Data Preprocessing for Predicting House Prices

Step 1: Load the Dataset

- **Objective**: Import the dataset into a usable format, like a Pandas DataFrame.
- **Action**: Use a tool like Pandas to read your dataset (typically a `.csv` file) and inspect its structure. This will help you identify the features and target variable.
    - **Inspect the first few rows**: Check the data to understand what each column represents.
    - **Check data types**: Look at the data types for each feature (numerical, categorical, etc.).

Step 2: Handle Missing Values

- **Objective**: Clean the dataset by handling missing or incomplete data.
- **Action**:
    - **Identify missing values**: Look for null or missing values in your dataset. You can use the `.isnull()` method in Pandas.
    - **Decide how to handle missing data**:
        - For **numerical features**, you can either fill in the missing values with the **mean**, **median**, or **mode**, or remove rows or columns with too many missing values.
        - For **categorical features**, you can fill in missing values with the most frequent category or drop rows/columns if they are too sparse.

Step 3: Handle Categorical Data

- **Objective**: Convert categorical data (non-numeric) into numeric values because neural networks require numerical input.
- **Action**:
    - **Identify categorical features**: These could include columns like `Neighborhood`, `HouseStyle`, etc.
    - **Use encoding techniques**:
        - **Label Encoding**: Convert each category into a unique integer (e.g., "Suburban" = 0, "Urban" = 1, etc.).
        - **One-Hot Encoding**: Convert categorical variables into a binary matrix (e.g., if a "Neighborhood" column has categories `A`, `B`, and `C`, create 3 separate columns indicating whether each row is in one of those neighborhoods). Use `pd.get_dummies()` in Pandas for this.
    - **Note**: One-Hot Encoding is preferable if the categorical variable does not have a natural ordinal relationship (e.g., "Red", "Green", "Blue" colors).

Step 4: Handle Outliers

- **Objective**: Identify and handle extreme values that may distort the training process.
- **Action**:
    - **Visualize the data**: Use box plots or histograms to visually check for outliers, particularly in numerical columns (e.g., `GrLivArea` or `SalePrice`).
    - **Remove or cap outliers**: If an outlier is due to data entry errors, you can remove the row. If the outlier is valid but extreme, you can choose to cap the value (e.g., setting values above a certain threshold to that threshold).

Step 5: Feature Scaling (Normalization or Standardization)

- **Objective**: Ensure that numerical features are on the same scale, which is important for neural networks to perform well.
- **Action**:
    - **Normalization**: Scale features to a range of 0 to 1 (common for most machine learning models). This is useful for features like square footage or year of construction.
    - **Standardization**: Scale features to have a mean of 0 and a standard deviation of 1 (useful when your data is normally distributed).
    - Use Scikit-learn's `MinMaxScaler` or `StandardScaler` to apply these transformations.

Step 6: Feature Engineering (Optional, But Useful)

- **Objective**: Create new features that could help the model perform better.
- **Action**:
    - **Example 1 - Create a new feature**: If you have a `YearBuilt` column, you could create a new feature for the **age of the house** by subtracting `YearBuilt` from the current year.
    - **Example 2 - Log transformation**: For heavily skewed data (e.g., house prices), you can apply a **log transformation** to make the data more normal (e.g., `log(SalePrice)`).

Step 7: Split the Data

- **Objective**: Split the dataset into **training** and **testing** sets.
- **Action**:
    - **Training set**: This is the data used to train the model (typically 80% of the data).
    - **Testing set**: This is the data used to evaluate how well the model performs on unseen data (typically 20% of the data).
    - Use Scikit-learn’s `train_test_split()` function to randomly split the data.

Step 8: Verify and Finalize Data

- **Objective**: Make sure that everything is in the correct format and ready for training.

- **Action**:
    - **Check data types again**: Ensure all features are in the correct format (numerical or categorical as needed).
    - **Confirm there are no missing values**: Run the `.isnull()` method again to ensure there are no missing values after all preprocessing steps.
    - **Verify shapes**: Check that the training and testing sets have the correct dimensions and that the features match the target variable.