Aprendizaje Computacional
Curso 24-25

Digit Recognition

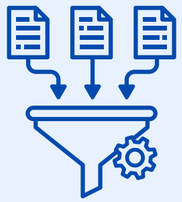
Participantes:

Fernando Javier López Cerezo
Emilio Gómez Esteban
Alberto Trigueros Postigo
María Peinado Toledo

Resumen del proyecto

El conjunto de datos MNIST es un recurso clásico para la clasificación de imágenes de dígitos escritos a mano, utilizado en la investigación y enseñanza de la visión artificial. En esta proyecto, el objetivo es clasificar correctamente las imágenes de dígitos. Mientras debemos experimentar con diferentes algoritmos para entender su rendimiento y compararlos.

Objetivos



Procesamiento de datos del problema

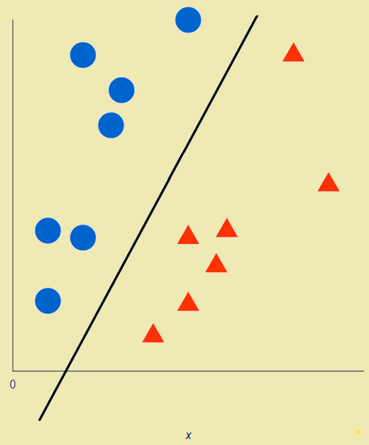
Resolver el problema con distintos modelos de machine learning

Determinar el mejor modelo con mayor accuracy y menor consumo de recursos

Modelos

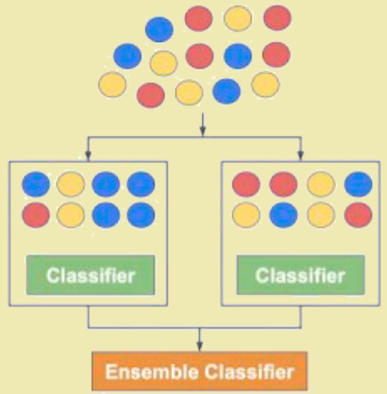
SVM

Alberto



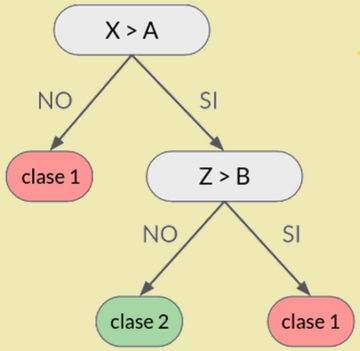
Bagging

María



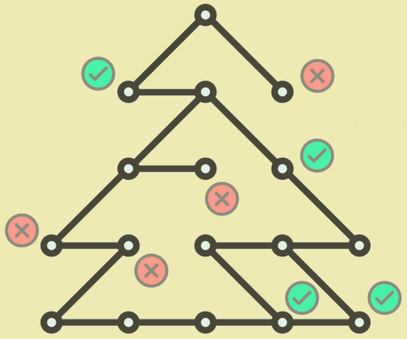
Rpart

Emilio



Random Forest

Javier



Procesamiento de datos

Carga

```
train <- read.csv("train.csv")
test <- read.csv("test.csv")
```

Inspección

```
prop.table(table(train$label)) * 100
train$label <- factor(train$label)
```

División

```
d_ size <- nrow ( train )
dtest _ size <- ceiling (0.2 * d_ size )
samples <- sample (1: d_size , d_size ,
replace = FALSE )
indexes <- samples [1: dtest _ size ]
dtrain <- train [ - indexes ,]
dtest <- train [ indexes ,]
```

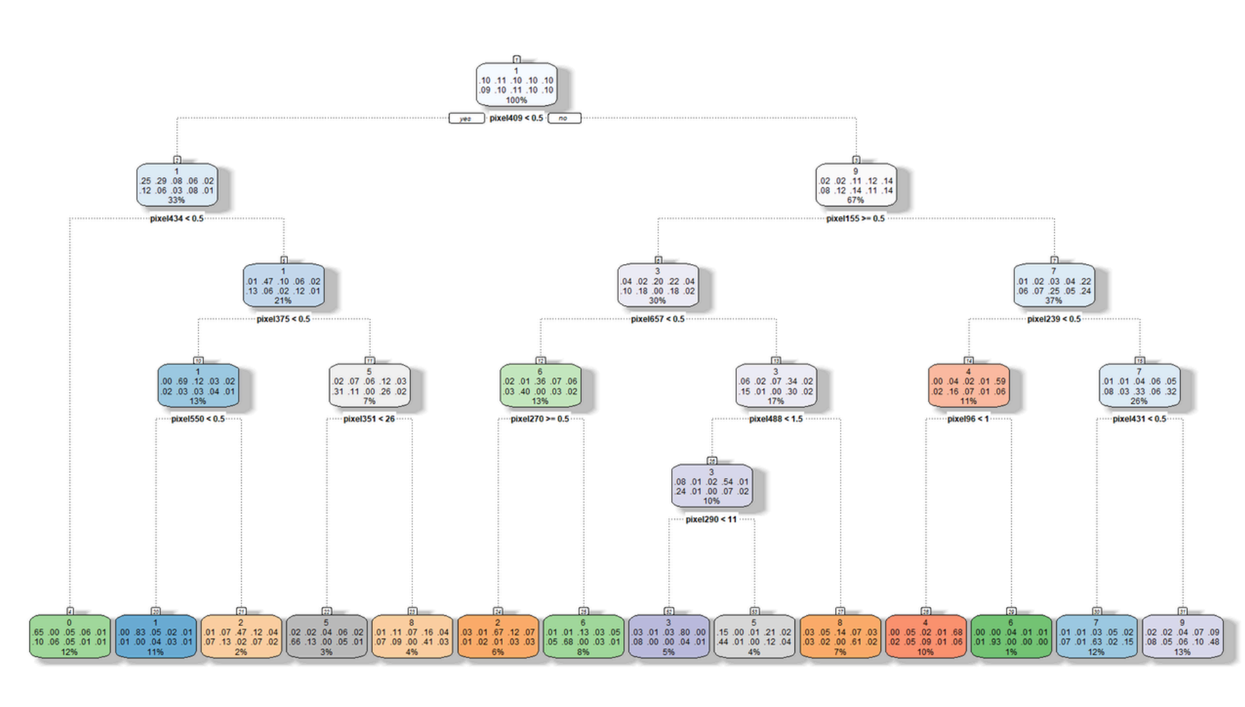
Train	Test
80%	20%

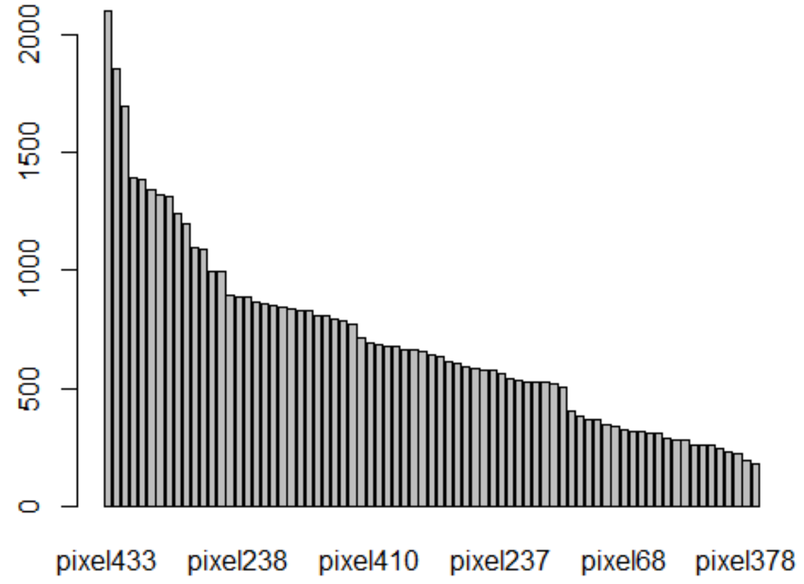
Rpart

```
start _ time <- Sys . time ()
tree = rpart ( label ~. , data = dtrain , method = " class ")
end _ time <- Sys . time ()
```

```
matrizconfusiontree <- table ( predict ( tree , newdata = dtest , type = " class ") ,
dtest $ label )
accuracytree <- sum ( diag ( matrizconfusiontree ) ) / sum ( matrizconfusiontree )
accuracytree
fancyRpartPlot ( tree )
barplot ( tree $ variable . importance )
```

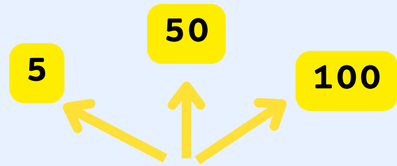
Rpart


pixel409 < 0.5
yes
no
pixel434 < 0.5
pixel155 >= 0.5
pixel375 < 0.5
pixel657 < 0.5
pixel239 < 0.5
pixel550 < 0.5
pixel351 < 26
pixel270 >= 0.5
pixel488 < 1.5
pixel96 < 1
pixel431 < 0.5
pixel290 < 11


0
500
1000
1500
2000
pixel433
pixel238
pixel410
pixel237
pixel68
pixel378

Random Forest

```
start _ time <- Sys . time ()
rf. model <- randomForest ( label ~ . , data = dtrain , ntree = 5)
end _ time <- Sys . time ()
save (rf.model , file ="rf5 .rda ")
end _ time - start _ time
```



```
rf. predict <- predict (rf.model , dtest )
matrizconfusionRF <- table (rf. predict , dtest $ label )
accuracyRF <- sum( diag ( matrizconfusionRF ) ) / sum( matrizconfusionRF )
accuracyRF
```

SVM

```
start _ time <- Sys . time ()
filter <- ksvm ( label ~ . , data = dtrain , kernel = " polydot ", kpar = list
( degree = 3) , cross = 3)
end _ time <- Sys . time ()
```

```
matrizconfusionKSVMpol <- table ( predict ( filter , dtest ) , dtest $ label )
accuracyksvmpol <- sum(diag (matrizconfusionKSVMpol)) / sum(matrizconfusionKSVMpol)
# Mostrar resultados
print ( matrizconfusionKSVMpol )
cat (" Accuracy del modelo :", accuracyksvmpol , "\n")
cat (" Tiempo de ejecución SVM :", end_time - start _time , "\n")
```

Bagging

```
start _ time <- Sys . time ()
Modelo _ AdaBag <- bagging ( label ~. , data = dtrain , na. action = na.omit ,
mfinal =9 , control = rpart . control ( cp = 0.001 , minsplit =7) )
end _ time <- Sys . time ()
```

```
matrizconfusionBag <- table ( dtest [ , " label "] , predict ( Modelo _AdaBag,
                    newdata = dtest , type = " class ")$ class )
accuracyBag <- sum( diag ( matrizconfusionBag ) ) / sum( matrizconfusionBag )
print ( matrizconfusionBag )
cat (" Accuracy del modelo :", accuracyBag , "\n")
cat (" Tiempo de ejecución Bagging :", end_time - start _time , "\n")
```

Resultados obtenidos de los modelos

Rpart

Accuracy: 0.6333
Time: 53.2402 seg

Random Forest

5 Árboles
Accuracy: 0.9104
Time: 24.9980 seg

50 Árboles
Accuracy: 0.9632
Time: 4.4706 min

100 Árboles
Accuracy: 0.9663
Time: 16.855 min

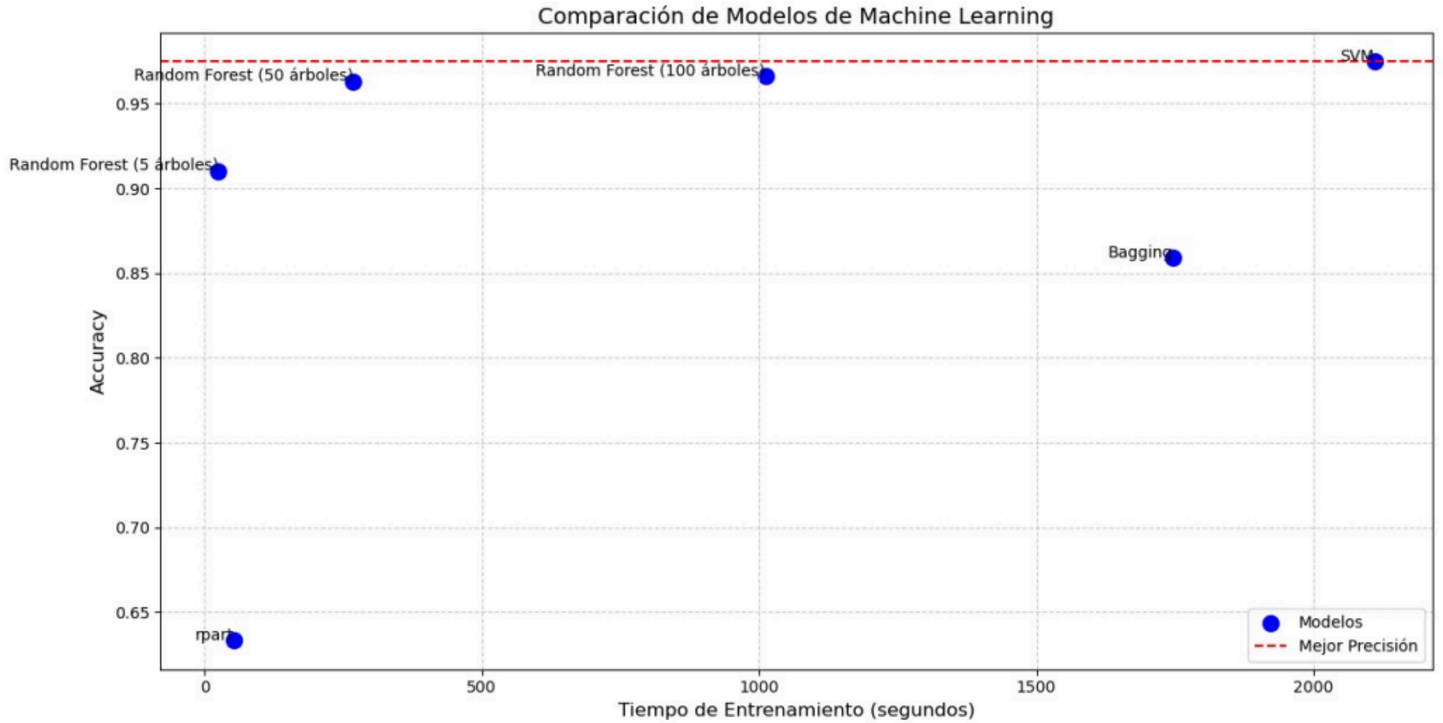
SVM

Accuracy: 0.9748
Time: 35.1554 min

Bagging

Accuracy: 0.8594
Time: 29.1157 min

Conclusión



Depende de los requerimientos del problema

Maximizar precisión sin tener en cuenta el tiempo

SVM

Compromiso entre precisión y eficiencia computacional

Random Forest

50 Árboles

Maximizar eficiencia computacional con menor eficiencia

Rpart