UNIVERSIDAD DE ANTIOQUIA

Inteligencia Artificial para las Ciencias e Ingenierías

Proyecto de semestre – Entrega Final

INTEGRANTES:

Silvio Otero Guzmán

Daniela Gonzáles Estrada

Medellín - Antioquia 2023

Introducción

Descripción del problema

Los datos que se trabajarán fueron obtenidos a partir de cuatro áreas silvestres localizadas en el bosque nacional Roosevelt en carolina del norte. Áreas representadas en bosques con mínima interferencia humana, el follaje actual es más un resultado de procesos ecológicos que de prácticas de manejo de bosques. Con la información suministrada se puede determinar cuáles arboles predominan en el área de cobertura, los datos fueron obtenidos del US Forest Service y del US Geological Survey. Mediante esta información se desea generar un modelo predictivo para determinar qué tipo de árboles tienden a la extinción por ser los de menor cantidad en las áreas analizadas, teniendo en cuenta que en los datos ofrecen información de distintos tipos de árboles y las condiciones en las que se encuentran.

Dataset

El dataset a utilizar será Forest Cover Type Prediction https://www.kaggle.com/competitions/forest-cover-type-prediction/data?select=train.csv que consta de 118 columnas y más de 15 mil instancias, cumpliendo los requisitos solicitados por el proyecto.

Métricas de desempeño

Tomando en cuenta que el problema se ha planteado como una clasificación multi clase las métricas de desempeño a usar para entender de mejor manera el modelo pueden ser las mismas que las usadas con una clasificación binaria. La métrica se calcula para cada clase al procesarla como un problema de clasificación binaria después de agrupar todas las otras clases como pertenecientes a la segunda clase. A continuación, se calcula el promedio de la métrica entre todas las clases para obtener una métrica de promedio macro o de media ponderada. Las que se usarán son las siguientes:

- Accuracy
- Matriz de confusión
- Precisión
- Medición F1

Métricas del negocio

La extinción de ciertas especies de árboles es en algunos casos un fenómeno inevitable, al conocer cuáles de ellos tienden a no ser capaces de adaptarse al cambio en el mundo y tener más posibilidad de extinción plantea una reevaluación en la escogencia de árboles con el fin de reforestar áreas con condiciones que podrían llegar a asemejarse a las mostradas en estos datos.

Exploración descriptiva del dataset

En el archivo *01 - Data Simulation.ipynb* se puede evidenciar las distribuciones que presentan los datos.

En el notebook *02 - Data Exploration.ipynb* es posible evidenciar que a partir del archivo *raw_data.parquet* se comienza a realizar una exploración de datos para conocer los datos, identificar patrones en los datos y graficarlos, y se encontró lo siguiente:

- **Tamaño del dataset:** El tamaño del dataset es (5013).

- **Tipos de datos:** Los tipos de datos que están presentes son int64 y otros object en las variables que se convirtieron a categóricas.

- **Inspección de datos numéricos:** Se calcula el promedio, la desviación estándar, valor mínimo, máximo y percentiles de las columnas numéricas, además, se graficó la matriz de correlaciones.

- **Inspección de variables categóricas:** En las variables transformadas a categóricas se realizó un conteo de las filas correspondientes a cada categoría (High, Medium y Low). Se incluyen también histogramas para encontrar relaciones entre la variable de salida (id) y cada una de las variables categóricas.

Iteraciones de desarrollo

Todas las iteraciones que se realizaron se hicieron con base en el archivo *clean_data.parquet,* archivo de salida generado tras haber realizado un preprocesamiento de datos previo (ver notebook *03 - Data Cleaning.ipynb*) que consistió en lo siguiente:

● **Llenado de datos faltantes:** Debido a que las columnas con datos faltantes son numéricas, el proceso se realiza con la media. Esto a través de la función fillna().

● **Conversión variables categóricas a numéricas:** Se utiliza la estrategia One Hot Encoding, la cual consiste en la creación de una columna para cada valor distinto que exista en la característica que estamos codificando y, para cada registro se marca con un 1 la columna a la que pertenezca dicho registro y se dejan las demás con un valor de 0. Esto es posible gracias a la función get_dummies().

Teniendo esto en cuenta, se enuncian las distintas iteraciones:

Iteración 1: Modelos sin SMOTE

En la primera iteración se llevaron a cabo experimentos dividiendo los datos mediante StratifiedKFold, esto con el fin de que los experimentos se realicen conservando la estratificación de los datos, lo cual es posible visualizarlo en el archivo *04 - Models without SMOTE.ipynb* Los modelos probados fueron los siguientes:

RF (Random Forest)

Para este modelo, se realizaron corridas con árboles [5,10,20,50,100, 150] y variables para la selección del mejor umbral [5,10,15,20,25] a través de combinaciones entre sí.
Por ejemplo, con un número de árboles 50 y variables para selección de mejor umbral 5 se obtuvo una eficiencia (accuracy) de prueba de 94,4317%, y en cuanto a la matriz de confusión se obtuvo lo siguiente:

Para la matriz de confusión se emplean porcentajes para visualizar mejor los resultados de las predicciones del modelo. En cuanto a los verdaderos negativos (TN) se observa

que el modelo clasifica correctamente el árbol más presente con el 73% de los datos pertenecientes a esa clase. A nivel de verdaderos positivos
(TP) no se obtiene un resultado muy favorable, ya que se observa que el modelo predice correctamente el 15% de las veces cuando otros arboles son los que predominan.
En los falsos negativos se obtiene una tasa del 85% mientras que en los falsos positivos se visualiza una tasa del 1.6%.

Gradient Boosting

Con la función GradientBoostingClassifier de sklearn se varía el n_estimators entre los valores [20,50,100,200,300], por ejemplo, empleando un valor de n_estimators en 300, obtenemos una eficiencia de prueba de 95.3579% y con relación a la matriz de confusión se observa lo siguiente:
En cuanto a los verdaderos negativos (TN) se observa que el modelo clasifica
correctamente la clase 0 con el 99% de los datos pertenecientes a esa clase. A nivel de verdaderos positivos (TP) no se obtiene un resultado muy favorable, ya que se observa que el modelo predice correctamente el 22% de las veces cuando un árbol será predominante. En los falsos negativos se obtiene una tasa del 78% mientras que en los falsos positivos se visualiza una tasa del 0.93%.

SVM (Support Vector Machine)

Para este caso se realizó un experimento con kernel ‘rbf’, gamma en 0.01 y C en 0.01 y se obtuvo una eficiencia de prueba de 96.7737% y la siguiente matriz de confusión:
Para este caso, la matriz de confusión muestra que el 100% el modelo realizó
predicciones con el valor de 0, un resultado no muy esperado.

Iteración 2: Modelos con SMOTE

Esta iteración puede ser observada en el notebook *05 - Models with SMOTE.ipynb* en el cual inicialmente se llevó a cabo una división de datos de train y de test. Además, para esta se buscaba mejorar el desempeño de los modelos a través de nuevas técnicas como **SMOTE** la cual es una técnica estadística de sobremuestreo de minorías sintéticas para aumentar el número de casos de un conjunto de datos de forma equilibrada [1]. Además de esto también se dividieron los datos mediante **StratifiedKFold** (n_splits en 5), para que los experimentos se realicen conservando la estratificación de los datos.

Posterior a esto se realizaron experimentos con los modelos regresión logística, XGBoost, Random Forest, Gradient Boosting y Catboost haciendo uso de la función **RandomizedSearchCV** para encontrar los parámetros óptimos de cada modelo mencionado. A continuación, se observan las curvas ROC de los cinco modelos con parámetros óptimos:

Es posible evidenciar el modelo con mejor AUC es Gradient Boosting Tree, y de segundo se encuentra Random Forest con un valor de 0.7291. Además de esto se graficaron las matrices de confusión a través del parámetro 'normalize' en true, para así observar los porcentajes y poder comparar con los resultados de la anterior iteración: Lo que se buscaba mejorar para esta iteración eran los verdaderos positivos (TP) ya que esta era una falencia para nosotros en la anterior iteración, por lo cual, optamos por elegir los modelos Random Forest (con los parámetros 'random_state': 42, 'max_features': 'sqrt', 'criterion': 'entropy', 'class_weight': 'balanced', 'bootstrap': False)
en el que se observa que el modelo predice correctamente el 47.5% de las veces cuando un determinado árbol predominará, además del modelo Gradient Boosting (con los parámetros 'random_state': 42, 'n_estimators': 100, 'max_features': 'auto') con elcual se evidencia que el modelo predice correctamente el 57.5% de las veces cuando un determinado árbol predominará.
Con esta elección, se emplean los datos de testing separados inicialmente para validar estos dos modelos, obteniéndose así las matrices de confusión.

Por último, se hace un reporte de las métricas de ambos:

Metricas de Random Forest
Metricas de Gradient Boosting

Ambos modelos tuvieron resultados sumamente similares, sin embargo por una mínima diferencia, el modelo Random Forest con los parámetros 'random_state': 42, 'max_features': 'sqrt', 'criterion': 'entropy', 'class_weight': 'balanced', 'bootstrap': False es el mejor modelo.

Retos y consideraciones de despliegue

Los retos que se tenían al principio del proyecto en cuanto al rendimiento del modelo eran:

- Tener una tasa de acierto del 85%.
- Valor de sensibilidad mayor al 80%.
- Valor de especificidad mayor al 90%.

La tasa de acierto se observa mayor al 85% debido a que los modelos están clasificando de forma desbalanceada siendo la clase mayoritaria la que nos indica que se tiene un rendimiento mucho mayor al 85%, por lo tanto, no es una métrica que tenga mucha influencia a la hora de desplegar el modelo. Los valores de sensibilidad de todos los modelos son inferiores al 80% por lo tanto no se cumple con el requerimiento asociado a esta métrica que nos indica la fracción de los verdaderos positivos clasificados. Los valores de especificidad de todos los modelos son inferiores al 90% por lo tanto no se cumple con el requerimiento asociado a esta métrica que nos indica la fracción de los verdaderos negativos. Se concluye que no se pueden desplegar los modelos debido a que no se cumplieron los requerimientos mínimos para que se tenga un buen rendimiento a nivel de producción.

Conclusiones

● La mejor manera de obtener un buen modelo de machine learning es aplicando diferentes algoritmos predictivos y combinación de hiperparámetros, en este caso de clasificación binaria sobre el mismo conjunto de datos, debido a que los algoritmos más comunes como regresión logística no siempre ofrecen el mejor resultado. Análogamente es recomendable evaluar el desempeño de cada modelo con diversas métricas de validación y no tomar decisiones solo con la exactitud (accuracy), sino también analizar otras métricas como el AUC, curva ROC, precisión y matriz de confusión.

● A través de técnicas de reducción de dimensionalidad como PCA es posible obtener modelos de machine learning más simplificados incluso con un mejor rendimiento tanto en cuanto a resultados como en tiempo de ejecución.

● El data cleaning es un proceso que demanda gran parte del tiempo en un proyecto de machine learning y es un proceso crucial para poder obtener modelos con buenos resultados.

● El modelo empleado de redes neuronales utilizado no se adecúa bien a los datos y por lo tanto no es útil para el problema en cuestión, esto se puede deber a que el modelo probablemente es demasiado complejo para los datos utilizados.

● En datos que se encuentran desbalanceados, como lo es el caso del dataset empleado, es sumamente importante utilizar técnicas de sobremuestreo como SMOTE para obtener mejores resultados. Esto se evidenció en el desarrollo del proyecto debido a que a nivel de verdaderos positivos (TP) sin SMOTE el modelo Random Forest predijo correctamente el 15% de las veces cuando un determinado árbol predominará, y con la técnica SMOTE se mejoró considerablemente los verdaderos positivos (TP) con el modelo CatBoost prediciendo correctamente el 54.54% de las veces cuando un determinado árbol si predominará.

● La realización de este estudio permitió enriquecer los conocimientos sobre gran parte de lo relacionado al machine learning, lo cual genera cierta experticia afrontar proyectos de la industria.

Referencias

[1] likebupt, «SMOTE - Azure Machine Learning». https://learn.microsoft.com/es-es/azure/machine-learning/component-reference/smote

[2] «Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE». https://www.cienciadedatos.net/documentos/35_principal_component_analysis