```python
import pandas as pd

data = """brokered_by;status;price;bed;bath;acre_lot;street;city;state;zip_code;house_size;prev_sold_date
26012;sold;259900;4;2;0.13;391 Bridge Rd;Great Falls;Montana;59405;1632;2021-11-09
97679;sold;440000;3;3;0.07;82 Lincoln Ave;Charlotte;North Carolina;28210;2053;2022-03-07
59082;sold;249900;3;2;0.41;672 West Pl;Stuarts Draft;Virginia;24477;1827;2022-02-14
969817;sold;42500;3;2;0.6;333 Pine Ave;Alessandria;irginia;22209;20075;2022-2-3
687;sold;355000;3;2;0.19;272 Terrace Ln;Orlando;Florida;32828;1864;2021-12-08
..."""  # Rest of your data goes here

df = pd.read_csv(pd.compat.StringIO(data), sep=';')

# Deduplicate based on all columns
deduped_df = df.drop_duplicates()

# If you want to deduplicate based on a subset of columns, specify them:
# For example, to deduplicate based on street, city, state, and zip_code:
# deduped_df = df.drop_duplicates(subset=['street', 'city', 'state', 'zip_code'])

print(deduped_df)

# Save the deduplicated data to a new CSV file
deduped_df.to_csv('deduped_data.csv', index=False, sep=';')
```

Explanation:

1. **Import pandas:** We use the pandas library for data manipulation.
2. **Read the data:** The `pd.read_csv` function reads the data from the string (you can replace `pd.compat.StringIO(data)` with the actual file path if you're reading from a file). The `sep=';'` argument specifies that the semicolon is the delimiter.
3. **Deduplicate:** The `drop_duplicates()` method removes duplicate rows. By default, it considers all columns.
4. **Subset (Optional):** If you only want to deduplicate based on certain columns, use the `subset` argument and provide a list of column names. This is useful if, for example, slight variations in other fields (like `price` or `prev_sold_date`) shouldn't prevent deduplication.
5. **Print and Save:** The code prints the deduplicated DataFrame and saves it to a new CSV file named `deduped_data.csv` using the same semicolon delimiter.

Key Improvements over simply using `df.drop_duplicates()`:

- **Handles data input:** The code now correctly reads the provided data string.
- **Flexibility:** It shows how to deduplicate based on all columns or a selected subset.
- **Output:** It saves the cleaned data to a new file, which is essential for most real-world applications.
- **Clearer explanation:** The comments explain each step and the purpose of the `subset` argument.

This improved code provides a more robust and practical solution for data deduplication. Remember to replace the placeholder data string with your actual data or filepath.