**README FOR PROJECT-1**

**import numpy as np:** This statement imports the NumPy library, which is widely used for numerical computing in Python. It's a common convention to import NumPy as np to make it shorter and easier to reference in the code. NumPy provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.   
In the code, NumPy is used for various array operations, such as reshaping arrays: np.array() and reshape() functions are used to reshape data into appropriate formats for further processing. Array operations: Various mathematical operations and array manipulations are performed using NumPy functions.

**import pandas as pd :** This statement imports the Pandas library, which provides high-performance data manipulation and analysis tools for Python. Pandas is particularly useful for working with structured data such as tabular data (e.g., CSV files, Excel spreadsheets). Similar to the NumPy import, as pd creates an alias pd for the Pandas module.  
In the code it is mainly used in loading and handling data: pd.DataFrame() is used to create DataFrames from arrays, and pd.concat() may be used for concatenating DataFrames.

Manipulating data: Column selection (iloc[]), adding new columns, and accessing DataFrame elements.

**from sklearn.metrics import accuracy\_score, confusion\_matrix, precision\_score, recall\_score**:   
This statement imports specific functions (accuracy\_score, confusion\_matrix, precision\_score, recall\_score) from the sklearn.metrics module. sklearn (scikit-learn) is a popular machine learning library in Python, and these functions are commonly used to evaluate the performance of machine learning models.   
In code specifically, used in Evaluating model performance: After making predictions, these functions are used to calculate accuracy, precision, recall, and confusion matrices based on the predicted and true labels.

**from sklearn.model\_selection import KFold:** This statement imports the KFold class from the sklearn.model\_selection module. KFold is a method for cross-validation, which is a technique used to assess how well a predictive model generalizes to an independent dataset. It splits the dataset into 'k' consecutive folds, then uses 'k-1' folds for training and the remaining fold for testing, repeating the process 'k' times. KFold is imported for performing k-fold cross-validation.  
In code it specifically used in splitting data: KFold.split() is used to generate train/test indices for k-fold cross-validation.

**from data\_processor import DataProcessor:**

This statement imports the DataProcessor class from a module named data\_processor. This module is a custom module defined in the project, and it contains functions or classes related to processing data. A custom class or module named DataProcessor is imported from data\_processor.py.  
In code it Specifically used in data preprocessing methods from DataProcessor are used for rotating and translating data, which likely involves some geometric transformations or manipulations.

**import os:** This statement imports the os module, which provides a portable way of using operating system-dependent functionality. In code, the os module is primarily used for interacting with the file system. It helps in constructing file paths and checking for the existence of directories before processing their contents.   
In the code, within the DataProcessor class, os.path.join() is used to create file paths by joining directory paths and file names. Additionally, os.path.exists() is employed to ensure that a directory exists before attempting to read its contents. This ensures robustness in handling directory structures and prevents potential errors that may occur if directories are missing.

**import glob:** This statement imports the glob module, which provides a way to search for files whose names match a specified pattern. In this code, glob.glob() is not explicitly used, but it could be used to search for files matching a certain pattern in a directory.

**import sys:** This statement imports the sys module, which provides access to some variables used or maintained by the Python interpreter and to functions that interact strongly with the interpreter.   
In this code, sys.argv is used to access command-line arguments passed to the script when it is executed. This allows the script to take input parameters from the command line.

**from classifier\_trainer import ClassifierTrainer:** This statement imports the ClassifierTrainer class from the classifier\_trainer module. The ClassifierTrainer class contains methods for training classifiers and evaluating their performance. It is used in the script to train different classifiers based on the provided algorithm.

**from sklearn.ensemble import RandomForestClassifier:** This statement imports the RandomForestClassifier class from the sklearn.ensemble module. RandomForestClassifier is a machine learning model implemented in scikit-learn, which is a popular library for machine learning in Python. It is one of the classifiers that can be used in the script depending on the specified algorithm.

**from sklearn.svm import SVC :** This statement imports the SVC class from the sklearn.svm module. SVC stands for Support Vector Classifier, which is a type of Support Vector Machine (SVM) classifier. It is another classifier option available in scikit-learn and can be used based on the specified algorithm in the script.

**from sklearn.tree import DecisionTreeClassifier:** This statement imports the DecisionTreeClassifier class from the sklearn.tree module. DecisionTreeClassifier is a classifier based on decision tree algorithms. Like the other classifiers, it is imported as an option for training based on the specified algorithm.

The scikit-learn classifiers (RandomForestClassifier, SVC, DecisionTreeClassifier) are imported and stored in a dictionary called classifiers. Each classifier class serves as a value in this dictionary, with the algorithm names (RF, SVM, DT) serving as keys.

classifiers = {'RF': RandomForestClassifier, 'SVM': SVC, 'DT': DecisionTreeClassifier}: This dictionary maps algorithm names to their corresponding scikit-learn classifier classes.

ClassifierTrainer.train\_classifier(classifiers[algo], x\_data, y\_data.values.ravel()): This line of code selects the appropriate classifier class based on the specified algorithm (algo) and trains the classifier using the processed features (x\_data) and labels (y\_data)

**import matplotlib.pyplot as plt:** This statement imports the matplotlib.pyplot module, which provides a MATLAB-like plotting framework. It is typically used for creating static, interactive, and animated visualizations in Python.

**from mpl\_toolkits.mplot3d import Axes3D:** This statement imports the Axes3D class from the mpl\_toolkits.mplot3d module. The mpl\_toolkits. mplot3d module provides tools for creating 3D plots in Matplotlib. The Axes3D class specifically allows for the creation of 3D axes in Matplotlib figures, enabling the visualization of three-dimensional data.