cancelled_flights_and_weather_preprocessing

October 18, 2018

In [1]:
```
import pandas as pd
from collections import Counter
from numpy import arcsin, sin, cos, sqrt, pi, round_
import numpy as np
```

In [2]:
```
earths_radius = 6.371e3 # Earth's radius in km
```

In [3]:
```
def distance(coordinates): #(lat1,lat2,long1,long2)
    """Calculates the distance from longitude and latitude
    between two points using the Haversine formula, result in km."""

    phi1 = coordinates[0]*pi/180 # Converting to radians
    phi2 = coordinates[1]*pi/180
    lambda1 = coordinates[2]*pi/180
    lambda2 = coordinates[3]*pi/180

    distance = 2*earths_radius*arcsin(sqrt(sin((phi2-phi1)/2)**2 +\
                cos(phi1)*cos(phi2)*sin((lambda2-lambda1)/2)**2))

    return distance
```

In [4]:
```
"""Import the data files"""
#airlines = pd.read_csv("data/airlines.csv")
airports = pd.read_csv("data/airports.csv")
```

```
flights = pd.read_csv("data/flights.csv")
weather = pd.read_csv("data/2015_usa_weather.csv")
```

In [5]:
```
# Use only useful columns
airports = airports[['IATA_CODE', 'LATITUDE', 'LONGITUDE']]
flights = flights[['YEAR', 'MONTH', 'DAY', 'ORIGIN_AIRPORT', 'DESTINATION_AIRPORT', 'CANCELLED']]
weather = weather[['LATITUDE', 'LONGITUDE', 'AvgTemp', 'MaxTemp', 'MinTemp', 'Date']]

# Manipulation for date indices for weather
weather.columns = map(str.upper, weather.columns) # Set uppercase for labels
weather['DATE'] = pd.to_datetime(weather['DATE'])
```

In [6]:
```
# Set latitude/longitude for given precision
weather['LATITUDE'] = round_(weather['LATITUDE'],4)
weather['LONGITUDE'] = round_(weather['LONGITUDE'],4)

airports['LATITUDE'] = round_(airports['LATITUDE'],4)
airports['LONGITUDE'] = round_(airports['LONGITUDE'],4)
```

In [7]:
```
# Calculate the closest weather measurement (station) to the given airport
weather_data = pd.DataFrame()
weather_data['COORDINATES'] = list(zip(weather['LATITUDE'],weather['LONGITUDE'])) # create one coordinate
weather_data = weather_data.drop_duplicates(subset='COORDINATES') # for coordiante wise drop, location fixed in time
weather_data[['LATITUDE','LONGITUDE']] =  weather_data['COORDINATES'].apply(pd.Series)
weather_data = weather_data.drop('COORDINATES',axis=1)

length = len(weather_data['LATITUDE'].values)
closest_weather = []

for airport in airports.index.values:

    coordinates = list(zip([airports.loc[airport]['LATITUDE']]*length,weather_data['LATITUDE'],
                           [airports.loc[airport]['LONGITUDE']]*length,weather_data['LONGITUDE']))
    mapped = list(map(distance, coordinates))
    index = mapped.index(min(mapped))
```

```
            closest_weather.append((coordinates[index][1],coordinates[index][3]))

        airports['WEATHER_COORDINATES'] = closest_weather

        # Separate coordinates
        airports[['WEATHER_LATITUDE', 'WEATHER_LONGITUDE']] = airports['WEATHER_COORDINATES'].apply(pd.Series)
        airports = airports.drop('WEATHER_COORDINATES',axis=1)
```

In [8]:
```
airports.columns
```

Out[8]:
```
Index(['IATA_CODE', 'LATITUDE', 'LONGITUDE', 'WEATHER_LATITUDE',
       'WEATHER_LONGITUDE'],
      dtype='object')
```

In [9]:
```
# Set required labels for merging
airports.columns = ['IATA_CODE', 'ORIGIN_LATITUDE', 'ORIGIN_LONGITUDE',
                    'ORIGIN_WEATHER_LATITUDE', 'ORIGIN_WEATHER_LONGITUDE']
```

In [10]:
```
# Merge data to get origin airport coordinates
flights = pd.merge(flights, airports, left_on='ORIGIN_AIRPORT', right_on='IATA_CODE',
                   how='right').drop('IATA_CODE', axis=1)
```

In [11]:
```
c+c1# Set required labels for merging
airports.columns = ['IATA_CODE', 'DESTINATION_LATITUDE', 'DESTINATION_LONGITUDE',
                    'DESTINATION_WEATHER_LATITUDE', 'DESTINATION_WEATHER_LONGITUDE']
```

In [12]:
```
# Merge data to get destination airport coordinates
flights = pd.merge(flights, airports, left_on='DESTINATION_AIRPORT', right_on='IATA_CODE',
                   how='right').drop('IATA_CODE', axis=1)
```

In [13]:
```
# Set the original labels
airports.columns = ['IATA_CODE', 'LATITUDE', 'LONGITUDE', 'WEATHER_LATITUDE', 'WEATHER_LONGITUDE']
```

In [14]:
```
# Drop flights with missing latitude and longitude data
flights = flights[pd.notnull(flights['ORIGIN_LATITUDE'])]
flights = flights[pd.notnull(flights['DESTINATION_LATITUDE'])]
```

In [15]:
```
# Manipulation for date indices for flights
flights['DATE'] = flights['YEAR'].astype(str)+"-"+flights['MONTH'].astype(str)+"-"+flights['DAY'].astype(str)
flights = flights.drop(['YEAR','MONTH','DAY'],axis=1)

flights['DATE'] = pd.to_datetime(flights['DATE'])
```

In [16]:
```
# Calculate distance between airports and create a separate column for it
distances = list(map(distance,list(zip(flights['ORIGIN_LATITUDE'],flights['DESTINATION_LATITUDE'],
                                       flights['ORIGIN_LONGITUDE'],flights['DESTINATION_LONGITUDE']))))

flights['DISTANCE'] = distances
```

In [17]:
```
weather.columns
```

Out[17]:
```
Index(['LATITUDE', 'LONGITUDE', 'AVGTEMP', 'MAXTEMP', 'MINTEMP', 'DATE'], dtype='object')
```

In [18]:
```
# Set required labels for merging
weather.columns = ['LATITUDE', 'LONGITUDE', 'ORIGIN_AVGTEMP', 'ORIGIN_MAXTEMP', 'ORIGIN_MINTEMP', 'DATE']
```

In [19]:
```
# Merge for origin weather at given days
flights = pd.merge(flights,weather,left_on=['ORIGIN_WEATHER_LATITUDE','ORIGIN_WEATHER_LONGITUDE','DATE'],
        right_on=['LATITUDE','LONGITUDE','DATE'])
```

In [20]:
```
# Set required labels for merging
weather.columns = ['LATITUDE', 'LONGITUDE', 'DESTINATION_AVGTEMP', 'DESTINATION_MAXTEMP', 'DESTINATION_MINTEMP', 'DATE'
```

In [21]:
```
# Merge for destination weather at given days
flights = pd.merge(flights,weather,left_on=['DESTINATION_WEATHER_LATITUDE','DESTINATION_WEATHER_LONGITUDE','DATE'],
        right_on=['LATITUDE','LONGITUDE','DATE'])
```

In [22]:
```
flights.columns
```

Out[22]:
```
Index(['ORIGIN_AIRPORT', 'DESTINATION_AIRPORT', 'CANCELLED', 'ORIGIN_LATITUDE',
       'ORIGIN_LONGITUDE', 'ORIGIN_WEATHER_LATITUDE',
       'ORIGIN_WEATHER_LONGITUDE', 'DESTINATION_LATITUDE',
       'DESTINATION_LONGITUDE', 'DESTINATION_WEATHER_LATITUDE',
```

```
       'DESTINATION_WEATHER_LONGITUDE', 'DATE', 'DISTANCE', 'LATITUDE_x',
       'LONGITUDE_x', 'ORIGIN_AVGTEMP', 'ORIGIN_MAXTEMP', 'ORIGIN_MINTEMP',
       'LATITUDE_y', 'LONGITUDE_y', 'DESTINATION_AVGTEMP',
       'DESTINATION_MAXTEMP', 'DESTINATION_MINTEMP'],
      dtype='object')
```

In [23]:
```
# Use the useful columns
flights = flights[['ORIGIN_AIRPORT', 'DESTINATION_AIRPORT', 'DISTANCE', 'ORIGIN_AVGTEMP', 'ORIGIN_MAXTEMP',
                   'ORIGIN_MINTEMP', 'DESTINATION_AVGTEMP', 'DESTINATION_MAXTEMP', 'DESTINATION_MINTEMP', 'CANCELLED']]
```

In [24]:
```
flights.head()
```

Out[24]:

	ORIGIN_AIRPORT	DESTINATION_AIRPORT	DISTANCE	ORIGIN_AVGTEMP	ORIGIN_MAXTEMP	ORIGIN_MINTEMP	DESTINATION_AVGTEMP	DESTINATION_MAXTEMP	DESTINATION_MINTEMP	CANCELLED
0	ANC	SEA	2325.630321	35	35	27	33	42	26	0
1	ANC	SEA	2325.630321	35	35	27	33	42	26	0
2	ANC	SEA	2325.630321	35	35	27	33	42	26	0
3	ANC	SEA	2325.630321	35	35	27	33	42	26	1
4	ANC	SEA	2325.630321	35	35	27	33	42	26	0

In [25]:
```
# Save to a pickle file
flights.to_pickle('./flights_dataframe.p')
```