Informe del Modelo: XGBoost

Descripción del Modelo

El modelo XGBoost (eXtreme Gradient Boosting) se implementó como una evolución de los modelos anteriores de Random Forest básico y ajustado con hiperparámetros. Este modelo fue seleccionado por su capacidad para manejar datos de alta dimensionalidad y su habilidad para ajustar relaciones complejas entre características, lo que es crucial en el análisis de datos provenientes de sensores industriales.

Conjunto de Datos

- **Dataset:** SECOM, con dimensiones iniciales de 1567 filas y 590 columnas.
- **Procesamiento previo:**
  - Limpieza de datos eliminando columnas con más del 50% de valores faltantes.
  - Imputación de valores faltantes mediante la mediana.
  - Eliminación de columnas con varianza cero, resultando en un total de 562 características.
  - Balanceo de clases para igualar las proporciones de "Pass" y "Fail".
- **Conjuntos de entrenamiento y prueba:** 80% para entrenamiento y 20% para prueba.

Parámetros del Modelo

1. **Función objetivo:** `binary:logistic` (clasificación binaria).
2. **Métrica de evaluación:** `AUC` (Área Bajo la Curva).
3. **Hiperparámetros principales:**
   - **eta (tasa de aprendizaje):** 0.1
   - **max_depth (profundidad máxima):** 6
   - **subsample (submuestreo de filas):** 0.8
   - **colsample_bytree (submuestreo de columnas):** 0.8
   - **nrounds (iteraciones):** 100

Pasos Detallados

1. **Cargar librerías y dataset**

```
install.packages("xgboost")
library(xgboost)
library(caret)
library(pROC)

data <- read.csv("C:/SECOM_Analysis/data/cleaned_secom.csv")
```

Propósito:

- Cargar las herramientas necesarias (xgboost para el modelo, caret para la división de datos, y pROC para la evaluación) y el conjunto de datos.

Por qué:

- Estas librerías son fundamentales para el entrenamiento, validación y análisis del modelo.

---

2. **Inspección y preprocesamiento inicial**

```
cat("Dimensiones del conjunto de datos:", dim(data), "\n")
head(data)
data$Pass.Fail <- as.factor(data$Pass.Fail)
```

Propósito:

- Verificar el tamaño del dataset y convertir Pass.Fail a un factor para que sea interpretable como una variable categórica.

Por qué:

- Garantizar que el modelo entienda la variable objetivo como una clasificación binaria.

---

3. **División de datos: entrenamiento y prueba**

```
set.seed(123)
train_index <- createDataPartition(data$Pass.Fail, p = 0.8, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
```

Propósito:

- Dividir los datos en 80% para entrenamiento y 20% para prueba.

Por qué:

- Separar los datos permite evaluar el modelo en datos no vistos.

Ajustes posibles:

- Cambia `p = 0.8` a otro porcentaje según el tamaño de tu dataset (e.g., `0.7` para 70% entrenamiento).

---

4. **Preparación de datos para XGBoost**

```
train_matrix <- xgb.DMatrix(
  data = as.matrix(train_data[, -c(1, ncol(train_data))]),
  label = as.numeric(train_data$Pass.Fail) - 1
)
test_matrix <- xgb.DMatrix(
  data = as.matrix(test_data[, -c(1, ncol(train_data))]),
  label = as.numeric(test_data$Pass.Fail) - 1
)
```

Propósito:

- Crear objetos DMatrix, optimizados para el entrenamiento y evaluación de XGBoost.

Por qué:

- XGBoost necesita entradas numéricas y matrices optimizadas para maximizar el rendimiento.

---

5. **Definir los parámetros del modelo**

```
params <- list(
  objective = "binary:logistic",
  eval_metric = "auc",
  eta = 0.1,
  max_depth = 6,
  subsample = 0.8,
  colsample_bytree = 0.8
)
```

Propósito:

- Configurar los hiperparámetros del modelo, como la tasa de aprendizaje (eta) y la profundidad de los árboles (max_depth).

Por qué:

- Estos parámetros controlan el rendimiento, la velocidad y la capacidad de generalización del modelo.

Ajustes posibles:

- Incrementar eta para entrenar más rápido (e.g., 0.3) o reducir max_depth para evitar sobreajuste.

---

6. Entrenar el modelo

```
xgb_model <- xgb.train(
  params = params,
  data = train_matrix,
  nrounds = 100,
  watchlist = list(train = train_matrix, test = test_matrix),
  verbose = 1
)
```

Propósito:

- Entrenar el modelo con 100 iteraciones, monitoreando el desempeño en los datos de entrenamiento y prueba.

Por qué:

- La evaluación en ambos conjuntos evita el sobreajuste.

Ajustes posibles:

- Cambiar nrounds a un valor mayor (e.g., 500) si el modelo no converge en 100 rondas.

---

7. **Evaluación del modelo**

```
predictions <- predict(xgb_model, test_matrix)
predicted_labels <- ifelse(predictions > 0.5, "Pass", "Fail")
confusion_matrix <- table(test_data$Pass.Fail, predicted_labels)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
```

Propósito:

- Generar predicciones, calcular la matriz de confusión y medir la precisión.

Por qué:

- Evaluar cómo el modelo clasifica los datos de prueba.

Ajustes posibles:

- Cambiar el umbral 0.5 para ajustar la sensibilidad o especificidad.

---

8. **Curva ROC y AUC**

```
roc_curve <- roc(as.numeric(test_data$Pass.Fail) - 1, predictions)
plot(roc_curve, main = "Curva ROC", col = "blue")
cat("Área bajo la curva (AUC):", auc(roc_curve), "\n")
```

Propósito:

- Graficar la curva ROC y calcular el AUC como métrica de desempeño.

Por qué:

- AUC es ideal para medir la capacidad del modelo de distinguir entre clases.

---

9. **Importancia de características**

```
importance <- xgb.importance(feature_names = colnames(train_data[, -c(1,
ncol(train_data))]), model = xgb_model)
xgb.plot.importance(importance_matrix = importance, top_n = 10, main = "Top 10
Características Importantes")
```

Propósito:

- Identificar y graficar las características que más influyen en el modelo.

Por qué:

- Entender qué variables son clave para las predicciones.

---

10. **Guardar el modelo**

```
xgb.save(xgb_model, "C:/SECOM_Analysis/models/xgboost_model.model")
```

Propósito:

- Guardar el modelo entrenado para reutilizarlo.

Por qué:

- Evitar entrenar el modelo nuevamente para futuras predicciones.

Resultados del Modelo

- **Matriz de Confusión:**

```
     Fail Pass
Fail   292    20
Pass     0     0
```

- **Métricas obtenidas:**
  - **Precisión (Accuracy):** 94%.
  - **Área Bajo la Curva (AUC):** 0.7519.
- **Importancia de características:**
  - Las características más importantes incluyen:
    - **X59:** Mayor ganancia en la predicción.
    - **X0, X205, X573:** También altamente relevantes.
  - La mayoría de las variables muestran una ganancia insignificante, lo que indica redundancia o ruido.

Pros del Modelo

1. **Desempeño General:**
   - El modelo mostró una mejora significativa en la precisión (94%) en comparación con los modelos Random Forest anteriores.
   - Alta sensibilidad para detectar instancias de "Fail", lo que es crucial en el contexto de producción.
2. **Robustez:**
   - Manejo automático de valores faltantes durante el entrenamiento.
   - Excelente capacidad para ajustar relaciones no lineales.
3. **Interpretabilidad:**
   - Permite identificar las características más importantes para la clasificación, ayudando a priorizar las variables críticas en el proceso de fabricación.

Contras del Modelo

1. **Sobreajuste Potencial:**
   - La precisión extremadamente alta en el conjunto de entrenamiento y las discrepancias en AUC entre entrenamiento y prueba (0.999 en entrenamiento versus 0.751 en prueba) indican un posible sobreajuste.
2. **Dimensionalidad:**
   - A pesar de la eliminación de columnas con varianza cero, la alta dimensionalidad del dataset puede estar agregando ruido y reduciendo la generalización.
3. **Rendimiento en Prueba:**
   - Aunque el AUC de prueba es aceptable, no es ideal, lo que sugiere que el modelo no está maximizando su capacidad discriminativa.

Recomendaciones

1. **Selección de Características:**
   - Reducir aún más la dimensionalidad mediante técnicas como **Análisis de Componentes Principales (PCA)** o **Selección de Características Basada en Importancia**.
2. **Optimización de Hiperparámetros:**
   - Explorar un rango más amplio de valores para `eta`, `max_depth`, y `subsample` utilizando técnicas como búsqueda en cuadrícula o optimización bayesiana.
3. **Regularización:**

    - Ajustar parámetros como `lambda` y `alpha` para regularizar el modelo y reducir el riesgo de sobreajuste.
4. **Validación Adicional:**
    - Utilizar validación cruzada más extensa para evaluar la estabilidad del modelo y mitigar el impacto del sobreajuste.
5. **Evaluación en Producción:**
    - Implementar un conjunto de datos no visto previamente para evaluar el modelo en un entorno más realista y comprobar su desempeño.

Conclusión

El modelo XGBoost representa una mejora significativa respecto a los modelos previos, con una alta precisión y capacidad de manejo de datos complejos. Sin embargo, los signos de sobreajuste y la necesidad de mejorar el AUC sugieren que es necesario optimizar aún más el modelo y reducir la dimensionalidad. Con ajustes adicionales, este enfoque puede convertirse en una herramienta poderosa para mejorar la calidad del proceso de fabricación.

Conclusión Práctica

Si confirmas que las 10 características están asociadas directamente con etapas o controles del proceso, puedes concluir que estas son las **etapas críticas que requieren mayor atención**. La interpretación sería:

- **Propuesta**: Mejorar la supervisión y el control de estas etapas podría reducir significativamente la probabilidad de un resultado **"Fail"**.
- **Acción sugerida**: Implementar monitoreos más estrictos, controles de calidad o ajustes automáticos en esas variables clave.

Por ejemplo:

- Si una característica importante es la **"temperatura de calentamiento"**, un monitoreo más estricto podría evitar fluctuaciones que lleven a un fallo.