

COMP 4331 Tut 2

Hongming Zhang

Outline

- Brief introduction to anaconda
- How to install anaconda in AWS server
- Brief introduction to Jupyter Notebook
- How to install Jupyter Notebook in AWS server

What is Anaconda

Anaconda is a Python distribution that is particularly popular for data analysis and scientific computing

- Open source project developed by Continuum Analytics, Inc
- Available for Windows, Mac OS X and Linux
- Includes many popular packages: NumPy, SciPy, Matplotlib, Pandas, IPython, Cython
- Includes Spyder, a Python development environment
- Includes conda, a platform-independent package manager

Why do we need Anaconda

Simplifies installation of Python packages

- Platform-independent package manager
- Doesn't require administrative privileges
- Installs non-Python library dependencies (MKL, HDF5, Boost)
- Provides "virtual environment" capabilities
- Many channels exist that support additional packages

For more information, you can refer to: <https://www.anaconda.com/>

How to install anaconda in AWS server

- Connect to the server via ssh (ssh -i "your own key file" [ubuntu@ec2-\[your own address\].us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-[your own address].us-east-2.compute.amazonaws.com))
- Download the anaconda file (curl -O https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh)
- Check the downloaded file (sha256sum Anaconda3-5.2.0-Linux-x86_64.sh)
- Install the anaconda (bash Anaconda3-5.2.0-Linux-x86_64.sh)

Follow the instructions, choose yes for all the choices except the last one. (we do not need to install the VS code)

How to use Anaconda

- Create a new environment (conda create -n comp4331 python=3.6)
- Activate the environment (source activate comp4331)
- Install any package you want (e.g., pip install pandas/torch)
- Deactivate the environment when you finish (source deactivate)

For more information, you can refer to:

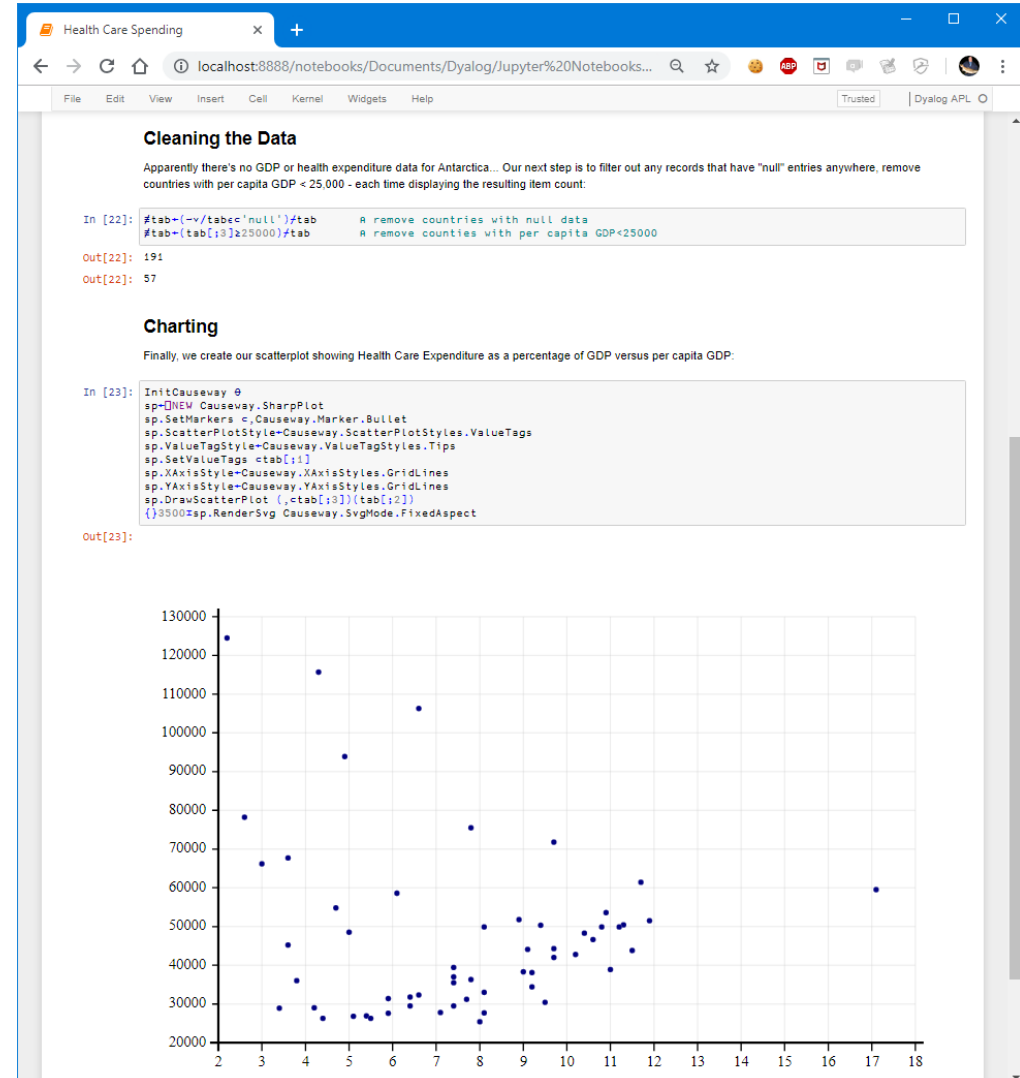
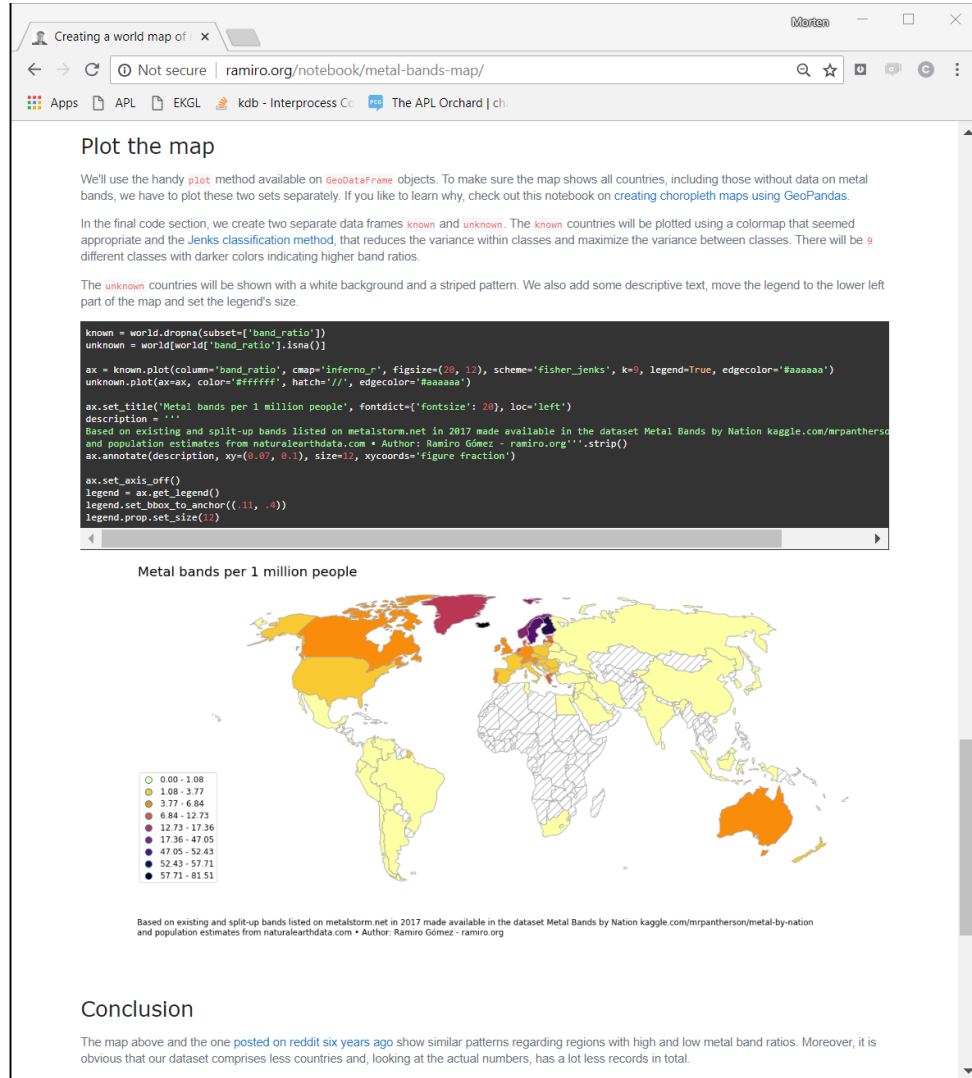
<https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>

What is Jupyter Notebook

A notebook combines the functionality of

- a word processor — handles formatted text
- a "shell" or "kernel" — executes statements in a programming language and includes output inline
- a rendering engine — renders HTML in addition to plain text

Jupyter Notebook Example



How to install it in aws

- SSH to the server like we did before (`ssh -i "your own key file" ubuntu@ec2-\[your own address\].us-east-2.compute.amazonaws.com`)
- Enter the python terminal (`ipython`)
- Generate your own key (`from IPython.lib import passwd; passwd()`) and **save** the generated key to you local PC.
- Exit the python terminal (`exit()`)

How to install it in aws

Generate the config file

- `jupyter notebook --generate-config`
- `mkdir certs`
- `cd certs`
- `openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem`
- `cd ~/.jupyter/`

How to install it in aws

Modify the config file by (vim jupyter_notebook_config.py), Copy the following text and paste them to the front of the config file.

```
c = get_config()
c.IPKernelApp.pylab = 'inline'
c.NotebookApp.certfile = u'/home/ubuntu/certs/mycert.pem'c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
# Your password below will be whatever you copied earlier
c.NotebookApp.password = u'[the key you just saved]'
c.NotebookApp.port = 8888
```

How to operate in Vim: (1) type 'i'; (2) paste all the above staff; (3) type 'Esc' to enter the control mode; (4) type ':wq' to save the modification and exit.

How to open port 8888 in aws

1

2

3

4

The screenshot shows the AWS Management Console interface. On the left, the 'EC2 Dashboard' sidebar is visible, with 'Security Groups' highlighted under the 'NETWORK & SECURITY' section. The main content area displays a list of security groups. The first group, 'launch-wizard-1' (ID: sg-00ca1798c055b60aa), is selected. Below the list, the 'Inbound' tab is active, and the 'Edit' button is visible. The 'Type' field in the inbound rule configuration is highlighted.

Name	Group ID	Group Name	VPC ID	Owner	Description
launch-wizard-1	sg-00ca1798c055b60aa	launch-wizard-1	vpc-33e51258	347795767038	launch-wizard-1 created 2019-09-22T11:13:55.082+08:00
default	sg-12c6d271	default	vpc-33e51258	347795767038	default VPC security group

Security Group: sg-00ca1798c055b60aa

Description Inbound Outbound Tags

Edit

Type	Protocol	Port Range	Source	Description
------	----------	------------	--------	-------------

How to open port 8888 in aws

1



Edit inbound rules ✕

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ	
Custom TCP ⓘ	TCP	8888	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop ✕
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop ✕

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel

Save



2

Start the server

- In the terminal (jupyter notebook)
- In your own browser ([https://\[your ip address\]:8888/](https://[your ip address]:8888/))

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name
		i-00271aa45d74c9374	t2.micro	us-east-2b	<div>running</div>	<div>2/2 checks ...</div>	None	ec2-3-15-204-231.us-e...	3.15.204.231	-	comp4:

The IP address of
your aws server

Instance: i-00271aa45d74c9374		Public DNS: ec2-3-15-204-231.us-east-2.compute.amazonaws.com	
Description	Status Checks	Monitoring	Tags
Instance ID	i-00271aa45d74c9374		
Instance state	running		
Instance type	t2.micro		
Elastic IPs			
Availability zone	us-east-2b		
Security groups	launch-wizard-1. view inbound rules . view outbound rules		
Scheduled events	No scheduled events		
AMI ID	ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20190722.1 (ami-05c1fa8df71875112)		
Public DNS (IPv4)	ec2-3-15-204-231.us-east-2.compute.amazonaws.com		
IPv4 Public IP	3.15.204.231		
IPv6 IPs	-		
Private DNS	ip-172-31-22-73.us-east-2.compute.internal		
Private IPs	172.31.22.73		
Secondary private IPs			
VPC ID	vpc-33e51258		
Subnet ID	subnet-deaaffa4		

The end

After typing your password, you should get something like this.



For more information, you can refer to:

<https://medium.com/@GalarnykMichael/aws-ec2-part-4-starting-a-jupyter-ipython-notebook-server-on-aws-549d87a55ba9#.ylckaikgc>

PS: the only difference between my instruction with this blog is that I didn't use sudo for the openssl command. Do it following me instead of the blog. This is important.

Practice (optional for this tutorial)

Kaggle Titanic competition

Competition Description

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

You can download the dataset and see the full code here:

<https://www.kaggle.com/dmilla/introduction-to-decision-trees-titanic-dataset>

Load data

```
# Imports needed for the script
import numpy as np
import pandas as pd
import re
import xgboost as xgb
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls

from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from IPython.display import Image as PImage
from subprocess import check_call
from PIL import Image, ImageDraw, ImageFont

# Loading the data
train = pd.read_csv('../input/train.csv')
test = pd.read_csv('../input/test.csv')

# Store our test passenger IDs for easy access
PassengerId = test['PassengerId']

# Showing overview of the train dataset
train.head(3)
```

Get a decision tree model

```
cv = KFold(n_splits=10)          # Desired number of Cross Validation folds
accuracies = list()
max_attributes = len(list(test))
depth_range = range(1, max_attributes + 1)

# Testing max_depths from 1 to max attributes
# Uncomment prints for details about each Cross Validation pass
for depth in depth_range:
    fold_accuracy = []
    tree_model = tree.DecisionTreeClassifier(max_depth = depth)
    # print("Current max depth: ", depth, "\n")
    for train_fold, valid_fold in cv.split(train):
        f_train = train.loc[train_fold] # Extract train data with cv indices
        f_valid = train.loc[valid_fold] # Extract valid data with cv indices

        model = tree_model.fit(X = f_train.drop(['Survived'], axis=1),
                                y = f_train["Survived"]) # We fit the model with the fold train data

        valid_acc = model.score(X = f_valid.drop(['Survived'], axis=1),
                                y = f_valid["Survived"]) # We calculate accuracy with the fold validation data

        fold_accuracy.append(valid_acc)

    avg = sum(fold_accuracy)/len(fold_accuracy)
    accuracies.append(avg)
    # print("Accuracy per fold: ", fold_accuracy, "\n")
    # print("Average accuracy: ", avg)
    # print("\n")

# Just to show results conveniently
df = pd.DataFrame({"Max Depth": depth_range, "Average Accuracy": accuracies})
df = df[["Max Depth", "Average Accuracy"]]
print(df.to_string(index=False))
```