

Data Cleaning (Missing Values, Duplicates, Types)

◆ Reminder

- Yesterday: we learned how to **import & export data** (CSV, Excel, JSON).
 - Today: imported datasets are often **messy** → we must **clean before analysis**.
-

◆ Why Cleaning Matters

Notes

- **Garbage in = garbage out** → ML models and visualizations are only as good as the data.
- Real-world issues:
 - Missing values → NaN (Not a Number).
 - Duplicates → repeated rows.
 - Wrong data types → e.g., numbers stored as text.

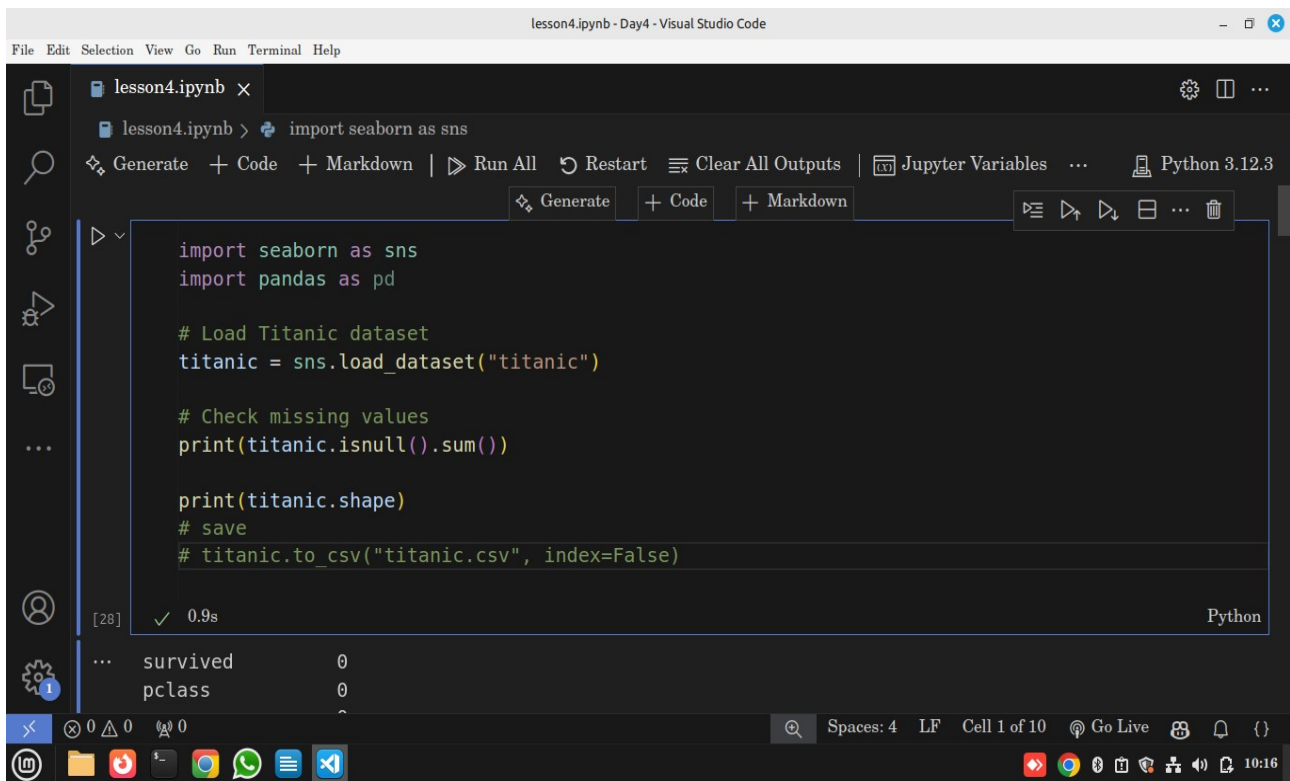
Use in Data Science

- Data cleaning can take **50–80% of project time**.
 - A well-cleaned dataset improves **accuracy, reliability, and reproducibility**.
-

◆ Handling Missing Values

Practical

Check for missing values



The screenshot shows a Jupyter Notebook in Visual Studio Code. The code in the cell is as follows:

```
import seaborn as sns
import pandas as pd

# Load Titanic dataset
titanic = sns.load_dataset("titanic")

# Check missing values
print(titanic.isnull().sum())

print(titanic.shape)
# save
# titanic.to_csv("titanic.csv", index=False)
```

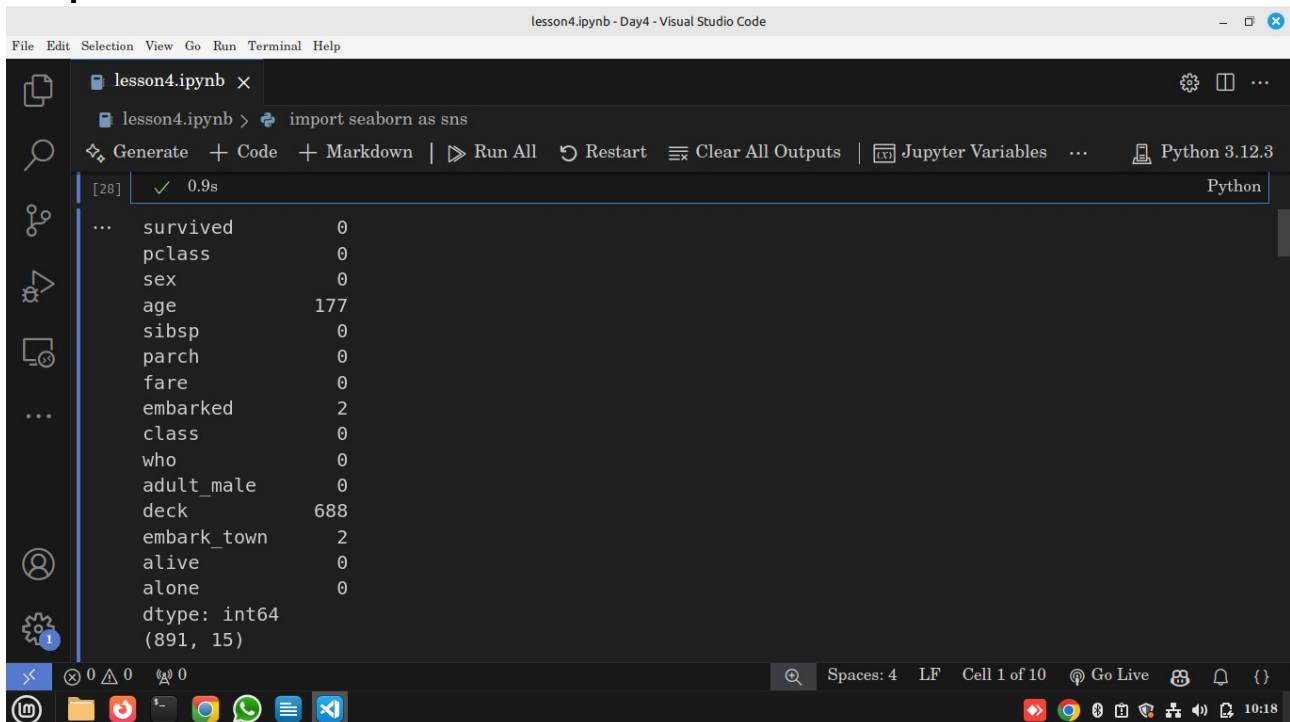
The output of the code is shown below the cell:

```
[28] ✓ 0.9s
```

The output shows the sum of missing values for each column:

```
survived    0
pclass      0
```

Output



The screenshot shows the same Jupyter Notebook in Visual Studio Code, but now the output of the code is visible. The output is as follows:

```
[28] ✓ 0.9s
```

The output shows the sum of missing values for each column:

```
survived    0
pclass      0
sex          0
age         177
sibsp       0
parch       0
fare        0
embarked    2
class       0
who         0
adult_male  0
deck       688
embark_town 2
alive       0
alone       0
dtype: int64
(891, 15)
```

- We have some missing values in age ,deck, and embark_town
- we have a total of 891 rows and 15 columns

Option 1: Drop the empty rows

The screenshot shows a Jupyter Notebook in Visual Studio Code. The first cell contains the following code:

```
import seaborn as sns
adult_male      0
deck            688
embark_town      2
alive           0
alone           0
dtype: int64
(891, 15)
```

The second cell contains the following code:

```
# Drop rows with missing values
titanic_drop = titanic.dropna()
print("After drop:", titanic_drop.shape)
```

The output of the second cell is:

```
[2] ✓ 0.0s Python
... After drop: (182, 15)
```

Option2 : Fill the missing values with Mean or Most common used value (mode)

The screenshot shows a Jupyter Notebook in Visual Studio Code. The first cell contains the same code as the previous screenshot:

```
import seaborn as sns
adult_male      0
deck            688
embark_town      2
alive           0
alone           0
dtype: int64
(891, 15)
```

The second cell contains the following code:

```
# Fill missing age with mean
titanic_fill = titanic.copy()
titanic_fill["age"] = titanic_fill["age"].fillna(titanic["age"].mean())

# Fill missing embarked with most common value
titanic_fill["embarked"] = titanic_fill["embarked"].fillna(titanic["embarked"].mode()[0])
print(titanic_fill.shape)
# Check missing values
```

The output of the second cell is:

```
[15] ✓ 0.0s Python
... (891, 15)
```

Explanation

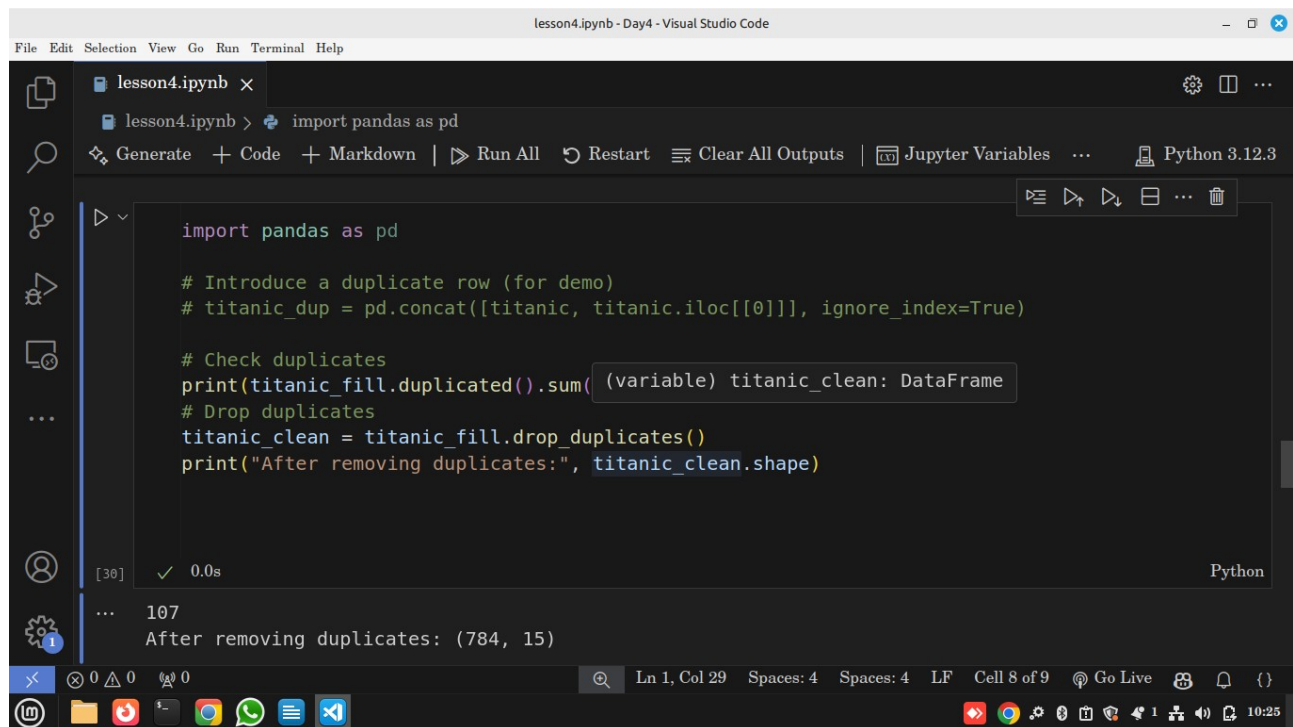
- `.isnull().sum()` → counts missing values per column.
- `.dropna()` → removes rows with NaN.
- `.fillna()` → replaces missing values.
- `.mode()[0]` → fills with the most frequent category.

Use in Data Science

- Dropping vs filling depends on dataset size.
 - Filling prevents data loss and makes models work.
-

◆ Handling Duplicates

Practical



```
import pandas as pd

# Introduce a duplicate row (for demo)
# titanic_dup = pd.concat([titanic, titanic.iloc[[0]]], ignore_index=True)

# Check duplicates
print(titanic_fill.duplicated().sum())
# Drop duplicates
titanic_clean = titanic_fill.drop_duplicates()
print("After removing duplicates:", titanic_clean.shape)
```

The screenshot shows a Jupyter Notebook interface with the following elements:

- File Explorer:** Shows the notebook file 'lesson4.ipynb'.
- Code Editor:** Contains the Python code for handling duplicates. A tooltip for `titanic_clean` shows its type as `DataFrame`.
- Output Console:** Shows the execution result of the `print` statement: `107` and `After removing duplicates: (784, 15)`.
- Status Bar:** Indicates the current cell is 'Cell 8 of 9' and the file is 'Ln 1, Col 29'.

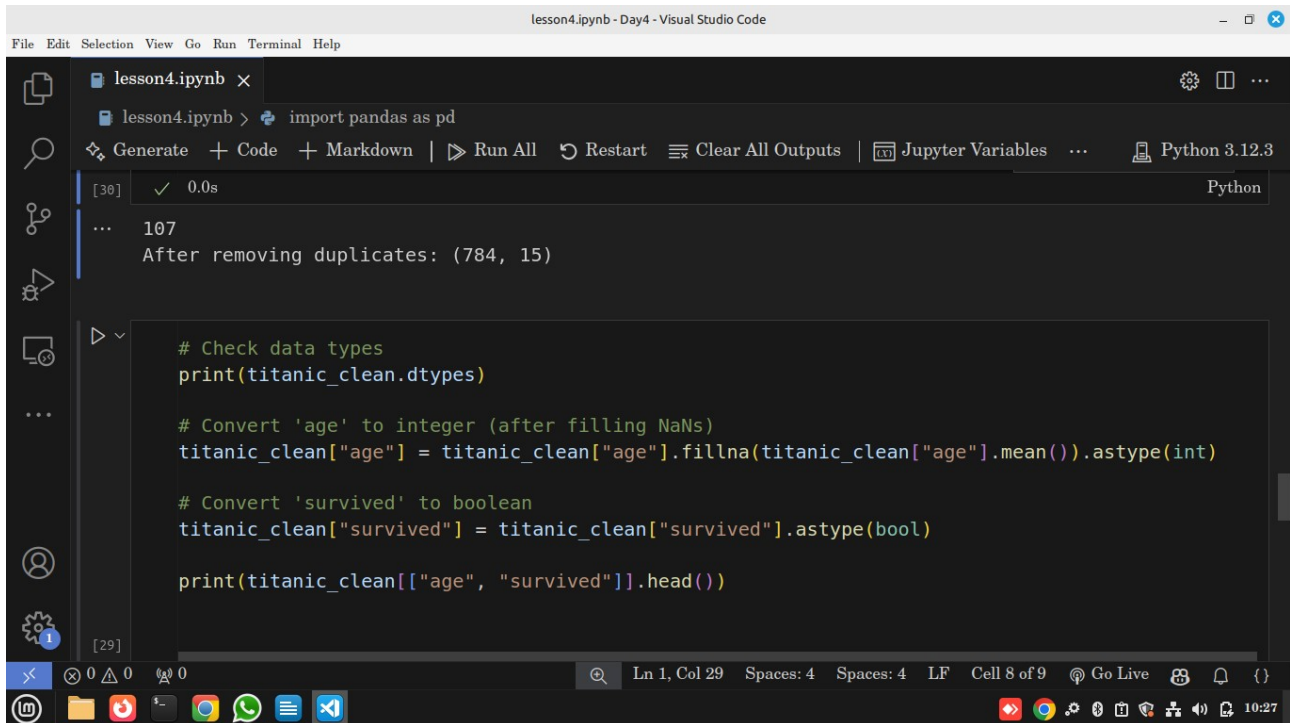
- `.duplicated()` → marks duplicates as True/False.
- `.drop_duplicates()` → removes them.

Use in Data Science

- Duplicates can bias results (e.g., same passenger counted twice).
-

◆ Fixing Data Types

Practical



The screenshot shows a Visual Studio Code window with a Jupyter notebook named 'lesson4.ipynb'. The notebook has two cells. The first cell contains the code `import pandas as pd` and has been executed, showing an output of `107` and a message 'After removing duplicates: (784, 15)'. The second cell contains the following code:

```
# Check data types
print(titanic_clean.dtypes)

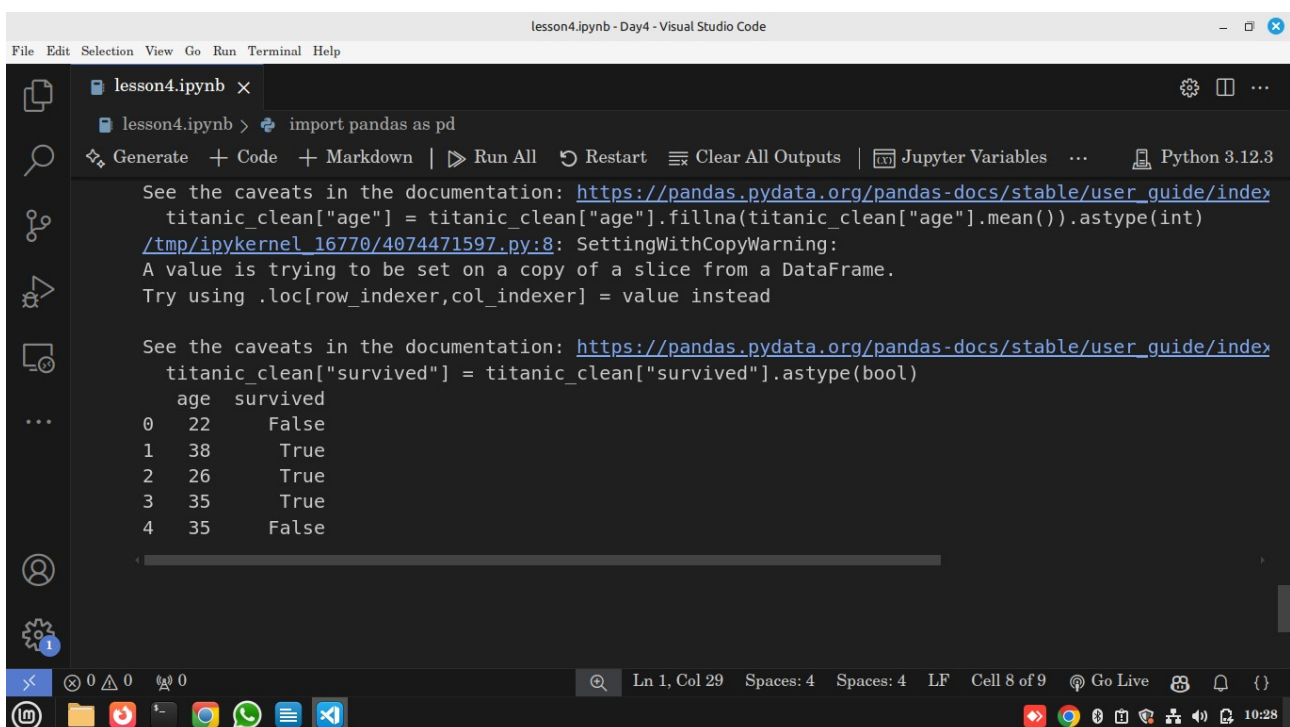
# Convert 'age' to integer (after filling NaNs)
titanic_clean["age"] = titanic_clean["age"].fillna(titanic_clean["age"].mean()).astype(int)

# Convert 'survived' to boolean
titanic_clean["survived"] = titanic_clean["survived"].astype(bool)

print(titanic_clean[["age", "survived"]].head())
```

The status bar at the bottom indicates the cursor is at line 1, column 29, with 4 spaces, 4 spaces, and LF line ending. It also shows 'Cell 8 of 9' and 'Go Live' button.

Output



The screenshot shows the same Visual Studio Code window with the Jupyter notebook. The second cell has been executed, and the output is displayed. It includes two warnings from pandas and a preview of the data.

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/index
titanic_clean["age"] = titanic_clean["age"].fillna(titanic_clean["age"].mean()).astype(int)
[/tmp/ipykernel_16770/4074471597.py:8](#): SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/index
titanic_clean["survived"] = titanic_clean["survived"].astype(bool)

	age	survived
0	22	False
1	38	True
2	26	True
3	35	True
4	35	False

The status bar at the bottom is the same as in the previous screenshot.

Explanation

- `.dtypes` → shows column data types.
- `.astype(int)` → ensures numerical consistency.
- `.astype(bool)` → makes survival easier to understand (True/False).

Use in Data Science

- Correct data types = fewer bugs + more efficient memory usage.
-

◆ Mini Challenge

- On Titanic dataset:
 1. Fill missing `age` with median instead of mean.
 2. Drop duplicate rows.
 3. Convert `class` column to category type.
 4. Save cleaned dataset as `titanic_clean.csv`.
-

Reflection

- Why does cleaning take 80% of time?
- Would you always drop missing values?
- How does fixing data types help later in **visualization & ML**?