**CS584 Machine Learning**

**Project Report**

Saurabh Patni

A20428510

Illinois Institute of Technology

**Phase 1:**

Firstly, we took the data which consisted of around 13,233 images of size 64 X 64, and grey scale photos of random people including those of Bush and Williams.

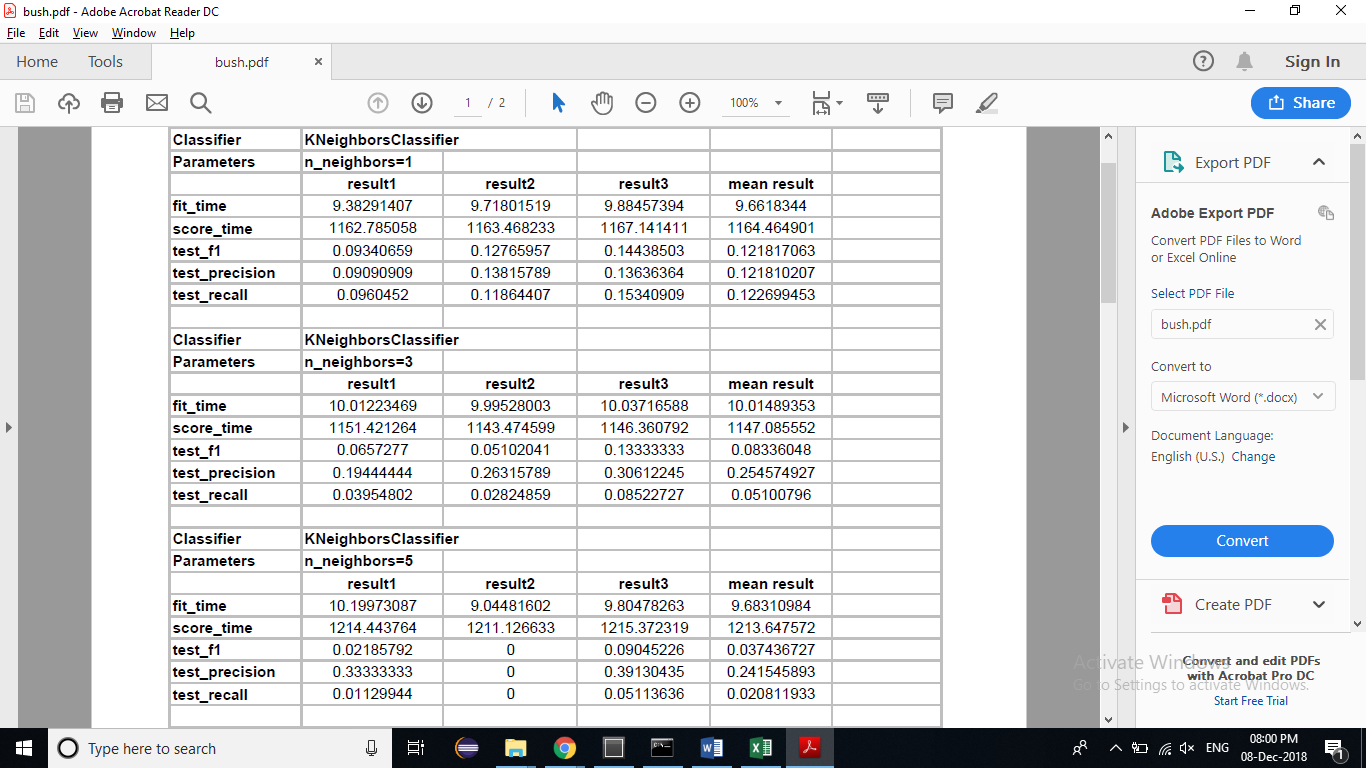
We split the data into training set and test set using three-fold cross validation which splits 2/3rd data into training set and rest in test set. The cross validation gives us the records of precision, recall, and test f1. When we set n\_jobs parameter to -1, it helps us perform the experiment faster.

We than use the classifiers - KNeighbour Classifier, a useful technique which is used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

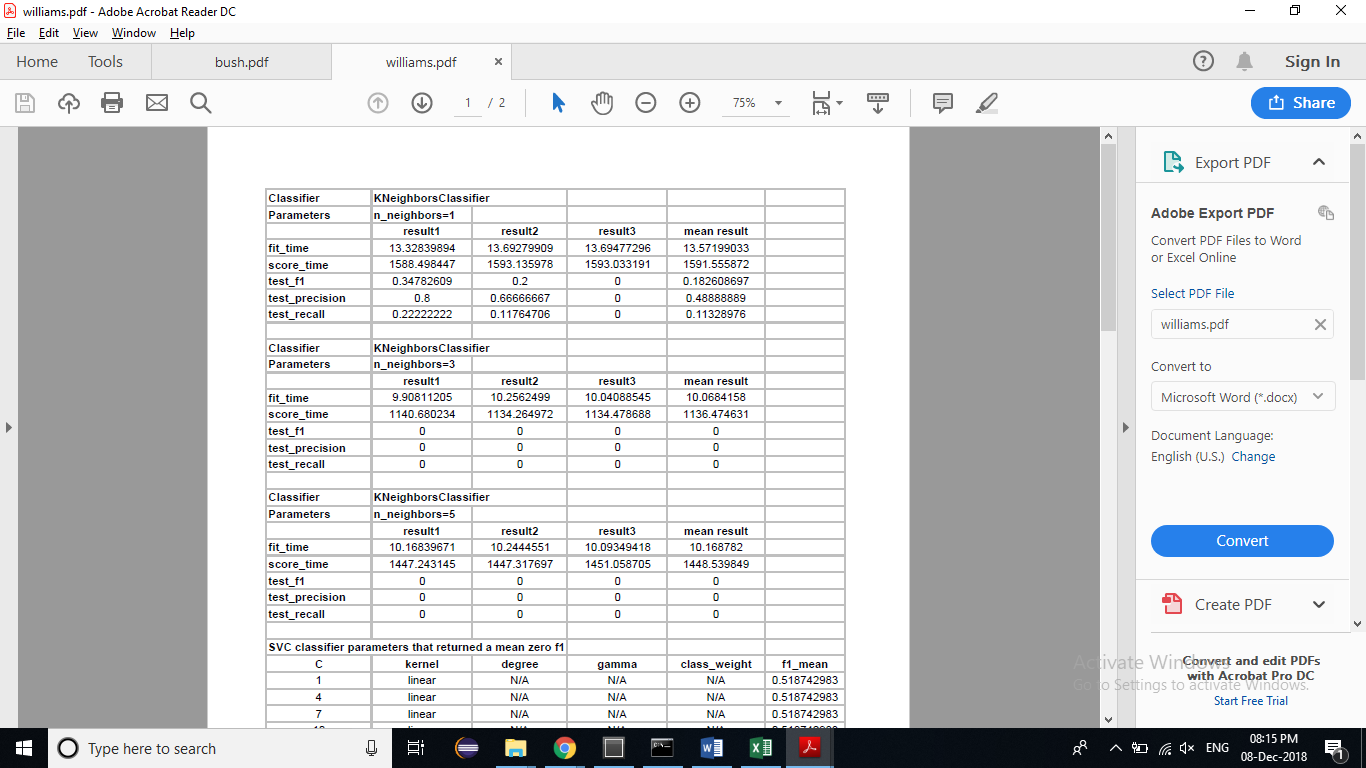
The neighbors are taken from a set of objects for which the class or the object property value is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

We have used 3 different values of the parameter n\_neighbours, i.e. 1, 3 and 5 and computed the result. We observed that for the case of Bush, the mean F1 score was observed to be 0.1218 when we kept the value of n\_neighbours = 1. It kept on decreasing as we increased the value of n\_neighbours. The same result was observed for mean recall. It kept on decreasing when we increase the values of n\_neighbours from 1 to 5. While the mean precision gave fluctuating results as firstly the value was 0.1218 when n\_neighbours was 1. It increased when the value of n\_neighbours was 3, while decreased a bit when the value of n\_neighbour was equal to 5.

Following are the results which we computed when we applied knn classifier on the data with bush.



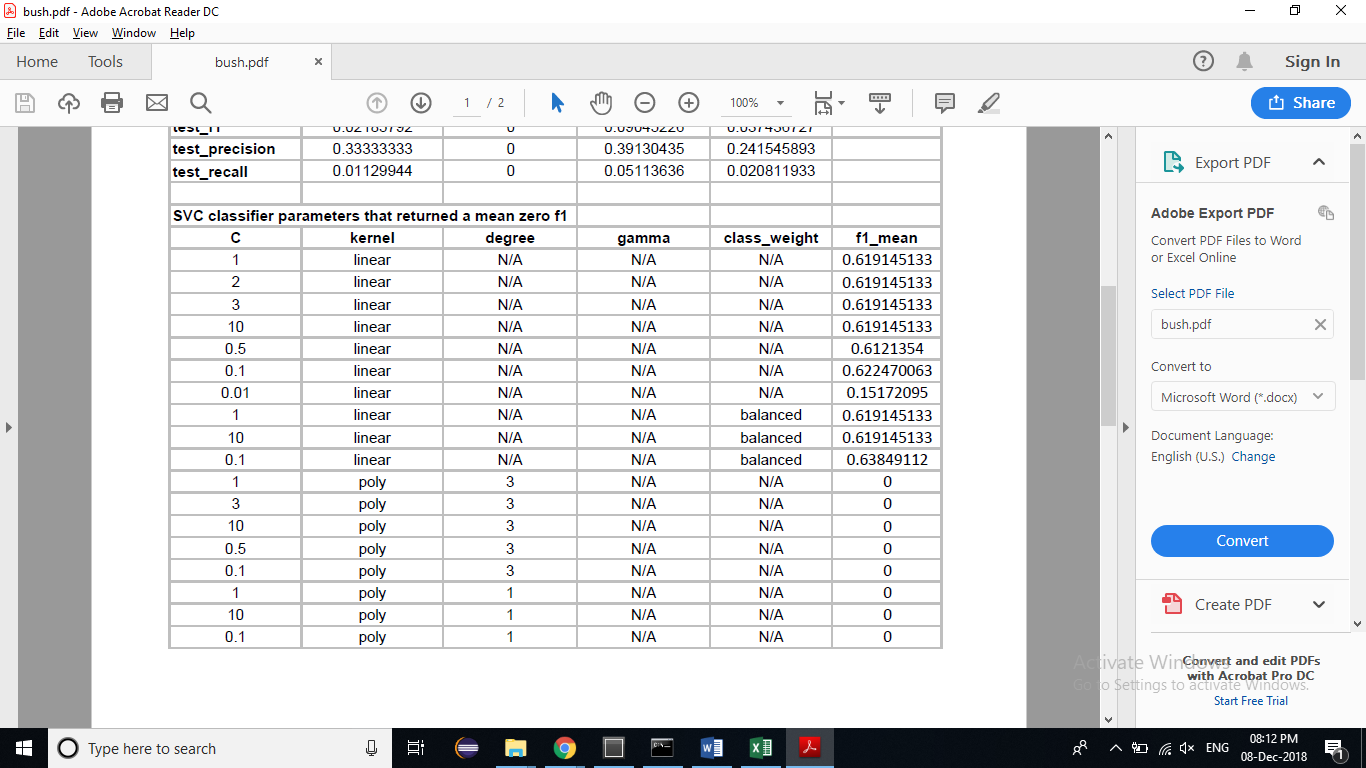
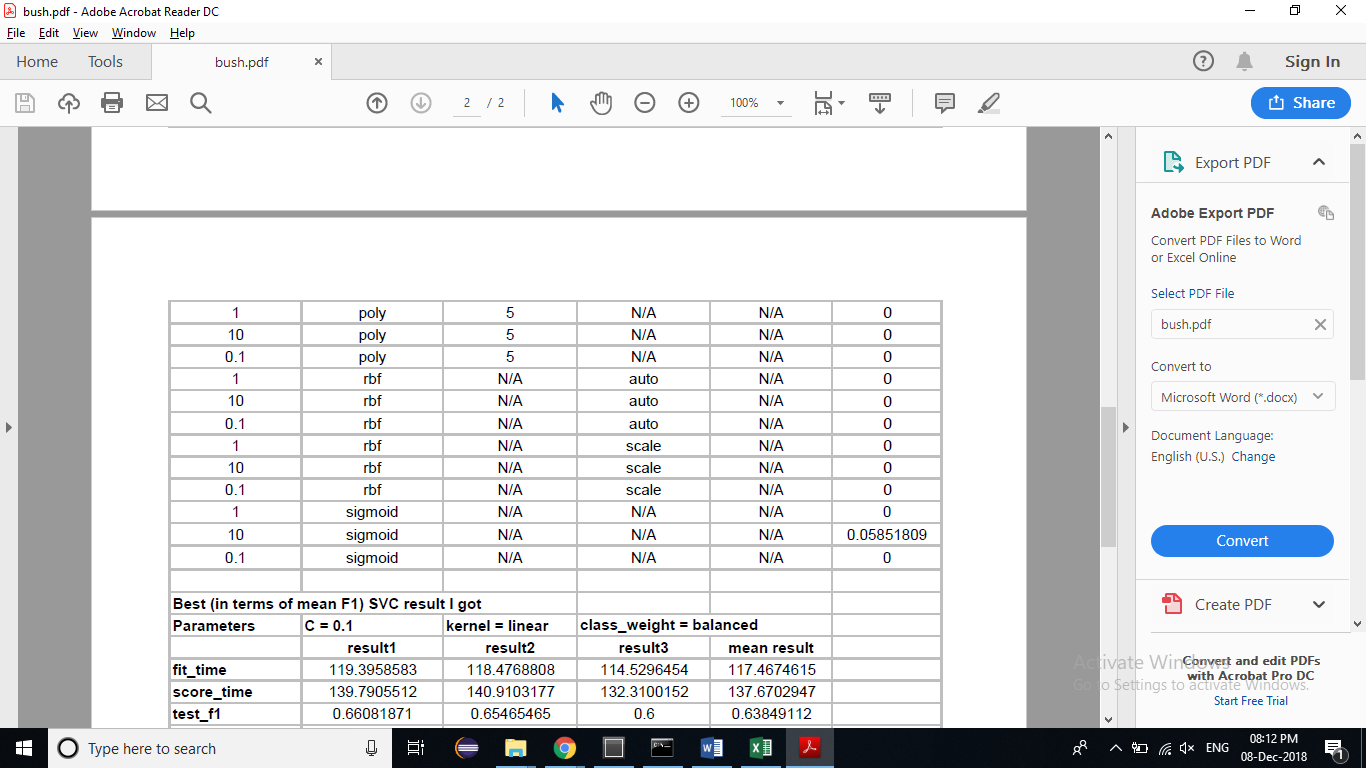
For Williams

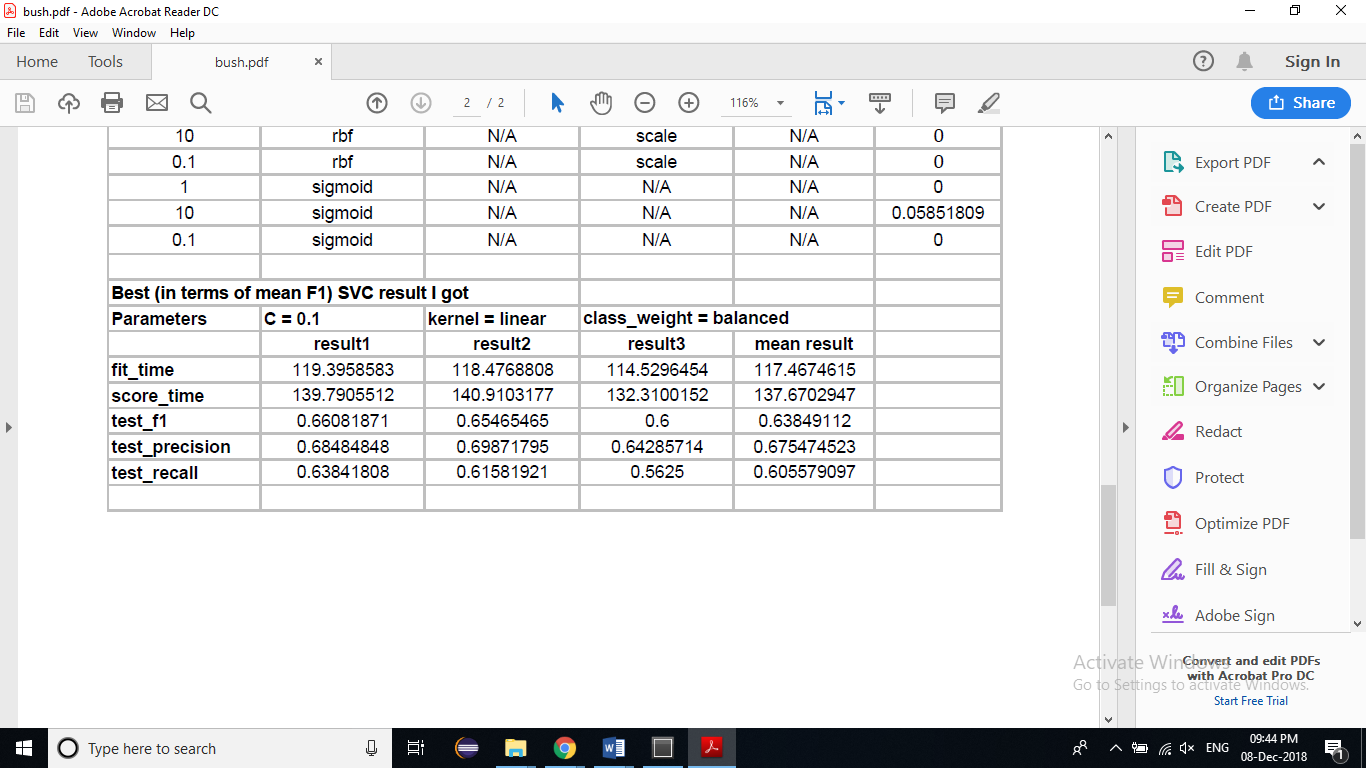


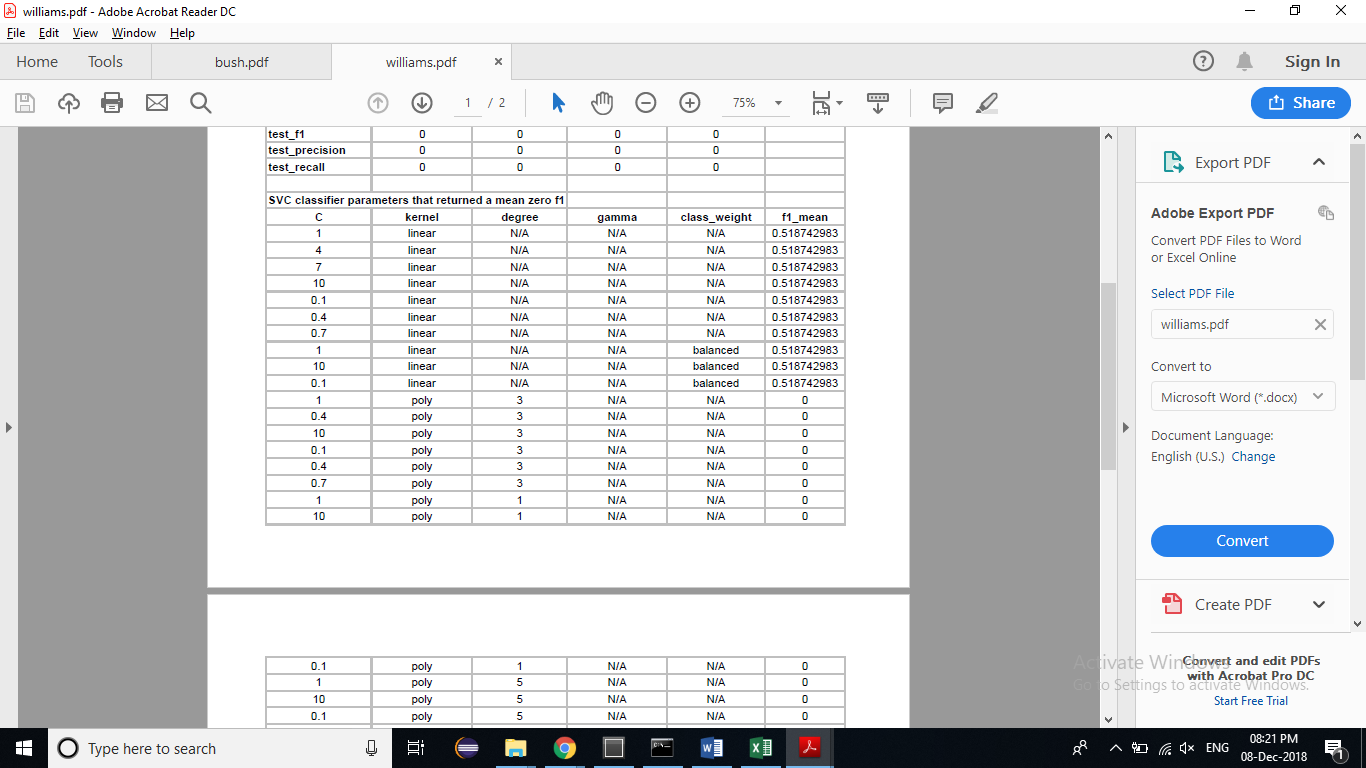
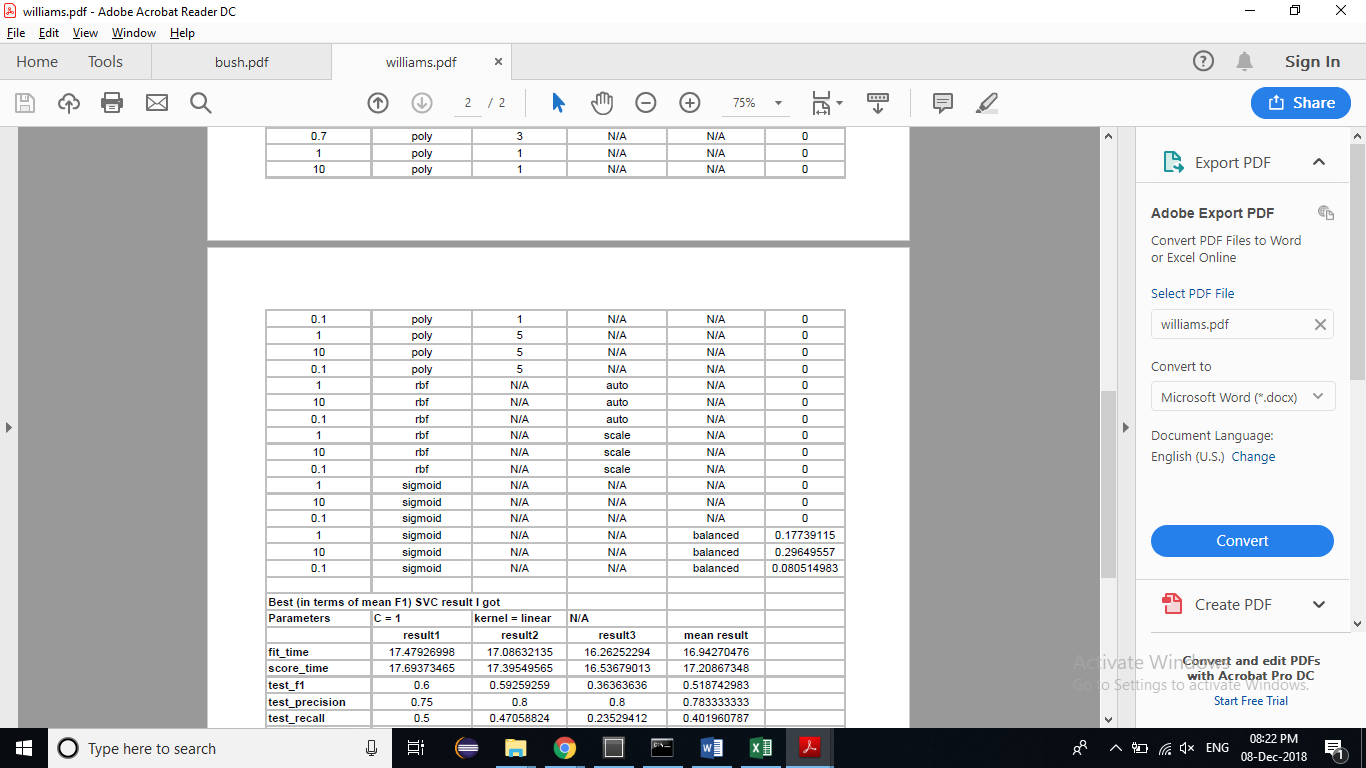
We also used another classifier named SVC (Support Vector Classifier). It is used to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our uses, though you can use this for many situations.

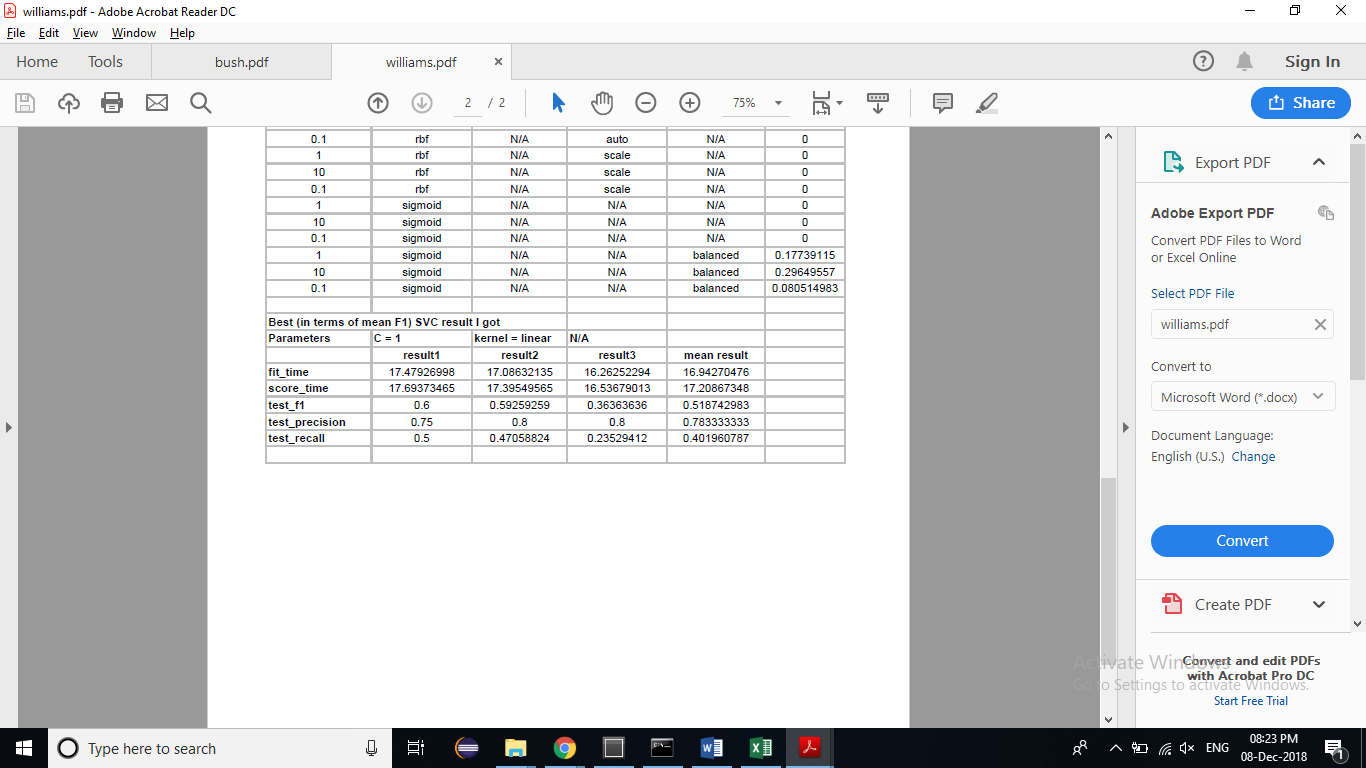
We computed the results by changing the value of C parameter which is a penalty parameter of the error term. We computed the result by changing the values of kernel parameter like linear, poly, rbf and sigmoid also changing the values of degree gave us different results.

Here is the list of combinations we tried to get the best fit suitable for

Bush



For Williams

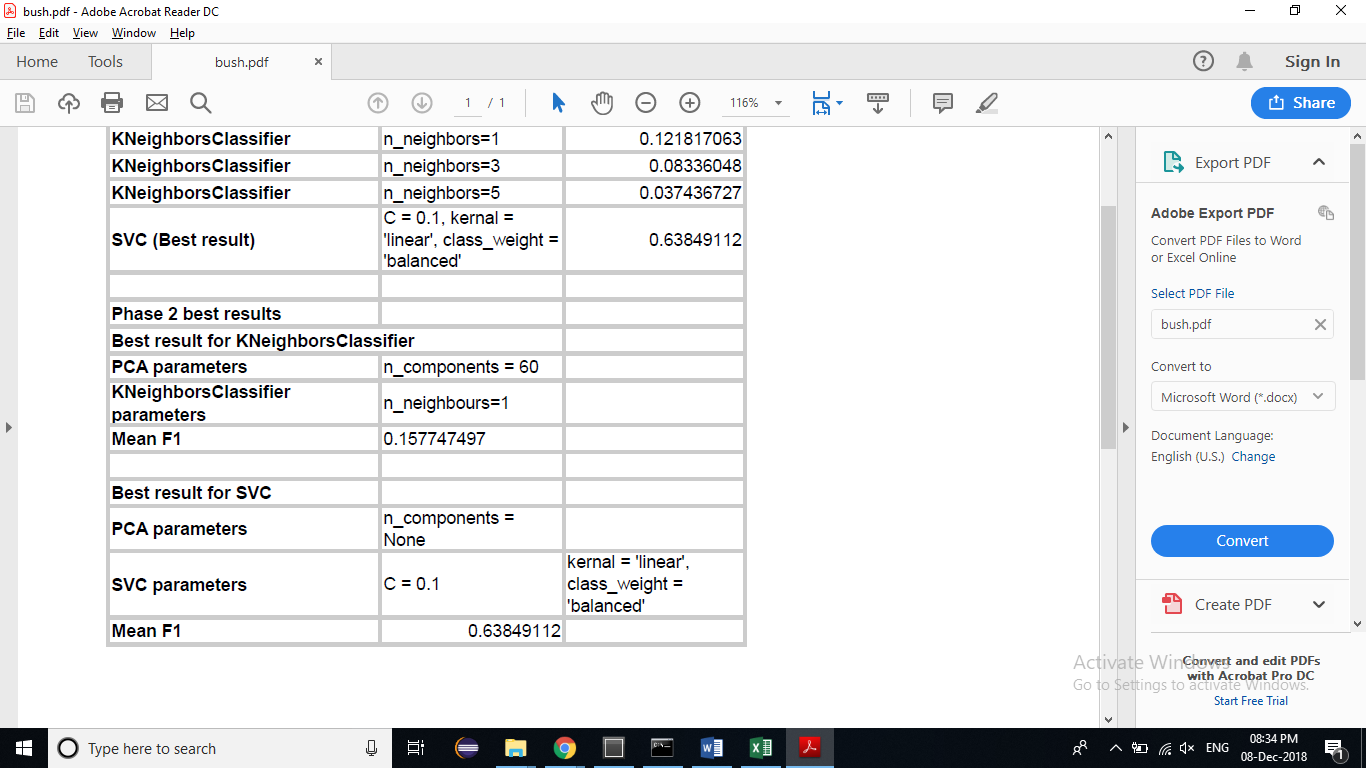


**Phase 2:**

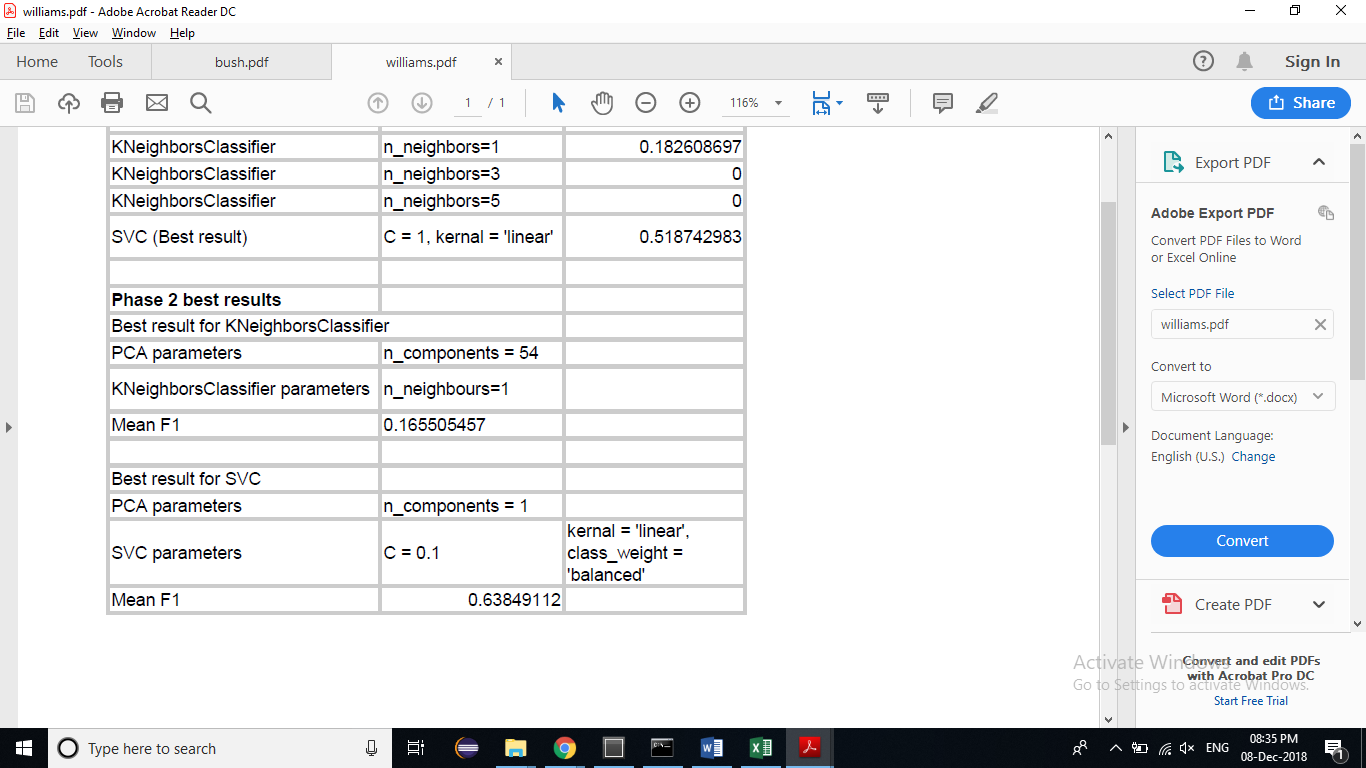
For this phase, we transformed the data using **Principal component analysis** (**PCA**) which is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called **principal components**. It reduces the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent.

By taking the transformed data we got the following results when we computed with both the classifiers, KNeighbours classifier and SVC.

We got the following results when we computed for bush



For Williams



**Phase 3:**

In this phase we emphasised on the use of deep learning, Convolutional Neural Network specifically. It is very similar to normal neural network as they are made of neurons that have weights and biases which can be learned. Each neuron gets some inputs and perform dot products and follows it optionally. A differentiable score function is observed by the whole network starting from from the raw image pixels on one end to class scores at the other.

The CNN model that we built is very simple since the number of examples with true labels is less, with 1 Convolutional Neural Network, 1 Pooling and 1 Dense layer.

**Bush model structure**

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 60, 60, 32) 832

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 30, 30, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 28800) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 28800) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 1) 28801

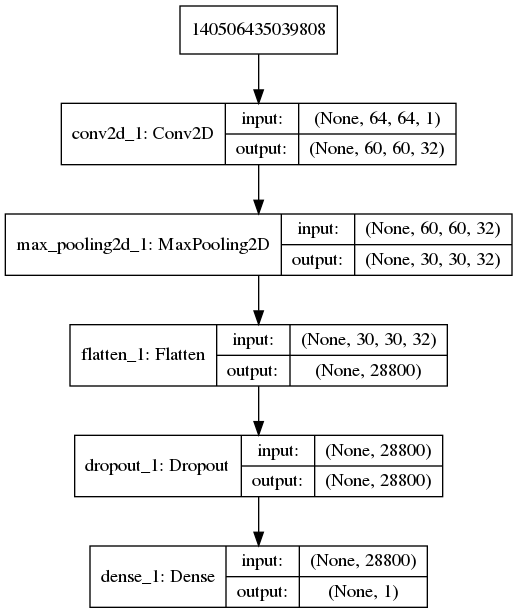
=================================================================

Total params: 29,633

Trainable params: 29,633

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



We tried different architectures and concluded that this added more complexity f1 score of 40~50%, but the current architecture gave me the best f1 score. The score came out as follow for bush

Train F1 Score: 0.961764705882353

Test F1 Score: 0.6643109540636042

**Williams model structure**

Layer (type) Output Shape Param #

=================================================================

conv2d\_1 (Conv2D) (None, 60, 60, 32) 832

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 30, 30, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 28800) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 1) 28801

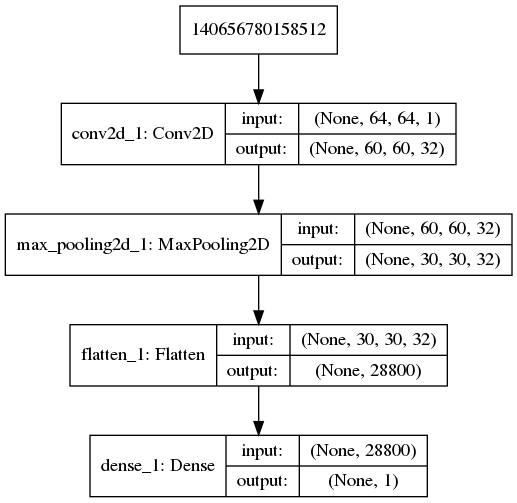
=================================================================

Total params: 29,633

Trainable params: 29,633

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Williams results were very challenging as it contained very few training data. I tried many combinations and kept my Network very minimal which resulted in the best f1 score as compared to the ones I got earlier. The current architecture is very simple. The results for Williams came as follow

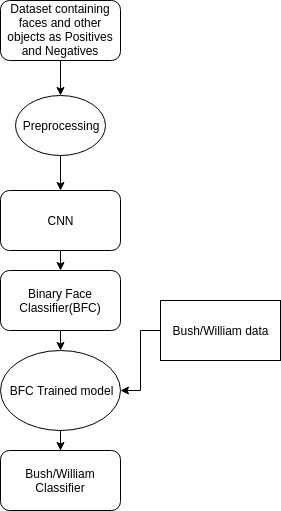
Train F1 Score: 0.955223880597015

Test F1 Score: 0.5

**Phase 4:**

In this phase, I made a Binary Face Classifier. I didn’t get a dataset for this, which should have faces as positive and other objects such as vehicles, animals etc. So, I made my own dataset using various resources.

For the negative/false labels I used different sources such as images of cars, train, animals and some other objects, which had totals images as ~2900 images for Negative.

The following figure shows the process, of how this phase was performed:

The optimizer used was **adadelta**, loss as ‘**binar\_crossentrophy**’ and epochs was 5.

The resultant model ‘**initialised\_model.model**’ was then used to train on the Bush/William data,

I used sgd, adam, adagrad and were terrible at both bush and Williams.

On training with Bush, following are the results obtained after changing hyperparameters with best ones:

For Bush

Train F1 Score: 0.9914529914529915

Test F1 Score: 0.7735849056603775

We trained the iniatialised\_model, on Bush examples for 20 epochs, which gave the best f1 score as compared to other parameters.

For Williams

Train F1 Score: 0.9855072463768115

Test F1 Score: 0.6896551724137931

We trained the iniatialised\_model, on Bush examples for 20 epochs.