Relatório Trabalho Prático

Arthur Haickel Nina (18203783) e
Universidade Federal de Santa Catarina (UFSC)

I. Pré-processamento

Primeiramente, era necessário regularizar o dataset [1], para isso, adicionou-se a linha de cabeçalho dele, que separava todas as *features e labels*. Em seguida, percebeu-se que havia linhas praticamente iguais 1, exceto por um atributo, mas que nada mudava na classificação daquele objeto e também não alterava valores de outras *features* mais proeminentes, portanto linhas foram apagadas utilizando como critério os valores de $capital\_run\_lenght$.

char_freq_4	char_freq_5	char_freq_6	capital_run_length_average	capital_run_length_longest	capital_run_length_total	Class
0.276	0.184	0.01	9.821	485	2259	1
0.137	0.000	0.00	3.537	40	191	1
0.135	0.000	0.00	3.537	40	191	1

Figura 1: Dataset apontando linhas praticamente duplicadas.

Dataset foi separado entre *features e labels* e em seguida, valores faltantes de atributos foram todos imputados utilizando o algoritmo KNN. Após a imputação, era necessário verificar se a base de dados estava com proporções equilibradas das classes para o treinamento. Fazendo a contagem, verificou-se que estava levemente desbalanceado:

Tabela I: Porcentagem de classes presentes no dataset

Classe	Percentual
Legítimo	58.895706%
Spam	41.104294%

Para balancear o dataset, ele deveria ser normalizado ou padronizado, escolheu-se a padronização (média é igualada a zero e desvio padrão limitado a um), pois apresentou resultados melhores em acurácia.

Tabela II: Resultados provisórios em técnicas de pré-processamento

Técnica	Acurácia
Normalização	0.69186
Padronização	0.88120

II. Validação Cruzada

Utilizou-se KFolds[2] como técnica de validação cruzada, onde se divide o conjunto de dados em pastas (folds) para teste e validação. Optou-se por 5 folds e 8 testes para equilibrar tempo de execução e boa avaliação dos algoritmos. Analisaram-se técnicas de validação cruzada (CV) aninhada e não aninhada. A CV aninhada foi usada para desenvolver um modelo com otimização de hiperparâmetros. Ela estima o erro de generalização do modelo e seu *grid search*, pois a não aninhada tende a superestimar a performance dos modelos, o que acarretar *overfitting*.[3] Essas avaliações otimistas podem ser observadas nas curvas vermelhas dos gráficos. No caso da CV não aninhada, as divisões nos conjuntos de treinamento, teste e validação ocorrem nos loops internos. No loop externo, o erro de generalização é estimado pela média das pontuações do conjunto de teste em várias divisões do conjunto de dados, garantindo separação completa entre conjuntos de treinamento e teste.[4] Portanto ela é usada apenas para avaliar o desempenho final do modelo, após todos os hiperparâmetros terem sido ajustados. Para a implementação da CV, utilizou-se um exemplo da biblioteca *Scikit Learn*, com o F1-score como métrica para observar o viés não aninhado. Além disso, parâmetros ótimos de cada teste foram adicionados à visualização dos gráficos. [5]

III. Métricas de avaliação

- Acurácia - Taxa de acerto do modelo, medida pela proporção de predições corretas.
- Score F1 - Média harmônica entre precisão e *recall*, significativo em casos de classes desbalanceadas, indicativo de como o modelo se sai ao detectar casos positivos (spam).
- Área sob Curva ROC - Descreve de forma generalizada o desempenho do modelo em separar classes ao traçar a taxa de verdadeiros positivos em função da taxa de falsos positivos em diferentes pontos de corte do modelo. Probabilidade do modelo atribuir um ranqueamento mais alto para um objeto positivo do que para um objeto negativo.[2]

IV. Performance dos modelos

A. Support Vector Machine

Modelo reaproveitado do exemplo na biblioteca do *Scikit Learn* [5], utilizando o kernel RBF. Mais custoso computacionalmente, tempo de execução de 9min e 4s.

1) Hiperparâmetros:

- C - penalidade para erros de classificação, cada incremento suaviza mais a região de decisão.
- $gamma$ - aponta similaridade no raio em torno do objeto, facilitando o agrupamento destes juntos.

Tabela III: Grade de hiperparâmetros

Hiperparâmetro	Valor 1	Valor 2	Valor 3
C	1	10	100
$gamma$	0.001	0.01	0.1

Modelo convergiu para seus melhores resultados nas tentativas 0 e 7, com os hiperparâmetros C = 10 e $gamma$ = 0.01.

Fold 4 teve os melhores resultados no geral, altos valores de F1 e AUC ROC indicam distribuição equilibrada de classes.

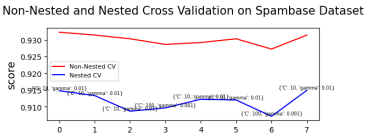

Figura 2: Scores de validação.

Tabela IV: Métricas em KFolds

Folds	Acurácia	F1	Área sob curva ROC
Fold 1	0.93314763	0.91695502	0.9745532
Fold 2	0.93444909	0.92047377	0.97587968
Fold 3	0.91352859	0.89419795	0.9663068
Fold 4	0.93863319	0.92307692	0.97302638
Fold 5	0.93584379	0.91986063	0.97404934

B. *Árvore de Decisão*

Algoritmo menos custoso computacionalmente, tempo de execução de 48.7 s.

1) Hiperparâmetros:

- Profundidade - número de divisões que a árvore deve fazer no dataset.
- Critério - métricas de pureza para cada nó de decisão, pode indicar quantidade de informação necessária para descrever a distribuição de classes (entropia).

Tabela V: Grade de hiperparâmetros

Hiperparâmetro	Valor 1	Valor 2	Valor 3
Profundidade	6	8	10
Critério	Gini	Entropia	-

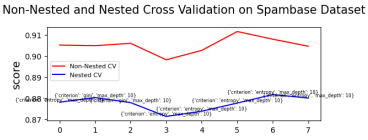

Figura 3: Scores de validação.

Maiores *scores* F1 foram para profundidade = 10 nas tentativas 1 e 6, entropia foi levemente melhor que Gini. Curiosamente a mesma combinação ganhadora também foi a perdedora na tentativa 3, folds poderiam estar desbalanceados.

Fold 5 apresentou os melhores resultados, inclusive em acurácia, diferente do caso anterior.

C. *Perceptron simples*

Originalmente desenvolvido para problemas lineares, teve o segundo maior tempo de execução de 8min e 22s.

1) Hiperparâmetros:

- Tolerância - critéro de parada, diferença máxima permitida entre as previsões atualizadas em iterações consecutivas.
- alpha - termo de penalização aplicado aos pesos do modelo durante o treinamento para evitar *overfitting*; foram testados valores considerados baixos, médios e grandes.
- Taxa de aprendizado - controla o tamanho dos incrementos nos pesos do modelo durante o treinamento.

Tabela VI: Métricas em KFolds

Folds	Acurácia	F1	Área sob curva ROC
Fold 1	0.88022284	0.85273973	0.91462792
Fold 2	0.91073919	0.89383562	0.92574005
Fold 3	0.89539749	0.87889273	0.89707437
Fold 4	0.90097629	0.87477954	0.92116091
Fold 5	0.92050209	0.89964158	0.93957617

- Peso da classe - atribuídos às classes para ajustar o desequilíbrio da distribuição de dados.

Tabela VII: Grade de hiperparâmetros

Hiperparâmetro	Valor 1	Valor 2	Valor 3
Tolerância	0.001	0.01	0.1
Taxa de aprendizado	0.001	0.01	0.1
alpha	0.001	0.5	10
Peso	balanced	None	-

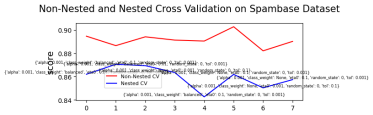

Figura 4: Scores de validação.

alpha 0.001, weight balanced, eta 0.1, tol 0.001

Trial 1 teve o melhor *score*, com Tolerância = 0.001, alpha = 0.001, taxa de aprendizado = 0.1 e pesos balanceados. Coincidentemente a mesma combinação foi otimizada na *trial* 4 com o pior resultado F1, indicativo de desbalanceio de classes.

Tabela VIII: Métricas em KFolds

Folds	Acurácia	F1	Área sob curva ROC
Fold 1	0.88300836	0.86	0.95526856
Fold 2	0.88981869	0.87027915	0.93768451
Fold 3	0.87866109	0.85475793	0.93843603
Fold 4	0.89121339	0.86956522	0.95374557
Fold 5	0.87168759	0.83211679	0.93528064

Desempenho no geral foi consistente, mas mais aprimorado nos primeiros folds, ao contrário dos outros dois algoritmos, que performaram melhor à medida que cobriam mais o dataset.

Referências

[1] "Spambase." Open Source, 1999. [Online]. Available: https://archive-beta.ics.uci.edu/dataset/94/spambase

[2] L. WEIHMANN and P. ANDRETTA, "Tópicos em aprendizado de máquina - notas de aula," 2023.

[3] S. VARMA and R. SIMON, "Bias in error estimation when using cross-validation for model selection." BMC Bioinformatics, 2006. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-7-91

[4] "Validação cruzada aninhada versus não aninhada." Runebook.dev. [Online]. Available: https://runebook.dev/pt/docs/scikit_learn/auto_examples/model_selection/plot_nested_cross_validation_iris

[5] "scikit-learn: Machine learning in python." Open Source. [Online]. Available: https://scikit-learn.org/stable/index.html