Data Preprocessing

Introduction

Data preprocessing is a fundamental step in data analysis and machine learning. It ensures raw data is transformed into a clean, structured, and analyzable format. This report highlights the steps performed on a dataset to prepare it for machine learning, including handling missing values, encoding categorical variables, and analyzing feature importance.

Steps Performed

1. Data Loading

The dataset was loaded into a pandas DataFrame to facilitate analysis. An initial inspection revealed the structure of the dataset, including column names, data types, and the presence of missing values.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
file_path =r'C:\Users\sauda\Downloads\netflix_titles.csv'
df = pd.read_csv(file_path)
```

2. Handling Missing Values

Missing values were identified in both numerical and categorical columns. These were handled as follows:

- For numerical columns, missing values were replaced with the median of the respective column to prevent skewing the data.
- For categorical columns, missing values were replaced with the most frequently occurring value (mode) to maintain consistency.

This ensured that the dataset was complete and no rows contained missing data.

```python
df.info()


df.describe()


missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)
```

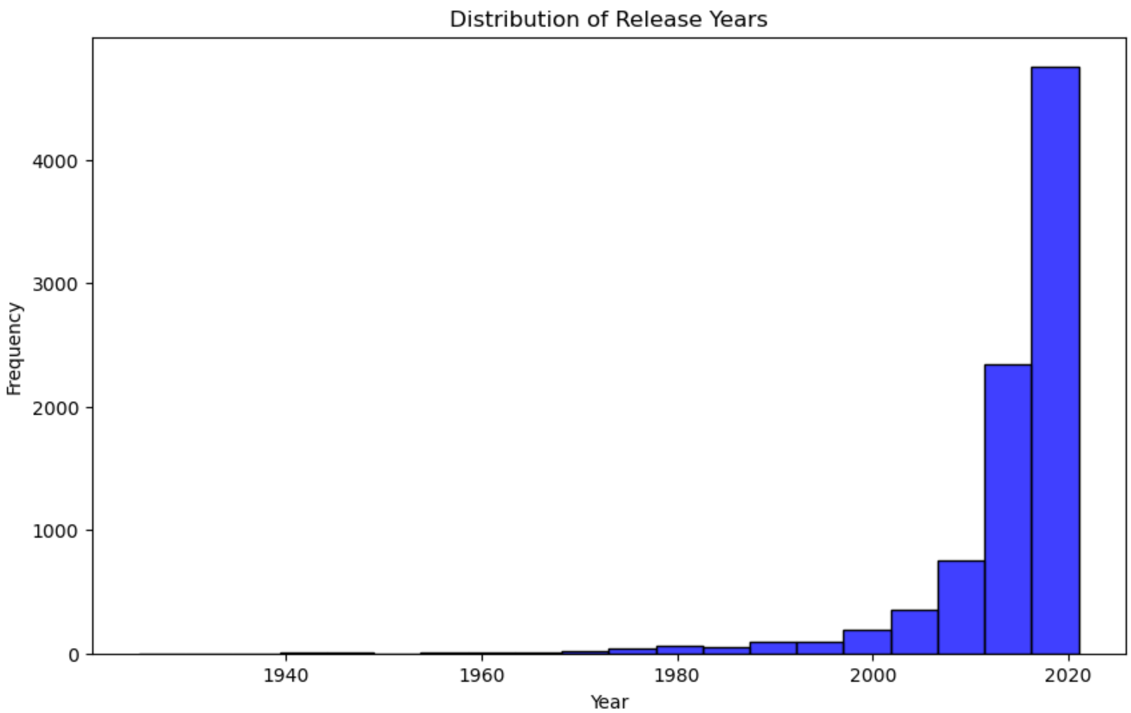
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
Missing Values:
 show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

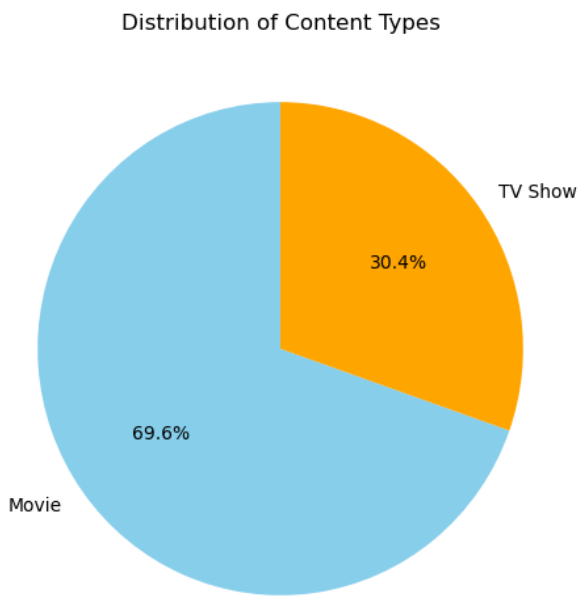
```python
for column in df.select_dtypes(include=['float64', 'int64']).columns:
    df[column].fillna(df[column].median(), inplace=True)


for column in df.select_dtypes(include=['object']).columns:
    df[column].fillna('Unknown', inplace=True)


print("Missing Values After Handling:\n", df.isnull().sum())
```

```
Missing Values After Handling:
 show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

3. Visualization

The dataset contained several non-numeric columns, primarily representing categorical data. These columns needed transformation to numeric formats to be compatible with machine learning algorithms.





4. Encoding Categorical Variables

Since machine learning models require numerical inputs, all categorical variables were transformed into numeric representations using Label Encoding. This process assigned a unique integer to each category, ensuring the data could be processed effectively.

```
Acateg_col=df.select_dtypes(include=['object']).columns
encoder= LabelEncoder()
```

```
for col in categ_col:
    data[col]=encoder.fit_transform(data[col])
```

5. Checking for Missing Values After Transformation

A thorough validation was conducted after handling missing values and encoding categorical variables to ensure there were no remaining gaps in the dataset.

```
for column in df.select_dtypes(include=['float64', 'int64']).columns:
    df[column].fillna(df[column].median(), inplace=True)


for column in df.select_dtypes(include=['object']).columns:
    df[column].fillna('Unknown', inplace=True)


print("Missing Values After Handling:\n", df.isnull().sum())
```

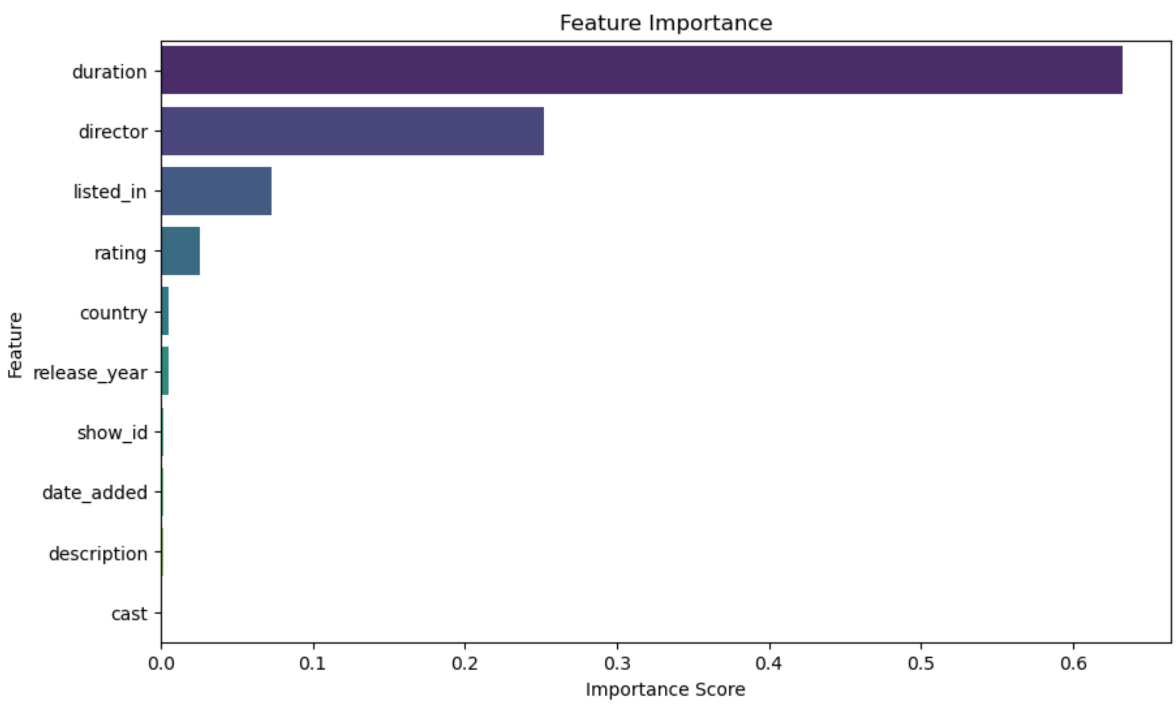
```
Missing Values After Handling:
 show_id        0
type           0
title          0
director       0
cast           0
country        0
date_added     0
release_year   0
rating         0
duration       0
listed_in      0
description    0
dtype: int64
```

6. Feature Importance Analysis

To understand the contribution of each feature to the target variable, a Random Forest Classifier was applied to the processed dataset. This provided insights into feature importance:

- Irrelevant columns, such as identifiers, were excluded from the analysis.
- The classifier ranked features based on their importance to the prediction of the target variable.
- The results were visualized using a bar plot to highlight the most and least influential features.

```python
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns


model = RandomForestClassifier(random_state=42)


model.fit(X, y)


importances = model.feature_importances_


feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)


print(feature_importance_df)

plt.figure(figsize=(10, 6))
sns.barplot(data=feature_importance_df, x='Importance', y='Feature', palette='viridis')
plt.title('Feature Importance')
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.show()
```



Results

1. **Missing Values**: All missing values in numerical and categorical columns were successfully handled, ensuring a complete dataset.

2. **Categorical Encoding**: Non-numeric columns were converted into numeric formats, making the dataset compatible with machine learning algorithms.

3. **Feature Importance**:
    - Key features contributing significantly to the target variable were identified.
    - Visualization provided clarity on the relative importance of each feature.

Conclusion

The preprocessing steps ensured the dataset was clean, consistent, and machine learning-ready. Handling missing values, encoding categorical variables, and analyzing feature importance were critical to preparing the data. This structured approach not only improves model performance but also ensures reliable insights can be derived from the data.