Universidade Federal de Santa Catarina

Mineração e Classificaçãode Páginas Web

DAS410058 - Aprendizado de Máquina

Alunos
Andrei Donati
Angeo Baruffi

Professor
Jomi Hubner

Outubro de 2017

Sumário

1 Dataset

O desafio do trabalho proposto é a classificação de 8282 páginas da web divididos em 7 tipos de páginas ('course', 'department', 'faculty', 'project', 'staff', 'student'). Oficialmente, o nome do dataset é *webkb-4-universities-wisconsin-test*. Para melhor entendimento dos dados, foi feito a visualização de quantas documentos existiam em cada classe:

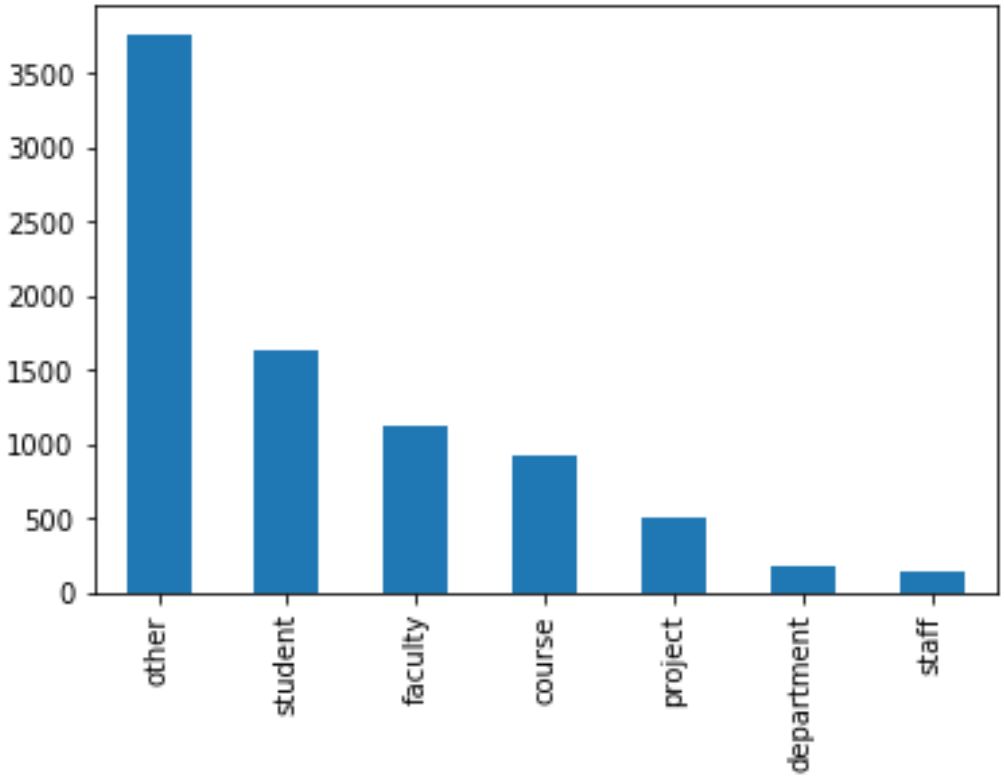


Figura 1: Número de exemplos por classe

Com a imagem acima já é possível perceber que o Dataset é bastante desbalanceado, o que faz pode gerar graves problemas de classificação para alguns algorítmos de classificação.

Se excluírmos a classe "Other"do Dataset, ficamos com a seguinte divisão:

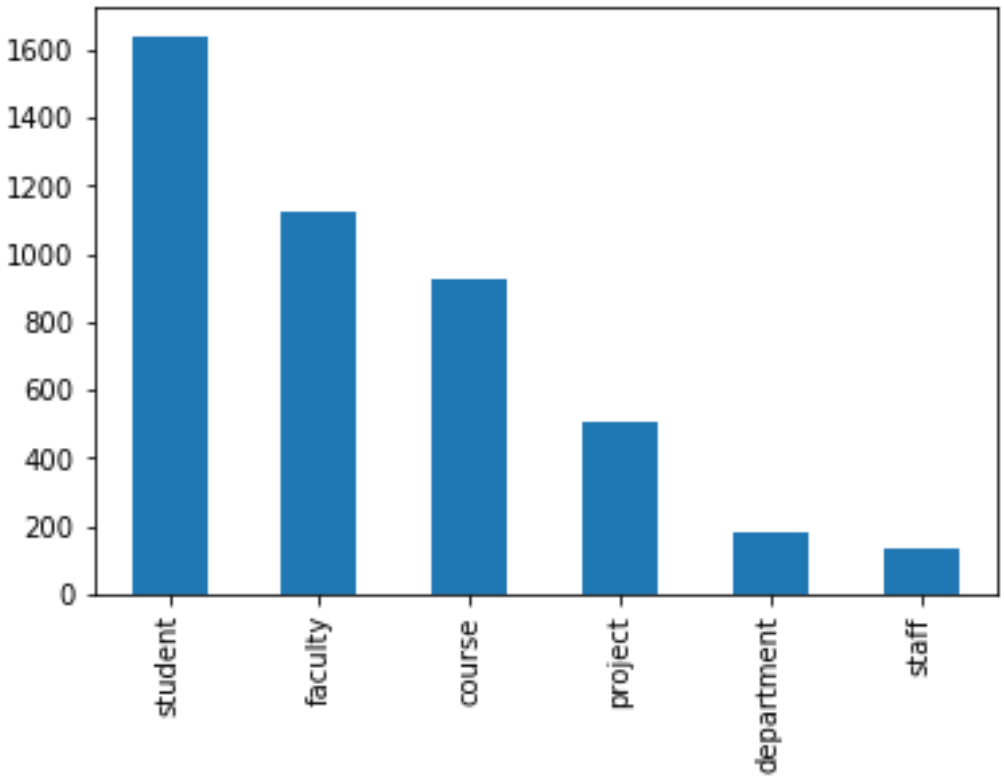


Figura 2: Número de exemplos por classe

O total de indíduos, neste caso, é de 4518 páginas web.

Os dados também eram divididos em universidades (não é o objetivo de classificação do trabalho) e dos quais:

- Cornell: 867 páginas
- Texas: 827 páginas
- Washington: 1205 páginas
- Wisconsin: 1263 páginas

Analisando as páginas, foi possível notar que a classe "other" tinha páginas muito distintas, que não apresentavam nenhum padrão entre si e poderiam prejudicar bastante a classificação das demais páginas.

2 Tratamento e Limpeza dos Dados

Como primeiro passo para a classificação dos de textos, é necessário fazer uma grande limpeza e extração de características, que posteriormente serão usados como features nos algoritmos de classificação.

Todo o código desenvolvido foi feito na linguaguem python, usando várias bibliottecas auxiliares e a sequencia apresentada a seguir é a mesma utilizado pelo grupo para o desenvolvimento do trabalho.

2.1 Aquisição e Compilação dos Dados

Os dados foram adquiridos usando a biblioteca Python BeatifulSoup, a qual tem capacidade de extrair features como número de imagens, número de tags H1 e o título das páginas. A saída desta etapa é um arquivo csv contendo várias caracteristicas das páginas, incluindo sua classe e o texto da página:

id	class	all_text	all_text_count	title	title_count	h1	h1_count	h2	h2_count	h3	h3_count	a	a_count	img_coun
0	project	network c	182	network comput s	1	network comput	1	search the ncst	3		0	ncstrl partici	10	1
1	project	the desigr	66	the design researc	1	design research i	1		0		0	about the dr	15	10
2	project	the horus	569	the horus project	1		0	introduct to hc	1	the horus	1	isi paper ab	23	20
3	project	welcom tc	138	welcom to prema	2		0	prema portabl	1	overview	4	prema porta	11	0
4	project	seq home	1367	seq home page	1	the seq project q	1	document cont	11	time to pu	1	project objec	37	12
5	project	mathemat	1227	mathemat for com	1	aster demonstr	1	how to use this	19		0	titl in audio g	232	1
6	project	tom henzi	11	tom henzinger hyt	1	hytech the hybric	1		0		0	we have mo	1	0
7	project	qmg proje	128	qmg project	1	qmg mesh genera	1		0		0	qmg releas	7	0
8	project		0	cachet relat projec	1	deriv increment g	1	theme cachet s	5		0	increment sy	9	0
9	project	cornel act	1033	cornel activ messa	1	cornel activ mess	1	cornel activ me	3	activ mess	2	unet project	27	6
10	project	cornel csr	615	cornel csrvl	1	cornel robot and	1	about the csrvl	5		0	prof daniel h	53	5
11	project	horus dist	519	horus distribut cor	1	horus distribut cc	1		0		0	kenneth birr	11	3
12	project	inform ca	213	inform captur and	1	inform captur anc	1	current area of	2		0	www server	3	0
13	project	cornel me	161	cornel medianet p	1	medianet a high p	1		0		0	unet cmhoru	6	1
14	project	cornel nu	77	cornel nuprl autor	1		0	nuprl project	1		0	introduct to	13	2
15	project	cornel mo	203	cornel model and s	1	cornel model and	1	overview of th	6		0	overview of	15	1

Figura 3: Dataset das páginas HTML

Na esquerda para direita, temos o id da página, todo o seu texto (sem nenhum tratamento), número de palavras do corpo da página, título, número de palavras no título, texto das tags H1, H2 e H3 bem como o número de palavras com essas tags e por último o número de imagens em todo a página.

2.2 Limpeza de palavras sem sentido

Algumas palavras dentro das páginas não tem sentido para a tarefa de classificação. Dentre elas estão nomes próprios, números, abreviações e pontuação. Dessa forma, todo o texto foi tratado para retirar estas palavras. Além disso, verbos conjugados em tempos diferentes ou para pessoas diferentes podem ter o mesmo significado, dessa forma é importante transformá-los para o infinitivo para auxiliar o classificador. Esse processo foi realizado utilizando a função "SnowballStemmer". Também foram retiradas palavras

que apareciam em muitos texto ou que apareciam em pouquíssimos textos, pois elas ou eram muito específicas (por exemplo um nome próprio) ou muito genéricas (por exemplo o conectivo "and", o artigo "the").

2.3 TF-IDF

Os classificadores, em geral, trabalham bem com features numéricas. Dessa forma é interessante gerar um conjunto de características numéricas Palavras que aparecem em muitos textos (mas que ainda assim apresentam algum sentido para a classificação) devem ser um peso para o classificador pequeno. Palavras que aparecem em poucos textos devem contar mais para o classificador. Assim, para fazer essa ponderação foi tulizado um algoritmo já consagrado na literatura. A fórmula que ele utiliza é:

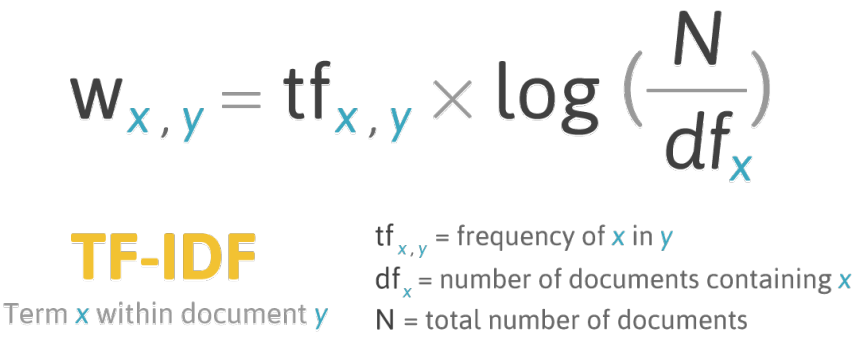
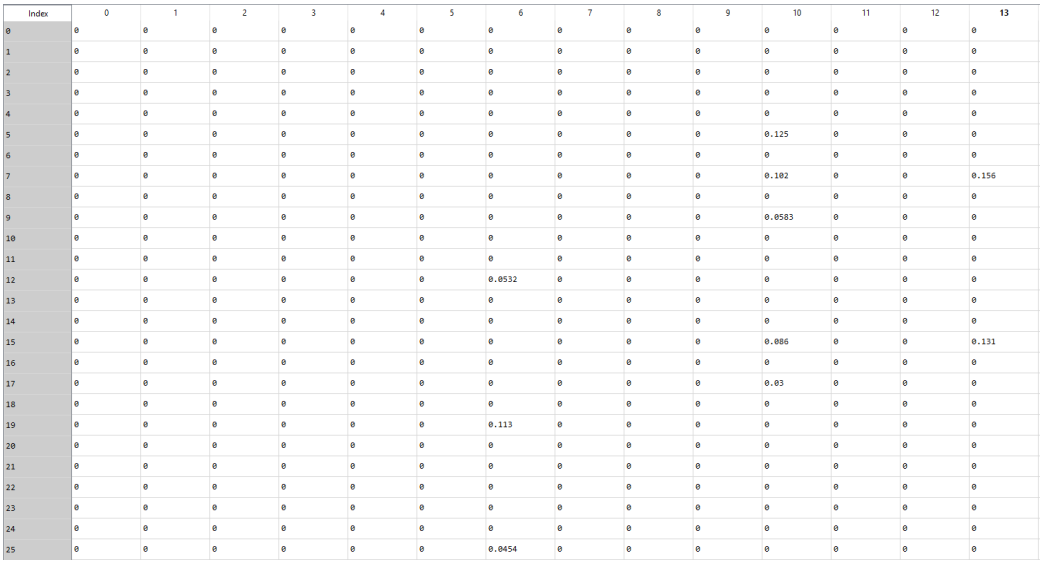


Figura 4: Fórmula TF-IDF (crédito [1])

Como saída dessa etapa é obtido uma matriz com as colunas sendo todos as páginas e as colunas todas as palavras diferentes de todos os textos e os valores são o resultado da fórmula TF-IDF. Segue uma imagem da matriz obtida pelo grupo:



Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0.125	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0.102	0	0	0.156
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0.0583	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0.0532	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0.086	0	0	0.131
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0.03	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0.113	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0.0454	0	0	0	0	0	0	0

Figura 5: Matriz de pesos segundo o TF-IDF

A matriz toda tem 6676 colunas (palavras diferentes).

2.4 Geração de Datset de Teste e Treino

Para treino e teste dos algorimos de classificação as páginas web foram dividas em um conjunto de treino e um conjunto de teste. Para o treino foi reservado 65% do teste de forma que as classes continuem com a mesma proporção de indivíduos.

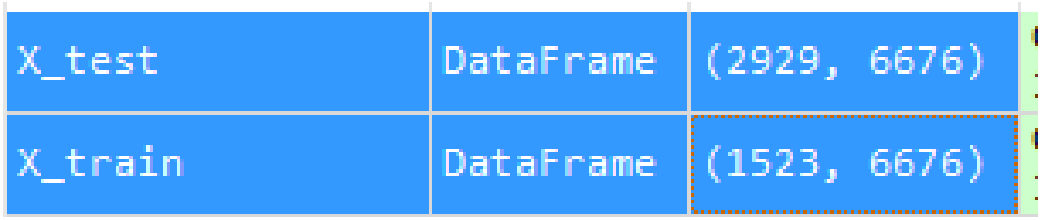


Figura 6: Matriz de treino e teste

As dimensões das matrizes são equivalentes ao número de exemplos e a quantidade de palavras diferentes no dataset.

2.5 Exclusão da classe ”other”

A equipe avaliou vários documentos da classe ”other”e por ser uma classe que contém pequenas quantidades de vários tipos de páginas é bastante difícil

de encontrar um padrão entre os documentos. Isso faz com que todos os classificação de todas as classes fiquem prejudicadas. Para escluir este problema e possibilitar a equipe focar na pesquisa e comparação de algoritmos de classificação que busquem padrões consistentes, essa classe foi excluída dos datasets de treinos e teste.

3 Técnicas de Classificação

Foram testadas várias tecnicas de classificação, várias delas presentes em [3]. As tecnicas expostas neste relatório foram as que melhor desempenharam no conjunto de teste.

Classifier	Acc_train	Acc_test	Time
ExtraTreesClassifier	99,21%	86,89%	2,05
Random Forest	98,62%	89,28%	4,63
XGBoost	99,41%	90,65%	17,64

Figura 7: Acurácia dos algoritmos

Por serem todos baseados em árvores de decisão, todos apresentam desempenho bastante similar porém o XGBoost apresentou desempenho levemente superior que os outros dois.

Algumas classes apresentaram desempenho muito ruim. Isto se deve a uma falta de padrão (em geral, alunos não seguem um padrão de apresentação em suas páginas pessoais, cursos, por exemplo têm uma estrutura básica que se repete com bastante frequência).

3.1 Random Forest

O algoritmo foi executado no conjunto de testes e apresentou um resultado satisfatório. Ele apresentou overfitting aos dados, como foi visto acima (acurácia no conjntop de treino muito superior que no conjunto de testes). Dessa forma, foi obtido:

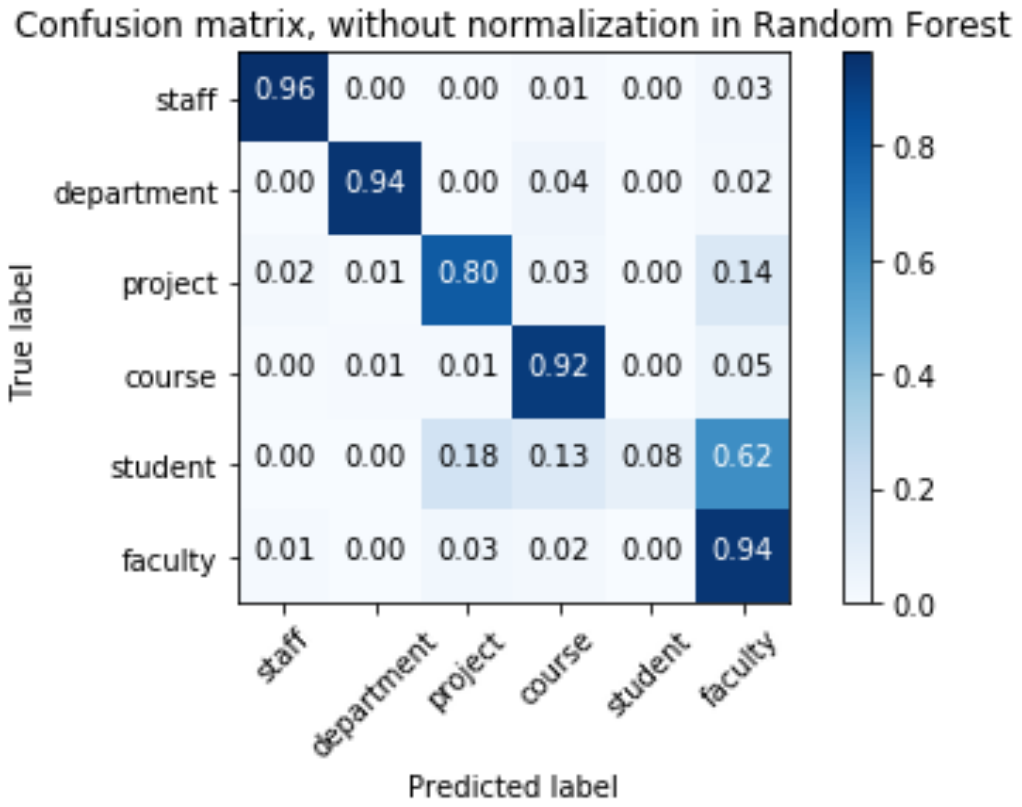


Figura 8: Matriz de confusão

A matriz de confusão acima mostra que algumas classes estão com um desempenho excelente (staff com acurácia de 96%) porém a classe "student"está muito ruim (levemente superior a um classificador randômico).

3.2 Extra Trees Classifier

O algoritmo foi executado no conjunto de testes e apresentou um resultado muito similar ao anterior. Dessa forma, foi obtido:

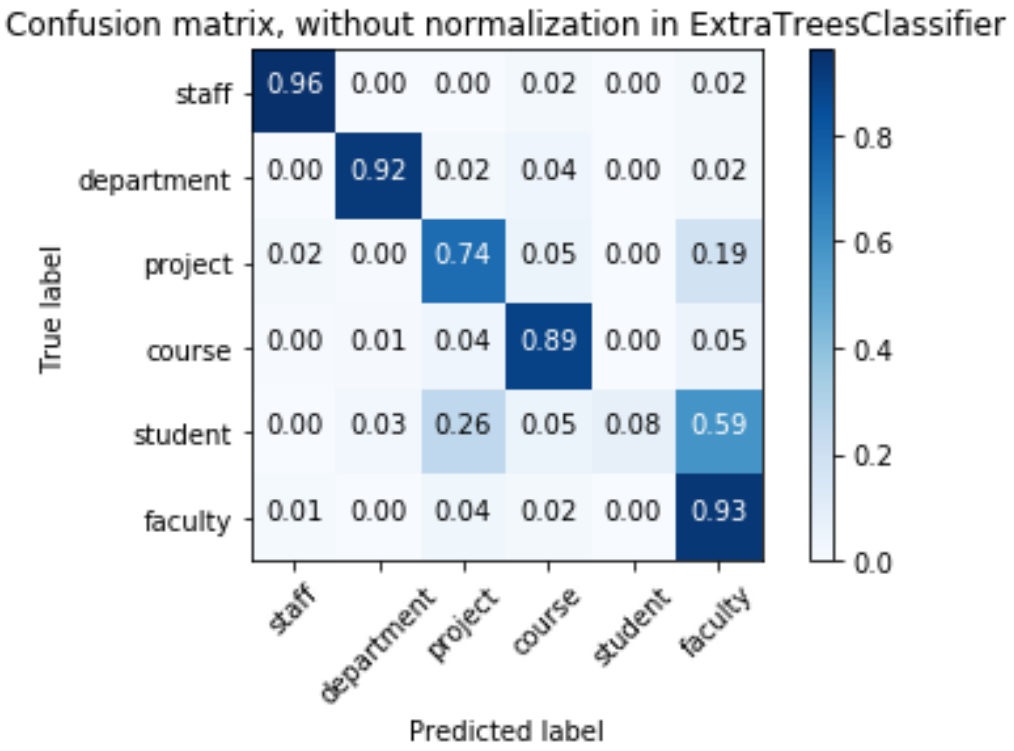


Figura 9: Matriz de confusão

A matriz de confusão acima mostra novamente que o problema com a classe "student"persiste e a melhor classe continua "staff"com acurácia de 96%.

3.3 XGBoost

O algoritmo foi executado no conjunto de testes e apresentou o melhor resultado. Alguns parâmetros tiveram que ser ajustados como a profundidade máxima das árvores e o número máximo de features (definido para raiz quadrado no número original de features). Dessa forma, foi obtido:

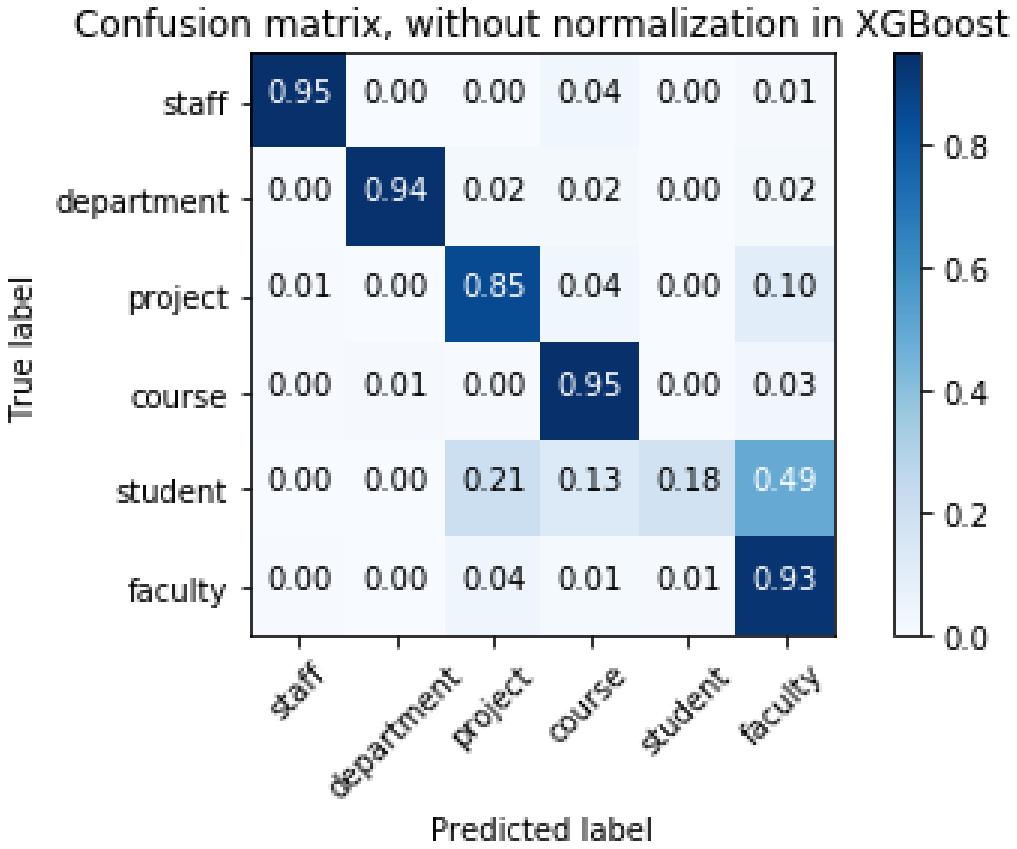


Figura 10: Matriz de confusão

A matriz de confusão acima mostra que o algoritmo está melhor ponderado (a classe "student"estão um pouco melhor e a classe "staff"está um pouco pior). No geral, ele está significativamente superior aos outros.

4 Trabalho Futuro

Como próximos trabalhos que poderiam ser desenvolvidos é o teste de técnicas de aprendizado profundo (redes neurais profundas e redes neurais recorrentes, como a Long short-term memory). Além disso, poderia ser tentado abordagens na qual vários classificadores são aplicados no conjunto de teste e, utilizando o resultado destes, é utilizado uma votação (ou até votação ponderada) para, no final, decidir a classe do texto. Esse método é chamado de Ensemble learning.

5 Código

O código está dividido em 3 arquivos Python. Para executá-lo é necessário ter as bibliotecas apresentadas no arquivo "requiriments.txt". O arquivo HTMLClassification.py é o responsável pela leitura e visualização básica dos dados. Este arquivo importa o arquivo csv descrito na seção 2.1 que foi gerado pela equipe e otimiza o tempo de execução do código. A arquivo mining.py contém toda a limpeza, tratamento e geração de features dos dados para os classificadores. Além disso, é neste arquivo que há os classificadores utlizados pela equipe. Em funcs.py há todas as funções desenvolvidas e utilizadas pela equipe para fazer extração, tratamento, limpeza e geração de features dos dados. A sequência de execução deve ser HTMLClassification.py para depois mining.py

O código também pode ser obtido em *github.com/Angelo-Baruffi/HTML-classification*

Referências

[1] http://filotechnologia.blogspot.com.br/2014/01/a-simple-java-class-for-tfidf-scoring.html - Acessado em 30 de setembro de 2017.

[2] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/ - Acessado em 25 de setembro de 2017.

[3] Scikit Learn Documentation. Disponível em http://scikit-learn.org/stable/documentation.html - Acessado em 27 de setembro de 2017.

[4] RAMOS, Juan et al. Using tf-idf to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning. 2003. p. 133-142.