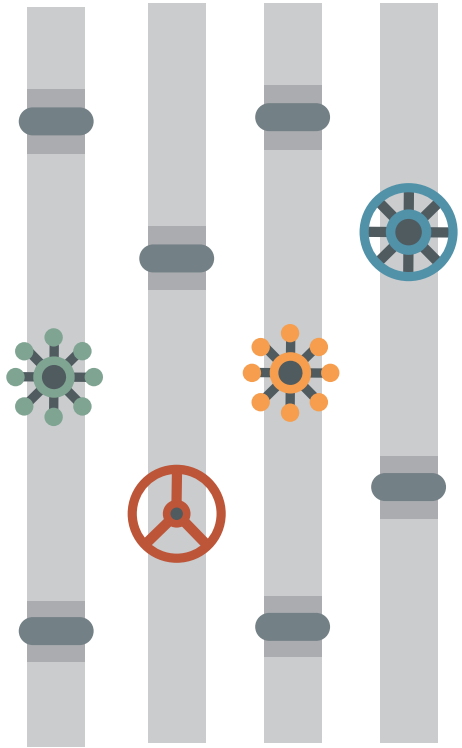
Team Challenge:

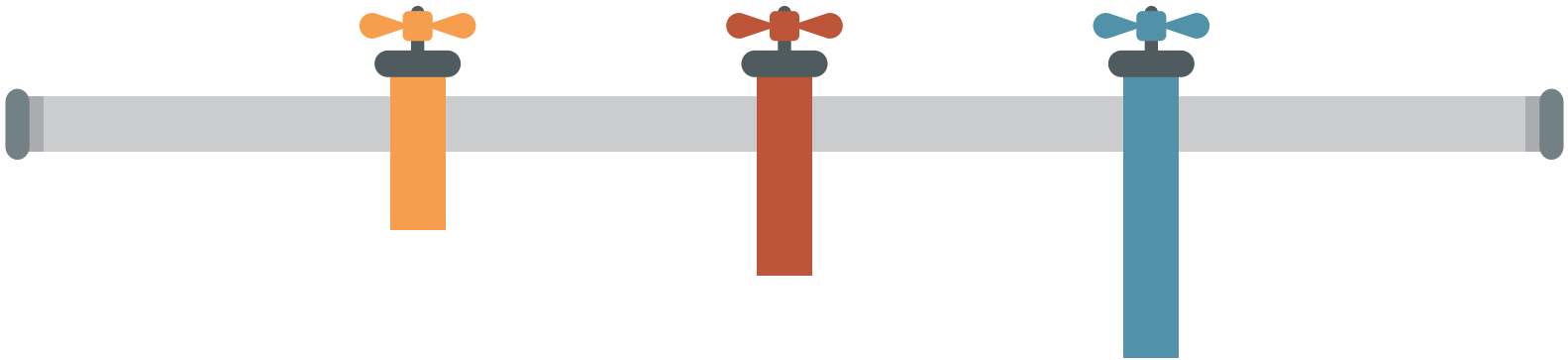
Pipelines Sprint 17

Plumbers Enterprise

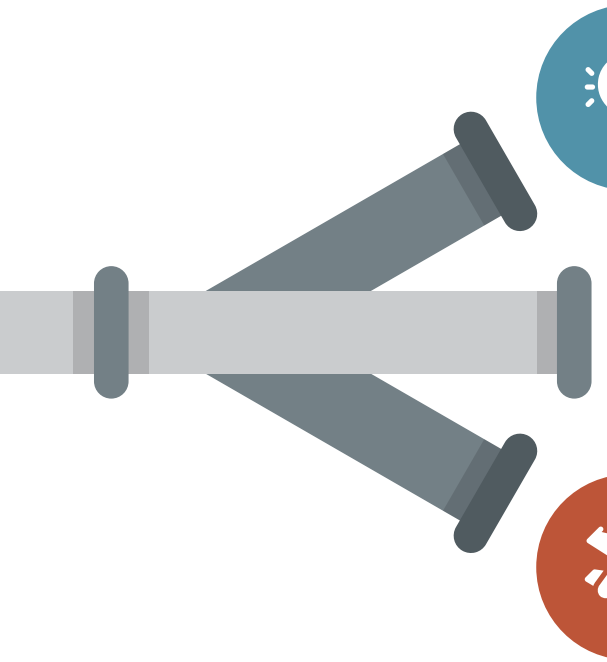
Álvaro Sánchez, Juan Moreno, Xián Mosquera,
Vicky Sequeira y Dani Castillo



Índice



Objetivo



Meta principal

A

Aplicar Machine Learning para segmentar clientes en grupos homogéneos mediante **clustering**.

Enfoque

B

Uso de **Pipelines** para optimizar el flujo de datos, transformación y modelado.

Resultados esperados

C

Identificar patrones de comportamiento en los clientes y **mejorar** estrategias de negocio.

¿De dónde partimos?

```
Información del dataset de entrenamiento:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8068 entries, 0 to 8067
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               8068 non-null   int64
 1   Gender           8068 non-null   object
 2   Ever_Married     7928 non-null   object
 3   Age              8068 non-null   int64
 4   Graduated        7990 non-null   object
 5   Profession       7944 non-null   object
 6   Work_Experience  7239 non-null   float64
 7   Spending_Score   8068 non-null   object
 8   Family_Size      7733 non-null   float64
 9   Var_1            7992 non-null   object
 10  Segmentation     8068 non-null   object
dtypes: float64(2), int64(2), object(7)
memory usage: 693.5+ KB
None
```

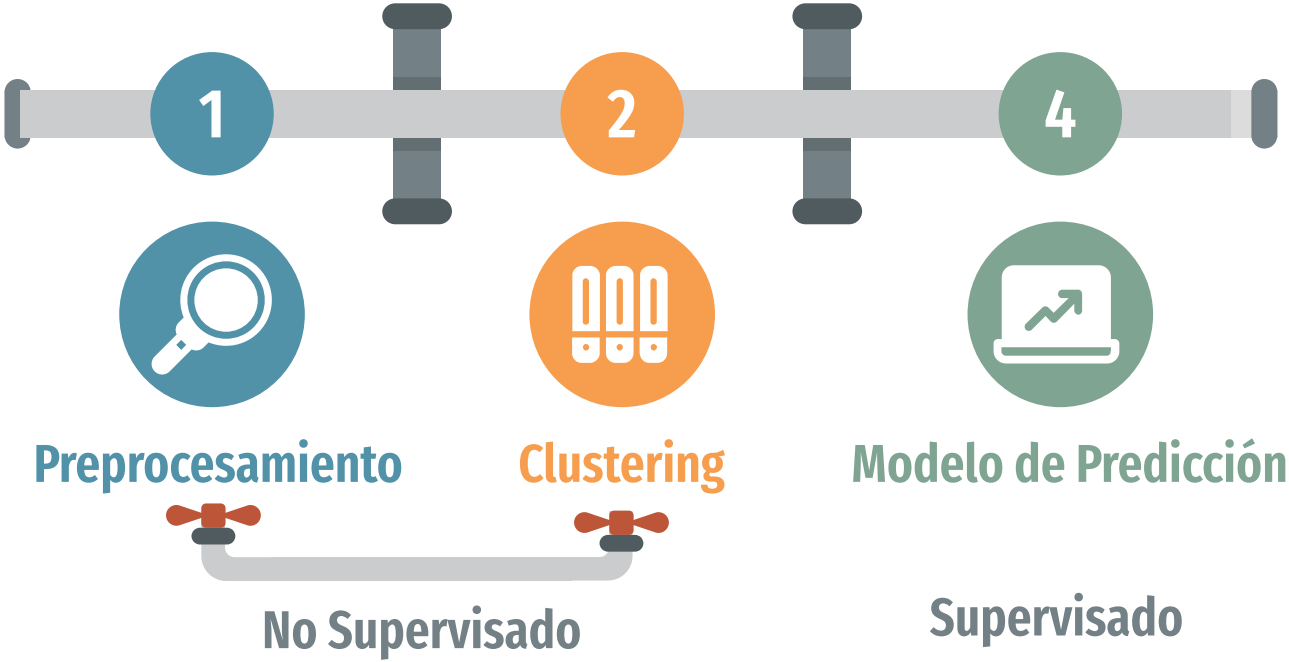
Valores Nulos

Variables no númericas

Target

	ID	Gender	Ever_Married	Age	Graduated	Profession	Work_Experience	Spending_Score	Family_Size	Var_1	Segmentation
0	462809	Male	No	22	No	Healthcare	1.0	Low	4.0	Cat_4	D
1	462643	Female	Yes	38	Yes	Engineer	NaN	Average	3.0	Cat_4	A
2	466315	Female	Yes	67	Yes	Engineer	1.0	Low	1.0	Cat_6	B
3	461735	Male	Yes	67	Yes	Lawyer	0.0	High	2.0	Cat_6	B
4	462669	Female	Yes	40	Yes	Entertainment	NaN	High	6.0	Cat_6	A

Usos de Pipelines



Usos de Pipelines

Preprocesamiento

Limpiar y preparar los datos
antes de entrenar modelos d
Machine Learning

Pasos del preprocesamiento en el pipeline:

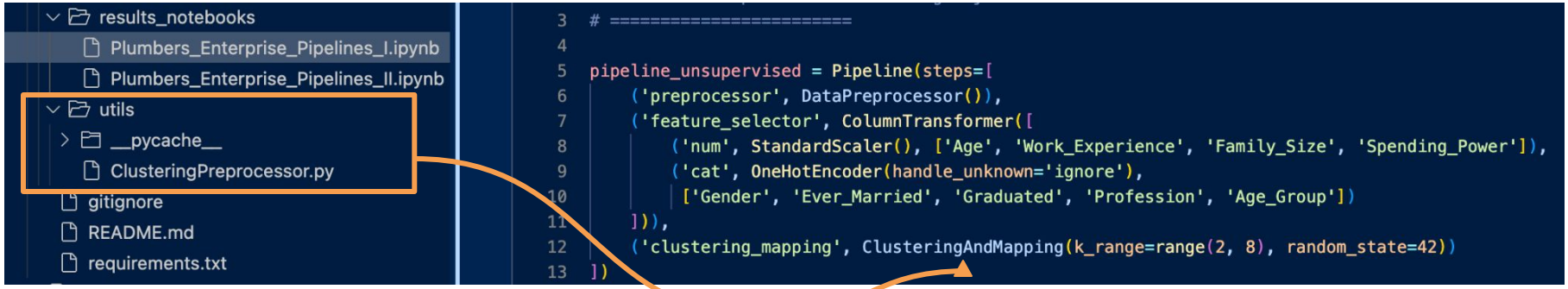
1. **Manejo de valores faltantes**
   - Para variables numéricas: SimpleImputer(strategy="median")
   - Para variables categóricas: SimpleImputer(strategy="most_frequent")
2. **Estandarización de características numéricas**
   - Se aplica StandardScaler() para que todas las variables numéricas tienen media 0 y varianza 1.
3. **Codificación de variables categóricas**
   - Se usa OneHotEncoder(handle_unknown="ignore") para transformar categorías en variables binarias.

Técnicas utilizadas en el pipeline:

- Se usó ColumnTransformer para elegir las variables más importantes:

```
preprocessor = ColumnTransformer([
    ("num", Pipeline([
        ("imputer", SimpleImputer(strategy="median")),  # Imputar valores nulos con la mediana
        ("scaler", StandardScaler())  # Escalar variables numéricas
    ]), numeric_features),

    ("cat", Pipeline([
        ("imputer", SimpleImputer(strategy="most_frequent")),  # Imputar valores nulos con el valor más frecuente
        ("encoder", OneHotEncoder(handle_unknown="ignore"))  # Codificar variables categóricas
    ]), categorical_features)
])
```

Usos de Pipelines



2

Clustering

Encontrar el número de clusters óptimo para la mejor segmentación de clientes.

ClusteringAndMapping()

Este transformador personalizado se encarga de:

Escalar los datos para mejorar la agrupación.
Determinar automáticamente el número óptimo de clusters.
Evaluar la calidad del clustering con métricas como **Silhouette Score y Davies-Bouldin Score**.
Asignar etiquetas a los clusters en función de las características más representativas.
Realizar un análisis detallado de los clusters, incluyendo:

- Tamaño del grupo.
- Edad promedio.
- Distribución de gasto.
- Profesiones más comunes.
- Proporción de clientes casados y graduados.

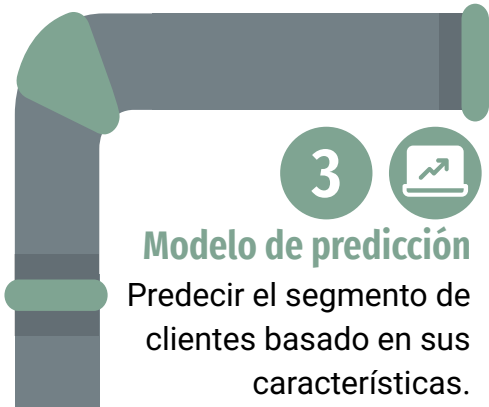
Usos de Pipelines

```python
param_grids = {
    "RandomForest": {
        "classifier__n_estimators": [100, 200],
        "classifier__max_depth": [10, 20, None]
    },
    "XGBoost": {
        "classifier__learning_rate": [0.01, 0.1],
        "classifier__n_estimators": [100, 200],
        "classifier__max_depth": [3, 6]
    },
    "LightGBM": {
        "classifier__learning_rate": [0.01, 0.1],
        "classifier__n_estimators": [100, 200],
        "classifier__num_leaves": [31, 50]
    }
}

models = {
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric="logloss", random_state=42),
    "LightGBM": LGBMClassifier(random_state=42)
}

best_models = {}
```

```python
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', model)
])

grid_search = GridSearchCV(pipeline,
                           param_grids[model_name],
                           cv=5,
                           scoring="accuracy",
                           n_jobs=-1)

grid_search.fit(X_train, y_train)

best_models[model_name] = grid_search.best_estimator_
```

Componentes del pipeline:

1. **Preprocesamiento:** (similar a la parte no supervisada)
   - **Imputación de valores nulos** con SimpleImputer.
   - **Estandarización y codificación** (StandardScaler y OneHotEncoder)
   - **Selección de características** con ColumnTransformer.
2. **Modelos supervisados entrenados dentro del pipeline:**
   - **Clasificación:** RandomForestClassifier, LogisticRegression.
   - **Regresión:** GradientBoostingRegressor.
3. **Optimización del modelo:**
   - Búsqueda de hiperparámetros con GridSearchCV.
   - Validación cruzada con cross_val_score.



3

Modelo de predicción

Predecir el segmento de clientes basado en sus características.

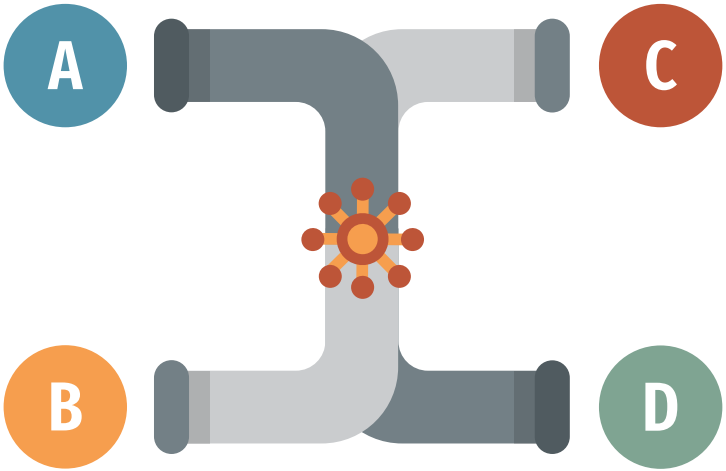
Ventajas

Automatización del flujo de trabajo

Reducción errores humanos y hace el proceso más eficiente

Reproducibilidad

Se pueden ejecutar los mismos pasos con diferentes datos sin rehacer el código.



Modularidad

Permite agregar, modificar o eliminar pasos sin afectar todo el código.

Optimización del tiempo

Se evitan tareas repetitivas y se mejora la velocidad de ejecución.

FIN