Master's Thesis

Double Master's degree in Industrial Engineering and Organization Engineering

Data analysis of the stock management of a manufacturing company

ANNEX

Author: Marc Gisbert Juárez
Director: Ernest Benedito Benet

Summons: January 2022

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Annex: code to perform a neural network

Below is the code used to perform all the neural networks of the project with all its steps explained. The programming language used is Python.

```
# load libraries
from pandas import read_csv
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import timeit

# load dataset
dataframe = read_csv("Training data.txt", delim_whitespace=True, header=None)
dataset = dataframe.values

# split into input (X) and output (Y) variables the training data
# NV = number of variables
NV = 4
X = dataset[:,0:NV]
Y = dataset[:,NV]

# training data normalization
sc = StandardScaler()
X = sc.fit_transform(X)

# load dataset
dataframe_cro_val = read_csv("Cross-validation data.txt", delim_whitespace=True,
header=None)
dataset_cro_val = dataframe_cro_val.values

# split into input (X) and output (Y) variables the cross-validation data
X_cro_val = dataset_cro_val[:,0:NV]
Y_cro_val = dataset_cro_val[:,NV]
```

```
# cross-validation data normalization
sc_cro_val = StandardScaler()
X_cro_val = sc_cro_val.fit_transform(X_cro_val)

# define base model
# ep = number of epochs of the neural network
# bs = batch size of the neural network
# hl = number of layers of the neural network
# ne = number of neurons of each layer
def baseline_model(ep,bs,hl,ne):
    initial_time=timeit.default_timer()

    # create model
    model = Sequential()
    model.add(Dense(ne, input_dim=NV, kernel_initializer='normal', activation='relu'))

    #add hidden layers
    i=1
    while i<hl:
        model.add(Dense(ne, kernel_initializer='normal', activation='relu'))
        i=i+1

    model.add(Dense(1, kernel_initializer='normal'))

    # Compile model
    model.compile(loss='mean_squared_error', optimizer='adam',metrics=['accuracy'])

    # fit the model on the dataset
    model.fit(X, Y, epochs=ep, batch_size=bs,verbose=0)

    # output calculation
    y_pred=model.predict(X)

    # evaluate model
    error,accuracy = model.evaluate(X,Y,verbose=0)

    # predict output of cross-validation data
    y_pred_cro_val=model.predict(X_cro_val)
```

```
# calculate average prediction error
ape=0
for p in range(len(y_pred_cro_val)):
    rel_error=abs((y_pred_cro_val[p][0]-Y_cro_val[p])*100/(Y_cro_val[p]))
    ape=ape+rel_error
ape=ape/len(Y_cro_val)

# writing predictions in a file
f=open('predictions.txt','w')
for k in y_pred_cro_val:
    f.write(str(k[0])+'\n')
f.close()

error_cro_val,accuracy_cro_val = model.evaluate(X_cro_val,Y_cro_val,verbose=0)

final_time=timeit.default_timer()
time=final_time-initial_time

# plot training data results
plt.plot(Y, color = 'red', label = 'Real data')
plt.plot(y_pred, color = 'blue', label = 'Predicted data')
plt.title('Prediction training dataset')
plt.legend()
plt.show()

# plot cross-validation data results
plt.plot(Y_cro_val, color = 'red', label = 'Real data')
plt.plot(y_pred_cro_val, color = 'blue', label = 'Predicted data')
plt.title('Prediction cross validation dataset')
plt.legend()
plt.show()

return error,ape,time
```