Ajuste de redes neuronales

Ejercicio 1

In [5]:
```
#Import libraries

import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.metrics import mean_squared_error, mean_absolute_error,class
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV, cross_val_predict
```

In [24]:
```
df = pd.read_csv('/home/alanv/Documents/7/omar/crime_data.csv')
df.drop(['State','MR','S'],inplace=True,axis=1)
# Dependiente es VR
df.head(5)
```

Out[24]:

	VR	M	W	H	P
0	761	41.8	75.2	86.6	9.1
1	780	67.4	73.5	66.9	17.4
2	593	44.7	82.9	66.3	20.0
3	715	84.7	88.6	78.7	15.4
4	1078	96.7	79.3	76.2	18.2

In [25]:
```
#Declare data
x = df.iloc[:, 1:].values
y = df.iloc[:,0].values
```

1. Evalúa con validación cruzada un modelo pereceptrón multicapa para las variables que se te asignaron para este ejercicio.

In [26]:
```
#   Multilayer perceptron

# Fit regression model using the complete dataset and calculate MSE and M
regr = MLPRegressor(hidden_layer_sizes=(20, 20), max_iter=20000)
regr.fit(x, y)

y_pred = regr.predict(x)
print('MSE: ', mean_squared_error(y, y_pred))
print("MAE: ", mean_absolute_error(y, y_pred))

# Evaluate regression model using k-fold cross-validation
n_folds = 5
kf = KFold(n_splits=n_folds, shuffle = True)
```

```
mse_cv = []
mae_cv = []
for train_index, test_index in kf.split(x):

    # Training phase
    x_train = x[train_index, :]
    y_train = y[train_index]

    regr_cv = MLPRegressor(hidden_layer_sizes=(20, 20), max_iter=20000)
    regr_cv.fit(x_train, y_train)

    # Test phase
    x_test = x[test_index, :]
    y_test = y[test_index]

    y_pred = regr_cv.predict(x_test)

    # Calculate MSE and MAE

    mse_i = mean_squared_error(y_test, y_pred)
    print('mse = ', mse_i)
    mse_cv.append(mse_i)

    mae_i = mean_absolute_error(y_test, y_pred)
    print('mae = ', mae_i)
    mae_cv.append(mae_i)


print('MSE:', np.average(mse_cv), '  MAE:', np.average(mae_cv))
```

```
MSE:  11017.329011130674
MAE:  74.44618494694208
mse =  501532.82211680186
mae =  326.85163386317384
mse =  201075.60967884207
mae =  283.4168814200769
mse =  36654.09200584439
mae =  165.92298617690136
mse =  92688.12342096666
mae =  223.7341728290521
mse =  22014.07734081662
mae =  118.1156326197759
MSE: 170792.9449126543   MAE: 223.608261381796
```

2. Agrega al conjunto de datos columnas que representen los cuadrados de las variables predictoras (por ejemplo, M2, W2), así como los productos entre pares de variables (por ejemplo, PxS, MxW). Evalúa un modelo perceptrón multicapa para este nuevo conjunto de datos.

In [27]:
```
#x2 = np.array([df['M'],df['W'],df['H'],df['P'],df['P']**2, df['H']**3,df
x2 = np.array([df['M'],df['W'],df['H'],df['P'],df['P']**2, df['H']**2,df[
y2 = np.array(df['VR'])
```

In [28]:
```
#   Multilayer perceptron square dataset

# Fit regression model using the complete dataset and calculate MSE and M
regr = MLPRegressor(hidden_layer_sizes=(20, 20), max_iter=20000)
```

```
regr.fit(x2, y2)

y_pred = regr.predict(x2)
print('MSE: ', mean_squared_error(y2, y_pred))
print("MAE: ", mean_absolute_error(y2, y_pred))

# Evaluate regression model using k-fold cross-validation
n_folds = 5
kf = KFold(n_splits=n_folds, shuffle = True)

mse_cv = []
mae_cv = []
for train_index, test_index in kf.split(x2):

    # Training phase
    x_train = x2[train_index, :]
    y_train = y2[train_index]

    regr_cv = MLPRegressor(hidden_layer_sizes=(20, 20), max_iter=20000)
    regr_cv.fit(x_train, y_train)

    # Test phase
    x_test = x2[test_index, :]
    y_test = y2[test_index]

    y_pred = regr_cv.predict(x_test)

    # Calculate MSE and MAE

    mse_i = mean_squared_error(y_test, y_pred)
    print('mse = ', mse_i)
    mse_cv.append(mse_i)

    mae_i = mean_absolute_error(y_test, y_pred)
    print('mae = ', mae_i)
    mae_cv.append(mae_i)


print('MSE:', np.average(mse_cv), '  MAE:', np.average(mae_cv))
```

```
MSE:  126110.93092449554
MAE:  289.0910393419879
mse =  197506.67236286792
mae =  401.07721179535906
mse =  354974.09683260426
mae =  362.71752229054766
mse =  439590.20785016345
mae =  396.71731685927637
mse =  55624.218179279895
mae =  198.31444760152553
mse =  88390.01871564526
mae =  247.65618951167343
MSE: 227217.04278811216   MAE: 321.2965376116764
```

3. Viendo los resultados de regresión, desarrolla una conclusión sobre los siguientes puntos:
    ¿Consideras que el modelo perceptrón multicapa es efectivo para modelar los datos del problema? ¿Por qué?
    ¿Qué modelo es mejor para los datos de criminalidad,

el lineal o el perceptrón multicapa? ¿Por qué?

No creo que sea un buen modelo, principalmente porque me da errores muy grandes, cuando los datos son relativamente pequeños, y lo que creo que sucede es que no hay suficientes datos para generar un buen modelo o los datos no representan una relación con la variable dependiente. Por lo tanto yo creo que ni el modelo lineal o el perecptron muticapas representan de una manera precisa los datos.

Ejercicio 2

In [2]:
```
#Load database
df = pd.read_csv('/home/alanv/Documents/7/omar/M_4.txt', delimiter='\t',
column_names = [f"V{i}" for i in range(1, df.shape[1] + 1)]
df.columns = column_names
df.drop(['V2','V633'],inplace=True,axis=1)
# Initializate x and y
x = df.iloc[:, 1:].values
y = df.iloc[:,0].values
```

1. Evalúa un modelo perceptrón multicapa con validación cruzada utilizando al menos 5 capas de 20 neuronas.

In [33]:
```
kf = StratifiedKFold(n_splits=5, shuffle = True)
cv_y_test = []
cv_y_pred = []
for train_index, test_index in kf.split(x, y):
    # Training phase
    x_train = x[train_index, :]
    y_train = y[train_index]
    clf_cv = MLPClassifier(hidden_layer_sizes=(20, 20, 20, 20, 20), max_i
    clf_cv.fit(x_train, y_train)
    # Test phase
    x_test = x[test_index, :]
    y_test = y[test_index]
    y_pred = clf_cv.predict(x_test)
    cv_y_test.append(y_test)
    cv_y_pred.append(y_pred)
print(classification_report(np.concatenate(cv_y_test), np.concatenate(cv_
```

```
              precision    recall  f1-score   support

           1       0.94      0.92      0.93        90
           2       0.81      0.84      0.83        90
           3       0.92      0.89      0.90        90
           4       0.88      0.91      0.90        90
           5       0.89      0.90      0.90        90
           6       0.93      0.84      0.88        90
           7       0.86      0.91      0.89        90

    accuracy                           0.89       630
   macro avg       0.89      0.89      0.89       630
weighted avg       0.89      0.89      0.89       630
```

2. Evalúa un modelo perceptrón multicapa con validación cruzada, pero encontrando el número óptimo de capas y neuronas de la red.

In [3]:
```
# Optimal number of layers and neurons
#########################################################################
num_layers = np.arange(1, 20, 5)
num_neurons = np.arange(10, 110, 20)
layers = []
for l in num_layers:
    for n in num_neurons:
        layers.append(l*[n])
clf = GridSearchCV(MLPClassifier(max_iter=10000), {'hidden_layer_sizes':
cv = 5)
clf.fit(x, y)
print(clf.best_estimator_)
```

```
MLPClassifier(hidden_layer_sizes=[70, 70, 70, 70, 70, 70], max_iter=10000)
```

3. Prepara el modelo perceptrón multicapa:

Opten los hiperparámetros óptimos de capas y neuronas de la red.
Con los hiperparámetros óptimos, ajusta el modelo con todos los datos.

In [7]:
```
#########################################################################
# Evaluate model finding the optimal number of layers and neurons
#########################################################################
clf = MLPClassifier(hidden_layer_sizes=(70, 70, 70, 70, 70, 70), max_iter
y_pred = cross_val_predict(clf, x, y, cv = 5)
print(classification_report(y, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.97      0.94      0.96        90
           2       0.80      0.88      0.84        90
           3       0.91      0.83      0.87        90
           4       0.88      0.94      0.91        90
           5       0.94      0.94      0.94        90
           6       0.95      0.86      0.90        90
           7       0.90      0.93      0.92        90

    accuracy                           0.90       630
   macro avg       0.91      0.90      0.91       630
weighted avg       0.91      0.90      0.91       630
```

4. Contesta lo siguientes:

¿Observas alguna mejora importante al optimizar el tamaño de la red? ¿Es el resultado que esperabas? Argumenta tu respuesta.

Se observa una mejora, pero no es considerable, aunque era lo que esperaba, ya

que al poner 5 capas de 20 neuronas dio un resultado muy bueno. Por lo tanto, pienso que el modelo no tiene mucho más que mejorar.

¿Qué inconvenientes hay al encontrar el tamaño óptimo de la red? ¿Por qué?

El costo computacional puede llegar a ser muy elevado si se quiere probar cada una de las combinaciones de hiperparámetros. Además, se podría ocasionar un sobreajuste y hacer que el modelo no sea preciso con datos nuevos.