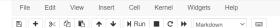




jupyter C1_W1_Lab_2_multi-output Last Checkpoint: 05/09/2021 (autosaved)



Not Trusted Python 3 O



Ungraded Lab: Build a Multi-output Model

In this lab, we'll show how you can build models with more than one output. The dataset we will be working on is available from the <u>UCI Machine Learning Repository</u>. It is an Energy Efficiency dataset which uses the bulding features (e.g. wall area, roof area) as inputs and has two outputs: Cooling Load and Heating Load. Let's see how we can build a model to train on this data.

Imports

Utilities

We define a few utilities for data conversion and visualization to make our code more neat.

```
In []: W

def format_output(data):
    y1 = data.pop('Y1')
    y2 = ny.array(y1)
    y2 = ny.array(y2)
    return y1, y2

def norm(x):
    return (x - train_stats['mean']) / train_stats['std']

def plot_diff(y_true, y_pred, title=''):
    plt.scatter(y_true, y_pred)
    plt.title(title)
    plt.xlabel('True values')
    plt.ylabel('Predictions')
    plt.axis('equal')
    plt.axis('equal')
    plt.xlim(plt.xlim())
    plt.ylim(plt.ylim())
    plt.ylim(plt.ylim())
    plt.plot([-100, 100], [-100, 100])

def plot_metrics(metric_name, title, ylim=5):
    plt.title(title)
    plt.ylim(opt.ylim())
    plt.ylim(opt.
```

Prepare the Data

We download the dataset and format it for training.

Build the Model

Here is how we'll build the model using the functional syntax. Notice that we can specify a list of outputs (i.e. $[y1_output, y2_output]$) when we instantiate the Model() class.

```
y2_output = Dense(units='1', name='y2_output')(third_dense)

# Define the model with the input layer and a list of output layers
model = Model(inputs=input_layer, outputs=[y1_output, y2_output])
print(model.summary())
```

Configure parameters

We specify the optimizer as well as the loss and metrics for each output.

Train the Model

Evaluate the Model and Plot Metrics