Regression using ANN

Dataset: Graduates Admission Prediction

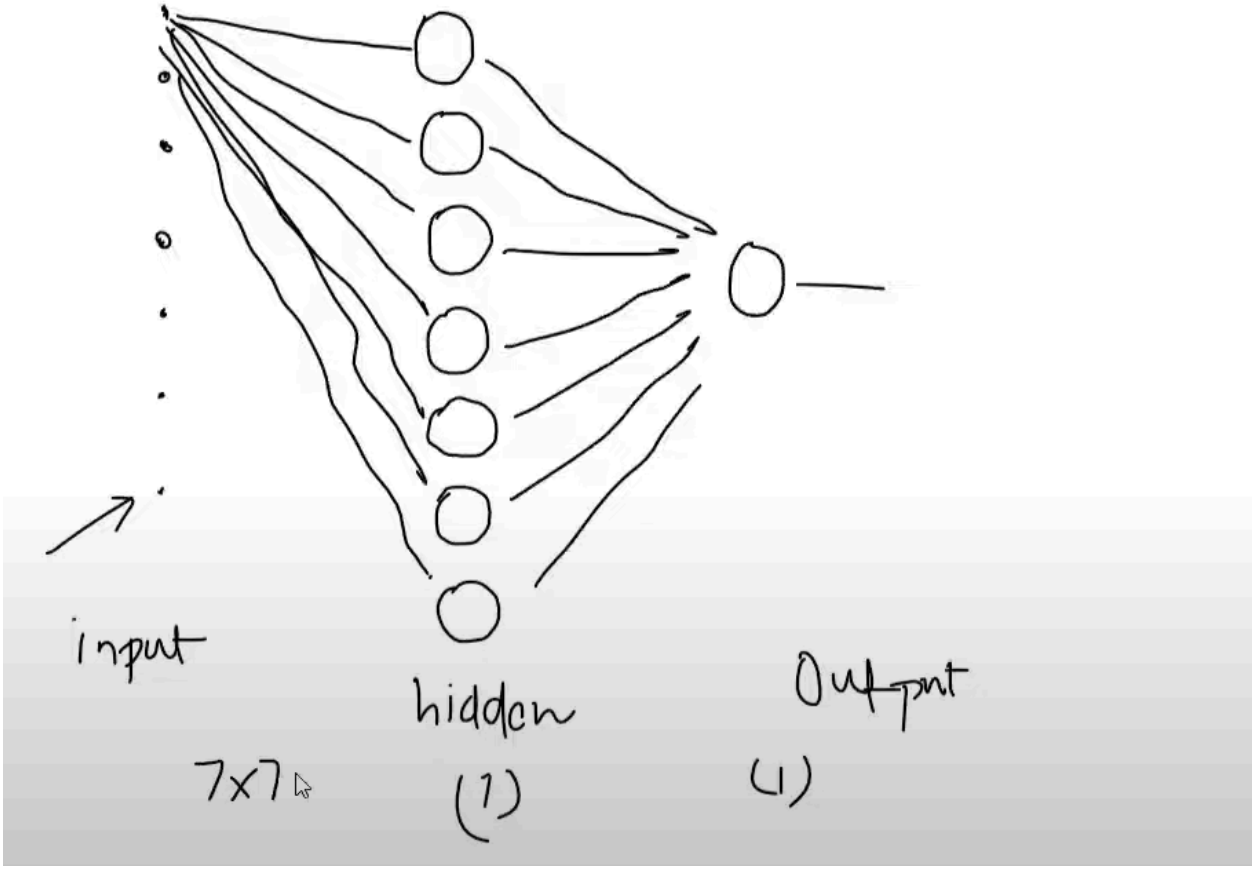
Problem: Regression

```
df = pd.read_csv(r'https://raw.githubusercontent.com/G1Codes/Datasets/refs/heads/main/Graduates%20Admission%20Prediction.csv')

df.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

1. GRE Scores (out of 340)
2. TOEFL Scores (out of 120)
3. University Rating (out of 5)
4. Statement of Purpose -(SOP) Strength (out of 5)
5. Letter of Recommendation-(LOR) Strength (out of 5)
6. Undergraduate GPA-CGPA (out of 10)
7. Research Experience (either 0 or 1)
8. Chance of Admit (ranging from 0 to 1)




```
df.shape
```

```
(500, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          500 non-null    int64
 1   TOEFL Score        500 non-null    int64
 2   University Rating  500 non-null    int64
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

Define X & y:

```
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

train_test_split:

```
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=1)
```

Scale the data

- We'll use `MinMaxScaler` because we know the upper and lower bound.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Import Libraries:

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

Create Model:

```
model = Sequential()

model.add(Dense(7,activation='relu',input_dim=7))
model.add(Dense(7,activation='relu'))
model.add(Dense(1,activation='linear'))
```

💡 **For regression →** `activation='linear'`

```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	56
dense_1 (Dense)	(None, 7)	56
dense_2 (Dense)	(None, 1)	8

```
Total params: 120 (480.00 B)

Trainable params: 120 (480.00 B)
```

Compile

```
model.compile(loss='mean_squared_error',optimizer='Adam')
```

Fit the model:

```
history = model.fit(X_train_scaled,y_train,epochs=100,validation_split=0.2)
```

```
history = model.fit(X_train_scaled,y_train,epochs=100,validation_split=0.2)
✓ 13.0s

Epoch 1/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.2398 - val_loss: 0.1999
Epoch 2/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1746 - val_loss: 0.1349
Epoch 3/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1136 - val_loss: 0.0834
Epoch 4/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0752 - val_loss: 0.0492
Epoch 5/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0441 - val_loss: 0.0329
Epoch 6/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0321 - val_loss: 0.0259
Epoch 7/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0266 - val_loss: 0.0211
Epoch 8/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0201 - val_loss: 0.0173
Epoch 9/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0168 - val_loss: 0.0145
Epoch 10/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0168 - val_loss: 0.0123
Epoch 11/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0125 - val_loss: 0.0109
Epoch 12/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0135 - val_loss: 0.0099
Epoch 13/100
...
Epoch 99/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0042 - val_loss: 0.0044
Epoch 100/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0037 - val_loss: 0.0045
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Predict:

```
y_pred = model.predict(X_test_scaled)
```

Calculate the r2 score:

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.8031579622886069
```

Plot Graph

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```