

# **MACHINE LEARNING ASSIGNMENT - 1**

**Darshan Agarwal**

**201225189**

## **Problem Setting**

Image annotation is one of the major research areas in the field of computer vision. It aims at predicting a set of textual labels for an image that describe its semantics. It is used in image retrieval systems to organize and locate images of interest from a database. With increase in the visual data there is a high demand of correlating two images semantically. This method can be regarded as a type of multi-class image classification with a very large number of classes - as large as the vocabulary size. In this assignment, we try to solve the problem of annotating images with multiple labels using three different approaches, namely : JEC, 2PKNN and Tag-propagation. These algorithms are built upon Nearest neighbour approach.

## **Problem Space**

Given an unseen image the task is to assign multiple labels relevant to that image.

## **Dataset**

We have used Corel-5k dataset for the assignment. It has 5000 manually labelled images. We have used 4500 images as training data and 499 images as testing data for all the experiments.

## **Features Used**

There are two types of features we are using :

- Feat1 : Standard features like colour, texture, shape.(HSV, RGB, LAB, Densehue, Gist, etc.) Using each of these features, pairwise distances among the samples
- are computed.
- Feat2 : By combining all the basic features, a single feature vector is computed. This is used for computing a single pair-wise distance matrix.

## **Nearest Neighbours Techniques**

Nearest Neighbour techniques have been found to achieve good performance in the image annotation task. These are based on simple idea that “similar images share similar labels”. To predict the image for a given label we first identify a set of its similar images and propagate the labels using different methods.

## **Techniques under study**

### **Joint Equal Contribution(JEC):**

In this method, to find the similar images we take equal contribution of all the features and use the distance between the images as the measure for similarity. After finding the k nearest neighbors of the test image it transfers the labels by following the below steps:

- Assign all the labels of the nearest image.
- If still more labels are to be assigned to the test image, sort the labels of the remaining neighboring images on 2 criterias, one on the basis of co-occurrence frequency with the labels of the nearest image and second on the basis of local frequency of the labels.

### **Tag Propagation(Tagprop):**

This method predicts tags by taking a weighted combination of the tag absence/presence among neighbors. The weights for neighbors are either determined based on the neighbor rank or its distance, and set automatically by maximizing the likelihood of annotations in a set of training images.

### **Tag Propagation using SD (Tagprop SD)**

This requires the distance between the training set and the test vector. In this case,  $w$ , the weight vector is a scalar, i.e., we are taking equal contributions of all the base distances.

### **Tag Propagation using sigma SD (Tagprop sigma SD)**

Sometimes, frequent tags skew the results, suppressing the effect of rare tags. To overcome this, we define a sigmoid type of function that enhances of rare tags and suppresses the results of very frequent tags.

### **Tag Propagation using ML:**

It is similar to above approach. The only difference is that the weight vector  $w$ , is not a scalar, hence the number number of base distances.

### **Tag Propagation using $\sigma$ ML:**

Similar to Tag Propagation using ML, it uses the weight vector ' $w$ '. It also takes into account the frequency of tags using the sigmoid function as described above.

### **2PKNN (Two-pass K-Nearest Neighbour):**

In this approach, instead of finding the nearest cluster, we filter out those results which are closer to the test vector. So, we find those  $R$  images which are closest in each cluster, and cluster those  $R$  images, and use them to classify the relevant tags.

## **Evaluation criterion**

To analyze the computation performance, we compute precision and recall of each label in the dataset. Suppose a label  $y_i$  is present in the ground truth of the  $m_1$  image and is predicted for  $m_2$  images during testing out of which  $m_3$  predictions are correct, then its precision will be  $= m_3/m_2$  and its recall will be  $= m_3/m_1$ . We average these values over all the labels of a dataset and get percentage mean precision  $P$  and percentage mean recall  $R$ .

The evaluation criterion used

- Average per-label precision
- Average per-label recall
- Per-label F1 score
- Number of labels with positive recall

For label  $l_i$ , precision  $P_i$  and recall  $R_i$  are defined as:

$$P = (\text{True positive})/(\text{True positive} + \text{false positive})$$

$$R = (\text{True Positive})/(\text{True Positive} + \text{false Negative})$$

F1score per label is defined as :

$$F1 = 2PR/(P + R)$$

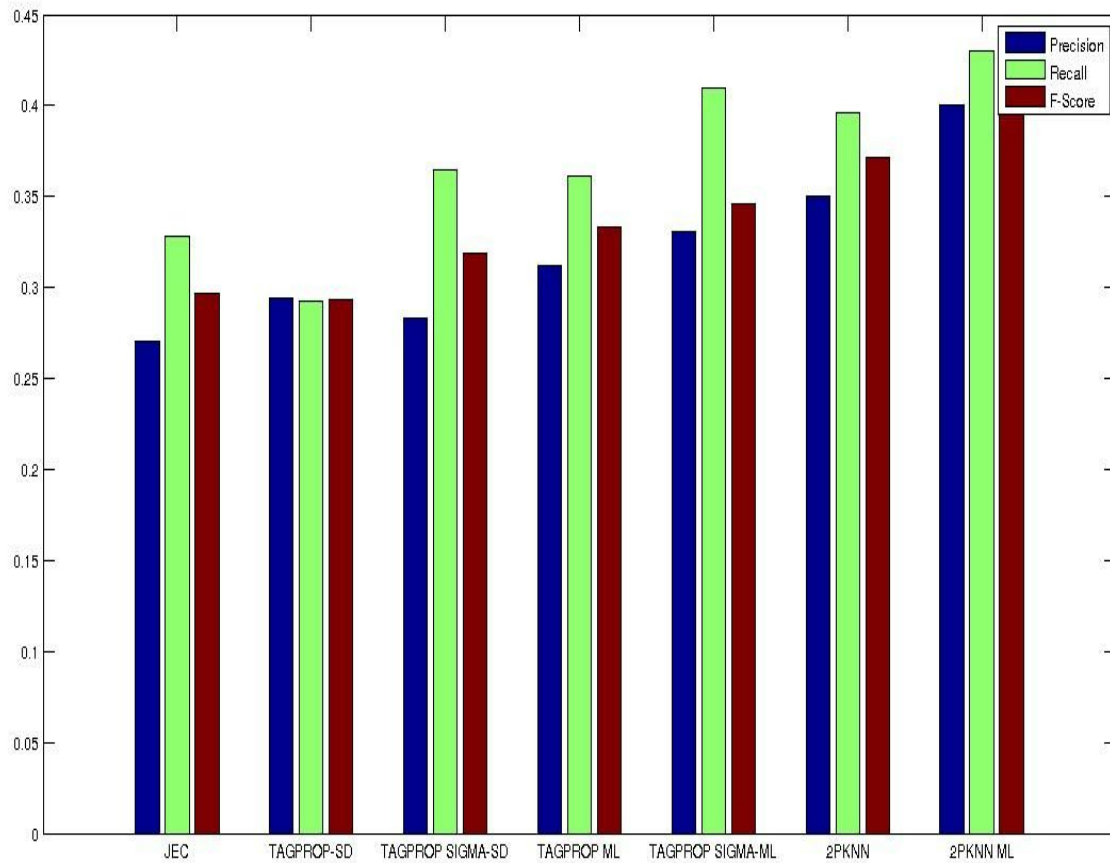
The following combinations of evaluation criteria are used for doing various experiments:

- (e1) v1 using Feat1
- (e2) v2 to v5 using Feat1
- (e3) v6 and v7 using Feat1
- (e4) The best of (1), (2) and (3) using Feat1
- (e5) v1 using Feat2
- (e6) v2 and v3 using Feat2
- (e7) v6 using Feat2
- (e8) v1, v2, v3 and v6 using Feat2

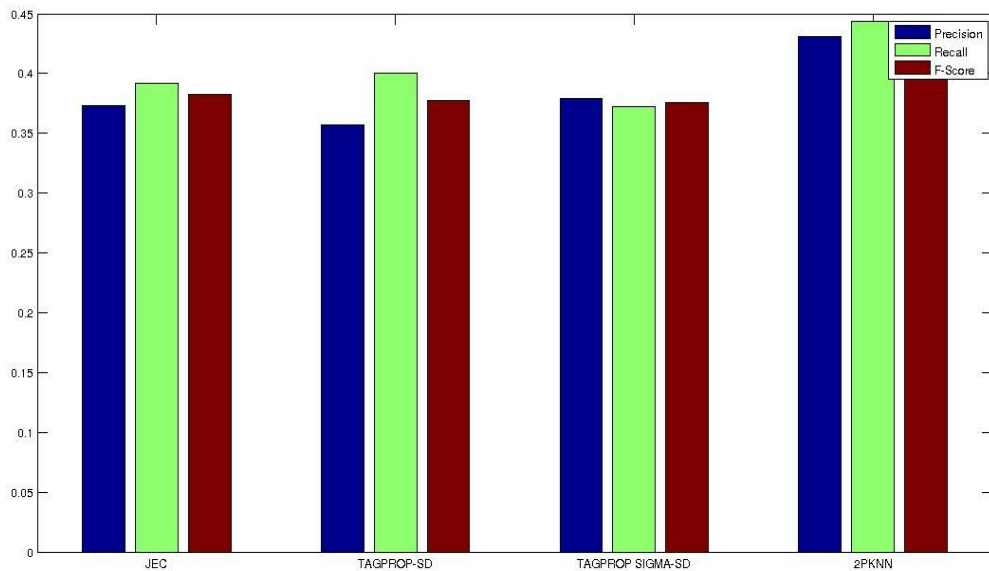
### **Explorations (Part - A)**

| <u>Method</u>    | <u>Features</u> | <u>Precision</u> | <u>Recall</u> | <u>F-score</u> | <u>N+</u> |
|------------------|-----------------|------------------|---------------|----------------|-----------|
| JEC              | Feat 1          | 0.2703           | 0.328         | 0.2965         | 141       |
| TagProp-SD       | Feat 1          | 0.2943           | 0.2927        | 0.2935         | 125       |
| Tagprop sigma-SD | Feat 1          | 0.2835           | 0.3649        | 0.3191         | 149       |
| TagProp ML       | Feat 1          | 0.3121           | 0.3611        | 0.333          | 143       |
| Tagprop sigma-ML | Feat 1          | 0.331            | 0.41          | 0.346          | 156       |
| 2PKNN            | Feat 1          | 0.3501           | 0.396         | 0.3716         | 172       |
| 2PKNN + ML       | Feat 1          | 0.40             | 0.43          | 0.414          | 166       |
| JEC              | Feat 2          | 0.3730           | 0.3919        | 0.3822         | 150       |
| TagProp-SD       | Feat 2          | 0.357            | 0.4006        | 0.3775         | 150       |
| Tagprop sigma-SD | Feat 2          | 0.3792           | 0.3725        | 0.3758         | 146       |
| 2PKNN            | Feat 2          | 0.4309           | 0.4438        | 0.4372         | 173       |

Bar Graph showing the Precision(Blue) , Recall(green), F-score(Red) of all the methods for Feat1



Bar Graph showing the Precision(Blue) , Recall(green), F-score(Red) of all the methods for Feat2



From the above, results and graphs, we can see that 2PKNN performs the best among all the methods. The results are relatively consistent with the results given in papers, though there are variations in the exact values.

### **Experiment-2: Role of Validation**

Cross-validation, is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. Validation dataset is defined to test the model in the training phase, in order to limit problems like overfitting, and to give an insight on how the model will generalize to an unknown dataset.

| Validation Set Used | Precision | Recall | F-Score | N Plus | Algorithm -<br>Feature used |
|---------------------|-----------|--------|---------|--------|-----------------------------|
| 10 percent          | 0.3325    | 0.3474 | 0.3398  | 104    | JEC - Feat2                 |
| 20 percent          | 0.4867    | 0.5143 | 0.5002  | 160    | JEC - Feat2                 |
| 30 percent          | 0.5439    | 0.5826 | 0.5626  | 180    | JEC - Feat2                 |
| 10 percent          | 0.5279    | 0.4954 | 0.5112  | 165    | 2PKNN - Feat2               |

Precision, Recall, F-Score and N+ on actual test data.

| Algorithm -<br>Featue | Precision | Recall | F-Score | NPlus |
|-----------------------|-----------|--------|---------|-------|
| JEC - Feat2           | 0.3730    | 0.3919 | 0.3822  | 150   |
| 2PKNN - Feat2         | 0.4309    | 0.4438 | 0.4372  | 173   |

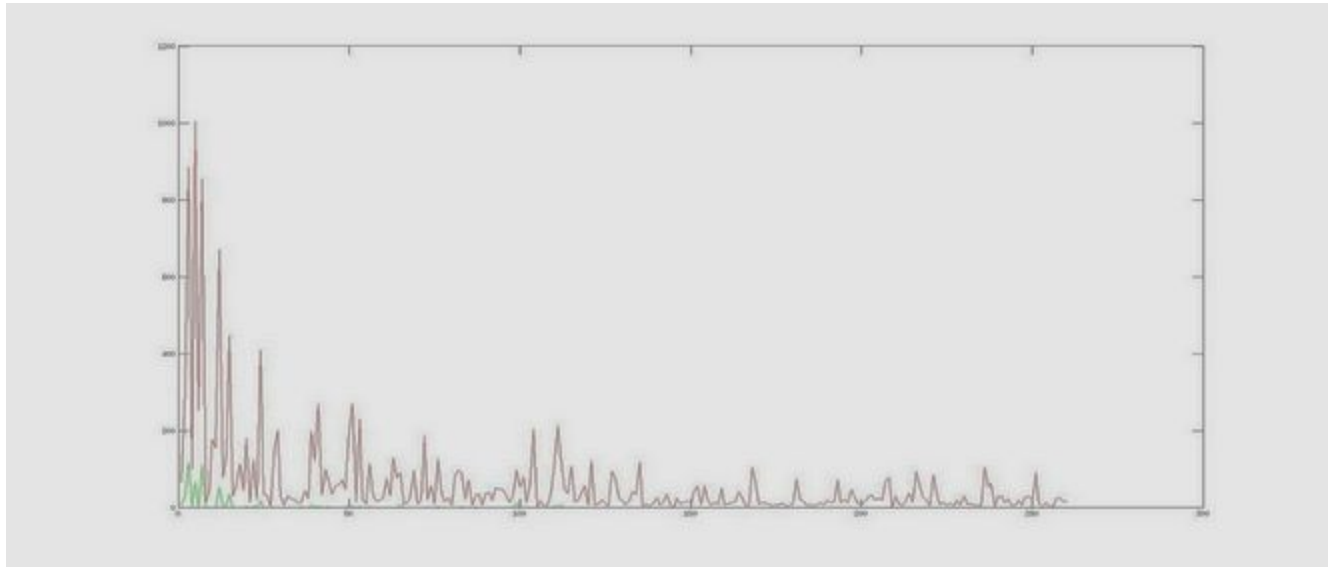
There are cases when the model has learned certain features from the training data which are not applicable to a larger dataset which is unknown, due to which performance of model goes down.

We call a model as overfit model when the accuracy on training data is much higher than the testing accuracy. Hence to avoid this, we separate usually 10% of the training data and treat it as validation data to avoid overfitting.

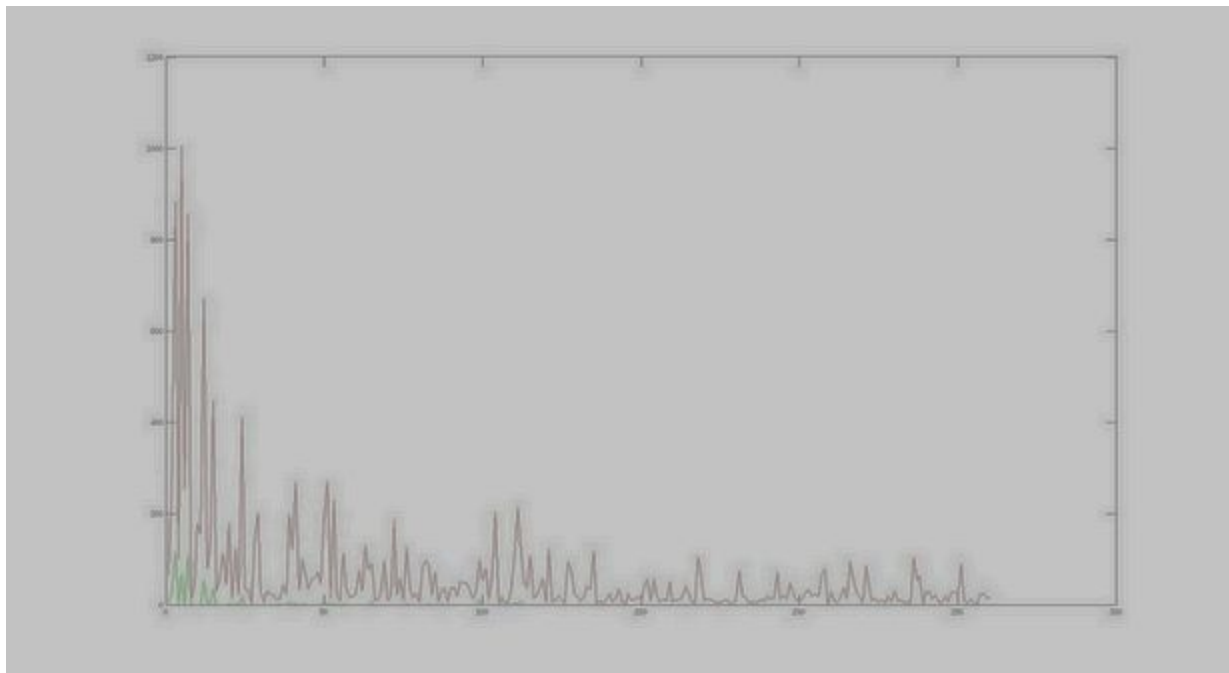
From the above results, the evaluation results of validation data is closer to the testing data when only 10% of the training data is used as validation data. We can observe that as the validation data increases the results move away from the test data and it overfits with the training data. In our case, validation data also helps in deciding the values of the parameters. In JEC, validation data helps in deciding the perfect value of  $k$  (i.e. number of neighbors). While in Tagprop, it helps in getting the values of parameters  $k$  and  $w$ .

## Per Label Performance

The label frequency of the JEC(in green) and the label frequency of the training data is in red  
JEC - Feat1 : Label index (x-axis) vs Frequency



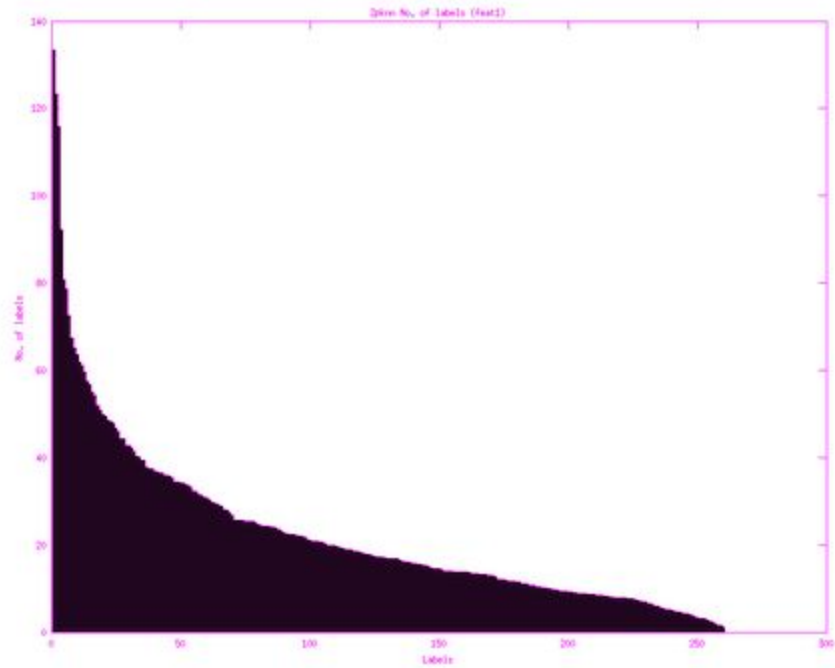
JEC Feat2 : Label index vs Frequency



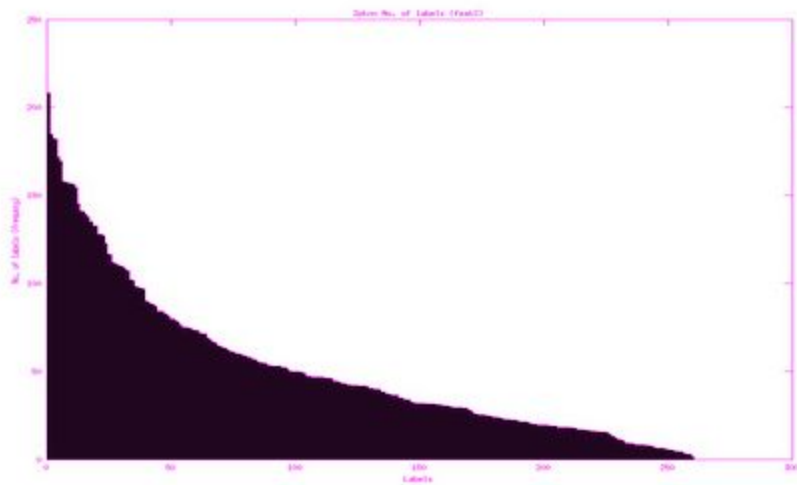


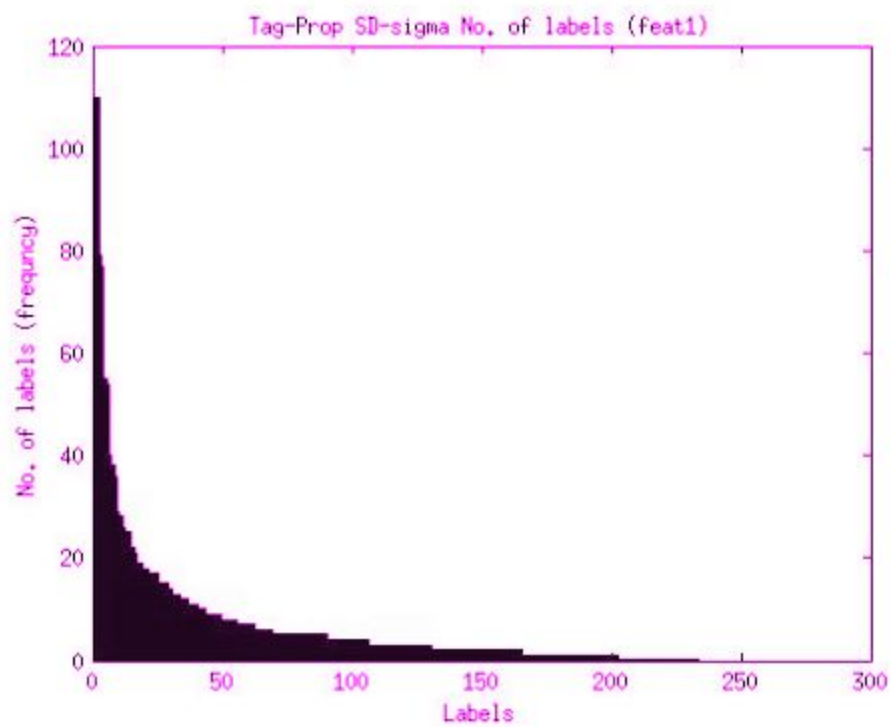
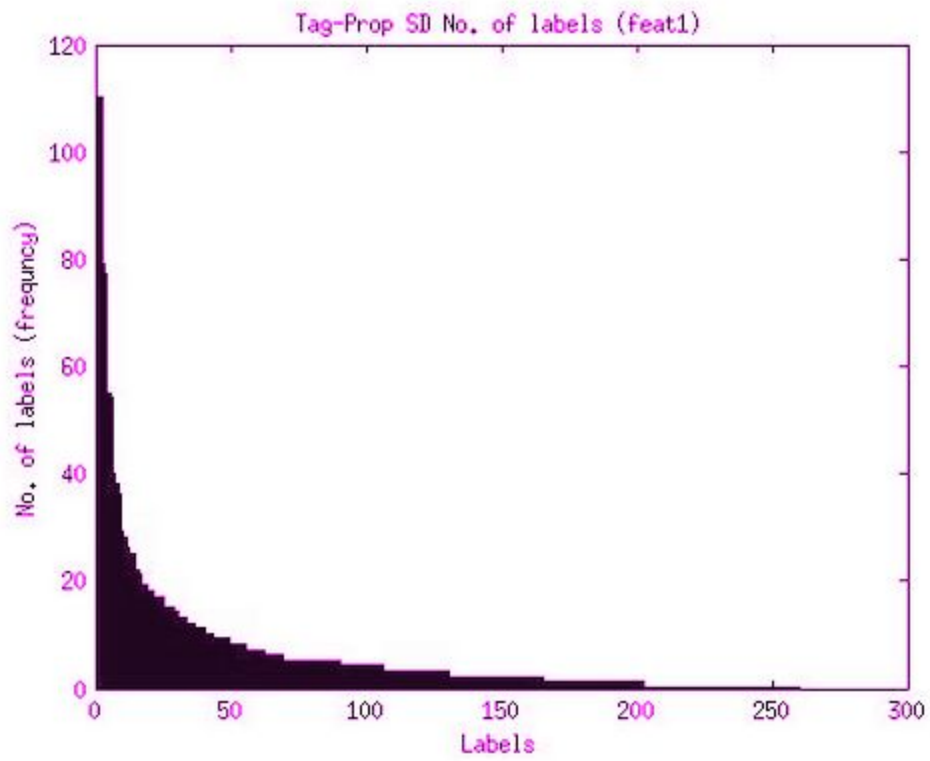
From the graph we can observe that per label performance of JEC is similar to the per label frequency of the training data.

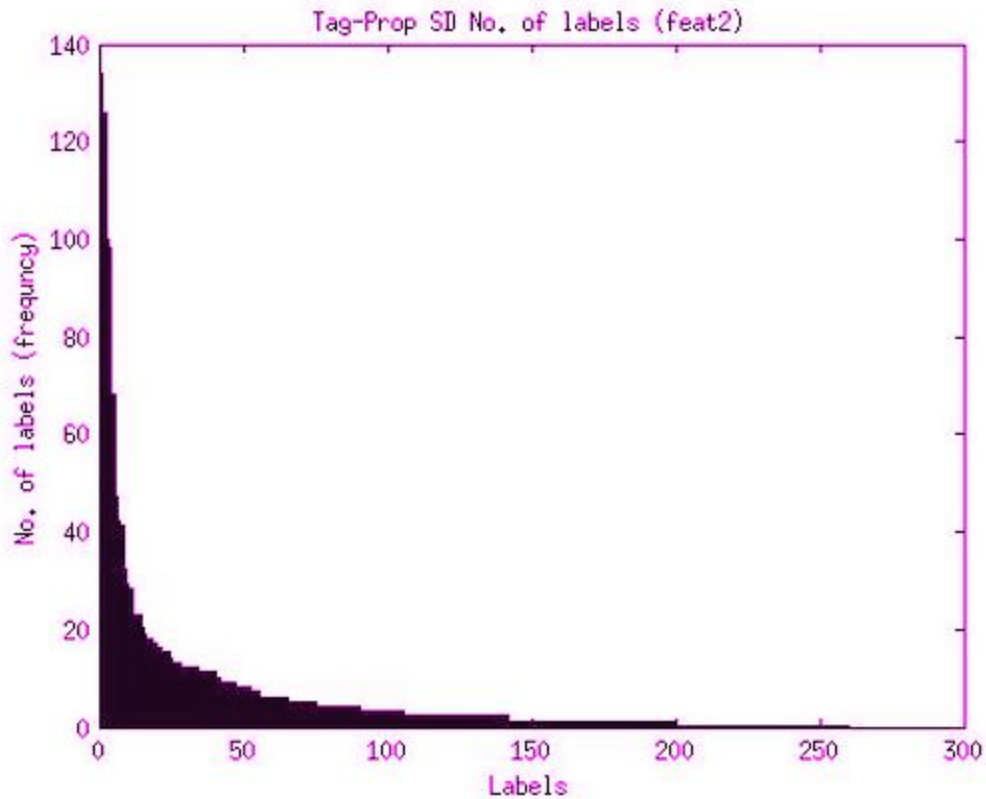
2PKNN feat1: Label-index vs Frequency



2PKNN feat2 : Label-index vs Frequency



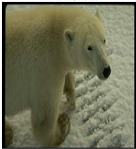








The label frequency plot of training set shows that there is significant amount of class imbalance i.e lot of variation among frequency of classes. The issue of class imbalance is successfully handled by 2p-kNN as visible from the frequency plot. The rare frequency labels occur in more number of test images.

#### **Experiment-4: Qualitative analysis:**

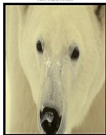



Improvement of v3 over v1 and v6


| Image   | Ground Truth                        | v1   | v3   | v6   |
|---|-------------------------------------|--|--|--|
|    | bear, polar<br>snow, face           | city,<br>mountain,<br>sky, sun,<br>water, clouds,<br>tree, lake, sea,<br>ice, frost,<br>frozen | bear, polar,<br>snow, frost,<br>frozen         | tree, nets,<br>barn,<br>lighthouse,<br>porcupine |
|   | leaf, close-up,<br>plant            | city,<br>mountain,<br>sky, water,<br>clouds, tree,<br>lake, sea, leaf,<br>flowers,<br>plants   | tree, leaf,<br>flowers,<br>close-up,<br>plants | basket,<br>clothes, vines,<br>column, baby       |
|  | water, beach,<br>people,<br>horizon | city,<br>mountain,<br>sky, water,<br>clouds, tree,<br>lake, sea,<br>beach, people,<br>sand     | sky, water,<br>beach, people,<br>sand          | head, shrubs,<br>sheep,<br>column,<br>lighthouse |

|   |                             |  |                                      |  |
|---|-----------------------------|--|--------------------------------------|--|
|   |                             |  |                                      |  |
|  | bear, polar<br>snow, face   | city,<br>mountain,<br>sky, sun,<br>water, clouds,<br>tree, lake, sea,<br>ice, frost,<br>frozen | coral,<br>anemone,<br>ocean, reefs   | tree, dress,<br>plaza, lion,<br>lynx       |
|  | sun, water,<br>clouds, tree | sun, water,<br>clouds, tree,<br>palm   | sun, water,<br>clouds, tree,<br>palm | tree, dress,<br>market,<br>indian, african |

In the above examples, Tagprop-SD algorithm performs better than other two using 'Feat2'. It is observed that in the given dataset the images need to be identified uniquely with labels rather and class imbalance seems on a lower side. Hence, we need to consider the weights of each label towards the same, but considering the uniqueness of the label at the same time which is handled by 'sigmoid' function in Tagprop. Thus, it performs better in such images, as depicted in the example.

## Improvement of v6 over v1 and v3

| Image   | Ground Truth                       | v1  | v3                                     | v6                                      |
|---|------------------------------------|---|--|---|
|    | bear, polar, close-up, face        | city, mountain, sky, sun, water, clouds, tree, lake, sea, bear, polar, snow           | grass, bear, polar, snow, ice          | bear, polar, snow, head, face           |
|   | smoke, train, locomotive, railroad | city, mountain, sky, sun, water, clouds, tree, lake, sea, train, locomotive, railroad | sky, tree, train, locomotive, railroad | sky, smoke, train, locomotive, railroad |
|  | tree, path, garden, cottage        | city, mountain, sky, sun, water, clouds, tree, lake, sea, house, garden, lawn         | tree, flowers, garden, cottage, lawn   | tree, path, garden, cottage, lawn       |
|  | buildings, clothes, shops, street  | city, mountain, sky, sun, water, clouds,  | tree, people, buildings, shops, street | buildings, clothes, shops, street, sign |

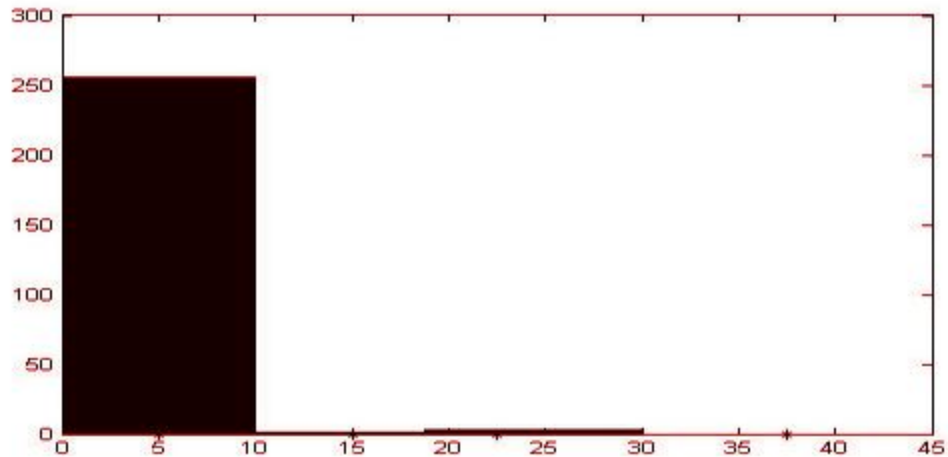
|   |  |   |   |   |
|---|--|---|---|---|
|   |  | tree, lake,<br>people,<br>buildings,<br>shops, street   |   |   |
|  | buildings,<br>skyline, street,<br>cars | city,<br>mountain,<br>sky, sun,<br>water, clouds,<br>tree, lake, sea,<br>buildings,<br>street, cars | sky, tree,<br>people,<br>buildings,<br>street | sky, tree,<br>buildings,<br>skyline, street |

Here, we have v6(2p knn) dominating over the other two methods used for comparison. 2pknn is efficient in handling both class imbalance and weak labelling issues, and hence the choice of the examples. Under case 3, the ‘weak labelling’ problem is clearly highlighted where one can easily observe the label ‘path’ which is often missed out, due to the stated issue. Although, v3 results are at par with v6 in this example but misses out on the point mentioned formerly. v1 on the other hand, clearly suffers with class imbalance problem in the selected examples.

## Explorations (Part-B)

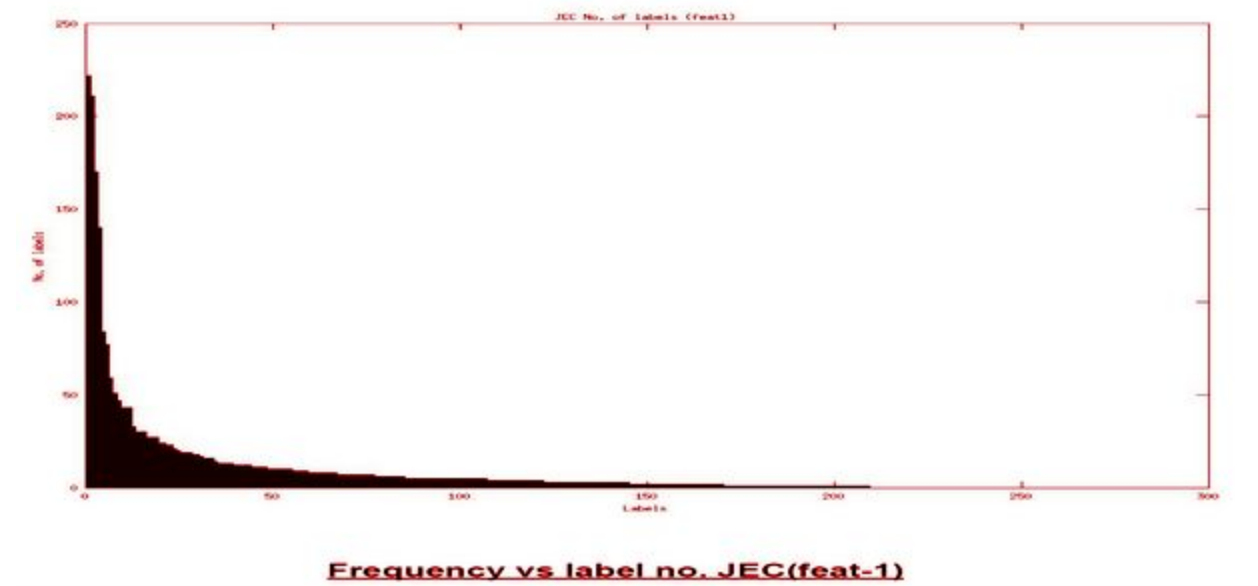
### Experiment-1: Analyzing “class-imbalance”:

Bar plot for number of labels falling in each bin

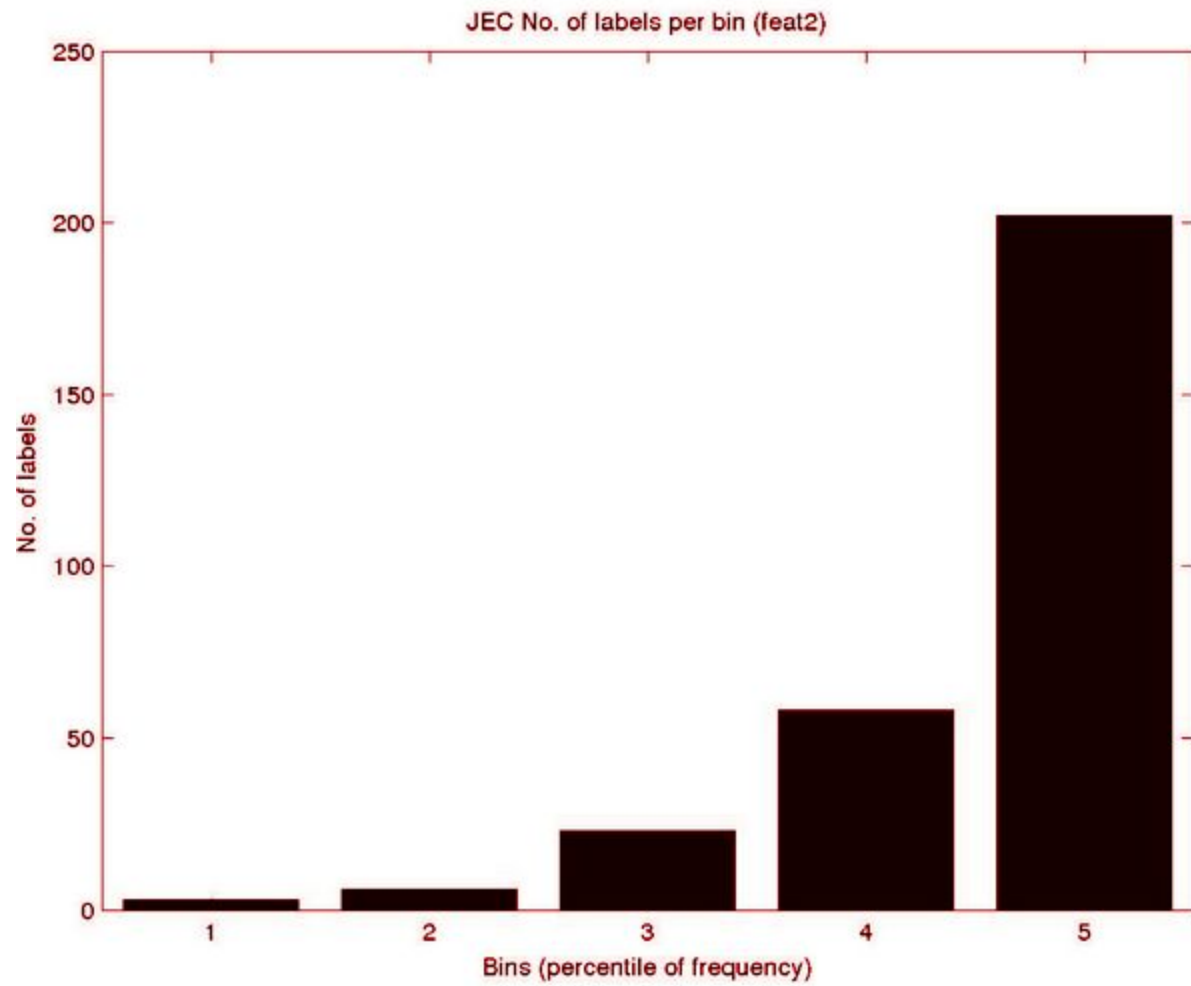


Number of Labels Vs Frequency

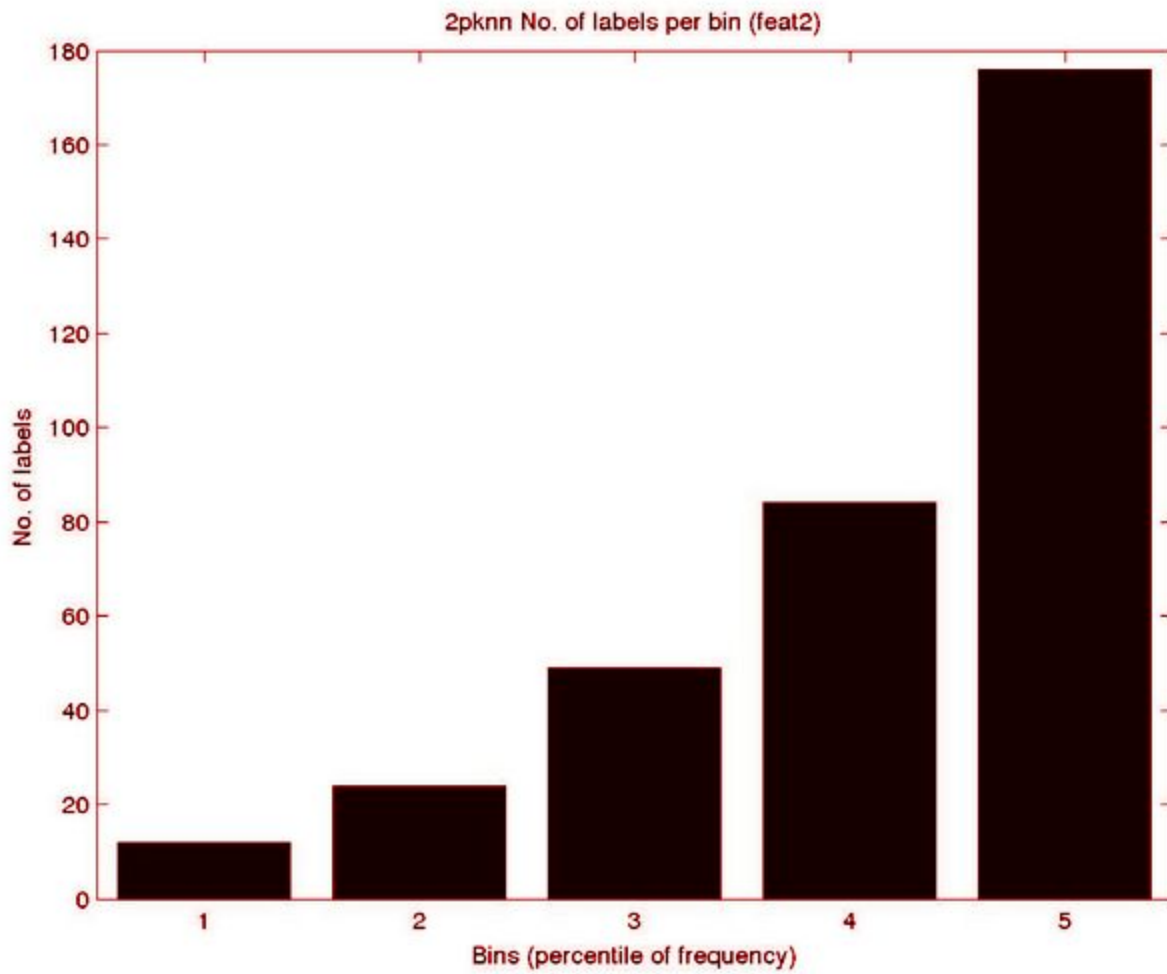
2. Bar-plots for each of the eight evaluation criteria after removing 20% of the labels



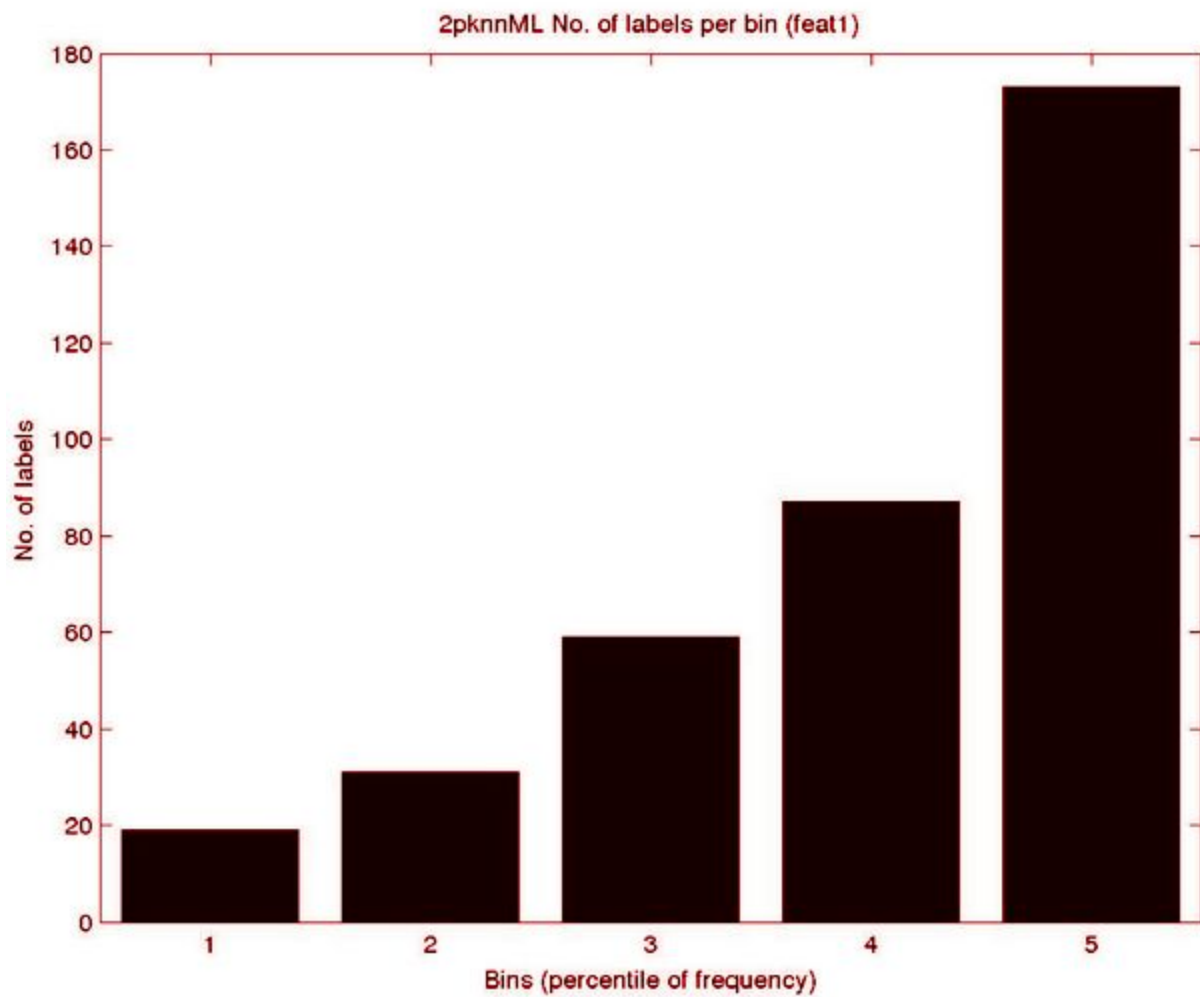




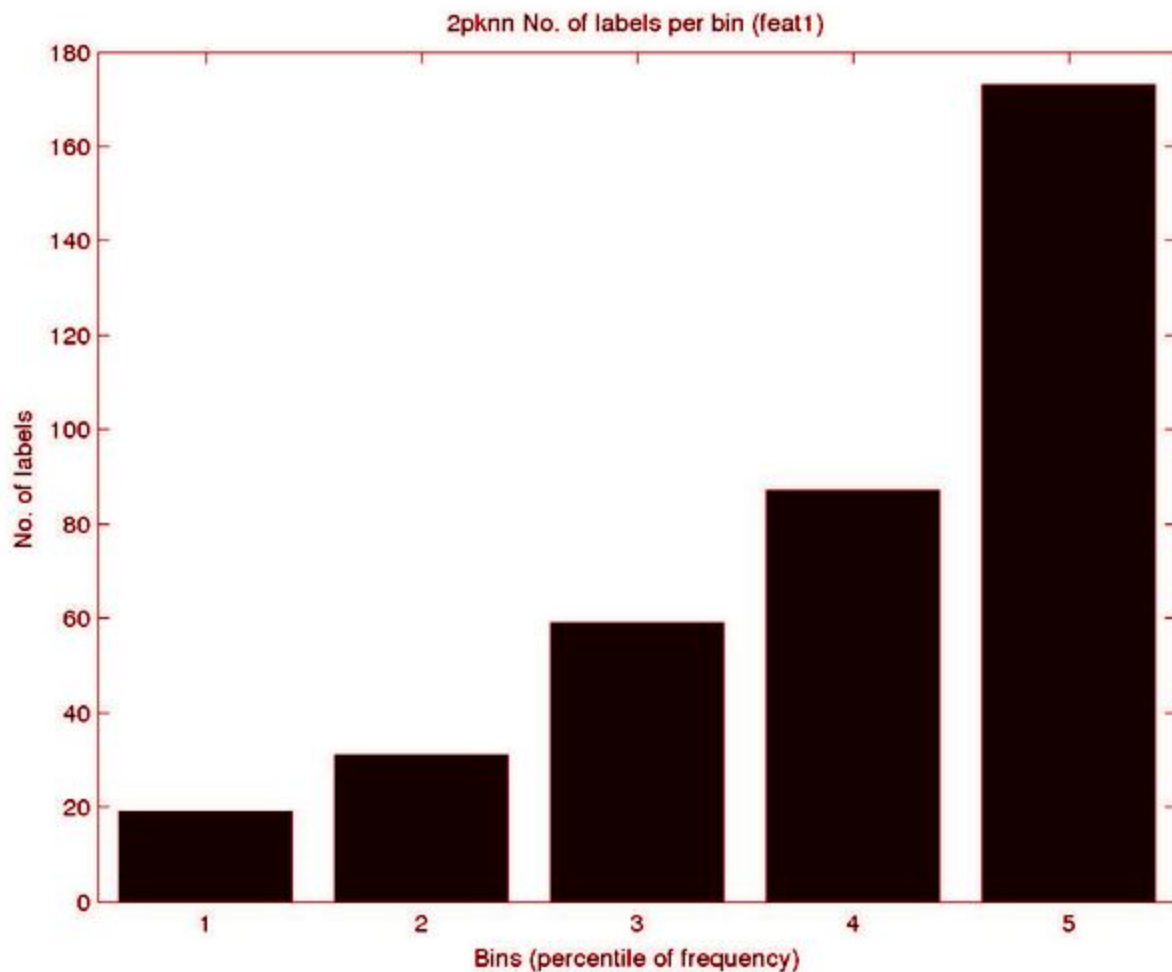
**Frequency vs label no. JEC (feat-2)**



Frequency vs label no. 2pknn (feat-2)



Frequency vs label no. 2pknn-ML (feat-1)

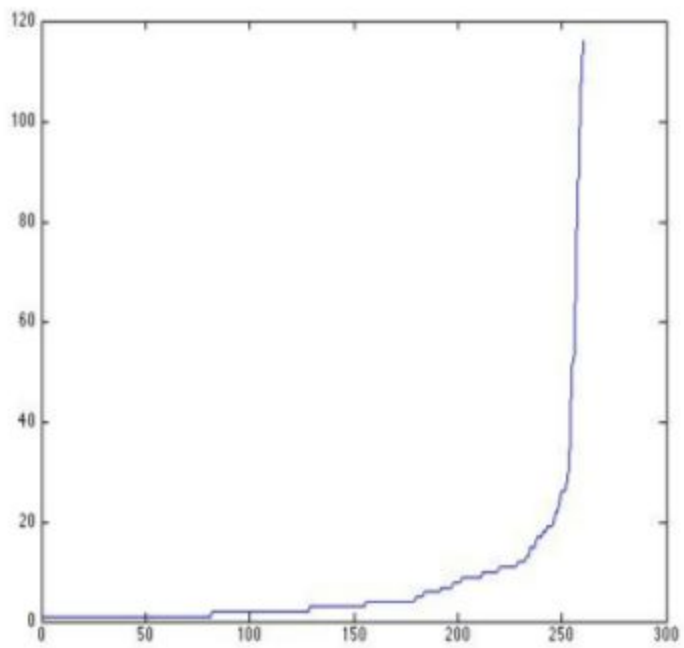


**Frequency vs label no. 2pknn (feat-1)**

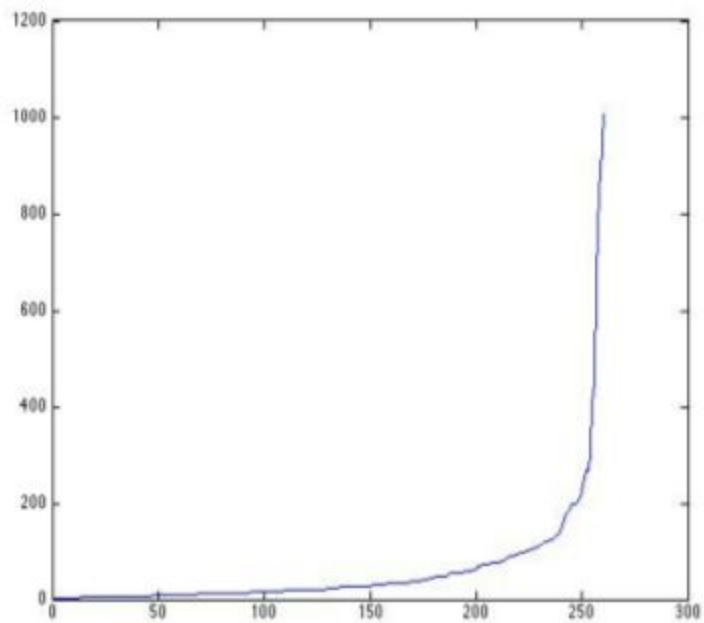
**Experiment-2: Randomly removing labels from the ground-truth of training data**

Roll number: 201225189%6 = 3 so 30% of the labels are removed from both training and testing images.

A line plot of the label-index(x axis) and frequency(y-axis) of the original frequency:

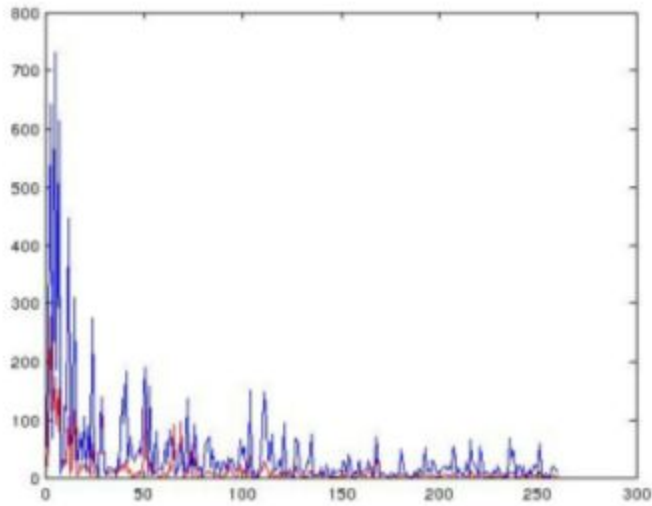


A line plot of the label-index(x axis) and frequency(y-axis) after removing 30%

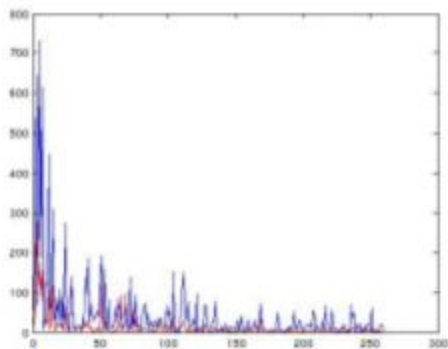


## 1. Line plot comparison

feat1 training data 30% same for all

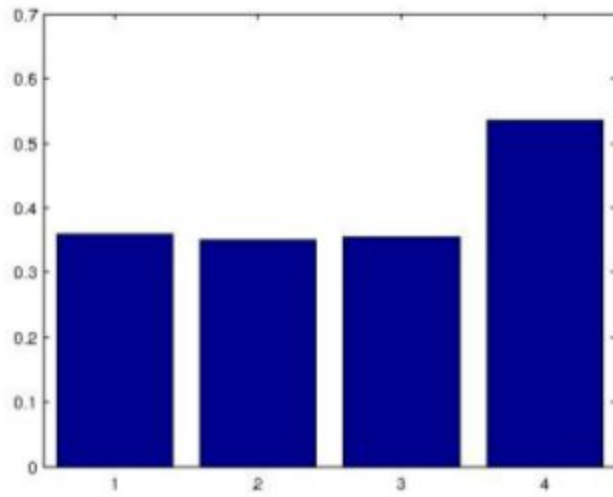


feat2 training data 30% same for all

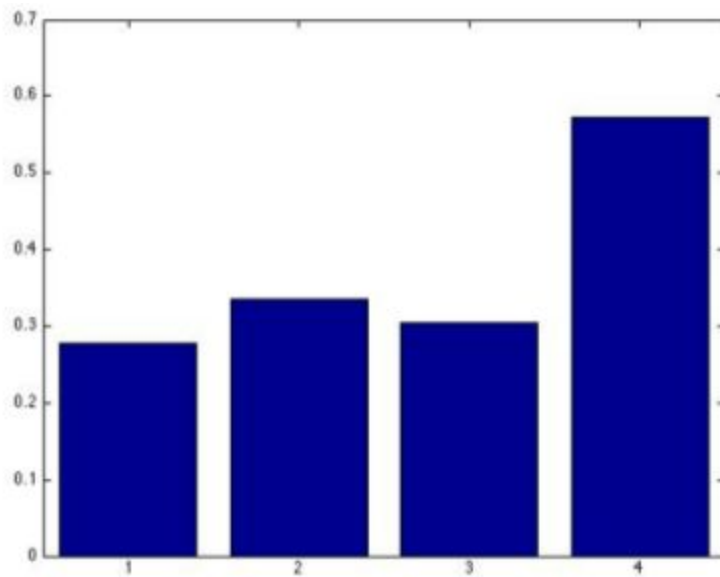


Individual bar-plot for each of the eight evaluation criteria comparing the average performance:

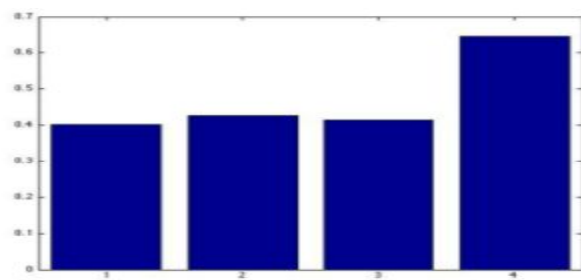
**feat1 testing 30% JEC**



**Feat2 - Testing 30% JEV**



**Feat1 using 30% PKNN**



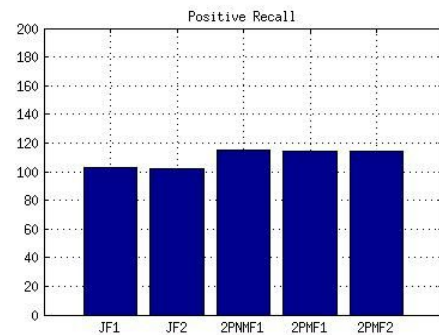
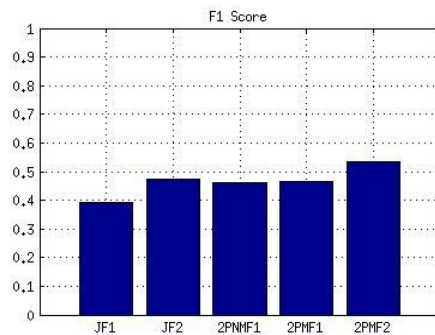
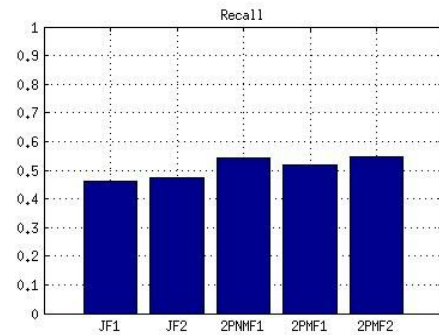
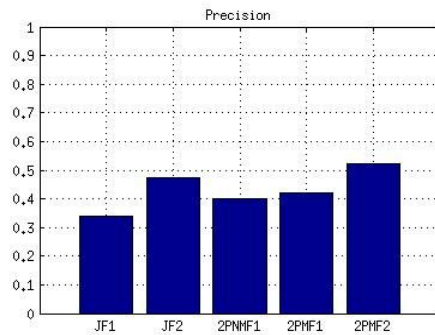
**feat2 30% pknn**

| <u>Method</u> | <u>Feature</u> | <u>Percent</u> | <u>Precision</u> | <u>Recall</u> | <u>F-Score</u> | <u>N+</u>       |
|---------------|----------------|----------------|------------------|---------------|----------------|-----------------|
| <u>JEC</u>    | <u>feat1</u>   | <u>30%</u>     | <u>0.2294</u>    | <u>0.2893</u> | <u>0.2559</u>  | <u>132.0000</u> |
| <u>JEC</u>    | <u>feat2</u>   | <u>30%</u>     | <u>0.3609</u>    | <u>0.3519</u> | <u>0.3564</u>  | <u>140.0000</u> |
| <u>2pknn</u>  | <u>feat1</u>   | <u>30%</u>     | <u>0.2778</u>    | <u>0.3350</u> | <u>0.3038</u>  | <u>149.0000</u> |
| <u>2pknn</u>  | <u>feat2</u>   | <u>30%</u>     | <u>0.4016</u>    | <u>0.4261</u> | <u>0.4135</u>  | <u>168.000</u>  |

### Shrinking the training/test data by considering only frequent labels

Roll number: 2012251892%3 = 0 so 75% of the labels are removed from both training and testing images

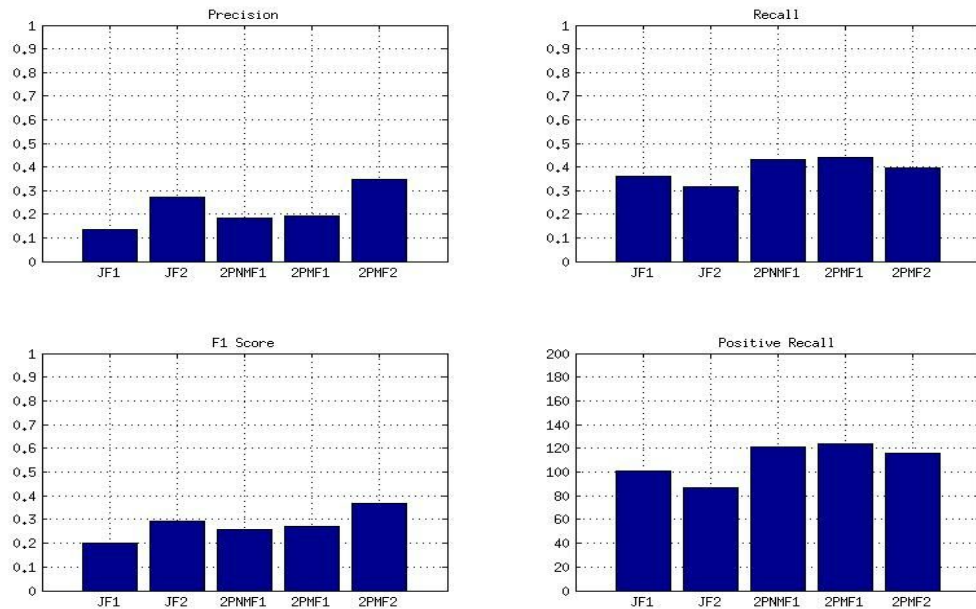
Individual bar-plot for each of the eight evaluation criteria comparing the average performance.





| <u>Method</u> | <u>Feature</u> | <u>Percent</u> | <u>Precision</u> | <u>Recall</u> | <u>F-Score</u> | <u>N+</u>       |
|---------------|----------------|----------------|------------------|---------------|----------------|-----------------|
| <b>JEC</b>    | feat1          | 75%            | 0.2341           | <u>0.2812</u> | <u>0.2555</u>  | <u>127.0000</u> |
| <b>JEC</b>    | feat2          | 75%            | 0.3212           | <u>0.3015</u> | <u>0.3110</u>  | <u>126.0000</u> |
| <b>2pknn</b>  | feat1          | 75%            | 0.2341           | <u>0.2812</u> | <u>0.2555</u>  | <u>127.0000</u> |
| <b>2pknn</b>  | feat2          | 75%            | 0.3212           | <u>0.3015</u> | <u>0.3110</u>  | <u>126.0000</u> |

#### Experiment-4: Shrinking the training/test data by considering only rare labels



| <u>Method</u> | <u>Feature</u> | <u>Percent</u> | <u>Precision</u> | <u>Recall</u> | <u>F-Score</u> | <u>N+</u>      |
|---------------|----------------|----------------|------------------|---------------|----------------|----------------|
| <b>JEC</b>    | feat1          | 25%            | 0.0368           | <u>0.0722</u> | <u>0.0487</u>  | <u>23.0000</u> |
| <b>JEC</b>    | feat2          | 25%            | 0.0500           | <u>0.0790</u> | <u>0.0612</u>  | <u>23.0000</u> |
| <b>2pknn</b>  | feat1          | 25%            | 0.0064           | <u>0.1045</u> | <u>0.0120</u>  | <u>31.0000</u> |
| <b>2pknn</b>  | feat2          | 25%            | 0.0310           | <u>0.0955</u> | <u>0.0468</u>  | <u>28.0000</u> |

#### 5. Extensions:

Demonstrate how the annotation problem can be casted and solved using any one of (i) Decision Trees (ii) Neural Networks (iii) SVMs. Do explore thoroughly with the aim of obtaining superior results. (At least report the best results that you can obtain with your method.) Implement, test and analyze.

Multilabel classification can be done using any of the techniques mentioned in the assignment question. There can be two ways in which this problem can be solved:

### 1. Algorithm adaptation methods:

Decision tree based method: Modify the entropy definition:

$$\text{Entropy} = - \sum \{P(\lambda_j) \log P(\lambda_j) + (1 - P(\lambda_j)) \log(1 - P(\lambda_j))\}$$

where,  $P(\lambda_j)$  = probability of class  $\lambda_j$ . This allows to estimate the uncertainty in terms of number of bits in multi-label setting. This modified method also allows multiple labels at the leaves.

BP-MLL(Backpropagation multi label learning): The error function for the very common neural network learning algorithm, back-propagation has been modified to account for multi-label data. Multi-layer perceptron is easy to extend for multi-label data where one output node is maintained for each class label.

SVM based method: The first idea is to have an extended dataset with  $k$  ( $= |L|$ ) additional features which are actually the predictions of each binary classifier at the first round. The  $k$  new binary classifiers are trained on this extended dataset. In this way the extended BR takes into account potential label dependencies. The second idea, ConfMat, based on a confusion matrix, removes negative training examples of a complete label if it is very similar to the positive label. The third idea is called BandSVM, removes very similar negative examples that are within a threshold distance from the learned decision hyperplane, and this helps building better models especially in the presence of overlapping classes.

## 2. Dataset transformation methods:

Label Powerset: It is a straightforward method that considers each unique set of labels in a multi-label training data as one class in the new transformed data. Therefore, the new transformed problem is a single label classification task. For a new instance, LP outputs the most probable class which actually is a set of classes in the original data.

Binary relevance: It creates  $k$  datasets, each for one class label and trains the classifier on each of these datasets. Each of these datasets contains the same number of instances as the original data, but each dataset  $D_{\lambda_j}$ ,  $1 \leq j \leq k$  positively labels instances that belong to class  $\lambda_j$  and negative otherwise.

LP and BR based random decision tree approaches prove to be best overall. They take significantly less time in training the samples. This makes them relatively feasible for classification problems when number of labels are high. For the problem being addressed in this assignment, i.e., image annotation, LP-RDT and BR-RDT perform better than most algorithms for image annotation on the scene dataset.

Random Decision Tree:

RDT constructs decision trees randomly and does not use label information much. That nature makes RDT fast and the computational cost independent of labels. When constructing each tree, the algorithm picks a remaining feature randomly at each node expansion. Once a categorical feature is chosen, it is useless to pick it again on the same decision path as it will be the same throughout the remaining path. However, continuous features can be chosen lower down the path. Classification is done at the leaf node level. Each tree outputs a class probability distribution and the final output for each class is the average of all the values obtained from each decision tree. The two types of RDT that have been implemented are:

- LP-RDT: Label Powerset considers each unique subset of labels that exists in the multi-label dataset as a single label. Let  $L$  be the set of all labels,  $L = \{l_1, l_2, \dots, l_{|L|}\}$ .  $P(L)$  is the power set of  $L$ , and  $|P(L)| = 2^{|L|}$ . Each element in  $P(L)$  stands for one possible combination of labels. During the procedure to construct

random trees, except for a leaf node that summarizes class label distribution, there is no need to use the training examples' class labels. Except on leaf node, the multi-class random tree building procedure is the same with binary classification tree.

- BR-RDT: Binary Relevance (or BR) learns one binary classifier  $h_k$  for each label  $k$ . It builds  $|L|$  datasets by using all the instances with one label  $k$  each time. To classify an unlabeled instance, BR generates the final multi-label prediction by summarizing the output of  $|L|$  classifiers. The problem of BR method is that it needs as many classifiers as the number of labels. When the number of labels is large, the training and test computation cost becomes significant. As a random tree constructed is actually independent from class labels, it can instead classify all labels, and an ensemble of random trees can be used for the problem with one to even hundreds of labels. The added trivial computation is only the label probability distribution counting on each leaf node.

Risk Analysis and Time Complexity:

Learning risk of RDT (both BR and LP) is stable. The classified errors for training instances decrease monotonically with the increasing of the number of trees, the expected training risk of RDT will also decrease. Risk bound of RDT will not increase with the increasing of the number of trees and the risk doesn't exceed the average risk of all the trees. Under the optimistic situation, RDT can minimize the generalization error with the increasing of the number of trees.

Training complexity of LP-RDT is  $O(mn[\log(mn)])$ .  $n$  is the number of leaf nodes and  $m$  is the number of decision trees. While that of BR-RDT is  $O(m(n \log(n) + tn))$ . Note that the time complexities are independent of the number of labels. The test complexity of both LP and BR RDT is  $O(mnq)$ , where  $m$  is the number of decision trees,  $n$  is the number of test instances and  $q$  is the average height of the decision trees.

The algorithm was tested on scene dataset which is a dataset for image annotation problem. Each instance represents an image with multiple class labels.

Metric figures for LP-RDT and BR-RDT

| Metric    | LP-RDT | BR-RDT |
|-----------|--------|--------|
| HamLoss   | 0.080  | 0.084  |
| Accuracy  | 0.751  | 0.737  |
| Recall    | 0.751  | 0.737  |
| Precision | 0.788  | 0.775  |

References:

[1] Mohammad S Sorower, "A Literature Survey on Algorithms for Multi-label Learning"

[2] Xiatian Zhang, Quan Yuan, Shiwan Zhao, Wei Fan, Wentao Zheng, Zhong Wang, "Multi-label Classification without the Multi-label Cost"

**Video Link:**

<https://drive.google.com/folderview?id=0B8pEDddkr6cyMUJPb2h5RXRiVXc&usp=sharing>