Open in Colab

T81-558: Applications of Deep Neural Networks

Module 2: Python for Machine Learning

- Instructor: Jeff Heaton, McKelvey School of Engineering, Washington University in St. Louis
- For more information visit the class website.

Module 2 Material

Main video lecture:

- Part 2.1: Introduction to Pandas [Video] [Notebook]
- Part 2.2: Categorical Values [Video] [Notebook]
- Part 2.3: Grouping, Sorting, and Shuffling in Python Pandas [Video] [Notebook]
- Part 2.4: Using Apply and Map in Pandas for Keras [Video] [Notebook]
- **Part 2.5: Feature Engineering in Pandas for Deep Learning in Keras** [Video] [Notebook]

Google CoLab Instructions

The following code ensures that Google CoLab is running the correct version of TensorFlow.

In [1]:
```
try:
    %tensorflow_version 2.x
    COLAB = True
    print("Note: using Google CoLab")
except:
    print("Note: not using Google CoLab")
    COLAB = False
```

```
Note: not using Google CoLab
```

Part 2.5: Feature Engineering

Feature engineering is an essential part of machine learning. For now, we will manually engineer features. However, later in this course, we will see some techniques for automatic feature engineering.

Calculated Fields

It is possible to add new fields to the data frame that your program calculates from the other fields. We can create a new column that gives the weight in kilograms. The equation to calculate a metric weight, given weight in pounds, is:

$$m_{(kg)} = m_{(lb)} \times 0.45359237$$

The following Python code performs this transformation:

In [2]:
```
import os
import pandas as pd

df = pd.read_csv(
    "https://data.heatonresearch.com/data/t81-558/auto-mpg.csv",
    na_values=['NA', '?'])

df.insert(1, 'weight_kg', (df['weight'] * 0.45359237).astype(int))
pd.set_option('display.max_columns', 6)
pd.set_option('display.max_rows', 5)
df
```

Out[2]:

	mpg	weight_kg	cylinders	...	year	origin	name
0	18.0	1589	8	...	70	1	chevrolet chevelle malibu
1	15.0	1675	8	...	70	1	buick skylark 320
...	...	...	...	...	...	...	...
396	28.0	1190	4	...	82	1	ford ranger
397	31.0	1233	4	...	82	1	chevy s-10

398 rows × 10 columns

Google API Keys

Sometimes you will use external APIs to obtain data. The following examples show how to use the Google API keys to encode addresses for use with neural networks. To use these, you will need your own Google API key. The key I have below is not a real key; you need to put your own there. Google will ask for a credit card, but there will be no actual cost unless you use a massive number of lookups. YOU ARE NOT required to get a Google API key for this class; this only shows you how. If you want to get a Google API key, visit this site and obtain one for **geocode**.

You can obtain your key from this link: Google API Keys.

In [3]:
```
if 'GOOGLE_API_KEY' in os.environ:
    # If the API key is defined in an environmental variable,
    # the use the env variable.
    GOOGLE_KEY = os.environ['GOOGLE_API_KEY']
else:
    # If you have a Google API key of your own, you can also just
    # put it here:
    GOOGLE_KEY = 'REPLACE WITH YOUR GOOGLE API KEY'
```

Other Examples: Dealing with Addresses

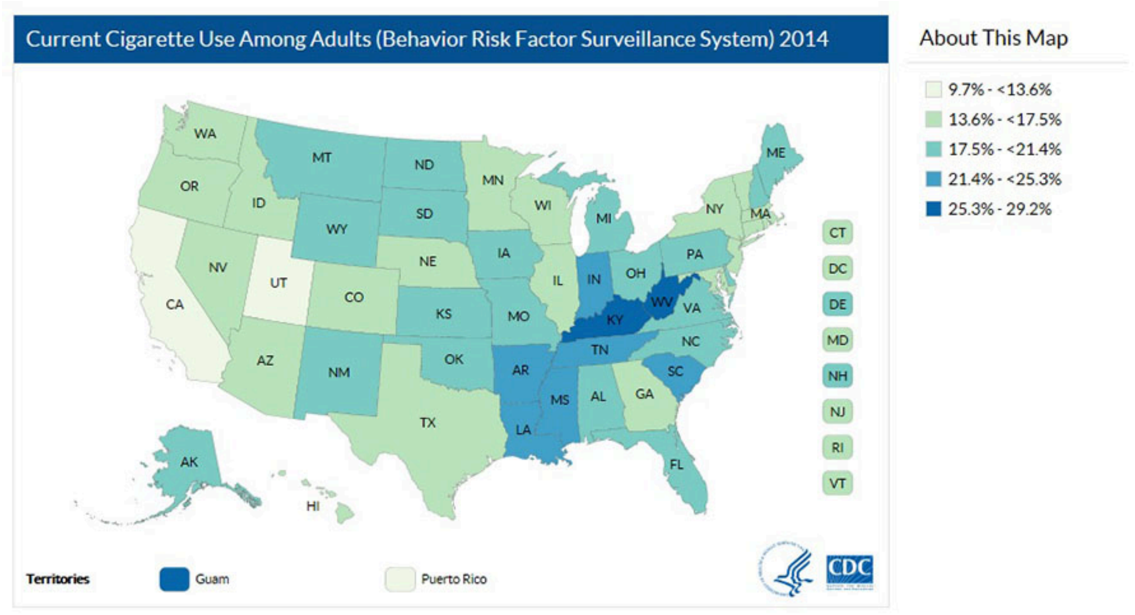
Addresses can be difficult to encode into a neural network. There are many different approaches, and you must consider how you can transform the address into something more meaningful. Map coordinates can be a good approach. latitude and longitude can be a useful encoding. Thanks to the power of the Internet, it is relatively easy to transform an address into its latitude and longitude values. The following code determines the coordinates of Washington University:

In [4]:
```
import requests

address = "1 Brookings Dr, St. Louis, MO 63130"

response = requests.get(
    'https://maps.googleapis.com/maps/api/geocode/json?key={}&address={}' \
    .format(GOOGLE_KEY,address))

resp_json_payload = response.json()

if 'error_message' in resp_json_payload:
    print(resp_json_payload['error_message'])
else:
    print(resp_json_payload['results'][0]['geometry']['location'])
```

```
{'lat': 38.6481653, 'lng': -90.3049506}
```

They might not be overly helpful if you feed latitude and longitude into the neural network as two features. These two values would allow your neural network to cluster locations on a map. Sometimes cluster locations on a map can be useful. Figure 2.SMK shows the percentage of the population that smokes in the USA by state.

Figure 2.SMK: Smokers by State



The above map shows that certain behaviors, like smoking, can be clustered by the global region.

However, often you will want to transform the coordinates into distances. It is reasonably easy to estimate the distance between any two points on Earth by using the great circle distance between any two points on a sphere:

The following code implements this formula:

$$\Delta\sigma = \arccos\bigl(\sin\phi_1 \cdot \sin\phi_2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \cos(\Delta\lambda)\bigr)$$

$$d = r\,\Delta\sigma$$

In [5]:
```python
from math import sin, cos, sqrt, atan2, radians

URL='https://maps.googleapis.com' + \
    '/maps/api/geocode/json?key={}&address={}'

# Distance function
def distance_lat_lng(lat1,lng1,lat2,lng2):
    # approximate radius of earth in km
    R = 6373.0

    # degrees to radians (lat/lon are in degrees)
    lat1 = radians(lat1)
    lng1 = radians(lng1)
    lat2 = radians(lat2)
    lng2 = radians(lng2)

    dlng = lng2 - lng1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlng / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    return R * c

# Find lat lon for address
def lookup_lat_lng(address):
    response = requests.get( \
        URL.format(GOOGLE_KEY,address))
    json = response.json()
    if len(json['results']) == 0:
        raise ValueError("Google API error on: {}".format(address))
    map = json['results'][0]['geometry']['location']
    return map['lat'],map['lng']


# Distance between two locations

import requests

address1 = "1 Brookings Dr, St. Louis, MO 63130"
address2 = "3301 College Ave, Fort Lauderdale, FL 33314"

lat1, lng1 = lookup_lat_lng(address1)
lat2, lng2 = lookup_lat_lng(address2)

print("Distance, St. Louis, MO to Ft. Lauderdale, FL: {} km".format(
        distance_lat_lng(lat1,lng1,lat2,lng2)))
```

```
Distance, St. Louis, MO to Ft. Lauderdale, FL: 1685.3019808607426 km
```

Distances can be a useful means to encode addresses. It would help if you considered what distance might be helpful for your dataset. Consider:

- Distance to a major metropolitan area
- Distance to a competitor
- Distance to a distribution center
- Distance to a retail outlet

The following code calculates the distance between 10 universities and Washington University in St. Louis:

In [6]:
```
# Encoding other universities by their distance to Washington University

schools = [
    ["Princeton University, Princeton, NJ 08544", 'Princeton'],
    ["Massachusetts Hall, Cambridge, MA 02138", 'Harvard'],
    ["5801 S Ellis Ave, Chicago, IL 60637", 'University of Chicago'],
    ["Yale, New Haven, CT 06520", 'Yale'],
    ["116th St & Broadway, New York, NY 10027", 'Columbia University'],
    ["450 Serra Mall, Stanford, CA 94305", 'Stanford'],
    ["77 Massachusetts Ave, Cambridge, MA 02139", 'MIT'],
    ["Duke University, Durham, NC 27708", 'Duke University'],
    ["University of Pennsylvania, Philadelphia, PA 19104",
         'University of Pennsylvania'],
    ["Johns Hopkins University, Baltimore, MD 21218", 'Johns Hopkins']
]

lat1, lng1 = lookup_lat_lng("1 Brookings Dr, St. Louis, MO 63130")

for address, name in schools:
    lat2,lng2 = lookup_lat_lng(address)
    dist = distance_lat_lng(lat1,lng1,lat2,lng2)
    print("School '{}', distance to wustl is: {}".format(name,dist))
```

```
School 'Princeton', distance to wustl is: 1354.4830895052746
School 'Harvard', distance to wustl is: 1670.6297027161022
School 'University of Chicago', distance to wustl is: 418.0815972177934
School 'Yale', distance to wustl is: 1508.217831712127
School 'Columbia University', distance to wustl is: 1418.2264083295695
School 'Stanford', distance to wustl is: 2780.6829398114114
School 'MIT', distance to wustl is: 1672.4444489665696
School 'Duke University', distance to wustl is: 1046.7970984423719
School 'University of Pennsylvania', distance to wustl is: 1307.19541200423
School 'Johns Hopkins', distance to wustl is: 1184.3831076555425
```

```
In [ ]:
```