



ITESM- Campus Puebla

NAATIK

Momento de Retroalimentación: Reto Modelo y Refinamiento

Inteligencia artificial avanzada para la ciencia Datos II

Integrantes Equipo 1:

Myroslava Sánchez Andrade A01730712
José Antonio Bobadilla García A01734433
Karen Rugerio Armenta A01733228
Alejandro Castro Reus A01731065

Fecha: 11/11/2022

NAATIK es una empresa enfocada en el desarrollo y aplicación de Inteligencia Artificial y Ciencia de Datos para brindar soluciones. El reto consiste en el análisis de un conjunto de datos de una compañía telefónica que contiene información de los clientes, tales como servicios contratados, información de cuenta, información demográfica y su permanencia (booleano).

La solución consiste en la predicción de la permanencia de un grupo cliente dados un set de datos de churn, y análisis de clientes para el desarrollo de un modelo de retención. Para poder seleccionar el modelo de clasificación se realizaron 5 modelos de Machine Learning:

- Regresión Logística (**Modelo de Benchmark**)
- Multilayer Perceptron
- Decision Tree
- Random Forest
- Convolutional Neural Network

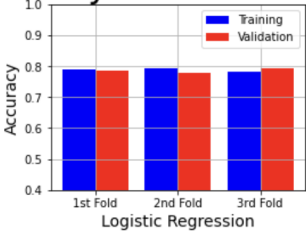
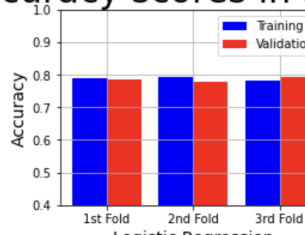
Es importante mencionar que como parte de la solución del reto y por requerimiento de los socios formadores, se necesita tener un modelo “generalizado” que permita la entrada de cualquier archivo churn. Es por ello que la elección del entrenamiento del modelo la tomaremos haciendo uso del dataset ‘WA_Fn-UseC_-Telco-Customer-Churn’ esperando que su performance sea similar con los nuevos inputs de los socios formadores.

Para la selección de este modelo se probarán distintas configuraciones, esperando obtener un buen nivel de precisión con la clasificación (si es churn o no es churn). Así mismo, como parte de la selección del modelo se considera el requerimiento de que el modelo debe ser fácil y rápido de procesar, ya que no se estará trabajando con un servidor sino con una computadora perteneciente a los socios formadores. Es importante resaltar que en el dataset probado se tiene un total de 7,043 datos, de los cuales el 73% son clientes sin churn (5,174) y el 27% son clientes con churn (1,869).

Regresión logística (modelo de benchmark)

Como modelo de benchmarking hemos decidido utilizar una regresión logística, que es una técnica simple de aprendizaje automático para clasificación y se utilizará como punto de comparación para otros modelos a generar.

Modelo	Regresión logística	Regresión logística (mejorada)
Configuración	penalty: none tolerance: 1e-10 regularization strength: 0.5 random state: 100	penalty: l2 tolerance: 1e-4 regularization strength: 1.0 random state: 100
MSE Train	0.2123	0.2127
Train Accuracy	0.7877	0.7873
MSE Test	0.2112	0.2107

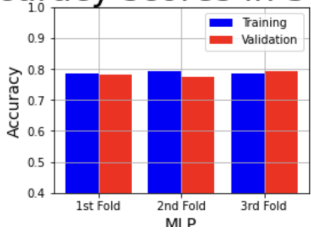
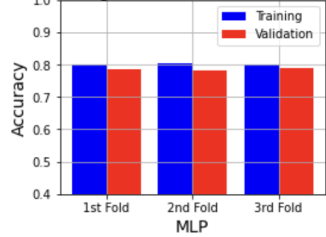
Test Accuracy	0.7888	0.7893																		
Folds Validation	Accuracy scores in 3 Folds 	Accuracy scores in 3 Folds 																		
Matriz de confusión	<table> <tr> <td></td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1419</td><td>314</td></tr> <tr> <td>1</td><td>132</td><td>247</td></tr> </table>		0	1	0	1419	314	1	132	247	<table> <tr> <td></td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1420</td><td>314</td></tr> <tr> <td>1</td><td>131</td><td>247</td></tr> </table>		0	1	0	1420	314	1	131	247
	0	1																		
0	1419	314																		
1	132	247																		
	0	1																		
0	1420	314																		
1	131	247																		

Sobre los resultados, podemos observar, que las métricas tanto de error como de accuracy, resultan ser mejores en el train para el modelo de regresión logística inicial y en el test para el modelo de regresión logística mejorado. La diferencia realmente no es significativa ya que varían por milésimas, así mismo en el k-folds validation test ambos modelos tienen resultados similares, en cuanto a la matriz de confusión se puede observar que en ambos modelos se detecta la misma cantidad de positivos que si son positivos, sin embargo el primer modelo detecta un falso negativo más que el segundo modelo, lo que hace el segundo modelo uno mejor para detectar y retener clientes con churn. Por lo tanto, se ha decidido tomar como modelo de benchmark el modelo mejorado, por su performance en el set de prueba y sus resultados en la matriz de confusión.

Multilayer Perceptron

Un MLP puede ser ocupado como modelo de clasificación cuando los inputs tienen asignada una clase o etiqueta. En este caso se considera el churn como etiqueta. Las configuraciones y los resultados se encuentran a continuación.

Modelo	Multilayer Perceptron	Multilayer Perceptron (mejorado)
Configuración	Maximum number of iterations: 100 Hidden layer sizes: (50,50) activation: Logistic Solver: Adam Random state: 1	Maximum number of iterations: 200 Hidden layers: (7, 5, 3) neuronas activation: relu Solver: lbfgs Random state: 1

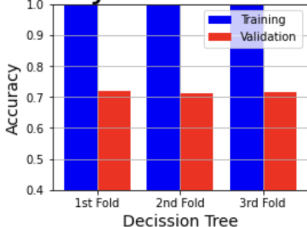
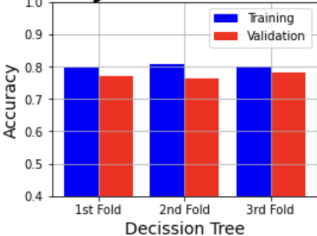
MSE Train	0.2170	0.2024																		
Train Accuracy	0.7830	0.79760																		
MSE Test	0.2050	0.2027																		
Test Accuracy	0.7950	0.7973																		
Folds Validation	<p>Accuracy scores in 3 Folds</p> 	<p>Accuracy scores in 3 Folds</p> 																		
Matriz de confusión	<table> <thead> <tr> <th></th><th>0</th><th>1</th></tr> </thead> <tbody> <tr> <th>0</th><td>1381</td><td>263</td></tr> <tr> <th>1</th><td>170</td><td>298</td></tr> </tbody> </table>		0	1	0	1381	263	1	170	298	<table> <thead> <tr> <th></th><th>0</th><th>1</th></tr> </thead> <tbody> <tr> <th>0</th><td>1412</td><td>289</td></tr> <tr> <th>1</th><td>139</td><td>272</td></tr> </tbody> </table>		0	1	0	1412	289	1	139	272
	0	1																		
0	1381	263																		
1	170	298																		
	0	1																		
0	1412	289																		
1	139	272																		

Para este modelo, los resultados del MLP mejorado se observan considerablemente mejor, ya que tanto las métricas de accuracy y error obtienen mejores resultados en test y train de este modelo. Así mismo se obtuvo una mejora de 1.31% en el Train Accuracy y de 1.01% en Test Accuracy en comparación con el modelo de benchmark. Por otro lado, los resultados del k-folds validation lucen similares, pero se puede observar menos variación en el modelo mejorado.

Sobre la matriz de confusión, podemos observar que el modelo mejorado detecta 25 positivos que son positivos más, lo que equivale a una mejora del 10.12% en detección de churn, sin embargo en la detección de negativos, detectó 8 negativos que si son negativos menos, lo que no es problema, ya que estos entraron en los falsos positivos y no es una cantidad considerable, tomando en cuenta el total de los datos.

Decision Tree

Los árboles de decisión son un modelo de predicción utilizado en la inteligencia artificial para realizar clasificaciones. La estructura natural de un árbol se recorre secuencialmente evaluando el estatuto lógico de cada nodo hasta llegar a la final de la predicción y evaluar un “sí” o un “no”, en este caso de la variable churn.

Modelo	Decision Tree	Decision Tree (mejorado)																								
Configuración	Criterion: log_loss Splitter: best min_samples_split: 2 Max depth: None Random state: 0	Criterion: gini Splitter: random min_samples_split: 50 Max depth: None Random state: 0																								
MSE Train	0.0063	0.1892																								
Train Accuracy	0.9937	0.8108																								
MSE Test	0.2670	0.2202																								
Test Accuracy	0.7330	0.7799																								
Folds Validation	<div>Accuracy scores in 3 Folds</div>  <table><thead><tr><th>Fold</th><th>Training Accuracy</th><th>Validation Accuracy</th></tr></thead><tbody><tr><td>1st Fold</td><td>0.9937</td><td>0.7330</td></tr><tr><td>2nd Fold</td><td>0.9937</td><td>0.7330</td></tr><tr><td>3rd Fold</td><td>0.9937</td><td>0.7330</td></tr></tbody></table>	Fold	Training Accuracy	Validation Accuracy	1st Fold	0.9937	0.7330	2nd Fold	0.9937	0.7330	3rd Fold	0.9937	0.7330	<div>Accuracy scores in 3 Folds</div>  <table><thead><tr><th>Fold</th><th>Training Accuracy</th><th>Validation Accuracy</th></tr></thead><tbody><tr><td>1st Fold</td><td>0.8108</td><td>0.7799</td></tr><tr><td>2nd Fold</td><td>0.8108</td><td>0.7799</td></tr><tr><td>3rd Fold</td><td>0.8108</td><td>0.7799</td></tr></tbody></table>	Fold	Training Accuracy	Validation Accuracy	1st Fold	0.8108	0.7799	2nd Fold	0.8108	0.7799	3rd Fold	0.8108	0.7799
Fold	Training Accuracy	Validation Accuracy																								
1st Fold	0.9937	0.7330																								
2nd Fold	0.9937	0.7330																								
3rd Fold	0.9937	0.7330																								
Fold	Training Accuracy	Validation Accuracy																								
1st Fold	0.8108	0.7799																								
2nd Fold	0.8108	0.7799																								
3rd Fold	0.8108	0.7799																								
Matriz de confusión	<table><tr><td></td><td></td><td>0</td><td>1</td></tr><tr><td>0</td><td>1301</td><td>279</td><td></td></tr><tr><td>1</td><td>285</td><td>247</td><td></td></tr></table>			0	1	0	1301	279		1	285	247		<table><tr><td></td><td></td><td>0</td><td>1</td></tr><tr><td>0</td><td>1430</td><td>362</td><td></td></tr><tr><td>1</td><td>121</td><td>199</td><td></td></tr></table>			0	1	0	1430	362		1	121	199	
		0	1																							
0	1301	279																								
1	285	247																								
		0	1																							
0	1430	362																								
1	121	199																								

En este caso nos encontramos con resultados parecidos al modelo de benchmark, en donde los resultados tanto del MSE como del Accuracy del Train son muy buenos, en realidad el resultado del set de prueba con 99% de accuracy es muy prometedor, sin embargo con el set de Test sólo obtiene un 73% lo que significa un overfitting en el modelo de entrenamiento. Sin embargo con el algoritmo mejorado, aunque sólo alcanza un 81% con el set de prueba, con el set de entrenamiento se alcanza un accuracy de casi 78% lo que lo hace un modelo más estable pero aún así no mejor que el modelo de benchmark, el cual obtiene un 78.73% en train accuracy y un 78.93% en test accuracy. Así mismo, si se analiza el error el primer modelo tiene un error muy bajo, lo que es ideal, pero con el set de prueba se dispara el valor más alto de error que se haya observado hasta ahora.

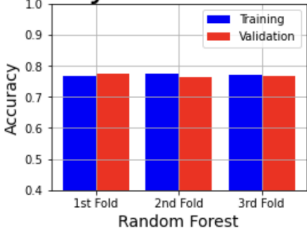
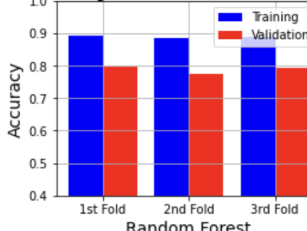
Aplicando k-folds validation observamos el mismo comportamiento, básicamente el primer modelo tiene un performance impresionante con el set de train y un performance muy pobre con el set de validación. Mientras que con el segundo modelo se observa ese equilibrio que se busca entre los resultados de train y validación. La matriz de confusión, por otro lado, nos muestra resultados muy decepcionantes, ya que detecta 48 positivos menos, lo que

equivale a un 19.43% menos, pero se tiene una mejora de 10 usuarios en la detección de negativos que si son negativos, sin embargo lo que queremos es detectar clientes con churn, por lo cual es una “mejora” poco considerable para el modelo.

Analizando la matriz de confusión del primero modelo (el no mejorado) no se tiene más que un resultado igual en positivos que si son positivos en comparación con el benchmark y una cantidad mucho menor en detección de negativos que son negativos (119) lo que equivale a un 8.38% más de falsos positivos, poco ideal para los clientes.

Random Forest

Los bosques aleatorios o los bosques de decisiones aleatorias son un método de aprendizaje comúnmente utilizado para la clasificación. Funciona construyendo un grupo de árboles de decisión al momento del entrenamiento. Para propósitos de clasificación, la salida resultante de un bosque aleatorio es la clase seleccionada por la mayoría de los árboles.

Modelo	Random Forest	Random Forest (mejorado)																		
Configuración	max_depth: 2 criterion: gini random_state: 0 max_features: sqrt	max_depth: 10 criterion: entropy random_state: 0 max_features: log2																		
MSE Train	0.2294	0.1320																		
Train Accuracy	0.7706	0.8680																		
MSE Test	0.2287	0.2093																		
Test Accuracy	0.7713	0.7907																		
Folds Validation	Accuracy scores in 3 Folds 	Accuracy scores in 3 Folds 																		
Matriz de confusión	<table> <tr> <td></td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1507</td><td>439</td></tr> <tr> <td>1</td><td>44</td><td>122</td></tr> </table>		0	1	0	1507	439	1	44	122	<table> <tr> <td></td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1412</td><td>303</td></tr> <tr> <td>1</td><td>139</td><td>258</td></tr> </table>		0	1	0	1412	303	1	139	258
	0	1																		
0	1507	439																		
1	44	122																		
	0	1																		
0	1412	303																		
1	139	258																		

Este modelo obtiene mejores resultados con la configuración mejorada. En general, es decir tanto en test y train se obtienen mejores resultados de MSE y Accuracy. Comparado con el modelo de benchmark se tiene una mejora del 10.25% en los resultados del train accuracy y de un 0.18% en el test accuracy. En cuanto a los resultados del MSE se obtiene un 0.66% menos de error en el modelo del benchmark del test y un 37.94% menos de error en el entrenamiento comparado con el benchmark. Sin embargo se observa un desequilibrio importante al momento de realizar el k-folds validation, ya que el training se observa con valores de accuracy elevados, mientras que la validación obtiene un accuracy muy por debajo del valor de entrenamiento.

Por otra parte, analizando la matriz de confusión, se observa un total de 11 positivos que si son positivos más que el modelo de benchmark, lo que equivale a un 4.45% más y un total de 8 negativos que si son negativos menos, lo que equivale a un 0.66% no detectado que entran a la categoría de falsos positivos.

Convolutional Neural Network (del arte de Deep Learning)

Una red neuronal típica tendrá una capa de entrada, capas ocultas y una capa de salida. Las CNN están inspiradas en la arquitectura del cerebro. Al igual que una neurona en el cerebro procesa y transmite información por todo el cuerpo, las neuronas artificiales o nodos en las CNN toman entradas, las procesan y envían el resultado como salida. La red neuronal convolucional o CNN es un tipo de red neuronal artificial, que se usa ampliamente para la clasificación de imágenes/objetos.

Modelo	CNN	CNN (mejorado)
Configuración	<pre>-Conv1D(128, 3, activation='relu', input_shape=(x_train.shape[1],1)) -MaxPooling1D(2) -LeakyReLU() -Dropout(0.5) -Dense(32, activation='relu') -LeakyReLU() -Dropout(0.5) -Flatten() -Dense(64, activation='relu') -Dense(2)</pre>	<pre>-Conv1D(32, 2, activation='relu', input_shape=(x_train.shape[1],1)) -Dense(16, activation='relu') -MaxPooling1D(1) -Dropout(0.3) -Conv1D(16, 2, activation='relu') -Flatten() -Dense(16, activation='relu') -Dense(2)</pre>
MSE Train	0.2198	0.2028

Train Accuracy	0.7802	0.7972																		
MSE Test	0.2311	0.2216																		
Test Accuracy	0.7689	0.7784																		
Matriz de confusión	<table> <tr> <td></td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1413</td><td>157</td></tr> <tr> <td>1</td><td>325</td><td>217</td></tr> </table>		0	1	0	1413	157	1	325	217	<table> <tr> <td></td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1389</td><td>181</td></tr> <tr> <td>1</td><td>294</td><td>248</td></tr> </table>		0	1	0	1389	181	1	294	248
	0	1																		
0	1413	157																		
1	325	217																		
	0	1																		
0	1389	181																		
1	294	248																		

En primera instancia se optó por una red robusta para el entrenamiento del modelo, esto resultó en una precisión del 76.89% para el test y 78.02% para el entrenamiento. Esto nos indicaba que había un claro overfit, por lo que optamos por hacerle una modificación y disminuir el número de neuronas que se tenían en las capas oculta, como resultado obtuvimos un 79.72% en el entrenamiento y 77.84% en el test de precisión; a pesar de que esto también indica un overfit, podemos evaluar como mejor modelo al modificado. También podemos observar en las matrices de confusión que se tiene una mejor predicción para las instancias sin churn por más del 1% y que se tiene la misma precisión para las instancias con churn para el modelo mejorado.

Selección del modelo

Ya que es necesario hacer la selección del modelo lo más generalizada posible y comparando los resultados de los 4 modelos con el modelo de benchmark, se ha llegado a la conclusión de utilizar el modelo Multilayer Perceptron (MLP). Ya que es el modelo que detecta la mayor cantidad de churn positivo que si es positivo, lo cual es lo ideal para la resolución de este reto que pretende prolongar la estadía de los clientes haciendo uso de un servicio. De igual manera, cuenta con un accuracy de predicción (con el set de datos de prueba) de un 80% lo cual es bastante bueno, considerando que el dataset cuenta con un 73% de no churn, es el 7% más.

Es importante mencionar que la selección del modelo está pensado en dos factores fundamentales, el primero es que sea lo más generalizado posible, esta es la razón por la cual no se está considerando una modificación de threshold con curvas ROC. Y el segundo es que sea lo mejor posible detectando usuarios con Churn.

Finalmente es importante mencionar que estas configuraciones dan resultados positivos con el dataset mencionado anteriormente, sin embargo, al ingresar un nuevo archivo puede que se tengan resultados similares, lo cual sería lo ideal, pero al ser tan generalizado, también es posible llegar a obtener un resultado poco satisfactorio.