E01 – Correcciones

Ana Sofía Avila Gálvez
745247

1. ¿Qué es un pipeline?

Es una secuencia ordenada de pasos que se aplican a los datos. Sirve para automatizar procesos como: limpiar datos, escalarlos, entrenar un modelo y evaluarlo, todo dentro de un mismo flujo.

2. ¿Cuál es el propósito de realizar regresiones? Explica las ventajas y desventajas de los dos planteamientos vistos en clase.

Las regresiones buscan explicar o predecir una variable en función de otras.

3. ¿En qué consiste el proceso de escalamiento de factores?

Es transformar las variables para que estén en la misma escala para evitar que variables con valores grandes dominen sobre las más pequeñas.

4. Explica el propósito de penalizar factores en una regresión.

Agrega un “costo” por tener coeficientes muy grandes. Para reducir el sobreajuste, ayudar a que el modelo sea más estable y generalizable.

5. ¿Cuál es la relación entre escalamiento y penalización?

El escalamiento es necesario porque la penalización se aplica sobre los coeficientes. Si las variables no están en la misma escala, la penalización castigará más a unas que a otras de forma injusta.

6. Explica el concepto de una prueba de hipótesis.

Es un procedimiento usado en la estadística para decidir, con base en datos, si hay suficiente evidencia para rechazar una afirmación inicial frente a una alternativa.

7. Explica la interpretación de un p-value de una prueba de hipótesis que compara contra una media μ.

P-value indica la probabilidad de observar datos tan extremos como los que tenemos, suponiendo que H0 es cierta.

8. Describe el propósito de realizar cross-validation.

Es una técnica para validar la calidad de un modelo dividiendo los datos en varios subconjuntos. Te ayuda a verificar el modelo no solo funciona en los datos de entrenamiento, sino también en datos nuevos.

9. Describe los pasos que seguirías al hacer un análisis exploratorio de datos. Justifica cada paso.

- Cargar y revisar datos → entender su estructura.
- Limpiar datos → tratar valores faltantes o raros.
- Describir con estadísticas básicas → medias, varianzas, correlaciones.
- Visualizar → gráficos para detectar patrones o anomalías.
- Hipótesis iniciales → preparar lo que probarás con modelos.

10. ¿Qué es el teorema del límite central?

El teorema del límite central dice que, aunque los datos originales no sigan una forma normal, si tomamos muchas muestras y calculamos sus promedios, esos promedios se van a parecer cada vez más a una curva normal (de campana).

Parte práctica pregunta 3:

Original:

```
X = data[["Released_Year", "Runtime", "Meta_score", "Gross"]]
y = data["IMDB_Rating"]
```

```
X = X.apply(pd.to_numeric, errors="coerce")
y = pd.to_numeric(y, errors="coerce")
df = pd.concat([X, y], axis=1).dropna()
```

```
X = sm.add_constant(df[["Released_Year", "Runtime", "Meta_score", "Gross"]])
y = df["IMDB_Rating"]
```

```
modelo = sm.OLS(y, X).fit()
modelo.summary()
```

Corrección:

```
X = data[["Released_Year", "Runtime", "Meta_score", "Gross"]]
y = data["IMDB_Rating"]
```

```
df = data.drop(columns=["Series_Title", "Director", "Unnamed: 0"])
```

```
#De runtime quite "min"
df["Runtime"] = df["Runtime"].str.replace(" min", "").astype(float)
```

```
#Quitar comas y convertir a numérico
df["Gross"] = df["Gross"].str.replace(",", "")
df["Gross"] = pd.to_numeric(df["Gross"], errors="coerce")
```

```
#Asegurar que sea numérica
df["Released_Year"] = pd.to_numeric(df["Released_Year"], errors="coerce")
```

```
df = df.dropna()
```

```
X = df.drop(columns=["IMDB_Rating"])
y = df["IMDB_Rating"]
```

```
X = pd.get_dummies(X, drop_first=True)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
modelo = LinearRegression()
modelo.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = modelo.predict(X_test)
r2 = r2_score(y_test, y_pred)
```

```
r2
```

```
0.14302346223683582
```

```
#P-values

X2 = sm.add_constant(X)
```

```
#Volver todo numérico
X2 = X2.apply(pd.to_numeric, errors="coerce").astype(float)
y = pd.to_numeric(df["IMDB_Rating"], errors="coerce").astype(float)
```

```
#Limpiar
df_model = pd.concat([X2, y], axis=1).dropna()
X2_nuevo = df_model.drop("IMDB_Rating", axis=1).astype(float)
y_nuevo = df_model["IMDB_Rating"].astype(float)
```

```
modelo_ols = sm.OLS(y_nuevo, X2_nuevo).fit()
modelo_ols.summary()
```

- **¿Cuál fue el error?**

Usé “statsmodels.OLS” directamente en lugar de construir primero el modelo con sklearn y preparar los datos como se pedía. Además, en el primer código me daba error porque las variables no estaban convertidas correctamente a valores numéricos y había columnas innecesarias en el dataset.

- **¿Cuál fue la correción?**

-Convetir las variables (Runtime, Gross, Released_Year) para que fueran numéricas.

-Eliminar columnas irrelevantes (Series_Title, Director, etc.).

-Separar correctamente X y y.

-Ajustar primero el modelo con LinearRegression de sklearn.

-Después, calcular los p-values con statsmodels usando los datos ya transformados y limpios.

- **¿Por qué se cometió el error?**

Porque intenté aplicar el método de otra librería (statsmodels) sin respetar la estructura que pedía el ejercicio (usar sklearn). Además, no validé que todas las variables fueran numéricas y limpias antes de entrenar el modelo, lo cual provocaba errores en la ejecución.

- **¿Cómo se puede evitar este error en el futuro?**

-Revisar el dataset antes de modelar: usar df.info() y df.head() para confirmar que todas las variables están en formato correcto.

-Usar pasos de limpieza estandarizados: siempre convertir variables con pd.to_numeric(errors="coerce") y usar dropna() antes de entrenar.