Programming Assignment #2

T81-559: Applications of Deep Neural Networks, Washington University September 11, 2016

Listing 1 shows a sample submission skeleton that you can use as a starting point for this assignment.

Listing 1: Sample Submission Skeleton

```
# Programming Assignment #2 by Jeff Heaton
# T81-558: Application of Deep Learning
import os
import sklearn
import pandas as pd
import numpy as np
import tensorflow.contrib.learn as skflow
from sklearn.cross_validation import KFold
from scipy.stats import zscore
from sklearn import metrics
from sklearn import preprocessing
from sklearn.cross_validation import KFold
from sklearn.cross_validation import train_test_split

path = "./data/"

# These four functions will help you, they were covered in class.
# Encode a text field to dummy variables
def encode_text_dummy(df,name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = "{}-{}".format(name,x)
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)

# Encode a text field to a single index value
def encode_text_index(df,name):
    le = preprocessing.LabelEncoder()
    df[name] = le.fit_transform(df[name])
    return le.classes_

# Encode a numeric field to Z-Scores
def encode_numeric_zscore(df,name,mean=None,sd=None):
    if mean is None:
        mean = df[name].mean()

    if sd is None:
        sd = df[name].std()

    df[name] = (df[name]-mean)/sd

# Encode a numeric field to fill missing values with the median.
def missing_median(df, name):
    med = df[name].median()
    df[name] = df[name].fillna(med)

# Convert a dataframe to x/y suitable for training.
def to_xy(df,target):
    result = []
    for x in df.columns:
        if x != target:
            result.append(x)
    return df.as_matrix(result),df[target]

# Encode the toy dataset
def question1():
    print()
    print("***Question 1***")

    path = "./data/"

    filename_read = os.path.join(path,"toy1.csv")
    filename_write = os.path.join(path,"submit-jheaton-prog2q1.csv")
    df = encode_toy_dataset(filename_read) # You just have to implement encode_toy_dataset above
    df.to_csv(filename_write,index=False)
    print("Wrote {} lines.".format(len(df)))


# Model the toy dataset, no cross validation
def question2():
    print()
    print("***Question 2***")

def question3():
    print()
    print("***Question 3***")

    # Z-Score encode these using the mean/sd from the dataset (you got this in question 2)
    testDF = pd.DataFrame([
            {'length':1, 'width':2, 'height': 3},
            {'length':3, 'width':2, 'height': 5},
```

```python
            {'length':4, 'width':1, 'height': 3}
        ])


def question4():
    print()
    print("***Question 4***")


def question5():
    print()
    print("***Question 5***")


question1()
question2()
question3()
question4()
question5()
```

Listing 2 shows what the output from this assignment would look like. Your numbers might differ from mine slightly. Every question, except 2, also generates an output CSV file. For your submission please include your Jupyter notebook and any generated CSV files that the questions specified. Name your output CSV files something such as **submit-jheaton-prog2q1.csv**. Submit a ZIP file that contains your Jupyter notebook and 4 CSV files to Blackboard. This will be 5 files total.

Listing 2: Expected Output

```
***Question 1***
Wrote 10000 lines.

***Question 2***
Step #99, avg. train loss: 1428641.75000
...Lines removed for space...
Step #10000, epoch #42, avg. train loss: 31899.88477
Out of sample (RMSE): 249.13940526005305

***Question 3***
length: (5.5259, 2.8610396265648745)
width: (5.5336999999999996, 2.8597695953020104)
height: (5.5338000000000003, 2.8721215970510063)
     height    length     width
0 -0.882205 -1.581907 -1.235659
1 -0.185856 -0.882861 -1.235659
2 -0.882205 -0.533338 -1.585338
```

```

***Question 4***
Fold #1
Step #50, epoch #12, avg. train loss: 0.07184, avg. val loss: 0.05284
...Lines removed for space...
Step #250, epoch #62, avg. train loss: 0.01217, avg. val loss: 0.01708
Stopping. Best step:
 step 97 with loss 0.013468504883348942
Fold score (RMSE): 0.17531210760832266
Fold #2
Step #50, epoch #12, avg. train loss: 0.07019, avg. val loss: 0.04958
...Lines removed for space...
Step #550, epoch #137, avg. train loss: 0.01109, avg. val loss: 0.01040
Fold score (RMSE): 0.14272974831031332
Fold #3
Stopping. Best step:
 step 362 with loss 0.009098603390157223
Step #50, epoch #12, avg. train loss: 0.06759, avg. val loss: 0.06043
...Lines removed for space...
Step #1300, epoch #325, avg. train loss: 0.00812, avg. val loss: 0.01476
Fold score (RMSE): 0.161749386296102
Fold #4
Stopping. Best step:
 step 1106 with loss 0.011062948033213615
Step #50, epoch #12, avg. train loss: 0.06714, avg. val loss: 0.07087
...
Step #600, epoch #150, avg. train loss: 0.00964, avg. val loss: 0.02152
Stopping. Best step:
 step 438 with loss 0.020158855244517326
Fold score (RMSE): 0.21273141903164636
Fold #5
Step #50, epoch #12, avg. train loss: 0.06887, avg. val loss: 0.05595
...
Step #950, epoch #237, avg. train loss: 0.00902, avg. val loss: 0.01661
Stopping. Best step:
 step 786 with loss 0.011288278736174107
Fold score (RMSE): 0.18455804191270428
Final, out of sample score (RMSE): 0.17696627225086287

***Question 5***
Fold #1
Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.37356
...Lines removed for space...
Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.03402
Fold score: 0.9875
Fold #2
Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.44299
```

```
...Lines removed for space...
Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.03660
Fold score: 1.0
Fold #3
Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.44690
...Lines removed for space...
Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.06254
Fold score: 0.9625
Fold #4
Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.64565
...Lines removed for space...
Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.18395
Fold score: 0.9367088607594937
Fold #5
Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.50485
...Lines removed for space...
Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.11250
Fold score: 0.9620253164556962
Final, out of sample score: 0.9698492462311558
```

Question 1

Use the dataset found here for this question: [click for toy dataset].

Encode the **toy1.csv** dataset. Generate dummy variables for the shape and metal. Encode height, width and length as z-scores. Include, but do not encode the weight. If this encoding is performed in a function, named **encode_toy_dataset**, you will have an easier time reusing the code from question 1 in question 2.

Write the output to a CSV file that you will submit with this assignment. The CSV file will look similar to Listing 3.

Listing 3: Question 2 Output Sample

```
height,length,width,weight,metal-bronze,metal-gold,metal-platinum,metal-↩
    silver,metal-tin,shape-box,shape-cylinder,shape-sphere
-0.18585564084337075, -0.18381430131795315, -0.1866234261937586, ↩
    729.63,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0
-0.5340303145851667, -0.18381430131795315, 0.16305509393694917, ↩
    2530.8,0.0,0.0,1.0,0.0,0.0,1.0,0.0,0.0
-1.2303796620687586, -1.2323841890415879, -1.5853375067165896, ↩
    58.37,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0
-0.8822049883269626, -0.8828608931337095, -0.8859804664551741, ↩
    74.15,0.0,0.0,0.0,1.0,0.0,0.0,0.0,1.0
...
```

Question 2

Use the dataset found here for this question: [click for toy dataset].

Use the encoded dataset from question 1 and train a neural network to predict weight. Use 25% of the data as validation and 75% as training, make sure you shuffle the data. Report the RMSE error for the validation set. No CSV file need be generated for this question.

Question 3

Use the dataset found here for this question: [click for toy dataset].

Using the **toy1.csv** dataset calculate and report the mean and standard deviation for height, width and length. Calculate the z-scores for the dataframe given by Listing 4. Make sure that you use the mean and standard deviations you reported for this question. Write the results to a CSV file.

Listing 4: Question 3 Input Data

```
    testDF = pd.DataFrame([
            {'length':1, 'width':2, 'height': 3},
            {'length':3, 'width':2, 'height': 5},
            {'length':4, 'width':1, 'height': 3}
        ])
...
```

Your resulting CSV file should look almost exactly like Listing 5.

Listing 5: Question 3 Output Sample

```
height,length,width
-0.8822049883269626,-1.5819074849494659,-1.2356589865858818
-0.18585564084337075,-0.8828608931337095,-1.2356589865858818
-0.8822049883269626,-0.5333375972258314,-1.5853375067165896
```

Question 4

Use the dataset found here for this question: [click for iris dataset].

Usually the **iris.csv** dataset is used to classify the species. Not this time! Use the fields species, sepal-l, sepal-w, and petal-l to predict petal-w. Use a 5-fold cross validation and report ONLY out-of-sample predictions to a CSV file. Make sure to shuffle the data. Your generated CSV file should look similar to Listing 6. Encode each of the inputs in a way that makes sense (e.g. dummies, z-scores).

Listing 6: Question 4 Output Sample

```
sepal_l,sepal_w,petal_l,petal_w,species-Iris-setosa,species-Iris-↩
    versicolor,species-Iris-virginica,0,0
0.30995914214417364, -0.5903951331558184, 0.5336208818725668, ↩
    1.2,0.0,1.0,0.0,1.2,1.444551944732666
-0.1730940663922016, 1.7038864723719687, -1.1658086782311483, ↩
    0.3,1.0,0.0,0.0,0.3,0.\
...
```

Question 5

Use the dataset found here for this question: [click for auto mpg dataset].

Usually the **auto-mpg.csv** dataset is used to regress the mpg. Not this time! Use the fields to predict how many cylinders the car has. Treat this as a classification problem, where there is a class for each number of cylinders. Use a 5-fold cross validation and report ONLY out-of-sample predictions to a CSV file. Make sure to shuffle the data. Your generated CSV file should look similar to Listing 7. Encode each of the inputs in a way that makes sense (e.g. dummies, z-scores). Report the final out of sample accuracy score.

Listing 7: Question 4 Output Sample

```
mpg,cylinders,displacement,horsepower,weight,acceleration,year,origin,name↩
    ,ideal,predict
-0.7055506566787514, 8, 1.0892327311042995, 0.6722714619460141, ↩
    0.6300768256149949, -1.2938698102195594, 70, -0.7142457922976494,↩
    chevrolet chevelle malibu,8,8
-1.0893794720944747, 8, 1.5016242793620063, 1.5879594901955474, ↩
    0.8532590135498572, -1.4751810504376373, 70, -0.7142457922976494,buick↩
     skylark 320,8,8
-0.7055506566787514, 8, 1.1947282434492943, 1.19552176380289, ↩
    0.5497784722839334, -1.6564922906557151, 70, -0.7142457922976494,↩
    plymouth satellite,8,8
...
```