R Notebook

Survey selection

```
survey_3 = df_clean[df_clean$survey == "3",]
```

Variables aggregation

```
survey_3 <- survey_3 %>%
  mutate(operazione_aggregata = case_when(
    operazione %in% c(2, 3, 5, 6, 9, 10) ~ as.factor(operazione),
    TRUE ~ "Altro"
  ))
survey_3$operazione_aggregata = as.factor((survey_3$operazione_aggregata))
```

```
survey_3 <- survey_3 %>%
  mutate(regione_aggregata = case_when(
    regione_des_ana %in% c(1, 12, 3, 17, 19) ~ as.factor(regione_des_ana),
    TRUE ~ "Altro"
  ))
survey_3$regione_aggregata = as.factor((survey_3$regione_aggregata))
```

```
survey_3 <- survey_3 %>%
mutate(cs_aggregata = case_when(
    cs_abi_num_comm %in% c(1,2,3,4,5,6,7,8,9) ~ as.factor(cs_abi_num_comm),
    TRUE ~ "Altro"
  ))

survey_3$cs_abi_num_comm <- as.factor(survey_3$cs_aggregata)
```

Nas Imputation

```
survey_3 <- kNN(survey_3, variable = "risk_rating_comm")
```

Dual target variable

```r
survey_3 <- survey_3 %>%
  mutate(nps_class = case_when(
      nps < 9 ~ 0,
    nps %in% c(9, 10) ~ 1

  ))

survey_3$nps_class = as.numeric(survey_3$nps_class)
```

Data cleaning

```r
survey_3_clean_3 <- survey_3[complete.cases(survey_3[, c("nps_class", "operazione_aggregata", "fascia_et
                           c("nps_class", "operazione_aggregata", "fascia_eta_code_ana","regione_aggreg
```

Random Forest

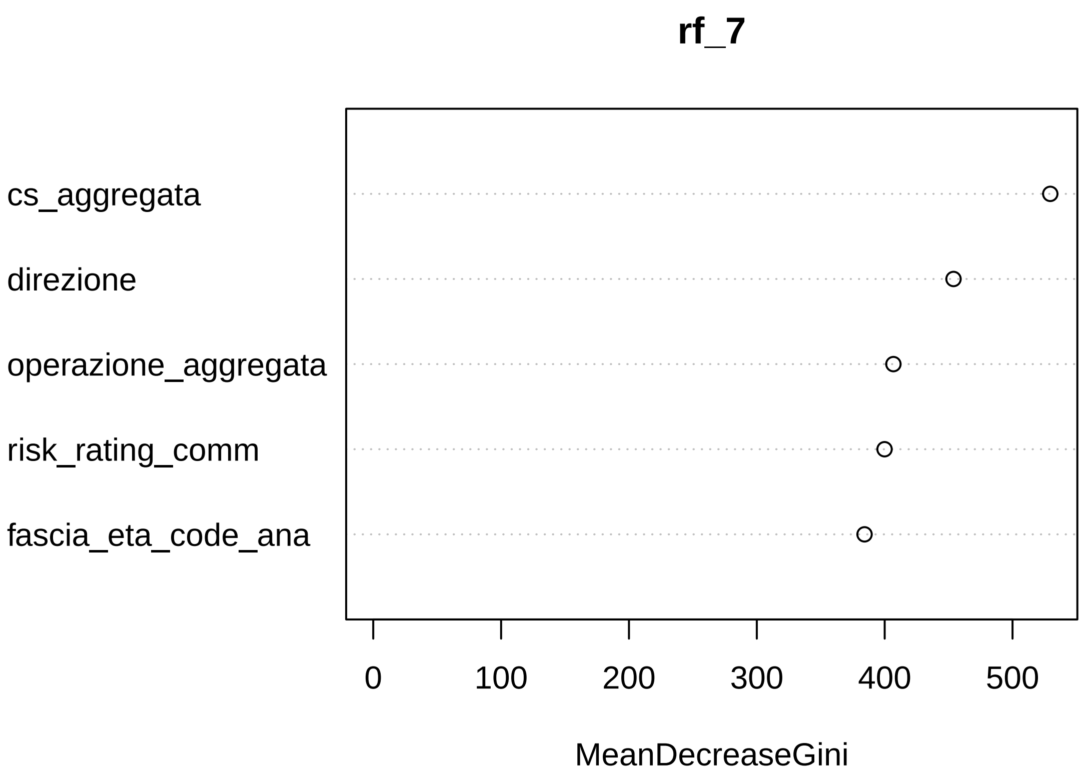
```r
set.seed(123)  # Per riproducibilità
rf_7 <- randomForest(nps_class ~ operazione_aggregata +  fascia_eta_code_ana    +   cs_aggregata +  dire

# Visualizza i risultati
print(rf_7)
```

```
##
## Call:
##  randomForest(formula = nps_class ~ operazione_aggregata + fascia_eta_code_ana +      cs_aggregata +
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 38.72%
## Confusion matrix:
##      0    1 class.error
## 0 1047 2983   0.7401985
## 1 1077 5379   0.1668216
```

```r
# Visualizza l'importanza delle variabili
importance(rf_7)
```

```
##                      MeanDecreaseGini
## operazione_aggregata         406.8350
## fascia_eta_code_ana          384.2439
## cs_aggregata                 529.5237
## direzione                    453.8514
## risk_rating_comm             399.9021
```

```r
varImpPlot(rf_7)
```



Predictions

```r
set.seed(123)  # Per riproducibilità
train_index <- sample(1:nrow(survey_3_clean_3), 0.75 * nrow(survey_3_clean_3))
train_set <- survey_3_clean_3[train_index, ]
test_set <- survey_3_clean_3[-train_index, ]



# Assicurati che nps_class sia un fattore con livelli coerenti
train_set$nps_class <- factor(train_set$nps_class, levels = c("0", "1"))
test_set$nps_class <- factor(test_set$nps_class, levels = c("0", "1"))

# Predizioni basate su soglie
thresholds <- seq(0, 1, by = 0.01)
results <- data.frame(threshold = thresholds, accuracy = NA)

library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following object is masked from 'package:colorspace':
##
##     coords

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# Prevedi le probabilità per la classe positiva
predictions_prob <- predict(rf_7, newdata = train_set, type = "prob")[, 2]

# Calcola la curva ROC
roc_curve <- roc(train_set$nps_class, predictions_prob)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
# Trova la soglia che massimizza la Youden's J statistic (TPR - FPR)
best_threshold <- coords(roc_curve, "best", ret = "threshold", best.method = "youden")

# Stampa la soglia ottimale
print(paste("treshold which maximizes the Roc - Curve:", best_threshold))
```

```
## [1] "treshold which maximizes the Roc - Curve: 0.735"
```

```r
# Calcola e stampa l'AUC
auc_value <- auc(roc_curve)
print(paste("AUC:", auc_value))
```

```
## [1] "AUC: 0.865330817406744"
```

```r
# Prevedi le probabilità per la classe positiva nel test set
predictions_prob_test <- predict(rf_7, newdata = test_set, type = "prob")[, 2]
predictions_test <- ifelse(predictions_prob_test > 0.675, "1", "0")

# Calcola la matrice di confusione per il test set
cm_test <- confusionMatrix(as.factor(predictions_test), test_set$nps_class)

# Stampa i risultati della matrice di confusione
print(cm_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 785 261
```

```
##          1  227 1349
##
##                Accuracy : 0.8139
##                  95% CI : (0.7984, 0.8286)
##     No Information Rate : 0.614
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6098
##
##  Mcnemar's Test P-Value : 0.1352
##
##             Sensitivity : 0.7757
##             Specificity : 0.8379
##          Pos Pred Value : 0.7505
##          Neg Pred Value : 0.8560
##              Prevalence : 0.3860
##          Detection Rate : 0.2994
##    Detection Prevalence : 0.3989
##       Balanced Accuracy : 0.8068
##
##        'Positive' Class : 0
##
```

```r
# Calcola l'accuratezza, precisione, richiamo e F1-score sul test set
accuracy <- cm_test$overall['Accuracy']
precision <- cm_test$byClass['Pos Pred Value']
recall <- cm_test$byClass['Sensitivity']
f1_score <- 2 * (precision * recall) / (precision + recall)
specificity <- cm_test$byClass["Specificity"]

cat("Model Evaluation Metrics on Test Set:\n",
    "Accuracy   :", round(accuracy, 4), "\n",
    "Precision  :", round(precision, 4), "\n",
    "Recall     :", round(recall, 4), "\n",
    "F1-Score   :", round(f1_score, 4), "\n",
    "Specificity:", round(specificity, 4), "\n")
```

```
## Model Evaluation Metrics on Test Set:
##  Accuracy   : 0.8139
##  Precision  : 0.7505
##  Recall     : 0.7757
##  F1-Score   : 0.7629
##  Specificity: 0.8379
```