In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as Tnr
#Read the training data
train_data = pd.read_csv('salary_data_cleaned.csv')
```

In [2]:

```
#Print the data
print(train_data.head())
```

```
   Rating  min_salary  max_salary  avg_salary  same_state  age  Founded
0     3.8          53          91        72.0           0   47     1973
1     3.4          63         112        87.5           0   36     1984
2     4.8          80          90        85.0           1   10     2010
3     3.8          56          97        76.5           1   55     1965
4     2.9          86         143       114.5           1   22     1998
```

In [3]:

```
#Print the dimesnion of the data
train_data.shape
```

Out[3]:

```
(690, 7)
```

In [4]:

```
#separating X train and Y train
X_train = train_data[['min_salary','max_salary','avg_salary','same_state','age']]
Y_train = train_data[['Rating']]
```

In [5]:

```
# importing train_test_split from sklearn
from sklearn.model_selection import train_test_split
```

In [6]:

```
# splitting the data
X_train, X_test, Y_train, Y_test = train_test_split(X_train, Y_train, test_size = 0.3, random_state = 0)
```

In [7]:

```
#print the shape of train and test data after spltting
print (X_train.shape)
print (Y_train.shape)
print (X_test.shape)
print (Y_test.shape)
```

```
(483, 5)
(483, 1)
(207, 5)
(207, 1)
```

In [8]:

```
from keras.layers import Dense
from keras.models import Sequential
```

In [9]:

```
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='tanh'))
model.add(Dense(32, activation='tanh'))
model.add(Dense(1, activation='tanh'))
```

In [10]:
```
model.compile(optimizer=Tnr.keras.optimizers.Adam(learning_rate=0.001),
                  loss=Tnr.keras.losses.BinaryCrossentropy(), metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=200, batch_size=32, verbose=1)
```

```
Epoch 1/200
16/16 [==============================] - 1s 2ms/step - loss: -5.6536 - accuracy: 0.0000e+00
Epoch 2/200
16/16 [==============================] - 0s 1ms/step - loss: -21.7389 - accuracy: 0.0000e+00
Epoch 3/200
16/16 [==============================] - 0s 1ms/step - loss: -32.8103 - accuracy: 0.0000e+00
Epoch 4/200
16/16 [==============================] - 0s 1ms/step - loss: -39.0781 - accuracy: 0.0000e+00
Epoch 5/200
16/16 [==============================] - 0s 1ms/step - loss: -41.1261 - accuracy: 0.0000e+00
Epoch 6/200
16/16 [==============================] - 0s 1ms/step - loss: -41.2284 - accuracy: 0.0000e+00
Epoch 7/200
16/16 [==============================] - 0s 1ms/step - loss: -41.2410 - accuracy: 0.0000e+00
Epoch 8/200
16/16 [==============================] - 0s 2ms/step - loss: -41.2426 - accuracy: 0.0000e+00
Epoch 9/200
16/16 [==============================] - 0s 1ms/step - loss: -41.2521 - accuracy: 0.0000e+00
Epoch 10/200
16/16 [==============================] - 0s 1ms/step - loss: -41.2603 - accuracy: 0.0000e+00
```

In [11]:
```
# Evaluate the model on the test set
test_loss, test_acc1 = model.evaluate(X_test, Y_test, verbose=0)
```

In [12]:
```
# Build the model with ReLU activation function
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='relu'))
```

In [13]:
```
# Compile the model
model.compile(optimizer=Tnr.keras.optimizers.Adam(learning_rate=0.001),
                  loss=Tnr.keras.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

In [20]:

```
# Train the model
model.fit(X_train, Y_train, epochs=200, batch_size=700, verbose=2)
```

```
Epoch 1/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 7ms/epoch - 7ms/step
Epoch 2/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 4ms/epoch - 4ms/step
Epoch 3/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 4ms/epoch - 4ms/step
Epoch 4/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 5ms/epoch - 5ms/step
Epoch 5/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 4ms/epoch - 4ms/step
Epoch 6/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 5ms/epoch - 5ms/step
Epoch 7/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 4ms/epoch - 4ms/step
Epoch 8/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 3ms/epoch - 3ms/step
Epoch 9/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 5ms/epoch - 5ms/step
Epoch 10/200
1/1 - 0s - loss: -4.1268e+01 - accuracy: 0.0000e+00 - 4ms/epoch - 4ms/step
```

In [28]:

```
# Evaluate the model on the test set
test_loss, test_acc2 = model.evaluate(X_test, Y_test, verbose=2)
print("Test accuracy with sigmoid activation:", test_acc1)
print("Test accuracy with ReLU activation:", test_acc2)
```

```
7/7 - 0s - loss: -4.1114e+01 - accuracy: 0.0000e+00 - 39ms/epoch - 6ms/step
Test accuracy with sigmoid activation: 0.0
Test accuracy with ReLU activation: 0.0
```