Laboratório Exame

CMC-13 Introdução à ciência de dados

Grupo:
Felipe Sato
Lucas Dias
Valério Augusto
Vinícius José de Menezes Pereira

1. Preparação de dados

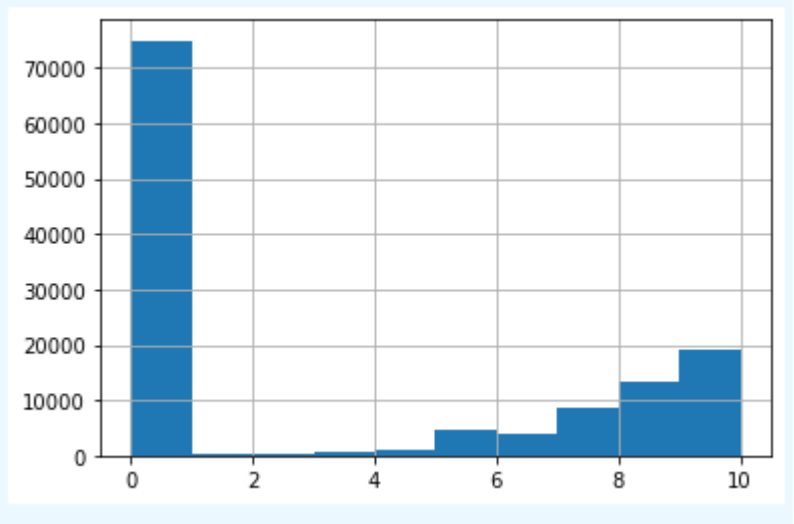
- As duas primeiras colunas do dataset que estavam sem nome não tinham utilidade alguma, por isso, foram descartadas.
- Não desejamos usar o atributo ‘user_id’, pois queríamos generalizar o modelo. Ou seja, queremos prever a nota de uma pessoa qualquer, dados seus atributos e não ficar enviesados para o dataset que nós tínhamos.
- Para a simplificação do modelo, o atributo ‘age’ foi discretizado com as idades 1, 18, 24, 30, 40 e 50. Mais detalhes no notebook.
- O ‘book_title’ e o ‘img_l’ eram atributos redundantes, pois para a identificação do livro já tinhamos o atributo isbn. Esses dois atributos, portanto, foram descartados.
- No atributo ‘Language’ tínhamos dois valores: ‘EN’ e 9. 9 provavelmente era uma maneira de definir valores faltantes. No fim, grande parte desta coluna era valor faltante. Dessa forma, esta coluna não nos dá nenhum ganho de informação, podendo ser descartada.
- As colunas ‘Category’,‘book_author’, ‘publisher’estão ok. Foi feito ‘Encoding’. Category possui 9 com NA.
- Para o local, utilizaremos a feature ‘state’ como referência. Isso porque ela não tem tantos valores únicos como ‘city’ nem é tão geral quanto ‘country’, em que a maioria das variáveis são ‘usa’, ficando enviesados. Foram feitas várias alterações nesta feature, pois ela possui amostras ‘ ‘,’,’ ou até ‘_’, que eram poucas, portanto foram removidas. Também tinham amostras como por exemplo ‘b s’ e ‘bs’, que eram a mesma coisa, portanto foram unidas. Havia um outro problema: para fazer o encoding, haviam labels do ‘state’ do teste que não haviam no treino. As linhas com essas labels diferentes eram poucas e foram removidas, sem perda significativa no número de amostras do teste.
- Normalização aplicada em ‘year_of_publication’, retirando o mínimo de cada amostra
- A quantidade de dados descartados foi razoável, cerca de 3.7%. Isso não deve comprometer a distribuição dos dados.
- Como há várias variáveis categóricas no dataset, como ‘state’, ‘isbn’, ‘book_author’, etc, seria talvez recomendado realizar OneHotEncoding. Isso, porém, não foi possível, pois deixaria o dataset com mais de 1000 colunas, o que seria inviável em termos de memória para o computador. Realizou-se, portanto, somente o Encoding.

- Para os dados serem ajustados para a rede neural, todas as colunas foram transformadas em numéricas pelo Encoding

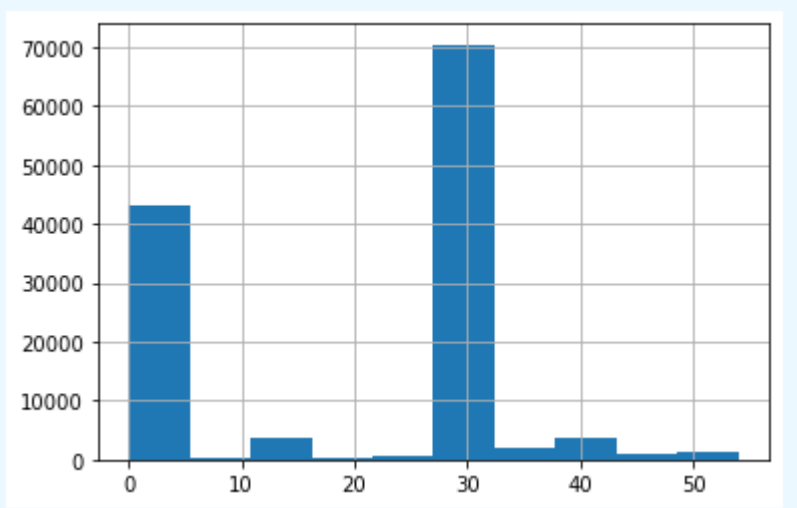
2. Exploração

Exploramos um pouco o dataset e descobrimos alguns dados interessantes.

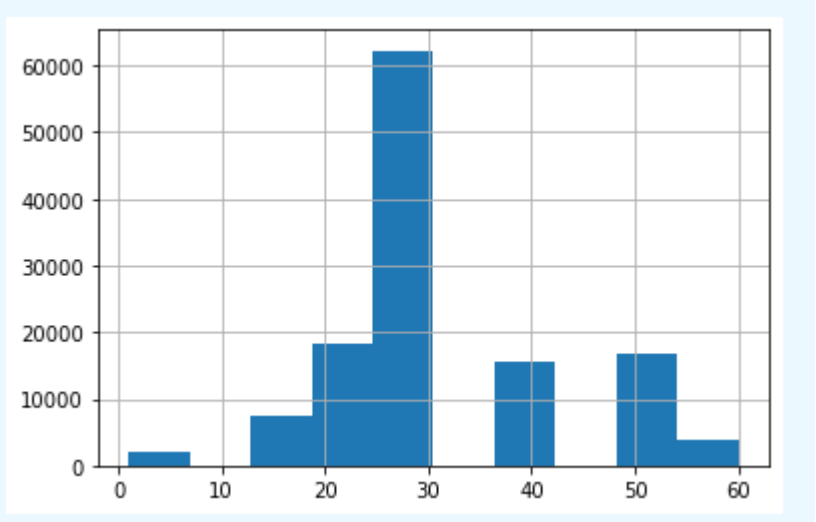
A tabela de ‘rating’ estava desbalanceada, com mais de 50% dos valores sendo 0. Há cerca de 15% com nota 10.



‘Category’ possui grande parte das amostras como ‘Fiction’ e outra parte como 9, o que provavelmente são dados faltantes. Essa coluna, portanto, tende a não nos dar muita informação



A maioria dos usuários são jovens adultos



3. Random Forests

Pela simplicidade, testamos modelos de random forest do sklearn utilizando gini como métrica de ganho com até 7 árvores, que é o mesmo número de atributos que o dataframe possui. Para escolher o melhor modelo, calculamos o chi quadrado e a acurácia do modelo conforme a imagem a seguir:

```
n_estimators: 1, chi_2: 966.11, acurracy: 0.42
n_estimators: 2, chi_2: 936.33, acurracy: 0.47
n_estimators: 3, chi_2: 952.37, acurracy: 0.45
n_estimators: 4, chi_2: 946.64, acurracy: 0.46
n_estimators: 5, chi_2: 947.94, acurracy: 0.46
n_estimators: 6, chi_2: 949.52, acurracy: 0.46
n_estimators: 7, chi_2: 950.43, acurracy: 0.46
```

A floresta com duas árvores foi a melhor que desempenhou, portanto, escolhemos essa. Para comparar ainda se não tínhamos escolhido um modelo simples demais, fizemos essas métricas para uma random forest de 500 árvores e o resultado obtido foi:

```
n_estimators: 500, chi_2: 947.35, acurracy: 0.47
```

Portanto, o modelo simples de 2 árvores continua sendo o melhor. Além disso, calculamos para esse modelo as seguintes métricas:

```
n_estimators: 2, chi_2: 938.0, acurracy: 0.47

kappa_score: 0.03,
f1_score: [0.66978253 0.         0.         0.         0.00514139 0.04358844
 0.03119584 0.06630728 0.09471567 0.08002177 0.09107856]

confusion matrix:[[14044    21    31    59    86   482   419   862  1268   747   718]
 [   43     0     0     0     0     0     1     1     1     3     0]
 [   64     0     0     2     0     2     1     3     6     6     0]
 [  117     1     1     0     1     5     3     9     7     5    14]
 [  174     1     1     2     1     5     5     8    19    10     7]
 [  847     2     4     6     8    43    30    64    66    42    45]
 [  720     0     2     4     4    32    27    54    81    36    35]
 [ 1595     0     1     4    12    53    60   123   160    93    88]
 [ 2337     2     4    11    16    95    77   149   259   158   134]
 [ 1594     3     5     8    17    54    55   121   166   147   126]
 [ 1664     2     4    10    11    45    58   127   194   131   171]]
```

Percebe-se na matriz de confusão que a primeira coluna possui números muito altos, isso deve-se ao fato de que o dataset utilizado para treino estava desbalanceado conforme descrito anteriormente. Assim, o modelo acabou incorporando esse viés de que a maioria das classificações é 0, o que prejudica o seu desempenho. Por conta disso, a acurácia não é alta.

É possível observar pelo baixo valor do kappa que esse viés foi incorporado, pois o algoritmo que chuta a maioria como 0 obteve desempenho semelhante ao classificador treinado.

4. Rede neural

Com o algoritmo sklearn, foi montada uma rede neural multiperceptron com 10 camadas de 300 neurônios. Isso porque redes mais simples com 3 camadas de 8 neurônios eram muito simples e não conseguiam aprender nada, com uma taxa de acerto de 0.3% quando o modelo era aplicado nos testes. Camadas mais profundas e com mais neurônios foram a solução encontrada para resolver tal problema.

Foi utilizada uma `MLPClassifier`, pois a regressão acabava dando notas decimais para os livros. Isso nos deu um score de 59% tanto nas amostras teste quanto nas de treino. Tal fato indica que teoricamente não há um enviesamento para os dados de treino especificamente.

No fim, infelizmente a rede acabou aprendendo o ‘chutar’ zeros para quase todas as amostras, como pode ser observado na matriz de confusão abaixo nos dados de teste:

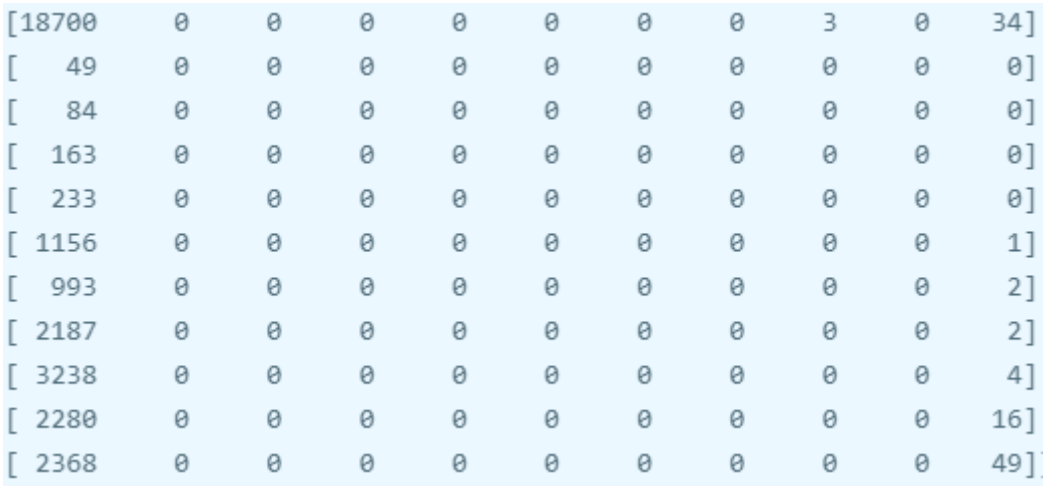


Figura 1: Matriz de confusão do teste

Isto também pode ser observado na matriz de confusão dos dados treino, conforme o modelo:

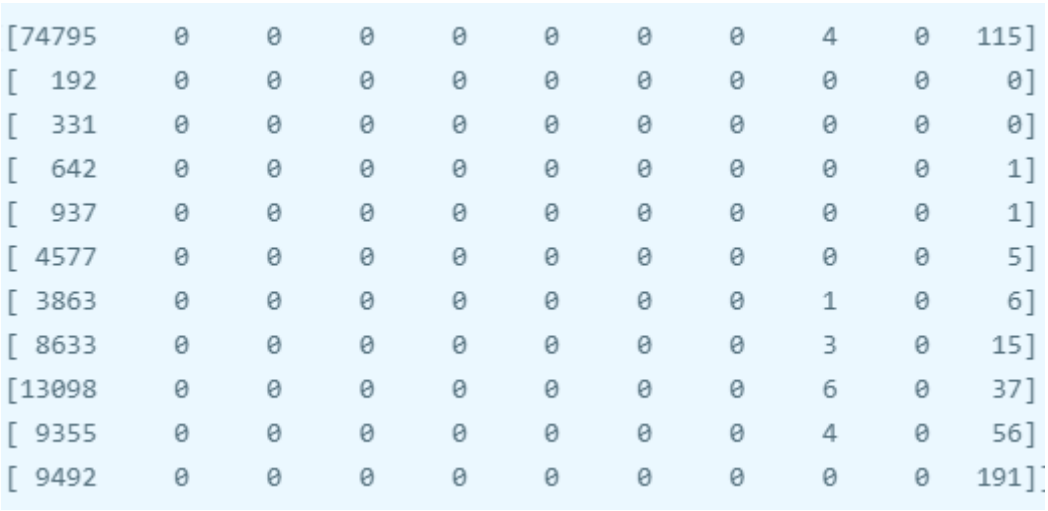


Figura 2: Matriz de confusão do treino

Isso acontece porque os dados das notas estão desbalanceados, cerca de mais de 50% dos dados de treino tendo nota 0. Um fato semelhante também acontece nos dados de teste. Daí podemos entender bem a natureza do fenômeno: a alta taxa de acerto não indica que o modelo está bom, pois os dados estão desbalanceados, o que dificulta e confunde bastante a rede neural.

É interessante notar que mesmo uma rede neural relativamente profunda, com 10 camadas, e com muitos neurônios, 300 em cada camada, que demorou 70 minutos, não conseguiu aprender bem os parâmetros relevantes para determinar a nota do usuário. Isso mostra como o desbalanceamento do dataset é ruim para o aprendizado.

Ao analisarmos o recall, vemos que a quantidade de falsos negativos para o ‘0’ deve ser bem baixa, o que pode explicar porque a rede neural acabou aprendendo que a melhor decisão praticamente sempre seria chutar ‘0’ como resposta.

```
[0.9980253  0.         0.         0.         0.         0.
0.         0.         0.         0.         0.02027307]
```

Figura 3: Recall

Ao analisarmos a precision, vemos que a quantidade de falso positivo ‘0’ deve ser bem alta. Isto indica que muitos valores que deveriam resultar em outras notas, tais como 1,2,3,...10 foram apenas ‘chutados’ como ‘0’, o que indica enviesamento do modelo.

```
[0.59457569 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.4537037 ]
```

Figura 4: Precision

Ao analisarmos a estatística de kohen kappa como `0.0053 ,` vemos que a correlação entre as notas previstas e as reais do teste é muito baixa, indicando insuficiência do modelo para descrever o problema.

5. Análise comparativa entre modelos

Quando se observa a acurácia, o modelo de redes neurais desempenhou melhor. No entanto, quando olhamos o fator kappa, o random forest tem um resultado consideravelmente melhor. O MLP desempenhou melhor porque chuta 0 para quase todas situações e sua acurácia fica boa porque o dataset está desbalanceado. A random forest, portanto, está, por certo modo, menos enviesada, mas acabou por prever muito mal o problema. Às vezes, o bias não é tão ruim quanto se pensa, principalmente quando os dados da vida real já são enviesados, como é o que vemos. A rede neural absorveu esse viés e acabou fazendo um modelo que conseguiu prever bem os testes. Sabemos, porém, que algoritmos de aprendizado de máquina não lidam bem com dados desbalanceados, o que explica nosso baixo desempenho com os dois métodos. Assim, quando olhamos o aprendizado em si, vemos que a random forest conseguiu um melhor desempenho, mesmo com uma acurácia menor. Quando olhamos isso com o viés, vemos que a rede neural apresenta melhor acurácia.

6. Conclusão

Obrigado, professor `Paulo André Lima de Castro`, pela oportunidade de realizar este curso e executar estas tarefas que nos ajudam em nosso aprendizado. Neste laboratório, aprendemos que nem sempre é possível prever com muita precisão o comportamento dos dados com aprendizado de máquina, sendo a técnica por muitas vezes falha e limitada. Isso não significa que deva ser descartada: pelo contrário, deve ser aprimorada e utilizada nos momentos corretos, com muita cautela e senso crítico.