**Cross validation Function (evaluate\_cv\_model):**

* **Arguments**:
  + - model (an object from the classifier)
    - data
    - target
    - kFolds
* **Return**:
  + - accuracy (accuracy of the model)
* **Description:**

Train the given model using the required number of Folds and calculate the accuracy.

**Confusion Matrix & Classification Report Function (plot\_confusionMatrix)**:

* **Arguments**:
  + - clf (an object from the classifier)
    - y\_test (testing targets)
    - X\_test (testing data)
* **Return**:
  + - report (Classification Report)
* **Description:**

This function can be used to plot the confusion matrix of the model and returns the classification report.

**KNN Function:**

* **Arguments**:
  + - X\_train (training data)
    - y\_train (training tragets)
    - X\_test (testing data)
    - y\_test (testing targets)
    - num\_neighbors (number of neighbors parameter)
* **Return**:
  + - test\_accuracy (accuracy of the testing data)
    - train\_accuracy (accuracy of the training data)
    - prediction (values of prediction by the model)
    - KNN (object from the classifier)
* **Description:**

Create the model by taking an object from **KNeighborsClassifier** with the required number of neighbors**,** then train the model using the training data and targets, then calculate the accuracy of both the training and testing, then get the predicted values of the testing data using the model.

Finally return the needed value.

**SVM Function:**

* **Arguments**:
  + - X\_train (training data)
    - y\_train (training tragets)
    - X\_test (testing data)
    - y\_test (testing targets)
    - kernel (the type of the kernel)
* **Return**:
  + - test\_accuracy (accuracy of the testing data)
    - train\_accuracy (accuracy of the training data)
    - prediction (values of prediction by the model)
    - SVM (object from the classifier)
* **Description:**

Create the model by taking an object from **svm.SVC** with the required kernel and using function shape “ovo” to be used for multiclass classification**,** then train the model using the training data and targets, then calculate the accuracy of both the training and testing, then get the predicted values of the testing data using the model.

Finally return the needed values.

**Logisitc\_Regression Function:**

* **Arguments**:
  + - X\_train (training data)
    - y\_train (training tragets)
    - X\_test (testing data)
    - y\_test (testing targets)
* **Return**:
  + - test\_accuracy (accuracy of the testing data)
    - train\_accuracy (accuracy of the training data)
    - prediction (values of prediction by the model)
    - log\_reg (object from the classifier)
* **Description:**

Create the model by taking an object from **LogisticRegression** with “multinomial” multi\_class parameter and “lbfgs” solver parameter**,** then train the model using the training data and targets, then calculate the accuracy of both the training and testing, then get the predicted values of the testing data using the model.

Finally return the needed values.

**Data section:**

* Read ECG, EEG, EEG+EEG selected features files in 3 data frames.
* Drop the unnecessary columns from each dataframe.
* Specify the data and target for each dataframe.
* Get number of **Kfolds** to be used for cross validation.
* Split the data using **train\_test\_split()**.

**KNN model section:**

* **EEG Data**
* Training multiple models with different number of neighbors and visualizing the output and then choose the best **k** value.
* Evaluate the model without cross validation using the **KNN** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.
* **ECG Data**
* Training multiple models with different number of neighbors and visualizing the output and then choose the best **k** value.
* Evaluate the model without cross validation using the **KNN** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.
* **ECG + EEG Data**
* Training multiple models with different number of neighbors and visualizing the output and then choose the best **k** value.
* Evaluate the model without cross validation using the **KNN** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.

**SVM models section:**

* **EEG Data (for each kernel type we make the same steps)**
* Train the SVM model with the required kernel type using **SVM** function.
* Evaluate the model without cross validation using the **SVM** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.
* **ECG Data (for each kernel type we make the same steps)**
* Train the SVM model with the required kernel type using SVM function.
* Evaluate the model without cross validation using the **SVM** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.
* **ECG + EEG Data (for each kernel type we make the same steps)**
* Train the SVM model with the required kernel type using SVM function.
* Evaluate the model without cross validation using the **SVM** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.

**Logistic Regression model section:**

* **EEG Data**
* Train the model using **Logistic\_Regression** function.
* Evaluate the model without cross validation using the **Logistic\_Regression** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.
* **ECG Data**
* Train the model using **Logistic\_Regression** function.
* Evaluate the model without cross validation using the **Logistic\_Regression** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.
* **ECG + EEG Data**
* Train the model using **Logistic\_Regression** function.
* Evaluate the model without cross validation using the **Logistic\_Regression** function and print the test and train accuracy.
* Evaluate the model with cross validation using **evaluate\_cv\_model** function and print the accuracy.
* Plot confusion matrix using **plot\_confussionMatrix** function.
* Display classification report returned from **plot\_confussionMatrix** function.