MASTER OF TECHNOLOGY

U 2/6: Computational Intelligence – I

CA1: Regression and Classification using Neural Networks

Submitted to:

Dr. Zhu Fangming
Institute of System Science
National University of Singapore

Prepared by:

Gopesh Dwivedi (A0178338R)
Saurabh Semwal (A0178339N)
Tanmoy Chakraborty (A0178252B)

INSTITUTE OF SYSTEMS SCIENCE

Contents

Part I: Regression Using Neural Networks

1.1 Problem Statement and Dataset

The dataset in consideration summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. The goal is to predict the number of shares in social networks (popularity) which is a continuous variable. The articles were published by Mashable (www.mashable.com) and their content as the rights to reproduce it belongs to them. Hence, this dataset does not share the original content but some statistics associated with it. The original content be publicly accessed and retrieved using the provided URLs. There are **61 attributes** of which 58 are predictive attributes, 2 non-predictive, 1 target variable i.e. Number of shares.

Here is the description of the data set:

No.	Name	Description
1	url	URL of the article (non-predictive)
2	timedelta	Days between the article publication and the dataset acquisition (non-predictive)
3	n_tokens_title	Number of words in the title
4	n_tokens_content	Number of words in the content
5	n_unique_tokens	Rate of unique words in the content
6	n_non_stop_words	Rate of non-stop words in the content
7	n_non_stop_unique_tokens	Rate of unique non-stop words in the content
8	num_hrefs	Number of links
9	num_self_hrefs	Number of links to other articles published by Mashable
10	num_imgs	Number of images
11	num_videos	Number of videos
12	average_token_length	Average length of the words in the content
13	num_keywords	Number of keywords in the metadata
14	data_channel_is_lifestyle	Is data channel 'Lifestyle'?
15	data_channel_is_entertainment	Is data channel 'Entertainment'?
16	data_channel_is_bus	Is data channel 'Business'?
17	data_channel_is_socmed	Is data channel 'Social Media'?
18	data_channel_is_tech	Is data channel 'Tech'?
19	data_channel_is_world	Is data channel 'World'?
20	kw_min_min	Worst keyword (minimum shares)
21	kw_max_min	Worst keyword (maximum shares)
22	kw_avg_min	Worst keyword (Average shares)
23	kw_min_max	Best keyword (minimum shares)
24	kw_max_max	Best keyword (maximum shares)
25	kw_avg_max	Best keyword (Average shares)
26	kw_min_avg	Average keyword (minimum shares)
27	kw_max_avg	Average keyword (maximum shares)
28	kw_avg_avg	Average keyword (Average shares)
29	self_reference_min_shares	minimum shares of referenced articles in Mashable
30	self_reference_max_shares	maximum shares of referenced articles in Mashable
31	self_reference_avg_sharess	Average shares of referenced articles in Mashable
32	weekday_is_monday	Was the article published on a Monday?

33	weekday_is_tuesday	Was the article published on a Tuesday?
34	weekday_is_wednesday	Was the article published on a Wednesday?
35	weekday_is_thursday	Was the article published on a Thursday?
36	weekday_is_friday	Was the article published on a Friday?
37	weekday_is_saturday	Was the article published on a Saturday?
38	weekday_is_sunday	Was the article published on a Sunday?
39	is_weekend	Was the article published on the weekend?
40	LDA_00	Closeness to LDA topic 0
41	LDA_01	Closeness to LDA topic 1
42	LDA_02	Closeness to LDA topic 2
43	LDA_03	Closeness to LDA topic 3
44	LDA_04	Closeness to LDA topic 4
45	global_subjectivity	Text subjectivity
46	global_sentiment_polarity	Text sentiment polarity
47	global_rate_positive_words	Rate of positive words in the content
48	global_rate_negative_words	Rate of negative words in the content
49	rate_positive_words	Rate of positive words among non-neutral tokens
50	rate_negative_words	Rate of negative words among non-neutral tokens
51	avg_positive_polarity	Average polarity of positive words
52	min_positive_polarity	minimum polarity of positive words
53	max_positive_polarity	maximum polarity of positive words
54	avg_negative_polarity	Average polarity of negative words
55	min_negative_polarity	minimum polarity of negative words
56	max_negative_polarity	maximum polarity of negative words
57	title_subjectivity	Title subjectivity
58	title_sentiment_polarity	Title polarity
59	abs_title_subjectivity	Absolute subjectivity level
60	abs_title_sentiment_polarity	Absolute polarity level
61	**shares**	**Number of shares (target)**

Table 1: Data Dictionary of Regression Dataset

1.2 Feature Engineering

The dataset was checked for missing values, data inconsistencies and outliers. No missing values were found in the data. However, we scaled and normalized the predictor numerical variables. Also, the target variable was scaled between 0-1 to meet our requirements. We removed some redundant attributes which have near zero variance or have linear relationship i.e. high correlation. One such variable was "is_weekend" which is already present in attributes for Saturday and Sunday. We also split the data into 80:20 ratio for training and testing respectively.

1.2.1 Dimensionality Reduction

Due to large number of input variables, we applied principal component analysis to reduce the number of predictor variables. After reduction, 30 Principal components were used to train the model which contained 92.4% information value.

1.3 Tools Used

We have done the implementation in **R tool** using **"RSNNS"** and **"neuralnet" packages**. Neuralnet provides with the training of neural networks using the backpropagation, resilient backpropagation with or without weight backtracking. The Stuttgart Neural Network Simulator (SNNS) is a library containing many standard implementations of neural networks. RSNNS package wraps the SNNS functionality to make it available from within R. A radial basis function network is an artificial neural network that uses radial basis functions as activation functions. We have implemented RBF using RSNNS package.

1.4 Neural Networks and Ensemble Architecture

1.4.1 MLFF Neural Network using resilient backpropagation with weight backtracking

The first Neural Network that we trained is a Multi-Layer Feed Forward Neural network using resilient backpropagation with weight backtracking. In a MLFF network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (of each layer) and to the output nodes. There are no cycles or loops in the network.

Rprop, short for resilient backpropagation, is a learning heuristic for supervised learning in feedforward artificial neural networks. This is a first-order optimization algorithm. Rprop takes into account only the sign of the partial derivative over all patterns (not the magnitude), and acts independently on each "weight". For each weight, if there was a sign change of the partial derivative of the total error function compared to the last iteration, the update value for that weight is multiplied by a factor $\eta-$, where $\eta- < 1$. If the last iteration produced the same sign, the update value is multiplied by a factor of $\eta+$, where $\eta+ > 1$. The update values are calculated for each weight in the above manner, and finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative, to minimize the total error function. $\eta+$ is empirically set to 1.2 and $\eta-$ to 0.5.

To overcome the inherent disadvantages of pure gradient-descent, RPROP performs a local adaptation of the weight-updates according to the behaviour of the error function. In substantial difference to other adaptive techniques, the effect of the RPROP adaptation process is not blurred by the unforeseeable influence of the size of the derivative but only dependent on the temporal behaviour of its sign. This leads to an efficient and transparent adaptation process.

We created a MLFF network with two hidden layers of size 10 neuron each, with error function as SSE

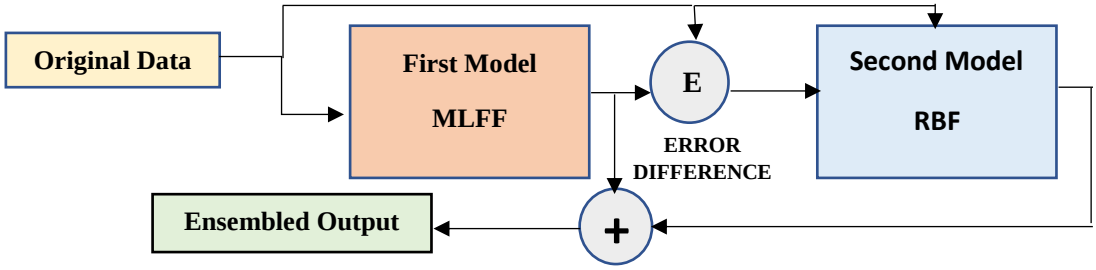
1.4.2 Radial Basis Function Neural Network

We implemented a RBF NeuralNet as the second neural network for regression. We used a single hidden layer with 40 nodes. RBF networks are feed-forward networks with one hidden layer. Their activation is not sigmoid (as in MLP), but radially symmetric (gaussian). Thereby, information is represented locally in the network (in contrast to MLP, where it is globally represented). Advantages of RBF networks in comparison to MLPs are mainly, that the networks are more interpretable, training ought to be easier and faster, and the network only activates in areas of the feature space where it was trained.

1.4.3 Ensemble Architecture

The goal of ensemble regression is to combine several models to improve the prediction accuracy on learning problems with a numerical target variable. The process of ensemble learning for regression can be divided into three phases: the generation phase, in which a set of candidate models is induced, the pruning phase, to select of a subset of those models and the integration phase, in which the output of the models is combined to generate a prediction.

We used an alternative approach to creating Ensembles. First, a model was generated using the original data. Second, another model was generated that estimates the error of the predictions of the first model and generates an ensemble that combines the prediction of the previous model with the correction of the current one. Finally, the output of the two ensembles is added up to give the final predictions.



1.5 Results and Conclusion

Model	Training Set RMSE	Testing Set RMSE
Base model on raw data	0.0311	0.0354
MLFF with Rprop on engineered data	0.0132	0.0138
RBF on engineered data	0.0134	0.0132
Ensembled Model	0.0127	0.0130

Table 2: Comparison of Various Models Architectures

RBF and MLFF+BP both being nonlinear networks can approximate arbitrary nonlinear functional mappings between multidimensional spaces. RBF are often considered to be superior to MLFF+BP in case of complex mappings but here using adaptive method for BP we got almost comparable results. We created an ensemble of these two models. MLFF+ rprop approximations were subtracted from the response values and the second model(RBF) learned this error. As a result, we did a summation of the result of first model and error approximations of the MLFF+RProp model learned by RBF. The final results were an improvement over the results of the individual models.

Part 2: Classification Using Neural Networks

2.1 Problem Statement and Dataset

Considering a hypothetical client which is a prominent brand of lady's hand bag, who want to develop a Machine Learning Model which will help them in identifying female users of a popular social networking site using their profile display pictures. This would assist them in targeting relevant campaigns and advertisement of their products.

To train such classification model, we used a publicly available repository of Human face images from Github. We implemented a deep learning models (combining CNN and ANN), then created an ensemble of the model where final category is selected by majority voting which led to better sensitivity ("Female" being the positive class) and precision, which is our final objective.

2.2 Tools Used

The model is built on **Python tool** using **the Keras** toolbox or library. We trained a combination of **CNN** i.e. convolution neural network and **ANN** (Artificial Neural Network). Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

2.3 Implementation

Various steps involved in our approach are as follows:

1. Data Collection and segregation of the data into train, validation and test sets.
2. Deciding the evaluation parameters
3. Building classifiers (we build three classifiers/models) using deep learning techniques (CNN, ANN)
4. Developing an ensemble of the three classifiers

These are explained in detail in this section.

Step 1- Data Collection and segregation of the data into train, validation and test sets.

As mentioned before we collected woman and man pictures which looks like display pictures from Github (hypothetically from the social networking site), the total number of images was 1888 out of which 1183 (~63 %) are woman pictures and 705 are male pictures (~37%). Please find the segregation details in the table below.

Datasets	Image Count	%(Total data)	# (woman images)	# (man images)	% (woman images)	% (man images)
Full Dataset	1,888	100	1,183	705	63	37
Train	1,272	67	828	444	65	35
Validation	301	16	196	105	65	35
Test	315	17	159	156	53	47

We have purposefully kept the Test data less biased as because the social networking site has almost equal number members. The train and validation data (for parameter tuning) is kept biases and we want to train and validate our models in a way that they learn our female features well and thus serve our ultimate objective of identifying female display pictures.

Note: Due to insufficient data we were not able to train, validate our model on different sets of data for the three classifiers, though we are aware the problems associated with correlation between the models.

Step 2- Deciding the evaluation parameters

As our objective was more focused on identifying woman class accurately, we identify the following three evaluation parameters:

Sensitivity/Recall: Woman being the positive class we focused our top most focus is on sensitivity score so that our client doesn't lose a potential customer (our target on test data for the ensembled model was >90 %)

Precision: We also must keep in mind that our ensembled model should not give high false positives. In such case our client will lose money advertising to the wrong target. (our target on test data for the ensembled model was >60 %)

F1 score: To have little bit of balance between the above two metrics, so that we don't compromise heavily on precision to achieve a high recall, we focused on a fair balanced value of f1 score (our target on test data for the ensembled model was >0.60).

Note: To satisfy our ensemble model evaluation target, we gave individual focus of the above three parameters in individual single standalone models. In other words, we made three independent classifiers/models).

Step3-Building classifiers (we build three classifiers/models) using deep learning techniques (CNN, ANN)

We built all the classifier from scratch (as transfer learning with vgg16 was not giving us a satisfactory result on this data, we decided not to go with transfer learning and build classifiers from scratch)

2.4 Results and Evaluation

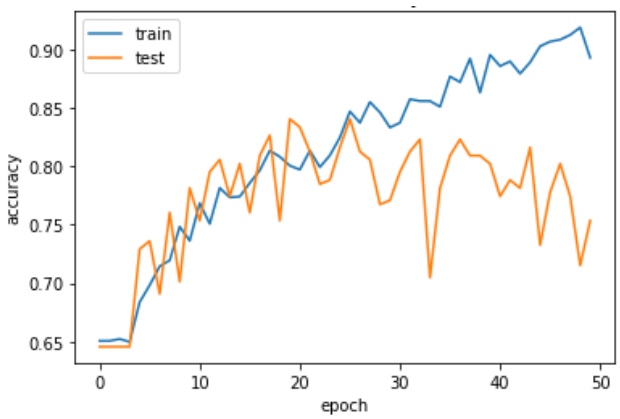
Classifier No.1

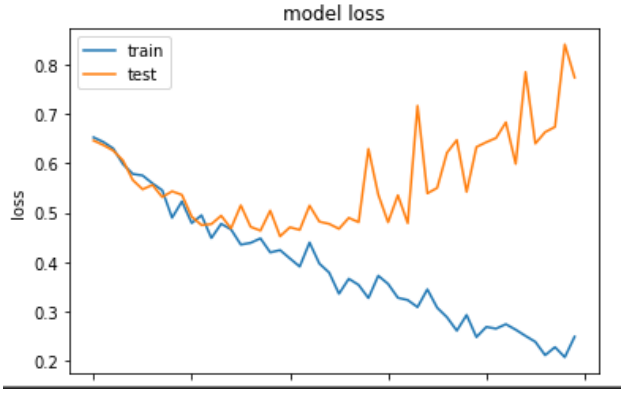
we built this model using with 3 convolution layers and 7 dense layers joined by a flatten layer. Please find the model architecture summary below

Layer (type)	Output Shape	Param #
conv2d_216 (Conv2D)	(None, 30, 30, 64)	1792
max_pooling2d_170 (MaxPoolin	(None, 15, 15, 64)	0
zero_padding2d_135 (ZeroPadd	(None, 17, 17, 64)	0
conv2d_217 (Conv2D)	(None, 15, 15, 64)	36928
conv2d_218 (Conv2D)	(None, 13, 13, 64)	36928
max_pooling2d_171 (MaxPoolin	(None, 6, 6, 64)	0
dropout_154 (Dropout)	(None, 6, 6, 64)	0
flatten_65 (Flatten)	(None, 2304)	0
dense_604 (Dense)	(None, 64)	147520
dropout_155 (Dropout)	(None, 64)	0
dense_605 (Dense)	(None, 64)	4160
dense_606 (Dense)	(None, 64)	4160
dense_607 (Dense)	(None, 64)	4160
dense_608 (Dense)	(None, 64)	4160
dense_609 (Dense)	(None, 64)	4160
dropout_156 (Dropout)	(None, 64)	0

```
_________________________________________________________________
dense_610 (Dense)            (None, 1)                 65
=================================================================
Total params: 244,033
Trainable params: 244,033
Non-trainable params: 0
```

So, our model1 have 244,033 trainable parameters.
Optmizer- adam
Loss-binary_cross entropy
Learning rate-0.001
Batch_size train=32
Batch_size_validation=32
Batch_size_test=1
Data Augmentation Train-rescale- rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip
Image size- 32 * 32*3
Data Augmentation Test- rescale=1./255
Method-Flow from directory Keras
No of epochs- 1st we run our classifier for 50 epochs and checked the loss and accuracy graph for train vs validation.





From the graphs, we can see that around after 25 epochs our model starts overfitting. So, we re-run our model with same initial weights till 25 epochs.

We got a validation overall accuracy of **78 %.** Then, we tested our model against the test data, please find below the test result on the above-mentioned evolution parameters after we see the confusion matrix.

Actual	Predicted	
	Man	Woman
Man	50	106
Women	8	151

```
: sensitivity_arch1=(tp1/(tp1+fn1))*100
 a women
: sensitivity_arch1#Recall
: 94.9685534591195

precision_arch1=(tp1/(tp1+fp1))*100
precision_arch1
58.754863813229576

f1_acore_arch1
0.5966256290438534
```

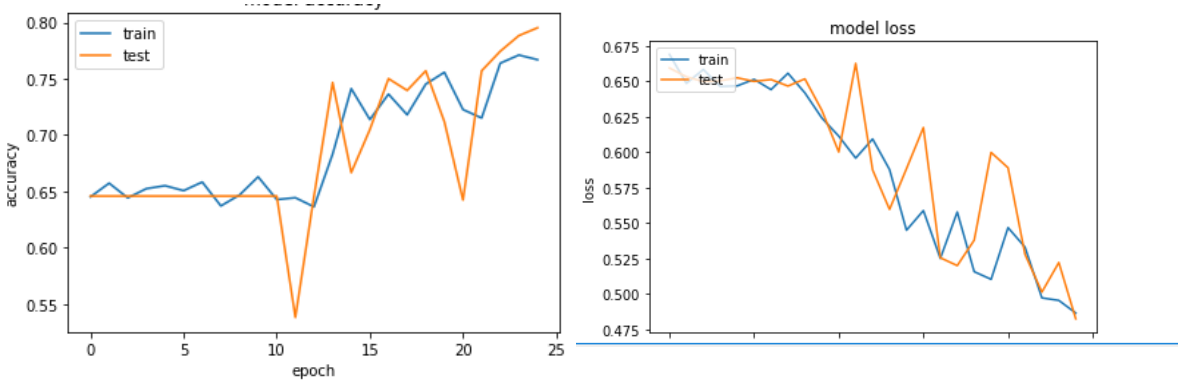
Classifier No. 2

we built this model using with 4 convolution layers and 7 dense layers joined by a flatten layer. Please find the model architecture summary below

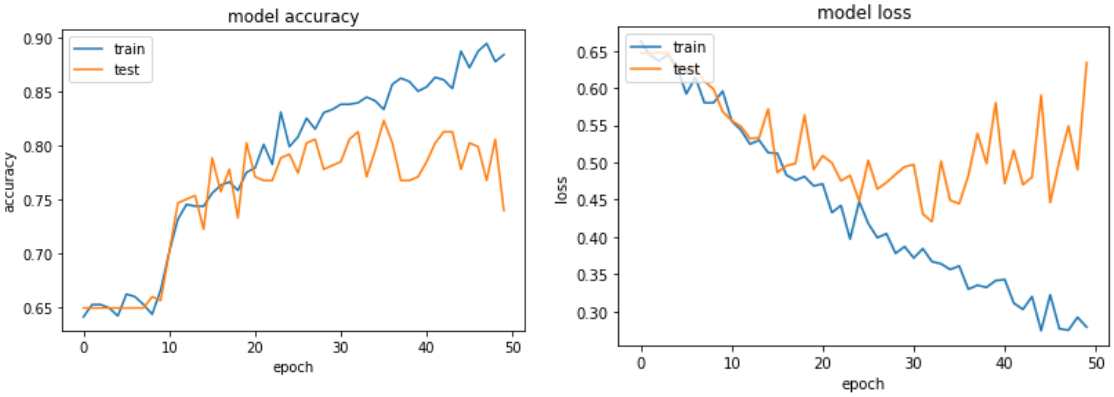
Layer (type)	Output Shape	Param #
conv2d_227 (Conv2D)	(None, 30, 30, 64)	1792
max_pooling2d_180 (MaxPoolin	(None, 15, 15, 64)	0
zero_padding2d_144 (ZeroPadd	(None, 17, 17, 64)	0
conv2d_228 (Conv2D)	(None, 15, 15, 64)	36928
zero_padding2d_145 (ZeroPadd	(None, 17, 17, 64)	0
max_pooling2d_181 (MaxPoolin	(None, 8, 8, 64)	0
conv2d_229 (Conv2D)	(None, 6, 6, 64)	36928
max_pooling2d_182 (MaxPoolin	(None, 3, 3, 64)	0
zero_padding2d_146 (ZeroPadd	(None, 5, 5, 64)	0
dropout_167 (Dropout)	(None, 5, 5, 64)	0
conv2d_230 (Conv2D)	(None, 3, 3, 64)	36928
zero_padding2d_147 (ZeroPadd	(None, 5, 5, 64)	0
max_pooling2d_183 (MaxPoolin	(None, 2, 2, 64)	0
dropout_168 (Dropout)	(None, 2, 2, 64)	0
flatten_68 (Flatten)	(None, 256)	0
dense_627 (Dense)	(None, 64)	16448
dropout_169 (Dropout)	(None, 64)	0
dense_628 (Dense)	(None, 64)	4160
dense_629 (Dense)	(None, 64)	4160
dense_630 (Dense)	(None, 64)	4160
dropout_170 (Dropout)	(None, 64)	0
dense_631 (Dense)	(None, 64)	4160
dense_632 (Dense)	(None, 64)	4160
dense_633 (Dense)	(None, 64)	4160
dropout_171 (Dropout)	(None, 64)	0
dense_634 (Dense)	(None, 1)	65

Total params: 154,049
Trainable params: 154,049
Non-trainable params: 0

So, our model2 have 154,049 trainable parameters.
Optimizer- adam
Loss- binary_cross entropy
Learning rate-0.001
Image size- 32 * 32*3
Data Augmentation Train-rescale- rescale=1./255,shear_range=0.2,zoom_range=0.2, horizontal flip
Data Augmentation Test- rescale=1./255
Method-Flow from directory Keras
No of epochs- 1st we run our classifier for 25 epochs going by the 1st model theory and checked the loss and accuracy graph for train vs validation.



From these graph, we can see that at 25 epochs our is still improving, accuracy increasing and loss decreasing. So, we increased the number of epochs to 50 and rerun our model with same initial weights.



From the graph, we can see that around after 35 epochs our model starts overfitting. So, we re-run our model with same initial weights till 35 epochs.

We got a validation overall accuracy of **81.94%.** Then we tested our model against the test data, please find below the test result on the above-mentioned evaluation parameters.

Actual	Predicted	
	Man	Woman
Man	65	91
Women	12	147

```
sensitivity_arch2=(tp2/(tp2+fn2))*100 #
# women
sensitivity_arch2
92.45283018867924

precision_arch2=(tp2/(tp2+fp2))*100
precision_arch2
61.76470588235294

f1_score_arch2
0.6492470351671873
```

So, model 2 has a higher precision and lower sensitivity compared to model1, F1 score of model2 is better.

Classifier No. 3

we built this model using with 3 convolution layers and 7 dense layers joined by a flatten layer. It's architecture is almost same as that of architecture 1 which gave us a higher sensitivity (our major objective), the only difference is here our algorithm is different, our optimizer is RMSprop, our loss function is mean-squared error.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_231 (Conv2D)          (None, 30, 30, 64)        1792
_________________________________________________________________
max_pooling2d_184 (MaxPoolin (None, 15, 15, 64)        0
_________________________________________________________________
zero_padding2d_148 (ZeroPadd (None, 17, 17, 64)        0
_________________________________________________________________
conv2d_232 (Conv2D)          (None, 15, 15, 64)        36928
_________________________________________________________________
conv2d_233 (Conv2D)          (None, 13, 13, 64)        36928
_________________________________________________________________
zero_padding2d_149 (ZeroPadd (None, 15, 15, 64)        0
_________________________________________________________________
max_pooling2d_185 (MaxPoolin (None, 7, 7, 64)          0
_________________________________________________________________
dropout_172 (Dropout)        (None, 7, 7, 64)          0
_________________________________________________________________
flatten_69 (Flatten)         (None, 3136)              0
_________________________________________________________________
dense_635 (Dense)            (None, 64)                200768
_________________________________________________________________
dropout_173 (Dropout)        (None, 64)                0
_________________________________________________________________
dense_636 (Dense)            (None, 64)                4160
_________________________________________________________________
dense_637 (Dense)            (None, 64)                4160
_________________________________________________________________
dropout_174 (Dropout)        (None, 64)                0
_________________________________________________________________
dense_638 (Dense)            (None, 64)                4160
_________________________________________________________________
dropout_175 (Dropout)        (None, 64)                0
_________________________________________________________________
dense_639 (Dense)            (None, 64)                4160
_________________________________________________________________
dense_640 (Dense)            (None, 64)                4160
_________________________________________________________________
dropout_176 (Dropout)        (None, 64)                0
_________________________________________________________________
dense_641 (Dense)            (None, 1)                 65
=================================================================
Total params: 297,281
Trainable params: 297,281
Non-trainable params: 0
```

So, our model1 have 297,281 trainable parameters.
Optimizer- RMSprop
Loss-binary mean-squared error
Learning rate-default
Batch size train=32
Batch size validation=32
Batch size test=1
Image size- 32 * 32*3
Data Augmentation Train-rescale- rescale=1./255,shear_range=0.2,zoom_range=0.2 ,horizontal flip
Data Augmentation Test- rescale=1./255
Method-Flow from directory Keras
No of epochs- Like we did for our previous classifiers we checked the loss and accuracy graph for train vs test and in this case, we got the optimal number to be 30. We got a validation overall accuracy of 78 % after running the train data for 30 epochs.
Then we tested our model against the test data, please find below the test result on the above-mentioned evaluation parameters after we see the confusion matrix.

Actual	Predicted	
	Man	Woman
Man	22	134
Women	4	155

```
sensitivity_arch3=(tp3/(tp3+fn3))*100
 women
sensitivity_arch3#Recall
97.48427672955975
precision_arch3=(tp3/(tp3+fp3))*100
precision_arch3
53.63321799307958
f1_acore_arch1      f1_acore_arch3
0.5966256290438534  0.4668612637362637
```

Thus, we can see tough the 3rd classifier gave us a very high sensitivity but our precision and overall f1 score decreased

Ensemble Model

All the above three models are stronger than the others in some evaluation parameters, so to build an optimal classification network we used the ensemble of these three classifiers. The method we choose is voting where the majority wins, for example if claasifier1 and 3 predict an image as a woman the ensemble network will predict the output as a woman even if classsifier2 would have predicted it as a man.

We tested our ensemble against the test data to see the improvement over individual model result, please find below the test result on the above-mentioned evaluation parameters after we see the confusion matrix.

Actual	Predicted	
	Man	Woman
Man	39	117
Women	7	152

```
precision_ensemble=(tp_e/(tp_e+fp_e))*100
precision_ensemble
58.01526717557252
f1_acore_ensemble
0.5482094938465809
sensitivity_ensemble#Recall
95.59748427672956
```

2.5 Conclusion

By creating the proposed Ensemble network architecture, we have achieved our main goal as the proposed model has a good sensitivity for target class. Though we fall a bit short from our expected F1 score and precision of 60 % but at the end the result is optimal and we hope the client will be satisfied with the results.