3. Implement a regression based supervised learning model for a data set of your choice, through a Multi Layer Perceptron.

```
import warnings
warnings.simplefilter("ignore")
```

```
from __future__ import print_function
from keras.models import load_model
import keras
from keras.utils import np_utils
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from sklearn.linear_model import LogisticRegressionCV
from keras.layers.core import Activation
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import RMSprop
import numpy as np
```

```
iris = load_iris()
X, y = iris.data[:, :4], iris.target
# Split both independent and dependent variables in half for cross-validation
train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.5, random_state=0)
#print(type(train_X),len(train_y),len(test_X),len(test_y))
lr = LogisticRegressionCV()
lr.fit(train_X, train_y)
pred_y = lr.predict(test_X)
print("Test fraction correct (LR-Accuracy) = {:.2f}".format(lr.score(test_X, test_y)))
```

```
Test fraction correct (LR-Accuracy) = 0.93
```

```
def one_hot_encode_object_array(arr):
    uniques, ids = np.unique(arr, return_inverse=True)
```

```
    return np_utils.to_categorical(ids, len(uniques))


# Dividing data into train and test data
train_y_ohe = one_hot_encode_object_array(train_y)
test_y_ohe = one_hot_encode_object_array(test_y)


#Creating a model
model = Sequential()
model.add(Dense(16, input_shape=(4,)))
model.add(Activation('sigmoid'))
model.add(Dense(3))
model.add(Activation('softmax'))


# Compiling the model
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')


# Actual modelling
model.fit(train_X, train_y_ohe, verbose=0, batch_size=1, nb_epoch=100)

score, accuracy = model.evaluate(test_X, test_y_ohe, batch_size=16, verbose=0)

print("\n Test fraction correct (LR-Accuracy) logistic regression = {:.2f}".format(lr.score(test_X, test_y)))
print("Test fraction correct (NN-Accuracy) keras  = {:.2f}".format(accuracy))
```

```
 Test fraction correct (LR-Accuracy) logistic regression = 0.93
Test fraction correct (NN-Accuracy) keras  = 0.97
```