```
# 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
  Building wheel for tensorflow-docs (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()
```

```
--2024-07-08 23:51:10--  https://cdn.freecodecamp.org/project-data/health-costs/
Resolving cdn.freecodecamp.org (cdn.freecodecamp.org)... 172.67.70.149, 104.26.3
Connecting to cdn.freecodecamp.org (cdn.freecodecamp.org)|172.67.70.149|:443...
HTTP request sent, awaiting response... 200 OK
Length: 50264 (49K) [text/csv]
Saving to: 'insurance.csv'

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

2024-07-08 23:51:10 (4.21 MB/s) - 'insurance.csv' 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 Categorical Data into Numbers
from sklearn.utils import shuffle
dataset["sex"].replace(
    ["female", "male"],
    [0, 1],
    inplace=True
)

dataset["smoker"].replace(
    ["no", "yes"],
    [0, 1],
    inplace=True
)

dataset["region"].replace(
    ['southwest', 'southeast', 'northwest', 'northeast'],
    [0, 1, 2, 3],
    inplace=True
)
dataset = shuffle(dataset).reset_index(drop=True)
```

```
# Seperating into training and test data
train_dataset  = dataset[0:int(0.8*dataset.shape[0])]
test_dataset = dataset[int(0.8*dataset.shape[0]):dataset.shape[0] - 1]

train_labels = train_dataset.pop("expenses")
test_labels = test_dataset.pop("expenses")

```

```
# Creating a model
normalizer = layers.experimental.preprocessing.Normalization()
normalizer.adapt(np.array(train_dataset))

model = keras.Sequential([
    normalizer,
    layers.Dense(32, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1)
])

model.compile(
    optimizer=tf.optimizers.Adam(learning_rate=0.1),
    loss='mae',
    metrics=['mae', 'mse']
)
model.build()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization (Normalizati  (None, 6)                 13
 on)

 dense (Dense)               (None, 32)                224

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 1)                 17

=================================================================
Total params: 782 (3.06 KB)
Trainable params: 769 (3.00 KB)
Non-trainable params: 13 (56.00 Byte)
_________________________________________________________________
```

```
# Training a model
history = model.fit(
    train_dataset,
    train_labels,
    epochs=100
)
```

```
Epoch 72/100
34/34 [==============================] - 0s 2ms/step - loss: 1922.2421 - mae: 1922.2421 - mse: 22669486.0000
Epoch 73/100
34/34 [==============================] - 0s 2ms/step - loss: 1897.0394 - mae: 1897.0394 - mse: 22772438.0000
Epoch 74/100
34/34 [==============================] - 0s 3ms/step - loss: 1958.0732 - mae: 1958.0732 - mse: 22763400.0000
Epoch 75/100
34/34 [==============================] - 0s 2ms/step - loss: 1952.7014 - mae: 1952.7014 - mse: 22890322.0000
Epoch 76/100
34/34 [==============================] - 0s 2ms/step - loss: 1901.6957 - mae: 1901.6957 - mse: 22527670.0000
Epoch 77/100
34/34 [==============================] - 0s 2ms/step - loss: 1893.3260 - mae: 1893.3260 - mse: 22433424.0000
Epoch 78/100
34/34 [==============================] - 0s 2ms/step - loss: 1915.0585 - mae: 1915.0585 - mse: 22063888.0000
Epoch 79/100
34/34 [==============================] - 0s 2ms/step - loss: 1915.0327 - mae: 1915.0327 - mse: 22320650.0000
Epoch 80/100
34/34 [==============================] - 0s 2ms/step - loss: 1914.0562 - mae: 1914.0562 - mse: 22208646.0000
Epoch 81/100
34/34 [==============================] - 0s 3ms/step - loss: 1869.4950 - mae: 1869.4950 - mse: 22302742.0000
Epoch 82/100
34/34 [==============================] - 0s 2ms/step - loss: 1856.0974 - mae: 1856.0974 - mse: 21946944.0000
Epoch 83/100
34/34 [==============================] - 0s 2ms/step - loss: 1867.3002 - mae: 1867.3002 - mse: 22066264.0000
Epoch 84/100
34/34 [==============================] - 0s 2ms/step - loss: 1845.6095 - mae: 1845.6095 - mse: 21942600.0000
Epoch 85/100
34/34 [==============================] - 0s 2ms/step - loss: 1819.3542 - mae: 1819.3542 - mse: 22194226.0000
Epoch 86/100
34/34 [==============================] - 0s 3ms/step - loss: 1871.0062 - mae: 1871.0062 - mse: 22297846.0000
Epoch 87/100
34/34 [==============================] - 0s 2ms/step - loss: 1831.1361 - mae: 1831.1361 - mse: 22024522.0000
Epoch 88/100
34/34 [==============================] - 0s 2ms/step - loss: 1824.2125 - mae: 1824.2125 - mse: 22069246.0000
Epoch 89/100
34/34 [==============================] - 0s 2ms/step - loss: 1802.5834 - mae: 1802.5834 - mse: 21877862.0000
Epoch 90/100
34/34 [==============================] - 0s 2ms/step - loss: 1901.1235 - mae: 1901.1235 - mse: 22222952.0000
Epoch 91/100
34/34 [==============================] - 0s 2ms/step - loss: 1931.0582 - mae: 1931.0582 - mse: 21864618.0000
Epoch 92/100
34/34 [==============================] - 0s 2ms/step - loss: 1877.7990 - mae: 1877.7990 - mse: 21988342.0000
Epoch 93/100
34/34 [==============================] - 0s 3ms/step - loss: 1965.6224 - mae: 1965.6224 - mse: 22445764.0000
Epoch 94/100
34/34 [==============================] - 0s 2ms/step - loss: 1813.6625 - mae: 1813.6625 - mse: 21672738.0000
Epoch 95/100
34/34 [==============================] - 0s 2ms/step - loss: 1839.8890 - mae: 1839.8890 - mse: 21950926.0000
Epoch 96/100
34/34 [==============================] - 0s 2ms/step - loss: 1830.4393 - mae: 1830.4393 - mse: 22055432.0000
Epoch 97/100
34/34 [==============================] - 0s 3ms/step - loss: 1812.6365 - mae: 1812.6365 - mse: 21842410.0000
Epoch 98/100
34/34 [==============================] - 0s 3ms/step - loss: 1822.8140 - mae: 1822.8140 - mse: 21997764.0000
Epoch 99/100
34/34 [==============================] - 0s 2ms/step - loss: 1780.9214 - mae: 1780.9214 - mse: 21870160.0000
Epoch 100/100
34/34 [==============================] - 0s 2ms/step - loss: 1758.3546 - mae: 1758.3546 - mse: 21668244.0000
```

```
# 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 - loss: 1942.5685 - mae: 1942.5685 - mse: 25610614.0000 - 207ms/epoch - 23ms/step
Testing set Mean Abs Error: 1942.57 expenses
You passed the challenge. Great job!
9/9 [==============================] - 0s 2ms/step
```