# Prepare Notebook for Predicting Budget Votes

* In the collateral directory from the github project copy the house-votes-84.csv to the notebooks directory
* In a browser open the Juypiter app, this is likely at <http://localhost:8888/>
* Enter the notebook directory
* Click new “Python 3” to create a new notebook and save it as “HouseVotes”
* Add the following imports

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow import keras

# Load the csv Data

* Use the CsvDataset to load the data from the csv files

#Read in Data

filename = "house-votes-84.csv"

csvData = tf.data.experimental.CsvDataset(

filename,

[tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string,tf.string],

header=True

)

Notice that there is a large number of columns in here, all of type string. Some are y/n and one is the party affiliation, democrat or republican.

# Split the data

* Split up the data into two sets. 80% going to training and 20% to scoring.

#Split the data

def is\_test(x, y):

return x % 5 == 0

def is\_train(x, y):

return not is\_test(x, y)

def get\_party(data):

if (data.decode("utf-8") == 'democrat'):

return 1

else:

return 0

def get\_isTrue(data):

if (data.decode("utf-8") == 'y'):

return 1

else:

return 0

def convert\_to\_Array(data, colMapping):

returnValue = []

for item in (data.enumerate().map(colMapping)).\_\_iter\_\_():

returnValue.append(get\_isTrue(item[1][3].numpy()))

return tf.constant(returnValue, dtype=tf.int16)

def convert\_to\_2Array(data):

returnValue = []

for item1, item2 in (data.enumerate()).\_\_iter\_\_():

returnValue.append([get\_party(item2[1][0].numpy()), get\_isTrue(item2[1][1].numpy()), get\_isTrue(item2[1][2].numpy()), get\_isTrue(item2[1][4].numpy()), get\_isTrue(item2[1][5].numpy()), get\_isTrue(item2[1][6].numpy())])

return tf.constant(returnValue, dtype=tf.int16)

test\_dataset = csvData.enumerate().filter(is\_test)

train\_dataset = csvData.enumerate().filter(is\_train)

train\_x = convert\_to\_2Array(train\_dataset)

train\_y = convert\_to\_Array(train\_dataset, lambda x,y: y)

test\_x = convert\_to\_2Array(test\_dataset)

test\_y = convert\_to\_Array(test\_dataset, lambda x, y: y)

in our prepare the data step, we are not only splitting the data but converting the values to numerics, ones and zeros

* Now we want to put toget the model

#compile model

model = keras.Sequential()

model.add(keras.layers.Dense(8, input\_dim=6, activation='relu'))

model.add(keras.layers.Dense(3, activation='softmax'))

# Compile model

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

Pay attention to the loss model we are using. That will come up in a bit

* Now that our model is set up, let’s train it.

#train model

history = model.fit(train\_x, train\_y,

epochs=500,

batch\_size=100,

validation\_data=(test\_x, test\_y))

* Finally, we’ll look at the results

#view accuracy

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

plt.figure(figsize=(8, 8))

plt.subplot(2, 1, 1)

plt.plot(acc, label='Training Accuracy')

plt.plot(val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.ylabel('Accuracy')

plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)

plt.plot(loss, label='Training Loss')

plt.plot(val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.ylabel('Cross Entropy')

plt.title('Training and Validation Loss')

plt.xlabel('epoch')

plt.show()

How happy were you with the results? I’m betting you were not. Try changing the loss tool to from 'categorical\_crossentropy' to ‘sparse\_categorical\_crossentropy’. Now run the model compile, training and accuracy steps. Were you happier with the results? If so, what does that tell you about the importance of properly setting up our models?

* If you want, when you are complete you can save your trained model.

tf.saved\_model.save(model, "budget\_votes/1/")