```
# Import libraries. You may or may not use all of these.
!pip install -q git+https://github.com/tensorflow/docs
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers

import tensorflow_docs as tfdocs
import tensorflow_docs.plots
import tensorflow_docs.modeling
```

```
  Preparing metadata (setup.py) ... done
Colab only includes TensorFlow 2.x; %tensorflow_version has no effect.
```

```
# Import data
!wget https://cdn.freecodecamp.org/project-data/health-costs/insurance.csv
dataset = pd.read_csv('insurance.csv')
dataset.tail()
```

```
--2024-10-31 14:35:22--  https://cdn.freecodecamp.org/project-data/health-costs/insurance.csv
Resolving cdn.freecodecamp.org (cdn.freecodecamp.org)... 172.67.70.149, 104.26.3.33, 104.26.2.33, ...
Connecting to cdn.freecodecamp.org (cdn.freecodecamp.org)|172.67.70.149|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 50264 (49K) [text/csv]
Saving to: ‘insurance.csv.4’

insurance.csv.4     100%[===================>]  49.09K  --.-KB/s    in 0.01s

2024-10-31 14:35:22 (3.61 MB/s) - ‘insurance.csv.4’ saved [50264/50264]
```

	age	sex	bmi	children	smoker	region	expenses
1333	50	male	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83
1336	21	female	25.8	0	no	southwest	2007.95
1337	61	female	29.1	0	yes	northwest	29141.36

```
# Converting catagorical data into numbers
#can even use dataset['Origin'] = dataset['Origin'].map({1: 'USA', 2: 'Europe', 3: 'Japan'})
#dataset["sex"] = dataset["sex"].astype('category').cat.codes
#dataset["smoker"] = dataset["smoker"].astype('category').cat.codes
#dataset['region'] = dataset['region'].astype('category').cat.codes

catColumns = ["sex", "smoker", "region"]
dataset = pd.get_dummies(dataset, columns = catColumns, drop_first=True)
dataset
```

	age	bmi	children	expenses	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest
0	19	27.9	0	16884.92	False	True	False	False	True
1	18	33.8	1	1725.55	True	False	False	True	False
2	28	33.0	3	4449.46	True	False	False	True	False
3	33	22.7	0	21984.47	True	False	True	False	False
4	32	28.9	0	3866.86	True	False	True	False	False
...	...	...	...	...	...	...	...	...	...
1333	50	31.0	3	10600.55	True	False	True	False	False
1334	18	31.9	0	2205.98	False	False	False	False	False
1335	18	36.9	0	1629.83	False	False	False	True	False
1336	21	25.8	0	2007.95	False	False	False	False	True
1337	61	29.1	0	29141.36	False	True	True	False	False

1338 rows × 9 columns

Next steps: Generate code with dataset | View recommended plots | New interactive sheet

```
#Split train-test data as 80-20
train_dataset = dataset.sample(frac=0.8,random_state=0)
test_dataset = dataset.drop(train_dataset.index)


#Pop the y label for train and test sets
train_labels = train_dataset.pop('expenses')
test_labels = test_dataset.pop('expenses')


def build_model():
  model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=[len(train_dataset.keys())]),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
  ])

  optimizer = tf.keras.optimizers.RMSprop(0.01)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model


model = build_model()
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_15"
```

Layer (type)	Output Shape	Param #
dense_51 (Dense)	(None, 64)	576
dense_52 (Dense)	(None, 64)	4,160
dense_53 (Dense)	(None, 1)	65

```
Total params: 4,801 (18.75 KB)
Trainable params: 4,801 (18.75 KB)
Non-trainable params: 0 (0.00 B)
```

```
example_batch = train_dataset[:10]
example_result = model.predict(example_batch)
example_result
```

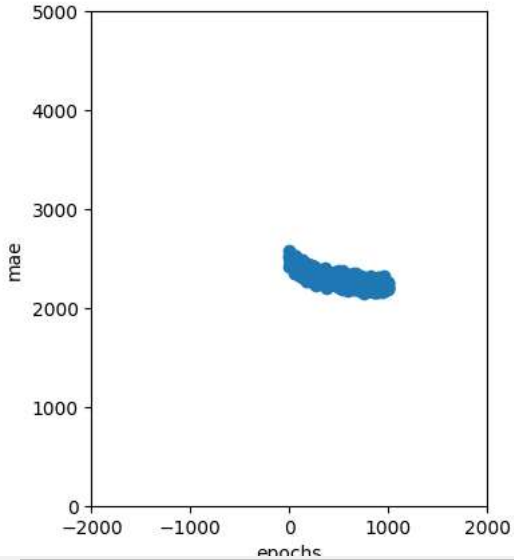
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step
array([[-0.4536562 ],
       [-0.45163822],
```

```
            [-1.3636885 ],
            [-0.4970541 ],
            [ 0.79911995],
            [-0.4287305 ],
            [-0.34012985],
            [-0.824224  ],
            [-0.9008274 ],
            [-1.1088824 ]], dtype=float32)


EPOCHS = 1000

history = model.fit(
  train_dataset, train_labels,
  epochs=EPOCHS, validation_split = 0.2, verbose=0,
  callbacks=[tfdocs.modeling.EpochDots()])
```

```
Epoch: 0, loss:19148498.0000,  mae:2584.6040,  mse:19148498.0000,  val_loss:19345152.0000,  val_mae:2561.7991,  val_mse:19345152.0000,
....................................................................................
Epoch: 100, loss:17432372.0000,  mae:2340.9719,  mse:17432372.0000,  val_loss:24390928.0000,  val_mae:3809.0217,  val_mse:24390928.0000,
....................................................................................
Epoch: 200, loss:17186174.0000,  mae:2304.0054,  mse:17186174.0000,  val_loss:20374920.0000,  val_mae:2682.5957,  val_mse:20374920.0000,
....................................................................................
Epoch: 300, loss:17217248.0000,  mae:2300.4822,  mse:17217248.0000,  val_loss:19903026.0000,  val_mae:2613.5701,  val_mse:19903026.0000,
....................................................................................
Epoch: 400, loss:16647792.0000,  mae:2313.1887,  mse:16647792.0000,  val_loss:19935612.0000,  val_mae:2428.1799,  val_mse:19935612.0000,
....................................................................................
Epoch: 500, loss:16814572.0000,  mae:2360.7905,  mse:16814572.0000,  val_loss:20978872.0000,  val_mae:2167.9541,  val_mse:20978872.0000,
....................................................................................
Epoch: 600, loss:16322841.0000,  mae:2292.9912,  mse:16322841.0000,  val_loss:23539696.0000,  val_mae:2338.2107,  val_mse:23539696.0000,
....................................................................................
Epoch: 700, loss:15618956.0000,  mae:2216.1912,  mse:15618956.0000,  val_loss:21679786.0000,  val_mae:2472.9253,  val_mse:21679786.0000,
....................................................................................
Epoch: 800, loss:15659093.0000,  mae:2272.3149,  mse:15659093.0000,  val_loss:22537606.0000,  val_mae:2891.4402,  val_mse:22537606.0000,
....................................................................................
Epoch: 900, loss:14861685.0000,  mae:2211.2170,  mse:14861685.0000,  val_loss:22693328.0000,  val_mae:3010.0679,  val_mse:22693328.0000,
....................................................................................
```

```
mae = history.history["mae"]
loss = history.history["loss"]
epoch = history.epoch

a = plt.axes(aspect='equal')
plt.scatter(epoch, mae)
plt.xlabel('epochs')
plt.ylabel('mae')
plt.xlim([-2000, 2000])
plt.ylim([0, 5000])
```

```
(0.0, 5000.0)
```



```
# RUN THIS CELL TO TEST YOUR MODEL. DO NOT MODIFY CONTENTS.
# Test model by checking how well the model generalizes using the test set.
loss, mae, mse = model.evaluate(test_dataset, test_labels, verbose=2)

print("Testing set Mean Abs Error: {:5.2f} expenses".format(mae))

if mae < 3500:
  print("You passed the challenge. Great job!")
else:
  print("The Mean Abs Error must be less than 3500. Keep trying.")

# Plot predictions.
test_predictions = model.predict(test_dataset).flatten()

a = plt.axes(aspect='equal')
plt.scatter(test_labels, test_predictions)
plt.xlabel('True values (expenses)')
plt.ylabel('Predictions (expenses)')
lims = [0, 50000]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims,lims)
```

```
9/9 - 0s - 4ms/step - loss: 35466520.0000 - mae: 3222.1812 - mse: 35466520.0000
Testing set Mean Abs Error: 3222.18 expenses
You passed the challenge. Great job!
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```