	0	0.00	0	NaN	Overcast	cloudy	
1	0	0.00	0	NaN	Overcast	cloudy	
2	0	0.00	0	NaN	Overcast	cloudy	
3	0	0.00	0	NaN	Overcast	cloudy	
4	0	0.00	0	NaN	Overcast	cloudy	

	stations
0	72534014819,KORD,KMDW,72530094846,74466504838,…
1	72534014819,KORD,KMDW,72530094846,74466504838,…
2	72534014819,KORD,KMDW,72530094846,74466504838,…
3	72534014819,KORD,KMDW,72530094846,74466504838,…
4	72534014819,KORD,KMDW,72530094846,74466504838,…

[5 rows x 24 columns]

[11]:
```
merged_weather.shape
```

[11]: (70080, 24)

[12]:
```
# Replacing city names with airport names
dic = {'syracuse': 'SYR', 'new york': 'JFK', 'orlando': 'MCO', 'chicago': 'ORD'}

merged_weather['name'] = merged_weather['name'].apply(lambda row: dic[row.
 ↪lower()])
```

```
merged_weather.head()
```

[12]:

	name	datetime	temp	feelslike	dew	humidity	precip	\
0	ORD	2022-01-01T00:00:00	40.90	33.40	36.50	84.14	0.00	
1	ORD	2022-01-01T01:00:00	39.50	30.80	36.00	86.96	0.00	
2	ORD	2022-01-01T02:00:00	38.30	31.40	34.70	86.89	0.00	
3	ORD	2022-01-01T03:00:00	38.50	30.80	34.40	85.10	0.00	
4	ORD	2022-01-01T04:00:00	38.30	29.90	33.50	82.72	0.00	

	precipprob	preciptype	snow	…	sealevelpressure	cloudcover	visibility	\
0	0	NaN	0.00	…	1,006.70	100.00	5.40	
1	0	NaN	0.00	…	1,007.10	100.00	3.30	
2	0	NaN	0.00	…	1,007.90	100.00	3.30	
3	0	NaN	0.00	…	1,008.50	100.00	5.60	
4	0	NaN	0.00	…	1,008.40	100.00	8.40	

	solarradiation	solarenergy	uvindex	severerisk	conditions	icon	\
0	0	0.00	0	NaN	Overcast	cloudy	
1	0	0.00	0	NaN	Overcast	cloudy	
2	0	0.00	0	NaN	Overcast	cloudy	
3	0	0.00	0	NaN	Overcast	cloudy	
4	0	0.00	0	NaN	Overcast	cloudy	

	stations
0	72534014819,KORD,KMDW,72530094846,74466504838,…
1	72534014819,KORD,KMDW,72530094846,74466504838,…
2	72534014819,KORD,KMDW,72530094846,74466504838,…
3	72534014819,KORD,KMDW,72530094846,74466504838,…
4	72534014819,KORD,KMDW,72530094846,74466504838,…

[5 rows x 24 columns]

0.3 Filtering Weather Data

0.3.1 Required / Selected weather columns:

- name
- datetime
- temp

- feelslike
- precip

- precipprob
- preciptype
- snow
- windgust

- windspeed
- winddir
- severerisk
- cloudcover
- visibility
- conditions

[13]:
```
merged_weather.columns
weather_columns_to_drop = ['dew','humidity','snowdepth','sealevelpressure',␣
↪'solarradiation', 'solarenergy','uvindex','icon','stations']
merged_weather_filtered = merged_weather.drop(columns = weather_columns_to_drop)
merged_weather_filtered.head()
```

[13]:
```
Index(['name', 'datetime', 'temp', 'feelslike', 'dew', 'humidity', 'precip',
       'precipprob', 'preciptype', 'snow', 'snowdepth', 'windgust',
       'windspeed', 'winddir', 'sealevelpressure', 'cloudcover', 'visibility',
       'solarradiation', 'solarenergy', 'uvindex', 'severerisk', 'conditions',
       'icon', 'stations'],
      dtype='object')
```

[13]:
```
  name             datetime  temp  feelslike  precip  precipprob preciptype  \
0  ORD  2022-01-01T00:00:00 40.90      33.40    0.00           0        NaN
1  ORD  2022-01-01T01:00:00 39.50      30.80    0.00           0        NaN
2  ORD  2022-01-01T02:00:00 38.30      31.40    0.00           0        NaN
3  ORD  2022-01-01T03:00:00 38.50      30.80    0.00           0        NaN
4  ORD  2022-01-01T04:00:00 38.30      29.90    0.00           0        NaN

   snow  windgust  windspeed  winddir  cloudcover  visibility  severerisk  \
0  0.00     21.90      13.80    12.00      100.00        5.40         NaN
1  0.00     26.20      16.30    15.00      100.00        3.30         NaN
2  0.00       NaN      10.40     5.00      100.00        3.30         NaN
3  0.00       NaN      12.40    11.00      100.00        5.60         NaN
4  0.00     26.40      14.10    10.00      100.00        8.40         NaN

  conditions
0   Overcast
1   Overcast
2   Overcast
3   Overcast
4   Overcast
```

[14]:
```
merged_weather_filtered.columns
```

[14]:
```
Index(['name', 'datetime', 'temp', 'feelslike', 'precip', 'precipprob',
       'preciptype', 'snow', 'windgust', 'windspeed', 'winddir', 'cloudcover',
       'visibility', 'severerisk', 'conditions'],
      dtype='object')
```

[15]:
```
merged_weather_filtered.to_csv('../dataset/weather_data/
 ↪merged_weather_hourly_2022_to_2023_filtered_columns.csv', index=False)
```

0.4 Flight Date and Time Processing and Formatting

0.4.1 Making a combined datetime column for arrival and departure

[16]:
```
# fetch data
flight_data = pd.read_csv('../dataset/bts_data/
 ↪filtered_jan2022_to_december2023_bts.csv')
flight_data.shape

# Convert 'FL_DATE' to datetime column
flight_data['FL_DATE'] = pd.to_datetime(flight_data['FL_DATE'].str[:-4])

# Append leading zero to time columns. ex 23 -> '0023'
flight_data['CRS_DEP_TIME'] = flight_data['CRS_DEP_TIME'].astype(str).
 ↪apply(lambda x: x.zfill(4))
flight_data['CRS_ARR_TIME'] = flight_data['CRS_ARR_TIME'].astype(str).
 ↪apply(lambda x: x.zfill(4))

# Calculate scheduled departure and arrival times
flight_data['SCH_DEP_TIME'] = flight_data.apply(lambda row: row['FL_DATE'] + pd.
 ↪Timedelta(hours=int(row['CRS_DEP_TIME'][:2]),␣
 ↪minutes=int(row['CRS_DEP_TIME'][2:])), axis=1)
flight_data['SCH_ARR_TIME'] = flight_data.apply(lambda row: row['FL_DATE'] + pd.
 ↪Timedelta(hours=int(row['CRS_ARR_TIME'][:2]),␣
 ↪minutes=int(row['CRS_ARR_TIME'][2:])), axis=1)

# Drop unnecessary columns
flight_data.drop(columns=['FL_DATE', 'CRS_DEP_TIME', 'CRS_ARR_TIME', 'MONTH'],␣
 ↪axis=1, inplace=True)
flight_data.head()
flight_data.shape
```

[16]: (7475, 21)

```
/var/folders/6b/tmvvscrj5wb5r4ngnndf1gtm0000gn/T/ipykernel_14019/2141439913.py:6
: UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is consistent and
as-expected, please specify a format.
  flight_data['FL_DATE'] = pd.to_datetime(flight_data['FL_DATE'].str[:-4])
```

[16]:

	DAY_OF_WEEK	MKT_UNIQUE_CARRIER	OP_UNIQUE_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	\
0	1	AA	MQ	4134	ORD	
1	1	AA	MQ	4253	ORD	
2	1	AA	MQ	4316	ORD	
3	1	B6	B6	116	JFK	

```
4            1                 B6                 B6               2516    JFK

  DEST  ARR_TIME  ARR_DELAY  CANCELLED  DIVERTED DUP  DISTANCE  WEATHER_DELAY  \
0  SYR  1,257.00     -19.00          0         0   N       607            NaN
1  SYR  2,024.00      12.00          0         0   N       607            NaN
2  SYR  1,813.00      -2.00          0         0   N       607            NaN
3  SYR       NaN        NaN          1         0   N       209            NaN
4  SYR  2,338.00      39.00          0         0   N       209           0.00

   NAS_DELAY  LATE_AIRCRAFT_DELAY  DIV_AIRPORT_LANDINGS  CRS_ELAPSED_TIME  \
0        NaN                  NaN                     0               NaN
1        NaN                  NaN                     0               NaN
2        NaN                  NaN                     0               NaN
3        NaN                  NaN                     0               NaN
4       0.00                 0.00                     0               NaN

         SCH_DEP_TIME        SCH_ARR_TIME
0 2022-01-03 10:25:00 2022-01-03 13:16:00
1 2022-01-03 17:25:00 2022-01-03 20:12:00
2 2022-01-03 15:28:00 2022-01-03 18:15:00
3 2022-01-03 08:29:00 2022-01-03 09:50:00
4 2022-01-03 21:45:00 2022-01-03 22:59:00
```

[16]:
```
(7475, 19)
```

0.5 Now we filter out all flights which arrive to Syracuse Airport on next Day.

0.5.1 i.e. Remove all late night flights

[17]:
```
# Negative "diff_arrival_departure" indicates that flight arrive next day.
diff_arrival_departure = flight_data['SCH_ARR_TIME'] -␣
 ↪flight_data['SCH_DEP_TIME']


flight_data = flight_data[diff_arrival_departure > pd.Timedelta(0)]

flight_data.shape
flight_data.dtypes
```

[17]:
```
(6969, 19)
```

[17]:
```
DAY_OF_WEEK                 int64
MKT_UNIQUE_CARRIER         object
OP_UNIQUE_CARRIER          object
OP_CARRIER_FL_NUM           int64
ORIGIN                     object
DEST                       object
ARR_TIME                  float64
```

```
ARR_DELAY                      float64
CANCELLED                        int64
DIVERTED                         int64
DUP                             object
DISTANCE                         int64
WEATHER_DELAY                  float64
NAS_DELAY                      float64
LATE_AIRCRAFT_DELAY            float64
DIV_AIRPORT_LANDINGS             int64
CRS_ELAPSED_TIME               float64
SCH_DEP_TIME            datetime64[ns]
SCH_ARR_TIME            datetime64[ns]
dtype: object
```

[18]:
```
weather_data = pd.read_csv('../dataset/weather_data/
 ↪merged_weather_hourly_2022_to_2023_filtered_columns.csv',␣
 ↪parse_dates=['datetime'])
weather_data.head()
weather_data.shape
weather_data.dtypes
```

[18]:
```
  name            datetime  temp  feelslike  precip  precipprob preciptype  \
0  ORD 2022-01-01 00:00:00 40.90      33.40    0.00           0        NaN
1  ORD 2022-01-01 01:00:00 39.50      30.80    0.00           0        NaN
2  ORD 2022-01-01 02:00:00 38.30      31.40    0.00           0        NaN
3  ORD 2022-01-01 03:00:00 38.50      30.80    0.00           0        NaN
4  ORD 2022-01-01 04:00:00 38.30      29.90    0.00           0        NaN

   snow  windgust  windspeed  winddir  cloudcover  visibility  severerisk  \
0  0.00     21.90      13.80    12.00      100.00        5.40         NaN
1  0.00     26.20      16.30    15.00      100.00        3.30         NaN
2  0.00       NaN      10.40     5.00      100.00        3.30         NaN
3  0.00       NaN      12.40    11.00      100.00        5.60         NaN
4  0.00     26.40      14.10    10.00      100.00        8.40         NaN

  conditions
0   Overcast
1   Overcast
2   Overcast
3   Overcast
4   Overcast
```

[18]:
```
(70080, 15)
```

[18]:
```
name                    object
datetime        datetime64[ns]
temp                   float64
```

```
feelslike            float64
precip               float64
precipprob             int64
preciptype            object
snow                 float64
windgust             float64
windspeed            float64
winddir              float64
cloudcover           float64
visibility           float64
severerisk           float64
conditions            object
dtype: object
```

0.6 Merging Flight and Weather Data

We're combining weather and airline data, needing both origin and destination weather information.

0.6.1 Steps for Merging:

- **Load Weather Data:** Load two data frames for weather data, one for origin and one for destination.
- **Column Renaming:** Rename origin columns with prefix `ORGIN_WTH_` and destination columns with prefix `DEST_WTH_`.
- **Merge Origin and Destination Weather:** Merge origin and destination weather data one by one.
- **Create Join Columns:** Add join columns in both datasets.
  - **For Origin:**
    - Flight data `ORGIN_WTH_JOIN` = `SCH_DEP_TIME` (rounded to nearest hour) + `ORIGIN`
    - Origin Weather data `ORGIN_WTH_JOIN` = `ORGIN_WTH_datetime` + `name`
  - **For Destination:**
    - Flight data `DEST_WTH_JOIN` = `SCH_ARR_TIME` (rounded to nearest hour) + `DEST`
    - Destination Weather data `DEST_WTH_JOIN` = `DEST_WTH_datetime` + `name`
- **Final Merge:** Join origin and destination data sequentially. Drop the join columns added previously.

[19]:
```
# Create two data frame for origin and destination. Also rename the columns by␣
↪adding the prefix.
rename_origin = {}
rename_dest = {}
for col in weather_data.columns:
    rename_origin[col] = 'ORGIN_WTH_' + col
    rename_dest[col] = 'DEST_WTH_' + col

org_weather_data = weather_data.rename(columns=rename_origin)
dst_weather_data = weather_data.rename(columns=rename_dest)
```

[20]:
```
# Adding Join columns
flight_data['ORGIN_WTH_JOIN'] = flight_data['SCH_DEP_TIME'].dt.round('H').
 ↪astype(str) + flight_data['ORIGIN']
flight_data['DEST_WTH_JOIN'] = flight_data['SCH_ARR_TIME'].dt.round('H').
 ↪astype(str) + flight_data['DEST']
org_weather_data['ORGIN_WTH_JOIN'] = org_weather_data['ORGIN_WTH_datetime'].
 ↪astype(str) + org_weather_data['ORGIN_WTH_name']
dst_weather_data['DEST_WTH_JOIN'] = dst_weather_data['DEST_WTH_datetime'].
 ↪astype(str) + dst_weather_data['DEST_WTH_name']
flight_data.head()
org_weather_data.head()
dst_weather_data.head()
```

[20]:

	DAY_OF_WEEK	MKT_UNIQUE_CARRIER	OP_UNIQUE_CARRIER	OP_CARRIER_FL_NUM	ORIGIN
0	1	AA	MQ	4134	ORD
1	1	AA	MQ	4253	ORD
2	1	AA	MQ	4316	ORD
3	1	B6	B6	116	JFK
4	1	B6	B6	2516	JFK

	DEST	ARR_TIME	ARR_DELAY	CANCELLED	DIVERTED	…	DISTANCE	WEATHER_DELAY
0	SYR	1,257.00	-19.00	0	0	…	607	NaN
1	SYR	2,024.00	12.00	0	0	…	607	NaN
2	SYR	1,813.00	-2.00	0	0	…	607	NaN
3	SYR	NaN	NaN	1	0	…	209	NaN
4	SYR	2,338.00	39.00	0	0	…	209	0.00

	NAS_DELAY	LATE_AIRCRAFT_DELAY	DIV_AIRPORT_LANDINGS	CRS_ELAPSED_TIME
0	NaN	NaN	0	NaN
1	NaN	NaN	0	NaN
2	NaN	NaN	0	NaN
3	NaN	NaN	0	NaN
4	0.00	0.00	0	NaN

	SCH_DEP_TIME	SCH_ARR_TIME	ORGIN_WTH_JOIN
0	2022-01-03 10:25:00	2022-01-03 13:16:00	2022-01-03 10:00:00ORD
1	2022-01-03 17:25:00	2022-01-03 20:12:00	2022-01-03 17:00:00ORD
2	2022-01-03 15:28:00	2022-01-03 18:15:00	2022-01-03 15:00:00ORD
3	2022-01-03 08:29:00	2022-01-03 09:50:00	2022-01-03 08:00:00JFK
4	2022-01-03 21:45:00	2022-01-03 22:59:00	2022-01-03 22:00:00JFK

	DEST_WTH_JOIN
0	2022-01-03 13:00:00SYR
1	2022-01-03 20:00:00SYR
2	2022-01-03 18:00:00SYR
3	2022-01-03 10:00:00SYR
4	2022-01-03 23:00:00SYR

```
[5 rows x 21 columns]
```

[20]:
```
  ORGIN_WTH_name  ORGIN_WTH_datetime  ORGIN_WTH_temp  ORGIN_WTH_feelslike  \
0            ORD 2022-01-01 00:00:00           40.90                33.40
1            ORD 2022-01-01 01:00:00           39.50                30.80
2            ORD 2022-01-01 02:00:00           38.30                31.40
3            ORD 2022-01-01 03:00:00           38.50                30.80
4            ORD 2022-01-01 04:00:00           38.30                29.90

   ORGIN_WTH_precip  ORGIN_WTH_precipprob ORGIN_WTH_preciptype  \
0              0.00                     0                  NaN
1              0.00                     0                  NaN
2              0.00                     0                  NaN
3              0.00                     0                  NaN
4              0.00                     0                  NaN

   ORGIN_WTH_snow  ORGIN_WTH_windgust  ORGIN_WTH_windspeed  ORGIN_WTH_winddir  \
0            0.00               21.90                13.80              12.00
1            0.00               26.20                16.30              15.00
2            0.00                 NaN                10.40               5.00
3            0.00                 NaN                12.40              11.00
4            0.00               26.40                14.10              10.00

   ORGIN_WTH_cloudcover  ORGIN_WTH_visibility  ORGIN_WTH_severerisk  \
0                100.00                  5.40                   NaN
1                100.00                  3.30                   NaN
2                100.00                  3.30                   NaN
3                100.00                  5.60                   NaN
4                100.00                  8.40                   NaN

  ORGIN_WTH_conditions          ORGIN_WTH_JOIN
0             Overcast  2022-01-01 00:00:00ORD
1             Overcast  2022-01-01 01:00:00ORD
2             Overcast  2022-01-01 02:00:00ORD
3             Overcast  2022-01-01 03:00:00ORD
4             Overcast  2022-01-01 04:00:00ORD
```

[20]:
```
  DEST_WTH_name   DEST_WTH_datetime  DEST_WTH_temp  DEST_WTH_feelslike  \
0           ORD 2022-01-01 00:00:00          40.90                33.40
1           ORD 2022-01-01 01:00:00          39.50                30.80
2           ORD 2022-01-01 02:00:00          38.30                31.40
3           ORD 2022-01-01 03:00:00          38.50                30.80
4           ORD 2022-01-01 04:00:00          38.30                29.90

   DEST_WTH_precip  DEST_WTH_precipprob DEST_WTH_preciptype  DEST_WTH_snow  \
0             0.00                    0                 NaN           0.00
```

```
1             0.00                    0                  NaN           0.00
2             0.00                    0                  NaN           0.00
3             0.00                    0                  NaN           0.00
4             0.00                    0                  NaN           0.00

   DEST_WTH_windgust  DEST_WTH_windspeed  DEST_WTH_winddir  \
0              21.90               13.80             12.00
1              26.20               16.30             15.00
2                NaN               10.40              5.00
3                NaN               12.40             11.00
4              26.40               14.10             10.00

   DEST_WTH_cloudcover  DEST_WTH_visibility  DEST_WTH_severerisk  \
0               100.00                 5.40                  NaN
1               100.00                 3.30                  NaN
2               100.00                 3.30                  NaN
3               100.00                 5.60                  NaN
4               100.00                 8.40                  NaN

  DEST_WTH_conditions           DEST_WTH_JOIN
0            Overcast  2022-01-01 00:00:00ORD
1            Overcast  2022-01-01 01:00:00ORD
2            Overcast  2022-01-01 02:00:00ORD
3            Overcast  2022-01-01 03:00:00ORD
4            Overcast  2022-01-01 04:00:00ORD
```

[21]:
```
# Join data set
merged_flight_weather = pd.merge(flight_data, org_weather_data,␣
 ↪on='ORGIN_WTH_JOIN')
merged_flight_weather = pd.merge(merged_flight_weather, dst_weather_data,␣
 ↪on='DEST_WTH_JOIN')

merged_flight_weather.head()
merged_flight_weather.shape
merged_flight_weather.columns
```

[21]:
```
   DAY_OF_WEEK MKT_UNIQUE_CARRIER OP_UNIQUE_CARRIER  OP_CARRIER_FL_NUM ORIGIN  \
0            1                 AA                MQ               4134    ORD
1            1                 AA                MQ               4253    ORD
2            1                 AA                MQ               4316    ORD
3            1                 UA                G7               4576    ORD
4            1                 B6                B6                116    JFK

  DEST  ARR_TIME  ARR_DELAY  CANCELLED  DIVERTED  … DEST_WTH_precipprob  \
0  SYR  1,257.00     -19.00          0         0  …                    0
1  SYR  2,024.00      12.00          0         0  …                    0
2  SYR  1,813.00      -2.00          0         0  …                    0
```

```
3  SYR  1,722.00     -14.00          0         0  …                   0
4  SYR       NaN        NaN          1         0  …                   0

   DEST_WTH_preciptype  DEST_WTH_snow  DEST_WTH_windgust  DEST_WTH_windspeed  \
0                  NaN           0.00                NaN                6.90
1                  NaN           0.00                NaN                5.80
2                  NaN           0.00                NaN                5.70
3                  NaN           0.00                NaN                5.70
4                  NaN           0.00                NaN                6.80

   DEST_WTH_winddir  DEST_WTH_cloudcover DEST_WTH_visibility  \
0            309.00                84.30               9.90
1            335.00                29.60               9.90
2            292.00                47.30               9.90
3            292.00                47.30               9.90
4            301.00                84.30               9.90

  DEST_WTH_severerisk DEST_WTH_conditions
0                 NaN    Partially cloudy
1                 NaN    Partially cloudy
2                 NaN    Partially cloudy
3                 NaN    Partially cloudy
4                 NaN    Partially cloudy

[5 rows x 51 columns]
```

[21]:
```
(6969, 51)
```

[21]:
```
Index(['DAY_OF_WEEK', 'MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER',
       'OP_CARRIER_FL_NUM', 'ORIGIN', 'DEST', 'ARR_TIME', 'ARR_DELAY',
       'CANCELLED', 'DIVERTED', 'DUP', 'DISTANCE', 'WEATHER_DELAY',
       'NAS_DELAY', 'LATE_AIRCRAFT_DELAY', 'DIV_AIRPORT_LANDINGS',
       'CRS_ELAPSED_TIME', 'SCH_DEP_TIME', 'SCH_ARR_TIME', 'ORGIN_WTH_JOIN',
       'DEST_WTH_JOIN', 'ORGIN_WTH_name', 'ORGIN_WTH_datetime',
       'ORGIN_WTH_temp', 'ORGIN_WTH_feelslike', 'ORGIN_WTH_precip',
       'ORGIN_WTH_precipprob', 'ORGIN_WTH_preciptype', 'ORGIN_WTH_snow',
       'ORGIN_WTH_windgust', 'ORGIN_WTH_windspeed', 'ORGIN_WTH_winddir',
       'ORGIN_WTH_cloudcover', 'ORGIN_WTH_visibility', 'ORGIN_WTH_severerisk',
       'ORGIN_WTH_conditions', 'DEST_WTH_name', 'DEST_WTH_datetime',
       'DEST_WTH_temp', 'DEST_WTH_feelslike', 'DEST_WTH_precip',
       'DEST_WTH_precipprob', 'DEST_WTH_preciptype', 'DEST_WTH_snow',
       'DEST_WTH_windgust', 'DEST_WTH_windspeed', 'DEST_WTH_winddir',
       'DEST_WTH_cloudcover', 'DEST_WTH_visibility', 'DEST_WTH_severerisk',
       'DEST_WTH_conditions'],
      dtype='object')
```

[22]:
```
# Drop unnecessary columns
merged_flight_weather_updated = merged_flight_weather.
 ↪drop(columns=['ORGIN_WTH_JOIN', 'DEST_WTH_JOIN', 'ORGIN_WTH_datetime',␣
 ↪'DEST_WTH_datetime', 'ORGIN_WTH_name', 'DEST_WTH_name'])
```

[23]:
```
merged_flight_weather_updated.to_csv('../dataset/merged_data/
 ↪merged_flight_weather_hourly_jan2022_dec2023.csv', index=False)
```

[24]:
```
carrier_counts = merged_flight_weather_updated['MKT_UNIQUE_CARRIER'].
 ↪value_counts()
print(carrier_counts)
```

```
MKT_UNIQUE_CARRIER
UA    1979
AA    1692
B6    1644
DL    1475
WN     179
Name: count, dtype: int64
```

0.7 Data filtering and preparing for TRAINING:

[25]:
```
merged_flight_weather_updated.shape
merged_flight_weather_updated.columns
```

[25]:
```
(6969, 45)
```

[25]:
```
Index(['DAY_OF_WEEK', 'MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER',
       'OP_CARRIER_FL_NUM', 'ORIGIN', 'DEST', 'ARR_TIME', 'ARR_DELAY',
       'CANCELLED', 'DIVERTED', 'DUP', 'DISTANCE', 'WEATHER_DELAY',
       'NAS_DELAY', 'LATE_AIRCRAFT_DELAY', 'DIV_AIRPORT_LANDINGS',
       'CRS_ELAPSED_TIME', 'SCH_DEP_TIME', 'SCH_ARR_TIME', 'ORGIN_WTH_temp',
       'ORGIN_WTH_feelslike', 'ORGIN_WTH_precip', 'ORGIN_WTH_precipprob',
       'ORGIN_WTH_preciptype', 'ORGIN_WTH_snow', 'ORGIN_WTH_windgust',
       'ORGIN_WTH_windspeed', 'ORGIN_WTH_winddir', 'ORGIN_WTH_cloudcover',
       'ORGIN_WTH_visibility', 'ORGIN_WTH_severerisk', 'ORGIN_WTH_conditions',
       'DEST_WTH_temp', 'DEST_WTH_feelslike', 'DEST_WTH_precip',
       'DEST_WTH_precipprob', 'DEST_WTH_preciptype', 'DEST_WTH_snow',
       'DEST_WTH_windgust', 'DEST_WTH_windspeed', 'DEST_WTH_winddir',
       'DEST_WTH_cloudcover', 'DEST_WTH_visibility', 'DEST_WTH_severerisk',
       'DEST_WTH_conditions'],
      dtype='object')
```

[26]:
```
flight_weather_data = merged_flight_weather_updated[
                        (merged_flight_weather_updated['DIV_AIRPORT_LANDINGS']␣
 ↪== 0)
                        & (merged_flight_weather_updated['CANCELLED'] ==0)
                    ]
```

```
flight_weather_data.shape
```

[26]:
```
(6773, 45)
```

[27]:
```
columns_to_drop = ['OP_CARRIER_FL_NUM', 'CANCELLED', 'DUP', 'OP_CARRIER_FL_NUM',
                   'DIV_AIRPORT_LANDINGS', 'DISTANCE', 'WEATHER_DELAY',␣
 ↪'NAS_DELAY',
                   'LATE_AIRCRAFT_DELAY', 'CRS_ELAPSED_TIME',␣
 ↪'ORGIN_WTH_feelslike',
                   'ORGIN_WTH_windgust', 'DEST_WTH_feelslike',␣
 ↪'DEST_WTH_windgust',
                   'DEST', 'DEST_WTH_preciptype', 'ORGIN_WTH_preciptype',␣
 ↪'DIVERTED',
                   'DEST_WTH_conditions', 'ORGIN_WTH_conditions', 'ARR_TIME']
flight_weather_data_updated =  flight_weather_data.drop(columns =␣
 ↪columns_to_drop)
flight_weather_data_updated.shape
flight_weather_data_updated.columns
```

[27]:
```
(6773, 25)
```

[27]:
```
Index(['DAY_OF_WEEK', 'MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER', 'ORIGIN',
       'ARR_DELAY', 'SCH_DEP_TIME', 'SCH_ARR_TIME', 'ORGIN_WTH_temp',
       'ORGIN_WTH_precip', 'ORGIN_WTH_precipprob', 'ORGIN_WTH_snow',
       'ORGIN_WTH_windspeed', 'ORGIN_WTH_winddir', 'ORGIN_WTH_cloudcover',
       'ORGIN_WTH_visibility', 'ORGIN_WTH_severerisk', 'DEST_WTH_temp',
       'DEST_WTH_precip', 'DEST_WTH_precipprob', 'DEST_WTH_snow',
       'DEST_WTH_windspeed', 'DEST_WTH_winddir', 'DEST_WTH_cloudcover',
       'DEST_WTH_visibility', 'DEST_WTH_severerisk'],
      dtype='object')
```

[28]:
```
# print(set(flight_weather_data_updated['DEST_WTH_conditions']))
# print(set(flight_weather_data_updated['ORGIN_WTH_conditions']))
```

[29]:
```
# flight_weather_data_updated['DEST_WTH_conditions'] =␣
 ↪flight_weather_data_updated['DEST_WTH_conditions'] \
#                                                     .replace('Rain,␣
 ↪Overcast', 'Rain') \
#                                                     .replace('Rain,␣
 ↪Partially cloudy', 'Rain') \
#                                                     .replace('Snow,␣
 ↪Overcast', 'Snow') \
#                                                     .replace('Snow,␣
 ↪Partially cloudy', 'Snow') \
#                                                     .replace('Snow, Rain,␣
 ↪Overcast', 'Overcast')
```

```
# flight_weather_data_updated['ORGIN_WTH_conditions'] =␣
 ↪flight_weather_data_updated['ORGIN_WTH_conditions'] \
#                                                     .replace('Freezing␣
 ↪Drizzle/Freezing Rain, Overcast', 'Rain') \
#                                                     .replace('Ice,␣
 ↪Overcast', 'Snow') \
#                                                     .replace('Rain,␣
 ↪Overcast', 'Rain') \
#                                                     .replace('Rain,␣
 ↪Partially cloudy', 'Rain') \
#                                                     .replace('Snow,␣
 ↪Overcast', 'Snow') \
#                                                     .replace('Snow,␣
 ↪Partially cloudy', 'Snow') \
#                                                     .replace('Snow, Rain,␣
 ↪Overcast', 'Overcast')
```

[30]:
```
# print(set(flight_weather_data_updated['DEST_WTH_conditions']))
# print(set(flight_weather_data_updated['ORGIN_WTH_conditions']))
flight_weather_data_updated.columns
```

[30]:
```
Index(['DAY_OF_WEEK', 'MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER', 'ORIGIN',
       'ARR_DELAY', 'SCH_DEP_TIME', 'SCH_ARR_TIME', 'ORGIN_WTH_temp',
       'ORGIN_WTH_precip', 'ORGIN_WTH_precipprob', 'ORGIN_WTH_snow',
       'ORGIN_WTH_windspeed', 'ORGIN_WTH_winddir', 'ORGIN_WTH_cloudcover',
       'ORGIN_WTH_visibility', 'ORGIN_WTH_severerisk', 'DEST_WTH_temp',
       'DEST_WTH_precip', 'DEST_WTH_precipprob', 'DEST_WTH_snow',
       'DEST_WTH_windspeed', 'DEST_WTH_winddir', 'DEST_WTH_cloudcover',
       'DEST_WTH_visibility', 'DEST_WTH_severerisk'],
      dtype='object')
```

[31]:
```
flight_weather_data_updated.isna().sum()
```

[31]:
```
DAY_OF_WEEK              0
MKT_UNIQUE_CARRIER       0
OP_UNIQUE_CARRIER        0
ORIGIN                   0
ARR_DELAY                0
SCH_DEP_TIME             0
SCH_ARR_TIME             0
ORGIN_WTH_temp           0
ORGIN_WTH_precip         0
ORGIN_WTH_precipprob     0
ORGIN_WTH_snow           0
ORGIN_WTH_windspeed      0
ORGIN_WTH_winddir        0
ORGIN_WTH_cloudcover     0
```

```
ORGIN_WTH_visibility     0
ORGIN_WTH_severerisk    76
DEST_WTH_temp            0
DEST_WTH_precip          0
DEST_WTH_precipprob      0
DEST_WTH_snow            0
DEST_WTH_windspeed       0
DEST_WTH_winddir         0
DEST_WTH_cloudcover      0
DEST_WTH_visibility      0
DEST_WTH_severerisk     80
dtype: int64
```

Filling severerisk NaN in weather data with its minimum value

[32]:
```
flight_weather_data_updated['ORGIN_WTH_severerisk'] =␣
  ↪flight_weather_data_updated['ORGIN_WTH_severerisk'].
  ↪fillna(flight_weather_data_updated['ORGIN_WTH_severerisk'].min())
flight_weather_data_updated['DEST_WTH_severerisk'] =␣
  ↪flight_weather_data_updated['DEST_WTH_severerisk'].
  ↪fillna(flight_weather_data_updated['DEST_WTH_severerisk'].min())
flight_weather_data_updated.isna().sum()
flight_weather_data_updated.shape
```

[32]:
```
DAY_OF_WEEK             0
MKT_UNIQUE_CARRIER      0
OP_UNIQUE_CARRIER       0
ORIGIN                  0
ARR_DELAY               0
SCH_DEP_TIME            0
SCH_ARR_TIME            0
ORGIN_WTH_temp          0
ORGIN_WTH_precip        0
ORGIN_WTH_precipprob    0
ORGIN_WTH_snow          0
ORGIN_WTH_windspeed     0
ORGIN_WTH_winddir       0
ORGIN_WTH_cloudcover    0
ORGIN_WTH_visibility    0
ORGIN_WTH_severerisk    0
DEST_WTH_temp           0
DEST_WTH_precip         0
DEST_WTH_precipprob     0
DEST_WTH_snow           0
DEST_WTH_windspeed      0
DEST_WTH_winddir        0
DEST_WTH_cloudcover     0
DEST_WTH_visibility     0
```

```
DEST_WTH_severerisk     0
dtype: int64
```

[32]:
```
(6773, 25)
```

[33]:
```
flight_weather_data_updated.to_csv('../dataset/merged_data/former_flight_data.
↪csv', index=False)
```

0.8 For latter flights prediction model, we need one additional feature – status of the former flight

Steps to add column – `FORMER_FLIGHT_STATUS`

- Things we consider:
  - For any given flight – `FORMER_FLIGHT_STATUS` = Status of the preceding flight **just before the given flight** on **same day** and **same origin - destination**.
- We first sort the data, according scheduled arrival time. We reset index after that.
- Then for every given row, we figure its `FORMER_FLIGHT_STATUS` based on above consideration.

[34]:
```
flight_data = pd.read_csv('../dataset/merged_data/former_flight_data.csv')
flight_data['SCH_ARR_TIME'] = pd.to_datetime(flight_data['SCH_ARR_TIME'])
flight_data['SCH_DEP_TIME'] = pd.to_datetime(flight_data['SCH_DEP_TIME'])
flight_data = flight_data.sort_values(by='SCH_ARR_TIME').reset_index(drop=True)
flight_data.head(10)
```

[34]:

	DAY_OF_WEEK	MKT_UNIQUE_CARRIER	OP_UNIQUE_CARRIER	ORIGIN	ARR_DELAY	\
0	6	WN	WN	MCO	-26.00	
1	6	UA	OO	ORD	-25.00	
2	6	B6	B6	MCO	22.00	
3	6	B6	B6	JFK	36.00	
4	7	B6	B6	JFK	-12.00	
5	7	AA	MQ	ORD	31.00	
6	7	UA	OO	ORD	48.00	
7	7	DL	9E	JFK	180.00	
8	7	B6	B6	MCO	64.00	
9	7	AA	MQ	ORD	35.00	

	SCH_DEP_TIME	SCH_ARR_TIME	ORGIN_WTH_temp	ORGIN_WTH_precip	\
0	2022-01-01 10:30:00	2022-01-01 13:20:00	74.00	0.00	
1	2022-01-01 10:40:00	2022-01-01 13:32:00	36.10	0.00	
2	2022-01-01 13:13:00	2022-01-01 15:56:00	83.00	0.00	
3	2022-01-01 21:45:00	2022-01-01 22:59:00	52.80	0.15	
4	2022-01-02 08:29:00	2022-01-02 09:50:00	52.10	0.00	
5	2022-01-02 10:25:00	2022-01-02 13:16:00	22.30	0.00	
6	2022-01-02 10:40:00	2022-01-02 13:32:00	23.50	0.00	
7	2022-01-02 12:55:00	2022-01-02 14:12:00	57.20	0.00	
8	2022-01-02 13:13:00	2022-01-02 15:56:00	82.10	0.00	
9	2022-01-02 17:25:00	2022-01-02 20:12:00	25.70	0.00	

	ORGIN_WTH_precipprob	…	ORGIN_WTH_severerisk	DEST_WTH_temp	\
0	0	…	3.00	48.00	
1	0	…	3.00	47.90	
2	0	…	3.00	47.70	
3	100	…	3.00	37.90	
4	0	…	3.00	25.00	
5	0	…	3.00	23.00	
6	0	…	3.00	23.00	
7	0	…	3.00	23.00	
8	0	…	3.00	23.00	
9	0	…	3.00	24.00	

	DEST_WTH_precip	DEST_WTH_precipprob	DEST_WTH_snow	DEST_WTH_windspeed	\
0	0.00	0	0.00	3.60	
1	0.00	0	0.00	0.40	
2	0.00	0	0.00	7.90	
3	0.02	100	0.00	6.10	
4	0.00	0	0.01	13.80	
5	0.01	100	0.01	10.20	
6	0.00	0	0.01	11.20	
7	0.00	0	0.01	11.20	
8	0.00	0	0.01	10.10	
9	0.00	0	0.01	6.80	

	DEST_WTH_winddir	DEST_WTH_cloudcover	DEST_WTH_visibility	\
0	8.00	100.00	9.80	
1	358.00	100.00	9.70	
2	311.00	100.00	7.80	
3	303.00	100.00	6.80	
4	303.00	100.00	1.20	
5	283.00	100.00	4.90	
6	301.00	100.00	8.50	
7	301.00	100.00	8.50	
8	273.00	100.00	3.10	
9	338.00	99.90	9.90	

	DEST_WTH_severerisk
0	3.00
1	3.00
2	3.00
3	3.00
4	3.00
5	3.00
6	3.00
7	3.00
8	3.00

```
9                 3.00

[10 rows x 25 columns]
```

[35]:
```
def get_former_flight_status(row: pd.Series):
    previous_flight_rows = flight_data[(flight_data.index < row.name)\
                                & (flight_data['SCH_ARR_TIME'].dt.date ==␣
 ↪row['SCH_ARR_TIME'].date())\
                                & (flight_data['ORIGIN'] == row['ORIGIN'])]
    if previous_flight_rows.shape[0] <= 0:
        return np.nan
    else:
        arr_delay = previous_flight_rows.iloc[-1]['ARR_DELAY']
        if arr_delay < -5:
            return 'early'
        elif arr_delay > 5:
            return 'late'
        else:
            return 'on-time'
```

[36]:
```
flight_data['FORMER_FLIGHT_STATUS'] = flight_data.
 ↪apply(get_former_flight_status, axis=1)
```

[37]:
```
flight_data[['ORIGIN', 'SCH_ARR_TIME','FORMER_FLIGHT_STATUS', 'ARR_DELAY']].
 ↪head(10)
```

[37]:

	ORIGIN	SCH_ARR_TIME	FORMER_FLIGHT_STATUS	ARR_DELAY
0	MCO	2022-01-01 13:20:00	NaN	-26.00
1	ORD	2022-01-01 13:32:00	NaN	-25.00
2	MCO	2022-01-01 15:56:00	early	22.00
3	JFK	2022-01-01 22:59:00	NaN	36.00
4	JFK	2022-01-02 09:50:00	NaN	-12.00
5	ORD	2022-01-02 13:16:00	NaN	31.00
6	ORD	2022-01-02 13:32:00	late	48.00
7	JFK	2022-01-02 14:12:00	early	180.00
8	MCO	2022-01-02 15:56:00	NaN	64.00
9	ORD	2022-01-02 20:12:00	late	35.00

[38]:
```
flight_data.isna().sum()
```

[38]:
```
DAY_OF_WEEK               0
MKT_UNIQUE_CARRIER        0
OP_UNIQUE_CARRIER         0
ORIGIN                    0
ARR_DELAY                 0
SCH_DEP_TIME              0
SCH_ARR_TIME              0
```

```
ORGIN_WTH_temp             0
ORGIN_WTH_precip           0
ORGIN_WTH_precipprob       0
ORGIN_WTH_snow             0
ORGIN_WTH_windspeed        0
ORGIN_WTH_winddir          0
ORGIN_WTH_cloudcover       0
ORGIN_WTH_visibility       0
ORGIN_WTH_severerisk       0
DEST_WTH_temp              0
DEST_WTH_precip            0
DEST_WTH_precipprob        0
DEST_WTH_snow              0
DEST_WTH_windspeed         0
DEST_WTH_winddir           0
DEST_WTH_cloudcover        0
DEST_WTH_visibility        0
DEST_WTH_severerisk        0
FORMER_FLIGHT_STATUS    2041
dtype: int64
```

[39]:
```
flight_data.to_csv('../dataset/merged_data/latter_flight_data.csv', index=False)
```