```
1 # import required libraries
2 import tensorflow as tf
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
```

```
1 # Read in the insurance data set
2 insurance = pd.read_csv("https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master
3 insurance
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

```
1 insurance["smoker"],insurance["age"]
```

```
(0       yes
 1        no
 2        no
 3        no
 4        no
        ...
 1333     no
 1334     no
 1335     no
 1336     no
 1337    yes
 Name: smoker, Length: 1338, dtype: object,
 0       19
 1       18
 2       28
 3       33
 4       32
         ..
 1333    50
 1334    18
 1335    18
 1336    21
 1337    61
 Name: age, Length: 1338, dtype: int64)
```

```
1 # Let's try one-hot encode our data frame  so it's all number
2 insurance_one_hot=pd.get_dummies(insurance)
3 insurance_one_hot.head()
```

	age	bmi	children	charges	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	re
0	19	27.900	0	16884.92400	1	0	0	1	0	
1	18	33.770	1	1725.55230	0	1	1	0	0	
2	28	33.000	3	4449.46200	0	1	1	0	0	
3	33	22.705	0	21984.47061	0	1	1	0	0	
4	32	28.880	0	3866.85520	0	1	1	0	0	

```
# Create X & y values(features and labels)
X= insurance_one_hot.drop("charges",axis=1)
y= insurance_one_hot["charges"]
```

```
# View X
X.head()
```

	age	bmi	children	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwes
0	19	27.900	0	1	0	0	1	0	
1	18	33.770	1	0	1	1	0	0	
2	28	33.000	3	0	1	1	0	0	
3	33	22.705	0	0	1	1	0	0	
4	32	28.880	0	0	1	1	0	0	

```
# view y
y.head()
```

```
0    16884.92400
1     1725.55230
2     4449.46200
3    21984.47061
4     3866.85520
Name: charges, dtype: float64
```

```
# Create a training anad test sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
len(X),len(X_train),len(X_test)
```

```
(1338, 1070, 268)
```

```
X_train
```

	age	bmi	children	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_north
560	46	19.950	2	1	0	1	0	0	
1285	47	24.320	0	1	0	1	0	1	
1142	52	24.860	0	1	0	1	0	0	
969	39	34.320	5	1	0	1	0	0	
486	54	21.470	3	1	0	1	0	0	
...	...	...	...	...	...	...	...	...	
1095	18	31.350	4	1	0	1	0	1	
1130	39	23.870	5	1	0	1	0	0	
1294	58	25.175	0	0	1	1	0	1	
860	37	47.600	2	1	0	0	1	0	
1126	55	29.900	0	0	1	1	0	0	

1070 rows × 11 columns

```
# Build a neural network
tf.random.set_seed(42)

# 1. Create a model
insurance_model= tf.keras.Sequential([
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1)
])

# 2. Compile the model
insurance_model.compile(loss=tf.keras.losses.mae,
                        optimizer=tf.keras.optimizers.SGD(),
                        metrics=["mae"])

```

```
# 3. Fit the model
insurance_model.fit(X_train,y_train,epochs=100)

```

```
Epoch 55/100
34/34 [==============================] - 0s 2ms/step - loss: 7238.4326 - mae: 7238.4326
Epoch 56/100
34/34 [==============================] - 0s 2ms/step - loss: 7390.6289 - mae: 7390.6289
Epoch 57/100
34/34 [==============================] - 0s 2ms/step - loss: 7428.5566 - mae: 7428.5566
Epoch 58/100
34/34 [==============================] - 0s 2ms/step - loss: 7414.1123 - mae: 7414.1123
Epoch 59/100
34/34 [==============================] - 0s 2ms/step - loss: 7657.4102 - mae: 7657.4102
Epoch 60/100
34/34 [==============================] - 0s 2ms/step - loss: 7389.7661 - mae: 7389.7661
Epoch 61/100
34/34 [==============================] - 0s 2ms/step - loss: 7586.9004 - mae: 7586.9004
Epoch 62/100
34/34 [==============================] - 0s 2ms/step - loss: 7412.1011 - mae: 7412.1011
Epoch 63/100
34/34 [==============================] - 0s 2ms/step - loss: 7130.7554 - mae: 7130.7554
Epoch 64/100
34/34 [==============================] - 0s 2ms/step - loss: 7307.0986 - mae: 7307.0986
Epoch 65/100
34/34 [==============================] - 0s 2ms/step - loss: 7361.6797 - mae: 7361.6797
Epoch 66/100
34/34 [==============================] - 0s 2ms/step - loss: 7446.7954 - mae: 7446.7954
Epoch 67/100
34/34 [==============================] - 0s 2ms/step - loss: 7175.2803 - mae: 7175.2803
Epoch 68/100
34/34 [==============================] - 0s 2ms/step - loss: 7464.6191 - mae: 7464.6191
Epoch 69/100
34/34 [==============================] - 0s 2ms/step - loss: 7460.7158 - mae: 7460.7158
Epoch 70/100
34/34 [==============================] - 0s 2ms/step - loss: 7489.4443 - mae: 7489.4443
Epoch 71/100
34/34 [==============================] - 0s 2ms/step - loss: 7449.6904 - mae: 7449.6904
Epoch 72/100
34/34 [==============================] - 0s 2ms/step - loss: 7248.8364 - mae: 7248.8364
Epoch 73/100
34/34 [==============================] - 0s 2ms/step - loss: 7352.2988 - mae: 7352.2988
Epoch 74/100
34/34 [==============================] - 0s 2ms/step - loss: 7488.3115 - mae: 7488.3115
Epoch 75/100
34/34 [==============================] - 0s 2ms/step - loss: 7126.4116 - mae: 7126.4116
Epoch 76/100
34/34 [==============================] - 0s 2ms/step - loss: 7190.6333 - mae: 7190.6333
Epoch 77/100
34/34 [==============================] - 0s 2ms/step - loss: 7439.6084 - mae: 7439.6084
Epoch 78/100
34/34 [==============================] - 0s 2ms/step - loss: 6883.9873 - mae: 6883.9873
Epoch 79/100
34/34 [==============================] - 0s 2ms/step - loss: 7535.5942 - mae: 7535.5942
Epoch 80/100
34/34 [==============================] - 0s 2ms/step - loss: 7422.2339 - mae: 7422.2339
Epoch 81/100
34/34 [==============================] - 0s 2ms/step - loss: 7149.4351 - mae: 7149.4351
Epoch 82/100
34/34 [==============================] - 0s 2ms/step - loss: 7250.4336 - mae: 7250.4336
Epoch 83/100
34/34 [==============================] - 0s 2ms/step - loss: 7507.2935 - mae: 7507.2935
Epoch 84/100
```

```
# check the results of insurance model on test data
insurance_model.evaluate(X_test,y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 8372.2754 - mae: 8372.2754
[8372.275390625, 8372.275390625]
```

```
y_train.median(),y_train.mean()
```

```
(9575.4421, 13346.089736364485)
```

Right now it looks liike our model is not performing well..let's try and improve it!

To (try) improve our model,we'll run 2 experiments:

1. Add an extra layer with more hidden units and use the Adam optimizer.
2. Train for longer
3. (insert your own experiment here)

```
# Set random seed
tf.random.set_seed(42)

# 1. create the model
insurance_model_2= tf.keras.Sequential([
    tf.keras.layers.Dense(100),
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1),
])

# 2. Compile the model
insurance_model_2.compile(loss=tf.keras.losses.mae,
                          optimizer=tf.keras.optimizers.Adam(),
                          metrics=["mae"])

# 3. Fit the model
insurance_model_2.fit(X_train,y_train,epochs=100,verbose=1)
```

```
34/34 [==============================] - 0s 4ms/step - loss: 6083.9292 - mae: 6083.9292
Epoch 73/100
34/34 [==============================] - 0s 3ms/step - loss: 6068.9634 - mae: 6068.9634
Epoch 74/100
34/34 [==============================] - 0s 3ms/step - loss: 6042.2856 - mae: 6042.2856
Epoch 75/100
34/34 [==============================] - 0s 4ms/step - loss: 6021.2661 - mae: 6021.2661
Epoch 76/100
34/34 [==============================] - 0s 4ms/step - loss: 6007.1904 - mae: 6007.1904
Epoch 77/100
34/34 [==============================] - 0s 6ms/step - loss: 5974.7412 - mae: 5974.7412
Epoch 78/100
34/34 [==============================] - 0s 4ms/step - loss: 5952.7207 - mae: 5952.7207
Epoch 79/100
34/34 [==============================] - 0s 7ms/step - loss: 5928.2656 - mae: 5928.2656
Epoch 80/100
34/34 [==============================] - 0s 5ms/step - loss: 5901.9487 - mae: 5901.9487
Epoch 81/100
34/34 [==============================] - 0s 4ms/step - loss: 5876.8101 - mae: 5876.8101
Epoch 82/100
34/34 [==============================] - 0s 4ms/step - loss: 5849.2944 - mae: 5849.2944
Epoch 83/100
34/34 [==============================] - 0s 6ms/step - loss: 5821.8589 - mae: 5821.8589
Epoch 84/100
34/34 [==============================] - 0s 8ms/step - loss: 5788.8101 - mae: 5788.8101
Epoch 85/100
34/34 [==============================] - 0s 4ms/step - loss: 5762.5049 - mae: 5762.5049
Epoch 86/100
34/34 [==============================] - 0s 5ms/step - loss: 5728.9771 - mae: 5728.9771
Epoch 87/100
34/34 [==============================] - 0s 6ms/step - loss: 5693.7437 - mae: 5693.7437
Epoch 88/100
34/34 [==============================] - 0s 4ms/step - loss: 5660.2822 - mae: 5660.2822
Epoch 89/100
34/34 [==============================] - 0s 4ms/step - loss: 5622.0054 - mae: 5622.0054
Epoch 90/100
34/34 [==============================] - 0s 6ms/step - loss: 5581.8906 - mae: 5581.8906
Epoch 91/100
34/34 [==============================] - 0s 3ms/step - loss: 5548.8403 - mae: 5548.8403
Epoch 92/100
34/34 [==============================] - 0s 5ms/step - loss: 5498.9194 - mae: 5498.9194
Epoch 93/100
34/34 [==============================] - 0s 5ms/step - loss: 5459.4111 - mae: 5459.4111
Epoch 94/100
34/34 [==============================] - 0s 4ms/step - loss: 5416.3892 - mae: 5416.3892
Epoch 95/100
34/34 [==============================] - 0s 5ms/step - loss: 5363.0635 - mae: 5363.0635
Epoch 96/100
34/34 [==============================] - 0s 4ms/step - loss: 5320.0010 - mae: 5320.0010
Epoch 97/100
34/34 [==============================] - 0s 3ms/step - loss: 5268.0181 - mae: 5268.0181
Epoch 98/100
34/34 [==============================] - 0s 4ms/step - loss: 5208.8267 - mae: 5208.8267
Epoch 99/100
34/34 [==============================] - 0s 5ms/step - loss: 5150.3325 - mae: 5150.3325
Epoch 100/100
34/34 [==============================] - 0s 5ms/step - loss: 5095.5630 - mae: 5095.5630
<keras.callbacks.History at 0x7fde5f709d20>
```

```
# Evaluate the model
insurance_model_2.evaluate(X_test,y_test)
```

```
9/9 [==============================] - 0s 4ms/step - loss: 4963.6123 - mae: 4963.6123
[4963.6123046875, 4963.6123046875]
```

```
# Set random seed
tf.random.set_seed(42)

# 1. create the model
insurance_model_3= tf.keras.Sequential([
    tf.keras.layers.Dense(100),
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1),
])

# 2. Compile the model
insurance_model_3.compile(loss=tf.keras.losses.mae,
                          optimizer=tf.keras.optimizers.Adam(),
                          metrics=["mae"])

# 3. Fit the model
history=insurance_model_3.fit(X_train,y_train,epochs=200)
```

```
34/34 [==============================] - 0s 6ms/step - loss: 3740.8586 - mae: 3740.8586
Epoch 151/200
34/34 [==============================] - 0s 6ms/step - loss: 3742.8582 - mae: 3742.8582
Epoch 152/200
34/34 [==============================] - 0s 7ms/step - loss: 3744.4946 - mae: 3744.4946
Epoch 153/200
34/34 [==============================] - 0s 6ms/step - loss: 3745.1584 - mae: 3745.1584
Epoch 154/200
34/34 [==============================] - 0s 6ms/step - loss: 3739.2522 - mae: 3739.2522
Epoch 155/200
34/34 [==============================] - 0s 6ms/step - loss: 3742.7021 - mae: 3742.7021
Epoch 156/200
34/34 [==============================] - 0s 9ms/step - loss: 3737.8555 - mae: 3737.8555
Epoch 157/200
34/34 [==============================] - 0s 12ms/step - loss: 3735.1296 - mae: 3735.1296
Epoch 158/200
34/34 [==============================] - 0s 9ms/step - loss: 3732.5378 - mae: 3732.5378
Epoch 159/200
34/34 [==============================] - 0s 10ms/step - loss: 3729.0229 - mae: 3729.0229
Epoch 160/200
34/34 [==============================] - 0s 7ms/step - loss: 3736.1531 - mae: 3736.1531
Epoch 161/200
34/34 [==============================] - 0s 6ms/step - loss: 3730.7847 - mae: 3730.7847
Epoch 162/200
34/34 [==============================] - 0s 5ms/step - loss: 3727.5432 - mae: 3727.5432
Epoch 163/200
34/34 [==============================] - 0s 9ms/step - loss: 3725.0820 - mae: 3725.0820
Epoch 164/200
34/34 [==============================] - 0s 11ms/step - loss: 3729.1099 - mae: 3729.1099
Epoch 165/200
34/34 [==============================] - 0s 10ms/step - loss: 3720.3237 - mae: 3720.3237
Epoch 166/200
34/34 [==============================] - 0s 11ms/step - loss: 3722.5479 - mae: 3722.5479
Epoch 167/200
34/34 [==============================] - 0s 14ms/step - loss: 3724.1365 - mae: 3724.1365
Epoch 168/200
34/34 [==============================] - 0s 9ms/step - loss: 3718.1836 - mae: 3718.1836
Epoch 169/200
34/34 [==============================] - 0s 9ms/step - loss: 3717.4670 - mae: 3717.4670
Epoch 170/200
34/34 [==============================] - 0s 9ms/step - loss: 3711.0815 - mae: 3711.0815
Epoch 171/200
34/34 [==============================] - 0s 9ms/step - loss: 3710.9958 - mae: 3710.9958
Epoch 172/200
34/34 [==============================] - 0s 9ms/step - loss: 3711.0173 - mae: 3711.0173
Epoch 173/200
34/34 [==============================] - 0s 9ms/step - loss: 3708.3420 - mae: 3708.3420
Epoch 174/200
34/34 [==============================] - 0s 8ms/step - loss: 3711.6448 - mae: 3711.6448
Epoch 175/200
34/34 [==============================] - 0s 7ms/step - loss: 3713.5789 - mae: 3713.5789
Epoch 176/200
34/34 [==============================] - 0s 9ms/step - loss: 3708.7476 - mae: 3708.7476
Epoch 177/200
34/34 [==============================] - 0s 6ms/step - loss: 3704.4399 - mae: 3704.4399
Epoch 178/200
34/34 [==============================] - 0s 6ms/step - loss: 3701.1604 - mae: 3701.1604
Epoch 179/200
34/34 [==============================] - 0s 5ms/step - loss: 3707.8538 - mae: 3707.8538
```

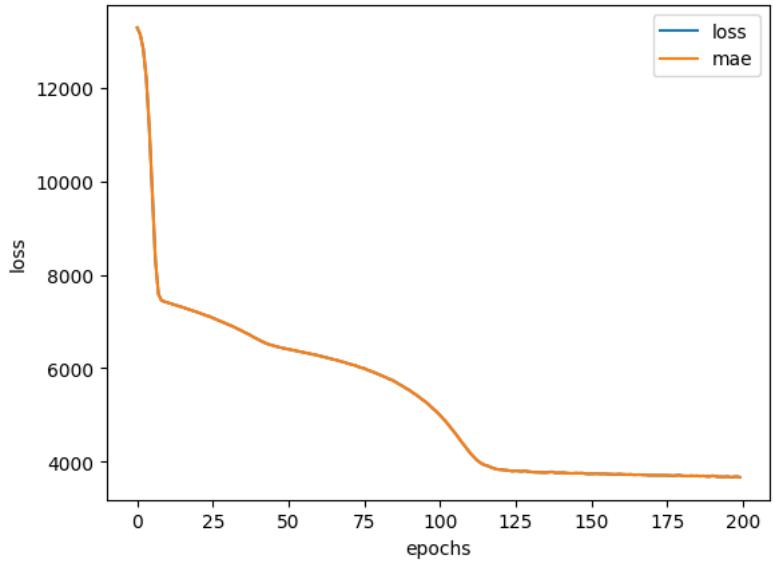
```
# Evaluate the third model
insurance_model_3.evaluate(X_test,y_test)
```

```
9/9 [==============================] - 0s 3ms/step - loss: 3496.1213 - mae: 3496.1213
[3496.121337890625, 3496.121337890625]
```

```
insurance_model.evaluate(X_test,y_test)
```

```
9/9 [==============================] - 0s 4ms/step - loss: 8372.2754 - mae: 8372.2754
[8372.275390625, 8372.275390625]
```

```
# Plot history (also known as a loss curve)
pd.DataFrame(history.history).plot()
plt.ylabel("loss")
plt.xlabel("epochs");
```



🤔 Question: How long should you train for?

It depends on what problem you're working on. Sometimes training won't take very long, other times it'll take longer than you expect. A common method is to set your model training for a very long time (e.g. 1000's of epochs) but set it up with an EarlyStopping callback so it stops automatically when it stops improving a certain metric.

Preprocessing data(normalization and standardization)

In terms of scaling values,neural networks tend to prefer normalization.

If you're not sure on which to use,you could try both and see which perfroms better. https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02

```
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf

# Read in the insurance data set
insurance = pd.read_csv("https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/maste
insurance
```

To prepare our data we can borrow few classes from scikit-learn.

```
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import MinMaxScaler,OneHotEncoder
from sklearn.model_selection import train_test_split

# Create a column transformer
ct=make_column_transformer(
    (MinMaxScaler(),["age","bmi","children"])  #turn all values in these columns between 0 and 1
    ,(OneHotEncoder(handle_unknown="ignore"),["sex","smoker","region"])
)

# Create X & y
X= insurance.drop("charges",axis=1)
y= insurance["charges"]

# Build our train and test sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

# fit the column transformer to our training data
ct.fit(X_train)

# Transform training and test data wih normalization(MinMaxScaler)and OneHotEncoder
X_train_normal=ct.transform(X_train)
X_test_normal = ct.transform(X_test)

```

```
# What does our data look like now?
X_train.loc[0]
```

```
age               19
sex           female
bmi             27.9
children           0
smoker           yes
region     southwest
Name: 0, dtype: object
```

```
X_train_normal[0]
```

```
array([0.60869565, 0.10734463, 0.4       , 1.        , 0.        ,
       1.        , 0.        , 0.        , 1.        , 0.        ,
       0.        ])
```

```
X_train.shape,X_train_normal.shape
```

```
((1070, 6), (1070, 11))
```

Our data has been normalized and one hot encoded. Now et's build a neural network model on it and see how it goes.

```
# build a neural network model to fit on our normalized data
tf.random.set_seed(42)

# 1. create the model
insurance_model_4=tf.keras.Sequential([
    tf.keras.layers.Dense(100),
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1)
])

# 2.compile the data
insurance_model_4.compile(loss=tf.keras.losses.mae,
                          optimizer=tf.keras.optimizers.Adam(),
                          metrics=["mae"])
# 3. Fit the model
insurance_model_4.fit(X_train_normal,y_train,epochs=100)
```

```
Epoch 1/100
34/34 [==============================] - 1s 4ms/step - loss: 13344.0762 - mae: 13344.0762
Epoch 2/100
34/34 [==============================] - 0s 3ms/step - loss: 13335.9180 - mae: 13335.9180
Epoch 3/100
34/34 [==============================] - 0s 4ms/step - loss: 13315.5146 - mae: 13315.5146
Epoch 4/100
34/34 [==============================] - 0s 3ms/step - loss: 13272.3369 - mae: 13272.3369
Epoch 5/100
34/34 [==============================] - 0s 3ms/step - loss: 13195.1064 - mae: 13195.1064
Epoch 6/100
34/34 [==============================] - 0s 3ms/step - loss: 13072.4014 - mae: 13072.4014
Epoch 7/100
34/34 [==============================] - 0s 4ms/step - loss: 12893.7852 - mae: 12893.7852
Epoch 8/100
34/34 [==============================] - 0s 3ms/step - loss: 12649.1680 - mae: 12649.1680
Epoch 9/100
34/34 [==============================] - 0s 3ms/step - loss: 12328.3975 - mae: 12328.3975
Epoch 10/100
34/34 [==============================] - 0s 4ms/step - loss: 11926.6855 - mae: 11926.6855
Epoch 11/100
34/34 [==============================] - 0s 3ms/step - loss: 11453.1719 - mae: 11453.1719
Epoch 12/100
34/34 [==============================] - 0s 3ms/step - loss: 10946.7520 - mae: 10946.7520
Epoch 13/100
34/34 [==============================] - 0s 3ms/step - loss: 10444.3008 - mae: 10444.3008
Epoch 14/100
34/34 [==============================] - 0s 3ms/step - loss: 9947.9414 - mae: 9947.9414
Epoch 15/100
34/34 [==============================] - 0s 4ms/step - loss: 9480.2754 - mae: 9480.2754
Epoch 16/100
34/34 [==============================] - 0s 4ms/step - loss: 9065.0439 - mae: 9065.0439
Epoch 17/100
34/34 [==============================] - 0s 4ms/step - loss: 8719.4248 - mae: 8719.4248
Epoch 18/100
34/34 [==============================] - 0s 3ms/step - loss: 8438.3389 - mae: 8438.3389
Epoch 19/100
34/34 [==============================] - 0s 3ms/step - loss: 8224.1338 - mae: 8224.1338
Epoch 20/100
34/34 [==============================] - 0s 3ms/step - loss: 8077.8018 - mae: 8077.8018
Epoch 21/100
34/34 [==============================] - 0s 4ms/step - loss: 7969.2271 - mae: 7969.2271
Epoch 22/100
34/34 [==============================] - 0s 3ms/step - loss: 7894.3066 - mae: 7894.3066
Epoch 23/100
34/34 [==============================] - 0s 4ms/step - loss: 7835.4468 - mae: 7835.4468
Epoch 24/100
34/34 [==============================] - 0s 3ms/step - loss: 7782.5552 - mae: 7782.5552
Epoch 25/100
34/34 [==============================] - 0s 2ms/step - loss: 7743.9800 - mae: 7743.9800
Epoch 26/100
34/34 [==============================] - 0s 2ms/step - loss: 7692.2397 - mae: 7692.2397
Epoch 27/100
34/34 [==============================] - 0s 2ms/step - loss: 7649.6465 - mae: 7649.6465
Epoch 28/100
34/34 [==============================] - 0s 2ms/step - loss: 7607.0825 - mae: 7607.0825
Epoch 29/100
34/34 [==============================] - 0s 2ms/step - loss: 7564.2417 - mae: 7564.2417
```

```
# Evaluate our insurance model trained on normalized data
insurance_model_4.evaluate(X_test_normal,y_test)
```

```
9/9 [==============================] - 0s 3ms/step - loss: 3435.2017 - mae: 3435.2017
[3435.20166015625, 3435.20166015625]
```

```
# insurance_model_2 results
#9/9 [==============================] - 0s 4ms/step - loss: 4963.6123 - mae: 4963.6123
```

```
insurance_model_2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 100)               1200

 dense_3 (Dense)             (None, 10)                1010

 dense_4 (Dense)             (None, 1)                 11

=================================================================
Total params: 2,221
Trainable params: 2,221
Non-trainable params: 0
_________________________________________________________________
```

✓ 0s completed at 11:15 PM