Práctica 4: Clasificación

Curso 2020/2021

PEDRO MANUEL FLORES CRESPO

Tratamiento Inteligente de Datos

Máster en Ingeniería Informática

Índice

1. Introducción

Esta práctica está dedicada a estudiar algoritmos de clasificación. Tenemos entre manos información acerca de una serie de subastas llevadas a cabo en la página web `eBay`. Nuestro objetivo es poder detectar aquellas que sean competitivas (reciben más de dos ofertas) frente a aquellas que no lo son. Así, podremos ver qué características favorecen que una subasta sea competitiva y proponer estrategias tanto para vendedores como a la compañía para mejorar el resultado de las subastas.

Como las prácticas anteriores, se ha llevado a cabo en el lenguaje `Python` y haciendo uso de la plataforma *Google Colaboratory*, cuyo cuaderno se adjunta. Durante el desarrollo de la práctica veremos diferentes algoritmos y los compararemos para decidir cuál es el más adecuado. Utilizaremos técnicas como validación cruzada (con 5 particiones), matrices de confusión, etc.

2. Preprocesamiento

En primer lugar, antes de poder aplicar los algoritmos tenemos que realizar un preprocesamiento adecuado de los datos. Primero vamos a pasar los atributos enteros a categóricos [1]. Estos atributos son `Category`, `currency` y `endDay`. Para el resto de atributos que sí son numéricos se ha realizado una discretización. En todos ellos se ha reducido a 10 el número de valores. Esta se ha hecho mediante la técnica *quantile-based discretization* [2] que fija los intervalos en función de los cuantiles.

3. Árboles de decisión

El primer método que vamos a probar va a ser la clasificación mediante un árbol de decisión [9]. Algunos de los parámetros que recibe es `max_depth` que indica la profundidad máxima del árbol y otro es `criterion` para medir la calidad de la separación.Este último puede tomar valores `gini` o `entropy`. Para no tener que fijar manualmente dichos valores y obtener una medida más objetiva de la bondad de cada una de las configuraciones se va a proceder a hacer una validación cruzada [3]. Para ello, se van a suponer que la profundidad máxima va de 3 a 14 para cada uno de los tipos de criterio. En cada combinación haremos una validación cruzada (de tamaño 5) y calcularemos la media. Nos quedaremos finalmente con el que mejor resultado en media tenga.

Este proceso se puede consultar en el cuaderno con el procedimiento. Una vez realizado, vemos que el mejor resultado se ha obtenido con profundidad 8 y criterio `gini`. Construimos entonces el árbol de decisión obtenido y lo entremos con el 80 % de las instancias ya que el otro 20 % actuará como conjunto de test. Los resultados obtenidos se muestran en la tabla 1.

clase/pred	0	1
0	164	19
1	32	180

clase	*precision*	*recall*	*f1-score*
0	0.84	0.90	0.87
1	0.9	0.85	0.88

Exactitud	0.87

Tabla 1: Resultados del mejor árbol de decisión obtenido tras la validación cruzada (profundidad = 8 y criterio =`ginit`).

4. Regresión logística

El segundo procedimiento a probar es la regresión logística [14]. A pesar de su nombre, es un modelo lineal que se utiliza para clasificación y no para regresión [15]. Como en este caso no tenemos unos parámetros tan claros que especificar como en el caso anterior, vamos a probar directamente los resultados sobre la implementación por defecto. Las matriz de confusión y otra serie de medidas se encuentran en la tabla 2.

clase/pred	0	1
0	158	25
1	45	167

clase	*precision*	*recall*	*f1-score*
0	0.78	0.86	0.82
1	0.87	0.79	0.83

Exactitud	0.82

Tabla 2: Resultados tras aplicar una regresión logística.

5. Gaussian Naive Bayes

Otro método a probar es `Gaussian Naive Bayes` [10]. Como en el caso de la regresión logística, no hay parámetros que "tunear" por lo que vemos los resultados directamente en la tabla 3.

clase/pred	0	1
0	135	48
1	58	154

clase	*precision*	*recall*	*f1-score*
0	0.70	0.74	0.72
1	0.76	0.73	0.74

Exactitud	0.73

Tabla 3: Resultados tras aplicar `Gaussian Naive Bayes`.

6. Máquinas de vector soporte

Nuestra cuarto método de clasificación son las máquinas de vector soporte [11]. En este caso sí nos encontramos como en la situación de los árboles de decisión donde tenemos que decidir alguno de los parámetros del procedimiento. En este caso, se trata del parámetro notado como `C` y que se trata del factor de regularización. Volvemos a repetir por tanto la validación cruzada (de 5 particiones) y vamos a variar dicho parámetro entre 1 y 2.75.

El mejor resultado obtenido (se puede consultar en el cuaderno) es para `C` = 2.5. Los resultados estableciendo ese parámetro se muestran en las tablas 4.

clase/pred	0	1
0	161	22
1	42	170

clase	*precision*	*recall*	*f1-score*
0	0.79	0.88	0.83
1	0.89	0.80	0.84

Exactitud	0.84

Tabla 4: Resultados tras aplicar una máquina de vector soporte son parámetro de regularización 2.5.

7. *Stochastic Gradient Descent*

El siguiente método a probar es clasificación con entrenamiento mediante descenso del gradiente estocástico [13]. En esta ocasión, el hiperparámetro para el que se va aplicar la validación cruzada es el que nota como `alpha`. Se trata de una constante que multiplica al término de regularización. A mayor valor, mayor es la regularización. Para la validación lo hemos variado entre 0.0001 y 0.2 y el mejor resultado en media se ha conseguido para `alpha` = 0.0281. En la tabla 5 se muestran los resultados de las pruebas.

clase/pred	0	1
0	161	22
1	42	170

clase	*precision*	*recall*	*f1-score*
0	0.79	0.88	0.83
1	0.89	0.80	0.84

Exactitud	0.84

Tabla 5: Resultados tras aplicar una clasificación usando SGD con parámetro `alpha` = 0.0281.

8. *K-Neighbors Classifier*

Por último se va a utilizar una clasificación que implementa el procedimiento *k-nearest neighbors* [16]. Para la validación cruzada se ha variado el número de vecinos (de 3 a 14) y la función de peso usada durante la predicción (`uniform` y `distance`). Tras la validación, los mejores resultados se han conseguido para 12 vecinos y usando la función `distance`. Los resultados se pueden ver en 6.

clase/pred	0	1
0	152	31
1	47	165

clase	*precision*	*recall*	*f1-score*
0	0.76	0.83	0.80
1	0.84	0.78	0.81

Exactitud	0.80

Tabla 6: Resultados tras aplicar una clasificación usando SGD con parámetro `alpha` = 0.0281.

9. Selección del mejor método

En las diferentes tablas hemos estado viendo las matrices de confusión obtenidas y los valores de exactitud, precisión, *recall* y *f1-score*. Parece que el árbol de decisión es que mejor resultados aporta. Para afianzar dicha conclusión vamos a usar las curvas ROC [7] y el área que encierra [8].

Clasificador	Área
Árbol de decisión	0.869
Regresión logística	0.825
Naive Bayes	0.732
Máquinas Vector Soporte	0.840
SGD	0.840
KNeighborsClassifier	0.804

Tanto en la figura 1 como en la tabla 9 vemos que el mejor resultado se obtiene con el árbol de decisión.

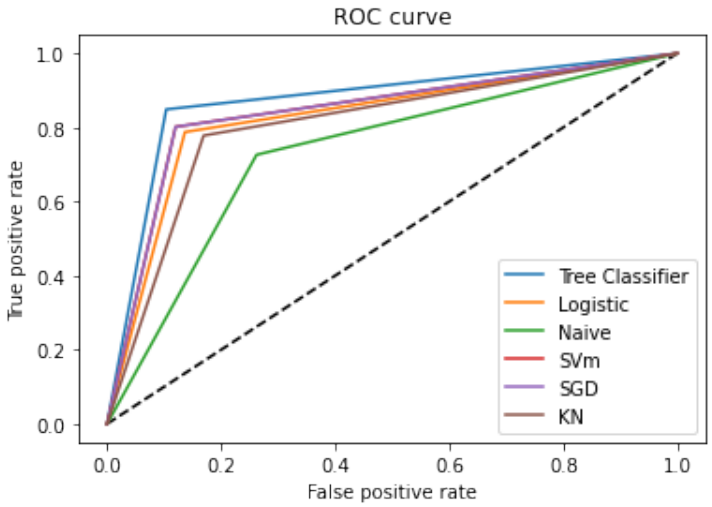

Figura 1: Curvas ROC para los modelos obtenidos.

10. Conclusiones

Durante la práctica hemos visto diferentes procedimientos de clasificación. Se han usado métodos como la validación cruzada para obtener los mejores parámetros en algunos casos. También se han usado matrices de confusión y la curva ROC para ver cuál es el que mejor resultados aporta. Tras ello, el mejor resultado se ha conseguido con un árbol de decisión con profundidad máxima 8 y criterio `gini`.

Para obtener conclusiones acerca de las mejores estrategias que deberían tomar los vendedores y la propia empresa para que las subastas sean más competitivas podemos mirar el árbol de decisión obtenido. Este se adjunta en el fichero `subastas.pdf`. Tras ver las correspondencia después de convertir las variables categóricas y la discretización, para establecer las conclusiones sobre los datos originales, podemos decir a grandes rasgos que:

- Si el precio inicial es menor de 1.15 y la categoría es automotivación, libros, negocios, ropa, monedas/sellos o coleccionables tiene gran probabilidad de ser competitiva. De lo contrario, puede que no lo sea.
- Si el precio de comienzo está entre 1.15 y 5.685. Para el precio final menor que 4.75 es probable que sea competitiva. Pero si el precio final está entre 4.75 y 12.45 no lo será.
- Si el precio de inicio es mayor de 5.685 y el de salida inferior a 12.45 es probable que sea competitiva.
- De otro modo, si el precio de inicio está entre 5.685 y 18.5 es probable que la subasta no sea competitiva.
- Para precios de inicio mayores de 18.5, si el *rating* del vendedor es menor de 573 la subasta no será competitiva. Si es mayor, es muy probable que lo sea excepto para las categorías de música, fotografía, deporte, juguetes y porcelana.

Vemos que las características que más influyen son el precio de salida, la valoración del vendedor y en algunos casos la categoría. En base a los resultados anteriores un vendedor se podría ajustar según el tipo de producto que desee vender. Por ejemplo, si tiene una valoración alta (mayor que 573) podría poner un precio más elevado. Por su parte, la empresa podría fomentar la valoración de los vendedores para que estos tengan un *rating* más alto y de ese modo, los artículos podrían tener un precio de salida más alto. También podría enfocarse en algunas categorías (como música, fotografía, deporte . . .) para mejorar la calidad de los productos ofertados y que tengan más probabilidades de ser comprados.

Además, en algunos rangos de precios podría fijar unos incrementos en las pujas. Por ejemplo, vemos que entre precios de salida de 1.15 y 5.6 si hay un precio muy alto desde el principio la puja no es competitiva.

Referencias

1. Guide to Encoding Categorical Values in Python: https://pbpython.com/categorical-encoding.html
2. pandas.qcut https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.qcut.html
3. Cross-validation: evaluating estimator performance https://scikit-learn.org/stable/modules/cross_validation.html
4. sklearn.model_selection.cross_val_score https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html#sklearn.model_selection.cross_val_score
5. sklearn.metrics.roc_curve https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
6. sklearn.model_selection.KFold https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html?highlight=kfold#sklearn.model_selection.KFold
7. Feature transformations with ensembles of trees https://scikit-learn.org/stable/auto_examples/ensemble/plot_feature_transformation.html#sphx-glr-auto-examples-ensemble-plot-feature-transformation-py
8. sklearn.metrics.roc_auc_score https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#sklearn.metrics.roc_auc_score
9. sklearn.tree.DecisionTreeClassifier https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
10. sklearn.naive_bayes.GaussianNB https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB
11. sklearn.svm.SVC https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
12. Supervised Learning in scikitLearn https://scikit-learn.org/stable/supervised_learning.html
13. sklearn.linear_model.SGDClassifier https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier
14. sklearn.linear_model.LogisticRegression https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
15. *Logistic Regression* https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
16. sklearn.neighbors.KNeighborsClassifier https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html