Data Science – Report Task 2

Task - Data Cleaning and Transformation

Use a messy dataset with missing values, duplicate entries, and inconsistent formatting to clean and transform it into a usable form.

Summary - This code cleans and transforms the Titanic dataset from Kaggle for analytical and machine learning purposes. Key tasks include handling missing values (e.g., filling Age and Fare with medians), encoding categorical variables (Sex, Embarked), and simplifying columns like Cabin and Ticket.Advanced features include phonetic clustering for deduplication, title extraction from names, and outlier detection for Age and Fare. New features (FamilySize, Is_Alone, Title) are created, and numerical columns are normalized for consistency. The final cleaned dataset is saved as cleaned_data.csv, ready for predictive modeling or exploratory analysis.

Import Libraries and Load Data

```
[1] # Import Libraries
    import numpy as np
    import pandas as pd
    import jellyfish  # For phonetic clustering

    # Load Data
    file_path = '/content/tested.csv'  # Update the path for Google Colab
    data = pd.read_csv(file_path)
    data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Data Cleaning and Transformation

```
# Fill missing 'Age' with median
age_median = data['Age'].median()
data['Age'].fillna(age_median, inplace=True)

# Fill missing 'Fare' with median
fare_median = data['Fare'].median()
data['Fare'].fillna(fare_median, inplace=True)

# Create a binary column for 'Cabin' indicating whether a cabin was assigned
data['Has_Cabin'] = data['Cabin'].notnull().astype(int)
data.drop(columns=['Cabin'], inplace=True)
```

```
<ipython-input-2-164cdda5e5b4>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform

  data['Age'].fillna(age_median, inplace=True)
<ipython-input-2-164cdda5e5b4>:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform

  data['Fare'].fillna(fare_median, inplace=True)
```

Simplify and Transform Features

```
[3] # Simplify 'Ticket' by extracting its length
    data['Ticket_Length'] = data['Ticket'].apply(lambda x: len(str(x)))
    data.drop(columns=['Ticket'], inplace=True)

    # Encode 'Sex' and 'Embarked'
    data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
    data['Embarked'] = data['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})
```

OpenRefine-Like Transformations

```
# 4.1 Standardize and Clean Text Columns
data['Name'] = data['Name'].str.strip().str.title()  # Standardize to title case and strip whitespace

# Split 'Name' into 'LastName' and 'OtherNames'
data[['LastName', 'OtherNames']] = data['Name'].str.split(',', n=1, expand=True)

# Deduplicate based on 'LastName' using clustering-like methods
data['LastName'] = data['LastName'].str.lower().str.strip()
data['LastName'] = data['LastName'].str.replace(r'[^a-z]', '', regex=True)

# 4.2 Phonetic Clustering
data['Soundex'] = data['LastName'].apply(lambda x: jellyfish.soundex(x))
soundex_map = data.groupby('Soundex')['LastName'].first().to_dict()
data['LastName'] = data['Soundex'].map(soundex_map)

# Remove duplicates while keeping the first occurrence
data = data.drop_duplicates(subset='LastName', keep='first')

# 4.3 Standardize Unique Values in 'Embarked'
data['Embarked'] = data['Embarked'].fillna(0)
data['Embarked'] = data['Embarked'].astype(int)
```

Feature Engineering

```
[5] # Create 'FamilySize' feature
    data['FamilySize'] = data['SibSp'] + data['Parch'] + 1

    # Create 'Is_Alone' feature
    data['Is_Alone'] = (data['FamilySize'] == 1).astype(int)

    # Extract Titles from Names
    data['Title'] = data['OtherNames'].str.extract(r'([A-Za-z]+)\.', expand=False).fillna('Unknown')

    # Bin Titles into Categories
    title_mapping = {
        'Mr': 'Mr', 'Miss': 'Miss', 'Mrs': 'Mrs', 'Master': 'Master',
        'Dr': 'Officer', 'Rev': 'Officer', 'Col': 'Officer',
        'Major': 'Officer', 'Mlle': 'Miss', 'Ms': 'Miss', 'Mme': 'Mrs',
        'Capt': 'Officer', 'Lady': 'Noble', 'Sir': 'Noble', 'Don': 'Noble'
    }
    data['Title'] = data['Title'].map(title_mapping).fillna('Other')
```

Data Profiling

```
# Detect Outliers in 'Age' using IQR
Q1 = data['Age'].quantile(0.25)
Q3 = data['Age'].quantile(0.75)
IQR = Q3 - Q1
data['Age_Outlier'] = ((data['Age'] < (Q1 - 1.5 * IQR)) | (data['Age'] > (Q3 + 1.5 * IQR))).astype(int)

# Detect Outliers in 'Fare'
Q1_fare = data['Fare'].quantile(0.25)
Q3_fare = data['Fare'].quantile(0.75)
IQR_fare = Q3_fare - Q1_fare
data['Fare_Outlier'] = ((data['Fare'] < (Q1_fare - 1.5 * IQR_fare)) | (data['Fare'] > (Q3_fare + 1.5 * IQR_fare))).astype(int)
```

Normalize Numerical Features

```
[7] # Normalize 'Age' and 'Fare'
    data['Age'] = (data['Age'] - data['Age'].mean()) / data['Age'].std()
    data['Fare'] = (data['Fare'] - data['Fare'].mean()) / data['Fare'].std()
```

Finalize and Save Data

```
# Drop unused columns and save cleaned data
data.drop(columns=['PassengerId', 'Name', 'Soundex'], inplace=True)
data.to_csv('/content/cleaned_data.csv', index=False)

print("Data cleaning and transformation complete. Cleaned data saved as 'cleaned_data.csv'.")
```

```
Data cleaning and transformation complete. Cleaned data saved as 'cleaned_data.csv'.
```

Output –

The output is a cleaned and transformed version of the Titanic dataset from Kaggle. It includes no missing values, encoded categorical variables (Sex, Embarked), and engineered features like FamilySize, Is_Alone, and Title. Numerical columns (Age, Fare) are normalized, and outliers are flagged. The dataset is deduplicated using phonetic clustering on last names. This cleaned data, saved as cleaned_data.csv, is ready for machine learning, predictive modeling, or detailed analysis.

Skills Learned - Data cleaning, handling missing data.

Tools - Python (Pandas, OpenRefine).

Conclusion - The cleaned and transformed Titanic dataset provides a robust foundation for analytical and predictive tasks. By addressing missing values, simplifying features, and engineering new ones, the dataset is optimized for machine learning models and detailed exploration. The transformations ensure consistency, reduce redundancy, and enhance data usability, making it ideal for extracting meaningful insights or building accurate predictions.