```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub
markmedhat_titanic_path = kagglehub.dataset_download('markmedhat/titanic')

print('Data source import complete.')

```

```
Data source import complete.
```

```
# This Python 3 environment comes with many helpful analytics libraries installe
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/doc
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets
# You can also write temporary files to /kaggle/temp/, but they won't be saved o
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
!kaggle datasets download brendan45774/test-file
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this,
Dataset URL: https://www.kaggle.com/datasets/brendan45774/test-file
License(s): CC0-1.0
Downloading test-file.zip to /content
  0% 0.00/11.2k [00:00<?, ?B/s]
100% 11.2k/11.2k [00:00<00:00, 22.9MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile('/content/test-file.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

Understanding Your Data in Machine Learning

Importance of Initial Exploration

- **Analogy**: Just like a detective gathers initial clues before solving a case, data scientists need to ask basic questions to get an initial understanding of their data.
- **Goal**: Establish a foundational understanding of the dataset to guide further analysis.

Dataset Used

- **Titanic Dataset**: A famous dataset from Kaggle, commonly used by beginners in machine learning.
- **Reason for Choice**: Well-known structure and provides a good example for initial data exploration.

Key Questions to Ask When You First Get Your Data

1. **How Big Is the Data?**
    - **Method**: Use `df.shape` to find out the number of rows and columns.
    - **Analogy**: Knowing the size of a book before reading helps you plan your reading schedule.
    - **Purpose**: Helps in planning resources and time needed for analysis.
2. **What Does the Data Look Like?**
    - **Method**:
        - Use `df.head()` to view the first few rows.
        - Use `df.sample(n)` to view random samples.
    - **Analogy**: Skimming through a few pages at different sections of a book to get a sense of the content.
    - **Purpose**: Detect any initial biases and understand data distribution.
3. **What Are the Data Types of Each Column?**
    - **Method**: Use `df.info()` to get data types and non-null counts.
    - **Analogy**: Checking the ingredients before cooking to ensure you have everything you need.
    - **Purpose**: Identifies numerical vs. categorical variables and potential data type optimizations.
4. **Are There Missing Values?**
    - **Method**:
        - Use `df.isnull().sum()` to count missing values per column.
        - Use `df.isnull().mean()` to get the percentage of missing values.

    - **Analogy**: Inspecting a puzzle to see if any pieces are missing before starting to assemble it.
    - **Purpose**: Missing values can significantly affect analysis and modeling; knowing their presence is crucial.

5. **What Is the Statistical Summary of Numerical Columns?**
    - **Method**: Use `df.describe()` to get count, mean, standard deviation, min, max, and quartiles.
    - **Analogy**: Reading the nutrition facts on food packaging to understand what you're consuming.
    - **Purpose**: Provides insights into the distribution and range of numerical data.

6. **Are There Duplicate Rows?**
    - **Method**: Use `df.duplicated().sum()` to count duplicate rows.
    - **Analogy**: Checking for duplicate entries in your contact list to avoid confusion.
    - **Purpose**: Duplicate data can skew analysis and should be handled appropriately.

7. **What Are the Relationships Between Variables?**
    - **Method**: Use `df.corr()` to compute pairwise correlation of numerical columns.
    - **Analogy**: Understanding the relationships in a family tree to see how members are connected.
    - **Purpose**: Identifies which variables may influence each other, aiding in feature selection.

```
df = pd.read_csv("data/titaninc.csv")
```

```

```

Detailed Exploration of Each Question

1. How Big Is the Data?

```
df.shape
```

```
(418, 12)
```

- **Code Example**:

```
df.shape
```

- **Output Interpretation**:
  - Returns a tuple `(rows, columns)`.
  - For the Titanic dataset, it might be `(891, 12)`.

2. What Does the Data Look Like?

```
1 df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.829
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.000

Next steps: View recommended plots New interactive sheet

```
1 df.sample(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
353	1245	0	2	Herman, Mr. Samuel	male	49.0	1	2	220845
211	1103	0	3	Finoli, Mr. Luigi	male	NaN	0	0	SOTON/O.Q 3101308
18	910	1	3	Ilmakangas, Miss. Ida Livija	female	27.0	1	0	STON/O2 3101270

- **Using `head()`**:
  - Shows the first five rows.
  - Potential Bias: May not represent the entire dataset.
- **Using `sample(n)`**:
  - Provides a random sample.
  - **Code Example**:

    ```
    df.sample(5)
    ```

  - **Benefit**: Reduces the chance of biased impressions.

3. What Are the Data Types of Each Column?

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

- **Using `info()`**:
  - Lists all columns with their data types and non-null counts.
  - **Data Types**:
    - `int64`, `float64`: Numerical data.
    - `object`: Categorical or text data.
- **Optimization Tip**:
  - Convert unnecessary `float` types to `int` to save memory.
  - **Analogy**: Using exact change instead of larger bills when small denominations suffice.

4. Are There Missing Values?

```
1 df.isnull().sum()
```

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	86
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	327
Embarked	0

dtype: int64

- **Identifying Missing Values**:
  - **Code Example**:

    ```
    df.isnull().sum()
    ```

  - Provides a count of missing values per column.
- **Handling Missing Values**:
  - **Options**:
    - Remove columns or rows with too many missing values.
    - Impute missing values using mean, median, or mode.
- **Analogy**: Deciding whether to repair or discard a damaged item based on the extent of the damage.

5. What Is the Statistical Summary of Numerical Columns?

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fa
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000(
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000(
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500(
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329

- **Using `describe()`**:
  - Provides statistical metrics like mean, standard deviation, min, max, quartiles.
  - **Insights**:
    - Detect outliers by comparing mean and median.
    - Understand the spread and distribution.
- **Analogy**: Looking at a city's weather averages to plan your wardrobe accordingly.

6. Are There Duplicate Rows?

```
df.duplicated().sum()
```

```
0
```

Double-click (or enter) to edit

- **Identifying Duplicates**:
  - **Code Example**:

    ```
    df.duplicated().sum()
    ```

  - **Handling Duplicates**:
    - Remove duplicates using `df.drop_duplicates()`.
- **Analogy**: Removing duplicate files from your computer to free up space and avoid confusion.

7. What Are the Relationships Between Variables?

```
numeric_df = df.select_dtypes(include=['float64', 'int64'])
```

```

numeric_df.corr()['Survived']
```

	Survived
PassengerId	-0.023245
Survived	1.000000
Pclass	-0.108615
Age	-0.000013
SibSp	0.099943
Parch	0.159120
Fare	0.191514

dtype: float64

- **Using `corr()`**:
  - Computes the correlation matrix.
  - **Interpretation**:
    - Values range from `-1` to `1`.
    - **Positive Correlation**: As one variable increases, so does the other.
    - **Negative Correlation**: As one variable increases, the other decreases.
    - **Zero Correlation**: No linear relationship.
- **Example**:
  - **Survived vs. Fare**:
    - Positive correlation suggests higher fare passengers had a higher survival rate.
    - **Analogy**: In a luxury hotel, guests paying more might receive better services.
  - **Survived vs. Pclass**:
    - Negative correlation indicates lower-class passengers had lower survival rates.
    - **Analogy**: In an emergency, first-class passengers might have better access to lifeboats.

Additional Insights

- **Data Types Affect Memory Usage**:

    - Optimizing data types can improve processing speed and reduce memory consumption.
    - **Example**: Converting `float64` to `int32` where applicable.

- **Missing Values Require Careful Handling**:
    - Ignoring missing values can lead to inaccurate models.
    - **Strategies**:
        - **Dropping**: If a column has too many missing values.
        - **Imputation**: Filling missing values based on statistical methods.

- **Correlation Does Not Imply Causation**:
    - High correlation between two variables doesn't mean one causes the other.
    - **Analogy**: Ice cream sales and drowning incidents both increase in summer but are not causally related.

Conclusion

- **Recap**:
    - Asking basic questions about your data is a crucial first step in any data science project.
    - These questions help identify potential issues and guide your analysis strategy.

- **Key Takeaways**:
    - Always inspect data size and structure.
    - Check data types and optimize if necessary.
    - Identify and handle missing and duplicate values.
    - Explore statistical summaries and variable relationships.

- **Final Thought**:
    - **Analogy**: Think of your dataset as a new city you're exploring. Before diving into the details, get a map (overview), understand the neighborhoods (variables), and identify areas of interest (key insights) to make the most of your journey.
