ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

CS 584: Machine Learning

Project Report

Phase 4

Table of Contents

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Phase 1 - KNeighborsClassifier and SVC

The goal of this phase was to experiment with different parameters of KNeighborsClassifer and SVC and to find the best parameters in terms of better mean f1 score:

KNeighborsClassifer Steps (For both Bush and Williams):

1. Firstly, loaded the dataset given
2. Picked KNeighborsClassifer as the classifier (For 1,3 and 5 n_neighbors and set n_jobs = -1 when required)
3. Then performed three-fold cross validation using:
   stratified_cv_results = cross_validate(clf, X, y, cv=StratifiedKFold(n_splits = 3, shuffle=True, random_state = 6099), scoring=('precision', 'recall', 'f1'), return_train_score=False)
4. Recorded fit_time, score_time , test_f1 , test_precision , test_recall for 1 , 3, 5 neighbors
5. Stored mean f1 score for 1,3,5 neighbors to a list and pickled it

KNeighborsClassifer OUTPUTS:

The following are the results for KNeighborsClassifer with different neighbors for Bush and Williams:

BUSH - KNeighborsClassifer with 1 neighbor:

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=1**			
	result1	**result2**	**result3**	**mean result**
fit_time	9.39785838	9.36794519	9.08584237	9.28388198
score_time	376.3758667	386.180898	387.8513379	383.4693675
test_f1	0.14647887	0.13855422	0.11299435	0.132675813
test_precision	0.14606742	0.1483871	0.11235955	0.13560469
test_recall	0.14689266	0.1299435	0.11363636	0.130157507

BUSH - KNeighborsClassifer with 3 neighbors:

Classifier	**KNeighborsClassifier**			
Parameters	**n_neighbors=3**			
	result1	**result2**	**result3**	**mean result**
fit_time	8.37858987	11.26188612	13.15082645	10.93043415
score_time	399.0834658	398.775702	427.8648906	408.5746861
test_f1	0.10232558	0.04784689	0.09433962	0.08150403
test_precision	0.28947368	0.15625	0.27777778	0.241167153
test_recall	0.06214689	0.02824859	0.05681818	0.04907122

BUSH - KNeighborsClassifer with 5 neighbors:

Classifier	**KNeighborsClassifier**			

Parameters	n_neighbors=5			
	result1	result2	result3	mean result
fit_time	7.71137547	9.49564648	8.13723755	8.4480865
score_time	363.3123326	375.3063574	367.0173855	368.5453585
test_f1	0.0212766	0.03174603	0.06349206	0.03883823
test_precision	0.18181818	0.25	0.46153846	0.297785547
test_recall	0.01129944	0.01694915	0.03409091	0.020779833

WILLIAMS - KNeighborsClassifer with 1 neighbor:

Classifier	KNeighborsClassifier			
Parameters	n_neighbors=1			
	result1	result2	result3	mean result
fit_time	10.11867356	10.63974762	9.55846453	10.10562857
score_time	390.4468646	391.8257494	457.1624386	413.1450175
test_f1	0.24	0.0952381	0.3	0.21174603
test_precision	0.42857143	0.25	1	0.55952381
test_recall	0.16666667	0.05882353	0.17647059	0.13398693

WILLIAMS - KNeighborsClassifer with 3 neighbors:

Classifier	KNeighborsClassifier			
Parameters	n_neighbors=3			
	result1	result2	result3	mean result
fit_time	10.88711643	11.21104455	10.48717833	10.86177977
score_time	476.5946572	447.5135305	458.3112814	460.8064897
test_f1	0	0	0	0
test_precision	0	0	0	0
test_recall	0	0	0	0

WILLIAMS - KNeighborsClassifer with 5 neighbors:

Classifier	KNeighborsClassifier			
Parameters	n_neighbors=5			
	result1	result2	result3	mean result
fit_time	12.9892602	11.68674159	8.6438818	11.10662786
score_time	400.2301948	416.3879404	391.7707279	402.7962877
test_f1	0	0	0	0
test_precision	0	0	0	0
test_recall	0	0	0	0

SVC Steps (For both Bush and Williams):

1. Firstly, loaded the dataset given
2. Picked SVC as the classifier (set n_jobs = -1 when required)

3. Then performed three-fold cross validation using:
   stratified_cv_results = cross_validate(clf, X, y, cv=StratifiedKFold(n_splits = 3, shuffle=True, random_state = 6099), scoring=('precision', 'recall', 'f1'), return_train_score=False)
4. Recorded fit_time , score_time , test_f1 , test_precision , test_recall for best SVC classifier parameters that gave better f1 score
5. Stored mean f1 score for best SVC classifier parameters used to a list and pickled it

SVC OUTPUTS:

The following are the results for SVC with different neighbors for Bush and Williams:

BUSH – Best SVC results:

Best (in terms of mean F1) SVC result I got					
Parameters	C=4.0	kernel= poly	coef0= 5.0	degree = 10	gamma=1/(X.shape[1]*X.std())
	result1	result2	result3	mean result	
fit_time	151.0265942	155.3023119	148.9125176	151.7471412	
score_time	203.0234728	184.6456382	187.7555344	191.8082151	
test_f1	0.60666667	0.64705882	0.56428571	0.60600373	
test_precision	0.7398374	0.76744186	0.75961538	0.755631547	
test_recall	0.51412429	0.55932203	0.44886364	0.507436653	

BUSH –SVC classifier parameters that returned a mean zero f1:

SVC classifier parameters that returned a mean zero f1				
C	**kernel**	**degree**	**gamma**	**coef0**
1.0	poly	3	N/A	-
1.0	rbf	N/A	1/(X.shape[1]*X.std())	-
1.0	sigmoid	N/A	1/(X.shape[1]*X.std())	-
1.0	N/A	N/A	1/(X.shape[1]*X.std())	-
2.0	poly	2	auto_deprecated	-
2.0	rbf	N/A	auto	-
0.001	rbf	N/A	auto	-
0.001	sigmoid	N/A	auto	-
0.001	linear	N/A	auto	-
0.001	poly	N/A	auto	2.0

WILLIAMS – Best SVC results:

Best (in terms of mean F1) SVC result I got				

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Parameters	C=1.0	kernel=linear		
	result1	result2	result3	mean result
fit_time	20.27377844	18.7149477	20.57497048	19.85456554
score_time	20.9429853	20.2867372	20.48222017	20.57064756
test_f1	0.48	0.5	0.58064516	0.520215053
test_precision	0.85714286	0.85714286	0.64285714	0.785714287
test_recall	0.33333333	0.35294118	0.52941176	0.405228757

WILLIAMS –SVC classifier parameters that returned a mean zero f1:

SVC classifier parameters that returned a mean zero f1			
C	kernel	degree	gamma
1	poly	3	N/A
1.0	rbf	N/A	1/(X.shape[1]*X.std())
1.0	N/A	N/A	1/(X.shape[1]*X.std())
9.0	sigmoid	N/A	auto
1.0	sigmoid	N/A	auto
24.0	sigmoid	N/A	1/(X.shape[1]*X.std())
2.0	poly	2	auto_deprecated
2.0	rbf	N/A	auto
0.001	rbf	N/A	auto
0.001	sigmoid	N/A	auto
0.001	linear	N/A	auto

Result Discussion Phase – 1:

For KNeighborsClassifer, the best mean f1 score came for 1 neighbor for both Bush and Williams which was 0.1326 and 0.2117 respectively. 3 and 5 neighbors performed bad in terms of mean f1 score for Bush and Williams.

For SVC classifier, the best mean f1 score came was 0.6060 for bush with parameters as C = 4.0, kernel = poly , coef0 =5.0 , degree = 10, gamma = 1/(X.shape[1]*X.std() and may other parameter combinations I tried gave 0 f1 score which are listed in the table above. Hence, poly kernel performed better for me than other kernels. For Williams, 0.5202 was the best I for as mean f1 score and with C = 1.0 and kernel = linear. Hence, for Williams the better kernel was linear.

But the mean f1 score increased for both Williams and Bush when I switched from KNeighborsClassifer to SVC. Hence, for the given dataset for this project SVC performed well than KNeighborsClassifer.

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Phase 2 – PCA Dimensionality reduction

The goal of this phase was to perform the dimensionality reduction on X.csv file using fit and transform and then use the KNeighborsClassifer and SVC classifier followed by three-fold cross validation and then getting the f1 score and observe the Phase 1 and this Phase results in terms of whether the mean f1 score increases or not after reducing the dimensions.

KNeighborsClassifer Steps (For both Bush and Williams):

1. Loaded the dataset
2. Used PCA for dimensionality (fit_transform) reduction and played with n_components parameter and other parameters as well
3. Picked KNeighborsClassifier as classifier and found the best parameters that can give better mean f1 score
4. Performed three-fold cross validation using the following code: stratified_cv_results = cross_validate(clf, X, y, cv=StratifiedKFold(n_splits = 3, shuffle=True, random_state = 6099), scoring=('precision', 'recall', 'f1'), return_train_score=False) where X is the PCA transformed data
5. Stored the best mean f1 score in a list and then pickled it for bush and Williams in bush.pickle and Williams.pickle repectively

KNeighborsClassifer OUTPUTS:

The following are the results for KNeighborsClassifer using the best parameters that are mentioned in table for Bush and Williams:

BUSH – KNeighborsClassifer best result:

Best result for KNeighborsClassifier	
PCA parameters	n_components= 55 , copy=False, whiten=True, svd_solver='auto', iterated_power='auto'
KNeighborsClassifier parameters	n_neighbors = 1, n_jobs = -1
Mean F1	0.151631059162092

WILLAIMS – KNeighborsClassifer best result:

Best result for KNeighborsClassifier	
PCA parameters	n_components= 55 , copy=False, whiten=True, svd_solver='auto', iterated_power='auto'
KNeighborsClassifier parameters	n_neighbors = 1, n_jobs = -1
Mean F1	0.225845410628019

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Result Discussion – KneighborsClassifer - Phase 2:

The following are the mean f1 score obtained by using KNeighborsClassifer as the classifier for Bush and Williams:

	Before PCA (Phase 1)	**After PCA (Phase 2)**
Bush	0.13267	0.15163
Williams	0.21174	0.22584

So, I observe that there is increase in mean f1 score after using the PCA i.e. after performing the dimensionality reduction. For Bush it was 0.13 before PCA and after it became 0.15 which is increase of approx. 14.29%. On the other hand, for Williams it was 0.21 before PCA and after it became 0.22 which is increase of approx. 6.66%. Hence, it concludes that for this dataset the mean f1 score increased after the dataset went through the dimensionality reduction using PCA and KNeighborsClassifer classifier.

SVC Steps (For both Bush and Williams):

1. Loaded the dataset
2. Used PCA for dimensionality (fit_transform) reduction and played with n_components parameter and other parameters as well
3. Picked SVC as classifier and found the best classifier using various different parameters
4. Performed three-fold cross validation using the following code: stratified_cv_results = cross_validate(clf, X, y, cv=StratifiedKFold(n_splits = 3, shuffle=True, random_state = 6099), scoring=('precision', 'recall', 'f1'), return_train_score=False) where X is the PCA transformed data
5. Stored the best mean f1 score in a list and then pickled it for bush and Williams in bush.pickle and Williams.pickle respectively

SVC OUTPUTS:

The following are the results for SVC using the best parameters that are mentioned in table for Bush and Williams:

BUSH –SVC best result:

Best result for SVC	
PCA parameters	n_components= 3000
SVC parameters	C = 4.0 , kernel = 'poly', coef0= 5.0 ,degree = 10 , gamma=1/(X.shape[1]*X.std())
Mean F1	0.609185913807599

WILLAIMS –SVC best result:

Best result for SVC	
PCA parameters	n_components= 0.95
SVC parameters	C = 4.0 , kernel = 'poly', coef0= 5.0 ,degree = 10 , gamma=1/(X.shape[1]*X.std())
Mean F1	0.558943907675791

Result Discussion – SVC – Phase 2:

The following are the mean f1 score obtained by using SVC as the classifier for Bush and Williams:

	Before PCA (Phase 1)	**After PCA (Phase 2)**
Bush	0.60600	0. 60918
Williams	0.52021	0.55894

So, I observe again that there is increase in mean f1 score after using the PCA i.e. after performing the dimensionality reduction. For Bush it was 0.606 before PCA and after it became 0.609 which is increase of approx. 0.52%. On the other hand, for Williams it was 0.520 before PCA and after it became 0.558 which is increase of approx. 7.45%. Hence, it concludes that for this dataset the mean f1 score increased after the dataset went through the dimensionality reduction using PCA and SVC classifier.

Phase 3 – Develop Deep Learning Model

The goal was to develop the deep learning model for both Bush and Williams. The built model uses multiple layers of CNNs (2D CNN) and maxpooling and a dense layer at the end followed by output layer with single node and sigmoid as activation function. The final f1 score was to be reported on both train and test.

BUSH MODEL:

The following is the summary of the model for Bush that I used, to get f1 score as 0.995 on Train and 0.848 on Test.

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Model Summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2305
=================================================================
Total params: 30,369
Trainable params: 30,369
Non-trainable params: 0
_________________________________________________________________
None
```

Model Layers Details:

```
[<keras.layers.convolutional.Conv2D at 0x7f8290835710>,
 <keras.layers.pooling.MaxPooling2D at 0x7f824ee4d588>,
 <keras.layers.convolutional.Conv2D at 0x7f824ee4d2e8>,
 <keras.layers.pooling.MaxPooling2D at 0x7f824e5e3f98>,
 <keras.layers.convolutional.Conv2D at 0x7f824ee4dda0>,
 <keras.layers.pooling.MaxPooling2D at 0x7f824e5e3d30>,
 <keras.layers.core.Flatten at 0x7f824e604278>,
 <keras.layers.core.Dense at 0x7f824e5987b8>]
```

Model Inputs and Model Outputs:

```
[<tf.Tensor 'conv2d_1_input:0' shape=(?, 64, 64, 1) dtype=float32>]

[<tf.Tensor 'dense_1/Sigmoid:0' shape=(?, 1) dtype=float32>]
```

Model Description:

Talking about the model for Bush it has the following type of the layers and the dimensions:

1. Input is the reshaped X file whose dimension after reshaping are 64*64*1
2. Conv2D - with activation function as relu , input_shape = (64, 64, 1), filter = 32 and kernel_size = (3,3) and this layer has output dimensions of 62*62*32

3. MaxPooling2 - that has pool_size of 2,2 and output dimension of 31*31*32 (The first two dimension of output shape will reduce to half of the one in previous step as we are using maxpooling and this applies to all the below steps where maxpooling is used)
4. Conv2D with activation function as relu, filter = 32 and kernel_size = (3,3) and this layer has output dimensions of 29*29*32
5. MaxPooling2 - that has pool_size of 2,2 and output dimension of 14*14*32
6. Conv2D with filter = 64 and kernel_size = (3,3) has output dimensions of f 12*12*64
7. MaxPooling2 - that has pool_size of 2,2 and output dimension of 6*6*64
8. Flatten, dimension of 2304
9. Then got Output layer by using sigmoid activation function and 1 as Dense and this has dimension of 1

Also, while compiling the model used optimizer='rmsprop',loss='binary_crossentropy' and while fitting the model used epochs=10, batch_size=32

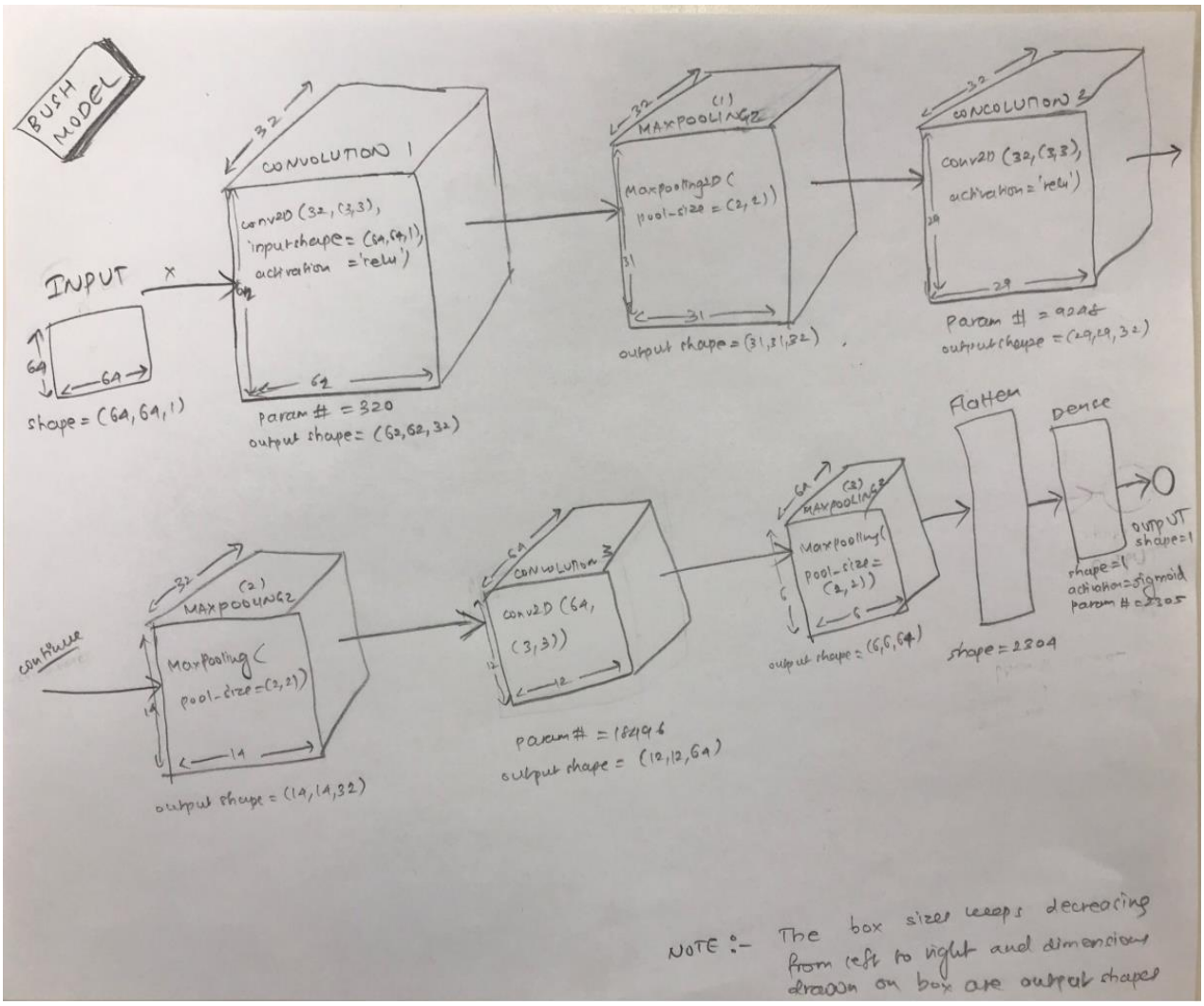
OUTPUT for my model for Bush:

```
Train on 8822 samples, validate on 4411 samples
Epoch 1/10
8822/8822 [==============================] - 8s 908us/step - loss: 0.1708 - acc: 0.9599 - val_loss: 0.2514 - val_acc: 0.9599
Epoch 2/10
8822/8822 [==============================] - 4s 459us/step - loss: 0.1236 - acc: 0.9619 - val_loss: 0.0960 - val_acc: 0.9726
Epoch 3/10
8822/8822 [==============================] - 4s 461us/step - loss: 0.0763 - acc: 0.9729 - val_loss: 0.0759 - val_acc: 0.9723
Epoch 4/10
8822/8822 [==============================] - 4s 461us/step - loss: 0.0546 - acc: 0.9812 - val_loss: 0.0538 - val_acc: 0.9825
Epoch 5/10
8822/8822 [==============================] - 4s 464us/step - loss: 0.0391 - acc: 0.9867 - val_loss: 0.0475 - val_acc: 0.9830
Epoch 6/10
8822/8822 [==============================] - 4s 463us/step - loss: 0.0275 - acc: 0.9899 - val_loss: 0.0372 - val_acc: 0.9880
Epoch 7/10
8822/8822 [==============================] - 4s 456us/step - loss: 0.0196 - acc: 0.9932 - val_loss: 0.0374 - val_acc: 0.9880
Epoch 8/10
8822/8822 [==============================] - 4s 452us/step - loss: 0.0145 - acc: 0.9959 - val_loss: 0.0326 - val_acc: 0.9893
Epoch 9/10
8822/8822 [==============================] - 4s 453us/step - loss: 0.0086 - acc: 0.9980 - val_loss: 0.0378 - val_acc: 0.9891
Epoch 10/10
8822/8822 [==============================] - 4s 450us/step - loss: 0.0053 - acc: 0.9983 - val_loss: 0.0411 - val_acc: 0.9875
```

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Model Diagram:



ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

WILLIAMS MODEL:

The following is the summary of the model for Bush that I used to get f1 score 1.0 on Train and 0.709 on Test.

Model Summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 62, 62, 32)        320
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 2305
=================================================================
Total params: 30,369
Trainable params: 30,369
Non-trainable params: 0
_________________________________________________________________
```

Model Layers Details:

```
[<keras.layers.convolutional.Conv2D at 0x7f8243f2be48>,
 <keras.layers.pooling.MaxPooling2D at 0x7f8240ba6898>,
 <keras.layers.convolutional.Conv2D at 0x7f8242150828>,
 <keras.layers.pooling.MaxPooling2D at 0x7f8240bbbcc0>,
 <keras.layers.convolutional.Conv2D at 0x7f8240bbb6a0>,
 <keras.layers.pooling.MaxPooling2D at 0x7f8240bbbbe0>,
 <keras.layers.core.Flatten at 0x7f8240bc52e8>,
 <keras.layers.core.Dense at 0x7f8240b5e780>]
```

Model Inputs and Model Outputs:

```
[<tf.Tensor 'conv2d_10_input:0' shape=(?, 64, 64, 1) dtype=float32>]
[<tf.Tensor 'dense_4/Sigmoid:0' shape=(?, 1) dtype=float32>]
```

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Model Description:

Talking about the model for Bush it has the following type of the layers and the dimensions:

1. Input is the reshaped X file whose dimension after reshaping are 64*64*1
2. Conv2D - with activation function as relu , input_shape = (64, 64, 1), filter = 32 and kernel_size = (3,3) and this layer has output dimensions of 62*62*32
10. MaxPooling2 - that has pool_size of 2,2 and output dimension of 31*31*32 (The first two dimension of output shape will reduce to half of the one in previous step as we are using maxpooling and this applies to all the below steps where maxpooling is used)
3. Conv2D with activation function as relu, filter = 32 and kernel_size = (3,3) and this layer has output dimensions of 29*29*32
4. MaxPooling2 - that has pool_size of 2,2 and output dimension of 14*14*32
5. Conv2D with filter = 64 and kernel_size = (3,3) has output dimensions of f 12*12*64
6. MaxPooling2 - that has pool_size of 2,2 and output dimension of 6*6*64
7. Flatten, dimension of 2304
8. Then got Output layer by using sigmoid activation function and 1 as Dense and this has dimension of 1

Also, while compiling the model used optimizer='rmsprop', loss='binary_crossentropy' and while fitting the model used epochs=15, batch_size=32

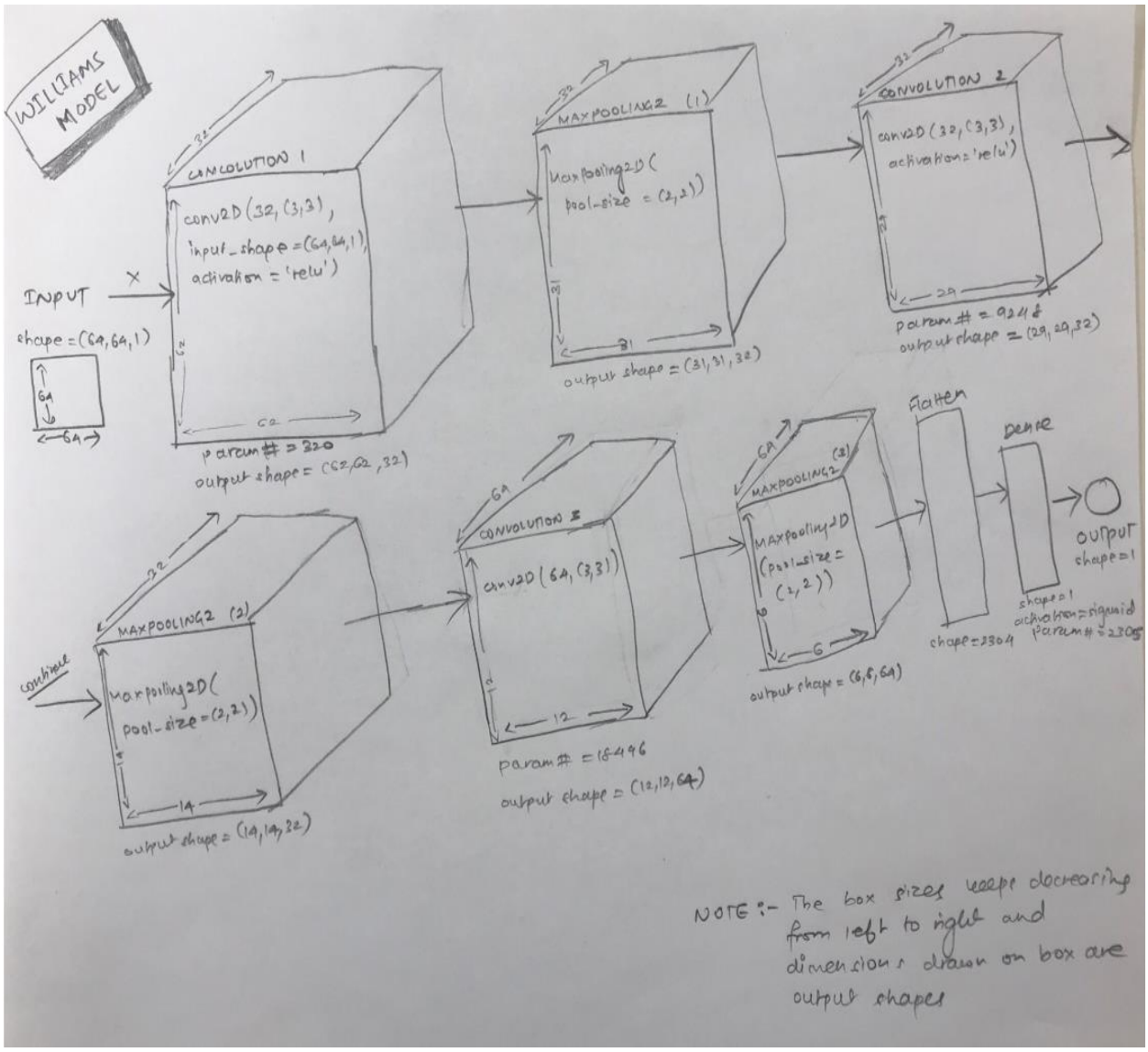
OUTPUT for my model for Williams:

```
Train on 8822 samples, validate on 4411 samples
Epoch 1/15
8822/8822 [==============================] - 6s 681us/step - loss: 0.0322 - acc: 0.9924 - val_loss: 0.0266 - val_acc: 0.9961
Epoch 2/15
8822/8822 [==============================] - 4s 485us/step - loss: 0.0209 - acc: 0.9959 - val_loss: 0.0153 - val_acc: 0.9961
Epoch 3/15
8822/8822 [==============================] - 4s 481us/step - loss: 0.0181 - acc: 0.9961 - val_loss: 0.0130 - val_acc: 0.9964
Epoch 4/15
8822/8822 [==============================] - 4s 484us/step - loss: 0.0147 - acc: 0.9965 - val_loss: 0.0130 - val_acc: 0.9961
Epoch 5/15
8822/8822 [==============================] - 4s 484us/step - loss: 0.0112 - acc: 0.9965 - val_loss: 0.0080 - val_acc: 0.9977
Epoch 6/15
8822/8822 [==============================] - 4s 480us/step - loss: 0.0087 - acc: 0.9977 - val_loss: 0.0132 - val_acc: 0.9964
Epoch 7/15
8822/8822 [==============================] - 4s 478us/step - loss: 0.0065 - acc: 0.9980 - val_loss: 0.0147 - val_acc: 0.9957
Epoch 8/15
8822/8822 [==============================] - 4s 483us/step - loss: 0.0047 - acc: 0.9989 - val_loss: 0.0177 - val_acc: 0.9975
Epoch 9/15
8822/8822 [==============================] - 4s 482us/step - loss: 0.0041 - acc: 0.9991 - val_loss: 0.0142 - val_acc: 0.9980
Epoch 10/15
8822/8822 [==============================] - 4s 481us/step - loss: 0.0029 - acc: 0.9993 - val_loss: 0.0088 - val_acc: 0.9980
Epoch 11/15
8822/8822 [==============================] - 4s 474us/step - loss: 0.0016 - acc: 0.9997 - val_loss: 0.0098 - val_acc: 0.9980
Epoch 12/15
8822/8822 [==============================] - 4s 474us/step - loss: 0.0012 - acc: 0.9997 - val_loss: 0.0202 - val_acc: 0.9977
Epoch 13/15
8822/8822 [==============================] - 4s 475us/step - loss: 9.2650e-04 - acc: 0.9997 - val_loss: 0.0111 - val_acc: 0.9966
Epoch 14/15
8822/8822 [==============================] - 4s 474us/step - loss: 0.0012 - acc: 0.9998 - val_loss: 0.0128 - val_acc: 0.9982
Epoch 15/15
8822/8822 [==============================] - 4s 478us/step - loss: 7.9282e-04 - acc: 0.9999 - val_loss: 0.0125 - val_acc: 0.9980
```

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Model Diagram:



ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Result Discussion – Phase 3:

The following are the mean f1 score obtained by building a deep learning model for Bush and Williams:

	Train	Test
Bush	0.995	0. 848
Williams	1.0	0.709

So, I observe that the deep learning model that I build for Bush is better on Train and on test it is0.995 and 0.848 respectively which are also a good f1 score and for Williams the f1 score is 1.0 on train and on test 0.709 on the model that I made for Williams. Bush has better f1 on test because may be Bush has more images than Williams and hence the model learns better if there are more images of the person.

Also, the F1 score increased for both Bush and Williams than the one is Phase 2 which was 0.609 for Bush and 0.5589 on Williams (using SVC). Now it is 0.848 for Bush and 0.709 for Williams on test.

There is increase of 39.24 % of the f1 score for Bush and increase of 26.86% for Williams from previous Phase and hence the deep learning mode that I made was good and able make the predictions better than the SVC classifier.

Phase 4 - Transfer Learning

The goal of this Phase is to find face or image classification dataset. Then process the data so that the images are 64x64 and gray-scale, and process the labels so that it is binary. Pre-train the model on processed image dataset and save the model. Then load the saved model and train it on Bush and Williams train split. Obtain the f1 score on train and test on both Bush and Williams.

Other Dataset Used – Dogs vs Cats:

Dataset used: Dogs vs Cats
Referred link: https://www.kaggle.com/c/dogs-vs-cats/data
About Dataset: The training archive contains 24,500 images of dogs and cats and 500 images are used for testing
 # Classes: 2

Prepressing of Dogs vs Cats dataset:

1. Downloaded the dataset from https://www.kaggle.com/c/dogs-vs-cats/data website
2. Uploaded the zipped folder of the dataset on google colab
3. Unzipped the folder using Python
4. Took the train set of the dataset which were labelled and set 0 for cat and 1 for Dog as follows:

```
'''Labelling the dataset'''
def label_img(img):
    word_label = img.split('.')[-3]
    # DIY One hot encoder
    if word_label == 'cat': return 0
    elif word_label == 'dog': return 1
```

5. Converted image to gray scale using : img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
6. Reshaped the image to 64*64 using : img = cv2.resize(img, (64, 64))
7. Stored the images and the labels in a list and that result I saved it in train_data.npy
8. Loaded train_data.npy
9. Split this file into train and test (Train has 24500 images and test has 500 images, Note: I did not use the test images on website as test because they were not labelled)
10. Built the deep learning model on this and used it for Williams and then Bush train split and got the f1 score on train and test for both

Initial Model Used:

The following is the deep learning model that was made on Dogs vs Cats dataset and then used for Williams train split and Bush train split to get the f1 score

Initial Model Summary:

```
[8]  model10.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 29, 29, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 2305
=================================================================
Total params: 58,049
Trainable params: 58,049
Non-trainable params: 0
```

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Initial Model Layers:

```
[9]  model10.layers
```

```
[<keras.layers.convolutional.Conv2D at 0x7f1ae0cf41d0>,
 <keras.layers.pooling.MaxPooling2D at 0x7f1ae0cf46a0>,
 <keras.layers.convolutional.Conv2D at 0x7f1ae0cf4198>,
 <keras.layers.pooling.MaxPooling2D at 0x7f1ae0ceeb00>,
 <keras.layers.convolutional.Conv2D at 0x7f1ae0cf4da0>,
 <keras.layers.pooling.MaxPooling2D at 0x7f1ae0db1e48>,
 <keras.layers.core.Flatten at 0x7f1ae0ceecf8>,
 <keras.layers.core.Dense at 0x7f1ae226ef98>]
```

Initial Model Inputs and Outputs:

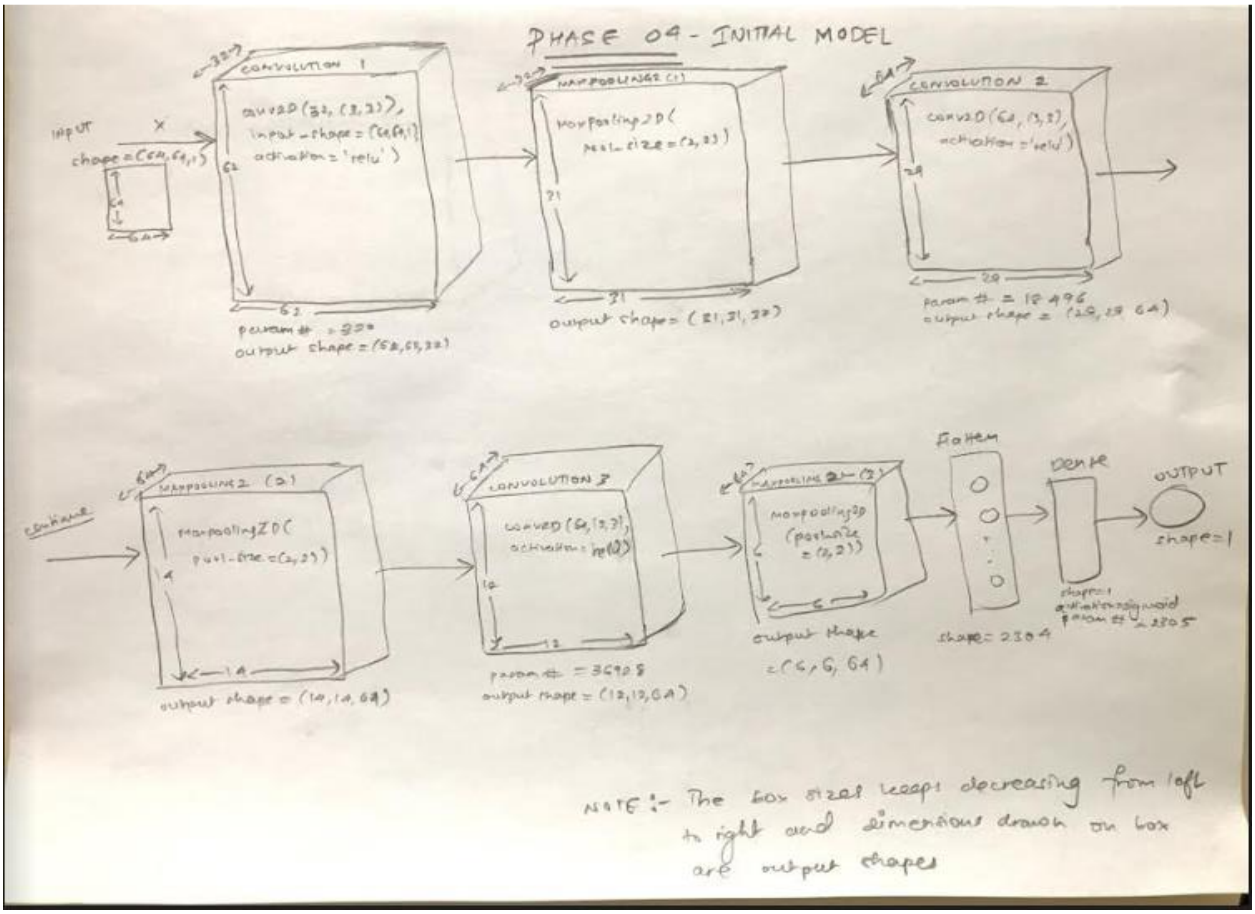
```
[10]  model10.inputs
```

```
[<tf.Tensor 'conv2d_4_input:0' shape=(?, 64, 64, 1) dtype=float32>]
```

```
[11]  model10.outputs
```

```
[<tf.Tensor 'dense_2/Sigmoid:0' shape=(?, 1) dtype=float32>]
```

Initial Model Diagram:



Then after, Compiled the model using optimizer='adam',loss='binary_crossentropy' and fit the model by using : model10.fit(X_train,y_train, epochs=50, batch_size=32, validation_data = (X_test, y_test))

Model Description:

Talking about the initial model it has the following type of the layers and the dimensions:

1. Input is the reshaped X file whose dimension after reshaping are 64*64*1
2. Conv2D - with activation function as relu , input_shape = (64, 64, 1), filter = 32 and kernel_size = (3,3) and this layer has output dimensions of 62*62*32
3. MaxPooling2 - that has pool_size of 2,2 and output dimension of 31*31*32 (The first two dimension of output shape will reduce to half of the one in previous step as we are using maxpooling and this applies to all the below steps where maxpooling is used)
4. Conv2D with activation function as relu, filter = 64 and kernel_size = (3,3) and this layer has output dimensions of 29*29*64
5. MaxPooling2 - that has pool_size of 2,2 and output dimension of 14*14*64
6. Conv2D with filter = 64 and kernel_size = (3,3) has output dimensions of f 12*12*64
7. MaxPooling2 - that has pool_size of 2,2 and output dimension of 6*6*64
8. Flatten, dimension of 2304
9. Then got Output layer by using sigmoid activation function and 1 as Dense and this has dimension of 1

Then after, Compiled the model using optimizer='adam',loss='binary_crossentropy' and fit the model by using: model10.fit(X_train,y_train, epochs=50, batch_size=32, validation_data = (X_test, y_test))

BUSH Model:

Steps performed to get f1 score for Bush Model:

1. Loaded the Initial model (Shown above)
2. Fit it on Bush train split with epochs=150, batch_size=32, validation_data = (X_test_b, y_test_b)
3. Predicted the classes for X_train_b and X_test_b using predict_classes
4. Calculated the F1 score for train and test using f1_score from sklearn.metrics

WILLIAMS Model:

Steps performed to get f1 score for Bush Model:

1. Loaded the Initial model (Shown above)
2. Fit it on Williams train split with epochs=30, batch_size=32, validation_data = (X_test_w, y_test_w)
3. Predicted the classes for X_train_w and X_test_w using predict_classes
4. Calculated the F1 score for train and test using f1_score from sklearn.metrics

ILLINOIS INSTITUTE OF TECHNOLOGY

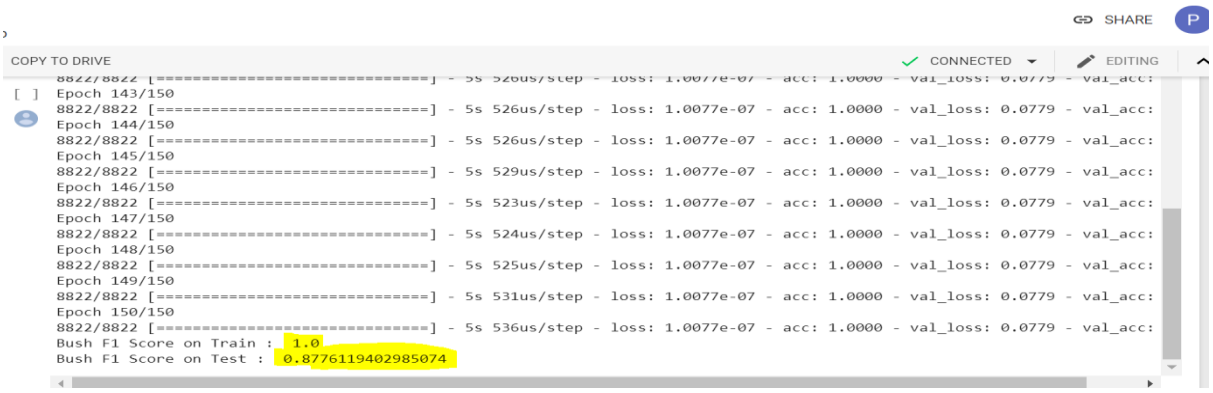
Pooja Patel
A20396099

BUSH Results on Model:

The following table shows the f1 score on train and test for Bush:

	F1 Score
Train	1.0
Test	0.8776

Screenshot of the output:



After doing transfer learning the f1 score increased both on train and test which was 0.995 and 0.848 in Phase 3 and now it is 1.0 and 0.8776 respectively. Hence, on train it increased by 0.5% and on test it increased by 3.49%. Hence, transfer learning helped in terms of getting better predictions and hence increasing the f1 score on both test and train for Bush.

WILLIAMS Results on Model:

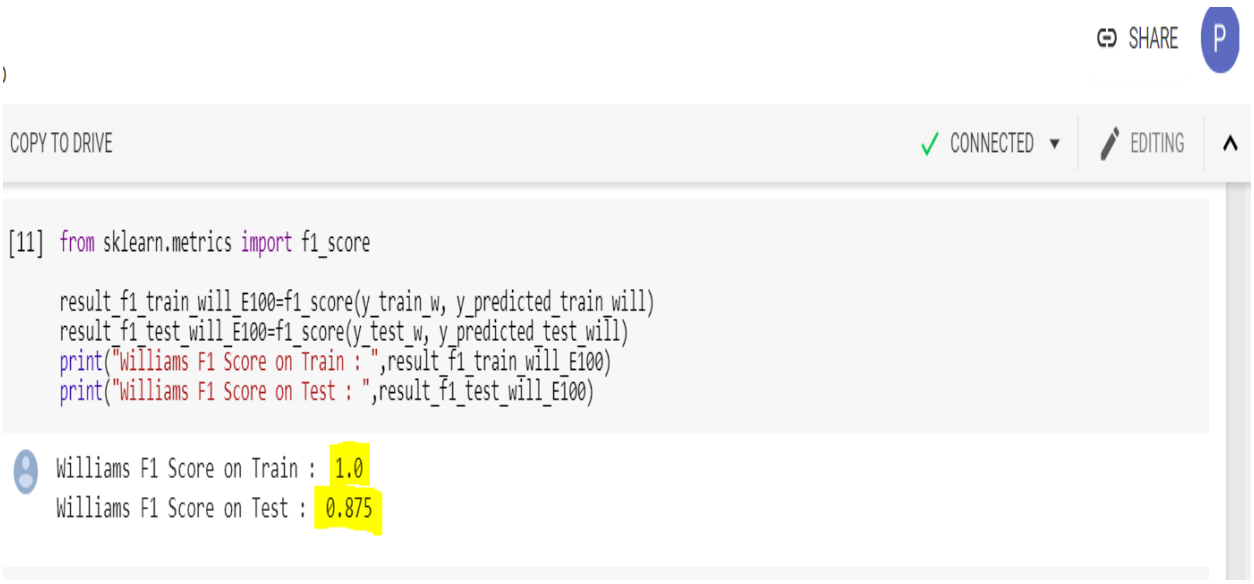
The following table shows the f1 score on train and test for Williams:

	F1 Score
Train	1.0
Test	0.875

ILLINOIS INSTITUTE OF TECHNOLOGY

Pooja Patel
A20396099

Screenshot of the output:



After doing transfer learning the f1 score increased on test which was 1.0 and 0.709 in Phase 3 and now it is 1.0 and 0.875 respectively. Hence, on test it increased by 23.41%. Hence, transfer learning helped in terms of getting better predictions and hence increasing the f1 score on test for Williams.

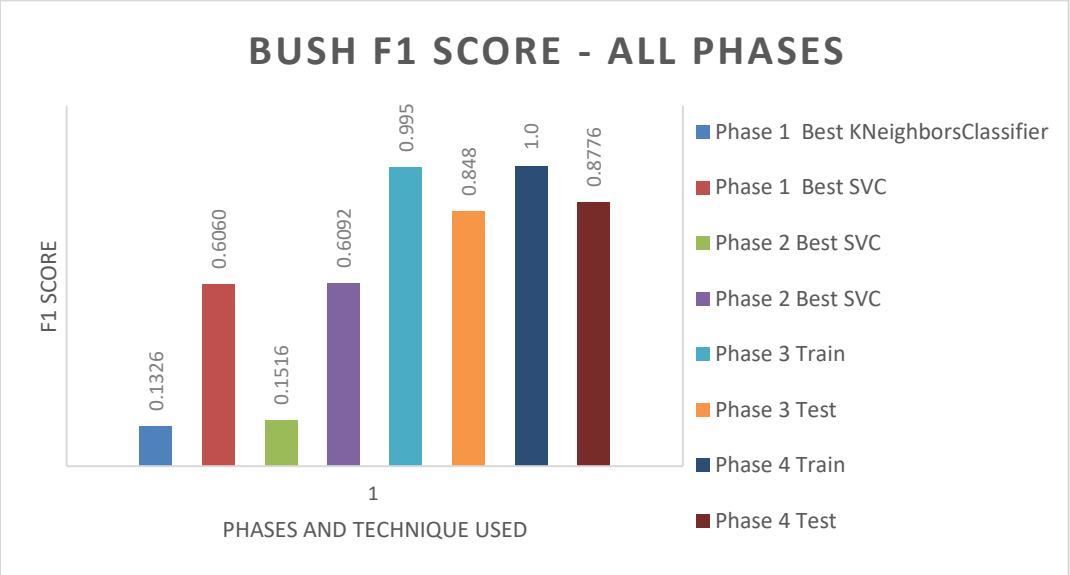
RESULT SUMMARY – ALL PHASES:

The following table shows the best f1 scores obtained in each Phase of the project.

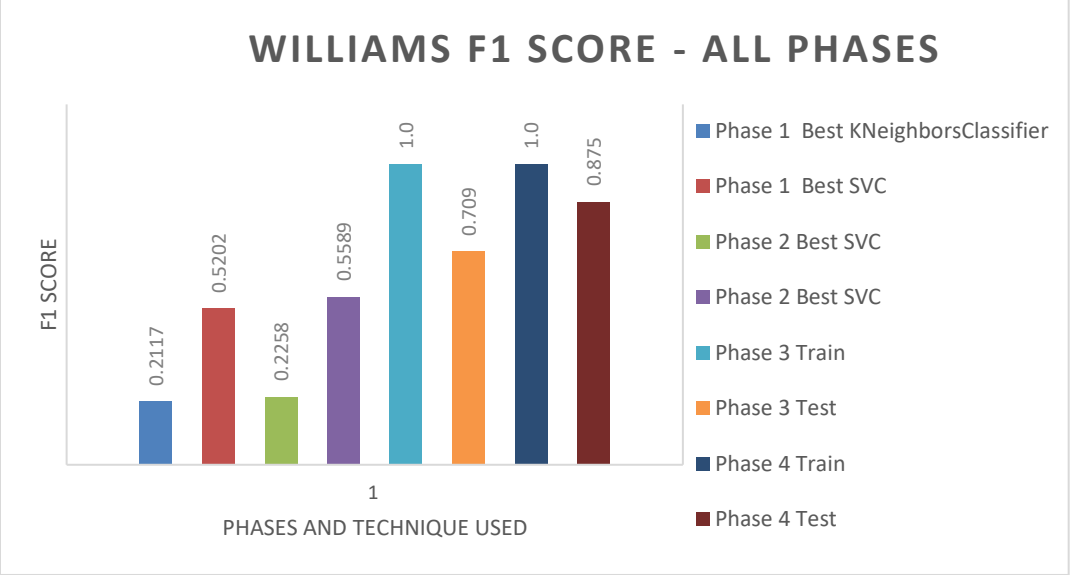
F1 Scores for all Phases			
		Bush	**Williams**
Phase 1	**Best KNeighborsClassifier**	0.1326	0.2117
	Best SVC	0.6060	0.5202
Phase 2	**Best KNeighborsClassifier**	0.1516	0.2258
	Best SVC	0.60918	0.5589
Phase 3	**Train**	0.995	1.0
	Test	0.848	0.709
Phase 4	**Train**	1.0	1.0
	Test	0.8776	0.875

Pooja Patel
A20396099

Visualizing the increase in F1 Score for Bush:



Visualizing the increase in F1 Score for Williams:



As I moved from 1st Phase to 4th Phase there was increase in the f1 score for each Phase for Bush as well as Williams. For 1st Phase for KneighborsClassifier the better performance was with 1 neighbor. Also, SVC turned out to be better classifier for this dataset. In the 2nd phase after dimensionality reduction

both KneighborsClassifier and SVC performed better than Phase 1 but among these two classifier again SVC performed better in terms of f1 score. Then went on to make the deep learning model for Bush and Williams and again it turned to perform better than the previous phases. Thereafter, in 4th Phase performed transfer learning by making a deep learning model on Dogs vs Cats dataset and using it on Williams and Bush train split and then calculating the f1 score which also turned out to be better than the previous Phases for both Bush and Williams. Hence, each phase helped to increase the f1 score for this Project.