Project: Linear Regression Health Costs Calculator

Objective: predict healthcare costs using a regression algorithm.

We are given a dataset that contains information about different people including their healthcare costs. Use the data to predict healthcare costs based on new data.

In [ ]:
```
# 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
```

```
DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the near future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebrew-core/issues/76621
DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the near future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebrew-core/issues/76621
2023-07-03 21:21:08.417426: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

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

```
--2023-07-03 21:21:12--  https://cdn.freecodecamp.org/project-data/health-costs/insurance.csv
Resolving cdn.freecodecamp.org (cdn.freecodecamp.org)... 104.26.3.33, 104.26.2.33, 172.67.70.149
Connecting to cdn.freecodecamp.org (cdn.freecodecamp.org)|104.26.3.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 50264 (49K) [text/csv]
Saving to: 'insurance.csv'

insurance.csv       100%[===================>]  49,09K  --.-KB/s    in 0,04s

2023-07-03 21:21:13 (1,30 MB/s) - 'insurance.csv' saved [50264/50264]
```

Out[ ]:

	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

In [ ]:
```
dataset.shape
```

Out[ ]:
```
(1338, 7)
```

In [ ]:
```
dataset.head()
```

Out[ ]:

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86

In [ ]:
```
dataset.info
```

Out[ ]:
```
<bound method DataFrame.info of       age     sex   bmi  children smoker     region  expenses
0      19  female  27.9         0    yes  southwest  16884.92
1      18    male  33.8         1     no  southeast   1725.55
2      28    male  33.0         3     no  southeast   4449.46
3      33    male  22.7         0     no  northwest  21984.47
4      32    male  28.9         0     no  northwest   3866.86
...   ...     ...   ...       ...    ...        ...       ...
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

[1338 rows x 7 columns]>
```

Data has some text columns, we need to convert the text values to numeric.

In [ ]:
```
df = dataset
df["sex"] = pd.factorize(df["sex"])[0]
df["region"] = pd.factorize(df["region"])[0]
df["smoker"] = pd.factorize(df["smoker"])[0]
dataset = df
dataset.head()
```

Out[ ]:

	age	sex	bmi	children	smoker	region	expenses
0	19	0	27.9	0	0	0	16884.92
1	18	1	33.8	1	1	1	1725.55
2	28	1	33.0	3	1	1	4449.46
3	33	1	22.7	0	1	2	21984.47
4	32	1	28.9	0	1	2	3866.86

Randomly 20% record to make our test_dataset first

In [ ]:
```
test_dataset = dataset.sample(frac=0.2)
len(test_dataset)
```

Out[ ]:
```
268
```

Select the remaining 80% to make train_dataset

In [ ]:
```
train_dataset = dataset[~dataset.isin(test_dataset)].dropna()
len(train_dataset)
```

Out[ ]:
```
1070
```

In [ ]:
```
train_dataset.head()
```

Out[ ]:

	age	sex	bmi	children	smoker	region	expenses
0	19.0	0.0	27.9	0.0	0.0	0.0	16884.92
1	18.0	1.0	33.8	1.0	1.0	1.0	1725.55
2	28.0	1.0	33.0	3.0	1.0	1.0	4449.46
3	33.0	1.0	22.7	0.0	1.0	2.0	21984.47
4	32.0	1.0	28.9	0.0	1.0	2.0	3866.86

Prepare the labels

In [ ]:
```
train_labels = train_dataset.pop("expenses")
train_labels.head()
```

Out[ ]:
```
0    16884.92
1     1725.55
2     4449.46
3    21984.47
4     3866.86
Name: expenses, dtype: float64
```

In [ ]:
```
train_dataset.head()
```

Out[ ]:

	age	sex	bmi	children	smoker	region
0	19.0	0.0	27.9	0.0	0.0	0.0
1	18.0	1.0	33.8	1.0	1.0	1.0
2	28.0	1.0	33.0	3.0	1.0	1.0
3	33.0	1.0	22.7	0.0	1.0	2.0
4	32.0	1.0	28.9	0.0	1.0	2.0

In [ ]:
```
test_labels = test_dataset.pop("expenses")
test_labels.head()
```

Out[ ]:
```
688     26236.58
251     47305.31
813      4428.89
1027    21595.38
883     46255.11
Name: expenses, dtype: float64
```

In [ ]:
```
test_dataset.head()
```

Out[ ]:

	age	sex	bmi	children	smoker	region
688	47	0	24.1	1	1	0
251	63	0	32.2	2	0	0
813	28	1	22.5	2	1	3
1027	23	1	18.7	0	1	2
883	51	0	37.1	3	0	3

Prepare the model

In [ ]:
```
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(8, activation='relu'),
    layers.Dense(1),
])
```

In [ ]:
```
model.compile(
    optimizer=tf.optimizers.Adam(learning_rate=0.01),
    loss='mae',
    metrics=['mae', 'mse']
)

model.build()
model.summary()
```

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

dense (Dense)                (None, 32)                224

dense_1 (Dense)              (None, 16)                528

dense_2 (Dense)              (None, 8)                 136

dense_3 (Dense)              (None, 1)                 9

=================================================================
Total params: 910
Trainable params: 897
Non-trainable params: 13
_________________________________________________________________
```

In [ ]:
```
history = model.fit(
    train_dataset,
    train_labels,
    epochs=200,  # Increase the number of epochs
    validation_split=0.5,
    verbose=0
)

print(history)
```

```
<keras.callbacks.History object at 0x12cef7eb0>
```

Test

In [ ]:
```
# 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))

# 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: 2054.1250 - mae: 2054.1250 - mse: 27074448.0000 - 34ms/epoch - 4ms/step
Testing set Mean Abs Error: 2054.12 expenses
9/9 [==============================] - 0s 1ms/step
```