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	Vehicle Size_Medsize
0	0	2763.519279	56274	1	384.811147	32	0	1
1	1	6979.535903	0	8	1131.464935	13	0	1
2	2	12887.431650	48767	2	566.472247	18	0	1
3	3	7645.861827	0	7	529.881344	18	0	1
4	4	2813.692575	43836	1	138.130879	12	0	1
...	...	...	...	...	...	...	...	...
9129	9129	23405.987980	71941	2	198.234764	18	0	1
9130	9130	3096.511217	21604	1	379.200000	14	0	1
9131	9131	8163.890428	0	2	790.784983	9	0	1
9132	9132	7524.442436	21941	3	691.200000	34	1	0
9133	9133	2611.836866	0	1	369.600000	3	0	1

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 = .25)
```

```
X_train
```

	Income	Number of Policies	Total Claim Amount	Months Since Last Claim	Vehicle Size_Large	Vehicle Size_Medsize	Gender_M	EmploymentSta
2340	0	4	753.225650	3	0	1	1	
8618	60964	4	327.723713	21	0	1	1	

```
from sklearn.preprocessing import StandardScaler

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

```
X_train
```

```
array([[-1.24179629,  0.42679261,  1.11194514, ..., -0.41924196,
         0.84105759, -0.60017207],
       [ 0.7683049 ,  0.42679261, -0.36732926, ..., -0.41924196,
         0.84105759, -0.60017207],
       [-1.24179629,  1.26220607,  1.49683167, ..., -0.41924196,
        -1.18897923,  1.66618883],
       ...,
       [-1.24179629, -0.40862085,  1.85354469, ..., -0.41924196,
         0.84105759, -0.60017207],
       [-0.62825448, -0.82632757, -0.32186903, ...,  2.38525745,
        -1.18897923, -0.60017207],
       [ 0.98628237, -0.40862085, -0.11046826, ..., -0.41924196,
         0.84105759, -0.60017207]])
```

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?
```

```
clf = MLPRegressor()

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

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

```
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
```

```
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer: M
  warnings.warn(
GridSearchCV(cv=5, estimator=MLPRegressor(),
             param_grid={'activation': ['relu', 'logistic'],
                         'hidden_layer_sizes': [(10,), (20, 5)]})
```

```
print('Best Parameters:',grid.best_params_)
print('Best Score:',grid.best_score_)
```

```
Best Parameters: {'activation': 'relu', 'hidden_layer_sizes': (20, 5)}
Best Score: 0.055823478826140804
```

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

```
opt_model = MLPRegressor(**grid.best_params_)
opt_model.fit(X, y)
```

```
MLPRegressor(hidden_layer_sizes=(20, 5))
```

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

- The in sample MSE is 111085646.23568861 and the out of sample MSE: 109367046.87773126

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

```
y_train_prediction = opt_model.predict(X_train)
y_test_prediction = opt_model.predict(X_test)
mse_train = mean_squared_error(y_train, y_train_prediction)
mse_test = mean_squared_error(y_test, y_test_prediction)
print('In sample MSE:', mse_train)
print('Out of sample MSE:', mse_test)
```

```
In sample MSE: 111085646.23568861
Out of sample MSE: 109367046.87773126
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but MLPRegressor was fitte
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but MLPRegressor was fitte
  warnings.warn(
```

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

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

```
model = Sequential()
model.add(Dense(20, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(5, activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer= Adam(lr=.01))
model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=0)
```

```
/usr/local/lib/python3.8/dist-packages/keras/optimizers/optimizer_v2/adam.py:117: UserWarning: The `lr` argument is deprecated, use `lea
  super().__init__(name, **kwargs)
<keras.callbacks.History at 0x7f530fae3340>
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 20)                340

 dense_4 (Dense)             (None, 5)                 105

 dense_5 (Dense)             (None, 1)                 6

=================================================================
Total params: 451
Trainable params: 451
Non-trainable params: 0
_________________________________________________________________
```

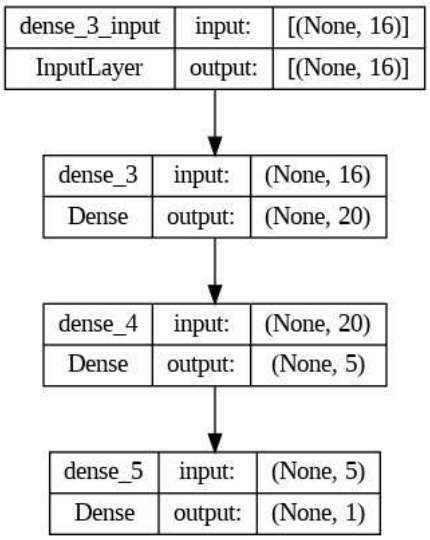
6.) Make two visualizations of your NN using “plot_model” and “ann_viz”

```
pip install ann_visualizer
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ann_visualizer
  Downloading ann_visualizer-2.5.tar.gz (4.7 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: ann_visualizer
  Building wheel for ann_visualizer (setup.py) ... done
  Created wheel for ann_visualizer: filename=ann_visualizer-2.5-py3-none-any.whl size=4168 sha256=905306236e75c4ec94189430f7bf5107e82d2c
  Stored in directory: /root/.cache/pip/wheels/4b/ef/77/9b8c4ae2f9a11de19957b80bc5c684accd99114bb8dc6b374c
Successfully built ann_visualizer
Installing collected packages: ann_visualizer
Successfully installed ann_visualizer-2.5
```

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

```
plot_model(model, to_file='plot_model.png', show_shapes=True, show_layer_names=True)
```



```
from ann_visualizer.visualize import ann_viz
from IPython.display import Image


ann_viz(model, title="Neural Network Visualization", view=True)
```