Assignment 6 (key)

1. Enable GPU:

```
import tensorflow as tf

# display tf version and test if GPU is active
tf.__version__, tf.test.gpu_device_name()
```

```
('2.3.0', '/device:GPU:0')
```

2. Load Boston data from Scikit-Learn:

```
from sklearn import datasets

dataset = datasets.load_boston()
data, target = dataset.data, dataset.target
```

3. Inspect data.

   a. get keys:

```
dataset.keys()
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

   b. get feature names:

```
feature_names = dataset.feature_name
feature_names
```

   c. display datatype of feature and target datasets:

```
type(dataset.data), type(dataset.target)
```

```
(numpy.ndarray, numpy.ndarray)
```

   d. get shapes of feature and target data:

```
dataset.data.shape, dataset.target.shape
```

```
((506, 13), (506,))
```

4. Convert data to pandas dataframes:

a. convert feature dataset to dataframe:

```
import pandas as pd

df_sklearn = pd.DataFrame(dataset.data, columns=feature_names)
```

b. add target dataset to dataframe:

```
df_sklearn['MEDV'] = dataset.target
```

5. Inspect dataframe.

a. display a few records:

```
df_sklearn.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

b. display basic information:

```
df_sklearn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

6. Remove noise.

   a. get shape before removing noise:

```
df_sklearn.shape
```

```
(506, 14)
```

   b. remove data with target price greater or equal to 50:

```
noise = df_sklearn.loc[df_sklearn['MEDV'] >= 50]
df_clean = df_sklearn.drop(noise.index)
```

   c. get shape after removing noise:

```
df_clean.shape
```

```
(490, 14)
```

7. Build the input pipeline:

   a. create a copy of the dataframe:

```
df = df_clean.copy()
```

b. create the target:

```
target = df.pop('MEDV')
```

c. convert feature and target data to numpy values:

```
features = df.values
labels = target.values
```

d. create train and test sets:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    features, labels, test_size=0.33, random_state=0)
```

e. scale feature data:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.fit_transform(X_test)
```

f. convert data to TensorFlow consumable slices:

```
train = tf.data.Dataset.from_tensor_slices(
    (X_train_std, y_train))
test = tf.data.Dataset.from_tensor_slices(
    (X_test_std, y_test))
```

g. shuffle (where appropriate), batch, and prefetch:

```
BATCH_SIZE = 16
SHUFFLE_BUFFER_SIZE = 100

train_ds = train.shuffle(
    SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE).prefetch(1)
test_ds = test.batch(BATCH_SIZE).prefetch(1)
```

8. Build model:

a. clear models and generate seed:

```
import numpy as np

tf.keras.backend.clear_session()
np.random.seed(0)
tf.random.set_seed(0)
```

b. get input shape:

```
in_shape = X_train.shape[1:]
in_shape
```

```
(13,)
```

c. import libraries:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

d. create model:

```
model = Sequential([
  Dense(64, activation='relu', input_shape=in_shape),
  Dense(64, activation='relu'),
  Dense(1)
])
```

e. inspect model:

```
model.summary()
```

f. compile (include **rmse** metric):

```
rmse = tf.keras.metrics.RootMeanSquaredError()

model.compile(loss='mse', optimizer='RMSProp',
              metrics=[rmse, 'mae', 'mse'])
```

g. train (include early stop with patience):

```
n = 4
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                              patience=n)

history = model.fit(train_ds, epochs=50,
                    validation_data=test_ds,
                    callbacks=[early_stop])
```

h. generalize:

```
model.evaluate(test_ds)
```

9. Predictions.

a. get predictions based on test data:

```
predictions = model.predict(test_ds)
```

b. compare first prediction against first actual label:

```
# predicted housing price
first = predictions[0]
print ('predicted price:', first[0], 'thousand')

# actual housing price
print ('actual price:', y_test[0], 'thousand')
```

c. plot predictions against actual labels:

(hint: use scatter plot)

```
import matplotlib.pyplot as plt

plt.scatter(y_test, predictions)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")
plt.show()
```