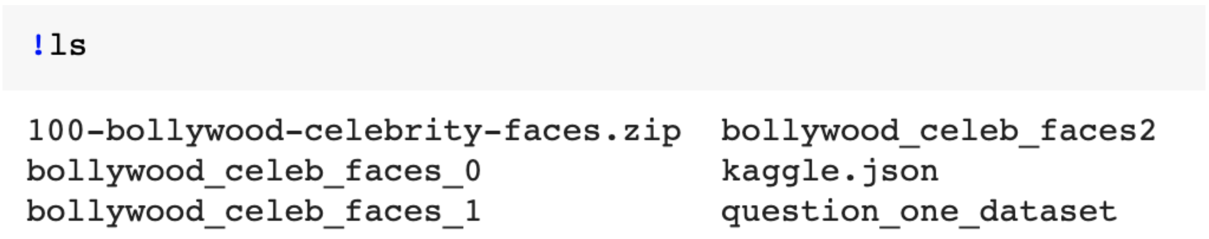
DAI_ASSIGNMENT_1_Q1_Report.pdf

**Dependable AI CSL7370 – Assignment 1**

# Task 1:

**➔Importing, Understanding and Pre-Processing the Data :**

First things first, we want to import the Celeb dataset into our workspace. In order to import the celeb dataset into the workspace we need to use google drive space. Using google colab's inherent library `from google.colab import drive` Its very easy to mount google drive storage. Once mounted we get access to the storage. Auth token must be pulled when prompted while running it. In addition, the dataset available on the Kaggle has to be copied to the google drive so that we can unzip it and use it as physical files. Luckily, we have Kaggle dataset URL as well as the API tokens generated from the Kaggle such that we can send the Kaggle commands to bash to dump the dataset into our mounted Kaggle folder in Google Drive. On successful completion of the data import, it should look like as in figure 1.



```
!ls
```

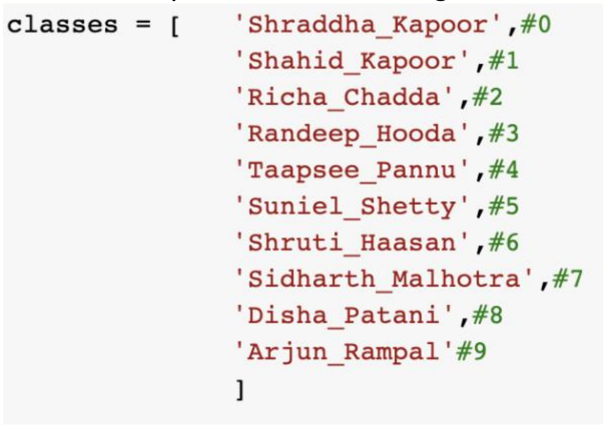
```
100-bollywood-celebrity-faces.zip  bollywood_celeb_faces2
bollywood_celeb_faces_0            kaggle.json
bollywood_celeb_faces_1            question_one_dataset
```

*Figure 1 List show of the present working directory*

Now that we have the dataset in our hands, import it. For ease, I have created a separate folder for 10 celebrities so to restrict ourselves within the single directory. The directory path looks like:

`"/content/gdrive/My Drive/Kaggle/question_one_dataset"`

Also, we can fetch all the subfolder names in the main directory using `os.listdir` to list the classes into single variable. In my case its shown in figure 2



```
classes = [    'Shraddha_Kapoor',#0
               'Shahid_Kapoor',#1
               'Richa_Chadda',#2
               'Randeep_Hooda',#3
               'Taapsee_Pannu',#4
               'Suniel_Shetty',#5
               'Shruti_Haasan',#6
               'Sidharth_Malhotra',#7
               'Disha_Patani',#8
               'Arjun_Rampal'#9
               ]
```

*Figure 2 Ten Classes focussed on*

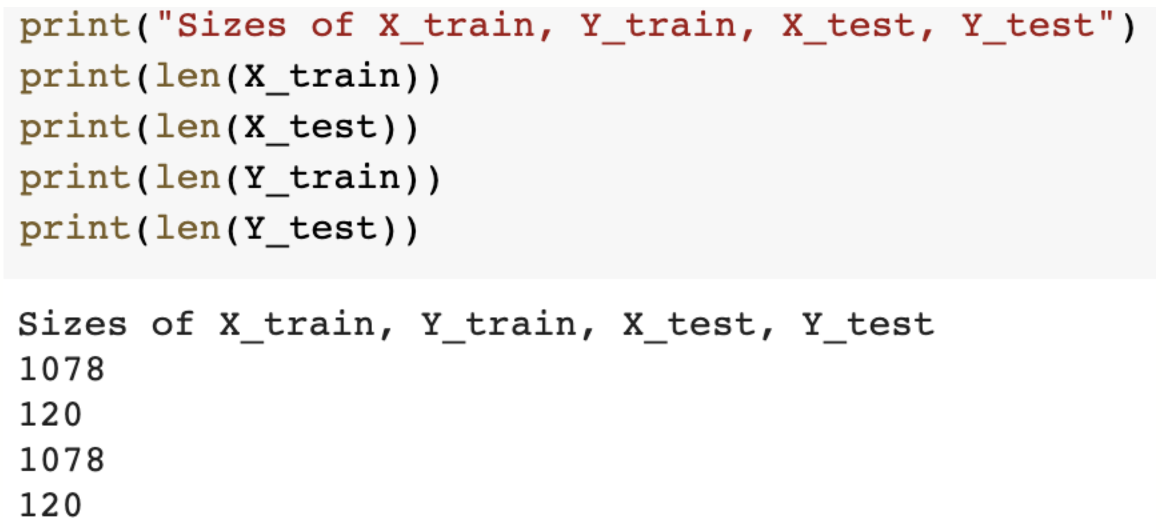
One more check just to ensure we are in right place, we shall perform this command:

```
cv2_imshow(cv2.imread("/content/gdrive/My
Drive/Kaggle/question_one_dataset/Shraddha_Kapoor/1.jpg"))
```

This results the image display of the file which is passed as path inside cv2.imread(). `cv2.imread` returns the image display of the file when given path of the file. Also `cv2_imshow` is native to google colab which helps the image to render inside the colab notebook. Now that we have all images in jpg format inside the respective directories, we will now pre process them. Preprocessing includes pixel resizing, reading images as numpy arrays and most importantly assigning each image or pixel array of the image with its class label. Class labels are indexed from 0 to 9 for the ease. In order to perform this, the stepwise process is as follows

```
Initialize empty_array[]
For each class_label_folder inside the present_directory:
        Assign path to the image,
        Assign label as the index_iteration_number,
        For each_image inside the class_label_folder:
                Assign image_path
                Actor_image ← Read each_image
                Actor_image ← numpy_array(Actor_image)
                Append_Inside_Array[Actor_image, class_label]
```

** reading the image using cv2 process has to be closed within the try catch block because if the file is not present it will return empty buffer.
Now at this stage we are just having the data on which we are going to apply training. Before that, as a best approach to generalize the learning process we can shuffle the dataset (stored inside the empty_array[]). Finally, in order to divide the dataset (In our case we have 1198 images), perform `train_test_split` from sklearn library. As parameters you need to mention, test_size which will tell the ratio how much data need to be assigned with test data. The status as of now should look like as in figure 3:



```
print("Sizes of X_train, Y_train, X_test, Y_test")
print(len(X_train))
print(len(X_test))
print(len(Y_train))
print(len(Y_test))
```

```
Sizes of X_train, Y_train, X_test, Y_test
1078
120
1078
120
```

*Figure 3 Train-Test-Split from Sklearn*

## ➔ Training

With the help of Sklearn library we can import SVC – Support Vector Classifier. The SVC() function trains the Training data `model.fit(X_train, Y_train)` and with the help of `model.predict` we can predict the test data. In order to evaluate the model we perform the training on training dataset such that first we evaluate the model on test dataset and also the training dataset itself. It is very intuitive that it will perform well when predicted with training data. And, with low accuracy on test data. One thing to note here is that I used various types of kernels in the SVM Learning process. I tried "rbf", "linear" and "ploy". Each of them are describes as: RBF – Radial Basis Functionis kind of gaussian methodology where the data is presented in higher dimension or in gassian space. Also it performs best when classes are separable. Linear Kernel also applied on linearly separable data but have poor optimization. Moreover on checking with multiple kernels the rbf performed better. `decision_function_shape='ovo'` Defines one over one classifier. In other words it identifies the underlying data and creates classifiers classwise, that is the inside classification is being done by checking the class vs class process. In our case since we have total 10 classes, the inner classifier has created a total of 10 choose 2 that is 45 different classifiers separating 2 classes. On evaluation the result we got is `Testing accuracy: 0.333` and `Training accuracy: 0.7597`. This was done for overall accuracy. Now to find the class wise accuracy we will perform the training explicitly giving only 2 classes to the model. So that each time our model learns 2 classes and calculates the accuracy.

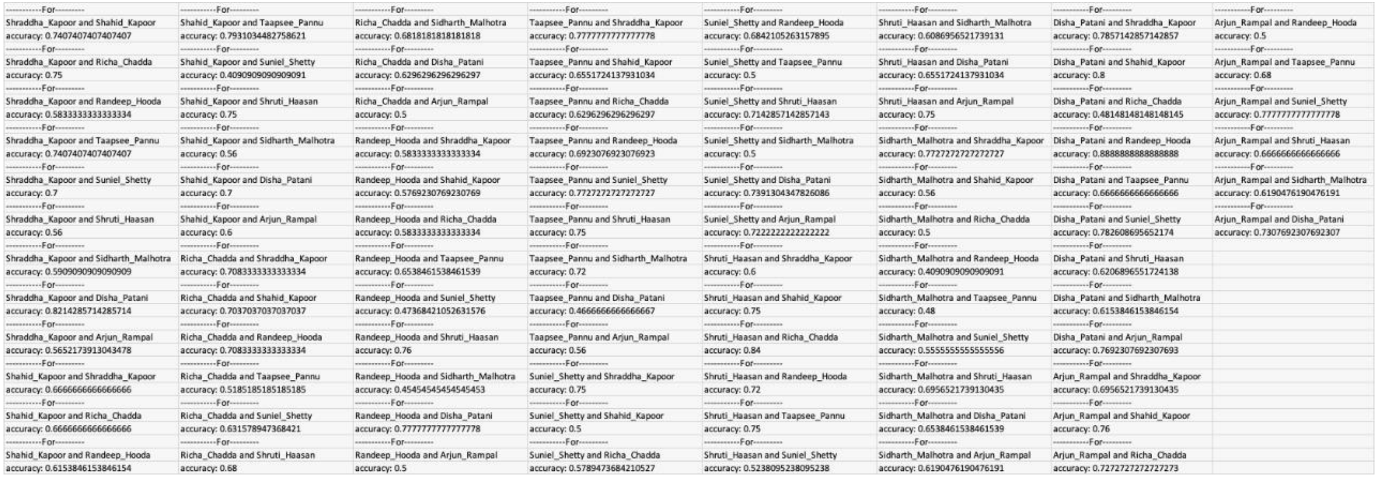
```
------------------------------------------------
-----------Doing for ---------
Shraddha_Kapoor and Shahid_Kapoor
------------------------------------------------
0
1
264
Sizes of X_train, Y_train, X_test, Y_test
237
27
237
27
accuracy: 0.7407407407407407
------------------------------------------------
-----------Doing for ---------
Shraddha_Kapoor and Richa_Chadda
------------------------------------------------
0
1
237
Sizes of X_train, Y_train, X_test, Y_test
213
24
```

*Figure 4 Text Box with classwise training results*

On further checking singleton results the results were not so good. The model mis-classifies most of the stars. Even on multiple EPOCHS, the training results were not so good.

**➔ Results and Inferences:**

Overall Training Accuracy = ~76%
Overall Testing Accuracy = ~33%
Class-wise Accuracies are attached as text box in figure 4.

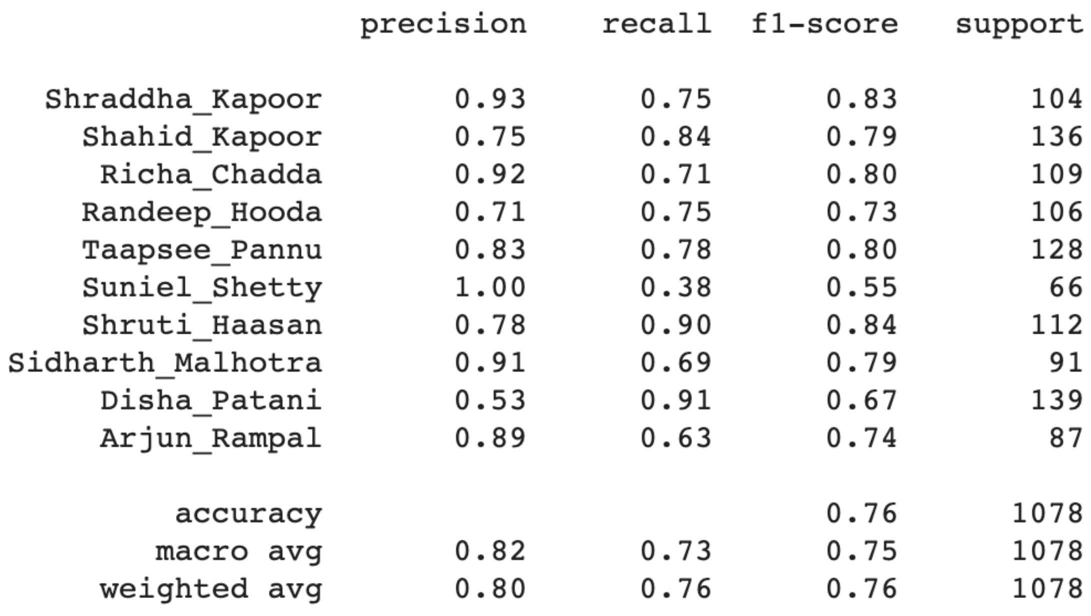


```
-----------For---------
Shraddha_Kapoor and Shahid_Kapoor
accuracy: 0.7407407407407407
-----------For---------
Shraddha_Kapoor and Richa_Chadda
accuracy: 0.75
-----------For---------
Shraddha_Kapoor and Randeep_Hooda
accuracy: 0.5833333333333334
-----------For---------
Shraddha_Kapoor and Taapsee_Pannu
accuracy: 0.7407407407407407
-----------For---------
Shraddha_Kapoor and Suniel_Shetty
accuracy: 0.7
-----------For---------
Shraddha_Kapoor and Shruti_Haasan
accuracy: 0.56
-----------For---------
Shraddha_Kapoor and Sidharth_Malhotra
accuracy: 0.5909090909090909
-----------For---------
Shraddha_Kapoor and Disha_Patani
accuracy: 0.8214285714285714
-----------For---------
Shraddha_Kapoor and Arjun_Rampal
accuracy: 0.5652173913043478
-----------For---------
Shahid_Kapoor and Shraddha_Kapoor
accuracy: 0.6666666666666666
-----------For---------
Shahid_Kapoor and Richa_Chadda
accuracy: 0.6666666666666666
-----------For---------
Shahid_Kapoor and Randeep_Hooda
accuracy: 0.6153846153846154
-----------For---------
Shahid_Kapoor and Taapsee_Pannu
accuracy: 0.7931034482758621
-----------For---------
Shahid_Kapoor and Suniel_Shetty
accuracy: 0.4090909090909091
-----------For---------
Shahid_Kapoor and Shruti_Haasan
accuracy: 0.75
-----------For---------
Shahid_Kapoor and Sidharth_Malhotra
accuracy: 0.56
-----------For---------
Shahid_Kapoor and Disha_Patani
accuracy: 0.7
-----------For---------
Shahid_Kapoor and Arjun_Rampal
accuracy: 0.6
-----------For---------
Richa_Chadda and Shraddha_Kapoor
accuracy: 0.7083333333333334
-----------For---------
Richa_Chadda and Shahid_Kapoor
accuracy: 0.7037037037037037
-----------For---------
Richa_Chadda and Randeep_Hooda
accuracy: 0.7083333333333334
-----------For---------
Richa_Chadda and Taapsee_Pannu
accuracy: 0.5185185185185185
-----------For---------
Richa_Chadda and Suniel_Shetty
accuracy: 0.631578947368421
-----------For---------
Richa_Chadda and Shruti_Haasan
accuracy: 0.68
-----------For---------
Richa_Chadda and Sidharth_Malhotra
accuracy: 0.6818181818181818
-----------For---------
Richa_Chadda and Disha_Patani
accuracy: 0.6296296296296297
-----------For---------
Richa_Chadda and Arjun_Rampal
accuracy: 0.5
-----------For---------
Randeep_Hooda and Shraddha_Kapoor
accuracy: 0.5833333333333334
-----------For---------
Randeep_Hooda and Shahid_Kapoor
accuracy: 0.5769230769230769
-----------For---------
Randeep_Hooda and Richa_Chadda
accuracy: 0.5833333333333334
-----------For---------
Randeep_Hooda and Taapsee_Pannu
accuracy: 0.6538461538461539
-----------For---------
Randeep_Hooda and Suniel_Shetty
accuracy: 0.47368421052631576
-----------For---------
Randeep_Hooda and Shruti_Haasan
accuracy: 0.76
-----------For---------
Randeep_Hooda and Sidharth_Malhotra
accuracy: 0.45454545454545453
-----------For---------
Randeep_Hooda and Disha_Patani
accuracy: 0.7777777777777778
-----------For---------
Randeep_Hooda and Arjun_Rampal
accuracy: 0.5
-----------For---------
Taapsee_Pannu and Shraddha_Kapoor
accuracy: 0.7777777777777778
-----------For---------
Taapsee_Pannu and Shahid_Kapoor
accuracy: 0.6551724137931034
-----------For---------
Taapsee_Pannu and Richa_Chadda
accuracy: 0.6296296296296297
-----------For---------
Taapsee_Pannu and Randeep_Hooda
accuracy: 0.6923076923076923
-----------For---------
Taapsee_Pannu and Suniel_Shetty
accuracy: 0.7727272727272727
-----------For---------
Taapsee_Pannu and Shruti_Haasan
accuracy: 0.75
-----------For---------
Taapsee_Pannu and Sidharth_Malhotra
accuracy: 0.72
-----------For---------
Taapsee_Pannu and Disha_Patani
accuracy: 0.4666666666666667
-----------For---------
Taapsee_Pannu and Arjun_Rampal
accuracy: 0.56
-----------For---------
Suniel_Shetty and Shraddha_Kapoor
accuracy: 0.75
-----------For---------
Suniel_Shetty and Shahid_Kapoor
accuracy: 0.5
-----------For---------
Suniel_Shetty and Richa_Chadda
accuracy: 0.5789473684210527
-----------For---------
Suniel_Shetty and Randeep_Hooda
accuracy: 0.6842105263157895
-----------For---------
Suniel_Shetty and Taapsee_Pannu
accuracy: 0.5
-----------For---------
Suniel_Shetty and Shruti_Haasan
accuracy: 0.7142857142857143
-----------For---------
Suniel_Shetty and Sidharth_Malhotra
accuracy: 0.5
-----------For---------
Suniel_Shetty and Disha_Patani
accuracy: 0.7391304347826086
-----------For---------
Suniel_Shetty and Arjun_Rampal
accuracy: 0.7222222222222222
-----------For---------
Shruti_Haasan and Shraddha_Kapoor
accuracy: 0.6
-----------For---------
Shruti_Haasan and Shahid_Kapoor
accuracy: 0.75
-----------For---------
Shruti_Haasan and Richa_Chadda
accuracy: 0.84
-----------For---------
Shruti_Haasan and Randeep_Hooda
accuracy: 0.72
-----------For---------
Shruti_Haasan and Taapsee_Pannu
accuracy: 0.75
-----------For---------
Shruti_Haasan and Suniel_Shetty
accuracy: 0.5238095238095238
-----------For---------
Shruti_Haasan and Sidharth_Malhotra
accuracy: 0.6086956521739131
-----------For---------
Shruti_Haasan and Disha_Patani
accuracy: 0.6551724137931034
-----------For---------
Shruti_Haasan and Arjun_Rampal
accuracy: 0.75
-----------For---------
Sidharth_Malhotra and Shraddha_Kapoor
accuracy: 0.7727272727272727
-----------For---------
Sidharth_Malhotra and Shahid_Kapoor
accuracy: 0.56
-----------For---------
Sidharth_Malhotra and Richa_Chadda
accuracy: 0.5
-----------For---------
Sidharth_Malhotra and Randeep_Hooda
accuracy: 0.4090909090909091
-----------For---------
Sidharth_Malhotra and Taapsee_Pannu
accuracy: 0.48
-----------For---------
Sidharth_Malhotra and Suniel_Shetty
accuracy: 0.5555555555555556
-----------For---------
Sidharth_Malhotra and Shruti_Haasan
accuracy: 0.6956521739130435
-----------For---------
Sidharth_Malhotra and Disha_Patani
accuracy: 0.6538461538461539
-----------For---------
Sidharth_Malhotra and Arjun_Rampal
accuracy: 0.6190476190476191
-----------For---------
Disha_Patani and Shraddha_Kapoor
accuracy: 0.7857142857142857
-----------For---------
Disha_Patani and Shahid_Kapoor
accuracy: 0.8
-----------For---------
Disha_Patani and Richa_Chadda
accuracy: 0.48148148148148145
-----------For---------
Disha_Patani and Randeep_Hooda
accuracy: 0.8888888888888888
-----------For---------
Disha_Patani and Taapsee_Pannu
accuracy: 0.6666666666666666
-----------For---------
Disha_Patani and Suniel_Shetty
accuracy: 0.782608695652174
-----------For---------
Disha_Patani and Shruti_Haasan
accuracy: 0.6206896551724138
-----------For---------
Disha_Patani and Sidharth_Malhotra
accuracy: 0.6153846153846154
-----------For---------
Disha_Patani and Arjun_Rampal
accuracy: 0.7692307692307693
-----------For---------
Arjun_Rampal and Shraddha_Kapoor
accuracy: 0.6956521739130435
-----------For---------
Arjun_Rampal and Shahid_Kapoor
accuracy: 0.76
-----------For---------
Arjun_Rampal and Richa_Chadda
accuracy: 0.7272727272727273
-----------For---------
Arjun_Rampal and Randeep_Hooda
accuracy: 0.5
-----------For---------
Arjun_Rampal and Taapsee_Pannu
accuracy: 0.68
-----------For---------
Arjun_Rampal and Suniel_Shetty
accuracy: 0.7777777777777778
-----------For---------
Arjun_Rampal and Shruti_Haasan
accuracy: 0.6666666666666666
-----------For---------
Arjun_Rampal and Sidharth_Malhotra
accuracy: 0.6190476190476191
-----------For---------
Arjun_Rampal and Disha_Patani
accuracy: 0.7307692307692307
```

*Figure 5 Output of Classwise training and accuracies*

Two Metrics used are:

1. Classification report (from sklearn library)
2. Confusion Matrix (from sklearn library)



```
                   precision    recall  f1-score   support

  Shraddha_Kapoor       0.93      0.75      0.83       104
    Shahid_Kapoor       0.75      0.84      0.79       136
     Richa_Chadda       0.92      0.71      0.80       109
    Randeep_Hooda       0.71      0.75      0.73       106
    Taapsee_Pannu       0.83      0.78      0.80       128
    Suniel_Shetty       1.00      0.38      0.55        66
    Shruti_Haasan       0.78      0.90      0.84       112
Sidharth_Malhotra       0.91      0.69      0.79        91
     Disha_Patani       0.53      0.91      0.67       139
     Arjun_Rampal       0.89      0.63      0.74        87

         accuracy                           0.76      1078
        macro avg       0.82      0.73      0.75      1078
     weighted avg       0.80      0.76      0.76      1078
```

*Figure 6 Training Metrics*

From figure 6 defines the key metrics of the classification problem of training dataset. To define the terminologies, **precision** is the accuracy within class or in other words, how many

are correctly classified within that class. That is you predicted positive and it turned out to be positive in labels as well. Precision = number of positives / total positive labels.
For example, for Shraddha Kapoor 93 examples out of 100 samples are correctly classified or for Richa Chadda 92 examples out of 100 samples are correctly classified where 100 samples are from their respective classes. Accuracy = # of correct classifications / total examples. And **recall** can be defined as number of positives given that the labels are positive or the ground truth is also positive and the predicted output is also positive.

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

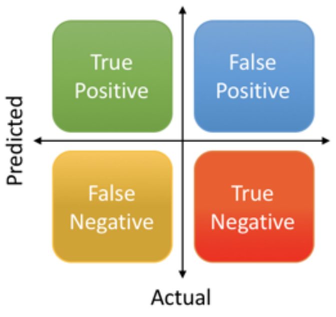
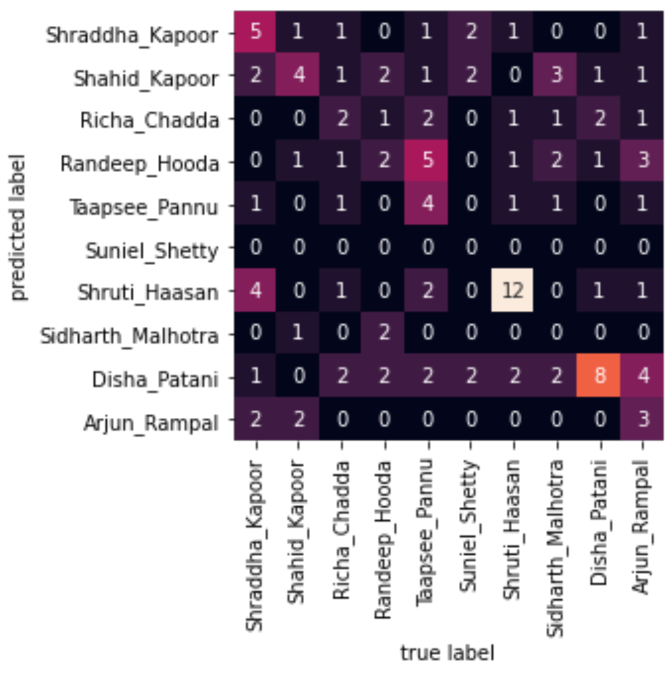


*Figure 7 Precision Recall chart*

In more simpler words, **Precision** is the number of times the actor (say Shraddha Kapoor) got correctly classified divided by number of times the model assumed it was Shraddha Kapoor. However **Recall** is number of times model classified it was Shraddha Kapoor divided by the number of times it was actually Shraddha Kapoor. **F1** score is the average of precision and recall. In addition, support is the amount of percentage of that class present in the dataset (here it is training set).



| | Shraddha_Kapoor | Shahid_Kapoor | Richa_Chadda | Randeep_Hooda | Taapsee_Pannu | Suniel_Shetty | Shruti_Haasan | Sidharth_Malhotra | Disha_Patani | Arjun_Rampal |
|---|---|---|---|---|---|---|---|---|---|---|
| Shraddha_Kapoor | 5 | 1 | 1 | 0 | 1 | 2 | 1 | 0 | 0 | 1 |
| Shahid_Kapoor | 2 | 4 | 1 | 2 | 1 | 2 | 0 | 3 | 1 | 1 |
| Richa_Chadda | 0 | 0 | 2 | 1 | 2 | 0 | 1 | 1 | 2 | 1 |
| Randeep_Hooda | 0 | 1 | 1 | 2 | 5 | 0 | 1 | 2 | 1 | 3 |
| Taapsee_Pannu | 1 | 0 | 1 | 0 | 4 | 0 | 1 | 1 | 0 | 1 |
| Suniel_Shetty | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Shruti_Haasan | 4 | 0 | 1 | 0 | 2 | 0 | 12 | 0 | 1 | 1 |
| Sidharth_Malhotra | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Disha_Patani | 1 | 0 | 2 | 2 | 2 | 2 | 2 | 2 | 8 | 4 |
| Arjun_Rampal | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |

*Figure 8 Confusion Matrix for Celeb data*

**Confusion Matrix** is also one of the metrics to check how our model performed overall. When analyzed, it is clearly seen as the model performed best on Shruti Haasan. We can find following conclusions:

1. The model predicted Shruti Haasan 12 times correctly. That is the model said the sample is Shruti Haasan and in ground truth it is being Shruti Haasan.
2. The model predicted Disha 8 times correctly. That is the model said the sample is Disha and in ground truth it is being Disha.
3. The model predicted Tapsee Pannu as Randeep Hooda 5 times. Which is more than its true positive (that is 4).
4. Even if you calculate the prediction row of Disha Patani the row sum is highest of Disha Patani. Additionally, in the training data we had the highest number of Disha Patani examples of images. This is clearly a bias towards Disha Patani being predicted for multiple actors.
5. In second observation, of Shruti Haasan, there is a bias towards it. This can be checked as Shraddha Kapoor being classified as Shruti Haasan.
6. Seems like For Suniel Shetty the model learned nothing. The model predicted Suneil Shetty being Suniel Shetty is 0 and model predicted Suniel Shetty being Disha Patani is 2. Clearly a bias towards Disha Patani.

Of course there are numerous observations that can be carried from the data. I have mentioned the highest possibilities of bias towards some classes.

DAI_ASSIGNMENT_1_Q2_Report.pdf

**Dependable AI CSL7370 – Assignment 1**

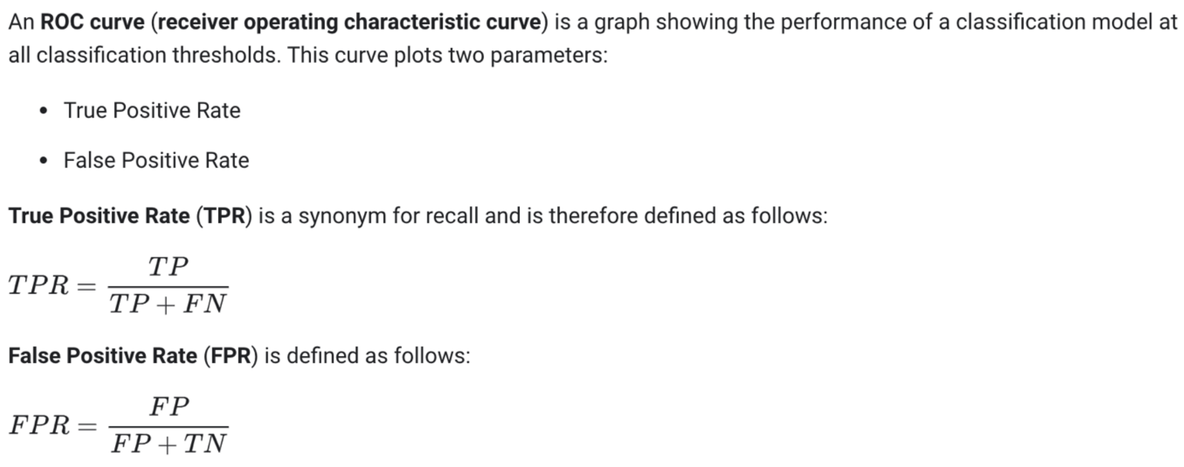
# Task 2:

## Contents:

## Theory

1. **Bias:** Originally, bias is anything that if its value is high then the Machine Learning model will oversimplify the training and test cases. Bias in machine learning models turns high error. "*Model with high bias pays very less attention to the training data.*" In paradigm of Dependable AI we introduce bias as a partiality of a machine learning model towards the query data. This is because the model has learned such patterns during the training. May be because of the training samples are inadequate or called as data bias. Apart from this there are numerous biases devised out by research community. Some of them are: Representation bias, Evaluation bias, Population bias, Sample bias, Prejudice bias, Confirmation bias, Hypothesis bias. Specifically, Bias in data includes, Data Gathering or Geographical factors bias (example where data is gathered specifically from freely available resources mainly western country data and then applied on different parts of the world), Data pre-processing bias, Confirmation bias.
2. **Fashion MNIST**: Originated from a European based E-commerce company, the dataset is freely available with MNIST. Zalando SE published 10 class of items with a total of 70000 example images (28x28) spread over 10 classes. The dataset can be accessed bias tensor-flow package as well.
3. **Cross Validation**: Evaluating the test dataset (How the model works on an unseen dataset) Sometimes with limited data examples we resample the same dataset with different thresholds such that we get to learn more knowledge about the dataset. One can think that cross validation will overfit the dataset since it is learning complete dataset iteratively. Howebver this is not the case, because you are not considering whole dataset at a time. You try to create multiple train test splits so as to fine tune your learning model.
4. **Linear SVM**: A faster Support Vector Machine model with linear kernel. (Mostly used when the data is linearly separable.)
5. **5 Layer ANN/MLP**: Before jumping what we require to use in our dataset let us know the differences between MLP and ANN. MLP is a subset of ANN. Most of the times both words are used interchangeably. The inherent difference is that, MLP are always feed forward while ANN can have loops. More on : here
6. **Testing Performance** (mean ± STD)**:** Mean accuracy and Deviation tells the metrics about the average accuracy of the model (Since we trained with 3 different training sets our each of the SVM and ANN models). This tells the average accuracy of the models. Additionally, Deviation is the percentage amount that the model is not performing well.
7. **Confusion matrix:** Confusion Matrix tells about the true positives, true negatives, false positive, false negatives. In simple terms, Confusion matrix tells about how many times "Coat" class is classified as pullover and coat actually. And same for "Pullover". This is best way to evaluate your model. That is How many times our model was confusion with pullover but it was coat in reality. This is where we need to create a predicted label array along with the original labels. So as to identify the parameters.

8. **TP,FP,TN,FN, TPR, FPR,TNR,FNR:** for better performance, TPR and TNR should be high and rest low. Because we want prediction to be similar to the ground truth.
    - **a.** True Positive: # of times coat is predicted as coat
    - **b.** False Positive: # of times pullover is predicted as coat
    - **c.** True Negative: # of times pullover is predicted as pullover
    - **d.** False Negative: # of times coat is predicted as pullover.
    - **e.** True Positive Rate: #tp/#tp+#fn
    - **f.** False Positive Rate: #fp/#tn+#fp
    - **g.** True Negative Rate: #tn/#tn+#fp
    - **h.** False negative Rate: #fn/#tp+#fn
9. ROC Curve and its importance: ROC Curve is simpler measure of confusion matrices. Receiver Operator Characteristics curve is simple way to summarize. The vertical axis of the graph is sensitivity (TPR) and x axis is (1-specificity) FPR with varying thresholds. Threshold is a kind of decision boundary which enables the samples to classify into their respective classes. With increasing threshold you allow more samples of class 0 (in our case 0 is pullover) The importance of ROC curve is that, Classifiers that give curves closer to the top-left corner indicate a better performance. We envision higher TPR because we want more predictions to match with ground truth. ROC curve is suitable for balanced datasets. Because ROC curve may give you wrong hopes of model being accurate. However in reality you were dealing with 90% data of coats and 10% data of pullover.


An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

**True Positive Rate** (**TPR**) is a synonym for recall and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

**False Positive Rate** (**FPR**) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$

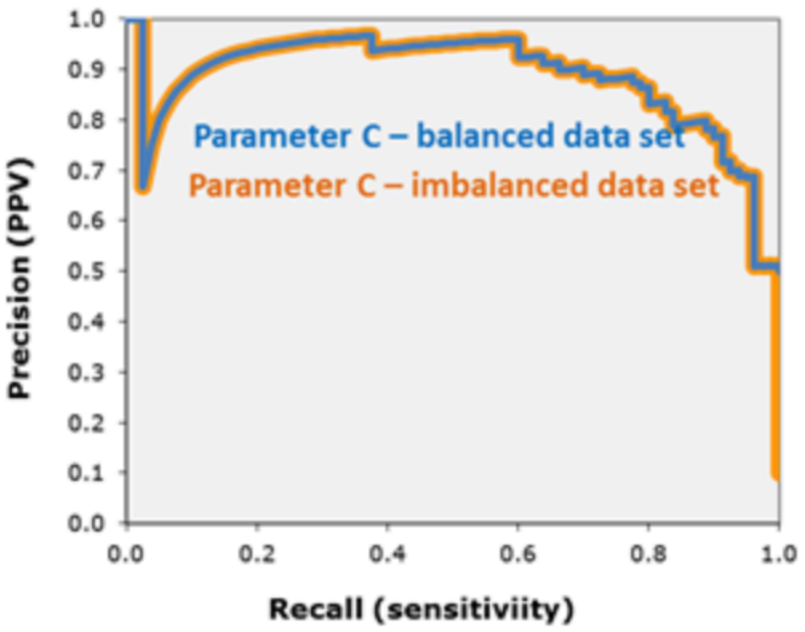
*Figure 1 from google developers machine learning crash course*

10. EER Equal Error Rate and curve importance: The point where the graph intersects is the point where the # of false acceptance is equal to # of false rejections. The lower the equal error rate value, the higher the accuracy of the out model. Below the intersection point, we accept low false predictions and higher false rejections, and above the intersection point, we accept more false predictions and less false rejections.
11. Precision Recall and curve importance: Suitable for imbalanced data. (From question 1) In more simpler words, **Precision** is the number of times the coat got correctly classified divided by number of times the model assumed it was coat. However (TPR) **Recall** is number of times model classified it was coat divided by the number of times it was actually coat.

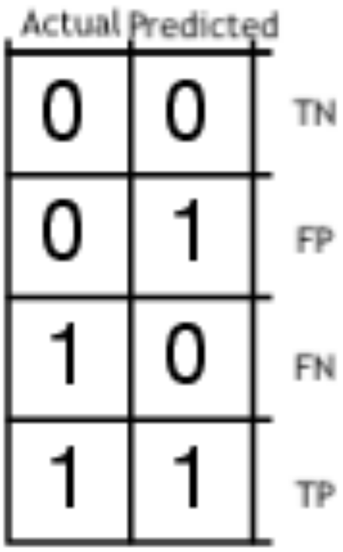
$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

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

12. PR Curve is a plot of Precision (vertical axis) and Recall (horizontal axis) measures at different thresholds. Because of this we can rely on PR Curve for imbalanced dataset. Even if the dataset is not balanced or the dataset is biased, we will not deal with different proportions of the data. We will check how our model performed within that particular class. As precision checks for correctness of the classification to the assumption for that class of classifier whereas recall checks for correctness of the classification to actual ground truth of that class of classifier. In general PR Curve overlaps for balanced and imbalanced datasets. However, the PR curve drop point tells the proportion of the classes you incorporated. In simpler terms, PR Curve tells about the biasness by the droppage of the curve to some value very high or very low according to the class your envision. However for balanced dataset it drops at 0.5. Additionally, we can say that when recall = 1, we shall check the precision such that we get to know about balanced and imbalanced datasets.



Whereas it is not a possible way to detect biasness in ROC Curves. Because of the biased dataset you will see a beautiful and perfect ROC Curve such that the curve for balanced dataset will be lower top left than to the curve of imbalanced dataset which will be higher top left. Thus we would first look at the recall values, then precision values. We are more cantered towards the actual values.



| Actual | Predicted | |
|---|---|---|
| 0 | 0 | TN |
| 0 | 1 | FP |
| 1 | 0 | FN |
| 1 | 1 | TP |

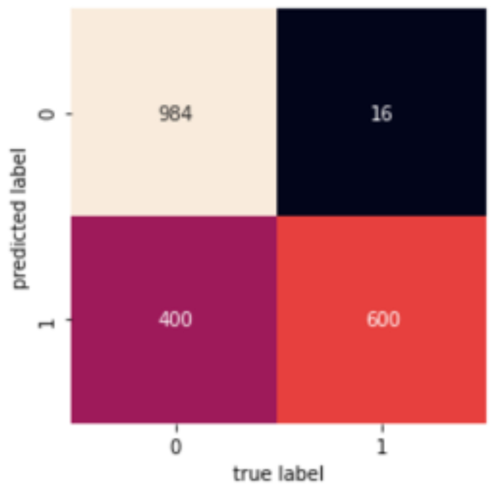
## Results

**1 and 2**
**Testing Performance (Mean ± STD) and** Testing Accuracy comparison of SVM vs ANN:
**SVM1 Accuracy = 71.3%**
**SVM2 Accuracy = 73.2%**
**SVM3 Accuracy = 69.55%**
**Mean Accuracy=** (71.3+73.2+69.55)/3 = **71.35%**
**SVM1 Deviation = 71.3%-71.35% = -0.5**
**SVM2 Deviation = 73.2%-71.35% = 1.85**
**SVM3 Deviation = 69.55%-71.35% = -1.8 (This one was only shuffled)** clearly we can see the change/deviation with shuffled data

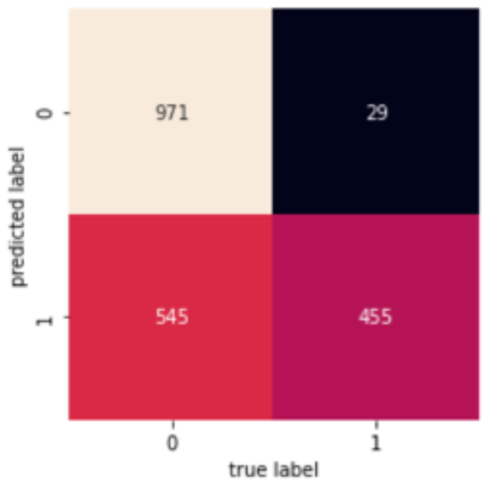
**ANN1 Accuracy = 79.2%**
**ANN2 Accuracy = 80.9%**
**ANN3 Accuracy = 75.75%**
**Mean Accuracy=** (**79.2**+**80.9**+**75.75**)/3 = 78.616%
**ANN1 Deviation = 79.2%-**78.616%**= 0.584**
**ANN2 Deviation = 80.9%-**78.616%**= 2.284**
**ANN3 Deviation = 75.75%-**78.616% **= -2.866 (This one was only shuffled)** clearly we can see the change/deviation with shuffled data

**Confusion Matrices**

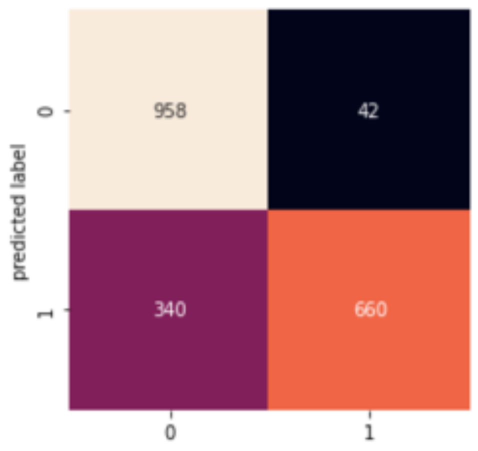
**ANN1**



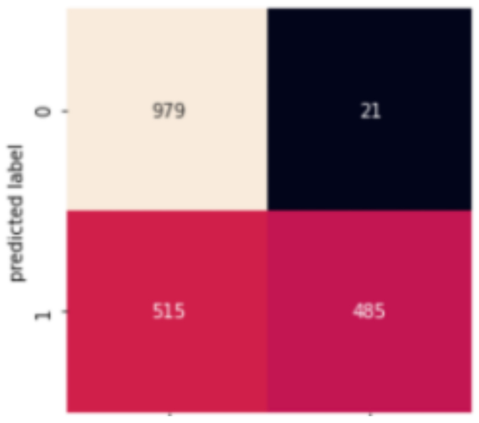
**SVM1**



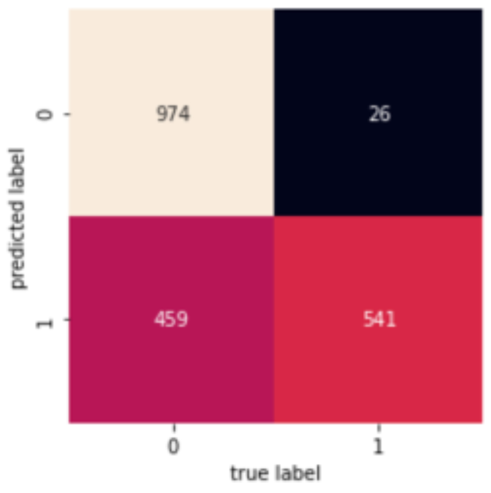
**ANN2**



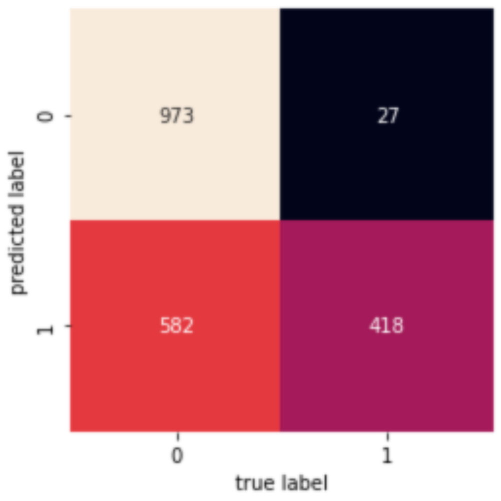
**SVM2**



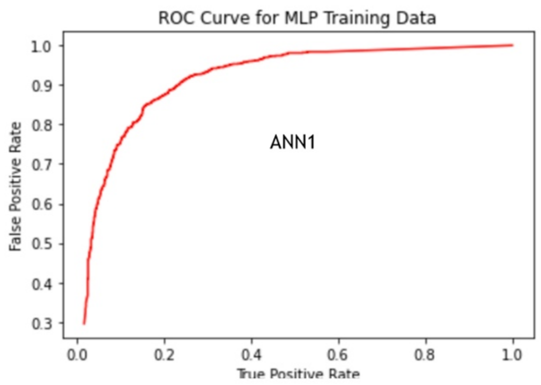
**ANN3**

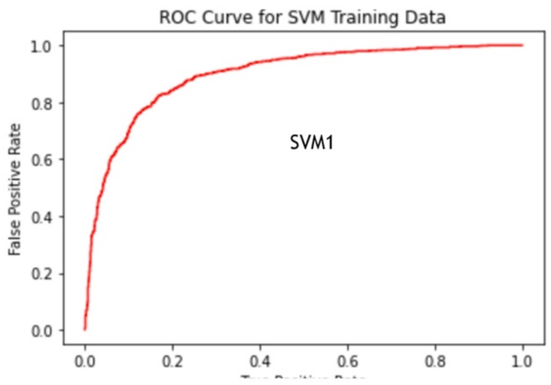


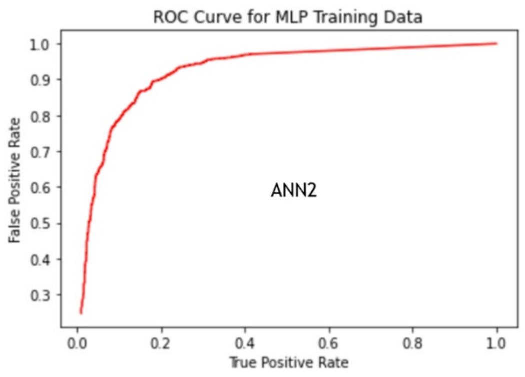
**SVM3**

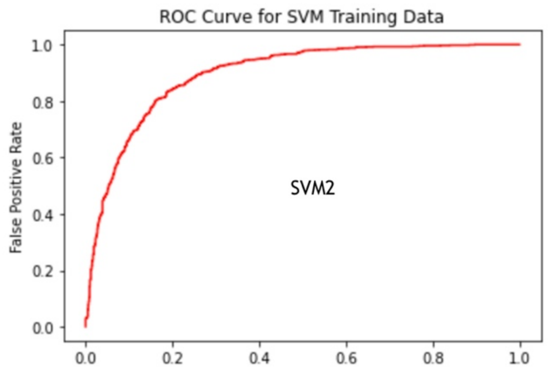


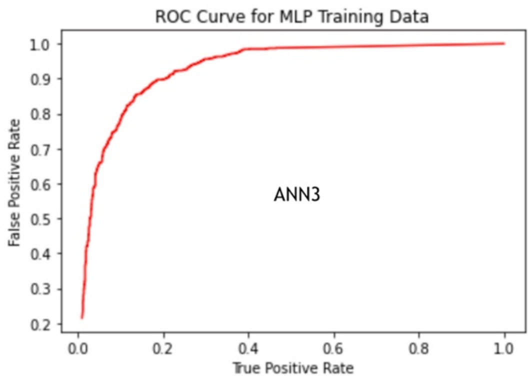
ROC Curves:

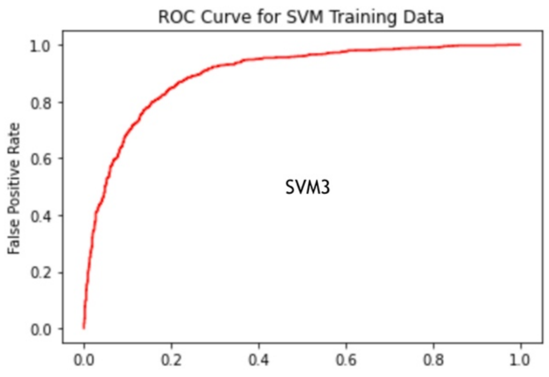

ROC Curve for MLP Training Data
ANN1
False Positive Rate
True Positive Rate
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve for SVM Training Data
SVM1
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

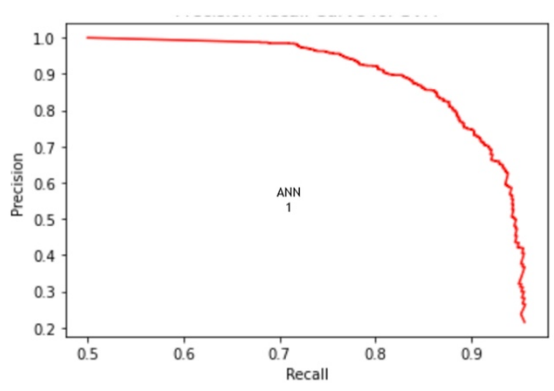

ROC Curve for MLP Training Data
ANN2
False Positive Rate
True Positive Rate
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0

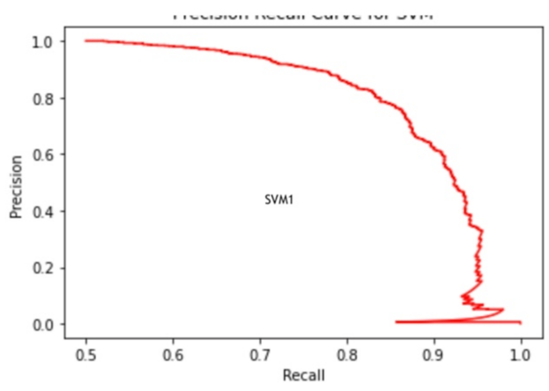

ROC Curve for SVM Training Data
SVM2
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

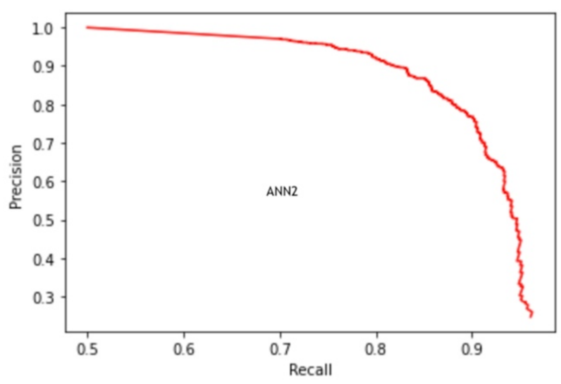

ROC Curve for MLP Training Data
ANN3
False Positive Rate
True Positive Rate
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
0.2
0.4
0.6
0.8
1.0

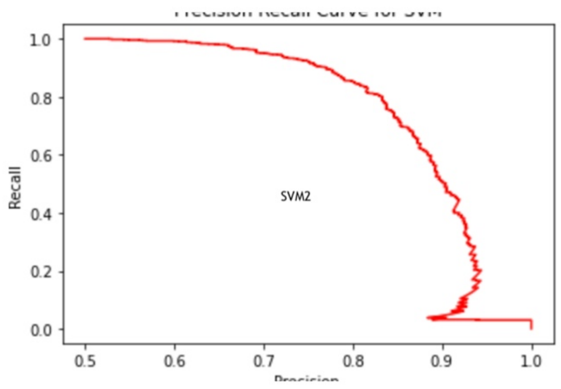

ROC Curve for SVM Training Data
SVM3
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

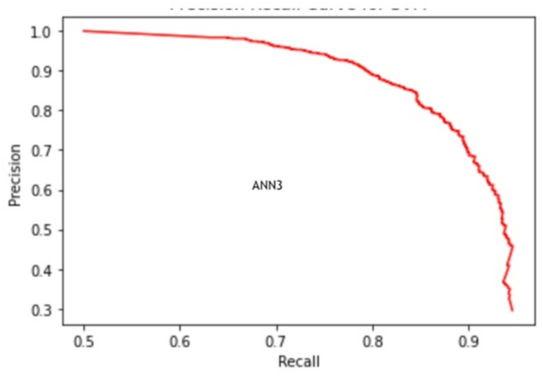
PR Curves:

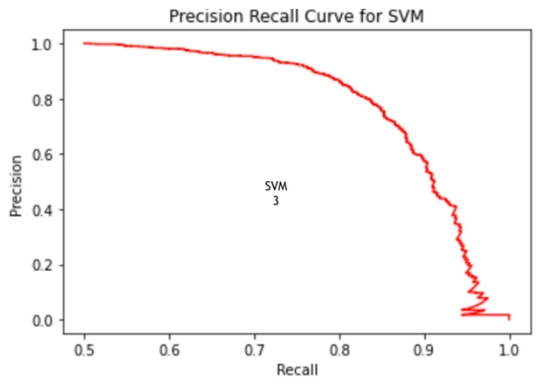












Inferences:

From the complete experiment we come to conclusions:

1. Having large number of samples of a specific class creates a bias.
2. DNN performs better than SVM because Linear SVMs just uses a basic hyperplane methodologies and DNN learns the features. Yes, you can create non linear hyperplanes with different kernels however, DNN still outperforms without overfitting the dataset.
3. When we shuffled the dataset we got smaller deviations than non shuffled dataset.
4. ROC, PR and Confusion matrix are drawn to evaluate the metrics of different models.
5. Out of which we use ROC for balanced dataset and PR Curve for Imbalanced dataset.

Reference:

1. http://www.davidsbatista.net/blog/2018/08/19/NLP_Metrics/

2. https://acutecaretesting.org/en/articles/precision-recall-curves-what-are-they-and-how-are-they-used

DAI_ASSIGNMENT_1_Q3_Report.pdf

**Dependable AI CSL7370 – Assignment 1**

# Task 3:

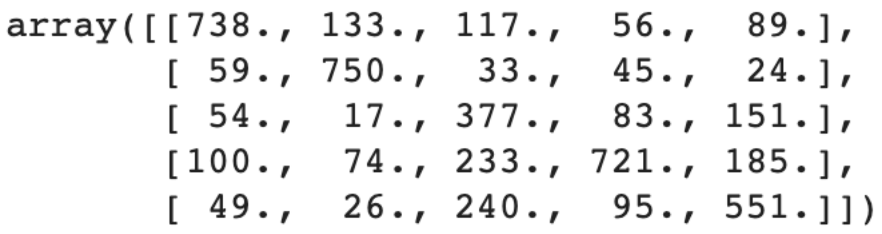
➔ **Preprocessing Data:**

This step includes fetching of data from library of tensorflow. Steps included are as follows:

1. Importing the datasets `from tensorflow.keras import datasets` using this command and then load cifar10 dataset. This process is simple as you just have to load the dataset into variables `(train_images, train_labels), (test_images, test_labels)`. Till this step we are splitting the dataset into variables. 50000 in training and 10000 in testing.
2. Now we need to drop 5 classes of choice. (In my case I have dropped bottom 5 classes)
3. In order to drop the classes, you can also find the indices of the classes which you need to incorporate for training and learning.
4. Indices of the requires classes are used to filter the dataset and fetch only those tuples which align to the respective index. This will get us all samples from the dataset of that particular class.
5. Similarly all training, testing datasets can be filtered as per requirement.
6. Once this process is done, you can concatenate for the sake of collaborative learning process. Such that, our X_train, Y_train, X_test and Y_test will have 25000, 25000, 5000, 5000 samples respectively.
7.

8. Just to visualize the image we define a function : `show_label_and_image`
9. Now that we have training and testing dataset, we can proceed to training process. The training process includes 3 layers in a simple Sequential Artificial Neural Network.
10. Additionally, the model uses Stochastic Gradient Descent to learn the updates. Loss used is CEL. This is preferred for classification datasets. To show the learning metrics we display accuracy measure. [Epochs = 10]
11. To print confusion matrix there are 3 methods. First: using inbuilt function (from sklearn), Second: using pandas.crosstab, Thirdly: from scratch using for loop
12. We shall focus on third method (I have solved with first 2 methods in notebook).
13. From scratch we just require to start with 2D array. Initialize 2D array using np.zeros. using : `np.zeros((len(labels), len(labels)))`
14. Once done we can start iterating rows and columns such that storing the counts of each comparison from predicted label and the truth label by comparing. The

condition returns 0 if not match or 1 if match. `np.sum((Y_actual == labels[i]) & (Y_pred == labels[j]))` This line of code will store the sums of “1” in each iteration and updates the 2D array. We are checking 2 conditions here, first one will check whether the actual label and the label in our row iteration matches. Second condition checks whether predicted label matches with the label in our column iteration. (The comparison was possible because of comparing numpy array with numpy integer and not with native python integer)



```
array([[738., 133., 117.,  56.,  89.],
       [ 59., 750.,  33.,  45.,  24.],
       [ 54.,  17., 377.,  83., 151.],
       [100.,  74., 233., 721., 185.],
       [ 49.,  26., 240.,  95., 551.]])
```