ARTIFICIAL NEURAL NETWORKS – CAR SALES PREDICTION

Student:

SAI KUMAR REDDY NOSSAM

05/02/2022

Course:

MACHINE LEARNING AND DEEP LEARNING

Under the Guidance:

Dr. RUOMING JIN

***Abstract*—** This project is used to anticipate a customer's purchasing power based on their characteristics. Study consumer behavior by predicting how much they are willing to spend based on customer characteristics such as age, credit card debt, annual salary, and net worth. Artificial neural networks will be used to complete a regression task in this project.

I. INTRODUCTION

The purpose of this project is to develop a model that predicts the total amount that customers are willing to spend based on the following characteristics:

- Customer Name & e-mail.
- Age
- Gender
- Country
- Credit Car Debt
- Annual Salary
- Net Worth

The first question is whether this is a classification problem or a regression problem.
Answer: Because predicting the purchasing amount of a car is a continuous variable, this is a regression problem.

II. Motivation

I went to a bike dealership two years ago to buy a bike. The salesman questioned me about my annual earnings, credit card debt, and other details. The salesperson showed me the motorcycles that fit my budget after I answered all the questions. So, I've come up with a reason why I shouldn't perform this project of forecasting vehicle pricing based on client characteristics. That is why I chose to take on this initiative.

III. WHAT IS REGRESSION?

Regression predicts the value of one variable, based on the value of another variable. The independent variable is X, whereas the dependent variable is y.

$$y = b + mX$$

X: Independent Variable

y: Dependent variable

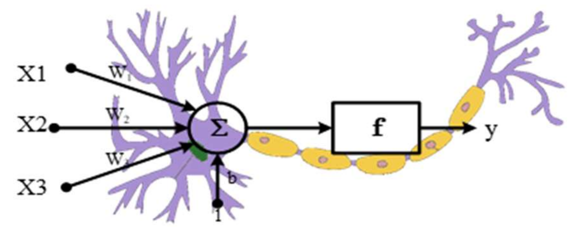
m: Slope of Line

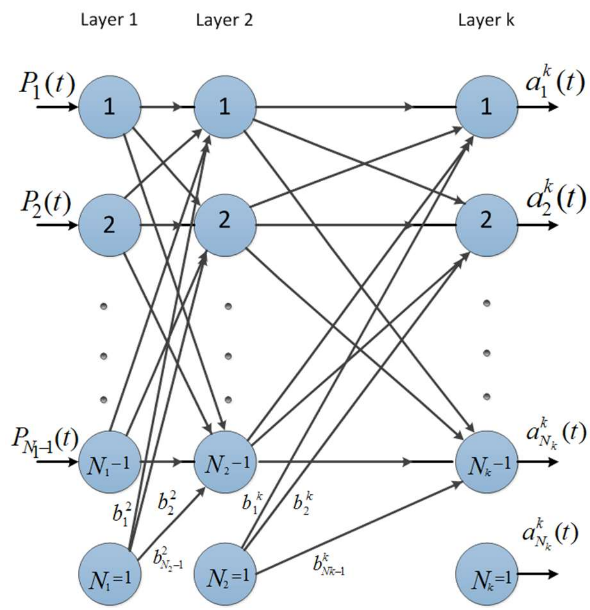
b: y Intercept

We have a regression model after the coefficients 'm' and 'b' have been computed. This trained model can then be used to predict the amount a customer will spend on a car based on the costumer's attributes.

IV. ARTIFICIAL NEURAL NETWORK

The more than 100 billion neurons in the brain communicate through electrical and chemical impulses. Neurons communicate with each other and help us see, think, and come up with new ideas. The human brain learns by forming connections between its neurons. The information processing model inspired by the human brain is called ANNs.



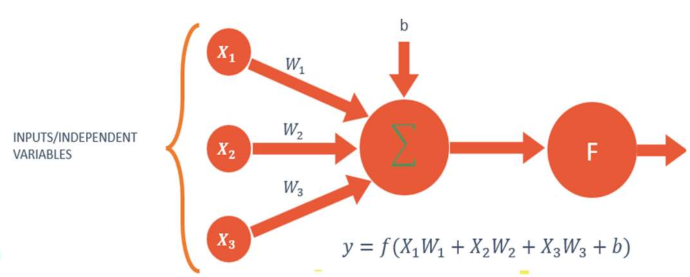


V. NEURON MATHEMTICAL MODEL

Neurons take signals from dendrites, analyze the information in their nuclei, and then form outputs in the axon, which is a long branch.

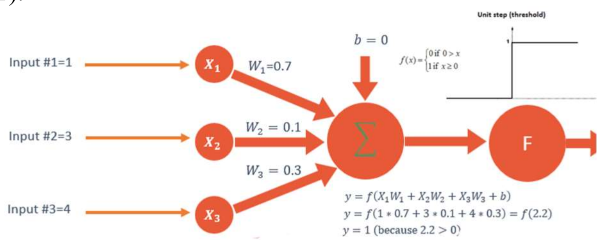
Example:

- The activation function curve can be shifted up or down using the bias parameter.
- The number of parameters that can be adjusted is four (3 weights and 1 bias)
- Activation function "F".



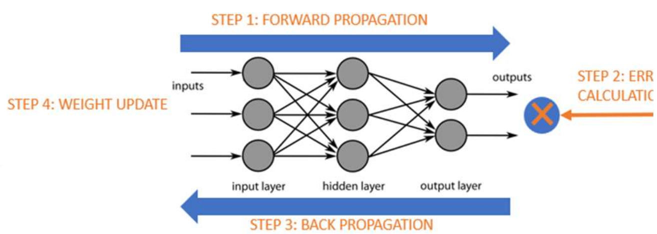
VI. SINGLE NEURON MODEL IN ACTION

- Assume the activation function is of the type of Unit Step Activation Function.
- The input is mapped between the activation functions (0, 1).



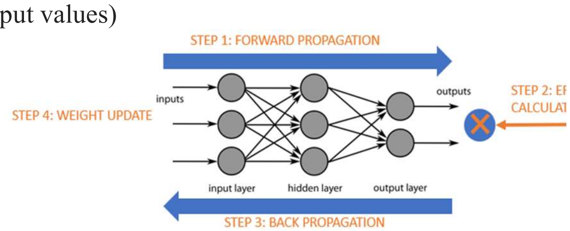
VII. NETWORK TRAINING: BACK PROPAGATION

- Backpropagation is a technique for training artificial neural networks that involves computing the gradients needed to update the weights of the network.
- Gradient descent optimization strategies are widely used to change the weights of neurons by computing the gradient of the loss function.



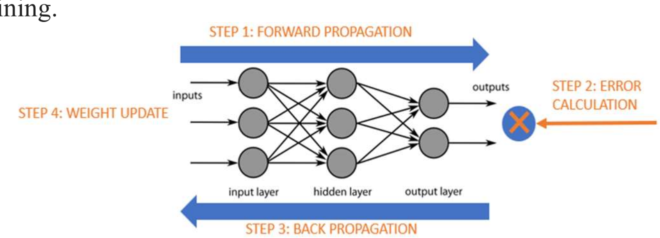
Back propagation Phase1: propagation

- Forward propagation through the network to produce the output value (s)
- The cost is calculated (error term)
- In order to create the deltas, the output activations are propagated back through the network using the training pattern target (difference between targeted and actual output values)



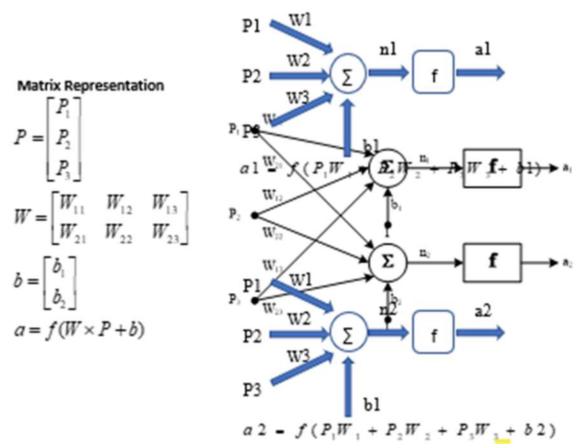
Phase 2: weight update

- Determine the weight gradient.
- The weight is removed by a ratio (%) of the weight's gradient.
- The learning rate is a ratio that determines the pace and quality of learning. The higher the ratio, the quicker the neuron trains; yet the lower the ratio, the more precise the training.



VIII. TWO NEURON MODEL: MATRIX REPRESENTATION

- A matrix of weights, inputs, and outputs represents the network.
- Total amount of parameters that may be adjusted = 8:
  - Weights = 6
  - Biases = 2



IX. DETAILED DATASET DESCRIPTION USED IN THE PROJECT

The Dataset contains the following fields:

1. Customer Name
2. Customer e-mail
3. Country
4. Gender
5. Age
6. Annual Salary
7. Credit Card Debt
8. Net Worth
9. Car Purchase Amount

The Dataset contains 500 Rows. The Data Types of the above fields:

```
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Customer Name        500 non-null    object 
 1   Customer e-mail      500 non-null    object 
 2   Country              500 non-null    object 
 3   Gender               500 non-null    int64  
 4   Age                  500 non-null    float64
 5   Annual Salary        500 non-null    float64
 6   Credit Card Debt     500 non-null    float64
 7   Net Worth            500 non-null    float64
 8   Car Purchase Amount  500 non-null    float64
dtypes: float64(5), int64(1), object(3)
```

X. FUTURE WORK AND THOUGHTS

In the future, a web version of this project should be created. Customers will be able to enter all their information and search for suitable vehicles within their budget. In the future, the initiative should be able to estimate how much money a bank might lend to consumers who want to buy a car.

REFERENCES

1. C. Chupong and B. Plangklang, "Comparison Study on Artificial Neural Network and Online Sequential Extreme Learning Machine in Regression Problem," 2019 7th International Electrical Engineering Congress (iEECON), 2019, pp. 1-4, doi: 10.1109/iEECON45304.2019.8938990.

2. Y. Celik, S. Guney and B. Dengiz, "Obesity Level Estimation based on Machine Learning Methods and Artificial Neural Networks," 2021 44th International Conference on Telecommunications and Signal Processing (TSP), 2021, pp. 329-332, doi: 10.1109/TSP52935.2021.9522628.

3. A. M. Trunin, A. N. Ragozin and S. N. Darovskih, "An Investigation of the Application of an Artificial Neural Network and Machine Learning to Improve the Efficiency of Gas Analyzer Systems in Assessing the State of the Environment," 2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2021, pp. 571-575, doi: 10.1109/ICIEAM51226.2021.9446406.

4. M. Soykan and P. S. Bölük, "Tor Network Detection By Using Machine Learning And Artificial Neural Network," 2021 International Symposium on Networks, Computers and Communications (ISNCC), 2021, pp. 1-4, doi: 10.1109/ISNCC52172.2021.9615730.

5. B. Alić, L. Gurbeta and A. Badnjević, "Machine learning techniques for classification of diabetes and cardiovascular diseases," 2017 6th Mediterranean Conference on Embedded Computing (MECO), 2017, pp. 1-4, doi: 10.1109/MECO.2017.7977152.

6. S. Ravikumar and P. Saraf, "Prediction of Stock Prices using Machine Learning (Regression, Classification) Algorithms," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-5, doi: 10.1109/INCET49848.2020.9154061.

7. H. A. Mesrabadi and K. Faez, "Improving early prostate cancer diagnosis by using Artificial Neural Networks and Deep Learning," 2018 4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), 2018, pp. 39-42, doi: 10.1109/ICSPIS.2018.8700542.

8. T. Thomas, N. Pradhan and V. S. Dhaka, "Comparative Analysis to Predict Breast Cancer using Machine Learning Algorithms: A Survey," 2020 International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 192-196, doi: 10.1109/ICICT48043.2020.9112464.

9. M. Susanty, Sahrul, A. F. Rahman, M. D. Normansyah and A. Irawan, "Offensive Language Detection using Artificial Neural Network," 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT), 2019, pp. 350-353, doi: 10.1109/ICAIIT.2019.8834452.

10. Yu-Xue Sun and Guang-Hui Guo, "Application of artificial neural network on prediction reservoir sensitivity," 2005 International Conference on Machine Learning and Cybernetics, 2005, pp. 4770-4773 Vol. 8, doi: 10.1109/ICMLC.2005.1527781.

11. Yih-Fang Huang, "Artificial neural networks-learning and generalization," Proceedings of APCCAS'94 - 1994 Asia Pacific Conference on Circuits and Systems, 1994, pp. 162-, doi: 10.1109/APCCAS.1994.514542.

12. K. T. Islam, G. Mujtaba, R. G. Raj and H. F. Nweke, "Handwritten digits recognition with artificial neural network," 2017 International Conference on Engineering Technology and Technopreneurship (ICE2T), 2017, pp. 1-4, doi: 10.1109/ICE2T.2017.8215993.

13. X. -F. Gu, L. Liu, J. -P. Li, Y. -Y. Huang and J. Lin, "Data Classification based on Artificial Neural Networks," 2008 International Conference on Apperceiving Computing and Intelligence Analysis, 2008, pp. 223-226, doi: 10.1109/ICACIA.2008.4770010.

ARTIFICIAL NEURAL NETWORKS – CAR SALES PREDICTION

May 8, 2022

1 Implementation in Python

2 ARTIFICIAL NEURAL NETWORKS – CAR SALES PREDICTION

3 Step 1: Import Libraries

[1]:
```
#Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

4 Step 2: Load Dataset

[2]:
```
#Load Dataset
Car_Purchasing_Data = pd.read_csv('C:
↪\\Users\\saiku\\OneDrive\\Desktop\\P74-Project-1\\Car_Purchasing_Data.
↪csv',encoding='ISO-8859-1')
```

[3]:
```
Car_Purchasing_Data
```

[3]:
```
        Customer Name                                    Customer e-mail  \
0       Martina Avila  cubilia.Curae.Phasellus@quisaccumsanconvallis.edu
1       Harlan Barnes                                 eu.dolor@diam.co.uk
2     Naomi Rodriquez  vulputate.mauris.sagittis@ametconsectetueradip…
3     Jade Cunningham                            malesuada@dignissim.com
4         Cedric Leach     felis.ullamcorper.viverra@egetmollislectus.net
..                …                                                  …
495            Walter                              ligula@Cumsociis.ca
496             Vanna                     Cum.sociis.natoque@Sedmolestie.edu
497             Pearl                        penatibus.et@massanonante.com
498              Nell                       Quisque.varius@arcuVivamussit.net
499             Marla                           Camaron.marla@hotmail.com
```

```
          Country  Gender        Age  Annual Salary  Credit Card Debt  \
0        Bulgaria       0  41.851720    62812.09301      11609.380910
1          Belize       0  40.870623    66646.89292       9572.957136
2         Algeria       1  43.152897    53798.55112      11160.355060
3    Cook Islands       1  58.271369    79370.03798      14426.164850
4          Brazil       1  57.313749    59729.15130       5358.712177
..            …       …        …            …               …
495         Nepal       0  41.462515    71942.40291       6995.902524
496      Zimbabwe       1  37.642000    56039.49793      12301.456790
497   Philippines       1  53.943497    68888.77805      10611.606860
498      Botswana       1  59.160509    49811.99062      14013.034510
499        marlal       1  46.731152    61370.67766       9391.341628

       Net Worth  Car Purchase Amount
0    238961.2505          35321.45877
1    530973.9078          45115.52566
2    638467.1773          42925.70921
3    548599.0524          67422.36313
4    560304.0671          55915.46248
..           …                  …
495  541670.1016          48901.44342
496  360419.0988          31491.41457
497  764531.3203          64147.28888
498  337826.6382          45442.15353
499  462946.4924          45107.22566

[500 rows x 9 columns]
```

[4]:
```python
Car_Purchasing_Data.head()
```

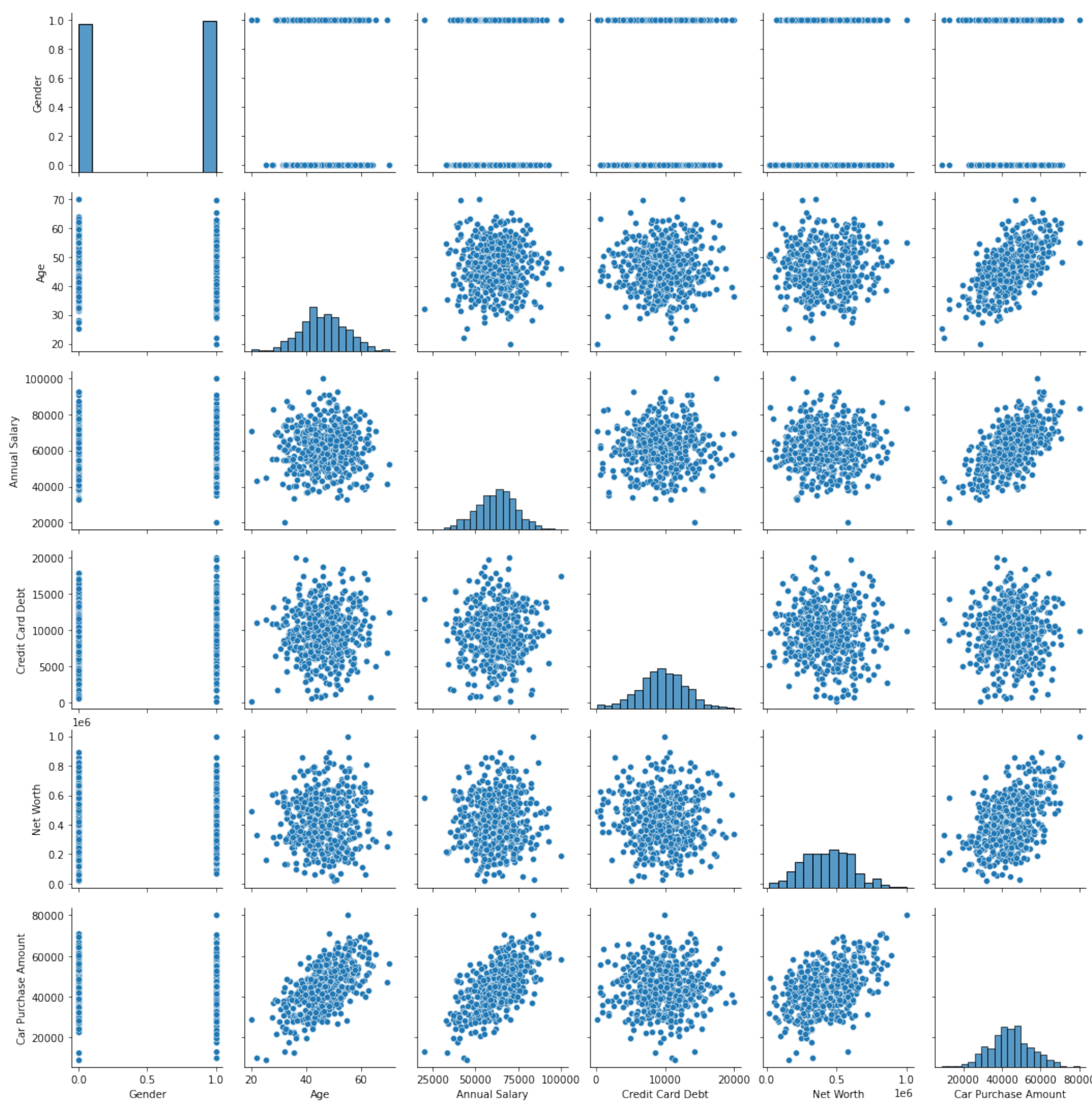
[4]:
```
     Customer Name                                    Customer e-mail  \
0    Martina Avila  cubilia.Curae.Phasellus@quisaccumsanconvallis.edu
1    Harlan Barnes                                eu.dolor@diam.co.uk
2  Naomi Rodriquez  vulputate.mauris.sagittis@ametconsectetueradip…
3  Jade Cunningham                            malesuada@dignissim.com
4     Cedric Leach     felis.ullamcorper.viverra@egetmollislectus.net

        Country  Gender        Age  Annual Salary  Credit Card Debt  \
0      Bulgaria       0  41.851720    62812.09301      11609.380910
1        Belize       0  40.870623    66646.89292       9572.957136
2       Algeria       1  43.152897    53798.55112      11160.355060
3  Cook Islands       1  58.271369    79370.03798      14426.164850
4        Brazil       1  57.313749    59729.15130       5358.712177

     Net Worth  Car Purchase Amount
0  238961.2505          35321.45877
```

1	530973.9078	45115.52566
2	638467.1773	42925.70921
3	548599.0524	67422.36313
4	560304.0671	55915.46248

5 Step 3: Visualize dataset

[5]:
```python
#Visualize dataset
sns.pairplot(Car_Purchasing_Data)
```

[5]: <seaborn.axisgrid.PairGrid at 0x2980eda6730>



6 Step 4: Data Cleaning

[6]:
```
#Data Cleaning

#Dropping Customer Name, Customer e-mail, and Country -> Because, the Car␣
↪Purchasing ability of a customer wouldn't depend on these characteristics
#Dropping Car Purchase Amount as it is our output
#X is for the input

X=Car_Purchasing_Data.drop(['Customer Name','Customer e-mail','Country','Car␣
↪Purchase Amount'], axis=1)
```

[7]:
```
X
```

[7]:
```
     Gender        Age  Annual Salary  Credit Card Debt    Net Worth
0         0  41.851720    62812.09301      11609.380910  238961.2505
1         0  40.870623    66646.89292       9572.957136  530973.9078
2         1  43.152897    53798.55112      11160.355060  638467.1773
3         1  58.271369    79370.03798      14426.164850  548599.0524
4         1  57.313749    59729.15130       5358.712177  560304.0671
..      …          …            …                 …            …
495       0  41.462515    71942.40291       6995.902524  541670.1016
496       1  37.642000    56039.49793      12301.456790  360419.0988
497       1  53.943497    68888.77805      10611.606860  764531.3203
498       1  59.160509    49811.99062      14013.034510  337826.6382
499       1  46.731152    61370.67766       9391.341628  462946.4924

[500 rows x 5 columns]
```

[8]:
```
#y is for the output
y=Car_Purchasing_Data['Car Purchase Amount']
```

[9]:
```
y
```

[9]:
```
0      35321.45877
1      45115.52566
2      42925.70921
3      67422.36313
4      55915.46248
          …
495    48901.44342
496    31491.41457
497    64147.28888
498    45442.15353
499    45107.22566
Name: Car Purchase Amount, Length: 500, dtype: float64
```

[10]:
```python
#Normalizing the Data
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaled_X=scaler.fit_transform(X)
```

[11]:
```python
scaled_X
```

[11]:
```
array([[0.        , 0.4370344 , 0.53515116, 0.57836085, 0.22342985],
       [0.        , 0.41741247, 0.58308616, 0.476028  , 0.52140195],
       [1.        , 0.46305795, 0.42248189, 0.55579674, 0.63108896],
       …,
       [1.        , 0.67886994, 0.61110973, 0.52822145, 0.75972584],
       [1.        , 0.78321017, 0.37264988, 0.69914746, 0.3243129 ],
       [1.        , 0.53462305, 0.51713347, 0.46690159, 0.45198622]])
```

[12]:
```python
scaler.data_max_
```

[12]:
```
array([1.e+00, 7.e+01, 1.e+05, 2.e+04, 1.e+06])
```

[13]:
```python
scaler.data_min_
```

[13]:
```
array([    0.,    20., 20000.,   100., 20000.])
```

[14]:
```python
y=y.values.reshape(-1,1)
```

[15]:
```python
y_scaled=scaler.fit_transform(y)
```

7 Step 5: Training the Model

[16]:
```python
scaled_X.shape
```

[16]:
```
(500, 5)
```

[17]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(scaled_X,y_scaled)
```

[18]:
```python
X_train.shape
```

[18]:
```
(375, 5)
```

[19]:
```python
X_test.shape
```

[19]:
```
(125, 5)
```

[20]:
```python
#Building Artificial Neural Network in Sequential Form
import tensorflow.keras
```

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

model=Sequential()
model.add(Dense(25, input_dim=5,activation='relu'))
model.add(Dense(25,activation='relu'))
model.add(Dense(1, activation='linear'))
```

[21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 25)                150
_________________________________________________________________
dense_1 (Dense)              (None, 25)                650
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 26
=================================================================
Total params: 826
Trainable params: 826
Non-trainable params: 0
_________________________________________________________________
```

[22]:
```
model.compile(optimizer='adam',loss='mean_squared_error')
```

[23]:
```
epochs_hist=model.fit(X_train, y_train, epochs=100,batch_size=25, verbose =1,
 ↪validation_split=0.2)
```

```
Epoch 1/100
12/12 [==============================] - 1s 20ms/step - loss: 0.1310 - val_loss:
0.1150
Epoch 2/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0639 - val_loss:
0.0503
Epoch 3/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0296 - val_loss:
0.0225
Epoch 4/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss:
0.0162
Epoch 5/100
12/12 [==============================] - 0s 5ms/step - loss: 0.0154 - val_loss:
0.0152
Epoch 6/100
12/12 [==============================] - ETA: 0s - loss: 0.013 - 0s 4ms/step -
```

```
loss: 0.0127 - val_loss: 0.0141
Epoch 7/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0113 - val_loss:
0.0119
Epoch 8/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0096 - val_loss:
0.0100
Epoch 9/100
12/12 [==============================] - 0s 5ms/step - loss: 0.0082 - val_loss:
0.0086
Epoch 10/100
12/12 [==============================] - 0s 5ms/step - loss: 0.0069 - val_loss:
0.0070
Epoch 11/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0056 - val_loss:
0.0057
Epoch 12/100
12/12 [==============================] - 0s 5ms/step - loss: 0.0044 - val_loss:
0.0048
Epoch 13/100
12/12 [==============================] - ETA: 0s - loss: 0.002 - 0s 4ms/step -
loss: 0.0036 - val_loss: 0.0041
Epoch 14/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0030 - val_loss:
0.0033
Epoch 15/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0025 - val_loss:
0.0031
Epoch 16/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0024 - val_loss:
0.0029
Epoch 17/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0022 - val_loss:
0.0027
Epoch 18/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0021 - val_loss:
0.0025
Epoch 19/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0019 - val_loss:
0.0024
Epoch 20/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0018 - val_loss:
0.0023
Epoch 21/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0017 - val_loss:
0.0021
Epoch 22/100
12/12 [==============================] - 0s 5ms/step - loss: 0.0016 - val_loss:
```

```
0.0021
Epoch 23/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0015 - val_loss:
0.0019
Epoch 24/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0014 - val_loss:
0.0019
Epoch 25/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0013 - val_loss:
0.0018
Epoch 26/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0012 - val_loss:
0.0017
Epoch 27/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0012 - val_loss:
0.0016
Epoch 28/100
12/12 [==============================] - 0s 4ms/step - loss: 0.0011 - val_loss:
0.0016
Epoch 29/100
12/12 [==============================] - 0s 5ms/step - loss: 0.0010 - val_loss:
0.0015
Epoch 30/100
12/12 [==============================] - 0s 4ms/step - loss: 9.7060e-04 -
val_loss: 0.0014
Epoch 31/100
12/12 [==============================] - 0s 5ms/step - loss: 9.0256e-04 -
val_loss: 0.0014
Epoch 32/100
12/12 [==============================] - 0s 4ms/step - loss: 8.3891e-04 -
val_loss: 0.0013
Epoch 33/100
12/12 [==============================] - 0s 4ms/step - loss: 7.9175e-04 -
val_loss: 0.0012
Epoch 34/100
12/12 [==============================] - 0s 4ms/step - loss: 7.3941e-04 -
val_loss: 0.0012
Epoch 35/100
12/12 [==============================] - 0s 4ms/step - loss: 6.9982e-04 -
val_loss: 0.0011
Epoch 36/100
12/12 [==============================] - 0s 4ms/step - loss: 6.3926e-04 -
val_loss: 0.0011
Epoch 37/100
12/12 [==============================] - 0s 4ms/step - loss: 5.8662e-04 -
val_loss: 9.4975e-04
Epoch 38/100
12/12 [==============================] - 0s 4ms/step - loss: 5.4724e-04 -
```

```
val_loss: 9.1071e-04
Epoch 39/100
12/12 [==============================] - 0s 5ms/step - loss: 5.1061e-04 -
val_loss: 8.5432e-04
Epoch 40/100
12/12 [==============================] - 0s 4ms/step - loss: 4.7122e-04 -
val_loss: 7.9405e-04
Epoch 41/100
12/12 [==============================] - 0s 5ms/step - loss: 4.2317e-04 -
val_loss: 7.5594e-04
Epoch 42/100
12/12 [==============================] - 0s 4ms/step - loss: 3.9148e-04 -
val_loss: 7.0011e-04
Epoch 43/100
12/12 [==============================] - 0s 4ms/step - loss: 3.8155e-04 -
val_loss: 6.7911e-04
Epoch 44/100
12/12 [==============================] - 0s 4ms/step - loss: 3.5142e-04 -
val_loss: 6.2109e-04
Epoch 45/100
12/12 [==============================] - 0s 3ms/step - loss: 2.9338e-04 -
val_loss: 5.8986e-04
Epoch 46/100
12/12 [==============================] - 0s 4ms/step - loss: 2.7886e-04 -
val_loss: 5.6591e-04
Epoch 47/100
12/12 [==============================] - 0s 4ms/step - loss: 2.5440e-04 -
val_loss: 5.1809e-04
Epoch 48/100
12/12 [==============================] - 0s 4ms/step - loss: 2.3750e-04 -
val_loss: 4.8724e-04
Epoch 49/100
12/12 [==============================] - 0s 4ms/step - loss: 2.2729e-04 -
val_loss: 4.6482e-04
Epoch 50/100
12/12 [==============================] - 0s 4ms/step - loss: 1.9680e-04 -
val_loss: 4.4012e-04
Epoch 51/100
12/12 [==============================] - 0s 4ms/step - loss: 1.8315e-04 -
val_loss: 4.0284e-04
Epoch 52/100
12/12 [==============================] - 0s 4ms/step - loss: 1.7291e-04 -
val_loss: 4.1777e-04
Epoch 53/100
12/12 [==============================] - 0s 4ms/step - loss: 1.8807e-04 -
val_loss: 3.8933e-04
Epoch 54/100
12/12 [==============================] - 0s 4ms/step - loss: 1.5006e-04 -
```

```
val_loss: 3.4955e-04
Epoch 55/100
12/12 [==============================] - 0s 5ms/step - loss: 1.4077e-04 -
val_loss: 3.3669e-04
Epoch 56/100
12/12 [==============================] - 0s 4ms/step - loss: 1.2751e-04 -
val_loss: 3.1666e-04
Epoch 57/100
12/12 [==============================] - 0s 4ms/step - loss: 1.2291e-04 -
val_loss: 3.0607e-04
Epoch 58/100
12/12 [==============================] - 0s 4ms/step - loss: 1.1087e-04 -
val_loss: 2.8506e-04
Epoch 59/100
12/12 [==============================] - 0s 4ms/step - loss: 1.0602e-04 -
val_loss: 2.8154e-04
Epoch 60/100
12/12 [==============================] - 0s 4ms/step - loss: 9.6773e-05 -
val_loss: 2.6415e-04
Epoch 61/100
12/12 [==============================] - 0s 4ms/step - loss: 9.2578e-05 -
val_loss: 2.5311e-04
Epoch 62/100
12/12 [==============================] - 0s 4ms/step - loss: 8.6630e-05 -
val_loss: 2.4155e-04
Epoch 63/100
12/12 [==============================] - 0s 4ms/step - loss: 8.2745e-05 -
val_loss: 2.3546e-04
Epoch 64/100
12/12 [==============================] - ETA: 0s - loss: 6.4606e-0 - 0s 4ms/step
- loss: 7.9923e-05 - val_loss: 2.2396e-04
Epoch 65/100
12/12 [==============================] - 0s 4ms/step - loss: 7.7305e-05 -
val_loss: 2.1619e-04
Epoch 66/100
12/12 [==============================] - 0s 5ms/step - loss: 7.2703e-05 -
val_loss: 2.0823e-04
Epoch 67/100
12/12 [==============================] - 0s 4ms/step - loss: 6.7175e-05 -
val_loss: 2.0193e-04
Epoch 68/100
12/12 [==============================] - 0s 5ms/step - loss: 6.4044e-05 -
val_loss: 1.9649e-04
Epoch 69/100
12/12 [==============================] - 0s 4ms/step - loss: 5.9985e-05 -
val_loss: 2.0862e-04
Epoch 70/100
12/12 [==============================] - 0s 5ms/step - loss: 6.3307e-05 -
```

```
val_loss: 1.8541e-04
Epoch 71/100
12/12 [==============================] - 0s 4ms/step - loss: 6.0317e-05 -
val_loss: 1.7783e-04
Epoch 72/100
12/12 [==============================] - 0s 4ms/step - loss: 5.2159e-05 -
val_loss: 1.7228e-04
Epoch 73/100
12/12 [==============================] - 0s 5ms/step - loss: 4.9247e-05 -
val_loss: 1.6835e-04
Epoch 74/100
12/12 [==============================] - 0s 5ms/step - loss: 4.8191e-05 -
val_loss: 1.6240e-04
Epoch 75/100
12/12 [==============================] - 0s 4ms/step - loss: 4.8057e-05 -
val_loss: 1.5844e-04
Epoch 76/100
12/12 [==============================] - 0s 4ms/step - loss: 5.0909e-05 -
val_loss: 1.6030e-04
Epoch 77/100
12/12 [==============================] - 0s 5ms/step - loss: 4.6153e-05 -
val_loss: 1.5057e-04
Epoch 78/100
12/12 [==============================] - 0s 4ms/step - loss: 4.5378e-05 -
val_loss: 1.5047e-04
Epoch 79/100
12/12 [==============================] - 0s 4ms/step - loss: 4.2189e-05 -
val_loss: 1.5889e-04
Epoch 80/100
12/12 [==============================] - 0s 4ms/step - loss: 3.9370e-05 -
val_loss: 1.4492e-04
Epoch 81/100
12/12 [==============================] - 0s 4ms/step - loss: 3.7113e-05 -
val_loss: 1.3656e-04
Epoch 82/100
12/12 [==============================] - 0s 4ms/step - loss: 3.7538e-05 -
val_loss: 1.3082e-04
Epoch 83/100
12/12 [==============================] - 0s 4ms/step - loss: 3.4388e-05 -
val_loss: 1.2760e-04
Epoch 84/100
12/12 [==============================] - 0s 4ms/step - loss: 3.5271e-05 -
val_loss: 1.2748e-04
Epoch 85/100
12/12 [==============================] - 0s 4ms/step - loss: 3.3804e-05 -
val_loss: 1.2387e-04
Epoch 86/100
12/12 [==============================] - 0s 4ms/step - loss: 3.2910e-05 -
```

```
val_loss: 1.2886e-04
Epoch 87/100
12/12 [==============================] - 0s 4ms/step - loss: 3.3410e-05 -
val_loss: 1.1901e-04
Epoch 88/100
12/12 [==============================] - 0s 4ms/step - loss: 3.0973e-05 -
val_loss: 1.1720e-04
Epoch 89/100
12/12 [==============================] - 0s 5ms/step - loss: 3.4701e-05 -
val_loss: 1.1751e-04
Epoch 90/100
12/12 [==============================] - 0s 4ms/step - loss: 2.9246e-05 -
val_loss: 1.1114e-04
Epoch 91/100
12/12 [==============================] - 0s 5ms/step - loss: 3.0648e-05 -
val_loss: 1.1143e-04
Epoch 92/100
12/12 [==============================] - 0s 4ms/step - loss: 3.4208e-05 -
val_loss: 1.0786e-04
Epoch 93/100
12/12 [==============================] - 0s 4ms/step - loss: 2.9210e-05 -
val_loss: 1.0908e-04
Epoch 94/100
12/12 [==============================] - 0s 5ms/step - loss: 2.6781e-05 -
val_loss: 1.0267e-04
Epoch 95/100
12/12 [==============================] - 0s 4ms/step - loss: 2.5226e-05 -
val_loss: 9.8934e-05
Epoch 96/100
12/12 [==============================] - 0s 4ms/step - loss: 2.4091e-05 -
val_loss: 9.8986e-05
Epoch 97/100
12/12 [==============================] - 0s 4ms/step - loss: 2.5505e-05 -
val_loss: 1.0432e-04
Epoch 98/100
12/12 [==============================] - 0s 4ms/step - loss: 2.8381e-05 -
val_loss: 1.0170e-04
Epoch 99/100
12/12 [==============================] - 0s 4ms/step - loss: 3.1544e-05 -
val_loss: 1.0765e-04
Epoch 100/100
12/12 [==============================] - 0s 4ms/step - loss: 2.8311e-05 -
val_loss: 9.1828e-05
```

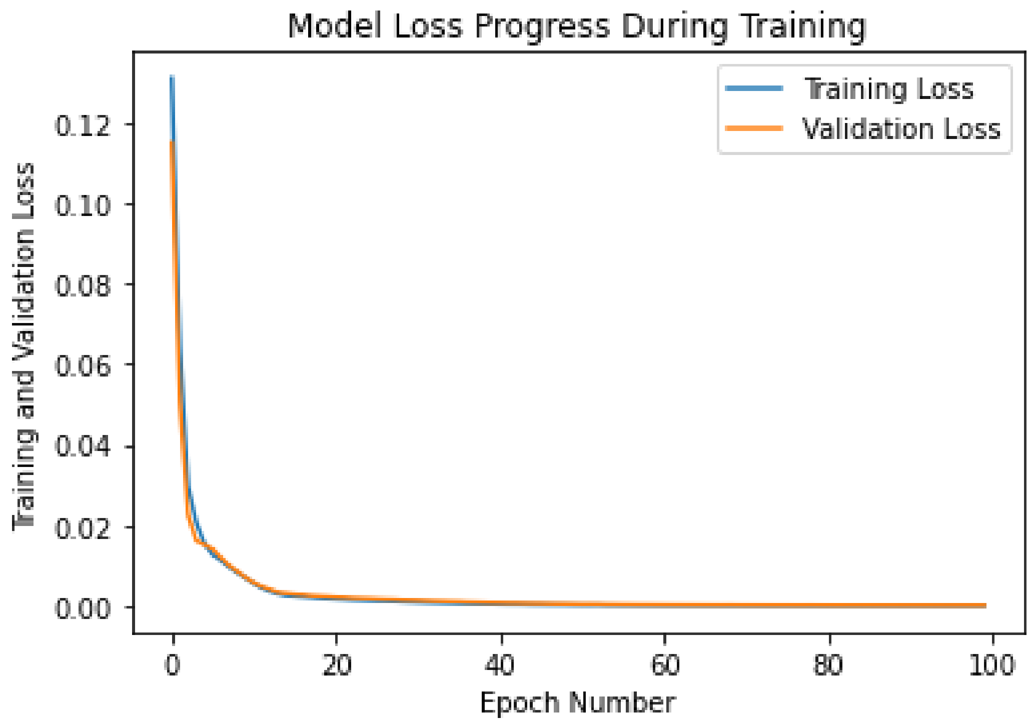
8 Step 6: Evaluating the Model

[24]:
```
epochs_hist.history.keys()
```

[24]:
```
dict_keys(['loss', 'val_loss'])
```

[25]:
```
plt.plot(epochs_hist.history['loss'])
plt.plot(epochs_hist.history['val_loss'])
plt.title('Model Loss Progress During Training')
plt.ylabel('Training and Validation Loss')
plt.xlabel('Epoch Number')
plt.legend(['Training Loss','Validation Loss'])
```

[25]:
```
<matplotlib.legend.Legend at 0x2981a3b3160>
```



[26]:
```
#Gender, age, annual salary, credit card debt, net worth
X_test=np.array([[1,50,40000,5000,500000]])
y_predict=model.predict(X_test)
```

[27]:
```
print('Expected Purchase Amount',y_predict)
```

```
Expected Purchase Amount [[240944.]]
```