Data Science – Learning Management Questions

**Week One – Question #1 (Survivor Data Set attached)**

import numpy

import pandas

import statsmodels.api as sm

def simple\_heuristic(file\_path):

'''

In this exercise, we will perform some rudimentary practices similar to those of

an actual data scientist.

Part of a data scientist's job is to use her or his intuition and insight to

write algorithms and heuristics. A data scientist also creates mathematical models

to make predictions based on some attributes from the data that they are examining.

We would like for you to take your knowledge and intuition about the Titanic

and its passengers' attributes to predict whether or not the passengers survived

or perished. You can read more about the Titanic and specifics about this dataset at:

http://en.wikipedia.org/wiki/RMS\_Titanic

http://www.kaggle.com/c/titanic-gettingStarted

In this exercise and the following ones, you are given a list of Titantic passengers

and their associated information. More information about the data can be seen at the

link below:

http://www.kaggle.com/c/titanic-gettingStarted/data.

For this exercise, you need to write a simple heuristic that will use

the passengers' gender to predict if that person survived the Titanic disaster.

You prediction should be 78% accurate or higher.

Here's a simple heuristic to start off:

1) If the passenger is female, your heuristic should assume that the

passenger survived.

2) If the passenger is male, you heuristic should

assume that the passenger did not survive.

You can access the gender of a passenger via passenger['Sex'].

If the passenger is male, passenger['Sex'] will return a string "male".

If the passenger is female, passenger['Sex'] will return a string "female".

Write your prediction back into the "predictions" dictionary. The

key of the dictionary should be the passenger's id (which can be accessed

via passenger["PassengerId"]) and the associated value should be 1 if the

passenger survied or 0 otherwise.

For example, if a passenger is predicted to have survived:

passenger\_id = passenger['PassengerId']

predictions[passenger\_id] = 1

And if a passenger is predicted to have perished in the disaster:

passenger\_id = passenger['PassengerId']

predictions[passenger\_id] = 0

You can also look at the Titantic data that you will be working with

at the link below:

https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/titanic\_data.csv

'''

predictions = {}

df = pandas.read\_excel(file\_path)

for passenger\_index, passenger in df.iterrows():

passenger\_id = passenger['PassengerId']

# If the passenger is a female, then the passenger survived otherwise not servived.

if passenger['Sex'] == 'female':

predictions[passenger\_id] = 1

elif(passenger['Sex']=='male'):

predictions[passenger\_id] = 0

return predictions

if \_\_name\_\_ == "\_\_main\_\_":

file\_path=r'Survivor Data Set.xlsx'

output=simple\_heuristic(file\_path)

print(output)

**Week One – Question #2 (Survivor Data Set attached)**

import numpy

import pandas

import statsmodels.api as sm

def complex\_heuristic(file\_path):

'''

You are given a list of Titantic passengers and their associated

information. More information about the data can be seen at the link below:

http://www.kaggle.com/c/titanic-gettingStarted/data

For this exercise, you need to write a more sophisticated algorithm

that will use the passengers' gender and their socioeconomical class and age

to predict if they survived the Titanic diaster.

You prediction should be 79% accurate or higher.

Here's the algorithm, predict the passenger survived if:

1) If the passenger is female or

2) if his/her socioeconomic status is high AND if the passenger is under 18

Otherwise, your algorithm should predict that the passenger perished in the disaster.

Or more specifically in terms of coding:

female or (high status and under 18)

You can access the gender of a passenger via passenger['Sex'].

If the passenger is male, passenger['Sex'] will return a string "male".

If the passenger is female, passenger['Sex'] will return a string "female".

You can access the socioeconomic status of a passenger via passenger['Pclass']:

High socioeconomic status -- passenger['Pclass'] is 1

Medium socioeconomic status -- passenger['Pclass'] is 2

Low socioeconomic status -- passenger['Pclass'] is 3

You can access the age of a passenger via passenger['Age'].

Write your prediction back into the "predictions" dictionary. The

key of the dictionary should be the Passenger's id (which can be accessed

via passenger["PassengerId"]) and the associated value should be 1 if the

passenger survived or 0 otherwise.

For example, if a passenger is predicted to have survived:

passenger\_id = passenger['PassengerId']

predictions[passenger\_id] = 1

And if a passenger is predicted to have perished in the disaster:

passenger\_id = passenger['PassengerId']

predictions[passenger\_id] = 0

You can also look at the Titantic data that you will be working with

at the link below:

https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/titanic\_data.csv

'''

predictions = {}

df = pandas.read\_csv(file\_path)

for passenger\_index, passenger in df.iterrows():

passenger\_id = passenger['PassengerId']

# Here's the algorithm, predict the passenger survived

# 1) If the passenger is female or

# 2) if his/her socioeconomic status is high AND if the passenger is under 18

# Otherwise, your algorithm should predict that the passenger perished in the disaster

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

predictions[passenger\_id] = 1

else:

predictions[passenger\_id] = 0

return predictions

if \_\_name\_\_ == "\_\_main\_\_":

file\_path='your path'

output=complex\_heuristic(file\_path)

**Week One – Question #3 (Survivor Data Set attached)**

import numpy

import pandas

import statsmodels.api as sm

def custom\_heuristic(file\_path):

'''

You are given a list of Titantic passengers and their associated

information. More information about the data can be seen at the link below:

http://www.kaggle.com/c/titanic-gettingStarted/data

For this exercise, you need to write a custom heuristic that will take

in some combination of the passenger's attributes and predict if the passenger

survived the Titanic diaster.

Can your custom heuristic beat 80% accuracy?

The available attributes are:

Pclass Passenger Class

(1 = 1st; 2 = 2nd; 3 = 3rd)

Name Name

Sex Sex

Age Age

SibSp Number of Siblings/Spouses Aboard

Parch Number of Parents/Children Aboard

Ticket Ticket Number

Fare Passenger Fare

Cabin Cabin

Embarked Port of Embarkation

(C = Cherbourg; Q = Queenstown; S = Southampton)

SPECIAL NOTES:

Pclass is a proxy for socioeconomic status (SES)

1st ~ Upper; 2nd ~ Middle; 3rd ~ Lower

Age is in years; fractional if age less than one

If the age is estimated, it is in the form xx.5

With respect to the family relation variables (i.e. SibSp and Parch)

some relations were ignored. The following are the definitions used

for SibSp and Parch.

Sibling: brother, sister, stepbrother, or stepsister of passenger aboard Titanic

Spouse: husband or wife of passenger aboard Titanic (mistresses and fiancees ignored)

Parent: mother or father of passenger aboard Titanic

Child: son, daughter, stepson, or stepdaughter of passenger aboard Titanic

Write your prediction back into the "predictions" dictionary. The

key of the dictionary should be the passenger's id (which can be accessed

via passenger["PassengerId"]) and the associating value should be 1 if the

passenger survvied or 0 otherwise.

For example, if a passenger is predicted to have survived:

passenger\_id = passenger['PassengerId']

predictions[passenger\_id] = 1

And if a passenger is predicted to have perished in the disaster:

passenger\_id = passenger['PassengerId']

predictions[passenger\_id] = 0

You can also look at the Titantic data that you will be working with

at the link below:

https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/titanic\_data.csv

'''

predictions = {}

df = pandas.read\_csv(file\_path)

for passenger\_index, passenger in df.iterrows():

passenger\_id = passenger['PassengerId']

# default assumption: the passenger perished

predictions[passenger\_id] = 0

# assume that all women and children not in 3rd class survived

if (passenger['Sex']=='female' or passenger['Age'] < 15) and passenger['Pclass'] != 3:

predictions[passenger\_id] = 1

return predictions

if \_\_name\_\_ == "\_\_main\_\_":

file\_path='Your path'

output=custom\_heuristic(file\_path)

print(output)

**Week One – Question #4 (Data Wrangling)**

import pandas

def add\_full\_name(path\_to\_csv, path\_to\_new\_csv):

df=pandas.read\_csv(path\_to\_csv)

# for Hank Aaron, nameFull would be 'Hank Aaron',

df['nameLast']=df['nameFirst'].map(str)+' '+df['nameLast'].map(str)

#2) The data in the pandas dataFrame to a new csv file located at path\_to\_new\_csv

df.to\_csv(path\_to\_new\_csv)

if \_\_name\_\_ == "\_\_main\_\_":

# For local use only

# If you are running this on your own machine add the path to the

# Lahman baseball csv and a path for the new csv.

# The dataset can be downloaded from this website: http://www.seanlahman.com/baseball-archive/statistics

# We are using the file Master/people.csv

path\_to\_csv = r"Master.csv"

path\_to\_new\_csv = r"New file path"

add\_full\_name(path\_to\_csv,path\_to\_new\_csv)

names=pandas.read\_csv(path\_to\_new\_csv)

print(names['nameLast'])

**Week One – Question #5 (Data Wrangling)**

import pandas

import pandasql

def select\_first\_50(filename):

# Read in our aadhaar\_data csv to a pandas dataframe. Afterwards, we rename the columns

aadhaar\_data = pandas.read\_csv(filename)

aadhaar\_data.rename(columns = lambda x: x.replace(' ', '\_').lower(), inplace=True)

# print(aadhaar\_data)

# Select out the first 50 values for "registrar" and "enrolment\_agency"

# in the aadhaar\_data table using SQL syntax.

#

# Note that "enrolment\_agency" is spelled with one l. Also, the order

# of the select does matter. Make sure you select registrar then enrolment agency

# in your query.

#

# You can download a copy of the aadhaar data that we are passing

# into this exercise below:

# https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/aadhaar\_data.csv

q = """

SELECT registrar,enrolment\_agency FROM aadhaar\_data LIMIT 50;

"""

#Execute your SQL command against the pandas frame

aadhaar\_solution = pandasql.sqldf(q.lower(), locals())

return aadhaar\_solution

if \_\_name\_\_ == "\_\_main\_\_":

path=”Your path”

rows=select\_first\_50(path)

print(rows)

**Week One – Question #6 (Data Wrangling)**

import json

import requests

def api\_get\_request(url):

# In this exercise, you want to call the last.fm API to get a list of the

# top artists in Spain. The grader will supply the URL as an argument to

# the function; assume get method returns json data with key “top\_10”

# you do not need to construct the address or call this

# function in your grader submission.

#

# Once you've done this, return the name of the number 1 top artist in

# Spain.

return requests.get(url).json()[“top\_10”]

# return the top artist in Spain

**Week One – Question #7 (Data Wrangling)**

import pandas as pd

import numpy as np

def imputation(filename):

# Pandas dataframes have a method called 'fillna(value)', such that you can

# pass in a single value to replace any NAs in a dataframe or series. You

# can call it like this:

# dataframe['column'] = dataframe['column'].fillna(value)

#

# Using the numpy.mean function, which calculates the mean of a numpy

# array, impute any missing values in our Lahman baseball

# data sets 'weight' column by setting them equal to the average weight.

#

# You can access the 'weight' colum in the baseball data frame by

# calling baseball['weight']

baseball = pd.read\_csv(filename)

baseball['weight'].fillna(baseball['weight'].mean(), inplace=True)

return baseball

if \_\_name\_\_ == "\_\_main\_\_":

filename = 'People.csv'

output=imputation(filename)

print(output['weight'])