Udacity: Data Science

Contents

1 Introduction to Data Science

1.1 Data Wrangling

SQL Queries in Python:

Click here to download csv for data.

```
import pandas
import pandasql

def num_rainy_days(filename):
  # The SQL query should return the number of days in rained.
  weather_data = pandas.read_csv(filename)

  q = """
  SELECT COUNT(rain)
  FROM weather_data
  WHERE CAST(rain AS INTEGER) = 1
  """
  rainy_days = pandasql.sqldf(q.lower(), locals())
  return rainy_days

def max_temp_aggregate_by_fog(filename):
  # The SQL query should return the maximum max temp for both foggy and non-foggy
  days
  weather_data = pandas.read_csv(filename)

  q = """
  SELECT fog, MAX(maxtempi)
  FROM weather_data
  GROUP BY 1
  """
  foggy_days = pandasql.sqldf(q.lower(), locals())
  return foggy_days

def avg_weekend_temperature(filename):
  # The SQL query should return the average mean temperature on weekends).
  weather_data = pandas.read_csv(filename)

  q = """
  SELECT AVG(CAST(meantempi AS INTEGER))
  FROM weather_data
  WHERE strftime('%w', date) IN ('0', '6')
  """
  mean_temp_weekends = pandasql.sqldf(q.lower(), locals())
  return mean_temp_weekends

def avg_min_temperature(filename):
  # The SQL query should find the average minimum temperature on
  # rainy days where the minimum temperature is greater than 55 degrees.
  weather_data = pandas.read_csv(filename)

  q = """
  SELECT AVG(mintempi)
  FROM weather_data
  WHERE (rain = 1) AND (mintempi > 55)
  """
  avg_min_temp_rainy = pandasql.sqldf(q.lower(), locals())
  return avg_min_temp_rainy

```

Working with Turnstile Data:

Click here to see an example of an input turnstile file.
Click here to see an example of the function output turnstile file.

```
import csv

def fix_turnstile_data(filenames):
  '''
  Filenames is a list of MTA Subway turnstile text files. As you can see, there
  are numerous data points included in each row. You want to write a function
  that will update each row in the text file so there is only one entry per row.
  '''
  for name in filenames: # go through each file
    with open(name, 'r') as f_in, open(''.join(['updated_'+name]), 'w') as f_out:
      reader = csv.reader(f_in) # csv object to read file
      writer = csv.writer(f_out) # csv object to write file
      for row in reader: # go through each row in file
        for i in range(3, len(row), 5): # 3 elements repeated, 5 elements unique
          writer.writerow(row[0:3] + row[i:i+5])
  return None

def create_master_turnstile_file(filenames, output_file):
  '''
  Write a function that takes the files in the list filenames, which all have the
  columns 'C/A, UNIT, SCP, DATEn, TIMEn, DESCn, ENTRIESn, EXITSn', and consolidates
  them into one file located at output_file.  There should be ONE row with the column
  headers, located at the top of the file. The input files do not have column header
  rows of their own.
  '''
  with open(output_file, 'w') as master_file: # open file to write to
    master_file.write('C/A,UNIT,SCP,DATEn,TIMEn,DESCn,ENTRIESn,EXITSn\n')
    for filename in filenames: # go through each file
      with open(filename, 'r') as f_in: # open the file to be read
        master_file.write(f_in.read()) # write all rows in file to output
  return None

```

Filtering, Creating, and Re-Formatting Data:

```
import pandas
import datetime

def filter_by_regular(filename):
  '''
  Read the csv file located at filename into a pandas dataframe, and filter
  the dataframe to only rows where the 'DESCn' column has the value 'REGULAR'.
  '''
  turnstile_data = pandas.read_csv(filename)
  turnstile_data = turnstile_data[turnstile_data['DESCn'] == 'REGULAR']
  return turnstile_data

def get_hourly_entries(df):
  # This function should count the number of entries since the last reading
  df['ENTRIESn_hourly'] = df['ENTRIESn'].diff().fillna(1)
  return df

def get_hourly_exits(df):
  # This function should count the number of exits since the last reading
  df['EXITSn_hourly'] = df['EXITSn'].diff().fillna(0)
  return df
```

```
def time_to_hour(time):
  # Given a time in the format of "00:00:00" (H:M:S) return hour as an integer.
  hour = int(time[0:2])
  return hour

def reformat_subway_dates(date):
  '''
  The dates in our subway data are formatted in the format month-day-year.
  The dates in our weather underground data are formatted year-month-day.
  Write a function that takes as its input a date in the MTA Subway
  data format, and returns a date in the weather underground format.
  '''
  date_formatted = datetime.datetime.strptime(date, '%M-%d-%y').strftime('%Y-%M-%d')
  return date_formatted

```

1.2 Data Analysis

Click here to download the data we will be using.

```
import numpy
import scipy.stats as s
import pandas
import matplotlib.pyplot as plt

def entries_histogram(turnstile_weather):
  '''
  Let's examine the hourly entries in our NYC subway data and determine what
  distribution the data follows. Plot two histograms on the same axes to show
  hourly entries when raining vs. when not raining.
  '''
  plt.figure()
  turnstile_weather['ENTRIESn_hourly'].loc[turnstile_weather['rain'] == 1].hist(bins
 =30)
  turnstile_weather['ENTRIESn_hourly'].loc[turnstile_weather['rain'] == 0].hist(bins
 =30)
  plt.title('Histogram of ENTRIESn_hourly')
  plt.legend(['Rain', 'No Rain'])
  plt.xlabel('ENTRIESn_hourly')
  plt.ylabel('Frequency')
  return plt

def mann_whitney_plus_means(turnstile_weather):
  '''
  Take the means and run the Mann Whitney U-test on the ENTRIESn_hourly column.
  We use this test instead of the Welch t-Test since our data does not follow
  the normal distribution (seen from graph outputted above).
  '''
  with_rain = turnstile_weather['ENTRIESn_hourly'].loc[
              turnstile_weather['rain'] == 1].values
  without_rain = turnstile_weather['ENTRIESn_hourly'].loc[
                 turnstile_weather['rain'] == 0].values

  with_rain_mean = np.mean(with_rain)
  without_rain_mean = np.mean(without_rain)

  U,p = scipy.stats.mannwhitneyu(with_rain, without_rain)
  return with_rain_mean, without_rain_mean, U, p
```

Note that there is a statistical difference since our $p < 0.05$, so we reject $H_0 : \mu_1 = \mu_2$

Linear Regression:

```
1
2
3
4
```