

# **DATA PREPARATION FOR MACHINE LEARNING IN DATABRICKS**

# Databricks for data preparation

Об'єднана платформа для даних та ML

Delta Lake: забезпечує надійне зберігання даних, контроль версій

Autoloader: автоматично завантажує потокові дані.

Feature Store: для зберігання й повторного використання ML-фіч з повною історією змін.

MLflow: відстеження експериментів, моделей та метрик

# Архітектура пайплайну (Medallion)

Для підготовки даних використовується medallion архітектура:

Raw → Bronze → Silver → Gold ( ML ready)

# Підготовка даних

Основні кроки:

- Очищення даних (виявлення та виправлення помилок, видалення дублікатів, обробка пропусків)
- Імпутація даних
- Стандартизація даних (приведення всіх ознак до однакової шкали, масштабування числових змінних)
- Оптимізація точності (налаштування балансу класів, вибір метрик оцінювання, адаптація датасету під конкретну ML-задачу)
- Feature Engineering (створення нових ознак або модифікація існуючих)

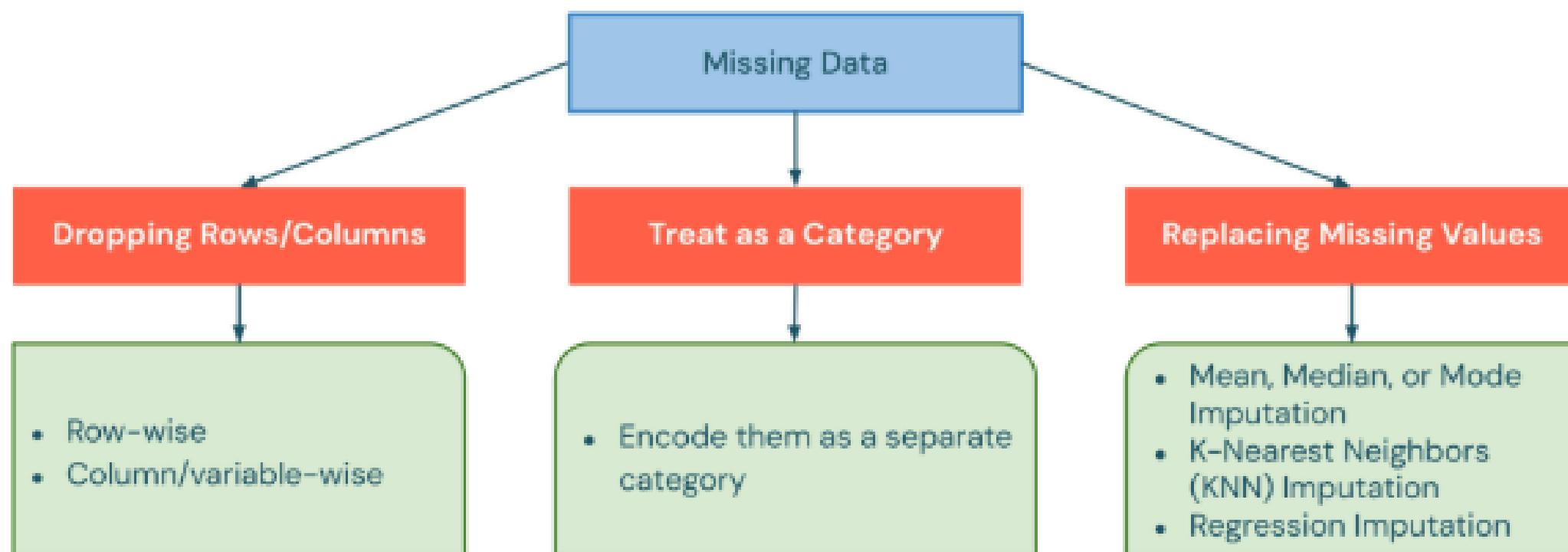
# Data imputation

data imputation - процес заповнення пропущених даних, для того щоб покращити якість датасету.

Проблеми пропущених даних:

## How to Handle Missing Data

Data imputation methods



- Знижена продуктивність моделі
- Упереджені висновки
- Незбалансовані представлення
- Підвищена складність в обробці моделі

# Replacing Missing Values

Data imputation methods

## Mean - Mode Imputation

Before	After
10	10.0
15	15.0
-	18.3
20	20.0
25	25.0

## K-Nearest Neighbors (KNN with k=2)

Before	After
8	8.0
-	10.0
12	12.0
15	15.0
-	13.0

## Multiple Imputation (Regression)

F1	F2	Before	After
X	X	10	10.0
X	X	15	15.0
X	$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$	20	Y
X	X	25	25.0

# Data imputation

## Factors Influencing Imputation Method

There isn't a definitive best approach

### Nature of Data

- Is data type is continuous, ordinal or categorical?
- Data distribution: For example, median imp. Might work better for non-normal distributions

### Amount of Missing Data

- How much of data is missing?
- Will imputing data improve the quality of dataset or will it add more bias?
- Multiple imp. and KNN might better for large missingness.

### Form of Missingness

- Is data is missing at random or is there a missingness pattern.
- Domain knowledge and how would imputation affect downstream analysis.

# Data encoding

Encoding - перетворення категоріальних або нечислових даних у числові представлення, які може обробляти ML-модель

- One-Hot encoding: гожна категорія - окремий бінарний стовпець Переваги: не створює штучного порядку між категоріями Недоліки: збільшує розмірність даних .
- Label encoding: кожній категорії унікальне число Переваги: Компактне представлення. Недоліки: Створює штучний порядок.
- Ordinal encoding: числові значення на основі природного порядку категорій. Використання: Коли категорії мають природний порядок (низький/середній/високий, оцінки тощо)

## Working with Categorical Features

### Encoding Categories

#### Issue: String types

Models require numerical input, and categorical variables need to be encoded.

### Possible Solutions:

#### One-Hot Encoding

ID	Category	ID	Cat_A	Cat_B	Cat_C
0	A	0	1	0	0
1	B	1	0	1	0
2	A	2	1	0	0
3	C	3	0	0	1
4	A	4	1	0	0

#### Label Encoding:

ID	Category	Label
0	A	0
1	B	1
2	A	0
3	C	2
4	A	0

#### Ordinal Encoding:

ID	Category	Ordinal
0	A	0.0
1	B	1.0
2	A	0.0
3	C	2.0
4	A	0.0

# Data encoding

## Working with Categorical Features

### High Cardinality

#### Issue: Large set of categories

Many unique values in a categorical feature can lead to a large number of dummy variables, increasing dimensionality and potentially causing issues. This is known as the **high-cardinality problem**.

#### Possible Solutions:

##### Group Rare Categories:

ID	Category
0	A
1	B
2	A
3	C
4	B
5	D
6	D
7	B
8	D
9	D

ID	Category	Cat_Group
0	A	A
1	B	Rare
2	A	A
3	C	Rare
4	B	B
5	D	D
6	D	D
7	B	B
8	D	D
9	D	D

##### Top-N Categories:

ID	Category
0	A
1	B
2	A
3	C
4	B
5	D
6	D
7	B
8	D
9	D

ID	Category	Cat_Group
0	A	A
1	B	Other
2	A	A
3	C	Other
4	B	B
5	D	D
6	D	D
7	B	B
8	D	D
9	D	D

# Embeddings

- Embeddings - числові представлення даних, що фіксують семантичні зв'язки
- На відміну від традиційного кодування, embeddings розміщують схожі елементи більше у низьковимірному просторі.
- Перетворюють високорозмірні категоріальні/текстові дані у компактний, щільний векторний простір
- Ці представлення фіксують зв'язки та контекст між різними об'єктами

## Embeddings vs. One-Hot Encoding

Choosing the Right Encoding for Categorical Data

Feature	One-Hot Encoding	Embeddings
Representation	Sparse binary vectors (high-dimensional)	Dense numerical vectors (low-dimensional)
Semantic Relationships	Does not capture relationships between categories	Places similar categories closer in vector space
Scalability	Inefficient for high-cardinality data	Scales well with large category sets
Efficiency	Inefficient for high-cardinality features (sparse matrix)	Efficient for high-cardinality features
Model Suitability	Suitable for simple models like Decision Trees	Best for Neural Networks and deep learning models
Example:	$\text{TV} \rightarrow [1, 0, 0, 0]$ $\text{Laptop} \rightarrow [0, 1, 0, 0]$ $\text{Phone} \rightarrow [0, 0, 1, 0]$	$\text{TV} \rightarrow [0.6, 1.2, -0.8]$ $\text{Laptop} \rightarrow [0.5, 1.1, -0.7]$ $\text{Phone} \rightarrow [0.2, -0.4, 1.5]$

# Data standarization

Стандартизація даних - це процес створення стандартів та перетворення даних з різних джерел у узгоджений формат.

Переваги стандартизації даних:

- Легше розуміти та порівнювати ознаки
- Жодна ознака не домінує через масштаб
- Алгоритми навчаються швидше та стабільніше

# Feature Store

Feature Store управляє ознаками або вхідними даними для моделі машинного навчання.

Types of Features:

- Transformed Features (закодовані категорії)
- Counted Features (інформація про день тижня)
- Feature Augmentation (зовнішні дані, наприклад, погода)

**Databricks Feature Store:**

- Інтеграція з іншими компонентами Databricks.
- UI Feature Store, доступний з workspace Databricks, дозволяє переглядати та шукати існуючі ознаки.
- Коли створюються feature table, джерела даних зберігаються та доступні.
- Коли кілька команд керують обчисленням ознак і ML-моделями в production, навіть незначний перекіс у вхідних даних може бути важко виявити та виправити.

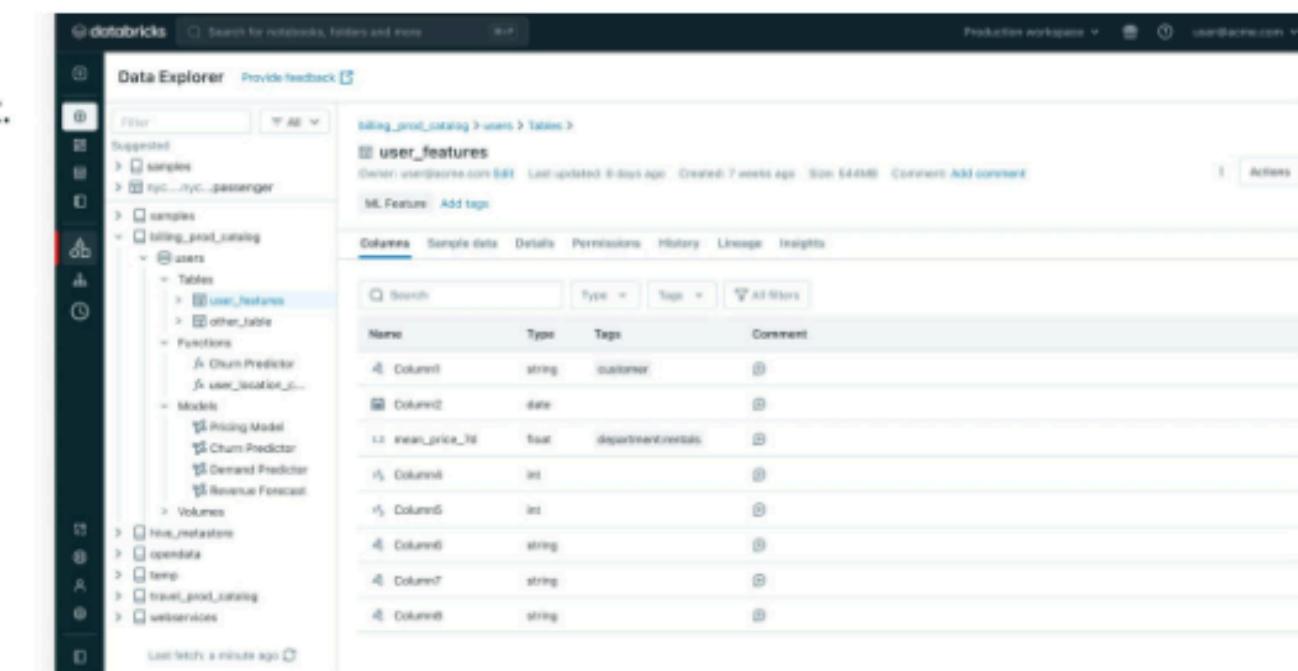
# Feature Store

Feature Store повністю інтегрується з Unity Catalog, тому будь-яка таблиця Unity Catalog може використовуватися як feature table. Feature tables поводяться як звичайні таблиці Unity Catalog, але включають додаткові метадані для управління ознаками.

## Complete Integration-FS with **Unity Catalog**

Any table can be a feature table

- Feature Tables become regular UC Tables with additional metadata.
- Shared properties are unified.
  - Feature table description == table comment.
  - Feature table schema == table schema
- Three-level namespace convention



Name	Type	Tags	Comment
Column1	string	customer	
Column2	date		
mean_price_7d	float	department_level	
Column4	int		
Column5	int		
Column6	string		
Column7	string		
Column8	string		

# Датасет: Amazon Product Reviews

датасет з /databricks-datasets/amazon/test4K

- review: Текст відгуку (Unstructured text).
- rating: Оцінка (Target variable)
- product\_category: Категорія (Electronics, Books...).
- price: Ціна

# Завантаження даних

```
raw_df = spark.read.parquet("/databricks-datasets/amazon/test4K")

bronze_df = raw_df \
    .withColumn("category", cat_udf()) \
    .withColumn("price", price_udf())

display(bronze_df)
```

# Cleaning and Imputation

- Drop Duplicates: Видалення повторних відгуків
- Empty Reviews: Фільтрація рядків без тексту.
- Missing Price: Використання Imputer (mean) для заповнення пропусків.

```
silver_filtered = bronze_df \
    .filter(F.length("review") > 0) \
    .dropDuplicates(["review_body"])

imputer = Imputer(
    inputCols=["price"],
    outputCols=["price_imputed"]
).setStrategy("mean")

silver_df = imputer.fit(silver_filtered).transform(silver_filtered)
```

# Encoding Categorical Data

Перетворення категорій у числовий формат:

Electronics -> [1.0, 0.0, 0.0]

Books -> [0.0, 1.0, 0.0]

Home -> [0.0, 0.0, 1.0]

```
indexer = StringIndexer(inputCol="category", outputCol="category_idx")
encoder = OneHotEncoder(inputCols=["category_idx"], outputCols=["category_vec"])

assembler = VectorAssembler(inputCols=["price_imputed"], outputCol="price_vec")
scaler = StandardScaler(inputCol="price_vec", outputCol="price_scaled", withStd=True, withMean=True)

pipeline = Pipeline(stages=[indexer, encoder, assembler, scaler])
pipeline_model = pipeline.fit(silver_df)
gold_df = pipeline_model.transform(silver_df)

display(gold_df.select("category", "category_vec", "price_imputed", "price_scaled"))

>  See performance (1)
```

# Standardization (scaling)

Масштабування числових ознак.

Ціна (наприклад, 200.0) значно більша за інші ознаки (0 або 1).

Це може викривити модель.

Приведення даних до: Mean = 0, Std Dev = 1.

```
assembler = VectorAssembler(inputCols=["price_imputed"], outputCol="price_vec")
scaler = StandardScaler(inputCol="price_vec", outputCol="price_scaled", withStd=True, withMean=True)

pipeline = Pipeline(stages=[indexer, encoder, assembler, scaler])
pipeline_model = pipeline.fit(silver_df)
gold_df = pipeline_model.transform(silver_df)

display(gold_df.select("category", "category_vec", "price_imputed", "price_scaled"))
"
```

# Text Embeddings

Для закодування текстів відгуків: Transformer models.

Модель: all-MiniLM-L6-v2 (HuggingFace).

Щільний вектор для кожного відгуку.

Дозволяє моделі розуміти сенс.

```
pdf = gold_df.select("product_id", "review_body").toPandas()

model = SentenceTransformer('all-MiniLM-L6-v2')

pdf['review_embedding'] = pdf['review_body'].apply(lambda x: model.encode(x).tolist())

embedding_schema = StructType([
    StructField("product_id", StringType(), True),
    StructField("review_body", StringType(), True),
    StructField("review_embedding", ArrayType(FloatType()), True)
])

embeddings_spark_df = spark.createDataFrame(pdf[["product_id", "review_embedding"]], schema="product_id string, review_embedding array<float>")

gold_df_with_embeddings = gold_df.join(embeddings_spark_df, on="product_id", how="inner")
display(gold_df_with_embeddings.select("review_body", "review_embedding"))
```

# Feature Store (simulated)

Збереження ознак для повторного використання.

Збереження готових векторів та скейлерів у Delta таблицю. Це дозволяє іншим моделям брати ці дані без повторних обчислень.

```
table_name = "amazon_features"

feature_table = gold_df_with_embeddings.select(
    "product_id",
    "timestamp",
    "price_scaled",
    "category_vec",
    "review_embedding",
    "star_rating"
)

feature_table.write.format("delta").mode("overwrite").saveAsTable(table_name)
print(f"Features saved to table: {table_name}")
```

# Training set creation

## Підготовка фінального набору для навчання

```
features_df = spark.table(table_name)
```

```
training_df = features_df.select(  
    "product_id",  
    "star_rating",  
    "price_scaled",  
    "category_vec",  
    "review_embedding"  
)
```

```
display(training_df)
```

	product_id	star_rating	price_scaled	category_vec
1	3676018b-5c3e-4224-a849-08372d6ef788	5	> {"type":1,"size":null,"indices":null,"values":[-0.05051375709697...]	> {"type":0,"size":4,"indices":[]}
2	39451006-5baf-4800-850c-a5dbe821b37f	5	> {"type":1,"size":null,"indices":null,"values":["1.498681351241177...]	> {"type":0,"size":4,"indices":[]}
3	5f8f22b7-2a10-4056-8bb3-51291941aa23	4	> {"type":1,"size":null,"indices":null,"values":[-1.02910174176834...]	> {"type":0,"size":4,"indices":[]}
4	04b2a2ed-328b-4983-a3d1-6184962154...	1	> {"type":1,"size":null,"indices":null,"values":[-1.06377682420659...]	> {"type":0,"size":4,"indices":[]}
5	02628f8c-6593-4097-9098-91e122edc507	4	> {"type":1,"size":null,"indices":null,"values":["1.613960791978871...]	> {"type":0,"size":4,"indices":[]}
6	f09a7fa6-21ef-4b7e-a4e4-df63e29cd344	2	> {"type":1,"size":null,"indices":null,"values":["1.305230891322513...]	> {"type":0,"size":4,"indices":[]}
7	1ee234c1-54db-48d7-9203-51eb0d2056f8	1	> {"type":1,"size":null,"indices":null,"values":[-1.04142052105561...]	> {"type":0,"size":4,"indices":[]}
8	bb4cbf0e-f4ed-469f-9741-7e4ce12a9dae	5	> {"type":1,"size":null,"indices":null,"values":["0.793621341663405...]	> {"type":0,"size":4,"indices":[]}
9	c91fab74-b703-4d88-a0ec-5e1d16f7de8a	5	> {"type":1,"size":null,"indices":null,"values":[-0.05051375709697...]	> {"type":0,"size":4,"indices":[]}
10	eed9fc8c-a0ea-4144-848e-c39c78e8af2	4	> {"type":1,"size":null,"indices":null,"values":["0.139053118793964...]	> {"type":0,"size":4,"indices":[]}
11	e6111f44-d382-4895-b9de-d1c8a7556ce8	1	> {"type":1,"size":null,"indices":null,"values":["0.094568638034365...]	> {"type":0,"size":4,"indices":[]}
12	3d009919-a4d2-4611-ad20-707101043c55	4	> {"type":1,"size":null,"indices":null,"values":[-0.05051375709697...]	> {"type":0,"size":4,"indices":[]}

**Thank you**