Tarea Individual II

Ciencia de Datos con R

Introducción

El objetivo de esta tarea es aplicar técnicas de **aprendizaje supervisado** utilizando `tidymodels`. Se plantean dos problemas: uno de **regresión** y otro de **clasificación**, utilizando modelos de **árbol de decisión** y **random forest**. Además, se debe interpretar y comunicar los resultados obtenidos.

Parte 1 – Regresión

Dataset: `ames` (paquete `{modeldata}`)
Variable objetivo: `Sale_Price` (precio de venta de la vivienda)

1.1 Preparación de datos

1.1.1. Cargar el dataset `ames` y convertirlo en tibble. Guardar como `ames_data`.

1.1.2. Utilizar `Sale_Price` como variable respuesta. Guarda la formula como `formula_ames`.

1.1.3. Dividir los datos en entrenamiento (80%) y test (20%). Guarda los conjuntos como `ames_train` y `ames_test`.

1.2 Entrenamiento del modelo

1.2.1. Definir un modelo de árbol de decisión. Guardar como `tree_ames`.

1.2.2. Entrenar el modelo utilizando `fit()`. Guardar el modelo entrenado como `fit_tree_ames`.

1.3 Evaluación del modelo

1.3.1. Predecir sobre el conjunto de test. Guardar las predicciones como `predictions_ames`.

1.3.2. Calcular **RMSE** y $\mathbf{R^2}$ tanto para train y test. En el caso de `ames_train` guardar como `metrics_train_ames` y en el caso de `ames_test` como `metrics_test_ames`.

1.3.3. Graficar valores reales vs. estimados (`Sale_Price`) con línea de identidad, utilizando los datos de entrenamiento. Mostrar el gráfico y guardarlo como `plot_tree_ames`.

1.3.4. Interpretar las métricas y el gráfico. (En este caso, no es necesario un chunk de código, pero sí una breve explicación en el texto).

1.3.5. Evaluar si hay sobreajuste o subajuste. Esta respuesta es libre y puede realizar el códogo que considere necesario para evaluar el modelo.

1.4 Interpretación del árbol

1.4.1. Visualizar el árbol. Guardar el gráfico como `plot_tree_ames_final`.

1.4.2. Interpretar brevemente las decisiones del árbol.

Parte 2 – Clasificación

Dataset: `attrition` (paquete `{modeldata}`)
Variable objetivo: `Attrition` (abandono laboral: "Yes" o "No")

2.1 Preparación de datos

2.1.1. Cargar el dataset `attrition` y convertirlo en tibble. Guardar como `attrition_data`.

2.1.2. Asegurarse de que `Attrition` sea un factor. Sobreescribir la variable si es necesario.

2.1.3. Dividir los datos en entrenamiento (80%) y test (20%) manteniendo la misma proporción de clases en ambos datasets. Guarda los conjuntos como `attrition_train` y `attrition_test`.

2.2 Entrenamiento del modelo

2.2.1. Definir un modelo de random forest. Llevar a cabo la definición utilizando `rand_forest()` y `set_engine()`. Guardar como `rf_attrition`.

2.2.2. Entrenar el modelo. Llevar a cabo el entrenamiento utilizando `fit()`. Utilizar la fórmula `Attrition ~ .` para incluir todas las variables predictoras. Guardar el modelo entrenado como `fit_rf_attrition`.

2.3 Evaluación del modelo

2.3.1. Predecir sobre el conjunto de test. Guardar las predicciones como `predictions_attrition`.

2.3.2. Calcular **accuracy**, **precision**, **recall** y **F1**. Guardar las métricas de entrenamiento como `metrics_train_attrition` y las de test como `metrics_test_attrition`.

2.3.3. Visualizar matriz de confusión con `autoplot(type = "heatmap")`. Imprimir el gráfico y guardarlo como `plot_confusion_attrition`.

2.3.4. Visualizar variables importantes con `vip()`. Imprimir el gráfico y guardarlo como `plot_vip_attrition`.

2.3.5. Interpretar las métricas y los errores. (En este caso, no es necesario un chunk de código, pero sí una breve explicación en el texto).

2.3.6. Evaluar si hay sobreajuste o subajuste. Esta respuesta es libre y puede realizar el código que considere necesario para evaluar el modelo.

Entrega

- **La fecha limite de entrega es el 20 de junio de 2025.**
- Las respuestas deben estar en este mismo archivo `.qmd`, el contenido deber ser completamente reproducible, es decir, cada `chunk` debe de funcionar sin errores para poder replicar los resultados.

- No se aceptan archivos `.Rmd` o `.R` para la entrega. Solamente subir al repositorio el archivo `.qmd` con las respuestas.
- Cada respuesta del ejercicio debe estar en el chunk correspondiente, no borrar la etiqueta del chunk `#| label: ejercicio_XX`.
- Puede realizar pasos intermedios los que sean necesarios dentro del chunk pero debe de respetar el nombre del objeto final en el caso que se indique.
- Los gráficos deben ser guardados en objetos y luego impresos en el caso que se indique que lo almacenen en un objeto. En el caso que no se indique, pueden ser impresos directamente.
- Para comenzar la tarea deben de ir al siguiente link: GitHub Classroom. Una vez allí les va a pedir que indiquen su cuenta de GitHub y luego les va a crear un repositorio en su cuenta. Una vez creado el repositorio, deben de clonar el repositorio en su computadora y abrirlo con RStudio