Analyzing the NYC Subway Dataset

CODE FOR PROBLEM SETS 2-4

Wesley Strange

AT&T | WS2234@ATT.COM

Contents

Problem Set 1: Titanic Survivor Data

1 - A Simple Heuristic

```
import numpy
import pandas
import statsmodels.api as sm

def simple_heuristic(file_path):

    predictions = {}
    df = pandas.read_csv(file_path)
    for passenger_index, passenger in df.iterrows():
        passenger_id = passenger['PassengerId']

        if passenger['Sex'] == 'female':
            predictions[passenger_id] = 1
        else:
            predictions[passenger_id] = 0

    return predictions
```

2 - A More Complex Heuristic

```
import numpy
import pandas
import statsmodels.api as sm

def complex_heuristic(file_path):

    predictions = {}
    df = pandas.read_csv(file_path)
    for passenger_index, passenger in df.iterrows():
        passenger_id = passenger['PassengerId']

        if passenger['Sex'] == 'female' or (passenger['Age'] < 18 and passenger['Pclass'] == 1):
            predictions[passenger_id] = 1
        else:
            predictions[passenger_id] = 0

    return predictions
```

3 - Your Custom Heuristic

```
import numpy
import pandas
import statsmodels.api as sm
```

```
def custom_heuristic(file_path):

    predictions = {}
    df = pandas.read_csv(file_path)
    for passenger_index, passenger in df.iterrows():
        passenger_id = passenger['PassengerId']

        if passenger['Sex'] == 'female' or (passenger['Age'] < 18 and passenger['Pclass'] == 1):
            predictions[passenger_id] = 1
        else:
            if passenger['Age'] <= 6 or passenger['Age'] >= 80:
                predictions[passenger_id] = 1
            else:
                if passenger['Fare'] > 500:
                    predictions[passenger_id] = 1
                else:
                    predictions[passenger_id] = 0

    return predictions
```

Problem Set 2: Wrangling Subway Data

1 - Number of Rainy Days

```
import pandas
import pandasql

def num_rainy_days(filename):
    weather_data = pandas.read_csv(filename)

    q = """
    SELECT count(cast(rain as integer))
    FROM weather_data
    WHERE rain = 1
    """

    rainy_days = pandasql.sqldf(q.lower(), locals())
    return rainy_days
```

2 - Temp on Foggy and Nonfoggy Days

```
import pandas
import pandasql
```

```
def max_temp_aggregate_by_fog(filename):
    weather_data = pandas.read_csv(filename)

    q = """
    SELECT fog, max(maxtempi)
    FROM weather_data
    GROUP BY fog;
    """

    foggy_days = pandasql.sqldf(q.lower(), locals())
    return foggy_days
```

3 - Mean Temp on Weekends

```
import pandas
import pandasql

def avg_weekend_temperature(filename):
    weather_data = pandas.read_csv(filename)

    q = """
    SELECT avg(meantempi)
    FROM weather_data
    WHERE cast(strftime('%w', date) as integer) = 0 or cast(strftime('%w', date) as integer) = 6
    """

    mean_temp_weekends = pandasql.sqldf(q.lower(), locals())
    return mean_temp_weekends
```

4 - Mean Temp on Rainy Days

```
import pandas
import pandasql

def avg_min_temperature(filename):
    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
```

5 - Fixing Turnstile Data

```
import csv

def fix_turnstile_data(filenames):
    for name in filenames:
        f_in = open(name, 'r')
        f_out = open("updated_"+name, 'w')
        reader_in = csv.reader(f_in, delimiter = ",")
        writer_out = csv.writer(f_out, delimiter = ",")
        for line in reader_in:
            temp = 3
            length = len(line)
            line_header = line[0:3]
            while temp < length:
                full_line = line_header + line[temp:temp+5]
                writer_out.writerow(full_line)
                temp = temp + 5
        f_in.close()
        f_out.close()
```

6 - Combining Turnstile Data

```
def create_master_turnstile_file(filenames, output_file):
    with open(output_file, 'w') as master_file:
       master_file.write('C/A,UNIT,SCP,DATEn,TIMEn,DESCn,ENTRIESn,EXITSn\n')
       for filename in filenames:
            f_in = open(filename, 'r')
            for line in f_in:
                master_file.write(line)
            f_in.close()
```

7 - Filtering Irregular Data

```
import pandas

def filter_by_regular(filename):
    turnstile_data = pandas.read_csv(filename)
    turnstile_data = turnstile_data[turnstile_data.DESCn == 'REGULAR']
    return turnstile_data
```

8 - Get Hourly Entries

```
import pandas

def get_hourly_entries(df):
    df['ENTRIESn_hourly'] = df['ENTRIESn'] - df['ENTRIESn'].shift(1)
    df['ENTRIESn_hourly'].fillna(1, inplace=True)
    return df
```

9 - Get Hourly Exits

```
import pandas

def get_hourly_exits(df):
    df['EXITSn_hourly'] = df['EXITSn'] - df['EXITSn'].shift(1)
    df['EXITSn_hourly'].fillna(0, inplace=True)
    return df
```

10 - Time to Hour

```
import pandas

def time_to_hour(time):
    hour = int(time[0:2])
    return hour
```

11 - Reformat Subway Dates

```
import datetime

def reformat_subway_dates(date):
    date_formatted = datetime.datetime.strptime(date, '%m-%d-%y').strftime("%Y-%m-%d")
    return date_formatted
```

Problem Set 3: Analyzing Subway Data

1 - Exploratory Data Analysis

```
import numpy as np
import pandas
import matplotlib.pyplot as plt

def entries_histogram(turnstile_weather):
    plt.figure()

    turnstile_weather['ENTRIESn_hourly'][turnstile_weather.rain == 0].hist(bins=200, label="No
Rain").set_xlim(0,6000)

    turnstile_weather['ENTRIESn_hourly'][turnstile_weather.rain ==
1].hist(bins=200,label="Rain").set_xlim(0,6000)
```

```
    plt.legend()

    return plt
```

3 - Mann-Whitney U-Test

```
import numpy as np
import scipy
import scipy.stats
import pandas

def mann_whitney_plus_means(turnstile_weather):
    with_rain_mean = np.mean(turnstile_weather['ENTRIESn_hourly'][turnstile_weather.rain == 1])
    without_rain_mean = np.mean(turnstile_weather['ENTRIESn_hourly'][turnstile_weather.rain == 0])

    U, p = scipy.stats.mannwhitneyu(turnstile_weather['ENTRIESn_hourly'][turnstile_weather.rain == 1], 
turnstile_weather['ENTRIESn_hourly'][turnstile_weather.rain == 0])

    return with_rain_mean, without_rain_mean, U, p
```

5 - Linear Regression

```
import numpy as np
import pandas
import statsmodels.api as sm

def linear_regression(features, values):
    features = sm.add_constant(features)
    model = sm.OLS(values,features)
    results = model.fit()
    intercept = results.params[0]
    params = results.params[1:]

    return intercept, params

def predictions(dataframe):
    features = dataframe[['Hour', 'mintempi', 'rain', 'meanwindspdi']]

    dummy_units = pandas.get_dummies(dataframe['UNIT'], prefix='unit')
    features = features.join(dummy_units)

    values = dataframe['ENTRIESn_hourly']

    features_array = features.values
    values_array = values.values

    intercept, params = linear_regression(features_array, values_array)
    predictions = intercept + np.dot(features_array, params)

    return predictions
```

6 - Plotting Residuals

```
import numpy as np
import scipy
import matplotlib.pyplot as plt

def plot_residuals(turnstile_weather, predictions):
    plt.figure()
    (turnstile_weather['ENTRIESn_hourly'] - predictions).hist(bins=25)
    return plt
```

7 - Compute R^2

```
import numpy as np
import scipy
import matplotlib.pyplot as plt
import sys

def compute_r_squared(data, predictions):
    SST = ((data - np.mean(data))**2).sum()
    SSReg = ((predictions - data)**2).sum()
    r_squared = 1 - SSReg / SST

    return r_squared
```

Problem Set 4: Visualizing Subway Data

Exercise - Visualization 1

```
from pandas import *
from ggplot import *
import numpy as np

def plot_weather_data(turnstile_weather):
    turnstile_weather_no_rain = turnstile_weather[turnstile_weather.rain==0].reset_index()
    turnstile_weather_rain = turnstile_weather[turnstile_weather.rain==1].reset_index()

    plot = ggplot(turnstile_weather_no_rain, aes('ENTRIESn_hourly')) + geom_bar(fill="red", binwidth=50)
+ xlim(low=0, high=5000) + ylim(low=0, high=7500) + xlab("Volume of Ridership") + ylab("Frequency of
Occurrence") + ggtitle("Histogram of ENTRIESn_Hourly - No Rain")
```

```
    #plot = ggplot(turnstile_weather_rain, aes('ENTRIESn_hourly')) + geom_bar(fill="blue", binwidth=50) + xlim(low=0, high=5000) + ylim(low=0, high=7500) + xlab("Volume of Ridership") + ylab("Frequency of Occurrence") + ggtitle("Histogram of ENTRIESn_Hourly - Rain")

    return plot
```

2 - Make Another Visualization

```
from pandas import *
from ggplot import *

def plot_weather_data(turnstile_weather):
    ENTRIESn_by_hour = turnstile_weather[['Hour','ENTRIESn_hourly']].groupby('Hour', as_index = False).mean()

    plot = ggplot(ENTRIESn_by_hour, aes('Hour', 'ENTRIESn_hourly')) + geom_bar(stat="bar") + xlab("Hour") + ggtitle("Average Ridership by Hour") + ylab("Ridership")

    return plot
```