Satisfacción de pasajeros en aerolíneas

Airline Passenger Satisfaction

Proyecto Final Aprendizaje Automático y Big Data

Curso 2021-2022

Autores

Unai Piris Ibañez

Javier Gómez Moraleda

Facultad de Informática

Universidad Complutense de Madrid

ÍNDICE DE CONTENIDOS

1. Análisis y preprocesamiento de los datos

1.1. Introducción

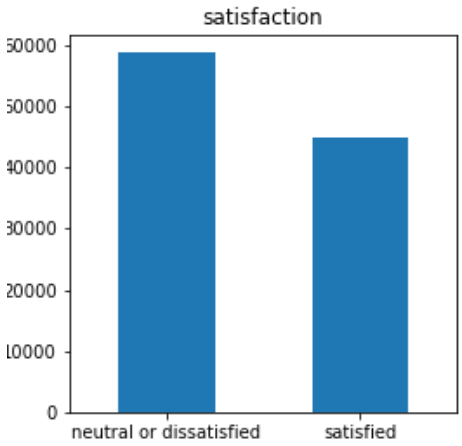
Este proyecto consiste en aplicar las técnicas y conocimientos adquiridos en la asignatura de Aprendizaje Automático y Big Data sobre un dataset de Kaggle.

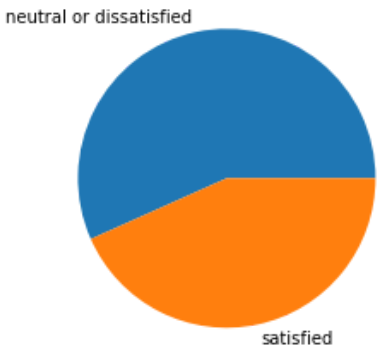
El dataset elegido consiste en la satisfacción de más de los pasajeros que realizaron un vuelo en una aerolínea, junto con 25 columnas de información como la edad del pasajero, la distancia o la clase en la que viajaba. La variable a predecir es la satisfacción, que puede tener dos valores posibles: satisfecho o neutral/no satisfecho. El dataset puede encontrarse en el siguiente enlace: https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction

Nuestro objetivo es determinar cuál será la satisfacción de un pasajero en base a la información contenida en las columnas.

1.2. Análisis de los datos

En primer lugar, observando la distribución de la variable a predecir, hay un mayor número de personas neutrales o no satisfechas en comparación con las satisfechas, pero sin demasiada diferencia.

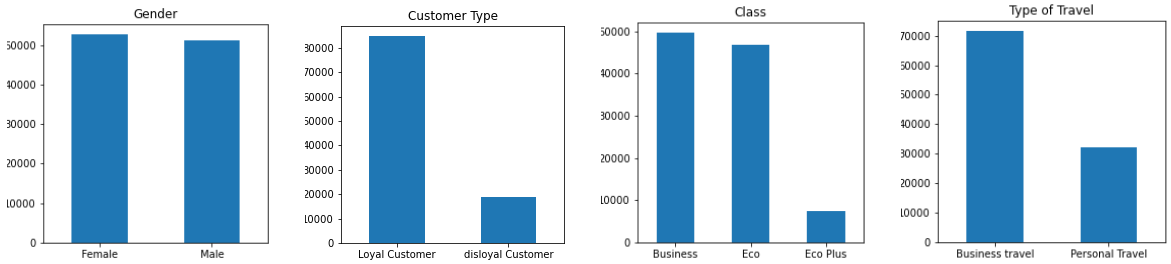




En segundo lugar, haciendo uso de un histograma, se va a realizar un análisis de las distribuciones de todas las variables.

Tanto la columna del número de fila (*Unnamed*) como la del identificador, no aportan nada, ya que cada fila tiene un valor único y deberán ser eliminadas.

Además, faltan las distribuciones de aquellas variables que no son numéricas. Se muestran a continuación.



Como los valores de estas columnas no son numéricos, será necesario aplicarles una transformación.

1.3. Preprocesamiento

El dataset original estaba formado por dos ficheros, uno de entrenamiento y otro de test. El primero de ellos tenía aproximadamente 104k filas, y el segundo 25k. Hemos considerado que el primero es suficientemente grande como para descartar el segundo. Además, como los datos son aleatorios, ambos ficheros tenían una distribución de variables casi idéntica.

El primer procesamiento que se realizó fue la eliminación de la columna que indica el número de fila y la columna con los identificadores de cada una, puesto que no aportan ningún valor.

Después, se realizó un análisis de valores nulos y se observó que la columna "Arrival Delay in Minutes" contenía 310 valores nulos, por lo que directamente fueron eliminadas, ya que era la opción más sencilla y no son representativas respecto al número total de filas.

A la hora de aplicar los algoritmos nos dimos cuenta de que el número de filas era un poco desorbitado, y no teníamos la suficiente potencia computacional para su aplicación, por lo que se ha decidido reducir el dataset a un total de 10k filas. Para ello, hemos seleccionado las primeras 10k y descartado las demás, ya que, como se ha mencionado anteriormente, los datos no siguen un orden. Mediante una

comprobación de distribuciones se ha comprobado que apenas había variación respecto al dataset original.

Por último, como algunas variables no eran numéricas, hemos aplicado el método *onehot* sobre ellas, generando una columna por cada valor diferente de la fila. De esta forma las variables *Gender, Customer Type* y *Type of Travel,* han pasado a ser dos columnas booleanas y la variable *Class,* tres columnas.

El dataset resultante tendrá un total de 10k filas x 28 columnas.

1.4. Separación en conjuntos

Para realizar la separación en conjuntos, vamos a realizar una separación del 60% para el conjunto de entrenamiento, 20% para el de validación y el último 20% para el de test. El resultado es que destinamos 6000 ejemplos para el entrenamiento, 2000 para la validación y 2000 para el test.

Para terminar, será necesario realizar una normalización de las variables, ya que los diferentes rangos pueden afectar al modelo entrenado. Lo único que habría que hacer es obtener la media y desviación típica de los datos de entrenamiento para después, aplicar la siguiente fórmula.

$$z = \frac{x_i - \mu}{\sigma}$$

En general, teníamos dudas sobre la normalización de las variables booleanas, pero hemos decidido aplicar la normalización a todas las columnas ya que al probarlo, la diferencia era mínima.

2. Regresión Logística

En esta sección vamos a aplicar el algoritmo de Regresión logística regularizada y sin regularizar visto en la prácticas 2 y 3.

Aplicaremos dos tipos de Hipótesis:

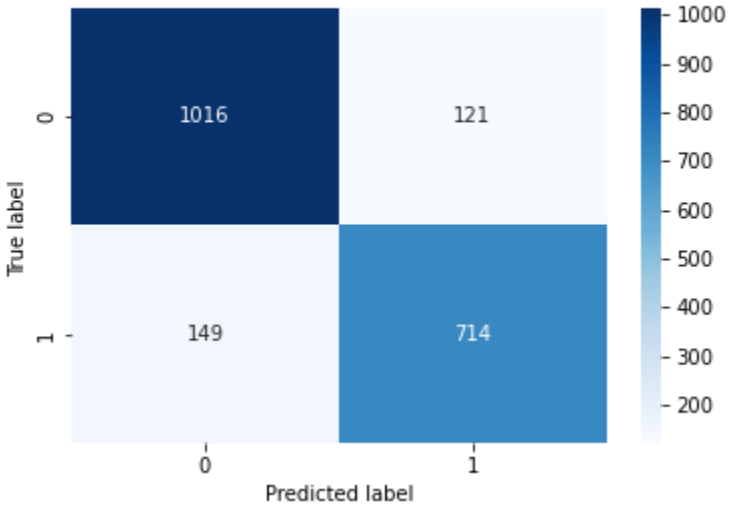
- Hipótesis lineal: Tomamos como atributos únicamente los datos originales.
- Hipótesis no lineal: Tomamos como atributos una combinación polinómica de los datos originales.

Para realizar el entrenamiento, vamos a hacer pruebas con los siguientes valores del parámetro λ: 0.01, 0.1, 1, 3, 7 y 10.

2.1. Regresión Logística sin regularizar

En general los resultados obtenidos son bastante buenos, en el caso de la hipótesis lineal se obtiene un porcentaje de aciertos del 86.5% sobre los datos de test.

La siguiente figura ilustra la matriz de confusión de los resultados obtenidos.



En este caso, clasifica correctamente 1016 ejemplos de personas neutrales insatisfechas y 714 satisfechas, obteniendo sólo 149+121 ejemplos mal clasificados.

Por último, se muestra la tabla con las distintas métricas de evaluación, que, en general, realiza una buena clasificación para las distintas clases.

Clase	Accuracy	Precision	Recall	F1-Score
Neutral or dissatisfied	-	0.89	0.87	0.88
Satisfied	-	0.83	0.86	0.84
Total	0.86	0.86	0.86	0.86

En el caso de la hipótesis no lineal los resultados mejoran respecto a la hipótesis lineal. Obtenemos un porcentaje de aciertos de 94.1% pero esta vez sobre los datos de validación, ya que como veremos a continuación la versión regularizada da un resultado mejor y es esta la que utilizamos para los resultados definitivos.

2.2. Regresión Logística regularizada

Para la elección de los valores óptimos se ha realizado un entrenamiento utilizando el conjunto de entrenamiento y se han obtenido los valores del parámetro λ escogiendo aquellos que daban un mayor porcentaje de acierto sobre el conjunto de validación.

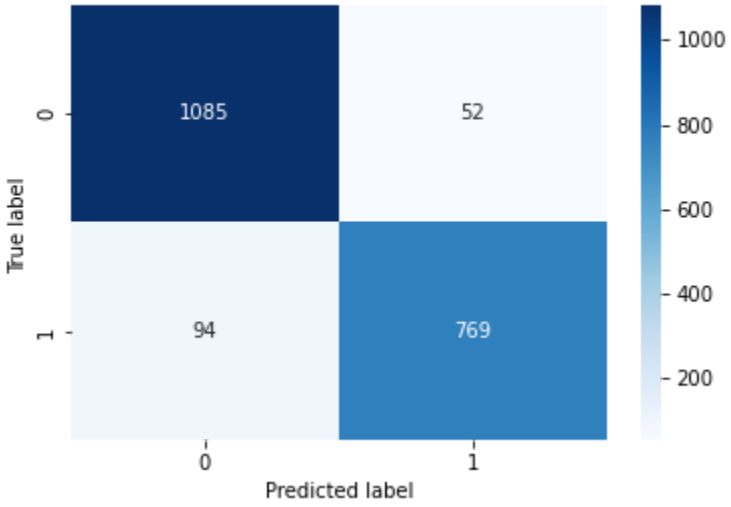
En el caso de la hipótesis lineal el parámetro de regularización no mejora el resultado obtenido con ninguno de los valores del del parámetro λ escogidos. Por tanto la versión óptima es con $\lambda = 0$ cuya matriz de confusión es la mostrada anteriormente.

En el caso de la hipótesis no lineal el parámetro de regularización óptimo lo obtenemos con $\lambda = 10$ con un porcentaje de aciertos de 94.15% sobre los datos de validación,

obteniendo una pequeña mejoría del 0.05% respecto a la versión no lineal sin regularizar.

Una vez aplicado sobre el conjunto de test se observa un porcentaje de aciertos de 92,7% obteniendo así una gran mejora sobre las versiones lineales del 7.65%.

La siguiente figura ilustra la matriz de confusión de los resultados obtenidos.



En este caso, clasifica correctamente 1085 ejemplos de personas neutrales insatisfechas y 769 satisfechas, obteniendo sólo 52+94 ejemplos mal clasificados.

Por último, se muestra la tabla con las distintas métricas de evaluación, que, en general, realiza una buena clasificación para las distintas clases.

Clase	Accuracy	Precision	Recall	F1-Score
Neutral or dissatisfied	-	0.95	0.92	0.94
Satisfied	-	0.89	0.94	0.91
Total	0.93	0.92	0.93	0.93

En conclusión podemos observar que tanto en el caso de las hipótesis lineales como el las no lineales el parámetro λ apenas afecta al resultado final de nuestras predicciones y que las hipótesis no lineales mejoran un 7.65% a las hipótesis no lineales.

3. Redes Neuronales

En este apartado vamos a aplicar los conocimientos aprendidos en las prácticas 3 y 4, en especial en esta última. Para ello, vamos a utilizar todo el código desarrollado en las mismas, aunque con algunas modificaciones.

En la práctica 4, el objetivo era clasificar números del 0 al 9, por lo que el número de neuronas de la última capa eran 10. En este caso, sólo tendremos dos resultados posibles, por lo que aplicaremos el método *onehot* en la salida. También hemos eliminado algunos fragmentos concretos de código que se realizaban para ajustarse al dataset original.

Para llevar a cabo el entrenamiento, vamos a realizar pruebas con distintos valores de los parámetros: λ (parámetros de regularización) y el número de iteraciones. Además, también probaremos a variar el número de neuronas de la capa oculta, analizando cómo afecta al resultado final. Los valores elegidos son los siguientes:

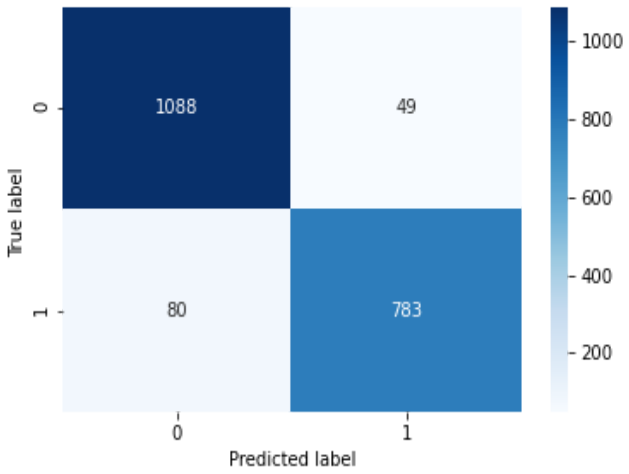
- Parámetro de regularización λ: 0.01, 0.1, 1, 3, 7 y 10.
- Número de iteraciones: desde 0 hasta 200 en intervalos de 10.
- Número de neuronas: 5, 15, 25 y 50.

Para la elección de los valores óptimos, se ha realizado un entrenamiento utilizando el conjunto de entrenamiento y se han obtenido los valores de los parámetros escogiendo aquellos que daban un mayor porcentaje de acierto sobre el conjunto de validación. En general, los resultados son bastante buenos independientemente del número de neuronas, ya que todos están en torno al 94%, por lo que no lo consideramos un parámetro decisivo. El término de regularización óptimo se encuentra entre 1 y 3. Y el número de iteraciones se encuentra en el intervalo de 140-190. En la siguiente tabla se pueden observar los resultados detallados.

Número de neuronas	Término de regularización (λ)	Número de iteraciones	Porcentaje de aciertos (%)
5	1	170	94.45
15	1	140	94.55
25	3	190	94.75
50	3	150	94.85

Puesto que todos los resultados son bastante buenos, hemos considerado seleccionar 25 neuronas, con término de regularización 3 y con 200 iteraciones. Después, hemos realizado un entrenamiento utilizando el conjunto de entrenamiento y los parámetros mencionados. Como la elección de los parámetros se ha utilizado seleccionando el mejor porcentaje de aciertos que se obtenía con el conjunto de validación, ahora necesitamos realizar dicha comprobación con otro conjunto distinto, el de test.

La siguiente figura ilustra la matriz de confusión de los resultados obtenidos, con una predominancia de los verdaderos positivos y falsos positivos, indicando un buen funcionamiento. El porcentaje de aciertos obtenidos en este caso es del 93.55%.



En este caso, clasifica correctamente 1088 ejemplos de personas neutrales insatisfechas y 783 satisfechas, obteniendo sólo 49 + 80 ejemplos mal clasificados.

Por último, se muestra la tabla con las distintas métricas de evaluación, que, en general, realiza una buena clasificación para las distintas clases.

Clase	Accuracy	Precision	Recall	F1-Score
Neutral or dissatisfied	-	0.96	0.93	0.94
Satisfied	-	0.91	0.94	0.92
Total	0.94	0.93	0.94	0.94

En conclusión, consideramos que realiza una excelente clasificación y mejora los modelos de regresión logística, en especial los que no utilizan polinomios. Sin embargo, las redes neuronales tienen un componente aleatorio a la hora de realizar la inicialización de los valores de theta, por lo que es posible la obtención de resultados distintos. En nuestro caso esta aleatoriedad puede variar los resultados en torno a un 2 puntos, pero generalmente siempre ha superado el 90% de aciertos en distintas ejecuciones.

4. Support Vector Machines

En este apartado vamos a aplicar los conocimientos aprendidos en la práctica 6. Para ello, vamos a utilizar todo el código desarrollado, aplicando el algoritmo SVM de la librería *sckit-learn*.

Para aplicar nuestro dataset, hemos realizado cambios mínimos sobre el código y añadido los diagramas para conseguir una mejor visualización de los resultados.

En el primer punto aplicaremos el kernel lineal y en el segundo punto el kernel gaussiano y, en general, hemos obtenido buenos resultados con ambos.

4.1. Kernel Lineal

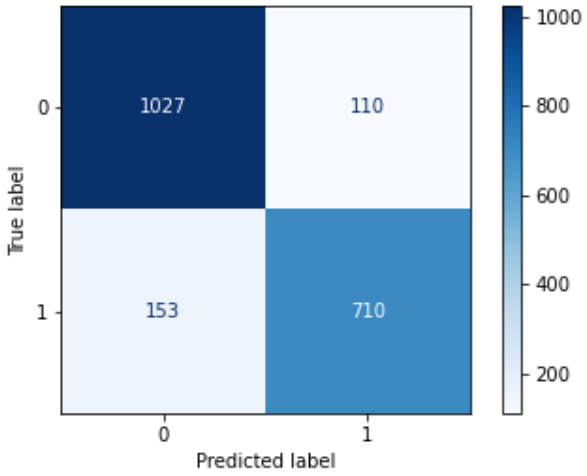
Para el desarrollo del kernel lineal, vamos a probar cómo se comporta ante distintos valores del parámetro de regularización C. Un C más alto indica un mayor sobreajuste al modelo, por lo que obtendremos un mejor resultado sobre el conjunto de entrenamiento.

Los valores de C utilizados son los siguientes: 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30 y 100.

En primer lugar, hemos entrenado el modelo utilizando el conjunto de entrenamiento para después comprobar el porcentaje de aciertos sobre el de validación. De esta forma, y probando diferentes valores de C, hemos escogido aquel que nos daba una mayor tasa de aciertos. El valor de C óptimo obtenido fue 0.03, con un porcentaje de aciertos del 86,85% sobre el conjunto de validación.

Una vez seleccionado el parámetro, toca entrenar el modelo de nuevo sobre el conjunto de entrenamiento, y realizar el porcentaje de aciertos sobre otro conjunto distinto al de validación, el de test.

La siguiente figura ilustra la matriz de confusión de los resultados obtenidos, con una predominancia de los verdaderos positivos y falsos positivos, indicando un buen funcionamiento. El porcentaje de aciertos obtenidos en este caso es del 87%.



En este caso, clasifica correctamente 1027 ejemplos de personas neutrales insatisfechas y 710 satisfechas, obteniendo sólo 153 + 110 ejemplos mal clasificados.

Por último, se muestra la tabla con las distintas métricas de evaluación, que, en general, realiza una buena clasificación para las distintas clases.

Clase	Accuracy	Precision	Recall	F1-Score
Neutral or dissatisfied	-	0.90	0.87	0.89
Satisfied	-	0.82	0.87	0.84
Total	0.87	0.86	0.87	0.87

En conclusión, hace una buena clasificación para ambas clases, aunque funciona un poco peor que las redes neuronales.

4.2. Kernel Gaussiano

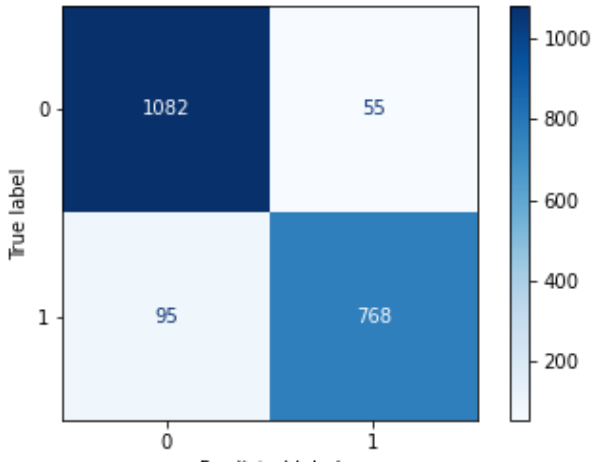
Para el desarrollo del kernel gaussiano, vamos a probar cómo se comporta ante distintos valores del parámetro de regularización C y σ. Los valores elegidos son los siguientes:

- Parámetro de regularización 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30 y 100.
- Parámetro σ: 0.01, 0.03, 0.1, 0.3, 1, 3, 10 y 30.

En primer lugar, hemos entrenado el modelo utilizando el conjunto de entrenamiento para después comprobar el porcentaje de aciertos sobre el de validación. De esta forma, y probando diferentes valores de C y de σ, hemos escogido aquel que nos daba una mayor tasa de aciertos. El valor de C óptimo obtenido fue 30 y de σ fue 10, con un porcentaje de aciertos del 92,5% sobre el conjunto de validación.

Una vez seleccionado los parámetros, toca entrenar el modelo de nuevo sobre el conjunto de entrenamiento, y realizar el porcentaje de aciertos sobre otro conjunto distinto al de validación, el de test.

La siguiente figura ilustra la matriz de confusión de los resultados obtenidos, con una predominancia de los verdaderos positivos y falsos positivos, indicando un buen funcionamiento. El porcentaje de aciertos obtenidos en este caso es del 93%.



En este caso, clasifica correctamente 1082 ejemplos de personas neutrales insatisfechas y 768 satisfechas, obteniendo sólo 95 + 55 ejemplos mal clasificados.

Por último, se muestra la tabla con las distintas métricas de evaluación, que, en general, realiza una buena clasificación para las distintas clases.

Clase	Accuracy	Precision	Recall	F1-Score
Neutral or dissatisfied	-	0.95	0.92	0.94
Satisfied	-	0.89	0.93	0.91
Total	0.93	0.92	0.93	0.92

Observando los resultados de la tabla, gracias a la utilización del kernel gaussiano, hemos conseguido mejorar los resultados de la clasificación media en cinco puntos, lo que es una mejora bastante significativa, acercándonos al resultado del modelo de redes neuronales.

5. Conclusiones

Por último, vamos a recopilar todos los resultados obtenidos en una única tabla.

En general, los modelos lineales dan peores resultados que los no lineales. Es por eso que tanto la regresión logística lineal comparada con la no lineal, como el SVM lineal comparado con el Gaussiano, dan unos resultados de 5 puntos menos. El modelo con el que hemos obtenido mejores resultados ha sido las redes neuronales, aunque con poca diferencia respecto a los modelos no lineales.

Modelo	Accuracy	Precision	Recall	F1-Score
Regresión logística lineal	0.86	0.86	0.86	0.86
Regresión logística no lineal	0.93	0.92	0.93	0.93
Redes neuronales	0.94	0.93	0.94	0.94
SVM Lineal	0.87	0.86	0.87	0.87
SVM Gaussiano	0.93	0.92	0.93	0.92