Chapter 3: ANN

Ex1: Cars - Linear activation function

- Cho dữ liệu cars.csv
- Áp dụng ANN để dự đoán giá trị của một "potential car sale"

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS8_DeepLearning/Practice/Chapter3/'
```

In [3]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
import tensorflow as tf
from tensorflow import keras
```

In [5]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor
```

In [6]:
```
print(tf.__version__)
print(keras.__version__)
```

```
2.5.0
2.5.0
```

In [7]:
```
# Load Dataset
#Variables
dataset=pd.read_csv("cars.csv", delimiter=",", header=None)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 963 entries, 0 to 962
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       963 non-null    int64
 1   1       963 non-null    int64
 2   2       963 non-null    int64
 3   3       963 non-null    int64
 4   4       963 non-null    int64
 5   5       963 non-null    int64
dtypes: int64(6)
memory usage: 45.3 KB
```

In [8]:
```
dataset.head()
```

Out[8]:

	0	1	2	3	4	5
0	28	0	23	0	4099	620
1	26	0	27	0	2677	1792
2	30	1	58	41576	6215	27754
3	26	1	25	43172	7626	28256
4	20	1	17	6979	8071	4438

In [9]:
```
dataset.describe()
```

Out[9]:

	0	1	2	3	4	5
count	963.000000	963.000000	963.000000	963.000000	963.000000	963.000000
mean	37.971963	0.512980	27.704050	14109.004154	6176.047767	11689.860852
std	12.290838	0.500091	13.378181	18273.702481	3260.670142	8986.896921
min	19.000000	0.000000	10.000000	0.000000	0.000000	500.000000
25%	27.000000	0.000000	20.000000	1475.000000	3506.500000	3554.000000
50%	37.000000	1.000000	25.000000	6236.000000	6360.000000	9130.000000
75%	49.000000	1.000000	32.000000	16686.000000	8649.500000	19245.000000
max	60.000000	1.000000	97.000000	59770.000000	11970.000000	29926.000000

In [10]:
```
x=dataset.iloc[:,0:5]
y=dataset.iloc[:,5]
print(type(x))
print(type(y))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [11]:
```
y=np.reshape(np.array(y), (-1,1))

scaler_x = MinMaxScaler()
scaler_y = MinMaxScaler()
# print(scaler.fit(x))
# print(scaler.fit(y))
```

In [12]:
```
xscale=scaler_x.fit_transform(x)
yscale=scaler_y.fit_transform(y)
#print(xscale)
#print(yscale)
```

In [13]:
```
# Keras Model Configuration: Neural Network API
model = Sequential()
model.add(Dense(3, input_dim=5, kernel_initializer='normal', activation='relu'))
# model.add(Dense(12, input_dim=5, kernel_initializer='normal', activation='relu
# tham số đầu tiên là units: là số nguyên, nodes của đầu ra
model.add(Dense(3, activation='relu'))
model.add(Dense(1, activation='linear'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 18
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 12
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 4
=================================================================
Total params: 34
Trainable params: 34
Non-trainable params: 0
_________________________________________________________________
```

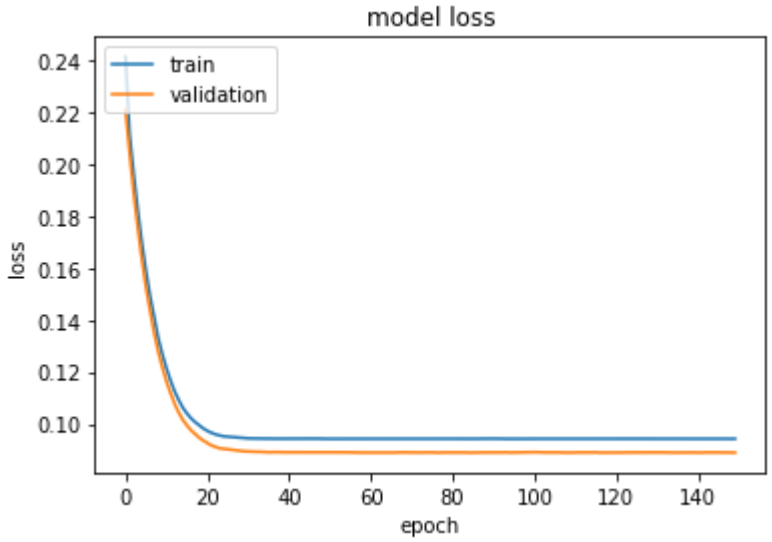
In [14]:
```
model.compile(loss='mse', optimizer='adam', metrics=['mse','mae'])
```

In [15]:
```
# fit model
history = model.fit(xscale, yscale, epochs=150,
                    batch_size=32,
                    verbose=1,
                    validation_split=0.2)
```

```
Epoch 1/150
25/25 [==============================] - 0s 6ms/step - loss: 0.2415 - mse: 0.
2415 - mae: 0.3786 - val_loss: 0.2211 - val_mse: 0.2211 - val_mae: 0.3642
Epoch 2/150
25/25 [==============================] - 0s 2ms/step - loss: 0.2118 - mse: 0.
2118 - mae: 0.3469 - val_loss: 0.2044 - val_mse: 0.2044 - val_mae: 0.3430
Epoch 3/150
25/25 [==============================] - 0s 2ms/step - loss: 0.1965 - mse: 0.
1965 - mae: 0.3292 - val_loss: 0.1890 - val_mse: 0.1890 - val_mae: 0.3246
Epoch 4/150
25/25 [==============================] - 0s 2ms/step - loss: 0.1825 - mse: 0.
1825 - mae: 0.3144 - val_loss: 0.1756 - val_mse: 0.1756 - val_mae: 0.3099
Epoch 5/150
25/25 [==============================] - 0s 2ms/step - loss: 0.1700 - mse: 0.
1700 - mae: 0.3028 - val_loss: 0.1635 - val_mse: 0.1635 - val_mae: 0.2969
Epoch 6/150
25/25 [==============================] - 0s 2ms/step - loss: 0.1591 - mse: 0.
1591 - mae: 0.2938 - val_loss: 0.1527 - val_mse: 0.1527 - val_mae: 0.2856
Epoch 7/150
```

In [16]:
```
print(history.history.keys())
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'mse', 'mae', 'val_loss', 'val_mse', 'val_mae'])
```



In [17]:
```
# Evaluate the model on the test data using `evaluate`
print('\n# Evaluate on test data')
results = model.evaluate(xscale, yscale)
print('loss, mse, mae:', results)
```

```
# Evaluate on test data
31/31 [==============================] - 0s 831us/step - loss: 0.0932 - mse: 0.
0932 - mae: 0.2699
loss, mse, mae: [0.09317748993635178, 0.09317748993635178, 0.2698577344417572]
```

In [20]:
```
# Predictions
Xnew = np.array([[40, 0, 26, 9000, 8000]])
X_new = scaler_x.transform(Xnew)
ynew=model.predict(X_new)
y_new = scaler_y.inverse_transform(ynew)
print("X=%s, Predicted=%s" % (Xnew[0], y_new[0]))
```

```
X=[  40    0   26 9000 8000], Predicted=[11720.976]
```

In [ ]:
```

```