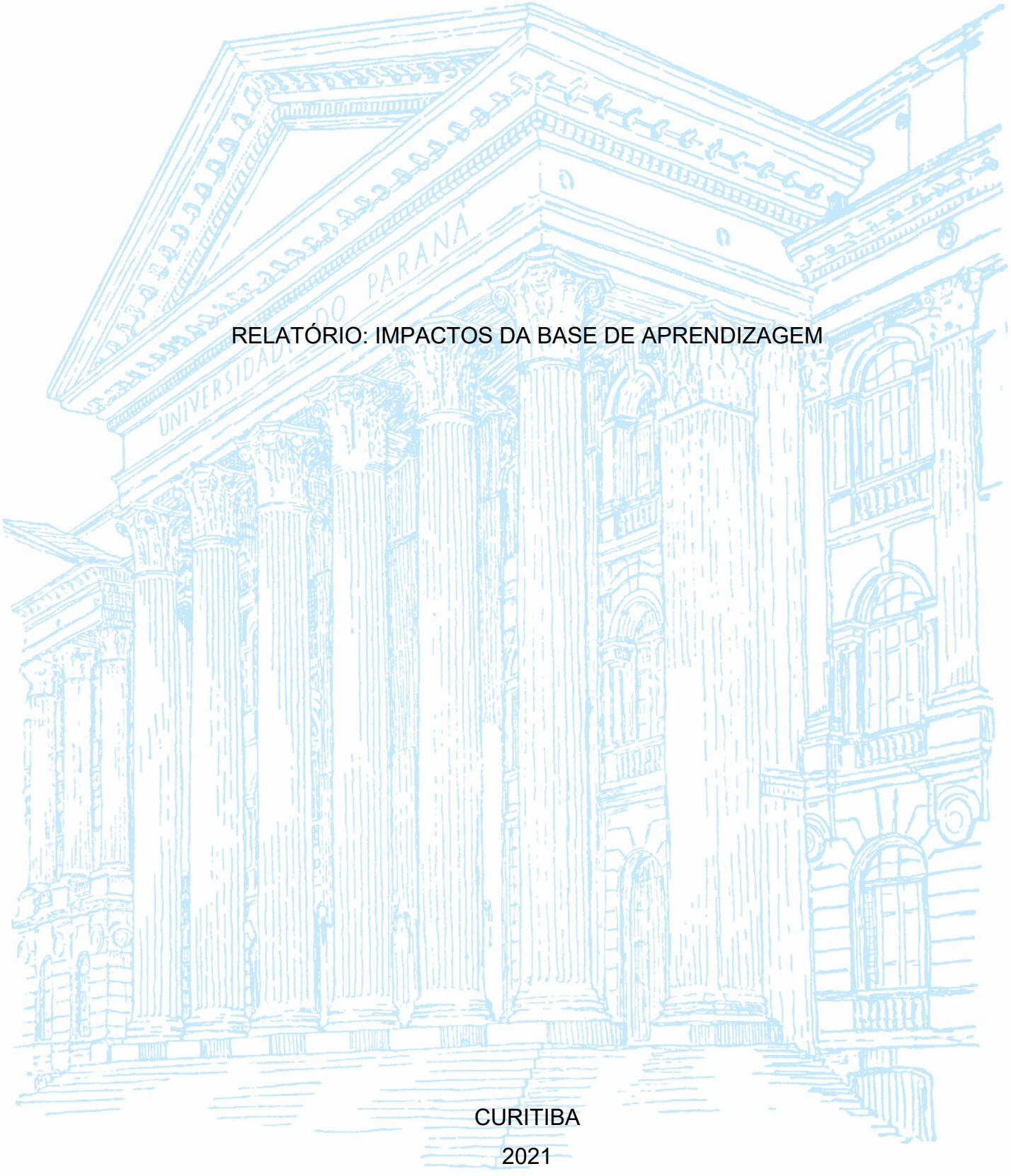
UNIVERSIDADE FEDERAL DO PARANÁ

MATEUS FELIPE DE CÁSSIO FERREIRA



RELATÓRIO: IMPACTOS DA BASE DE APRENDIZAGEM

CURITIBA

2021

MATEUS FELIPE DE CÁSSIO FERREIRA

RELATÓRIO: IMPACTOS DA BASE DE APRENDIZAGEM

Relatório apresentado como requisito parcial à conclusão da disciplina CI171 – Aprendizagem de Máquina, no Curso de Bacharelado em Informática Biomédica, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Orientador: Prof. Dr. Luiz Eduardo Soares de Oliveira

CURITIBA

2021

SUMÁRIO

1 INTRODUÇÃO

Este relatório busca apresentar os resultados obtidos referentes ao segundo Laboratório da disciplina de CI171 – Aprendizagem de Máquina, que consiste em identificar os impactos da quantidade de exemplos da base de aprendizagem na acurácia final de um classificador.

O objetivo deste relatório é o de analisar o desempenho de cinco classificadores em uma base de teste, alimentando o treinamento do classificador de 1.000 em 1.000 blocos, até chegar ao seu limite. Ainda, pretende-se identificar o classificador que tem o melhor desempenho para poucos e todos os dados, além de identificar o classificador mais rápido para classificar os 58.646 exemplos na base de teste.

2 METODOLOGIA

Este relatório tomou o arquivo *train.txt* como base de treinamento, com um total de 20.000 exemplos de 10 classes balanceadas, contendo 132 características. A base de teste do classificador, por outro lado, contém 58.646 exemplos.

Os blocos de representação da base de treinamento foram obtidos sequencialmente após a leitura da base. A Figura 1 apresenta a forma de obtenção dos dados de treinamento ao longo de todos os classificadores.

FIGURA 1 – SEPARAÇÃO DO BLOCO DE TREINAMENTO

```
#carrega os dados da base de treinamento
x_data, y_data = load_svmlight_file(train)
x_train = x_data[0:train_block]
y_train = y_data[0:train_block]
```

FONTE: O autor (2021).

Por outro lado, a avaliação do tempo de classificação de toda a base de teste de um determinado classificador foi feita dentro da própria função que definia um classificador. Foi utilizado a função *timeit.default_timer()*, contida na biblioteca *timeit* para o cálculo do tempo gasto em classificação. A Figura 2 apresenta a forma em que foi calculo esse tempo ao longo de todos os classificadores.

FIGURA 2 – EXEMPLO DO CÁLCULO DO TEMPO DE CLASSIFICAÇÃO

```
#------------------MÉDIÇÃO DO TEMPO(INÍCIO)
starttime = timeit.default_timer()
# predicao do classificador
y_pred = neigh.predict(x_test)

elapsed = timeit.default_timer() - starttime
vet_time.append(elapsed)
#------------------MÉDIÇÃO DO TEMPO(FIM)
```

FONTE: O autor (2021).

Uma vez que se trata de um problema balanceado, em que o número de representações de todas as classes é bastante semelhante, a métrica utilizada para avaliar o melhor classificador para a base de treinamento será a medida de acurácia

Os classificadores que serão avaliados neste relatório são: k-Nearest Neighbors (kNN), *Naïve Bayes*, *Linear Discriminant Analysis* (LDA), *Logistic Regression* e o *Perceptron*.

2.1 K-NEAREST NEIGHBORS (KNN)

Foi construído um kNN com um número de vizinhos mais próximos (**K**) de 1 e a métrica para o cálculo de distância utilizada foi a *euclidean*. Esses dois valores são parâmetros da função *KNeighborsClassifier*(), que constrói um kNN.

2.2 NAÏVE BAYES

Foi construído um Gaussian Naive Bayes, utilizando a função *GaussianNB()* presente na biblioteca do ScikitLearn. Essa função não recebeu nenhum parâmetro adicional. Assim, por exemplo, a probabilidade *a priori* da classe foi ajustada de acordo com os dados.

2.3 LINEAR DISCRIMINANT ANALYSIS (LDA)

Foi construído um LDA utilizando a função *LinearDiscriminantAnalysis()* presente na biblioteca do ScikitLearn. Essa função não recebeu nenhum parâmetro adicional.

2.4 LOGISTIC REGRESSION

Foi construído uma Regressão Logística utilizando a função *LogisticRegression()* presente na biblioteca do ScikitLearn. Essa função recebeu, como parâmetros adicionais, o valor de tolerância (*tol* = 0.0001) e máximo de iterações (*max_iter* = 100). Para fazer um embaralhamento nos dados de treinamento, foi utilizado um valor de (*random_state = random.randint(0, 42)*).

2.5 PERCEPTRON

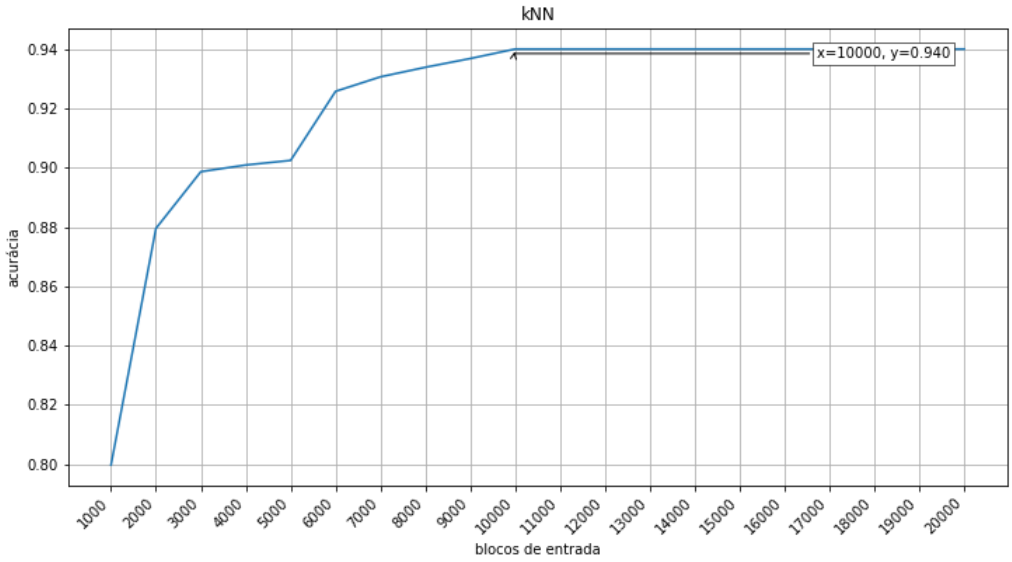
Foi construído um Perceptron utilizando a função *Perceptron()* presente na biblioteca do ScikitLearn. Essa função recebeu, como parâmetros adicionais, o valor de tolerância, máximo de iterações e embaralhamento semelhantes ao classificador *Logistic Regression.* Ainda, foi setado o valor de (*shufle* = True), número de CPUs utilizadas para o OVA (*One Versus All*) para problemas com mais de duas classes

(*n_jobs* = 4) e o número de iterações em que não há um melhoramento no treinamento (*n_iter_no_change* = 10).

3 RESULTADOS

A Figura 3 apresenta o gráfico da medida de acurácia, por bloco de entrada, para o classificador kNN. Percebe-se que a partir do tamanho do bloco 10.000 a acurácia se mantém constante em 0,940. Em um outro experimento realizado, com a mesma métrica de distância e o valor de número de vizinhos mais próximos (**K**) de 3, o classificador só atingiu o mesmo valor de acurácia para o tamanho do bloco de 20.000, ou seja, utilizando toda a base de treinamento.

FIGURA 3 – MEDIDAS DE ACURÁCIA POR BLOCO DE TREINAMENTO PARA O CLASSIFICADOR KNN



FONTE: O autor (2021).

A Figura 4 apresenta os resultados de acurácia para o classificador *Naïve Bayes*. Nota-se que, congruente ao classificador kNN, para blocos de treinamento acima de 10.000, não há um ganho de acurácia no treinamento do modelo.

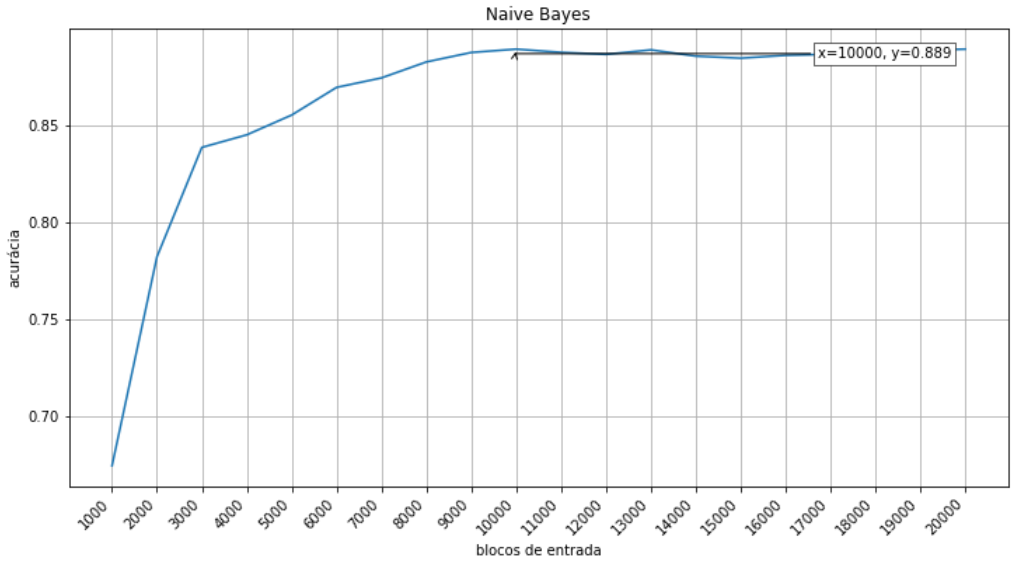
A Figura 5 apresenta os resultados de acurácia para o classificador LDA. Nota-se que, utilizando como critério uma acurácia acima de 90%, o classificador atingiu esse número com um bloco de 6.000 exemplos na base de treinamento. A partir de então, o classificador conseguiu o seu pico de acurácia, com 0,928 com o valor de bloco igual a 10.000. A partir daí, o desempenho do classificador sofreu uma queda e só voltou a subir no bloco de tamanho 20.000, com uma acurácia de 0,927.

A Figura 6 apresenta os resultados de acurácia para o classificador *Logistic Regression*. Percebe-se que, ainda utilizando o critério de acurácia acima de 90%, o classificador atingiu esse valor com um bloco de 16.000 exemplos. A partir daí, a

acurácia do modelo não sofreu grandes alterações, atingindo o seu pico de acurácia de 0,912 com um bloco de 20.000 exemplos. Ainda, uma constatação interessante para esse modelo é que a partir do tamanho de bloco com 7.000 exemplos, o modelo atinge um número máximo de iterações definidas como parâmetro para o classificador (100), levantando um aviso de convergência para o modelo, conforme apresentado na Figura 7.

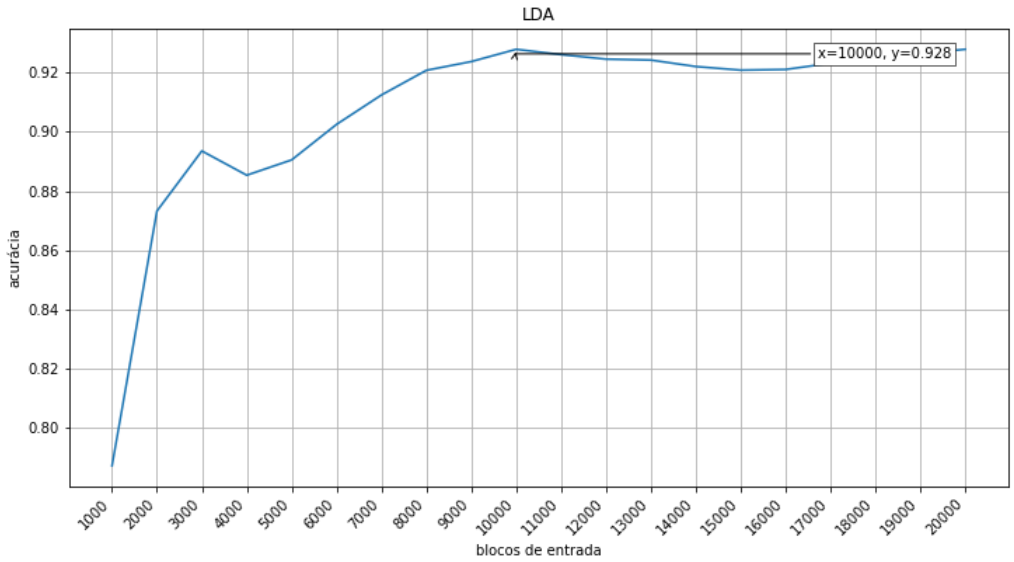
Por fim, a Figura 8 apresenta os resultados de acurácia para o classificador *Perceptron*. Para esse classificador, percebe-se uma irregularidade nos valores de acurácia apresentados, tendo o seu pico, de 0,935 com um bloco de 12.000 exemplos. A irregularidade apresentada por esse classificador pode estar associada ao fato de que o uso este classificador é indicado para problemas de apenas duas classes. Neste problema, existem 10 classes balanceadas ao todo. No entanto, apesar disso, os resultados desse classificador se mostraram satisfatórios quando comparados a outros modelos.

FIGURA 4 – MEDIDAS DE ACURÁCIA POR BLOCO DE TREINAMENTO PARA O CLASSIFICADOR NAÏVE BAYES



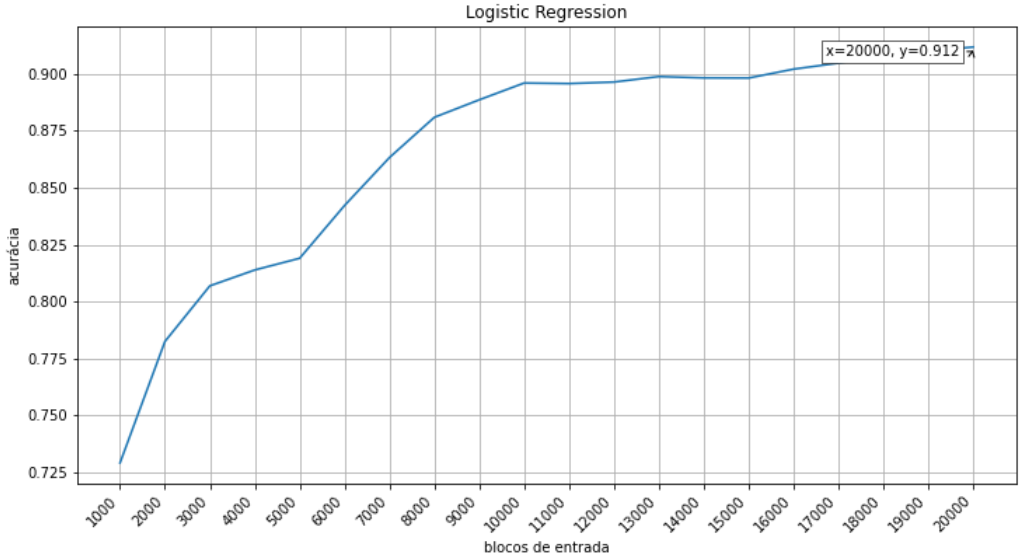
FONTE: O autor (2021).

FIGURA 5 – MEDIDAS DE ACURÁCIA POR BLOCO DE TREINAMENTO PARA O CLASSIFICADOR LDA



FONTE: O autor (2021).

FIGURA 6 – MEDIDAS DE ACURÁCIA POR BLOCO DE TREINAMENTO PARA O CLASSIFICADOR LOGISTIC REGRESSION



FONTE: O autor (2021).

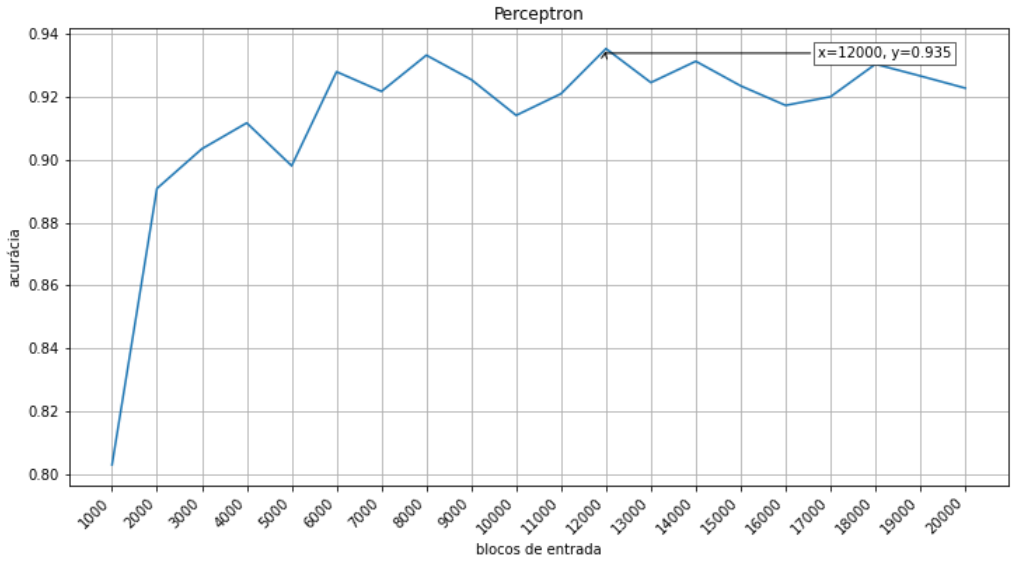
FIGURA 7 – AVISO DE CONVERGÊNCIA PARA O CLASSIFICADOR LOGISTIC REGRESSION

```
LOGISTIC block training =  7000
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

FONTE: O autor (2021).

FIGURA 8 – MEDIDAS DE ACURÁCIA POR BLOCO DE TREINAMENTO PARA O CLASSIFICADOR PERCEPTRON



FONTE: O autor (2021).

Considerando toda a base de treinamento, o algoritmo com melhor desempenho no valor de acurácia é o classificador kNN, com o valor de 0,940.

A análise das matrizes de confusão, para toda a base de treinamento nos classificadores propostos evidenciou que não são os mesmos erros cometidos por todos os classificadores ao longo da base de teste. No entanto, alguns erros dos classificadores de assemelham. A Figura 9 apresenta a matriz de confusão para o classificador kNN. Esse classificador confundiu a classificação da classe 0 com a classe 7, como mostrado em vermelho. Esse mesmo erro também foi cometido pelo classificador *Naïve Bayes*. Os outros classificadores, para o mesmo exemplo em questão, não erraram essa classificação.

FIGURA 9 – MATRIZ DE CONFUSÃO PARA O CLASSIFICADOR KNN

```
BLOCK SIZE = 20000
[[5466    3    1   12    4    2   25    2   42    3]
 [   0 6113  157  121   55   10   32   74   38   55]
 [  14   14 5616  143    4    1   16   50   27    3]
 [   4    1   22 5628    0   62    1   54   25   22]
 [  14    7   20    1 5292   21  113   29   14  211]
 [  10    3    8  377    2 4939   38   10   81   71]
 [  26    9    5    2    4   44 5723    0   45    0]
 [   1   16   45  123   47    2    1 5762    5   95]
 [  34   22   33  114   34   29   54   25 5176  174]
 [  11    8   15   93   91   11    7  119   46 5412]]
```

FONTE: O autor (2021).

No que diz respeito ao tempo de classificação de todos os 58.646 exemplos na base de teste, a Tabela 1 apresenta os resultados de acurácia e tempo de todos os classificadores, todos eles utilizando todos os 20.000 exemplos na base de treinamento. Nota-se que o classificador mais rápido é o *Perceptron*. Entre todos os classificadores propostos, o classificador de melhor "custo benefício" é o LDA, que apresenta uma acurácia de 0,928 com um tempo um pouco maior que o *Perceptron*, de 0,016.

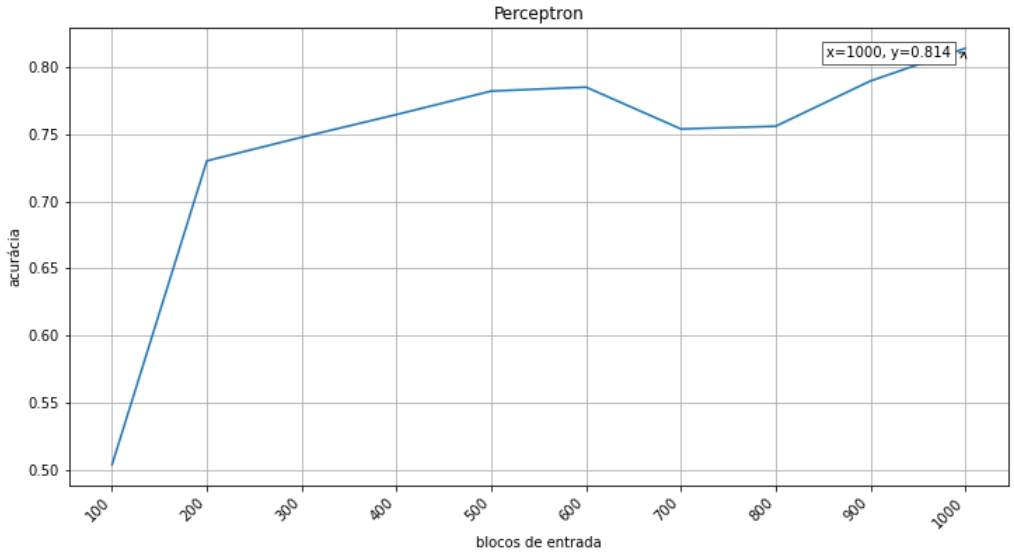
TABELA 1 – MEDIDAS DE ACURÁCIA E TEMPO PARA DIFERENTES CLASSIFICADORES, UTILIZANDO TODA A BASE DE TREINAMENTO

CLASSIFICADOR	ACURÁCIA	TEMPO (s)
kNN	0,94	13
LDA	0,928	0,016
Logistic Regression	0,912	0,015
Naive Bayes	0,89	0,55
Perceptron	0,922	0,014

FONTE: O autor (2021).

Com relação a poucos dados, ou seja, quando os classificadores são alimentados com blocos de dados abaixo de 1.000 exemplos, o classificador que obteve a maior acurácia foi o *Perceptron*, com um valor de 0,814. A Figura 10 apresenta as medidas de acurácia à medida que o classificador é alimentado, com blocos de 100 em 100 até 1.000 exemplos.

FIGURA 10 – MEDIDAS DE ACURÁCIA PARA BLOCOS ABAIXO DE 1.000 EXEMPLOS NA BASE DE TREINAMENTO



FONTE: O autor (2021).

4 CONCLUSÃO

A partir dos resultados do experimento nota-se que existe um significativo impacto na escolha do tamanho da base de dados para treinamento de um classificador. A escolha do tamanho deve levar em consideração as particularidades de cada classificador, bem como o problema que você está tentando resolver. Nesse laboratório, foi preciso levar em consideração o balanceamento dos dados nessa base e a quantidade de classes presentes no problema. Verificou-se que, para uma maximização da acurácia, não foi preciso utilizar toda a base para treinamento do classificador kNN, visto o uso de mais dados, para esse modelo, não implicou em uma melhoria de *performance*.