Universidade de Évora

Engenharia Informática

Dataset - Student dropout

Rodrigo Alves, 48681

António Nanita, 48407

Eugeniu Musteata, 45824

OBJETIVO

Utilizando informação do histórico académico dum conjunto de alunos (curso, ECTS matriculados e concluídos e notas médias ao longo de vários semestres), construir um modelo preditivo que responda à pergunta: "quais os alunos em risco de abandonar os estudos?"

Descrição do Trabalho

Para este trabalho foram utilizados dois ficheiros (trabalho2.py e teste.py) com o objetivo de determinar qual o melhor modelo relativamente para um conjunto de dados.
Como foi descrito anteriormente, são usados dois ficheiros, sendo que um serve para averiguar qual de entre os vários modelos que possui as seguintes características:

- Maximizar a cobertura
- Garantindo um mínimo de 70% de precisão.

Desta forma, é feita a descrição do que se define como o melhor modelo. A descoberta do melhor modelo é feito pelo ficheiro trabalho2.py, enquanto no ficheiro teste.py é feito o calculo da precisão e da cobertura entre predição do modelo escolhido, que é devolvido pelo método *predict(),* e o conjunto de testes (*X_test*).

Decisões tomadas

Foram utilizados diferentes algoritmos de classificação, dos mais básicos aos mais complexos de modo a determinar qual o melhor modelo. Desta forma conseguimos ter a garantia de que o algoritmo escolhido é o mais adequado.

Algoritmos utilizados:

- K-neighbors Classifier
- Decision Tree Classifier
- Gaussian Naive Bayes
- Logistic Regression
- Gradient Boosting
- Random Forest
- Dummy Classifier

Modelo Class

Nesta classe **Modelo**, que se encontra no ficheiro trabalho2.py, é onde é feito o estudo de qual o melhor modelo. Este processo de averiguação é definido pelas seguintes etapas:

1. Em primeiro lugar é definido o ficheiro que vai ser utilizado para po0steriormente serem extraídos os dados. Esta operação é efetuada no construtor da classe, o nome do ficheiro é guardado pela variável *self.data_file.*

2. De seguida, fazemos a leitura do conjunto de dados que está presente no ficheiro, esta etapa é realizada pelo método *read().* A tabela que foi lida vai ser armazenada na variável *self.dataset*.

3. A variável acima definida vai permitir treinar o conjunto de dados e definir o *X_train* e o *y_train.* Este processo é definido pelo uso do *train_test_split(X,y,test_size=0.2)*. A utilização de 20% dos dados deve-se ao facto de tentamos diminuir a probabilidade de ocorrer overfitting.

4. Após termos o *X_train* e o *y_train,* podemos calcular qual é o melhor modelo, esta decisão é feita no método *best_model()*. Neste método referido é feito em primeiro lugar o calculo do *predict* para cada modelo, para isso utilizamos os seguintes métodos:
   pred_model_KNN(),pred_model_DecisionTree(),pred_model_GaussianNB(),pred_model_LogisticRegression(),pred_model_GradientBoosting(),pred_model_Random_Forest(),pred_model_DummyClassifier(), pred_model_RandomForest_2atributes().

5. *O objetivo secundário deste trabalho pretende demonstrar a utilização de um conjunto de dados alterado, ou seja, o conjunto de dados que foi utilizado no objetivo principal vai ser alterado, isto é, vão ser eliminados atributos/colunas, e vai ser adicionada uma nova coluna denominada media, que ira representar a media de cada aluno. Desta forma, o nosso X_train vai possuir apenas 2 atributos que vão ser, licenciatura do aluno em causa e a media.*

6. Depois de terem sido calculadas todas as predict's vamos iniciar a avaliação de cada modelo, em primeiro lugar vamos testar se cada modelo respeita uma das condições necessárias para ser classificado como um forte candidato, que é ter uma precision_score maior ou igual a 0.7 (70%). Esta etapa é realizada quando é feita a passagem de um algoritmo do *array* de algoritmos definido como *First_Verificacion para o array de algoritmos* Second_Verification. Desta forma só calculamos os valores de cobertura para os modelos que respeitem a primeira condição avaliada.

7. No Second_Verification será avaliado qual é o modelo que possui a maior cobertura, este será retornado pelo método *best_model()*.

O *predict* do modelo classificado como o melhor vai retornado pelo método *predict().*

Valores Obtidos

	Modelo	Precision	Recall
1	KNN	0,762	0,353
2	DecisonTree	0,842	0,874
3	GaussianNB	0,0	0,0
4	LogisticRegression	0,0	0,0
5	GradientBoosting	0,923	0,892
6	Random_Forest	0,914	0,856
7	DummyClassifier	0,277	0,485

Perante este caso, podemos concluir que o melhor modelo a será o GradientBoosting pois possui um valor de cobertura maior do que os outros modelos com a precisão superior ou igual a 0,7. Desta forma, será retornado o predict do algoritmo GradientBoosting que vai ser armazenado na variável *y_pred_teste* que vai ser utilizada para calcular a precisão e a cobertura com o *y_teste* do ficheiro "dropout_teste.csv".

Decisões tomadas

Foram utilizados diferentes algoritmos de classificação, dos mais básicos aos mais complexos de modo a determinar qual o melhor modelo. Desta forma conseguimos ter a garantia de que o algoritmo escolhido é o mais adequado.

Algoritmos utilizados:

- K-neighbors Classifier
- Decision Tree Classifier
- Gaussian Naive Bayes
- Logistic Regression
- Gradient Boosting
- Random Forest
- Dummy Classifier

Random Forest :

-No Random Forest o único parâmetro que foi alterado foi o estimador sendo utilizados como valores de teste: 100, 200, 500 e 800.

Estimador	Precision	Recall
800	0,934	0,865
500	0,913	0,867
200	0,922	0,872
100	0,910	0,890

A partir da tabela podemos concluir que os valores da precisão são próximos, sendo que o valor de precisão para um estimador igual a 200 “engloba” os valores de precisão calculados com estimador de 100 e de 500. Assim, decidimos que o melhor estimador a utilizar seria o de 200 pois é abrangente.

Decisão: Estimador = 200

Gradient Boosting :

-No Gradient Boosting o único parâmetro que foi alterado foi o estimador sendo utilizados como valores de teste: 100, 200, 500 e 800.

Estimador	Precision	Recall
800	0,907	0,901
500	0,911	0,901
200	0,913	0,894
100	0,908	0,889

A partir da tabela podemos concluir que os valores da precisão são próximos, sendo que o valor de precisão para um estimador igual a 200 “engloba” os valores de precisão calculados com estimador de 100, 500 e 800. Assim, decidimos que o melhor estimador a utilizar seria o de 200 pois é abrangente.

Decisão: Estimador = 200

K-neighbors Classifier :

- No KNN o único parâmetro que foi alterado foi o n_neighbor sendo utilizados como valores de teste: 1, 2, 3, 4 e 5.

n_neighbor	Precision	Recall
5	0,620	0,380
4	0,680	0,340
3	0,700	0,411
2	0,830	0,370
1	0,650	0,602

A partir da visualização da tabela, podemos concluir que o valor de n_neighbor 2 é o que se destaca pelo que é o valor mais alto e que engloba os outros valores de precisão de n_neighbor superiores ou inferiores.

Decisão: n_neighbor = 2

Dummy Classifier :

- No Dummy Classifier o único parâmetro que foi alterado foi o strategy sendo utilizados como valores de teste: uniform, most_frequent e stratified.

Modelo	Strategy	Precision	Recall
Dummy Classifier	uniform	0,263	0,480
Dummy Classifier	most_frequent	0,0	0,0
Dummy Classifier	stratified	0,291	0,263

Concluímos que a estratégia uniform comparada com as outras estratégias que foram testadas, esta possui um valor de precisão menor do que a stratified mas sobressai no valor de cobertura.

Decisão: Strategy = uniform

Conclusão

Concluindo, neste trabalho foram aplicados diferentes algoritmos de classificação relativamente a um conjunto de dados. Desta forma permitiu-nos aumentar o conhecimento de cada algoritmo, bem como da importância, em certos casos, dos valores dos parâmetros definidos para cada um deles.