**Installing TensorFlow in R with reticulate**

**Do this in R.** Install and load tidyverse, reticulate, and tensorflow.

# R

library(tidyverse)

library(reticulate)

library(tensorflow)

Next, run install\_tensorflow() in your R environment. This will take about 3-5 minutes to install TensorFlow in a new Conda Environment named **“py3.6”.**

# R

install\_tensorflow(

method = "conda",

version = "default", # Installs TF 2.0.0 (as of May 15, 2020)

envname = "py3.6",

conda\_python\_version = "3.6",

extra\_packages = c("matplotlib", "numpy", "pandas", "scikit-learn")

)

***Side note:*** You can actually specify which TensorFlow version to install with the version arg. This can be helpful to switch from the CPU vesion to GPU version (greater power, greater responsibility) or to access older versions of TF.

We can check to see that py3.6 conda environment has been created.

# R

conda\_list()

## name python

## 1 anaconda3 /Users/mdancho/opt/anaconda3/bin/python

## 2 py3.6 /Users/mdancho/opt/anaconda3/envs/py3.6/bin/python

## 3 py3.7 /Users/mdancho/opt/anaconda3/envs/py3.7/bin/python

## 4 py3.8 /Users/mdancho/opt/anaconda3/envs/py3.8/bin/python

Next, we tell reticulate to use the py3.6 conda environment.

# R

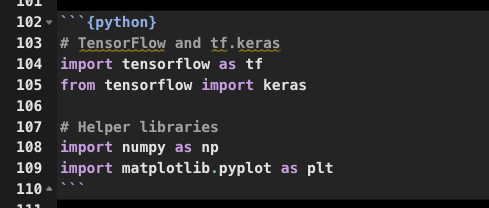
use\_condaenv("py3.6", required = TRUE)

**Congrats on the Installation is now complete!** - Now Let’s Use TensorFlow to classify images.

**Image Recognition Analysis   
To Verify TensorFlow is Working**

**Step 1 - Make a Python Code Chunk**

Use [Pro-Tip #1 Below](https://www.business-science.io/learn-r/2020/05/15/setup-tensorflow-python-in-r.html#pro-tips) to make a *“Python Code Chunk”*.



Python Code Chunk

**Step 2 - Import Libraries**

Import the libraries needed:

* Deep Learning: tensorflow and keras
* Math: numpy
* Visualization: matplotlib

# Python

# TensorFlow and tf.keras

import tensorflow as tf

from tensorflow import keras

# Helper libraries

import numpy as np

import matplotlib.pyplot as plt

Check the version of tensorflow to make sure we’re using 2.0.0+.

# Python

print(tf.\_\_version\_\_)

## 2.0.0

**Step 3 - Load the Fashion Images**

Load the fashion\_mnist dataset from keras.

# Python

fashion\_mnist = keras.datasets.fashion\_mnist

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

We have 60,000 training images that have been labeled.

# Python

train\_images.shape

## (60000, 28, 28)

We can check the unique labels to see what classifications the images belong to. Note that these are numeric values ranging from 0 to 9.

# Python

np.unique(train\_labels)

## array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=uint8)

The corresponding labels are:

# Python

class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',

'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

We can see what the first image looks like using matplotlib.

# Python

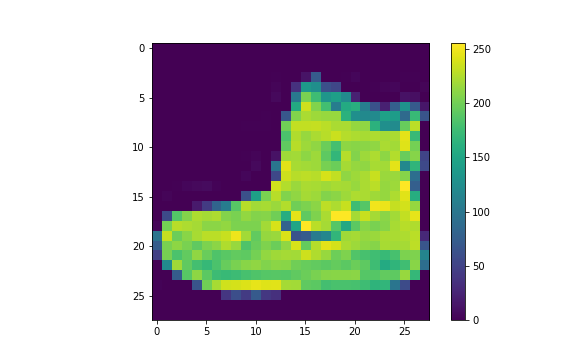
plt.figure()

plt.imshow(train\_images[0])

plt.colorbar()

plt.grid(False)

plt.show()



And we can also check out the first 25 images.

# Python

plt.figure(figsize=(10,10))

for i in range(25):

plt.subplot(5,5,i+1)

plt.xticks([])

plt.yticks([])

plt.grid(False)

plt.imshow(train\_images[i], cmap=plt.cm.binary)

plt.xlabel(class\_names[train\_labels[i]])

plt.show()



**Step 4 - Modeling with Keras**

Make a keras model using the Sequential() with 3 steps: Flatten, Dense, and Dense.

# Python

model = keras.Sequential([

keras.layers.Flatten(input\_shape=(28, 28)),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(10)

])

Next, compile the model with the “adam” optimizer.

# Python

model.compile(

optimizer = 'adam',

loss = tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics = ['accuracy']

)

Inspect the model summary.

# Python

model.summary()

## Model: "sequential"

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Layer (type) Output Shape Param #

## =================================================================

## flatten (Flatten) (None, 784) 0

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## dense (Dense) (None, 128) 100480

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## dense\_1 (Dense) (None, 10) 1290

## =================================================================

## Total params: 101,770

## Trainable params: 101,770

## Non-trainable params: 0

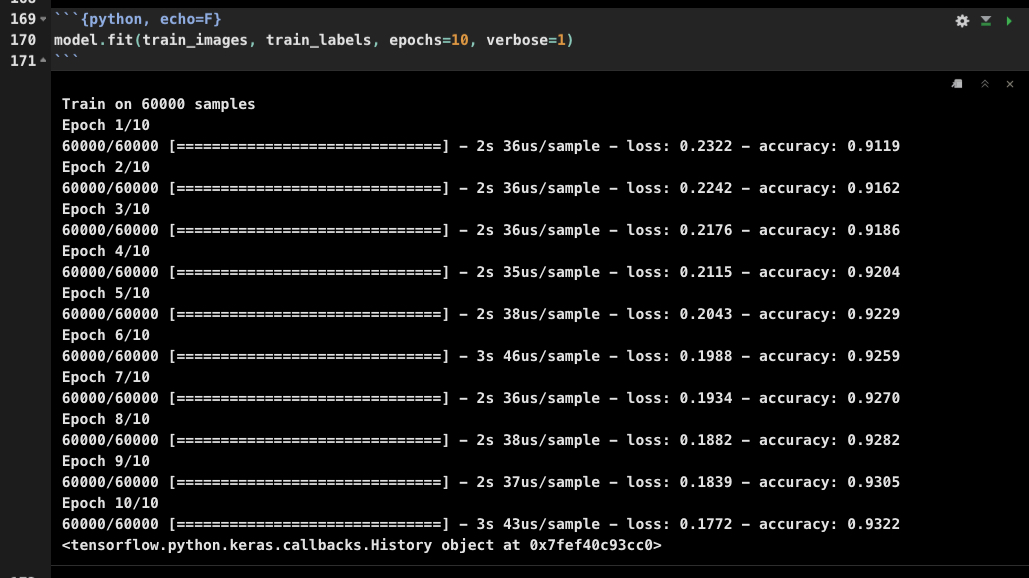
## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Step 5 - Fit the Keras Model**

CRITICAL STEP - Fit the model. Make sure this step works!</p>

# Python

model.fit(train\_images, train\_labels, epochs=10, verbose=1)



TensorFlow Model Training

**Step 6 - Training History**

# Python

history = model.history.history

history

## {'loss': [3.1226694132328032, 0.6653605920394262, 0.5747007430752118, 0.5286751741568247, 0.508751327864329, 0.5023731174985567, 0.48900006746848423, 0.4814090680360794, 0.4810072046995163, 0.47561218699614205], 'accuracy': [0.68145, 0.74085, 0.79331666, 0.8143, 0.8228833, 0.8244333, 0.8283167, 0.83428335, 0.8348, 0.83521664]}

I’ll plot this using R. Note - This is an R Code Chunk (not a Python Code Chunk).

# R Code Chunk (not Python)

py$history %>%

as\_tibble() %>%

unnest(loss, accuracy) %>%

rowid\_to\_column() %>%

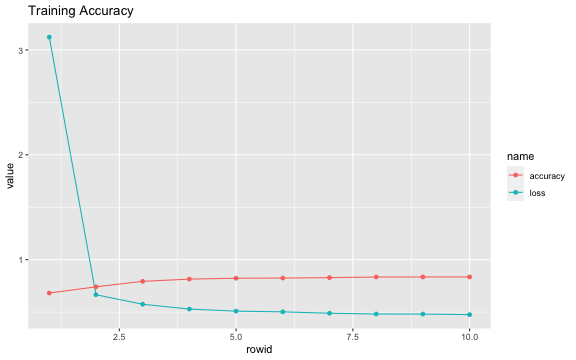
pivot\_longer(-rowid) %>%

ggplot(aes(rowid, value, color = name)) +

geom\_line() +

geom\_point() +

labs(title = "Training Accuracy")



**Step 7 - Test Accuracy**

Evaluate accuracy on the out-of-sample images.

# Python

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

## 10000/1 - 0s - loss: 0.4470 - accuracy: 0.8098

**Step 8 - Make Predictions**

The model produces linear outputs cakked “logits”. The softmax layer to converts the logits to probabilities.

# Python

probability\_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])

We can then classify all of the test images (held out)

# Python

predictions = probability\_model.predict(test\_images)

We can make a prediction for the first image.

# Python

predictions[0]

## array([7.7921860e-21, 3.3554016e-13, 0.0000000e+00, 1.8183892e-15,

## 0.0000000e+00, 4.0730215e-03, 8.1443369e-20, 4.2793788e-03,

## 2.6940727e-18, 9.9164760e-01], dtype=float32)

Use np.argmax() to determine which index has the highest probability.

# Python

np.argmax(predictions[0])

## 9

The index value can be retrieved with np.max().

# Python

np.max(predictions[0])

## 0.9916476

Get the class name.

# Python

class\_names[np.argmax(predictions[0])]

## 'Ankle boot'

And visualize the image.

# Python

plt.figure()

plt.imshow(test\_images[0])

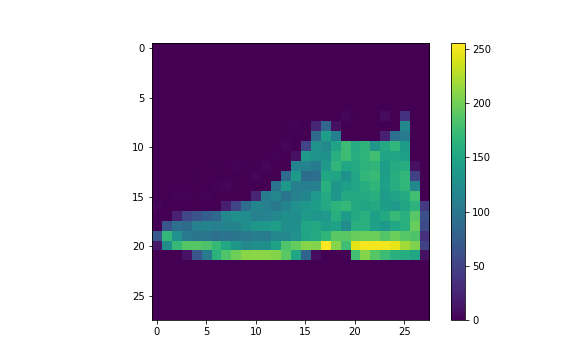
plt.colorbar()

## <matplotlib.colorbar.Colorbar object at 0x7fee14906240>

# Python

plt.grid(False)

plt.show()



**Nice work** - If you made it through this tutorial unscathed, then you are doing well!

**Pro Tips (Python in R)**

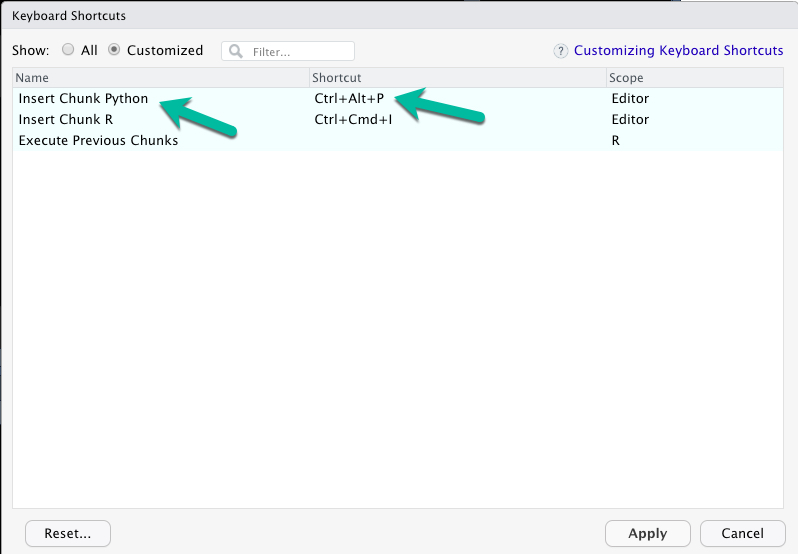
Now that you have python running in R, use these pro-tips to make your experience way more enjoyable.

**Pro-Tip #1 - Python Chunk Keyboard Shortcut**

I can’t stress this one enough - **Set up a Keyboard shortcut for Python Code Chunks.** This is a massive productivity booster for Rmarkdown documents.

* My preference: Ctrl + Alt + P

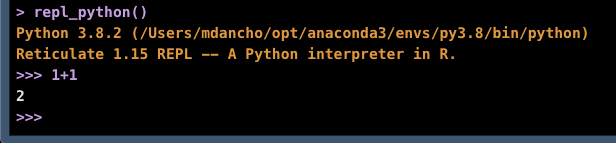
When you hit Ctrl + Alt + P, a {python} code chunk will appear in your R Markdown document.



**Pro-Tip #2 - Use Python Interactively**

For debugging Python Code Chunks in R Markdown, it can help to use the repl\_python() to convert your Console to a Python Code Console. To do so:

* In R Console, you can run python interactively using repl\_python(). You will see >>> indicating you are in Python Mode.
* Make sure the correct Python / Conda Environment is selected.
* To escape Python in the console, just hit escape.



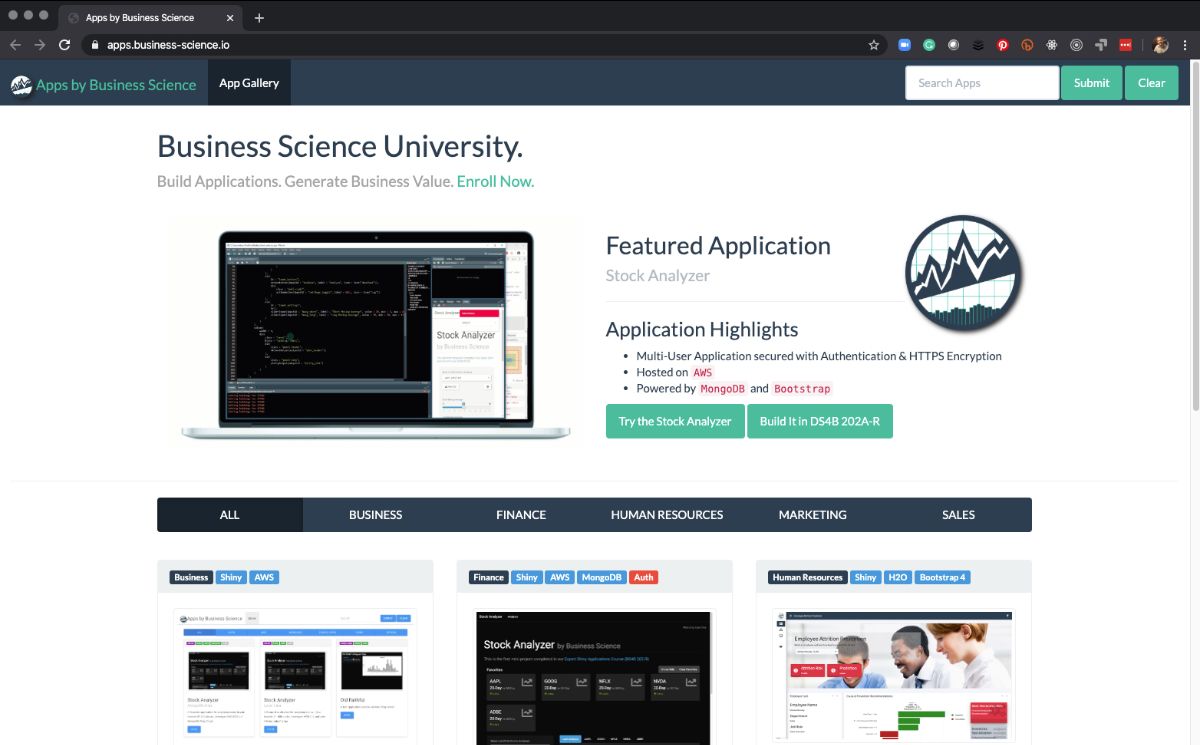
**Pro-Tip #3 - My Top 4 Conda Terminal Commands**

At some point you will need to create, modify, add more packages to your Conda Environment(s). Here are 4 useful commands:

1. Run conda env list to list the available conda environments
2. Run conda activate <env\_name> to activate a conda environment
3. Run conda update --all to update all python packages in a conda environment.
4. Run conda install <package\_name> to install a new package

**Use Python inside Shiny Apps**

Up until now we haven’t talked about Shiny - the ***web application framework*** that is used to take your python and R machine learning models into ***Production.***

[](https://apps.business-science.io/)

[Business Science Application Library](https://apps.business-science.io/)   
A Meta-Application that houses Shiny Apps

**R Shiny needs to be in your toolbox if you want to productionize Data Science.** You simply cannot put machine learning applications into production with other “BI” Tools like Tableau, PowerBI, and QlikView.

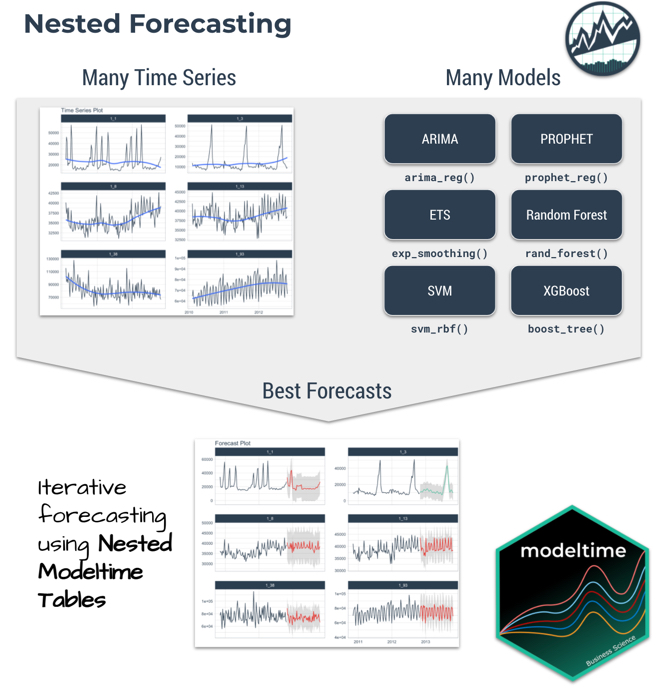
**CRITICAL POINT: You can USE SHINY to productionize Scikit Learn and TensorFlow models.**

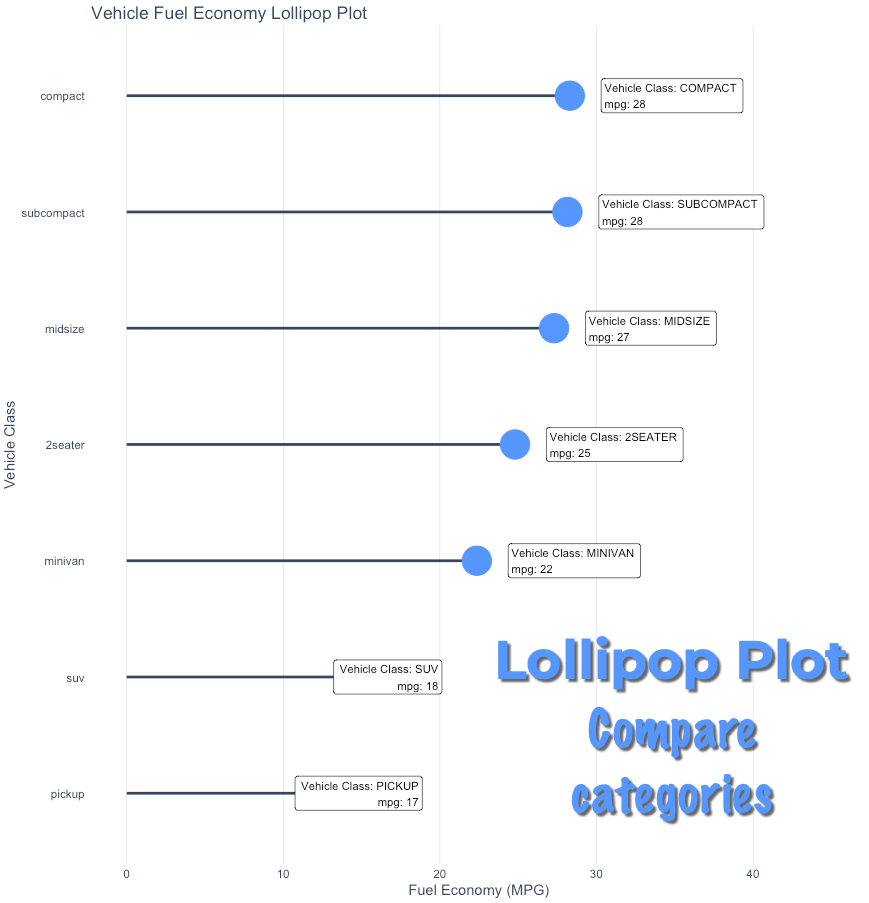
**If you need to learn R Shiny as fast as possible**, I have the perfect program for you. It will accelerate your career.

**Our program is different. We've helped 1000+ students like you.**

**Expert-level Machine Learning skills** and **Predictive Web Application skills**,  
**No experience required!**

**Finding it Difficult to Connect Data Science to Business?**

[[](https://www.business-science.io/code-tools/2021/08/26/modeltime-iterative-forecasting.html)](https://www.business-science.io/code-tools/2021/08/26/modeltime-iterative-forecasting.html)

[[](https://www.business-science.io/r/2021/08/24/ggalt-lollipop-plots-ggplot2.html)](https://www.business-science.io/r/2021/08/24/ggalt-lollipop-plots-ggplot2.html)