Problem Statement:

We are working with a car salesman to develop a model to predict the total dollar amount that customers are willing to pay given the following attributes:

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

The model will predict Car Purchase Amount

This is a regression task, we are predicting a continuous value.

In [1]:

```
# Step 1: Importing libraries and dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv(r'C:\Users\jihad\Desktop\ML\CarSalePrediction\Car_Purchasing_Data.csv'
```

In [3]:

```
df.head()
```

Out[3]:

	Customer Name	Customer e-mail	Country	Gender	Age	Annual Salary
0	Martina Avila	cubilia.Curae.Phasellus@quisaccumsanconvallis.edu	Bulgaria	0	41.851720	62812.09301
1	Harlan Barnes	eu.dolor@diam.co.uk	Belize	0	40.870623	66646.89292
2	Naomi Rodriquez	vulputate.mauris.sagittis@ametconsectetueradip...	Algeria	1	43.152897	53798.55112
3	Jade Cunningham	malesuada@dignissim.com	Cook Islands	1	58.271369	79370.03798
4	Cedric Leach	felis.ullamcorper.viverra@egetmollislectus.net	Brazil	1	57.313749	59729.15130

In [4]:

```
df.describe()
```

Out[4]:

	Gender	Age	Annual Salary	Credit Card Debt	Net Worth	Car Purchase Amount
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000
mean	0.506000	46.241674	62127.239608	9607.645049	431475.713625	44209.799218
std	0.500465	7.978862	11703.378228	3489.187973	173536.756340	10773.178744
min	0.000000	20.000000	20000.000000	100.000000	20000.000000	9000.000000
25%	0.000000	40.949969	54391.977195	7397.515792	299824.195900	37629.896040
50%	1.000000	46.049901	62915.497035	9655.035568	426750.120650	43997.783390
75%	1.000000	51.612263	70117.862005	11798.867487	557324.478725	51254.709517
max	1.000000	70.000000	100000.000000	20000.000000	1000000.000000	80000.000000

In [5]:

```
# We have no missing values.

print(" \n Total NaN in each column in df : \n\n",
      df.isnull().sum())

print(" \n Total NaN values in df : \n\n",
       df.isnull().sum().sum())
```

```
 Total NaN in each column in df :

 Customer Name          0
Customer e-mail        0
Country                0
Gender                 0
Age                    0
Annual Salary          0
Credit Card Debt       0
Net Worth              0
Car Purchase Amount    0
dtype: int64

 Total NaN values in df :

 0
```

In [6]:

```
df.tail()
```

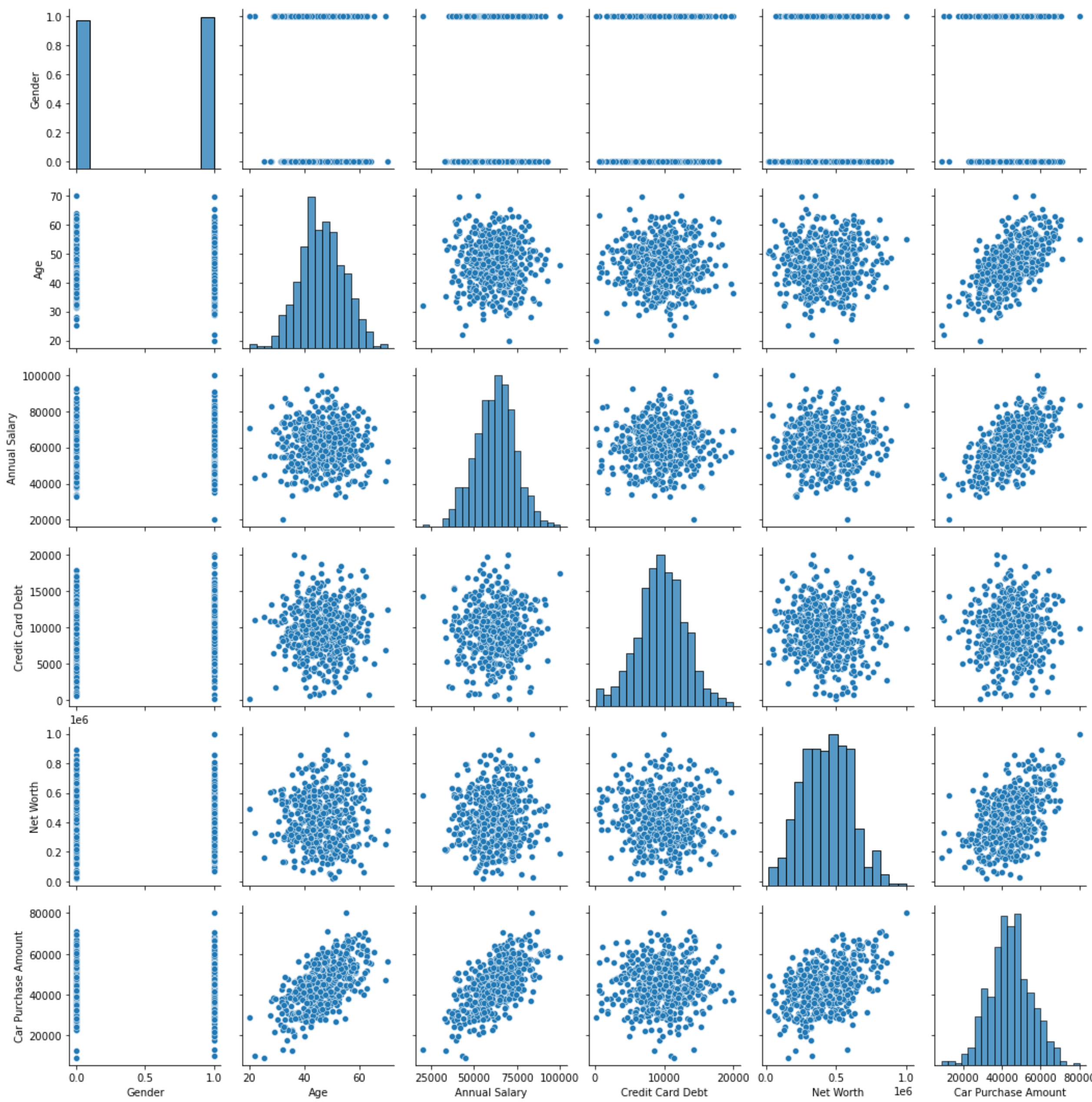
Out[6]:

	Customer Name	Customer e-mail	Country	Gender	Age	Annual Salary	Credit
495	Walter	ligula@Cumsociis.ca	Nepal	0	41.462515	71942.40291	6995.9(
496	Vanna	Cum.sociis.natoque@Sedmolestie.edu	Zimbabwe	1	37.642000	56039.49793	12301.4!
497	Pearl	penatibus.et@massanonante.com	Philippines	1	53.943497	68888.77805	10611.6(
498	Nell	Quisque.varius@arcuVivamussit.net	Botswana	1	59.160509	49811.99062	14013.0:
499	Marla	Camaron.marla@hotmail.com	marlal	1	46.731152	61370.67766	9391.3‹

In [7]:

```
# Lets visualize the dataset.

# We can see correlation between age and purchase amount, annual salary and purchase am
# and net worth with purchase amount, which makes sense intuitively.
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x261f7014cc8>



In [8]:

```
# Step 3: Create testing, training split and clean up the dataset.
# For our goal certain columns have no influence on the regression, so we're going to d

# Drop these columns (axis=1)
```

In [9]:

```
# Step 3: Create testing, training split and clean up the dataset.
```

```
# the dataframe X will be the input variables from df we use in regression to predict Y
# Y= Car Purchase Amount.

# For our goal certain attributes are assumed to have no influence on the regression,
#  we're going to drop these columns(axis=1) and the output ('Car Purchase Amount')

X = df.drop(['Customer Name', 'Customer e-mail', 'Country', 'Car Purchase Amount'], axi
Y = df['Car Purchase Amount']
```

In [10]:

```
Y.head() # Output we will train the model to predict
```

Out[10]:

```
0    35321.45877
1    45115.52566
2    42925.70921
3    67422.36313
4    55915.46248
Name: Car Purchase Amount, dtype: float64
```

In [11]:

```
X.head() # Inputs
```

Out[11]:

	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

In [12]:

```
X.shape
```

Out[12]:

```
(500, 5)
```

In [13]:

```
Y.shape
```

Out[13]:

```
(500,)
```

In [14]:

```
# We now normalize the values

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

In [15]:

```
# Maximum values

scaler.data_max_
```

Out[15]:

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

In [16]:
```
# Minimum values
scaler.data_min_
```

Out[16]:
```
array([    0.,    20., 20000.,   100., 20000.])
```

In [17]:
```
# In order for fit_transform to work with the ouput Y, we need to reshape Y (transpose)

Y= Y.values.reshape(-1,1)
Y.shape   # shapes Y to  (500,1)
```

Out[17]:
```
(500, 1)
```

In [18]:
```
Y_scaled = scaler.fit_transform(Y)
```

In [19]:
```
# Step 4. Training the Model

# We split the dataset into training and test datasets.
# The test dataset is never seen by the model in the training.


from sklearn.model_selection import train_test_split

# 25% for testing data and 75% for the training data.
X_train, X_test, y_train, y_test = train_test_split(X_scaled,Y_scaled,test_size=0.25)
```

In [20]:
```
X_train.shape # 375 samples for testing
X_test.shape  # 125 samples for testing
```

Out[20]:
```
(125, 5)
```

In [21]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

# All the outputs from one layer will be fully connected to the next (hidden) layer (de
#input_dim = 5; 5 attribute columns we use for regression

model = Sequential()

# Specify inputs and neurons in the first layer, we also specify the activation functio
model.add(Dense(25, input_dim = 5, activation='relu'))


# Lets add another hidden layer.
# 25 neurons connected to the next 25 neurons
model.add(Dense(25, activation='relu'))

# We need to predict a value, we don't want to kill the output with impulse function
model.add(Dense(1,activation='linear'))
```

In [22]:
```
# Overview of parameters in model
# We have a bias associated with each of the 25 neurons, so 150 = 125(X_samples) + 25(B
# Next layer we have 25*25 = 625 neurons + 25(bias) = 650 param
```

```
# Output layer 25 neurons connected to one output, (25 weights) + (1 bias) = 26 param

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
_________________________________________________________________
```

In [23]:

```
 # Training the model

# Using adam optimizier to speed up the gradient descent
# We take subsets of training data to avoid overfitting (validation_split)
# 100 epochs and a batch size of 50


model.compile(optimizer ='adam', loss ='mean_squared_error')
epochs_hist = model.fit(X_train, y_train, epochs = 100, batch_size = 50, verbose =1, va
```

```
Epoch 1/100
6/6 [==============================] - 0s 35ms/step - loss: 0.2100 - val_loss: 0.1459
Epoch 2/100
6/6 [==============================] - 0s 14ms/step - loss: 0.1230 - val_loss: 0.0798
Epoch 3/100
6/6 [==============================] - 0s 14ms/step - loss: 0.0624 - val_loss: 0.0420
Epoch 4/100
6/6 [==============================] - 0s 14ms/step - loss: 0.0313 - val_loss: 0.0304
Epoch 5/100
6/6 [==============================] - 0s 13ms/step - loss: 0.0230 - val_loss: 0.0320
Epoch 6/100
6/6 [==============================] - 0s 13ms/step - loss: 0.0232 - val_loss: 0.0306
Epoch 7/100
6/6 [==============================] - 0s 15ms/step - loss: 0.0210 - val_loss: 0.0250
Epoch 8/100
6/6 [==============================] - 0s 15ms/step - loss: 0.0176 - val_loss: 0.0213
Epoch 9/100
6/6 [==============================] - 0s 15ms/step - loss: 0.0161 - val_loss: 0.0194
Epoch 10/100
6/6 [==============================] - 0s 12ms/step - loss: 0.0154 - val_loss: 0.0183
Epoch 11/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0149 - val_loss: 0.0171
Epoch 12/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0141 - val_loss: 0.0162
Epoch 13/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0135 - val_loss: 0.0153
Epoch 14/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0129 - val_loss: 0.0145
Epoch 15/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0124 - val_loss: 0.0137
```

```
Epoch 16/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0118 - val_loss: 0.0130
Epoch 17/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0113 - val_loss: 0.0123
Epoch 18/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0107 - val_loss: 0.0117
Epoch 19/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0100 - val_loss: 0.0111
Epoch 20/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0093 - val_loss: 0.0105
Epoch 21/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0087 - val_loss: 0.0099
Epoch 22/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0081 - val_loss: 0.0091
Epoch 23/100
6/6 [==============================] - ETA: 0s - loss: 0.008 - 0s 3ms/step - loss: 0.007
6 - val_loss: 0.0083
Epoch 24/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0071 - val_loss: 0.0076
Epoch 25/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0067 - val_loss: 0.0070
Epoch 26/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0062 - val_loss: 0.0064
Epoch 27/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0058 - val_loss: 0.0058
Epoch 28/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0054 - val_loss: 0.0053
Epoch 29/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0050 - val_loss: 0.0049
Epoch 30/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0047 - val_loss: 0.0046
Epoch 31/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0043 - val_loss: 0.0043
Epoch 32/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0039 - val_loss: 0.0039
Epoch 33/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0036 - val_loss: 0.0036
Epoch 34/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0034 - val_loss: 0.0034
Epoch 35/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0031 - val_loss: 0.0032
Epoch 36/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0029 - val_loss: 0.0030
Epoch 37/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0027 - val_loss: 0.0028
Epoch 38/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0025 - val_loss: 0.0028
Epoch 39/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0024 - val_loss: 0.0025
Epoch 40/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0021 - val_loss: 0.0024
Epoch 41/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0020 - val_loss: 0.0022
Epoch 42/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0018 - val_loss: 0.0021
Epoch 43/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0017 - val_loss: 0.0019
Epoch 44/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0016 - val_loss: 0.0018
Epoch 45/100
```

```
6/6 [==============================] - 0s 4ms/step - loss: 0.0014 - val_loss: 0.0017
Epoch 46/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0013 - val_loss: 0.0015
Epoch 47/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0012 - val_loss: 0.0014
Epoch 48/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0011 - val_loss: 0.0012
Epoch 49/100
6/6 [==============================] - 0s 5ms/step - loss: 0.0010 - val_loss: 0.0012
Epoch 50/100
6/6 [==============================] - 0s 5ms/step - loss: 9.3841e-04 - val_loss: 0.0010
Epoch 51/100
6/6 [==============================] - 0s 4ms/step - loss: 8.6705e-04 - val_loss: 9.5371
e-04
Epoch 52/100
6/6 [==============================] - 0s 4ms/step - loss: 8.0008e-04 - val_loss: 8.6270
e-04
Epoch 53/100
6/6 [==============================] - 0s 4ms/step - loss: 7.7435e-04 - val_loss: 7.9497
e-04
Epoch 54/100
6/6 [==============================] - 0s 5ms/step - loss: 7.1351e-04 - val_loss: 7.7328
e-04
Epoch 55/100
6/6 [==============================] - 0s 4ms/step - loss: 6.5005e-04 - val_loss: 6.8705
e-04
Epoch 56/100
6/6 [==============================] - 0s 3ms/step - loss: 6.0174e-04 - val_loss: 6.6018
e-04
Epoch 57/100
6/6 [==============================] - 0s 3ms/step - loss: 5.4491e-04 - val_loss: 5.8118
e-04
Epoch 58/100
6/6 [==============================] - 0s 4ms/step - loss: 5.0000e-04 - val_loss: 5.6932
e-04
Epoch 59/100
6/6 [==============================] - 0s 4ms/step - loss: 4.6675e-04 - val_loss: 5.1422
e-04
Epoch 60/100
6/6 [==============================] - 0s 4ms/step - loss: 4.3191e-04 - val_loss: 4.7129
e-04
Epoch 61/100
6/6 [==============================] - 0s 3ms/step - loss: 3.9898e-04 - val_loss: 4.5125
e-04
Epoch 62/100
6/6 [==============================] - 0s 3ms/step - loss: 3.7095e-04 - val_loss: 4.0311
e-04
Epoch 63/100
6/6 [==============================] - 0s 3ms/step - loss: 3.4847e-04 - val_loss: 3.7797
e-04
Epoch 64/100
6/6 [==============================] - 0s 4ms/step - loss: 3.2403e-04 - val_loss: 3.6862
e-04
Epoch 65/100
6/6 [==============================] - 0s 4ms/step - loss: 3.0569e-04 - val_loss: 3.4469
e-04
Epoch 66/100
6/6 [==============================] - 0s 3ms/step - loss: 2.8711e-04 - val_loss: 3.1784
e-04
Epoch 67/100
```

```
6/6 [==============================] - 0s 4ms/step - loss: 2.6911e-04 - val_loss: 3.1357
e-04
Epoch 68/100
6/6 [==============================] - 0s 4ms/step - loss: 2.5082e-04 - val_loss: 3.0055
e-04
Epoch 69/100
6/6 [==============================] - 0s 4ms/step - loss: 2.3513e-04 - val_loss: 2.8021
e-04
Epoch 70/100
6/6 [==============================] - 0s 4ms/step - loss: 2.2410e-04 - val_loss: 2.6324
e-04
Epoch 71/100
6/6 [==============================] - 0s 4ms/step - loss: 2.1308e-04 - val_loss: 2.5157
e-04
Epoch 72/100
6/6 [==============================] - 0s 4ms/step - loss: 2.0244e-04 - val_loss: 2.4065
e-04
Epoch 73/100
6/6 [==============================] - 0s 4ms/step - loss: 1.9522e-04 - val_loss: 2.3563
e-04
Epoch 74/100
6/6 [==============================] - 0s 4ms/step - loss: 1.8347e-04 - val_loss: 2.2579
e-04
Epoch 75/100
6/6 [==============================] - 0s 3ms/step - loss: 1.7611e-04 - val_loss: 2.1631
e-04
Epoch 76/100
6/6 [==============================] - 0s 4ms/step - loss: 1.7092e-04 - val_loss: 2.0513
e-04
Epoch 77/100
6/6 [==============================] - 0s 4ms/step - loss: 1.6995e-04 - val_loss: 2.0109
e-04
Epoch 78/100
6/6 [==============================] - 0s 4ms/step - loss: 1.6187e-04 - val_loss: 1.9605
e-04
Epoch 79/100
6/6 [==============================] - 0s 4ms/step - loss: 1.5434e-04 - val_loss: 1.8651
e-04
Epoch 80/100
6/6 [==============================] - 0s 3ms/step - loss: 1.4900e-04 - val_loss: 1.7947
e-04
Epoch 81/100
6/6 [==============================] - 0s 4ms/step - loss: 1.4076e-04 - val_loss: 1.8120
e-04
Epoch 82/100
6/6 [==============================] - 0s 3ms/step - loss: 1.4014e-04 - val_loss: 1.7969
e-04
Epoch 83/100
6/6 [==============================] - 0s 3ms/step - loss: 1.3537e-04 - val_loss: 1.7032
e-04
Epoch 84/100
6/6 [==============================] - 0s 4ms/step - loss: 1.2930e-04 - val_loss: 1.6850
e-04
Epoch 85/100
6/6 [==============================] - 0s 4ms/step - loss: 1.2765e-04 - val_loss: 1.6756
e-04
Epoch 86/100
6/6 [==============================] - 0s 3ms/step - loss: 1.2283e-04 - val_loss: 1.5755
e-04
Epoch 87/100
```

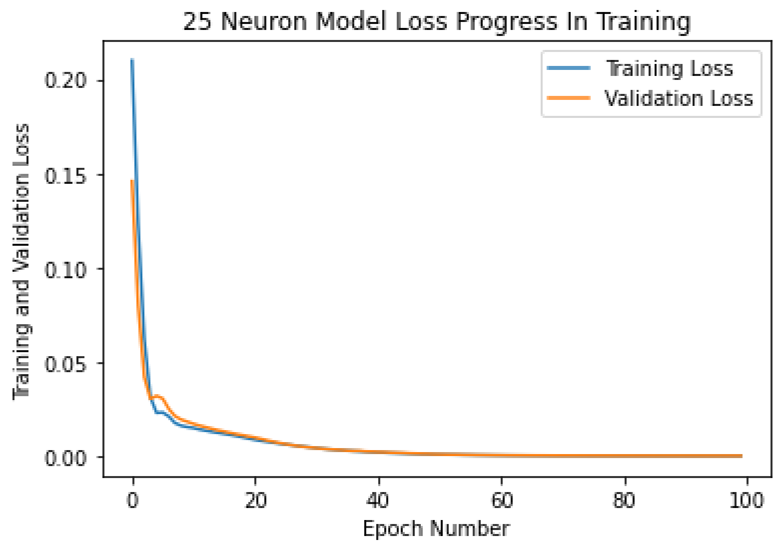
```
6/6 [==============================] - 0s 3ms/step - loss: 1.2125e-04 - val_loss: 1.5630
e-04
Epoch 88/100
6/6 [==============================] - 0s 3ms/step - loss: 1.1873e-04 - val_loss: 1.5528
e-04
Epoch 89/100
6/6 [==============================] - 0s 3ms/step - loss: 1.1594e-04 - val_loss: 1.5754
e-04
Epoch 90/100
6/6 [==============================] - 0s 4ms/step - loss: 1.1236e-04 - val_loss: 1.5308
e-04
Epoch 91/100
6/6 [==============================] - 0s 4ms/step - loss: 1.1158e-04 - val_loss: 1.4655
e-04
Epoch 92/100
6/6 [==============================] - 0s 3ms/step - loss: 1.0745e-04 - val_loss: 1.4680
e-04
Epoch 93/100
6/6 [==============================] - 0s 3ms/step - loss: 1.0725e-04 - val_loss: 1.4930
e-04
Epoch 94/100
6/6 [==============================] - 0s 4ms/step - loss: 1.0550e-04 - val_loss: 1.4617
e-04
Epoch 95/100
6/6 [==============================] - 0s 4ms/step - loss: 1.0351e-04 - val_loss: 1.4484
e-04
Epoch 96/100
6/6 [==============================] - 0s 3ms/step - loss: 1.0160e-04 - val_loss: 1.3819
e-04
Epoch 97/100
6/6 [==============================] - 0s 3ms/step - loss: 9.9596e-05 - val_loss: 1.3741
e-04
Epoch 98/100
6/6 [==============================] - 0s 4ms/step - loss: 9.8326e-05 - val_loss: 1.4481
e-04
Epoch 99/100
6/6 [==============================] - 0s 3ms/step - loss: 9.7393e-05 - val_loss: 1.3872
e-04
Epoch 100/100
6/6 [==============================] - 0s 4ms/step - loss: 9.6212e-05 - val_loss: 1.3281
e-04
```

In [24]:

```
# Using matplotlib to visualize the epoch history

epochs_hist.history.keys()


plt.plot(epochs_hist.history['loss'])
plt.plot(epochs_hist.history['val_loss'])

plt.title('25 Neuron Model Loss Progress In Training')
plt.ylabel('Training and Validation Loss')
plt.xlabel('Epoch Number')
plt.legend(['Training Loss', 'Validation Loss'])
```

Out[24]:

```
<matplotlib.legend.Legend at 0x26181a76b88>
```



In [25]:

```
# Changing the architecture of the network,
# Reducing the number of neurons in connected layers to 5

model5 = Sequential()
model5.add(Dense(5, input_dim = 5, activation='relu'))
model5.add(Dense(5, activation='relu'))

# Last layer
model5.add(Dense(1,activation='linear'))
model5.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 30
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 6
=================================================================
Total params: 66
Trainable params: 66
Non-trainable params: 0
_________________________________________________________________
```

In [26]:

```
model5.compile(optimizer ='adam', loss ='mean_squared_error')
epochs_chart = model5.fit(X_train, y_train, epochs =100, batch_size = 50, verbose = 1,
```

```
Epoch 1/100
6/6 [==============================] - 0s 16ms/step - loss: 0.0361 - val_loss: 0.0315
Epoch 2/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0262 - val_loss: 0.0264
Epoch 3/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0237 - val_loss: 0.0247
Epoch 4/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0231
Epoch 5/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0201 - val_loss: 0.0222
Epoch 6/100
```

```
6/6 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0218
Epoch 7/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0211
Epoch 8/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0203
Epoch 9/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0168 - val_loss: 0.0195
Epoch 10/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0187
Epoch 11/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0178
Epoch 12/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0144 - val_loss: 0.0169
Epoch 13/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0136 - val_loss: 0.0158
Epoch 14/100
6/6 [==============================] - 0s 5ms/step - loss: 0.0127 - val_loss: 0.0147
Epoch 15/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0118 - val_loss: 0.0137
Epoch 16/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0110 - val_loss: 0.0127
Epoch 17/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0102 - val_loss: 0.0117
Epoch 18/100
6/6 [==============================] - 0s 22ms/step - loss: 0.0095 - val_loss: 0.0109
Epoch 19/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0088 - val_loss: 0.0102
Epoch 20/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0082 - val_loss: 0.0095
Epoch 21/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0076 - val_loss: 0.0089
Epoch 22/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0071 - val_loss: 0.0083
Epoch 23/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0066 - val_loss: 0.0078
Epoch 24/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0062 - val_loss: 0.0073
Epoch 25/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0058 - val_loss: 0.0069
Epoch 26/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0054 - val_loss: 0.0065
Epoch 27/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0052 - val_loss: 0.0062
Epoch 28/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0049 - val_loss: 0.0058
Epoch 29/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0046 - val_loss: 0.0056
Epoch 30/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0044 - val_loss: 0.0054
Epoch 31/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0042 - val_loss: 0.0051
Epoch 32/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0040 - val_loss: 0.0049
Epoch 33/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0039 - val_loss: 0.0047
Epoch 34/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0037 - val_loss: 0.0045
Epoch 35/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0036 - val_loss: 0.0043
Epoch 36/100
```

```
6/6 [==============================] - 0s 3ms/step - loss: 0.0034 - val_loss: 0.0042
Epoch 37/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0033 - val_loss: 0.0040
Epoch 38/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0032 - val_loss: 0.0039
Epoch 39/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0031 - val_loss: 0.0038
Epoch 40/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0030 - val_loss: 0.0036
Epoch 41/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0029 - val_loss: 0.0034
Epoch 42/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0028 - val_loss: 0.0034
Epoch 43/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0027 - val_loss: 0.0032
Epoch 44/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0026 - val_loss: 0.0031
Epoch 45/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 46/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0024 - val_loss: 0.0029
Epoch 47/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0024 - val_loss: 0.0028
Epoch 48/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0023 - val_loss: 0.0027
Epoch 49/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0022 - val_loss: 0.0026
Epoch 50/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0022 - val_loss: 0.0025
Epoch 51/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0021 - val_loss: 0.0024
Epoch 52/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0020 - val_loss: 0.0023
Epoch 53/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0020 - val_loss: 0.0023
Epoch 54/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0019 - val_loss: 0.0021
Epoch 55/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0019 - val_loss: 0.0021
Epoch 56/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0018 - val_loss: 0.0020
Epoch 57/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0017 - val_loss: 0.0019
Epoch 58/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0017 - val_loss: 0.0018
Epoch 59/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0016 - val_loss: 0.0018
Epoch 60/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0016 - val_loss: 0.0017
Epoch 61/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0015 - val_loss: 0.0016
Epoch 62/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0015 - val_loss: 0.0016
Epoch 63/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0014 - val_loss: 0.0015
Epoch 64/100
6/6 [==============================] - 0s 4ms/step - loss: 0.0014 - val_loss: 0.0015
Epoch 65/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0013 - val_loss: 0.0014
Epoch 66/100
```

```
6/6 [==============================] - 0s 4ms/step - loss: 0.0013 - val_loss: 0.0014
Epoch 67/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0012 - val_loss: 0.0013
Epoch 68/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0012 - val_loss: 0.0013
Epoch 69/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0012 - val_loss: 0.0012
Epoch 70/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0011 - val_loss: 0.0012
Epoch 71/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0011 - val_loss: 0.0011
Epoch 72/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0010 - val_loss: 0.0011
Epoch 73/100
6/6 [==============================] - 0s 3ms/step - loss: 0.0010 - val_loss: 0.0010
Epoch 74/100
6/6 [==============================] - 0s 3ms/step - loss: 9.7459e-04 - val_loss: 9.9572
e-04
Epoch 75/100
6/6 [==============================] - 0s 4ms/step - loss: 9.4172e-04 - val_loss: 9.5484
e-04
Epoch 76/100
6/6 [==============================] - 0s 3ms/step - loss: 9.0997e-04 - val_loss: 9.2351
e-04
Epoch 77/100
6/6 [==============================] - 0s 4ms/step - loss: 8.8030e-04 - val_loss: 8.9230
e-04
Epoch 78/100
6/6 [==============================] - 0s 3ms/step - loss: 8.5119e-04 - val_loss: 8.4638
e-04
Epoch 79/100
6/6 [==============================] - 0s 3ms/step - loss: 8.2107e-04 - val_loss: 8.1767
e-04
Epoch 80/100
6/6 [==============================] - 0s 3ms/step - loss: 7.9561e-04 - val_loss: 7.7663
e-04
Epoch 81/100
6/6 [==============================] - 0s 3ms/step - loss: 7.6362e-04 - val_loss: 7.4292
e-04
Epoch 82/100
6/6 [==============================] - 0s 3ms/step - loss: 7.3566e-04 - val_loss: 7.0983
e-04
Epoch 83/100
6/6 [==============================] - 0s 3ms/step - loss: 7.1270e-04 - val_loss: 6.7627
e-04
Epoch 84/100
6/6 [==============================] - 0s 4ms/step - loss: 6.8674e-04 - val_loss: 6.4332
e-04
Epoch 85/100
6/6 [==============================] - 0s 4ms/step - loss: 6.6811e-04 - val_loss: 6.2181
e-04
Epoch 86/100
6/6 [==============================] - 0s 3ms/step - loss: 6.4791e-04 - val_loss: 5.8497
e-04
Epoch 87/100
6/6 [==============================] - 0s 3ms/step - loss: 6.2388e-04 - val_loss: 5.7475
e-04
Epoch 88/100
6/6 [==============================] - 0s 3ms/step - loss: 5.9930e-04 - val_loss: 5.2880
e-04
```

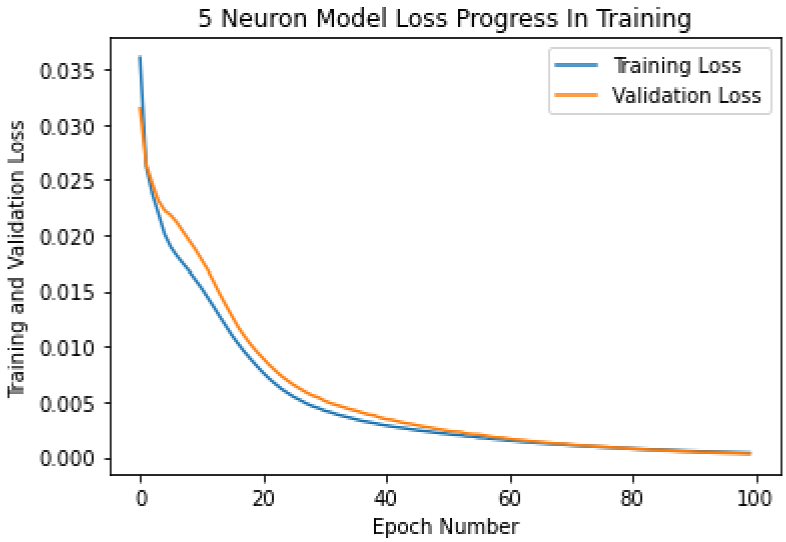
```
Epoch 89/100
6/6 [==============================] - 0s 3ms/step - loss: 5.7554e-04 - val_loss: 5.0272
e-04
Epoch 90/100
6/6 [==============================] - 0s 3ms/step - loss: 5.4818e-04 - val_loss: 4.9696
e-04
Epoch 91/100
6/6 [==============================] - 0s 4ms/step - loss: 5.3204e-04 - val_loss: 4.5658
e-04
Epoch 92/100
6/6 [==============================] - 0s 3ms/step - loss: 5.1086e-04 - val_loss: 4.3620
e-04
Epoch 93/100
6/6 [==============================] - 0s 4ms/step - loss: 4.9026e-04 - val_loss: 4.1858
e-04
Epoch 94/100
6/6 [==============================] - 0s 3ms/step - loss: 4.7570e-04 - val_loss: 3.9919
e-04
Epoch 95/100
6/6 [==============================] - 0s 4ms/step - loss: 4.5697e-04 - val_loss: 3.7777
e-04
Epoch 96/100
6/6 [==============================] - 0s 3ms/step - loss: 4.4237e-04 - val_loss: 3.5964
e-04
Epoch 97/100
6/6 [==============================] - 0s 4ms/step - loss: 4.2317e-04 - val_loss: 3.4093
e-04
Epoch 98/100
6/6 [==============================] - 0s 3ms/step - loss: 4.0967e-04 - val_loss: 3.2799
e-04
Epoch 99/100
6/6 [==============================] - 0s 3ms/step - loss: 3.9163e-04 - val_loss: 3.0729
e-04
Epoch 100/100
6/6 [==============================] - 0s 4ms/step - loss: 3.7913e-04 - val_loss: 2.9361
e-04
```

In [27]:

```
plt.plot(epochs_chart.history['loss'])
plt.plot(epochs_chart.history['val_loss'])
plt.title('5 Neuron Model Loss Progress In Training')
plt.ylabel('Training and Validation Loss')
plt.xlabel('Epoch Number')
plt.legend(['Training Loss', 'Validation Loss'])
```

Out[27]:

```
<matplotlib.legend.Legend at 0x2618177fd48>
```



In [28]:

```
# Lets predict with the models.
# We use scaled values for input and use an inverse transformation on the predicted val

# Test the model with the first row of X_scaled

# Gender, Age, Annual Salary, Credit Card Debt, Net Worth
X_test = np.array([[0, 0.4370344,  0.53515116, 0.57836085, 0.22342985]])

# Predicitions
y_predict = model.predict(X_test)
y_predict5 = model5.predict(X_test)

# Printing values before inverting
print('(25NN) Expected Purchase Amount= ', y_predict)
print('(5NN) Expected Purchase Amount  = ', y_predict5)


# Inverting predictions
y_predict_sample_actual = scaler.inverse_transform(y_predict)
y_predict_sample_actual5 = scaler.inverse_transform(y_predict5)
```

```
(25NN) Expected Purchase Amount=  [[0.3685331]]
(5NN) Expected Purchase Amount  =  [[0.37696424]]
```

In [29]:

```
# Printing usable expected values from both models
print('(25NN) Actual Expected Purchase Amount= ', y_predict_sample_actual)
print('(5NN) Actual Expected Purchase Amount= ', y_predict_sample_actual5)
```

```
(25NN) Actual Expected Purchase Amount=  [[35165.85]]
(5NN) Actual Expected Purchase Amount=  [[35764.46]]
```

Conclusion:

For a customer with X_test values for attributes, we show them cars in the price range of y_predict.

In this way the model can help the salesmen specifically target customers better, leading to increased efficiency and more sales.