

Previsão de novos comentários



Métodos Estatísticos

Turma 7 - Grupo D

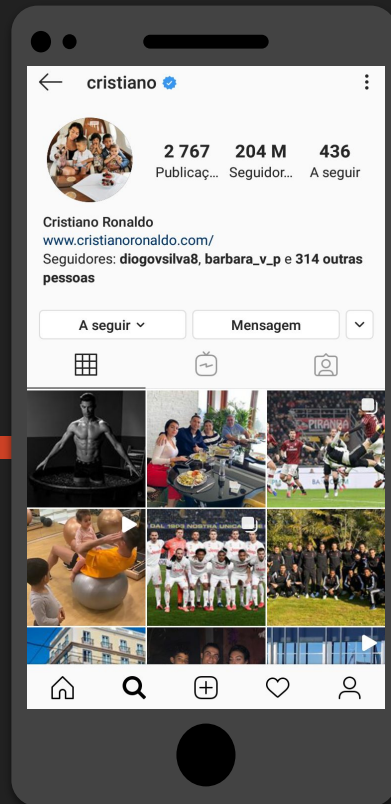
Trabalho realizado por:

Filipe Campos | up201905609
Francisco Cerqueira | up201905337
João Rocha | up201806261
Vasco Alves | up201808031

Problema



Prever se, numa dada publicação recente, haverão novos comentários nas seguintes 24 horas, tendo como ponto de partida um conjunto de dados sobre publicações anteriores.



Dados



Page Likes
Page Interaction
Page Category



Nº Comments
Delta 48 24h
Character Count



Day Posted
Current Day



Number of
Shares



Has New
Comments

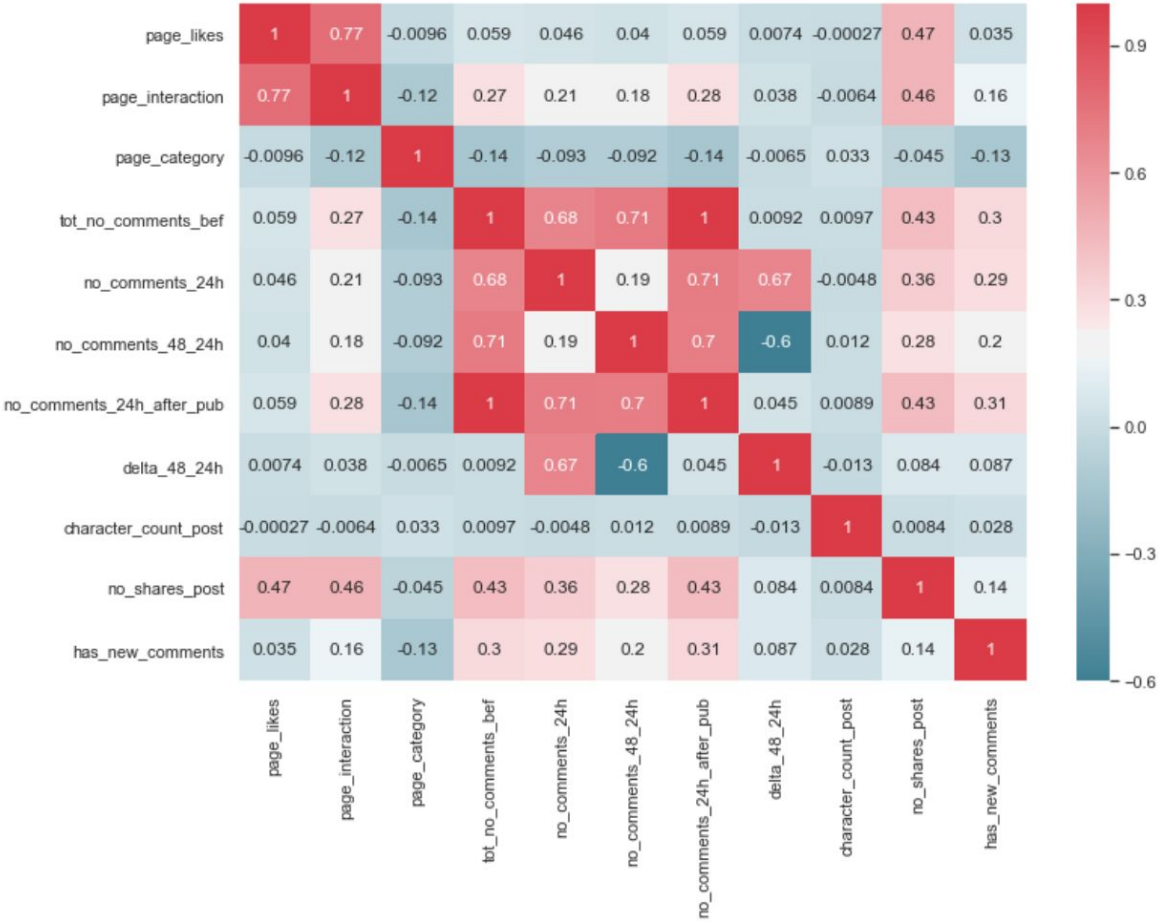


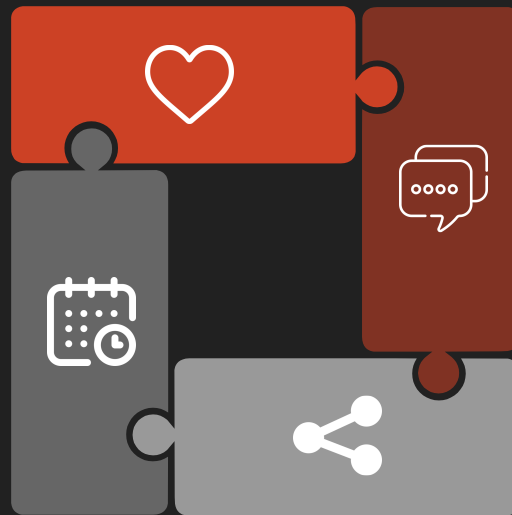
Gráfico 1 - Relação entre as variáveis

Preparação dos Dados

Inicialmente, foi realizada uma avaliação individual da importância de cada uma das variáveis.

Posteriormente, foi analisada a existência de outliers utilizando boxplots individuais de cada dado. Em seguida, foram removidos usando o método IQR (Interquartile Range).

No entanto, a precisão dos modelos testados diminuiu relativamente em relação a modelos idênticos com outliers presentes.



Preparação dos Dados

Page Category - métodos testados:

- One Hot Encoding - Transformar cada categoria numa coluna binária.
- Binary Encoding - Criar 7 colunas binárias, correspondendo o valor representado pelas colunas a uma categoria.
- Criação de 2 colunas binárias:
 - A categoria tem, em média, "has_new_comments" = "yes" > 70%
 - A categoria tem, em média, "has_new_comments" = "yes" < 30%



Para o mesmo modelo, Binary Encoding foi o método que apresentou maior precisão.

Modelação

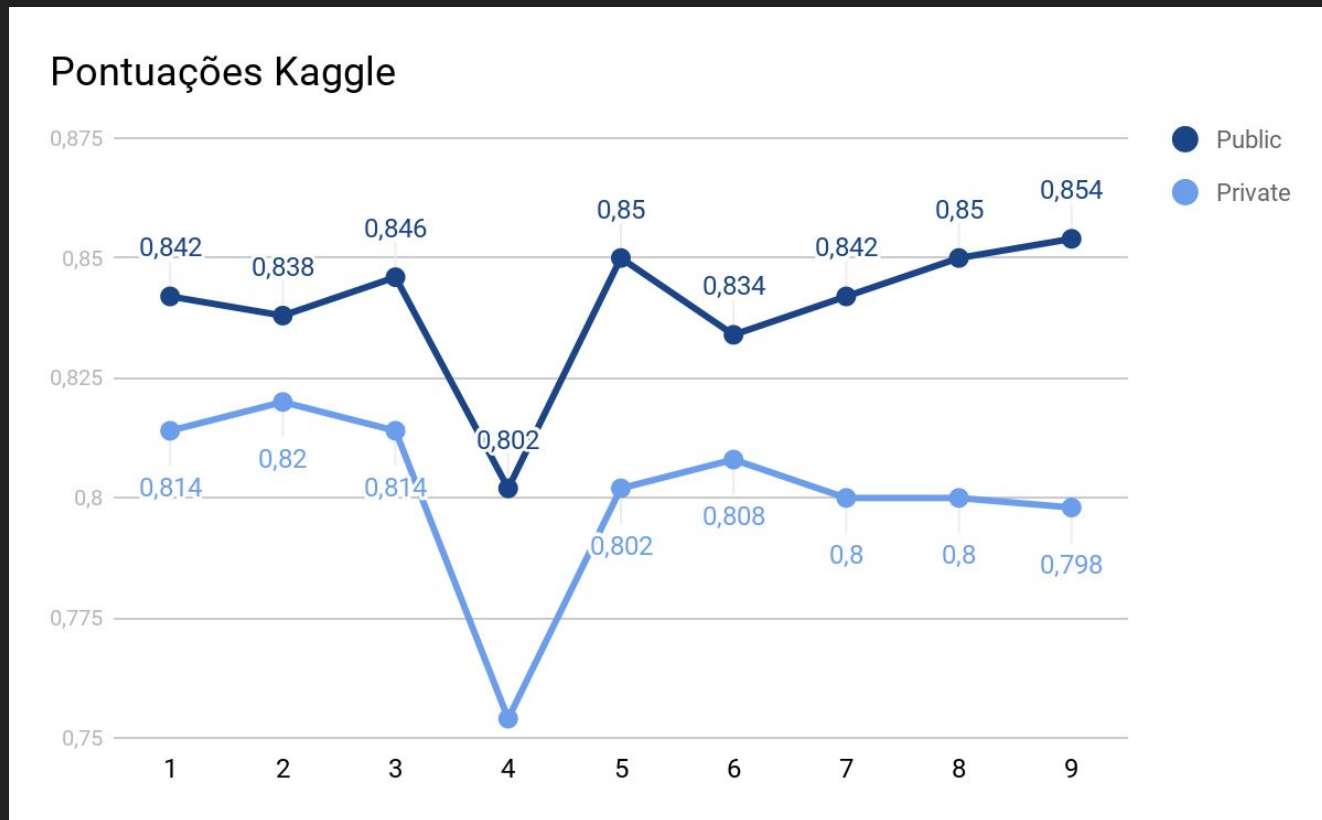
Attempt	1	2	3	4	5	6	7	8	9
Public	84.2%	83.8%	84.6%	80.2%	85.0%	83.4%	84.2%	85.0%	85.4%
Private	81.4%	82.0%	81.4%	75.4%	80.2%	80.8%	80.0%	80.0%	79.8%
Method	Decision Tree	Deep Learning	Random Forest	Gradient Boosting Classifier	Random Forest	Random Tree	Hard Vote Classifier	Hard Vote Classifier	XGB Classifier

A metodologia deste trabalho teve como base a utilização de vários algoritmos, aplicando, na maioria dos casos, 80% dos dados como treino e 20% para avaliação da previsão realizada.

No final, os 2 métodos com maior percentagem de acerto foram os escolhidos para submissão no Kaggle.

Resultados

Gráfico 2 - Pontuações Kaggle



Resultados



O melhor resultado foi obtido através da utilização do operador Deep Learning no Rapidminer, com um score de 0.82000.




O pior resultado foi obtido através da utilização do operador Gradient Boosting Classifier no Rapidminer, com um score de 0.75400.

Observação: Os resultados privados, na sua totalidade, foram inferiores aos resultados públicos.

Conclusões

A realização deste projeto permitiu-nos compreender a importância do conhecimento sobre estatística e que, dependendo de cada situação, podem ser utilizados vários modelos para realizar previsões, sendo importante ter um bom conhecimento sobre análise de dados de modo a maximizar a utilização destes modelos.





Previsão de novos comentários



Parte Privada

Python



Nos exemplos apresentados, a variável data é um *DataFrame* com os dados do ficheiro dev.csv

Remoção de outliers

```
for col in data.columns:
    if col not in ["page_category", "day_posted", "current_day", "has_new_comments", "ID"]:
        Q1 = np.percentile(data[col], 25)
        Q3 = np.percentile(data[col], 75)
        IQR = Q3 - Q1
        upper_bound = Q3 + 1.5 * IQR
        lower_bound = Q1 - 1.5 * IQR

        data[col] = data[col][data[col].between(lower_bound, upper_bound)]
```

Teste de vários algoritmos

Este algoritmo permitiu
testar rapidamente a
accuracy de diversos
modelos de classificação,
usando *cross validation*.

```
from sklearn import ensemble, tree, svm, neighbors, discriminant_analysis, naive_bayes
from sklearn.model_selection import cross_val_score
import xgboost
```

```
ALG = [
    svm.SVC(probability = True),
    svm.LinearSVC(),

    neighbors.KNeighborsClassifier(weights='distance'),

    discriminant_analysis.LinearDiscriminantAnalysis(),
    discriminant_analysis.QuadraticDiscriminantAnalysis(),

    naive_bayes.BernoulliNB(),
    naive_bayes.GaussianNB(),

    tree.DecisionTreeClassifier(),

    ensemble.AdaBoostClassifier(),
    ensemble.BaggingClassifier(),
    ensemble.ExtraTreesClassifier(),
    ensemble.GradientBoostingClassifier(),
    ensemble.RandomForestClassifier(),

    xgboost.XGBClassifier(),
]
```

```
X = data.drop(["has_new_comments", "ID"], axis=1)
y = data["has_new_comments"]
```

```
for alg in ALG:
    print(alg.__class__.__name__, end=":")
    print(cross_val_score(alg, X, y, cv=3))
```

Optimização de parâmetros

Exemplo de otimização de parâmetros de *XGBClassifier* usando *GridSearchCV* (sklearn.model_selection)

```
X = data.drop(["has_new_comments", "ID"], axis=1)
y = data["has_new_comments"]

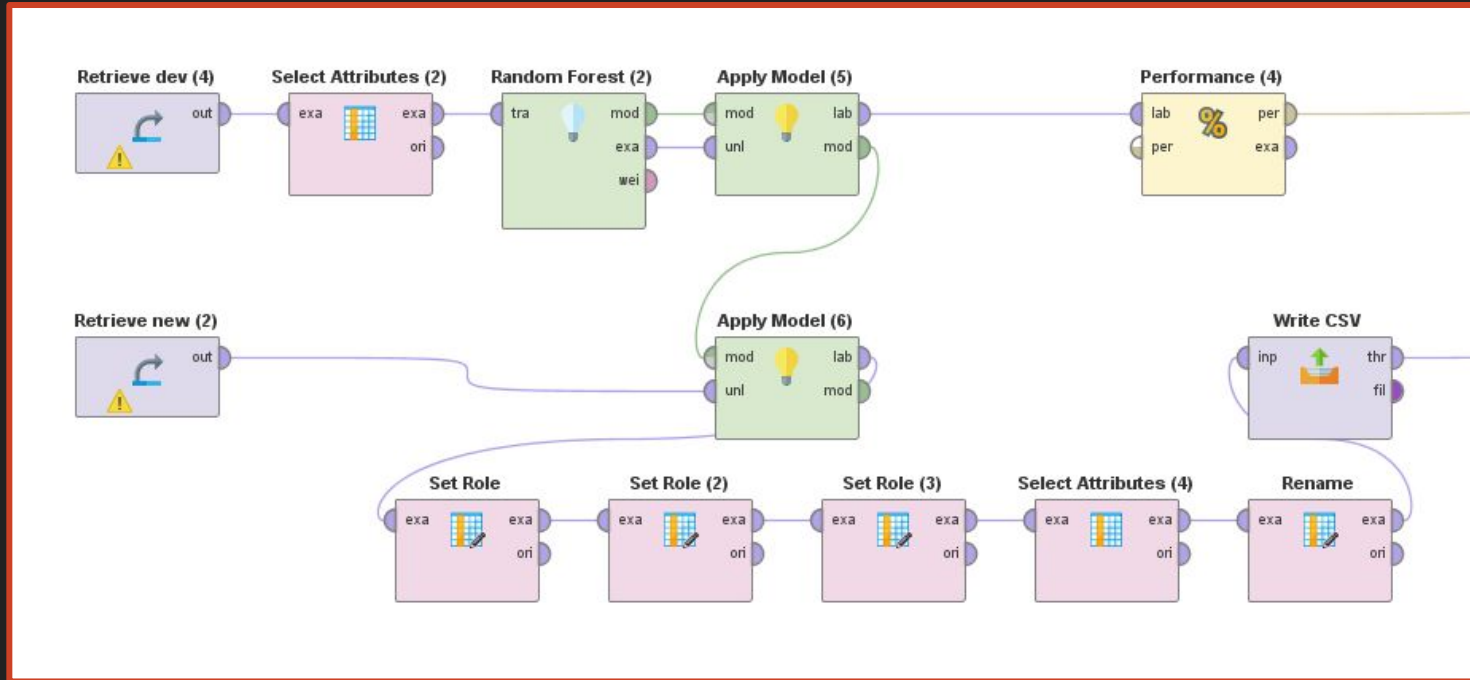
params = {"learning_rate"      : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30],
          "max_depth"         : [ 3, 4, 5, 6, 8, 10, 12, 15],
          "min_child_weight"  : [ 1, 3, 5, 7],
          "gamma"             : [ 0.0, 0.1, 0.2 , 0.3, 0.4],
          "colsample_bytree"  : [ 0.3, 0.4, 0.5 , 0.7] }

xgb = xgboost.XGBClassifier()
clf = GridSearchCV(xgb, params)
clf.fit(X, y)
```

Rapid Miner



Random Forest



Deep Learning

