# Práctica 3
# Programación

---

Aprendizaje Automático - GII
Grupo 2
Curso 19-20

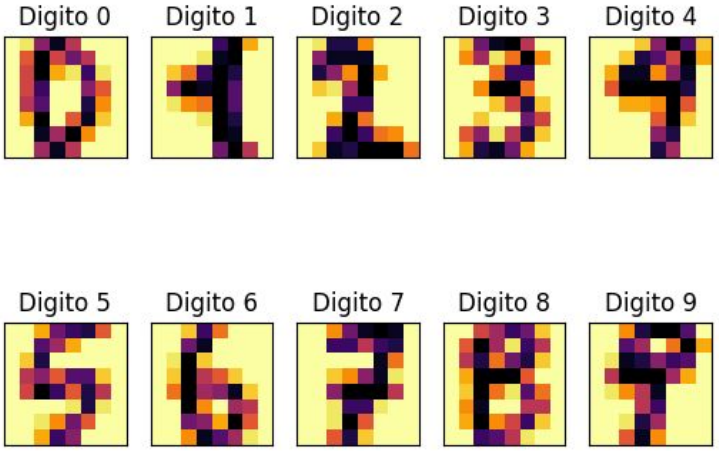
**Universidad de Granada**
Cumbreras Torrente, Paula
49087324-B

# Índice de contenidos

# Optical Recognition of Handwritten Digits

## Comprensión del problema

El problema de clasificación consiste en ajustar un modelo lineal a un conjunto de datos y analizar los pasos para ello. Los datos son un conjunto de dígitos escritos a mano (un mapa de bits) junto a su clasificación (0-9). A continuación se muestra un ejemplo de los mapas de bits mencionados clasificados generados con la función imshow de matplotlib:




Como se muestra en el repositorio de donde se obtiene la base de datos, ésta consta de 5620 instancias con 65 columnas: 64 de ellas son atributos numéricos que representan cada uno de los bits de una matriz 8x8 (como se puede observar en la figura anterior) y la última representa la variable de clase, es decir, el dígito. Los datos están completos, no falta información por lo que el problema ha de ser, y han sido previamente divididos en los conjuntos training (aproximadamente un 68% de las instancias) y test (el 32% restante).

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 5620 | **Area:** | Computer |
| **Attribute Characteristics:** | Integer | **Number of Attributes:** | 64 | **Date Donated** | 1998-07-01 |
| **Associated Tasks:** | Classification | **Missing Values?** | No | **Number of Web Hits:** | 271836 |

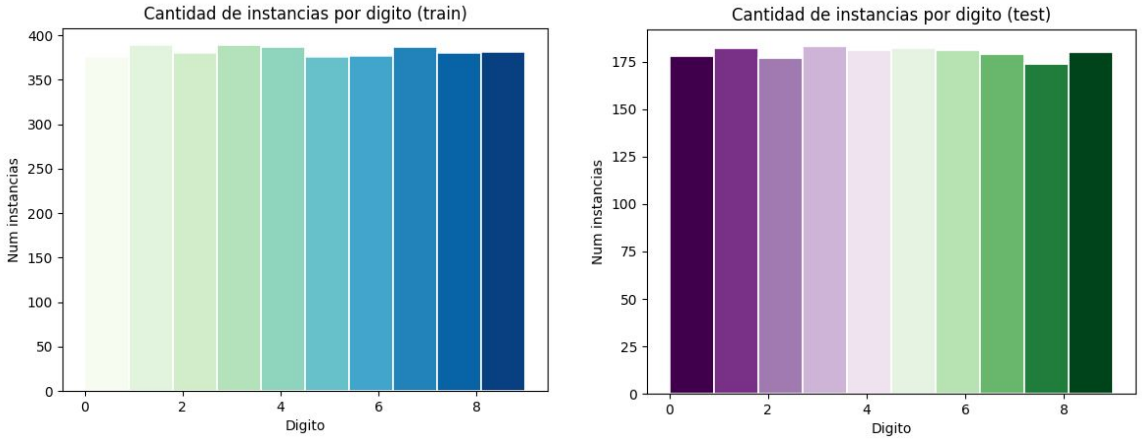
Tabla obtenida de la página web https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits

```
Tamaño train: 3823
Tamaño test: 1797
```

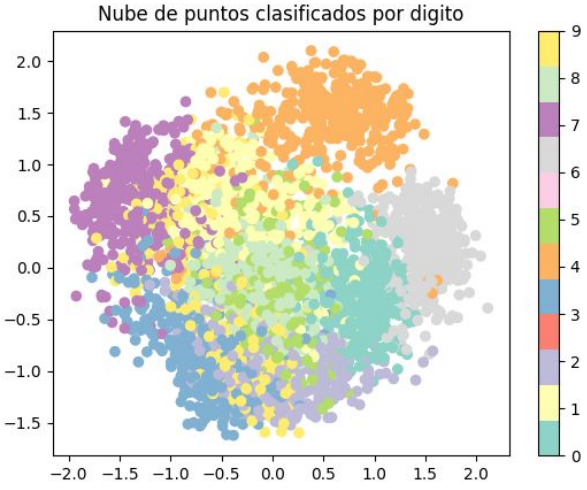
Para separar X e Y, gracias a lo mencionado anteriormente acerca de las columnas solamente ha sido necesario hacer esa distinción: tanto para el conjunto de entrenamiento

como el de prueba, la última columna será nuestra Y o etiqueta y el resto de columnas compondrán X.

A continuación se muestra la distribución de valores de las etiquetas para ambos conjuntos:




Podemos afirmar que se trata de un problema de clasificación y de aprendizaje supervisado, puesto que las clases a las que pertenecen las instancias son conocidas antes de su clasificación.




## Clases de funciones

La clase de funciones empleada para la práctica es la clase de las funciones lineales.

## Training, Test y Validación

Como ya se comentaba anteriormente, los conjuntos Training y Test ya habían sido separados previamente. No se ha considerado necesaria la separación de un conjunto Validación debido a la partición propuesta inicialmente y puesto que al usar GridSearchCV

de sklearn ya se nos proporciona la opción de realizar la validación cruzada sin necesidad de separar los conjuntos.

## Preprocesamiento

Con el fin de obtener mejores resultados, es conveniente reducir el conjunto. Para ello se han empleado dos funciones de sklearn: para reducir las características se ha usado VarianceThreshold, que es un selector de características que elimina aquellas que tengan una baja varianza; para disminuir la complejidad de las características restantes usamos MinMaxScaler que es un estimador que escala las características de forma que estas queden comprendidas en un rango (por defecto [0-1]).

| Preprocesado | Filas | Columnas | Ejemplo_valores |
|---|---|---|---|
| Antes | 3823 | 64 | 16.0 |
| Despues | 3823 | 54 | 0.375 |

Aunque el ejemplo de valores seleccionado antes del preprocesado pueda no corresponder con el que se encuentra en la misma posición tras el mismo debido a la alteración de las columnas, se ha añadido para ilustrar el cambio realizado: los valores inicialmente eran enteros.

## Métrica

A continuación se explican las métricas utilizadas para este problema:

- Confusion Matrix. Consiste en una tabla C en la que cada casilla $C_{ij}$ toma el valor del número de datos clasificados en i y predichos en j, permitiendo destacar dónde se suelen confundir dos clases. Se ha usado la función confusion_matrix de sklearn.
- Accuracy. Es el porcentaje de elementos que han sido clasificados correctamente, tomando el valor de (tp + tn) / (tp + tn + fp + fn). Se ha usado la función accuracy_score de sklearn.
- F1 score. Es el promedio ponderado de accuracy y recall, que es otra métrica que no ha sido incluida en la práctica (tp / (tp + fn)) Se ha usado la función f1_score de sklearn. La fórmula para la puntuación F1 es:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

Pese a haber considerado más de una métrica en este ejercicio, será f1 score aquella a la que le daremos más importancia en caso de obtener valores muy similares con las otras métricas.

## Regularización

La regularización es el método usado para medir la complejidad de un modelo. Seleccionar correctamente la regularización nos permitirá minimizar la función de coste. Trabajaremos con dos tipos:

- Regularización Lasso (‘l1’) mide la complejidad como la media del valor absoluto de los coeficientes del modelo. Esta regularización es especialmente útil cuando consideramos que pueden haber características irrelevantes (pues facilita la diferenciación de los mismos) y cuando los atributos no están fuertemente relacionados.

$$C = \frac{1}{N} \sum_{j=1}^{N} |w_i|$$

- Regularización Ridge (‘l2’) mide la complejidad como la media del cuadrado de los coeficientes del modelo. Será útil cuando los atributos estén fuertemente relacionados entre ellos (pues disminuye la correlación) y cuando todos o la mayoría de los datos sean relevantes.

$$C = \frac{1}{N} \sum_{j=1}^{N} |w_i^2|$$

## Modelos

Para este problema se han usado los siguientes modelos:

- Regresión Logística: se ha usado el modelo LogisticRegression de sklearn.
- Perceptron: se ha usado el modelo Perceptron de sklearn.
- Clasificador SGD: aunque SGD no sea un modelo, sino un método de optimización, sklearn proporciona el modelo SGDClassifier que consiste en un clasificador lineal con entrenamiento SGD. Por defecto, el modelo lineal que utiliza es SVM.

A continuación se muestran los pasos generales para la generación, ajuste y prueba de los diferentes modelos. Para ello usamos GridSearchCV de sklearn, que realiza una búsqueda exhaustiva sobre los valores de los hiperparámetros especificados para encontrar los mejores para un modelo dado.

```
Establecer hiperparametros
Generar modelo
Aplicar GridSearchCV
Volver a generar el modelo, esta vez con los mejores parámetros
Ajustar al conjunto train
Predecir con el modelo x_test
Obtener hit score sobre el conjunto test
```

## Hiperparámetros y selección de modelo

### Regresión Logística

Parámetros del modelo:

- penalty: Se utiliza para especificar la norma utilizada en la regularización.
- C: Inverso de la fuerza de regularización; debe ser un flot positivo.
- tol: El criterio de detención.
- solver: Algoritmo a utilizar en el problema de optimización.

- multi_class: Si la opción elegida es "ovr", entonces se ajusta un problema binario para cada etiqueta. Para "multinomial", la pérdida minimizada es el ajuste de pérdida multinomial en toda la distribución de probabilidad, incluso cuando los datos son binarios. "Auto" selecciona "ovr" o "multinomial".

Puesto que la regularización Lasso no es compatible con el solver ‘newton-cg’ se ha diferenciado, creando dos conjuntos de hiperparámetros. Comenzando con la regularización Lasso:

```
[{'penalty':[reg],'C':[1,10,100,1000,10000], 'tol':[1e-3,1e-4], 'solver':['liblinear'], 'multi_class':['auto']}]
```

El mejor modelo obtenido tiene las siguientes características:

```
Mejores parametros:  {'penalty': 'l1', 'multi_class': 'auto', 'C': 10, 'tol': 0.001, 'solver': 'liblinear'}
Validacion cruzada:  0.9649489929374836
Ein : 0.035051007062516404
```

Hiperparámetros para regularización Ridge:

```
[{'penalty':[reg],'C':[1,10,100,1000,10000], 'tol':[1e-3,1e-4], 'solver':['newton-cg'], 'multi_class':['auto']}]
```

El mejor modelo obtenido tiene las siguientes características:

```
Mejores parametros:, {'penalty': 'l1', 'alpha': 1e-05, 'tol': 0.0001}
Validacion cruzada:, 0.9466387653675125
Ein : 0.05336123463248754
```

**Perceptron**

Parámetros del modelo:

- penalty: Se utiliza para especificar la norma utilizada en la regularización.
- alpha: Constante que multiplica el término de regularización si se utiliza la regularización.
- tol: El criterio de detención.

Hiperparámetros:

```
[{'penalty':[reg],'alpha':[0.1,0.001,0.0001,0.00001],'tol':[1e-3,1e-4]}]
```

Mejor modelo obtenido con regularización Lasso:

```
Mejores parametros:  {'penalty': 'l2', 'multi_class': 'auto', 'C': 10, 'tol': 0.001, 'solver': 'newton-cg'}
Validacion cruzada:  0.9659952916557677
Ein: 0.034004708344232304
```

Mejor modelo obtenido con regularización Ridge:

```
Mejores parametros:  {'penalty': 'l2', 'alpha': 1e-05, 'tol': 0.001}
Validacion cruzada:  0.9445461679309443
Ein : 0.05545383206905574
```

**SGDClassifier**

Parámetros del modelo:

- loss: La función de pérdida que se utilizará.
- penalty: Se utiliza para especificar la norma utilizada en la regularización.
- n_iter: El número real de iteraciones antes de alcanzar el criterio de detención.

Hiperparámetros:

```
[{'loss': ['log'], 'penalty':[reg], 'n_iter':[1000]}]
```

Mejor modelo obtenido con regularización Lasso:

```
Mejores parametros:  {'penalty': 'l1', 'loss': 'log', 'n_iter': 100}
Validacion cruzada:  0.9589327753073502
Ein : 0.041067224692649784
```

Mejor modelo obtenido con regularización Ridge:

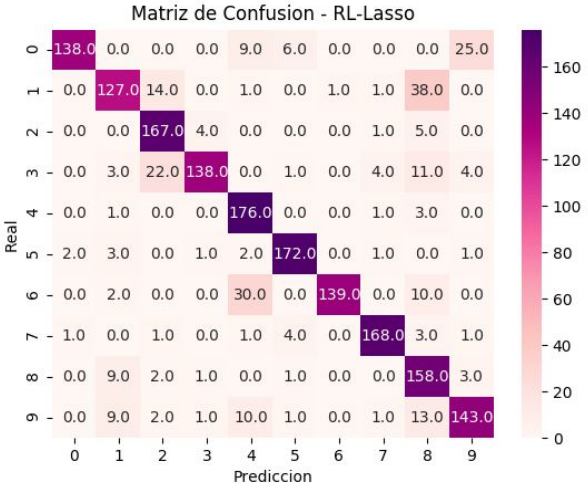
```
Mejores parametros:  {'penalty': 'l2', 'loss': 'log', 'n_iter': 1000}
Validacion cruzada:  0.9639026942191996
Ein : 0.03609730578080039
```

## Estimación del error

### Regresión Logística

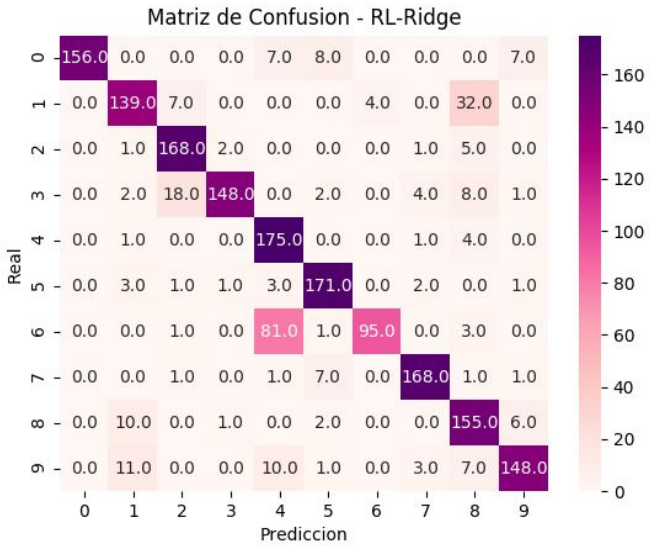
Valores obtenidos con regularización Lasso:

```
Hit score:  0.8491930996104619
Exactitud:  0.8491930996104619
Error en clasificacion:, 0.15080690038953815
Cohen's Kappa:  0.8324660554094612
F1 (weighted):, 0.8500338025603569
```




Valores obtenidos con regularización Ridge:

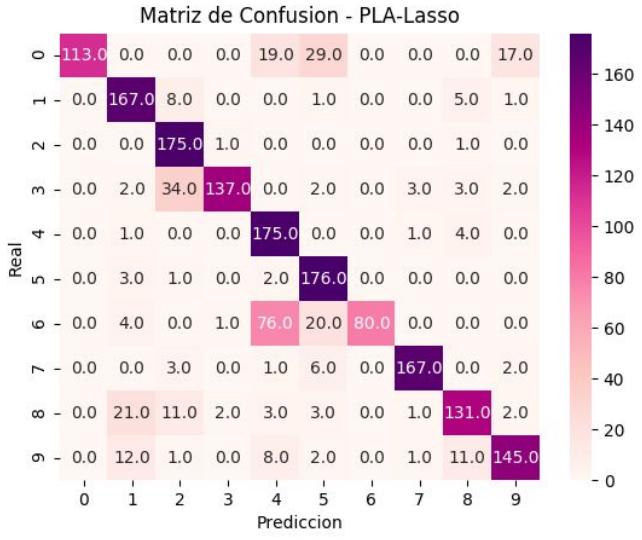
```
Hit score:  0.8491930996104619
Exactitud:  0.8491930996104619
Error en clasificacion:, 0.15080690038953815
Cohen's Kappa:  0.8324660554094612
F1 (weighted):, 0.8500338025603569
```




**Perceptron**

Valores obtenidos con regularización Lasso:

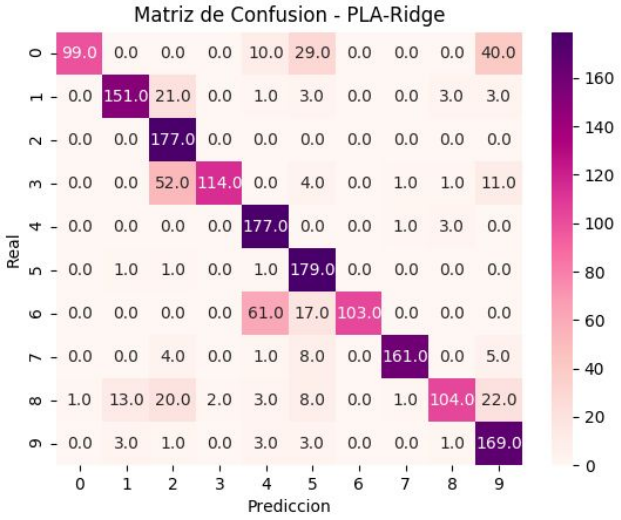
```
Hit score: ', 0.8158041179744018
Exactitud: ', 0.8158041179744018
Error en clasificacion: ', 0.18419588202559822
Cohen's Kappa:  0.7953240984800206
F1 (weighted):, 0.8105266413935968
```




Valores obtenidos con regularización Ridge:

```
Hit score:  0.7979966611018364
Exactitud:  0.7979966611018364
Error en clasificacion:, 0.20200333889816358
Cohen's Kappa:  0.7755347691570773
```

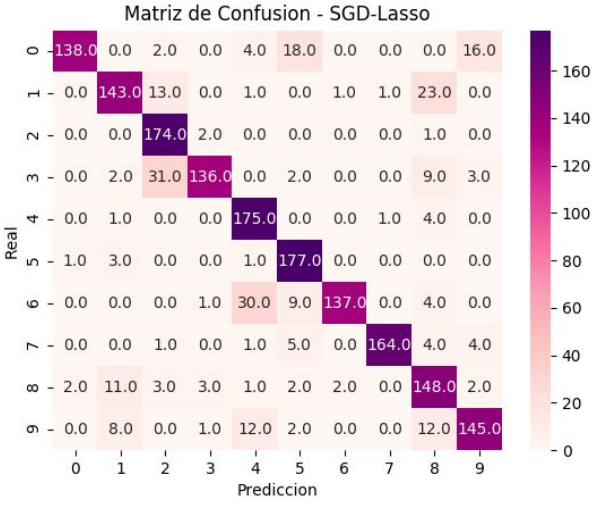
```
F1 (weighted)' 0.7934372881932438
```




**SGDClassifier**

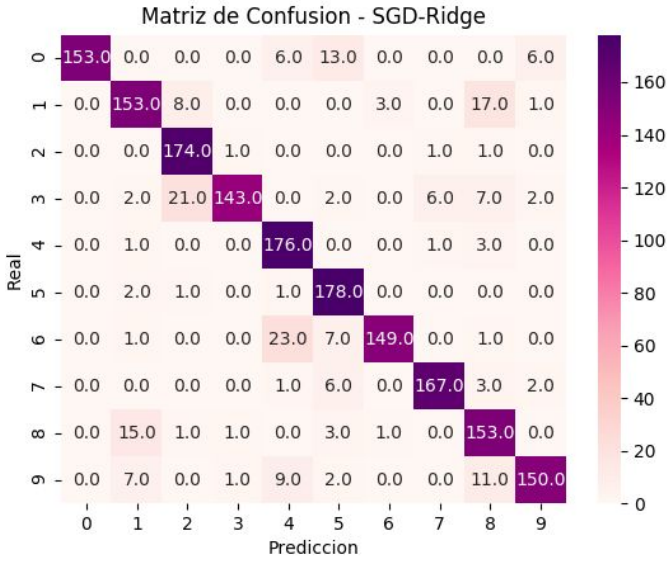
Valores obtenidos con regularización Lasso:

```
Hit score:  0.8553144129104062
Exactitud:, 0.8553144129104062
Error en clasificacion:, 0.14468558708959378
Cohen's Kappa:, 0.8392521139403256
F1 (weighted):  0.8553458410481926
```



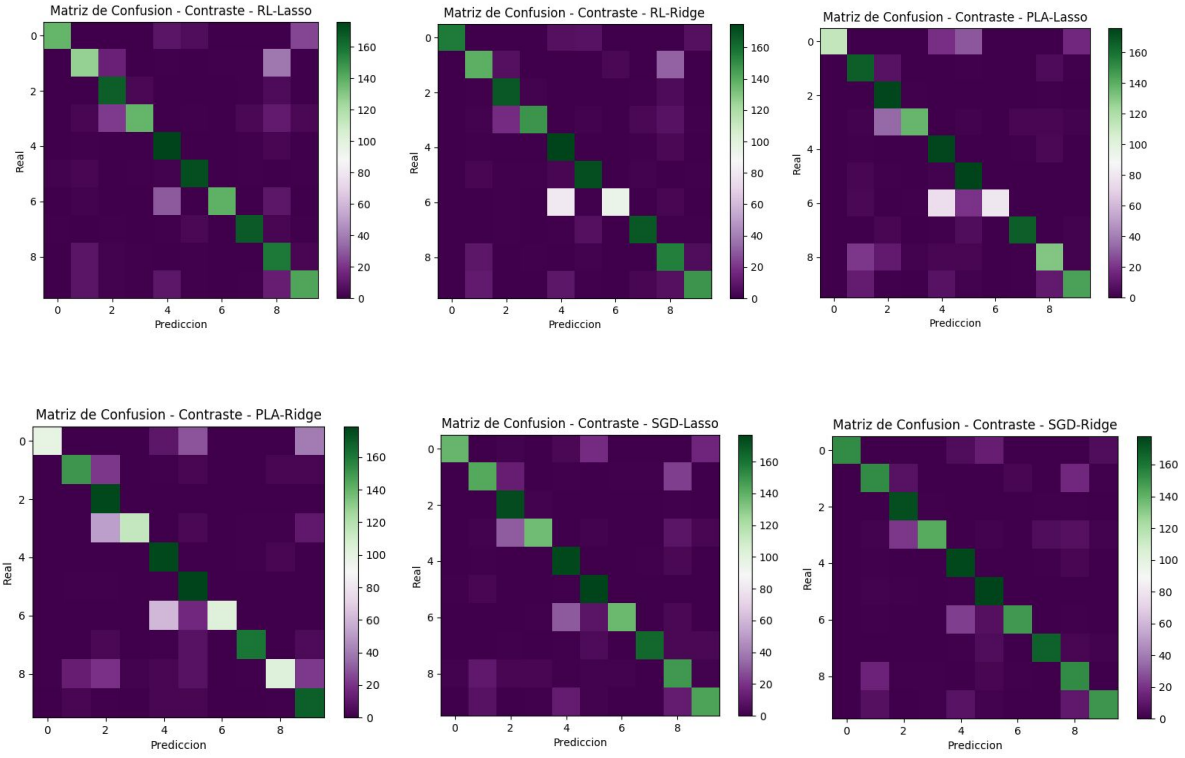

Valores obtenidos con regularización Ridge:

```
Hit score:  0.8881469115191987
Exactitud:  0.8881469115191987
Error en clasificacion:  0.11185308848080133
Cohen's Kappa:  0.8757248583313549
F1 (weighted):  0.8883098407013555
```

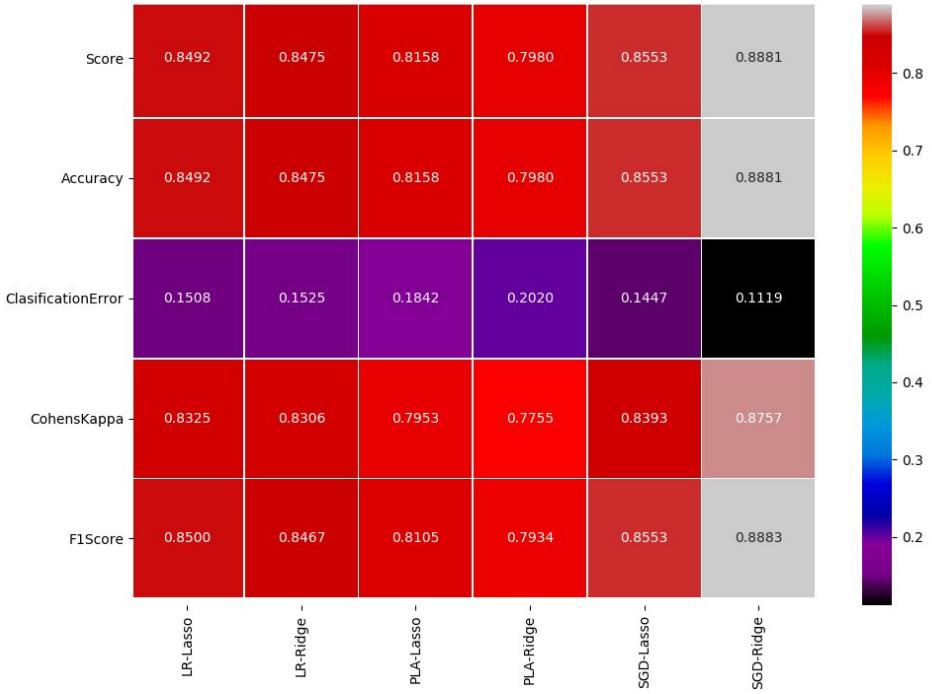



# Técnica de ajuste y justificación

## Comparación de matrices de confusión



## Comparación de resultados




El modelo que seleccionaría sería SGDClassifier con la regularización Ridge pues es el que mejores prestaciones obtiene con diferencia, seguido por el mismo modelo pero con regularización Lasso. Tal y como se mencionó anteriormente, SGDClassifier usa SVM como modelo lineal.

# Communities and Crime

## Comprensión del problema

El problema de regresión consiste en ajustar un modelo lineal a un conjunto de datos y analizar los pasos para ello. El conjunto de datos sobre el que trabajaremos contiene información relacionadas con poblaciones y tasas de crímenes con el fin de predecir el porcentaje de crímenes violentos por población.

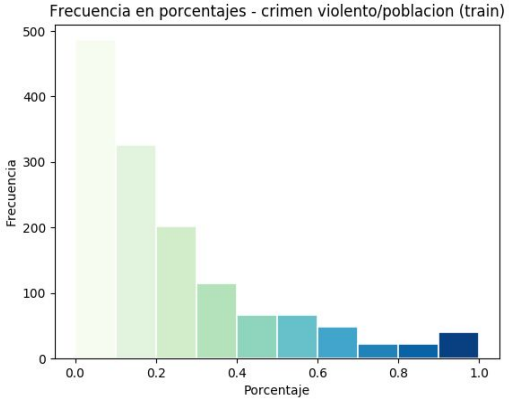
Como se muestra en el repositorio de donde se obtiene la base de datos, ésta consta de 1994 instancias con 128 columnas, algunas de ellas categóricas. En la última columna se encuentra el porcentaje mencionado anteriormente, que será nuestra Y. Las primeras 5 columnas no nos son útiles, por lo que no las incluiremos al leer los datos. La información se encuentra incompleta, por lo que se han eliminado también las columnas en las que faltan muchos datos. En aquellas columnas en las que solo falte un dato, éste se ha sustituído por la media.

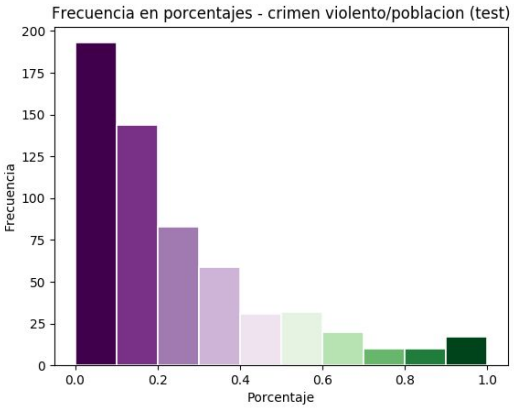
| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 1994 | **Area:** | Social |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 128 | **Date Donated** | 2009-07-13 |
| **Associated Tasks:** | Regression | **Missing Values?** | Yes | **Number of Web Hits:** | 281881 |

Tabla obtenida de la página web http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

Una vez habiendo eliminado toda la información comentada anteriormente, nos habremos quedado solo con 100 columnas. El conjunto X estará formado por las 99 primeras y el conjunto Y por la última (ViolentCrimesPerPob).

A continuación se muestra la frecuencia con la que aparece cada porcentaje del conjunto Y:



## Training y test

Para obtener los conjuntos Train y Test se ha usado la función train_test_split de sklearn. Esta función permite obtener los subconjuntos de prueba y entrenamiento a partir de un conjunto, indicando el tamaño del Test (Train tendrá la totalidad del conjunto exceptuando los seleccionados para el Test). En nuestro caso se ha seleccionado test_size=0.3, obteniendo los siguientes tamaños:

```
Tamaño train:  1395
Tamaño test:  599
```

## Preprocesado los datos

Para este problema, la función VarianceThreshold no proporcionaba ninguna mejora. En su lugar se ha añadido otra función de sklearn, Normalizer, la cual reescala cada fila que tenga al menos un componente distinto de 0 para normalizarla.

| Preprocesado | Columnas | Filas | Ejemplo_valores |
|---|---|---|---|
| Antes | 1395 | 100 | 0.7 |
| Despues | 1395 | 100 | 0.6030743 |

## Métrica

A continuación se muestran las métricas utilizadas para el problema

- $R^2$ score: se ha usado la función r2_score de sklearn.

$$R^2 = 1 - \left[\frac{(1-R2)(n-1)}{n-k-1}\right]$$

- Valor de la varianza explicada: se ha usado la función explained_variance_score de sklearn.

$$EVS = 1 - \frac{Var[\hat{y}i - yi]}{Var[yi]}$$

- Error mínimo cuadrado: se ha usado la función mean_squared_error de sklearn.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y_i - \hat{y}_i)^2$$

- Error absoluto medio: se ha usado la función median_absolute_error de sklearn

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_i - \hat{y}_i|$$

## Modelos a usar

Los modelos considerados para este problema han sido los siguientes:

- Regresión Lineal: se ha usado el modelo LinearRegression de sklearn.

- Regresor SGD: modelo de sklearn consistente en un modelo lineal ajustado con SGD.
- Ridge: modelo de sklearn, resuelve un modelo de regresión donde la función de pérdida es la función lineal de mínimos cuadrados y la regularización viene dada por la norma l2.

## Hiperparámetros y selección del modelo

### Regresión Lineal

Parámetros del modelo:

- fit_intercept: Si se debe calcular la intercepción para este modelo. Si se establece en False, no se utilizará ninguna intercepción en los cálculos.
- normalize: Este parámetro se ignora cuando fit_intercept se establece en False.

Hiperparámetros:

```
[{'fit_intercept':[True,False],'normalize':[True,False]}]
```

Mejor modelo obtenido:

```
Mejores parametros:  {'normalize': False, 'fit_intercept': True}
Validacion cruzada:  0.6434549848124842
Ein : 0.3565450151875158
```

### SGDRegressor

Parámetros del modelo:

- loss: La función de pérdida que se utilizará.
- penalty: Se utiliza para especificar la norma utilizada en la regularización.
- n_iter: El número real de iteraciones antes de alcanzar el criterio de detención.

Hiperparámetros:

```
[{'loss': ['epsilon_insensitive'], 'penalty':[reg], 'n_iter':[1,10,100,1000]}]
```

Mejor modelo obtenido con regularización Lasso:

```
Mejores parametros:  {'penalty': 'l1', 'loss': 'epsilon_insensitive', 'n_iter': 10}
Validacion cruzada:  0.6455648466401066
Ein : 0.3544351533598934
```

Mejor modelo obtenido con regularización Ridge:

```
Mejores parametros:  {'penalty': 'l2', 'loss': 'epsilon_insensitive', 'n_iter': 10}
Validacion cruzada:  0.6459556329612804
Ein : 0.35404436703871955
```

### Ridge

Parámetros del modelo:

- alpha: Constante que multiplica el término de regularización si se utiliza la regularización.
- solver: Algoritmo a utilizar en el problema de optimización.
- tol: El criterio de detención.

Hiperparámetros:

```
[{'alpha':[0.1,0.05,0.01,0.005,0.001],'solver':['cholesky','sag'],'tol':[1e-3,1e-4]}]
```

Mejor modelo obtenido:

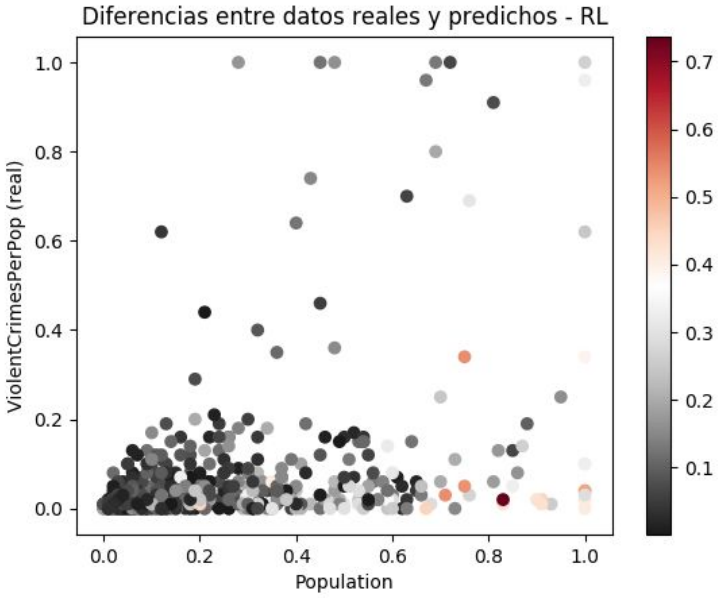
```
Mejores parametros:  {'alpha': 0.1, 'tol': 0.001, 'solver': 'sag'}
Validacion cruzada:  0.6499059624163119
Ein : 0.35009403758368807
```

## Estimación del error

### Regresión Lineal

Valores obtenidos:

```
Hit score:  0.6115031049006632
Valor r2:  0.6115031049006632
Valor de la varianza explicada 0.6222391128540039
Error de los minimos cuadrados:  0.02099231816828251
Error absoluto medio:  0.0731852650642395
```

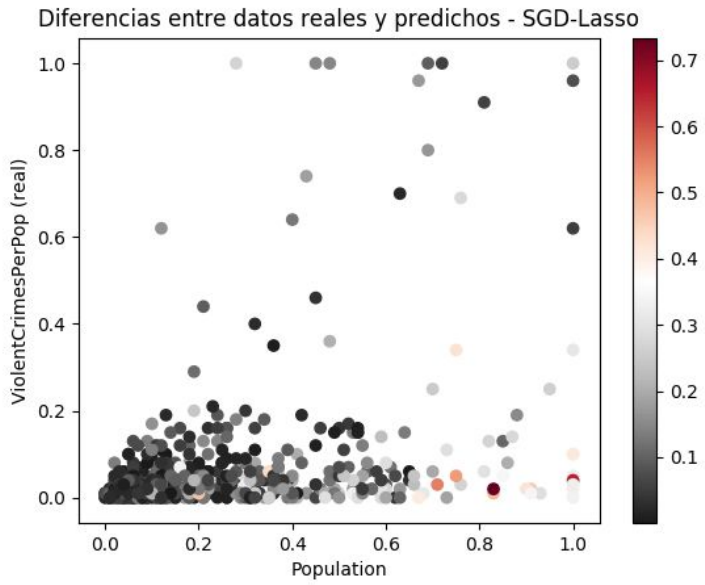



### SGDRegressor

Valores obtenidos con regularización Lasso:

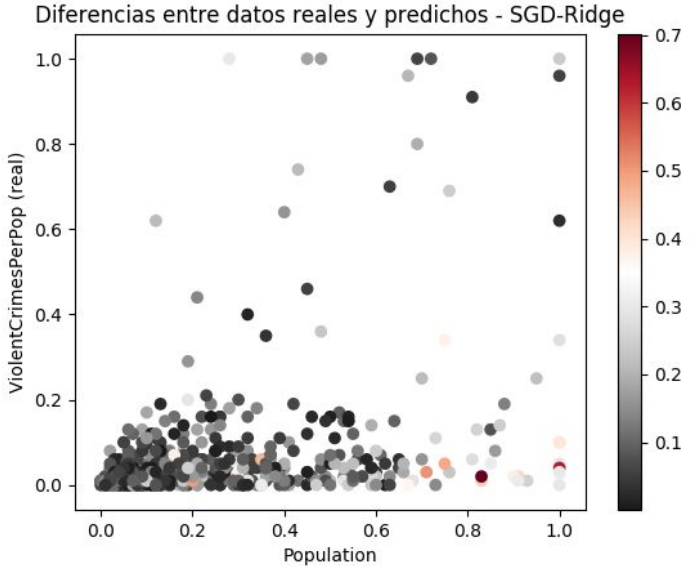
```
Hit score:  0.6388859256876528
Valor r2:  0.6388859256876528
```

```
Valor de la varianza explicada 0.6405873862205969
Error de los minimos cuadrados:  0.019512695021700995
Error absoluto medio:  0.06261260664787593
```




Valores obtenidos con regularización Ridge:

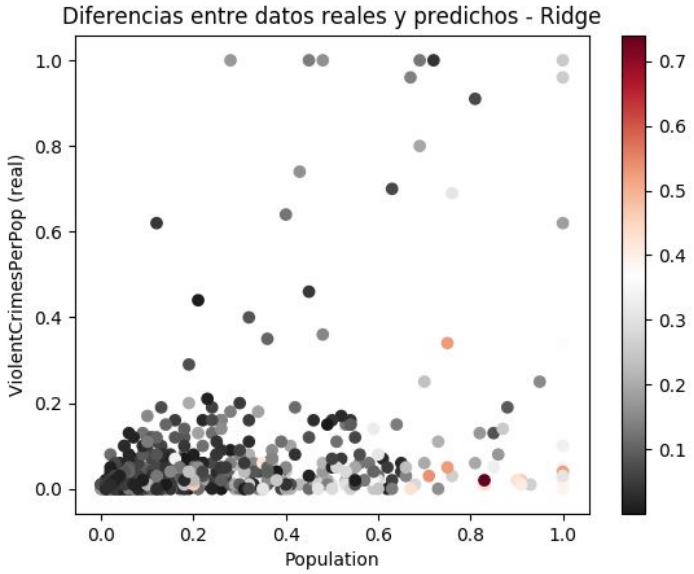
```
Hit score:  0.6177858663605443
Valor r2:  0.6177858663605443
Valor de la varianza explicada 0.6377973330712119
Error de los minimos cuadrados:  0.0206528306516226
Error absoluto medio:  0.08244379996087398
```




**Ridge**

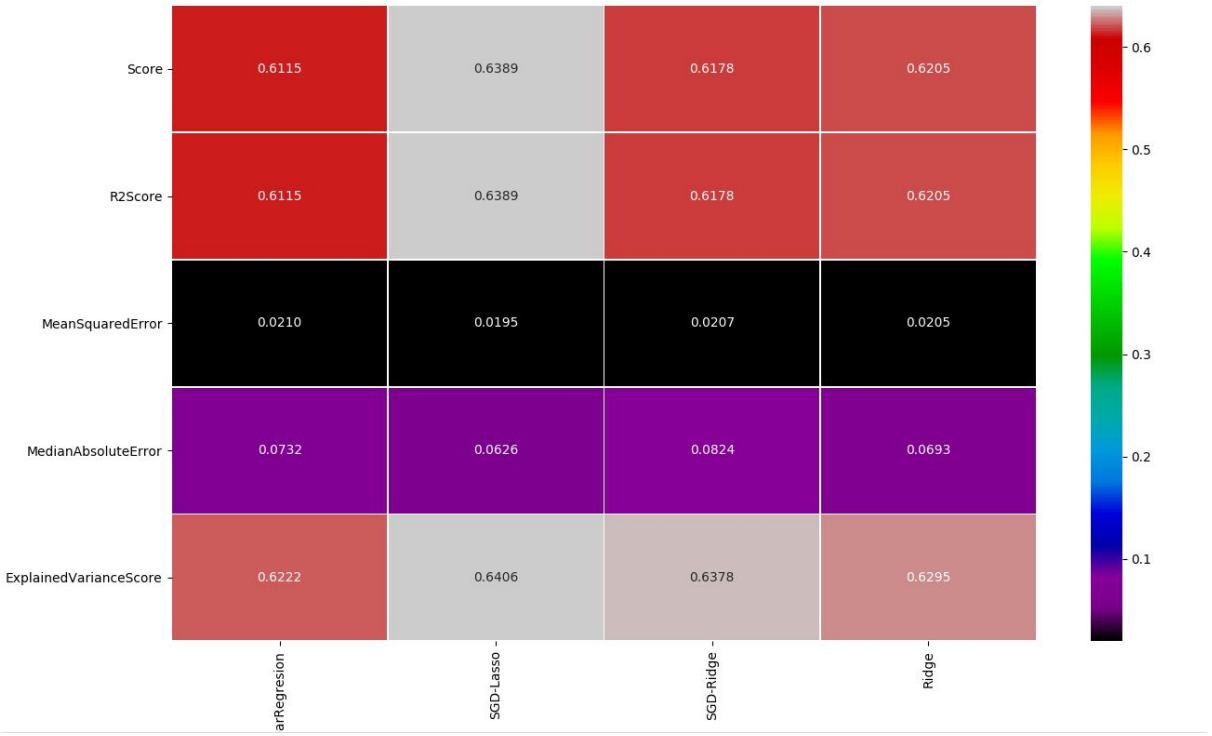
Valores obtenidos:

```
Hit score:  0.6205070075693904
Valor r2:  0.6205070075693904
Valor de la varianza explicada 0.6294556026718421
Error de los minimos cuadrados:  0.02050579456996252
Error absoluto medio:  0.06933642759717054
```




## Técnica de ajuste y justificación

### Comparación de resultados




Ningún modelo obtiene resultados especialmente, considero que esto se debe al propio conjunto de datos. El modelo que seleccionaría sería SGDRegressor con regularización Lasso, seguido otra vez por el mismo SGDRegressor, pero con regularización Ridge.

# Bibliografía


- https://scikit-learn.org
- http://archive.ics.uci.edu
- https://sitiobigdata.com
- https://www.aprendemachinelearning.com
- https://towardsdatascience.com
- https://medium.com
- https://www.interactivechaos.com