Introdução ao aprendizado de máquina (MAC5832)
Projeto - Entrega final

Dayanne Gomes e Gustavo Álvarez

28 de julho de 2022

Resumo

Neste trabalho, é estudado o desempenho dos algoritmos de **Regressão Logística** e **SVM** (*Support Vector Machine*) para resolver um problema de classificação. Para tanto, o conjunto de dados Fashion-MNIST será explorado, o qual já fornece uma divisão dos dados em um conjunto de treinamento e um conjunto de teste. No treinamento dos modelos de Regressão Logística e SVM serão usadas rotinas prontas da biblioteca Scikit-learn do Python.

1 Equipe

A equipe é composta pelos alunos de pós-graduação: Dayanne Cristina Pereira Gomes, NUSP: 13796211, e Gustavo David Quintero Alvarez, NUSP: 11350395.

2 Informações sobre a exploração

Para o trabalho, foi feita uma exploração do tipo 1, que é um problema para o qual existe um dataset público, o Fashion MNIST. Em termos gerais, o problema abordado nesse dataset é o de identificação de peças de vestuários em um conjunto de imagens.

2.1 Fashion MNIST

O dataset a ser utilizado é o Fashion MNIST. O Fashion MNIST é um dataset criado por Zalando que se constitui de imagens de peças de vestuário. Esse dataset é dividido em conjunto de treinamento e teste. O conjunto de treinamento tem 60.000 amostras, ao passo que o de teste tem 10.000 amostras. Ele possui o mesmo tamanho de imagem (28x28) e estrutura de treinamento e teste que o dataset MNIST. As imagens estão em escala de cinza e são classificadas em:

1. T-shirt/top
2. Trouser/pants
3. Pullover shirt
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

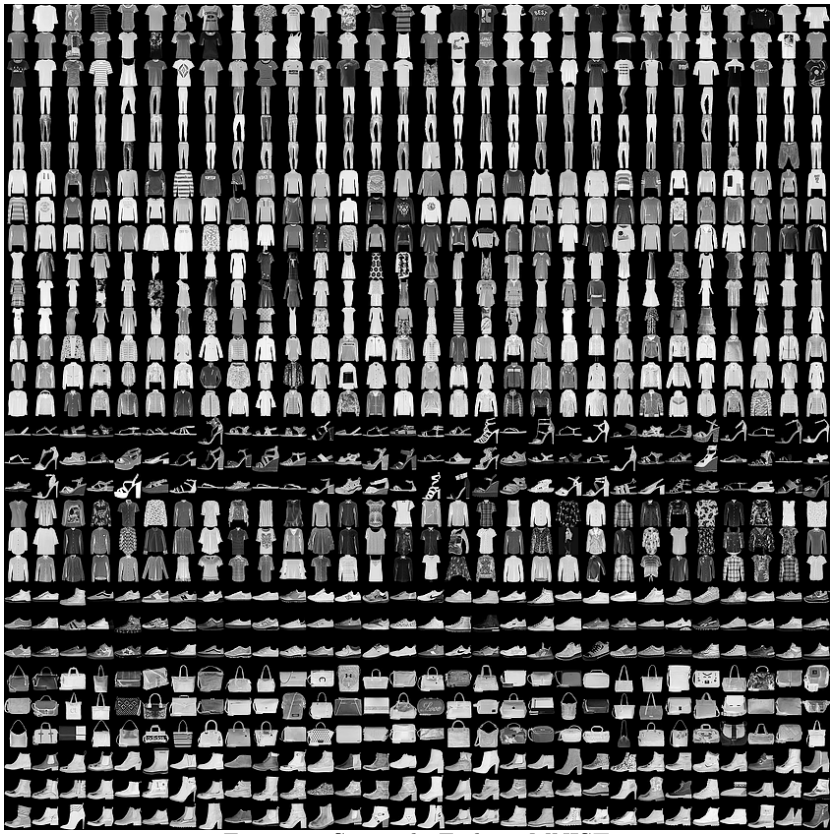

Figura 1: Sprite do Fashion MNIST

A Figura 1 acima, retirada de [7], mostra o formato das imagens no dataset. Individualmente, cada imagem tem a representação de 28×28 pixels na forma:

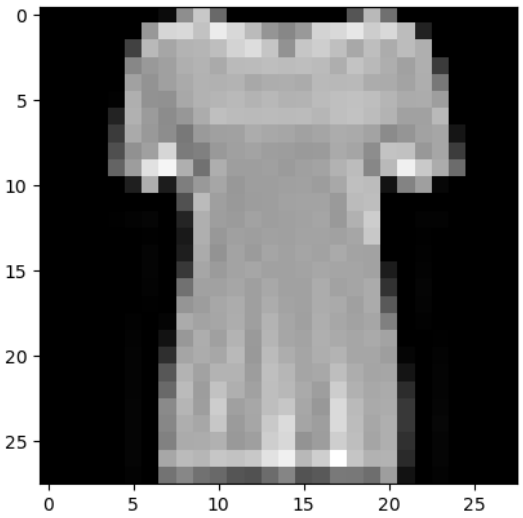

Figura 2: Sprite de camiseta

3 Plano

Para implementar a solução com o algoritmo de Regressão Logística e SVM, usamos a biblioteca Scikit-learn. Para isso, importamos apenas as principais funções a serem utilizadas, visto que a mesma

é muito grande e seria desnecessário trazer tudo para a execução do problema. Além das usuais bibliotecas pandas, numpy e matplotlib, também usamos Tensorflow para lidar com o dataset. Como orientação para preparação dos testes, utilizamos referência de trabalhos anteriores da disciplina, bem como o material do engenheiro de software François Chollet, disponível em [2].

3.1 Preparação do dataset

Para lidar com o conjunto de dados usamos o método *fashion_mnist.load_data()* da biblioteca Tensorflow, o qual já fornece uma divisão dos dados em um conjunto de treinamento com 60.000 amostras que armazenamos no array *xTrain*, um conjunto de teste com 10.000 amostras que armazenamos no array *xTest*, e as correspondentes classes dos conjuntos de treinamento e teste que armazenamos, respectivamente, nos vetores *yTrain* e *yTest*. Após a normalização dos conjuntos de treinamento e teste, usamos o método *train_test_split()* para obter uma divisão estratificada de *xTrain* em um novo conjunto de treinamento *D_train* e validação *D_val*, seguindo a divisão em 80% e 20% (48000 e 12000, respectivamente) conforme foi recomendado pelo professor Abu Mostafa (vide [1], p. 140). Ademais, seguindo a sugestão da professora da disciplina, optamos por fazer uma divisão extra em 75& para treinamento e 25% para validação. Como preparação paralela, também reduzimos o tamanho das imagens utilizadas, que de 28 x 28 pixels, caiu para 14 x 14 pixels, para que pudéssemos estudar a execução de outros testes em menos tempo, além de obter scores parecidos com os originais. Isso pode ser obtido ao fazermos o reshape das imagens, disponíveis nos conjuntos *xTrainReduced* e *xTestReduced*. Por fim, foi observado que as 10 classes estão perfeitamente balanceadas, conforme mostra a figura abaixo, e estas proporções foram mantidas nos dois novos conjuntos criados.

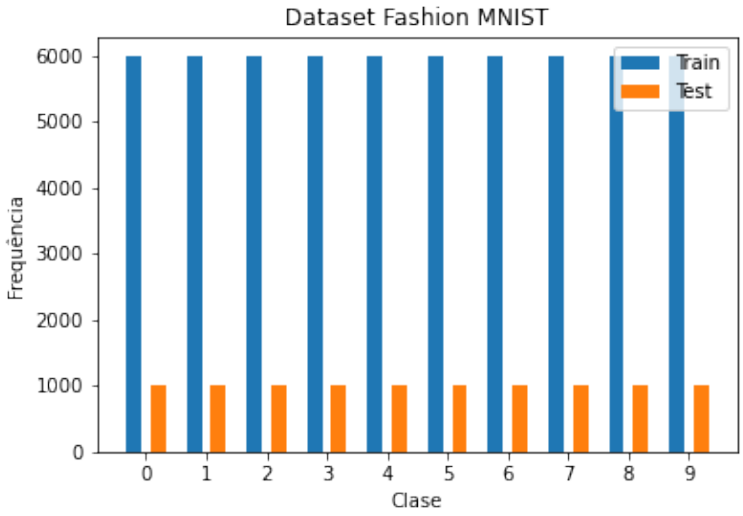

Figura 3: Frequência das 10 classes de Fashion MNIST

3.2 Detalhamento da solução

Tanto no algoritmo de Regressão Logística como no SVM são usados métodos de otimização para encontrar o minimizador de uma certa função de custo. No caso de Regressão Logística, na aula e nos vídeos fornecidos pela Caltech vimos o método de gradiente descendente, o qual pertence a uma família de métodos conhecidos como métodos de descida (veja [3], [4] ou [5]), e que consistem em encontrar uma direção $d \in \mathbb{R}^n$ que satisfaça a condição $\nabla f(x)^T d < 0$ (uma tal d é chamada de *direção de descida*) para uma certa função objetivo $f(x)$, para todo $x \in \mathbb{R}^n$. No caso do método de gradiente descendente a direção de descida é dada por $d = -\nabla f(x)$.

Usamos o método GridSearchCV para treinamento dos dois modelos escolhidos (Regressão Logística e SVM). Com ele, podemos escolher um parâmetro para ranquear os modelos. Assim, optamos pela acurácia (accuracy), que nos dá o total de classificações corretas entre o y_true e o y_pred. Também podemos usar a precisão, que é a habilidade de não classificar como positiva uma amostra negativa. Neste método, um dos parâmetros de entrada é o modelo a ser usado, o qual definimos através da

variável *model_lgreg*. Para o modelo de Regressão Logística, forneceremos também um vetor de strings, o qual armazena os solvers a serem utilizados na fase de otimização; dentre as opções permitidas pelo método *LogisticRegression* do Scikit-learn, faremos uso dos solvers: newton-cg e lbfgs.

- **newton-cg:** Este solver corresponde ao conhecido Método de Newton, o qual consiste em encontrar uma direção de descida $d \in \mathbb{R}^n$ que resolva o sistema linear

$$\nabla^2 f(x)\, d = \nabla f(x),$$

  em que f é a função a ser minimizada (função de custo) para todo $x \in \mathbb{R}^n$. Embora o algoritmo de descida definido com este método tenha convergência quadrática (que é melhor do que a convergência linear do algoritmo de gradiente descendente), o calculo das segundas derivadas pode ser muito custoso.

- **lbfgs:** Este solver usa o método Quase-Newton chamado BFGS (Broyden, Fletcher, Goldfarb e Shanno), e consiste em definir uma direção $d = -H\,\nabla f(x)$, em que H é uma aproximação da matriz Hessiana da função objetivo que contém informação das segundas derivadas dela, mas cujo computo é mais barato (para todos os detalhes veja [3], [5] ou [6]). Pode-se mostrar que a direção d definida por este método é uma direção de descida e que a ordem de convergência do algoritmo é pelo menos superlinear.

Já no caso do modelo SVM usaremos o método SVC() e se fornecerá também uma lista de strings contendo os kernels a serem utilizados. Nesse caso, optamos por utilizar **linear** e **rbf**. Havíamos cogitado inicialmente também o **sigmoid**, mas, após experimentações, recebemos resultados NaN (Not a Number), o que nos fez descartar tais resultados para efeitos de comparação.

4 Testes

Como foi mencionado anteriormente, pretendemos observar o desempenho dos algoritmos de Regressão Logística e SVM em diferentes situações, para tanto, realizamos diversos testes variando o parâmetro de regularização C, a saber $C = 10^k$ para $k = -2, -1, 0, 1, 2$, dado no problema de otimização. Além dos solvers e kernels mencionados acima, também variamos o valor de k-folds da cross-validations em função do tipo de divisão feita (4 no caso 80% e 20%, e 5 no caso 75% e 25%). Os testes foram divididos de acordo com o balanceamento de classes (classes balanceadas e desbalanceadas). Cada teste foi executado individualmente, utilizando os dois algoritmos propostos. Após a execução, comparamos os scores para escolher o modelo final, ranqueado de acordo com o parâmetro definido anteriormente (acurácia), que nos gerou algumas tabelas com os scores médios de cross-validation e matrizes de confusão, bem como os tempos de execução. Uma vez obtido o modelo final, pegamos o conjunto completo de treinamento e teste para fazer o retreinamento final, a fim de analisar se há diferenças significativas entre treinamentos realizados anteriormente.

5 Resultados

5.1 Classes balanceadas

Para os testes utilizando classes balanceadas, mantivemos as proporções iniciais do dataset, ou seja, cada uma das 10 classes possui uma quantidade média de elementos bem próxima das outras (6000 no no conjunto de treinamento, 10000 no conjunto de amostras).

- **Teste 1:** 4 folds, 48000 amostras de treinamento e 12000 amostras de validação (80% e 20%). Os resultados obtidos com cada um dos solvers, kernels e parâmetros de regularização são mostrados nas tabelas abaixo, junto com as respectivas matrizes de confusão.

Tabela 1: Regressão logística

C	Solver	Score médio
100.00	newton-cg	0.8306
100.00	lbfgs	0.8527
10.00	newton-cg	0.8395
10.00	lbfgs	0.8531
1.00	newton-cg	0.8487
1.00	lbfgs	0.8539
0.10	newton-cg	0.8551
0.10	lbfgs	0.8550
0.01	newton-cg	0.8484
0.01	lbfgs	0.8488

Tabela 2: SVM

C	Kernel	Score médio
100.00	linear	0.8288
100.00	rbf	0.8919
10.00	linear	0.8369
10.00	rbf	0.8998
1.00	linear	0.8496
1.00	rbf	0.8839
0.10	linear	0.8622
0.10	rbf	0.8420
0.01	linear	0.8601
0.01	rbf	0.7668

Matriz de confusão Regressão Logística:

$$\begin{pmatrix} 990 & 1 & 19 & 55 & 5 & 1 & 110 & 0 & 17 & 2 \\ 3 & 1159 & 8 & 23 & 1 & 0 & 5 & 1 & 0 & 0 \\ 20 & 2 & 902 & 16 & 139 & 2 & 111 & 1 & 7 & 0 \\ 43 & 10 & 14 & 1064 & 42 & 0 & 23 & 1 & 3 & 0 \\ 5 & 2 & 104 & 46 & 933 & 0 & 106 & 0 & 4 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1117 & 0 & 50 & 10 & 21 \\ 172 & 4 & 136 & 44 & 111 & 1 & 715 & 0 & 17 & 0 \\ 0 & 0 & 0 & 0 & 0 & 40 & 0 & 1129 & 2 & 29 \\ 1 & 0 & 8 & 9 & 2 & 5 & 16 & 6 & 1152 & 1 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 42 & 0 & 1139 \end{pmatrix}$$

Matriz de confusão SVM:

$$\begin{pmatrix} 1050 & 0 & 12 & 25 & 2 & 1 & 104 & 0 & 6 & 0 \\ 1 & 1179 & 1 & 13 & 2 & 0 & 3 & 0 & 1 & 0 \\ 23 & 0 & 998 & 12 & 82 & 2 & 80 & 0 & 3 & 0 \\ 24 & 4 & 10 & 1104 & 34 & 0 & 18 & 0 & 6 & 0 \\ 2 & 3 & 90 & 33 & 1015 & 0 & 56 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1163 & 0 & 20 & 5 & 12 \\ 134 & 0 & 78 & 34 & 58 & 0 & 893 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18 & 0 & 1155 & 2 & 25 \\ 4 & 1 & 2 & 2 & 3 & 3 & 8 & 2 & 1175 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & 0 & 31 & 0 & 1160 \end{pmatrix}$$

Tempo de execução Regressão Logística: 1980 segundos ou 33 minutos
Tempo de execução SVM: 12913 segundos ou 215.23 minutos

Modelo final:

O melhor modelo é: SVM (C = 10.0)
Score: 0.89510000
Acurácia no conjunto de teste: 0.89510000
Acurácia no conjunto de validação: 0.90766667
Precisão no conjunto de teste: 0.89486765
Precisão no conjunto de validação: 0.90743775
Erro no conjunto de teste (Eout): 0.10490000

Matriz de confusão modelo final:

$$\begin{pmatrix} 844 & 1 & 10 & 23 & 4 & 1 & 108 & 0 & 9 & 0 \\ 3 & 972 & 3 & 17 & 2 & 0 & 3 & 0 & 0 & 0 \\ 24 & 2 & 825 & 11 & 77 & 0 & 59 & 0 & 2 & 0 \\ 25 & 6 & 14 & 903 & 28 & 0 & 19 & 0 & 5 & 0 \\ 1 & 1 & 81 & 22 & 834 & 0 & 60 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 966 & 0 & 22 & 2 & 9 \\ 122 & 1 & 84 & 25 & 58 & 0 & 701 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 & 0 & 967 & 0 & 18 \\ 4 & 1 & 2 & 3 & 4 & 2 & 2 & 4 & 978 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 1 & 31 & 0 & 961 \end{pmatrix}$$

Retreinamento:

Score do modelo final: 0.90020000
Acurácia no conjunto teste: 0.90020000
Acurácia no conjunto de validação: 0.97316667
Precisão no conjunto de teste: 0.90014351
Precisão no conjunto de validação: 0.97315360
Erro no conjunto de teste (Eout): 0.09980000

Matriz de confusão modelo retreinado:

$$\begin{pmatrix} 853 & 0 & 13 & 21 & 6 & 1 & 100 & 0 & 6 & 0 \\ 3 & 973 & 2 & 17 & 3 & 0 & 2 & 0 & 0 & 0 \\ 20 & 3 & 836 & 9 & 74 & 0 & 56 & 0 & 2 & 0 \\ 24 & 2 & 13 & 910 & 27 & 0 & 19 & 0 & 5 & 0 \\ 0 & 1 & 82 & 21 & 835 & 0 & 60 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 974 & 0 & 17 & 1 & 7 \\ 114 & 1 & 82 & 24 & 54 & 0 & 718 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 962 & 0 & 22 \\ 3 & 0 & 3 & 3 & 3 & 2 & 3 & 3 & 980 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 1 & 30 & 0 & 961 \end{pmatrix}$$

- **Teste 2:** 5 folds, 48000 amostras de treinamento e 12000 amostras de validação (75% e 25%). Os resultados obtidos com cada um dos solvers, kernels e parâmetros de regularização são mostrados nas tabelas abaixo, junto com as respectivas matrizes de confusão.

Tabela 3: Regressão logística

C	Solver	Score médio
100.00	newton-cg	0.8320
100.00	lbfgs	0.8541
10.00	newton-cg	0.8394
10.00	lbfgs	0.8538
1.00	newton-cg	0.8492
1.00	lbfgs	0.8536
0.10	newton-cg	0.8549
0.10	lbfgs	0.8552
0.01	newton-cg	0.8474
0.01	lbfgs	0.8479

Tabela 4: SVM

C	Kernel	Score médio
100.00	linear	0.8302
100.00	rbf	0.8922
10.00	linear	0.8378
10.00	rbf	0.9000
1.00	linear	0.8518
1.00	rbf	0.8847
0.10	linear	0.8636
0.10	rbf	0.8406
0.01	linear	0.8598
0.01	rbf	0.7665

Matriz de confusão Regressão Logística:

$$\begin{pmatrix} 1256 & 2 & 27 & 62 & 5 & 1 & 131 & 0 & 15 & 1 \\ 2 & 1453 & 10 & 26 & 1 & 0 & 6 & 1 & 1 & 0 \\ 21 & 2 & 1128 & 22 & 172 & 1 & 143 & 0 & 11 & 0 \\ 43 & 15 & 14 & 1329 & 58 & 0 & 33 & 1 & 7 & 0 \\ 6 & 4 & 133 & 59 & 1163 & 0 & 127 & 0 & 8 & 0 \\ 2 & 1 & 0 & 0 & 0 & 1385 & 0 & 68 & 14 & 30 \\ 216 & 6 & 168 & 54 & 139 & 0 & 897 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 56 & 0 & 1396 & 4 & 44 \\ 2 & 1 & 6 & 14 & 5 & 8 & 26 & 6 & 1430 & 2 \\ 0 & 0 & 0 & 1 & 0 & 19 & 0 & 57 & 0 & 1423 \end{pmatrix}$$

Matriz de confusão SVM:

$$\begin{pmatrix} 1327 & 0 & 12 & 32 & 2 & 1 & 116 & 0 & 10 & 0 \\ 3 & 1472 & 1 & 19 & 2 & 0 & 2 & 0 & 1 & 0 \\ 25 & 0 & 1257 & 15 & 100 & 2 & 95 & 0 & 6 & 0 \\ 33 & 8 & 8 & 1377 & 41 & 0 & 23 & 0 & 10 & 0 \\ 4 & 3 & 111 & 44 & 1263 & 0 & 74 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1449 & 0 & 32 & 4 & 15 \\ 175 & 1 & 107 & 43 & 68 & 0 & 1101 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 1449 & 2 & 29 \\ 2 & 1 & 2 & 4 & 4 & 4 & 10 & 2 & 1470 & 1 \\ 0 & 0 & 1 & 0 & 0 & 10 & 0 & 45 & 0 & 1444 \end{pmatrix}$$

Tempo de execução Regressão Logística: 2807 segundos ou 46.8 minutos
Tempo de execução SVM: 15481 segundos ou 258.03 minutos

Modelo final:

O melhor modelo é: SVM (C = 10.0)
Score: 0.89460000
Acurácia no conjunto de teste: 0.89460000
Acurácia no conjunto de validação: 0.90726667
Precisão no conjunto de teste: 0.89432975
Precisão no conjunto de validação: 0.90687373
Erro no conjunto de teste (Eout): 0.10540000

Matriz de confusão modelo final:

$$\begin{pmatrix} 849 & 0 & 11 & 22 & 4 & 1 & 102 & 0 & 11 & 0 \\ 4 & 970 & 2 & 19 & 2 & 0 & 3 & 0 & 0 & 0 \\ 21 & 2 & 820 & 11 & 79 & 0 & 64 & 0 & 3 & 0 \\ 24 & 4 & 13 & 908 & 27 & 0 & 19 & 0 & 5 & 0 \\ 1 & 1 & 80 & 26 & 827 & 0 & 63 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 966 & 0 & 22 & 2 & 9 \\ 122 & 1 & 84 & 26 & 55 & 0 & 702 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 & 0 & 967 & 0 & 18 \\ 3 & 1 & 3 & 3 & 3 & 2 & 3 & 4 & 978 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 1 & 32 & 0 & 959 \end{pmatrix}$$

Retreinamento:

Score do modelo final: 0.90020000
Acurácia no conjunto de teste: 0.90020000
Acurácia no conjunto de validação: 0.97240000
Precisão no conjunto de teste: 0.90014351
Precisão no conjunto de validação: 0.97237738
Erro no conjunto de teste (Eout): 0.09980000

Matriz de confusão modelo retreinado:

$$\begin{pmatrix} 853 & 0 & 13 & 21 & 6 & 1 & 100 & 0 & 6 & 0 \\ 3 & 973 & 2 & 17 & 3 & 0 & 2 & 0 & 0 & 0 \\ 20 & 3 & 836 & 9 & 74 & 0 & 56 & 0 & 2 & 0 \\ 24 & 2 & 13 & 910 & 27 & 0 & 19 & 0 & 5 & 0 \\ 0 & 1 & 82 & 21 & 835 & 0 & 60 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 974 & 0 & 17 & 1 & 7 \\ 114 & 1 & 82 & 24 & 54 & 0 & 718 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 962 & 0 & 22 \\ 3 & 0 & 3 & 3 & 3 & 2 & 3 & 3 & 980 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 1 & 30 & 0 & 961 \end{pmatrix}$$

- **Teste 3:** Imagens reduzidas pela metade com 4 folds, 48000 amostras de treinamento e 12000 amostras de validação (80% e 20%). Os resultados obtidos com cada um dos solvers, kernels e parâmetros de regularização são mostrados nas tabelas abaixo, junto com as respectivas matrizes de confusão.

Tabela 5: Regressão logística

C	Solver	Score médio
100.00	newton-cg	0.8257
100.00	lbfgs	0.8254
10.00	newton-cg	0.8268
10.00	lbfgs	0.8250
1.00	newton-cg	0.8277
1.00	lbfgs	0.8261
0.10	newton-cg	0.8250
0.10	lbfgs	0.8241
0.01	newton-cg	0.8053
0.01	lbfgs	0.8052

Tabela 6: SVM

C	Kernel	Score médio
100.00	linear	0.8308
100.00	rbf	0.8747
10.00	linear	0.8329
10.00	rbf	0.8808
1.00	linear	0.8365
1.00	rbf	0.8650
0.10	linear	0.8351
0.10	rbf	0.8234
0.01	linear	0.8185
0.01	rbf	0.7579

Matriz de confusão Regressão Logística:

$$\begin{pmatrix} 977 & 12 & 18 & 55 & 12 & 4 & 103 & 1 & 18 & 0 \\ 7 & 1124 & 7 & 54 & 1 & 0 & 5 & 1 & 1 & 0 \\ 18 & 6 & 832 & 11 & 170 & 4 & 149 & 1 & 7 & 2 \\ 45 & 40 & 13 & 1016 & 49 & 0 & 33 & 1 & 3 & 0 \\ 4 & 7 & 121 & 42 & 887 & 0 & 130 & 0 & 8 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1094 & 0 & 74 & 8 & 24 \\ 187 & 7 & 140 & 52 & 131 & 0 & 665 & 0 & 17 & 1 \\ 0 & 0 & 0 & 0 & 0 & 49 & 0 & 1103 & 5 & 43 \\ 2 & 1 & 10 & 11 & 4 & 6 & 19 & 3 & 1144 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 52 & 0 & 1129 \end{pmatrix}$$

Matriz de confusão SVM:

$$\begin{pmatrix} 1046 & 2 & 14 & 31 & 2 & 1 & 96 & 0 & 8 & 0 \\ 2 & 1161 & 3 & 30 & 2 & 0 & 2 & 0 & 0 & 0 \\ 18 & 1 & 974 & 10 & 103 & 2 & 89 & 0 & 3 & 0 \\ 33 & 10 & 11 & 1090 & 35 & 0 & 18 & 0 & 3 & 0 \\ 4 & 2 & 113 & 41 & 967 & 0 & 71 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1153 & 0 & 31 & 3 & 13 \\ 172 & 4 & 96 & 36 & 77 & 0 & 809 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 1150 & 2 & 29 \\ 4 & 1 & 5 & 4 & 4 & 5 & 8 & 3 & 1166 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 40 & 0 & 1152 \end{pmatrix}$$

Tempo de execução Regressão Logística: 519 segundos ou 8.67 minutos
Tempo de execução SVM: 3732 segundos ou 62.21 minutos

Modelo final:

O melhor modelo é: SVM (C = 10.0)
Score: 0.87520000
Acurácia no conjunto de teste: 0.87520000
Acurácia no conjunto de validação: 0.88900000
Precisão no conjunto de teste: 0.87494731
Precisão no conjunto de validação: 0.88845382
Erro no conjunto de teste (Eout): 0.12480000

Matriz de confusão modelo final:

$$\begin{pmatrix} 837 & 3 & 7 & 26 & 3 & 1 & 114 & 0 & 9 & 0 \\ 6 & 959 & 4 & 22 & 3 & 0 & 5 & 0 & 1 & 0 \\ 22 & 2 & 786 & 13 & 93 & 0 & 83 & 0 & 1 & 0 \\ 23 & 9 & 7 & 900 & 28 & 0 & 30 & 0 & 3 & 0 \\ 1 & 1 & 87 & 34 & 794 & 0 & 82 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 963 & 0 & 21 & 1 & 14 \\ 135 & 4 & 98 & 32 & 79 & 0 & 643 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 957 & 0 & 23 \\ 4 & 0 & 7 & 3 & 7 & 3 & 4 & 3 & 968 & 1 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 44 & 0 & 945 \end{pmatrix}$$

Retreinamento:

Score do modelo final: 0.87830000
Acurácia no conjunto de teste: 0.87830000
Acurácia no conjunto de validação: 0.95025000
Precisão no conjunto de teste: 0.87801563
Precisão no conjunto de validação: 0.95019828
Erro no conjunto de teste (Eout): 0.12170000

Matriz de confusão modelo retreinado:

$$\begin{pmatrix} 838 & 1 & 11 & 22 & 4 & 1 & 114 & 0 & 9 & 0 \\ 3 & 960 & 4 & 24 & 3 & 0 & 5 & 0 & 1 & 0 \\ 16 & 2 & 790 & 14 & 89 & 0 & 85 & 0 & 4 & 0 \\ 21 & 9 & 14 & 904 & 26 & 0 & 23 & 0 & 3 & 0 \\ 1 & 1 & 85 & 33 & 802 & 0 & 77 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 965 & 0 & 19 & 2 & 13 \\ 138 & 3 & 99 & 31 & 74 & 0 & 647 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 958 & 0 & 23 \\ 4 & 0 & 8 & 3 & 5 & 2 & 6 & 3 & 969 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 39 & 0 & 950 \end{pmatrix}$$

- **Teste 4:** Imagens reduzidas pela metade com 5 folds, 48000 amostras de treinamento e 12000 amostras de validação (75% e 25%). Os resultados obtidos com cada um dos solvers, kernels e parâmetros de regularização são mostrados nas tabelas abaixo, junto com as respectivas matrizes de confusão.

Tabela 7: Regressão logística

C	Solver	Score médio
100.00	newton-cg	0.8254
100.00	lbfgs	0.8233
10.00	newton-cg	0.8262
10.00	lbfgs	0.8246
1.00	newton-cg	0.8270
1.00	lbfgs	0.8247
0.10	newton-cg	0.8249
0.10	lbfgs	0.8247
0.01	newton-cg	0.8054
0.01	lbfgs	0.8055

Tabela 8: SVM

C	Kernel	Score médio
100.00	linear	0.8311
100.00	rbf	0.8748
10.00	linear	0.8332
10.00	rbf	0.8808
1.00	linear	0.8350
1.00	rbf	0.8649
0.10	linear	0.8346
0.10	rbf	0.8223
0.01	linear	0.8181
0.01	rbf	0.7571

Matriz de confusão Regressão Logística:

$$\begin{pmatrix} 1232 & 11 & 22 & 76 & 13 & 6 & 113 & 0 & 27 & 0 \\ 7 & 1406 & 9 & 68 & 1 & 0 & 7 & 1 & 1 & 0 \\ 22 & 5 & 1042 & 15 & 214 & 6 & 182 & 1 & 12 & 1 \\ 50 & 51 & 16 & 1273 & 62 & 1 & 38 & 1 & 8 & 0 \\ 4 & 11 & 146 & 54 & 1111 & 0 & 164 & 0 & 9 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1365 & 0 & 88 & 14 & 33 \\ 233 & 10 & 181 & 66 & 161 & 0 & 827 & 0 & 22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 63 & 0 & 1379 & 5 & 53 \\ 3 & 2 & 9 & 15 & 7 & 8 & 26 & 4 & 1424 & 2 \\ 0 & 0 & 0 & 1 & 0 & 22 & 0 & 67 & 0 & 1410 \end{pmatrix}$$

Matriz de confusão SVM:

$$\begin{pmatrix} 1310 & 2 & 18 & 30 & 3 & 1 & 125 & 0 & 11 & 0 \\ 3 & 1457 & 3 & 31 & 3 & 0 & 3 & 0 & 0 & 0 \\ 26 & 1 & 1217 & 17 & 117 & 2 & 115 & 0 & 5 & 0 \\ 41 & 12 & 13 & 1359 & 47 & 0 & 24 & 0 & 4 & 0 \\ 4 & 3 & 136 & 48 & 1213 & 0 & 93 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1442 & 0 & 41 & 3 & 14 \\ 216 & 6 & 120 & 43 & 93 & 0 & 1016 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 27 & 0 & 1438 & 2 & 33 \\ 5 & 1 & 4 & 4 & 3 & 7 & 11 & 3 & 1461 & 1 \\ 1 & 0 & 0 & 0 & 0 & 11 & 0 & 53 & 0 & 1435 \end{pmatrix}$$

Tempo de execução Regressão Logística: 660 segundos ou 11.01 minutos
Tempo de execução SVM: 4355 segundos ou 72.59 minutos

Modelo final:

O melhor modelo é: SVM (C = 10.0)
Score: 0.87470000
Acurácia no conjunto de teste: 0.87470000
Acurácia no conjunto de validação: 0.88986667
Precisão no conjunto de teste: 0.87427179
Precisão no conjunto de validação: 0.88934734
Erro no conjunto de teste (Eout): 0.12530000

Matriz de confusão modelo final:

$$\begin{pmatrix} 843 & 3 & 10 & 26 & 3 & 1 & 105 & 0 & 9 & 0 \\ 4 & 957 & 4 & 25 & 3 & 0 & 6 & 0 & 1 & 0 \\ 22 & 2 & 780 & 13 & 94 & 0 & 87 & 0 & 2 & 0 \\ 22 & 11 & 10 & 898 & 27 & 0 & 28 & 0 & 4 & 0 \\ 1 & 1 & 85 & 35 & 796 & 0 & 81 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 963 & 0 & 20 & 2 & 14 \\ 138 & 4 & 104 & 31 & 75 & 0 & 639 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 957 & 0 & 23 \\ 4 & 0 & 8 & 3 & 5 & 2 & 5 & 3 & 969 & 1 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 44 & 0 & 945 \end{pmatrix}$$

Retreinamento:

Score do modelo final: 0.87830000
Acurácia no conjunto de teste: 0.87830000
Acurácia no conjunto de validação: 0.94973333
Precisão no conjunto de teste: 0.87801563
Precisão no conjunto de validação: 0.94967041
Erro no conjunto de teste (Eout): 0.12170000

Matriz de confusão modelo retreinado:

$$\begin{pmatrix} 838 & 1 & 11 & 22 & 4 & 1 & 114 & 0 & 9 & 0 \\ 3 & 960 & 4 & 24 & 3 & 0 & 5 & 0 & 1 & 0 \\ 16 & 2 & 790 & 14 & 89 & 0 & 85 & 0 & 4 & 0 \\ 21 & 9 & 14 & 904 & 26 & 0 & 23 & 0 & 3 & 0 \\ 1 & 1 & 85 & 33 & 802 & 0 & 77 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 965 & 0 & 19 & 2 & 13 \\ 138 & 3 & 99 & 31 & 74 & 0 & 647 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 958 & 0 & 23 \\ 4 & 0 & 8 & 3 & 5 & 2 & 6 & 3 & 969 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 39 & 0 & 950 \end{pmatrix}$$

5.2 Classes desbalanceadas

Para os testes utilizando classes balanceadas, redistribuímos as proporções iniciais do dataset, ou seja, algumas classes possuem mais amostras que outras. Para fazer isso, as classes 1, 4, 7 e 9 mantêm-se, manipulando apenas as outras. Retiramos 1000 elementos das classes 2, 5 e 8 e colocamos 1000 elementos nas classes 0, 3, e 6, mantendo o total de elementos de todos os conjuntos somados igual. Finalmente, mantivemos balanceado o conjunto de teste, conforme figura abaixo.

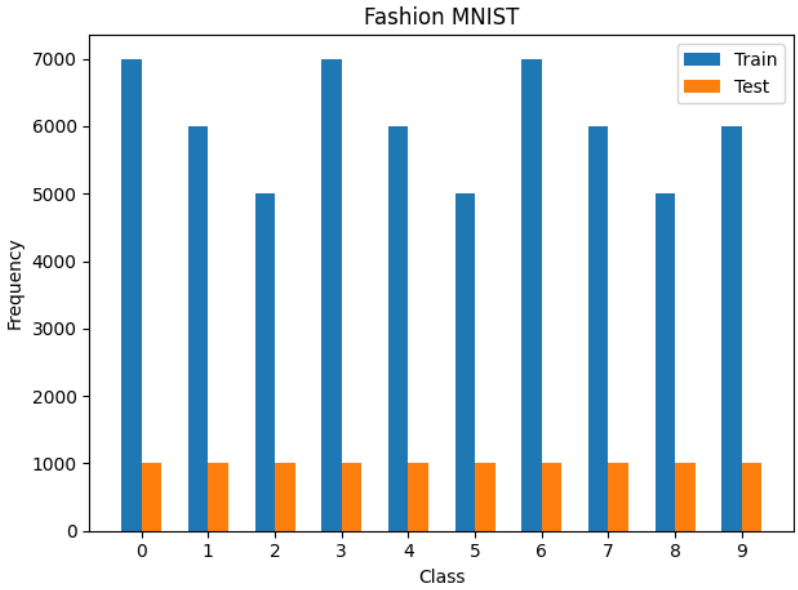

Figura 4: Conjunto de Treinamento desbalanceado.

- **Teste 1**: Imagens reduzidas com 4 folds, 48000 amostras de treinamento e 12000 amostras de validação (80% e 20%). Os resultados obtidos com cada um dos solvers, kernels e parâmetros de regularização são mostrados nas tabelas abaixo, junto com as respectivas matrizes de confusão.

Tabela 9: Regressão logística

C	Solver	Score médio
100.00	newton-cg	0.8257
100.00	lbfgs	0.8254
10.00	newton-cg	0.8268
10.00	lbfgs	0.8250
1.00	newton-cg	0.8277
1.00	lbfgs	0.8261
0.10	newton-cg	0.8250
0.10	lbfgs	0.8241
0.01	newton-cg	0.8053
0.01	lbfgs	0.8052

Tabela 10: SVM

C	Kernel	Score médio
100.00	linear	0.8308
100.00	rbf	0.8747
10.00	linear	0.8329
10.00	rbf	0.8808
1.00	linear	0.8365
1.00	rbf	0.8650
0.10	linear	0.8351
0.10	rbf	0.8234
0.01	linear	0.8185
0.01	rbf	0.7579

Matriz de confusão Regressão Logística:

$$\begin{pmatrix} 977 & 12 & 18 & 55 & 12 & 4 & 103 & 1 & 18 & 0 \\ 7 & 1124 & 7 & 54 & 1 & 0 & 5 & 1 & 1 & 0 \\ 18 & 6 & 832 & 11 & 170 & 4 & 149 & 1 & 7 & 2 \\ 45 & 40 & 13 & 1016 & 49 & 0 & 33 & 1 & 3 & 0 \\ 4 & 7 & 121 & 42 & 887 & 0 & 130 & 0 & 8 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1094 & 0 & 74 & 8 & 24 \\ 187 & 7 & 140 & 52 & 131 & 0 & 665 & 0 & 17 & 1 \\ 0 & 0 & 0 & 0 & 0 & 49 & 0 & 1103 & 5 & 43 \\ 2 & 1 & 10 & 11 & 4 & 6 & 19 & 3 & 1144 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 52 & 0 & 1129 \end{pmatrix}$$

Matriz de confusão SVM:

$$\begin{pmatrix} 1046 & 2 & 14 & 31 & 2 & 1 & 96 & 0 & 8 & 0 \\ 2 & 1161 & 3 & 30 & 2 & 0 & 2 & 0 & 0 & 0 \\ 18 & 1 & 974 & 10 & 103 & 2 & 89 & 0 & 3 & 0 \\ 33 & 10 & 11 & 1090 & 35 & 0 & 18 & 0 & 3 & 0 \\ 4 & 2 & 113 & 41 & 967 & 0 & 71 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1153 & 0 & 31 & 3 & 13 \\ 172 & 4 & 96 & 36 & 77 & 0 & 809 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 1150 & 2 & 29 \\ 4 & 1 & 5 & 4 & 4 & 5 & 8 & 3 & 1166 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 40 & 0 & 1152 \end{pmatrix}$$

Tempo de execução Regressão Logística: 515.18 segundos ou 8.58 minutos
Tempo de execução SVM: 3701.47 segundos ou 61.69 minutos

Modelo final:

O melhor modelo é: SVM (C = 10.0)
Score: 0.87520000
Acurácia no conjunto de teste: 0.87520000
Acurácia no conjunto de validação: 0.88900000
Precisão no conjunto de teste: 0.87494731
Precisão no conjunto de validação: 0.88845382
Erro no conjunto de teste (Eout): 0.12480000

Matriz de confusão modelo final:

$$\begin{pmatrix} 837 & 3 & 7 & 26 & 3 & 1 & 114 & 0 & 9 & 0 \\ 6 & 959 & 4 & 22 & 3 & 0 & 5 & 0 & 1 & 0 \\ 22 & 2 & 786 & 13 & 93 & 0 & 83 & 0 & 1 & 0 \\ 23 & 9 & 7 & 900 & 28 & 0 & 30 & 0 & 3 & 0 \\ 1 & 1 & 87 & 34 & 794 & 0 & 82 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 963 & 0 & 21 & 1 & 14 \\ 135 & 4 & 98 & 32 & 79 & 0 & 643 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 957 & 0 & 23 \\ 4 & 0 & 7 & 3 & 7 & 3 & 4 & 3 & 968 & 1 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 44 & 0 & 945 \end{pmatrix}$$

Retreinamento:

Score do modelo final: 0.87830000
Acurácia no conjunto de teste: 0.87830000
Acurácia no conjunto de validação: 0.95025000
Precisão no conjunto de teste: 0.87801563
Precisão no conjunto de validação: 0.95019828
Erro no conjunto de teste (Eout): 0.12170000

Matriz de confusão modelo retreinado:

$$\begin{pmatrix} 838 & 1 & 11 & 22 & 4 & 1 & 114 & 0 & 9 & 0 \\ 3 & 960 & 4 & 24 & 3 & 0 & 5 & 0 & 1 & 0 \\ 16 & 2 & 790 & 14 & 89 & 0 & 85 & 0 & 4 & 0 \\ 21 & 9 & 14 & 904 & 26 & 0 & 23 & 0 & 3 & 0 \\ 1 & 1 & 85 & 33 & 802 & 0 & 77 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 965 & 0 & 19 & 2 & 13 \\ 138 & 3 & 99 & 31 & 74 & 0 & 647 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 958 & 0 & 23 \\ 4 & 0 & 8 & 3 & 5 & 2 & 6 & 3 & 969 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 39 & 0 & 950 \end{pmatrix}$$

- **Teste 2:** Imagens reduzidas com 5 folds, 45000 amostras de treinamento e 15000 amostras de validação (75% e 25%). Os resultados obtidos com cada um dos solvers, kernels e parâmetros de regularização são mostrados nas tabelas abaixo, junto com as respectivas matrizes de confusão.

Tabela 11: Regressão logística

C	Solver	Score médio
100.00	newton-cg	0.8254
100.00	lbfgs	0.8233
10.00	newton-cg	0.8262
10.00	lbfgs	0.8246
1.00	newton-cg	0.8270
1.00	lbfgs	0.8247
0.10	newton-cg	0.8249
0.10	lbfgs	0.8247
0.01	newton-cg	0.8054
0.01	lbfgs	0.8055

Tabela 12: SVM

C	Kernel	Score médio
100.00	linear	0.8311
100.00	rbf	0.8748
10.00	linear	0.8332
10.00	rbf	0.8808
1.00	linear	0.8350
1.00	rbf	0.8649
0.10	linear	0.8346
0.10	rbf	0.8223
0.01	linear	0.8181
0.01	rbf	0.7571

Matriz de confusão Regressão Logística:

$$\begin{pmatrix} 1232 & 11 & 22 & 76 & 13 & 6 & 113 & 0 & 27 & 0 \\ 7 & 1406 & 9 & 68 & 1 & 0 & 7 & 1 & 1 & 0 \\ 22 & 5 & 1042 & 15 & 214 & 6 & 182 & 1 & 12 & 1 \\ 50 & 51 & 16 & 1273 & 62 & 1 & 38 & 1 & 8 & 0 \\ 4 & 11 & 146 & 54 & 1111 & 0 & 164 & 0 & 9 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1365 & 0 & 88 & 14 & 33 \\ 233 & 10 & 181 & 66 & 161 & 0 & 827 & 0 & 22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 63 & 0 & 1379 & 5 & 53 \\ 3 & 2 & 9 & 15 & 7 & 8 & 26 & 4 & 1424 & 2 \\ 0 & 0 & 0 & 1 & 0 & 22 & 0 & 67 & 0 & 1410 \end{pmatrix}$$

Matriz de confusão SVM:

$$\begin{pmatrix} 1310 & 2 & 18 & 30 & 3 & 1 & 125 & 0 & 11 & 0 \\ 3 & 1457 & 3 & 31 & 3 & 0 & 3 & 0 & 0 & 0 \\ 26 & 1 & 1217 & 17 & 117 & 2 & 115 & 0 & 5 & 0 \\ 41 & 12 & 13 & 1359 & 47 & 0 & 24 & 0 & 4 & 0 \\ 4 & 3 & 136 & 48 & 1213 & 0 & 93 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1442 & 0 & 41 & 3 & 14 \\ 216 & 6 & 120 & 43 & 93 & 0 & 1016 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 27 & 0 & 1438 & 2 & 33 \\ 5 & 1 & 4 & 4 & 3 & 7 & 11 & 3 & 1461 & 1 \\ 1 & 0 & 0 & 0 & 0 & 11 & 0 & 53 & 0 & 1435 \end{pmatrix}$$

Tempo de execução Regressão Logística: 660.02 segundos ou 11 minutos
Tempo de execução SVM: 4328.86 segundos ou 72.14 minutos

Modelo final:

O melhor modelo é: SVM (C = 10.0)
Score: 0.87470000
Acurácia no conjunto de teste: 0.87470000
Acurácia no conjunto de validação: 0.88986667
Precisão no conjunto de teste: 0.87427179
Precisão no conjunto de validação: 0.88934734
Erro no conjunto de teste (Eout): 0.12530000

Matriz de confusão modelo final:

$$\begin{pmatrix} 843 & 3 & 10 & 26 & 3 & 1 & 105 & 0 & 9 & 0 \\ 4 & 957 & 4 & 25 & 3 & 0 & 6 & 0 & 1 & 0 \\ 22 & 2 & 780 & 13 & 94 & 0 & 87 & 0 & 2 & 0 \\ 22 & 11 & 10 & 898 & 27 & 0 & 28 & 0 & 4 & 0 \\ 1 & 1 & 85 & 35 & 796 & 0 & 81 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 963 & 0 & 20 & 2 & 14 \\ 138 & 4 & 104 & 31 & 75 & 0 & 639 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 957 & 0 & 23 \\ 4 & 0 & 8 & 3 & 5 & 2 & 5 & 3 & 969 & 1 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 44 & 0 & 945 \end{pmatrix}$$

Retreinamento:

Score do modelo final: 0.87830000
Acurácia no conjunto de teste: 0.87830000
Acurácia no conjunto de validação: 0.94973333
Precisão no conjunto de teste: 0.87801563
Precisão no conjunto de validação: 0.94967041
Erro no conjunto de teste (Eout): 0.12170000

Matriz de confusão modelo retreinado:

$$\begin{pmatrix} 838 & 1 & 11 & 22 & 4 & 1 & 114 & 0 & 9 & 0 \\ 3 & 960 & 4 & 24 & 3 & 0 & 5 & 0 & 1 & 0 \\ 16 & 2 & 790 & 14 & 89 & 0 & 85 & 0 & 4 & 0 \\ 21 & 9 & 14 & 904 & 26 & 0 & 23 & 0 & 3 & 0 \\ 1 & 1 & 85 & 33 & 802 & 0 & 77 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 965 & 0 & 19 & 2 & 13 \\ 138 & 3 & 99 & 31 & 74 & 0 & 647 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 958 & 0 & 23 \\ 4 & 0 & 8 & 3 & 5 & 2 & 6 & 3 & 969 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 1 & 39 & 0 & 950 \end{pmatrix}$$

6 Conclusões

Após a realização dos testes, a seguir apresentamos algumas conclusões que estão divididas de acordo com as classes observadas. Como observado no geral, tivemos como resultado que as classes balanceadas tiveram um desempenho melhor, apesar de que também mostra-se interessante um estudo breve do comportamento quando há um desbalanceamento das mesmas.

6.1 Classes balanceadas

- **Tempo de computação:** O custo em termos de tempo de computação é, em alguns casos, 8 vezes maior que em regressão logística, o que pode ser relevante em sistemas com baixo poder computacional. Uma forma de contornar essa dificuldade foi de diminuir o tamanho das imagens, conforme feito nos testes 3 e 4 e estabelecido no plano. Observamos que o tempo de computação diminuiu de 215 para 62 minutos, sem que houvesse uma perda significativa de score.

- **Score:** Observamos que o resultado geral em função do score dos modelos favorece o SVM, fato que pode ser constatado olhando as matrizes de confusão (o número de acertos sempre foi superior usando o SVM). Em relação aos testes 3 e 4, vemos também que o score não foi muito afetado, de 0.90 para 0.89.

- **Cross-validation:** Nos testes, como considerado anteriormente, fizemos uma divisão nos dois primeiros testes de 80% e 20%, e nos dois subsequentes 75% e 25%. Os resultados obtidos não mostram uma grande diferença nos scores, porém o melhor resultado foi usando 80% e 20%. É interessante notar que, após o retreinamento, o score usando as duas divisões foi exatamente o mesmo.

6.2 Classes desbalanceadas

Conforme observado no item 6.1, ao diminuir o tamanho das imagens, o score ficou parecido (com uma melhora no tempo de computação), o que nos levou a executar apenas dois testes, que consideram as divisões de 80% e 20%, e 75% e 25% para treinamento e validação.

Tempo de computação: De forma semelhante, o tempo observado utilizando o SVM foi muito maior.

Score: O score permanece maior em SVM que em regressão logística.

Cross-validation: Similarmente, usando a proporção de 80% e 20%, temos um resultado melhor que 75% e 25%.

6.3 Comentários finais

Em geral, o desempenho dos modelos foi muito bom. Em comparação com alguns scores presentes na literatura (seção de Benchmark por [7]), o score mostra-se consistente e dentro de um valor aceitável. Porém, o leitor pode escolher qual modelo se ajusta melhor à sua necessidade (tempo de computação ou score). Se o tempo de computação não for um problema, sugerimos a escolha do modelo de SVM, com o tamanho original das imagens (28x28), o qual mostrou o melhor desempenho. Ainda assim,

o modelo de regressão logística também mostra-se como uma boa alternativa, já que os scores não sofreram diminuição significativa, bem como tem um tempo de execução menor. Por fim, observamos também que, ao desbalancear as classes, o score permanece alto nos dois modelos. No entanto, um experimento mais próximo da realidade utilizaria um desbalanceamento mais aleatório, através de alguma função própria das bibliotecas ou alguma rotina similar programada pelo usuário que siga alguma proposta.

Referências

[1] Yaser S Abu-Mostafa. *Learning from data: a short course*. 2012.

[2] François Chollet. URL: https://github.com/fchollet/deep-learning-with-python-notebooks.git.

[3] Ana Friedlander. ≪Elementos de programaçao nao-linear≫. Em: *Livro-texto. UNICAMP* (2012).

[4] David G Luenberger, Yinyu Ye et al. *Linear and nonlinear programming*. Vol. 2. Springer, 1984.

[5] Jorge Nocedal e Stephen J Wright. *Numerical optimization*. Springer, 1999.

[6] Wikipedia. URL: https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm.

[7] Zalando. *Fashion-MNIST: A mnist-like fashion product database*. URL: https://github.com/zalandoresearch/fashion-mnist.