Dataset Details

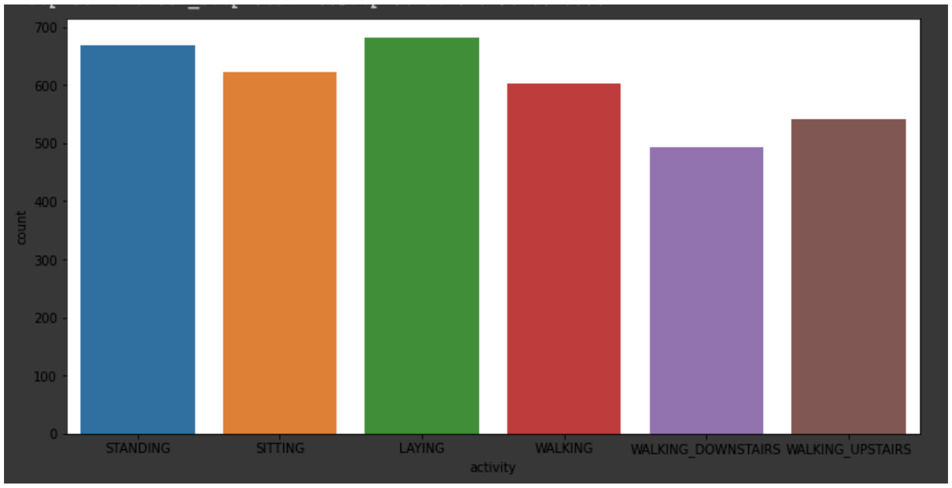
Human Activity Recognition
For each record in the dataset these details are provided:
Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.Triaxial Angular velocity from the gyroscope. Dataset contains 561-feature vector with time and frequency domain variables. Its activity label and an identifier of the subject who carried out the experiment. This dataset contains six labels which are Each person performed six activities walking, walking*upstairs, walking* downstairs, sitting, standing and laying.

Digit Recognizer
Each image has a height of 28 pixels and a width of 28 pixels, for a total of 784 pixel. There is a single pixel-value correlated with each pixel, showing the pixel's lightness or darkness, with higher numbers suggesting darker. This pixel-value is an integer ranging from 0 to 255, inclusive.The data set for training, has 785 columns. The first column is the digit that was drawn by the individual, labeled "label". The remainder of the columns contain the corresponding picture pixel values.This dataset consists of 42000 rows and 784 column of pixel.

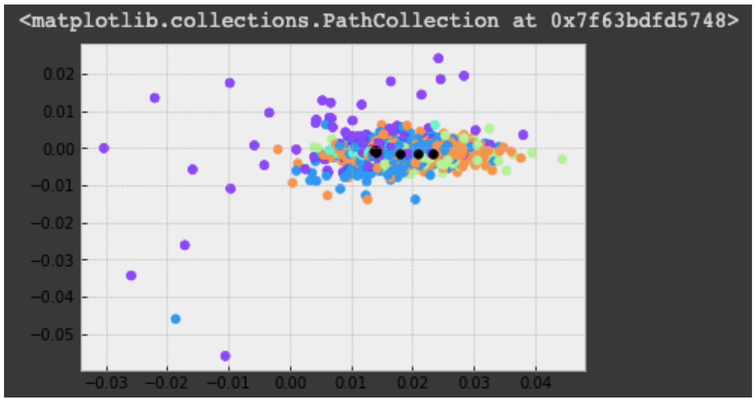
Data Visualization

Human Activity Recognition

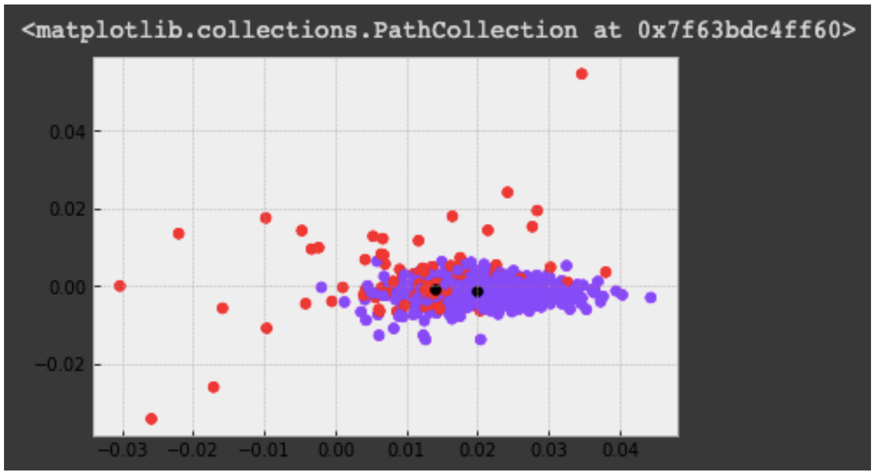
This graph shows the activity vs count variable that is how much data we have for each category



Scatter plot with 6 centroids

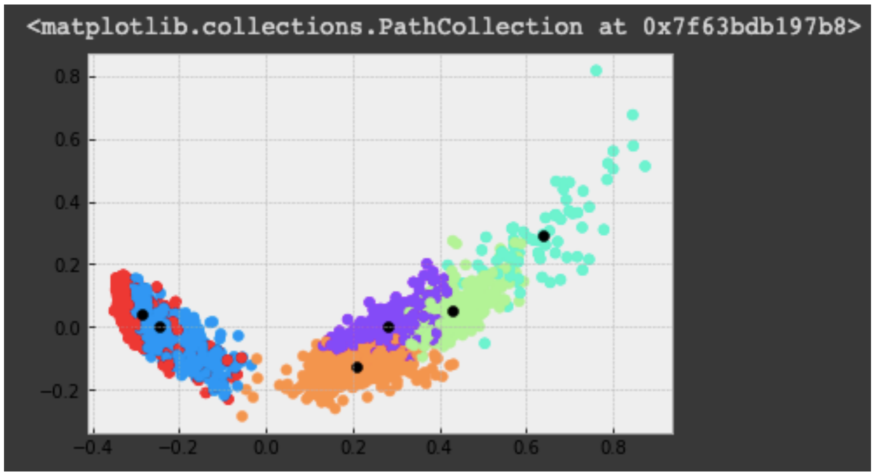

```
<matplotlib.collections.PathCollection at 0x7f63bdfd5748>
```

Scatter plot with 2 centroids



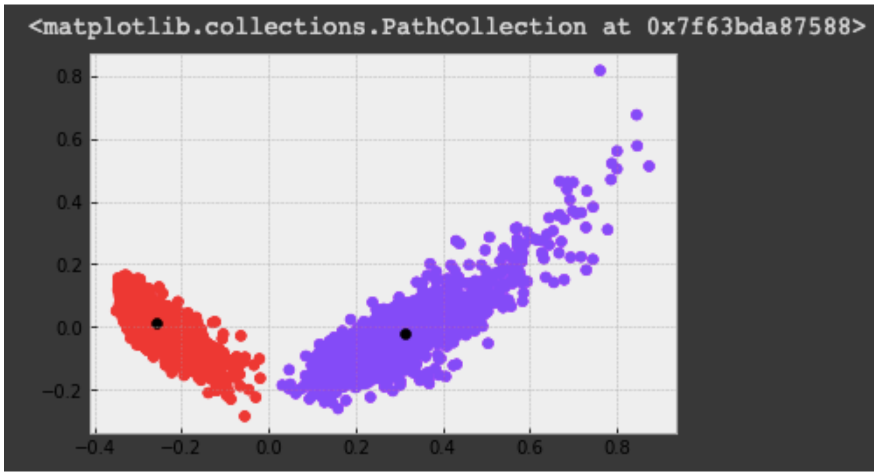
```
<matplotlib.collections.PathCollection at 0x7f63bdc4ff60>
```

Scatter plot after PCA with 6 centroids



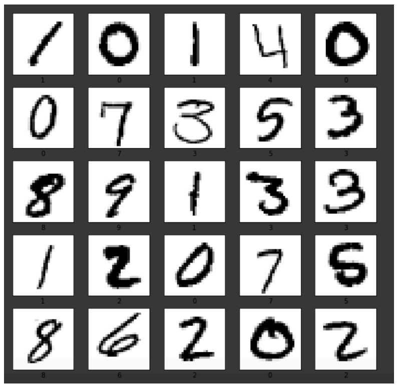
```
<matplotlib.collections.PathCollection at 0x7f63bdb197b8>
```

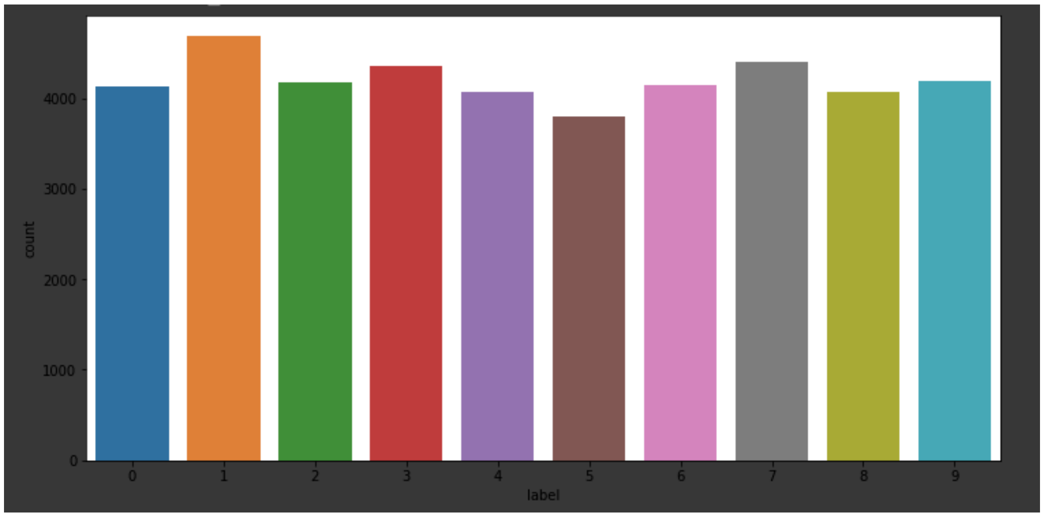
Scatter plot after PCA with 2 centroids



```
<matplotlib.collections.PathCollection at 0x7f63bda87588>
```

Digit Recognizer





This dataset consists of handwritten digits from 1-9.

Train-Test Split of Data

The train-test split is a technique for testing a machine learning algorithm's output. This involves a dataset being taken and divided into two parts. First part is used to train/fit the model and is referred to as the training dataset. The second part is not used to train the algorithm instead, it is used to give input element of the dataset, then predictions are made and the predicted values are compared. This second dataset is known as test dataset.

In this dataset I have choose a 20% split percentage. That is 80% would be my training set and 20% would be my test set. I have taken this split by considering several parameters like computational cost in both training and testing the model. As well as representativeness of training set.

Algorithm Description

Feature Scaling

Data Normalization

Normalization refers to rescaling real valued numeric attributes into the range 0 and 1.It is useful to scale the input attributes for a model that relies on the magnitude of values, such as distance measures used in k-nearest neighbors and in the preparation of coefficients in regression.

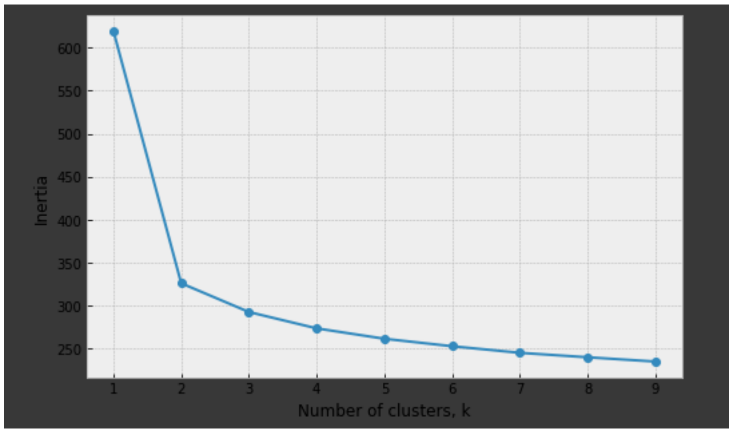
KMeans Algorithm
Conventional *k*-means requires only a few steps. The first step is to randomly select *k* centroids, where *k* is equal to the number of clusters you choose. **Centroids** are data points representing the center of a cluster.

The main element of the algorithm works by a two-step process called **expectation-maximization**.
The **expectation** step assigns each data point to its nearest centroid. Then, the **maximization** step computes the mean of all the points for each cluster and sets the new centroid.

```
Algorithm 1 k-means algorithm
1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: repeat
4:     expectation: Assign each point to its closest centroid.
5:     maximization: Compute the new centroid (mean) of each cluster.
6: until The centroid positions do not change.
```

The quality of the cluster assignments is determined by computing the sum of the squared error (SSE) after the centroids converge, or match the previous iteration's assignment. The SSE is defined as the sum of the squared Euclidean distances of each point to its closest centroid. Since this is a measure of error, the objective of *k*-means is to try to minimize this value.

Elbow Method
Elbow method gives us an idea on what a good *k* number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids. We pick *k* at the spot where SSE starts to flatten out and forming an elbow. We'll use the geyser dataset and evaluate SSE for different values of *k* and see where the curve might form an elbow and flatten out. The graph below shows that k=2 is good choice.



Principal Component Analysis
Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation which converts a set of correlated variables to a set of uncorrelated variables. PCA is a most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

Principal Component analysis also known as PCA is such a feature extraction method where we create new independent features from the old features and from combination of both keep only those features that are most important in predicting the target. New features are extracted from old features and any feature can be dropped that is considered to be less dependent on the target variable.

PCA is such a technique which groups the different variables in a way that we can drop the least important feature. All the features that are created are independent of each other.

Principal Component Analysis steps

- Initially start with standardization of data.
- Create a correlation matrix or covariance matrix for all the desired dimensions.
- Calculate eigenvectors that are the principal component and respective eigenvalues that apprehend the magnitude of variance.

- Arrange the eigen pairs in decreasing order of respective eigenvalues and pick the value which has the maximum value, this is the first principal component that protects the maximum information from the original data.

KNN Algorithm
In K, meaning algorithm, we would look at the K nearest training data points for each test data point and take the most frequently occurring classes and assign the test data to that class. K reflects the sum of training data points that lie near the evaluation data point that we are going to use to locate the class.

```
1. Load the training and test data
2. Choose the value of K
3. For each point in test data:
    - find the distance to all training data points
    - store the distances in a list and sort it
    - choose the first k points
    - assign a class to the test point based on the majority of classes present in the chosen points
4. End
```

Random Forest Algorithm
Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.
The Working process can be explained in the below steps and diagram:
Step-1: Select random K data points from the training set.
Step-2: Build the decision trees associated with the selected data points (Subsets).
Step-3: Choose the number N for decision trees that you want to build.
Step-4: Repeat Step 1 & 2.
Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

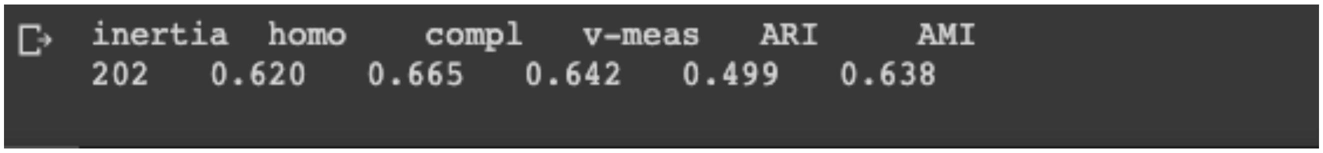
Algorithm Results

Human Activity Recognition

For this dataset I have performed with both the possibilities having all 6 labels and 2 labels. I have categorised standing, sitting or laying in non-moving label (0 – binary value) and rest in moving lable (1 – binary value).

Kmeans

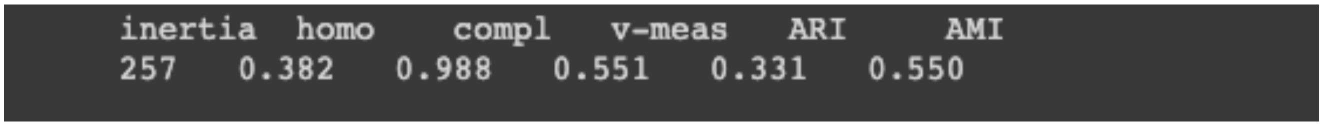
Results when there are **6** clusters



```
inertia  homo    compl   v-meas   ARI     AMI
202      0.620   0.665   0.642    0.499   0.638
```

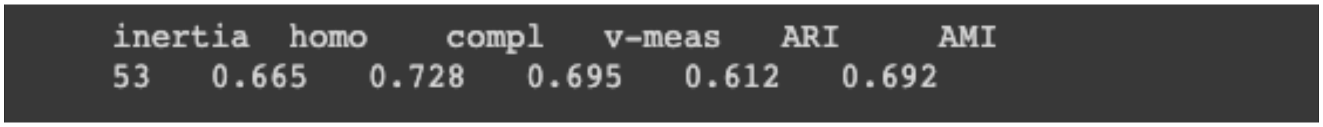
Runtime of the program is **1.69 s**.

Results when there are **2** clusters



```
inertia  homo    compl   v-meas   ARI     AMI
257      0.382   0.988   0.551    0.331   0.550
```

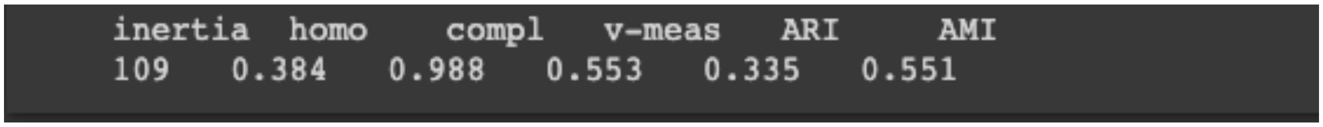
Results after performing **PCA** and then applying Kmeans on the processed data with **6** clusters



```
inertia  homo    compl   v-meas   ARI     AMI
53   0.665   0.728   0.695   0.612   0.692
```

Runtime of the program is **0.20 s**.

Results after performing **PCA** and then applying Kmeans on the processed data with **2** clusters



```
inertia  homo    compl   v-meas   ARI     AMI
109   0.384   0.988   0.553   0.335   0.551
```

Homogeneity Score
This score is useful to check whether the clustering algorithm meets an important requirement: a cluster should contain only samples belonging to a single class.

Completeness Score
This score is complementary to the previous one. Its purpose is to provide a piece of information about the assignment of samples belonging to the same class. More precisely, a good clustering algorithm should assign all samples with the same true label to the same cluster.

V-measure
The V-Measure is defined as the harmonic mean of homogeneity h and completeness c of the clustering. Both these measures can be expressed in terms of the mutual information and entropy measures of the information theory.

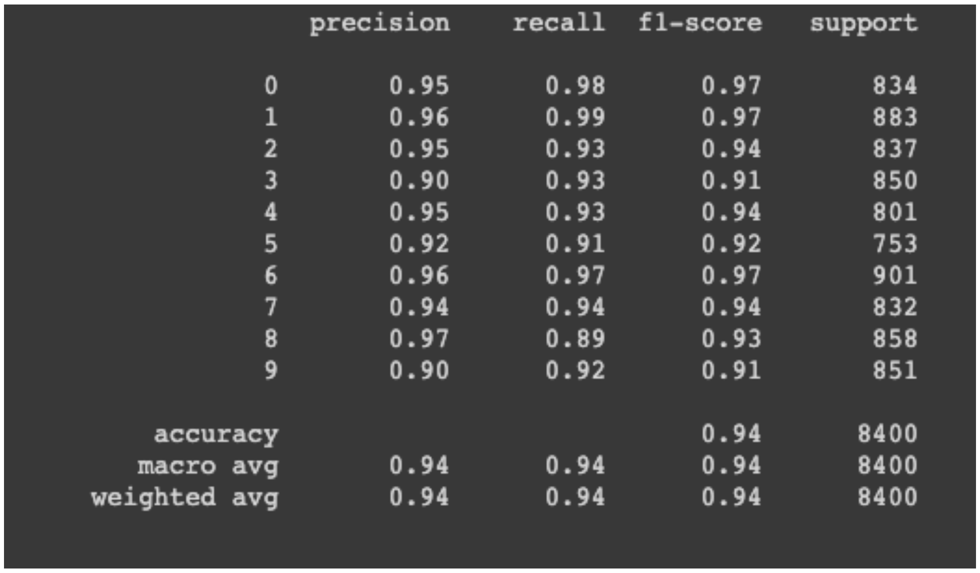
Rand Index
The Rand Index computes a similarity measure between two clustering by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering.

The adjusted Rand index is thus ensured to have a value close to 0.0 for random labeling independently of the number of clusters and samples and exactly 1.0 when the clusterings are identical (up to a permutation).

When we are having 2 features/components only then the completeness score is really high. Which shows that the classification/assignment to cluster is better when only 2 component (moving/not moving) are considered.

Digit Recognizer

KNN Algorithm

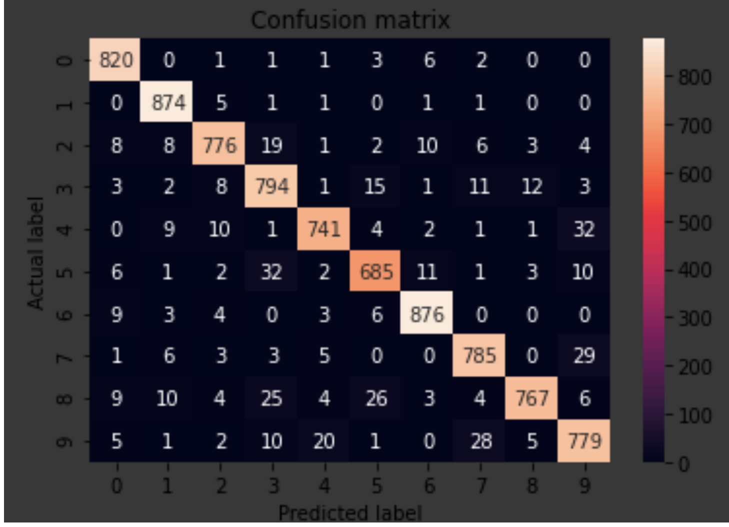


```
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       834
           1       0.96      0.99      0.97       883
           2       0.95      0.93      0.94       837
           3       0.90      0.93      0.91       850
           4       0.95      0.93      0.94       801
           5       0.92      0.91      0.92       753
           6       0.96      0.97      0.97       901
           7       0.94      0.94      0.94       832
           8       0.97      0.89      0.93       858
           9       0.90      0.92      0.91       851

    accuracy                           0.94      8400
   macro avg       0.94      0.94      0.94      8400
weighted avg       0.94      0.94      0.94      8400
```

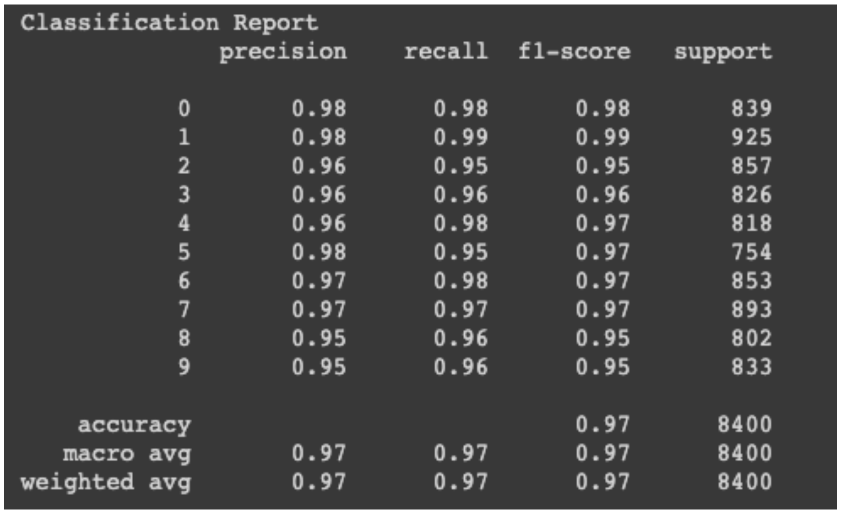
```
Confusion Matrix
Text(0.5, 15.0, 'Predicted label')
```



	0	1	2	3	4	5	6	7	8	9
0	820	0	1	1	1	3	6	2	0	0
1	0	874	5	1	1	0	1	1	0	0
2	8	8	776	19	1	2	10	6	3	4
3	3	2	8	794	1	15	1	11	12	3
4	0	9	10	1	741	4	2	1	1	32
5	6	1	2	32	2	685	11	1	3	10
6	9	3	4	0	3	6	876	0	0	0
7	1	6	3	3	5	0	0	785	0	29
8	9	10	4	25	4	26	3	4	767	6
9	5	1	2	10	20	1	0	28	5	779

Runtime of the program is **6.41 s**.

Random Forest Algorithm

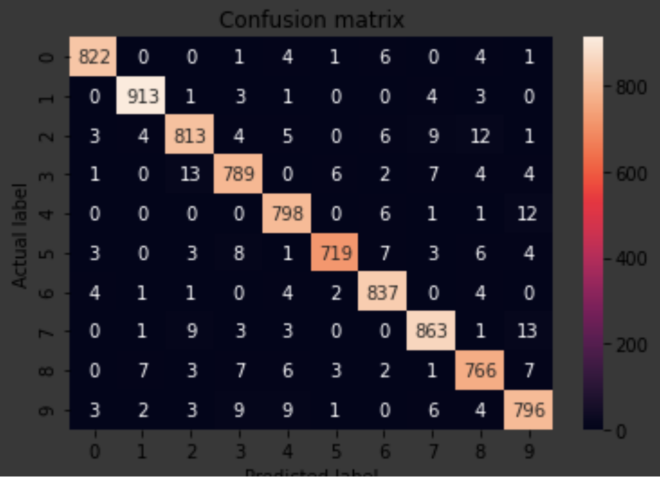


```
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       839
           1       0.98      0.99      0.99       925
           2       0.96      0.95      0.95       857
           3       0.96      0.96      0.96       826
           4       0.96      0.98      0.97       818
           5       0.98      0.95      0.97       754
           6       0.97      0.98      0.97       853
           7       0.97      0.97      0.97       893
           8       0.95      0.96      0.95       802
           9       0.95      0.96      0.95       833

    accuracy                           0.97      8400
   macro avg       0.97      0.97      0.97      8400
weighted avg       0.97      0.97      0.97      8400
```

```
Confusion Matrix
Text(0.5, 15.0, 'Predicted label')
```



	0	1	2	3	4	5	6	7	8	9
0	822	0	0	1	4	1	6	0	4	1
1	0	913	1	3	1	0	0	4	3	0
2	3	4	813	4	5	0	6	9	12	1
3	1	0	13	789	0	6	2	7	4	4
4	0	0	0	0	798	0	6	1	1	12
5	3	0	3	8	1	719	7	3	6	4
6	4	1	1	0	4	2	837	0	4	0
7	0	1	9	3	3	0	0	863	1	13
8	0	7	3	7	6	3	2	1	766	7
9	3	2	3	9	9	1	0	6	4	796

Runtime of the program is **22.99 s**.