LinkedIn Predictor App in R

Justin Brooks

2023-12-13

```
s<- read.csv('social_media_usage.csv')
```

```
clean_sm <- function(x) {
  ifelse(x == 1, 1, 0)
}
```

```
s$sm_li <- clean_sm(s$web1h)

# Transform the variables for prediction.
ss <- s %>%
  mutate(
    income = ifelse(income > 9, NA, income),
    education = ifelse(educ2 > 8, NA, educ2),
    parent = ifelse(par == 1, 1, 0),
    married = ifelse(marital == 1, 1, 0),
    female = ifelse(gender == 2, 1, 0),
    age = ifelse(age > 98, NA, age),
    sm_li = ifelse(sm_li == 1, 1, 0)
  ) %>%
  select(sm_li, income, education, parent, married, female, age) %>%
  drop_na()
```

```
print(dim(ss))
```

```
## [1] 1260    7
```

```
print(head(ss))
```

```
##   sm_li income education parent married female age
## 1     0      6         4      0       0      1  77
## 2     0      5         3      0       0      0  59
## 3     0      8         4      0       1      1  60
## 4     0      8         8      0       0      0  73
## 5     1      7         8      0       1      1  65
## 6     0      7         6      0       0      1  62
```

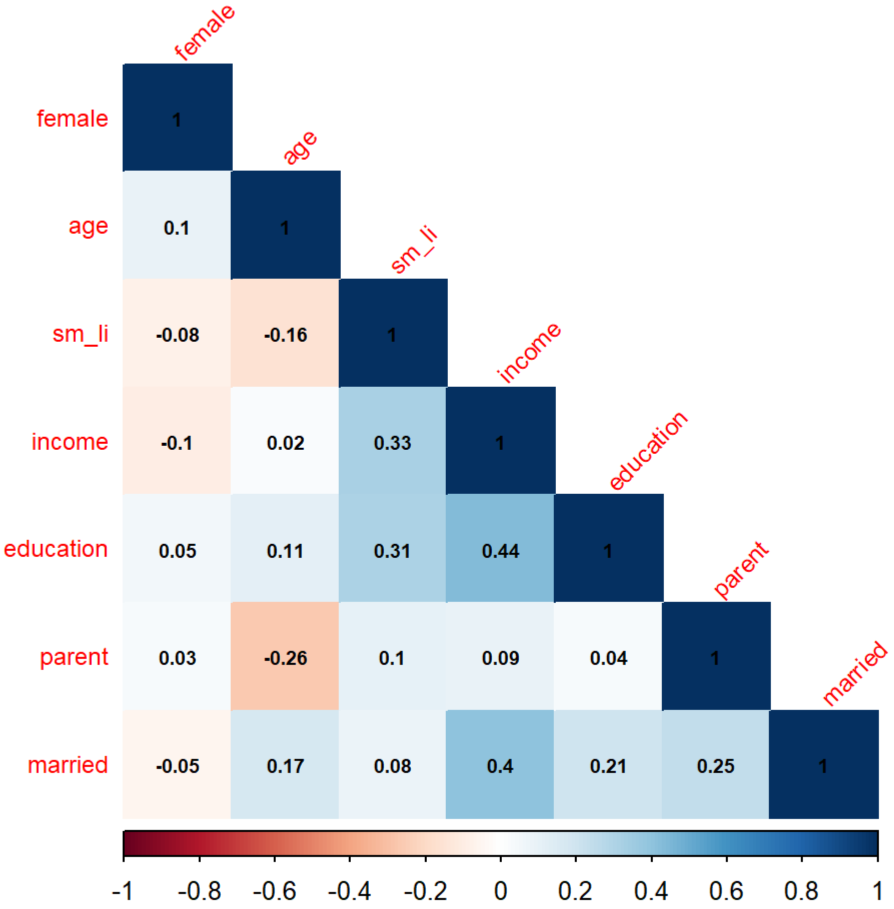
```
# Prep data for correlation plot
continuous_vars <- ss %>% select_if(is.numeric)
correlations <- cor(continuous_vars)
ss_correlations <- correlations['sm_li',]
ss_correlations_df <- data.frame(Variable = names(ss_correlations),
                                        Correlation = ss_correlations) %>%
  arrange(desc(Correlation))

# Remove columns with zero variance or too many NAs
ss_filtered_continuous_vars <- continuous_vars %>%
  select_if(function(x) var(x, na.rm = TRUE) > 0 & mean(is.na(x)) < 0.5)

# Recalculate correlations
ss_fixed_correlations <- cor(ss_filtered_continuous_vars, use = "complete.obs")

# Replace NA with 0 in correlation matrix
ss_fixed_correlations[is.na(ss_fixed_correlations)] <- 0

# Plotting correlations among continuous variables-Heatmap
corrplot(ss_fixed_correlations, method = "color", order = "hclust", addCoef.col = "black", tl.ce
x = 0.75, tl.srt = 45, type= 'lower',number.cex= 0.60)
```



```
y <- ss$sm_li
X <- ss %>% select(age, education, income, parent, married, female)
```

```r
set.seed(3125)
trainIndex <- createDataPartition(y, p = .8, list = FALSE, times = 1)
X_train <- X[trainIndex, ]
X_test <- X[-trainIndex, ]
y_train <- y[trainIndex]
y_test <- y[-trainIndex]
```

```r
model <- glm(sm_li ~ ., family = binomial, data = ss[trainIndex, ])
summary(model)
```

```
##
## Call:
## glm(formula = sm_li ~ ., family = binomial, data = ss[trainIndex,
##     ])
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.680509   0.347699  -7.709 1.27e-14 ***
## income       0.290441   0.039652   7.325 2.39e-13 ***
## education    0.301171   0.046777   6.438 1.21e-10 ***
## parent       0.012099   0.184984   0.065    0.948
## married     -0.002871   0.173112  -0.017    0.987
## female      -0.106500   0.155575  -0.685    0.494
## age         -0.029201   0.004793  -6.093 1.11e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1268.9  on 1007  degrees of freedom
## Residual deviance: 1058.7  on 1001  degrees of freedom
## AIC: 1072.7
##
## Number of Fisher Scoring iterations: 4
```

```r
y_pred <- predict(model, newdata = X_test, type = "response")
y_pred_class <- ifelse(y_pred > 0.5, 1, 0)
confusionMatrix <- table(y_test, y_pred_class)
print(confusionMatrix)
```

```
##       y_pred_class
## y_test   0   1
##      0 130  28
##      1  58  36
```

```r
# Calculating accuracy
accuracy <- sum(diag(confusionMatrix)) / sum(confusionMatrix)
print(paste('Accuracy:', accuracy))
```

```
## [1] "Accuracy: 0.658730158730159"
```

```
y_test <- factor(y_test, levels = c(0, 1))
y_pred_class <- factor(y_pred_class, levels = c(0, 1))

# Creating the confusion matrix
confusionMatrix <- table(y_test, y_pred_class)

# Now create the dataframe from the confusion matrix with names
confusionMatrix_df <- as.data.frame.matrix(confusionMatrix)
rownames(confusionMatrix_df) <- c("Actual negative", "Actual positive")
colnames(confusionMatrix_df) <- c("Predicted negative", "Predicted positive")

# Print the confusion matrix dataframe
print(confusionMatrix_df)
```

```
##                 Predicted negative Predicted positive
## Actual negative                130                 28
## Actual positive                 58                 36
```

```
precision <- confusionMatrix[2,2] / sum(confusionMatrix[2,])
recall <- confusionMatrix[2,2] / sum(confusionMatrix[,2])
f1 <- 2 * (precision * recall) / (precision + recall)

print(paste('Precision:', precision))
```

```
## [1] "Precision: 0.382978723404255"
```

```
print(paste('Recall:', recall))
```

```
## [1] "Recall: 0.5625"
```

```
print(paste('F1 Score:', f1))
```

```
## [1] "F1 Score: 0.455696202531646"
```

```
newdata <- data.frame(
  age = c(42, 82),
  education = c(7, 7),
  income = c(8, 8),
  parent = c(0, 0),
  married = c(1, 1),
  female = c(1, 1)
)
```

```
newdata$prediction_linkedin_user <- predict(model, newdata=newdata, type="response")

newdata
```

```
##   age education income parent married female prediction_linkedin_user
## 1  42         7      8      0       1      1                0.6023854
## 2  82         7      8      0       1      1                0.3202519
```

```
newdata$prediction_linkedin_user <- ifelse(newdata$prediction_linkedin_user >= 0.5, 1, 0)


print(newdata)
```

```
##   age education income parent married female prediction_linkedin_user
## 1  42         7      8      0       1      1                        1
## 2  82         7      8      0       1      1                        0
```