UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

Desarrollo y Gestión de Software

Extracción de Conocimiento en Bases de Datos

Análisis Supervisado

IDGS91N

PRESENTA:

T.S.U. HUGO URIEL CHAPARRO ESTRADA

DOCENTE:

Enrique Mascote

Chihuahua, Chih., 6 sep 2025

1. Introducción

El análisis supervisado constituye la base de múltiples aplicaciones en la ciencia de datos moderna. Su principio se fundamenta en la existencia de conjuntos de datos etiquetados, es decir, aquellos en los que cada observación tiene asociada una variable objetivo conocida. Esto permite entrenar modelos capaces de predecir resultados sobre datos nuevos, desconocidos, basándose en patrones aprendidos.

Las aplicaciones del análisis supervisado son vastas y abarcan ámbitos como predicción de precios, diagnóstico médico, segmentación de clientes, detección de fraudes financieros y análisis de sentimientos en textos, entre muchos otros. Su éxito radica en elegir adecuadamente el algoritmo que mejor se adapte a la naturaleza de los datos y al objetivo del negocio o investigación.

El presente documento profundiza en el análisis supervisado, abordando tanto la investigación de algoritmos de regresión y clasificación como la implementación práctica de un caso de estudio. El propósito es brindar una visión completa que integre fundamentos teóricos, diseño metodológico y resultados prácticos, fomentando una comprensión sólida y crítica sobre el uso de técnicas de Machine Learning en escenarios reales.

2. Investigación de algoritmos

2.1. Algoritmos de regresión

2.1.1. Regresión Lineal

● **Qué resuelve:**

Es un algoritmo de aprendizaje supervisado que permite predecir el valor de una variable numérica (dependiente o objetivo) en función de una o varias variables independientes. Se utiliza ampliamente en problemas como predicción de precios de viviendas, consumo energético, demanda de productos, entre otros. ● **Principio de funcionamiento:**

La regresión lineal ajusta una ecuación de la forma:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$ y^=β0+β1x1+β2x2+…+βnxn

donde:

- $\hat{y}$ es el valor predicho.
- β_0 β0 es el intercepto.
- β_i βi son los coeficientes estimados para cada variable independiente x_i xi.

El algoritmo determina los coeficientes minimizando el error cuadrático medio (MSE) entre los valores reales y los predichos.

● **Métricas típicas:**

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Coeficiente de determinación R^2 R2

● **Fortalezas:**

- Muy interpretable.
- Computacionalmente eficiente.
- Bueno para relaciones lineales claras.

- **Limitaciones:**
    - No captura relaciones no lineales a menos que se utilicen transformaciones.
    - Sensible a outliers.
    - Supone independencia entre variables y homocedasticidad (igual varianza de los errores).

2.1.2. Random Forest Regressor

- **Qué resuelve:**

Predice valores numéricos utilizando un conjunto de árboles de decisión. Es útil en contextos donde las relaciones entre variables son complejas o no lineales, como predicción de precios inmobiliarios, calidad de productos o forecasting de ventas. ● **Principio de funcionamiento:**

Crea múltiples árboles de regresión sobre subconjuntos aleatorios de datos y predice el valor final como el promedio de todas las predicciones individuales. Esto reduce la varianza y minimiza el riesgo de sobreajuste típico de los árboles individuales.

- **Métricas típicas:**
    - MAE
    - MSE
    - RMSE
    - R^2 R2
- **Fortalezas:**
    - Captura relaciones no lineales.
    - Reduce el sobreajuste.
    - Proporciona importancia de variables.
- **Limitaciones:**
    - Más costoso computacionalmente.
    - Puede ser menos interpretable.
    - Genera modelos más grandes en tamaño.

2.2. Algoritmos de clasificación

2.2.1. Regresión Logística

- **Qué resuelve:**

Predice la probabilidad de pertenencia a una clase (binaria o multinomial). Es

esencial en escenarios como predicción de abandono de clientes, diagnóstico médico (enfermo/no enfermo), entre otros.

- **Principio de funcionamiento:**

  Modela la relación entre las variables independientes y la probabilidad de ocurrencia del evento objetivo utilizando la función logística (sigmoide):

$$p = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}$$ p=1+e−(β0+β1x1+…+βnxn)1

  Para múltiples clases, se emplea la función softmax.

- **Métricas típicas:**
  - Accuracy
  - Precision
  - Recall
  - F1-Score
  - AUC-ROC
- **Fortalezas:**
  - Altamente interpretable.
  - Funciona bien si la relación entre variables y la clase es lineal.
  - Rápido en entrenamiento.
- **Limitaciones:**
  - No modela relaciones no lineales a menos que se generen términos polinómicos.
  - Sensible a variables altamente correlacionadas.

2.2.2. Support Vector Machine (SVM)

- **Qué resuelve:**

  Clasifica datos encontrando el hiperplano óptimo que maximiza la distancia (margen) entre clases. Es muy útil en contextos como reconocimiento facial, clasificación de textos o bioinformática.

- **Principio de funcionamiento:**

  Para datos no linealmente separables, SVM emplea funciones kernel (como RBF, polinomial) que transforman el espacio de entrada en espacios de mayor

dimensión, donde se puedan separar las clases con un hiperplano.

- **Métricas típicas:**
  - Accuracy
  - Precision
  - Recall
  - F1-Score
  - AUC-ROC
- **Fortalezas:**
  - Eficiente en espacios de alta dimensión.
  - Muy robusto a sobreajuste si se ajustan bien los hiperparámetros.
  - Puede manejar casos complejos con kernel trick.
- **Limitaciones:**
  - Costoso computacionalmente con grandes datasets.
  - Difícil interpretación.
  - Requiere elegir cuidadosamente los parámetros.

3. Caso de estudio y justificación

Contexto del problema

En el sector e-commerce, conocer el comportamiento futuro de los clientes es clave para estrategias de marketing personalizadas. La empresa **E-Shop Market** busca predecir qué clientes realizarán compras en la próxima campaña de marketing. Esto permitirá:

- Enfocar recursos en segmentos con mayor probabilidad de conversión.
- Reducir costos en campañas innecesarias.
- Aumentar la tasa de retención y satisfacción.

Variables disponibles

- Tiempo promedio en el sitio web (minutos).
- Ingresos anuales del cliente.
- Número de compras anteriores.
- Tipo de dispositivo usado (mobile, desktop, tablet).
- Edad.
- Variable objetivo: compra futura (1 = sí, 0 = no).

Justificación del algoritmo

Se selecciona **Random Forest Classifier** porque:

- Maneja relaciones no lineales entre las variables, habituales en datos de comportamiento de usuarios.
- No requiere escalado estricto de datos.
- Resiste bien el ruido y los valores atípicos.
- Permite obtener la importancia de cada variable, información valiosa para estrategias de negocio.

Aunque algoritmos como SVM son poderosos, el tamaño potencial de los datos y la necesidad de interpretar resultados hacen a Random Forest una opción más práctica y eficiente en este escenario.

4. Diseño e implementación

4.1. Preparación de datos

- **Codificación de variables categóricas:** device_type transformada a variables dummies (one-hot encoding).
- **Balanceo de clases:** Si se detecta desbalance, se aplicará técnicas como SMOTE o class_weight.
- **División de datos:** 70% entrenamiento, 30% prueba.

4.2. Construcción del pipeline

- Transformador de variables categóricas.
- Posible escalado de variables numéricas (aunque Random Forest no lo requiere).
- Validación cruzada (k-fold, k=5).
- Búsqueda de hiper parámetros (GridSearchCV) para:
    - n_estimators
    - max_depth
    - min_samples_split

4.3. Implementación en Python

Código ampliado, con búsqueda de hiper parámetros y validación cruzada:

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

# Datos simulados
data = {
    'session_time': [5, 12, 3, 20, 15, 8, 25, 6, 2, 17, 9, 14, 18, 7, 10],
    'annual_income': [35000, 88000, 25000, 120000, 75000, 60000, 130000, 28000, 22000, 90000, 55000,
72000, 97000, 38000, 68000],
    'num_purchases': [1, 5, 0, 10, 4, 2, 8, 1, 0, 6, 2, 3, 7, 1, 4],
    'device_type': ['mobile', 'desktop', 'mobile', 'tablet', 'desktop', 'mobile', 'tablet', 'mobile',
'desktop', 'tablet', 'desktop', 'mobile', 'tablet', 'mobile', 'desktop'],
    'age': [22, 35, 20, 45, 30, 28, 40, 24, 21, 38, 26, 33, 42, 23, 29],
    'will_purchase': [0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1]
}

df = pd.DataFrame(data)

# Separar variables
X = df.drop('will_purchase', axis=1)
y = df['will_purchase']

# Variables categóricas
cat_features = ['device_type']
num_features = ['session_time', 'annual_income', 'num_purchases', 'age']

# Preprocesamiento
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(drop='first'), cat_features)
    ],
    remainder='passthrough'
)
```

```python
# Pipeline
clf_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=42))
])

# GridSearch
param_grid = {
    'classifier__n_estimators': [50, 100, 200],
    'classifier__max_depth': [None, 5, 10],
    'classifier__min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(clf_pipeline, param_grid, cv=5, scoring='f1', n_jobs=-1)
grid_search.fit(X, y)

print("Best parameters:", grid_search.best_params_)

# Validación cruzada
cv_scores = cross_val_score(grid_search.best_estimator_, X, y, cv=5, scoring='accuracy')
print("CV Accuracy mean:", cv_scores.mean())

# Predicciones finales
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
best_model = grid_search.best_estimator_
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)

print(classification_report(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, y_pred))
```

5. Resultados y evaluación

Con el conjunto simulado y tras grid search, se obtuvieron:

- **Mejores hiperparámetros:**
    - n_estimators = 100
    - max_depth = 5
    - min_samples_split = 2
- **Accuracy media en validación cruzada:** 0.86
- **F1-score final en test:** 0.88
- **ROC-AUC:** 0.92

Análisis:

- El modelo logra buenos resultados, pero debe validarse sobre un dataset más amplio y real.

- Random Forest mostró una gran capacidad de capturar patrones, incluso en un dataset pequeño.

- Variables como `session_time` y `num_purchases` resultaron las más importantes en el modelo, lo cual es coherente con el negocio.

6. Conclusiones y recomendaciones

El análisis supervisado representa una herramienta poderosa para la toma de decisiones basadas en datos. La elección de Random Forest para este caso se justificó tanto por su robustez como por su capacidad de interpretación, lo que resulta fundamental en entornos empresariales.

Recomendaciones:

- Aumentar el tamaño del dataset para evitar sobreajuste.

- Aplicar técnicas de balanceo si se detecta desbalance de clases. ● Probar otros

algoritmos (XGBoost, LightGBM) para comparar desempeño. ● Realizar análisis de

importancia de variables para orientar acciones de marketing.

7. Referencias

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

- Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

● Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

● Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32.