Esquema de pasos para un modelo de Machine Learning Supervisado

1️⃣ Cargar y explorar el dataset

- `df.head()`, `df.info()`, `df.describe()`
- `df.dtypes` para revisar tipos de variables
- `df.isnull().sum()` para detectar nulos
- `df.nunique()` para ver cardinalidad

2️⃣ Análisis exploratorio (EDA)

- 🔍 **Distribuciones**: `histplot`, `boxplot`, `df[col].value_counts()`
- 📊 **Correlación con la target**: `df.corr()['target']`, `heatmap`
- **Multicolinealidad**: revisar variables muy correladas entre sí
- 📌 ¿Qué variables parecen más relevantes? ¿Qué outliers aparecen? ¿Hay relaciones lineales?

3️⃣ Limpieza de datos

🔹 **a. Valores nulos**
- Eliminar columnas o filas (`dropna`) si hay muchos nulos
- Rellenar con:
  - Media / mediana (numéricas)
  - Moda (categóricas)
  - O usar `SimpleImputer`

🔹 **b. Outliers**
- Visualizar con `boxplot`, `describe()`, o `z-score`
  Tratar según caso: eliminar, limitar, o transformar

4️⃣ Tratamiento de variables categóricas

🔹 **a. Codificación**
- `pd.get_dummies()` (One-Hot Encoding)
- `LabelEncoder` para variables ordinales
- `OrdinalEncoder` si hay jerarquía

🔹 **b. Binning (agrupar categorías)**

- Reunir categorías poco frecuentes
- Usar `map()` o `replace()` para simplificar
- Útil si hay muchas clases con pocos ejemplos

5️⃣ Transformación de variables

- Escalado con `StandardScaler`, `MinMaxScaler` (imprescindible para SVM, KNN, KMeans...)
- Transformaciones estadísticas (log, box-cox...) si hay sesgo fuerte

6️⃣ Feature Engineering (si aplica)

- Crear nuevas variables a partir de otras
- Por ejemplo: ratios, diferencias, etc.

7️⃣ Selección de variables

- Eliminar columnas poco informativas o duplicadas
- Usar correlación (`df.corr()`), `SelectKBest`, `RFE` o `feature_importances_` (de modelos)

8️⃣ Separar variables X / y

```
X = df.drop('target', axis=1)
y = df['target']
```

9️⃣ Dividir en entrenamiento y prueba

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

🔟 Entrenar modelo

```
from sklearn.linear_model import LinearRegression, LogisticRegression
modelo = LinearRegression()  # o cualquier otro
modelo.fit(X_train, y_train)
```

1️⃣1️⃣ Predecir y evaluar (MÉTRICAS)

```
y_pred = modelo.predict(X_test)
```

- **a. Regresión**
  - `mean_squared_error`, `mean_absolute_error`, `r2_score`

- **b. Clasificación**
  - `accuracy_score`, `precision`, `recall`, `f1_score`
  - `confusion_matrix`, `classification_report`

1️⃣2️⃣ Visualización de resultados

- `scatter`, `residual plot`, `matriz de confusión`, `ROC curve`, `feature importances`

1️⃣3️⃣ Conclusiones

- Interpretar métricas
- Qué funcionó y qué no
- Posibles mejoras

Y probar de predecir sobre valores aleatorios.