Youssef Mahmoud 905854027

```
import pandas as pd
from google.colab import drive
import matplotlib.pyplot as plt
import numpy as np
import datetime
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
import keras.models
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import mean_squared_error
from ann_visualizer.visualize import ann_viz
from pdf2image import convert_from_path

from tensorflow.keras.utils import plot_model
```

Double-click (or enter) to edit

```
df = pd.read_csv("/content/gdrive/MyDrive/stroke/CLV.csv",parse_dates = True, index_col = 0)
```

```
df
```

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

9134 rows × 17 columns

```
X = df.drop(["Customer Lifetime Value"], axis =1 )
Y = df['Customer Lifetime Value']
```

```
 X_train,X_test,Y_train,Y_test = sklearn.model_selection.train_test_split( X, Y, test_size=0.3)
```

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

```
clf = MLPRegressor()
```

```
clf = MLPRegressor()
```

Double-click (or enter) to edit

```
params = {
    'hidden_layer_sizes':[(10,),(5,15,),(5,15,20,),(10,40,100,)],
    'activation':['relu','logistic'],
    'alpha':[0.0001,0.001,0.01]
}
```

```
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: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:692: ConvergenceWarning: Stochast
  warnings.warn(
GridSearchCV(cv=5, estimator=MLPRegressor(),
             param_grid={'activation': ['relu', 'logistic'],
                         'alpha': [0.0001, 0.001, 0.01],
                         'hidden_layer_sizes': [(10,), (5, 15), (5, 15, 20),
                                                (10, 40, 100)]})
```

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

```
Best Parameters: {'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': (5, 15, 20)}
Best Score: 0.06464502713548756
```

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

```
MLPRegressor(alpha=0.01, hidden_layer_sizes=(5, 15, 20))
```

```
p_dictionary = {
    "hidden_layer_sizes": (5, 15, 20),
    "activation" : 'relu',
    "alpha":(0.01),
}

MLP = MLPRegressor(**p_dictionary, solver='adam',max_iter = 1000)
```

```
MLP.fit(X_train,Y_train)
```

```
MLPRegressor(alpha=0.01, hidden_layer_sizes=(5, 15, 20), max_iter=1000)
```

```
Y_pred = MLP.predict(X_train)
Y_pred1 = MLP.predict(X_test)
```

```
Train_MSE = mean_squared_error(Y_train, Y_pred)
Test_MSE = mean_squared_error(Y_test, Y_pred1)
print('Train MSE = ',Train_MSE)
print('Test MSE = ', Test_MSE)
```

```
Train MSE =  42855364.40180898
Test MSE =  43755781.25363725
```

```
model = Sequential()
model.add(Dense(5, input_dim=X_train.shape[1],activation="relu"))
model.add(Dense(15, activation = 'relu'))
model.add(Dense(20, activation = 'relu'))
```

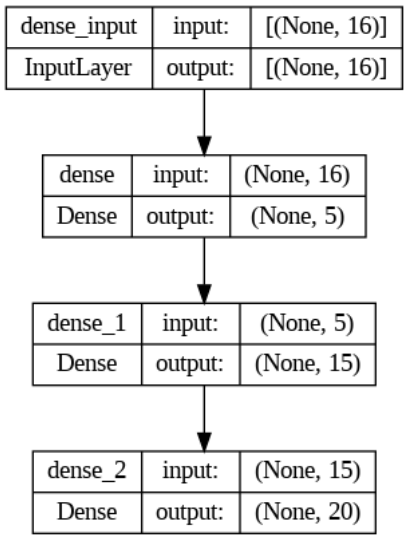
```
model.compile(loss='MSE',optimizer=Adam(lr=0.01))
model.fit(X_train,Y_train,batch_size=32,epochs=100)
```

```
Epoch 92/100
200/200 [==============================] - 1s 3ms/step - loss: 46330152.0000
Epoch 93/100
200/200 [==============================] - 1s 3ms/step - loss: 46325576.0000
Epoch 94/100
200/200 [==============================] - 1s 3ms/step - loss: 46311292.0000
Epoch 95/100
200/200 [==============================] - 1s 3ms/step - loss: 46307484.0000
Epoch 96/100
200/200 [==============================] - 0s 2ms/step - loss: 46364576.0000
Epoch 97/100
200/200 [==============================] - 0s 2ms/step - loss: 46300180.0000
Epoch 98/100
200/200 [==============================] - 0s 2ms/step - loss: 46312756.0000
Epoch 99/100
200/200 [==============================] - 0s 2ms/step - loss: 46341608.0000
Epoch 100/100
200/200 [==============================] - 0s 2ms/step - loss: 46287692.0000
```

```
model.summary()
```

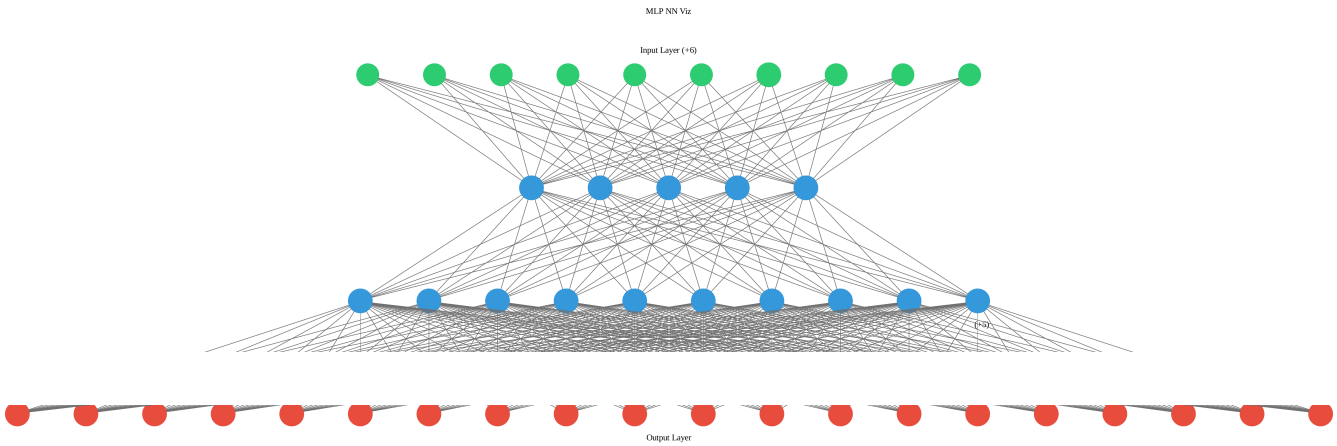
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 5)                 85

 dense_1 (Dense)             (None, 15)                90

 dense_2 (Dense)             (None, 20)                320

=================================================================
Total params: 495
Trainable params: 495
Non-trainable params: 0
_________________________________________________________________
```

```
plot_model(model, show_shapes=True)
```




```
ann_viz(model, title = "MLP NN Viz",filename="/content/mlp_model")
```

```
NN = convert_from_path("/content/mlp_model.pdf")
NN[0]
```



Colab paid products - Cancel contracts here

✓ 0s completed at 5:02 PM