Creating Simple Dataset Notebook here:

[https://colab.research.google.com/drive/14e2-UmfV2FGMmWCsp405NKXqDbzunBKZ?usp=sharin](https://colab.research.google.com/drive/14e2-UmfV2FGMmWCsp405NKXqDbzunBKZ?usp=sharing)g

Time series Notebook here:

<https://colab.research.google.com/drive/1CKYsULqCqnE9TpWP7xhTJunYw-C4vuMX?usp=sharing>

## Getting the Dataset (Xs and Ys)

def windowed\_dataset(series, window\_size, batch\_size, shuffle\_buffer\_size):

#Get the Dataset

ds = Dataset.from\_tensor\_slices(series)

#Get windows of size (window\_size + 1)

ds = ds.window(window\_size+1, shift=1,drop\_remainder=True)

#Batch the windows using the same window size

ds = ds.flat\_map(lambda window: window.batch(window\_size + 1))

#Shuffle and split into x and y values

ds = ds.shuffle(shuffle\_buffer\_size).map(lambda data: (data[:-1], data[-1:]))

#Batch the windows to use for training batches

ds = ds.batch(batch\_size).prefetch(1)

return ds

* Series is the set of values for your time series graph
* You pass in Series and it gives you a dataset containing the Xs and Y values (You directly feed ds into the model, without the need to split into x and y
* X will be of shape (32,20) and Y of shape (1,20) for every batch (Given that batch size is 32)
* X will be the first 20 values and Y will be the 21st value on the time series

## 

## Simple Training

l0 = tf.keras.layers.Dense(1,input\_shape=[window\_size])

model = tf.keras.models.Sequential([l0])

model.compile(loss='mse', optimizer=tf.keras.optimizers.SGD(lr=1e-6, momentum=0.9))

model.fit(dataset, epochs=200,verbose=0)

## Getting the predicted values

#i will be from i=0 to i=1440

for i in range(len(series)- window\_size):

#Must expand\_dims to change shape from (20) to (1,20)

pred\_series = np.expand\_dims(series[i:i+20],axis=0)

pred = model.predict(pred\_series)

#Append all predictions to the pred list. This will be predicted values at t=20 to t=1460

preds.append(pred)

* len(series) is the total no. of values in the time series (1461 in this case)
* Window\_size is the length of 1 X data (which comprises of 20 values in the time series)
* Pass the model the previous 20 timesteps to predict the current timestep (use t=0 to t=19 to predict the value at t=20)
* Expand\_dims to convert to a format that the model can predict on
* Use t=0 to t=1440 as the starting values of the prediction

(so last one will be t=1440 to t= 1459 to be used to predict t=1460)

* So, you predict values from t=20 to t=1460 (So, you predicted 1441 values in total)

(NOT 1440 values predicted since that doesn't account for the predicted value at t=20!)



## Getting the validation set predictions

#Only get the predicted values of the validation set

#preds[0] is the predicted value of t=20, using the first 20 values (t=0 to t=19)

#Index the preds[800:]this will be the predicted values of t=1000 all the way to t=1460

preds\_valid = preds[split\_time-window\_size:]

* Get the predicted values from t=1000 to t=1460 (Which is 461 values)

## Getting the 1D array of predicted values

#preds\_valid is a 3d array. You index all the elements in the first dimension, and index the first element of the second and third dimension

'''

[

[[t=0]]

[[t=1]]

[[t=2]]

]

'''

results = np.array(preds\_valid)[:, 0, 0]

* Get the 1D array containing all the predicted values. Do this by getting all the elements from the first dimension and indexing the first element for the 2nd and 3rd dimensions