Machine Learning: Lista de Exercícios 1

Paulo Orenstein

Monitores: Antônio Catão, Otávio Moreira, Melvin Poveda

Verão, 2025

Exercício 1 (Verdadeiro ou Falso)**.** Indique se as afirmações abaixo são verdadeiras ou falsas, e justifique brevemente.

(a) Se um modelo tem um erro de teste baixo, o valor do erro de treino não traz nenhuma informação adicional sobre como potencialmente melhorar o modelo.

(b) Numa regressão linear, não é possível testar uma hipótese $H_0 : \beta_j = 0$ usando o teste t sem fazer alguma hipótese sobre a distribuição dos erros.

(c) Um banco de varejo está interessado em reduzir transações fraudulentas e desenvolveu dois modelos para isso; o primeiro tem acurácia (isto é, um menos a perda 0-1) 98.5% e o outro 99%. Suponha que o banco se preocupe mais em barrar todas as transações fraudulentas do que em aceitar todas as transações legítimas. Nesse caso, é necessariamente preferível usar o modelo de maior acurácia.

(d) Existe sempre um valor de λ capaz de garantir que a performance de uma regressão ridge nunca seja pior do que uma regressão linear.

(e) Se, ao rodar uma regressão linear, os erros não tiverem uma distribuição Normal, então os intervalos de confiança gerados via bootstrap em geral passam a ser uma alternativa mais apropriada do que os intervalos de confiança $[\hat{\beta}_j - 2\sqrt{\hat{\sigma}^2[(X^TX)^{-1}]_{jj}}, \hat{\beta}_j + 2\sqrt{\hat{\sigma}^2[(X^TX)^{-1}]_{jj}}]$.

Exercício 2 (Regressão linear com resíduos correlacionados e heteroscedasticidade)**.** Considere uma amostra $\{(X_i, Y_i)\}_{i=1}^n$ satisfazendo o modelo linear:

$$Y_i = f(X_{i,1}, \ldots, X_{i,p}) + \varepsilon_i,$$

onde $f(X_{i,1}, \ldots, X_{i,p}) = \beta_0 + \sum_{j=1}^p \beta_j X_{i,j}$ e ε é um vetor tal que $\mathbb{E}[\varepsilon] = 0$ e $\mathbb{V}[\varepsilon] = \Sigma$, onde Σ é uma matriz positiva definida.

(a) Explique como a hipótese de resíduos não-correlacionados ($\mathbb{E}[\varepsilon_i \varepsilon_j] = 0$ para $i \neq j$) e homoscedasticidade ($\mathbb{V}[\varepsilon_i] = \mathbb{V}[\varepsilon_j]$) que tipicamente são feitas em modelos de regressão se traduzem respectivamente em hipóteses sobre Σ neste modelo.

(b) Considere a seguinte versão modificada do problema de mínimos quadrados para esta situação:

$$\hat{\beta}_\Sigma = \underset{\beta}{\operatorname{argmin}} (y - X\beta)^\intercal \Sigma^{-1} (y - X\beta). \tag{1}$$

Prove que:

(b-i)
$$\hat{\beta}_\Sigma = \left(X^\intercal \Sigma^{-1} X\right)^{-1} X^\intercal \Sigma^{-1} y.$$

(b-ii)
$$\mathbb{E}(\hat{\beta}_\Sigma) = \beta.$$

(b-iii)
$$\mathbb{V}[\hat{\beta}_\Sigma] = \left(X^\intercal \Sigma^{-1} X\right)^{-1}.$$

(b-iv) Se $\varepsilon \sim \mathcal{N}(0, \Sigma)$, então $\hat{\beta}_\Sigma \sim \mathcal{N}(\beta, \left(X^\intercal \Sigma^{-1} X\right)^{-1})$.

(c) Agora, vamos supor que os dados não são correlacionados, mas que há heterocedasticidade, no sentido de que $\Sigma = \operatorname{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$, ou seja $\mathbb{V}[\varepsilon_i] \neq \mathbb{V}[\varepsilon_j]$ para $i \neq j$.

(c-i) Mostre que, nesse caso, (1) acima é equivalente a:

$$\hat{\beta}_\Sigma = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n w_i^2 \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2,$$

onde as observações recebem peso $w_i^2 = 1/\sigma_i^2$.

(c-ii) Interprete a escolha dos pesos $w_i^2 = 1/\sigma_i^2$ no cálculo dos coeficientes.

(c-iii) Mais geralmente, prove que o problema de otimização em (1) é equivalente a:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} (\tilde{y} - \tilde{X}\beta)^T (\tilde{y} - \tilde{X}\beta) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n \left(\tilde{y}_i - \frac{\beta_0}{\sigma_i} - \sum_{j=1}^p \beta_j \tilde{x}_{ij} \right)^2,$$

onde $\tilde{y} = \Sigma^{-\frac{1}{2}} y$ e $\tilde{X} = \Sigma^{-\frac{1}{2}} X$.

(d) Considere o caso particular do modelo acima com $n = 50$, $p = 1$, $X_{i,1} \sim \mathcal{N}(0, 1)$ e $\varepsilon \sim \mathcal{N}(0, \Sigma)$ independente com Σ diagonal com diagonal $\Sigma_{ii} = \sigma_i^2 = 10^{\frac{i-20}{5}}$ e $\beta_0 = 1, \beta_1 = 0.25$. Vamos gerar amostras deste modelo e comparar o desempenho de $\hat{\beta}_\Sigma$ com o estimador de mínimos quadrados usual $\hat{\beta}_{ls}$.

(d-i) Rode os seguintes comandos para gerar um conjunto de dados:

```
import numpy as np

n = 50
Sigma = np.diag([10**((i-20)/5) for i in range(1,n+1)])

np.random.seed(0)
X = np.array([np.ones(n),np.random.normal(0,1,n)]).T
beta = np.array([1,0.25])
epsilon = np.random.multivariate_normal(np.zeros(n),Sigma)
y = X @ beta + epsilon
```

(d-ii) Calcule $\hat{\beta}_{\text{ls}}$ e $\hat{\beta}_{\Sigma}$ para o conjunto de dados acima e compare como ambos estimadores se saem em termos de $\|\beta - \hat{\beta}\|_2^2$.

(d-iii) Calcule o p-valor do teste t para o coeficiente $(\hat{\beta}_{\text{ls}})_0$ como se estivéssemos supondo erroneamente que os dados foram gerados com um ε vetor de resíduos com $\Sigma = \sigma^2 I$. Com base neste p-valor, você rejeitaria a hipótese nula?

(d-iv) Calcule a seguinte estatística:

$$Z = \frac{(\hat{\beta}_{\Sigma})_0}{\sqrt{(X^T \Sigma^{-1} X)^{-1}_{1,1}}}$$

(d-v) Argumente que, condicionado em X, vale que $Z \sim \mathcal{N}(0,1)$ sob a hipótese nula $H_0 : \beta_0 = 0$ e use isto para calcular o p-valor associado ao teste de $H_0 : \beta_0 = 0$ vs. $H_1 : \beta_0 \neq 0$. Agora, com base neste novo p-valor, você rejeitaria a hipótese nula?

Exercício 3 (Regressão com erros Laplacianos)**.** Usualmente, em regressão linear, assumimos que os erros tem distribuição $\varepsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$, de modo que os erros seguem a função de densidade

$$p_{\text{Normal}}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

O que acontece quando os erros seguem outra distribuição? Suponha que $\varepsilon \overset{\text{iid}}{\sim} \text{Laplace}(0, b)$, onde a distribuição Laplaciana é parametrizada pela média μ e um parâmetro de escala b, e tem densidade

$$p_{\text{Laplace}}(x; \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right).$$

A Figure 1 ilustra diferenças entre as densidades: a distribuição Laplaciana tem um pico pronunciado em torno da média e caudas pesadas (*i.e.*, grandes desvios da média são mais prováveis).

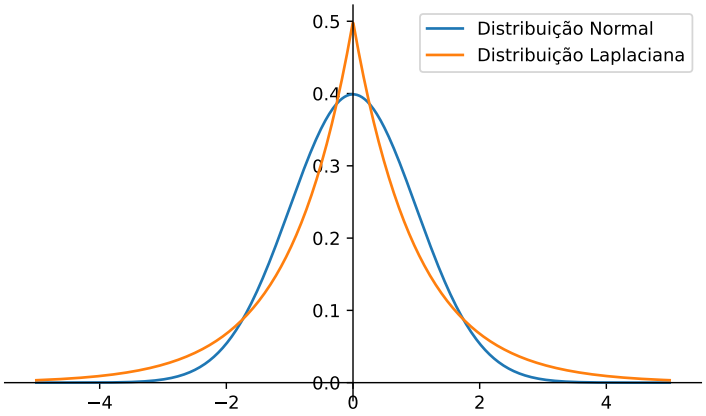

Figura 1: Funções de densidade da distribuição Laplaciana e Gaussiana.

Nesta questão, vamos supor que $\varepsilon_i \overset{\text{iid}}{\sim} \text{Laplace}(0, b)$, com $b > 0$, e que os dados são gerados via

$$Y_i = \sum_{j=1}^{p} \beta_j X_{ij} + \varepsilon_i = \beta^T X_i + \varepsilon_i.$$

(a) Argumente que $Y_i | X_i \sim \text{Laplace}(\beta^T X_i, b)$, e, dada uma amostra $\{(x_i, y_i)\}_{i=1}^n$, escreva a função de verossimilhança $\ell(\beta)$ do parâmetro β em termos de β, b e das amostras $\{(x_i, y_i)\}_{i=1}^n$.

(b) A partir do item (a), sugira uma função-perda para regressão linear com erros Laplacianos. Simplifique sua solução de tal modo que a função perda não dependa do parâmetro b.

(c) Mostre que a função-perda obtida no item anterior, no caso de erros Normais, é o erro médio quadrático. Cite uma vantagem de usar a função-perda do item (b) versus erro médio quadrático.

(d) Gostaríamos de aprender o parâmetro β usando o método de descida de gradiente. Escreva a regra de atualização de descida de gradiente para minimizar a função-perda do item (b). A regra de atualização deve ser uma fórmula para $\hat{\beta}^{(t)}$ (o valor de $\hat{\beta}$ na iteração t), em termos de $\hat{\beta}^{(t-1)}$ (o valor de $\hat{\beta}$ na iteração $t-1$) e η (a taxa de aprendizado). *Dica:* você pode assumir que $\frac{d}{dz}|z| = \text{sgn}(z)$, isto é, 1 se > 0, -1 se $z < 0$, e 0 se $z = 0$.

(e) Vamos comparar o passo de descida de gradiente encontrado em (d) com a regra de atualização no caso de regressão linear com erro Normal.

(i) Suponha $n = 1$, ou seja, que temos apenas uma amostra de treinamento (x, y). Quando a predição do modelo $\beta^T x$ está próxima do real valor y, qual modelo fará o maior passo na descida do gradiente: o modelo com erro Laplaciano ou o modelo com erro Gaussiano? (Assuma o mesmo valor de η para ambos os modelos.)

(ii) Ainda no caso $n = 1$, começando com o mesmo valor inicial de β, é verdade que a direção do passo do modelo Laplaciano é sempre a mesma da do modelo Normal?

(iii) Agora, considere o caso em que estamos realizando uma iteração de descida de gradiente para um conjunto com $n > 1$ amostras. Os modelos com erro Laplaciano e Gaussiano terão um passo de gradiente sempre na mesma direção? Justifique.

(f) Iremos comparar a regressão linear com erros Laplacianos e Gaussiano em um experimento com dados sintéticos.

(i) Complete o código abaixo com o cálculo de $\hat{\beta}$ com erros Gaussiano e Laplaciano para o dataset (`X`, `y`). (*Dica:* Use a função `minimize` para calcular `beta_hat_laplacian`)

```
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

np.random.seed(1)

beta = np.array([-1.5, 2.0])
input_range = np.linspace(-1, 1, 100)
X = np.vstack([np.ones(100), input_range]).T
y = X @ beta + np.random.normal(0, 0.3, 100)

def calculate_laplacian_loss(parameters, X, y):
    return np.sum(np.abs(y - X @ parameters))

# Complete as linhas abaixo
beta_hat_gaussian = ...
beta_hat_laplacian = ...
```

(ii) Plote os gráficos da regressão linear em cada caso. Considerando o erro $\|\beta - \hat{\beta}\|_2$, qual modelo você escolheria para esse conjunto de dados? Justifique.

(iii) Vamos adicionar um outlier ao conjunto de dados. Adicione a linha abaixo ao código anterior e repita o item anterior. Qual modelo você escolheria agora? Justifique.

```
y[80] = 10
```

Exercício 4 (Inferência para regressão logística)**.** Em regressão linear, vimos que se os dados vêm de um modelo da forma $Y_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi} + \varepsilon_i = \beta^T x_i + \varepsilon_i$ onde $\varepsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ não depende de X_i, então podemos fazer inferência acerca dos coeficientes estimados $\hat{\beta}_j$, isto é, podemos construir intervalos de confiança e realizar testes de hipótese. O objetivo desta questão é mostrar que isso também é possível para regressão logística assintoticamente.

Considere uma amostra iid $\{(x_i, y_i)\}_{i=1}^n$, onde $x_i \in \mathbb{R}^{p+1}$ e $y_i \in \{0, 1\}$. Vamos supor que os y_i possam ser modelados como realizações de variáveis binárias independentes

$$Y_i \sim \text{Bern}\left(p_\beta(x_i)\right), \; i = 1, \ldots, n, \tag{2}$$

onde

$$p_\beta(x_i) = \frac{1}{1 + e^{-\beta^T x_i}},$$

para $\beta \in \mathbb{R}^{p+1}$ um vetor de coeficientes desconhecido que queremos estimar. Como visto em aula, a log-verossimilhança é

$$\ell_n(\beta) = \sum_{i=1}^n Y_i \log(p_\beta(x_i)) + (1 - Y_i) \log(1 - p_\beta(x_i)).$$

com

$$\begin{aligned}
\nabla_\beta \ell_n(\beta) &= \sum_{i=1}^n (Y_i - p_\beta(x_i)) x_i \\
\nabla^2_\beta \ell_n(\beta) &= -\sum_{i=1}^n x_i x_i^T p_\beta(x_i)(1 - p_\beta(x_i)).
\end{aligned}$$

(a) Considerando a aleatoriedade de Y_i de acordo com (2), mostre que, para qualquer $\beta \in \mathbb{R}^{p+1}$,

$$\begin{aligned}
\mathbb{E}\left[\nabla_\beta \ell_n(\beta)\right] &= 0 \\
\mathbb{E}\left[\nabla^2_\beta \ell_n(\beta)\right] &= \nabla^2_\beta \ell_n(\beta) \\
\mathbb{V}\left[\nabla_\beta \ell_n(\beta)\right] &= -\nabla^2_\beta \ell_n(\beta).
\end{aligned}$$

Em geral, chamamos de $I_n(\beta) = -\mathbb{E}[\nabla^2_\beta \ell_n(\beta)]$ a matriz de informação de Fisher.

(b) Lembrando que o estimador de máxima verossimilhança satisfaz $\hat{\beta} = \text{argmax}_{\tilde{\beta}} \, \ell_n(\tilde{\beta})$, sob algumas condições de regularidade, podemos fazer uma expansão de Taylor:

$$0 = \nabla \ell_n(\hat{\beta}) \approx \nabla \ell_n(\beta) + \nabla^2 \ell_n(\beta)(\hat{\beta} - \beta). \tag{3}$$

Supondo que (3) valha com igualdade, use os resultados do item anterior para mostrar que

$$I_n(\beta)^{\frac{1}{2}}(\hat{\beta} - \beta) \xrightarrow{d} N(0, I_{p+1}), \tag{4}$$

onde $I_n(\beta)^{\frac{1}{2}}$ denota a raiz quadrada da matriz $I_n(\beta)$ (ou seja, é a única matriz M tal que $M \cdot M = I_n(\beta)$), e I_{p+1} é a matriz de identidade. (*Dica:* se definirmos $W_i = (Y_i - p_\beta(x_i))x_i$, então, pelo Teorema Central do Limite para vetores independentes, vale que $(\sum_{i=1}^n \mathbb{V}[W_i])^{-\frac{1}{2}} \sum_{i=1}^n (W_i - \mathbb{E}[W_i]) \xrightarrow{d} N(0, I_{p+1})$.)

(c) A partir de (4), justifique por que vale

$$\hat{\beta}_j - \beta_j \xrightarrow{d} N(0, [I_n^{-1}(\beta)]_{jj}), \tag{5}$$

onde $[A]_{jj}$ denote a entrada (j, j) da matriz A.

(d) A equação (5) implica em $([I_n^{-1}(\beta)]_{jj})^{-1/2}(\hat{\beta}_j - \beta_j) \xrightarrow{d} N(0,1)$. Como $\hat{\beta}$ é muito perto de β, vale que $I_n^{-1}(\hat{\beta})$ é próximo de $I_n^{-1}(\beta)$ de modo que, assintoticamente, também vale

$$\frac{1}{\sqrt{[I_n^{-1}(\hat{\beta})]_{jj}}}(\hat{\beta}_j - \beta_j) \xrightarrow{d} N(0,1). \tag{6}$$

Usando (6), sugira uma estatística de teste assintoticamente válida para testar $H_0 : \beta_j = 0$.

(e) Vamos verificar o resultado (6) computacionalmente. Utilize o seguinte código para gerar um conjunto artificial de *features*:

```
import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt

p = 5
n = 1000

np.random.seed(10)
beta = np.ones(p)
X = np.random.normal(loc=0,scale=1,size=(n,p))
```

Gere uma amostra de n observações segundo (2) através de:

```
logits = 1/(1+np.exp(-X@beta))
y = np.random.binomial(n=np.ones(n).astype(int),p=logits, size = n)
```

Rode um modelo de regressão logística sem intercepto e sem regularização nos dados acima, obtendo $\hat{\beta}$, e depois calcule:

$$\frac{1}{\sqrt{[I_n^{-1}(\hat{\beta})]_{11}}} \cdot (\hat{\beta}_1 - 1), \tag{7}$$

onde $[I_n^{-1}(\hat{\beta})]_{11}$ significa a entrada $(1,1)$ da matriz $I_n^{-1}(\hat{\beta})$.

(f) Repita a última linha de código do item acima 1000 vezes para obter $\{Y^{(i)}\}_{i=1}^{1000}$ e, a partir disso, calcular o conjunto de vetores de coeficientes $\{\hat{\beta}^{(i)}\}_{i=1}^{1000}$ (use `np.random.seed(10)` antes do `for` para que seus resultados sejam completamente reprodutíveis). Faça um histograma dos valores obtidos via (7) para cada $\hat{\beta}_1^{(i)}$, $i = 1, \ldots, 1000$. Os dados se parecem com uma Normal?

(g) A equação (4) sugere que a matriz de covariância empírica, $\hat{\mathbb{V}}[\hat{\beta}]$, é bem-aproximada por $I_n^{-1}(\beta)$ onde β = `np.ones(p)`; já (6) sugere que essa matriz também pode ser aproximada por $I_n^{-1}(\hat{\beta})$. Usando $\{\hat{\beta}^{(i)}\}_{i=1}^{1000}$, calcule essas três matrizes. As entradas são próximas?

(h) Utilize a amostra gerada no item (e) para realizar o teste de confiança proposto no item (c), isto é, $H_0 : \beta_1 = 0$. A hipótese nula é rejeitada? Compare o valor da sua estatística de teste com a obtida usando o pacote `statsmodels` via

```
import statsmodels.api as sm
logit_model = sm.Logit(y, X)
print(logit_model.fit().summary())
```

Exercício 5 (Comparando modelos de classificação)**.** Neste exercício vamos comparar 5 métodos de classificação diferentes: regressão logística, LDA, QDA, naive Bayes e kNN.

Faça o download do dataset `soccer.csv` e use o excerto de código abaixo para (i) carregar os pacotes que usaremos no exercício; (ii) carregar os dados de partidas de futebol entre seleções de diversos países; (iii) definir uma variável `X` para as features e `y` para a variável resposta e (iv) particionar os dados em treino e teste. Atente-se para o caminho `./soccer.csv` na chamada do `pd.read_csv`: você deve ajustá-lo para o caminho do arquivo no seu computador.

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis as QDA
from sklearn.linear_model import LogisticRegression as LR
from sklearn.naive_bayes import GaussianNB as NB
from sklearn.neighbors import KNeighborsClassifier as kNN
from sklearn import preprocessing

df = pd.read_csv("./soccer.csv")

X = df.drop("target", axis=1)
y = df[["target"]]

X_train, y_train = X.iloc[:2560], y.iloc[:2560]
X_test, y_test = X.iloc[2560:], y.iloc[2560:]
```

Nossa variável resposta `target` é 1 se o time da casa venceu a partida e 0 caso contrário. A descrição de todas as variáveis está disponível na Tabela 1.

Tabela 1: Descrição das variáveis do dataset `soccer`

Variável	Descrição
`home_team`	Time da casa
`away_team`	Time visitante
`target`	Se o time da casa venceu o jogo
`rank_dif`	Diferença de ranking entre os times
`goals_dif`	Dif. da média de gols entre o time da casa e o de fora
`goals_dif_l5`	Dif. da média de gols entre o time da casa e o de fora (últimos 5 jogos)
`goals_suf_dif`	Dif. da média de gols sofridos entre o time da casa e o de fora
`goals_suf_dif_l5`	Dif. da média de gols sofridos entre o time da casa e o de fora (5 jogos)
`goals_per_ranking_dif`	Dif. da razão média de gols / média de ranking entre os times
`dif_rank_agst`	Dif. entre ranking médio do time da casa e do time de fora
`dif_rank_agst_l5`	Dif. entre ranking médio do time da casa e do time de fora (5 jogos)
`dif_points_rank`	Diferença de pontos por ranking
`dif_points_rank_l5`	Diferença de pontos por ranking (últimos 5 jogos)
`is_friendly_1`	Se o jogo é amistoso

Processamos os dados para descartar features categóricas e normalizar as features usando a média e a variância do conjunto de treino.

```
X_train = X_train.drop(["home_team", "away_team"], axis=1)
```

```
X_test = X_test.drop(["home_team", "away_team"], axis=1)

scaler = preprocessing.StandardScaler().fit(X_train)
X_train = preprocessing.scale(X_train)
X_test = scaler.transform(X_test)
```

(a) Explique por que é necessário normalizar as features.

(b) Treine cada um dos 5 modelos (use $k = 5$ para o kNN) e faça previsões com base tanto em `X_train` quanto em `X_test`;

(c) Faça um único gráfico de dispersão com a taxa de erro de treino no eixo x e a taxa de erro de teste no eixo y para os modelos e discuta o resultado obtido (use cores ou símbolos diferentes para diferenciar os modelos e indique-os na legenda da figura);

(d) Repita os itens (b) e (c) para kNN com $k = 1, 2, 3, ..., 9, 10$. Explique o comportamento observado.

Exercício 6 (Seleção de variáveis e lasso)**.** Vamos comparar diferentes abordagens para seleção de variáveis para regressão linear: *best subset selection*, *forward stepwise selection*, *backward stepwise* e lasso. Faremos isso através da tarefa de analisar como o percentual de gordura corporal se relaciona com as outras métricas corporais da tabela 2 (exceto densidade por pesagem hidrostática, pois este percentual de gordura corporal já é definido como uma função linear desta variável).

Tabela 2: Descrição das variáveis do dataset de gordura corporal

Variável	Descrição
`Density`	Densidade determinada por pesagem hidrostática
`BodyFat`	Percentual de gordura corporal pela equação de Siri (1956)
`Age`	Idade (anos)
`Weight`	Peso (libras)
`Height`	Altura (polegadas)
`Neck`	Circunferência do pescoço (cm)
`Chest`	Circunferência do tórax (cm)
`Abdomen`	Circunferência do abdômen (cm)
`Hip`	Circunferência do quadril (cm)
`Thigh`	Circunferência da coxa (cm)
`Knee`	Circunferência do joelho (cm)
`Ankle`	Circunferência do tornozelo (cm)
`Biceps`	Circunferência do bíceps (cm)
`Forearm`	Circunferência do antebraço (cm)
`Wrist`	Circunferência do pulso (cm)

Faça download do dataset clicando em `bodyfat.csv`. Troque `"../bodyfat.csv"` no código abaixo pelo caminho no seu computador do `.csv` baixado e execute o trecho de comandos abaixo em Python para carregar os pacotes necessários, o dataset e gerar os conjuntos de treino e teste e a separação de amostras do conjunto de treino em 5 subconjuntos para realizar validação cruzada.

```
import statsmodels.api as sm
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, KFold
import matplotlib.pyplot as plt
from sklearn.linear_model import Lasso
from sklearn import preprocessing

bodyfat = pd.read_csv("../bodyfat.csv")

X = bodyfat.drop(columns=["BodyFat","Density"])
y = bodyfat["BodyFat"]
X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.2,
random_state = 10
)

kf = KFold(n_splits = 5, shuffle = True, random_state = 10)
cv_fold = np.zeros(len(y_train)).astype(int)
for i, (_, fold_indexes) in enumerate(kf.split(X_train)):
```

```
cv_fold[fold_indexes] = int(i)
```

(a) Um dos modelos considerados requer que os dados sejam normalizados para que cada feature tenha variância 1. Identifique esse modelo e justifique a necessidade da normalização. Nos itens abaixo, implemente a normalização para esse modelo, mas não para os demais.

(b) Seja o $\mathcal{M}_k^S$ o modelo de regressão linear treinado no conjunto de treino com k preditores obtidos via algum método de seleção S através da maximização do R^2 dentre as possíveis escolhas de subconjuntos de k preditores. Utilize o conjunto de treino para obter $\mathcal{M}_k^S$ para cada número possível de preditores k entre 1 e 13, e os métodos de seleção S dentre *best subset selection*, *forward stepwise selection* e *backward stepwise selection*.

(c) Faça um gráfico com três curvas, cada uma correspondendo a um dos métodos de seleção de variáveis S citados acima, contendo no eixo x cada possível valor de k entre 1 e 13 e no eixo y o R^2 do modelo $\mathcal{M}_k^S$ correspondente.

(d) Gere o seguinte conjunto de valores α:

```
alphas = 10**np.linspace(5,-2,100)
```

Faça um gráfico mostrando no eixo y o MSE estimado para o lasso via validação cruzada, com a separação em subconjuntos gerada no início do exercício, e no eixo x o parâmetro α usado dentre a malha de valores gerada acima. Obtenha α^* o valor de α que atinge o menor valor do MSE estimado via validação cruzada. Denotaremos por $\mathcal{M}^{\text{lasso}}$ o modelo de lasso ajustado em todo o conjunto de treino com $\alpha = \alpha^*$.

(e) Calcule o MSE no conjunto de teste de $\mathcal{M}^{\text{lasso}}$ e $\mathcal{M}_k^S$ para cada um dos três métodos de seleção S descritos acima.

(f) Qual é o melhor modelo em termos do MSE no conjunto de teste e quais são os preditores utilizados por ele?