**Code 1 (Random Forest)**

This code conducts a machine learning classification task using a Random Forest Classifier. It begins by importing necessary libraries and setting a random seed for reproducibility. Data is loaded from a CSV file and split into features (X) and labels (y). The class distribution of the labels is printed. The data is then split into training and test sets. A pipeline is constructed for data preprocessing (standardization) and classification using a Random Forest Classifier with specified hyperparameters. The classifier is trained on the training data, and learning curves are generated to visualize the model's performance. Mean and standard deviation is calculated of training and validation scores. Predictions are made on the test set, and a confusion matrix and classification report are generated to evaluate the model's performance further. Finally, these results are visualized using matplotlib and seaborn.

**Code 2 (Random Forest probability)**

This code trains a Random Forest Classifier on a dataset loaded from a CSV file. After loading the data and separating features and labels, a classifier pipeline is constructed, including standardization of features. The classifier is trained on the entire dataset, and then saved to a file using joblib for later use. It is then loaded back into memory from the saved file. Finally, the trained classifier is used to predict probabilities for each instance in the dataset, specifically for the class label representing melanoma, and these probabilities are printed.

**Code 3 (KNN)**

This code implements a classification task using the K-Nearest Neighbors (KNN) algorithm. It starts by importing necessary libraries and setting a random seed for reproducibility. Data is loaded from a CSV file and split into features and labels. The class distribution is then printed. Next, the data is split into training and test sets. A pipeline is constructed for data preprocessing (standardization) and classification using KNN with specified hyperparameters. The classifier is trained on the training data, and learning curves are generated to visualize the model's performance. Predictions are made on the test set, and a confusion matrix and classification report are generated to evaluate the model's performance further. Finally, these results are visualized using matplotlib and seaborn.

**Code 4 (KNN probability)**  
This code trains a K-Nearest Neighbors (KNN) classifier on a dataset loaded from a CSV file. After loading the data and separating features and labels, a classifier pipeline is constructed, including standardization of features. The classifier is trained on the entire dataset and then saved to a file using joblib for later use. It is then loaded back into memory from the saved file. Finally, the trained classifier is used to predict probabilities for each instance in the dataset, specifically for the class label representing melanoma, and these probabilities are printed.

**Code 5 (Decision tree)**

This code performs a classification task using a Decision Tree Classifier. It begins by importing necessary libraries and setting a random seed for reproducibility. Data is loaded from a CSV file and split into features and labels. The class distribution is printed. The data is then split into training and test sets. A pipeline is constructed for data preprocessing (standardization) and classification using a Decision Tree Classifier with specified hyperparameters. The classifier is trained on the training data, and learning curves are generated to visualize the model's performance. Predictions are made on the test set, and a confusion matrix and classification report are generated to evaluate the model's performance further. Finally, these results are visualized using matplotlib and seaborn.

**Code 6 (Decision tree)**

This code trains a Decision Tree Classifier on a dataset loaded from a CSV file. After loading the data and separating features and labels, a classifier pipeline is constructed, including standardization of features. The classifier is trained on the entire dataset and then saved to a file using joblib for later use. It is then loaded back into memory from the saved file. Finally, the trained classifier is used to predict probabilities for each instance in the dataset, specifically for the class label representing melanoma, and these probabilities are printed.