1.) Preprocess your data into scaled input variables and an output variable

```
import pandas as pd
from google.colab import drive
import matplotlib.pyplot as plt
import numpy as np
import datetime
```

```
drive.mount('/content/gdrive/', force_remount = True)
```

```
    Mounted at /content/gdrive/
```

```
df = pd.read_csv("/content/gdrive/MyDrive/ECON441B/CLV.csv")
```

```
df
```

	Unnamed: 0	Customer Lifetime Value	Income	Number of Policies	Total Claim Amount	Months Since Last Claim	Vehicle Size_Large	Si
0	0	2763.519279	56274	1	384.811147	32	0	
1	1	6979.535903	0	8	1131.464935	13	0	
2	2	12887.431650	48767	2	566.472247	18	0	
3	3	7645.861827	0	7	529.881344	18	0	
4	4	2813.692575	43836	1	138.130879	12	0	
...	...	...	...	...	...	...	...	
9129	9129	23405.987980	71941	2	198.234764	18	0	
9130	9130	3096.511217	21604	1	379.200000	14	0	
9131	9131	8163.890428	0	2	790.784983	9	0	
9132	9132	7524.442436	21941	3	691.200000	34	1	
9133	9133	2611.836866	0	1	369.600000	3	0	

9134 rows × 18 columns

```
X = df.drop(["Unnamed: 0","Customer Lifetime Value"],axis=1)
y = df['Customer Lifetime Value']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3)
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

2.) Run a GridSearch CV on at least 10 possible combinations of hyper parameters

```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV
```

```
MLPRegressor
```

```
    sklearn.neural_network._multilayer_perceptron.MLPRegressor
```

```
clf = MLPRegressor()
params = {"hidden_layer_sizes":[(10,),(10,50),(20,5)],
          "activation": ['relu','logistic','sigmoid','tanh']}
```

```
grid = GridSearchCV(clf,params, cv=5)
grid.fit(X_train, y_train)
```

```
print('Best parameters:',grid.best_params_)
print('Best score: {}'.format(grid.best_score_))
```

```
    Best parameters: {'activation': 'relu', 'hidden_layer_sizes': (10, 50)}
    Best score: 0.06571544775867602
```

3.) Train a model with the optimal solution from GridSearch

```
MLPRegressor(**grid.best_params_)
```

```
    MLPRegressor(hidden_layer_sizes=(10, 50))
```

```
p_dict = {"hidden_layer_sizes": (10,50),"activation": 'relu'}
model= MLPRegressor(**p_dict)
```

```
model.fit(X_train, y_train)
```

```
    /usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_per
      warnings.warn(
    MLPRegressor(hidden_layer_sizes=(10, 50))
```

4.) What are the in-sample and out of sample MSEs

```
from sklearn.metrics import mean_squared_error
```

```
y_pred_train = model.predict(X_train)
mse_train = mean_squared_error(y_train, y_pred_train)
print('In-sample MSE:', mse_train)
```

```
    In-sample MSE: 44357388.21973672
```

```
y_pred_test = model.predict(X_test)
mse_test = mean_squared_error(y_test, y_pred_test)
print('Out-of-sample MSE:', mse_test)
```

```
    Out-of-sample MSE: 41453159.33647162
```

5.) Build a Keras with the architecture defined by GridSearchCV

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

```
hidden_layer_sizes = (10,50)
activation_function = 'relu'


model1 = Sequential()


model1.add(Dense(hidden_layer_sizes[0], input_dim=X_train.shape[1], activation=acti

for layer_size in hidden_layer_sizes[1:]:
    model1.add(Dense(layer_size, activation=activation_function))

model1.add(Dense(1, activation='linear'))

model1.compile(loss='mean_squared_error', optimizer='adam')

model1.fit(X_train, y_train, epochs=100, batch_size=10, verbose=0)
```

```
<keras.callbacks.History at 0x7f36650a4100>
```

```
mse_in_sample = model1.evaluate(X_train, y_train, verbose=0)
mse_out_of_sample = model1.evaluate(X_test, y_test, verbose=0)

print("In-sample MSE: ", mse_in_sample)
print("Out-of-sample MSE: ", mse_out_of_sample)
```

```
In-sample MSE:  43738468.0
Out-of-sample MSE:  41054744.0
```

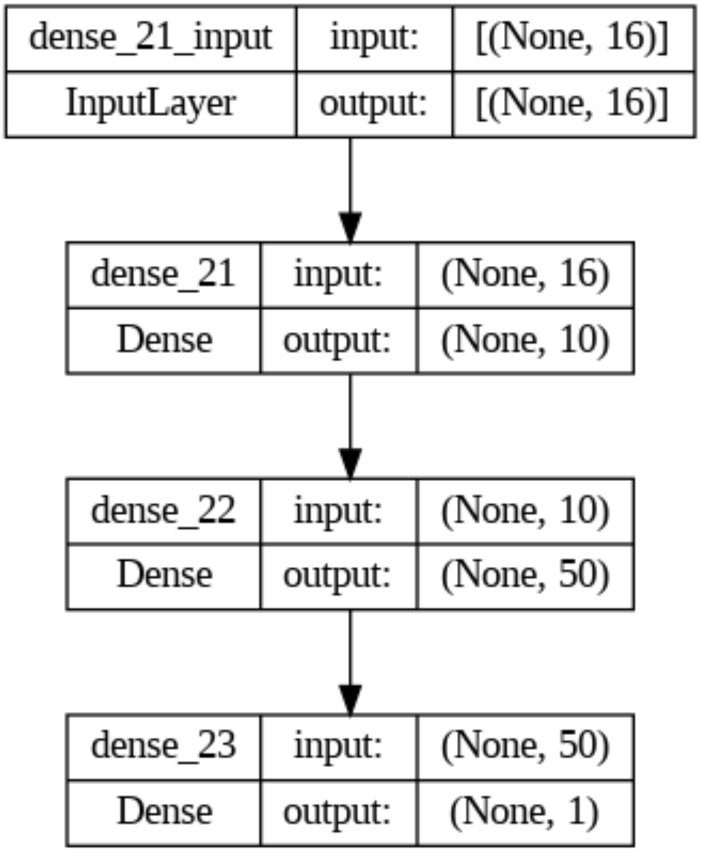
6.) Make two visualizations of your NN using “plot_model” and “ann_viz”

```
!pip install ann_visualizer
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-w
Requirement already satisfied: ann_visualizer in /usr/local/lib/python3.8/dist
```

```
from tensorflow.keras.utils import plot_model
from ann_visualizer.visualize import ann_viz
```

```
plot_model(model1, show_shapes=True, show_layer_names=True)
```



dense_21_input | input: | [(None, 16)]
InputLayer | output: | [(None, 16)]

dense_21 | input: | (None, 16)
Dense | output: | (None, 10)

dense_22 | input: | (None, 10)
Dense | output: | (None, 50)

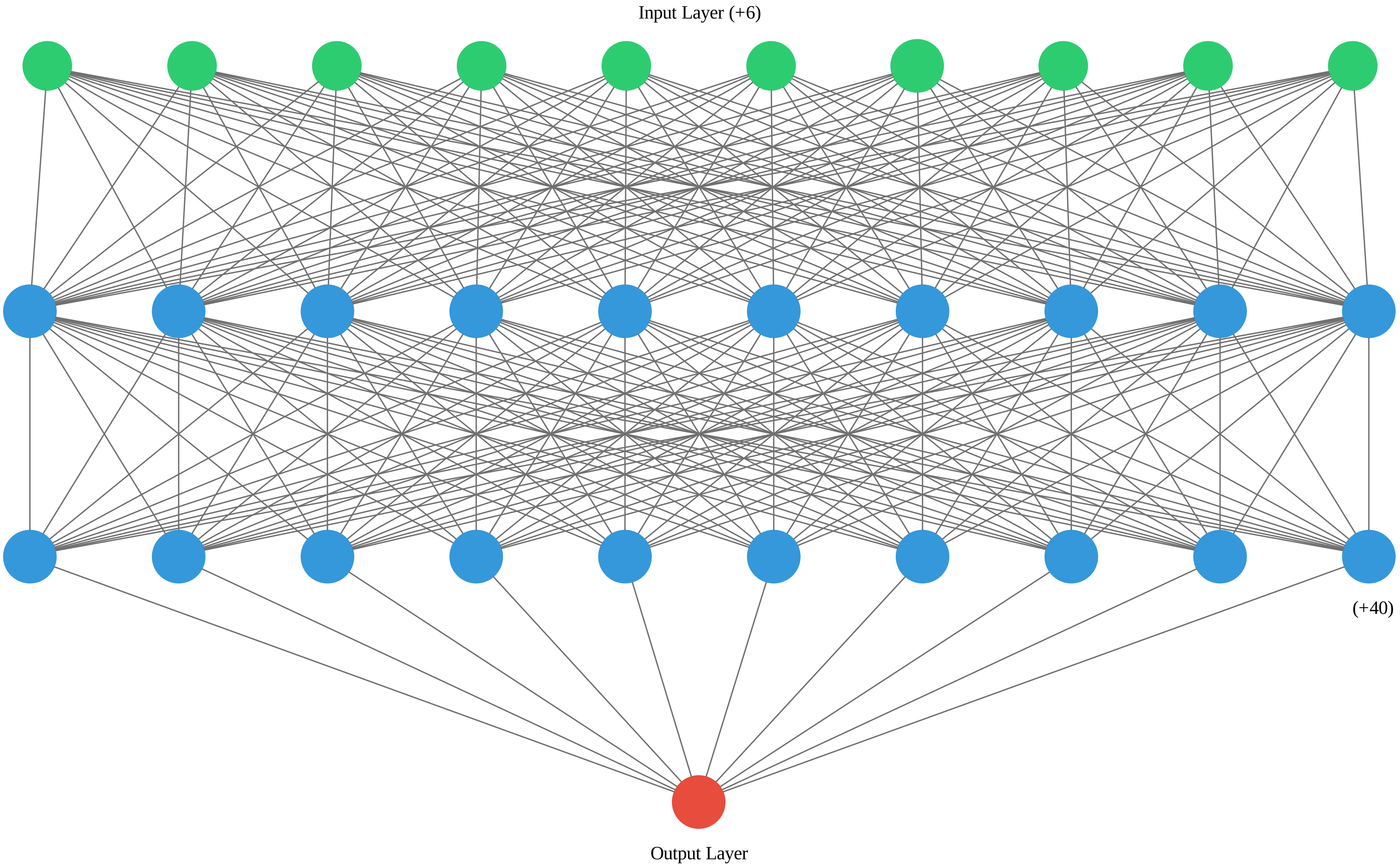
dense_23 | input: | (None, 50)
Dense | output: | (None, 1)

```
ann_viz(model1, title="My Neural Network", view=True)
```

```
!pip install google.colab
from google.colab import files
```

```
files.download("network.gv.pdf")
```

My Neural Network



Colab 付费产品 - 在此处取消合同

✓ 9 秒 完成时间：18:08 ● ✕