MÓDULO II

Aprendizaje Supervisado

Curso:

Machine Learning Supervisado

💰 Sesión 03: Del Accuracy al Profit

Optimización Bayesiana, Calibración y Business Analytics

Dejando de jugar a las adivinanzas y empezando a hacer dinero

📋 Agenda de Hoy

Bloque	Tema	Duración
1	🔍 El Infierno de GridSearch	15 min
2	🎯 Optimización Bayesiana con Optuna	30 min
3	⚖️ El Mito de SMOTE y Desbalance	20 min
☕	**Break**	15 min
4	🎯 Calibración de Probabilidades	25 min
5	💸 Business Analytics: Profit Curves	30 min
6	🧪 Hands-On: Del Modelo al Dinero	25 min

🎯 Objetivos de Aprendizaje

Al finalizar esta sesión podrás:

1. **Optimizar Inteligentemente:** Usar Optuna en lugar de GridSearch ciego
2. **Entender el Desbalance:** Por qué SMOTE casi nunca es la respuesta
3. **Calibrar Probabilidades:** Asegurar que un 80% sea realmente 80%
4. **Monetizar Modelos:** Convertir predicciones en decisiones rentables
5. **Threshold Tuning:** Encontrar el umbral óptimo para tu negocio

🔍 BLOQUE 1

El Infierno de GridSearch

¿Por qué la fuerza bruta no escala?

🔍 El Problema de la Búsqueda Manual

La Realidad

- `max_depth`: 10 valores
- `n_estimators`: 10 valores
- `learning_rate`: 10 valores
- `min_samples_leaf`: 10 valores

= 10,000 combinaciones 💥

```
# GridSearchCV ingenuo
param_grid = {
    'max_depth': range(1, 11),
    'n_estimators': range(50, 550, 50),
    'learning_rate': [0.01, 0.1, 0.2, ...],
    'min_samples_leaf': range(1, 11)
}
# ⏱️ Tiempo: ~8 horas en laptop
```

Métodos de Búsqueda

Método	Estrategia	Eficiencia
Manual	Intuición	❌ Sesgado
GridSearch	Todas las combinaciones	❌ Exponencial
RandomSearch	Pruebas aleatorias	⚠️ Mejor, pero ciego
Bayesiana	Aprende del pasado	✅ Inteligente

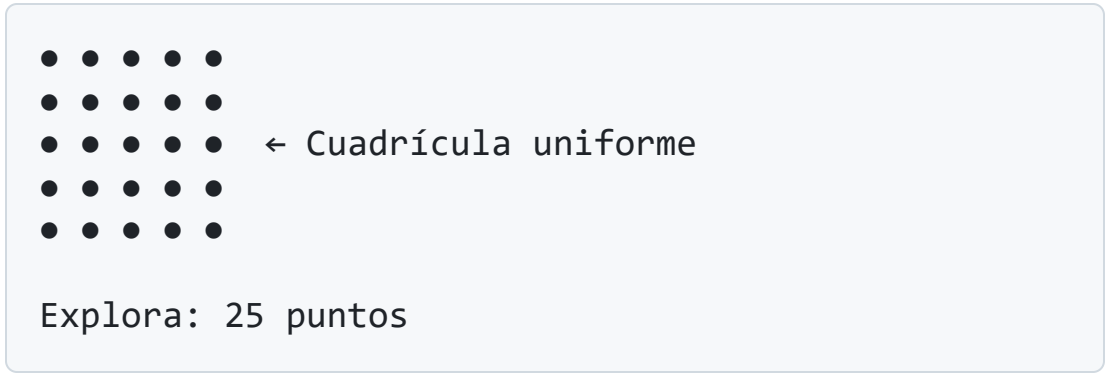
La Maldición de la Dimensionalidad

$$\text{Combinaciones} = \prod_{i=1}^{n} k_i$$

Cada hiperparámetro nuevo multiplica el espacio

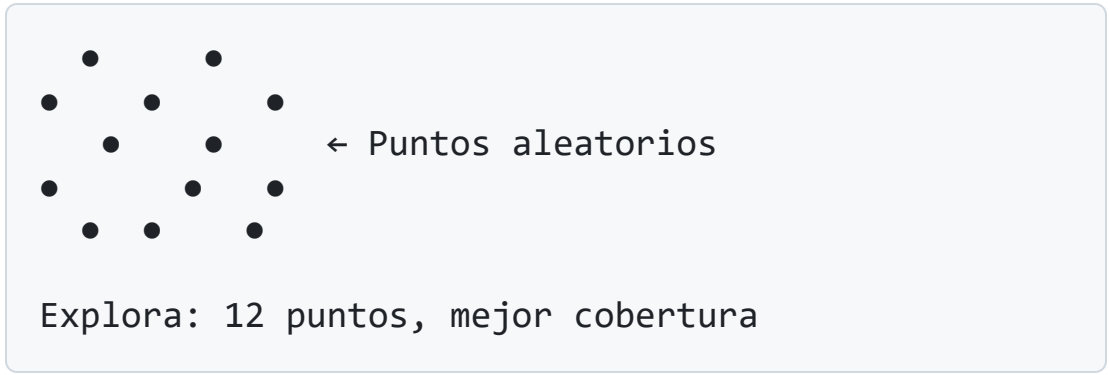
📊 GridSearch vs RandomSearch

GridSearch



Problema: Si el óptimo está entre puntos, lo pierde.

RandomSearch



Ventaja: Más eficiente en alta dimensión (Bergstra & Bengio, 2012)

> 💡 **Paper:** *Random Search for Hyper-Parameter Optimization* demostró que RandomSearch supera a GridSearch con menos evaluaciones.

🎯 BLOQUE 2

Optimización Bayesiana con Optuna

"Aprende de tus errores (computacionales)"

🎯 ¿Qué es Optuna?

La Idea Central

1. **Prueba** un conjunto de hiperparámetros
2. **Observa** el resultado (AUC, Accuracy, etc.)
3. **Aprende** qué regiones son prometedoras
4. **Sugiere** el siguiente punto inteligentemente

Algoritmos Internos

- **TPE (Tree-structured Parzen Estimator):** Default
- **CMA-ES:** Para espacios continuos
- **GP:** Gaussian Process

Diagrama Conceptual

```
Trial 1: max_depth=3 → AUC=0.75
Trial 2: max_depth=10 → AUC=0.82
Trial 3: max_depth=7 → AUC=0.88 ✓

"El modelo aprende que depth~7
 es una región prometedora"

Trial 4: max_depth=8 → AUC=0.89 ✓
Trial 5: max_depth=6 → AUC=0.87
...

Converge al óptimo en ~50 trials
(GridSearch necesitaría 10,000)
```

🎯 Optuna en Código

```python
import optuna

def objective(trial):
    params = {
        'max_depth': trial.suggest_int('max_depth', 2, 15),
        'n_estimators': trial.suggest_int('n_estimators', 50, 500),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, log=True),
        'num_leaves': trial.suggest_int('num_leaves', 10, 100),
        'min_child_samples': trial.suggest_int('min_child_samples', 5, 100),
    }

    model = lgb.LGBMClassifier(**params, random_state=42)
    score = cross_val_score(model, X_train, y_train, cv=5, scoring='roc_auc')
    return score.mean()

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100, show_progress_bar=True)

print(f"Mejor AUC: {study.best_value:.4f}")
print(f"Mejores hiperparámetros: {study.best_params}")
```

🎯 Tipos de `suggest` en Optuna

Método	Uso	Ejemplo
`suggest_int`	Enteros	`max_depth`, `n_estimators`
`suggest_float`	Decimales	`learning_rate`
`suggest_float(log=True)`	Escala logarítmica	Tasas de aprendizaje
`suggest_categorical`	Opciones discretas	`['gini', 'entropy']`

⚠️ Real-World Warning: Escala Logarítmica

```
# ❌ Mal: learning_rate uniformemente entre 0.01 y 0.3
trial.suggest_float('lr', 0.01, 0.3)  # Favorece valores altos

# ✅ Bien: escala logarítmica
trial.suggest_float('lr', 0.01, 0.3, log=True)  # Explora bien 0.01-0.1
```

📊 Visualización de Optuna

```
# Historia de optimización
optuna.visualization.plot_optimization_history(study)

# Importancia de hiperparámetros
optuna.visualization.plot_param_importances(study)

# Coordenadas paralelas
optuna.visualization.plot_parallel_coordinate(study)
```

Gráficos que Genera

Gráfico	¿Qué muestra?
Optimization History	Convergencia del AUC vs trials
Param Importances	¿Qué hiperparámetro impacta más?
Parallel Coordinate	Relaciones entre hiperparámetros
Contour Plot	Superficie de respuesta 2D

⚖️ BLOQUE 3

El Mito de SMOTE y el Desbalance

Por qué crear datos sintéticos es casi siempre una mala idea

⚖️ El Problema del Desbalance

Escenario Típico

- 95% clientes buenos ✅
- 5% clientes morosos ❌

El Modelo Perezoso

```
# Si siempre predice "No moroso"
accuracy = 0.95  # 🎉 Excelente!
recall_morosos = 0.00  # 💀 Inútil
```

Soluciones Comunes

1. **SMOTE:** Crear datos sintéticos
2. **Undersampling:** Eliminar mayoritarios
3. **Class Weights:** Penalizar errores
4. **Threshold Tuning:** Ajustar umbral

La Tentación de SMOTE

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(
    X_train, y_train
)
# Ahora tiene 50%-50%
# ¿Problema resuelto? ❌
```

Lo que Realmente Hace SMOTE

- Interpola puntos entre vecinos minoritarios
- **Crea ejemplos que nunca existieron**
- El modelo aprende de "fantasmas"

⚠️ Los Problemas de SMOTE

1. Rompe la Calibración de Probabilidades

```
Distribución Real:      5% morosos
Distribución con SMOTE: 50% morosos

El modelo aprende P(moroso) ≈ 0.5
Pero la realidad es P(moroso) ≈ 0.05

Resultado: Probabilidades infladas e inútiles
```

2. Introduce Ruido en Fronteras

- SMOTE crea puntos en **zonas de transición**
- Donde precisamente hay más incertidumbre
- El modelo puede aprender patrones **falsos**

3. Hace el Entrenamiento más Lento

- De 100,000 ejemplos a 200,000+
- Más datos = más tiempo de entrenamiento

✅ La Solución Pro: Class Weights

En LightGBM/XGBoost

```
# Calcula el peso automáticamente
neg_count = (y_train == 0).sum()
pos_count = (y_train == 1).sum()
scale_pos_weight = neg_count / pos_count

lgbm = lgb.LGBMClassifier(
    scale_pos_weight=scale_pos_weight
)
```

Lo que Hace

- FN cuesta `scale_pos_weight` veces más que FP
- El modelo se esfuerza más en la clase minoritaria
- **Sin crear datos falsos**

En Scikit-Learn

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    class_weight='balanced'  # Automático
)

# O manualmente
rf = RandomForestClassifier(
    class_weight={0: 1, 1: 19}  # 5% vs 95%
)
```

Ventajas

- ✅ Mantiene calibración intacta
- ✅ No aumenta tiempo de entrenamiento
- ✅ Matemáticamente correcto
- ✅ Más interpretable

📊 Comparativa: SMOTE vs Class Weights

Aspecto	SMOTE	Class Weights
Calibración	❌ Destruida	✅ Preservada
Tiempo de entrenamiento	↑ Aumenta	= Igual
Datos sintéticos	Sí (riesgo)	No
Complejidad	Alta	Baja
Cuándo usar	Muy raro	Por defecto

> 💡 **Pro-Tip:** Si necesitas usar SMOTE, **recalibra** el modelo después con `CalibratedClassifierCV`.

 BREAK

15 minutos

🎯 BLOQUE 4

Calibración de Probabilidades

¿Tu 80% es realmente 80%?

🎯 ¿Qué es Calibración?

Definición

Un modelo está **calibrado** si:

- De 100 predicciones con P=0.8
- Aproximadamente **80 son positivas**

Ejemplo Real

Predicción	Calibrado	No Calibrado
P=0.10	~10% positivos	~5% positivos
P=0.50	~50% positivos	~70% positivos
P=0.90	~90% positivos	~95% positivos

¿Por qué Importa?

> "El paciente tiene 80% de probabilidad de cáncer"

Si el modelo está mal calibrado:

- Podría ser realmente 40% → alarma innecesaria
- Podría ser realmente 95% → subestimamos el riesgo

Sectores Críticos

- 🏥 **Medicina:** Diagnósticos
- 🏦 **Banca:** Riesgo crediticio
- 🚗 **Seguros:** Pricing de primas
- ⚖️ **Legal:** Predicción de reincidencia

📈 El Diagrama de Calibración

Reliability Diagram (Curva de Calibración)

```
from sklearn.calibration import calibration_curve

prob_true, prob_pred = calibration_curve(
    y_test, y_pred_proba, n_bins=10, strategy='uniform'
)

plt.plot([0, 1], [0, 1], 'k--', label='Perfectamente calibrado')
plt.plot(prob_pred, prob_true, 's-', label='Modelo')
plt.xlabel('Probabilidad Predicha')
plt.ylabel('Fracción de Positivos Real')
plt.legend()
```

Interpretación

Curva	Significado
Sobre diagonal	Subestima (conservador)
Bajo diagonal	Sobreestima (sobre-confiado)
Diagonal	Perfectamente calibrado ✅

🛠️ Corrigiendo la Calibración

Método 1: Platt Scaling

```
from sklearn.calibration import CalibratedClassifierCV

calibrated = CalibratedClassifierCV(
    model,
    method='sigmoid',  # Platt
    cv=5
)
calibrated.fit(X_train, y_train)
```

Ajusta una **regresión logística** sobre los scores.

Método 2: Isotonic Regression

```
calibrated = CalibratedClassifierCV(
    model,
    method='isotonic',
    cv=5
)
```

Ajuste no-paramétrico más flexible.

¿Cuál Usar?

Método	Cuándo
Sigmoid	Pocos datos (<1000)
Isotonic	Más datos (>1000)

⚠️ Modelos y su Calibración Típica

Modelo	Calibración por Defecto
Logistic Regression	✅ Bien calibrado
Random Forest	⚠️ Tiende a subestimar extremos
Gradient Boosting	⚠️ Varía según hiperparámetros
SVM (Platt)	❌ Requiere calibración
Naive Bayes	❌ Muy mal calibrado
Neural Networks	⚠️ Tienden a sobreconfianza

> 💡 **Pro-Tip:** Siempre verifica la calibración antes de usar probabilidades en producción.

💸 BLOQUE 5

Business Analytics: Profit Curves

Donde el Machine Learning conoce al Balance General

💸 El Accuracy No Paga las Cuentas

El Problema

```
Modelo A: Accuracy 95%
Modelo B: Accuracy 90%

¿Cuál es mejor? 🤔
```

Depende del Costo de los Errores

Error	Crédito	Fraude
FP	Rechazar buen cliente: -$500	Bloquear tarjeta legítima: -$50
FN	Aprobar moroso: -$10,000	No detectar fraude: -$5,000

La Realidad Financiera

$$\text{Costo Total} = C_{FP} \cdot FP + C_{FN} \cdot FN$$

Ejemplo

- Modelo A (95%): 50 FP, 50 FN
- Modelo B (90%): 100 FP, 0 FN

Con $C_{FP} = \$500$ y $C_{FN} = \$10,000$:

- **Modelo A:** $50 \times 500 + 50 \times 10000 = \$525,000$
- **Modelo B:** $100 \times 500 + 0 \times 10000 = \$50,000$

Modelo B gana aunque tiene peor accuracy!

📊 La Matriz de Costos

Definiendo el Valor de Negocio

	Pred: No Moroso (0)	Pred: Moroso (1)
Real: No Moroso (0)	TN: $0	FP: -$500 (costo de oportunidad)
Real: Moroso (1)	FN: -$10,000 (pérdida capital)	TP: +$200 (interés cobrado)

En Código

```
cost_matrix = {
    'TP': 200,    # Beneficio de detectar moroso y no prestar
    'TN': 0,      # Cliente bueno, préstamo pagado
    'FP': -500,   # Rechazamos buen cliente
    'FN': -10000  # Aprobamos moroso, perdemos capital
}
```

📈 Profit Curves: Encontrando el Umbral Óptimo

El Umbral por Defecto (0.5)

```
# Predicción binaria estándar
y_pred = (y_proba >= 0.5).astype(int)
```

Pero... ¿Por qué 0.5?

- Es arbitrario
- Asume costos simétricos
- Ignora la realidad del negocio

Threshold Tuning

Barrer umbrales de 0 a 1 y calcular el **profit total** para cada uno.

Código de Profit Curve

```
def calculate_profit(y_true, y_proba, threshold, costs):
    y_pred = (y_proba >= threshold).astype(int)

    TP = ((y_pred == 1) & (y_true == 1)).sum()
    TN = ((y_pred == 0) & (y_true == 0)).sum()
    FP = ((y_pred == 1) & (y_true == 0)).sum()
    FN = ((y_pred == 0) & (y_true == 1)).sum()

    profit = (TP * costs['TP'] +
              TN * costs['TN'] +
              FP * costs['FP'] +
              FN * costs['FN'])
    return profit

# Barrer umbrales
thresholds = np.linspace(0, 1, 100)
profits = [calculate_profit(y_test, y_proba, t, costs)
           for t in thresholds]
```

📊 Visualizando la Profit Curve

```
import plotly.express as px

# Crear DataFrame
df_profits = pd.DataFrame({
    'Threshold': thresholds,
    'Profit': profits
})

# Encontrar óptimo
best_idx = np.argmax(profits)
best_threshold = thresholds[best_idx]
best_profit = profits[best_idx]

fig = px.line(df_profits, x='Threshold', y='Profit',
              title=f'Profit Curve - Umbral Óptimo: {best_threshold:.2f}')
fig.add_vline(x=best_threshold, line_dash="dash", line_color="red")
fig.show()
```

Resultado Típico

- **Umbral 0.5**: Profit = $50,000
- **Umbral óptimo (0.25)**: Profit = $120,000 🎉
- **Ganancia**: +$70,000 solo por cambiar el umbral

🎯 Threshold Tuning en Producción

Consideraciones Prácticas

Aspecto	Recomendación
Datos de calibración	Usar validation set, NO test
Estabilidad	Probar con bootstrap
Monitoreo	Re-evaluar umbral periódicamente
Restricciones	Considerar límites regulatorios

⚠️ Real-World Warning: Overfitting del Umbral

```
# ❌ MAL: Optimizar umbral en test set
best_threshold = optimize_threshold(y_test, y_proba_test)

# ✅ BIEN: Optimizar en validation, evaluar en test
best_threshold = optimize_threshold(y_val, y_proba_val)
final_profit = calculate_profit(y_test, y_proba_test, best_threshold)
```

💡 Framework de Decisión de Negocio

Flujo Completo

```
1. Entrenar Modelo
       ↓
2. Calibrar Probabilidades
       ↓
3. Definir Matriz de Costos
       ↓
4. Encontrar Umbral Óptimo (en validation)
       ↓
5. Evaluar Profit Final (en test)
       ↓
6. Monitorear en Producción
```

> 💡 **Regla de Oro:** El modelo más "preciso" no siempre es el más rentable. Optimiza para **lo que importa al negocio**.

🧪 BLOQUE 6

Hands-On: Del Modelo al Dinero

`01_Optimization_and_Money.ipynb`

🧪 Lo que Construiremos

Pipeline Completo

1. **Preprocesamiento Robusto**
    - Pipeline con ColumnTransformer
    - TargetEncoder para categóricas
2. **Optimización con Optuna**
    - 50 trials bayesianos
    - Visualización de resultados
3. **Manejo de Desbalance**
    - `scale_pos_weight` vs SMOTE (demostración)
4. **Calibración**
    - Verificar y corregir probabilidades
5. **Profit Curve**

🎯 Key Takeaways

1. **GridSearch es prehistórico:** Usa Optuna para búsqueda inteligente de hiperparámetros
2. **SMOTE casi nunca es la respuesta:** `class_weight` es más limpio y preserva calibración
3. **Verifica la calibración:** Un 80% debe ser realmente 80% antes de tomar decisiones
4. **El accuracy no paga cuentas:** Optimiza para el **profit**, no para métricas abstractas
5. **El umbral 0.5 es arbitrario:** El umbral óptimo depende de tu matriz de costos
6. **Siempre piensa en producción:** Calibración y umbrales se validan en datos separados

📚 Referencias y Lecturas

Papers Fundamentales

- Bergstra, J., & Bengio, Y. (2012). *Random Search for Hyper-Parameter Optimization*. JMLR.
- Akiba, T. et al. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. KDD.
- Niculescu-Mizil, A., & Caruana, R. (2005). *Predicting Good Probabilities with Supervised Learning*. ICML.

Libros

- Provost, F., & Fawcett, T. (2013). *Data Science for Business*. O'Reilly.
- Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly.

Documentación

- Optuna Documentation
- Scikit-Learn Calibration

🙋 ¿Preguntas?

📧 jrodriguezm216@gmail.com

🌐 jordandataexpert.com

💻 github.com/JordanKingPeru

💻 ¡Vamos al Código!

Notebook de Hoy:

`01_Optimization_and_Money.ipynb`

Abrir: `03Material/03_Sesion_Optimizacion_Negocio/notebooks/`