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

```
#reading the data
df = pd.read_csv('https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/
```

```
df.head(10)
```

	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
5	31	female	25.740	0	no	southeast	3756.62160
6	46	female	33.440	1	no	southeast	8240.58960
7	37	female	27.740	3	no	northwest	7281.50560
8	37	male	29.830	2	no	northeast	6406.41070
9	60	female	25.840	0	no	northwest	28923.13692

```
#one-hot encoding
data=pd.get_dummies(df)
```

```
data.head()
```

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

```
X = data.drop('charges',axis=1)
```

```
Y = data['charges']
```

```
X.shape,Y.shape
```

```
    ((1338, 11), (1338,))
```

```
X.isnull().sum()
```

```
    age                 0
    bmi                 0
    children            0
    sex_female          0
    sex_male            0
    smoker_no           0
    smoker_yes          0
    region_northeast    0
    region_northwest    0
    region_southeast    0
    region_southwest    0
    dtype: int64
```

```
Y.head()
```

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

```
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=12)
```

```
len(x_test),len(x_train),len(X)
```

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

```
#building of neural network
tf.random.set_seed(42)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(5),
    tf.keras.layers.Dense(1)
])

#model compile
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),
              loss=tf.keras.losses.mae,
              metrics=['mae'])
```

```
model.fit(x_train,y_train,epochs=100,verbose=1)
```

```
Epoch 1/100
34/34 [==============================] - 1s 2ms/step - loss: 13296.8994 - mae: 13296.
Epoch 2/100
34/34 [==============================] - 0s 2ms/step - loss: 13276.6768 - mae: 13276.
Epoch 3/100
34/34 [==============================] - 0s 2ms/step - loss: 13249.2158 - mae: 13249.
Epoch 4/100
34/34 [==============================] - 0s 2ms/step - loss: 13208.4844 - mae: 13208.
Epoch 5/100
34/34 [==============================] - 0s 2ms/step - loss: 13147.5322 - mae: 13147.
Epoch 6/100
34/34 [==============================] - 0s 2ms/step - loss: 13058.4346 - mae: 13058.
Epoch 7/100
34/34 [==============================] - 0s 2ms/step - loss: 12933.6240 - mae: 12933.
Epoch 8/100
34/34 [==============================] - 0s 2ms/step - loss: 12765.7529 - mae: 12765.
Epoch 9/100
34/34 [==============================] - 0s 2ms/step - loss: 12546.7510 - mae: 12546.
Epoch 10/100
34/34 [==============================] - 0s 2ms/step - loss: 12268.0410 - mae: 12268.
Epoch 11/100
34/34 [==============================] - 0s 2ms/step - loss: 11922.6074 - mae: 11922.
Epoch 12/100
34/34 [==============================] - 0s 2ms/step - loss: 11506.0449 - mae: 11506.
Epoch 13/100
34/34 [==============================] - 0s 2ms/step - loss: 11030.4854 - mae: 11030.
Epoch 14/100
34/34 [==============================] - 0s 2ms/step - loss: 10516.0449 - mae: 10516.
Epoch 15/100
34/34 [==============================] - 0s 2ms/step - loss: 9992.6104 - mae: 9992.61
Epoch 16/100
34/34 [==============================] - 0s 2ms/step - loss: 9476.2295 - mae: 9476.22
Epoch 17/100
34/34 [==============================] - 0s 2ms/step - loss: 8982.4619 - mae: 8982.46
Epoch 18/100
34/34 [==============================] - 0s 2ms/step - loss: 8560.6211 - mae: 8560.62
Epoch 19/100
34/34 [==============================] - 0s 2ms/step - loss: 8217.3086 - mae: 8217.30
Epoch 20/100
34/34 [==============================] - 0s 2ms/step - loss: 7961.0889 - mae: 7961.08
Epoch 21/100
34/34 [==============================] - 0s 2ms/step - loss: 7783.0542 - mae: 7783.05
```

```
34/34 [==============================] - 0s 2ms/step - loss: 7783.0542 - mae: 7783.05
Epoch 22/100
34/34 [==============================] - 0s 3ms/step - loss: 7676.7539 - mae: 7676.75
Epoch 23/100
34/34 [==============================] - 0s 2ms/step - loss: 7621.4028 - mae: 7621.40
Epoch 24/100
34/34 [==============================] - 0s 2ms/step - loss: 7590.7988 - mae: 7590.79
Epoch 25/100
34/34 [==============================] - 0s 2ms/step - loss: 7571.6118 - mae: 7571.61
Epoch 26/100
34/34 [==============================] - 0s 2ms/step - loss: 7558.0625 - mae: 7558.06
Epoch 27/100
34/34 [==============================] - 0s 2ms/step - loss: 7549.2397 - mae: 7549.23
Epoch 28/100
34/34 [==============================] - 0s 2ms/step - loss: 7541.9243 - mae: 7541.92
Epoch 29/100
34/34 [==============================] - 0s 2ms/step - loss: 7534.1299 - mae: 7534.12
```

```
model.evaluate(x_test,y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 6372.5566 - mae: 6372.5566
[6372.556640625, 6372.556640625]
```

```
#building of neural network
tf.random.set_seed(42)

model1 = tf.keras.Sequential([
    tf.keras.layers.Dense(200),
    tf.keras.layers.Dense(150),
    tf.keras.layers.Dense(100),
    tf.keras.layers.Dense(50),
    tf.keras.layers.Dense(25),
    tf.keras.layers.Dense(5),
    tf.keras.layers.Dense(1)
])

#model compile
model1.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.01),
              loss=tf.keras.losses.mae,
              metrics=['mae'])
```

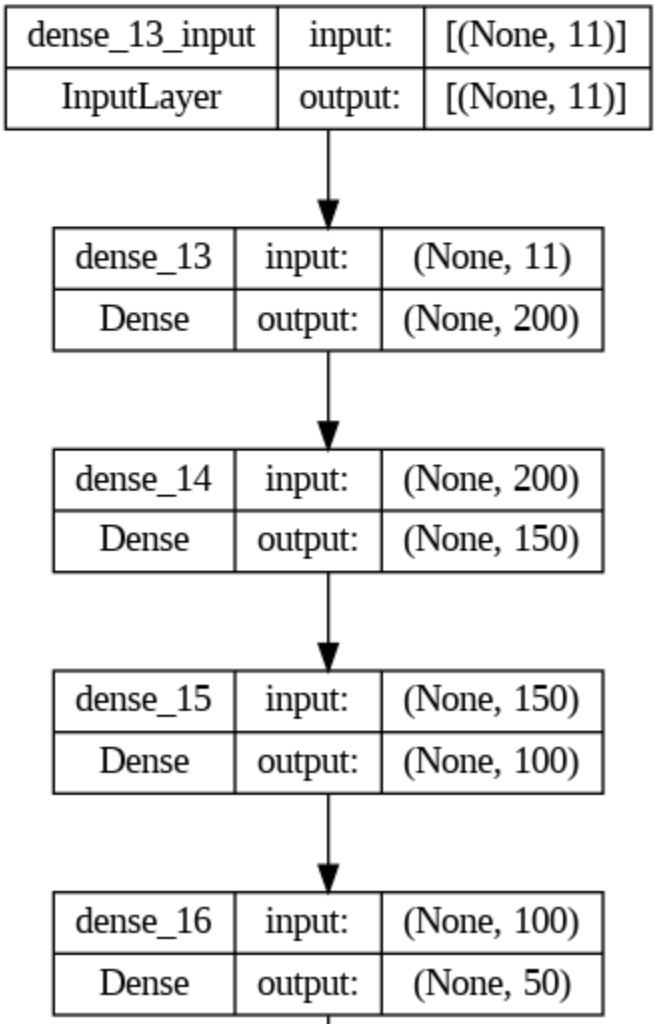
```
history=model1.fit(x_train,y_train,epochs=500)
```

```
Epoch 1/500
34/34 [==============================] - 2s 3ms/step - loss: 8517.8916 - mae: 8517.89
Epoch 2/500
34/34 [==============================] - 0s 3ms/step - loss: 6798.7236 - mae: 6798.72
Epoch 3/500
34/34 [==============================] - 0s 3ms/step - loss: 5794.1528 - mae: 5794.15
Epoch 4/500
34/34 [==============================] - 0s 3ms/step - loss: 4477.3628 - mae: 4477.36
Epoch 5/500
Epoch 5/500
34/34 [==============================] - 0s 4ms/step - loss: 4312.4199 - mae: 4312.41
Epoch 6/500
34/34 [==============================] - 0s 7ms/step - loss: 4819.9590 - mae: 4819.95
Epoch 7/500
34/34 [==============================] - 0s 5ms/step - loss: 4461.5122 - mae: 4461.51
Epoch 8/500
34/34 [==============================] - 0s 6ms/step - loss: 5152.4512 - mae: 5152.45
Epoch 9/500
34/34 [==============================] - 0s 6ms/step - loss: 3868.7495 - mae: 3868.74
Epoch 10/500
34/34 [==============================] - 0s 6ms/step - loss: 3867.8406 - mae: 3867.84
Epoch 11/500
34/34 [==============================] - 0s 7ms/step - loss: 3913.2742 - mae: 3913.27
Epoch 12/500
34/34 [==============================] - 0s 6ms/step - loss: 4198.9302 - mae: 4198.93
Epoch 13/500
34/34 [==============================] - 0s 8ms/step - loss: 4610.8765 - mae: 4610.87
Epoch 14/500
34/34 [==============================] - 0s 6ms/step - loss: 3867.9399 - mae: 3867.93
Epoch 15/500
34/34 [==============================] - 0s 6ms/step - loss: 3957.4131 - mae: 3957.41
Epoch 16/500
34/34 [==============================] - 0s 6ms/step - loss: 3984.2061 - mae: 3984.20
Epoch 17/500
34/34 [==============================] - 0s 6ms/step - loss: 3999.6370 - mae: 3999.63
Epoch 18/500
34/34 [==============================] - 0s 6ms/step - loss: 3873.9790 - mae: 3873.97
Epoch 19/500
34/34 [==============================] - 0s 7ms/step - loss: 3769.0115 - mae: 3769.01
Epoch 20/500
34/34 [==============================] - 0s 6ms/step - loss: 3617.2151 - mae: 3617.21
Epoch 21/500
34/34 [==============================] - 0s 8ms/step - loss: 4098.7314 - mae: 4098.73
Epoch 22/500
34/34 [==============================] - 0s 10ms/step - loss: 3726.7959 - mae: 3726.7
Epoch 23/500
34/34 [==============================] - 0s 9ms/step - loss: 3641.4900 - mae: 3641.49
Epoch 24/500
34/34 [==============================] - 0s 8ms/step - loss: 4062.8201 - mae: 4062.82
Epoch 25/500
34/34 [==============================] - 0s 8ms/step - loss: 3768.7891 - mae: 3768.78
Epoch 26/500
34/34 [==============================] - 0s 8ms/step - loss: 3601.5254 - mae: 3601.52
Epoch 27/500
34/34 [==============================] - 0s 8ms/step - loss: 3811.0088 - mae: 3811.00
Epoch 28/500
34/34 [==============================] - 0s 9ms/step - loss: 3675.4670 - mae: 3675.46
Epoch 29/500
34/34 [==============================] - 0s 9ms/step - loss: 3677.6379 - mae: 3677.63
```

```
model1.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
 Layer (type)                Output Shape              Param #
=================================================================
 dense_13 (Dense)            (None, 200)               2400

 dense_14 (Dense)            (None, 150)               30150

 dense_15 (Dense)            (None, 100)               15100

 dense_16 (Dense)            (None, 50)                5050

 dense_17 (Dense)            (None, 25)                1275

 dense_18 (Dense)            (None, 5)                 130

 dense_19 (Dense)            (None, 1)                 6

=================================================================
Total params: 54,111
Trainable params: 54,111
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.utils import plot_model
plot_model(model1,show_shapes=True)
```



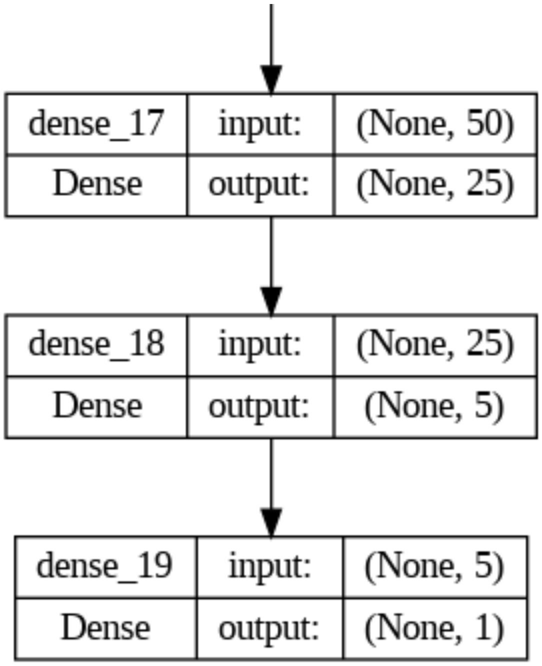
dense_13_input | input: | [(None, 11)]
InputLayer | output: | [(None, 11)]

dense_13 | input: | (None, 11)
Dense | output: | (None, 200)

dense_14 | input: | (None, 200)
Dense | output: | (None, 150)

dense_15 | input: | (None, 150)
Dense | output: | (None, 100)

dense_16 | input: | (None, 100)
Dense | output: | (None, 50)

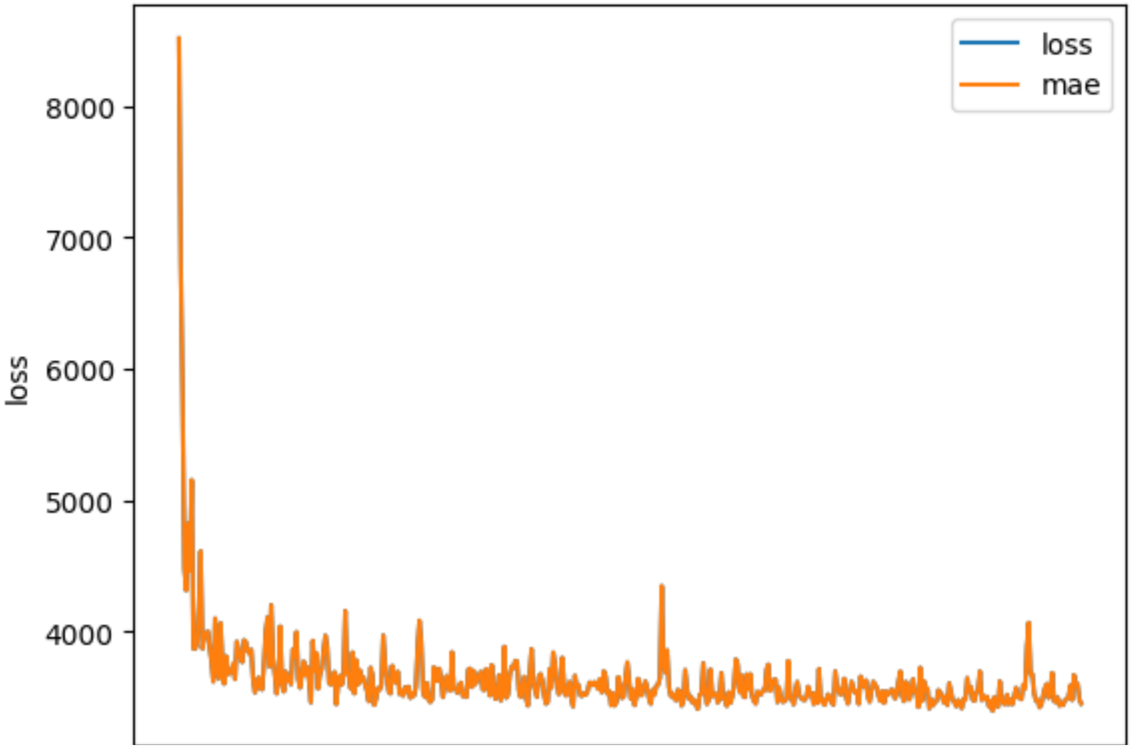


```
model1.evaluate(x_test,y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 3573.0610 - mae: 3573.0610
[3573.06103515625, 3573.06103515625]
```

```
#plot history or we can say training curve
pd.DataFrame(history.history).plot()
plt.ylabel('loss')
plt.xlabel('epochs')
```

```
Text(0.5, 0, 'epochs')
```



0
100
200
300
400
500
epochs