Abbreviated data preparation for the Wine Quality Problem (Project 1)

```
# Import Required Libraries
import matplotlib.pyplot as plt
import numpy as np
# This time we need to also import pandas
import pandas as pd

# Read in white wine data
# USES PANDAS (pd) to create a PANDAS DataFrame OBJECT:
white = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-

# Read in red wine data
# USES PANDAS (pd) to create a PANDAS DataFrame OBJECT:
red = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-re

# Add `type` column to `red` with price one  - done with PANDAS
red['type'] = 1

# Add `type` column to `white` with price zero - done with PANDAS
white['type'] = 0

# Append `white` to `red` - done with PANDAS
# AFTER THIS WE HAVE ALL WINES (red and white) in a SINGLE pandas DataFrame
wines = red.append(white, ignore_index = True)

# Import SKLEARN
import sklearn
# Import `train_test_split` from `sklearn.model_selection`
from sklearn.model_selection import train_test_split

# Specify the  data -
X1 = wines.iloc[:, 0:11]
X2 = wines.iloc[:,12]
X = pd.concat([X1,X2],axis = 1)

# Specify the QUALITY target labels and flatten the array
y = np.ravel(wines.quality)

# Splitting the data set for training and validating  - Done with SKLEARN
X_train, X_valid, y_train, y_valid = train_test_split(
                 X, y, test_size = 0.25, random_state = 45)



# CONVERTING X_train & X_test DataFrame s to TF tensors
# Will USE NumPy, TF & Keras after this
# import tensorflow as tf

Xtrain = X_train.to_numpy()
```

```
X_valid = X_valid.to_numpy()

# In reality:
# [1] ALL THE Xtrain patterns (with their y_train targets)
# will be used for TRAINING ([TR]), as Xtrain & y_train
# [2] MOST OF THE X_valid patterns (and their y_valid targets)
#  will be used for VALIDATION ([TT]),  as X_val & y_val
# BUT WE WILL SET ASIDE THE LAST 10 for "testing" ([TS])
#  as X_tst & y_tst


# To separate the last 10 in X_valid, let's first see the shape of X_valid
X_valid.shape
```

```
(1625, 12)
```

```
# And verify also the shape of y_valid
y_valid.shape
```

```
(1625,)
```

```
# Retain the first 1615 for validation ([TT])
Xval = X_valid[:1615]
Xval.shape
```

```
(1615, 12)
```

```
# and now set aside the last 10 for "test"
Xtst = X_valid[1615:]
Xtst.shape
```

```
(10, 12)
```

```
# SAME FOR THE CORRESPONDING TARGETS
# Retain the first 1615 for validation ([TT])
y_val = y_valid[:1615]
y_val.shape
```

```
(1615,)
```

```
y_tst = y_valid[1615:]
y_tst.shape
```

```
(10,)
```

```
y_tst
```

```
array([5, 5, 7, 6, 5, 5, 6, 6, 7, 6])
```

```
 # NOW, IN ADDITION, CREATE THE TARGETS AS ONE-HOT-ENCODED 4 quality LEVELS
```

```
# We will track these few targets through the conversion process
y_train[272:283]
```

```
array([5, 4, 6, 5, 5, 6, 7, 6, 5, 8, 5])
```

```
# Function create rank-1 arrays where 3,4,5,6,7,8,9 are mapped to 1 or 2 or 3 or 4
def to_4cs(x):
  lx = len(x)
  results = np.zeros(lx)
  for i in range(lx):
      # print( "start")
      xa = x[i];
      if xa <= 3:
          results[i] = 1      # 1, 2 and 3 map to Q-LEVEL 1 ( BAD Wine)
      elif xa <=6:
          results[i] = 2      # 4, 5 and 6 map to Q-LEVEL 2  (MEDIUM Wine)
      elif xa <=8:
          results[i] = 3      # 7 and 8 and 6 map to Q-LEVEL 3  (GOOD Wine)
      else:
          results[i] = 4      # 9 and above map to Q-LEVEL 4  (EXCELLENT Wine)
    # results[i, label] = 1.
  results = results.astype(int)
  return results


train_labels = to_4cs(y_train)
val_labels = to_4cs(y_val)
tst_labels = to_4cs(y_tst)
```

```
# Let's verify that the trainnig targets that we are tracking
# were converted to levels (1 = BAD; 2 = MEDIUM; 3 = GOOD; 4- EXCELLENT) correctly:
train_labels[272:283]
```

```
array([2, 2, 2, 2, 2, 2, 3, 2, 2, 3, 2])
```

```
# NOW, ONE-HOT ENCODING OF ALL 3  TARGET ARRAYS
# define a function to do the one-hot-encoding of output labels

def to_one_hot(labels, dimension=4):
    results = np.zeros((len(labels), dimension))
    for i, label in enumerate(labels-1):
        results[i, label] = 1.
    return results

one_hot_train_labels = to_one_hot(train_labels)
one_hot_val_labels = to_one_hot(val_labels)
one_hot_tst_labels = to_one_hot(tst_labels)
```

```
#Let's verify that the training targets we have tracked were
# one-hot encoded correctly
one_hot_train_labels[272:283,]
```

```
array([[0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 1., 0., 0.]])
```

```
# SO, AFTER EXECUTING THIS CELL, YOU WILL HAVE:
# FOR TRAINING:
#  Xtrain (4872, 12)...y_train (4872,)...train_labels(4872,)....one_hot_train_labels (4872,4)
# FOR VALIDATING:
#  Xval (1615, 12)...y_val (1615,)...val_labels(1615,)...one_hot_val_labels (1615,4)
# FOR TESTING:
#  Xtst (10, 12)...y_tst (10,)...tst_labels(10,)... one_hot_tst_labels (10,4)
# PLEASE DO NOT CHANGE THE NAMES OF THESE VARIABLES (So that instructor can use them)
```

++++ END OF THE DATA PREPARATION PART ++++

NOTE: THE COMMANDS HIGHLIGHTED IN YELLOW ARE JUST FOR "EXPLANATION". THEY CAN BE REMOVED IN THE STUDENT'S ACTUAL NOTEBOOKS FOR SUBMISSION