MSPA PREDICT 420

Graded Exercise 2: Flight Connections

Introduction

This document presents the results of the second graded exercise for the Masters of Science in Predictive Analytics course: PREDICT 420. This assessment required the student to perform some data w rangling exercises on airline/flight data retrieved from OpenFlights.org.

Assessment

1. Loading the Data

Load datasets into pandas dataframes.

```
In [1]: import pandas as pd
        import pickle

        df_airlines = pd.read_pickle("data/airlines.p")
        df_airports = pd.read_pickle("data/airports.p")
        df_routes = pd.read_pickle("data/routes.p")
```

2. Remove Duplicate Records

Find duplicate records w ithin the 'airlines' dataframe.

```
In [2]: #Duplicate airline records: 1
        df_airlines['Duplicate'] = df_airlines.duplicated(["airlName",
                                                           "airlIATA",
                                                           "airlICAO"])
```

Count the number of duplicate records w ithin the 'airlines' dataframe.

```
In [3]: duplicate_airlines_count = len(df_airlines[df_airlines.Duplicate == True])
        print("Duplicate airline records:", duplicate_airlines_count)
```

```
Duplicate airline records: 1
```

Find duplicate records w ithin the 'airports' dataframe.

```
In [4]: #Duplicate airports records: 54
        df_airports["Duplicate"] = df_airports.duplicated(["airpName",
                                                           "airpCity",
                                                           "airpCountry",
                                                           "airpIATAFAA",
                                                           "airpICAO"])
```

Count the number of duplicate records w ithin the 'airports' dataframe.

```
In [5]: duplicate_airports_count = len(df_airports[df_airports.Duplicate == True])
        print("Duplicate airports records:", duplicate_airports_count)
```

```
Duplicate airports records: 54
```

Find duplicate records w ithin the 'routes' dataframe.

```
In [6]: #Duplicate routes records: 0
        df_routes["Duplicate"] = df_routes.duplicated(["airlName",
                                                       "sourceAirpName",
                                                       "sourceAirpID",
                                                       "destAirpName",
                                                       "destAirpID"])
```

Count the number of duplicate records w ithin the 'routes' dataframe.

```
In [7]: duplicate_routes_count = len(df_routes[df_routes.Duplicate == True])
        print("Duplicate routes records:", duplicate_routes_count)
```

```
Duplicate routes records: 0
```

Eliminate duplicate records from each dataframe.

```
In [8]: df_airlines = df_airlines[df_airlines.Duplicate == False]
        df_airports = df_airports[df_airports.Duplicate == False]
        df_routes = df_routes[df_routes.Duplicate == False]
```

3. Print DataTypes for each Dataframe Column

Create reference table for datatypes and print data types for each column of each dataframe.

```
In [9]: #Source: https://en.wikibooks.org/wiki/Python_Programming/Data_Types
        import pandas as pd

        df_datatypes = pd.read_csv("data/datatypes.csv")
        df_datatypes
```

Out[9]:

	Native Data Type	Pandas Data Type	Class	Description
0	int	int64	Numeric types	Integers
1	float	float64	Numeric types	Floating-point numbers
2	str	object	Sequences	String

```
In [10]: print(df_airlines.drop('Duplicate', 1).dtypes)
```

```
airlID          int64
airlName       object
airlAlias      object
airlIATA       object
airlICAO       object
airlCallsign   object
airlCountry    object
airlActive     object
dtype: object
```

```
In [11]: print(df_airports.drop('Duplicate', 1).dtypes)
```

```
airpID            int64
airpName         object
airpCity         object
airpCountry      object
airpIATAFAA      object
airpICAO         object
airpLat         float64
airpLong        float64
airpAlt           int64
airpTimezone    float64
airpDST          object
airpTz           object
dtype: object
```

```
In [12]: print(df_routes.drop('Duplicate', 1).dtypes)
```

```
airlName          object
airlID            object
sourceAirpName    object
sourceAirpID      object
destAirpName      object
destAirpID        object
airlCodeshare     object
airlStops          int64
airlEquip         object
dtype: object
```

4. Print First 10 Values of the Row Index for each Dataframe

In [13]:
```
print(df_airlines.index[0:10]) #Index values
df_airlines[0:10] #Row values
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype='int64')
```

Out[13]:

	airlID	airlName	airlAlias	airlIATA	airlICAO	airlCallsign	airlCountry	airlActive	Duplicate
0	1	Private flight	\N	-	NaN	NaN	NaN	Y	False
1	2	135 Airways	\N	NaN	GNL	GENERAL	United States	N	False
2	3	1Time Airline	\N	1T	RNX	NEXTIME	South Africa	Y	False
3	4	2 Sqn No 1 Elementary Flying Training School	\N	NaN	WYT	NaN	United Kingdom	N	False
4	5	213 Flight Unit	\N	NaN	TFU	NaN	Russia	N	False
5	6	223 Flight Unit State Airline	\N	NaN	CHD	CHKALOVSK-AVIA	Russia	N	False
6	7	224th Flight Unit	\N	NaN	TTF	CARGO UNIT	Russia	N	False
7	8	247 Jet Ltd	\N	NaN	TWF	CLOUD RUNNER	United Kingdom	N	False
8	9	3D Aviation	\N	NaN	SEC	SECUREX	United States	N	False
9	10	40-Mile Air	\N	Q5	MLA	MILE-AIR	United States	Y	False

In [14]:
```
print(df_airports.index[0:10]) #Index values
df_airports[0:10] #Row values
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype='int64')
```

Out[14]:

	airpID	airpName	airpCity	airpCountry	airpIATAFAA	airpICAO	airpLat	airpLong	airpAlt	airpTimezone	airpDS
0	1	Goroka	Goroka	Papua New Guinea	GKA	AYGA	-6.081689	145.391881	5282	10	U
1	2	Madang	Madang	Papua New Guinea	MAG	AYMD	-5.207083	145.788700	20	10	U
2	3	Mount Hagen	Mount Hagen	Papua New Guinea	HGU	AYMH	-5.826789	144.295861	5388	10	U
3	4	Nadzab	Nadzab	Papua New Guinea	LAE	AYNZ	-6.569828	146.726242	239	10	U
4	5	Port Moresby Jacksons Intl	Port Moresby	Papua New Guinea	POM	AYPY	-9.443383	147.220050	146	10	U
5	6	Wewak Intl	Wewak	Papua New Guinea	WWK	AYWK	-3.583828	143.669186	19	10	U
6	7	Narsarsuaq	Narssarssuaq	Greenland	UAK	BGBW	61.160517	-45.425978	112	-3	E

In [15]:
```
print(df_routes.index[0:10]) #Index values
df_routes[0:10] #Row values
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype='int64')
```

Out[15]:

	airlName	airlID	sourceAirpName	sourceAirpID	destAirpName	destAirpID	airlCodeshare	airlStops	airlEquip	Duplicate
0	2B	410	AER	2965	KZN	2990	NaN	0	CR2	False
1	2B	410	ASF	2966	KZN	2990	NaN	0	CR2	False
2	2B	410	ASF	2966	MRV	2962	NaN	0	CR2	False
3	2B	410	CEK	2968	KZN	2990	NaN	0	CR2	False
4	2B	410	CEK	2968	OVB	4078	NaN	0	CR2	False
5	2B	410	DME	4029	KZN	2990	NaN	0	CR2	False
6	2B	410	DME	4029	NBC	6969	NaN	0	CR2	False
7	2B	410	DME	4029	TGK	\N	NaN	0	CR2	False
8	2B	410	DME	4029	UUA	6160	NaN	0	CR2	False
9	2B	410	EGO	6156	KGD	2952	NaN	0	CR2	False

5. Remove Defunct Records from Airlines Dataframe

Count the number of records within the 'airlines' dataframe.

```
In [16]: airlines_count = len(df_airlines)
         print("Number of airline records:", airlines_count)
```

```
Number of airline records: 6047
```

Count the number of defunct records within the 'airlines' dataframe.

```
In [17]: defunct_airlines_count = len(df_airlines[df_airlines.airlActive == "N"])
         print("Number of defunct airline records:", defunct_airlines_count)
```

```
Number of defunct airline records: 4885
```

Remove defunct records from the 'airlines' dataframe.

```
In [18]: df_airlines = df_airlines[df_airlines.airlActive == "Y"]
```

6. Remove 'Flights from Nowhere' Records from Routes Dataframe

Count the number of records within the 'routes' dataframe.

```
In [19]: routes_count = len(df_routes)
         print("Number of routes records:", routes_count)
```

```
Number of routes records: 67663
```

Find all 'sourceAirpName' records within the 'routes' dataframe which do not appear within 'airpIATAFAA' records within the 'airports' dataframe.

```
In [20]: df_routes["Matched"] = df_routes["sourceAirpName"].isin(df_airports["airpIATAFAA"])
```

Count the number of 'flights from nowhere' records within the 'routes' dataframe.

```
In [21]: nowhere_routes_count = len(df_routes[df_routes.Matched == False])
         print("Number of flights from nowhere routes records:", nowhere_routes_count)
```

```
Number of flights from nowhere routes records: 235
```

Remove 'flights from nowhere' records within the 'routes' dataframe.

```
In [22]: df_routes = df_routes[df_routes.Matched == True]
```

7. Pickle Final Dataframes.

```
In [23]: import pickle

         #df_airlines.to_pickle("data/airlines.p")
         #df_airports.to_pickle("data/airports.p")
         #df_routes.to_pickle("data/routes.p")
```

8. Find the 10 Longest Flight Routes from Chicago O'Hare

Import 'distance_on_unit_sphere' function to calculate distance betw een tw o latitude/longitude pairs.

In [24]:
```
#Source: http://www.johndcook.com/blog/python_longitude_latitude/
import math

def distance_on_unit_sphere(lat1, long1, lat2, long2):
    degrees_to_radians = math.pi/180.0
    phi1 = (90.0 - lat1)*degrees_to_radians
    phi2 = (90.0 - lat2)*degrees_to_radians
    theta1 = long1*degrees_to_radians
    theta2 = long2*degrees_to_radians

    cos = (math.sin(phi1)*math.sin(phi2)*math.cos(theta1 - theta2) +
           math.cos(phi1)*math.cos(phi2))
    arc = math.acos( cos )

    return arc*6373
```

Find the latitude/longitude pair for Chicago O'Hare

In [25]:
```
df_chicagoohare = df_airports[df_airports.airpName == "Chicago Ohare Intl"]

chicagoohare_lat = df_chicagoohare.airpLat
chicagoohare_long = df_chicagoohare.airpLong
```

Apply 'distance_on_unit_sphere' function to each airport w ithin the df_airports dataframe, in order to find the distance betw een each airport and 'Chicago O'Hare'.

In [26]:
```
import numpy as np

df_airports["distToChicago"] = np.vectorize(distance_on_unit_sphere)(df_airports["airpLat"],
                                                                     df_airports["airpLong"],

                                                                     chicagoohare_lat,
                                                                     chicagoohare_long)
```

Return the 10 airports w hich maximize distance to 'Chicago O'Hare' (distance in kilometres).

In [27]:
```
df_airports_distochicagoasc = df_airports.sort_values(by = "distToChicago", ascending = False)[
:10]
df_airports_distochicagoasc[["airpName", "distToChicago"]]
```

Out[27]:

	airpName	distToChicago
7655	Brusselton	17785.379366
6923	Rottnest Island	17672.610324
5108	Albany Airport	17659.811127
3248	Perth Jandakot	17651.854742
6418	Bursw ood Park Helipad	17644.395508
3255	Perth Intl	17635.275374
5416	RAAF Pearce	17614.200347
6922	Cunderdin	17517.892454
5141	Geraldton Airport	17513.563857
5150	Kalbarri Airport	17461.577215