GUÍA DETALLADA: PLANTILLA DE CÓDIGO PARA EDA

Ejemplo de Implementación de los 11 Pasos

Esta guía proporciona una plantilla de código que los estudiantes pueden adaptar a su dataset específico. Cada sección incluye el código base y explicaciones detalladas.

PASO 1: IMPORTAR LIBRERÍAS

Código:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from scipy.stats import shapiro
import warnings
warnings.filterwarnings('ignore')
```

✅ Qué hacer: Copiar este código al inicio de su notebook.

PASO 2: IMPORTAR DATOS

Código:

```
# Cargar el dataset (ajustar la URL según su grupo)
datos = pd.read_csv('URL_DE_SU_DATASET')

# Crear una copia de trabajo
datos_trabajo = datos.copy()
```

```
# Visualizar las primeras filas
print(datos_trabajo.head())
print(f"\nDimensiones del dataset: {datos_trabajo.shape}")
```

✅ Qué hacer: Reemplazar URL_DE_SU_DATASET con la URL asignada a su grupo.

✅ Análisis: Describir qué representa cada columna del dataset.

PASO 3: REVISAR DATOS NAN

Código:

```
# Información general del dataset
print("=== INFORMACIÓN DEL DATASET ===")
datos_trabajo.info()

# Estadísticos descriptivos
print("\n=== ESTADÍSTICOS DESCRIPTIVOS ===")
print(datos_trabajo.describe())

# Conteo de valores faltantes
print("\n=== VALORES FALTANTES ===")
print(datos_trabajo.isna().sum())

# Visualización de valores faltantes
plt.figure(figsize=(10, 6))
sns.heatmap(datos_trabajo.isna(), cbar=True, cmap='viridis')
plt.title('Mapa de Valores Faltantes')
plt.tight_layout()
plt.show()
```

✅ Análisis requerido:

- ¿Cuántos valores faltantes hay por columna?
- ¿Qué porcentaje del total representan?
- ¿Hay patrones en los valores faltantes?

PASO 4: LIMPIAR DATOS

Código (adaptar según necesidades):

```
# Ejemplo 1: Eliminar duplicados
datos_trabajo = datos_trabajo.drop_duplicates()

# Ejemplo 2: Tratar valores faltantes
# Opción A: Eliminar filas con NaN
# datos_trabajo = datos_trabajo.dropna()

# Opción B: Rellenar con mediana (variables numéricas)
for col in
datos_trabajo.select_dtypes(include=[np.number]).columns:
  if datos_trabajo[col].isna().sum() > 0:
    datos_trabajo[col].fillna(datos_trabajo[col].median(),
inplace=True)

# Opción C: Rellenar con moda (variables categóricas)
for col in datos_trabajo.select_dtypes(include=['object']).columns:
  if datos_trabajo[col].isna().sum() > 0:
    datos_trabajo[col].fillna(datos_trabajo[col].mode()[0],
inplace=True)

# Ejemplo 3: Estandarizar categorías (ajustar según su dataset)
# datos_trabajo['columna'] =
datos_trabajo['columna'].str.lower().str.strip()

# Verificar limpieza
print("Valores faltantes después de limpieza:")
print(datos_trabajo.isna().sum())
```

✅ Documentar: Explicar qué estrategia de limpieza usaron y por qué.

PASO 5: REALIZAR GRÁFICAS PARA ANALIZAR TENDENCIAS

Crear mínimo 5 visualizaciones diferentes. Ejemplos:

```
# Gráfica 1: Distribución de variable numérica
plt.figure(figsize=(10, 6))
sns.histplot(datos_trabajo['variable_numerica'], kde=True, bins=30)
plt.title('Distribución de [Nombre Variable]')
plt.xlabel('[Nombre Variable]')
plt.ylabel('Frecuencia')
plt.tight_layout()
plt.show()

# Gráfica 2: Conteo de variable categórica
plt.figure(figsize=(10, 6))
sns.countplot(data=datos_trabajo, y='variable_categorica',

order=datos_trabajo['variable_categorica'].value_counts().index)
plt.title('Distribución de [Nombre Variable]')
plt.xlabel('Cantidad')
plt.tight_layout()
plt.show()

# Gráfica 3: Boxplot para detectar outliers
plt.figure(figsize=(10, 6))
sns.boxplot(data=datos_trabajo, x='variable_numerica')
plt.title('Boxplot de [Nombre Variable]')
plt.tight_layout()
plt.show()

# Gráfica 4: Relación entre dos variables
plt.figure(figsize=(10, 6))
sns.scatterplot(data=datos_trabajo, x='variable1', y='variable2',
alpha=0.6)
plt.title('Relación entre [Variable 1] y [Variable 2]')
plt.tight_layout()
plt.show()

# Gráfica 5: Gráfica de pastel
plt.figure(figsize=(8, 8))
datos_trabajo['variable_categorica'].value_counts().plot(kind='pie',
autopct='%1.1f%%')
plt.title('Proporción de [Nombre Variable]')
plt.ylabel('')
```

```
plt.tight_layout()
plt.show()
```

✅ IMPORTANTE: Después de cada gráfica, escribir un análisis de 2-3 oraciones explicando:

- Qué muestra la gráfica
- Qué patrones o tendencias se observan
- Qué implicaciones tiene para el análisis

PASO 6: CONVERSIÓN DE VARIABLES A NÚMERO

Código:

```
# Identificar columnas categóricas
columnas_categoricas =
datos_trabajo.select_dtypes(include=['object']).columns
print(f"Columnas categóricas: {list(columnas_categoricas)}")

# Aplicar LabelEncoder
le = LabelEncoder()
for col in columnas_categoricas:
  datos_trabajo[col] = le.fit_transform(datos_trabajo[col])

# Verificar conversión
print("\nTipos de datos después de conversión:")
print(datos_trabajo.dtypes)
```

✅ Nota: Si tienen muchas categorías, consideren usar One-Hot Encoding en lugar de LabelEncoder.

PASO 7: NORMALIZACIÓN

Código:

```
# Ver estadísticos antes de normalizar
print("=== ANTES DE NORMALIZACIÓN ===")
print(datos_trabajo.describe().T)

# Aplicar MinMaxScaler (escala 0-1)
scaler = MinMaxScaler()
```

```
columnas_numericas = datos_trabajo.columns
datos_trabajo[columnas_numericas] =
scaler.fit_transform(datos_trabajo[columnas_numericas])

# Ver estadísticos después de normalizar
print("\n=== DESPUÉS DE NORMALIZACIÓN ===")
print(datos_trabajo.describe().T)
```

✅ Análisis: Explicar por qué es importante normalizar los datos.

PASO 8: CORRELACIÓN

Código:

```
# Calcular matriz de correlación (Spearman es más robusto)
correlacion = datos_trabajo.corr(method='spearman')

# Visualizar con heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlacion, annot=True, cmap='coolwarm',
        fmt='.2f', linewidths=0.5, center=0)
plt.title('Matriz de Correlación de Spearman', fontsize=16)
plt.tight_layout()
plt.show()

# Identificar correlaciones fuertes
print("\n=== CORRELACIONES MÁS FUERTES ===")
# Obtener pares de correlación (excluyendo diagonal)
correlaciones_fuertes = []
for i in range(len(correlacion.columns)):
  for j in range(i+1, len(correlacion.columns)):
    if abs(correlacion.iloc[i, j]) > 0.5:  # Umbral de 0.5
      correlaciones_fuertes.append((
        correlacion.columns[i],
        correlacion.columns[j],
        correlacion.iloc[i, j]
      ))

for var1, var2, corr in sorted(correlaciones_fuertes, key=lambda x:
abs(x[2]), reverse=True):
  print(f"{var1} <-> {var2}: {corr:.3f}")
```

✅ Análisis: Identificar las 3 correlaciones más fuertes y explicar su significado.

PASO 9: TEST DE NORMALIDAD

Código:

```
# Función para test de normalidad
def test_normalidad(dataframe, alpha=0.05):
  print("=== TEST DE SHAPIRO-WILK ===")
  print(f"Nivel de significancia: {alpha}\n")
```

```
    for col in dataframe.columns:
        if pd.api.types.is_numeric_dtype(dataframe[col]):
            data = dataframe[col].dropna()
            if len(data) >= 3:
                stat, p_value = shapiro(data)
                resultado = "NORMAL" if p_value > alpha else "NO NORMAL"
                print(f"{col}:")
                print(f"  Estadístico W: {stat:.4f}")
                print(f"  P-valor: {p_value:.4f}")
                print(f"  Conclusión: {resultado}\n")

# Aplicar test
test_normalidad(datos_trabajo)
```

✅ Análisis: Explicar qué variables siguen distribución normal y las implicaciones.

PASO 10: PREGUNTA DE INVESTIGACIÓN

Ejemplos de preguntas según el tipo de dataset:

- Dataset de clasificación: "¿Es posible predecir [variable objetivo] con base en [variables predictoras] y con qué precisión?"
- Dataset de regresión: "¿Qué variables tienen mayor influencia en [variable objetivo] y cuál es su impacto?"
- Dataset temporal: "¿Existen patrones estacionales en [variable] que permitan hacer predicciones?"
- Dataset de salud: "¿Qué factores de riesgo están más asociados con [condición médica]?"

✅ Su pregunta debe ser:

- Específica y medible
- Respondible con los datos disponibles
- Relevante para el contexto del dataset

PASO 11: EJERCICIO DE PREDICCIÓN

Código para Regresión:

```
# Definir variable objetivo y predictoras
X = datos_trabajo.drop('variable_objetivo', axis=1)  # Variables
predictoras
y = datos_trabajo['variable_objetivo']  # Variable a predecir

# Dividir en entrenamiento y prueba (80-20)
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.2, random_state=42
)

# Crear y entrenar modelo
modelo = LinearRegression()
modelo.fit(X_train, y_train)

# Realizar predicciones
y_pred = modelo.predict(X_test)

# Evaluar rendimiento
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("=== EVALUACIÓN DEL MODELO ===")
print(f'Error Cuadrático Medio (MSE): {mse:.4f}")
print(f'Raíz del MSE (RMSE): {rmse:.4f}")
print(f'Coeficiente R²: {r2:.4f}")
print(f"\nInterpretación R²: El modelo explica el {r2*100:.2f}% de la
variabilidad")

# Coeficientes del modelo
coeficientes = pd.DataFrame({
   'Variable': X.columns,
   'Coeficiente': modelo.coef_
}).sort_values('Coeficiente', key=abs, ascending=False)

print("\n=== IMPORTANCIA DE VARIABLES ===")
print(coeficientes)

# Visualización: Valores reales vs predichos
plt.figure(figsize=(10, 6))
```

```
plt.scatter(y_test, y_pred, alpha=0.6, color='blue',
label='Predicciones')
plt.plot([y_test.min(), y_test.max()],
      [y_test.min(), y_test.max()],
      'r--', linewidth=2, label='Predicción Perfecta')
plt.xlabel('Valores Reales')
plt.ylabel('Valores Predichos')
plt.title('Rendimiento del Modelo de Regresión')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

Código alternativo para Clasificación:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Preparar datos (igual que antes)
X = datos_trabajo.drop('variable_objetivo', axis=1)
y = datos_trabajo['variable_objetivo']
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.2, random_state=42
)

# Modelo de clasificación
modelo = LogisticRegression(max_iter=1000)
modelo.fit(X_train, y_train)
y_pred = modelo.predict(X_test)

# Evaluación
accuracy = accuracy_score(y_test, y_pred)
print(f"Precisión del modelo: {accuracy*100:.2f}%")
print("\nReporte de clasificación:")
print(classification_report(y_test, y_pred))

# Matriz de confusión
plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('Matriz de Confusión')
plt.ylabel('Valor Real')
plt.xlabel('Valor Predicho')
plt.tight_layout()
plt.show()
```

✅ Análisis final requerido:

- ¿Qué tan bueno es el modelo? (interpretar métricas)
- ¿Qué variables son más importantes?
- ¿Responde satisfactoriamente la pregunta de investigación?
- ¿Qué limitaciones tiene el modelo?

ESTRUCTURA DE CONCLUSIONES

Las conclusiones deben incluir:

1. Resumen del análisis: Breve descripción del dataset y los 11 pasos realizados

2. Hallazgos principales: Patrones, tendencias y correlaciones más importantes encontradas

3. Respuesta a la pregunta de investigación: Respuesta clara y fundamentada con los resultados del modelo

4. Interpretación de resultados: Qué significan los resultados en el contexto del problema

5. Limitaciones: Restricciones del dataset, del análisis o del modelo

6. Recomendaciones: Sugerencias para análisis futuros o mejoras del modelo

TIPS PARA UNA EXCELENTE PRESENTACIÓN

💡 Practiquen la presentación varias veces antes del día de exposición

💡 Cada integrante debe dominar una sección específica

💡 Usen visualizaciones claras y de alta calidad

💡 Eviten saturar los slides con texto; usen viñetas

💡 Preparen respuestas para posibles preguntas del profesor

💡 Lleguen 10 minutos antes para probar el equipo

💡 Mantengan contacto visual con la audiencia

💡 Hablen con claridad y a un ritmo moderado

CHECKLIST ANTES DE ENTREGAR

☐ Código completo con los 11 pasos implementados

☐ Comentarios explicativos en cada sección del código

☐ Mínimo 5 visualizaciones con sus respectivos análisis

☐ Pregunta de investigación claramente formulada

☐ Modelo predictivo implementado y evaluado

☐ Presentación PowerPoint con exactamente 8 slides

☐ Documento de conclusiones (1-2 páginas)

☐ Todos los archivos nombrados correctamente

☐ Revisión ortográfica y gramatical

☐ Prueba de que el código se ejecuta sin errores