Assignment 11

ISEN 613

1. Pruning, Random Forest and Boosting models

Code:

```
Train.data=read.csv("CancerData.csv",header=T)
Test.data=read.csv("CancerHoldoutData.csv",header=T)

#Analyze for missing/incorrect data
summary(Train.data)

##  TARGET_deathRate incidenceRate      medIncome      povertyPercent
##  Min.   : 59.7    Min.   : 201.3   Min.   : 22640   Min.   : 3.20
##  1st Qu.:161.0    1st Qu.: 420.2   1st Qu.: 38842   1st Qu.:12.10
##  Median :178.2    Median : 453.5   Median : 45224   Median :15.80
##  Mean   :178.6    Mean   : 447.9   Mean   : 47188   Mean   :16.85
##  3rd Qu.:195.2    3rd Qu.: 480.8   3rd Qu.: 52702   3rd Qu.:20.40
##  Max.   :362.8    Max.   :1206.9   Max.   :125635   Max.   :47.40
##
##    MedianAge      MedianAgeMale   MedianAgeFemale
##  Min.   : 22.30   Min.   :22.40   Min.   :22.30
##  1st Qu.: 37.80   1st Qu.:36.40   1st Qu.:39.10
##  Median : 41.00   Median :39.50   Median :42.40
##  Mean   : 44.73   Mean   :39.56   Mean   :42.15
##  3rd Qu.: 44.00   3rd Qu.:42.60   3rd Qu.:45.38
##  Max.   :624.00   Max.   :64.70   Max.   :65.70
##
##                            Geography    AvgHouseholdSize PercentMarried
##  Abbeville County, South Carolina:   1   Min.   :0.0222   Min.   :23.10
##  Acadia Parish, Louisiana        :   1   1st Qu.:2.3700   1st Qu.:47.90
##  Accomack County, Virginia       :   1   Median :2.5000   Median :52.40
##  Ada County, Idaho               :   1   Mean   :2.4896   Mean   :51.78
##  Adair County, Kentucky          :   1   3rd Qu.:2.6400   3rd Qu.:56.30
##  Adair County, Missouri          :   1   Max.   :3.9700   Max.   :72.50
##  (Other)                         :2584
##   PctNoHS18_24     PctHS18_24    PctSomeCol18_24 PctBachDeg18_24
##  Min.   : 0.00   Min.   : 0.00   Min.   : 7.10   Min.   : 0.00
##  1st Qu.:12.70   1st Qu.:29.20   1st Qu.:33.60   1st Qu.: 3.10
##  Median :17.10   Median :34.80   Median :40.30   Median : 5.30
##  Mean   :18.24   Mean   :34.96   Mean   :40.87   Mean   : 6.16
##  3rd Qu.:22.80   3rd Qu.:40.67   3rd Qu.:46.20   3rd Qu.: 8.20
##  Max.   :64.10   Max.   :72.50   Max.   :79.00   Max.   :51.80
##                                  NA's   :1938
##  PctPrivateCoverage PctPublicCoverage PctPublicCoverageAlone    PctWhite
```

```
##  Min.   :22.30     Min.   :11.20     Min.   : 2.60          Min.   : 10.2
0
##  1st Qu.:57.40     1st Qu.:30.80     1st Qu.:14.90          1st Qu.: 77.0
4
##  Median :65.20     Median :36.20     Median :18.70          Median : 89.9
9
##  Mean   :64.42     Mean   :36.18     Mean   :19.19          Mean   : 83.5
7
##  3rd Qu.:72.20     3rd Qu.:41.50     3rd Qu.:23.00          3rd Qu.: 95.3
6
##  Max.   :92.30     Max.   :65.10     Max.   :46.60          Max.   :100.0
0
##
##     PctBlack          PctAsian         PctOtherRace     PctMarriedHousehold
s
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   :22.99
##  1st Qu.: 0.6321   1st Qu.: 0.2556   1st Qu.: 0.2899   1st Qu.:47.83
##  Median : 2.2692   Median : 0.5507   Median : 0.8330   Median :51.71
##  Mean   : 9.0979   Mean   : 1.2690   Mean   : 2.0311   Mean   :51.25
##  3rd Qu.:10.3528   3rd Qu.: 1.2230   3rd Qu.: 2.1823   3rd Qu.:55.33
##  Max.   :85.9478   Max.   :42.6194   Max.   :41.9303   Max.   :78.08
##
```

```r
summary(Test.data)
```

```
##  TARGET_deathRate incidenceRate      medIncome      povertyPercent
##  Min.   :106.1    Min.   : 254.7   Min.   : 25807   Min.   : 3.90
##  1st Qu.:162.0    1st Qu.: 421.8   1st Qu.: 39017   1st Qu.:12.40
##  Median :177.9    Median : 453.5   Median : 45168   Median :16.30
##  Mean   :179.1    Mean   : 450.3   Mean   : 46358   Mean   :17.02
##  3rd Qu.:195.2    3rd Qu.: 482.4   3rd Qu.: 51911   3rd Qu.:20.60
##  Max.   :270.4    Max.   :1014.2   Max.   :122641   Max.   :40.60
##
##    MedianAge       MedianAgeMale   MedianAgeFemale
##  Min.   : 23.30   Min.   :23.00   Min.   :23.60
##  1st Qu.: 37.60   1st Qu.:36.30   1st Qu.:39.20
##  Median : 40.90   Median :39.70   Median :42.30
##  Mean   : 48.34   Mean   :39.62   Mean   :42.13
##  3rd Qu.: 44.00   3rd Qu.:42.30   3rd Qu.:45.20
##  Max.   :619.20   Max.   :58.60   Max.   :58.00
##
##                       Geography   AvgHouseholdSize PercentMarried
##  Adair County, Iowa         :  1   Min.   :0.0221   Min.   :26.20
##  Adair County, Oklahoma     :  1   1st Qu.:2.3600   1st Qu.:47.40
##  Adams County, Colorado     :  1   Median :2.4900   Median :52.60
##  Adams County, Indiana      :  1   Mean   :2.4235   Mean   :51.77
##  Adams County, Mississippi :  1   3rd Qu.:2.6200   3rd Qu.:56.70
##  Adams County, Pennsylvania:  1   Max.   :3.9700   Max.   :68.00
##  (Other)                    :451
##   PctNoHS18_24     PctHS18_24    PctSomeCol18_24 PctBachDeg18_24
```

```
##  Min.   : 1.50   Min.   :10.00   Min.   : 9.60   Min.   : 0.000  
##  1st Qu.:13.30   1st Qu.:29.20   1st Qu.:34.73   1st Qu.: 3.200  
##  Median :17.40   Median :34.50   Median :41.25   Median : 5.600  
##  Mean   :18.13   Mean   :35.25   Mean   :41.58   Mean   : 6.148  
##  3rd Qu.:22.00   3rd Qu.:40.80   3rd Qu.:47.25   3rd Qu.: 8.100  
##  Max.   :59.70   Max.   :72.10   Max.   :78.30   Max.   :28.500  
##                                  NA's   :347                     
##  PctPrivateCoverage PctPublicCoverage PctPublicCoverageAlone    PctWhite    
##  Min.   :25.0       Min.   :11.80     Min.   : 4.60          Min.   :11.01  
##  1st Qu.:56.6       1st Qu.:31.40     1st Qu.:14.80          1st Qu.:78.32  
##  Median :64.0       Median :37.00     Median :19.60          Median :90.32  
##  Mean   :64.0       Mean   :36.66     Mean   :19.55          Mean   :84.05  
##  3rd Qu.:71.7       3rd Qu.:41.80     3rd Qu.:23.60          3rd Qu.:95.66  
##  Max.   :86.9       Max.   :57.50     Max.   :39.70          Max.   :99.69  
##                                                                             
##     PctBlack          PctAsian         PctOtherRace     PctMarriedHousehold
s
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   :24.43      
##  1st Qu.: 0.5946   1st Qu.: 0.2419   1st Qu.: 0.3345   1st Qu.:47.10      
##  Median : 2.2221   Median : 0.5377   Median : 0.7860   Median :51.45      
##  Mean   : 9.1652   Mean   : 1.1689   Mean   : 1.7138   Mean   :51.19      
##  3rd Qu.:10.7674   3rd Qu.: 1.1962   3rd Qu.: 2.0944   3rd Qu.:55.75      
##  Max.   :80.6600   Max.   :33.7609   Max.   :22.4644   Max.   :68.28      
## 

#Removing Geography data since trees cannot be grown with more than 30 factor
s
Train.data$Geography=NULL
Test.data$Geography=NULL

#Correcting median age values for training and holdout data
i=1
for (i in 1:2590)
{
  if (Train.data$MedianAge[i]>130)
  {
    Train.data$MedianAge[i]=(Train.data$MedianAgeMale[i]+Train.data$MedianAge
Female[i])/2
  }
}

i=1
for (i in 1:457)
{
  if (Test.data$MedianAge[i]>130)
  {
    Test.data$MedianAge[i]=(Test.data$MedianAgeMale[i]+Test.data$MedianAgeFem
ale[i])/2
  }
}
```

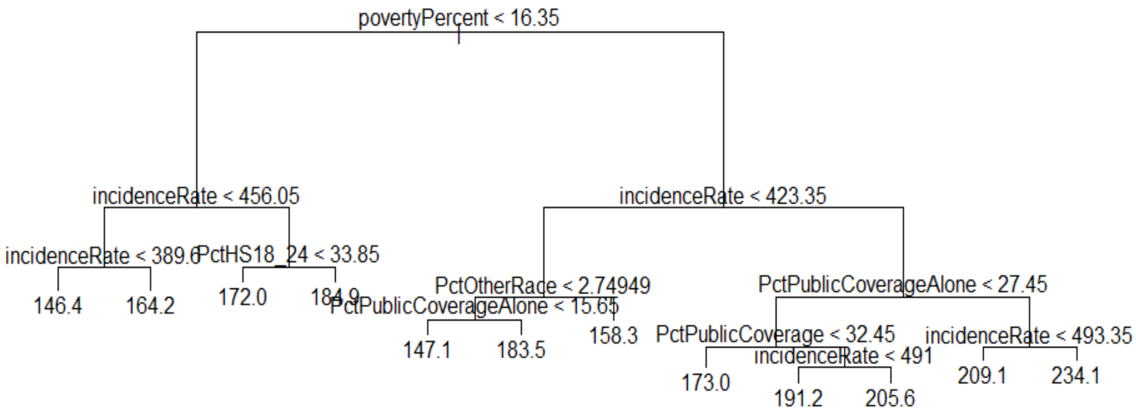
```
#Filling values for PctSomeCol 18_24
Train.data$PctSomeCol18_24=100-(Train.data$PctBachDeg18_24+Train.data$PctHS18
_24+Train.data$PctNoHS18_24)
Test.data$PctSomeCol18_24=100-(Test.data$PctBachDeg18_24+Test.data$PctHS18_24
+Test.data$PctNoHS18_24)

test.rate=Test.data$TARGET_deathRate
```

Explanation:

Data summary shows that there is are incorrect values in the MedianAge column (age cannot be >130 years). These values were corrected by taking average of MedianAgeMale and MedianAgeFemale. Geography data was removed since decision tree cannot take predictors as input that have more than 30 factors. Finally, the missing values in PctSomCol 18_24 were calculated by the other education data. The sum of PctSomCol 18_24, PctHS18_24, PctNoHS18_24 and PctBachDeg18_24 is 100. So values for PctSomCol 18_24 were calculated by subtracting the remaining ones from 100.

Decision Tree Code:

```
#Growing a tree------------------------------------------------
library(tree)

tree.cancer=tree(TARGET_deathRate~.,data=Train.data)
summary(tree.cancer)

##
## Regression tree:
## tree(formula = TARGET_deathRate ~ ., data = Train.data)
## Variables actually used in tree construction:
## [1] "povertyPercent"         "incidenceRate"          "PctHS18_24"
## [4] "PctOtherRace"           "PctPublicCoverageAlone" "PctPublicCoverage"
## Number of terminal nodes:  12
## Residual mean deviance:  454.5 = 1172000 / 2578
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -92.0000 -13.0900  -0.5493   0.0000  12.6700 157.2000

#Deviance:454.5

plot(tree.cancer)
text(tree.cancer,pretty=0)
```

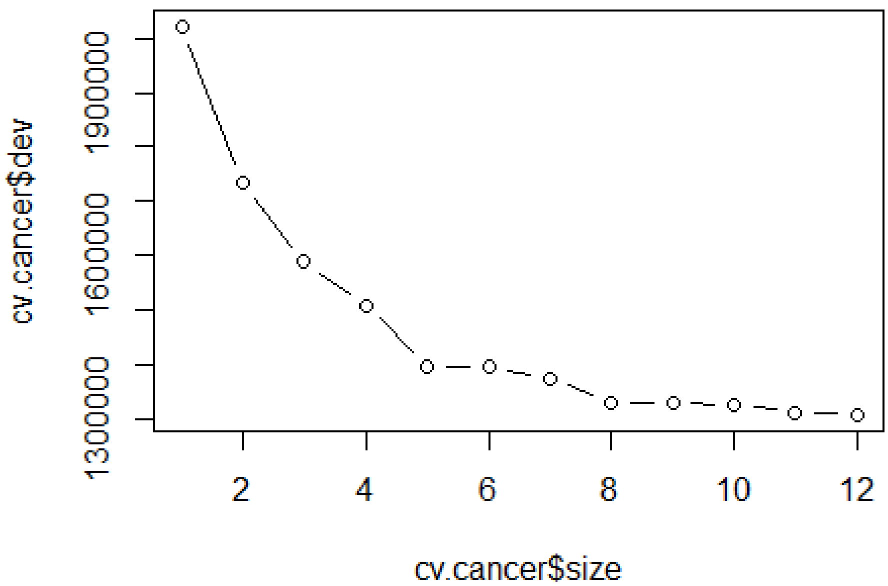


Explanation:

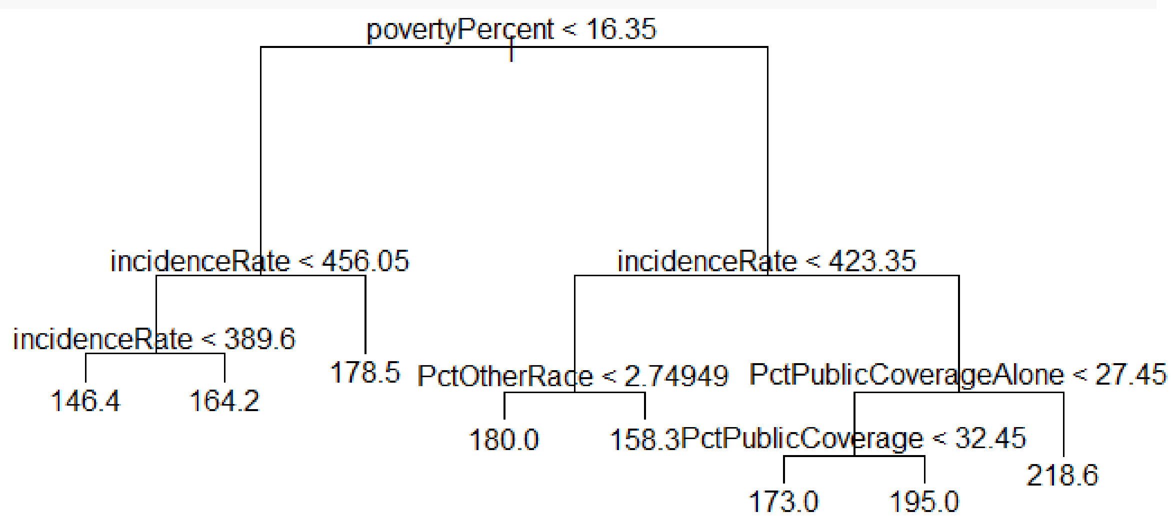
From the tree summary, it can be seen that only 6 out of 21 variables were used to grow the tree. The tree has 12 terminal nodes. povertyPercent was the most important predictor, followed by incidence rate. Some other factors like insurance coverage (PctPublicCoverage alone), PctOtherRace and PctHS18_24 were considered somewhat important as well. The training MSE (Residual mean deviance) was obtained to be 454.5.

Pruning code:

```
#Pruning
set.seed(3)
cv.cancer=cv.tree(tree.cancer,K=10)
cv.cancer

## $size
##  [1] 12 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 1306660 1311773 1325931 1328094 1328781 1374357 1395280 1395280 15091
45
## [10] 1588651 1733820 2021865
##
## $k
##  [1]      -Inf  22472.07  24428.53  25149.01  25624.69  34476.97  37230.64
##  [8]  37276.70  85242.13 102307.50 153492.11 301296.97
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"

plot(cv.cancer$size,cv.cancer$dev,type="b")
```




```
#Deviance is minimum for size=12. However, there is not much change in devian
ce after size=8
#So size=8 is taken for pruning
prune.cancer=prune.tree(tree.cancer,best=8)
plot(prune.cancer)
text(prune.cancer,pretty=0)
```




```
summary(prune.cancer)

##
## Regression tree:
## snip.tree(tree = tree.cancer, nodes = c(15L, 5L, 12L, 29L))
```

```
## Variables actually used in tree construction:
## [1] "povertyPercent"         "incidenceRate"          "PctOtherRace"
## [4] "PctPublicCoverageAlone" "PctPublicCoverage"
## Number of terminal nodes:  8
## Residual mean deviance:  491.7 = 1269000 / 2582
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -92.0000 -13.9100  -0.7271   0.0000  13.3500 167.8000

#MSE: 491.7 - slight increase
```

Explanation:

Tree pruning was done with 10-fold Cross Validation. From the CV, it can be seen that the deviance is minimum for a tree of size 12, which is the original tree size. From the deviance vs. size graph, it is seen that the deviance becomes nearly constant after size=8. Therefore, for pruning, 8 nodes were considered. For pruned tree, the training MSE obtained was 491.7, which is slightly higher than the unpruned tree. This was expected, since the deviance for size 12 tree was the minimum. However, pruning increases the interpretability of the tree, along with decreasing its tendency to overfit.

Method	Training MSE
Unpruned tree	454.5
Pruned tree	491.7

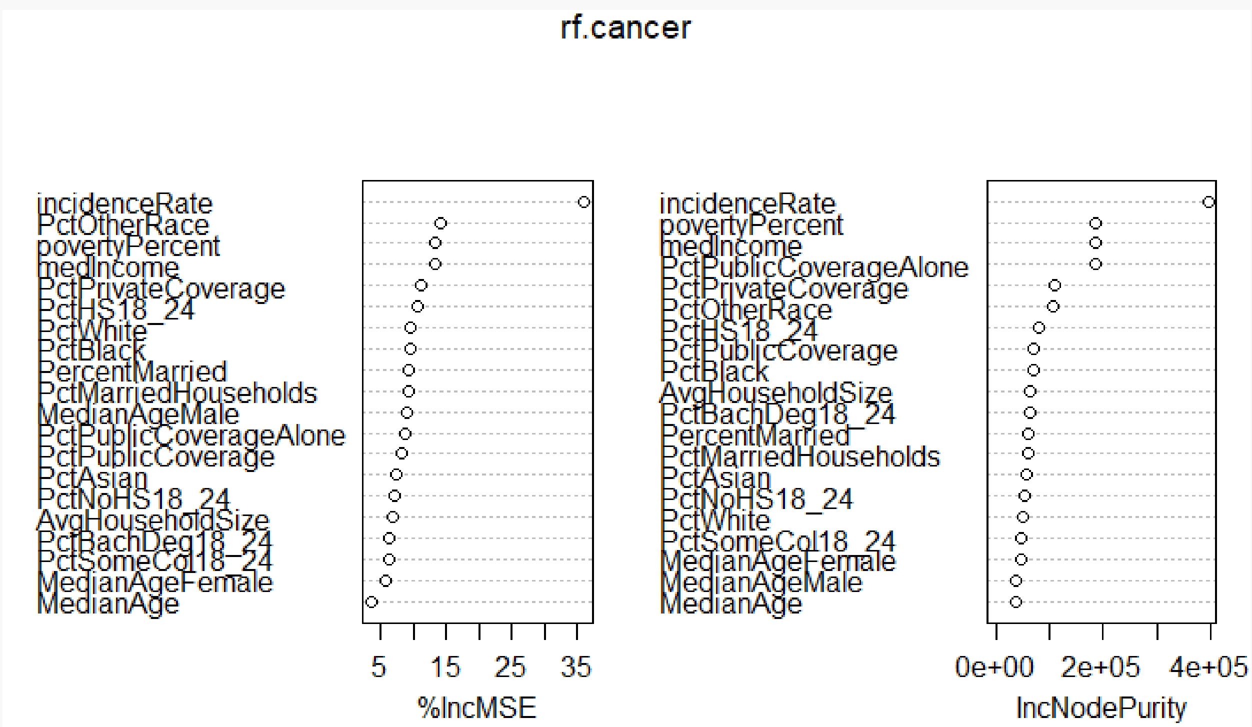
Code for Random Forest:

```
#RandomForest---------------------------------------------------
library(randomForest)

set.seed(1)
rf.cancer=randomForest(TARGET_deathRate~.,data=Train.data,mtry=7,ntree=140,im
portance=TRUE)
rf.cancer

##
## Call:
##  randomForest(formula = TARGET_deathRate ~ ., data = Train.data,      mtry
= 7, ntree = 140, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 140
## No. of variables tried at each split: 7
##
##           Mean of squared residuals: 395.3129
##                     % Var explained: 49.33

#MSE:395.3129
```

```r
#Important variables
importance(rf.cancer)
```

```
##                          %IncMSE IncNodePurity
## incidenceRate          35.884416     371518.78
## medIncome              15.964897     184124.77
## povertyPercent         14.067882     157451.47
## MedianAge               8.903654      44433.81
## MedianAgeMale           8.998722      45203.34
## MedianAgeFemale        10.393158      53111.15
## AvgHouseholdSize        7.272348      65507.31
## PercentMarried          9.186026      68078.35
## PctNoHS18_24            6.985515      51021.68
## PctHS18_24             11.432895      81819.22
## PctSomeCol18_24         6.791370      49141.59
## PctBachDeg18_24         8.939176      63515.29
## PctPrivateCoverage     15.129331     135645.28
## PctPublicCoverage       9.923905      76834.67
## PctPublicCoverageAlone 11.850455     176154.29
## PctWhite                8.305869      51544.59
## PctBlack               10.291992      70707.43
## PctAsian               10.130447      58283.25
## PctOtherRace           19.225398     107487.27
## PctMarriedHouseholds   10.802135      61895.80
```

```r
varImpPlot(rf.cancer)
```




```r
#Hyperparameter tuning
library(caret)
```

```r
caretGrid=expand.grid(mtry=5:12)

metric="RMSE"
trainControl=trainControl(method="cv",number=10)

set.seed(1)
i=100
for (i in seq(from=100,to=500,by=200))
{
  rf.caret=train(TARGET_deathRate~.,data=Train.data,
                 method="rf",trControl=trainControl,
                 verbose=FALSE,metric=metric,
                 tuneGrid=caretGrid,ntree=i)
  print(rf.caret)
}
```

```
## Random Forest
##
## 2590 samples
##   20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2332, 2331, 2332, 2330, 2330, 2331, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##    5    19.91423  0.4983666  14.69948
##    6    19.93097  0.4964389  14.70444
##    7    19.97488  0.4930750  14.73469
##    8    19.72091  0.5067305  14.59599
##    9    19.89085  0.4967043  14.69662
##   10    19.91840  0.4952377  14.72283
##   11    19.89895  0.4962201  14.68007
##   12    19.88138  0.4968491  14.68302
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 8.
## Random Forest
##
## 2590 samples
##   20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2332, 2331, 2330, 2331, 2330, 2332, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##    5    19.80430  0.5027482  14.65646
```

```
##    6    19.77310  0.5034082  14.60044
##    7    19.78130  0.5020135  14.60415
##    8    19.77053  0.5018676  14.63979
##    9    19.72287  0.5037393  14.58192
##   10    19.79645  0.4994972  14.61645
##   11    19.78168  0.5001087  14.61602
##   12    19.75478  0.5013361  14.62948
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 9.
## Random Forest
##
## 2590 samples
##   20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2332, 2330, 2333, 2331, 2331, 2331, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##    5    19.76883  0.5060964  14.60200
##    6    19.76403  0.5054037  14.62678
##    7    19.76633  0.5044493  14.61893
##    8    19.78045  0.5032309  14.61723
##    9    19.75724  0.5042610  14.61542
##   10    19.76671  0.5031056  14.63061
##   11    19.77119  0.5028916  14.62774
##   12    19.79496  0.5014372  14.65535
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 9.

#ntree=100, RMSE=19.721 mtry=8
#ntree=300, RMES:19.722 mtry=9
#ntree=500, RMSE:19.757 mtry=9

#Rebuilding model using ntree=100

rf.caret=train(TARGET_deathRate~.,data=Train.data,
               method="rf",trControl=trainControl,
               verbose=FALSE,metric=metric,
               tuneGrid=caretGrid,ntree=100)
```

Explanation:

Random forest was used to fit the training data. Initially, 140 trees and 7 variables were tried at each split, to get an estimate of error. 10-fold CV was used. Then, caret package was used to tune the value of mtry from 5 to 12 using a grid. Using caret for varying the number of trees was taking a lot of time. Due to limited computing resources, 3 values of ntrees were checked using a for loop. The results are shown in the table below:

Method		**Training MSE**	
Unpruned tree		454.5	
Pruned tree		491.7	
Random Forest			
Ntree	**Mtry**	**RMSE**	**MSE**
100	5	19.914	396.5674
	6	19.931	397.2448
	7	19.975	399.0006
	8	**19.720**	**388.8784**
	9	19.891	395.6519
	10	19.918	396.7267
	11	19.899	395.9702
	12	19.881	395.2542
300	5	19.804	392.1984
	6	19.773	390.9715
	7	19.781	391.288
	8	19.770	390.8529
	9	**19.723**	**388.9967**
	10	19.796	391.8816
	11	19.782	391.3275
	12	19.755	390.26
500	5	19.769	390.8134
	6	19.764	390.6157
	7	19.766	390.6948
	8	19.780	391.2484
	9	**19.757**	**390.339**
	10	19.767	390.7343
	11	19.771	390.8924
	12	19.795	391.842

It can be seen that minimum MSE is obtained for ntree=100 and mtry=8. This is close to the default value of mtry which is p/3 (in this case it is 7). Since there is not much change in MSE values for 100,300 and 500 trees, it indicates that saturation has been reached and the error will not decrease by introducing more trees.

Code for Boosting:

```
#Boosting-----------------------------------------------
library(gbm)

library(caret)
boost.caretGrid=expand.grid(interaction.depth=c(1,3,5),n.trees=(1:20)*50,
                            shrinkage=c(0.001,0.01,0.05,0.1),n.minobsinnode=c
```

```
(10,50,100))
metric="RMSE"

set.seed(1)
gbm.caret=train(TARGET_deathRate~ ., data=Train.data,method="gbm",
                trControl=trainControl, verbose=FALSE,
                tuneGrid=boost.caretGrid, metric=metric)

print(gbm.caret)
```

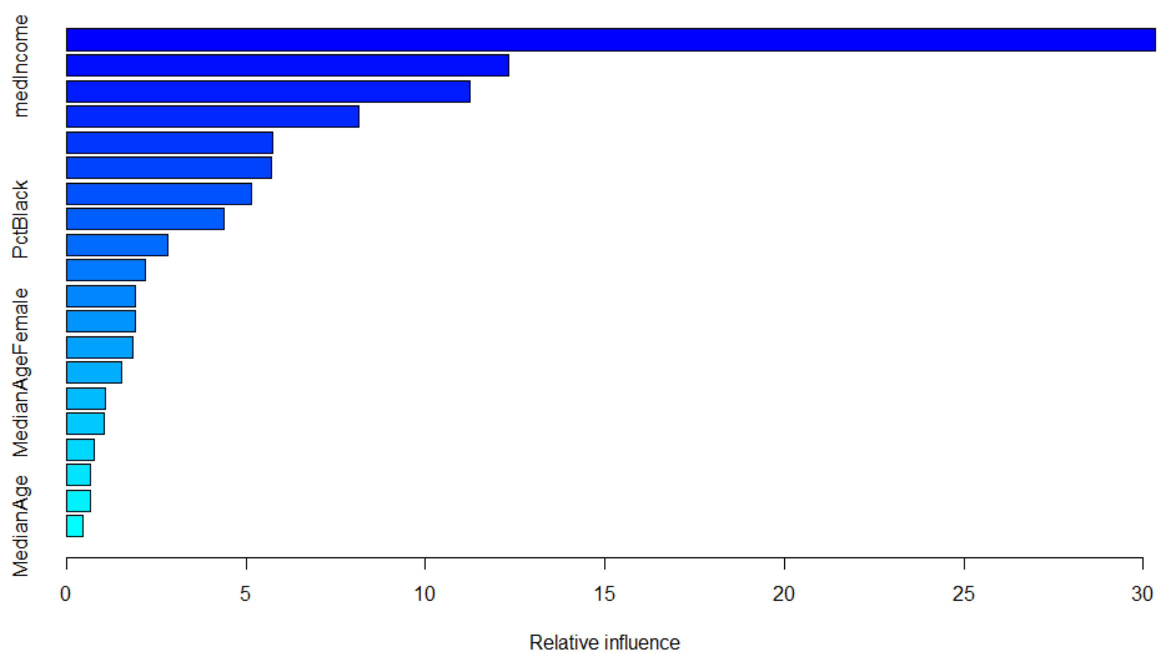
```
Output:
Stochastic Gradient Boosting

2590 samples
  20 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2332, 2331, 2332, 2330, 2330, 2331, ...
Resampling results across tuning parameters:

  shrinkage  interaction.depth  n.minobsinnode  n.trees  RMSE      Rsquared   MAE
  0.001      1                  10                50     27.71060  0.2511115  21.26311
  0.001      1                  10               100     27.52219  0.2666335  21.09806
  0.001      1                  10               150     27.34262  0.2757090  20.94290
  0.001      1                  10               200     27.16996  0.2851749  20.79336
  0.001      1                  10               250     27.00279  0.2936099  20.64875
  0.001      1                  10               300     26.84247  0.2988639  20.50985
  0.001      1                  10               350     26.68842  0.3043864  20.37847
  0.001      1                  10               400     26.53868  0.3104737  20.25075
  0.001      1                  10               450     26.39641  0.3151984  20.12781
  0.001      1                  10               500     26.25796  0.3198637  20.01084
  0.001      1                  10               550     26.12479  0.3239404  19.89995
  0.001      1                  10               600     25.99688  0.3279237  19.79459
  0.001      1                  10               650     25.87377  0.3314829  19.69177
  0.001      1                  10               700     25.75370  0.3353987  19.59369
  0.001      1                  10               750     25.63618  0.3385009  19.49520
  0.001      1                  10               800     25.52295  0.3421860  19.40182
  0.001      1                  10               850     25.41257  0.3453567  19.30963
  0.001      1                  10               900     25.30639  0.3481949  19.22118
  0.001      1                  10               950     25.20617  0.3503177  19.13802
  0.001      1                  10              1000     25.10577  0.3532316  19.05434
  0.001      1                  50                50     27.70859  0.2458375  21.26059
  0.001      1                  50               100     27.52408  0.2618135  21.09796
  0.001      1                  50               150     27.34338  0.2752894  20.94293
  0.001      1                  50               200     27.16920  0.2849950  20.79205
  0.001      1                  50               250     27.00202  0.2960837  20.64983
  0.001      1                  50               300     26.84181  0.3006080  20.51120
  0.001      1                  50               350     26.68876  0.3061622  20.37933
  0.001      1                  50               400     26.54108  0.3117302  20.25429
  0.001      1                  50               450     26.39840  0.3166058  20.13288
  0.001      1                  50               500     26.25790  0.3210503  20.01278
  0.001      1                  50               550     26.12319  0.3253758  19.89963
  0.001      1                  50               600     25.99671  0.3286083  19.79427
```

0.001	1	50	650	25.87156	0.3324770	19.69084
0.001	1	50	700	25.75157	0.3362286	19.59186
0.001	1	50	750	25.63508	0.3394485	19.49607
0.001	1	50	800	25.52350	0.3424225	19.40392
0.001	1	50	850	25.41596	0.3448550	19.31483
0.001	1	50	900	25.30982	0.3473032	19.22745
0.001	1	50	950	25.20657	0.3504616	19.14255
0.001	1	50	1000	25.10572	0.3532499	19.05889
0.001	1	100	50	27.70898	0.2495445	21.26153
0.001	1	100	100	27.52089	0.2660928	21.09576
0.001	1	100	150	27.34076	0.2735546	20.93796
0.001	1	100	200	27.16826	0.2853133	20.78923
0.001	1	100	250	27.00026	0.2948502	20.64454
0.001	1	100	300	26.84028	0.2999631	20.50689
0.001	1	100	350	26.68546	0.3062881	20.37497
0.001	1	100	400	26.53598	0.3114200	20.24779
0.001	1	100	450	26.39559	0.3147371	20.12742
0.001	1	100	500	26.25861	0.3193863	20.01214
0.001	1	100	550	26.12391	0.3242092	19.90065
0.001	1	100	600	25.99397	0.3293974	19.79266
0.001	1	100	650	25.86939	0.3333536	19.68984
0.001	1	100	700	25.74924	0.3372193	19.59257
0.001	1	100	750	25.63315	0.3398716	19.49590
0.001	1	100	800	25.52170	0.3423441	19.40298
0.001	1	100	850	25.41351	0.3451464	19.31172
0.001	1	100	900	25.30779	0.3478906	19.22450
0.001	1	100	950	25.20426	0.3503063	19.13788
0.001	1	100	1000	25.10293	0.3535104	19.05473
0.001	3	10	50	27.53836	0.3670177	21.13040
0.001	3	10	100	27.19647	0.3712799	20.84940
0.001	3	10	150	26.87199	0.3744379	20.58162
0.001	3	10	200	26.56720	0.3776036	20.32898
0.001	3	10	250	26.28044	0.3810652	20.09249
0.001	3	10	300	26.00724	0.3842980	19.86424
0.001	3	10	350	25.74855	0.3884682	19.64868
0.001	3	10	400	25.50263	0.3920959	19.44617
0.001	3	10	450	25.26944	0.3957108	19.25416
0.001	3	10	500	25.04814	0.3984437	19.07095
0.001	3	10	550	24.83534	0.4015642	18.89308
0.001	3	10	600	24.63390	0.4044241	18.72809
0.001	3	10	650	24.44388	0.4069065	18.57176
0.001	3	10	700	24.26325	0.4093739	18.42335
0.001	3	10	750	24.08963	0.4120227	18.27925
0.001	3	10	800	23.92501	0.4141939	18.14294
0.001	3	10	850	23.76664	0.4168081	18.01251
0.001	3	10	900	23.61511	0.4193702	17.88716
0.001	3	10	950	23.47249	0.4215820	17.77031
0.001	3	10	1000	23.33622	0.4236161	17.65783
0.001	3	50	50	27.53896	0.3619498	21.12944
0.001	3	50	100	27.19467	0.3683649	20.84467
0.001	3	50	150	26.86996	0.3729769	20.57701
0.001	3	50	200	26.56307	0.3761598	20.32150
0.001	3	50	250	26.27501	0.3799978	20.08166
0.001	3	50	300	26.00221	0.3832714	19.85341
0.001	3	50	350	25.73970	0.3872325	19.63510
0.001	3	50	400	25.49299	0.3904752	19.43155
0.001	3	50	450	25.26202	0.3931999	19.23942

0.001	3	50	500	25.04092	0.3966122	19.05562
0.001	3	50	550	24.83098	0.3994065	18.87959
0.001	3	50	600	24.62948	0.4024274	18.71319
0.001	3	50	650	24.43905	0.4052397	18.55628
0.001	3	50	700	24.25551	0.4079703	18.40458
0.001	3	50	750	24.08046	0.4108745	18.25848
0.001	3	50	800	23.91263	0.4137376	18.12041
0.001	3	50	850	23.75377	0.4163235	17.98778
0.001	3	50	900	23.60102	0.4189260	17.86202
0.001	3	50	950	23.45696	0.4210930	17.74233
0.001	3	50	1000	23.31916	0.4233645	17.63004
0.001	3	100	50	27.53900	0.3542025	21.12924
0.001	3	100	100	27.19360	0.3624446	20.84232
0.001	3	100	150	26.86891	0.3663223	20.57161
0.001	3	100	200	26.56354	0.3704815	20.31658
0.001	3	100	250	26.27443	0.3738675	20.07205
0.001	3	100	300	26.00276	0.3767611	19.84224
0.001	3	100	350	25.74363	0.3800527	19.62477
0.001	3	100	400	25.50011	0.3829051	19.42182
0.001	3	100	450	25.26875	0.3861833	19.22555
0.001	3	100	500	25.04913	0.3888740	19.03848
0.001	3	100	550	24.84105	0.3915965	18.86021
0.001	3	100	600	24.64268	0.3942102	18.69226
0.001	3	100	650	24.45430	0.3971825	18.53210
0.001	3	100	700	24.27490	0.3998456	18.38153
0.001	3	100	750	24.10316	0.4027385	18.23651
0.001	3	100	800	23.93892	0.4056512	18.09780
0.001	3	100	850	23.78465	0.4081432	17.96887
0.001	3	100	900	23.63641	0.4104691	17.84351
0.001	3	100	950	23.49562	0.4127021	17.72483
0.001	3	100	1000	23.35817	0.4153243	17.61241
0.001	5	10	50	27.46819	0.4070481	21.08992
0.001	5	10	100	27.06065	0.4111244	20.76506
0.001	5	10	150	26.67557	0.4145104	20.45799
0.001	5	10	200	26.31290	0.4170842	20.16724
0.001	5	10	250	25.97005	0.4194799	19.88865
0.001	5	10	300	25.64630	0.4221863	19.62160
0.001	5	10	350	25.34424	0.4243907	19.37403
0.001	5	10	400	25.05917	0.4268793	19.13923
0.001	5	10	450	24.79083	0.4289200	18.91804
0.001	5	10	500	24.53460	0.4316586	18.70587
0.001	5	10	550	24.29475	0.4336120	18.50713
0.001	5	10	600	24.07009	0.4358963	18.32190
0.001	5	10	650	23.85548	0.4380551	18.14049
0.001	5	10	700	23.65431	0.4398860	17.97289
0.001	5	10	750	23.46275	0.4418927	17.81370
0.001	5	10	800	23.28364	0.4435642	17.66469
0.001	5	10	850	23.11403	0.4455762	17.52488
0.001	5	10	900	22.95141	0.4476192	17.39095
0.001	5	10	950	22.79805	0.4494971	17.26469
0.001	5	10	1000	22.65289	0.4515244	17.14475
0.001	5	50	50	27.47074	0.4012485	21.08622
0.001	5	50	100	27.06097	0.4073004	20.75836

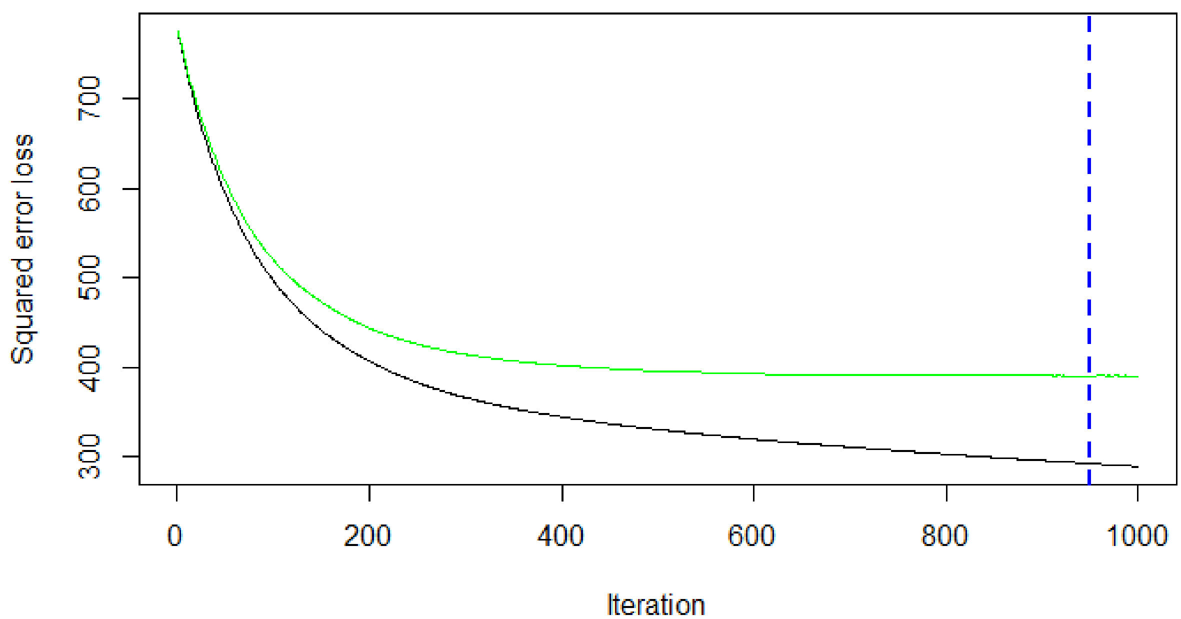
```
 [ reached getOption("max.print") -- omitted 578 rows ]

RMSE was used to select the optimal model using the smallest value.
```

```
The final values used for the model were n.trees = 900, interaction.depth = 5, shrink
age = 0.01
 and n.minobsinnode = 50.
#Above result says that best RMSE was obtained at ntree=900, interaction.dept
h=5
#Shrinkage=0.01 and min obs in node:50

set.seed(1)
boost.check=gbm(TARGET_deathRate~.,data=Train.data,n.trees=1000,interaction.d
epth=5,shrinkage=0.01,
                n.minobsinnode = 50,bag.fraction = 0.75,cv.folds = 10,distrib
ution = "gaussian")
summary(boost.check)
```




```
##                                             var    rel.inf
## incidenceRate                     incidenceRate 30.3424188
## medIncome                             medIncome 12.3191413
## PctPublicCoverageAlone PctPublicCoverageAlone 11.2461146
## povertyPercent                   povertyPercent  8.1585943
## PctOtherRace                       PctOtherRace  5.7510040
## PctPrivateCoverage           PctPrivateCoverage  5.7074812
## PctHS18_24                           PctHS18_24  5.1654407
## PctBlack                               PctBlack  4.3760381
## PctBachDeg18_24                 PctBachDeg18_24  2.8242274
## PctAsian                               PctAsian  2.1949864
## PercentMarried                   PercentMarried  1.9311735
## AvgHouseholdSize               AvgHouseholdSize  1.9006804
```

```
## PctMarriedHouseholds     PctMarriedHouseholds  1.8600662
## MedianAgeFemale                MedianAgeFemale  1.5308070
## PctPublicCoverage            PctPublicCoverage  1.0769080
## PctWhite                              PctWhite  1.0511391
## PctSomeCol18_24                PctSomeCol18_24  0.7726127
## MedianAgeMale                    MedianAgeMale  0.6763654
## PctNoHS18_24                      PctNoHS18_24  0.6599479
## MedianAge                            MedianAge  0.4548531

#Optimum number of trees for CV can be found using gbm.perf function
opt.cv=gbm.perf(boost.check,method="cv")
```




```
print(opt.cv) #Optimum trees is 950

## [1] 950

boost.caretGrid2=expand.grid(interaction.depth=5,n.trees=c(900,950),
                             shrinkage=0.01,n.minobsinnode=c(10,50,100))
metric="RMSE"

set.seed(1)
gbm.caret2=train(TARGET_deathRate~ ., data=Train.data,method="gbm",
                 trControl=trainControl, verbose=FALSE,
                 tuneGrid=boost.caretGrid2, metric=metric)
print(gbm.caret2)
```

```
## Stochastic Gradient Boosting
##
## 2590 samples
##   20 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2332, 2331, 2332, 2330, 2330, 2331, ...
## Resampling results across tuning parameters:
##
##   n.minobsinnode  n.trees  RMSE      Rsquared   MAE
##    10             900      19.72900  0.5020752  14.54982
##    10             950      19.73651  0.5017373  14.55702
##    50             900      19.62980  0.5068303  14.48239
##    50             950      19.61971  0.5073373  14.47706
##   100             900      19.87855  0.4948419  14.64623
##   100             950      19.86743  0.4953527  14.63780
##
## Tuning parameter 'interaction.depth' was held constant at a value of 5
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.01
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 950, interaction.depth
=
##  5, shrinkage = 0.01 and n.minobsinnode = 50.

#Therefore ntrees=950 and RMSE: 19.619
```

Explanation:

Caret function was used to tune parameters for boosting. Three interaction depth values:1,3,5 were considered. More values were not taken since studies show good results for smaller depth values. Shrinkage, or the learning rate was also varied. Ideally, slower learning rates ensure that the prediction is accurate. Values from different ranges were taken to ensure that the step size is not too large or too small. Minimum observation in node was varied from 10, 50 and 100. Number of trees were varied from 50 to 1000 in increments of 50.

Best fit was obtained for 900 trees with shrinkage=0.01, n.minobsinnode=50 and interaction depth=5. n.minobsinnode is the minimum number of observations that should be present on order to create a split. Higher n.minobsinnode value ensures that the model does not overfit the training data.

In gbm package, we can find the optimal number of trees to minimize the MSE. To determine this, a gbm model was fit to the data with 1000 trees. This model was input to gbm.perf( ), which gave a plot of training error and validation error. The black line in graph is training error while the green line is validation error. The dotted line represents when the validation error starts increasing. The corresponding number of trees gives the minimum validation error, which is 950 in this case. New caret model was created with 900 and 950 trees. This model is used in the final prediction.

Method	Training MSE	
Unpruned tree	454.5	
Pruned tree	491.7	
	RMSE	**MSE**
Random Forest Number of trees:100 Number of variables tried at each node: 8	19.720	388.8784
Boosting **Number of trees: 950** **Shrinkage: 0.01** **n.minobsinnode:50** **Interaction.depth:5**	**19.619**	**384.9051**

It is observed that Boosting and Random Forest have a similar performance on training data (Boosting gives a marginally better performance).

Prediction:

The models highlighted above were used for prediction on holdout data

Code:

```
#Applying models on HoldOut data

#Unpruned tree
tree.predict=predict(tree.cancer,Test.data)
mean((tree.predict-test.rate)^2)

## [1] 458.6729

#MSE:458.6729

#Pruned tree: Number of nodes=8
prune.predict=predict(prune.cancer,Test.data)
mean((prune.predict-test.rate)^2)

## [1] 489.1926

#MSE:489.1926

#Random Forest:ntrees=100, mtry=8
rf.predict=predict(rf.caret,Test.data)
mean((rf.predict-test.rate)^2)

## [1] 344.5213

#MSE:343.9803

#Boosting: ntree:950
```

```
boost.predict=predict(gbm.caret2,Test.data)
mean((boost.predict-test.rate)^2)

## [1] 336.4469

#MSE:336.4469
```

Explanation:

Method	Test MSE
Unpruned tree	458.6729
Pruned tree	489.1926
Random Forest	343.9803
Boosting	336.4469

It is seen that among all the models, boosting gives the minimum test MSE. Pruned tree has the highest MSE, followed by unpruned tree. This is because they use a single tree to make predictions whereas Random Forest and Boosting use multiple trees. Therefore, they are less prone to overfitting the training data.

2. Executive summary

The performance of all the models is compared in the table below:

Method	Test MSE
Unpruned tree	458.6729
Pruned tree	489.1926
Random Forest	343.9803
Boosting	**336.4469**
Linear Regression	414.3
KNN	410.183

From the table, it can be seen that Boosting gives the best prediction of cancer mortality rates across different counties. Boosting is based on high bias and low variance, which is gradually improved upon by building trees based on previous trees' residuals. Also, the variance is reduced by averaging across different number of trees. In Random Forest, only variance is lowered, so its accuracy is less compared to Boosting. The performance of random forest and Boosting is better than regression because it does not assume a linear relationship as they are non-parametric.

Important features

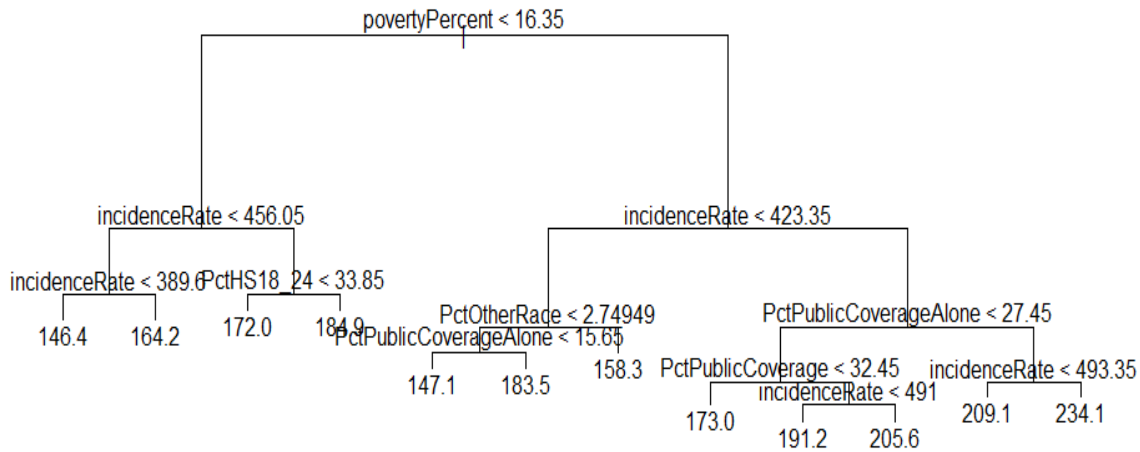
Previous models (**KNN and Linear Regression**) suggested following predictors as important: (Note: Linear Regression and KNN codes are attached in Appendix)

- incidenceRate
- medIncome
- PctHS18_24

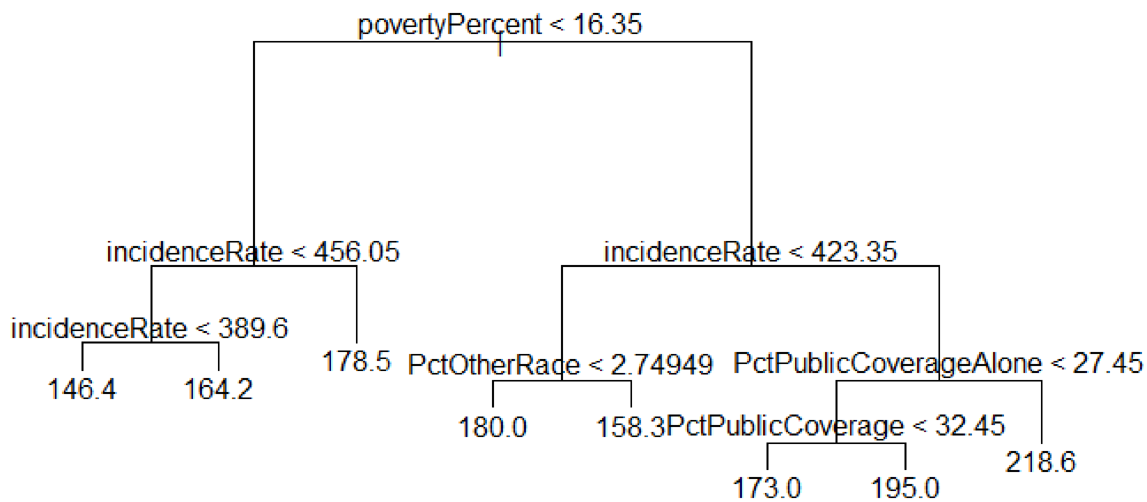
- PctBachDeg18_24
- PctPrivateCoverage
- PctPublicCoverageAlone
- PctOtherRace
- PctMarriedHouseholds

The unpruned and pruned decision trees are shown below:

Unpruned tree

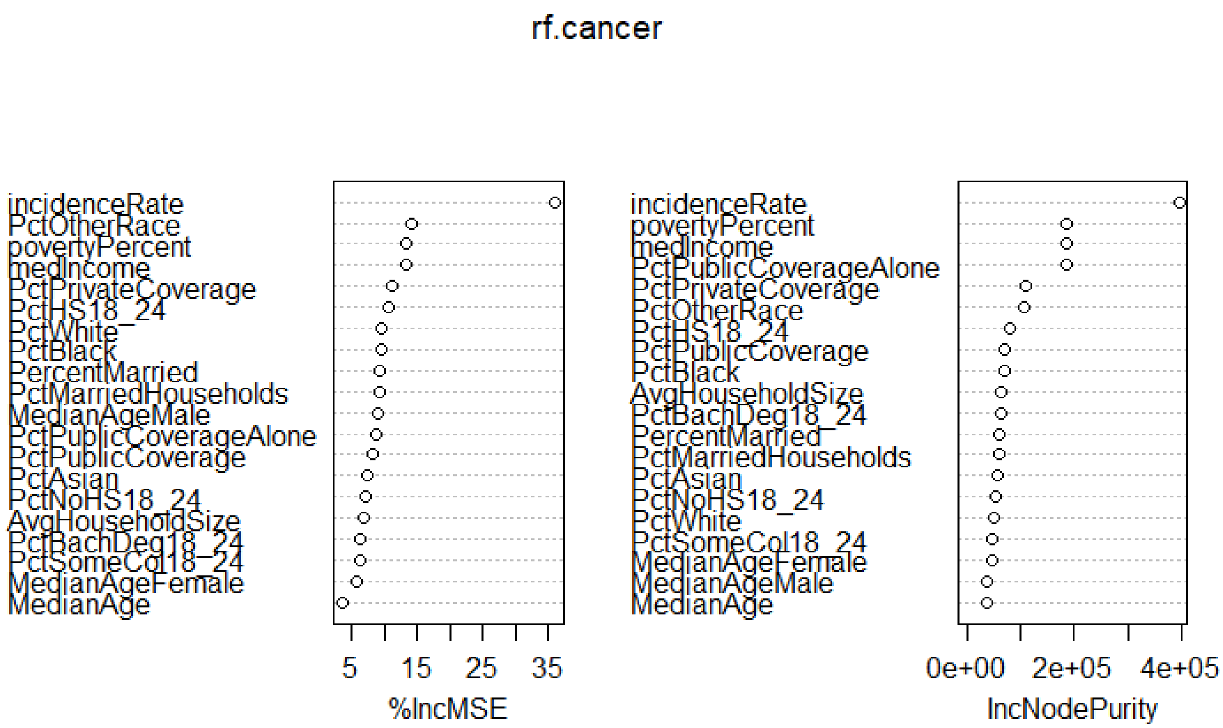


Pruned tree



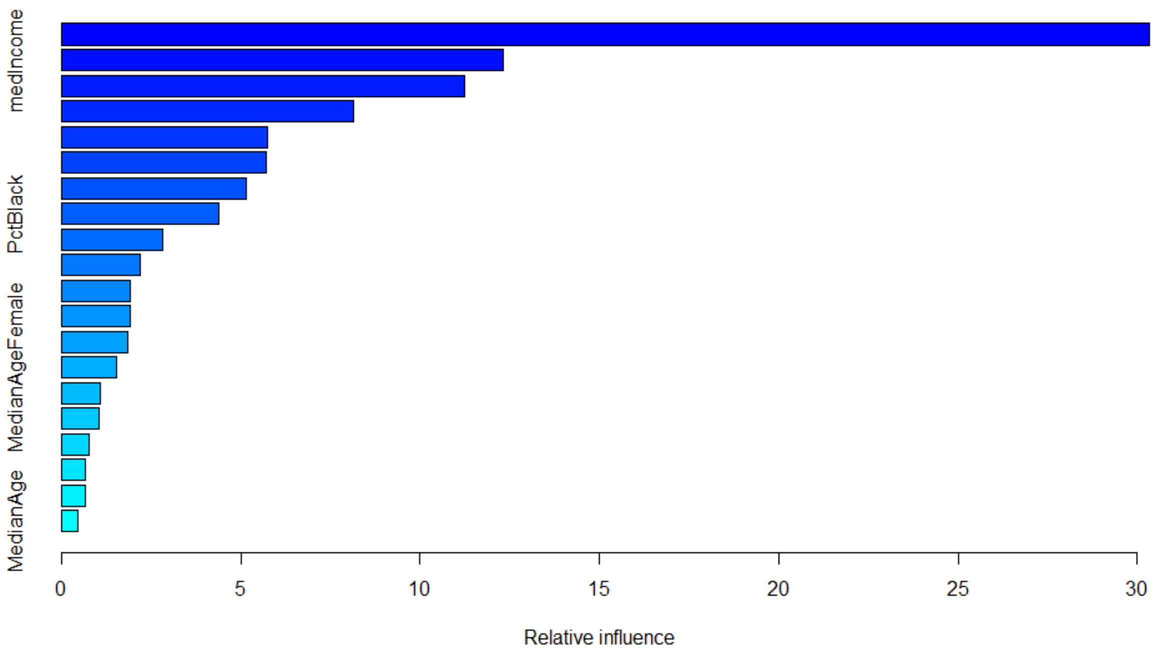
In decision trees, the predictors at the top are important. For the **unpruned tree**, povertyPercent was the most important predictor, followed by incidence rate. Some other factors like insurance coverage (PctPublicCoverage alone), PctOtherRace and PctHS18_24 were considered somewhat important as well. For the **pruned tree**, povertyPercent was the most important predictor, followed by incideneRate and PctPublicCoverage. High povertyPercent was found to increase the mortality rates. As expected, high incidenceRates would also lead to more deaths due to cancer. Also, if larger section of the population had only public health converage (more than 27%), the corresponding county had higher mortality rates (218 compared to 195).

The variable importance plot for **Random Forest** is shown below:



Here only %IncMSE is used to determine importance since IncNodePurity is based on Training data and not on Out Of Bag sample, like IncMSE. It can be seen that permuting the values of incidenceRate had the maximum % increase in training MSE. Therefore, it is the most significant predictor by far. PctOtherRace, povertyPercent and medIncome also had significant impact on the MSE.

The Importance plot for **Boosting** is shown below:




```
##                                              var    rel.inf
## incidenceRate                       incidenceRate 30.3424188
## medIncome                               medIncome 12.3191413
## PctPublicCoverageAlone PctPublicCoverageAlone 11.2461146
## povertyPercent                     povertyPercent  8.1585943
## PctOtherRace                         PctOtherRace  5.7510040
## PctPrivateCoverage             PctPrivateCoverage  5.7074812
## PctHS18_24                             PctHS18_24  5.1654407
## PctBlack                                 PctBlack  4.3760381
## PctBachDeg18_24                   PctBachDeg18_24  2.8242274
## PctAsian                                 PctAsian  2.1949864
## PercentMarried                     PercentMarried  1.9311735
## AvgHouseholdSize                 AvgHouseholdSize  1.9006804
```

From the above data, it is seen that for boosting, incidenceRate (Relative importance 30.34) was the most important predictor. medIncome (12.31) and PctPublicCoverageAlone (11.24) were also important. This was followed by povertyPercent (8.15), PctOtherRace (5.75), PctPrivateCoverage (5.70) and PctHS18_24 (5.16).

Comparison of predictors with KNN and regression model:

From the results of all models, it can be seen that the most important predictors common for all models were incidenceRate and medIncome. povertyPercent was determined important by all tree-based methods but it was not important in KNN and regression. Insurance coverage was also determined important across all models. Counties with higher segment of population relying on public coverage alone were more susceptible to death due to cancer.

PctOtherRace was important in all the models. Interestingly, age was not an important predictor in any of the models.

Unfortunately, the high accuracy in prediction for tree-based models comes at the cost of interpretability. Random Forest and Boosting are far more accurate compared to regression but interpreting the effect of each of the predictors is not possible, like in the case of linear regression. In linear regression, we can tell if the predictor has a positive or negative effect on the output. But it is not draw any such inferences from Random Forest or Boosting.

APPENDIX

Linear Regression code:

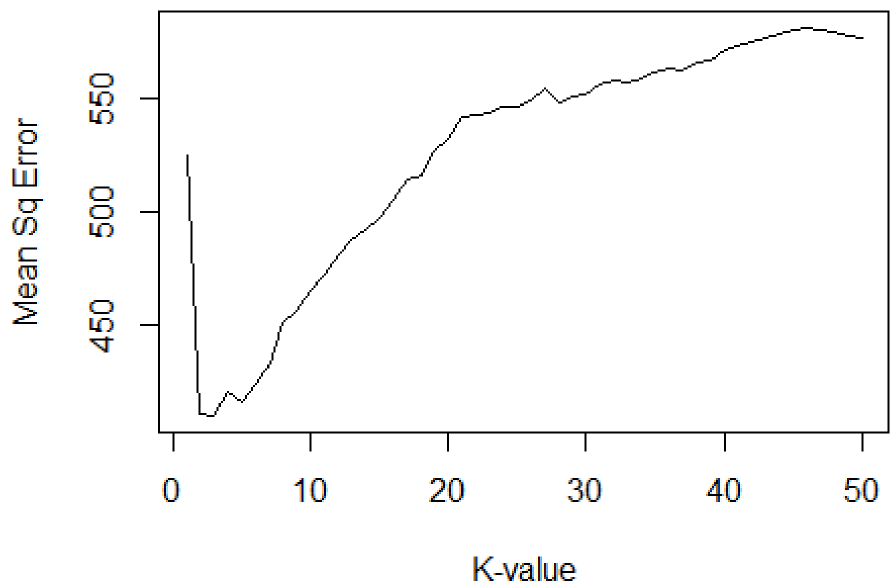
```
Train.data=read.csv("CancerData.csv",header=T)
Test.data=read.csv("CancerHoldoutData.csv",header=T)
library(ISLR)
library(car)

library(class)
library(FNN)

attach(Train.data)

#Filling values for PctSomeCol 18_24
Train.data$PctSomeCol18_24=100-(Train.data$PctBachDeg18_24+Train.data$PctHS18
_24+Train.data$PctNoHS18_24)
Test.data$PctSomeCol18_24=100-(Test.data$PctBachDeg18_24+Test.data$PctHS18_24
+Test.data$PctNoHS18_24)

#Correcting Median age values
i=1
for (i in 1:2590)
{
  if (Train.data$MedianAge[i]>130)
  {
    Train.data$MedianAge[i]=(Train.data$MedianAgeMale[i]+Train.data$MedianAge
Female[i])/2
  }
}

i=1
for (i in 1:457)
{
  if (Test.data$MedianAge[i]>130)
  {
    Test.data$MedianAge[i]=(Test.data$MedianAgeMale[i]+Test.data$MedianAgeFem
ale[i])/2
  }
}

#Removing geography data
Train.data=Train.data[,-c(8)]
attach(Train.data)
```

```
linear.fit=lm(TARGET_deathRate~.-PctSomeCol18_24 ,data=Train.data) #Removing
PctSomeCol18_24 as it is collinear with remaining education data
summary(linear.fit)

##
## Call:
## lm(formula = TARGET_deathRate ~ . - PctSomeCol18_24, data = Train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -86.299 -12.148  -0.113  11.640 127.367
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1.060e+02  1.423e+01   7.449 1.27e-13 ***
## incidenceRate           2.181e-01  8.246e-03  26.444  < 2e-16 ***
## medIncome              -2.647e-04  7.983e-05  -3.316 0.000927 ***
## povertyPercent          3.032e-01  1.702e-01   1.781 0.074975 .
## MedianAge              -4.836e-01  1.198e+00  -0.404 0.686468
## MedianAgeMale           6.259e-02  6.993e-01   0.090 0.928688
## MedianAgeFemale         7.985e-02  5.861e-01   0.136 0.891647
## AvgHouseholdSize        6.479e-01  1.204e+00   0.538 0.590540
## PercentMarried          1.913e-01  1.614e-01   1.185 0.236097
## PctNoHS18_24           -4.965e-02  6.253e-02  -0.794 0.427284
## PctHS18_24              4.562e-01  5.231e-02   8.721  < 2e-16 ***
## PctBachDeg18_24        -3.489e-01  1.184e-01  -2.947 0.003242 **
## PctPrivateCoverage     -2.791e-01  1.141e-01  -2.447 0.014489 *
## PctPublicCoverage       2.638e-02  2.136e-01   0.123 0.901723
## PctPublicCoverageAlone  5.644e-01  2.781e-01   2.030 0.042463 *
## PctWhite               -4.874e-02  6.359e-02  -0.766 0.443498
## PctBlack                3.859e-02  6.245e-02   0.618 0.536636
## PctAsian               -2.668e-01  1.990e-01  -1.341 0.180004
## PctOtherRace           -9.974e-01  1.296e-01  -7.699 1.95e-14 ***
## PctMarriedHouseholds   -3.104e-01  1.558e-01  -1.992 0.046428 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.36 on 2570 degrees of freedom
## Multiple R-squared:  0.4728, Adjusted R-squared:  0.4689
## F-statistic: 121.3 on 19 and 2570 DF,  p-value: < 2.2e-16

Test.data=Test.data[,-c(8)]
attach(Test.data)

lm.predict=predict(linear.fit,Test.data,interval="predict")

mean((lm.predict[,1]-Test.data$TARGET_deathRate)^2)

## [1] 414.3202
```

KNN Code:

```
Train.data=read.csv("CancerData.csv",header=T)
Test.data=read.csv("CancerHoldoutData.csv",header=T)
library(ISLR)
library(car)

library(class)
library(FNN)

attach(Train.data)

Train.data$PctSomeCol18_24=100-(Train.data$PctBachDeg18_24+Train.data$PctHS18
_24+Train.data$PctNoHS18_24)
Test.data$PctSomeCol18_24=100-(Test.data$PctBachDeg18_24+Test.data$PctHS18_24
+Test.data$PctNoHS18_24)

i=1
for (i in 1:2590)
{
  if (Train.data$MedianAge[i]>130)
  {
    Train.data$MedianAge[i]=(Train.data$MedianAgeMale[i]+Train.data$MedianAge
Female[i])/2
  }
}

i=1
for (i in 1:457)
{
  if (Test.data$MedianAge[i]>130)
  {
    Test.data$MedianAge[i]=(Test.data$MedianAgeMale[i]+Test.data$MedianAgeFem
ale[i])/2
  }
}

#Removing Geography data
train.x=Train.data[,-c(8)]
test.x=Test.data[,-c(8)]

train.drate=train.x$TARGET_deathRate
test.drate=test.x$TARGET_deathRate

set.seed(2)
i=1
mse=matrix(,nrow=50,ncol=2)
for (i in 1:nrow(mse))
{
  knn.pred=knn.reg(train.x,test.x,train.drate,k=i)
```

```
  mse[i,1]=mean((test.drate-knn.pred$pred)^2)
  mse[i,2]=i
}
plot(mse[,2],mse[,1],type="l",xlab="K-value",ylab="Mean Sq Error")
```




```
min(mse[,1])

## [1] 410.1837

head(mse)

##          [,1] [,2]
## [1,] 524.2508    1
## [2,] 411.0644    2
## [3,] 410.1837    3
## [4,] 420.8188    4
## [5,] 416.2928    5
## [6,] 424.6396    6
```