Overview

In this project, you will develop machine learning models to predict the likelihood of breast cancer based on provided features derived from a digitized image of a fine needle aspirate (FNA, based on the Breast Cancer Wisconsin Kaggle Competition): https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data.

In addition, if you have time, you may implement a web application to host your model.

You may choose one of the following options:

1. Use our skeleton code for guidance on
2. Try implementing everything on your own
    a. If you get stuck, you can always take a look at the step-by-step instructions and skeleton code for reference, ask in the Slack or email Kevin, or come to office hours!

We recommend creating an end-to-end solution first with only rudimentary steps in order to get a baseline prediction and test the entire workflow. Then, we would recommend proceeding with the usual machine learning pipeline, beginning with exploratory data analysis, data cleaning/transformations and feature engineering, and finally model building, evaluation, and error analysis - iterating over this pipeline until you reach a satisfactory stopping point.

Tasks:

1. Data Exploration and Preprocessing

- Load Data: Import the dataset using Pandas.
- Initial Data Exploration: Use `data.head()` to view the dataset structure. Identify columns and data types.
- Handle Missing Values: Check for and handle any missing values in the dataset.
- Drop Unnecessary Columns: Specifically, remove 'ID' and 'Unnamed: 32' columns as they are not useful for analysis.
- Data Normalization/Standardization: Apply normalization or standardization to the dataset to bring all variables to a comparable scale.

2. Advanced Exploratory Data Analysis (EDA)

- Feature Distribution Analysis: Use Seaborn's `countplot` to understand the distribution of diagnoses (Malignant/Benign).
- (Optional) Advanced Visualization Techniques:
    - Utilize `violinplot` and `swarmplot` to visualize the distribution of features with respect to the diagnosis.
    - Employ `pairplot` or `PairGrid` to observe the relationship between features.
- Correlation Analysis: Use Seaborn's `heatmap` to visualize the correlation matrix of the features.

3. Feature Selection Techniques

- Correlation-Based Selection: Identify highly correlated features and retain only one feature from each set of highly correlated ones.
- (Optional) SelectKBest Method: Apply `SelectKBest` from `sklearn.feature_selection` to select top features based on a chosen statistical measure.
- (Optional) Tree-Based Feature Selection: Use feature importances from a fitted RandomForestClassifier to understand the most significant features.

(Optional) 4. Feature Extraction with PCA

- Data Preparation for PCA: Ensure data normalization/standardization is applied.
- Applying PCA: Utilize PCA from `sklearn.decomposition` and fit it to the training data.

5. Model Development, Evaluation, and Optimization

- Splitting the Data: Divide the dataset into training and testing sets.
- Model Training: Train a RandomForestClassifier using the selected features.
- Model Evaluation:
    - Assess the model using accuracy and confusion matrices.
    - Perform cross-validation and compare the scores.
- Model Optimization: Based on evaluation, refine the model. Adjust parameters or consider additional feature selection/extraction as needed.
- (Optional) Try out other models

6. Project Reporting

- Methodology Documentation: Clearly document each step taken in the analysis, from data preprocessing to model evaluation.

- Results and Insights: Provide detailed findings from the feature selection/extraction process and model performance.
- Comparative Analysis: Discuss how different feature handling techniques impacted the model's effectiveness.

Deliverables

- Executable Python Notebook: Containing all code, comments, and visualizations.
- Detailed Report: Summarizing findings, methodologies, and insights from the project.

Final Steps

Push your code to a github repository!

Additional Features

1. Deployment to a live server - (e.g Heroku, PythonAnywhere)
    a. This allows you to have the script running at all times, and for anyone on the internet to view your project!