Data Preprocessing Using Pandas

Steps:

1) Remove Duplicate values.
2) Removing columns that have major NaN values.
3) Mean/Median value imputation for NaN values in numerical columns.
4) Mode/constant value imputation for NaN values in categorical columns.
5) One hot encoding.

1) Remove Duplicate Values

```
# Returns a series of boolean values, True->Duplicate, False->Unique
df.duplicated()

# Finding total number of duplicate values
df.duplicated().sum()

# Displaying the duplicate records
df[df.duplicated() == True]

# Removing duplicate values
df.drop_duplicates(inplace=True)
```

2) Removing columns that have major NaN values

```
# Returns a boolean dataframe. True: if NaN, False if Non NaN
df.isnull()

# Finding total number of null values in each column
df.isnull().sum()

# Finding total null values
df.isnull().sum().sum()

# Getting percentage of null values in each column
percent_of_NaN = df.isnull().sum() / df.shape[0] * 100

# Plotting heatmap
sns.heatmap(data.isnull())

# Getting all the column names that have NaN above 17%
# U can specify any value instead of 17
NaN_columns = percent_of_NaN[percent_of_NaN > 17].keys()
```

```
#Creating a copy of original df
data_copy = df.copy()

# Now we have do delete above columns which have majority of NaN values
data_copy = data_copy.drop(NaN_columns, axis=1)

# Now seeing reduced NaN values
data_copy.isnull().sum().sum()
```

3) Mean/Median value imputation for NaN values in numerical columns.

```
# Getting only numerical columns from the dataframe
num_df = data_copy.select_dtypes(['int', 'float'])
num_df.shape

# Getting numerical column names
num_cols = num_df.columns
num_cols

# Getting numerical columns that have NaN values
NaN_col_list = num_df.isnull().sum()[num_df.isnull().sum() > 0].keys()
NaN_col_list


# Mean imputation
data_copy[num_cols] = data_copy[num_cols].fillna(data_copy[num_cols].mean())


# Checking reduced NaN values
data_copy.shape
data_copy.isnull().sum().sum()

# Visualization of original vs imputation
plt.figure(figsize=(16,9))
for i,var in enumerate(NaN_col_list):
    plt.subplot(4,3,i+1)
    sns.histplot(data_copy[var],label="Impute", element='poly', fill=False)
    sns.histplot(data[var].dropna(),label="Original", element='poly', fill=False)
    plt.legend()
```

4) Mode/constant value imputation for NaN values in categorical columns.

```
# Dataframe containing only categorical variable
cat_df = data_copy.select_dtypes(['object'])

# Getting categorical column names
cat_cols = cat_df.columns
cat_cols

# Getting categorical columns that have NaN values
NaN_col_list = cat_df.isnull().sum()[cat_df.isnull().sum() > 0].keys()
NaN_col_list

# Displaying records that has NaN values
cat_df.loc[:, NaN_col_list]

# Mode imputation
data_copy[cat_cols] = data_copy[cat_cols].fillna(data_copy[cat_cols].mode().iloc[0])
```

Or

```
# Constant imputation
data_copy[cat_cols] = data_copy[cat_cols].fillna(“Missing”)


# Finally checking data shape and NaN values
data_copy.shape
data_copy.isnull().sum().sum()
sns.heatmap(data_copy.isnull())

plt.figure(figsize=(16,9))
for i,var in enumerate(NaN_col_list):
    plt.subplot(4,3,i+1)
    sns.distplot(data_copy[var],label="Impute", element='poly', fill=False)
    sns.distplot(data[var].dropna(),label="Original", element='poly', fill=False)
    plt.legend()
```

5) One hot encoding

```
data_copy = pd.get_dummies(data=data_copy)
data_copy.shape
data_copy.head()
```