ISEN 613-FALL 2019 COURSE PROJECT

PROJECT REPORT ON MODELS FOR OBJECT PREDICTION IN AN IMAGE

Submitted by

Kasturee Rajeev Chaphekar (327005571)

Soham Milind Deshpande (829009731)

Vaidehi Vivek Natu (830000028)

Vaishnavi Vilas Patki (430000405)

Executive Summary

To: President X
From: Data Consultant Y
Subject: Development of models to predict an object in an image

The purpose of this analysis is to find the best model to predict an object in an image and classify it in one of the 10 categories. Object prediction and recognition is gaining importance and has potential applications in various areas like automated vehicle systems, optical character recognition, medical imaging, object counting etc.

Initially, the data of 50,000 images was given. Each image was of 32X32 pixels and each pixel had some values of red, green and blue color which constitute overall appearance of that pixel. Such 32X32X3= 3072 values of red, green and blue colors constitute a total image. Data analysis starts with the identification of number of observation and number of variables. In this case, 50,000 is the number of observations and 3072 is the number of variables.

The first task was to determine which classification method can give the maximum accuracy on the test data within the training data. Principle Components Analysis (PCA) was performed to select the smaller number of representative variables among 3072. The number of Principle Components (PCs) which contain 90% of the total information about the observations were selected for further model development.

Several classification models like Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors (KNN), Random Forests and Support Vector Machines were developed and tested for accuracy. Among all these techniques, QDA resulted in maximum accuracy of 50.06%. It has the lowest computation time of 8 seconds. The second-best model with 48.16% accuracy was identified as Random Forests. The third-best model was KNN with an accuracy of 37.48%.

The highest accuracy model (QDA) model was then used to find the testing accuracy on the provided test data. The testing accuracy is reported as 50.34%. Efforts were taken to improve this testing accuracy by varying multiple parameters both individually and at the same time. This resulted in the final accuracy of the modified model as 51.74%. It is evident that there is an increase in accuracy of 1.4% after the improvements on previous model were performed.

All the team members took equal efforts for the successful completion of this project. Everyone was actively involved in all the steps i.e. from initial brainstorming to final refining of models. This report is also a joint effort of all the team members.

<u>Pre-Processing</u>

The goal of the project was to correctly identify the object in the given image. The model is trained using the dataset, which contains 50,000 color images coded in RGB values. There are 3072 columns in which first 32x32 columns correspond to the “RED” values of the images, second 32x32 columns correspond to the “GREEN” values of the images and, the third 32x32 columns correspond to the “BLUE” values of the images. First row Corresponds to the RGB values of first image and so on. The entry values have range from 0 to 255, defining the intensity of the corresponding R, G or B value. There 10 classes corresponding to different objects that an image from the dataset can contain. They are coded by labels 1 to 10.

Data Frame Creation:
The first step was to put the given dataset in the form of data frame in order to include the relevant data out of all non-required data and omit the null values. Data was then split into training and testing datasets having split as 75% and 25%.
Intuitively, if training data is taken for a fraction more than 0.75 of original data frame, the model might overfit and testing accuracy might be much affected. On the other hand, if training data is taken for a fraction less than 0.75 of the original data frame, the model might overestimate the test error.

Principle Component Analysis (PCA):
In order to deal with 3072 predictors, dimension reduction is a feasible solution. Amongst the top dimensional reduction algorithms, PCA gives good performance. PCA makes the work of feature manipulation easier and helps to improve the results of the classifier. Hence, first PCA was applied on the training data set. [2]

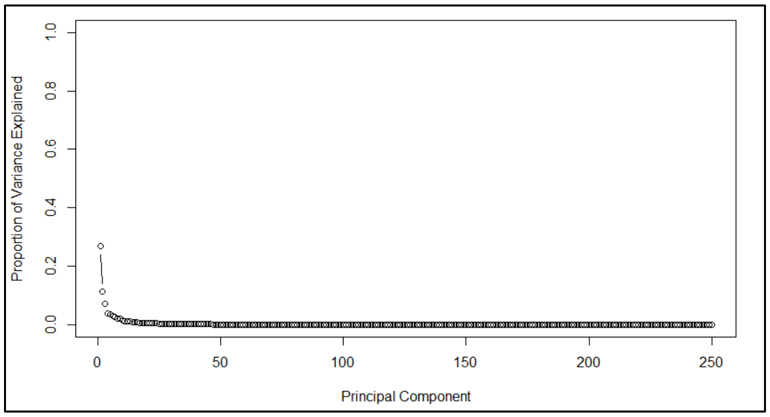
#R-Workspace

```
> pc.x <- prcomp (train.x , scale =TRUE)
```

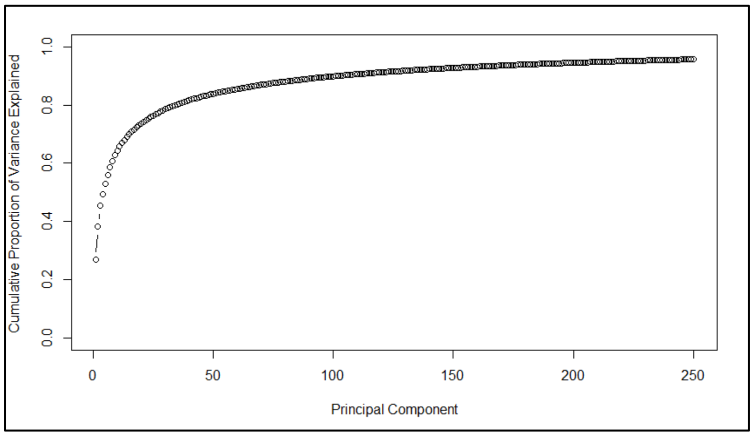
Plot of PCs against the Proportion of variance explained is given below.

#R-Workspace

```
> pr.var =pc.x$sdev ^2
> pve=pr.var/sum(pr.var)
> plot(pve [1:250], xlab="Principal Component", ylab="Proportion of Variance Explained", ylim=c(0,1) ,type="b")
```



```
> plot(cumsum (pve[1:250] ), xlab=" Principal Component ", ylab ="Cumulative Proportion of Variance Explained ", ylim=c(0,1)
 , type="b")
```



```
> cumsum (pve[1:250])*100
```

```
  [97] 89.60829 89.68668 89.76439 89.84179 89.91793 89.99348 90.06821 90.14174 90.21487 90.28576 90.35649 90.42530
 [109] 90.49391 90.56207 90.62889 90.69460 90.76015 90.82542 90.88937 90.95252 91.01506 91.07715 91.13832 91.19830
 [121] 91.25777 91.31605 91.37366 91.43101 91.48766 91.54340 91.59906 91.65428 91.70820 91.76190 91.81498 91.86744
 [133] 91.91959 91.97090 92.02200 92.07186 92.12088 92.16963 92.21818 92.26621 92.31381 92.36102 92.40787 92.45418
 [145] 92.50025 92.54596 92.59146 92.63644 92.68072 92.72482 92.76858 92.81192 92.85501 92.89729 92.93920 92.98074
 [157] 93.02186 93.06243 93.10267 93.14207 93.18127 93.22018 93.25871 93.29670 93.33438 93.37191 93.40901 93.44586
 [169] 93.48259 93.51913 93.55514 93.59074 93.62587 93.66054 93.69495 93.72905 93.76285 93.79659 93.83020 93.86355
 [181] 93.89653 93.92920 93.96172 93.99402 94.02598 94.05770 94.08936 94.12061 94.15180 94.18278 94.21353 94.24391
 [193] 94.27411 94.30407 94.33365 94.36280 94.39160 94.42023 94.44869 94.47706 94.50518 94.53309 94.56054 94.58771
 [205] 94.61469 94.64163 94.66851 94.69498 94.72131 94.74747 94.77350 94.79940 94.82504 94.85061 94.87610 94.90132
 [217] 94.92617 94.95081 94.97508 94.99924 95.02319 95.04710 95.07081 95.09433 95.11776 95.14098 95.16405 95.18697
 [229] 95.20986 95.23269 95.25540 95.27804 95.30044 95.32272 95.34491 95.36684 95.38863 95.41032 95.43195 95.45334
 [241] 95.47467 95.49584 95.51677 95.53759 95.55820 95.57878 95.59916 95.61944 95.63961 95.65971
```

Here, it can be observed that 95.66% of the variance is explained with around 250 PCs, but 105 PCs explain around 90% of the variance, so first 105 PCs were selected for modelling as there is no significant increase in the variance explanation after that.

TRAIN AND TEST DATA based on PCA:
As stated above, proportion of the variance explained by the 105 PCs is about 90%. Therefore, new training data frame containing first 105 PCs were created. Now to ensure that the Direction of the variation is same in test data as the train data, the new test data frame was created using the predict function.[2]

Model 1: QDA- Quadratic Discriminant Analysis

Discriminant analysis is popular when classification is to be done in more than 2 classes. The fundamental idea behind QDA is to find the posterior probability of the observation belonging to k^th class using Bayes' theorem. It assumes the observations from each class follows Normal distribution. Estimates for class-mean vector (μ_k) and class-covariance matrix (Σ_k) are used to find the classification boundary. The observation is assigned to a particular class for which the $\delta_k(x)$ is largest.[1]

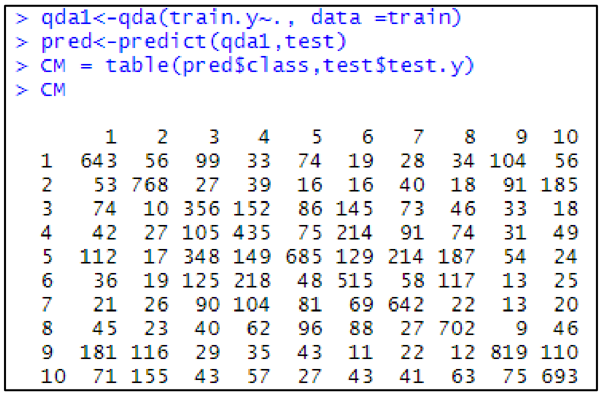
$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\mathbf{\Sigma}_k^{-1}(x-\mu_k)+\log\pi_k \\ &= -\frac{1}{2}x^T\mathbf{\Sigma}_k^{-1}x + x^T\mathbf{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}_k^{-1}\mu_k + \log\pi_k\end{aligned}$$ [1]

QDA assumes quadratic decision boundary. Hence, this method of classification lies in between non-parametric methods and linear methods.

While performing PCA, the data was standardized. The data containing PCs follows the basic assumption of normal distribution. Hence, QDA might be one of the best models to classify the images.

Training data for this object detection task has large number of observations i.e. n is large. In general, as 'n' is increased, variance decreases. This drives the intuition that the model will not suffer from high covariance and hence it is feasible to consider QDA which assumes separate covariance matrix for 10 classes.

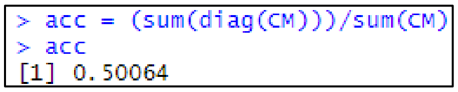
#R-Workspace


```
> qda1<-qda(train.y~., data =train)
> pred<-predict(qda1,test)
> CM = table(pred$class,test$test.y)
> CM

       1   2   3   4   5   6   7   8   9  10
  1  643  56  99  33  74  19  28  34 104  56
  2   53 768  27  39  16  16  40  18  91 185
  3   74  10 356 152  86 145  73  46  33  18
  4   42  27 105 435  75 214  91  74  31  49
  5  112  17 348 149 685 129 214 187  54  24
  6   36  19 125 218  48 515  58 117  13  25
  7   21  26  90 104  81  69 642  22  13  20
  8   45  23  40  62  96  88  27 702   9  46
  9  181 116  29  35  43  11  22  12 819 110
  10  71 155  43  57  27  43  41  63  75 693
```

The QDA is used to analyze a confusion matrix between true labels of the classes for our testing observations and the labels that are predicted by QDA model.

The numbers on the diagonal of the confusion matrix represent the number of images that were classified correctly by our model. Rest of the numbers indicate misclassified images.

The reported accuracy of the QDA model is 50.064%


```
> acc = (sum(diag(CM)))/sum(CM)
> acc
[1] 0.50064
```

QDA is the best model considering two major factors:

1. Prediction accuracy : Maximum test accuracy is obtained on a QDA model (50.064%)

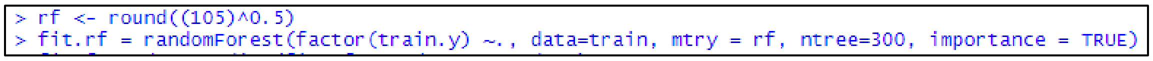
2. Computation time : Least computation time compared to other methods like Random Forests and KNN

Model 2: Random Forest

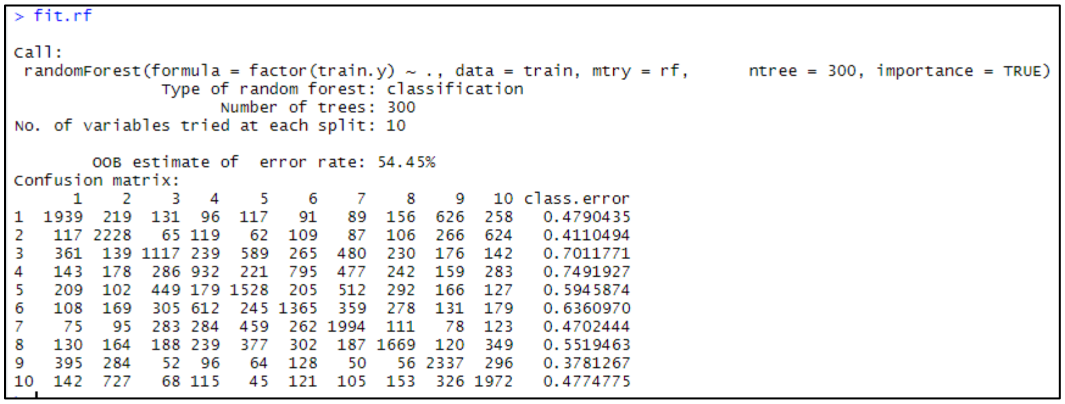
This classification algorithm averages the predictions from all the uncorrelated classification trees which are generated on different bootstrapped training data sets. The main aim of the random forests is reduction in variance.

The Random forest model was tuned by varying ‘mtry’ and ‘ntree’. The values were varied between 10 to 105 for the number of predictors to be considered and for number of trees the values considered were between 100 to 500 trees. There was no significant increase in accuracy after increasing the number of trees above 300, hence ntree=300 was selected. The best results were obtained when mtry was equal to square root of the number of PCs used.

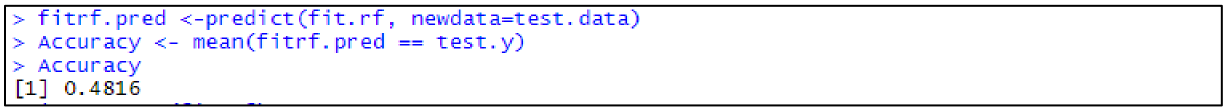
#R-Workspace:



```
> rf <- round((105)^0.5)
> fit.rf = randomForest(factor(train.y) ~., data=train, mtry = rf, ntree=300, importance = TRUE)
```



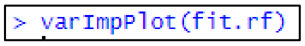
```
> fit.rf

Call:
 randomForest(formula = factor(train.y) ~ ., data = train, mtry = rf,      ntree = 300, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 300
No. of variables tried at each split: 10

        OOB estimate of  error rate: 54.45%
Confusion matrix:
      1    2    3   4    5    6    7    8    9   10 class.error
1  1939  219  131  96  117   91   89  156  626  258   0.4790435
2   117 2228   65 119   62  109   87  106  266  624   0.4110494
3   361  139 1117 239  589  265  480  230  176  142   0.7011771
4   143  178  286 932  221  795  477  242  159  283   0.7491927
5   209  102  449 179 1528  205  512  292  166  127   0.5945874
6   108  169  305 612  245 1365  359  278  131  179   0.6360970
7    75   95  283 284  459  262 1994  111   78  123   0.4702444
8   130  164  188 239  377  302  187 1669  120  349   0.5519463
9   395  284   52  96   64  128   50   56 2337  296   0.3781267
10  142  727   68 115   45  121  105  153  326 1972   0.4774775
```



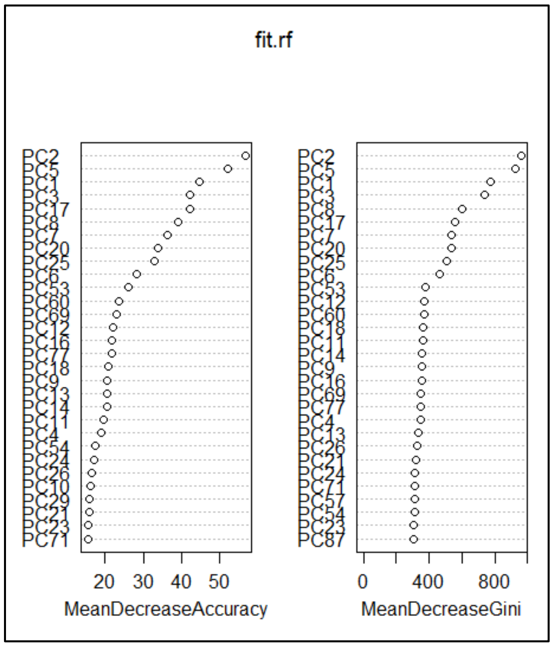
```
> fitrf.pred <-predict(fit.rf, newdata=test.data)
> Accuracy <- mean(fitrf.pred == test.y)
> Accuracy
[1] 0.4816
```

The error of the prediction is calculated in terms of Out-Of-Bag error. It is 54.45%. Hence the Out-Of-Bag accuracy is 45.55%. Testing accuracy is 48.16%. Another important observation can be drawn here. OOB error and Test error are comparable and are approximately equal.

The accuracy of the model can always be improved, but the random forests require more straightening out the given data. Across all the trees considered in the random forest, the important PCs are identified from the following graph of Importance. When PC2 and PC5 are excluded from the training sample, the accuracy decreases by an average of 50%. Hence, PC2 and PC5 are the most important PCs to consider.



```
> varImpPlot(fit.rf)
```

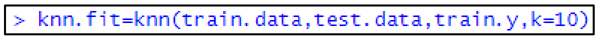


Model 3: K-Nearest Neighbors

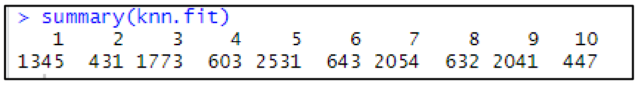
KNN algorithm firstly estimates the conditional probabilities by considering closest K number of neighbors for the given observation x_0. Taking majority of the classes from those K neighbors, conditional probability for x_0 is calculated. Then x_0 is assigned to a class of maximum posterior probability using Bayes' rule.

The KNN model was tuned based the number of K nearest neighbors, the value of k was found using for -loop and it was observed to be 10 for accuracy to be maximum.

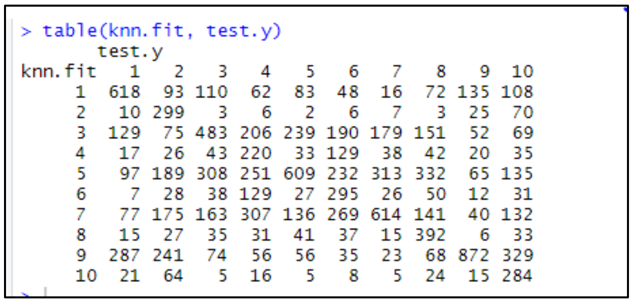
#R-Workspace:



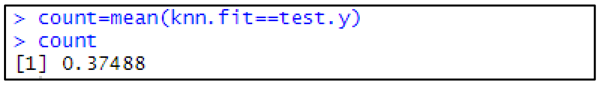
```
> knn.fit=knn(train.data,test.data,train.y,k=10)
```



```
> summary(knn.fit)
   1    2    3    4    5    6    7    8    9   10 
1345  431 1773  603 2531  643 2054  632 2041  447 
```



```
> table(knn.fit, test.y)
       test.y
knn.fit   1   2   3   4   5   6   7   8   9  10
     1  618  93 110  62  83  48  16  72 135 108
     2   10 299   3   6   2   6   7   3  25  70
     3  129  75 483 206 239 190 179 151  52  69
     4   17  26  43 220  33 129  38  42  20  35
     5   97 189 308 251 609 232 313 332  65 135
     6    7  28  38 129  27 295  26  50  12  31
     7   77 175 163 307 136 269 614 141  40 132
     8   15  27  35  31  41  37  15 392   6  33
     9  287 241  74  56  56  35  23  68 872 329
     10  21  64   5  16   5   8   5  24  15 284
```

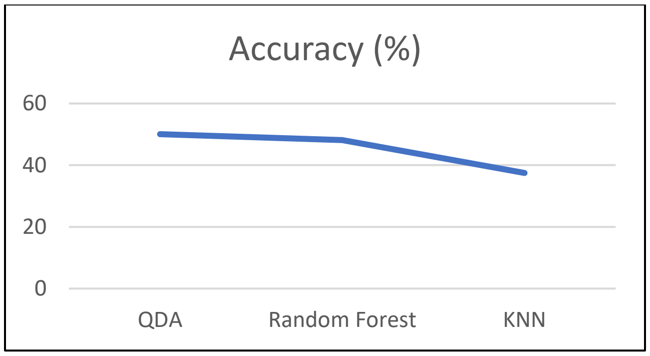


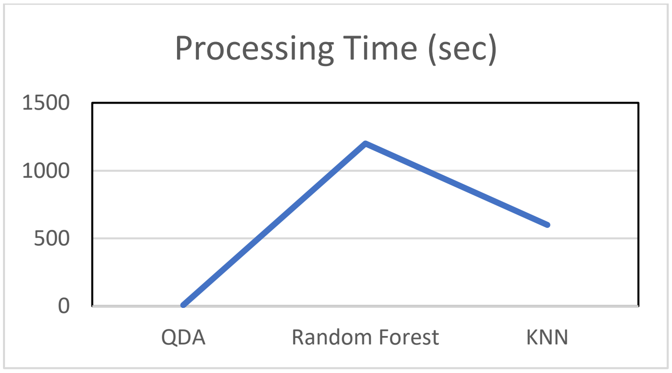
```
> count=mean(knn.fit==test.y)
> count
[1] 0.37488
```

Accuracy of object detection using KNN is 37.48%. KNN performs well when responses are sampled from complicated function. However, the PCs which are used for this analysis are standardized and hence follow normal distribution. Hence, it is observed that accuracy of the KNN with K=10 is not as good as QDA.

Comparison between QDA, Random Forest and KNN

Accuracy and processing time are the two parameters are most important while selecting the best model.





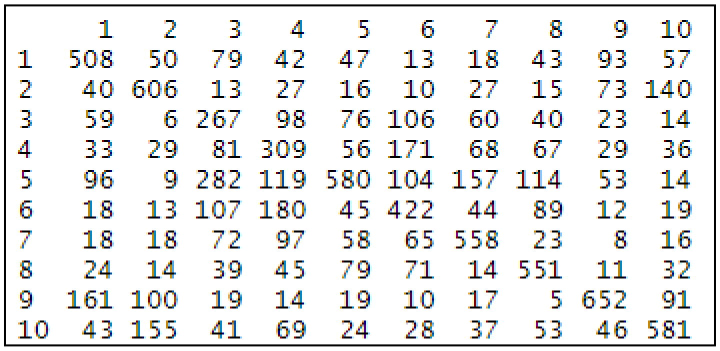
Here, from the above two graphs it is observed that the testing accuracy and the computation time is optimal for the QDA model and hence, we select that as our best model. The accuracy of random forests can be increased a bit by increase in number of trees but, the increase in accuracy in not very significant compared to the computation time which will be significant. And, as of the current RF model, the computation time is high. The KNN model has a lower computation time than RF but at an expense of decrease in accuracy.

So, concluding, for image classification problem on this given data set, we select the QDA model as it has higher interpretability than compared to other two models, lower computation time and higher accuracy.

Improvements

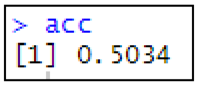
1. **Accuracy of existing program:**
   At first, the finalized QDA model was run on the new test data. Some minor but necessary additions to the program were done, as:
   a) Creating a data-frame for the new data set
   b) Replacing the 0.25 * training data by the actual test data

   After running the program, following results were obtained:
   Confusion Matrix:



```
     1   2   3   4   5   6   7   8   9  10
1  508  50  79  42  47  13  18  43  93  57
2   40 606  13  27  16  10  27  15  73 140
3   59   6 267  98  76 106  60  40  23  14
4   33  29  81 309  56 171  68  67  29  36
5   96   9 282 119 580 104 157 114  53  14
6   18  13 107 180  45 422  44  89  12  19
7   18  18  72  97  58  65 558  23   8  16
8   24  14  39  45  79  71  14 551  11  32
9  161 100  19  14  19  10  17   5 652  91
10  43 155  41  69  24  28  37  53  46 581
```

   Accuracy obtained:



```
> acc
[1] 0.5034
```

   The accuracy obtained on the test set (50.34%) is therefore comparable to the accuracy obtained in validation set (50.06%).

2. **Ways to improve accuracy:**
   To improve the accuracy, following ways were tested:
   a) Changing the split between training and test data:
   The train-test split used while training the model was 0.75 – 0.25. Several other split percentages could be checked for minimal test error (in-training). For all the splits, PCA was done on the training data and PCs explaining 90% of the total variance were used in further analysis. The same PCA was run on the test data.

Split Percentage	Accuracy
50% - 50%	49.09%
60% - 40%	49.37%
75% - 25%	50.34%
80% - 20%	49.98%

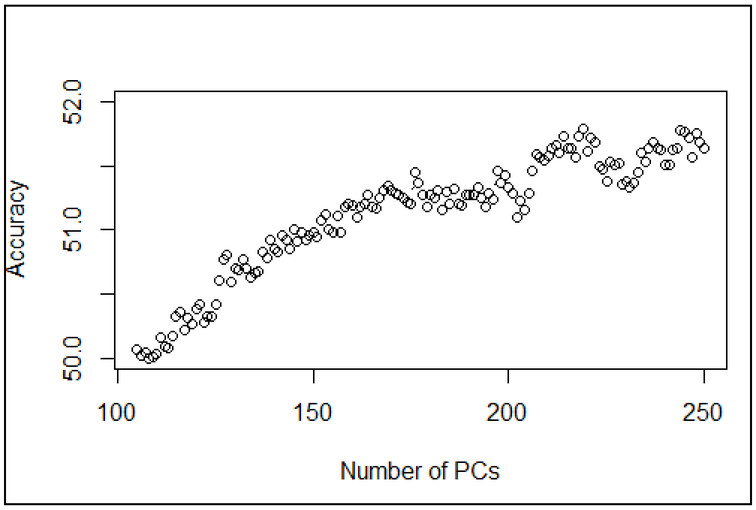
   The maximum accuracy split (75%-25%) was selected.

   b) Using Transformations of the input data:
   To check if the relation between the predictors and the response is highly non-linear, different transforms of the predictors were tested. For all the transforms, PCA was done on the training data and PCs explaining 90% of the total variance were used in further analysis. The same PCA was run on the test data.

Predictor Transform	Accuracy
$\sqrt{X}$	49.92%
X	50.34%
X^2	44.96%
X^3	40.31%

The maximum accuracy transform (X) was selected.

c) Changing the number of PCs considered:
Keeping the above-mentioned parameters constant, the number of principal components considered was varies. Due to high computation time required for the program, it was broken into splits and best accuracy for each split was checked.



An increasing trend is observed in accuracy as the number of Principal components increase, reaches maximum accuracy Number of PCs = 219 and starts decreasing again. This being the test error, the trend will not follow an increasing trend and will keep decreasing, opposite to the test error rate graph. The point with maximum accuracy in this trend can therefore be selected and used on the actual test data.

d) The accuracy can be further improved by changing the number of PCs considered for different split percentages. In that case, a combined effect of changing the split percentage and the number of principal components considered could make a difference and an optimal point can be achieved with higher accuracy. The same point can then be tested on the final test data.

3. Results:

After running the same code on final test data, the accuracy is obtained as **51.74%.**
The total increase in accuracy after improvements is **1.4%.**

References

1. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning: With Applications in R. Corrected edition New York: Springer, 2013.
2. https://www.analyticsvidhya.com/blog/2016/07/making-predictions-test-data-principal-component-analysis/