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.

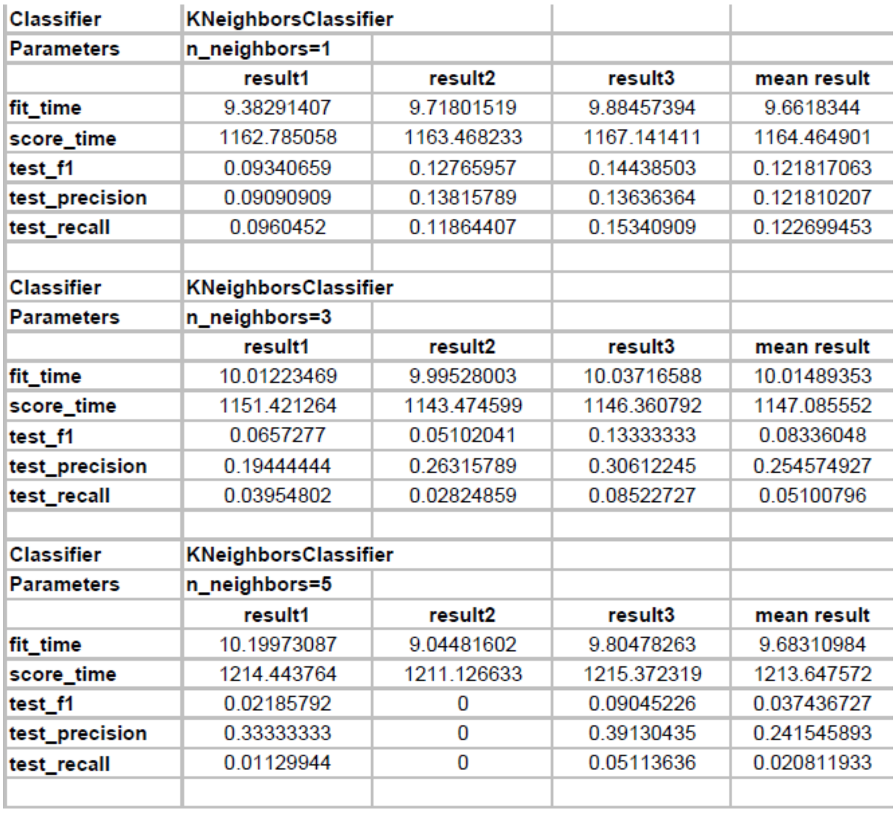
We split the data into training set and test set using three-fold cross validation which splits 2/3rd data into training set and rest in test set. The cross validation gives us the records of precision, recall, and test f1. When we set n_jobs parameter to -1, it helps us perform the experiment faster.

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.

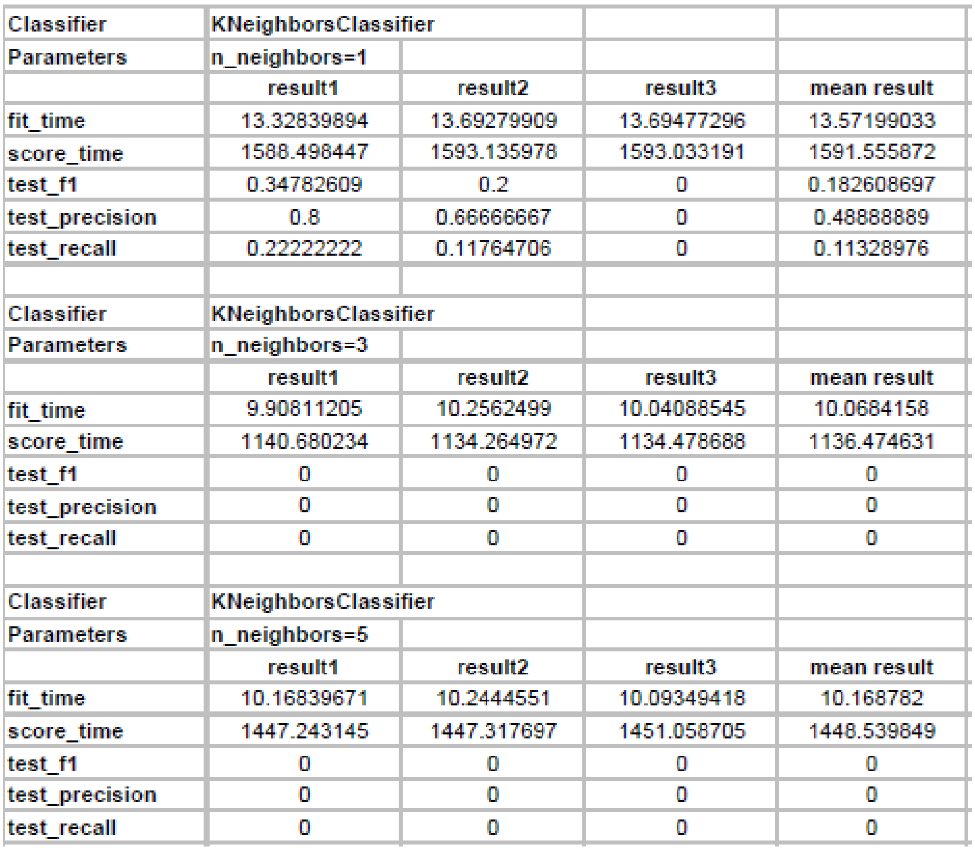
We have used 3 different values of the parameter n_neighbours, i.e. 1, 3 and 5 and computed the result. We observed that for the case of Bush, the mean F1 score was observed to be 0.1218 when we kept the value of n_neighbours = 1. It kept on decreasing as we increased the value of n_neighbours. The same 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.



Classifier	KNeighborsClassifier			
Parameters	n_neighbors=1			
	result1	result2	result3	mean result
fit_time	9.38291407	9.71801519	9.88457394	9.6618344
score_time	1162.785058	1163.468233	1167.141411	1164.464901
test_f1	0.09340659	0.12765957	0.14438503	0.121817063
test_precision	0.09090909	0.13815789	0.13636364	0.121810207
test_recall	0.0960452	0.11864407	0.15340909	0.122699453
Classifier	KNeighborsClassifier			
Parameters	n_neighbors=3			
	result1	result2	result3	mean result
fit_time	10.01223469	9.99528003	10.03716588	10.01489353
score_time	1151.421264	1143.474599	1146.360792	1147.085552
test_f1	0.0657277	0.05102041	0.13333333	0.08336048
test_precision	0.19444444	0.26315789	0.30612245	0.254574927
test_recall	0.03954802	0.02824859	0.08522727	0.05100796
Classifier	KNeighborsClassifier			
Parameters	n_neighbors=5			
	result1	result2	result3	mean result
fit_time	10.19973087	9.04481602	9.80478263	9.68310984
score_time	1214.443764	1211.126633	1215.372319	1213.647572
test_f1	0.02185792	0	0.09045226	0.037436727
test_precision	0.33333333	0	0.39130435	0.241545893
test_recall	0.01129944	0	0.05113636	0.020811933

For Williams



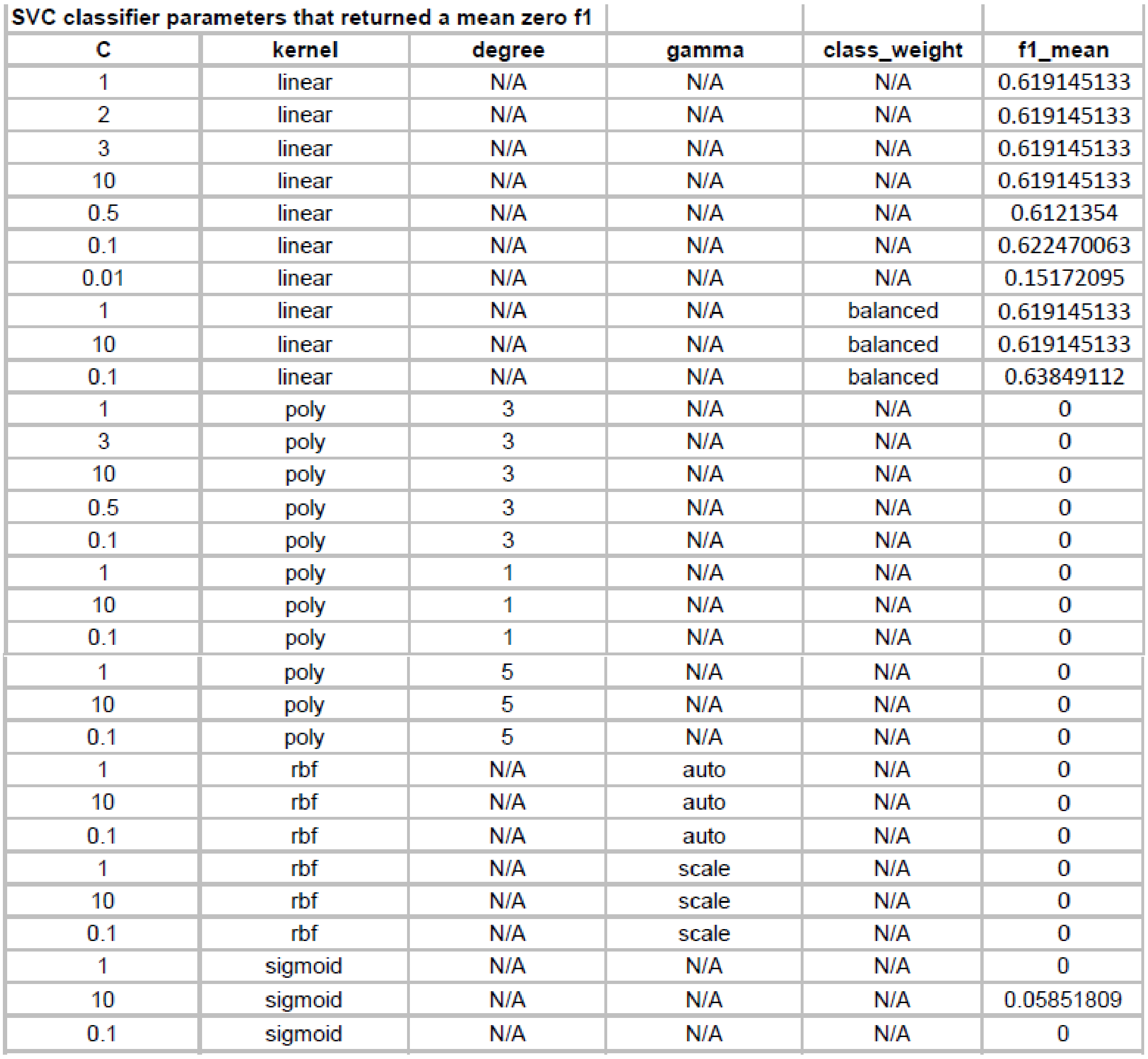
Classifier	KNeighborsClassifier			
Parameters	n_neighbors=1			
	result1	result2	result3	mean result
fit_time	13.32839894	13.69279909	13.69477296	13.57199033
score_time	1588.498447	1593.135978	1593.033191	1591.555872
test_f1	0.34782609	0.2	0	0.182608697
test_precision	0.8	0.66666667	0	0.48888889
test_recall	0.22222222	0.11764706	0	0.11328976
Classifier	KNeighborsClassifier			
Parameters	n_neighbors=3			
	result1	result2	result3	mean result
fit_time	9.90811205	10.2562499	10.04088545	10.0684158
score_time	1140.680234	1134.264972	1134.478688	1136.474631
test_f1	0	0	0	0
test_precision	0	0	0	0
test_recall	0	0	0	0
Classifier	KNeighborsClassifier			
Parameters	n_neighbors=5			
	result1	result2	result3	mean result
fit_time	10.16839671	10.2444551	10.09349418	10.168782
score_time	1447.243145	1447.317697	1451.058705	1448.539849
test_f1	0	0	0	0
test_precision	0	0	0	0
test_recall	0	0	0	0

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.

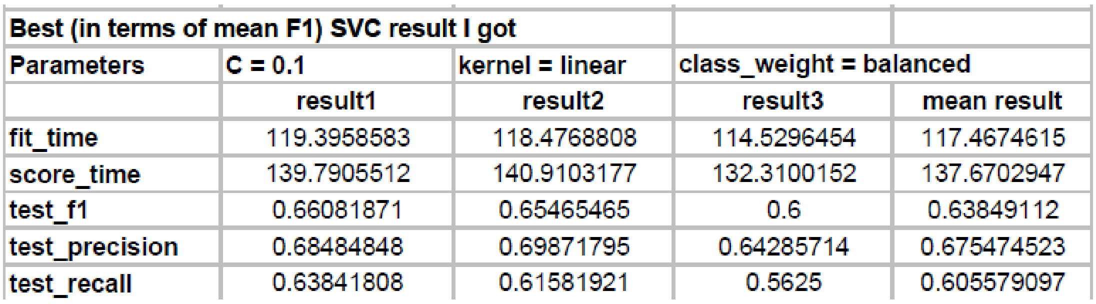
Here is the list of combinations we tried to get the best fit suitable for

Bush



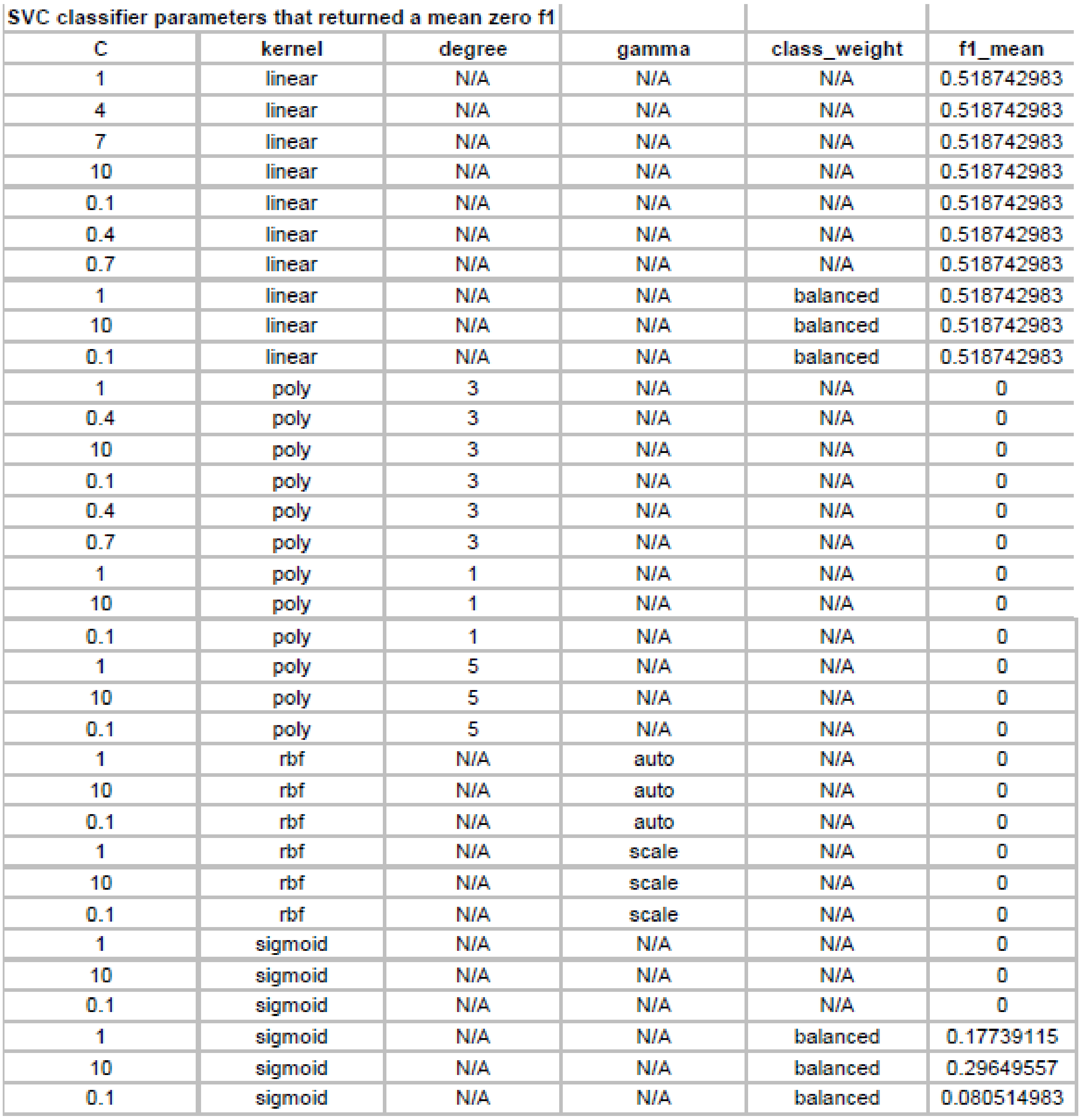
SVC classifier parameters that returned a mean zero f1

C	kernel	degree	gamma	class_weight	f1_mean
1	linear	N/A	N/A	N/A	0.619145133
2	linear	N/A	N/A	N/A	0.619145133
3	linear	N/A	N/A	N/A	0.619145133
10	linear	N/A	N/A	N/A	0.619145133
0.5	linear	N/A	N/A	N/A	0.6121354
0.1	linear	N/A	N/A	N/A	0.622470063
0.01	linear	N/A	N/A	N/A	0.15172095
1	linear	N/A	N/A	balanced	0.619145133
10	linear	N/A	N/A	balanced	0.619145133
0.1	linear	N/A	N/A	balanced	0.63849112
1	poly	3	N/A	N/A	0
3	poly	3	N/A	N/A	0
10	poly	3	N/A	N/A	0
0.5	poly	3	N/A	N/A	0
0.1	poly	3	N/A	N/A	0
1	poly	1	N/A	N/A	0
10	poly	1	N/A	N/A	0
0.1	poly	1	N/A	N/A	0
1	poly	5	N/A	N/A	0
10	poly	5	N/A	N/A	0
0.1	poly	5	N/A	N/A	0
1	rbf	N/A	auto	N/A	0
10	rbf	N/A	auto	N/A	0
0.1	rbf	N/A	auto	N/A	0
1	rbf	N/A	scale	N/A	0
10	rbf	N/A	scale	N/A	0
0.1	rbf	N/A	scale	N/A	0
1	sigmoid	N/A	N/A	N/A	0
10	sigmoid	N/A	N/A	N/A	0.05851809
0.1	sigmoid	N/A	N/A	N/A	0

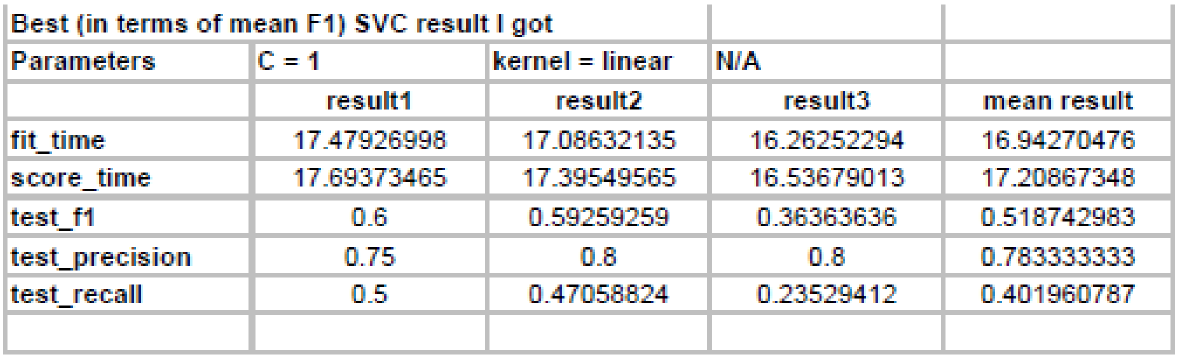


Best (in terms of mean F1) SVC result I got				
Parameters	C = 0.1	kernel = linear	class_weight = balanced	
	result1	result2	result3	mean result
fit_time	119.3958583	118.4768808	114.5296454	117.4674615
score_time	139.7905512	140.9103177	132.3100152	137.6702947
test_f1	0.66081871	0.65465465	0.6	0.63849112
test_precision	0.68484848	0.69871795	0.64285714	0.675474523
test_recall	0.63841808	0.61581921	0.5625	0.605579097

For Williams



SVC classifier parameters that returned a mean zero f1					
C	kernel	degree	gamma	class_weight	f1_mean
1	linear	N/A	N/A	N/A	0.518742983
4	linear	N/A	N/A	N/A	0.518742983
7	linear	N/A	N/A	N/A	0.518742983
10	linear	N/A	N/A	N/A	0.518742983
0.1	linear	N/A	N/A	N/A	0.518742983
0.4	linear	N/A	N/A	N/A	0.518742983
0.7	linear	N/A	N/A	N/A	0.518742983
1	linear	N/A	N/A	balanced	0.518742983
10	linear	N/A	N/A	balanced	0.518742983
0.1	linear	N/A	N/A	balanced	0.518742983
1	poly	3	N/A	N/A	0
0.4	poly	3	N/A	N/A	0
10	poly	3	N/A	N/A	0
0.1	poly	3	N/A	N/A	0
0.4	poly	3	N/A	N/A	0
0.7	poly	3	N/A	N/A	0
1	poly	1	N/A	N/A	0
10	poly	1	N/A	N/A	0
0.1	poly	1	N/A	N/A	0
1	poly	5	N/A	N/A	0
10	poly	5	N/A	N/A	0
0.1	poly	5	N/A	N/A	0
1	rbf	N/A	auto	N/A	0
10	rbf	N/A	auto	N/A	0
0.1	rbf	N/A	auto	N/A	0
1	rbf	N/A	scale	N/A	0
10	rbf	N/A	scale	N/A	0
0.1	rbf	N/A	scale	N/A	0
1	sigmoid	N/A	N/A	N/A	0
10	sigmoid	N/A	N/A	N/A	0
0.1	sigmoid	N/A	N/A	N/A	0
1	sigmoid	N/A	N/A	balanced	0.17739115
10	sigmoid	N/A	N/A	balanced	0.29649557
0.1	sigmoid	N/A	N/A	balanced	0.080514983



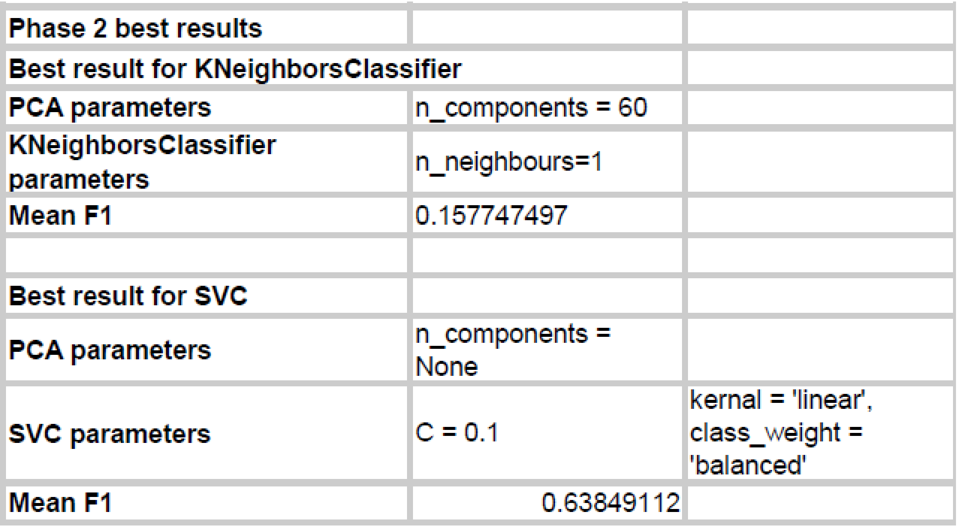
Best (in terms of mean F1) SVC result I got				
Parameters	C = 1	kernel = linear	N/A	
	result1	result2	result3	mean result
fit_time	17.47926998	17.08632135	16.26252294	16.94270476
score_time	17.69373465	17.39549565	16.53679013	17.20867348
test_f1	0.6	0.59259259	0.36363636	0.518742983
test_precision	0.75	0.8	0.8	0.783333333
test_recall	0.5	0.47058824	0.23529412	0.401960787

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.

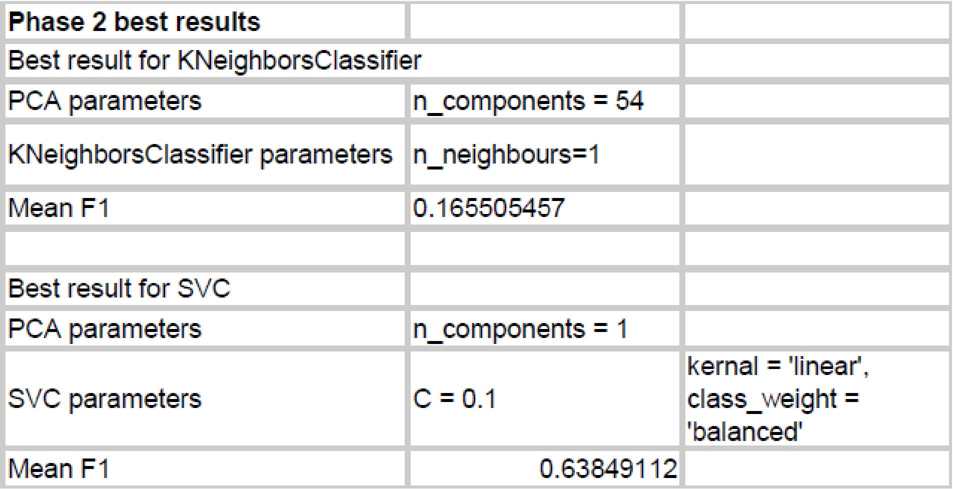
By taking the transformed data we got the following results when we computed with both the classifiers, KNeighbours classifier and SVC.

We got the following results when we computed for bush



Phase 2 best results		
Best result for KNeighborsClassifier		
PCA parameters	n_components = 60	
KNeighborsClassifier parameters	n_neighbours=1	
Mean F1	0.157747497	
Best result for SVC		
PCA parameters	n_components = None	
SVC parameters	C = 0.1	kernal = 'linear', class_weight = 'balanced'
Mean F1	0.63849112	

For Williams



Phase 2 best results		
Best result for KNeighborsClassifier		
PCA parameters	n_components = 54	
KNeighborsClassifier parameters	n_neighbours=1	
Mean F1	0.165505457	
Best result for SVC		
PCA parameters	n_components = 1	
SVC parameters	C = 0.1	kernal = 'linear', class_weight = 'balanced'
Mean F1	0.63849112	

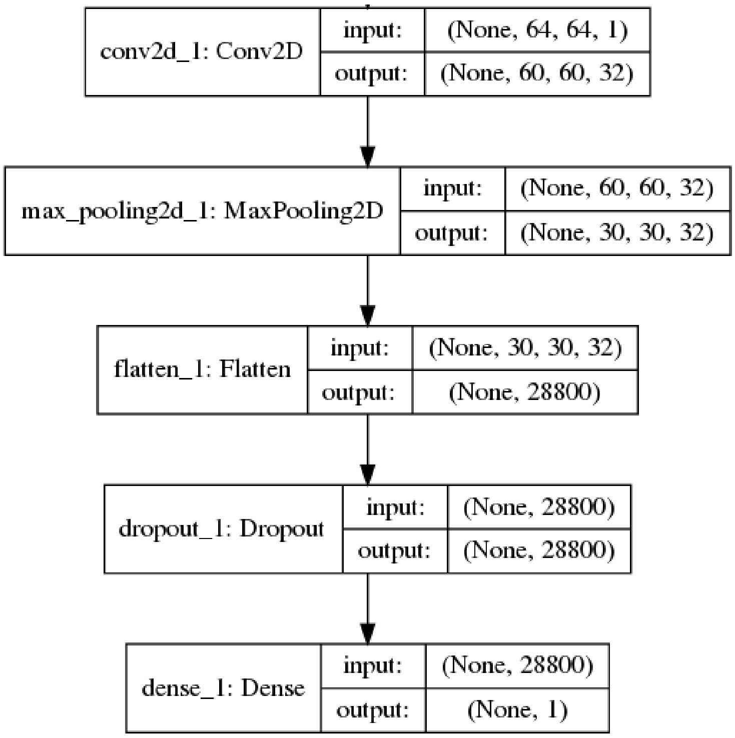
Phase 3:

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
_________________________________________________________________
```

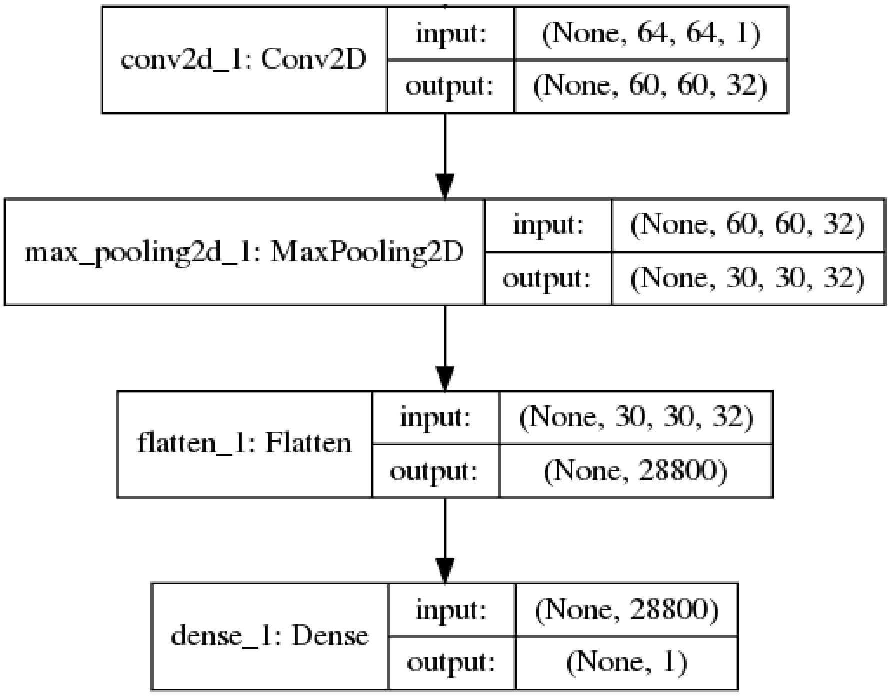


We tried different architectures and concluded that this added more complexity f1 score of 40~50%, but the current architecture gave me the best f1 score. The score came out as follow for bush

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

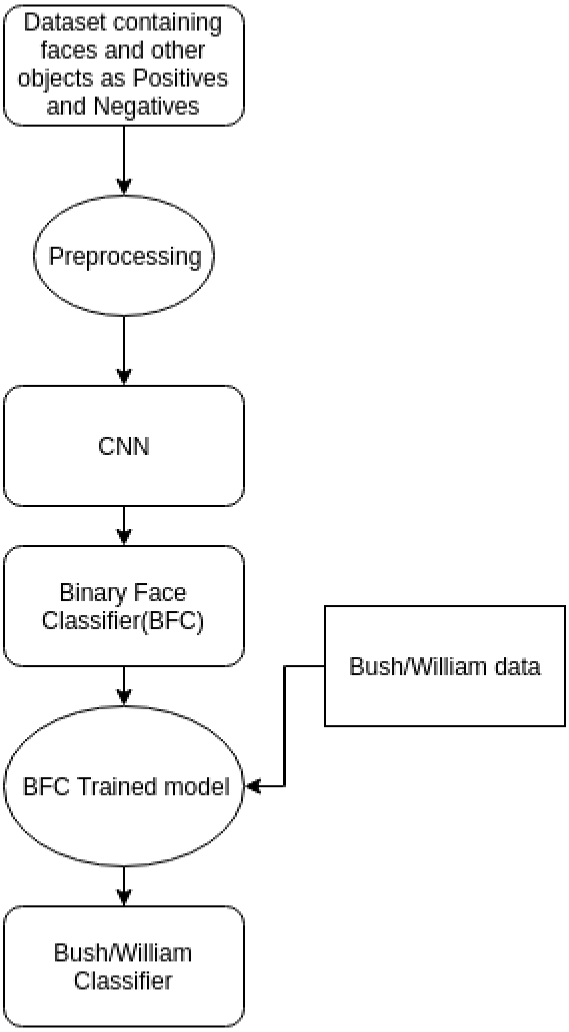
Train F1 Score: 0.955223880597015
Test F1 Score: 0.5

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:


Dataset containing faces and other objects as Positives and Negatives
Preprocessing
CNN
Binary Face Classifier(BFC)
Bush/William data
BFC Trained model
Bush/William Classifier

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.