```
# Import required libraries
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Read the dataset
insurance = pd.read_csv("insurance.csv")
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

We need to convert our categorical data into numerical data. For this, we will use pandas `get_dummies` method to convert categorical data into numerical data by one hot encoding.

```
# One hot encoding our DataFrame
insurance_one_hot = pd.get_dummies(insurance)
insurance_one_hot.head()
```

	age	bmi	children	charges	sex_female	sex_male	smoker_no	smoker_yes
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

Creating **Features** (X) and **Labels**(y)

```
X = insurance_one_hot.drop("charges", axis=1)
y = insurance_one_hot["charges"]
```

```
X.head()
```

	age	bmi	children	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_no
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	

```
y.head()
```

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

Now let's create a train and test set. For this, we will use sklearn's `train_test_split`

```
# Import train_test_split and split the data into train and test set
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)
```

```
# Let's check the shapes of our split data set
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((1070, 11), (1070,), (268, 11), (268,))
```

```
# Now let's build a neural network for this data
tf.random.set_seed=42

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

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

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

```
    34/34 [==============================] - 0s 1ms/step - loss: 7332.9751 - mae: 7332.9751
    Epoch 72/100
    34/34 [==============================] - 0s 1ms/step - loss: 7263.4653 - mae: 7263.4653
    Epoch 73/100
    34/34 [==============================] - 0s 1ms/step - loss: 7368.5991 - mae: 7368.5991
    Epoch 74/100
    34/34 [==============================] - 0s 1ms/step - loss: 7264.3057 - mae: 7264.3057
    Epoch 75/100
    34/34 [==============================] - 0s 1ms/step - loss: 7430.5791 - mae: 7430.5791
    Epoch 76/100
    34/34 [==============================] - 0s 1ms/step - loss: 7378.0332 - mae: 7378.0332
    Epoch 77/100
    34/34 [==============================] - 0s 1ms/step - loss: 7537.7036 - mae: 7537.7036
    Epoch 78/100
    34/34 [==============================] - 0s 2ms/step - loss: 7085.6025 - mae: 7085.6025
    Epoch 79/100
    34/34 [==============================] - 0s 1ms/step - loss: 7394.8496 - mae: 7394.8496
    Epoch 80/100
    34/34 [==============================] - 0s 1ms/step - loss: 7341.1792 - mae: 7341.1792
    Epoch 81/100
    34/34 [==============================] - 0s 1ms/step - loss: 7181.1934 - mae: 7181.1934
    Epoch 82/100
    34/34 [==============================] - 0s 1ms/step - loss: 7331.5215 - mae: 7331.5215
    Epoch 83/100
    34/34 [==============================] - 0s 1ms/step - loss: 7483.0605 - mae: 7483.0605
    Epoch 84/100
    34/34 [==============================] - 0s 1ms/step - loss: 7223.4634 - mae: 7223.4634
    Epoch 85/100
    34/34 [==============================] - 0s 1ms/step - loss: 7600.2646 - mae: 7600.2646
    Epoch 86/100
    34/34 [==============================] - 0s 1ms/step - loss: 7545.9370 - mae: 7545.9370
    Epoch 87/100
    34/34 [==============================] - 0s 1ms/step - loss: 7306.1094 - mae: 7306.1094
    Epoch 88/100
    34/34 [==============================] - 0s 1ms/step - loss: 7369.8037 - mae: 7369.8037
    Epoch 89/100
    34/34 [==============================] - 0s 2ms/step - loss: 7154.6943 - mae: 7154.6943
    Epoch 90/100
    34/34 [==============================] - 0s 1ms/step - loss: 7075.6602 - mae: 7075.6602
    Epoch 91/100
    34/34 [==============================] - 0s 2ms/step - loss: 7469.7051 - mae: 7469.7051
    Epoch 92/100
    34/34 [==============================] - 0s 2ms/step - loss: 7256.0098 - mae: 7256.0098
    Epoch 93/100
    34/34 [==============================] - 0s 1ms/step - loss: 7561.3916 - mae: 7561.3916
    Epoch 94/100
    34/34 [==============================] - 0s 1ms/step - loss: 7279.7432 - mae: 7279.7432
    Epoch 95/100
    34/34 [==============================] - 0s 1ms/step - loss: 7383.5327 - mae: 7383.5327
    Epoch 96/100
    34/34 [==============================] - 0s 1ms/step - loss: 7556.0991 - mae: 7556.0991
    Epoch 97/100
    34/34 [==============================] - 0s 2ms/step - loss: 7255.0537 - mae: 7255.0537
    Epoch 98/100
    34/34 [==============================] - 0s 1ms/step - loss: 7066.4873 - mae: 7066.4873
    Epoch 99/100
    34/34 [==============================] - 0s 1ms/step - loss: 7119.2231 - mae: 7119.2231
    Epoch 100/100
    34/34 [==============================] - 0s 1ms/step - loss: 7347.1284 - mae: 7347.1284
    <tensorflow.python.keras.callbacks.History at 0x7f0fb4275e90>
```

```
# Check the results of our model with test data
model.evaluate(X_test, y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 7982.7554 - mae: 7982.7554
[7982.75537109375, 7982.75537109375]
```

Well, it seems that the model is not performing up to the mark ! Let's tweak the model a bit and see if we get a better result!

```
# Let's try a model with a higher hidden layer and relu activation
tf.random.set_seed=42
# 1.Create a model
model_1 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(1)
])

# 2. Compile the model
model_1.compile(loss=tf.keras.losses.mae,
                optimizer=tf.keras.optimizers.SGD(),
                metrics=["mae"])

# 3. Fit the model
model_1.fit(X_train, y_train, epochs = 100)
```

```
Epoch 72/100
34/34 [==============================] - 0s 2ms/step - loss: 8283.7705 - mae: 8283.7705
Epoch 73/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.1973 - mae: 8283.1973
Epoch 74/100
34/34 [==============================] - 0s 1ms/step - loss: 8284.0312 - mae: 8284.0312
Epoch 75/100
34/34 [==============================] - 0s 1ms/step - loss: 8282.8877 - mae: 8282.8877
Epoch 76/100
34/34 [==============================] - 0s 1ms/step - loss: 8284.0068 - mae: 8284.0068
Epoch 77/100
34/34 [==============================] - 0s 1ms/step - loss: 8285.0781 - mae: 8285.0781
Epoch 78/100
34/34 [==============================] - 0s 2ms/step - loss: 8283.5732 - mae: 8283.5732
Epoch 79/100
34/34 [==============================] - 0s 1ms/step - loss: 8284.0332 - mae: 8284.0332
Epoch 80/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.8770 - mae: 8283.8770
Epoch 81/100
34/34 [==============================] - 0s 3ms/step - loss: 8284.0654 - mae: 8284.0654
Epoch 82/100
34/34 [==============================] - 0s 2ms/step - loss: 8283.0439 - mae: 8283.0439
Epoch 83/100
34/34 [==============================] - 0s 1ms/step - loss: 8282.4688 - mae: 8282.4688
Epoch 84/100
34/34 [==============================] - 0s 2ms/step - loss: 8284.3271 - mae: 8284.3271
Epoch 85/100
34/34 [==============================] - 0s 1ms/step - loss: 8284.0322 - mae: 8284.0322
Epoch 86/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.5283 - mae: 8283.5283
Epoch 87/100
```

```
34/34 [==============================] - 0s 1ms/step - loss: 8284.0781 - mae: 8284.0781
Epoch 88/100
34/34 [==============================] - 0s 1ms/step - loss: 8284.9541 - mae: 8284.9541
Epoch 89/100
34/34 [==============================] - 0s 1ms/step - loss: 8284.9355 - mae: 8284.9355
Epoch 90/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.2588 - mae: 8283.2588
Epoch 91/100
34/34 [==============================] - 0s 1ms/step - loss: 8285.0654 - mae: 8285.0654
Epoch 92/100
34/34 [==============================] - 0s 1ms/step - loss: 8284.2539 - mae: 8284.2539
Epoch 93/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.5557 - mae: 8283.5557
Epoch 94/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.3428 - mae: 8283.3428
Epoch 95/100
34/34 [==============================] - 0s 1ms/step - loss: 8282.9150 - mae: 8282.9150
Epoch 96/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.0645 - mae: 8283.0645
Epoch 97/100
34/34 [==============================] - 0s 1ms/step - loss: 8282.8350 - mae: 8282.8350
Epoch 98/100
34/34 [==============================] - 0s 2ms/step - loss: 8283.7832 - mae: 8283.7832
Epoch 99/100
34/34 [==============================] - 0s 2ms/step - loss: 8283.1016 - mae: 8283.1016
Epoch 100/100
34/34 [==============================] - 0s 1ms/step - loss: 8283.3799 - mae: 8283.3799
<tensorflow.python.keras.callbacks.History at 0x7f0fb3129050>
```

```
# Check the results of our model with test data
model_1.evaluate(X_test, y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 8651.7334 - mae: 8651.7334
[8651.7333984375, 8651.7333984375]
```

```
# Let's try a model with a different optimizer
tf.random.set_seed=42
# 1.Create a model
model_2 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(1)
])

# 2. Compile the model
model_2.compile(loss=tf.keras.losses.mae,
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
                metrics=["mae"])

# 3. Fit the model
model_2.fit(X_train, y_train, epochs = 100)
```

```
Epoch 1/100
34/34 [==============================] - 0s 2ms/step - loss: 13056.1895 - mae: 13056.1895
Epoch 2/100
34/34 [==============================] - 0s 3ms/step - loss: 9238.7588 - mae: 9238.7588
Epoch 3/100
34/34 [==============================] - 0s 2ms/step - loss: 7470.2544 - mae: 7470.2544
```

```
Epoch 4/100
34/34 [==============================] - 0s 2ms/step - loss: 7322.2998 - mae: 7322.2998
Epoch 5/100
34/34 [==============================] - 0s 2ms/step - loss: 7201.9595 - mae: 7201.9595
Epoch 6/100
34/34 [==============================] - 0s 1ms/step - loss: 7087.5200 - mae: 7087.5200
Epoch 7/100
34/34 [==============================] - 0s 1ms/step - loss: 6930.0894 - mae: 6930.0894
Epoch 8/100
34/34 [==============================] - 0s 2ms/step - loss: 6767.2153 - mae: 6767.2153
Epoch 9/100
34/34 [==============================] - 0s 2ms/step - loss: 6615.3237 - mae: 6615.3237
Epoch 10/100
34/34 [==============================] - 0s 2ms/step - loss: 6474.4204 - mae: 6474.4204
Epoch 11/100
34/34 [==============================] - 0s 2ms/step - loss: 6382.2559 - mae: 6382.2559
Epoch 12/100
34/34 [==============================] - 0s 1ms/step - loss: 6278.2817 - mae: 6278.2817
Epoch 13/100
34/34 [==============================] - 0s 2ms/step - loss: 6185.1782 - mae: 6185.1782
Epoch 14/100
34/34 [==============================] - 0s 1ms/step - loss: 6088.6411 - mae: 6088.6411
Epoch 15/100
34/34 [==============================] - 0s 1ms/step - loss: 5981.2485 - mae: 5981.2485
Epoch 16/100
34/34 [==============================] - 0s 2ms/step - loss: 5819.2983 - mae: 5819.2983
Epoch 17/100
34/34 [==============================] - 0s 2ms/step - loss: 5636.0405 - mae: 5636.0405
Epoch 18/100
34/34 [==============================] - 0s 2ms/step - loss: 5412.4688 - mae: 5412.4688
Epoch 19/100
34/34 [==============================] - 0s 1ms/step - loss: 5104.8608 - mae: 5104.8608
Epoch 20/100
34/34 [==============================] - 0s 1ms/step - loss: 4696.1636 - mae: 4696.1636
Epoch 21/100
34/34 [==============================] - 0s 2ms/step - loss: 4205.0234 - mae: 4205.0234
Epoch 22/100
34/34 [==============================] - 0s 1ms/step - loss: 3797.3997 - mae: 3797.3997
Epoch 23/100
34/34 [==============================] - 0s 1ms/step - loss: 3601.7830 - mae: 3601.7830
Epoch 24/100
34/34 [==============================] - 0s 1ms/step - loss: 3555.9775 - mae: 3555.9775
Epoch 25/100
34/34 [==============================] - 0s 2ms/step - loss: 3483.5774 - mae: 3483.5774
Epoch 26/100
34/34 [==============================] - 0s 2ms/step - loss: 3422.8474 - mae: 3422.8474
Epoch 27/100
34/34 [==============================] - 0s 1ms/step - loss: 3363.7134 - mae: 3363.7134
Epoch 28/100
34/34 [==============================] - 0s 1ms/step - loss: 3332.0598 - mae: 3332.0598
Epoch 29/100
34/34 [==============================] - 0s 2ms/step - loss: 3309.3633 - mae: 3309.3633
Epoch 30/100
```

```
# Check the results of our model with test data
model_2.evaluate(X_test, y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 1986.9711 - mae: 1986.9711
[1986.9710693359375, 1986.9710693359375]
```

We can see a clear improvement with the changed optimizer. Let's try some more

```
# Let's try a model with a different learning rate and one more hidden layer
tf.random.set_seed=42
# 1.Create a model
model_3 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(1)
])

# 2. Compile the model
model_3.compile(loss=tf.keras.losses.mae,
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
                metrics=["mae"])

# 3. Fit the model
model_3.fit(X_train, y_train, epochs = 100)
```

```
Epoch 1/100
34/34 [==============================] - 1s 2ms/step - loss: 11314.4160 - mae: 11314.4160
Epoch 2/100
34/34 [==============================] - 0s 2ms/step - loss: 7296.1606 - mae: 7296.1606
Epoch 3/100
34/34 [==============================] - 0s 2ms/step - loss: 6822.2778 - mae: 6822.2778
Epoch 4/100
34/34 [==============================] - 0s 2ms/step - loss: 6613.1118 - mae: 6613.1118
Epoch 5/100
34/34 [==============================] - 0s 2ms/step - loss: 6531.8442 - mae: 6531.8442
Epoch 6/100
34/34 [==============================] - 0s 2ms/step - loss: 6231.5352 - mae: 6231.5352
Epoch 7/100
34/34 [==============================] - 0s 2ms/step - loss: 5815.0703 - mae: 5815.0703
Epoch 8/100
34/34 [==============================] - 0s 2ms/step - loss: 4930.5410 - mae: 4930.5410
Epoch 9/100
34/34 [==============================] - 0s 2ms/step - loss: 3933.1089 - mae: 3933.1089
Epoch 10/100
34/34 [==============================] - 0s 2ms/step - loss: 3886.8911 - mae: 3886.8911
Epoch 11/100
34/34 [==============================] - 0s 2ms/step - loss: 3249.7505 - mae: 3249.7505
Epoch 12/100
34/34 [==============================] - 0s 2ms/step - loss: 3284.3779 - mae: 3284.3779
Epoch 13/100
34/34 [==============================] - 0s 2ms/step - loss: 3289.9590 - mae: 3289.9590
Epoch 14/100
34/34 [==============================] - 0s 2ms/step - loss: 2943.9331 - mae: 2943.9331
Epoch 15/100
34/34 [==============================] - 0s 3ms/step - loss: 2824.8767 - mae: 2824.8767
Epoch 16/100
34/34 [==============================] - 0s 2ms/step - loss: 3086.7166 - mae: 3086.7166
Epoch 17/100
34/34 [==============================] - 0s 2ms/step - loss: 2941.0498 - mae: 2941.0498
Epoch 18/100
34/34 [==============================] - 0s 2ms/step - loss: 2753.4429 - mae: 2753.4429
Epoch 19/100
```

```
34/34 [==============================] - 0s 2ms/step - loss: 2684.0005 - mae: 2684.0005
Epoch 20/100
34/34 [==============================] - 0s 2ms/step - loss: 2727.8430 - mae: 2727.8430
Epoch 21/100
34/34 [==============================] - 0s 2ms/step - loss: 2748.2822 - mae: 2748.2822
Epoch 22/100
34/34 [==============================] - 0s 2ms/step - loss: 2556.8987 - mae: 2556.8987
Epoch 23/100
34/34 [==============================] - 0s 2ms/step - loss: 2624.3076 - mae: 2624.3076
Epoch 24/100
34/34 [==============================] - 0s 2ms/step - loss: 2581.5723 - mae: 2581.5723
Epoch 25/100
34/34 [==============================] - 0s 2ms/step - loss: 2580.2134 - mae: 2580.2134
Epoch 26/100
34/34 [==============================] - 0s 2ms/step - loss: 2526.9045 - mae: 2526.9045
Epoch 27/100
34/34 [==============================] - 0s 2ms/step - loss: 2522.5085 - mae: 2522.5085
Epoch 28/100
34/34 [==============================] - 0s 2ms/step - loss: 2655.6672 - mae: 2655.6672
Epoch 29/100
34/34 [==============================] - 0s 2ms/step - loss: 2653.7476 - mae: 2653.7476
Epoch 30/100
```

```
# Check the results of our model with test data
model_3.evaluate(X_test, y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 1840.7301 - mae: 1840.7301
[1840.7301025390625, 1840.7301025390625]
```

```
# Let's try a model with more epochs
tf.random.set_seed=42
# 1.Create a model
model_4 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(1)
])

# 2. Compile the model
model_4.compile(loss=tf.keras.losses.mae,
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.005),
                metrics=["mae"])

# 3. Fit the model
model_4.fit(X_train, y_train, epochs = 300, verbose = 0)
```

```
<tensorflow.python.keras.callbacks.History at 0x7f0fb52ec8d0>
```

```
# Check the results of our model with test data
model_4.evaluate(X_test, y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 1483.9738 - mae: 1483.9738
[1483.9737548828125, 1483.9737548828125]
```

That's a pretty good improvement! Let's try more!

```
# Let's try a model with more epochs
tf.random.set_seed=42
# 1.Create a model
model_5 = tf.keras.Sequential([
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(1)
])

# 2. Compile the model
model_5.compile(loss=tf.keras.losses.mae,
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.005),
                metrics=["mae"])

# 3. Fit the model
model_5.fit(X_train, y_train, epochs = 300, verbose = 0)
```

```
<tensorflow.python.keras.callbacks.History at 0x7f0faed81410>
```

```
# Check the results of our model with test data
model_5.evaluate(X_test, y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 1556.0930 - mae: 1556.0930
[1556.093017578125, 1556.093017578125]
```

```
# Now let's compare all our models
all_models = [["model_1", tf.metrics.mean_absolute_error(y_test, tf.squeeze(model_1.predict(X_test))).n
              ["model_2", tf.metrics.mean_absolute_error(y_test, tf.squeeze(model_2.predict(X_test))).n
              ["model_3", tf.metrics.mean_absolute_error(y_test, tf.squeeze(model_3.predict(X_test))).n
              ["model_4", tf.metrics.mean_absolute_error(y_test, tf.squeeze(model_4.predict(X_test))).n
              ["model_5", tf.metrics.mean_absolute_error(y_test, tf.squeeze(model_5.predict(X_test))).n

models_df = pd.DataFrame(all_models, columns=["model", "mae", "mse"])

models_df
```

	model	mae	mse
2	model_3	1640.729980	21778872.0

Preprocessing data (normalization and standardization)

To prepare our data, we can use the scikit learn's `MinMaxScalar`

```
insurance.head()
```

	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

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

# Create column transformer
ct = make_column_transformer(
    (MinMaxScaler(), ["age", "bmi", "children"]),
    (OneHotEncoder(handle_unknown="ignore"), ["sex", "smoker", "region"])
)

# Set X and y again
X = insurance.drop("charges", axis=1)
y = insurance["charges"]

# Split the data again into 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 with our training data
ct.fit(X_train)

# Transforming training and test data with MinMaxScalar and OneHotEncoder
X_train_normal = ct.transform(X_train)
X_test_normal = ct.transform(X_test)
```

```
# Let's check how our data looks 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.        ])
```

```
# Let's check the shapes
X_train.shape, X_train_normal.shape
```

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

Let's build a neural network model to fit on our normalized and encoded data set

```
# Set random seed
tf.random.set_seed = 42

# 1. Create a model
model = tf.keras.Sequential([
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
      tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.mae,
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              metrics=["mae"])

# 3. Fit the model
model.fit(X_train_normal, y_train, epochs=200, verbose=0)
```

```
<tensorflow.python.keras.callbacks.History at 0x7f0f996ae210>
```

```
#Evaluate the model
model.evaluate(X_test_normal, y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 1589.6372 - mae: 1589.6372
[1589.63720703125, 1589.63720703125]
```

✓ 0s completed at 10:55 PM ● ×