Code Smells

- **Hardcoded Data**
  - In the old code, `english_words_set` was hardcoded within `password_generator.py`. This was refactored to allow more flexibility and easier updates.
- **Repetitive Code**
  - The old version repeated the character set construction logic in `generate_random_password`. This was refactored to reduce repetition.
- **Lack of Modularity**
  - Initially, all functionalities were embedded within two primary files. The refactoring introduced modular services, improving code organization and readability.
- **Bad Naming Conventions**
  - The old code had some inconsistencies in naming (like the `use_advanced`). These were made more consistent in the refactored version.
- **Limited Functionality**
  - The old version had limited functionality (like no custom character set feature). This was expanded in the refactoring process.
- **Lack of Input Validation**
  - Initially, there was no robust input validation. The refactoring introduced better validation mechanisms.
- **Global Variables**
  - The old code used global variables (like `english_words_set`). Refactoring encapsulated these within functions or classes.
- **Lack of Error Handling**
  - The initial version had minimal error handling, particularly for user input. The refactored version improved on this aspect.
- **Direct User Input Handling in Main Function**
  - The initial code handled user input directly in `main.py`. This was refactored to separate input handling from the main logic.
- **Lack of Entropy Calculation**
  - The old version did not calculate or display password entropy. This feature was added in the refactored code.

Refactoring Methods

- **Extract Method**
  - Extracting specific functionalities into separate methods/functions for better readability and reusability (like extracting the password generation logic into different methods based on password type).

- **Module Extraction**
    - Splitting the code into multiple modules (`services` directory) to enhance maintainability and modularity.
- **Rename Method/Variable**
    - Renaming methods and variables to more accurately describe their purpose and improve code readability.
- **Parameterize Method**
    - Changing methods to take parameters where appropriate, allowing more flexibility (like passing character sets as parameters).
- **Remove Magic Numbers/String**
    - Replacing hardcoded values and strings with named constants or configurable options.

Design Patterns Used

- **Factory Method**
    - Employing factory methods in `password_generator.py` to create different types of passwords based on user input.
- **Singleton**
    - The use of single instances for certain functionalities, like the English word set or the character set which ensures a single point of modification.
- **Strategy Pattern**
    - Implementing different strategies for password generation (like pronounceable vs. random) that can be switched out easily depending on user preferences.