Homework 3: Ensemble Techniques

Gayathri Prerepa – gp2254

This report covers 4 ensemble techniques: cross validation, random forest, stacking and gradient boosting

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. K = 10 means CV is performed by splitting the data in 10 random data sets.

Cross validation is done using the following in R:

train_control <- trainControl(method = "cv", number = 10)

train_control1 <- trainControl(method = "cv", number = 3)

ctrl <- trainControl(method = "cv", savePred=T, classProb=T)

I have explained the below 4 models in the previous report, therefore will be including information only about the new ones.

1.1. Naïve bayes without cross validation

nb.m1=train(x=x_nb,y=y_nb,method="nb",trControl=train_control1)

```
> nb.m1
Naive Bayes

336268 samples
     4 predictor
     5 classes: 'BRONX', 'BROOKLYN', 'MANHATTAN', 'QUEENS', 'STATEN ISLAND'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 224179, 224178, 224179
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9998870  0.9998524
   TRUE      0.9998245  0.9997709

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
```

```
predi$class1 <- predict(nb.ml, newdata = testing_nb)
pred = subset(predi, select = -c(1))
CM<-confusionMatrix(pred$class1, testing_nb$class1)
f1_nb <- F1_Score(pred$class1, testing_nb$class1)
CM
f1_nb
bias(as.numeric(testing_nb$class1), as.numeric(pred$class1))
var(as.numeric(pred$class1))
```

```
> f1_nb
[1] 0.9997664
>
> bias(as.numeric(testing_nb$class1), as.numeric(pred$class1))
[1] -0.0001070597
> var(as.numeric(pred$class1))
[1] 1.350647
> CM
Confusion Matrix and Statistics

              Reference
Prediction      BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
  BRONX         23538        4         0      0             0
  BROOKLYN          5    31605         1      0             1
  MANHATTAN         1        4     27659      0             0
  QUEENS            1        2         3  24539             1
  STATEN ISLAND     0        0         0      0          4723

Overall Statistics

               Accuracy : 0.9998
                 95% CI : (0.9997, 0.9999)
    No Information Rate : 0.2821
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9997

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: BRONX Class: BROOKLYN Class: MANHATTAN Class: QUEENS Class: STATEN ISLAND
Sensitivity                0.9997          0.9997           0.9999        1.0000              0.99958
Specificity                1.0000          0.9999           0.9999        0.9999              1.00000
Pos Pred Value             0.9998          0.9998           0.9998        0.9997              1.00000
Neg Pred Value             0.9999          0.9999           1.0000        1.0000              0.99998
Prevalence                 0.2101          0.2821           0.2468        0.2189              0.04215
Detection Rate             0.2100          0.2820           0.2468        0.2189              0.04214
Detection Prevalence       0.2100          0.2820           0.2468        0.2190              0.04214
Balanced Accuracy          0.9998          0.9998           0.9999        1.0000              0.99979
```

1.2. Naïve bayes with 10-fold cross validation

nb_cv <- train(x = x_nb,y = y_nb,method = "nb",trControl = train_control)

```
> nb_cv
Naive Bayes

336268 samples
     4 predictor
     5 classes: 'x1', 'x2', 'x3', 'x4', 'x5'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 302642, 302641, 302642, 302642, 302641, 302641, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.9999019  0.9998718
   TRUE      0.9998245  0.9997709

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 1.
```

predi$class1 <- predict(nb_cv, newdata = testing_nb)

pred = subset(predi, select = -c(1))

CM<-confusionMatrix(pred$class1, testing_nb$class1)

f1_nb <- F1_Score(pred$class1, testing_nb$class1)

CM

f1_nb

bias(as.numeric(testing_nb$class1), as.numeric(pred$class1))

var(as.numeric(pred$class1))

```
> f1_nb
[1] 0.9997664
>
> bias(as.numeric(testing_nb$class1), as.numeric(pred$class1))
[1] -0.0001070597
> var(as.numeric(pred$class1))
[1] 1.350647
> CM
Confusion Matrix and Statistics

               Reference
Prediction      BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
  BRONX         23538        4         0      0             0
  BROOKLYN          5    31605         1      0             1
  MANHATTAN         1        4     27659      0             0
  QUEENS            1        2         3  24539             1
  STATEN ISLAND     0        0         0      0          4723

Overall Statistics

               Accuracy : 0.9998
                 95% CI : (0.9997, 0.9999)
    No Information Rate : 0.2821
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9997

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: BRONX Class: BROOKLYN Class: MANHATTAN Class: QUEENS Class: STATEN ISLAND
Sensitivity                0.9997          0.9997           0.9999        1.0000                0.99958
Specificity                1.0000          0.9999           0.9999        0.9999                1.00000
Pos Pred Value             0.9998          0.9998           0.9998        0.9997                1.00000
Neg Pred Value             0.9999          0.9999           1.0000        1.0000                0.99998
Prevalence                 0.2101          0.2821           0.2468        0.2189                0.04215
Detection Rate             0.2100          0.2820           0.2468        0.2189                0.04214
Detection Prevalence       0.2100          0.2820           0.2468        0.2190                0.04214
Balanced Accuracy          0.9998          0.9998           0.9999        1.0000                0.99979
```

2.1. Decision tree without cross validation

dtt<- train(class1~. ,data = train_subset , method = "rpart")

```
CART

10000 samples
    4 predictor
    5 classes: 'x1', 'x2', 'x3', 'x4', 'x5'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 10000, 10000, 10000, 10000, 10000, 10000, ...
Resampling results across tuning parameters:

  cp         Accuracy   Kappa
  0.2947029  0.9010967  0.8688442
  0.3241312  0.5352102  0.3741069
  0.3267937  0.4222420  0.2115088

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.2947029.
```

```
pred <- predict(dtt, test_subset)

dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)

accuracy<- sum(diag(dataRev)) / sum(dataRev)

cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))

f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))

accuracy

cm

f1

var(as.numeric(pred))

bias(as.numeric(test_subset$class1), as.numeric(pred))
```

```
> accuracy
[1] 0.7504
> cm
      y_pred
y_true BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
    x1     0        0         0      0              0
    x2     0     2790         0      0              0
    x3  2060        0      2371      0            436
    x4     0        0         0   2343              0
    x5     0        0         0      0              0
> f1
[1] 1
> var(as.numeric(pred))
[1] 0.511353
> bias(as.numeric(test_subset$class1), as.numeric(pred))
[1] -0.3248
```

2.2. Decision tree with 10- fold cross validation:

```
dtt_cv <- train(class1~. ,data = train_subset , method = "rpart", trControl = train_control)
```

```
CART

10000 samples
    4 predictor
    5 classes: 'x1', 'x2', 'x3', 'x4', 'x5'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 8998, 9000, 9001, 8999, 8999, 9001, ...
Resampling results across tuning parameters:

  cp         Accuracy   Kappa
  0.2947029  0.8769003  0.8366424
  0.3241312  0.6582465  0.5455621
  0.3267937  0.4494768  0.2524323

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.2947029.
```

```
pred <- predict(dtt_cv, test_subset)
dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)
accuracy<- sum(diag(dataRev)) / sum(dataRev)
cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))
f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))
accuracy
cm
f1
var(as.numeric(pred))
bias(as.numeric(test_subset$class1), as.numeric(pred))
```

```
> accuracy
[1] 0.7504
> cm
      y_pred
y_true BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
    x1     0        0         0      0             0
    x2     0     2790         0      0             0
    x3  2060        0      2371      0           436
    x4     0        0         0   2343             0
    x5     0        0         0      0             0
> f1
[1] 1
> var(as.numeric(pred))
[1] 0.511353
> bias(as.numeric(test_subset$class1), as.numeric(pred))
[1] -0.3248
```

3.1. Svm without cross validation

```
svmm <- train(class1~. ,data = train_subset , method = 'svmLinear')
```

```
> svmm
Support Vector Machines with Linear Kernel

10000 samples
    4 predictor
    5 classes: 'x1', 'x2', 'x3', 'x4', 'x5'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 10000, 10000, 10000, 10000, 10000, 10000, ...
Resampling results:

  Accuracy  Kappa
  1         1

Tuning parameter 'C' was held constant at a value of 1
```

```
pred <- predict(svmm, test_subset)

dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)

accuracy<- sum(diag(dataRev)) / sum(dataRev)

cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))

f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))

accuracy

cm

f1

var(as.numeric(pred))

bias(as.numeric(test_subset$class1), as.numeric(pred))
```

```
> accuracy
[1] 1
> cm
       y_pred
y_true BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
    x1  2060        0         0      0             0
    x2     0     2790         0      0             0
    x3     0        0      2371      0             0
    x4     0        0         0   2343             0
    x5     0        0         0      0           436
> f1
[1] 1
> var(as.numeric(pred))
[1] 1.375307
> bias(as.numeric(test_subset$class1), as.numeric(pred))
[1] 0
```

3.2. Svm with 10-fold cross validation:

svm_cv<-train(class1~. ,data = train_subset , method = 'svmLinear',trControl = ctrl)

```
Support Vector Machines with Linear Kernel

10000 samples
    4 predictor
    5 classes: 'x1', 'x2', 'x3', 'x4', 'x5'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9001, 9000, 9000, 9002, 9000, 8999, ...
Resampling results:

  Accuracy   Kappa
  0.7667998  0.6970251

Tuning parameter 'C' was held constant at a value of 1
```

```
pred <- predict(svm_cv, test_subset)
dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)
accuracy<- sum(diag(dataRev)) / sum(dataRev)
cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))
f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))
accuracy
cm
f1
var(as.numeric(pred))
bias(as.numeric(test_subset$class1), as.numeric(pred))
```

```
> accuracy
[1] 0.7657
> cm
       y_pred
y_true BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
    x1  2060        0         0   2343             0
    x2     0     2790         0      0             0
    x3     0        0      2371      0             0
    x4     0        0         0      0             0
    x5     0        0         0      0           436
> f1
[1] 0.6374749
> var(as.numeric(pred))
[1] 1.064665
> bias(as.numeric(test_subset$class1), as.numeric(pred))
[1] 0.7029
```

3.1. Knn without cross validation:

```
knnf <- train(class1~., data = train_subset, method = "knn", trControl = trainControl(method = "cv", number = 3), preProcess = c("center","scale"), tuneLength = 20, na.action="na.omit")
pred <- predict(knnf, test_subset)
dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)
accuracy<- sum(diag(dataRev)) / sum(dataRev)
cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))
f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))
accuracy
cm
```

f1

var(as.numeric(pred))

bias(as.numeric(test_subset$class1), as.numeric(pred))

```
> knnf
k-Nearest Neighbors

10000 samples
    4 predictor
    5 classes: 'x1', 'x2', 'x3', 'x4', 'x5'

Pre-processing: centered (7), scaled (7)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 6667, 6667, 6666
Resampling results across tuning parameters:

  k   Accuracy  Kappa
   5  1         1
   7  1         1
   9  1         1
  11  1         1
  13  1         1
  15  1         1
  17  1         1
  19  1         1
  21  1         1
  23  1         1
  25  1         1
  27  1         1
  29  1         1
  31  1         1
  33  1         1
  35  1         1
  37  1         1
  39  1         1
  41  1         1
  43  1         1

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 43.
```

```
> accuracy_Knn
[1] 1
> cm_knn
       y_pred
y_true BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
    x1  2060        0         0      0             0
    x2     0     2790         0      0             0
    x3     0        0      2371      0             0
    x4     0        0         0   2343             0
    x5     0        0         0      0           436
> f1_knn
[1] 1
> var(as.numeric(pred_knn))
[1] 1.375307
> bias(as.numeric(test_subset$class1), as.numeric(pred_knn))
[1] 0
```

4.2. Knn with 10-fold cross validation:

knn_cv <- train(class1~., data = train_subset, method = "knn", trControl = train_control, preProcess = c("center","scale"), tuneLength = 20, na.action="na.omit")

```
> knn_cv
k-Nearest Neighbors

10000 samples
    4 predictor
    5 classes: 'x1', 'x2', 'x3', 'x4', 'x5'

Pre-processing: centered (7), scaled (7)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 9001, 9000, 9001, 9001, 9000, 8999, ...
Resampling results across tuning parameters:

  k   Accuracy  Kappa
   5  1         1
   7  1         1
   9  1         1
  11  1         1
  13  1         1
  15  1         1
  17  1         1
  19  1         1
  21  1         1
  23  1         1
  25  1         1
  27  1         1
  29  1         1
  31  1         1
  33  1         1
  35  1         1
  37  1         1
  39  1         1
  41  1         1
  43  1         1

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 43.
```

pred <- predict(knn_cv, test_subset)

dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)

accuracy<- sum(diag(dataRev)) / sum(dataRev)

cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))

f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))

accuracy

cm

f1

var(as.numeric(pred))

bias(as.numeric(test_subset$class1), as.numeric(pred))

```
> accuracy_Knn
[1] 1
> cm_knn
      y_pred
y_true BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
    x1  2060        0         0      0             0
    x2     0     2790         0      0             0
    x3     0        0      2371      0             0
    x4     0        0         0   2343             0
    x5     0        0         0      0           436
> f1_knn
[1] 1
> var(as.numeric(pred_knn))
[1] 1.375307
> bias(as.numeric(test_subset$class1), as.numeric(pred_knn))
[1] 0
```

With cross validation, the training data is varied and this reduces overfitting, Overall, in all the above models with and without CV, the models with CV had higher accuracy but more bias and overfitting. These errors were able to overcome by using cross validation ensemble technique. K = 10 means 10-fold cross-validation. The above results cannot be generalised to all cases because performance depends on many factors, but these were the results observed for my models Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data.

5. Random forest model:

A random forest consists of a group (an ensemble) of individual decision trees. Therefore, the technique is called Ensemble Learning. A large group of uncorrelated decision trees can produce more accurate and stable results than any of individual decision trees.

rf <- randomForest(y_nb~. ,data = x_nb, ntree = 250,trControl = train_control)

pred <- predict(rf, test_subset)

```
dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)

accuracy<- sum(diag(dataRev)) / sum(dataRev)

cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))

f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))

accuracy

cm

f1

var(as.numeric(pred))

bias(as.numeric(test_subset$class1), as.numeric(pred))
```

```
Confusion Matrix and Statistics

               Reference
Prediction      BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
  BRONX         23541        2         4      1             0
  BROOKLYN          2    31609         2      0             0
  MANHATTAN         2        4     27651      0             0
  QUEENS            0        0         6  24538             0
  STATEN ISLAND     0        0         0      0          4725

Overall Statistics

               Accuracy : 0.9998
                 95% CI : (0.9997, 0.9999)
    No Information Rate : 0.2821
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9997

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: BRONX Class: BROOKLYN Class: MANHATTAN Class: QUEENS Class: STATEN ISLAND
Sensitivity                0.9998          0.9998           0.9996        1.0000               1.00000
Specificity                0.9999          1.0000           0.9999        0.9999               1.00000
Pos Pred Value             0.9997          0.9999           0.9998        0.9998               1.00000
Neg Pred Value             1.0000          0.9999           0.9999        1.0000               1.00000
Prevalence                 0.2101          0.2821           0.2468        0.2189               0.04215
Detection Rate             0.2100          0.2820           0.2467        0.2189               0.04215
Detection Prevalence       0.2101          0.2820           0.2467        0.2190               0.04215
Balanced Accuracy          0.9999          0.9999           0.9997        0.9999               1.00000
> f1_nb <- F1_Score(pred$class1, testing_nb$class1)
> f1_nb
[1] 0.9997664
>
> bias(as.numeric(testing_nb$class1), as.numeric(pred$class1))
[1] -8.921641e-06
> var(as.numeric(pred$class1))
[1] 1.350845
```

Random forest achieved high accuracy but the **bias of the model was also high**, which is not desirable. This might be because I have trained the model on a smaller subset of the actual data to fasten the process.

6. Stacking technique:

Stacked Ensemble method is a supervised ensemble machine learning algorithm that finds the optimal combination of a collection of prediction algorithms using a process called stacking.

```
h2o.init()
train_df <-as.h2o(train_subset)
test_df<-as.h2o(test_subset)
y <- "class1"
x <- setdiff(names(train_df), y)
nfolds <- 5
my_gbm <- h2o.gbm(x = x,
                  y = y,
                  training_frame = train_df,
                  nfolds = nfolds,
                  keep_cross_validation_predictions = TRUE,
                  seed = 5)
my_rf <- h2o.randomForest(x = x,
                          y = y,
                          training_frame = train_df,
                          nfolds = nfolds,
                          keep_cross_validation_predictions = TRUE,
                          seed = 5)
my_lr <- h2o.glm(x = x,
                 y = y,
                 training_frame = train_df,
                 family = c("binomial"),
                 nfolds = nfolds,
                 keep_cross_validation_predictions = TRUE,
                 seed = 5)
```

```
ensemble <- h2o.stackedEnsemble(x = x,
                                y = y,
                                metalearner_algorithm="drf",
                                training_frame = train_df,
                                base_models = list(my_gbm, my_rf, my_lr))
```

I tried building a stacking ensemble model with 3 machine learning models, but the above code gives me an error as my data set is multiclass classification, hence I was unable to perform this.

```
Illegal argument(s) for GLM model: GLM_model_R_1651975304947_4.  Details: ERRR on field: _family:
onse to be a 2-class categorical or a binary column (0/1)
```

7. Gradient boosting:

It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error.

Gradient Boosting Algorithm is generally used when we want to decrease the Bias error.

```
gbmm <- train(y1~ ., data = train_subset, method = "gbm", trControl = train_control,verbose = FALSE)

pred <- predict(gbmm, test_subset)

dataRev <- table(actualclass=test_subset$class1, predictedclass= pred)

accuracy<- sum(diag(dataRev)) / sum(dataRev)

cm <- ConfusionMatrix(as.factor(test_subset$class1), as.factor(pred))

f1<- F1_Score(as.numeric(pred),as.numeric(test_subset$class1))

accuracy

cm

f1

var(as.numeric(pred))

bias(as.numeric(test_subset$class1), as.numeric(pred))
```

```
Stochastic Gradient Boosting

10000 samples
    4 predictor
    5 classes: 'BRONX', 'BROOKLYN', 'MANHATTAN', 'QUEENS', 'STATEN ISLAND'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 8999, 9001, 9000, 9001, 9000, 9000, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  Accuracy  Kappa
  1                   50      1         1
  1                  100      1         1
  1                  150      1         1
  2                   50      1         1
  2                  100      1         1
  2                  150      1         1
  3                   50      1         1
  3                  100      1         1
  3                  150      1         1

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant
 at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 50, interaction.depth = 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
> accuracy
[1] 1
> cm
                y_pred
y_true           BRONX BROOKLYN MANHATTAN QUEENS STATEN ISLAND
  BRONX           2005        0         0      0             0
  BROOKLYN           0     2747         0      0             0
  MANHATTAN          0        0      2415      0             0
  QUEENS             0        0         0   2411             0
  STATEN ISLAND      0        0         0      0           422
> f1
[1] 1
> var(as.numeric(pred))
[1] 1.364096
> bias(as.numeric(test_subset$class1), as.numeric(pred))
[1] 0
```

Gradient boosting has outperformed the random forest model, in my above analysis. This is as expected, theoretically. It also **reduced the bias** as expected.

<u>Something new in this homework:</u>

- One of the ensemble techniques which was not mentioned to us, but I have tried for analysis is to take a subset of the training set and train the model
- This makes the training faster but increases overfitting.
- This could be the reason for high accuracy and performance of the models.

Overall, these ensemble techniques have tuned the models and even though accuracy decreased it some cases, it **reduced the bias and variance** of the models.