Bounding box detection - Racoon data

Data files

- images_racoon.rar: contain images of racoons
- train_labels.cv: contains coordinates for bounding box for every image

Import the necessary libraries

```
# IMPORT LIBRARIES AND PACKAGES
import tensorflow as tf
import csv
import numpy as np
from PIL import Image

from keras import Model
from keras.applications.mobilenet import MobileNet, preprocess_input
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, Callback
from keras.layers import Conv2D, Reshape
from keras.utils import Sequence
from keras.backend import epsilon
```

```
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the
%tensorflow_version 1.x magic: more info.
Using TensorFlow backend.
```

```
pip install patool
```

```
Collecting patool
  Downloading https://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d811096432
     |████████████████████████████████| 81kB 2.6MB/s
Installing collected packages: patool
Successfully installed patool-1.12
```

```
import patoolib
patoolib.extract_archive("/content/drive/My Drive/Residency 9/images_racoon.rar", outdir="/cc
```

Change directory

```
patool: ... /content/drive/My Drive/Residency 9/images_racoon.rar extracted to `/content
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318
Enter your authorization code:
··········
Mounted at /content/drive
```

Load the training data from train.csv file

```
import pandas as pd
import numpy as np

df = pd.read_csv('/content/drive/My Drive/Residency 9/train_labels.csv')
```

Print the shape of the train dataset

```
df.shape
```

```
(173, 8)
```

Declare a variable IMAGE_SIZE = 128 as we will be using MobileNet which will be Input shape as 128 * 128

```
ALPHA = 1.0 # Width hyper parameter for MobileNet (0.25, 0.5, 0.75, 1.0). Higher width means

IMAGE_SIZE = 128 # MobileNet takes images of size 128*128*3

EPOCHS = 10 # Number of epochs. I got decent performance with just 5.
BATCH_SIZE = 32 # Depends on your GPU or CPU RAM.

DATASET_FOLDER = "/content/drive/My Drive/Residency 9/"
TRAIN_CSV = DATASET_FOLDER+"train_labels.csv"
#VALIDATION_CSV = DATASET_FOLDER+"validation.csv"

images_zip_path = DATASET_FOLDER + "images_racoon.zip"

from zipfile import ZipFile
```

With the help of csv.reader write a for loop which can load the train.csv file and s path, width, height, x0,y0,x1,y1 in induvidual variables.

1. Create a list variable known as 'path' which has all the path for all the training images
2. Create an array 'coords' which has the resized coordinates of the bounding box for the training images

Note: All the training images should be downsampled to 128 * 128 as it is the input shape of MobileN we will be using for Object detection). Hence the corresponding coordinates of the bounding boxes s changed to match the image dimension of 128 * 128

```
import csv
with open(TRAIN_CSV, 'r') as csvfile:
    paths = []
    coords = np.zeros((sum(1 for line in csvfile)-1, 4))
    reader = csv.reader(csvfile, delimiter=',')
    csvfile.seek(0)## Reading at zero line
    next(reader) ## TO SKIP THE HEADER

    for col, row in enumerate(reader):

     # print(row)
      #for i, r in enumerate(row[1:8]): # Parse row with seven entities
            #print([ line[0][0] for  line in r])
            #row[i+1] = (r)
            #print(int(r))
        path, image_width, image_height,_, x0, y0, x1, y1 = row
        path = '/content/drive/My Drive/Residency 9/images/'+str(path)

        #path = "./" + path.split('/')[-2] + "/" + path.split('/')[-1]

        coords[col, 0] = int(x0) * IMAGE_SIZE / int(image_width) # Normalize bounding box by
        coords[col, 1] = int(y0) * IMAGE_SIZE / int(image_height) # Normalize bounding box by
        coords[col, 2] = (int(x1) - int(x0))* IMAGE_SIZE / int(image_width) # Normalize bounc
        coords[col, 3] = (int(y1) - int(y0)) * IMAGE_SIZE / int(image_height)
        paths.append(path)
```

Write a for loop which can load all the training images into a variable 'batch_imag the paths from the 'paths' variable

Note: Convert the image to RGB scale as the MobileNet accepts 3 channels as inputs

```
batch_images = np.zeros((len(paths), IMAGE_SIZE, IMAGE_SIZE, 3), dtype=np.float32)
for i, f in enumerate(paths):
    img = Image.open(f) # Read image
    img = img.resize((IMAGE_SIZE, IMAGE_SIZE)) # Resize image
    img = img.convert('RGB')
```

```
    batch_images[i] = preprocess_input(np.array(img, dtype=np.float32))
```

Import MobileNet and load MobileNet into a variable named 'model' which takes shape of 128 * 128 * 3. Freeze all the layers. Add convolution and reshape layers end to ensure the output is 4 coordinates

```
model = MobileNet(input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3), include_top=False, alpha=ALPHA) #
# Do not include classification (top) layer

# to freeze layers, except the new top layer, of course, which will be added below
for layer in model.layers:
    layer.trainable = False

# Add new top layer which is a conv layer of the same size as the previous layer so that only
x = model.layers[-1].output
x = Conv2D(4, kernel_size=4, name="coords")(x)
# In the line above kernel size should be 3 for img size 96, 4 for img size 128, 5 for img si
x = Reshape((4,))(x) # These are the 4 predicted coordinates of one BBox

model = Model(inputs=model.input, outputs=x)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

Downloading data from https://github.com/fchollet/deep-learning-models/releases/downloac
17227776/17225924 [==============================] - 0s 0us/step
```

Define a custom loss function IoU which calculates Intersection Over Union

```
gt = coords
def loss(gt,pred):
    intersections = 0
    unions = 0
    diff_width = np.minimum(gt[:,0] + gt[:,2], pred[:,0] + pred[:,2]) - np.maximum(gt[:,0], p
    diff_height = np.minimum(gt[:,1] + gt[:,3], pred[:,1] + pred[:,3]) - np.maximum(gt[:,1],
    intersection = diff_width * diff_height

    # Compute union
    area_gt = gt[:,2] * gt[:,3]
    area_pred = pred[:,2] * pred[:,3]
    union = area_gt + area_pred - intersection

#     Compute intersection and union over multiple boxes
    for j, _ in enumerate(union):
        if union[j] > 0 and intersection[j] > 0 and union[j] >= intersection[j]:
            intersections += intersection[j]
            unions += union[j]

    # Compute IOU. Use epsilon to prevent division by zero
    iou = np.round(intersections / (unions + epsilon()), 4)
    iou = iou.astype(np.float32)
    return iou

def IoU(y_true, y_pred):
    iou = tf.py_func(loss, [y_true, y_pred], tf.float32)
    return iou
```

Write model.compile function & model.fit function with:

1. Optimizer = Adam, Loss = 'mse' and metrics = IoU
2. Epochs = 30, batch_size = 32, verbose = 1

```
model.compile(optimizer='Adam', loss='mse', metrics=[IoU])
```

```
model.fit(batch_images,gt,
            epochs=32,batch_size = 32,
            verbose=1)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_

Epoch 1/32
173/173 [==============================] - 6s 34ms/step - loss: 2815.6249 - IoU: 0.0815
Epoch 2/32
173/173 [==============================] - 5s 28ms/step - loss: 685.1006 - IoU: 0.4228
Epoch 3/32
173/173 [==============================] - 5s 28ms/step - loss: 622.7765 - IoU: 0.5374
Epoch 4/32
173/173 [==============================] - 5s 28ms/step - loss: 547.4101 - IoU: 0.5450
Epoch 5/32
173/173 [==============================] - 5s 28ms/step - loss: 347.0228 - IoU: 0.5940
Epoch 6/32
173/173 [==============================] - 5s 28ms/step - loss: 273.6478 - IoU: 0.5890
Epoch 7/32
173/173 [==============================] - 5s 27ms/step - loss: 234.7158 - IoU: 0.6166
Epoch 8/32
173/173 [==============================] - 5s 28ms/step - loss: 189.4690 - IoU: 0.6616
Epoch 9/32
173/173 [==============================] - 5s 28ms/step - loss: 163.9423 - IoU: 0.7030
Epoch 10/32
173/173 [==============================] - 5s 28ms/step - loss: 146.9907 - IoU: 0.7167
Epoch 11/32
173/173 [==============================] - 5s 28ms/step - loss: 130.7645 - IoU: 0.7349
Epoch 12/32
173/173 [==============================] - 5s 28ms/step - loss: 114.1170 - IoU: 0.7402
Epoch 13/32
173/173 [==============================] - 5s 28ms/step - loss: 101.9557 - IoU: 0.7576
Epoch 14/32
173/173 [==============================] - 5s 28ms/step - loss: 95.0449 - IoU: 0.7684
Epoch 15/32
173/173 [==============================] - 5s 28ms/step - loss: 92.6005 - IoU: 0.7725
Epoch 16/32
173/173 [==============================] - 5s 28ms/step - loss: 86.8316 - IoU: 0.7857
Epoch 17/32
173/173 [==============================] - 5s 28ms/step - loss: 77.1729 - IoU: 0.7926
Epoch 18/32
173/173 [==============================] - 5s 28ms/step - loss: 75.1863 - IoU: 0.7956
Epoch 19/32
173/173 [==============================] - 5s 28ms/step - loss: 74.1508 - IoU: 0.7986
Epoch 20/32
173/173 [==============================] - 5s 28ms/step - loss: 67.9819 - IoU: 0.8209
Epoch 21/32
173/173 [==============================] - 5s 28ms/step - loss: 67.9373 - IoU: 0.8095
Epoch 22/32
173/173 [==============================] - 5s 28ms/step - loss: 71.1175 - IoU: 0.7979
Epoch 23/32
173/173 [==============================] - 5s 28ms/step - loss: 61.8706 - IoU: 0.8131
Epoch 24/32
173/173 [==============================] - 5s 28ms/step - loss: 63.4638 - IoU: 0.8228
Epoch 25/32
173/173 [==============================] - 5s 27ms/step - loss: 65.4807 - IoU: 0.8249
Epoch 26/32
173/173 [==============================] - 5s 29ms/step - loss: 60.0771 - IoU: 0.8297
Epoch 27/32
```

```
173/173 [==============================] - 5s 28ms/step - loss: 61.1672 - IoU: 0.8105
Epoch 28/32
173/173 [==============================] - 5s 28ms/step - loss: 64.6286 - IoU: 0.8262
Epoch 29/32
173/173 [==============================] - 5s 28ms/step - loss: 59.3722 - IoU: 0.8311
Epoch 30/32
173/173 [==============================] - 5s 28ms/step - loss: 60.2982 - IoU: 0.8319
Epoch 31/32
173/173 [==============================] - 5s 28ms/step - loss: 60.1770 - IoU: 0.8264
Epoch 32/32
173/173 [==============================] - 5s 28ms/step - loss: 60.6307 - IoU: 0.8296
<keras.callbacks.History at 0x7fd14ec53c88>
```

```
model.summary()
```

conv_dw_3_relu (ReLU)	(None, 32, 32, 128)	0
conv_pw_3 (Conv2D)	(None, 32, 32, 128)	16384
conv_pw_3_bn (BatchNormaliza	(None, 32, 32, 128)	512
conv_pw_3_relu (ReLU)	(None, 32, 32, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, 33, 33, 128)	0
conv_dw_4 (DepthwiseConv2D)	(None, 16, 16, 128)	1152
conv_dw_4_bn (BatchNormaliza	(None, 16, 16, 128)	512
conv_dw_4_relu (ReLU)	(None, 16, 16, 128)	0
conv_pw_4 (Conv2D)	(None, 16, 16, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, 16, 16, 256)	1024
conv_pw_4_relu (ReLU)	(None, 16, 16, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, 16, 16, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, 16, 16, 256)	1024
conv_dw_5_relu (ReLU)	(None, 16, 16, 256)	0
conv_pw_5 (Conv2D)	(None, 16, 16, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, 16, 16, 256)	1024
conv_pw_5_relu (ReLU)	(None, 16, 16, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, 17, 17, 256)	0
conv_dw_6 (DepthwiseConv2D)	(None, 8, 8, 256)	2304
conv_dw_6_bn (BatchNormaliza	(None, 8, 8, 256)	1024
conv_dw_6_relu (ReLU)	(None, 8, 8, 256)	0
conv_pw_6 (Conv2D)	(None, 8, 8, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_pw_6_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_dw_7_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_7 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, 8, 8, 512)	2048

conv_pw_7_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_8_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_dw_8_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_8 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_pw_8_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_dw_9_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_9 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_9_bn (BatchNormaliza	(None, 8, 8, 512)	2048
conv_pw_9_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_10_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_dw_10_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_10 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_10_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_pw_10_relu (ReLU)	(None, 8, 8, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, 8, 8, 512)	4608
conv_dw_11_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_dw_11_relu (ReLU)	(None, 8, 8, 512)	0
conv_pw_11 (Conv2D)	(None, 8, 8, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, 8, 8, 512)	2048
conv_pw_11_relu (ReLU)	(None, 8, 8, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 9, 9, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, 4, 4, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, 4, 4, 512)	2048

```
conv_dw_12_relu (ReLU)       (None, 4, 4, 512)         0
_________________________________________________________________
conv_pw_12 (Conv2D)          (None, 4, 4, 1024)        524288
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 4, 4, 1024)        4096
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 4, 4, 1024)        0
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 4, 4, 1024)        9216
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 4, 4, 1024)        4096
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 4, 4, 1024)        0
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 4, 4, 1024)        1048576
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 4, 4, 1024)        4096
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 4, 4, 1024)        0
_________________________________________________________________
coords (Conv2D)              (None, 1, 1, 4)           65540
_________________________________________________________________
reshape_1 (Reshape)          (None, 4)                 0
=================================================================
Total params: 3,294,404
Trainable params: 65,540
Non-trainable params: 3,228,864
_________________________________________________________________
```

Pick a test image from the given data

```
import cv2
filename = '/content/drive/My Drive/Residency 9/images/raccoon-102.jpg'
unscaled = cv2.imread(filename)
```

Resize the image to 128 * 128 and preprocess the image for the MobileNet mode

```
image height, image width,   = unscaled.shape
```

```
image = cv2.resize(unscaled, (IMAGE_SIZE, IMAGE_SIZE)) # Rescaled image to run the network
feat_scaled = preprocess_input(np.array(image, dtype=np.float32))
```

Predict the coordinates of the bounding box for the given test image

```
region = model.predict(x=np.array([feat_scaled]))[0]
```

Plot the test image using .imshow and draw a boundary box around the image w coordinates obtained from the model

```
x0 = int(region[0] * image_width / IMAGE_SIZE) # Scale the BBox
y0 = int(region[1] * image_height / IMAGE_SIZE)

x1 = int((region[2]) * image_width / IMAGE_SIZE)
y1 = int((region[3]) * image_height / IMAGE_SIZE)


import matplotlib.pyplot as plt
import matplotlib.patches as patches
from PIL import Image
import numpy as np


# Create figure and axes
fig,ax = plt.subplots(1)

# Display the image
ax.imshow(unscaled)

# Create a Rectangle patch
rect = patches.Rectangle((x0, y0), (x1 - x0) , (y1 - y0) , linewidth=2, edgecolor='r', facecc

# Add the patch to the Axes
ax.add_patch(rect)

plt.show()
```

Time Series Prediction using LSTM

Download Data

Link: https://datamarket.com/data/set/2324/daily-minimum-temperatures-in-melbourne-australia-1990#!ds=2324&display=line

Description

Daily minimum temperatures in Melbourne, Australia, 1981-1990

Units: Degrees Celcius

Steps before loading

- Rename the column name with temprature values to "Temprature"
- In the last, there is one extra row in the data, remove it by opening the file and save it again.
- There are some values in Temprature column which have a "?" before them, they will give error, before them and save the file
- If you don't want to do these steps, just load the data file given by Great Learning.

Mount google drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

Change your present working directory

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from keras.utils import np_utils

## Directory is same as above so no need to change.
```

Load your data file

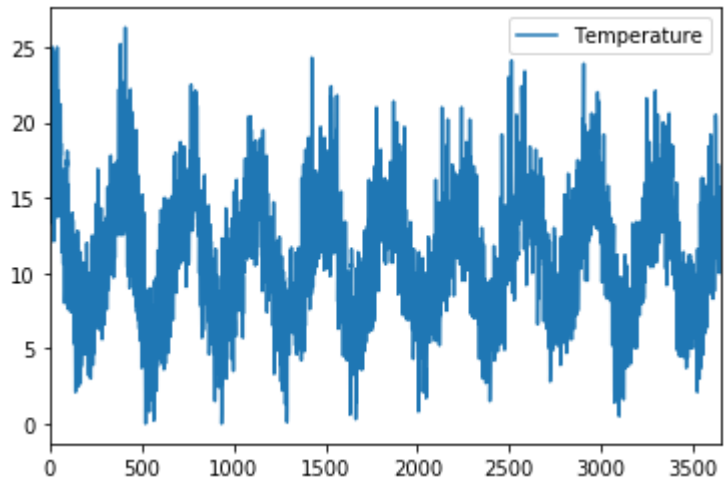
```
df = pd.read_csv('/content/drive/My Drive/Residency 9/daily-minimum-temperatures-in-me.csv')
df.sort_index(inplace=True)
df.tail(4)
```

	Date	Temperature
3646	1990-12-28	13.6
3647	1990-12-29	13.5
3648	1990-12-30	15.7
3649	1990-12-31	13.0

Plot data

```
df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd1451180b8>
```



Descibe your dataframe

```
df.shape
```

```
(3650, 2)
```

Check for null values

```
#Check for null values
df.isnull().sum()
```

```
Date           0
Temperature    0
dtype: int64
```

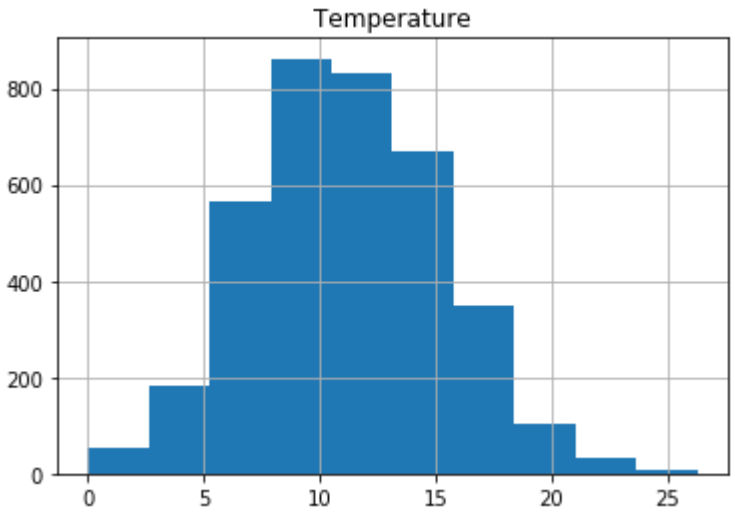
Drop null values

```
# Not required.
```

Get the representation of the distribution of data in the form of histogram

```
df.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd144065e10>]],
      dtype=object)
```



Check the maximum and minimum values

```
#Check Data Range
print('Min', np.min(df))
print('Max', np.max(df))
```

```
Min Date           1981-01-01
Temperature             0
dtype: object
Max Date           1990-12-31
Temperature          26.3
dtype: object
```

```
df.drop("Date", axis=1, inplace=True)
```

Normalize the data

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(df)
```

Check the maximum and minimum values of scaled data

```
#Check Data Range
print('Min', np.min(df))
print('Max', np.max(df))
```

```
Min Temperature    0.0
dtype: float64
Max Temperature    26.3
dtype: float64
```

Look into some of the scaled values

```
df.head(5)
```

	Temperature
0	20.7
1	17.9
2	18.8

Split data into Training and Testing

```
# 70% examples will used for training (in the begining)
train_size = int(len(scaled) * 0.70)

#30% will be used for Test
test_size = len(scaled - train_size)


#Split the data
train, test = scaled[0:train_size, :], scaled[train_size: len(scaled), :]
```

Print train and test size

```
print('train: {}\ntest: {}'.format(len(train), len(test)))
```

```
train: 2555
test: 1095
```

Create the sequential data

Map the temprature at a particular time t to the temprature at time t+n, where n is any number you de

For example: to map tempratures of consecutive days, use t+1, i.e. loop_back = 1

Define your function to create dataset

```
#window - how long the sequence will be
def create_dataset(dataset, window=1):

    dataX, dataY = [], []

    for i in range(len(dataset)-window):

        a = dataset[i:(i+window), 0]
        dataX.append(a)
        dataY.append(dataset[i + window, 0])
```

```
    return np.array(dataX), np.array(dataY)
```

Use function to get training and test set

```
#Create Input and Output
window_size = 1
X_train, y_train = create_dataset(train, window_size)
X_test, y_test = create_dataset(test, window_size)
```

Transform the prepared train and test input data into the expected structure using numpy.

```
y_train.shape

(X_train.shape[0], X_train.shape[1], 1)

X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
print(X_train.shape)
print(X_test.shape)
```

```
(2554, 1, 1)
(1094, 1, 1)
```

Define Model

Define sequntial model, add LSTM layer and compile the model

```
import tensorflow as tf


tf.keras.backend.clear_session()
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(32, input_shape=(window_size, 1)))
model.add(tf.keras.layers.Dense(1))
model.compile(optimizer='adam', loss='mse')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/op
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

```
model.fit(X_train, y_train, epochs=200, validation_data=(X_test, y_test), batch_size=32)
```

```
2554/2554 [==============================] - 0s 95us/sample - loss: 0.0100 - val_loss: 0
Epoch 173/200
2554/2554 [==============================] - 0s 96us/sample - loss: 0.0100 - val_loss: 0
Epoch 174/200
2554/2554 [==============================] - 0s 102us/sample - loss: 0.0100 - val_loss:
Epoch 175/200
2554/2554 [==============================] - 0s 107us/sample - loss: 0.0100 - val_loss:
Epoch 176/200
2554/2554 [==============================] - 0s 93us/sample - loss: 0.0100 - val_loss: 0
Epoch 177/200
2554/2554 [==============================] - 0s 95us/sample - loss: 0.0100 - val_loss: 0
Epoch 178/200
2554/2554 [==============================] - 0s 92us/sample - loss: 0.0100 - val_loss: 0
Epoch 179/200
2554/2554 [==============================] - 0s 108us/sample - loss: 0.0100 - val_loss:
Epoch 180/200
2554/2554 [==============================] - 0s 97us/sample - loss: 0.0100 - val_loss: 0
Epoch 181/200
2554/2554 [==============================] - 0s 97us/sample - loss: 0.0100 - val_loss: 0
Epoch 182/200
2554/2554 [==============================] - 0s 92us/sample - loss: 0.0100 - val_loss: 0
Epoch 183/200
2554/2554 [==============================] - 0s 94us/sample - loss: 0.0100 - val_loss: 0
Epoch 184/200
2554/2554 [==============================] - 0s 92us/sample - loss: 0.0100 - val_loss: 0
Epoch 185/200
2554/2554 [==============================] - 0s 105us/sample - loss: 0.0100 - val_loss:
Epoch 186/200
2554/2554 [==============================] - 0s 95us/sample - loss: 0.0100 - val_loss: 0
Epoch 187/200
2554/2554 [==============================] - 0s 101us/sample - loss: 0.0100 - val_loss:
Epoch 188/200
2554/2554 [==============================] - 0s 93us/sample - loss: 0.0100 - val_loss: 0
Epoch 189/200
2554/2554 [==============================] - 0s 94us/sample - loss: 0.0100 - val_loss: 0
Epoch 190/200
2554/2554 [==============================] - 0s 97us/sample - loss: 0.0100 - val_loss: 0
Epoch 191/200
2554/2554 [==============================] - 0s 94us/sample - loss: 0.0100 - val_loss: 0
Epoch 192/200
2554/2554 [==============================] - 0s 96us/sample - loss: 0.0100 - val_loss: 0
Epoch 193/200
2554/2554 [==============================] - 0s 98us/sample - loss: 0.0100 - val_loss: 0
Epoch 194/200
2554/2554 [==============================] - 0s 110us/sample - loss: 0.0100 - val_loss:
Epoch 195/200
2554/2554 [==============================] - 0s 93us/sample - loss: 0.0099 - val_loss: 0
Epoch 196/200
2554/2554 [==============================] - 0s 94us/sample - loss: 0.0100 - val_loss: 0
Epoch 197/200
2554/2554 [==============================] - 0s 98us/sample - loss: 0.0099 - val_loss: 0
Epoch 198/200
2554/2554 [==============================] - 0s 95us/sample - loss: 0.0100 - val_loss: 0
Epoch 199/200
2554/2554 [==============================] - 0s 105us/sample - loss: 0.0100 - val_loss:
Epoch 200/200
2554/2554 [==============================] - 0s 95us/sample - loss: 0.0099 - val_loss: 0
<tensorflow.python.keras.callbacks.History at 0x7fd142dc9dd8>
```

```
<tensorflow.python.keras.callbacks.History at 0x7fd143dc9da8>
```