Trabajo 3: Programación

AJUSTE DE MODELOS LINEALES

Sergio Aguilera Ramírez

13/05/2019

Grupo 2

Índice:

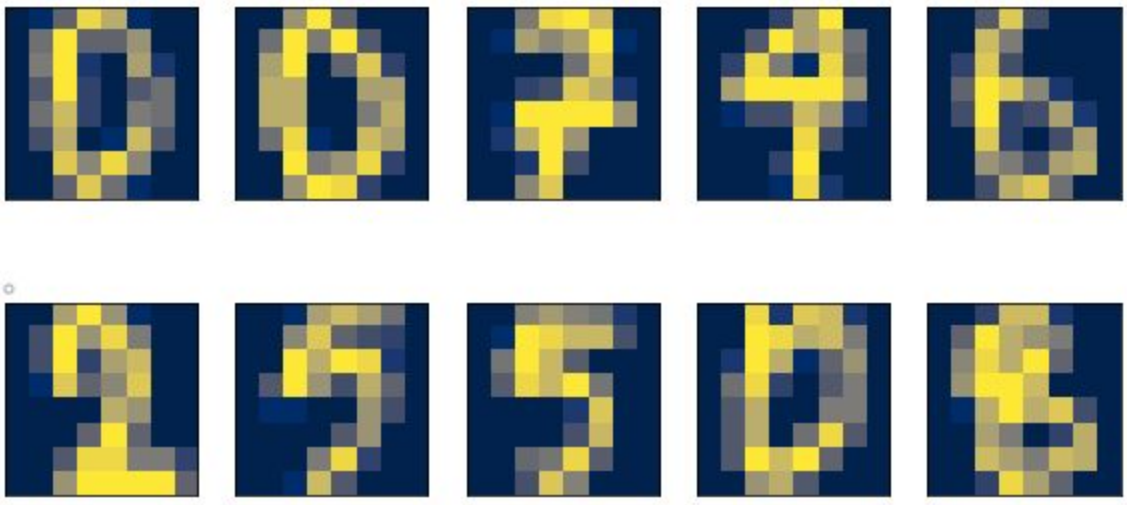
1- CLASIFICACIÓN

1.1- Comprender el problema a resolver

En primer lugar vamos a realizar un estudio sobre un problema de clasificación. Este problema de clasificación se va a realizar sobre el dataset de dígitos manuscritos (optical recognition of handwritten digits), el cual está compuesto por 5620 imágenes que representan dígitos del 0 al 9, representado por 64 atributos cada uno de las instancias.

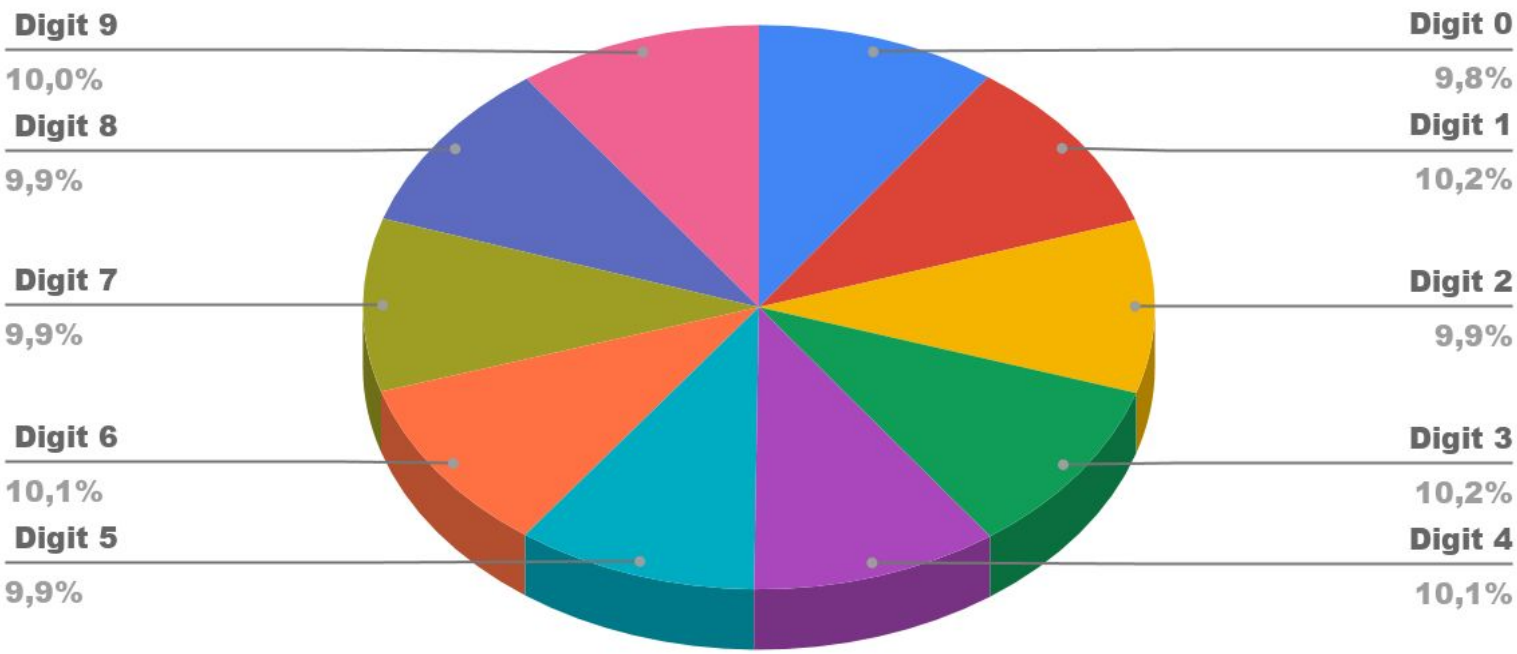
Estos 64 atributos son de tipo numérico, donde todos los valores están comprendidos en el rango [0,16], esto reduce la dimensionalidad y da invarianza a pequeñas distorsiones. Las etiquetas de cada instancia (imágen) están comprendidas en el rango [0,9], representando al dígito respectivo.

A continuación se muestran 10 dígitos, estos dígitos están contenidos en este dataset, como podemos observar los datos no son totalmente claros.

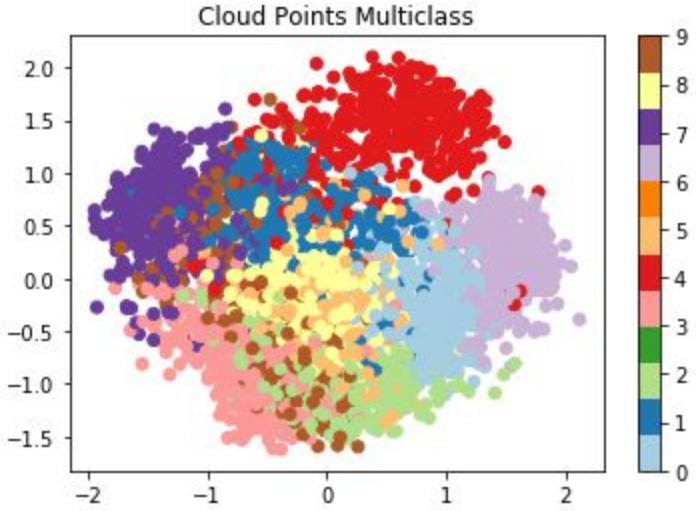


Disponemos de dos archivos .tra y .tes, esto nos da ventaja ya que los datos están “pre-divididos” en cuanto a división de los datos de entrenamiento y prueba, el tamaño de cada conjunto es de 3823 imágenes para el conjunto de entrenamiento y 1797 imágenes para el conjunto de prueba. El objetivo de este problema es determinar a qué dígito corresponde cada imagen fuera de la muestra de la forma más precisa posible.

Obtenemos la distribución de clases del conjunto de entrenamiento para comprobar que el número de ejemplos de todas las clases son equivalentes y no tenemos una clase con pocas instancias lo que supondría un problema en la clasificación respecto a esa clase.



La nube de puntos de las diferentes clases queda de la siguiente forma, se ha reducido la dimensionalidad en 2 columnas para poder representa dicha gráfica:



1.2- Preprocesado de los datos

El preprocesado es una de las fases más importantes de los problemas de maching learning, ya que en en esta fase procuramos obtener un conjunto de datos de mejor calidad para obtener una mejor extracción del conocimiento. En el preprocesado de datos tratamos de reducir de forma eficiente el conjunto de datos, ya que no siempre nuestro modelo va a ser mejor si tenemos más variables en cuenta. ya que si este conjunto contuviera variables irrelevantes o erróneas nos llevaría a un aprendizaje erróneo, también podría conducirnos al overfitting. Además tratamos de disminuir la complejidad del conjunto, esto se lleva a cabo eliminando datos duplicados o de baja variabilidad, normalizando los datos etc. En nuestro caso vamos a utilizar los siguientes métodos:

- **VarianceThreshold() :** este método elimina las características cuya varianza no supera el threshold establecido (0.1 en mi caso), también

elimina las características cuya variación es 0, como hemos comentado antes esto elimina datos duplicados.

- **MinMaxSacler() :** escala todos los valores del conjunto de datos a valores comprendidos entre el rango [0,1], siendo 0 el valor mínimo y 1 el valor máximo del conjunto (los valores intermedios son escalados), esto permite reducir la complejidad del conjunto de datos.

1.3- Selección de funciones

Las funciones utilizados en los diferentes modelos son la funciones lineales, tal y como indica el guión de prácticas. Los diferentes modelos a utilizar son modelos lineales por lo que estos utilizaran dichas funciones, por otro lado hago uso del modelo SVM el cual recibe como parámetro utilizar funciones lineales o polinómicas, en nuestro caso probaremos con los tipos de funciones.

1.4- Definición de los conjuntos training, validación y test

En este apartado voy a explicar cómo he desarrollado la división de conjuntos y he llevado a cabo la validación de los modelos para su selección. Para esto he utilizado la función GridSearchCV() de la librería sklearn, ya que esta función implementa la validación cruzada y a su vez realiza la combinación de los diferentes parámetros pasados como argumento. Esta función recibe como argumentos el modelo lineal (estimador) en nuestro caso, un conjunto de parámetros ({‘penal’:[‘l1’’l2’],’tol’:{1e-3,1e-4]}). La función realizará todas las validaciones con cada una de las combinaciones posibles de estos parámetros, también recibe como parámetro el ‘cv’ referente al número de particiones que realizará esta función en nuestra validación, las distintas validaciones cruzadas se llevarán a cabo sobre el conjunto de entrenamiento, la división de estos conjuntos tiene tamaño 80% para train y 20% para test.

Por otro lado, una vez terminado todas las validaciones podemos obtener la mejor puntuación media (modelo.best_score_) conseguida entre las validaciones, además podemos obtener los parámetros con los que hemos obtenido dicho porcentaje de acierto (modelo.best_params_).

1.5- Discutir regularización

Para nuestros modelos vamos a utilizar dos tipos de regularizaciones, en primer lugar utilizaremos la regularización Lasso basada en la asignación de un coeficiente a cada atributo a la hora de generar la combinación entre ellos, ésta regularización es útil cuando algunas de las variables del conjunto de datos no influyen mucho en el modelo, ya que cuando realiza la asignación de los coeficientes estos atributos serán igual a 0, por otro lado utilizaremos la regularización Ridge, esta regularización es eficaz cuando la mayoría de las características son importantes (relevantes), lo que supondrá la asignación de coeficientes de menor valor a todas las características.

1.6- Modelos lineales usados y sus parámetros

Se han utilizado tres tipos de modelos, siendo estos regresión logística, Perceptron y SVM (Support Vector machines). Como hemos comentado anteriormente cada modelo ha sido ejecutado con diferentes combinaciones de parámetros. Los parámetros utilizados son :

-Regresión logística:

```
parameters = [{'penalty':['l2'],'C':[1,10,100,1000,10000], 'tol':[1e-3,1e-4], 'solver':['newton-cg'],
'multi_class':['auto']}]
```

El parámetro penal da lugar al tipo de regularización en nuestro caso vamos a utilizar la regularización Lasso, ya que newton-cg solo acepta la regularización ‘l2’, newton-cg es el algoritmo utilizado para la optimización multiclase, tol es la tolerancia en la optimización.

-Perceptron:

```
parameters = [{'penalty':['l1','l2'],'alpha':[0.1,0.001,0.0001,0.00001,1e-05],'tol':[1e-3,1e-4]}]
```

Alpha es el valor que multiplicará al término de regularización.

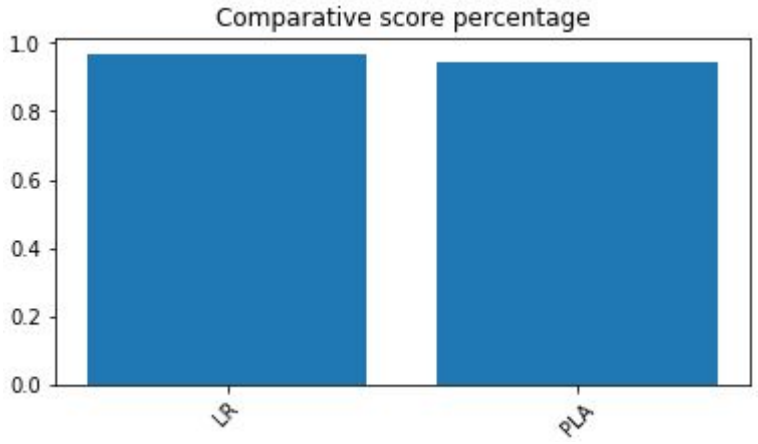
-SVM:

parameters =
[{'gamma':['scale'],'C':[1,10,100,1000,10000],'kernel':['linear','poly'],'degree':[2,6],'tol':[1e-3,1e-4]}]

Kernel es el parámetro para especificar el tipo de kernel utilizado en el algoritmo, en nuestro caso probamos el lineal y el polinomial, degree indica los grados del polinomio y gamma especifica el coeficiente kernel, es decir, en nuestro caso utilizamos scaled por lo que para evaluar gamma utilizamos (n_features * X.var()).

1.7- Selección y ajuste del modelo final

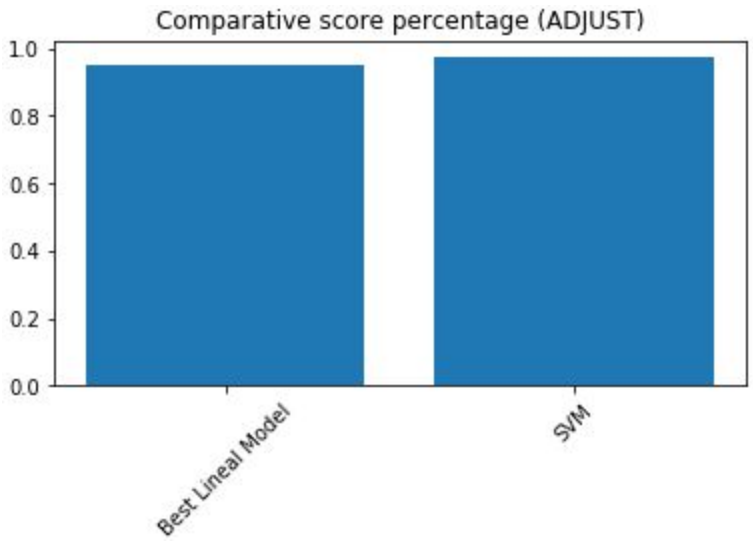
Una vez realizado las validaciones con los diferentes modelos lineales (la comparación con SVM se realiza después de obtener el ajuste sobre el mejor de los modelos lineales) comprobamos cuál de ellos ha obtenido un mejor porcentaje de acierto. La siguiente gráfica muestra los resultados obtenidos por los modelos perceptron y regresión logística:



Como podemos observar en la gráfica (se ve muy escasamente la diferencia) el porcentaje obtenido por el modelo de regresión logística es mejor con respecto al obtenido por el perceptron.

Por lo tanto el ajuste del modelo final se ha realizado sobre el modelo de regresión logística. Ajustamos los datos de prueba usando regresión logística con los parámetros con los que hemos obtenido la mejor tasa de acierto (modelo.best_params_) en la validación cruzada. El resultado obtenido sobre los datos prueba es 0.9510294936004452, por lo que el resultado es bastante bueno, esto nos permitirá clasificar las distintas imágenes fuera de la muestra de forma correcta.

Para aumentar el estudio de este problema he utilizado el modelo SVM para la clasificación de los datos y he comparado el resultado final (ajuste sobre los datos prueba) con el resultado obtenido del ajuste realizado por el mejor modelo lineal. Generando el diagrama de barras podemos observar como el modelo SVM mejora el porcentaje de acierto fuera de la muestra.



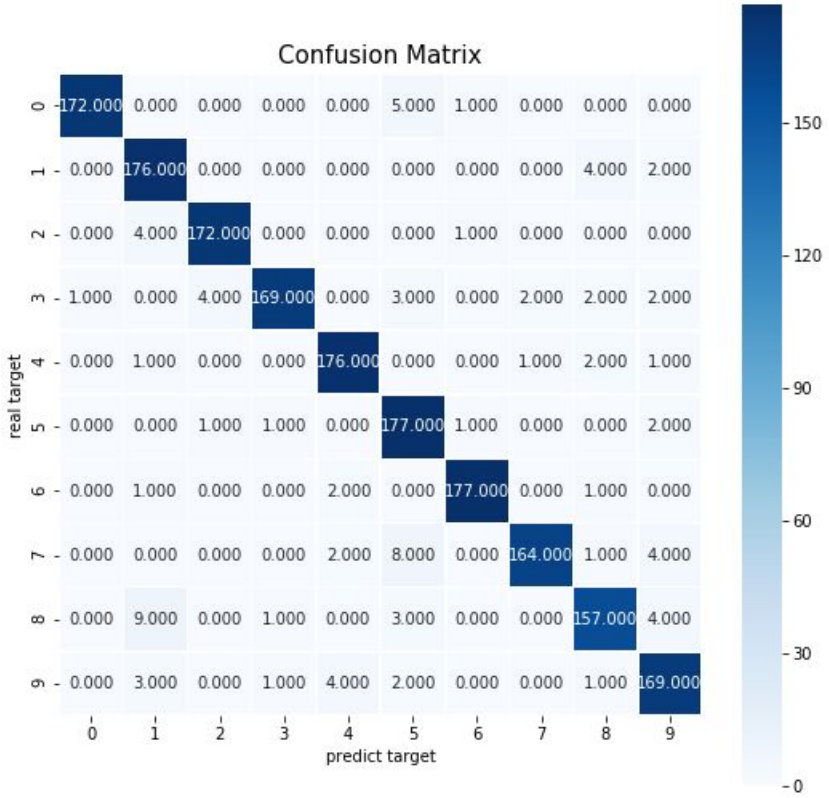
Como hemos visto en teoría, SVM es un clasificador que funciona mediante el modelado de las variables de entrada como puntos en un espacio geométrico, los cuales son mapeados de tal forma que los datos pertenecientes a diferentes clases son divididos de manera eficiente por un hiperplano de separación lo más ancho posible, permitiendo así una mejor división entre los datos que la regresión logística.

1.8- Discutir métrica

En este apartado se deben seleccionar los distintos tipos de métrica a usar, en mi caso he utilizado 4 tipos de métrica, matriz de confusión (2 variantes), curva de ROC multiclass, recall y precisión.

Matriz de confusión

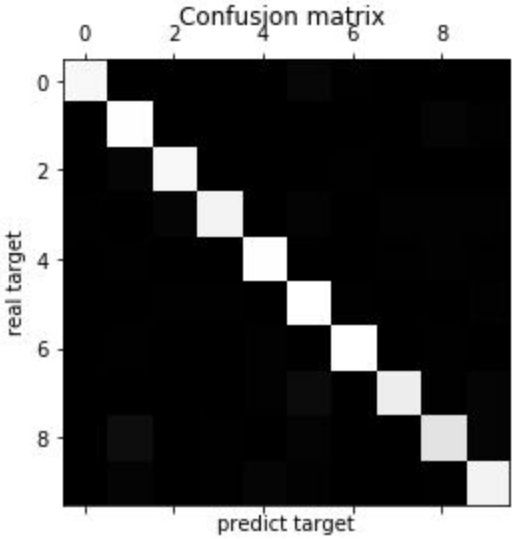
La matriz de confusión nos muestra el comportamiento de nuestro modelo con respecto a las diferentes clases de nuestro conjunto de datos. Las columnas de la matriz representan el número de predicciones de cada clase, mientras que las filas representan las etiquetas reales del conjunto de datos.



La diagonal representa el número de datos de esa clase que están bien clasificados, mientras que el resto de los valores muestran el número de datos que se han clasificado mal respecto a la clase real. Por ejemplo podemos observar que el dígito 8 se ha clasificado 9 veces como el dígito 1, y también vemos como el

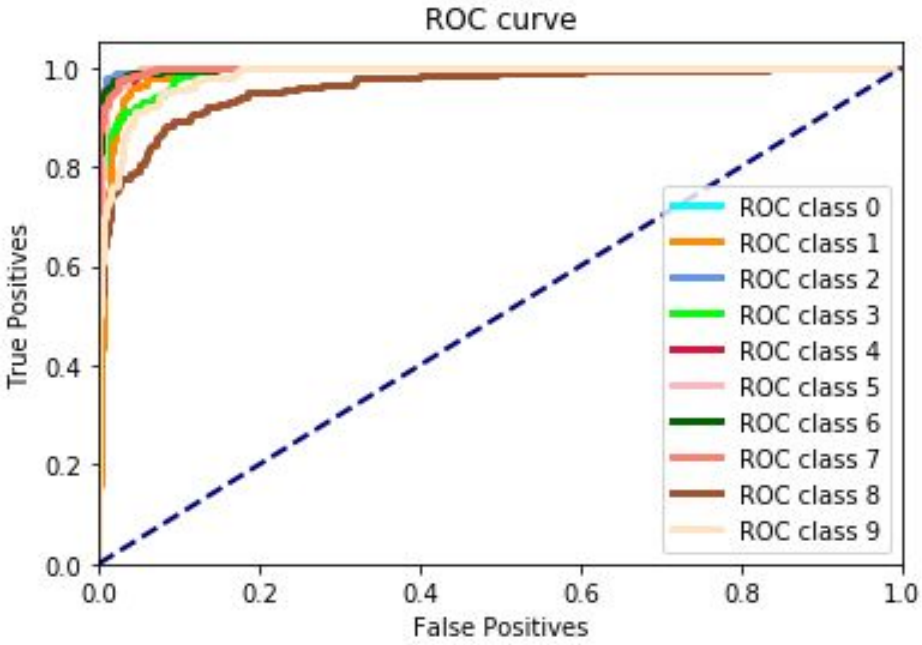
dígito 7 se ha clasificado 8 veces como el número 5. Esto nos ayuda visualizar de forma más clara el comportamiento de nuestro modelo sobre el modelo seleccionado.

Podemos representar la matriz de confusión de forma diferente, resaltando en blanco (de forma degradada) las zonas donde las etiquetas reales se han clasificado con respecto a las clases predichas.



Curva de ROC

La curva de ROC es la representación de trazar la tasa verdadera positiva (proporción de datos clasificados incorrectamente como negativos) con la tasa verdadera negativa (proporción de datos clasificados incorrectamente como positivos. La siguiente gráfica muestra las curvas de RoC respectivas de cada clase del conjunto de datos.



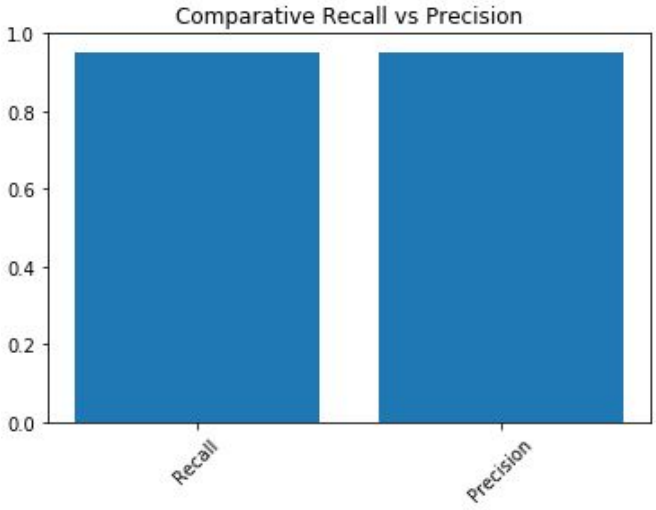
Observando la gráfica podemos ver como el dígito 8 es el peor clasificado con respecto a las demás clases.

Precisión y Recall

Recall nos aporta información sobre el rendimiento de un clasificador con respecto a los falsos negativos, mientras que la precisión nos muestra información sobre su rendimiento con respecto a los falsos positivos. Nuestro clasificador será mejor cuando estos valores sean los más cercanos a 1. Los valores obtenidos en nuestro clasificador son:

-Recall: 0.9508766354456801

-Precision: 0.9524578164929063



Por último vamos a mostrar los diferentes valores de precisión, recall, f1-score y support obtenidos en las distintas clases de nuestro conjunto de datos.

```
# METRICS CLASS #
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       178
           1       0.91      0.97      0.94       182
           2       0.97      0.97      0.97       177
           3       0.98      0.92      0.95       183
           4       0.96      0.97      0.96       181
           5       0.89      0.97      0.93       182
           6       0.98      0.98      0.98       181
           7       0.98      0.92      0.95       179
           8       0.93      0.90      0.92       174
           9       0.92      0.94      0.93       180

   micro avg       0.95      0.95      0.95      1797
   macro avg       0.95      0.95      0.95      1797
weighted avg       0.95      0.95      0.95      1797
```

Podemos observar como hemos comentado en apartados anteriores como los valores recall disminuyen en los dígitos 7,8,9 y 3 (falsos negativos) mientras que la precisión baja en los dígitos 1,8 y 9 (falsos positivos).

1.9- Estimación del error E_{out}

El error obtenido al realizar el ajuste sobre el mejor modelo seleccionado es de 0.0489705063995548, el resultado como podemos comprobar es bastante bueno lo que quiere decir que la clasificación de los datos fuera de la muestra será correcta en la mayoría de los casos.

1.10- Discutir y justificar calidad del modelo

En conclusión, podemos decir que nuestro modelo lineal seleccionado, siendo este regresión logística, representa de forma correcta el conjunto de datos con el que estamos trabajando, ya que como hemos comentado en apartados anteriores el porcentaje de acierto respecto a los datos de prueba (fuera de la muestra) es bastante bueno y de igual forma el error resultante es muy bajo. Por medio de la observación de la matriz de confusión y de las demás métricas podemos ver como este modelo clasifica la mayoría de las etiquetas de forma correcta, fallando de manera resaltada en los dígitos 8 y 7.

Por otro lado, comparando con otros modelos más avanzados como es el modelo support vector machines (SVM), vemos como la regresión logística queda algo por debajo de éste respecto a los porcentajes de acierto obtenidos. Esto es debido a que el modelo SVM trata de encontrar un hiperplano donde la distancia a los dos vectores de soporte (estos estarían en el límite de cada clase) sea la mayor posible, es decir, encontrar la posición del hiperplano donde la distancia al punto más cercano de cada clase esté a la mayor distancia del hiperplano, lo que mejora la clasificación con respecto a los datos test.

En conclusión, este dataset optical handwritten digits está representado de forma correcta mediante los modelos lineales usados en este estudio. Se podría ampliar el número de instancias del conjunto de prueba para ver si el error aumenta o disminuye, esto serviría para demostrar de una forma más segura la idoneidad de los modelos.

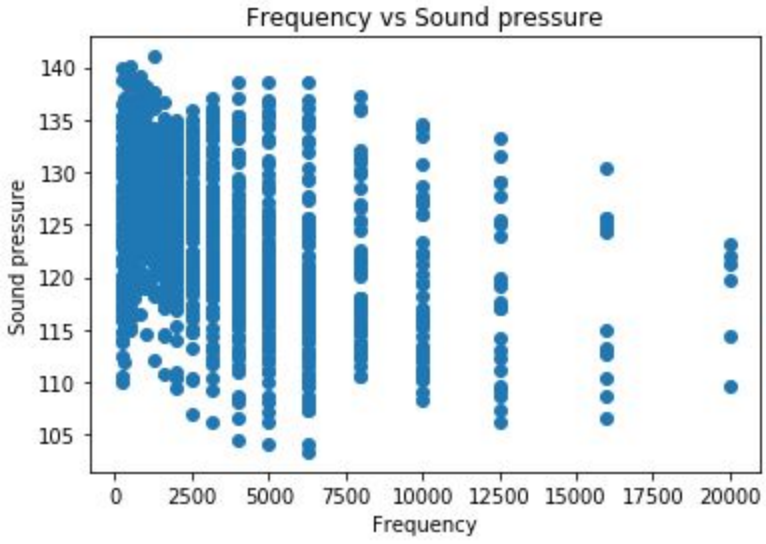
2- REGRESIÓN

2.1- Comprender el problema a resolver

En este ejercicio vamos a realizar el estudio sobre un problema de regresión. La base de datos a utilizar es airfoil self-noise. Esta base de datos recoge información sobre una serie de pruebas aerodinámicas y acústicas de aspas aerodinámicas. Este dataset está compuesto por 1503 instancias, cada instancia viene representada por 6 atributos de carácter numérico.

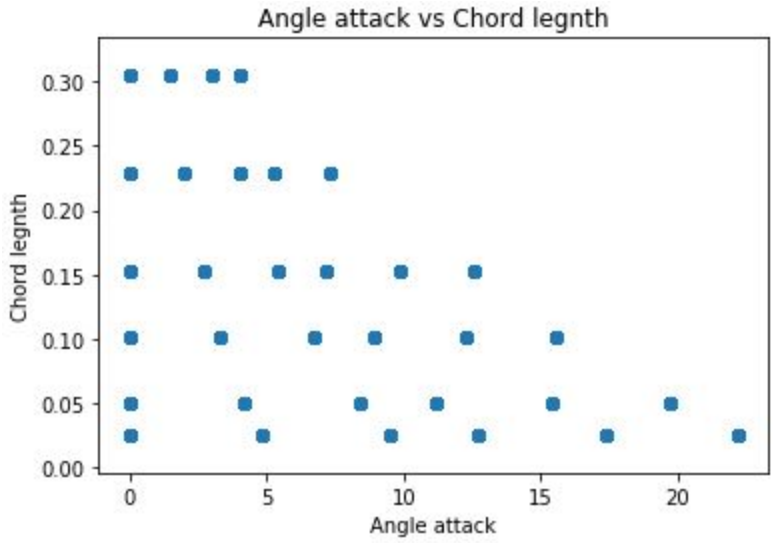
Estos 6 atributos representan las siguientes características tomadas en las pruebas: frecuencia (Hz), ángulo de ataque (grados), longitud de cuerda (metros), velocidad del flujo libre (metro/segundo), espesor de desplazamiento lateral (metros) y el nivel de presión sonora (decibelios) este último valor se toma como etiqueta de las instancias.

A continuación se muestran algunas gráficas de puntos tomando para su representación dos características de este conjunto.

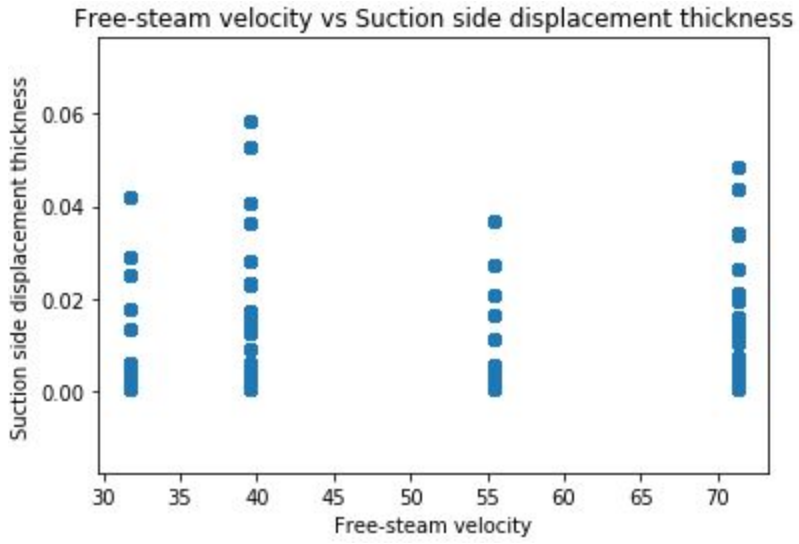


Esta gráfica representa los datos mediante las características de frecuencia y presión de sonido. Los valores de frecuencia oscilan entre 0 y 20000 hercios mientras que la presión del sonido está comprendida entre ~100 y ~145. Vemos

como la mayoría de los datos tienen una frecuencia entre 0 y 2500, ya que entre esos valores hay una gran concentración de instancias.



Con respecto al ángulo de ataque y longitud de cuerdo los valores obtenidos están menos dispersos, es decir, los resultados obtenidos se concentran en unos determinados valores.



En cuanto a la velocidad del flujo libre y el espesor de desplazamiento observamos como los resultados de velocidad obtenidos en las pruebas varía entre los valores ~32, 40, ~55 y ~73 m/s , en cuanto al espesor de desplazamiento lateral vemos como oscila entre los valores 0 y 0.06.

2.2- Preprocesado de los datos

En la fase de procesado se ha utilizado dos métodos, el aumento dimensionalidad y el escalado, ya que cada instancia consta de 6 características, siendo este un número escaso he pensado que una reducción de dimensionalidad no sería lo más eficiente en este dataset. Uno de los métodos en el preprocesado es realizar un aumento la dimensionalidad de los datos mediante la función PolynomialFeatures(), ésta función genera una nueva matriz de entidades compuesta por todas combinaciones polinomiales de las entidades con un grado menor o igual al grado especificado como parámetro (3), consiguiendo un conjunto más grande con el que poder validar nuestro modelo, las instancias pasan a tener 56 atributos (5 sin procesar). También he utilizado la función MinMaxScaler(), esta función escala todas las características entre 0 y 1, donde las característica de valor mínimo será 0 y la característica de mayor será 1, escalando los valores comprendidos entre el valor mínimo y valor máximo.

2.3- Selección de funciones

En nuestro estudio vamos a utilizar funciones lineales. Estas funciones son utilizadas por el modelo de regresión lineal el cual vamos a emplear en este problema junto con los modelos Lasso y Ridge, estos modelos utilizan combinaciones lineales en la representación del conjunto de datos.

2.4- Definición de los conjuntos training, validación y test

La división del conjunto de datos en training y test la he llevado a cabo mediante la función “train_test_split” la cual se encarga de dividir el conjunto de datos de forma que el conjunto de entrenamiento contiene el 80% del dataset y el conjunto de prueba el 20% restante, está proporción es especificado en los argumentos de entrada de la función (test_size).

Para la validación de los diferentes modelos a utilizar, al igual que en el problema de clasificación he utilizado la función GridSearchCV() que permite la validación del modelo mediante la división del conjunto de entrenamiento (5-fold cross validation) y la combinación de los hyperparámetros especificados.

2.5- Discutir regularización

Para el problema de regresión vamos a utilizar las regularizaciones Lasso y Ridge, como ya hemos comentado anteriormente sus formas de actuar sobre los conjunto de datos cabe decir que en este problema vamos a probar diferentes parámetros junto con estas regularizaciones. El principal motivo por el que usar regularización en nuestros conjuntos de datos es para disminuir la probabilidad de overfitting, esto se produce al entrenar el modelo demasiado o por el entrenamiento de dicho modelo con datos extraños (ruido, erróneos etc).

2.6- Modelos lineales usados y sus parámetros

En este problema vamos a utilizar tres modelos lineales diferentes. Estos modelos son regresión lineal, Ridge y Lasso. La razón por la que usar estos modelos se debe a que estos modelos son para problemas de tipo regresión. Los parámetros utilizados en los diferentes modelos son:

Regresión lineal:

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

Utilizamos fit_inercept para centrar los datos, el parámetro normalize sirve para normalizar los datos antes de la regresión, por lo que lo utilizaremos para ver si obtiene mejores resultados normalizando o no dichos datos.

Lasso:

```
[{'alpha':[1.0,0.5,0.1,0.05,0.01,0.005,0.001],'selection':['random','cyclic'],'tol':[1e-3,1e-4,1e-5,1e-6,1e-7]}]
```

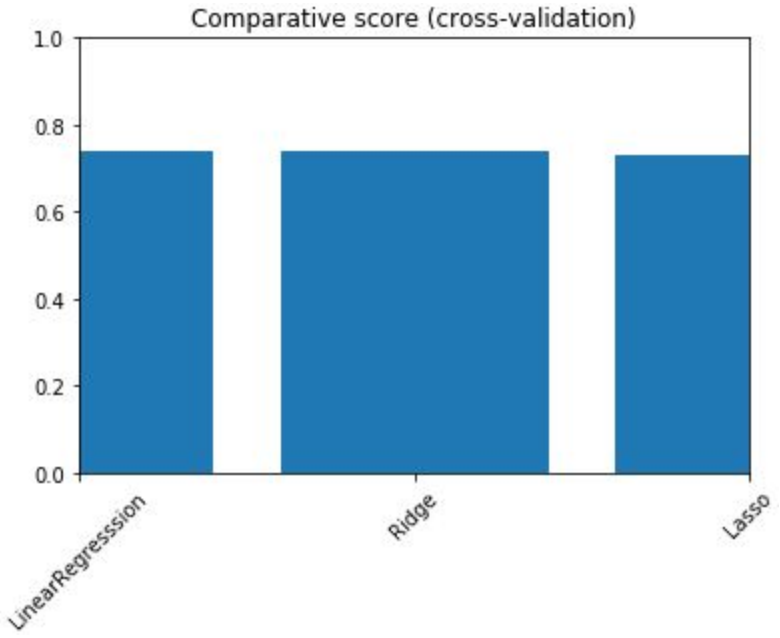
Ridge:

```
[{'alpha':[1.0,0.5,0.1,0.05,0.01,0.005,0.001],'solver':['cholesky','saga'],'tol':[1e-3,1e-4]}]
```

El parámetro tol en los modelos Ridge y Lasso es la tolerancia para la optimización, con respecto al solver de Ridge especificamos cholesky para que el modelo utilice la función estándar scipy.linalg.solve para calcular la solución y saga que utiliza el promedio del gradiente descendente estocástico.

2.7- Selección y ajuste del modelo final

Una vez hechas las validaciones de los diferentes modelos, tenemos que seleccionar el modelo final para ajustarlo con respecto al conjunto de prueba. Mediante la siguiente gráfica podemos observar cual de los tres modelos a obtenidos mejores resultados.



Como podemos ver el mejor resultado obtenido entre las validaciones de los modelos es Ridge, el resultado obtenido por este modelo no es un resultado muy bueno para la estimación de este dataset, dicho valor es 0.741645389505807.

Después de ajustar dicho modelo con el conjunto de datos fuera de la muestra, es decir con los datos de prueba, obtenemos un porcentaje de acierto del 0.5406273609937385, por lo que podemos comprobar, el porcentaje de acierto fuera de la muestra es bastante menor que el obtenido en los datos de entrenamiento, esto quiere decir que nuestro modelo de regresión no se ajusta de manera correcta a nuestro conjunto de datos. Este error podría reducirse si aumentamos el número de características y la diversidad de las instancias fuera mayor.

2.8- Discutir métrica

En cuanto a la métrica he utilizado los errores MSE y MAE, además también se ha utilizado el error R squared. MSE mide el error cuadrático promedio de nuestras predicciones, es decir, para cada punto calcula la diferencia cuadrada entre las predicciones y las etiquetas, y calcula el promedio de esos valores, cuanto mayor sea este valor peor es nuestro modelo. Por otro lado, MAE calcula error como el promedio de las diferencias absolutas entre los valores de las etiquetas reales y las etiquetas predecidas.

R squared (R^2) o coeficiente de determinación es similar a MSE pero cuando la ventaja con respecto a este es que si el valor de salida es muy grande MSE sesga dicha métrica mientra que el coeficiente de determinación no. Esto permite obtener el verdadero error cuadrático de nuestro modelo.

-MSE: 19.511303023419536

-MAE: 3.4102259247609488

-Coefficient of determination: 0.5406273609937385

2.9- Estimación del error E_{out}

El error obtenido en el ajuste final del modelo seleccionado es de Eout: 0.45937263900626146, siendo el error bastante malo, podemos concluir que nuestro modelo no será de gran calidad en la clasificación de los datos. El error puede verse reflejado por la falta de información en el dataset o por el mal ajuste de los modelos al conjunto.

2.10- Discutir y justificar calidad del modelo

En conclusión, según los resultados obtenidos mediante el estudio de este conjunto de datos con los diferentes modelos lineales, podemos concluir que estos algoritmos representan de forma correcta dicho conjunto. Hemos visto que el mejor valor obtenido ha sido el modelo Ridge, obteniendo un porcentaje de acierto de 0.5406273609937385 fuera de la muestra lo que supone que estos modelos no son de buena calidad con respecto a los datos.

Esto puede verse afectado por la poca diversidad de los datos dentro de la del conjunto de datos, ya que al entrenar nuestros modelos estos no obtienen un comportamiento correcto para la estimación de los valores fuera de la muestra.

Una mejora de la calidad de estos modelos sería aumentar el tamaño de instancias, siempre que esta información no contenga errores (ruido) ni inconsistencias, dicho de otra forma podemos añadir nuevos datos recogidos mediante pruebas sobre las aspas aerodinámicas, por lo contrario si añadimos información errónea o extraña y el conjunto de datos incrementa mucho podría aparecer overfitting.