Trabalho Computacional 01
TIP8311 - Reconhecimento de Padrões
Modelos de Classificação para Diagnóstico de Parkinson

Artur Rodrigues Rocha Neto - 431951
Mestrando em Engenharia de Teleinformática
artur.rodrigues26@gmail.com

7 de Dezembro de 2018

1 Introdução

A Doença de Parkinson é um condição neurológica causada pela perda de células produtoras de dopamina e que geralmente afeta seres humanos acima de 60 anos de idade. É caracterizada pela perda gradual das capacidades motoras e seus principais sintomas são: bradicinesia (lentidão dos movimentos em geral), rigidez muscular, instabilidade na postura e tremores ao longo do corpo. É uma doença crônica e progressiva, ou seja, é uma condição que tanto perdura por um longo período quanto seus sintomas se agravam constantemente [1].

Não existe um exame específico para o diagnóstico da Doença de Parkinson. Exames neurológicos e motores, aliados a um estudo do histórico do paciente, guiam o médico neurologista a diagnosticar a condição e a descartar outras doenças similares. Métodos baseados em modelos preditivos são comuns no auxílio desse procedimento [2, 3, 4].

Este trabalho aborda o diagnóstico da Doença de Parkinson através da detecção de disfonia (dificuldade de emissão vocal). Foi usado um conjunto de dados[1] contendo amostras de fonação de 31 indivíduos, dos quais 23 tinham Doença de Parkinson [5]. Para fins didáticos, foi efetuado um estudo comparativo entre alguns classificadores e métodos de redução de dimensionalidade.

2 Conjunto de Dados

O conjunto de dados `parkinsons.data` [5] agrega $N = 195$ amostras de 31 indivíduos, sendo 23 diagnosticados com Doença de Parkinson. Para cada observação existem $p = 22$ atributos mais a informação de classe. Os atributos quantitativos definem características vocais dos indivíduos, tais como máxima frequência fundamental da voz e variações de amplitude. A coluna `status` indica zero (0) para amostras de pacientes saudáveis e um (1) para aqueles com Parkinson. O conjunto foi manipulado em memória com a ajuda da biblioteca ***pandas*** [6].

[1] `https://archive.ics.uci.edu/ml/datasets/parkinsons`

A Figura 1 revela um desequilíbrio entre a quantidade de amostras de pacientes saudáveis (48) e aqueles com Doença de Parkinson (147). Esse é um problema comum em muitos problemas de classificação que pode gerar consequências indesejáveis, como:

1. A definição de um conjunto de testes fica compremetida, pois a técnica de amostragem usada pode desfavorecer a classe desbalanceada.

2. Quando usado em tempo real, o modelo pode ficar não-otimizado na tarefa de classificação de novas amostras da classe desbalanceada.

Do ponto de vista de distribuições normais, a Figura 2 mostra que alguns atributos (i.e Jitter:DDP, NHR, Shimmer:DDA) possuem distribuições assimétrias. Um pré-processamento que remove assimetrias pode ser efetuada afim de tornar os dados mais gaussianos.

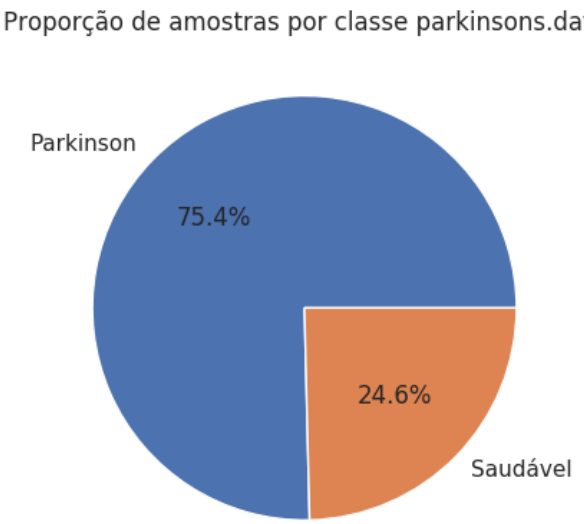

Figura 1: Número de amostras por classe

A análise dos atributos pode ajudar na montagem de vetores de característica otimizados para a classificação. Métricas estatísticas simples podem revelar muito sobre os dados. A Tabela 1 traz média, desvio padrão e assimetria para cada um dos 22 atributos do problema. Os três primeiros atributos listados possuem as maiores média e desvio padrão.

A Figura 3 mostra, em formato de mapa de calor, a correlação entre cada par de atributos. Altas correlações, sejam positivas ou negativas, representam redundância na capacidade descritiva dos atributos. Para melhor visualização, esse mapa mostra os valores absolutos das correlações. Quanto mais próximo de 1, mais correlato é aquele par. Analisando juntamente com a Tabela 1, vemos

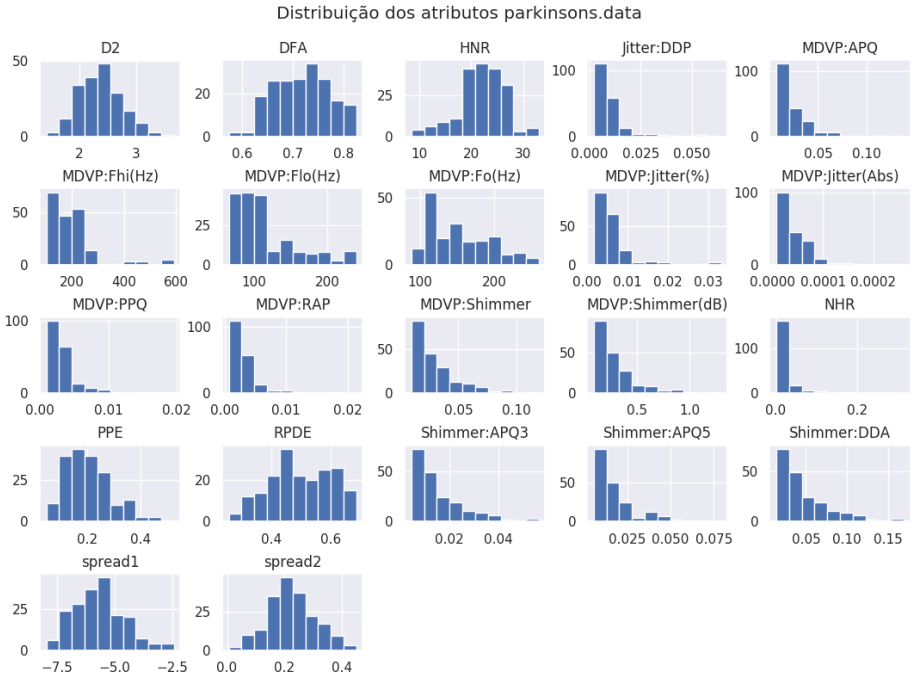

Figura 2: Distribuição dos dados

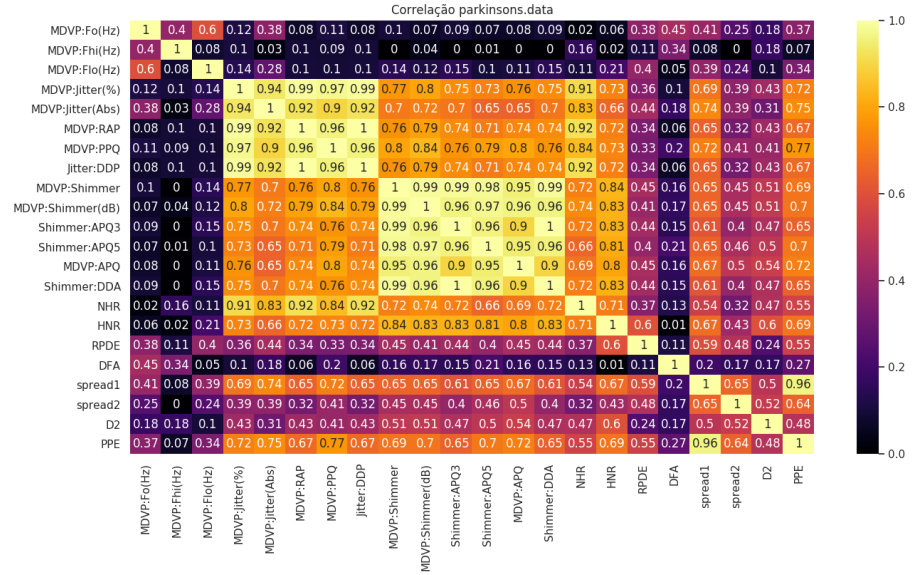

Figura 3: Correlação entre os atributos

que esse conjunto de atributos super correlatos coincide com aqueles de baixo valor de média e desvio padrão.

Atributo	**Média**	**Mediana**	**Min**	**Max**	**Desvio**
Atributo	Média	Mediana	Min	Max	Desvio
MDVP:Fo(Hz)	154.23	148.79	88.33	260.1	41.39
MDVP:Fhi(Hz)	197.1	175.83	102.14	592.03	91.49
MDVP:Flo(Hz)	116.32	104.32	65.48	239.17	43.52
MDVP:Jitter(%)	0.01	0.0	0.0	0.03	0.0
MDVP:Jitter(Abs)	0.0	0.0	0.0	0.0	0.0
MDVP:RAP	0.0	0.0	0.0	0.02	0.0
MDVP:PPQ	0.0	0.0	0.0	0.02	0.0
Jitter:DDP	0.01	0.01	0.0	0.06	0.01
MDVP:Shimmer	0.03	0.02	0.01	0.12	0.02
MDVP:Shimmer(dB)	0.28	0.22	0.08	1.3	0.19
Shimmer:APQ3	0.02	0.01	0.0	0.06	0.01
Shimmer:APQ5	0.02	0.01	0.01	0.08	0.01
MDVP:APQ	0.02	0.02	0.01	0.14	0.02
Shimmer:DDA	0.05	0.04	0.01	0.17	0.03
NHR	0.02	0.01	0.0	0.31	0.04
HNR	21.89	22.08	8.44	33.05	4.43
RPDE	0.5	0.5	0.26	0.69	0.1
DFA	0.72	0.72	0.57	0.83	0.06
spread1	-5.68	-5.72	-7.96	-2.43	1.09
spread2	0.23	0.22	0.01	0.45	0.08
D2	2.38	2.36	1.42	3.67	0.38
PPE	0.21	0.19	0.04	0.53	0.09

Tabela 1: Estatísticas gerais dos atributos

3 Metodologia

Foram comparados os desempenhos de três classificadores: Vizinho mais Próximo (NN), Distância Mínima ao Centróide (DMC) e Classificador Quadrático Gaussiano (CQG). Dois cenários de amostragem foram escolhidos: um com os dados originais e outro com uma redução de dimensionalidade usando Análise de Componentes Principais (PCA) e Análise de Discriminates Lineares (LDA). A comparação de desempenho foi realizada em termos de média, mediana, valores mínimo/máximo, desvio padrão, sensibildiade e especificidade. O ambiente de experimentação foi implementado em Python 3.5 com o auxílio da biblioteca de aprendizagem de máquina ***scikit-learn*** [7]. Todos os testes foram executados em uma máquina Debian GNU/Linux 9.5 com 8GB de RAM e processador i7-3770 @3.40GHz x4.

3.1 Classificador Vizinho Mais Próximo (NN)

O classificador Vizinho Mais Próximo (NN) é um classificador simples, porém muito poderoso. Baseado em distâncias, seu algoritmo consiste em [8]:

Passo 1: Armazenar em memória o conjunto de treinamento X
Passo 2: Para cada nova observação x, buscar em X pelo índice do vetor de

atributos mais próximo de x:

$$i^* = arg \min_{i=1,...,N} ||x - x_i||^2 \quad (1)$$

onde $||v||$ denota a norma euclidiana do vetor v.

Passo 3: Atribuir à observação x o índice de x_{i^*}

A Eq.1 pode ser modificada para usar normas de qualquer ordem, não apenas a euclidiana. Uma outra norma muito usada é a Manhattam (Minkowski de ordem 1), mostrada na Eq.2:

$$d(x, x_i) = \sum_{i=1}^{N} |x - x_i| \quad (2)$$

A implementação do NN no ***scikit-learn*** traz diversas otimizações, incluindo o particionamento do espaço de atributos usando *KD-Tree* [9] ou *Ball-Tree* [10] para acelerar a busca do vizinho mais próximo. A implementação do NN está contida na classe `sklearn.neighbors.KNeighborsClassifier`.

3.2 Classificador Distância Mínima ao Centróide (DMC)

O classificador Distância Mínima ao Centróide tem um funcionamento muito semelhante ao NN. A ideia agora é calcular a distância de uma nova observão aos centróides das massas de dados de cada classe. Dada uma nova observação x a ser classificada, o algoritmo faz:

Passo 1: Encontrar o vetor centróide de cada classe:

$$m_i = \frac{1}{N_i} \sum_{\forall x \in \omega_i} x \quad (3)$$

onde N_i é o número de amostras da i-ésima classe, cujo rótulo é ω_i, para $i = 1, ..., K$

Passo 2: Rotular a nova observação x como da classe de m_{i^*} se:

$$||x - m_{i^*}||^2 < ||x - m_i||^2, \forall i \neq i^* \quad (4)$$

onde $||v||$ denota a norma euclidiana do vetor v.

Assim como no NN, outras normas podem ser usadas para o cálculo da distância no classificador DMC. A distância é uma métrica de dissimilaridade e o conhecimento de suas diversas formas e respectivas interpretações geométricas é chave para a construção de modelos preditivos robustos [8]. A implementação do DMC está contida na classe `sklearn.neighbors.NearestCentroid`.

3.3 Classificador Quadrático Gaussiano (CQG)

A Regra de Bayes, ponto de partida para a demonstração do CQG, responde à seguinte pergunta: "de posse dos exemplos já observados X, qual a probabilidade de uma nova observação x pertencer à classe w_i, ou seja, quanto vale $p(w_i|x)$?"[11].

Seja $p(w_i)$ a probabilidade *a priori* de uma observação pertencer a w_i, $p(x)$ a probabilidade dos atributos e assumindo que os exemplos respeitam uma distribuição de probabilidade normal multivariada $p(x|w_i)$ de média μ_i e matriz de covariância Σ_i (também chamada função de verossimilhança), temos [12]:

$$p(x|w_i) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^-1(x-\mu_i)\right\} \tag{5}$$

$$p(w_i|x) = \frac{p(w_i)p(x|w_i)}{p(x)} \tag{6}$$

A classificação da nova observação x pode ser feita usando a regra da *máxima a posteriori*, ou critério MAP: x pertencerá à classe w_j se $p(w_j|x)$ for a maior densidade entre todas as K classes, ou seja:

$$w_j = arg\max_{i=1,...,K} p(w_i|x) \tag{7}$$

Uma forma mais geral do critério MAP pode ser escrita a partir de qualquer função que retorne o grau de pertinência da observação x a uma classe w_i, chamada *função discriminate* $g_i(x)$. Uma função discriminate muito utilizada é derivada a partir do logaritmo natural aplicado à Eq.5:

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^-1(x-\mu_i) - \frac{1}{2}\ln|\Sigma_i| + \ln p(w_i) \tag{8}$$

Assumindo que as classes são equiprováveis ($p(w_1) = p(w_2) = ... = p(w_k)$), o termo $\ln p(w_i)$ se torna irrelevante. Distribuindo os termos restantes do lado direito da Eq.8, temos:

$$\begin{aligned}
g_i(x) &= -\frac{1}{2}(x-\mu_i)^T\Sigma_i^-1(x-\mu_i) - \frac{1}{2}\ln|\Sigma_i| \\
g_i(x) &= -\frac{1}{2}[x^T\Sigma_i^-1x - x^T\Sigma_i^-1\mu_i - \mu_i^T\Sigma_i^-1x + m_i^T\Sigma_i^-1\mu_i] - \frac{1}{2}\ln|\Sigma_i| \\
g_i(x) &= -\frac{1}{2}[x^T\Sigma_i^-1x - 2\mu_i^T\Sigma_i^-1x + \mu_i^T\Sigma_i^-1\mu_i] - \frac{1}{2}\ln|\Sigma_i|
\end{aligned} \tag{9}$$

Podemos ver que a derivação final da Eq.9 representa a equação de um hiperparabolóide. Logo, o CQG não é um classificador linear, e sim quadrático como o próprio nome sugere. A invertibilidade da matriz Σ_i é um ponto crítico no uso desse classificador, logo recomenda-se o uso de alguns métodos para avaliar a sua invertibilidade e, em caso de necessidade, estimação da inversa. A classe `sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis` contém a implementação do CQG.

3.3.1 Invertibildiade da Matriz de Covariância

Garantir a existência da inversa de Σ_i é necessário para o uso do CQG. Podemos avaliar se uma matriz é invertível ou não através do seu posto.

O posto de uma matriz é o seu número máximo de colunas linearmente independentes. Dada uma matriz quadrada $M_{(p,p)}$, M é invertível se e somente se o posto de M for igual a p, ou seja, se diz que *"M tem posto completo"*. A biblioteca ***NumPy*** [13] possui um pacote de álgebra linear chamado `numpy.linalg` e neste encontra-se a função `rank()` para avaliar o posto de uma matriz.

3.4 Análise de Componentes Principais (PCA)

A Análise de Componenetes Principais (PCA) é uma técnica usada para dois fins: remoção da correlação entre os atributos e redução de dimensionalidade. O príncipio básico do PCA é decompor um conjunto multivariado em um conjunto de componentes ortogonais sucessivas, ordenadas da maior para a menor variância explicada. Podemos resumir o algoritmo do PCA como sendo a aplicação de uma transformação linear nos dados originais a partir de uma matriz montada pelos autovetores ordenados da matriz de covariância dos dados normalizados.

Seja um conjunto de dados X formado por N observações x_k, $k = 1, ..., N$, cada uma com p atributos. Desejamos transformar o conjunto X em um conjunto Y cujas observações possuem dimensão q, com $q \leq p$. Para isso, devemos:

Passo 1: Centralizar os dados do conjunto X:

$$\begin{aligned} \bar{x} &= \frac{1}{N}\sum_{k=1}^{N} x_k \\ x_k &= x_k - \bar{x}, k = 1, ..., N \end{aligned} \tag{10}$$

Passo 2: Estimar a matriz de covariância de X:

$$\begin{aligned} C_x &= E[xx^T] \\ C_x &\approx \frac{1}{N}\sum_{k=1}^{N} x_k x_k^T \end{aligned} \tag{11}$$

onde $E[.]$ é o operador valor esperado.

Passo 3: Determinar os p autovalores e autovetores da matrix C_x, o que significa resolver o seguinte sistema de equações:

$$C_x v = \lambda v \tag{12}$$

onde λ e v são os autovalores e autovetores, respectivamente.

Passo 4: Ordenar os autovalores em modo decrescente:

$$\lambda_1 > \lambda_2 > ... > \lambda_p \tag{13}$$

Passo 5: Montar a matriz V agrupando os autovetores nas colunas seguindo a mesma ordenação dos seus respectivos autovalores:

$$V = [v_1|v_2|...|v_p] \tag{14}$$

Passo 6: Por fim, aplicar a seguinte transformação no conjunto X gerando o novo conjunto Y:

$$Y = V^T X \tag{15}$$

É possível mostrar que a matriz de covariância dos dados transformados Y é diagonal, ou seja, os dados nesse novo espaço estão descorrelacionados. Podemos também reduzir o número de autovetores geradores da matriz V, satisfazendo $q \leq p$, o que gera uma matriz de transformação $V_{(p,q)}$ e, consequentemente, uma matriz de dados transformados $Y_{(q,q)}$, o que mostra a capacidade do PCA de ser uma técnica de redução de dimensionalidade [14]. A funcionalidade do PCA está implementada em `sklearn.decomposition.PCA`.

3.5 Análise de Discriminates Lineares (LDA)

A Análise de Discriminates Lineares (LDA), ou Discriminate de Fisher, é uma técnica que procura uma combinação linear dos dados de forma a definir superfície que separa as classes o máximo possível. Fisher percebeu que isso era possível quando a razão entre a variância intraclasse e interclasse é máxima [15].

Em termos de algoritmo, o LDA se assemelha muito ao PCA: procura-se uma matriz que carrega informações dos dados, resolve-se o problema do autovalor/autovetor em cima dela e, ao final, constrói-se uma matriz de transformação com os autovetores agregados em ordem. Precisamos saber qual matriz usar; no PCA, usamos a matriz de covariância dos dados normalizados C_x, já no LDA usarmos a matriz $S_W^{-1}S_B$ (inversa da matriz de dispersão intraclasse S_W pela matriz de dispersão interclasse S_B).

$$S_W = N_1C_1 + N_2C_2 + ... + N_KC_K = \sum_{k=1}^{K} N_kC_k \tag{16}$$

$$S_B = \sum_{k=1}^{K} (m_k - m)(m_k - m)^T \tag{17}$$

Nas Eq.16 e Eq.17 temos N_k como a quantidade de amostra da k-ésima classe, C_k a matriz de covariância da k-ésima classe, m_k o centróide da k-ésima classe e m o centróide do conjunto total de dados. Podemos, então executar os mesmos passos 4, 5 e 6 do PCA e teremos a matriz de transformação V para o LDA [16]. O LDA pode ser utilizado através da implementação `sklearn.discriminant_analysis.LinearDiscriminantAnalysis`.

3.6 Normalização

De forma a melhorar a estabilidade de certos cálculos, um dos passos de pré-processamento mais simples e utilizado em *pipelines* de predição é a normalização, também conhecida como escalamento ou padronização [11].

Para normalizar um conjunto de dados, primeiro fazemos uma transformação de centralização, subtraindo de cada valor de um atributo k a média dos valores do mesmo $\bar{x_k}$. Depois, uma operação de escala é efetuada dividindo todos os valores de cada atributo x_k pelo desvio padrão σ_k. Ambos os passos podem ser descritos com uma fórmula, mostrada em Eq.18.

$$x_k^* = \frac{x_k - \bar{x_k}}{\sigma_k} \tag{18}$$

A transformação de normalização é implementada no ***scikit-learn*** pela classe `sklearn.preprocessing.StandardScaler`.

3.7 Remoção de Assimetria

Um distribuição de dados é dita simétrica quando a probabilidade de um evento ocorrer de um lado da distribuição é praticamente a mesmo que na região equidistante oposta. A assimetria de uma distribuição é cáculada usando a Eq.19, onde x é o atributo, n o número de valores e $\bar{x}$ é a média do atributo. Um distribuição possui *assimetria à direita* quando apresenta uma maior densidade

de valores no lado esquerdo que do lado direito, e o seu valor é positivo. O análogo também serve para a *assimetria à esquerda* e o seu valor é negativo.

$$\begin{aligned} assimetria &= \frac{\sum(x_i - \bar{x})^3}{(n-1)v^{3/2}} \\ v &= \frac{\sum(x_i - \bar{x})^2}{(n-1)} \end{aligned} \tag{19}$$

Existe uma família de transformações que podem ser usadas para remover assimetria de dados. Aquela escolhida para esse trabalho foi a Yeo-Johnson [17], mostrada na Eq.20, onde λ é um parâmetro que define a transformação (pode ser estimado a partir de conjunto de treinamento [11]):

$$x^* = \begin{cases} ((x+1)^\lambda - 1)/\lambda & \text{se } \lambda \neq 0,\ x \geq 0 \\ \log(x+1)\text{se} & \text{se } \lambda = 0,\ x \geq 0 \\ -[(-x+1)^{2-\lambda} - 1]/(2-\lambda) & \text{se } \lambda \neq 2,\ x < 0 \\ -\log(-x+1) & \text{se } \lambda = 2,\ x < 0 \end{cases} \tag{20}$$

A transformação de remoção de assimetria é implementada no ***scikit-learn*** pela classe `sklearn.preprocessing.PowerTransformer`.

3.8 Avaliação de Performance dos Classificadores

O objetivo deste trabalho é o diagnóstico ou não da Doença de Parkinson. Portanto, podemos dizer que trata-se de um problema de classificação binária. Podemos agrupar o resultado de um conjunto de predições de um dado classificador na forma de uma Matriz de Confusão (Tabela 2). A matriz de confusão agrega de forma gráfica a quantidade de eventos preditos correta e incorretamente. Podemos extrair:

		Condição Real		
		Parkinson (1)	Saudável (0)	Total
Predição	Parkinson (1)	VP	FP	VP + FP
	Saudável (0)	FN	VN	FN + VN
	Total	VP + FN	FP + VN	N

Tabela 2: Matriz de confusão e seus elementos

- **Verdadeiros Positivos (VP):** pacientes com Parkinson corretamente diagnosticados com a doença;
- **Verdadeiros Negativos (VN):** pacientes saudáveis corretamente diagnosticados sem a doença;
- **Falsos Positivos (FP):** pacientes saudáveis diagnosticados incorretamente com a doença; e
- **Falsos Negativos (FN):** pacientes com Parkinson dignosticados incorretamente como saudáveis.

Precisão é a taxa de acerto geral (Eq.21). Sensibilidade mede o quanto as observações Verdadeira Positivas foram corretamente classificadas como tal (Eq.22). Especificidade é o análogo da sensibilidade, mas em relação aos Verdadeiros Negativos (Eq.23).

$$A = \frac{VP + VN}{VP + VN + FP + FN} \tag{21}$$

$$SENS = \frac{VP}{VP + FN} \tag{22}$$

$$SPEC = \frac{VN}{VN + FP} \tag{23}$$

A modelagem em classes do ***scikit-learn*** implementa, para todos os classificadores, um método `fit()` para treinamento, `score()` para teste e `predict()` para predição de novas observações. A função `sklearn.metrics.confusion_matrix` foi usada para extrair os valores de VP, VN, FP e FN.

4 Resultados

Apresentaremos agora os resultados comparativos entre os classificadores escolhidos e as técnicas de redução de dimensionalidade. Dois *scripts* Python foram criados: `recpad.py`, que agrega diversas funções utilitárias para classificação e visualização, e `tc1.py`, sequência do passo-a-passo de geração de resultados deste trabalho. Pedaços principais de código serão apresentados ao longo dos resultados. Para uma avaliação das implementações, ver arquivo `recpad.py`.

Todas as experimentações foram repetidos 100 vezes, os resultados de cada rodada armazenados e um conjunto de estatísticas, analisado. A proporção 70/30 entre amostras de treinamento e teste foi mantida constante em todos os cenários abordados.

```
from recpad import *

classifiers = {"NN"   : NN(n_neighbors=1),
               "DMC"  : DMC(),
               "CQG"  : CQG(store_covariance=True)}

dataset = "data/parkinsons.data"
test_s = 0.3
rounds = 100

df = pd.read_csv(dataset)
df = df.drop(["name"], axis=1)
X = df.drop(["status"], axis=1)
y = df["status"]

X0 = df.loc[df["status"] == 0]
X0 = X0.drop(["status"], axis=1)
X1 = df.loc[df["status"] == 1]
X1 = X1.drop(["status"], axis=1)

X_trans = super_normalize(X)
X_trans = super_unskew(X_trans)
X_trans = pd.DataFrame(X_trans, columns=X.columns)
```

Listing 1: Cabeçalho do código trabalho01

4.1 Invertibilidade da Matriz de Covariância C_X

Primeiro, calculamos a matriz de covariância de cada classe aplicando a função `numpy.cov`. Em seguida, usamos a função `numpy.linalg.matrix_rank` para calcular o posto. Vemos que a matriz C_X de ambas as classes tem posto 19. Como a dimensão dessas matrizes é $(22, 22)$, concluimos que elas não têm posto completo. Logo, não são inversíveis.

Para contornar esse problema, o ***scikit-learn*** usa *Singular Value Decomposition* (SVD) para gerar uma matriz diagonal ao invês da matriz de covariância. Em poucas palavras, o SVD representa a matriz de dados em um novo espaço onde a matriz de covariância é diagonal, ou seja, os atributos nesse espaço são descorrelacionados [18]. Outra alternativa é usar *métodos de encolhimento* [19], mas ambas as técnicas não podem ser usadas ao mesmo tempo por questões de implementação.

```
print("Posto cov C0: {}".format(rank_covmatrix(X0)))
print("Posto cov C1: {}".format(rank_covmatrix(X1)))

X0_norm = super_normalize(X0)
X1_norm = super_normalize(X1)
print("Posto cov C0 norm: {}".format(rank_covmatrix(X0_norm)))
print("Posto cov C1 norm: {}".format(rank_covmatrix(X1_norm)))

X0_unskew = super_normalize(X0)
X1_unskew = super_normalize(X1)
print("Posto cov C0 unskew: {}".format(rank_covmatrix(X0_unskew)))
print("Posto cov C1 unskew: {}".format(rank_covmatrix(X1_unskew)))
```

Listing 2: Calculando matrizes de covariância e seus postos

4.2 Desempenho com Dados Originais

A Tabela 3 mostra as taxas extraídas dos classificadores usando o conjunto original, sem qualquer pré-processamento. Podemos ver que o CQG venceu o NN por uma pequena diferença em termos de precisão média. Entretanto, o CQG apresentou um desequilíbrio entre sensibilidade e especificidade; ele se mostrou é um bom classificador para detectar casos de corretos de Parkinson $(98, 73\%)$, mas não muito confiável no diagnóstico de pacientes saudáveis $(49, 39\%)$.

```
ans = classify(classifiers, X, y, test_s, rounds)
sumary(ans, "Desempenho geral")
ans = classify(classifiers, X, y, test_s, rounds, nm=True, uk=True)
sumary(ans, "Desempenho geral (normalizado, unskew)")
```

Listing 3: Desempenho dos classificadores

Classif	Média	Mediana	Min/Max	Desvio	Sens	Espec
NN	83,12	83,05	71,19/94,92	4,75	89,22	66,22
DMC	71,15	71,19	61,02/86,44	5,00	74,36	62,8
CQG	85,88	88,14	67,80/96,61	5,93	98,73	49,92

Tabela 3: Desempenho dos classificadores (dados originais)

4.3 Desempenho com redução via PCA

A Tabela 4 traz a visão geral dos classificadores em estudo, dessa vez com o conjunto de dados reduzido usando PCA. O número de componentes q usado para cada resultado está entre parênteses ao lado do nome do classificador, respeitando $1 \leq q \leq p$, com $p = 22$. Foram testados todos os possíveis valores de q para cada um dos classificadores. A escolha da dimensão reduzida foi feita em termos de precisão. Analisando os melhores valores de q para cada modelo, o CQG permaneceu como melhor classificador $(88,24\%)$, seguido por NN $(84,37\%)$ e DMC $(72,36\%)$.

A Figura 4 mostra a evolução das taxas de cada classificador em função do número de componentes mantidas. No geral, todos os classificadores se valeram da redução de dimensionalidade e obtiveram melhores taxas de acerto. Podemos ver que o NN alcança sua melhor taxa média com 12 componentes, o DMC com 16 e o CQG, 18.

O DMC aprensetou um resultado interessante. Ele oscilou em torno do resultado geral para qualquer valor de q, mostrando constância nas taxas de acerto independente da redução de dimensionaliade. Como esse classificador trabalha com o centróide da massa de pontos, é natural pensar que, se os dados forem reduzidos igualmente, a separação (e.g a distância) entre os centróides transformados permanecerão a mesma.

Classif	Média	Mediana	Min/Max	Desvio	Sens	Espec
NN (12)	84,37	84,75	71,19/94,92	4,31	89,13	70,30
DMC (16)	72,36	72,88	59,32/88,14	4,96	75,56	62,27
CQG (18)	88,24	88,14	74,58/96,61	4,23	96,04	64,86

Tabela 4: Desempenho dos classificadores (redução dos dados com PCA)

```
find_best_pca(dataset, classifiers, test_s, "prec-pca-geral.png")
```

Listing 4: Desempenho com PCA

4.4 Desempenho com redução via LDA

Classif	Média	Mediana	Min/Max	Desvio	Sens	Espec
NN+LDA	85,27	85,59	74,58/91,53	3,62	89,84	71,59
DMC+LDA	85,59	86,44	76,27/91,53	3,62	89,84	71,59
CQG+LDA	91,20	91,53	84,75/96,61	2,62	97,28	72,50

Tabela 5: Desempenho dos classificadores (redução dos dados com LDA)

A Tabela 5 traz o cenário final de experimentação, onde os classificadores foram submetidos ao conjunto de dados transformados usando LDA. O DMC valeu-se sobremaneira com o usdo do LDA, alcançando uma taxa de precisão média de $85,59\%$. Ele obteve empate técnico om o NN $(85,27\%)$. O CQG evoluiu em perfornamence, tanto em taxa de precisão $(91,20\%)$, quanto na relação sensibilidade/especificidade de $97,28\%/72,50\%$.

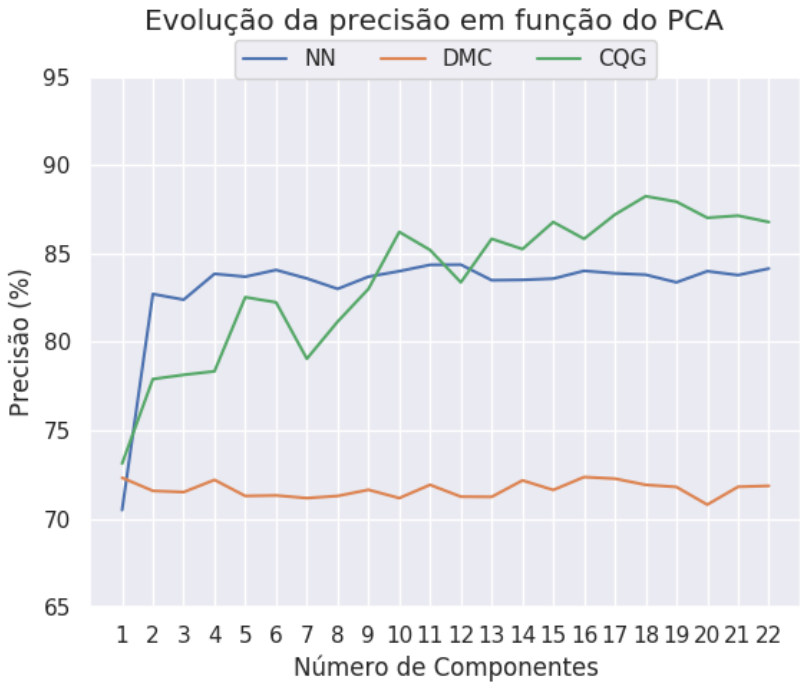

Figura 4: Precisão dos classificadores em função do número de componentes mantidas usando PCA

```
X_lda = reduction_lda(X, y, 1)
ans = classify(classifiers, X_lda, y, test_s, rounds)
sumary(ans, "Desempenho LDA")
```

Listing 5: Desempenho com LDA

4.5 Desempenho com Dados Pré-Processados

```
ans = classify(classifiers, X, y, test_s, rounds, nm=True, uk=True)
sumary(ans, "Desempenho geral (normalizado, unskew)")

find_best_pca(dataset, classifiers, test_s, "prec-pca-trans.png",
    nm=True, uk=True)

X_lda = reduction_lda(X, y, 1)
ans = classify(classifiers, X_lda, y, test_s, rounds, nm=True,
    uk=True)
sumary(ans, "Desempenho LDA (normalizado, unskew)")
```

Listing 6: Desempenho com normalização

Os mesmos ensaios mostrados até agora foram repetidos para o conjunto de dados resultante das transformações de normalização e remoção de assimetria. As matrizes de covariâncias das classes com posto igual a 22, ou seja, posto completo. A Figur 5 mostra que as tranformações de normalização e remoção de assimetria tornaram a distribuições de todos os atributos mais próximas de curvas gaussianas. A Tabela 6 traz os resultados com e sem redução de

dimensionalidade. A Figura 6 traz a evolução da taxa máxima precisão em função do número de componentes q mantidas sobre o conjunto transformado.

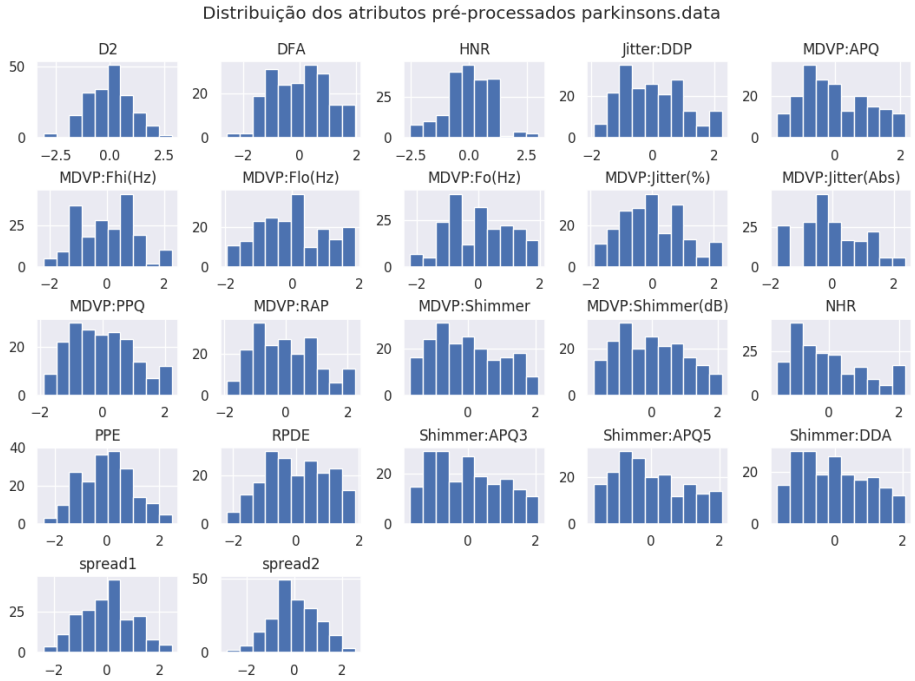

Figura 5: Distribuição dos atributos pré-processados

Classif	**Média**	**Mediana**	**Min/Max**	**Desvio**	**Sens**	**Espec**
NN	91,64	91,53	84,75/100,00	3,06	94,03	84,82
DMC	73,46	72,88	61,02/89,83	4,99	71,74	78,69
CQG	84,98	84,75	64,41/98,31	6,48	99,42	43,50
NN (20)	89,86	89,83	74,58/96,61	3,84	9,02	83,56
DMC (20)	79,59	79,66	67,80/89,83	5,18	81,86	72,89
CQG (18)	88,07	88,14	77,97/98,31	4,27	97,27	60,59
NN+LDA	85,49	86,44	77,97/93,22	3,58	90,06	72,49
DMC+LDA	83,17	83,05	71,19/96,61	4,37	82,90	84,04
CQG+LDA	91,07	91,53	83,05/98,31	3,21	96,99	73,59

Tabela 6: Desempenho dos classificadores (dados pré-processados)

O melhor resultado obtido foi o do NN, com $91,64\%$ e melhor relação sensibilidade/especificidade e menor desvio padrão, com dados sem redução de dimensionalidade, tornando-a a melhor dentre todas as apresentadas.

5 Conclusões

O uso de técnicas de pré-processamento e redução de dimensionalidade são ferramentas muito úteis na construção de modelos de predição. Foi apresentado um

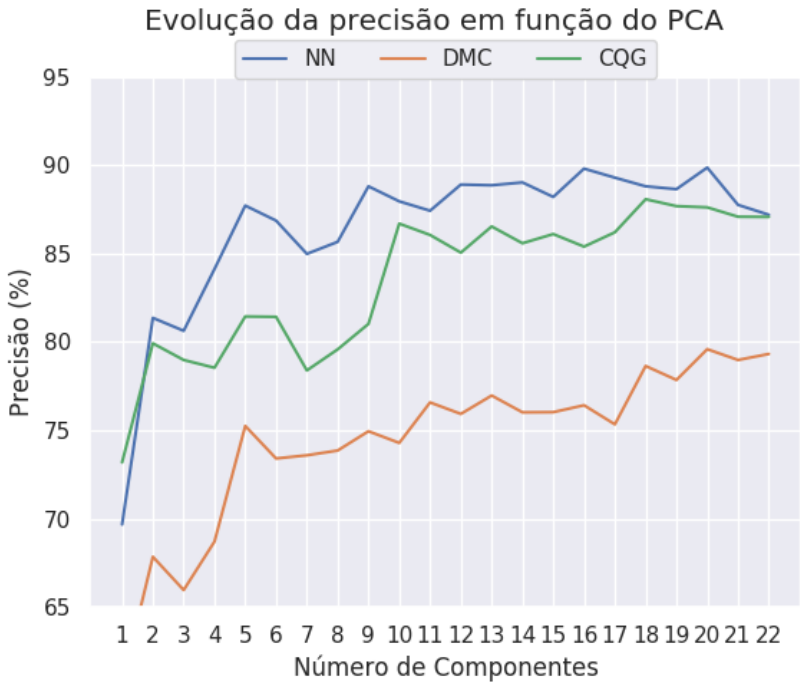

Figura 6: Desempenho com dados normalizados, tratadas as assimetrias e aplicação de PCA (evolução dos valores de q)

conjunto de comparações onde classificadores e métodos de transformação foram combinados e analisados afim de ajudar no diagnóstico da Doença de Parkison. O melhor resultado obtido foi com o Classificador Vizinho Mais Próximo na configuração de dados pré-processados (normalização e remoção de assimetria), sem redução de dimensionalidade, atingindo uma taxa de precisão de $91,64\%$ e relação sensibilidade/especificidade de $94,03\%/84,82\%$. Esse resultado se assemelha ao da bibliografia de referência [4], mas com uma taxa de especificidade menor.

Referências

[1] National Institute of Neurological Disorders and Stroke. *Parkinson's Disease Information Page (https://www.ninds.nih.gov/Disorders/All-Disorders/Parkinsons-Disease-Information-Page)*, 2016 (acessado 01 de dezembro de 2018).

[2] Akin Ozcift and Arif Gulten. Classifier ensemble construction with rotation forest to improve medical diagnosis performance of machine learning algorithms. *Computer methods and programs in biomedicine*, 104(3):443–451, 2011.

[3] Hui-Ling Chen, Chang-Cheng Huang, Xin-Gang Yu, Xin Xu, Xin Sun, Gang Wang, and Su-Jing Wang. An efficient diagnosis system for detection of parkinson's disease using fuzzy k-nearest neighbor approach. *Expert systems with applications*, 40(1):263–271, 2013.

[4] Christian Salvatore, Antonio Cerasa, Isabella Castiglioni, F Gallivanone, A Augimeri, M Lopez, G Arabia, M Morelli, MC Gilardi, and A Quattrone. Machine learning on brain mri data for differential diagnosis of parkinson's disease and progressive supranuclear palsy. *Journal of Neuroscience Methods*, 222:230–237, 2014.

[5] Max A Little, Patrick E McSharry, Eric J Hunter, Jennifer Spielman, Lorraine O Ramig, et al. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE transactions on biomedical engineering*, 56(4):1015–1022, 2009.

[6] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Guilherme de Alencar Barreto. Introdução à classificação de padrões. Slides da disciplina TIP8311 - Reconhecimento de Padrões, 2018.

[9] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[10] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[11] Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

[12] Guilherme de Alencar Barreto. Critério map e classificadores gaussianos. Notas de aula sobre classificadores gaussianos, 2014.

[13] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[14] Guilherme de Alencar Barreto. Exemplo de aplicação de transformações lineares: Análise das componentes principais. Slides da disciplina TIP8311 - Reconhecimento de Padrões, 2018.

[15] Andrew R Webb. *Statistical pattern recognition*. John Wiley & Sons, 2003.

[16] Guilherme de Alencar Barreto. Exemplo de aplicação de transformações lineares: Discriminante linear de fisher. Slides da disciplina TIP8311 - Reconhecimento de Padrões, 2018.

[17] In-Kwon Yeo and Richard A Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000.

[18] Desconhecido. *Singular Value Decomposition (SVD) tutorial (http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm)*, 2002 (acessado 02 de dezembro de 2018).

[19] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.