

Data 622 Assignment 3

Keith DeNivo

Comparing Decision Trees and SVM

Comparison between decision tree based methods and Support Vector Machines.

In “decision tree ensembles to predict COVID-19 infection decision” from Amir et al 2021, tree based models were trained based off a dataset of 600 patients with 19 features consisting of age and 18 laboratory tests such as (hematocrit, hemoglobin, platelets, red blood cells, lymphocytes, leukocytes) (1). This dataset had a positive to negative class imbalance ratio of 1:6.5 (1). The decision trees had to predict the Covid-19 positive cases based on the given features in the dataset (1). For initial comparison between the different tree models, this imbalanced dataset was used. When the decision tree ensembles were compared standard C4.5 trees (standard classifiers) performed better on accuracy and precision, however for 4 performance measures (F1-measure, recall, AU-ROC, and AUPRC) decision tree ensembles for imbalanced datasets performed better (1).

Different sampling techniques were applied to the models to offset the imbalance such as: SMOTE and RUS. RUS performed better for the bagging and random forest methods (1). Further comparisons between XGBoosted, AdaBoost, and random forest models were done by changing the ratio of positives and negatives in the data. Ensemble size was also varied. Age was a significant feature that improved predictions for a majority of the ensemble methods (1). In general the Balanced random Forest with RUS sampling performed the best with an accuracy of 0.816 for the imbalanced dataset (1). It is worth noting that accuracy

is a poor metric for the imbalanced, so the emphasis should be on recall, F1, AUROC, AUPRC.

Guhathakurata S et al 2021 used Support Vector Machine models to predict between 3 classes for the target variable: Not infected, mildly infected, severely infected (2). Eight attributes: (temp, breathing rate, hypertension, Heart beat rate, acute respiratory disease syndrome, chest pain, heart disease, cough with sputum) were used as features. The dataset they used had 200 records (2). The SVM model they used had a linear kernel to better separate the data (2). The SVM model generated had accuracy of 0.87 with a high f1 score (0.97) for those who are severely infected and a lower f1 score (0.75) for those who were not infected. They also compared several different modeling algorithms. SVM outperformed kNN, Naïve Bayes, Random Forest, AdaBoost, Binary Tree (2).

The differences between the datasets were highlighted to show that Guhathakurata S et al 2021 and Amir et al 2021 to show that the models cannot be directly compared, since they do not use the same features. Amir had a binary classification and Guhathakurata had 3 classes, so there scores were not directly comparable. Though Guhathakurata S et al. 2021 does demonstrate that SVM was better than decision trees, the same optimization most likely had not been applied. It would be interesting to see how the SVM and balanced random forest (RUS) would perform side by side. Their models suggest that RF works well for a binary classification and SVM works for a multi class prediction.

The following articles were chosen to be within the scope of analytical chemistry of the environment for public health. I am an analytical chemist working for the division of environmental sciences for the Department of Health. Articles related to chemistry, the environment or public health were chosen. These are broad topics, however each of the articles directly compared a decision tree based model and Support Vector machine model.

A similar study to the previous two compares different classification models for predicting death due to COVID-19, Gharehhasani et al. 2024 compared decision trees SVM, and Adaboost (3). Surprisingly the accuracy of decision trees was higher than SVM and Adaboost (3). “The results showed that the sensitivity, specificity, accuracy and the area under the receiver operating characteristic curve were respectively 0.60, 0.68, 0.71, and 0.75 in the DT model, 0.54, 0.62, 0.63, and 0.71 in the SVM model, and 0.59, 0.65, 0.69 and 0.74 in the AdaBoost model.” (3). 8.8% of the dataset of 23504 had succumbed to COVID. Variables such as age, chronic blood disease, fever, cough, muscle pain, immune system deficiency, low level of consciousness, diarrhea, vomiting, sense of taste, abdominal pain, chronic lung disease, and neurological disorders were used as features in their models (3).

Though looking at the study’s metrics there appears to be very little difference between the decision tree model and the AdaBoost model. Suggested future work would include random forest models. The AdaBoost model potentially could be more robust and adaptable to new data due to not relying on single splits.

A study was done that predicted the amount of dissolved oxygen in estuarine (brackish water). Random forest (RF), SVM, decision trees, and regression models were compared. For

the continuous data the RF model had overall better predictive ability compared to the others, this was most likely due to the limited data. SVM was second (4).

For predicting the amount of ozone based on climate variables, RF, DT regression, and support vector regression were compared. RF had the highest R^2 (0.970), lowest RMSE. SVR performed better than DT regression but below RF (5).

For another study, water quality was classified into: excellent, acceptable, slightly polluted, polluted, and heavily polluted. The study compared data mining techniques such as: Naive Bayes, DT, k-NN, SVM, ANN, and rule-based classifiers. Support vector machines and decision tree classifiers proved to be the best classifiers because they achieved statistically valid results (6). The study also suggested that when provided a broad spectrum of water parameters, support vector machines and decision tree classifiers will provide reliable and robust results. Remarkably the decision tree model created and an accuracy and kappa equal to 1 (6).

Clathrate hydrate is the tendency to form ice like crystalline structures of water that traps gases such as methane. This occurs under specific physical conditions (pressure temperature, concentration etc.) Machine learning was implemented to model when these conditions are met and the structures are formed. This study compared decision trees and SVM. The SVM model had better predictive capabilities than decision trees. SVM outperforms DT in complex chemical prediction tasks, which reinforces their robustness. There were multiple target variables that were continuous (such as temperature, pressure and concentration), and SVM generally performs well with higher dimensional data.

Out of the 6 studies that directly compared SVM vs decision tree methods. SVM (was preferred 2 times and decision tree/ensemble decision tree methods were preferred 4 times. In multi-class classification and some environmental classification tasks: SVM frequently achieves higher headline accuracy.

Ultimately it depends on the dataset and the properties of the features and the complexity of the prediction that will determine which machine learning model will perform the best. Performance varies across a large range of applications. When looking at the regression paper RF outperformed Support vector regression, however that was only one dataset, and the performance of the model is dependent on the content of that dataset.

1. Ahmad, Amir, Safi, Ourooj, Malebary, Sharaf, Alesawi, Sami, Alkayal, Entisar, Decision Tree Ensembles to Predict Coronavirus Disease 2019 Infection: A Comparative Study, *Complexity*, 2021, 5550344, 8 pages, 2021. <https://doi.org/10.1155/2021/5550344>
2. Guhathakurata S, Kundu S, Chakraborty A, Banerjee JS. A novel approach to predict COVID-19 using support vector machine. Data Science for COVID-19. 2021:351–64. doi: 10.1016/B978-0-12-824536-1.00014-9. Epub 2021 May 21. PMID: PMC8137961.
3. gharehhasani, Bita Shokri, Mansour Rezaei, Armin Naghipour, Nazanine Sayad, Shayan Mostafaei, and Ehsan Alimohammadi. “The Most Important Variables Associated with Death Due to COVID-19 Disease, Based on Three Data Mining Models Decision Tree, AdaBoost,

and Support Vector Machine: A Cross-sectional Study." *HEALTH SCIENCE REPORTS* 7, no. 7 (2024): e2266-n/a. <https://doi.org/10.1002/hsr2.2266>.

4. Siddik, Mohammad Abu Zafer. "Application of Machine Learning Approaches in Predicting Estuarine Dissolved Oxygen (DO) under a Limited Data Environment." *Water Quality Research Journal* 57, no. 3 (2022): 140–51. <https://doi.org/10.2166/wqrj.2022.002>.
5. Balogun, Abdul-Lateef, and Abdulwaheed Tella. "Modelling and Investigating the Impacts of Climatic Variables on Ozone Concentration in Malaysia Using Correlation Analysis with Random Forest, Decision Tree Regression, Linear Regression, and Support Vector Regression." *Chemosphere (Oxford)* 299 (2022). <https://doi.org/10.1016/j.chemosphere.2022.134250>.
6. Babbar, Richa, and Sakshi Babbar. "Predicting River Water Quality Index Using Data Mining Techniques." *Environmental Earth Sciences* 76, no. 14 (2017). <https://doi.org/10.1007/s12665-017-6845-9>.
7. Hsu, Chou-Yi, Jorge Sebastian Buñay Guaman, Amit Ved, Anupam Yadav, G Ezhilarasan, A Rameshbabu, Ahmad Alkhayyat, et al. "Prediction of Methane Hydrate Equilibrium in Saline Water Solutions Based on Support Vector Machine and Decision Tree Techniques." *Scientific Reports* 15, no. 1 (2025). <https://doi.org/10.1038/s41598-025-95969-w>.

DT, RF, ADABOOST, SVM

Read in Data

```
file_url <- "https://raw.githubusercontent.com/division-zero/Data-622/refs/heads/main/assignment1.csv"
# Download the file to a temporary location
temp_file <- tempfile(fileext = ".csv")
download.file(file_url, destfile = temp_file, mode = "wb")
# Read the csv file
fulldata <- read.delim(temp_file, sep = ";", header = TRUE, stringsAsFactors = FALSE)
# View the data
head(fulldata)
```

	age	job	marital	education	default	housing	loan	contact	month
1	56	housemaid	married	basic.4y	no	no	no	telephone	may
2	57	services	married	high.school	unknown	no	no	telephone	may
3	37	services	married	high.school	no	yes	no	telephone	may
4	40	admin.	married	basic.6y	no	no	no	telephone	may

```

5 56 services married high.school      no      no yes telephone may
6 45 services married basic.9y unknown      no      no telephone may
  day_of_week duration campaign pdays previous      poutcome emp.var.rate
1     mon        261       1    999      0 nonexistent      1.1
2     mon        149       1    999      0 nonexistent      1.1
3     mon        226       1    999      0 nonexistent      1.1
4     mon        151       1    999      0 nonexistent      1.1
5     mon        307       1    999      0 nonexistent      1.1
6     mon        198       1    999      0 nonexistent      1.1
  cons.price.idx cons.conf.idx euribor3m nr.employed y
1     93.994      -36.4     4.857      5191 no
2     93.994      -36.4     4.857      5191 no
3     93.994      -36.4     4.857      5191 no
4     93.994      -36.4     4.857      5191 no
5     93.994      -36.4     4.857      5191 no
6     93.994      -36.4     4.857      5191 no

```

```

# Clean up the temporary file
unlink(temp_file)

```

```

#smaller data set for training or testing
file_url <- "https://raw.githubusercontent.com/division-zero/Data-622/refs/heads/main/assignn
# Download the file to a temporary location
temp_file <- tempfile(fileext = ".csv")
download.file(file_url, destfile = temp_file, mode = "wb")
# Read the csv file
partdata <- read.delim(file_url, sep = ";", header = TRUE, stringsAsFactors = FALSE)
# View the data
head(partdata)

```

	age	job	marital	education	default	housing	loan	contact
1	30	blue-collar	married	basic.9y	no	yes	no	cellular
2	39	services	single	high.school	no	no	no	telephone
3	25	services	married	high.school	no	yes	no	telephone
4	38	services	married	basic.9y	no	unknown	unknown	telephone
5	47	admin.	married	university.degree	no	yes	no	cellular
6	32	services	single	university.degree	no	no	no	cellular
	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate
1	may	fri	487	2	999	0	nonexistent	-1.8
2	may	fri	346	4	999	0	nonexistent	1.1
3	jun	wed	227	1	999	0	nonexistent	1.4
4	jun	fri	17	3	999	0	nonexistent	1.4

```
5 nov mon 58 1 999 0 nonexistent -0.1
6 sep thu 128 3 999 2 failure -1.1
cons.price.idx cons.conf.idx euribor3m nr.employed y
1 92.893 -46.2 1.313 5099.1 no
2 93.994 -36.4 4.855 5191.0 no
3 94.465 -41.8 4.962 5228.1 no
4 94.465 -41.8 4.959 5228.1 no
5 93.200 -42.0 4.191 5195.8 no
6 94.199 -37.5 0.884 4963.6 no
```

```
# Clean up the temporary file
unlink(temp_file)
```

Basic data info

```
#converting yes to 1 and no to 0 for binary classification
```

```
partdata$term_deposit <- ifelse(partdata$y == "yes", 1, 0)
partdata$term_deposit_factor <- factor(partdata$term_deposit, levels = c(0, 1))
```

Data cleaning

```
fulldata$term_deposit <- ifelse(fulldata$y == "yes", 1, 0)
fulldata$term_deposit_factor <- factor(fulldata$term_deposit, levels = c(0, 1))
set.seed(42)
#removing columns/features that are directly related to target variable
modeldata <- fulldata |> dplyr::select(-y, -term_deposit, -duration)
#taking the full data set and splitting 0.7:0.3 train:test split.
train_index <- sample(seq_len(nrow(modeldata)), size = 0.7 * nrow(modeldata))
train_data <- modeldata[train_index, ]
test_data <- modeldata[-train_index, ]

#train on full data set without missing features
tree_model <- rpart(
  term_deposit_factor ~ .,
  data = train_data,
  method = "class",
  control = rpart.control(
    cp = 0.01,
    minsplit = 20,
    maxdepth = 30
  )
)

#tree_model
```

Accuracy : 0.8992

sensitivity 0.9898

specificity 0.1923

kappa 0.2667

```

#random forest model
set.seed(42)
rf_model <- randomForest(
  term_deposit_factor ~ .,
  data = train_data,
  ntree = 500,
  importance = TRUE
)

forrest_pred_class <- predict(rf_model, test_data, type = "class")

#rf_model
#confusion matrix of model
#table(Predicted = forrest_pred_class, Actual = test_data$term_deposit_factor)

#mean(forrest_pred_class == test_data$term_deposit_factor)

#confusionMatrix(forrest_pred_class, test_data$term_deposit_factor)

#varImpPlot(rf_model)

```

```

#adaboost
ada_partdata <- partdata |> dplyr::select(-y, -term_deposit, -default)
ada_partdata[] <- lapply(ada_partdata, function(x) {
  if (is.character(x)) factor(x) else x
})

set.seed(42)

```

```

idx <- createDataPartition(ada_partdata$term_deposit_factor, p = 0.7, list = FALSE)
part_train_data <- ada_partdata[idx, ]
part_test_data <- ada_partdata[-idx, ]


ada_model <- boosting(
  term_deposit_factor ~ .,
  data = part_train_data,
  mfinal = 100,      # number of boosting iterations
  boos = TRUE
)

#ada_model

ada_pred <- predict(ada_model, part_test_data)

ada_pred$class <- factor(
  ada_pred$class,
  levels = levels(part_test_data$term_deposit_factor)
)

#confusionMatrix(ada_pred$class, part_test_data$term_deposit_factor)

#ada_model$importance

```

Remove features for all models

remove duration, default, loan, contact, education, employment variation rate. make the number of yes equal to the number of nos for term deposit subscription. Randomly selected term deposit row containing yes to equal the number of term deposit rows containing nos.

```

set.seed(42)
mod_data <- modeldata |> dplyr::select( -default, -loan, -contact, -education, -emp.var.rate
#removing highly correlated features

term_deposit_no <- mod_data %>% filter(term_deposit_factor == 0)
term_deposit_yes <- mod_data %>% filter(term_deposit_factor == 1)
#separating the positives and negatives

```

```

num_yes <- nrow(term_deposit_yes)

no_sample <- term_deposit_no %>% sample_n(num_yes) #sample the same number of negatives as positive

equal_data <- bind_rows(term_deposit_yes, no_sample) #a dataset with equal number of positive and negative samples

#create the training dataset and test data set based off the equal yes and no data above
eqtrain_index <- sample(seq_len(nrow(equal_data)), size = 0.7 * nrow(equal_data))
eqtrain_data <- equal_data[eqtrain_index, ]
eqtest_data <- equal_data[-eqtrain_index, ]

#making 1 to be first therefore the positive class
eqtrain_data$term_deposit_factor <- relevel(eqtrain_data$term_deposit_factor, ref = "1")
eqtest_data$term_deposit_factor <- relevel(eqtest_data$term_deposit_factor, ref = "1")

```

Train and evaluate the Models

Decision Trees

```

set.seed(42)
#decision tree model

tree_model <- rpart(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  method = "class",
  control = rpart.control(
    cp = 0.01,
    minsplit = 30,
    maxdepth = 20
  )
)

tree_model

```

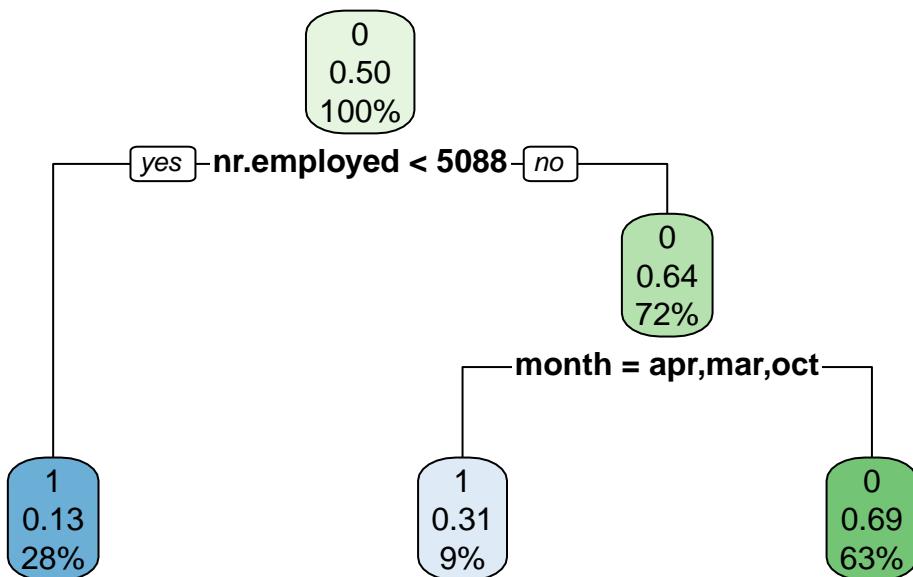
n= 6496

node), split, n, loss, yval, (yprob)

* denotes terminal node

- 1) root 6496 3246 0 (0.4996921 0.5003079)
- 2) nr.employed< 5087.65 1797 229 1 (0.8725654 0.1274346) *
- 3) nr.employed>=5087.65 4699 1678 0 (0.3570973 0.6429027)
- 6) month=apr,mar,oct 611 188 1 (0.6923077 0.3076923) *
- 7) month=aug,jul,jun,may,nov 4088 1255 0 (0.3069961 0.6930039) *

```
rpart.plot(tree_model)
```



```

pred_class <- predict(tree_model, eqtest_data, type = "class")
#pred_prob <- predict(tree_model, eqtest_data, type = "prob")

table(Predicted = pred_class, Actual = eqtest_data$term_deposit_factor)
  
```

	Actual	
Predicted	1	0
1	828	175
0	566	1215

```
mean(pred_class == eqtest_data$term_deposit_factor)
```

```
[1] 0.7338362
```

```
#confusionMatrix(pred_class$class, actual, positive = "1")
#confusion matrix contains most metrics for classifiers
cm <- confusionMatrix(pred_class, eqtest_data$term_deposit_factor)

cm
```

Confusion Matrix and Statistics

		Reference
Prediction	1	0
1	828	175
0	566	1215

Accuracy : 0.7338
95% CI : (0.717, 0.7502)

No Information Rate : 0.5007
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4679

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.5940
Specificity : 0.8741
Pos Pred Value : 0.8255
Neg Pred Value : 0.6822
Prevalence : 0.5007
Detection Rate : 0.2974
Detection Prevalence : 0.3603
Balanced Accuracy : 0.7340

'Positive' Class : 1

```
# calculate the F1 Score which is more useful than accuracy due to the focus on precision and
precision <- cm$byClass["Pos Pred Value"]
recall    <- cm$byClass["Sensitivity"]
```

```
f1 <- 2 * (precision * recall) / (precision + recall)
f1
```

Pos	Pred	Value
		0.6908636

0.7338 correct however the data is more balanced.

specificity: 0.8741 actual positives , sensitivity: 0.5940 actual negatives

kappa 0.4679

F1 0.6908636

Deeper Decision Tree

cp = 0.001 overfit

```
set.seed(42)
tree_model <- rpart(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  method = "class",
  control = rpart.control(
    cp = 0.001,
    minsplit = 30,
    maxdepth = 20
  )
)

tree_model
```

n= 6496

```
node), split, n, loss, yval, (yprob)
  * denotes terminal node
```

- 1) root 6496 3246 0 (0.4996921 0.5003079)
- 2) nr.employed< 5087.65 1797 229 1 (0.8725654 0.1274346) *
- 3) nr.employed>=5087.65 4699 1678 0 (0.3570973 0.6429027)

```

6) month=apr,mar,oct 611 188 1 (0.6923077 0.3076923)
12) job=admin.,housemaid,retired,self-employed,student,unemployed 300 49 1 (0.8366
13) job=blue-collar,entrepreneur,management,services,technician,unknown 311 139 1 (0
26) euribor3m< 1.3685 44 6 1 (0.8636364 0.1363636) *
27) euribor3m>=1.3685 267 133 1 (0.5018727 0.4981273)
54) euribor3m>=1.4905 67 15 1 (0.7761194 0.2238806) *
55) euribor3m< 1.4905 200 82 0 (0.4100000 0.5900000)
110) poutcome=nonexistent,success 145 67 0 (0.4620690 0.5379310)
220) day_of_week=thu,tue,wed 66 29 1 (0.5606061 0.4393939)
440) job=blue-collar 22 5 1 (0.7727273 0.2272727) *
441) job=entrepreneur,management,services,technician,unknown 44 20 0 (0.4
221) day_of_week=fri,mon 79 30 0 (0.3797468 0.6202532)
442) marital=single 21 8 1 (0.6190476 0.3809524) *
443) marital=divorced,married 58 17 0 (0.2931034 0.7068966) *
111) poutcome=failure 55 15 0 (0.2727273 0.7272727) *
7) month=aug,jul,jun,may,nov 4088 1255 0 (0.3069961 0.6930039)
14) cons.conf.idx< -42.35 1658 653 0 (0.3938480 0.6061520)
28) pdays< 505.5 57 16 1 (0.7192982 0.2807018) *
29) pdays>=505.5 1601 612 0 (0.3822611 0.6177389)
58) euribor3m< 1.2755 294 144 0 (0.4897959 0.5102041)
116) campaign< 5.5 275 133 1 (0.5163636 0.4836364)
232) day_of_week=fri,mon,tue,wed 201 88 1 (0.5621891 0.4378109)
464) euribor3m>=1.2545 65 17 1 (0.7384615 0.2615385) *
465) euribor3m< 1.2545 136 65 0 (0.4779412 0.5220588)
930) job=admin.,blue-collar,entrepreneur,self-employed,technician,unemploy
1860) campaign< 1.5 40 16 1 (0.6000000 0.4000000) *
1861) campaign>=1.5 65 31 0 (0.4769231 0.5230769)
3722) day_of_week=fri 42 18 1 (0.5714286 0.4285714) *
3723) day_of_week=mon 23 7 0 (0.3043478 0.6956522) *
931) job=housemaid,management,retired,services,student 31 10 0 (0.32258
233) day_of_week=thu 74 29 0 (0.3918919 0.6081081) *
117) campaign>=5.5 19 2 0 (0.1052632 0.8947368) *
59) euribor3m>=1.2755 1307 468 0 (0.3580719 0.6419281)
118) day_of_week=fri,mon,tue,wed 1060 400 0 (0.3773585 0.6226415)
236) previous< 0.5 926 363 0 (0.3920086 0.6079914)
472) month=may 294 131 0 (0.4455782 0.5544218)
944) euribor3m>=1.349 31 6 1 (0.8064516 0.1935484) *
945) euribor3m< 1.349 263 106 0 (0.4030418 0.5969582)
1890) job=entrepreneur,management,retired,self-employed,technician,unemp
1891) job=admin.,blue-collar,housemaid,services,student 204 73 0 (0.35
473) month=jul 632 232 0 (0.3670886 0.6329114)
946) job=admin.,entrepreneur,management,retired,self-employed,services 35
1892) euribor3m>=4.9625 68 29 1 (0.5735294 0.4264706)

```

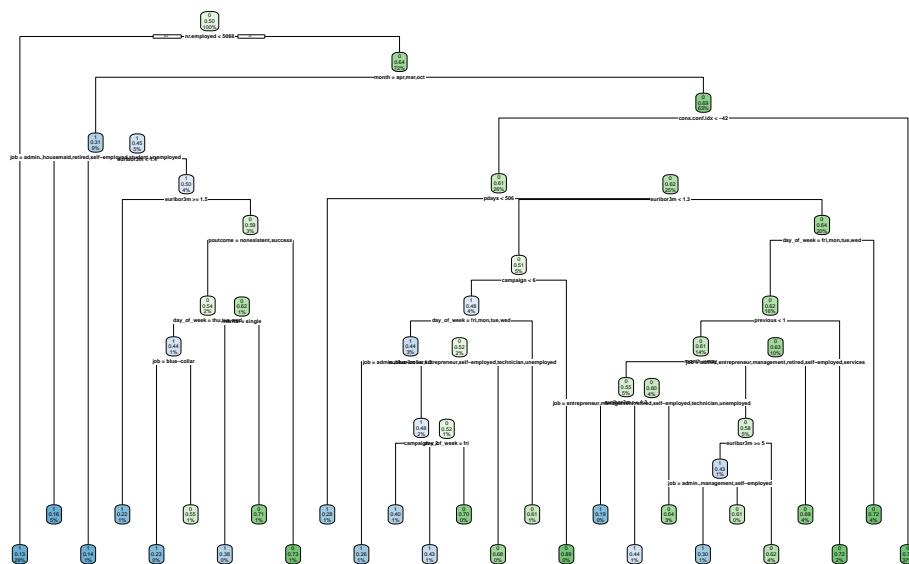
```

3784) job=admin.,management,self-employed 40 12 1 (0.7000000 0.3000000
3785) job=entrepreneur,retired,services 28 11 0 (0.3928571 0.6071429)
1893) euribor3m< 4.9625 283 107 0 (0.3780919 0.6219081) *
947) job=blue-collar,housemaid,student,technician,unemployed,unknown 281
237) previous>=0.5 134 37 0 (0.2761194 0.7238806) *
119) day_of_week=thu 247 68 0 (0.2753036 0.7246964) *
15) cons.conf.idx>=-42.35 2430 602 0 (0.2477366 0.7522634) *

```

```
rpart.plot(tree_model)
```

Warning: labs do not fit even at cex 0.15, there may be some overplotting



```
pred_class <- predict(tree_model, eqtest_data, type = "class")  
#pred_prob <- predict(tree_model, eqtest_data, type = "prob")
```

```
table(Predicted = pred_class, Actual = eqtest_data$term_deposit_factor)
```

	Predicted	Actual
--	-----------	--------

```
1 876 186  
0 518 1204
```

```
mean(pred_class == eqtest_data$term_deposit_factor)
```

```
[1] 0.7471264
```

```
#confusionMatrix(pred_class$class, actual, positive = "1")  
cm <- confusionMatrix(pred_class, eqtest_data$term_deposit_factor)  
cm
```

Confusion Matrix and Statistics

		Reference
Prediction	1	0
1	876	186
0	518	1204

Accuracy : 0.7471
95% CI : (0.7305, 0.7632)

No Information Rate : 0.5007

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4944

McNemar's Test P-Value : < 2.2e-16

Sensitivity : 0.6284

Specificity : 0.8662

Pos Pred Value : 0.8249

Neg Pred Value : 0.6992

Prevalence : 0.5007

Detection Rate : 0.3147

Detection Prevalence : 0.3815

Balanced Accuracy : 0.7473

'Positive' Class : 1

```

precision <- cm$byClass["Pos Pred Value"]
recall     <- cm$byClass["Sensitivity"]

f1_tree <- 2 * (precision * recall) / (precision + recall)
f1_tree

```

```

Pos Pred Value
0.713355

```

performs slightly better 0.7471 accuracy

specificity: 0.8662

sensitivity 0.6284

kappa 0.4944

F1 0.713355

Random forest

```

set.seed(42)
#random forest model

rf_model <- randomForest(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  #method = "class"
  ntree = 500,
  importance = TRUE
)

forrest_pred_class <- predict(rf_model, eqtest_data, type = "class")
forrest_pred_prob <- predict(rf_model, eqtest_data, type = "prob")

rf_model

```

```

Call:
randomForest(formula = term_deposit_factor ~ ., data = eqtrain_data,      ntree = 500, impo
              Type of random forest: classification
              Number of trees: 500
No. of variables tried at each split: 3

              OOB estimate of  error rate: 24.78%
Confusion matrix:
             1   0 class.error
1 2063 1183  0.3644486
0  427 2823  0.1313846







          Actual
Predicted    1    0
           1 864 170
           0 530 1220

mean(forrest_pred_class == eqtest_data$term_deposit_factor)

[1] 0.7485632

cm <- confusionMatrix(forrest_pred_class, eqtest_data$term_deposit_factor)

cm

Confusion Matrix and Statistics

          Reference
Prediction    1    0
           1 864 170
           0 530 1220

Accuracy : 0.7486
95% CI : (0.732, 0.7646)
No Information Rate : 0.5007
P-Value [Acc > NIR] : < 2.2e-16

```

```
Kappa : 0.4973

McNemar's Test P-Value : < 2.2e-16

Sensitivity : 0.6198
Specificity : 0.8777
Pos Pred Value : 0.8356
Neg Pred Value : 0.6971
Prevalence : 0.5007
Detection Rate : 0.3103
Detection Prevalence : 0.3714
Balanced Accuracy : 0.7487

'Positive' Class : 1
```

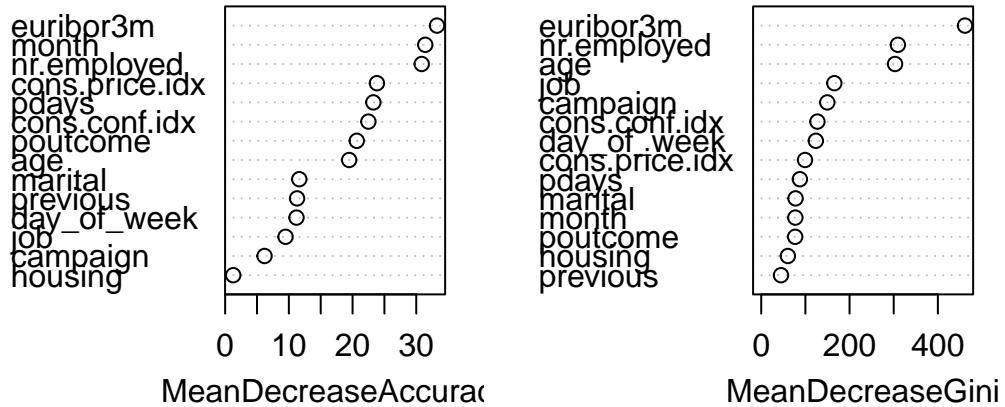
```
precision <- cm$byClass["Pos Pred Value"]
recall     <- cm$byClass["Sensitivity"]

f1_rf <- 2 * (precision * recall) / (precision + recall)
f1_rf
```

```
Pos Pred Value
0.7116969
```

```
varImpPlot(rf_model)
```

rf_model



surprisingly model performance is roughly equal to just one decision tree

kappa 0.4894, sensitivity 0.8791, specificity 0.6105 accuracy 0.7446

F1 0.7116969

Accuracy : 0.7453, Kappa : 0.4909, Sensitivity : 0.8827, Specificity : 0.6083
specificity decreased

Tuned Random Forest

```
yn_eqtrain_data <- eqtrain_data
yn_eqtest_data <- eqtest_data

set.seed(42)
yn_eqtrain_data$term_deposit_factor <- factor(
  ifelse(yn_eqtrain_data$term_deposit_factor == 1, "yes", "no"),
```

```

  levels = c("yes", "no")    # convert back to yes and no
)

yn_eqtest_data$term_deposit_factor <- factor(
  ifelse(yn_eqtest_data$term_deposit_factor == 1, "yes", "no"),
  levels = c("yes", "no")    # convert back to yes and no
)

tuned_rf <- train(
  term_deposit_factor ~ .,
  data = yn_eqtrain_data,
  method = "rf",
  trControl = trainControl(
    method = "cv",
    number = 5,
    classProbs = TRUE,
    summaryFunction = twoClassSummary
  ),
  metric = "ROC",
  tuneLength = 5
)

tuned_rf

```

Random Forest

6496 samples
 14 predictor
 2 classes: 'yes', 'no'

No pre-processing
 Resampling: Cross-Validated (5 fold)
 Summary of sample sizes: 5197, 5197, 5196, 5197, 5197
 Resampling results across tuning parameters:

mtry	ROC	Sens	Spec
2	0.7881693	0.6179917	0.8827692
11	0.7923191	0.6789831	0.7981538
20	0.7871650	0.6885338	0.7775385
29	0.7857010	0.6879185	0.7723077
39	0.7840828	0.6879194	0.7636923

ROC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 11.

```
varImp(tuned_rf)
```

rf variable importance

only 20 most important variables shown (out of 39)

	Overall
euribor3m	100.000
age	86.084
nr.employed	60.121
campaign	42.624
cons.conf.idx	19.935
housingyes	15.852
cons.price.idx	15.289
pdays	13.900
jobblue-collar	11.337
maritalmarried	11.082
jobtechnician	10.417
poutcomesuccess	9.626
day_of_weekthu	9.385
day_of_weekwed	9.291
maritalsingle	9.241
day_of_weekmon	9.073
day_of_weektue	8.848
jobservices	8.261
jobmanagement	7.271
previous	7.017

```
forrest_pred_class <- predict(tuned_rf, yn_eqtest_data, type = "raw")

#table(Predicted = forrest_pred_class, Actual = yn_eqtest_data$term_deposit_factor)

#mean(forrest_pred_class == yn_eqtest_data$term_deposit_factor)

cm <- confusionMatrix(forrest_pred_class, yn_eqtest_data$term_deposit_factor)
cm
```

Confusion Matrix and Statistics

```

    Reference
Prediction yes   no
      yes  922  253
      no   472 1137

    Accuracy : 0.7396
    95% CI  : (0.7229, 0.7558)
No Information Rate : 0.5007
P-Value [Acc > NIR] : < 2.2e-16

    Kappa : 0.4793

McNemar's Test P-Value : 5.665e-16

    Sensitivity : 0.6614
    Specificity  : 0.8180
    Pos Pred Value : 0.7847
    Neg Pred Value : 0.7067
    Prevalence    : 0.5007
    Detection Rate : 0.3312
    Detection Prevalence : 0.4221
    Balanced Accuracy : 0.7397

    'Positive' Class : yes

```

```

precision <- cm$byClass["Pos Pred Value"]
recall    <- cm$byClass["Sensitivity"]

f1_tune_rf <- 2 * (precision * recall) / (precision + recall)
f1_tune_rf

```

```

Pos Pred Value
0.717789

```

```

head(tuned_rf$resample)

```

```

ROC      Sens      Spec Resample
1 0.7831563 0.6671803 0.7861538      Fold1

```

```
2 0.7810454 0.6779661 0.7830769      Fold5
3 0.7937253 0.6748844 0.8015385      Fold2
4 0.8100473 0.7000000 0.8153846      Fold3
5 0.7936210 0.6748844 0.8046154      Fold4
```

```
sd(tuned_rf$resample$ROC)
```

```
[1] 0.01150017
```

```
sd(tuned_rf$resample$Kappa)
```

```
[1] NA
```

```
sd(tuned_rf$resample$accuracy)
```

```
[1] NA
```

```
Accuracy : 0.7478
Kappa : 0.4958
Sensitivity : 0.6413          Specificity : 0.8547
specificity improved and sensitivity decreased. F1: 0.717789
```

Adaboost

```
#adaptive boosting model

#ada_partdata <- partdata |> dplyr::select(-y, -term_deposit, -default)
#ada_partdata[] <- lapply(ada_partdata, function(x) {
#  if (is.character(x)) factor(x) else x
#})

##ada_partdata$default <- factor(ada_partdata$default, levels = c("no", "yes", "unknown"))

set.seed(42)
```

```

ada_model <- boosting(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  mfinal = 100,      # number of boosting iterations
  boos = TRUE
)

#ada_model

ada_pred <- predict(ada_model, eqtest_data)

ada_pred$class <- factor(
  ada_pred$class,
  levels = levels(eqtest_data$term_deposit_factor)
)

cm <- confusionMatrix(ada_pred$class, eqtest_data$term_deposit_factor)

cm

```

Confusion Matrix and Statistics

		Reference	
Prediction	1	0	
1	891	195	
0	503	1195	

Accuracy : 0.7493
 95% CI : (0.7327, 0.7653)
 No Information Rate : 0.5007
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4987

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.6392
 Specificity : 0.8597
 Pos Pred Value : 0.8204
 Neg Pred Value : 0.7038
 Prevalence : 0.5007

```
Detection Rate : 0.3200  
Detection Prevalence : 0.3901  
Balanced Accuracy : 0.7494
```

```
'Positive' Class : 1
```

```
precision <- cm$byClass["Pos Pred Value"]  
recall    <- cm$byClass["Sensitivity"]  
  
f1_ada <- 2 * (precision * recall) / (precision + recall)  
f1_ada
```

```
Pos Pred Value  
0.7185484
```

```
sort(ada_model$importance, decreasing = TRUE)
```

	nr.employed	pdays	euribor3m	month	poutcome
43.1271088	16.3860165	15.7885071	12.5864874	4.8985520	
job	cons.price.idx	age	campaign	day_of_week	
2.1103654	2.0172030	1.2179434	0.5869548	0.5002386	
previous	cons.conf.idx	housing	marital		
0.3590088	0.1722818	0.1329486	0.1163839		

```
F1: 0.717789
```

```
Accuracy : 0.7493
```

```
Kappa : 0.4987
```

```
Sensitivity : 0.8597          Specificity : 0.6392
```

SVM: Radial Basis kernel

```

set.seed(42)
#SVM model radial basis kernel

SVM_rbf_model <- ksvm(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  kernel = "rbfdot"
)

SVM_rbf_model

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)
parameter : cost C = 1

Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.0757110179780849

Number of Support Vectors : 3986

Objective Function Value : -3303.13
Training error : 0.245844

rbf_pred_class <- predict(SVM_rbf_model, eqtest_data, type = "response")

#table(Predicted = rbf_pred_class, Actual = eqtest_data$term_deposit_factor)

cm <- confusionMatrix(rbf_pred_class, eqtest_data$term_deposit_factor)

cm

```

Confusion Matrix and Statistics

```

    Reference
Prediction   1     0
           1 840 169
           0 554 1221

    Accuracy : 0.7403
    95% CI  : (0.7236, 0.7565)
No Information Rate : 0.5007
P-Value [Acc > NIR] : < 2.2e-16

    Kappa : 0.4808

McNemar's Test P-Value : < 2.2e-16

    Sensitivity : 0.6026
    Specificity : 0.8784
Pos Pred Value : 0.8325
Neg Pred Value : 0.6879
    Prevalence : 0.5007
    Detection Rate : 0.3017
Detection Prevalence : 0.3624
Balanced Accuracy : 0.7405

'Positive' Class : 1

```

```

#calculate f1 from precision and recall
precision <- cm$byClass["Pos Pred Value"]
recall    <- cm$byClass["Sensitivity"]

f1 <- 2 * (precision * recall) / (precision + recall)
f1

```

```

Pos Pred Value
0.6991261

```

radial f1 0.69912161

SVM: Linear kernel

```
set.seed(42)
#SVM model

SVM_lin_model <- ksvm(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  kernel = "vanilladot"
)
```

Setting default kernel parameters

```
SVM_lin_model
```

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)
parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 3758

Objective Function Value : -3521.749

Training error : 0.254618

```
lin_pred_class <- predict(SVM_lin_model, eqtest_data, type = "response")

#table(Predicted = rbf_pred_class, Actual = eqtest_data$term_deposit_factor)

cm <- confusionMatrix(lin_pred_class, eqtest_data$term_deposit_factor)

cm
```

Confusion Matrix and Statistics

		Reference
Prediction	1	0
1	843	180
0	551	1210

Accuracy : 0.7374
95% CI : (0.7207, 0.7537)
No Information Rate : 0.5007
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4751

McNemar's Test P-Value : < 2.2e-16

Sensitivity : 0.6047
Specificity : 0.8705
Pos Pred Value : 0.8240
Neg Pred Value : 0.6871
Prevalence : 0.5007
Detection Rate : 0.3028
Detection Prevalence : 0.3675
Balanced Accuracy : 0.7376

'Positive' Class : 1

```
#calculate f1 from precision and recall
precision <- cm$byClass["Pos Pred Value"]
recall   <- cm$byClass["Sensitivity"]

f1 <- 2 * (precision * recall) / (precision + recall)
f1
```

Pos Pred Value
0.697559

linear f1 0.697559

SVM: hyperbolic tangent sigmoid kernel

```
set.seed(42)
#SVM model

SVM_tanh_model <- ksvm(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  kernel = "tanhdot"
)
```

Setting default kernel parameters

```
SVM_tanh_model
```

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)
parameter : cost C = 1

Hyperbolic Tangent kernel function.
Hyperparameters : scale = 1 offset = 1

Number of Support Vectors : 2884

Objective Function Value : -440844.2
Training error : 0.443196

```
tanh_pred_class <- predict(SVM_tanh_model, eqtest_data, type = "response")

#table(Predicted = rbf_pred_class, Actual = eqtest_data$term_deposit_factor)

cm <- confusionMatrix(tanh_pred_class, eqtest_data$term_deposit_factor)
```

```
cm
```

Confusion Matrix and Statistics

```
    Reference
Prediction   1   0
             1 773 615
             0 621 775

Accuracy : 0.556
95% CI  : (0.5373, 0.5746)
No Information Rate : 0.5007
P-Value [Acc > NIR] : 2.865e-09

Kappa : 0.1121
```

McNemar's Test P-Value : 0.8869

```
Sensitivity : 0.5545
Specificity : 0.5576
Pos Pred Value : 0.5569
Neg Pred Value : 0.5552
Prevalence : 0.5007
Detection Rate : 0.2777
Detection Prevalence : 0.4986
Balanced Accuracy : 0.5560
```

'Positive' Class : 1

```
#calculate f1 from precision and recall
precision <- cm$byClass["Pos Pred Value"]
recall   <- cm$byClass["Sensitivity"]

f1 <- 2 * (precision * recall) / (precision + recall)
f1
```

```
Pos Pred Value
0.5557153
```

tanh f1 0.5557

radial is the best

Cost Parameters SVM

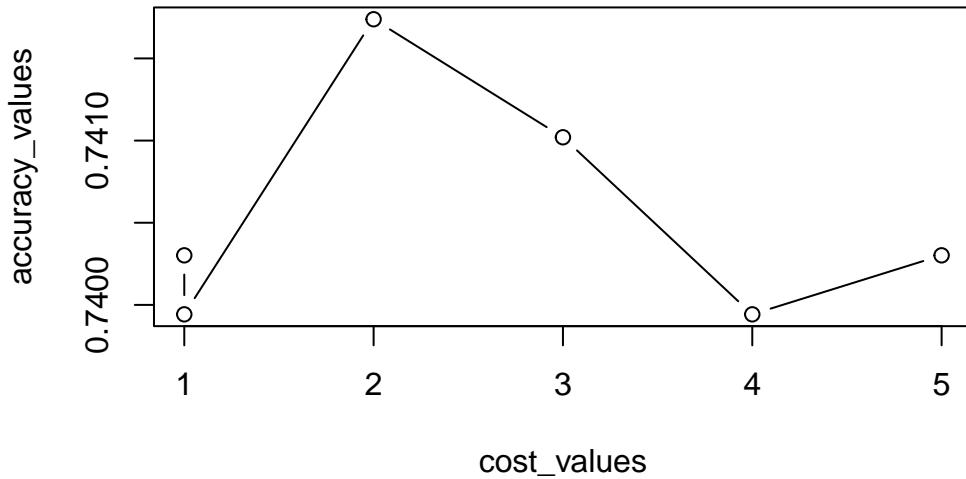
Cost Parameter based on accuracy

```
set.seed(42)

cost_values <- c(1, seq(from=1, to = 5, by = 1))
accuracy_values <- sapply(cost_values, function(x){
  SVM_rbf_model <- ksvm(
    term_deposit_factor ~ .,
    data = eqtrain_data,
    kernel = "rbfdot", C = x
  )
  rbf_pred <- predict(SVM_rbf_model, eqtest_data, type = "response")
  agree <- ifelse(rbf_pred == eqtest_data$term_deposit_factor, 1, 0)
  accuracy <- sum(agree) / nrow(eqtest_data)

  return (accuracy)
})

plot(cost_values, accuracy_values, type = "b")
```



Cost parameter of 2 was shown to have the highest accuracy

```

cost_values <- c(1, seq(from=1, to = 5, by = 1))
f1_values <- sapply(cost_values, function(x){
  SVM_rbf_model <- ksvm(
    term_deposit_factor ~ .,
    data = eqtrain_data,
    kernel = "rbfdot", C = x
  )
  rbf_pred <- predict(SVM_rbf_model, eqtest_data, type = "response")
  #agree <- ifelse(rbf_pred == eqtest_data$term_deposit_factor, 1, 0)
  #accuracy <- sum(agree) / nrow(eqtest_data)

  cm <- confusionMatrix(rbf_pred, eqtest_data$term_deposit_factor)

#cm

#calculate f1 from precision and recall
precision <- cm$byClass["Pos Pred Value"]
recall     <- cm$byClass["Sensitivity"]

f1 <- 2 * (precision * recall) / (precision + recall)

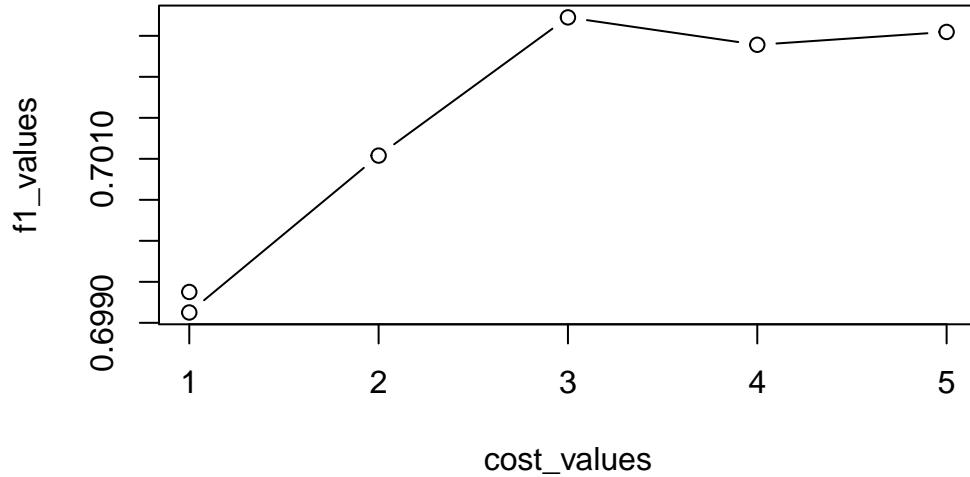
```

```

    return (f1)
})

plot(cost_values, f1_values, type = "b")

```



cost parameter of 3 had the highest F1 score. This was used as the best SVM model to compare to the other models.

```

set.seed(42)
#SVM model

SVM_rbf_model <- ksvm(
  term_deposit_factor ~ .,
  data = eqtrain_data,
  kernel = "rbfdot", C = 3
)
SVM_rbf_model

```

Support Vector Machine object of class "ksvm"

```

SV type: C-svc (classification)
parameter : cost C = 3

Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.0757110179780849

Number of Support Vectors : 3994

Objective Function Value : -9560.548
Training error : 0.23476

rbf_pred_class <- predict(SVM_rbf_model, eqtest_data, type = "response")

#table(Predicted = rbf_pred_class, Actual = eqtest_data$term_deposit_factor)

cm <- confusionMatrix(rbf_pred_class, eqtest_data$term_deposit_factor)

cm

```

Confusion Matrix and Statistics

		Reference	
Prediction	1	0	
1	852	177	
0	542	1213	

Accuracy : 0.7417
 95% CI : (0.7251, 0.7579)
 No Information Rate : 0.5007
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4837

McNemar's Test P-Value : < 2.2e-16

Sensitivity : 0.6112

```

    Specificity : 0.8727
    Pos Pred Value : 0.8280
    Neg Pred Value : 0.6912
        Prevalence : 0.5007
    Detection Rate : 0.3060
Detection Prevalence : 0.3696
Balanced Accuracy : 0.7419

'Positive' Class : 1

```

```

#calculate f1 from precision and recall
precision <- cm$byClass["Pos Pred Value"]
recall     <- cm$byClass["Sensitivity"]

f1_svm <- 2 * (precision * recall) / (precision + recall)
f1_svm

```

```

Pos Pred Value
0.7032604

```

For comparison of the models, all were trained and tested on the same dataset. The dataset had equal number of randomly selected subscribers and all nonsubscribers. I removed duration, default, loan, contact, education, and employment variation rate from the dataset. This was then split 3:7 for testing to training. F1 score was used primarily to determine the performance of the models. The F1 score balances precision and recall which is important for imbalanced data.

Tested several SVM kernels, compared their F1 score:

Linear

SVM kernel	F1 score
Radial basis	0.6991261
Linear	0.697559
Hyperbolic tangent sigmoid	0.5557153

Radial was the selected kernel for SVM due to having the highest F1 score. Then the cost parameter was optimized for accuracy. The cost parameter changes the width of the decision boundary and controls the penalty for misclassification. A cost value of 3 was found to maximize the F1 score. So, it was used for training the SVM model.

A comparison of the metrics for each of the previously trained and the optimized SVM model.

Model

Accuracy

Sensitivity

Specificity

kappa

F1

2 depth tree

0.7338

0.5940

0.8741

0.4679

0.6908636

~12 depth tree

0.7471

0.6284

0.8662

0.4944

0.713355

Random Forest

0.7486

0.6198

0.8777

0.4973

0.7116969

Tuned CV RF

0.717789
0.6614
0.8180
0.4793
0.717789
Adaptive boosting
0.7493
0.6392
0.8597
0.4987
0.7185484
SVM
0.7417
0.6112
0.8727
0.4837
0.7032604

The top three models with the highest F1 score are: Adaptive Boosting, Tuned cross validated Random Forest, Deep Decision tree. The adaptive boosting model also had the highest kappa indicating that its predictions are moderately better than random guessing and had the strongest agreement with the true labels beyond chance. The SVM may have performed better if it had a multi class prediction. How complex the data is, what features are used, and the content of the data makes a big difference in the performance of the models. Depending on how the data is preprocessed many different outcomes could occur. The models perform very similarly, which may indicate limitations or problems with the data itself. The features may not provide clear separability between the classes, or it has too much noise causing a performance ceiling that the algorithms cannot overcome.

reference:

Input variables: # bank client data: 1 - **age** (numeric) 2 - **job : type of job** (categorical: “admin.”,“blue-collar”,“entrepreneur”,“housemaid”,“management”,“retired”,“self-employed”,“services”,“student”,“technician”,“unemployed”,“unknown”) 3 - **marital : marital status** (categorical: “divorced”,“married”,“single”,“unknown”; note: “divorced” means divorced or widowed) 4 - **education** (categorical: “basic.4y”,“basic.6y”,“basic.9y”,“high.school”,“illiterate”,“profes-

5 - **default: has credit in default?** (categorical: “no”,“yes”,“unknown”) 6 - **housing: has housing loan?** (categorical: “no”,“yes”,“unknown”) 7 - **loan: has personal loan?** (categorical: “no”,“yes”,“unknown”) # related with the last contact of the current campaign: 8 - **contact: contact communication type** (categorical: “cellular”,“telephone”) 9 - **month: last contact month of year** (categorical: “jan”, “feb”, “mar”, ..., “nov”, “dec”) 10 - **day_of_week: last contact day of the week** (categorical: “mon”,“tue”,“wed”,“thu”,“fri”) 11 - **duration: last contact duration, in seconds (numeric)**. Important note: this attribute highly affects the output target (e.g., if duration=0 then y=“no”). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model. # other attributes: 12 - **campaign: number of contacts performed during this campaign and for this client** (numeric, includes last contact) 13 - **pdays: number of days that passed by after the client was last contacted from a previous campaign** (numeric; 999 means client was not previously contacted) 14 - **previous: number of contacts performed before this campaign and for this client** (numeric) 15 - **poutcome: outcome of the previous marketing campaign** (categorical: “failure”,“nonexistent”,“success”) # social and economic context attributes 16 - **emp.var.rate: employment variation rate** - quarterly indicator (numeric) 17 - **cons.price.idx: consumer price index** - monthly indicator (numeric) 18 - **cons.conf.idx: consumer confidence index** - monthly indicator (numeric) 19 - **euribor3m: euribor 3 month rate** - daily indicator (numeric) 20 - **nr.employed: number of employees** - quarterly indicator (numeric)

Output variable (desired target): 21 - **y** - has the client subscribed a term deposit? (binary: “yes”,“no”)