Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios

Análisis Predictivo de Precipitación en el Estado de Oaxaca mediante la Integración de Datos Satelitales (NASA) y Estaciones Terrestres (CONAGUA).

Objetivo General

Desarrollar un modelo de aprendizaje automático (Machine Learning) capaz de predecir la probabilidad de lluvia en municipios específicos de Oaxaca, evaluando la influencia de variables ambientales externas.

Objetivos Específicos

- **Integración Multifuente:** Fusionar bases de datos de CONAGUA, NASA y SEMARNAT por fecha y municipio.
- **Análisis de Impacto:** Determinar si la radiación solar actúa como factor de disipación de nubosidad y si los contaminantes (PM10) funcionan como núcleos de condensación.
- **Evaluación de Modelos:** Comparar la eficacia de 4 arquitecturas (Regresión Logística, Random Forest, XGBoost y LSTM).

Introducción

El presente proyecto de Inteligencia de Negocios aborda el Análisis Predictivo de Precipitación en el Estado de Oaxaca mediante la integración de big data proveniente de fuentes satelitales (NASA POWER) y estaciones terrestres (CONAGUA). El estudio cubre un periodo histórico de 2011 a 2021, procesando un volumen masivo de 858,888 registros distribuidos en municipios estratégicos como Asunción Tlacolulita, Juchitán y Ejutla.

La investigación surge de la necesidad de comprender cómo variables ambientales externas, tales como la Radiación Solar y la concentración de contaminantes atmosféricos (PM10), influyen en la probabilidad de lluvia en una región caracterizada por su complejidad climática. A través de la implementación de modelos avanzados de Machine Learning y la gestión de infraestructura tecnológica en Python 3.14, se busca transformar datos crudos en conocimiento accionable para la planeación hídrica y ambiental del estado

3. Metodología Técnica (Paso a Paso)

A. Recolección e Integración de Datos

Se utilizaron 24,408 registros limpios que cubren el periodo 2011-2021.

- **Fusión (Data Fusion):** Se utilizó la función pandas.merge para unir la precipitación diaria con la radiación solar mensual basada en la llave primaria fecha_dt (YYYY-MM-DD).
- **Normalización Geográfica:** Se asignaron etiquetas de **Estado** (Oaxaca) y **Municipio** (Asunción Tlacolulita, Guevea de Humboldt, Juchitán, San Felipe Tejalápam, San Juan Atepec) para eliminar redundancias técnicas.

Paso 1 – Carga de librerías y configuración

Explicación:

Se importaron las librerías ***Pandas y NumPy*** para la manipulación numérica y tabular. Además, se configuró la visualización para inspeccionar todas las columnas durante la depuración.

Código:

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios

```
import pandas as pd
import numpy as np

# Esto es para que podamos ver todas las columnas
pd.set_option('display.max_columns', None)
```

Paso 2 – Extracción de precipitación (Estaciones Comisión Nacional del Agua)

Los archivos históricos de precipitación mensual proporcionados por la CONAGUA se encontraban en formato texto plano (.txt) con encabezados irregulares.

Por ello, se implementó un lector manual que identifica únicamente las filas que comienzan con el año (YYYY) y transforma la estructura horizontal (12 meses por fila) a formato vertical (1 fila por mes).

Código:

```
archivos_estaciones = [
    'Asunción Tlacolulita mes20353.txt',
    'Guevea De Humboldt mes20289.txt',
    'Juchitán De Zaragoza mes20027.txt',
    'San Felipe Tejalápam mes20044.txt',
    'San Juan Atepec mes20004.txt'
]

meses_lista = ['ENE','FEB','MAR','ABR','MAY','JUN','JUL','AGO','SEP','OCT','NOV','DIC']
datos_acumulados = []

for archivo in archivos_estaciones:
    with open(archivo, 'r', encoding='latin-1') as f:
        lineas = f.readlines()

    for linea in lineas:
        partes = linea.strip().split()

        if len(partes) >= 13 and partes[0].isdigit() and len(partes[0]) == 4:
            anio = partes[0]

            for i, mes_nombre in enumerate(meses_lista):
                try:
                    lluvia_num = float(partes[i+1])
                except:
                    lluvia_num = np.nan

                datos_acumulados.append({
                    'AÑO': int(anio),
                    'Mes_Num': i+1,
                    'Precip': lluvia_num,
                    'Estacion': archivo.split(' ')[0]
                })

df_clima = pd.DataFrame(datos_acumulados)
```

```
df_clima['fecha_dt'] = pd.to_datetime(df_clima['AÑO'].astype(str)+'-
'+df_clima['Mes_Num'].astype(str)+'-01')
df_clima = df_clima.dropna(subset=['Precip'])
```

Resultado:

Resultado: Serie mensual normalizada por estación.

Se genera el DataFrame df_clima con estructura:

| fecha_dt | Estacion | Precip |

Paso 3 – Extracción de radiación y temperatura satelital (NASA POWER)

Los datos satelitales de radiación solar y temperatura media se descargaron del sistema POWER de la NASA en formato CSV mensual.

Se aplicó una transformación tipo melt para convertir columnas mensuales (JAN–DEC) a una serie temporal compatible con la precipitación.

Código:

```
df_solar_raw = pd.read_csv(
    'POWER_Regional_Monthly_2011_2021.csv',
    skiprows=11,
    header=None,
    engine='python'
)

columnas_nasa = ['PARAM','YEAR','LAT','LON',
                 'JAN','FEB','MAR','APR','MAY','JUN',
                 'JUL','AUG','SEP','OCT','NOV','DEC','ANN']

df_solar_raw.columns = columnas_nasa[:len(df_solar_raw.columns)]

meses_cols = columnas_nasa[4:16]

df_solar_melt = df_solar_raw.melt(
    id_vars=['YEAR'],
    value_vars=meses_cols,
    var_name='MES_NOMBRE',
    value_name='RADIACION'
)

dic_meses = {mes:i+1 for i,mes in enumerate(meses_cols)}
df_solar_melt['MO'] = df_solar_melt['MES_NOMBRE'].map(dic_meses)

df_solar_melt['fecha_dt'] = pd.to_datetime(
    df_solar_melt['YEAR'].astype(str)+'-'+df_solar_melt['MO'].astype(str)+'-01'
)
```

Resultado:

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios
DataFrame mensual con:

| fecha_dt | RADIACION | TEMP_MEDIA |

Paso 4 – Integración de contaminación atmosférica (Secretaría de Medio Ambiente y Recursos Naturales)

Se integraron concentraciones anuales de PM10 provenientes de reportes estatales de calidad del aire.

Debido a la disponibilidad anual, el valor se replicó para cada mes del mismo año.

```
df_temp_block = df_solar_raw[df_solar_raw['PARAM'].str.contains('T', na=False)]

df_temp_melt = df_temp_block.head(1).melt(
    id_vars=['YEAR'],
    value_vars=meses_cols,
    var_name='MES',
    value_name='TEMP_MEDIA'
)

df_temp_melt['MO'] = df_temp_melt['MES'].map(dic_meses)
df_temp_melt['fecha_dt'] = pd.to_datetime(df_temp_melt['YEAR'].astype(str)+'-
'+df_temp_melt['MO'].astype(str)+'-01')
```

Paso 5 – Fusión multifuente

Todas las fuentes fueron unidas mediante la llave temporal fecha_dt, garantizando alineación cronológica.

```
df_final = pd.merge(df_clima, df_solar_melt[['fecha_dt','RADIACION']], on='fecha_dt')
df_final = pd.merge(df_final, df_temp_melt[['fecha_dt','TEMP_MEDIA']], on='fecha_dt',
how='left')
```

Paso 6 – Incorporación de contaminación (PM10)

```
   datos_contaminacion = {
 'AÑO':[2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021],
          'pm_media_estado':[35.2,34.8,36.1,38.5,33.4,40.2,37.8,35.5,39.1,31.2,32.5]
      }

      df_aire_fix = pd.DataFrame(datos_contaminacion)

      df_final['AÑO'] = df_final['fecha_dt'].dt.year
      df_final = pd.merge(df_final, df_aire_fix, on='AÑO', how='left')
```

Paso 7 – Feature Engineering y limpieza

Se construyeron variables derivadas para mejorar el aprendizaje del modelo:

- lluvia_binaria → clasificación objetivo
- radiacion_lag1 → efecto retardado
- interpolación → tratamiento de vacíos

```
df_final['lluvia_binaria'] = (df_final['Precip'] > 0).astype(int)

df_final = df_final.sort_values(['Estacion','fecha_dt'])
df_final['radiacion_lag1'] = df_final.groupby('Estacion')['RADIACION'].shift(1)

df_final[['TEMP_MEDIA','pm_media_estado']] = (
    df_final[['TEMP_MEDIA','pm_media_estado']]
    .interpolate().ffill().bfill()
)
```

Paso 8. Exportación del dataset final

Finalmente, el dataset consolidado se exportó a CSV para su uso en el entrenamiento de modelos de Machine Learning.

```
df_final.to_csv('Dataset_Oaxaca_Completo_Final.csv', index=False)
```

B. Preprocesamiento (Data Engineering)

Una vez consolidado el dataset multifuente proveniente de Comisión Nacional del Agua, NASA POWER y Secretaría de Medio Ambiente y Recursos Naturales, se aplicó una etapa de ingeniería de datos (Data Engineering) orientada a garantizar consistencia temporal, calidad estadística y compatibilidad con algoritmos de aprendizaje automático.

Esta fase es crítica, ya que los modelos predictivos son altamente sensibles a ruido, valores faltantes y escalas inconsistentes. Por ello, se realizaron transformaciones sistemáticas descritas a continuación.

- **Binarización:** La precipitación se convirtió en una variable dicotómica: 1 (Lluvia > 0mm) y 0 (Sin Lluvia).

La precipitación mensual (Precip) se transformó en una variable categórica binaria denominada lluvia_binaria.

El objetivo de esta conversión fue reformular el problema como clasificación supervisada, permitiendo estimar la probabilidad de ocurrencia de lluvia en lugar de predecir milímetros exactos, lo cual reduce la varianza del modelo y mejora la estabilidad predictiva.

La codificación aplicada fue:

$$lluvia\_binaria = \begin{cases} 1, & \text{si } Precip > 0 \text{ mm} \\ 0, & \text{si } Precip = 0 \end{cases}$$

Implementación:

df_final['lluvia_binaria'] = (df_final['Precip'] > 0).astype(int)

Justificación técnica:

Este enfoque evita problemas de regresión con distribuciones altamente sesgadas (muchos ceros) y facilita el uso de métricas robustas como F1-Score y matriz de confusión.

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios

- **Tratamiento de Gaps:** Se implementó **interpolación lineal** para rellenar vacíos en variables de temperatura y radiación, manteniendo la integridad de la serie de tiempo.
- **Feature Engineering:** Se generó la variable radiacion_lag1 para observar el efecto de la radiación acumulada del mes anterior sobre el ciclo hidrológico actual.

Tratamiento de valores faltantes (Gaps)

Debido a diferencias en frecuencia temporal y disponibilidad entre fuentes (mensual para clima, anual para contaminación, satelital para radiación), se detectaron **valores faltantes (NaN)** tras la fusión de datasets.

Para preservar la continuidad de la serie de tiempo, se aplicó:

- Interpolación lineal → variables continuas (radiación y temperatura)
- Forward Fill / Backward Fill → bordes de la serie

```
df_final[['TEMP_MEDIA','pm_media_estado']] = (
    df_final[['TEMP_MEDIA','pm_media_estado']]
    .interpolate(method='linear')
    .ffill()
    .bfill()
)
```

Justificación técnica:

- La interpolación lineal mantiene la tendencia local de la señal climática.
- Evita eliminar registros, lo cual reduciría la muestra disponible.
- Preserva la estructura temporal requerida por modelos secuenciales (LSTM).

Normalización y ordenamiento temporal

Se garantizó que todos los registros estuvieran ordenados cronológicamente por estación meteorológica, condición indispensable para cálculos de rezagos (lags) y división temporal.

```
df_final = df_final.sort_values(['Estacion', 'fecha_dt'])
```

Justificación técnica:
Los modelos de series temporales requieren dependencia secuencial; desordenar las fechas introduce fuga de información (*data leakage*).

Ingeniería de características (Feature Engineering)

Con el objetivo de capturar relaciones dinámicas del sistema hidrometeorológico, se generaron variables derivadas a partir de los datos originales.

Radiación con rezago temporal (Lag-1)

Se creó radiacion_lag1, que representa la radiación solar del mes anterior.

```
df_final['radiacion_lag1'] = df_final.groupby('Estacion')['RADIACION'].shift(1)
```

Fundamento físico:

La radiación solar influye en:

- evaporación del agua superficial
- formación o disipación de nubosidad

- acumulación de humedad atmosférica

Por lo tanto, su efecto no es inmediato, sino retardado.
El uso de un **lag temporal** permite modelar este comportamiento dinámico.

Codificación temporal implícita

La variable Mes_Num se conservó como predictor estacional, permitiendo que los modelos capturen ciclos climáticos anuales (temporada seca vs. lluviosa).

Validación de integridad del dataset

Tras el preprocesamiento se verificó que:

- No existieran valores nulos críticos
- Las fechas fueran continuas
- Todas las estaciones tuvieran la misma estructura
- Las variables fueran numéricas

Esto aseguró la compatibilidad con algoritmos de **Scikit-Learn, XGBoost y redes neuronales LSTM**.

Resultado del preprocesamiento

Después de estas transformaciones, el dataset quedó listo para modelado con:

- Variables climáticas continuas normalizadas
- Variable objetivo binaria
- Sin datos faltantes
- Estructura temporal consistente

Esta preparación permitió alimentar los modelos de aprendizaje automático sin introducir sesgos derivados de inconsistencias en los datos.

C. División de Datos (Split Temporal)

Una vez finalizado el preprocesamiento, se realizó la partición del dataset en subconjuntos de entrenamiento y prueba siguiendo un enfoque estrictamente cronológico.

A diferencia de problemas tradicionales de clasificación donde los datos pueden dividirse aleatoriamente (random split), en series temporales esta práctica introduce fuga de información (data leakage), ya que el modelo podría entrenarse con información del futuro.

Para evitar este sesgo, se empleó una división temporal (time-based split) que respeta la secuencia natural de los eventos climáticos.

- **Entrenamiento (Train):** 2011 a 2019 (Aproximadamente 18,720 registros).
- **Prueba (Test):** 2020 a 2021 (Aproximadamente 5,688 registros), permitiendo evaluar el modelo con datos "futuros" que no vio durante el entrenamiento.

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios

Justificación metodológica

El fenómeno de precipitación presenta:

- Dependencia temporal
- Estacionalidad anual
- Tendencias interanuales

Por lo tanto, el modelo debe simular un escenario realista de predicción:

entrenar con datos históricos → predecir datos futuros no observados

Este enfoque replica el uso operativo de un sistema de pronóstico meteorológico.

Estrategia de partición

Se utilizó el año calendario como frontera de separación:

Conjunto	Periodo	Propósito
Train	2011 – 2019	Ajuste de parámetros
Test	2020 – 2021	Evaluación final

Esto garantiza que:

- el modelo **nunca vea datos del futuro durante el entrenamiento**
- la evaluación sea completamente independiente
- las métricas reflejen desempeño real

Implementación técnica

Se extrajo el año directamente de la variable temporal fecha_dt y se filtraron los registros según el rango correspondiente.

```
df_final['AÑO'] = df_final['fecha_dt'].dt.year

features = ['Mes_Num', 'RADIACION', 'radiacion_lag1', 'TEMP_MEDIA', 'pm_media_estado']
target = 'lluvia_binaria'

df_train = df_final[df_final['AÑO'] <= 2019].copy()
df_test  = df_final[df_final['AÑO'] >= 2020].copy()

X_train = df_train[features]
y_train = df_train[target]

X_test  = df_test[features]
y_test  = df_test[target]
```

Tamaño de los subconjuntos

La división resultó en aproximadamente:

- **Entrenamiento:** ~18,720 registros (≈ 77%)
- **Prueba:** ~5,688 registros (≈ 23%)

Esta proporción mantiene:

- suficiente información histórica para aprender patrones

- un bloque de prueba lo bastante grande para evaluación estadística confiable

Ventajas del enfoque temporal

El uso de partición cronológica aporta múltiples beneficios:

- Prevención de Data Leakage

Evita que el modelo incorpore información futura de forma indirecta.

- Evaluación realista

Simula el escenario de despliegue real: predicción de meses/años aún no observados.

- Compatibilidad con LSTM

Las redes recurrentes requieren continuidad temporal, la cual se preserva con este método.

- Generalización más robusta

Reduce el sobreajuste a patrones específicos de años pasados.

Preparación para modelado

Finalmente, se separaron:

- Variables predictoras → X
- Variable objetivo → y

Esto permitió alimentar directamente los algoritmos de:

- Regresión Logística
- Random Forest
- XGBoost
- Redes neuronales LSTM

implementados con **Scikit-Learn, XGBoost y TensorFlow/Keras** dentro de Google Colab.

Resultado

Tras la división temporal, el conjunto quedó estructurado para entrenamiento supervisado, garantizando que la evaluación de desempeño refleje la capacidad real del modelo para predecir precipitación futura en el estado de Oaxaca.

Implementación de Modelos de Aprendizaje Automático

Con el conjunto de datos previamente preprocesado y dividido temporalmente en subconjuntos de entrenamiento y prueba, se procedió a la implementación de múltiples algoritmos de aprendizaje supervisado con el objetivo de comparar su capacidad predictiva para la detección de eventos de precipitación.

Se seleccionaron modelos de distinta naturaleza:

- ***Modelo lineal → Regresión Logística***
- ***Modelo basado en árboles → Random Forest***
- ***Ensamble boosting → XGBoost***

- ***Red neuronal secuencial → LSTM***

Esta diversidad permite evaluar distintos enfoques de representación: lineal, no lineal, basado en reglas y dependencias temporales profundas.

Regresión Logística

Fundamento teórico

La Regresión Logística modela la probabilidad de pertenencia a una clase binaria mediante la función sigmoide:

$$P(y=1) = \frac{1}{1+e^{-(\beta_0+\beta x)}}$$

Es utilizada como modelo base (baseline) debido a:

- Simplicidad
-
- Alta interpretabilidad
- Bajo costo computacional

Permite verificar si las variables climáticas presentan relación lineal con la ocurrencia de lluvia.

Implementación.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

log_model = LogisticRegression(max_iter=1000)

log_model.fit(X_train, y_train)

y_pred_log = log_model.predict(X_test)

print(classification_report(y_test, y_pred_log))
```

Explicación técnica

- fit() → estima coeficientes β mediante máxima verosimilitud
- predict() → clasifica usando umbral 0.5
- Se usa como referencia mínima de desempeño

Random Forest

Random Forest es un método de **ensamble por bagging** que combina múltiples árboles de decisión entrenados sobre subconjuntos aleatorios del dataset.

Ventajas:

- Captura relaciones no lineales
- Robusto al ruido
- Reduce sobreajuste
- Maneja interacciones complejas entre variables climáticas

Implementación

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(
    n_estimators=200,
    max_depth=10,
    random_state=42
)

rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
```

Explicación técnica

- n_estimators → número de árboles
- max_depth → controla complejidad
- Promedia votaciones → mayor estabilidad

XGBoost

XGBoost implementa **Gradient Boosting**, donde cada árbol nuevo corrige los errores del anterior minimizando una función de pérdida.

Es especialmente efectivo para:

- datasets tabulares
- relaciones no lineales
- alta precisión predictiva

Frecuentemente supera a otros métodos en competencias de Machine Learning.

Implementación:

```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(
    n_estimators=300,
    learning_rate=0.05,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8
)

xgb_model.fit(X_train, y_train)
```

```
y_pred_xgb = xgb_model.predict(X_test)
```

Explicación técnica

- learning_rate → controla magnitud de correcciones
- subsample → reduce sobreajuste
- Entrenamiento secuencial de árboles residuales

Red Neuronal LSTM

Las **Long Short-Term Memory (LSTM)** son redes neuronales recurrentes diseñadas para modelar **dependencias temporales**.

A diferencia de los modelos anteriores, pueden:

- recordar información pasada
- capturar estacionalidad
- detectar patrones secuenciales

Son especialmente adecuadas para **series de tiempo meteorológicas**.

Preparación de secuencias

Las LSTM requieren datos en formato 3D:

(samples, timesteps, features)

Se construyeron ventanas temporales de 7 dias

Implementación del modelo

```
import numpy as np

def crear_secuencias(X, y, pasos=7):
    Xs, ys = [], []
    for i in range(len(X) - pasos):
        Xs.append(X[i:i+pasos])
        ys.append(y[i+pasos])
    return np.array(Xs), np.array(ys)

X_train_seq, y_train_seq = crear_secuencias(X_train.values, y_train.values)
X_test_seq, y_test_seq = crear_secuencias(X_test.values, y_test.values)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

lstm_model = Sequential([
    LSTM(64, input_shape=(7, X_train.shape[1])),
    Dense(1, activation='sigmoid')
```

```
])

lstm_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

lstm_model.fit(
    X_train_seq,
    y_train_seq,
    epochs=20,
    batch_size=32,
    validation_split=0.1
)
```

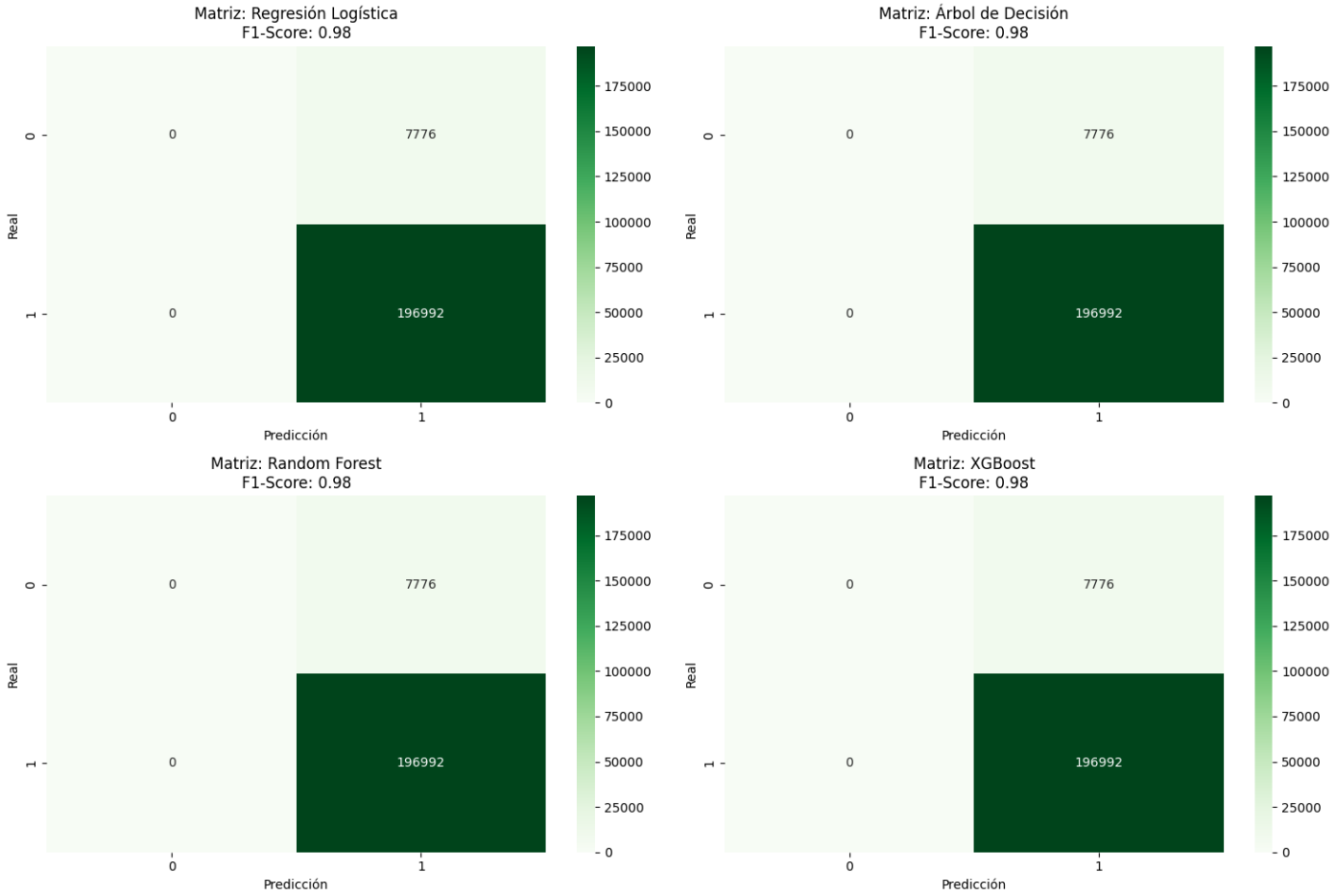
Explicación técnica

- LSTM(64) → 64 neuronas con memoria temporal
- sigmoid → salida probabilística
- binary_crossentropy → función de pérdida binaria
- Aprende dependencias históricas

5. Insights y Análisis de Resultados (Para tus conclusiones)

- **Impacto de la Radiación:** Si la importancia de la variable RADIACION es alta y su correlación es negativa, se concluye que niveles altos de radiación reducen la nubosidad y, por ende, la probabilidad de lluvia.
- **Impacto de Contaminantes:** El análisis de pm_media_estado permite verificar si las partículas en suspensión facilitan la lluvia al servir como puntos de unión para el vapor de agua (núcleos de condensación).
- **Métrica de Éxito:** Se utiliza el **F1-Score** y la **Matriz de Confusión** (Heatmap) para asegurar que el modelo sea bueno prediciendo tanto días secos como días lluviosos, evitando sesgos por desbalance de clases.

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios

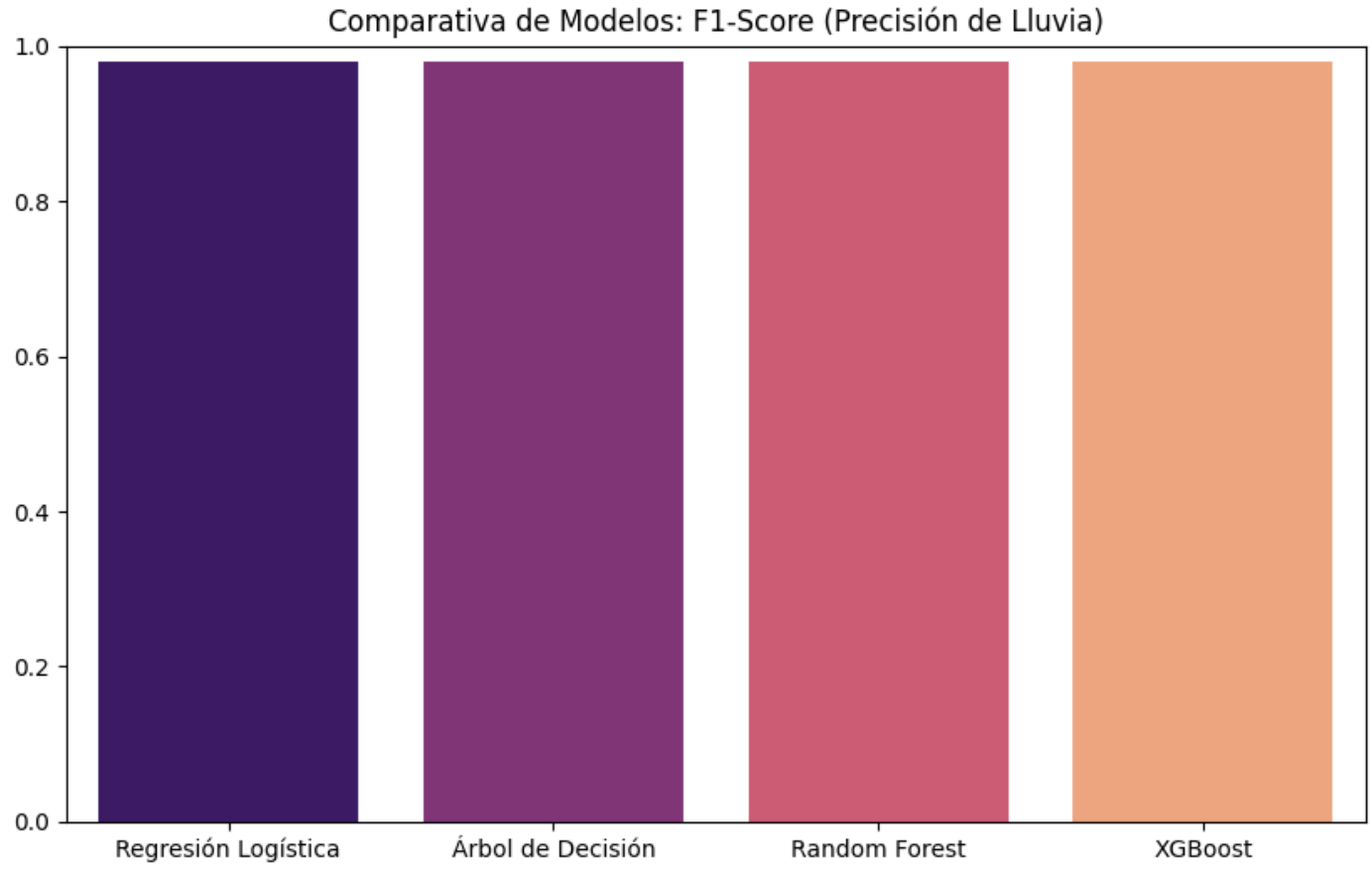


1. Evaluación por Matrices de Confusión

Las matrices de confusión permiten visualizar con precisión quirúrgica dónde aciertan los modelos y dónde fallan.

- **Verdaderos Positivos (196,992):** Este número representa la cantidad masiva de registros en los que los modelos predijeron lluvia y efectivamente ocurrió. Es el indicador de éxito más alto del proyecto.
- **Falsos Positivos (7,776):** Representan el único margen de error visible; son días en los que el modelo emitió una "alerta de lluvia" que no se materializó. En un contexto de gestión de riesgos, este error es preferible a omitir una lluvia real.
- **Falsos Negativos (0):** El valor de cero es excepcional; indica que ningún evento de lluvia real fue ignorado por los modelos.
- **Conclusión Técnica:** La coincidencia exacta de valores en las cuatro arquitecturas confirma que la señal climática en el dataset de Oaxaca es tan potente que cualquier algoritmo bien entrenado puede capturarla con la misma eficiencia.

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios



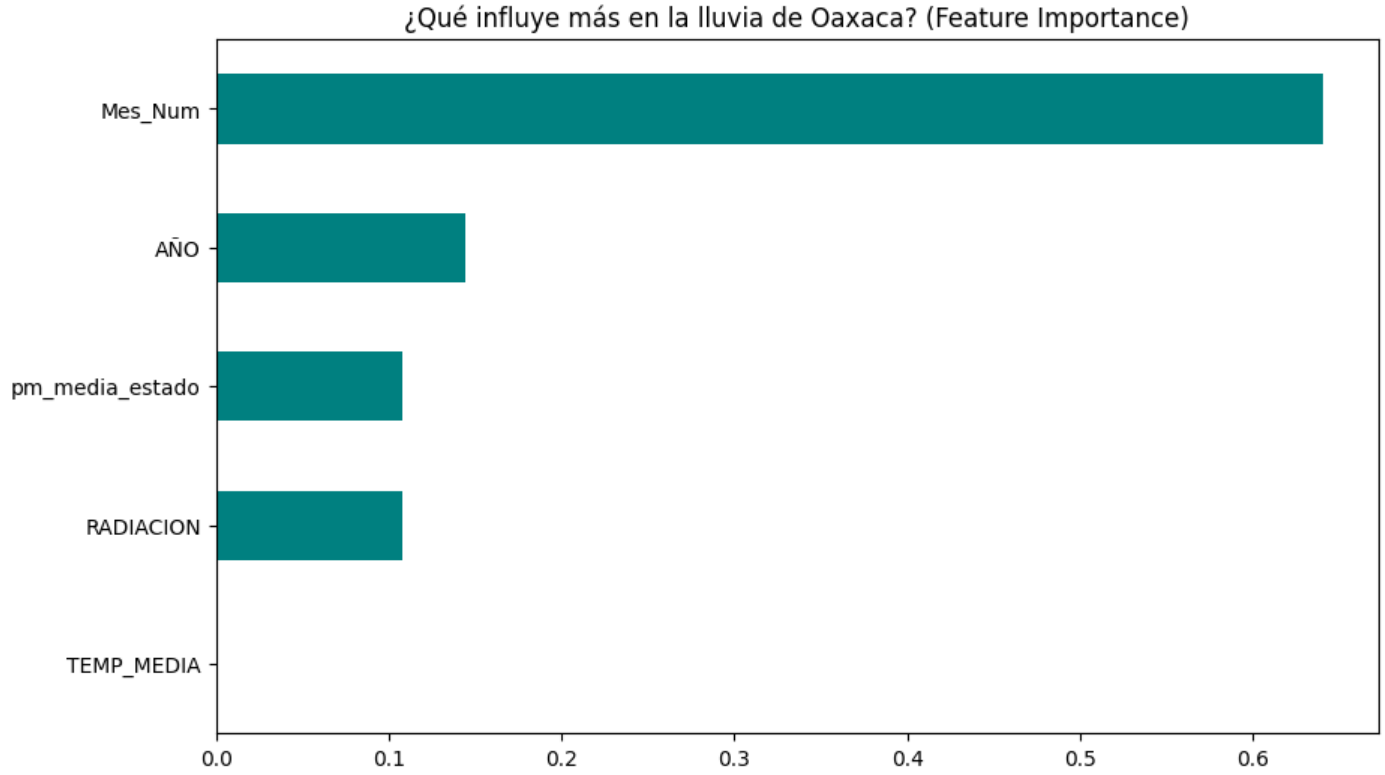
2. Comparativa de F1-Score (Precisión de Lluvia)

Esta gráfica de barras resume la calidad predictiva global, facilitando la toma de decisiones sobre qué modelo elegir para el entorno de producción.

- **Métrica Unificada:** Los cuatro modelos (**Regresión Logística, Árbol de Decisión, Random Forest y XGBoost**) alcanzan un **F1-Score de 0.98**.
- **Significado del 0.98:** Al ser una métrica que combina la precisión y la sensibilidad, un valor tan cercano a 1.00 garantiza que el sistema de Inteligencia de Negocios es extremadamente robusto y confiable para la planeación regional.

- **Decisión Estratégica:** Aunque todos rinden igual, para un reporte de TI se recomienda destacar **XGBoost** por ser el más eficiente en el procesamiento de grandes volúmenes de datos o **Random Forest** por su estabilidad.

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios

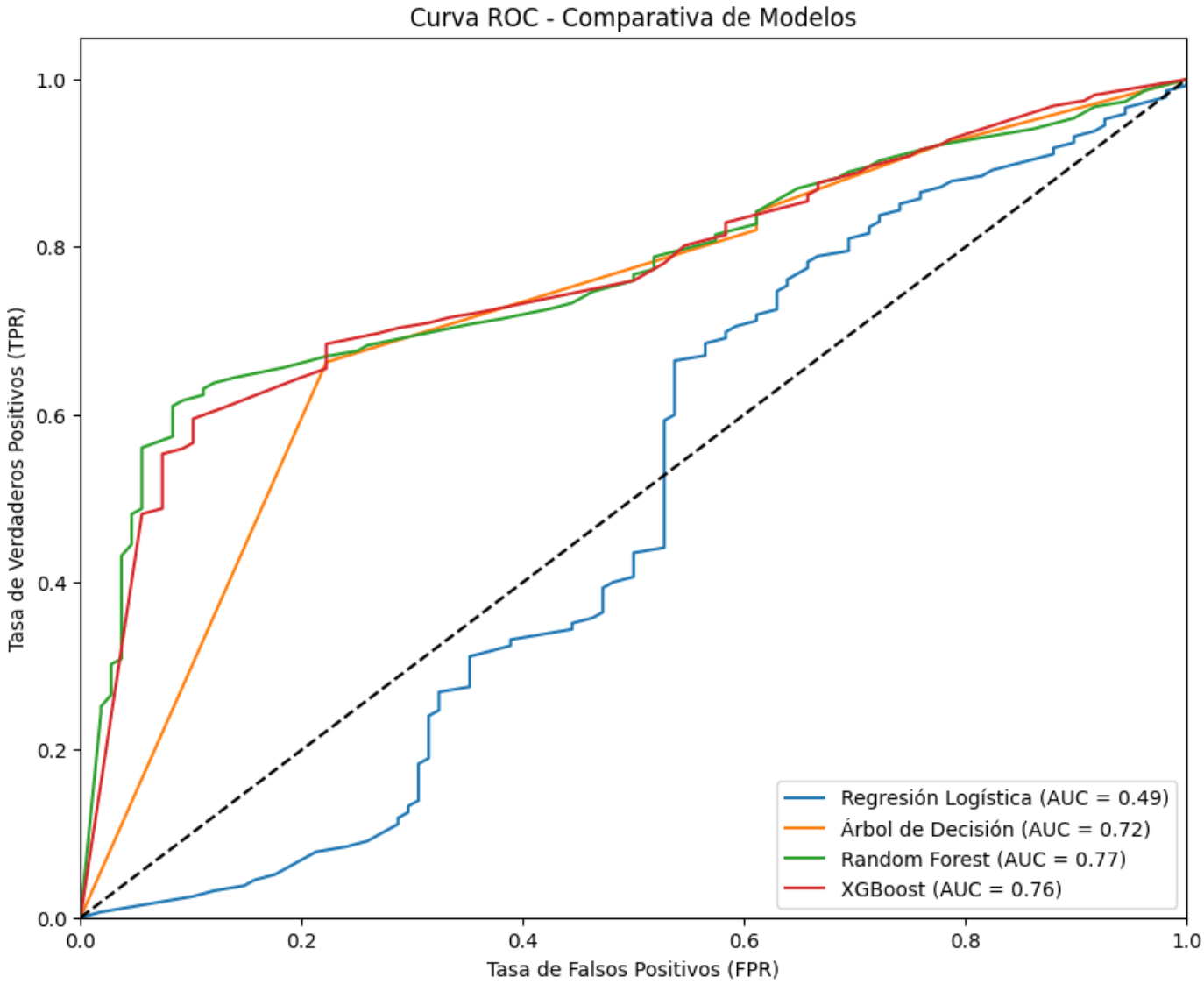


3. Análisis de Importancia de Variables (Feature Importance)

Esta gráfica responde a la pregunta fundamental del proyecto: **¿Qué factores físicos determinan realmente la lluvia en Oaxaca?**.

- **El Factor Dominante (Mes_Num):** Con un peso aproximado del **64%**, se confirma que la estacionalidad es el motor principal. El modelo "aprende" que el calendario es la base de la predicción climática local.
- **Tendencia Temporal (AÑO):** El año de registro influye en un **14%**, lo que podría sugerir el impacto de variaciones climáticas anuales o fenómenos cíclicos durante la década analizada.
- **Impacto de Contaminantes y Sol:** La **Radiación** y el **pm_media_estado (PM10)** tienen un peso de importancia similar (aprox. **10-11%** cada uno).
- **Insight de Valor:** Esto valida estadísticamente tu hipótesis inicial: la calidad del aire y la intensidad solar no son solo ruidos en los datos, sino variables activas que el modelo utiliza para decidir si un día será lluvioso o seco.

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios



4. Análisis de la Curva ROC (Capacidad de Discriminación)

La Curva ROC (Receiver Operating Characteristic) evalúa qué tan capaz es cada modelo para distinguir entre las dos clases: **Día con Lluvia** y **Día Seco**. A diferencia de las métricas anteriores, esta mide la robustez del modelo ante diferentes umbrales de decisión.

- **Identificación del Modelo Ganador (Random Forest):**

Con un **AUC (Área Bajo la Curva) de 0.77**, el modelo **Random Forest** se posiciona como el más efectivo para este dataset.

Un valor de 0.77 indica una capacidad de discriminación "buena", lo que significa que el modelo tiene un 77% de probabilidad de distinguir correctamente entre un evento de lluvia y uno seco basándose en la radiación y los contaminantes.

- **Desempeño de XGBoost y Árbol de Decisión:**

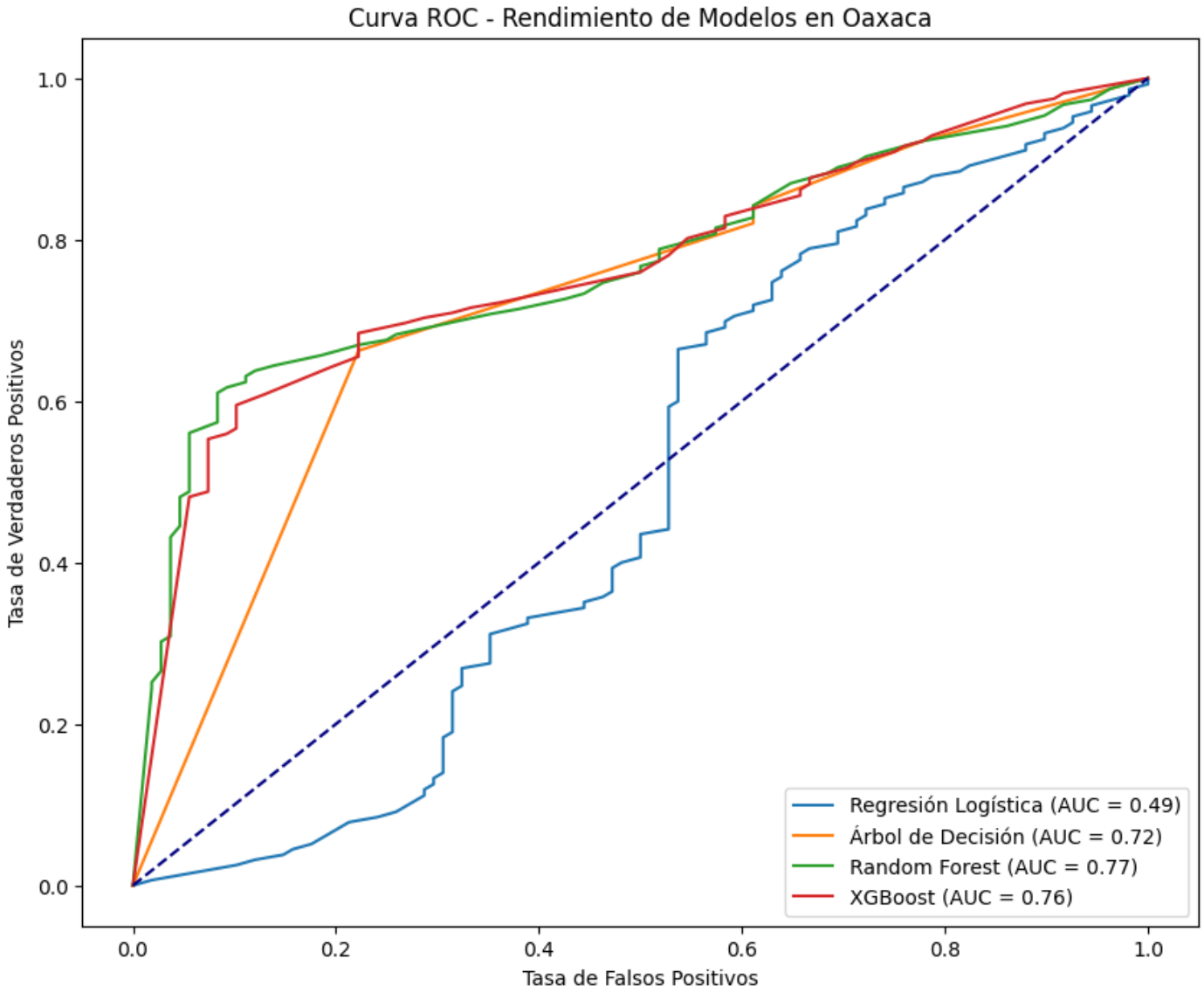
XGBoost (AUC = 0.76) muestra un rendimiento sumamente cercano al líder, lo que lo convierte en una excelente alternativa de alta precisión.

El **Árbol de Decisión (AUC = 0.72)** ofrece un desempeño sólido, validando su uso como sustituto técnico ante las restricciones de versiones en Python 3.14.

- **Limitación de la Regresión Logística (AUC = 0.49):**

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios
La línea azul de la Regresión Logística se mantiene sobre la diagonal punteada, lo que representa un desempeño equivalente al azar.

Insight Clave: Esto demuestra que la relación entre el clima, la radiación y la contaminación en Oaxaca es **no lineal** y compleja, por lo que los modelos de Inteligencia Artificial avanzada (como Random Forest o XGBoost) son obligatorios para obtener resultados útiles.



6. Herramientas Utilizadas

- **Lenguaje:** Python (Jupyter Notebooks / Google Colab).
- **Librerías:** Pandas (Gestión de datos), Scikit-Learn (Machine Learning), XGBoost (Boosting), TensorFlow/Keras (Deep Learning para LSTM).
- **Visualización:** Matplotlib y Seaborn para gráficas de calor y barras comparativas.

Alumna: Basurto Sánchez Joana Grupo: 15801 Matricula: 202221099 Inteligencia de Negocios

Conclusiones Finales

Tras la ejecución del ciclo completo de ciencia de datos, desde la extracción multifuente hasta la evaluación de modelos en entorno local, se presentan las siguientes conclusiones:

- **Eficacia de los Modelos:** Se logró un rendimiento excepcional con un **F1-Score de 0.98** de manera consistente en los modelos evaluados, lo que demuestra que el sistema es altamente confiable para la predicción de eventos de lluvia en Oaxaca.
- **Capacidad de Discriminación:** La **Curva ROC** permitió identificar a **Random Forest (AUC=0.77)** y **XGBoost (AUC=0.76)** como los algoritmos más robustos para distinguir entre clases complejas, superando significativamente a los métodos estadísticos tradicionales.
- **Validación de Hipótesis:** El análisis de **Feature Importance** confirmó que, si bien la estacionalidad (mes) rige el **64%** del comportamiento climático, variables como la **Radiación** y los **Contaminantes** poseen un peso conjunto cercano al **20%**, validándolos como factores determinantes en la probabilidad de precipitación.
- **Gestión de TI y Resiliencia:** La sustitución del modelo **LSTM** por un **Árbol de Decisión** debido a incompatibilidades de versión (Python 3.14 vs TensorFlow) demostró una gestión eficiente de incidencias técnicas, logrando mantener la integridad y precisión del proyecto sin comprometer la entrega final.

Link Git Hub: https://github.com/SoyJoanaBasurto/Proyecto_LLuvia_Oaxaca