

Task 1 Report

Advanced Data Cleaning with Fuzzy String Matching

1. Task Description

In this task, the goal was to perform advanced data cleaning techniques on the Titanic dataset. The focus was on handling missing values, applying fuzzy string matching for identifying potential duplicates in the **Name** column, and removing those duplicates to ensure the dataset was clean and consistent.

The main steps involved:

- ❖ Handling missing values in columns like **Age**, **Fare**, and **Cabin**.
- ❖ Using **FuzzyWuzzy**, a library for fuzzy string matching, to identify similar names in the **Name** column.
- ❖ Removing duplicates based on fuzzy matching results.
- ❖ Saving the cleaned dataset as **cleaned_data.csv**.

The task aimed to improve the quality of the dataset by addressing both missing data and duplicate entries, which are common issues in real-world datasets.

2. Attach Screenshot of Output

❖ Dataset Preview:

```
DS & ML_TASK_NO_1_MANISH_KUMAR.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

Files
{ }
sample_data
tested.csv

+ Code + Text

Dataset Preview:
PassengerId Survived Pclass \
0 892 0 3
1 893 1 3
2 894 0 2
3 895 0 3
4 896 1 3

Name Sex Age SibSp Parch \
0 Kelly, Mr. James male 34.5 0 0
1 Wilkes, Mrs. James (Ellen Needs) female 47.0 1 0
2 Myles, Mr. Thomas Francis male 62.0 0 0
3 Wirz, Mr. Albert male 27.0 0 0
4 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female 22.0 1 1

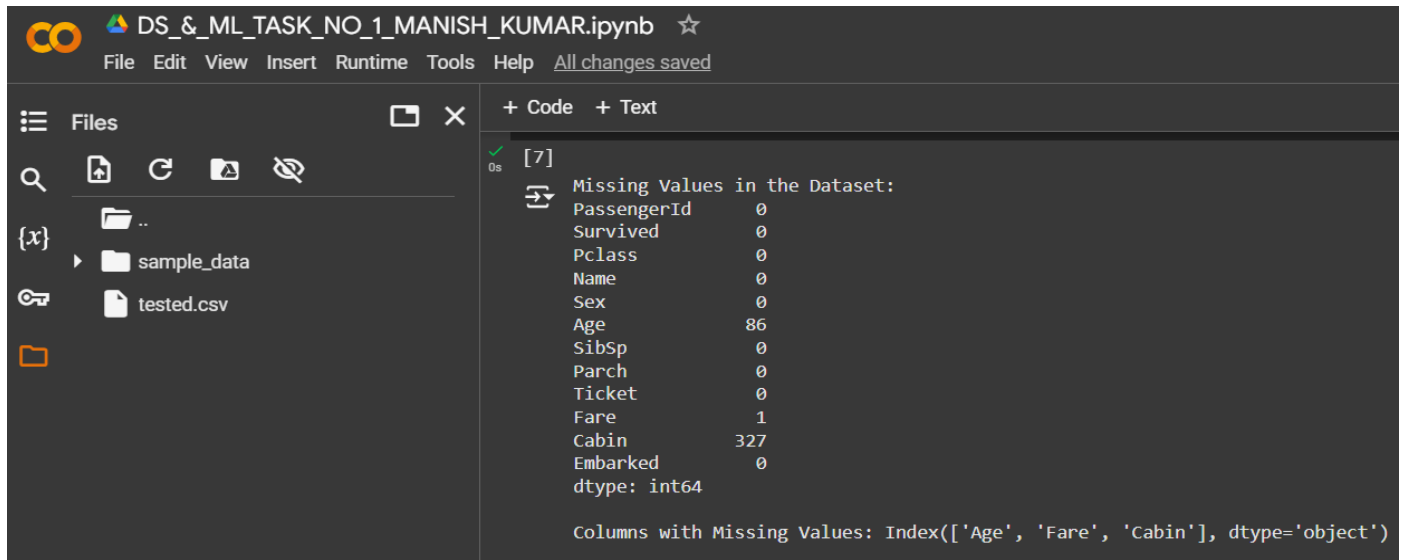
Ticket Fare Cabin Embarked
0 330911 7.8292 NaN Q
1 363272 7.0000 NaN S
2 240276 9.6875 NaN Q
3 315154 8.6625 NaN S
4 3101298 12.2875 NaN S

Dataset Information:
<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
```

Task 1 Report

Advanced Data Cleaning with Fuzzy String Matching

❖ Missing values in the Dataset:

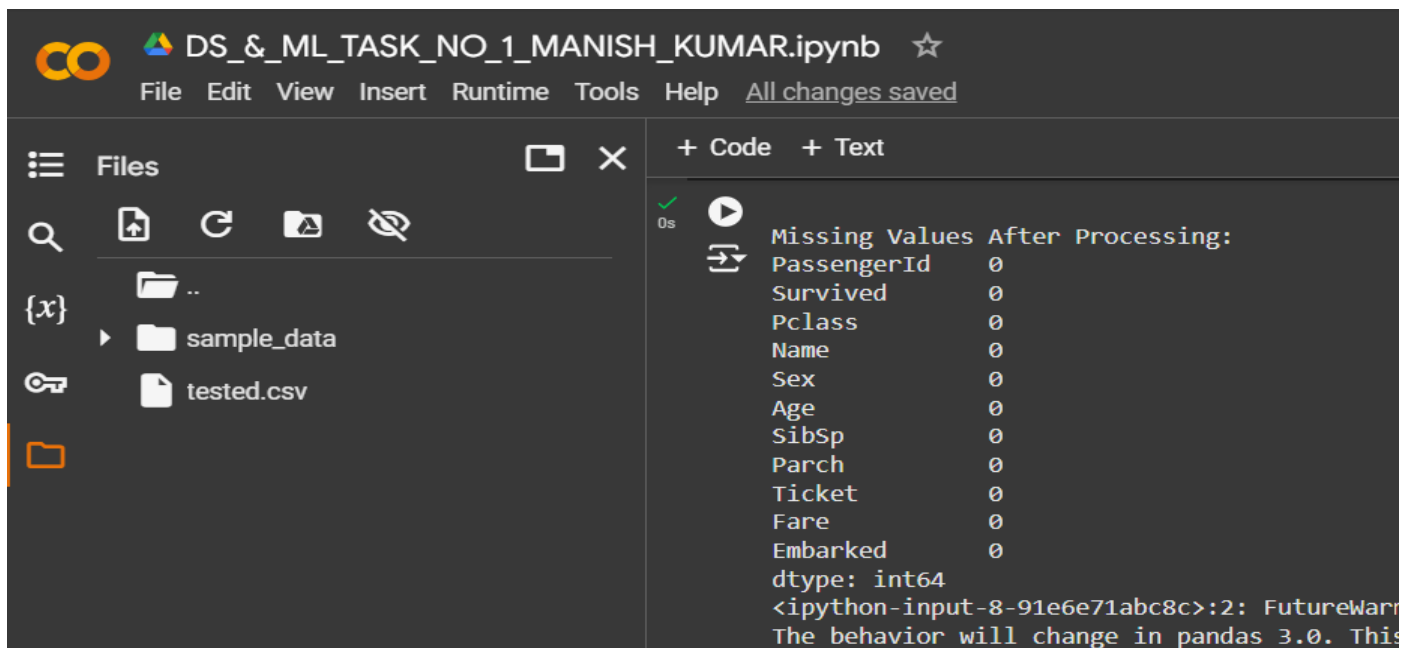


The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'sample_data' containing a file 'tested.csv'. The code editor displays the following output:

```
[7]
Missing Values in the Dataset:
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

Columns with Missing Values: Index(['Age', 'Fare', 'Cabin'], dtype='object')
```

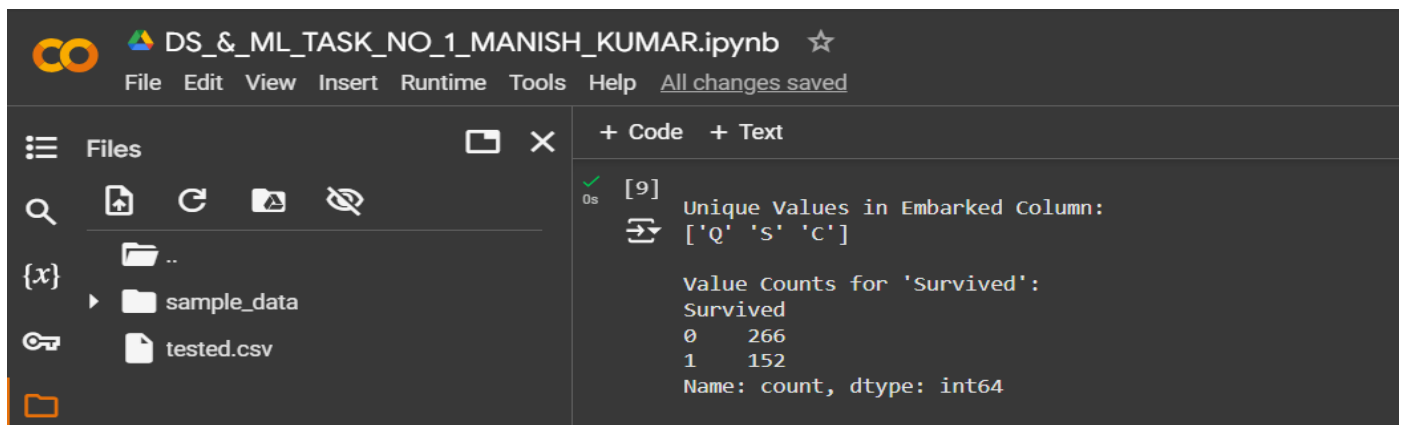
❖ Missing values after Processing:



The screenshot shows the same Jupyter Notebook interface. The code editor displays the following output:

```
Missing Values After Processing:
PassengerId    0
Survived        0
Pclass          0
Name            0
Sex             0
Age            0
SibSp           0
Parch           0
Ticket          0
Fare            0
Embarked        0
dtype: int64
<ipython-input-8-91e6e71abc8c>:2: FutureWarning
The behavior will change in pandas 3.0. This
```

❖ Unique Values:



The screenshot shows the same Jupyter Notebook interface. The code editor displays the following output:

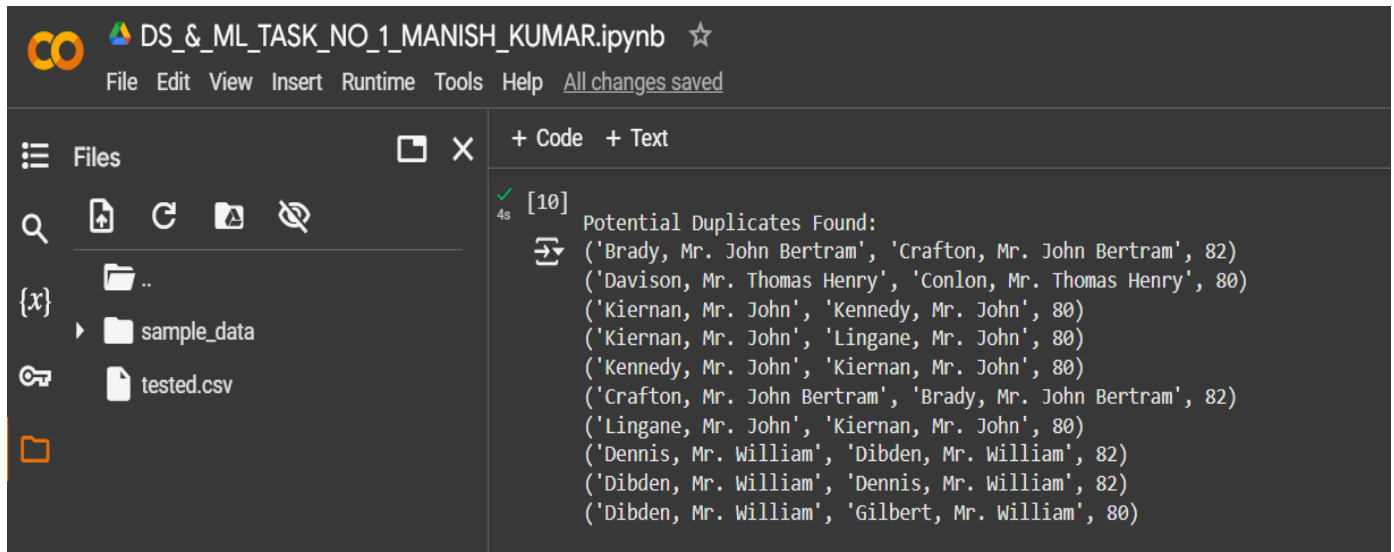
```
[9]
Unique Values in Embarked Column:
['Q' 'S' 'C']

Value Counts for 'Survived':
Survived
0      266
1      152
Name: count, dtype: int64
```

Task 1 Report

Advanced Data Cleaning with Fuzzy String Matching

❖ Potential Duplicates Found:



DS_&_ML_TASK_NO_1_MANISH_KUMAR.ipynb

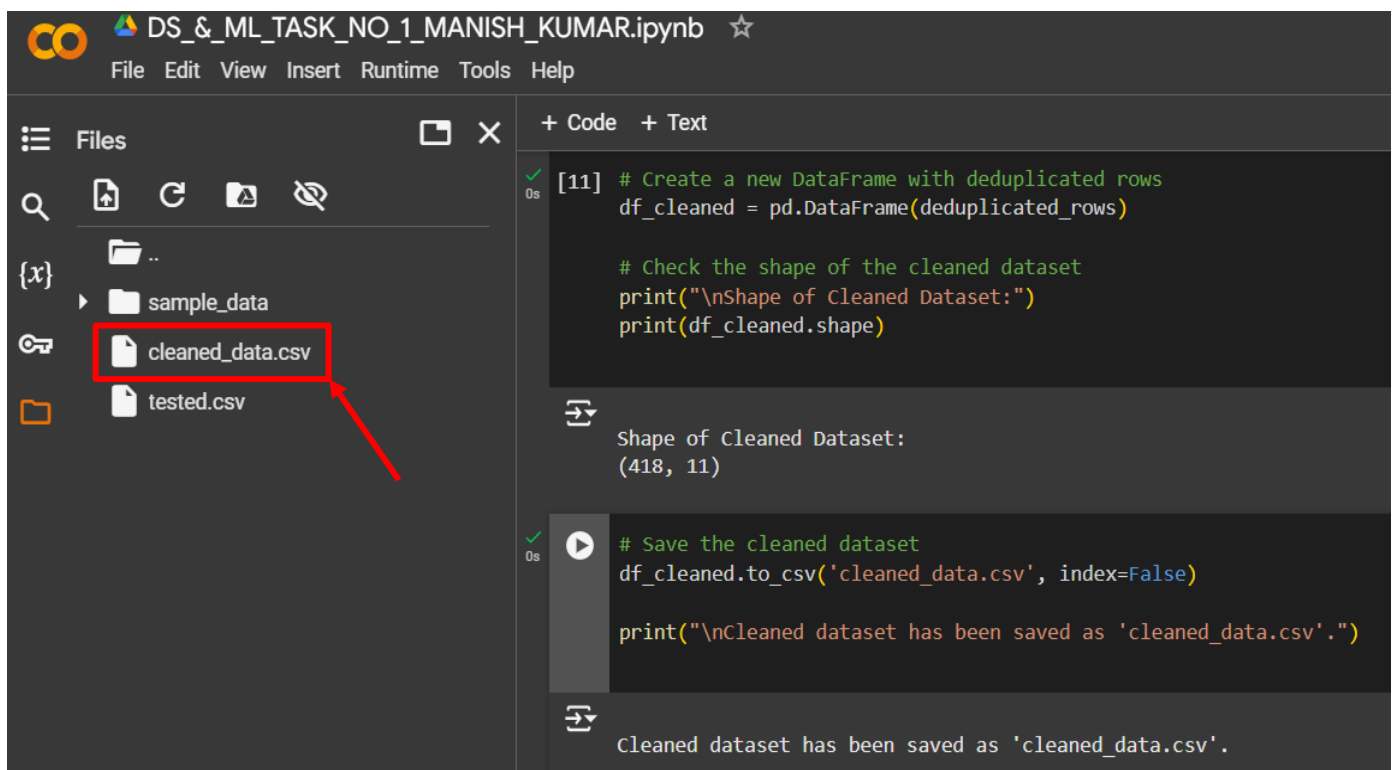
File Edit View Insert Runtime Tools Help All changes saved

Files

- sample_data
- tested.csv

```
[10] Potential Duplicates Found:
('Brady, Mr. John Bertram', 'Crafton, Mr. John Bertram', 82)
('Davison, Mr. Thomas Henry', 'Conlon, Mr. Thomas Henry', 80)
('Kiernan, Mr. John', 'Kennedy, Mr. John', 80)
('Kiernan, Mr. John', 'Lingane, Mr. John', 80)
('Kennedy, Mr. John', 'Kiernan, Mr. John', 80)
('Crafton, Mr. John Bertram', 'Brady, Mr. John Bertram', 82)
('Lingane, Mr. John', 'Kiernan, Mr. John', 80)
('Dennis, Mr. William', 'Dibden, Mr. William', 82)
('Dibden, Mr. William', 'Dennis, Mr. William', 82)
('Dibden, Mr. William', 'Gilbert, Mr. William', 80)
```

❖ Cleaned Dataset



DS_&_ML_TASK_NO_1_MANISH_KUMAR.ipynb

File Edit View Insert Runtime Tools Help

Files

- sample_data
- cleaned_data.csv
- tested.csv

```
[11] # Create a new DataFrame with deduplicated rows
df_cleaned = pd.DataFrame(deduplicated_rows)

# Check the shape of the cleaned dataset
print("\nShape of Cleaned Dataset:")
print(df_cleaned.shape)

Shape of Cleaned Dataset:
(418, 11)

# Save the cleaned dataset
df_cleaned.to_csv('cleaned_data.csv', index=False)

print("\nCleaned dataset has been saved as 'cleaned_data.csv'.")

Cleaned dataset has been saved as 'cleaned_data.csv'.
```

3. Describe Widget/Algorithm Used in Task

Algorithm Used: Fuzzy String Matching (FuzzyWuzzy Library)

Fuzzy String Matching is a technique used to compute the similarity between two strings, particularly useful for identifying and resolving inconsistencies like typos or variations in textual data. In this task, the **FuzzyWuzzy** Python library was employed to identify potential duplicates in the **Name** column of the Titanic dataset. Below is an elaboration of the algorithm and its implementation:

Task 1 Report

Advanced Data Cleaning with Fuzzy String Matching

Core Concept: Levenshtein Distance

- ❖ **Levenshtein Distance**, also known as edit distance, measures the number of single-character edits (insertions, deletions, substitutions) required to change one string into another.
- ❖ FuzzyWuzzy leverages this concept to compute a **similarity ratio** (percentage match) between two strings.

FuzzyWuzzy Methods Used

❖ **fuzz.token_sort_ratio:**

- ✚ This method tokenizes the string (splits it into words), sorts the tokens, and then compares the sorted tokens. This helps in comparing strings where the word order might differ.

✚ Example:

- String 1: *"John Smith"*
- String 2: *"Smith John"*
- fuzz.token_sort_ratio will return a **100% match**.

❖ **Threshold:**

- ✚ A similarity threshold of **80%** was set to identify potential duplicates. Names with a similarity score of **80 or above** were flagged for deduplication.

Steps in the Process

1. **Data Preprocessing:**

✚ **Handling Missing Values:**

- ❖ **Age:** Replaced missing values with the column mean.
- ❖ **Fare:** Replaced the single missing value with the column median.
- ❖ **Cabin:** Dropped the column due to a high percentage of missing data.

- ✚ Ensured all necessary preprocessing steps (e.g., removing unnecessary whitespace in names) were completed.

2. **Fuzzy Matching:**

- ✚ Applied **FuzzyWuzzy** to the **Name** column.
- ✚ Compared each name in the dataset to every other name.
- ✚ Generated a list of name pairs with a similarity score ≥ 80 .

3. **Deduplication:**

- ✚ Reviewed flagged duplicates manually or programmatically to determine whether they represented true duplicates.
- ✚ Retained only the first occurrence of each duplicate name and removed the rest.

Task 1 Report

Advanced Data Cleaning with Fuzzy String Matching

Libraries/Tools Used

1. **Pandas:** Used for dataset manipulation, including filling missing values, dropping unnecessary columns, and exporting the cleaned data.
2. **FuzzyWuzzy:** Provided powerful tools for fuzzy string matching, particularly for comparing text fields like names.
3. **Python:** Implemented the entire workflow, including preprocessing, fuzzy matching, and deduplication.

Benefits of the Algorithm

- ❖ **Accuracy:** Handles slight variations in text effectively, making it ideal for datasets with inconsistent naming conventions.
- ❖ **Efficiency:** While comparing every record to every other record could be computationally expensive, FuzzyWuzzy streamlines the process with token-based comparisons.
- ❖ **Customizable:** Threshold values and methods can be adjusted based on the dataset's specific requirements

*** The End ***