Lista 5

César A. Galvão - 190011572

Gabriela Carneiro - 180120816

João Vitor Vasconcelos - 170126064

Kevyn Andrade de Souza - 190015853

Índice

Questão 12

Revise as notas de aula e estude o Capítulo 9 de James et al. (with Applications in R ou with Applications in Python), disponível em https://www.statlearning.com/. Resolva os exercícios deste capítulo.

Questão 1

This problem involves hyperplanes in two dimensions.

Item a)

Sketch the hyperplane $1 + 3X_1 - X_2 = 0$. Indicate the set of points for which $1 + 3X_1 - X_2 > 0$, as well as the set of points for which $1 + 3X_1 - X_2 < 0$.

A Figura 1 mostra o hiperplano indicado, assim como as curvas de nível. Os pontos em que o valor da função que gera o hiperplano é maior que zero estão à direita da curva de nível com valor 0 e os demais estão à esquerda.

```
data <- data.frame(
  x1 = seq(-5, 5, length.out = 100),
  x2 = seq(-5, 5, length.out = 100)
)

ggplot(data, aes(x = x1, y = x2)) +
  geom_abline(intercept = 1, slope = 3, color = "blue") +
  geom_ribbon(aes(ymin = -Inf, ymax = (3*x1 + 1)), xmin = min(data$x1), xmax = Inf,
  ↪  fill = "blue", alpha = 0.2) +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_continuous(expand = c(0,0)) +
  labs(x = TeX("$x_1$"), y = TeX("$x_2$"))+
  annotate("text", x = 2, y = -5, label = "y > 0", color = "red", size = 5) +
  theme_classic()+
  theme(panel.grid.major = element_line(),
        axis.title.y = element_text(angle = 0, vjust = 0.5))
```

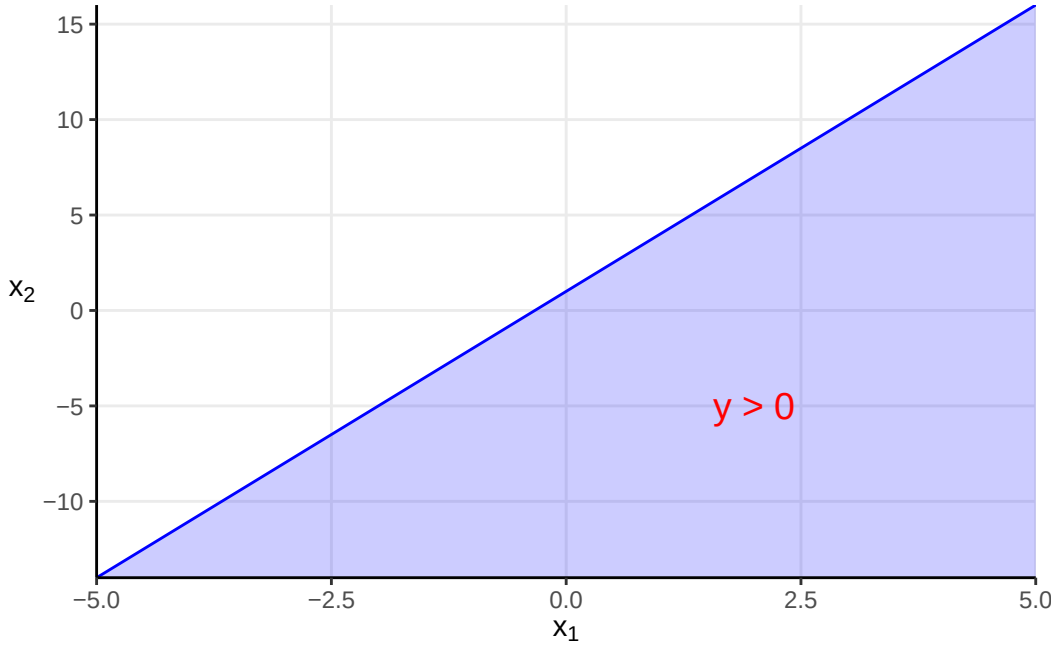


Figura 1: Hiperplano $1 + 3X_1 - X_2 = 0$

Item b)

On the same plot, sketch the hyperplane $-2 + X_1 + 2X_2 = 0$. Indicate the set of points for which $-2 + X_1 + 2X_2 > 0$, as well as the set of points for which $-2 + X_1 + 2X_2 < 0$.

A Figura 2 mostra o hiperplano indicado da mesma forma que no item anterior.

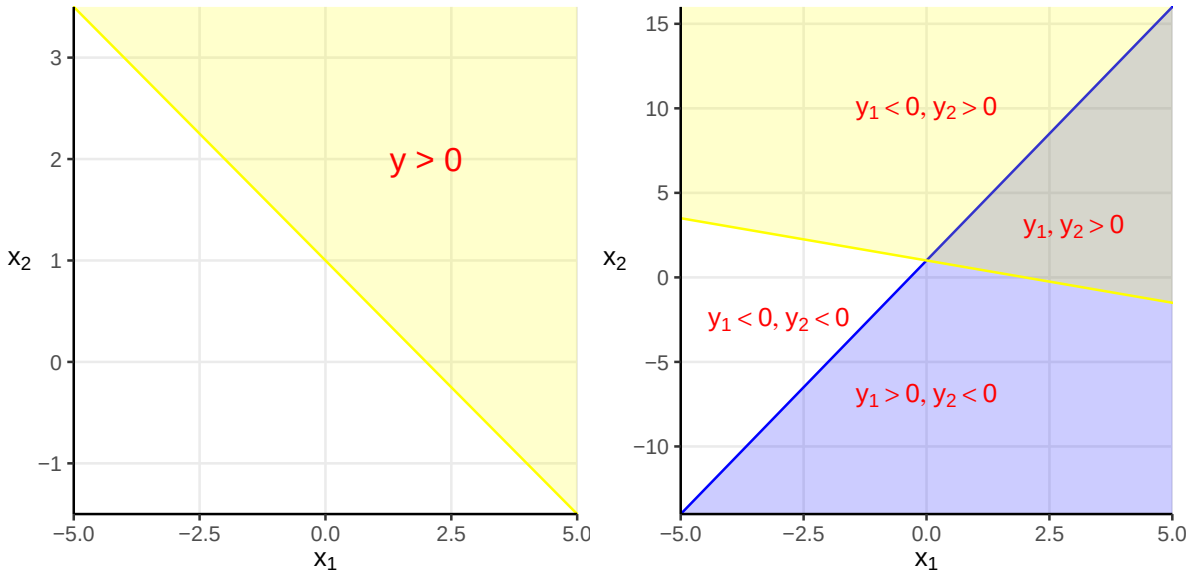


Figura 2: Hiperplano $-2 + X_1 + 2X_2 = 0$ e interseções entre os planos

Questão 4

Generate a simulated two-class data set with 100 observations and two features in which there is a visible but non-linear separation between the two classes. Show that in this setting, a support vector machine with a polynomial kernel (with degree greater than 1) or a radial kernel will outperform a support vector classifier on the training data. Which technique performs best on the test data? Make plots and report training and test error rates in order to back up your assertions.

Os dados são gerados a seguir, seguidos de um gráfico mostrando a separação entre as classes.

```
set.seed (1)
x <- matrix(rnorm (100 * 2), ncol = 2)
x[1:25, ] <- x[1:25, ] + 2
x[76:100, ] <- x[76:100, ] - 2
y <- c(rep(1, 25), rep(2, 50), rep(1, 25))
dat <- data.frame(x = x, y = as.factor(y))

plot(x, col = y)
```

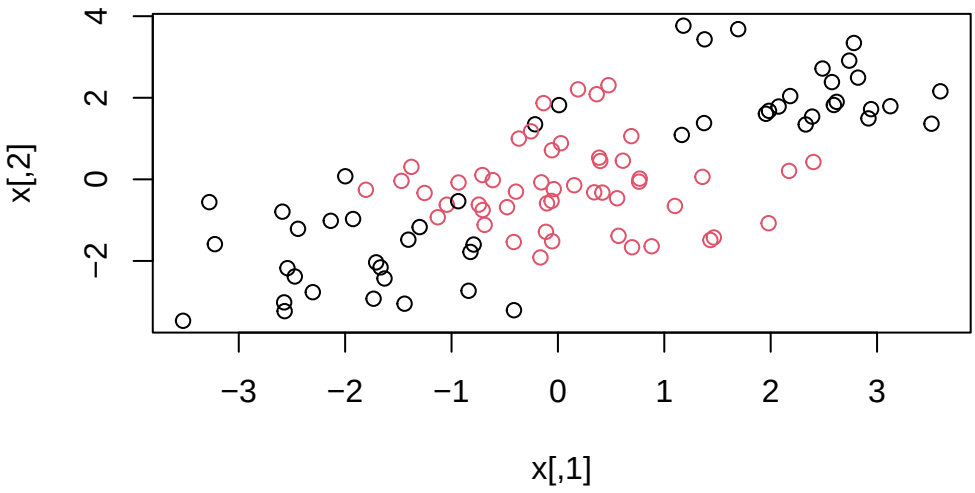
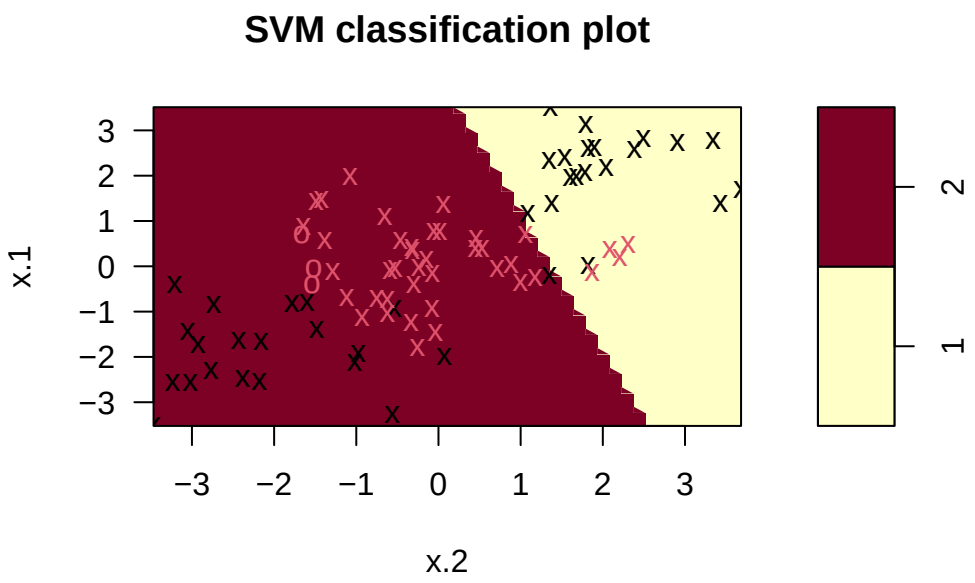


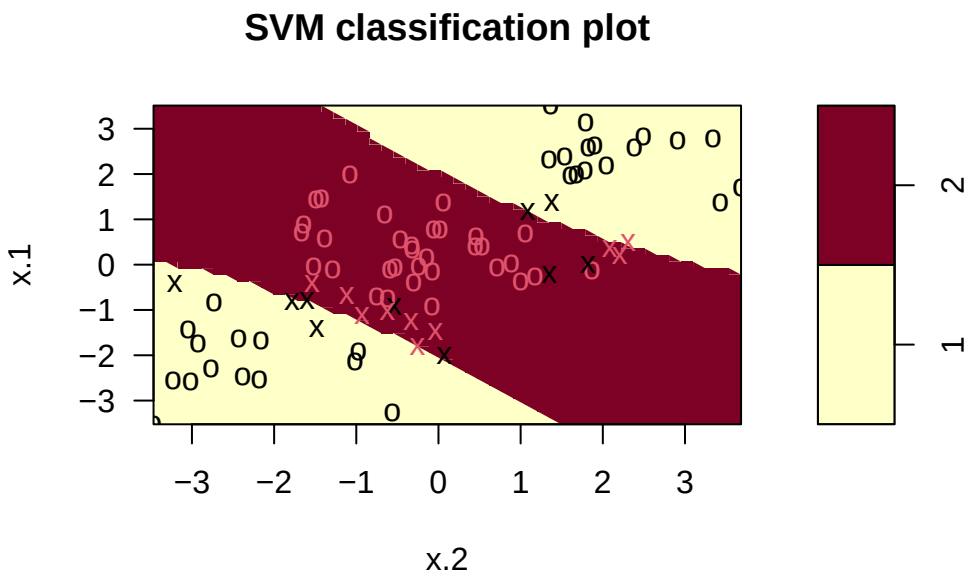
Figura 3

A seguir são ajustados os modelos SVM com kernel polinomial e radial, bem como o modelo SVM linear. Os resultados para uma partição de teste são apresentados na tabela a seguir utilizando custo igual a 1 em todos os casos, grau de polinômio igual a 2 no caso polinomial e γ igual a 1 no caso radial.

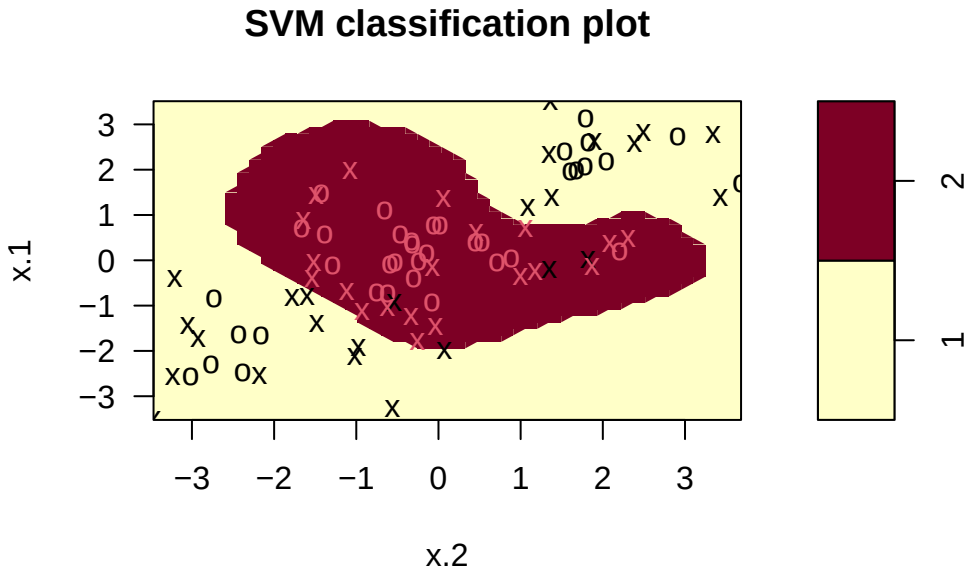
```
# seleciona parte da base como teste = 1
set.seed(1)
test <- sample(c(1, 0), 100, replace = TRUE, prob = c(0.2, 0.8))

# classificador linear
linearfit <- svm(y ~ ., data = dat[test == 0,] , kernel = "linear",
              cost = 1, scale = FALSE)

# classificador polinomial
polyfit <- svm(y ~ ., data = dat[test == 0,] , kernel = "polynomial",
              degree = 2, cost = 1, scale = FALSE)

# classificador radial
radialfit <- svm(y ~ ., data = dat[test == 0,] , kernel = "radial",
              gamma = 1, cost = 1, scale = FALSE)

# plots utilizando dados de treinamento
plot(linearfit, dat[test == 0,])
```




```
plot(polyfit, dat[test == 0,])
```




```
plot(radialfit, dat[test == 0,])
```



Visualmente, o modelo SVM com kernel polinomial parece ser o que melhor se ajusta aos dados, contando apenas os vetores próximos à fronteira, como esperado, como vetores de suporte. A tabela a seguir mostra os resultados para os modelos ajustados com os dados de treinamento.

```
# table for linear fit
tbl_linear <- table(
  true = dat[test == 0, "y"],
  pred = predict(
    linearfit , newdata = dat[test == 0, ]
  )
)

# table for poly fit
tbl_poly <- table(
  true = dat[test == 0, "y"],
  pred = predict(
    polyfit , newdata = dat[test == 0, ]
  )
)

# table for radial fit
tbl_radial <- table(
  true = dat[test == 0, "y"],
  pred = predict(
    radialfit , newdata = dat[test == 0, ]
  )
)

map(list(tbl_linear, tbl_poly, tbl_radial), function(x) {
  x %>%
```

```
    as.data.frame() %>%
    rename("True" = "true", "Predicted" = "pred") %>%
    mutate_all(as.character) %>%
    mutate(True = if_else(True == "1", "Classe real 1", "Classe real 2"),
           Predicted = if_else(Predicted == "1", "Classe fit 1", "Classe fit 2"))
            ↪  %>%
    pivot_wider(names_from = "True", values_from = "Freq")
}) %>%
  bind_rows() %>%
  mutate(Modelo = rep(c("Linear", "Polinomial", "Radial"), each = 2)) %>%
  select(Modelo, everything()) %>%
  knitr::kable(caption = "Matriz de confusão para os modelos ajustados com os dados
↪  de treinamento")
```

Tabela 1: Matriz de confusão para os modelos ajustados com os dados de treinamento

Modelo	Predicted	Classe real 1	Classe real 2
Linear	Classe fit 1	19	4
Linear	Classe fit 2	21	39
Polinomial	Classe fit 1	34	0
Polinomial	Classe fit 2	6	43
Radial	Classe fit 1	36	0
Radial	Classe fit 2	4	43

Novamente, a hipótese de que o kernel radial se ajusta melhor é suportada, visto que errou apenas 4 das 83 observações na base de treinamento, que corresponde a aproximadamente 5% de erro. O modelo polinomial teve desempenho marginalmente inferior, com aproximadamente 7% de erro.

A seguir é exposta a tabela com os resultados para os modelos ajustados com os dados de teste. Novamente, o desempenho do kerner radial é superior, com aproximadamente 5% de erro — 1 das 17 observações reservadas para teste.

```
# table for linear fit
tbl_linear <- table(
  true = dat[test == 1, "y"],
  pred = predict(
    linearfit , newdata = dat[test == 1, ]
  )
)

# table for poly fit
tbl_poly <- table(
  true = dat[test == 1, "y"],
  pred = predict(
    polyfit , newdata = dat[test == 1, ]
  )
)
```

```r
# table for radial fit
tbl_radial <- table(
  true = dat[test == 1, "y"],
  pred = predict(
    radialfit , newdata = dat[test == 1, ]
  )
)

map(list(tbl_linear, tbl_poly, tbl_radial), function(x) {
  x %>%
    as.data.frame() %>%
    rename("True" = "true", "Predicted" = "pred") %>%
    mutate_all(as.character) %>%
    mutate(True = if_else(True == "1", "Classe real 1", "Classe real 2"),
           Predicted = if_else(Predicted == "1", "Classe fit 1", "Classe fit 2"))
           ↪  %>%
    pivot_wider(names_from = "True", values_from = "Freq")
}) %>%
  bind_rows() %>%
  mutate(Modelo = rep(c("Linear", "Polinomial", "Radial"), each = 2)) %>%
  select(Modelo, everything()) %>%
  knitr::kable(caption = "Matriz de confusão para os modelos ajustados com os dados
↪  de teste")
```

Tabela 2: Matriz de confusão para os modelos ajustados com os dados de teste

Modelo	Predicted	Classe real 1	Classe real 2
Linear	Classe fit 1	5	0
Linear	Classe fit 2	5	7
Polinomial	Classe fit 1	10	2
Polinomial	Classe fit 2	0	5
Radial	Classe fit 1	10	1
Radial	Classe fit 2	0	6

CURVA ROC ENTRE OS 3 MODELOS

Questão 7

In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the `ISLR::Auto` data set.

Item a)

Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```r
auto <- ISLR::Auto %>%
  mutate(mpg01 = if_else(mpg > median(mpg), 1, 0))
```

Item b)

Fit a support vector classifier to the data with various values of `cost`, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results. Note you will need to fit the classifier without the gas mileage variable to produce sensible results.

A seguir é ajustado um classificador linear com valores de custo iguals a 0.1, 1, 10, 100 e 1000 e é usada validação cruzada com 10 folds para avaliar o desempenho do modelo. Isto é, a base de dados é particionada em 10 pedaços e o modelo é ajustado 10 vezes, cada vez utilizando 9 pedaços para treinamento e 1 para teste. O erro de classificação é calculado para cada ajuste e a média dos erros é reportada.

De acordo com os resultados do `summary()`, o modelo com cursto igual a 1 apresentou os melhores resultados, com 9,5% de erro.

```r
auto_nompg <- auto %>% select(-mpg)

set.seed (1)
tune.svc <- tune(svm, mpg01 ~ ., data = auto_nompg,
                  kernel = "linear",
                  ranges = list(
                    cost = c(0.1, 1, 10, 100, 1000)
                  )
)

summary(tune.svc)
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
    1

- best performance: 0.09603609

- Detailed performance results:
   cost      error dispersion
```

```
1 1e-01 0.10227373 0.03634911
2 1e+00 0.09603609 0.03666741
3 1e+01 0.10531309 0.03683207
4 1e+02 0.12079079 0.03864160
5 1e+03 0.12724775 0.03878303
```

Item c)

Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of `gamma` and `degree` and cost. Comment on your results.

A seguir são ajustados modelos SVM com kernel polinomial e radial, variando os valores de `cost` e `degree` para o polinomial e `cost` e `gamma` para o radial.

```
set.seed (1)
tune.svpoly <- tune(svm, mpg01 ~ ., data = auto_nompg,
                    kernel = "polynomial",
                    ranges = list(
                      cost = c(0.1, 1, 10, 100, 1000),
                      degree = c(2, 3, 4, 5)
                    )
)
```

```
set.seed (1)
tune.svradial <- tune(svm, mpg01 ~ ., data = auto_nompg,
                    kernel = "radial",
                    ranges = list(
                      cost = c(0.1, 1, 10, 100, 1000),
                      gamma = c(0.5, 1, 2, 3, 4)
                    )
)
```

Os resultados para os modelos ajustados com kernel polinomial e radial são apresentados a seguir. O modelo polinomial via de regra apresenta resultados precários. O melhor resultado, com 15% de erros, ocorre com um polinômio de grau 2 e erro igual a 1000. O modelo radial, por outro lado, apresenta resultados melhores, com erro de 6,5% para custo igual a 1 e γ igual a 0.5. Conclui-se que o modelo radial é o mais indicado, considerando apenas as simulações.

```
summary(tune.svpoly)
```

```
Parameter tuning of 'svm':
```

```
- sampling method: 10-fold cross validation

- best parameters:
 cost degree
 1000      2

- best performance: 0.1594745

- Detailed performance results:
    cost degree     error dispersion
1  1e-01      2 0.4960905 0.03974005
2  1e+00      2 0.4752293 0.04525479
3  1e+01      2 0.3375643 0.08311313
4  1e+02      2 0.2548390 0.07343884
5  1e+03      2 0.1594745 0.05452093
6  1e-01      3 0.4978768 0.03935396
7  1e+00      3 0.4924827 0.03986701
8  1e+01      3 0.4418838 0.04913560
9  1e+02      3 0.2465708 0.05122197
10 1e+03      3 0.1683651 0.03763027
11 1e-01      4 0.4984602 0.03930559
12 1e+00      4 0.4983427 0.03933257
13 1e+01      4 0.4971505 0.03961089
14 1e+02      4 0.4855727 0.04370976
15 1e+03      4 0.3998639 0.09315902
16 1e-01      5 0.4984718 0.03930295
17 1e+00      5 0.4984585 0.03930611
18 1e+01      5 0.4983260 0.03933773
19 1e+02      5 0.4969797 0.03965863
20 1e+03      5 0.4838647 0.04389044
```

```
summary(tune.svradial)
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.0657775

- Detailed performance results:
    cost gamma      error  dispersion
1  1e-01   0.5 0.08548406 0.021104890
2  1e+00   0.5 0.06577750 0.026197728
3  1e+01   0.5 0.07304343 0.027869128
4  1e+02   0.5 0.07667563 0.027824986
5  1e+03   0.5 0.07667563 0.027824986
```

```
6  1e-01   1.0 0.27951125 0.036343997
7  1e+00   1.0 0.09918732 0.020523476
8  1e+01   1.0 0.10442706 0.020690277
9  1e+02   1.0 0.10442540 0.020692046
10 1e+03   1.0 0.10442540 0.020692046
11 1e-01   2.0 0.42047779 0.044561140
12 1e+00   2.0 0.20409900 0.013765977
13 1e+01   2.0 0.20506214 0.013269495
14 1e+02   2.0 0.20506214 0.013269495
15 1e+03   2.0 0.20506214 0.013269495
16 1e-01   3.0 0.43453116 0.043273628
17 1e+00   3.0 0.23107129 0.010987668
18 1e+01   3.0 0.23126372 0.010794330
19 1e+02   3.0 0.23126372 0.010794330
20 1e+03   3.0 0.23126372 0.010794330
21 1e-01   4.0 0.43894907 0.041927995
22 1e+00   4.0 0.23592639 0.009193937
23 1e+01   4.0 0.23597971 0.009134275
24 1e+02   4.0 0.23597971 0.009134275
25 1e+03   4.0 0.23597971 0.009134275
```

Item d)

Make some plots to back up your assertions in (b) and (c).

A seguir, para comparar os modelos de forma sintética, são comparadas as curvas ROC para os melhores modelos de cada categoria. O que se pode observar na Figura 4 é uma confirmação de que o modelo polinomial tem um desempenho consideravelmente pior que os demais. Enquanto o modelo radial parece apresentar quase desempenho perfeito — o que pode ser um indicador de overfitting — o modelo linear apresenta um desempenho comparável a este e, por isso, pode apresentar uma flexibilidade maior para classificação de novas observações.

```r
fit_svc <- attributes(predict(tune.svc$best.model , auto_nompg, decision.values =
↪  T))$decision.values

fit_svpoly <- attributes(predict(tune.svpoly$best.model , auto_nompg,
↪  decision.values = T))$decision.values

fit_svradial <- attributes(predict(tune.svradial$best.model , auto_nompg,
↪  decision.values = T))$decision.values

rocplot(fit_svc , auto_nompg$mpg01, col = "red")
rocplot(fit_svpoly , auto_nompg$mpg01, add = T, col = "blue")
rocplot(fit_svradial , auto_nompg$mpg01, add = T, col = "green")
legend("bottomright", legend = c("Linear", "Poli.", "Radial"), col = c("red",
↪  "blue", "green"), pch = 16)
```

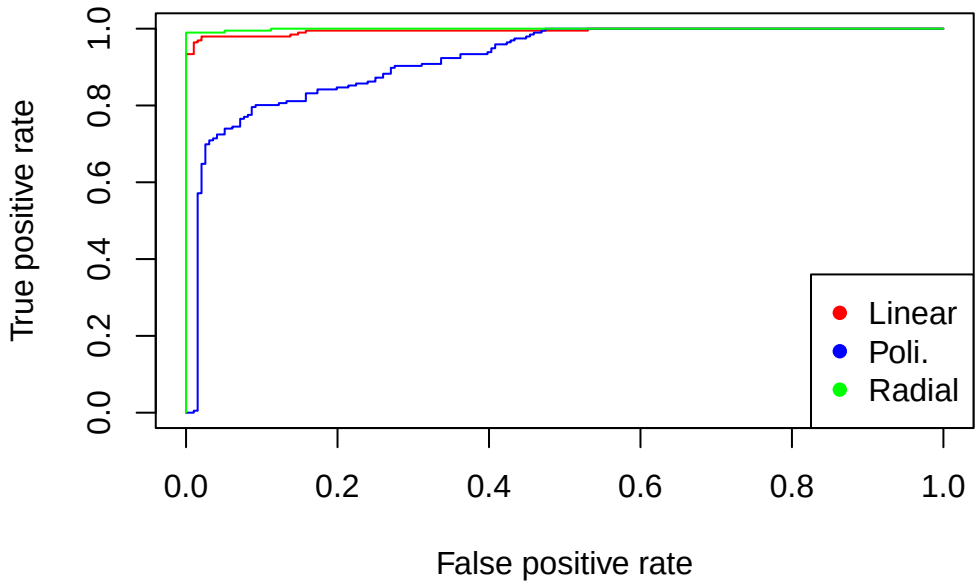


Figura 4: Curvas ROC para os modelos ajustados com SVMs

Questão 13

Escolha uma linguagem de programação (`R`, `Python`, `SAS`, `Matlab`, `Julia`) e apresente um exemplo de classificação com SVM utilizando Kernel Linear e outro utilizando Kernel não Linear.

No `SAS`, a função implementada para realizar análises utilizando algoritmos de suporte vetorial (SVM) é o `PROC HPSVM`. Essa função permite o uso de kernels lineares e não lineares nos dados de treinamento. O `PROC HPSVM` executa o algoritmo SVM de alta performance, possibilitando sua execução em computação paralela tanto em uma única máquina quanto em múltiplas máquinas.[1]

Utilizando o exemplo fornecido na documentação do `SAS` com um kernel linear, vamos ajustar um modelo utilizando um banco de dados chamado `SAMPSIO.DMAGECR`, um banco de dados de referência que traz informações sobre risco de crédito[2]. Esse banco de dados faz parte da biblioteca `SAMPSIO` e contém 1.000 observações, cada uma com informações detalhadas sobre os requerentes. O banco inclui a classificação do indivíduo como GOOD ou BAD em uma variável denominada GOOD_BAD, além de outras variáveis como histórico de crédito, duração do empréstimo, entre outras.

A tabela 'Matriz de Classificação' mostra que, entre as 1.000 observações totais, 700 são classificadas como boas e 300 como ruins. O número de observações BOAS corretamente previstas é 626, e o número de observações RUINS corretamente previstas é 158.

[1] https://documentation.sas.com/doc/en/emhpprcref/14.2/emhpprcref_hpsvm_overview.htm
[2] https://support.sas.com/documentation/cdl/en/emgs/59885/HTML/default/viewer.htm#a001026918.htm

Classification Matrix			
	Training Prediction		
Observed	good	bad	Total
good	626	74	700
bad	142	158	300
Total	768	232	1000

Assim, a precisão do modelo é de 78,4%, conforme indicado na tabela 'Estatísticas de Ajuste'.

Fit Statistics	
Statistic	Training
Accuracy	0.7840
Error	0.2160
Sensitivity	0.8943
Specificity	0.5267

Um modelo relativamente bom significa que a taxa de erro de classificação é baixa, enquanto a sensibilidade e a especificidade são altas. Com o PROC HPSVM, é possível ajustar os parâmetros de treinamento e usar diferentes kernels para obter um modelo melhor. O padrão do procedimento usa o kernel linear conforme especificado:

$$k(x_1, x_2) = \langle x_1, x_2 \rangle, \tag{1}$$

onde x_1 e x_2 são dois vetores e $\langle \cdot, \cdot \rangle$ é o produto interno. Para alterar o tipo de kernel utilizado, basta definir o argumento KERNEL, especificando o tipo de kernel e quaisquer parâmetros associados. Definindo o kernel como polinomial:

$$k(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^p, \tag{2}$$

onde p é o grau do polinômio, obtém-se o resultado:

Classification Matrix			
	Training Prediction		
Observed	good	bad	Total
good	699	1	700
bad	3	297	300
Total	702	298	1000

A matriz de classificação mostra que o número de observações BOAS corretamente previstas é 699, e o número de observações RUINS corretamente previstas é 297. Assim, a precisão do modelo é de 99,6%, conforme indicado na tabela 'Estatísticas de Ajuste', mostrando que o uso do kernel polinomial é mais acurado, nesse caso.

Fit Statistics	
Statistic	Training
Accuracy	0.9960
Error	0.0040
Sensitivity	0.9986
Specificity	0.9900

O código a seguir foi utilizado para gerar os resultados desta questão:

```
proc freq data = sampsio.dmagecr;
 tables GOOD_BAD;
 run;


 proc hpsvm data=sampsio.dmagecr;
     input checking history purpose savings employed marital coapp
           property other job housing telephon foreign/level=nominal;
     input duration amount installp resident existcr depends age/level=interval;
     target good_bad;
     KERNEL LINEAR;
 run;

/* Ajustando o modelo SVM
proc hpsvm data=sampsio.dmagecr;
    input checking history purpose savings employed marital coapp
          property other job housing telephon foreign / level=nominal;
    input duration amount installp resident existcr depends age / level=interval;
    target good_bad / level=nominal;
    id duration amount;
    savestate rstore=work.svm_model;
run;

Aplicando o modelo ajustado aos dados e salvando os resultados
proc astore;
    score data=sampsio.dmagecr out=svm_results rstore=work.svm_model;
run;

Gerando um gráfico de dispersão para visualizar a distribuição das classes previstas
↪
proc sgplot data=svm_results;
    scatter x=amount y=duration / group=P_good_bad;
    xaxis label="Amount";
    yaxis label="Duration";
    title "Gráfico de Dispersão das Classes Previstas pelo Modelo SVM";
run;*/

 proc hpsvm data=sampsio.dmagecr;
     input checking history purpose savings employed marital coapp
           property other job housing telephon foreign/level=nominal;
     input duration amount installp resident existcr depends age/level=interval;
     target good_bad;
     KERNEL POLYNOM;
 run;
```