```
import pandas as pd
import numpy as np
```

```
insurance=pd.read_csv('/content/insurance.csv')
```

```
insurance.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

```
insurance.head()
#insurance.tail()
```

	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

```
insurance.shape
```

```
(1338, 7)
```

```
insurance.dtypes
```

```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

```
insurance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
insurance.isna().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

```
insurance_one_hot=pd.get_dummies(insurance)
insurance_one_hot
```

	age	bmi	children	charges	sex_female	sex_male	smoker_no	smoker_yes	region_
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	
...	...	...	...	...	...	...	...	...	
1333	50	30.970	3	10600.54830	0	1	1	0	
1334	18	31.920	0	2205.98080	1	0	1	0	
1335	18	36.850	0	1629.83350	1	0	1	0	
1336	21	25.800	0	2007.94500	1	0	1	0	
1337	61	29.070	0	29141.36030	1	0	0	1	

1338 rows × 12 columns

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

```
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)
```

```
#Build the neural network
import tensorflow as tf
tf.random.set_seed(42)
#create a model
insurance_model=tf.keras.Sequential([
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1)
])
#compile the model
insurance_model.compile(loss=tf.keras.losses.mae,
                        optimizer=tf.keras.optimizers.SGD(),
                        metrics=["mae"]
)
#fit the model
insurance_model.fit(X_train,y_train,epochs=100)
#check the results with insurnce model test data
insurance_model.evaluate(X_test,y_test)
```

```
Epoch 1/100
34/34 [==============================] - 1s 2ms/step - loss: 8637.0996 - mae: 8637.0996
Epoch 2/100
34/34 [==============================] - 0s 2ms/step - loss: 7886.7769 - mae: 7886.7769
Epoch 3/100
34/34 [==============================] - 0s 2ms/step - loss: 7558.1475 - mae: 7558.1475
Epoch 4/100
34/34 [==============================] - 0s 2ms/step - loss: 7792.0220 - mae: 7792.0220
Epoch 5/100
34/34 [==============================] - 0s 2ms/step - loss: 7748.3892 - mae: 7748.3892
Epoch 6/100
34/34 [==============================] - 0s 2ms/step - loss: 7595.3940 - mae: 7595.3940
Epoch 7/100
34/34 [==============================] - 0s 2ms/step - loss: 7589.9849 - mae: 7589.9849
Epoch 8/100
34/34 [==============================] - 0s 2ms/step - loss: 7698.5591 - mae: 7698.5591
Epoch 9/100
34/34 [==============================] - 0s 2ms/step - loss: 7496.7788 - mae: 7496.7788
Epoch 10/100
34/34 [==============================] - 0s 2ms/step - loss: 7493.1743 - mae: 7493.1743
Epoch 11/100
34/34 [==============================] - 0s 2ms/step - loss: 7769.7314 - mae: 7769.7314
Epoch 12/100
34/34 [==============================] - 0s 2ms/step - loss: 7706.9033 - mae: 7706.9033
Epoch 13/100
34/34 [==============================] - 0s 2ms/step - loss: 7687.7227 - mae: 7687.7227
```

```
Epoch 14/100
34/34 [==============================] - 0s 2ms/step - loss: 7689.8999 - mae: 7689.8999
Epoch 15/100
34/34 [==============================] - 0s 2ms/step - loss: 7393.5322 - mae: 7393.5322
Epoch 16/100
34/34 [==============================] - 0s 2ms/step - loss: 7780.6982 - mae: 7780.6982
Epoch 17/100
34/34 [==============================] - 0s 2ms/step - loss: 7578.5093 - mae: 7578.5093
Epoch 18/100
34/34 [==============================] - 0s 2ms/step - loss: 7750.8350 - mae: 7750.8350
Epoch 19/100
34/34 [==============================] - 0s 2ms/step - loss: 7739.2134 - mae: 7739.2134
Epoch 20/100
34/34 [==============================] - 0s 2ms/step - loss: 7875.0635 - mae: 7875.0635
Epoch 21/100
34/34 [==============================] - 0s 2ms/step - loss: 7466.6768 - mae: 7466.6768
Epoch 22/100
34/34 [==============================] - 0s 2ms/step - loss: 7941.2310 - mae: 7941.2310
Epoch 23/100
34/34 [==============================] - 0s 2ms/step - loss: 7640.2725 - mae: 7640.2725
Epoch 24/100
34/34 [==============================] - 0s 2ms/step - loss: 7539.2656 - mae: 7539.2656
Epoch 25/100
34/34 [==============================] - 0s 2ms/step - loss: 7619.9658 - mae: 7619.9658
Epoch 26/100
34/34 [==============================] - 0s 2ms/step - loss: 7644.1709 - mae: 7644.1709
Epoch 27/100
34/34 [==============================] - 0s 2ms/step - loss: 7709.0361 - mae: 7709.0361
Epoch 28/100
34/34 [==============================] - 0s 2ms/step - loss: 7366.8662 - mae: 7366.8662
Epoch 29/100
34/34 [==============================] - 0s 2ms/step - loss: 7444.3135 - mae: 7444.3135
```

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

```
9/9 [==============================] - 0s 3ms/step - loss: 7023.3286 - mae: 7023.3286
[7023.32861328125, 7023.32861328125]
```

```
#Build the neural network
tf.random.set_seed(42)
#create a model
insurance_model_2=tf.keras.Sequential([
    tf.keras.layers.Dense(100),
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1)
])
#compile the model
insurance_model_2.compile(loss=tf.keras.losses.mae,
                          optimizer=tf.keras.optimizers.Adam(),
                          metrics=["mae"]
)
#fit the model
history=insurance_model_2.fit(X_train,y_train,epochs=300)
```

```
Epoch 287/300
34/34 [==============================] - 0s 3ms/step - loss: 3522.8616 - mae: 3522.8616
Epoch 288/300
34/34 [==============================] - 0s 2ms/step - loss: 3522.2529 - mae: 3522.2529
Epoch 289/300
34/34 [==============================] - 0s 3ms/step - loss: 3523.6272 - mae: 3523.6272
Epoch 290/300
34/34 [==============================] - 0s 3ms/step - loss: 3526.8005 - mae: 3526.8005
Epoch 291/300
34/34 [==============================] - 0s 3ms/step - loss: 3529.2947 - mae: 3529.2947
Epoch 292/300
34/34 [==============================] - 0s 2ms/step - loss: 3527.0535 - mae: 3527.0535
Epoch 293/300
34/34 [==============================] - 0s 2ms/step - loss: 3517.7183 - mae: 3517.7183
Epoch 294/300
34/34 [==============================] - 0s 2ms/step - loss: 3532.4072 - mae: 3532.4072
Epoch 295/300
34/34 [==============================] - 0s 2ms/step - loss: 3527.7683 - mae: 3527.7683
Epoch 296/300
34/34 [==============================] - 0s 2ms/step - loss: 3517.7471 - mae: 3517.7471
Epoch 297/300
34/34 [==============================] - 0s 2ms/step - loss: 3507.5869 - mae: 3507.5869
Epoch 298/300
34/34 [==============================] - 0s 2ms/step - loss: 3508.5842 - mae: 3508.5842
Epoch 299/300
34/34 [==============================] - 0s 3ms/step - loss: 3521.3545 - mae: 3521.3545
Epoch 300/300
34/34 [==============================] - 0s 3ms/step - loss: 3511.0706 - mae: 3511.0706
```

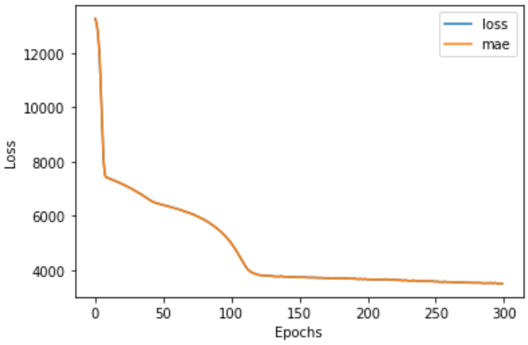
```
insurance_model_2.evaluate(X_test,y_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 3217.6089 - mae: 3217.6089
[3217.60888671875, 3217.60888671875]
```

```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
#plot history (Also known as loss  curve or a training curve)
pd.DataFrame(history.history).plot()
plt.ylabel("Loss")
plt.xlabel("Epochs")
```

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



```
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import MinMaxScaler,OneHotEncoder
from sklearn.model_selection import train_test_split
#create column transformer
ct=make_column_transformer(
   (MinMaxScaler(),["age","bmi","children"]),#turn the values betweeen range of 0 and 1
   (OneHotEncoder(handle_unknown="ignore"),      ["sex","smoker","region"])
)
```

```
#create X and y
X =insurance.drop("charges",axis=1)
y=insurance["charges"]
#build train and test data
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)
#tranfome training and test data with normailization(minmax scaler and normalization)
X_train_normal=ct.transform(X_train)
X_test_normal=ct.transform(X_test)
```

```
X_train_normal[0]
```

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

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

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

```
X_test_normal[0]
```

```
array([0.58695652, 0.24791499, 0.4       , 1.        , 0.        ,
       1.        , 0.        , 1.        , 0.        , 0.        ,
       0.        ])
```

```
#Build the neural network
tf.random.set_seed(42)
#create a model
insurance_model_4=tf.keras.Sequential([
    tf.keras.layers.Dense(100),
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1)
])
#compile the model
insurance_model_4.compile(
                         loss=tf.keras.losses.mae,
                         optimizer=tf.keras.optimizers.Adam(),
                         metrics=["mae"]
)
#fit the model
history=insurance_model_4.fit(X_train_normal,y_train,epochs=300)
```

```
Epoch 1/300
34/34 [==============================] - 1s 3ms/step - loss: 13342.6494 - mae: 13342.6494
Epoch 2/300
34/34 [==============================] - 0s 2ms/step - loss: 13333.4785 - mae: 13333.4785
Epoch 3/300
34/34 [==============================] - 0s 2ms/step - loss: 13312.0234 - mae: 13312.0234
Epoch 4/300
34/34 [==============================] - 0s 3ms/step - loss: 13267.7930 - mae: 13267.7930
Epoch 5/300
34/34 [==============================] - 0s 2ms/step - loss: 13189.5830 - mae: 13189.5830
Epoch 6/300
34/34 [==============================] - 0s 3ms/step - loss: 13066.4502 - mae: 13066.4502
Epoch 7/300
34/34 [==============================] - 0s 3ms/step - loss: 12888.1953 - mae: 12888.1953
Epoch 8/300
34/34 [==============================] - 0s 4ms/step - loss: 12644.6523 - mae: 12644.6523
Epoch 9/300
34/34 [==============================] - 0s 3ms/step - loss: 12325.5469 - mae: 12325.5469
Epoch 10/300
34/34 [==============================] - 0s 3ms/step - loss: 11925.9658 - mae: 11925.9658
Epoch 11/300
34/34 [==============================] - 0s 3ms/step - loss: 11454.3350 - mae: 11454.3350
Epoch 12/300
34/34 [==============================] - 0s 3ms/step - loss: 10949.8076 - mae: 10949.8076
Epoch 13/300
34/34 [==============================] - 0s 3ms/step - loss: 10448.9404 - mae: 10448.9404
Epoch 14/300
34/34 [==============================] - 0s 3ms/step - loss: 9951.6250 - mae: 9951.6250
Epoch 15/300
34/34 [==============================] - 0s 2ms/step - loss: 9482.7422 - mae: 9482.7422
Epoch 16/300
34/34 [==============================] - 0s 3ms/step - loss: 9066.7461 - mae: 9066.7461
Epoch 17/300
34/34 [==============================] - 0s 2ms/step - loss: 8721.9854 - mae: 8721.9854
Epoch 18/300
34/34 [==============================] - 0s 3ms/step - loss: 8441.2002 - mae: 8441.2002
Epoch 19/300
34/34 [==============================] - 0s 2ms/step - loss: 8227.5117 - mae: 8227.5117
Epoch 20/300
34/34 [==============================] - 0s 3ms/step - loss: 8081.9775 - mae: 8081.9775
Epoch 21/300
34/34 [==============================] - 0s 2ms/step - loss: 7973.8945 - mae: 7973.8945
Epoch 22/300
34/34 [==============================] - 0s 2ms/step - loss: 7899.1597 - mae: 7899.1597
Epoch 23/300
34/34 [==============================] - 0s 2ms/step - loss: 7840.3906 - mae: 7840.3906
Epoch 24/300
34/34 [==============================] - 0s 3ms/step - loss: 7787.9619 - mae: 7787.9619
Epoch 25/300
34/34 [==============================] - 0s 2ms/step - loss: 7749.2622 - mae: 7749.2622
Epoch 26/300
34/34 [==============================] - 0s 3ms/step - loss: 7697.9600 - mae: 7697.9600
Epoch 27/300
34/34 [==============================] - 0s 2ms/step - loss: 7656.0269 - mae: 7656.0269
```

```
Epoch 28/300
34/34 [==============================] - 0s 3ms/step - loss: 7613.4780 - mae: 7613.4780
Epoch 29/300
34/34 [==============================] - 0s 3ms/step - loss: 7570.9482 - mae: 7570.9482
```

✓ 34s completed at 8:20 PM