Práctica: Diferenciación entre Data Drift y Concept Drift

Objetivo:

- Simular y entender las diferencias entre el data drift y el concept drift.

Herramientas Utilizadas:

- Python
- Scikit-Learn

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

Generar Datos Iniciales:

- Genera un conjunto de datos inicial con características y etiquetas.

```
def generate_initial_data(num_samples=1000, num_features=5):
    np.random.seed(42)
    X = np.random.rand(num_samples, num_features)
    y = (X.sum(axis=1) > num_features / 2).astype(int)
    return X, y


X_initial, y_initial = generate_initial_data()
```

Entrenar un Modelo Inicial:

- Entrena un modelo de clasificación con los datos iniciales.

```
X_train, X_test, y_train, y_test = train_test_split(X_initial, y_initial, test_size=0.2,
random_state=42)

model_initial = RandomForestClassifier(random_state=42)

model_initial.fit(X_train, y_train)

y_pred_initial = model_initial.predict(X_test)
```

```
accuracy_initial = accuracy_score(y_test, y_pred_initial)

print(f'Precisión del modelo inicial: {accuracy_initial:.2f}')
```

Simular Data Drift:

- Genera nuevos datos con una distribución diferente para simular el data drift.

```
def simulate_data_drift(X_old, num_samples=200, num_features=5):
    X_drifted = np.random.rand(num_samples, num_features)
    return np.concatenate([X_old, X_drifted]), np.concatenate([y_initial,
np.ones(num_samples)])

# Simular data drift
X_drifted, y_drifted = simulate_data_drift(X_initial)
```

Evaluar el Modelo con Data Drift:

- Evalúa el modelo inicial en los nuevos datos y observa cómo cambia la precisión.

```
# Evaluar el modelo inicial con datos drift
y_pred_drifted = model_initial.predict(X_drifted)
accuracy_drifted = accuracy_score(y_drifted, y_pred_drifted)
print(f'Precisión del modelo con data drift: {accuracy_drifted:.2f}')
```

Simular Concept Drift:

- Modifica el concepto subyacente en los datos para simular el concept drift.

```
def simulate_concept_drift(X_old, num_samples=200, num_features=5):

    X_drifted = np.random.rand(num_samples, num_features) * 2  # Cambio en el
concepto

    return np.concatenate([X_old, X_drifted]), np.concatenate([y_initial,
np.ones(num_samples)])
```

 # Simular concept drift

X_drifted_concept, y_drifted_concept = simulate_concept_drift(X_initial)

Evaluar el Modelo con Concept Drift:

- Evalúa el modelo inicial en los nuevos datos con concept drift y observa cómo cambia la precisión**.**

y_pred_drifted_concept = model_initial.predict(X_drifted_concept)

accuracy_drifted_concept = accuracy_score(y_drifted_concept, y_pred_drifted_concept)

print(f'Precisión del modelo con concept drift: {accuracy_drifted_concept:.2f}')

¿Cómo cambia el rendimiento del modelo en situaciones de data drift y concept drift?

Argumenta tu respuesta