### Homework DL Starter Code and Instructions

### Tyler Echols

### Step 1: Description

This notebook performs regression to estimate the price of a car given various features.

### Step 2: Load the data

* upload the data
* put the data in a pandas dataframe
* output the data shape (rows, cols)
* output the first few rows of the data

[ ]

# load the csv file up into the cloud

[ ]

# load the data into a pandas data frame

# print the shape of the data frame

# display the first few rows

### Step 3 Data Exploration

[ ]

df.dtypes

model object

year int64

price int64

transmission object

mileage int64

fuelType object

tax int64

mpg float64

engineSize float64

dtype: object

[ ]

# change categorical column type from object to category

[ ]

# check for NAs

[ ]

# use describe() to examine the data

[ ]

# using seaborn, craete a lineplot() with year on the x axis and price on the y axis

[ ]

# create another plot exploring the data

# choose columns and plot type

### Step 4 Prepare Data

You can use the code below for this step.

[ ]

# set up X and y

X=df.drop(columns=['price'],axis=1)

y=df['price']

[ ]

import numpy as np

from sklearn.preprocessing import MinMaxScaler, LabelBinarizer

from sklearn.compose import ColumnTransformer

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X, y, test\_size=0.2)

# scale the numeric data

col\_list = ['year', 'mileage', 'tax', 'mpg', 'engineSize']

scaler = MinMaxScaler()

train\_numeric = scaler.fit\_transform(X\_train[col\_list])

test\_numeric = scaler.transform(X\_test[col\_list])

# one-hot encode the categorical data for model, transmission, and fuelType

# model

zipBinarizer = LabelBinarizer().fit(df['model'])

train\_model = zipBinarizer.transform(X\_train['model'])

test\_model = zipBinarizer.transform(X\_test['model'])

# transmission

zipBinarizer = LabelBinarizer().fit(df['transmission'])

train\_transmission = zipBinarizer.transform(X\_train['transmission'])

test\_transmission = zipBinarizer.transform(X\_test['transmission'])

# fuelType

zipBinarizer = LabelBinarizer().fit(df['fuelType'])

train\_fuelType = zipBinarizer.transform(X\_train['fuelType'])

test\_fuelType = zipBinarizer.transform(X\_test['fuelType'])

# concatenate

X\_train\_input = np.hstack([train\_numeric, train\_model, train\_transmission, train\_fuelType])

X\_test\_input = np.hstack([test\_numeric, test\_model, test\_transmission, test\_fuelType])

print(X\_train\_input[:3])

[[1. 0.02824203 0.25 0.14937759 0.28846154 0.

0. 0. 0. 0. 0. 0.

0. 1. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 1. 0. 0. 0.

1. ]

[0.95454545 0.03505107 0.25862069 0.06224066 0.48076923 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 1. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 1. 0. 0.

1. ]

[0.86363636 0.04766921 0.05172414 0.21873148 0.26923077 0.

0. 1. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 1. 0. 0. 0.

1. ]]

### Step 5 Linear regression

Run linear regression in sklearn.

[ ]

# train the algorithm

[ ]

# make predictions

[ ]

# evaluation on the test data using mse, mae, and r2\_score

[ ]

# display the first 5 predictions

[ ]

# display the first 5 actual values

### Regression in Keras

[ ]

# build a sequential model

# you choose the architecture

[ ]

# compile the model

[ ]

# train the model

[ ]

# output test mse score, test mae score

### Step 7 Commentary

Answer the following questions:

a. Compare metrics from sklearn and Keras.

b. Explore the data a bit more to speculate on why you achieved the results you got.

c. Describe all the architectures/hyperparameters you tried and the results. What do you conclude?