**Session24\_Assignment-1**

5. Problem Statement

1. Perform the below given activities:

a. Take a sample data set of your choice

b. Apply random forest, logistic regression using Spark R

c. Predict for new dataset

library(sparklyr)

library(ggplot2)

library(dplyr)

##

## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':

##

## filter, lag

## The following objects are masked from 'package:base':

##

## intersect, setdiff, setequal, union

sc <- spark\_connect(master = "local")

iris\_tbl <- copy\_to(sc, iris, "iris", overwrite = TRUE)

iris\_tbl

## # Source: table<iris> [?? x 5]

## # Database: spark\_shell\_connection

## Sepal\_Length Sepal\_Width Petal\_Length Petal\_Width Species

## <dbl> <dbl> <dbl> <dbl> <chr>

## 1 5.1 3.5 1.4 0.2 setosa

## 2 4.9 3 1.4 0.2 setosa

## 3 4.7 3.2 1.3 0.2 setosa

## 4 4.6 3.1 1.5 0.2 setosa

## 5 5 3.6 1.4 0.2 setosa

## 6 5.4 3.9 1.7 0.4 setosa

## 7 4.6 3.4 1.4 0.3 setosa

## 8 5 3.4 1.5 0.2 setosa

## 9 4.4 2.9 1.4 0.2 setosa

## 10 4.9 3.1 1.5 0.1 setosa

## # ... with more rows

lm\_model <- iris\_tbl %>%

select(Petal\_Width, Petal\_Length) %>%

ml\_linear\_regression(Petal\_Length ~ Petal\_Width)

iris\_tbl %>%

select(Petal\_Width, Petal\_Length) %>%

collect %>%

ggplot(aes(Petal\_Length, Petal\_Width)) +

geom\_point(aes(Petal\_Width, Petal\_Length), size = 2, alpha = 0.5) +

geom\_abline(aes(slope = coef(lm\_model)[["Petal\_Width"]],

intercept = coef(lm\_model)[["(Intercept)"]]),

color = "red") +

labs(

x = "Petal Width",

y = "Petal Length",

title = "Linear Regression: Petal Length ~ Petal Width",

subtitle = "Use Spark.ML linear regression to predict petal length as a f

unction of petal width."

)

pca\_model <- tbl(sc, "iris") %>%

select(-Species) %>%

ml\_pca()

print(pca\_model)

## Explained variance:

##

## PC1 PC2 PC3 PC4

## 0.924618723 0.053066483 0.017102610 0.005212184

##

## Rotation:

## PC1 PC2 PC3 PC4

## Sepal\_Length -0.36138659 -0.65658877 0.58202985 0.3154872

## Sepal\_Width 0.08452251 -0.73016143 -0.59791083 -0.3197231

## Petal\_Length -0.85667061 0.17337266 -0.07623608 -0.4798390

## Petal\_Width -0.35828920 0.07548102 -0.54583143 0.7536574

b. Apply random forest, logistic regression using Spark R

c. Predict for new dataset

#Random Forest

#Use Spark's Random Forest to perform regression or multicla

ss classification.

rf\_model <- iris\_tbl %>%

ml\_random\_forest(Species ~ Petal\_Length + Petal\_Width, type = "classificati

on")

rf\_predict <- sdf\_predict(rf\_model, iris\_tbl) %>%

ft\_string\_indexer("Species", "Species\_idx") %>%

collect

## Warning in sdf\_predict.ml\_model(rf\_model, iris\_tbl): The signature

## sdf\_predict(model, dataset) is deprecated and will be removed in a future

## version. Use sdf\_predict(dataset, model) or ml\_predict(model, dataset)

## instead.

table

## function (..., exclude = if (useNA == "no") c(NA, NaN), useNA = c("no",

## "ifany", "always"), dnn = list.names(...), deparse.level = 1)

## {

## list.names <- function(...) {

## l <- as.list(substitute(list(...)))[-1L]

## nm <- names(l)

## fixup <- if (is.null(nm))

## seq\_along(l)

## else nm == ""

## dep <- vapply(l[fixup], function(x) switch(deparse.level +

## 1, "", if (is.symbol(x)) as.character(x) else "",

## deparse(x, nlines = 1)[1L]), "")

## if (is.null(nm))

## dep

## else {

## nm[fixup] <- dep

## nm

## }

## }

## miss.use <- missing(useNA)

## miss.exc <- missing(exclude)

## useNA <- if (miss.use && !miss.exc && !match(NA, exclude,

## nomatch = 0L))

## "ifany"

## else match.arg(useNA)

## doNA <- useNA != "no"

## if (!miss.use && !miss.exc && doNA && match(NA, exclude,

## nomatch = 0L))

## warning("'exclude' containing NA and 'useNA' != \"no\"' are a bit

contradicting")

## args <- list(...)

## if (!length(args))

## stop("nothing to tabulate")

## if (length(args) == 1L && is.list(args[[1L]])) {

## args <- args[[1L]]

## if (length(dnn) != length(args))

## dnn <- if (!is.null(argn <- names(args)))

## argn

## else paste(dnn[1L], seq\_along(args), sep = ".")

## }

## bin <- 0L

## lens <- NULL

## dims <- integer()

## pd <- 1L

## dn <- NULL

## for (a in args) {

## if (is.null(lens))

## lens <- length(a)

## else if (length(a) != lens)

## stop("all arguments must have the same length")

## fact.a <- is.factor(a)

## if (doNA)

## aNA <- anyNA(a)

## if (!fact.a) {

## a0 <- a

## a <- factor(a, exclude = exclude)

## }

## add.na <- doNA

## if (add.na) {

## ifany <- (useNA == "ifany")

## anNAc <- anyNA(a)

## add.na <- if (!ifany || anNAc) {

## ll <- levels(a)

## if (add.ll <- !anyNA(ll)) {

## ll <- c(ll, NA)

## TRUE

## }

## else if (!ifany && !anNAc)

## FALSE

## else TRUE

## }

## else FALSE

## }

## if (add.na)

## a <- factor(a, levels = ll, exclude = NULL)

## else ll <- levels(a)

## a <- as.integer(a)

## if (fact.a && !miss.exc) {

## ll <- ll[keep <- which(match(ll, exclude, nomatch = 0L) ==

## 0L)]

## a <- match(a, keep)

## }

## else if (!fact.a && add.na) {

## if (ifany && !aNA && add.ll) {

## ll <- ll[!is.na(ll)]

## is.na(a) <- match(a0, c(exclude, NA), nomatch = 0L) >

## 0L

## }

## else {

## is.na(a) <- match(a0, exclude, nomatch = 0L) >

## 0L

## }

## }

## nl <- length(ll)

## dims <- c(dims, nl)

## if (prod(dims) > .Machine$integer.max)

## stop("attempt to make a table with >= 2^31 elements")

## dn <- c(dn, list(ll))

## bin <- bin + pd \* (a - 1L)

## pd <- pd \* nl

## }

## names(dn) <- dnn

## bin <- bin[!is.na(bin)]

## if (length(bin))

## bin <- bin + 1L

## y <- array(tabulate(bin, pd), dims, dimnames = dn)

## class(y) <- "table"

## y

## }

## <bytecode: 0x0000000019782370>

## <environment: namespace:base>

partitions <- tbl(sc, "iris") %>%

sdf\_partition(training = 0.75, test = 0.25, seed = 1099)

fit <- partitions$training %>%

ml\_linear\_regression(Petal\_Length ~ Petal\_Width)

estimate\_mse <- function(df){

sdf\_predict(fit, df) %>%

mutate(resid = Petal\_Length - prediction) %>%

summarize(mse = mean(resid ^ 2)) %>%

collect

}

sapply(partitions, estimate\_mse)

## Warning in sdf\_predict.ml\_model(fit, df): The signature sdf\_predict(model,

## dataset) is deprecated and will be removed in a future version. Use

## sdf\_predict(dataset, model) or ml\_predict(model, dataset) instead.

## Warning: Missing values are always removed in SQL.

## Use `AVG(x, na.rm = TRUE)` to silence this warning

## Warning in sdf\_predict.ml\_model(fit, df): The signature sdf\_predict(model,

## dataset) is deprecated and will be removed in a future version. Use

## sdf\_predict(dataset, model) or ml\_predict(model, dataset) instead.

## Warning: Missing values are always removed in SQL.

## Use `AVG(x, na.rm = TRUE)` to silence this warning

## $training.mse

## [1] 0.2374596

##

## $test.mse

## [1] 0.1898848

#Use ft\_string\_indexer and ft\_index\_to\_string to convert a character column i

nto a numeric column and back again.

ft\_string2idx <- iris\_tbl %>%

ft\_string\_indexer("Species", "Species\_idx") %>%

ft\_index\_to\_string("Species\_idx", "Species\_remap") %>%

collect

table(ft\_string2idx$Species, ft\_string2idx$Species\_remap)

##

## setosa versicolor virginica

## setosa 50 0 0

## versicolor 0 50 0

## virginica 0 0 50

ft\_string2idx <- iris\_tbl %>%

sdf\_mutate(Species\_idx = ft\_string\_indexer(Species)) %>%

sdf\_mutate(Species\_remap = ft\_index\_to\_string(Species\_idx)) %>%

collect

ft\_string2idx %>%

select(Species, Species\_idx, Species\_remap) %>%

distinct

## # A tibble: 3 x 3

## Species Species\_idx Species\_remap

## <chr> <dbl> <chr>

## 1 setosa 2 setosa

## 2 versicolor 0 versicolor

## 3 virginica 1 virginica

#Use Spark's logistic regression to perform logistic regression, modeling a b

inary outcome as a function of one or more explanatory variables.

# Prepare beaver dataset

beaver <- beaver2

beaver$activ <- factor(beaver$activ, labels = c("Non-Active", "Active"))

copy\_to(sc, beaver, "beaver")

## # Source: table<beaver> [?? x 4]

## # Database: spark\_shell\_connection

## day time temp activ

## <dbl> <dbl> <dbl> <chr>

## 1 307 930 36.6 Non-Active

## 2 307 940 36.7 Non-Active

## 3 307 950 36.9 Non-Active

## 4 307 1000 37.2 Non-Active

## 5 307 1010 37.2 Non-Active

## 6 307 1020 37.2 Non-Active

## 7 307 1030 37.2 Non-Active

## 8 307 1040 36.9 Non-Active

## 9 307 1050 37.0 Non-Active

## 10 307 1100 36.9 Non-Active

## # ... with more rows

beaver\_tbl <- tbl(sc, "beaver")

glm\_model <- beaver\_tbl %>%

mutate(binary\_response = as.numeric(activ == "Active")) %>%

ml\_logistic\_regression(binary\_response ~ temp)

glm\_model

## Formula: binary\_response ~ temp

##

## Coefficients:

## (Intercept) temp

## 550.52331 -14.69184

#First, we will copy the mtcars dataset into Spark.

mtcars\_tbl <- copy\_to(sc, mtcars, "mtcars")

# transform our data set, and then partition into 'training', 'test'

partitions <- mtcars\_tbl %>%

filter(hp >= 100) %>%

sdf\_mutate(cyl8 = ft\_bucketizer(cyl, c(0,8,12))) %>%

sdf\_partition(training = 0.5, test = 0.5, seed = 888)

# fit a linear mdoel to the training dataset

fit <- partitions$training %>%

ml\_linear\_regression(mpg ~ wt + cyl)

# summarize the model

summary(fit)

## Deviance Residuals:

## Min 1Q Median 3Q Max

## -2.0947 -1.2747 -0.1129 1.0876 2.2185

##

## Coefficients:

## (Intercept) wt cyl

## 33.795576 -1.596247 -1.580360

##

## R-Squared: 0.8267

## Root Mean Squared Error: 1.437