MLND Capstone

Tunesh verma

Definition

Project Overview

With the rapid advances in micro electro-mechanical systems (MEMS) technology, the size, weight and cost of commercially available inertial sensors have decreased considerably over the last two decades. A recent application domain of inertial sensing is automatic recognition and monitoring of human activities.

Miniature sensor units that contain accelerometers and gyroscopes are sometimes complemented by magnetometers. Gyroscopes provide angular rate information around an axis of sensitivity, whereas accelerometers provide linear or angular velocity rate information. There exist devices sensitive around a single axis, as well as two- and triaxial devices. Tri-axial magnetometers can detect the strength and direction of the Earth's magnetic field as a vector quantity. The main advantages of inertial sensors is that they are self-contained, non-radiating, non-jammable devices that provide dynamic motion information through direct measurements in 3D.
A recent application domain of inertial sensing is automatic recognition and monitoring of human activities. This is a research area with many challenges and one that has received tremendous interest, especially over the last decade. Several different approaches are commonly used for the purpose of activity monitoring. One such approach is to use sensing systems fixed to the environment, such as vision systems with multiple video cameras.

http://www.springer.com/cda/content/document/cda_downloaddocument/9781447141198-c2.pdf?SGWID=0-0-45-1332403-p174315570

These sensors are attached on body parts and certain activities are performed, data of the sensor signals are collected which can be used to predict activities. In this project, a multi-classifier was used that recognize daily and sports activities done by a person with the help of signals from inertia sensors dataset.

Problem Statement

Automatic recognition, representation and analysis of human activities based on video images has had a high impact on security and surveillance, entertainment and personal archiving applications.

Using cameras fixed to the environment (or other ambient intelligence solutions) may be acceptable when activities are confined to certain parts of an indoor environment. If cameras are being used, the environment needs to be well illuminated and almost studio like.

However, when activities are performed indoors and outdoors and involve going from place to place (e.g. commuting, shopping, jogging), fixed camera systems are not very practical because acquiring video data is difficult for long-term human motion analysis in such unconstrained environments. Recently, wearable camera systems have been proposed to overcome this problem; however, the other disadvantages of camera systems still exist, such as occlusion effects, the correspondence problem, the high cost of processing and storing images, the need for using multiple camera projections from 3D to 2D, the need for camera calibration and cameras' intrusions on privacy.

Using miniature inertial sensors that can be worn on the human body instead of employing sensor systems fixed to the environment has certain advantages. As activity can best be measured where it occurs.' Unlike visual motion capture systems that require a free line of sight, miniature inertial sensors can be flexibly used inside or behind objects without occlusion.

1D signals acquired from the multiple axes of inertial sensors can directly provide the required information in 3D. Wearable systems have the advantages of being with the user continuously and having low computation and power requirements.

In this project, multi-classifier is used that recognize daily and sports activities done by a person with the help of signals from inertia sensors dataset. Inputs are the signal data from the sensors and output is the activity.

After labeling and loading the data in notebook data will be split into training, test and validation set then per processing will be done with the help of MinMax scaler and label will be encoded. Data will be visualized to get some information of the data. In the end different algorithm will be using and results will be obtained, algorithm with the best results will be chosen and further it parameter will be tuned to improve the results.

Metrics

Evaluation is done by Confusion Metrix which is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

It is often convenient to combine precision and recall into a single metric called the F1 score, in particular if you need a simple way to compare two classifiers. The F1 score is the harmonic mean of precision and recall.

```
F1 = 2 * (precision * recall) / (precision + recall)
```

In the multi-class and multi-label case, this is the weighted average of the F1 score of each class.

In sports and daily activity dataset labels are uniformly distributed and f1_score is best to use when our class labels are uniformly distributed.

The **accuracy_score** function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions.

In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the fraction of correct predictions over n_{samples} is defined as

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Analysis

Data Exploration

Each of the 19 activities is performed by eight subjects (4 female, 4 male, between the ages 20 and 30) for 5 minutes. Total signal duration is 5 minutes for each activity of each subject. The subjects are asked to perform the activities in their own style and were not restricted on how the activities should be performed. For this reason, there are inter-subject variations in the speeds and amplitudes of some activities.

The activities are performed at the Bilkent University Sports Hall, in the Electrical and Electronics Engineering Building, and in a flat outdoor area on campus. Sensor units are calibrated to acquire data at 25 Hz sampling frequency. The 5-min signals are divided into 5-sec segments so that 480(=60x8) signal segments are obtained for each activity.

The 19 activities are:

- Sitting (A1),
- Standing (A2),
- Lying on back and on right side (A3 and A4),
- Ascending and descending stairs (A5 and A6),
- Standing in an elevator still (A7)
- Moving around in an elevator (A8),
- Walking in a parking lot (A9),
- Walking on a treadmill with a speed of 4 km/h (in flat and 15 degree inclined positions) (A10 and A11),

- Running on a treadmill with a speed of 8 km/h (A12),
- Exercising on a stepper (A13),
- Exercising on a cross trainer (A14),
- Cycling on an exercise bike in horizontal and vertical positions (A15 and A16),
- Rowing (A17),
- Jumping (A18),
- Playing basketball (A19).

File structure:

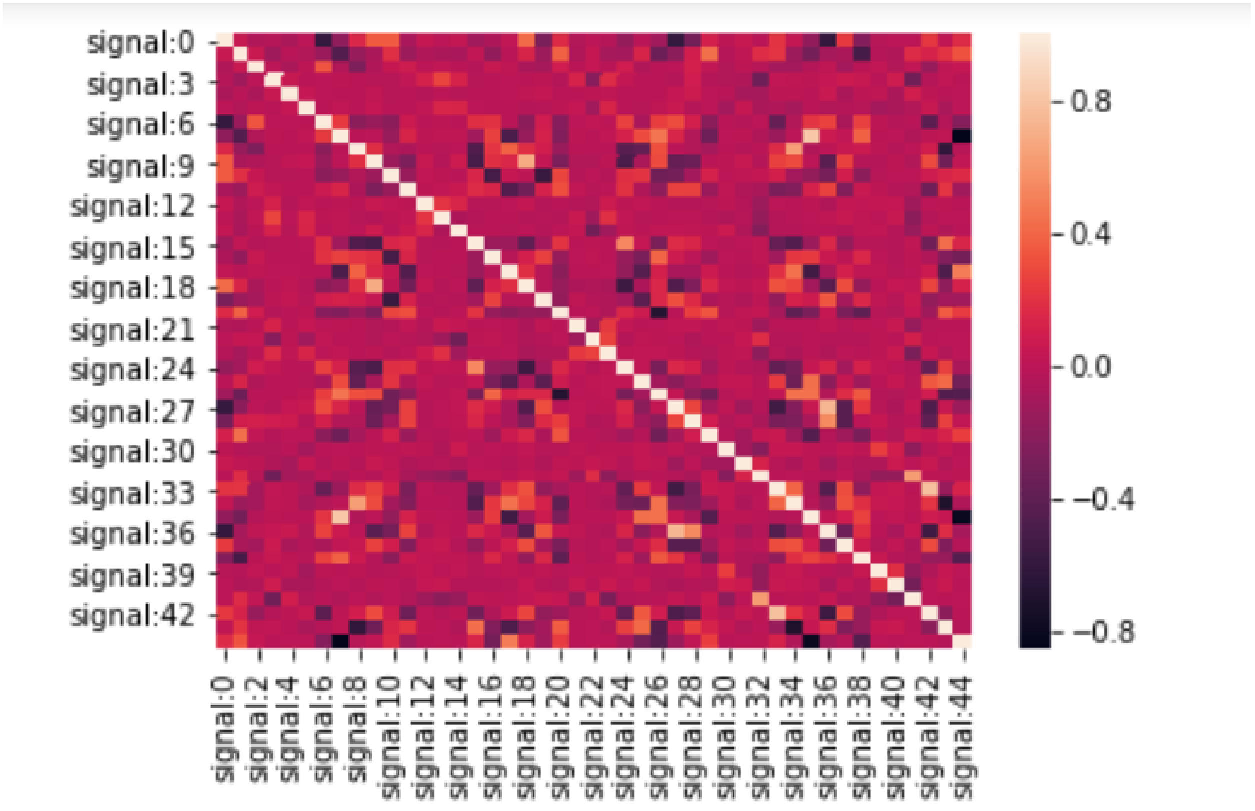
- 19 activities (a) (in the order given above)
- 8 subjects (p)
- 60 segments (s)
- 5 units on torso (T), right arm (RA), left arm (LA), right leg (RL), left leg (LL)
- 9 sensors on each unit (x, y, z accelerometers, x, y, z gyroscopes, x, y, z magnetometers)
- Folders a01, a02, ..., a19 contain data recorded from the 19 activities.
- For each activity, the subfolders p1, p2, ..., p8 contain data from each of the 8 subjects.
- In each subfolder, there are 60 text files s01, s02, ..., s60, one for each segment.
- In each text file, there are 5 units x 9 sensors = 45 columns and 5 sec x 25 Hz = 125 rows.
- Each column contains the 125 samples of data acquired from one of the sensors of one of the units over a period of 5 sec.
- Each row contains data acquired from all of the 45 sensor axes at a particular sampling instant separated by commas.

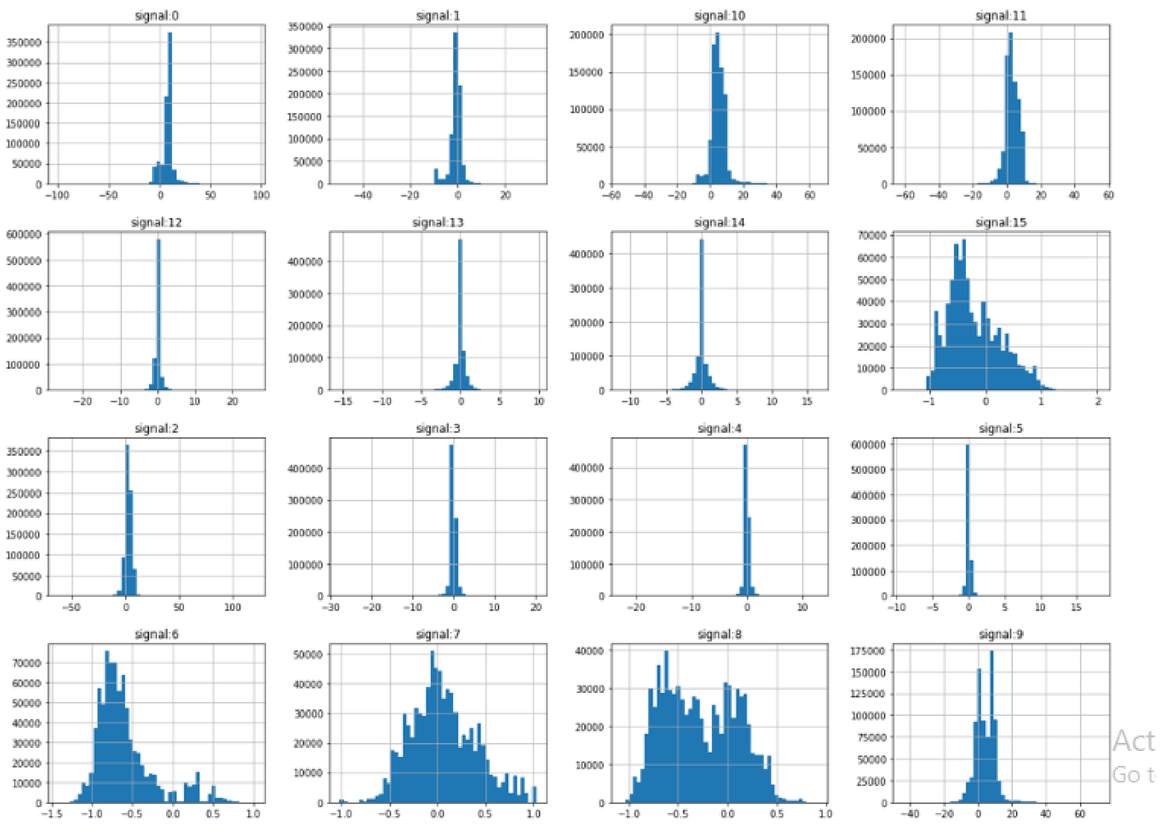
Download dataset from: -
https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities

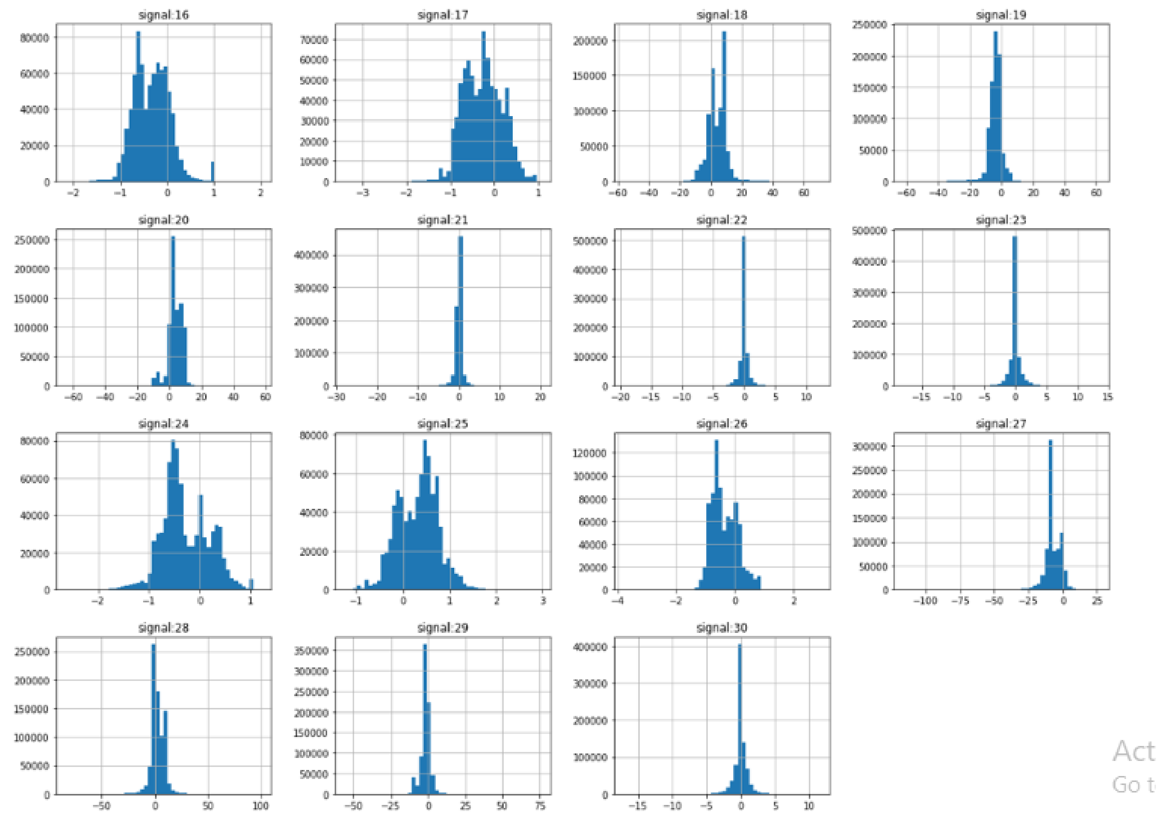
Visualization of Data

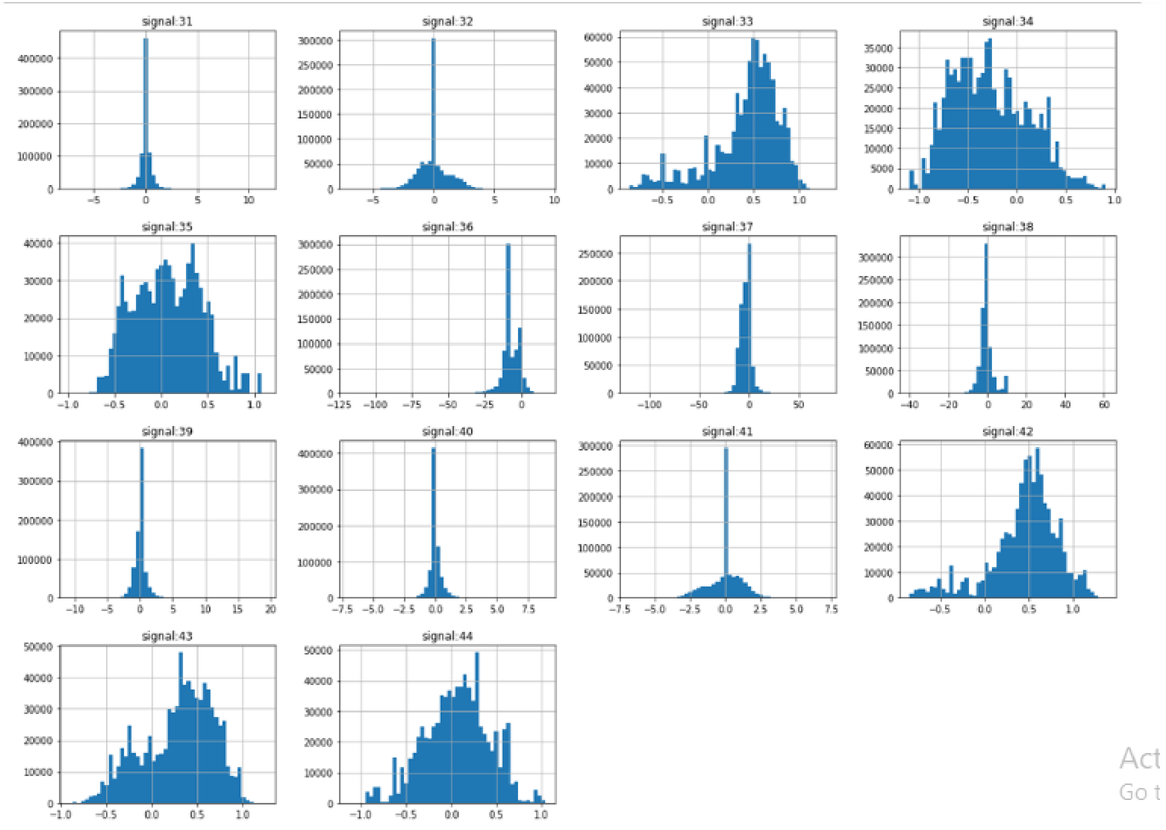
I plotted the data using matplotlib. Plotted the data using histogram plot to get information about the frequency of features. Plotted the scatter matrix plot to know about the correlation between the features.

This is correlation heat map




signal:0
signal:1
signal:10
signal:11
signal:12
signal:13
signal:14
signal:15
signal:2
signal:3
signal:4
signal:5
signal:6
signal:7
signal:8
signal:9
Act
Go t


signal:16
signal:17
signal:18
signal:19
signal:20
signal:21
signal:22
signal:23
signal:24
signal:25
signal:26
signal:27
signal:28
signal:29
signal:30
Act
Go t



Few features are right/left skewed. Some features are mostly zero because it might be an action with no movement and hence no sensor information.

Algorithm and Results

Different algorithms used to train data are as follows:-

1. KNeighborsClassifier

Neighbors-based classification is a type of *instance-based learning* or *non-generalizing learning*: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

The k-neighbors classification in **KNeighborsClassifier** is the more commonly used of the two techniques. The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

The basic nearest neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit. This can be accomplished through the weight keyword. The default value, weights = 'uniform', assigns uniform weights to each neighbor. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied which is used to compute the weights.

2. GaussianNB

GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

3. RandomForestClassifier

The **sklearn.ensemble** module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method. Both algorithms are perturb-and-combine techniques specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features.

As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

4. DNNClassifier

The Perceptron is one of the simplest ANN architectures, it is based on a slightly different artificial neuron called a linear threshold unit (LTU). The LTU computes a weighted sum of its inputs ($z = w1\ x1 + w2\ x2 + \cdots + wn\ xn = \mathbf{w}T \cdot \mathbf{x}$), then applies a step function to that sum and outputs the result: $h\mathbf{w}(\mathbf{x}) = \text{step}\ (z) = \text{step}\ (\mathbf{w}T\ .\mathbf{x})$. Perceptron is fed one training instance at a time, and for each instance it makes its predictions.

For every output neuron that produced a wrong prediction, it reinforces the connection weights from the inputs that would have contributed to the correct prediction.

Stacking multiple Perceptrons results to ANN is called a Multi-Layer Perceptron (MLP). An MLP is composed of one (passthrough) input layer, one or more layers of LTUs, called hidden layers, and one final layer of LTUs called the output layer. Every layer except the output layer includes a bias neuron and is fully connected to the next layer. When an ANN has two or more hidden layers, it is called a deep neural network (DNN).

DNN is best to use when the data is in large and sports and daily activities dataset is quiet large.

The problem being classification problem and you tried non-linear algorithms like NB, random forest, DNN to predict the probability for classes. DNN was tried because the input features were continuous and numeric and hence parameters weights and biases could be trained well for accurate predictions.

Benchmark

“BDM provides a correct classification rate of 99.2% with relatively small computational cost and storage requirements. The same rate of 99.2% is achieved with ANN and SVM in WEKA. The average correct classification rates previously reported for BDM using RRSS and 10-fold cross-validation techniques are 99.1 and 99.2%, respectively, whereas these rates drop to 96.5 and 96.6% for NB”.

Methodology

Data Processing

Anaconda jupyter environment was used for this project. I did some data wangling to make data ready for applying machine learning algorithms.

Made a loop to add data from different folders and load them in notebook. Used that loop for 19 times as there were 19 folder of different activities and labeled them as a01, a02, a03…..a19.

Added all the data with the help of concat command in pandas and renamed all the columns as signal: 1, signal: 2…..signal: 45 using rename command in pandas after shuffling the data.

- Total number of rows in dataset are 14,40,000.
- Total number of columns are 45, 45th column is of labels.

- Total number of activities in multi class classification are 19.

Then I separated labels from dataset and split the dataset into training and test set in the ratio of 70:30. Reset the index of all datasets.

Scale of the data

MinMax scaler, Transforms features by scaling each feature to a given range.

This estimator scales and translates each feature individually such that it is in the given range on the training set, i.e. between zero and one.

Encode of the labels

LabelEncoder was used to encode the labels.

Dimension Reduction

Principal component analysis (PCA). Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space, was used to reduce number features and amount of data to speed up the time taken by algorithm. Didn't have better results using PCA.

Implementation

After the data processing the data was ready for machine learning algorithm.

1. KNeighborsClassifier – Parameters (*n_neighbors=5*, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=1*, ***kwargs*). Results achieved through this algorithm are stated in table 1.

2. Naïve Bayes – Parameters (*priors=None*). Results achieved through this algorithm are stated in table 1.

3. Random Forest classifier – Parameters (*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*). Results achieved through this algorithm are stated in table 1.

4. DNN classifier - Parameters (model_dir=None, n_classes=2, weight_column_name=None, optimizer=None, activation_fn=tf.nn.relu, dropout=None, gradient_clip_norm=None, enable_centered_bias=False, config=None, feature_engineering_fn=None, embedding_lr_multipliers=None, input_layer_min_slice_size=None, label_keys=None) using tensorflow was used and best results were achieved, results achieved through this algorithm are stated in table 1.

Confusion metrics was used to evaluate the results, f1_score and accuracy was obtained after implementation of every algorithm.

Most interesting and difficult part which I found was the addition of data as they were in many folders and subfolders. Second most interesting and difficult part was to apply different algorithms and tune the parameters.

Results achieved using different algorithms:-

Algorithm	**F_score**	**Accuracy**
KNeighborsClassifier	0.15	0.16
GaussianNB	0.0369	0.1043
RandomForestClassifier	0.4303	0.4422
DNNClassifier	0.6861	0.7056

Table 1

Refinement

Initially with DNN classifier with the architecture of:-

Hidden units = [500, 250]

Batch size=50

Steps=40000

Results achieved were F_score = 0.60 and accuracy = 62.20%

With the architecture of:-

Hidden units = [1024, 512, 256]

Batch size=50

Steps=40000

Results achieved were F_score = 0.65 and accuracy = 66.20%

Finally with the architecture of:-

Hidden units = [2048, 1024, 512, 256]

Batch size=50

Steps=40000

Results achieved were F_score = 0.6864 and accuracy = 70.56%

Results

Model evaluation and validation

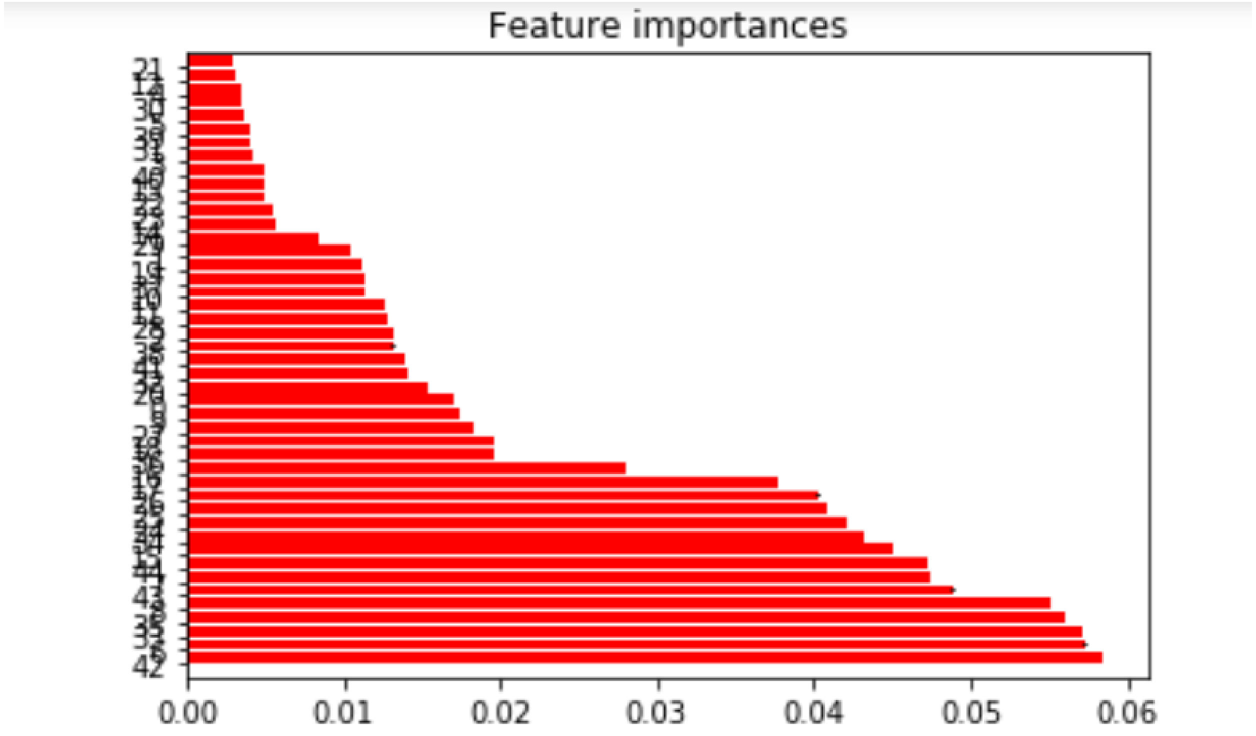
DNNClassifier was chosen because it has achieved the best f1_score = 0.68 and accuracy = 70.56%. This model is robust, when validated with the help of validation dataset, dataset other than test dataset, gives the f1_score = 0.6587 and accuracy = 67.43%. So this model generalize well to unseen data and can be trusted.

Justification

According to benchmark model best results achieved is 99.2% of accuracy by using ANN, SVM and BDM. I have got the best of accuracy 70.56% using DNNClassifier. Better results can be achieved using more hidden layers and more complex architecture.

Conclusion

Free-Form Visualization



Feature importance ranking is given in the code.

Reflection

First of all data was loaded in the notebook. Data was distributed in many folders and subfolders so addition of all the data was done. After the addition of all the data shuffling and renaming of data was done. Labels were separated from the data and then splitting of data set was done into training, test and validation set. After that visualization of data was done. Data processing was done by using MinMax scaler and label encoding was done. Implement of few algorithm were done and results were achieved using f1_score and accuracy. Results were improved by tuning the parameters.

Most interesting and difficult part which I found was the addition of data as they were in many folders and subfolders. Second most interesting and difficult part was to apply different algorithms and tune the parameters.

Improvement

Better results can be achieved by using more hidden layers and more complex architecture.

RandomForestClassifier has also given very good results of F_score = 0.43 and accuracy = 44.22%. Using GridSearchCV could increase the results.