import statements

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [2]:

```
import sklearn.datasets
```

In [3]:

```
Diabetes_Dataset = sklearn.datasets.load_diabetes()
```

In [4]:

```
print(Diabetes_Dataset)
```

```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990749, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06833155, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286131, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04688253,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452873, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00422151,  0.00306441]]), 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310., 1
01.,
        69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
        68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
        87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
       259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
       128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
       150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
       200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
        42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
        83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
       104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
       220.,  57.]), 'frame': None, 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen ba
seline variables, age, sex, body mass index, average blood\npressure, and six blood serum measurements were obtaine
d for each of n =\n442 diabetes patients, as well as the response of interest, a\nquantitative measure of disease p
rogression one year after baseline.\n\n**Data Set Characteristics:**\n\n  :Number of Instances: 442\n\n  :Number of
Attributes: First 10 columns are numeric predictive values\n\n  :Target: Column 11 is a quantitative measure of dis
ease progression one year after baseline\n\n  :Attribute Information:\n      - age     age in years\n      - sex\n
- bmi     body mass index\n      - bp      average blood pressure\n      - s1      tc, total serum cholesterol\n
- s2      ldl, low-density lipoproteins\n      - s3      hdl, high-density lipoproteins\n      - s4      tch, total
cholesterol / HDL\n      - s5      ltg, possibly log of serum triglycerides level\n      - s6      glu, blood sugar
level\n\nNote: Each of these 10 feature variables have been mean centered and scaled by the standard deviation time
s the square root of `n_samples` (i.e. the sum of squares of each column totals 1).\n\nSource URL:\nhttps://www4.st
at.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley Efron, Trevor Hastie, Iain Johnsto
ne and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussion), 407-499.\n(http
s://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)\n', 'feature_names': ['age', 'sex', 'bmi', 'bp', 's
1', 's2', 's3', 's4', 's5', 's6'], 'data_filename': 'diabetes_data_raw.csv.gz', 'target_filename': 'diabetes_targe
t.csv.gz', 'data_module': 'sklearn.datasets.data'}
```

In [5]:

```
list(Diabetes_Dataset.feature_names)
```

Out[5]:

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

In [6]:

```
import pandas as pd
df = pd.DataFrame(Diabetes_Dataset.data, columns= Diabetes_Dataset.feature_names)
```

In [7]:

```
df.head()
```

Out[7]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641

In [8]:

```
df.shape
```

Out[8]:

```
(442, 10)
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
age    0
sex    0
bmi    0
bp     0
s1     0
s2     0
s3     0
s4     0
s5     0
s6     0
dtype: int64
```

In [10]:

```
df.describe()
```

Out[10]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	
count	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.4200(
mean	-1.444295e-18	2.543215e-18	-2.255925e-16	-4.854086e-17	-1.428596e-17	3.898811e-17	-6.028360e-18	-1.788100e-17	9.243486e-17	1.3517
std	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.7619
min	-1.072256e-01	-4.464164e-02	-9.027530e-02	-1.123988e-01	-1.267807e-01	-1.156131e-01	-1.023071e-01	-7.639450e-02	-1.260971e-01	-1.37
25%	-3.729927e-02	-4.464164e-02	-3.422907e-02	-3.665608e-02	-3.424784e-02	-3.035840e-02	-3.511716e-02	-3.949338e-02	-3.324559e-02	-3.31
50%	5.383060e-03	-4.464164e-02	-7.283766e-03	-5.670422e-03	-4.320866e-03	-3.819065e-03	-6.584468e-03	-2.592262e-03	-1.947171e-03	-1.07
75%	3.807591e-02	5.068012e-02	3.124802e-02	3.564379e-02	2.835801e-02	2.984439e-02	2.931150e-02	3.430886e-02	3.243232e-02	2.7917
max	1.107267e-01	5.068012e-02	1.705552e-01	1.320436e-01	1.539137e-01	1.987880e-01	1.811791e-01	1.852344e-01	1.335973e-01	1.3561

In [11]:

```
df.dropna()
```

Out[11]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641
...	...	...	...	...	...	...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.031193	0.007207
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.018114	0.044485
439	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993	-0.011080	-0.046883	0.015491
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.044529	-0.025930
441	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816	-0.039493	-0.004222	0.003064

442 rows × 10 columns

In [12]:

```
# creating one hot encoding of the categorical columns.
file = pd.get_dummies(df, columns = ['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6'] )
```

In [13]:

```
file.head()
```

Out[13]:

	age_-0.1072256316073538	age_-0.10359309315633439	age_-0.09996055470531495	age_-0.09632801625429555	age_-0.09269547780327612	age_-0.
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	

5 rows × 1135 columns

In [14]:

```
file.columns
```

Out[14]:

```
Index(['age_-0.1072256316073538', 'age_-0.10359309315633439',
       'age_-0.09996055470531495', 'age_-0.09632801625429555',
       'age_-0.09269547780327612', 'age_-0.0890629393522567',
       'age_-0.08543040090123728', 'age_-0.08179786245021785',
       'age_-0.07816532399919843', 'age_-0.074532785548179',
       ...
       's6_0.08176444079622315', 's6_0.0859065477110576',
       's6_0.09004865462589207', 's6_0.09419076154072652',
       's6_0.09833286845556097', 's6_0.1066170822852299',
       's6_0.11904340302973325', 's6_0.12732761685940217',
       's6_0.13146972377423663', 's6_0.13561183068907107'],
      dtype='object', length=1135)
```

In [15]:

```
file.columns[35]
```

Out[15]:

```
'age_0.01991321417832592'
```

In [16]:

```
 X = file.iloc[:,1:]
```

In [17]:

```
X
```

Out[17]:

	age_-0.10359309315633439	age_-0.09996055470531495	age_-0.09632801625429555	age_-0.09269547780327612	age_-0.0890629393522567	age_
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	1	
4	0	0	0	0	0	
...	...	...	...	...	...	
437	0	0	0	0	0	
438	0	0	0	0	0	
439	0	0	0	0	0	
440	0	0	0	0	0	
441	0	0	0	0	0	

442 rows × 1134 columns

In [18]:

```
Y = file.iloc[:,0]
```

In [19]:

```
Y
```

Out[19]:

```
0      0
1      0
2      0
3      0
4      0
      ..
437    0
438    0
439    0
440    0
441    0
Name: age_-0.1072256316073538, Length: 442, dtype: uint8
```

In [20]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=0)
```

In [21]:

```
X_train, X_test, Y_train, Y_test
```

Out[21]:

```
(     age_-0.10359309315633439  age_-0.09996055470531495  \
 20                          0                         0
 353                         0                         0
 281                         0                         0
 14                          0                         0
 300                         0                         0
 ..                        ...                       ...
 323                         0                         0
 192                         0                         0
 117                         0                         0
 47                          0                         0
 172                         0                         0

     age_-0.09632801625429555  age_-0.09269547780327612  \
 20                          0                         0
 353                         0                         0
 281                         0                         1
 14                          0                         0
```

In [22]:

```
classifier = LogisticRegression(solver='lbfgs',random_state=0)
```

In [23]:

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

Out[23]:

```
LogisticRegression(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:

```
predicted_y = classifier.predict(X_test)
```

In [25]:

```
predicted_y
```

Out[25]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0], dtype=uint8)
```

In [27]:

```
print('Accuracy: {:.2f}'.format(classifier.score(X_test, Y_test)))
```

```
Accuracy: 0.98
```

In [28]:

```
####
```

ANN TO MODEL

In [29]:

```
X_train.shape
```

Out[29]:

```
(331, 1134)
```

In [30]:

```
X_test.shape
```

Out[30]:

```
(111, 1134)
```

In [31]:

```
Y_train.shape
```

Out[31]:

```
(331,)
```

In [32]:

```
Y_test.shape
```

Out[32]:

```
(111,)
```

In [34]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
#fit_transform() - calculates mean and standard dev of each column - (x-mean)/std
X_train_scaled = scaler.fit_transform(X_train)
#transform() - directly applies (x-mean)/std
X_test_scaled = scaler.transform(X_test)
```

In [35]:

```
X_train_scaled.shape
```

Out[35]:

```
(331, 1134)
```

In [36]:

```
X_test_scaled.shape
```

Out[36]:

```
(111, 1134)
```

In [38]:

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

In [39]:

```
diabetesANN =  Sequential()
```

In [40]:

```
#hidden layer
diabetesANN.add(Dense(units=250, activation= 'relu'))
diabetesANN.add(Dense(units=750, activation='relu'))
diabetesANN.add(Dense(units=1, activation='relu'))
```

In [41]:

```
diabetesANN.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_absolute_percentage_error'])
```

In [42]:

```
history = diabetesANN.fit(X_train,Y_train, epochs=50)
```

```
Epoch 1/50
11/11 [==============================] - 2s 16ms/step - loss: 0.0031 - mean_absolute_percentage_error: 1913597.6250
Epoch 2/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 3/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 4/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 5/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 6/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 7/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 8/50
11/11 [==============================] - 0s 16ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 9/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 10/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 11/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 12/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 13/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 14/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 15/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 16/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 17/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 18/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 19/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 20/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 21/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 22/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 23/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 24/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 25/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 26/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 27/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 28/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 29/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 30/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 31/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 32/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 33/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 34/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 35/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 36/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 37/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 38/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 39/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 40/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 41/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 42/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 43/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
```

```
Epoch 44/50
11/11 [==============================] - 0s 17ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 45/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 46/50
11/11 [==============================] - 0s 14ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 47/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 48/50
11/11 [==============================] - 0s 15ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 49/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
Epoch 50/50
11/11 [==============================] - 0s 13ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
```

In [43]:

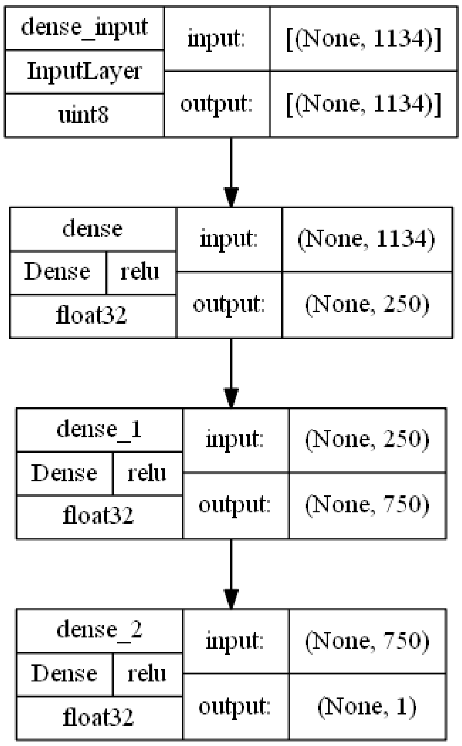
```
diabetesANN.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 250)               283750

 dense_1 (Dense)             (None, 750)               188250

 dense_2 (Dense)             (None, 1)                 751

=================================================================
Total params: 472,751
Trainable params: 472,751
Non-trainable params: 0
_________________________________________________________________
```

In [44]:

```
from tensorflow.keras.utils import plot_model
plot_model(diabetesANN, show_shapes=True, show_dtype=True, show_layer_activations=True, show_layer_names=True)
```

Out[44]:



In [46]:

```
diabetesANN.evaluate(X_train,Y_train)
```

```
11/11 [==============================] - 0s 5ms/step - loss: 0.0030 - mean_absolute_percentage_error: 0.3021
```

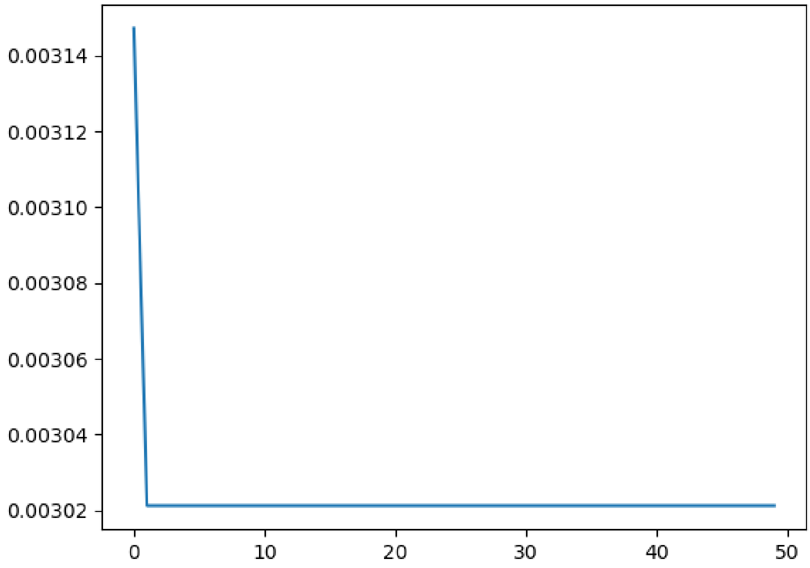
Out[46]:

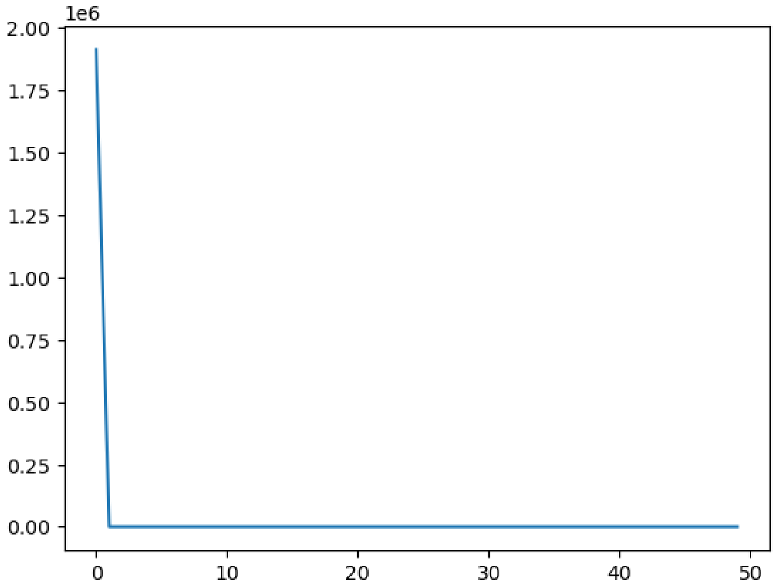
```
[0.003021148033440113, 0.3021148145198822]
```

Plotting graph

In [49]:

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.show()
plt.plot(history.history['mean_absolute_percentage_error'])
plt.show()
```





In [50]:

```
#ashwinikaldate1111.gmail.com
```

change the no of units in first layer:-

In [1]:

```
from sklearn.datasets import load_diabetes
X, y = load_diabetes(return_X_y= True, as_frame=True)
```

In [2]:

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y , test_size=0.1, random_state=22, shuffle=True)
```

ANN

In [3]:

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

In [4]:

```
diabetesANN = Sequential()
```

In [5]:

```
#hidden layer
diabetesANN.add(Dense(units=250, activation= 'relu'))
#output layer
diabetesANN.add(Dense(units=1, activation='relu'))
```

In [6]:

```
diabetesANN.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_absolute_percentage_error'])
```

In [7]:

```
history = diabetesANN.fit(Xtrain,ytrain, epochs=50)
```

```
Epoch 1/50
13/13 [==============================] - 2s 6ms/step - loss: 28982.3906 - mean_absolute_percentage_error: 99.9300
Epoch 2/50
13/13 [==============================] - 0s 4ms/step - loss: 28920.2324 - mean_absolute_percentage_error: 99.7569
Epoch 3/50
13/13 [==============================] - 0s 4ms/step - loss: 28834.2793 - mean_absolute_percentage_error: 99.5238
Epoch 4/50
13/13 [==============================] - 0s 5ms/step - loss: 28718.3203 - mean_absolute_percentage_error: 99.2071
Epoch 5/50
13/13 [==============================] - 0s 4ms/step - loss: 28566.5723 - mean_absolute_percentage_error: 98.8129
Epoch 6/50
13/13 [==============================] - 0s 4ms/step - loss: 28380.0488 - mean_absolute_percentage_error: 98.3135
Epoch 7/50
13/13 [==============================] - 0s 4ms/step - loss: 28155.6777 - mean_absolute_percentage_error: 97.6949
Epoch 8/50
13/13 [==============================] - 0s 5ms/step - loss: 27885.5195 - mean_absolute_percentage_error: 96.9799
Epoch 9/50
13/13 [==============================] - 0s 5ms/step - loss: 27579.3105 - mean_absolute_percentage_error: 96.1413
Epoch 10/50
13/13 [==============================] - 0s 6ms/step - loss: 27231.1621 - mean_absolute_percentage_error: 95.1991
Epoch 11/50
13/13 [==============================] - 0s 5ms/step - loss: 26845.2539 - mean_absolute_percentage_error: 94.1426
Epoch 12/50
13/13 [==============================] - 0s 4ms/step - loss: 26424.1543 - mean_absolute_percentage_error: 92.9719
Epoch 13/50
13/13 [==============================] - 0s 5ms/step - loss: 25972.4316 - mean_absolute_percentage_error: 91.6735
Epoch 14/50
13/13 [==============================] - 0s 3ms/step - loss: 25481.4785 - mean_absolute_percentage_error: 90.2858
Epoch 15/50
13/13 [==============================] - 0s 4ms/step - loss: 24955.2637 - mean_absolute_percentage_error: 88.8024
Epoch 16/50
13/13 [==============================] - 0s 4ms/step - loss: 24409.0508 - mean_absolute_percentage_error: 87.1651
Epoch 17/50
13/13 [==============================] - 0s 4ms/step - loss: 23830.6191 - mean_absolute_percentage_error: 85.4676
Epoch 18/50
13/13 [==============================] - 0s 4ms/step - loss: 23232.7109 - mean_absolute_percentage_error: 83.6834
Epoch 19/50
13/13 [==============================] - 0s 4ms/step - loss: 22609.4336 - mean_absolute_percentage_error: 81.7906
Epoch 20/50
13/13 [==============================] - 0s 5ms/step - loss: 21962.8203 - mean_absolute_percentage_error: 79.8610
Epoch 21/50
13/13 [==============================] - 0s 4ms/step - loss: 21305.3496 - mean_absolute_percentage_error: 77.7807
Epoch 22/50
13/13 [==============================] - 0s 5ms/step - loss: 20627.9707 - mean_absolute_percentage_error: 75.6256
Epoch 23/50
13/13 [==============================] - 0s 4ms/step - loss: 19930.0508 - mean_absolute_percentage_error: 73.3655
Epoch 24/50
13/13 [==============================] - 0s 6ms/step - loss: 19223.6230 - mean_absolute_percentage_error: 71.1230
Epoch 25/50
13/13 [==============================] - 0s 4ms/step - loss: 18520.5234 - mean_absolute_percentage_error: 68.7231
Epoch 26/50
13/13 [==============================] - 0s 4ms/step - loss: 17800.7246 - mean_absolute_percentage_error: 66.2888
Epoch 27/50
13/13 [==============================] - 0s 6ms/step - loss: 17085.7598 - mean_absolute_percentage_error: 63.8511
Epoch 28/50
13/13 [==============================] - 0s 4ms/step - loss: 16368.8779 - mean_absolute_percentage_error: 61.3942
Epoch 29/50
13/13 [==============================] - 0s 5ms/step - loss: 15671.1582 - mean_absolute_percentage_error: 59.0044
Epoch 30/50
13/13 [==============================] - 0s 4ms/step - loss: 14988.0166 - mean_absolute_percentage_error: 56.6219
Epoch 31/50
13/13 [==============================] - 0s 4ms/step - loss: 14296.5117 - mean_absolute_percentage_error: 54.4388
Epoch 32/50
13/13 [==============================] - 0s 4ms/step - loss: 13634.4355 - mean_absolute_percentage_error: 52.3743
Epoch 33/50
13/13 [==============================] - 0s 6ms/step - loss: 12968.5996 - mean_absolute_percentage_error: 50.6394
Epoch 34/50
13/13 [==============================] - 0s 4ms/step - loss: 12351.6914 - mean_absolute_percentage_error: 48.9257
Epoch 35/50
13/13 [==============================] - 0s 4ms/step - loss: 11744.9600 - mean_absolute_percentage_error: 47.3655
Epoch 36/50
13/13 [==============================] - 0s 5ms/step - loss: 11151.0039 - mean_absolute_percentage_error: 46.0117
Epoch 37/50
13/13 [==============================] - 0s 5ms/step - loss: 10585.8877 - mean_absolute_percentage_error: 44.7901
Epoch 38/50
13/13 [==============================] - 0s 5ms/step - loss: 10041.4639 - mean_absolute_percentage_error: 43.5912
Epoch 39/50
13/13 [==============================] - 0s 4ms/step - loss: 9512.1309 - mean_absolute_percentage_error: 42.6515
Epoch 40/50
13/13 [==============================] - 0s 4ms/step - loss: 9028.1670 - mean_absolute_percentage_error: 41.8068
Epoch 41/50
13/13 [==============================] - 0s 4ms/step - loss: 8553.0654 - mean_absolute_percentage_error: 41.1578
Epoch 42/50
13/13 [==============================] - 0s 4ms/step - loss: 8116.7568 - mean_absolute_percentage_error: 40.6130
Epoch 43/50
13/13 [==============================] - 0s 5ms/step - loss: 7688.8931 - mean_absolute_percentage_error: 40.1459
```

```
Epoch 44/50
13/13 [==============================] - 0s 6ms/step - loss: 7309.8110 - mean_absolute_percentage_error: 39.8516
Epoch 45/50
13/13 [==============================] - 0s 4ms/step - loss: 6948.8022 - mean_absolute_percentage_error: 39.5848
Epoch 46/50
13/13 [==============================] - 0s 5ms/step - loss: 6622.7041 - mean_absolute_percentage_error: 39.5372
Epoch 47/50
13/13 [==============================] - 0s 9ms/step - loss: 6313.2393 - mean_absolute_percentage_error: 39.6687
Epoch 48/50
13/13 [==============================] - 0s 3ms/step - loss: 6023.1421 - mean_absolute_percentage_error: 39.6877
Epoch 49/50
13/13 [==============================] - 0s 3ms/step - loss: 5769.0928 - mean_absolute_percentage_error: 39.9438
Epoch 50/50
13/13 [==============================] - 0s 4ms/step - loss: 5537.5938 - mean_absolute_percentage_error: 40.2159
```

In [8]:

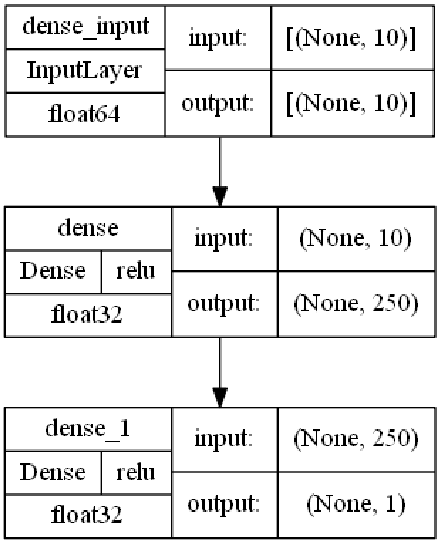
```
diabetesANN.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 250)               2750

 dense_1 (Dense)             (None, 1)                 251

=================================================================
Total params: 3,001
Trainable params: 3,001
Non-trainable params: 0
_________________________________________________________________
```

In [9]:

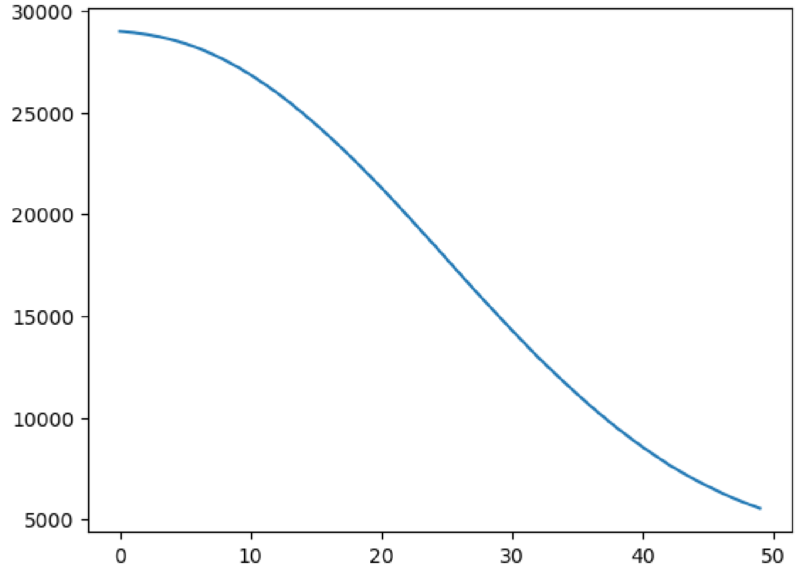
```
from tensorflow.keras.utils import plot_model
plot_model(diabetesANN, show_shapes=True, show_dtype=True, show_layer_activations=True, show_layer_names=True)
```

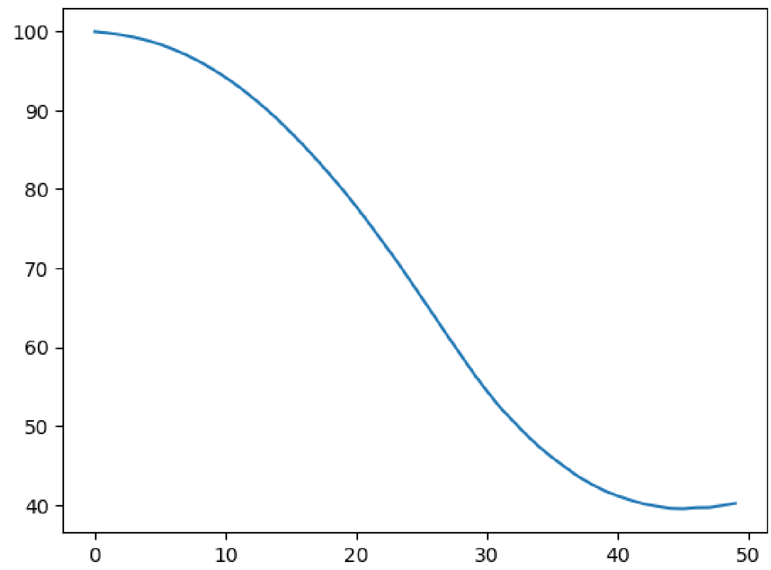
Out[9]:



In [10]:

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.show()
plt.plot(history.history['mean_absolute_percentage_error'])
plt.show()
```





In [11]:

```
diabetesANN.evaluate(Xtrain,ytrain)
```

```
13/13 [==============================] - 0s 2ms/step - loss: 5409.9136 - mean_absolute_percentage_error: 40.3506
```

Out[11]:

```
[5409.91357421875, 40.3505744934082]
```

In [12]:

```
###############
```

ANN3 :-

In [36]:

```
from sklearn.datasets import load_diabetes
X, y = load_diabetes(return_X_y= True, as_frame=True)
```

In [37]:

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y , test_size=0.1, random_state=22, shuffle=True)
```

In [38]:

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

In [39]:

```
diabetesANN = Sequential()
```

In [40]:

```
#hidden layer
diabetesANN.add(Dense(units=88, activation= 'relu'))

diabetesANN.add(Dense(units=1, activation='relu'))
```

In [41]:

```
diabetesANN.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_absolute_percentage_error'])
```

In [42]:

```
history = diabetesANN.fit(Xtrain,ytrain, epochs=50)
```

```
Epoch 1/50
13/13 [==============================] - 1s 5ms/step - loss: 28989.1465 - mean_absolute_percentage_error: 99.9525
Epoch 2/50
13/13 [==============================] - 0s 3ms/step - loss: 28955.8359 - mean_absolute_percentage_error: 99.8561
Epoch 3/50
13/13 [==============================] - 0s 3ms/step - loss: 28915.3066 - mean_absolute_percentage_error: 99.7395
Epoch 4/50
13/13 [==============================] - 0s 3ms/step - loss: 28861.0488 - mean_absolute_percentage_error: 99.5995
Epoch 5/50
13/13 [==============================] - 0s 4ms/step - loss: 28794.4980 - mean_absolute_percentage_error: 99.4222
Epoch 6/50
13/13 [==============================] - 0s 4ms/step - loss: 28715.4512 - mean_absolute_percentage_error: 99.2024
Epoch 7/50
13/13 [==============================] - 0s 4ms/step - loss: 28621.0762 - mean_absolute_percentage_error: 98.9522
Epoch 8/50
13/13 [==============================] - 0s 4ms/step - loss: 28513.8672 - mean_absolute_percentage_error: 98.6606
Epoch 9/50
13/13 [==============================] - 0s 5ms/step - loss: 28391.6113 - mean_absolute_percentage_error: 98.3364
Epoch 10/50
13/13 [==============================] - 0s 6ms/step - loss: 28257.4785 - mean_absolute_percentage_error: 97.9696
Epoch 11/50
13/13 [==============================] - 0s 4ms/step - loss: 28110.4414 - mean_absolute_percentage_error: 97.5549
Epoch 12/50
13/13 [==============================] - 0s 6ms/step - loss: 27943.8887 - mean_absolute_percentage_error: 97.1317
Epoch 13/50
13/13 [==============================] - 0s 5ms/step - loss: 27771.2402 - mean_absolute_percentage_error: 96.6494
Epoch 14/50
13/13 [==============================] - 0s 5ms/step - loss: 27579.8613 - mean_absolute_percentage_error: 96.1389
Epoch 15/50
13/13 [==============================] - 0s 4ms/step - loss: 27374.7891 - mean_absolute_percentage_error: 95.5950
Epoch 16/50
13/13 [==============================] - 0s 6ms/step - loss: 27159.7852 - mean_absolute_percentage_error: 95.0019
Epoch 17/50
13/13 [==============================] - 0s 4ms/step - loss: 26930.5137 - mean_absolute_percentage_error: 94.3610
Epoch 18/50
13/13 [==============================] - 0s 4ms/step - loss: 26691.9141 - mean_absolute_percentage_error: 93.6886
Epoch 19/50
13/13 [==============================] - 0s 4ms/step - loss: 26438.9824 - mean_absolute_percentage_error: 92.9974
Epoch 20/50
13/13 [==============================] - 0s 4ms/step - loss: 26177.0801 - mean_absolute_percentage_error: 92.2691
Epoch 21/50
13/13 [==============================] - 0s 5ms/step - loss: 25901.4844 - mean_absolute_percentage_error: 91.4942
Epoch 22/50
13/13 [==============================] - 0s 5ms/step - loss: 25619.4961 - mean_absolute_percentage_error: 90.6783
Epoch 23/50
13/13 [==============================] - 0s 4ms/step - loss: 25317.3555 - mean_absolute_percentage_error: 89.8518
Epoch 24/50
13/13 [==============================] - 0s 4ms/step - loss: 25013.8711 - mean_absolute_percentage_error: 88.9478
Epoch 25/50
13/13 [==============================] - 0s 3ms/step - loss: 24695.0352 - mean_absolute_percentage_error: 88.0257
Epoch 26/50
13/13 [==============================] - 0s 5ms/step - loss: 24367.2188 - mean_absolute_percentage_error: 87.0777
Epoch 27/50
13/13 [==============================] - 0s 5ms/step - loss: 24027.2324 - mean_absolute_percentage_error: 86.1092
Epoch 28/50
13/13 [==============================] - 0s 4ms/step - loss: 23683.6641 - mean_absolute_percentage_error: 85.0934
Epoch 29/50
13/13 [==============================] - 0s 4ms/step - loss: 23326.2422 - mean_absolute_percentage_error: 84.0367
Epoch 30/50
13/13 [==============================] - 0s 4ms/step - loss: 22964.0371 - mean_absolute_percentage_error: 82.9307
Epoch 31/50
13/13 [==============================] - 0s 5ms/step - loss: 22595.4629 - mean_absolute_percentage_error: 81.7898
Epoch 32/50
13/13 [==============================] - 0s 5ms/step - loss: 22218.4238 - mean_absolute_percentage_error: 80.6586
Epoch 33/50
13/13 [==============================] - 0s 5ms/step - loss: 21841.1641 - mean_absolute_percentage_error: 79.4739
Epoch 34/50
13/13 [==============================] - 0s 4ms/step - loss: 21451.9648 - mean_absolute_percentage_error: 78.3114
Epoch 35/50
13/13 [==============================] - 0s 6ms/step - loss: 21072.4062 - mean_absolute_percentage_error: 77.0622
Epoch 36/50
13/13 [==============================] - 0s 4ms/step - loss: 20677.0723 - mean_absolute_percentage_error: 75.8312
Epoch 37/50
13/13 [==============================] - 0s 5ms/step - loss: 20285.6152 - mean_absolute_percentage_error: 74.5818
Epoch 38/50
13/13 [==============================] - 0s 5ms/step - loss: 19884.3867 - mean_absolute_percentage_error: 73.2862
Epoch 39/50
13/13 [==============================] - 0s 4ms/step - loss: 19477.6406 - mean_absolute_percentage_error: 72.0002
Epoch 40/50
13/13 [==============================] - 0s 3ms/step - loss: 19073.9688 - mean_absolute_percentage_error: 70.6847
Epoch 41/50
13/13 [==============================] - 0s 3ms/step - loss: 18670.8320 - mean_absolute_percentage_error: 69.3259
Epoch 42/50
13/13 [==============================] - 0s 4ms/step - loss: 18272.1445 - mean_absolute_percentage_error: 67.9519
Epoch 43/50
13/13 [==============================] - 0s 4ms/step - loss: 17865.3047 - mean_absolute_percentage_error: 66.5838
```

```
Epoch 44/50
13/13 [==============================] - 0s 4ms/step - loss: 17465.3398 - mean_absolute_percentage_error: 65.1933
Epoch 45/50
13/13 [==============================] - 0s 5ms/step - loss: 17055.6973 - mean_absolute_percentage_error: 63.8429
Epoch 46/50
13/13 [==============================] - 0s 3ms/step - loss: 16662.9902 - mean_absolute_percentage_error: 62.4320
Epoch 47/50
13/13 [==============================] - 0s 5ms/step - loss: 16259.4707 - mean_absolute_percentage_error: 61.0833
Epoch 48/50
13/13 [==============================] - 0s 4ms/step - loss: 15866.3730 - mean_absolute_percentage_error: 59.7065
Epoch 49/50
13/13 [==============================] - 0s 2ms/step - loss: 15477.0508 - mean_absolute_percentage_error: 58.3449
Epoch 50/50
13/13 [==============================] - 0s 4ms/step - loss: 15082.5312 - mean_absolute_percentage_error: 57.0606
```

In [43]:

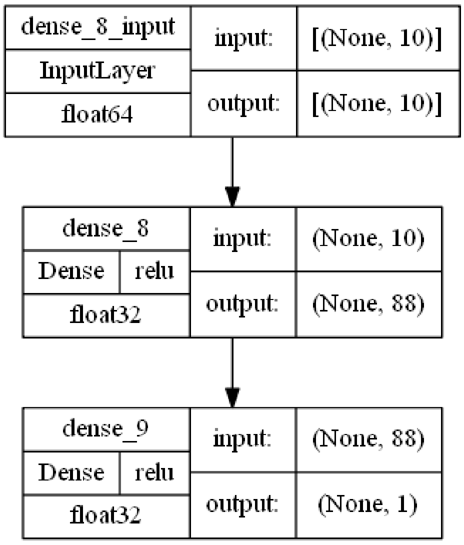
```
diabetesANN.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 88)                968

 dense_9 (Dense)             (None, 1)                 89

=================================================================
Total params: 1,057
Trainable params: 1,057
Non-trainable params: 0
_________________________________________________________________
```

In [44]:

```
from tensorflow.keras.utils import plot_model
plot_model(diabetesANN, show_shapes=True, show_dtype=True, show_layer_activations=True, show_layer_names=True)
```

Out[44]:



In [45]:

```
diabetesANN.evaluate(Xtrain,ytrain)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 14867.2490 - mean_absolute_percentage_error: 56.3452
```

Out[45]:

```
[14867.2490234375, 56.34523010253906]
```

In [ ]: