Trabajo 3 Programación BBDD 2

Víctor Manuel Arroyo Martín
Aprendizaje Automático

Esta segunda memoria debe verse como una sola junto a la de la base de datos anterior. Las he separado porque la justificación de muchas preguntas es distinta pero hay algunas que tienen la misma justificación.

1 Comprender el problema a resolver. Identificar los elementos X, Y and f del problema.

En este caso tenemos una base de datos numérica que contiene datos recogidos en comunidades de EEUU con el propósito de predecir el número total de crímenes violentos por cada 100.000 habitantes. La base de datos está compuesta por 1994 instancias y 128 atributos de los cuales los 5 primeros no sirven para predecir así que los he borrado al principio del programa.
X son los datos recogidos en la base de datos sobre cada población de 10000 habitantes (edad media, localización...) y la etiqueta Y es el número de casos violentos ocurridos en esas poblaciones. La f que queremos ajustar es aquella que nos diga con más exactitud cuántos crímenes violentos hay en una población de 10000 de la que disponemos de los datos de cada atributo.

2 Selección de las clase/s de funciones a usar. Identificar cuáles y porqué.

Al igual que en la base de datos categórica, en esta hago uso de funciones lineales y cuadráticas con las que se obtienen buenos resultados. Añadir complejidad a las funciones no mejora a penas y el programa aumenta mucho su tiempo de ejecución.

3 Fijar conjuntos de training y test que sean coherentes.

Los datos venían todos en un fichero así que haciendo uso de una función de sklearn, los mezclé y los dividí en 30 y 70% de test y train respectivamente. Con el de train hice crossvalidation para elegir el mejor modelo dividiendo el conjunto en 5.

4 Preprocesado los datos

Los datos que vienen en el fichero data vienen sin nungún preprocesado excepto el de no incluir variables claramente incorreladas. Por ello, tiene valores perdidos, datos con una varianza muy pequeña etc. así que es importante el preprocesamiento en esta base de datos.

Para el preprocesado de datos también he añadido las siguientes funciones de sklearn:

-SimpleImputer: Lo que hace es que una columna con datos perdidos (NaN) sustituye éstos por la media de los que no hay perdidos en ese mismo atributo con lo que el impacto en el entrenamiento al sustituir estos datos perdidos es nulo.

-Variancethreshold: Con el parámtero 0.01 indico que quiero eliminar las variables que tengan menos de esa varianza ya que una varianza tan baja no va a aportar nada en el aprendizaje de nuestra función ibjetivo.

-PolynomialFeatures: para generar un vector w de pesos con forma lineal.

-StandardScaler: Estandariza las características eliminando la media y escalando a la varianza unidad. Si u es la media y s la desviación típica de nuestros datos de entrenamiento, standardScaler transforma un dato x en

$$z=\frac{(x-u)}{s}$$

5 Fijar la métrica de error a usar. Discutir su idoneidad para el problema.

La métrica en este caso es distinta a la de clasificación. Ahora la métrica usada es para valores reales así que usamos una función de pérdida para regresión. En este caso he usado la pérdida de regresión de error cuadrático medio negativa, pues es la que mejores resultados me ha dado.

6 Discutir la técnica de ajuste elegida.

Como en la anterior base de datos, he usado psudoinversa y SGD pues se adaptan bien a cualquier situación. Tanto la regresión lineal como lasso y como ridge usan pseudoinversa y SGDRegressor usa SGD como su nombre indica.

7 Discutir la necesidad de regularización y en su caso la justificar la función usada para ello.

Misma justificación que en la BBDD 1.

8 Identificar los modelos a usar.

Como modelos he decidido utilizar varios que hacen uso de regresión lineal:
-SGDRegressor: Es regresión lineal con sgd en vez de pseudoinversa.
-LinearRegression: Regresión lienal.
-Ridge: Hace uso de l2 para la regresión y evitar el overfitting.Usa pseudoinversa.
-Lasso: Usa l1 lasso para evitar el overfitting y también usa pseudoinversa como ridge.
A pesar de que todos usan regresión lineal, las opciones y parámetros son distintos y por eso he comparado mediante crossvalidation los cuatro.

9 Estimación de hiperparámetros y selección del mejor modelo.

En SGDRegressor he usado los siguientes parámetros:
- loss: la función de pérdida. He puesto cuatro opciones para asegurar buenos resultados.
- penalty: es la regresión. He puesto como opciones lasso y ridge para asegurar que no habrá overfitting.
- alpha: la inversa de la fuerza de regresión. He puesto valores pequeños para que el impacto de la regresión sea notable. Esto también lo he usado en ridge y lasso.
- poly degree: El grado del polinomio de w para aproximar. He puesto las opciones 1 y 2 en todos los modelos.

Como mejor modelo, he seleccionado el devuelto por crossvalidation: Lasso con alpha=0.00316227, tolerancia de 0.0001 y máximo de iteraciones 1000. Los errores obtenidos son bastante bajos como veremos en la siguiente sección.

10 Estimación por validación cruzada del error E out del modelo. Compárela con E test , ¿que conclusiones obtiene?

Por validación cruzada, el error medio que devuelve sklearn es de un -1.742% siendo bastante bajo y ajustando muy bien al conjunto de validación. El error en el conjunto de trining es de -1.421%. El erro en test (que es desconocido para el modelo entrenado) es tan solo de -1.943%, con lo que nos confirma que el E_{out} es también bajo.

11 Suponga que Ud ha sido encargado de realizar este ajuste para una empresa. ¿Qué modelo les propondría y que error E out les diría que tiene?. Justifique las decisiones.

Misma justificación que en la BBDD 1.