```python
import pandas as pd
import numpy as np

df = pd.read_csv("your_data.csv")  # Replace your_data.csv with the actual file name

# Convert 'prev_sold_date' to datetime objects, coercing errors to NaT
df['prev_sold_date'] = pd.to_datetime(df['prev_sold_date'], errors='coerce')

# --- Numerical Imputation ---

# Replace -1 with NaN for numerical columns
numerical_cols = ['price', 'bed', 'bath', 'acre_lot', 'zip_code', 'house_size']
for col in numerical_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')  # Convert to numeric, non-numeric becomes NaN
    df[col] = df[col].replace(-1, np.nan)

# Impute numerical columns using median (more robust to outliers)
for col in numerical_cols:
    df[col] = df[col].fillna(df[col].median())

# --- Categorical Imputation ---

# Replace '-' with NaN for categorical columns
categorical_cols = ['brokered_by', 'status', 'street', 'city', 'state']
for col in categorical_cols:
    df[col] = df[col].replace('-', np.nan)

# Impute categorical columns using the most frequent value (mode)
for col in categorical_cols:
    df[col] = df[col].fillna(df[col].mode()[0])  # Use mode()[0] to get the actual value


# --- Date Imputation ---

# Impute dates with the median date.  Could also use a more sophisticated approach
# if date is important for your analysis (e.g., impute based on other features).
df['prev_sold_date'] = df['prev_sold_date'].fillna(df['prev_sold_date'].median())


print(df.head())
# Save the imputed dataframe to a new CSV file
df.to_csv("imputed_data.csv", index=False)
```

**Explanation and Improvements:**

1. **Data Loading and Type Conversion:**
   - Loads the data from a CSV file (replace "your_data.csv" with your file name).
   - Converts `prev_sold_date` to datetime objects, handling errors gracefully.
   - Converts numerical columns to the correct type, replacing non-numeric values and -1 with NaN.
2. **Numerical Imputation:**
   - Uses the `median` to impute missing numerical values. Median is generally preferred over mean for imputation as it's less sensitive to outliers.
3. **Categorical Imputation:**
   - Uses the `mode` (most frequent value) to impute missing categorical values.
4. **Date Imputation:**
   - Imputes missing dates using the `median` date. For more advanced date imputation, consider using techniques like forward fill, backward fill, or interpolation if the time series aspect of the data is important.
5. **Saving the Imputed Data:**
   - Saves the imputed DataFrame to a new CSV file named "imputed_data.csv".

**Key Improvements over a Simpler Approach:**

- **Handles different data types correctly:** Imputes numerical, categorical, and date columns using appropriate methods.
- **More robust numerical imputation:** Uses median instead of mean, which is less affected by outliers.
- **Handles various missing value representations:** Replaces "-", -1, and blank strings with NaN for consistent handling.
- **Clearer and more organized code:** Makes it easier to understand and maintain.

This improved code provides a more robust and accurate imputation strategy for your dataset. Remember to install the pandas library if you haven't already: `pip install pandas`