Aplicando Aprendizado de Máquina em uma competição do Kaggle

Abstract

A proposta deste artigo é demonstrar técnicas de aprendizado de máquina em um problema do mundo real. Para isso analisamos a mais recente competição promovida pela Expedia no site Kaggle. Demonstramos o uso de diferentes técnicas de classificação multi-classe e analisamos a performance de cada uma para o problema proposto.

1. Introdução

Planejamento de férias sempre demanda muito cuidado e na maioria das vezes gera dor de cabeça para o consumidor na hora de escolher o Hotel no qual vai se hospedar. Visando melhorar a experiência do consumidor, diversos tipos de serviços surgiram com o objetivo de facilitar e esclarecer a incógnita que é escolher um hotel em uma cidade na qual não conhece.

Um destes serviços oferecidos é o site Expedia, onde são ofertados preços de hospedagem, aluguel de carros, pacotes de viagens e voos baratos. Recentemente a Expedia lançou no Kaggle um desafio: a partir dos dados de comportamento do usuário no site prever em qual cluster de hotéis está o hotel que o usuário fará sua reserva. Kaggle é uma plataforma fundada em 2010 para competições de modelagem preditiva e analítica onde empresas e pesquisadores postam seus dados para que data miners do mundo inteiro possam competir sobre quem gera os melhores modelos.

Neste desafio eles disponibilizaram dados da navegação do usuário no site, incluindo o que o foi pesquisado, como eles interagiram com o resultado da pesquisa, se a pesquisa resultou em um pacote de viagem, entre outros. Eles também já disponibilizaram os clusters dos hotéis, onde hotéis similares estão agrupados.

Foram disponibilizados 4 arquivos para serem utilizados, $train.csv$, $test.csv$, $destination.csv$ e$sample_submission.cs$v. Onde o primeiro e o terceiro deverão ser utilizados para modelar o problema, o segundo deve ser utilizado para se gerar uma submissão para avaliação no Kaggle, e o último é um template de como deve ser feita essa submissão. Na próxima seção iremos caracterizar o dataset e a forma como os dados foram processados para servirem de entrada para os algoritmos. Depois disso discutiremos as diferentes técnicas de aprendizado de máquina utilizadas. Por fim iremos discutir os resultados e os desafios enfretados.

2. Caracterização do Dataset

A tabela 1 mostra as $features$ disponiveis, além da resposta que deve ser prevista, no caso o cluster do hotel definido pela coluna hotel_cluster.O conjunto de treino inicial dispõe de 36 milhões de linhas que representam os dados de navegação do usuário dentro do site da Expedia. Essa navegação pode resultar em duas ações, a reserva de um hotel ou o clique para ver mais informações referente ao hotel. Essa ação é definida pela coluna is_booking. Para is_booking = 1 a ação foi uma reserva, para is_booking = 0 a ação foi um clique. Para o conjunto de teste, que será avaliado para geração de uma pontuação, todas as ações são ações de reserva. Logo, para diminuir o conjunto de treino e deixar o problema mais tratável, foi gerado um subconjunto utilizando apenas os dados no qual is_booking = 1, com isso o conjunto de treino passou a ter 3 milhões de linhas.

Outros tratamentos no conjunto de dados foram realizados para facilitar a modelagem do problema. Inicialmente foi criado uma nova coluna que mostra o número de dias para os quais o hotel foi reservado. Além disso os dados que representavam datas foram divididos em ano, mês, dia e trimestre, já que os algoritmos de aprendizado não saberiam lidar com o tipo de dado que representam datas.

Antes de iniciarmos com a proposta para tratamento do problema, vamos analisar o balanceamento das classes. A figura 1 mostra a frequência com que cada cluster aparece no conjunto de dados. Podemos observar que embora alguns clusters apareçam mais que os outros, existe, em média, um relativo balanceamento entre eles. Discutiremos agora a metodologia adotada nesse trabalho.

Table 1. Descrição dos dados

Nome da Coluna	Descrição
date_time	Timestamp
site_name	ID do site da Expedia onde o evento ocorreu (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)
posa_continent	ID do continente associado ao site_name
user_location_country	ID do país onde o cliente está localizado
user_location_region	ID da região onde o cliente está localizado
user_location_city	ID da cidade onde o cliente está localizado
orig_destination_distance	Distância física entre o cliente e o hotel no momento da busca
user_id	ID do usuário
is_mobile	1 se o usuário está conectado utilizando um dispositivo móvel, 0 do contrário
is_package	1 se o evento (clique/reserva) foi feito como parte de um pacote, 0 do contrário
channel	ID do canal de marketing
srch_ci	Data de check-in
srch_co	Data de check-out
srch_adults_cnt	O número de adultos especificados para o quarto do hotel
srch_children_cnt	O número de crianças especificdas para o quarto do hotel
srch_rm_cnt	O número de quartos especificados na pesquisa
srch_destination_id	ID do destino onde o hotel se localiza
srch_destination_type_id	Tipo de destino
hotel_continent	Continente do hotel
hotel_country	Hotel country
hotel_market	Mercado do hotel
is_booking	1 se reserva, 0 se clique
cnt	Número de eventos similares dentro da mesma sessão do mesmo número
hotel_cluster	ID do cluster do hotel

3. Metodologia

A metodologia desenvolvida nesse trabalho envolve 5 passos, são eles:

- Pré processamento dos dados e divisão em treino e validação.
- Escolha do algoritmo de aprendizado de máquina a ser utilizado.
- Definição dos melhores parâmetros para o algoritmo escolhido.
- Geração do modelo a partir do algoritmo escolhido.

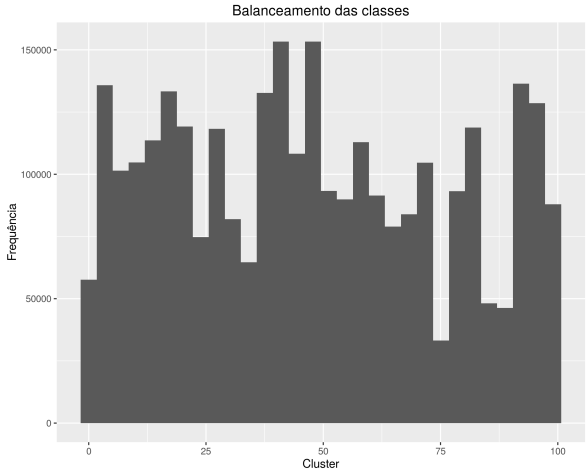

Figure 1. Balanceamento das classes

- Validação do modelo gerado e avaliação crítica.
- Tentativa de melhorar o modelo.

O pré processamento dos dados foi discutido na seção anterior. Devido ao grande número de dados disponíveis para treino foi realizado uma amostragem para a utilização na geração do modelo. Devido ao grande número de dados foi decidido não usar k-fold cross validation para realizar validação, e sim um conjunto de validação.

Sabemos que não existe uma abordagem melhor que a outra quando em aprendizado de máquina, por isso para o tratamento desse problema decidimos experimentar e observar como três abordagens diferentes se comportam. Inicialmente foram realizados experimentos utilizando regressão logística. Depois testamos duas abordagens que estão no estado da arte quando se fala de algoritmos de classificação, são elas: (1) random forest e (2) gradient tree boosting. Nas próximas seções iremos detalhar os experimentos realizados com cada algoritmo, mas antes vamos discutir a métrica de validação que será utilizada.

3.1. Métrica

A métrica de validação é a mesma utilizada pelo Kaggle para avaliação dos resultados. Nesse caso é a mean average precision @ 5 (MAP@5). Essa métrica é representada pela fórmula na figura 2. Nessa fórmula U representa o número

$$MAP@5 = \frac{1}{|U|}\sum_{u=1}^{|U|}\sum_{k=1}^{min(5,n)} P(k)$$

Figure 2. Formula da métrica utilizada.

de eventos de usuários (número de registros avaliados), n é

o número de clusters previstos para um evento e P(k) a precisão no corte k. Suponha U = 2, no qual temos as seguintes previsões e os seguintes valores reais conforme na tabela 2. Para esse caso teríamos $MAP@5 = (4/5 + 1/5)/2 = 0.5$.

Table 2. Valores de possível saída

Id	Cluster Previsto	Cluster Real
1	1 5 9 10 11	5
2	2 3 15 7 99	99

4. Regressão Logística

Nessa seção iremos discutir como foram realizados os experimentos para regressão logística e quais foram os resultados obtidos. Os experimentos foram feitos utilizando três técnicas de regularização: ridge regression, LASSO e elastic net. Para cada um foi variado os valores do parâmetro de penalidade da regularização (lambda) e os resultados foram validados através do conjunto de validação.

4.1. Experimentos

Para a execução dos experimentos foi utilizado a linguagem de programação R, mais especificamente o pacote h2o. Esse pacote é uma API para o R de uma engine open-source escrita em Java que implementa algoritmos de aprendizado de máquina e permite o processamento paralelo e distribuído para grandes volumes de dados. A escolha desse pacote se sustenta pelo fato de estarmos tratando de um conjunto de dados muito grande, e a forma de processamento utilizada auxilia no melhor tratamento do problema.

Inicialmente foram gerados modelos utilizando a regressão logística para cada uma das técnicas de regularização, vale ressaltar que os dados foram normalizados antes de serem utilizados para a geração do modelo. Além disso, os modelos foram utilizados para prever o cluster em duas situações diferentes, na primeira foram considerados todos os 100 clusters possíveis, na segunda os clusters foram limitados pelo srch_destination_id. Sabemos que o problema de previsão multi-classe para 100 classes diferentes é extremamente sujeito a erros, por isso, definimos uma técnica para diminuir o número de classes possíveis de serem previstas. Para isso utilizamos todo o conjunto de treino e agrupamos os clusters pela coluna srch_destination_id. Essa coluna representa o destino que o usuário está pesquisando, logo, é valido assumir que para um determinado destino apenas certos clusters fazem sentido. Dessa forma, quando realizando uma nova previsão os clusters possíveis passam a ser aqueles que já surgiram como resultado para o srch_destination_id do novo registro. Por fim o melhor modelo gerado foi utilizado para gerar uma submissão no Kaggle. Para esse último caso foi utilizado 10% do conjunto inicial (com is_booking = 1) para o treinamento.

4.2. Resultados

Como explicado anteriormente foram gerados vários modelos, utilizando diferentes técnicas de regularização, variando o valor da penalidade (lambda), além disso as previsões foram feitas em duas situações diferentes. Os três gráficos abaixo mostram o resultado para os dados de treino e validação, para cada técnica de regularização quando se varia o valor da penalidade (lambda), e para aos dois contextos definidos.

O gráfico da figura 3 mostra os resultados quando utilizamos $ridgeregression$, o gráfico da figura 4 quando utilizamos o $LASSO$ e o gráfico da figura 5 quando utilizamos a $elasticnet$. No eixo y temos a precisão medida utilizando $MAP@5$ e no eixo x a variação do valor da penalidade (lambda). No título dos gráficos temos a pontuação que o melhor modelo, para aquela técnica de regularização, gerou quando avaliado no conjunto de validação. Para o caso do $elasticnet$, que gerou os melhores resultados em validação, observamos também o resultado da avaliação no Kaggle. Podemos observar resultados parecidos para as três técnicas de regularização, o valor ótimo de lambda, dentre os testados, foi 0.0005, e temos uma performance ligeiramente melhor quando utilizando $elasticnet$.

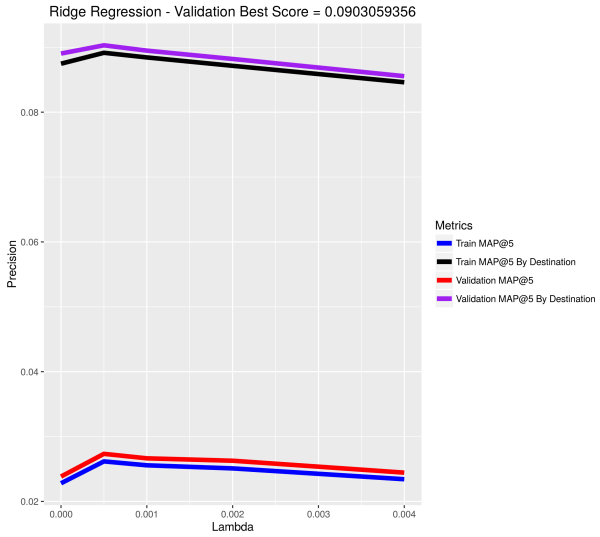

Figure 3. Resultados para ridge regression

5. Random Forest

Nessa seção iremos discutir os experimentos e resultados quando utilizando o algoritmo de random forest. Nesse caso variamos dois parâmetros na tentativa de escolher o melhor modelo, são eles: a profundidade máxima das árvores geradas e o número de features que são amostradas

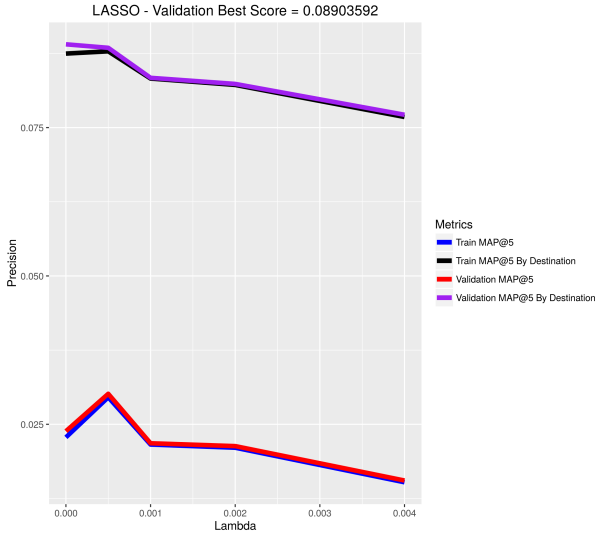

Figure 4. Resultados para lasso

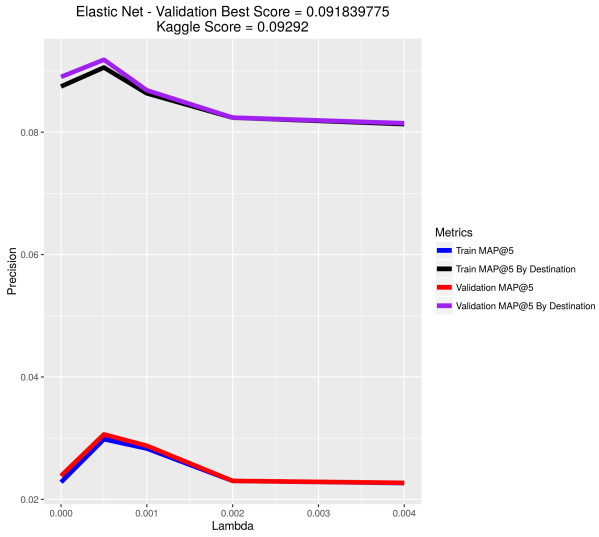

Figure 5. Resultados para elastic net.

como candidatas para divisão em cada nó.

5.1. Experimentos

Para a realização desses experimentos também foi utilizado a implementação do pacote h2o discutido anteriormente. Pelos experimentos realizados na regressão logística percebemos que quando limitamos o número de clusters por srch_destination_id os resultados melhoram significativamente. No entanto esse processo é muito caro computacionalmente. Por esse motivo, e como queremos apenas comparar a performance dos algoritmos, para a escolha do melhor modelo utilizando random forest não foi imposta essa limitação.

Foram utilizados três valores para profundidade máxima (10,20,30), e três valores para o número de features que são amostradas como candidatas para divisão em cada nó (4,5,6). Além disso foi definido um parâmetro fixo que é o número de árvores geradas (500), outros parâmetros poderiam ser variados mas foi decidido utilizar os valores padrão da implementação.

5.2. Resultados

O gráfico da figura 6 mostra os resultados alcançados. Percebemos que o aumento no número de features que são amostradas como candidatas para divisão em cada nó não afeta o resultado significativamente. No entanto o aumento da profundidade máxima das árvores geradas tem um efeito significativo. Notavelmente temos aumento da performance em treino, no entanto a performance em validação diminui, isso indica um clássico caso de overfitting. Também podemos notar uma performance significativamente melhor do que a regressão logística.

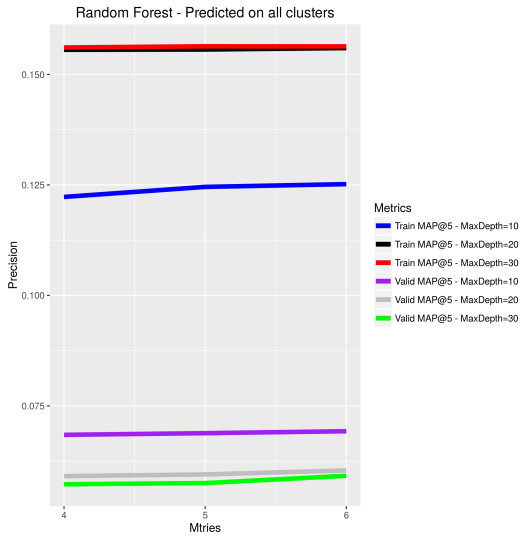

Figure 6. Resultados para random forest

6. Gradient Tree Boosting

Nessa seção iremos discutir os experimentos e resultados quando utilizando o algoritmo de gradient tree bosting. Nesse caso variamos três parâmetros na tentativa de escolher o melhor modelo, são eles: a profundidade máxima das árvores geradas, a amostra percentual dos dados de treino utilizados (quantidade) para crescer cada árvore e a amostra percentual das features utilizadas para crescer cada árvore (quais features utilizar).

6.1. Experimentos

Para a relização desses experimentos foi utilizado o pacote xgboost. Esse pacote implementa máquinas de boosting para árvores e modelos lineares, e vem sendo utilizado pela comunidade do Kaggle há algum tempo, o que se observa é que frequentemente os vencedores das competições utilizam esse pacote. Assim como no caso da random forest não foi imposta a limitação dos clusters por srch_destination_id.

Foram utilizados dois valores para profundidade máxima (4,6), três valores para amostra percentual dos dados de treino utilizados(1,0.75,0.5) e quatro valores para amostra percentual das features utilizadas (0.4, 0.6, 0.8, 1) . Além disso foram definidos dois parâmetro fixos: o número de árvores geradas (100) e a taxa de aprendizado (0.03). Outros parâmetros poderiam ser variados mas foi decidido utilizar os valores padrão da implementação.

6.2. Resultados

A figura 7 mostra os resultados obtidos. Percebeu-se que a amostragem percentual dos dados de treino não gera grandes diferenças nos resultados, por isso o gráfico gerado manteve fixo a profundidade máxima das árvores e variou a amostra percentual de quais features utilizar para crescer cada árvore.

Percebemos algo semelhante com o que ocorreu com a random forest. O aumento na profundidade máxima gera overfitting. Além disso também percebemos que os resultados são ótimos quando a amostra percentual de quais features utilizar é 0.6.

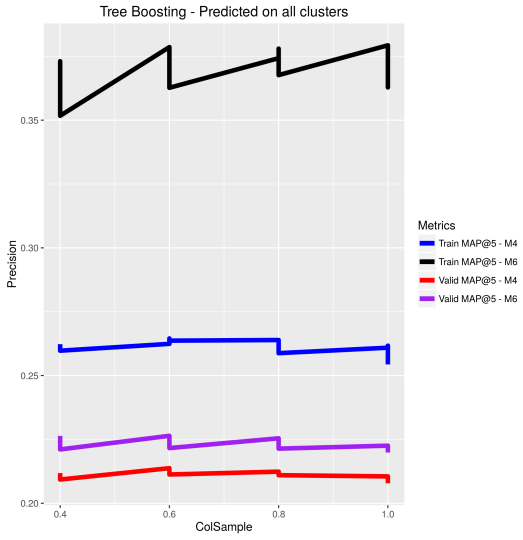

Figure 7. Resultados para gradient tree boosting

7. Discussão

Percebemos que o algoritmo de gradient tree boosting é significativamente melhor do que os outros. Com isso em mente tentamos observar se era possível melhorar o resultado aumentando a quantidade de dados de treino. O gráfico da figura 8 mostra os valores alcançados variando a quantidade de dados de treino para esse algoritmo. Percebemos que o aumento da quantidade de dados de treino melhora os resultados, diminuindo o overfitting e aumentando a precisão em validação, o que mostra que o algoritmo está sofrendo de alta variância.

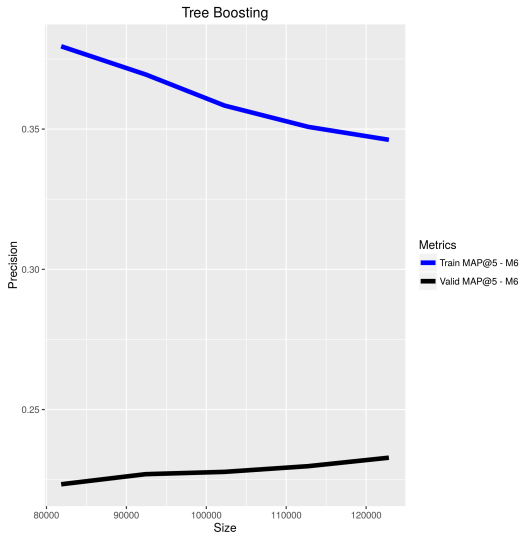

Figure 8. Resultados para gradient tree boosting variando quantidade de dados

O valor de referência para MAP@5 em uma escolha aleatória de clusters é em torno de 2.3%. Para cada algoritmo, testado, selecionamos os melhor modelo (melhores parâmetros) e utilizado a tática de limitar os clusters por srch_destination_id geramos os resultados mostrados na tabela 3. No entanto quando enviamos para o Kaggle, o melhor resultado gerado foi uma precisão de 11%. Na próxima seção discutimos o porque disso e os desafios encontrados na exploração dessa competição.

Table 3. Resultados alcançados

Model	**Train MAP**	**Valid MAP**
Logistic Regression (h2o)	0.0918	0.0906
Random Forest (h2o)	0.1856	0.1410
GTB (xgboost)	0.4127	0.2647

7.1. Desafios

Como discutido anteriormente o melhor resultado gerado ao enviarmos uma resposta ao Kaggle foi de 11%. No entanto, nos testes de validação obtivemos resultados muito melhores, porque isso ocorre?

Na seção anterior também mostramos como o aumento na quantidade de dados de treino melhorou os resultados gerados pelo algoritmo de gradient tree boosting. Vamos pensar em números, naquele caso variamos a quantidade de dados de treino entre 80000 e 120000 e prevemos, para validação, em aproximadamente 40000 exemplos. Esses valores por si só são grandes e demandam um certo poder computacional para serem gerados. No entanto quando pensamos na competição em si temos disponíveis 36 milhões de exemplos para treino e devemos prever em 2.6 milhões de dados para sermos avaliados pelo Kaggle. No entanto o modelo que usamos para gerar a submissão para o Kaggle foi treinado em 'apenas' 100 mil exemplos.

Esses números mostram o maior desafio dessa competição, como lidar com a quantidade de dados disponível. Embora essa quantidade seja razoável quando pensamos no mundo real, nós não tínhamos poder computacional suficiente para ser competitivo.

8. Conclusão e Trabalhos Futuros

Com essa competição foi possível estudar diferentes técnicas de aprendizado de máquina e observar como elas se comportam. Por exemplo, observamos que a regressão logística gera resultados muito ruins para o problema explorado, isso ocorre porque a relação entre as features e a resposta não é linear. Podemos confirmar isso observando os valores dos coeficientes de correlação de Pearson entre as $features$ e a resposta. Esse coeficiente mostra se duas variáveis são linearmente dependentes entre si, valores próximos de 0 indicam que não existe relação linear, no entanto pode existir uma relação que não seja linear. O gráfico da figura 9 mostra o valor do coeficiente de correlação de Pearson no eixo y e um índice que representa cada $feature$ no eixo x. Podemos observar que todas apresentam valores próximos a 0.

Passado o ponto que observamos que a relação não é linear entre as $features$, conseguimos resultados muito melhores utilizados técnicas sensíveis a não linearidade. No entanto, embora os resultados tenham sido relativamente satisfatórios, fomos impedidos de avançar mais devido a falta de poder computacional para tratar os dados disponíveis.

Como trabalhos futuros podemos focar no algoritmo de gradient tree boosting e tentar melhorar a performance utilizando técnicas de computação distribuída. Além disso outras técnicas de classificação podem ser testadas como

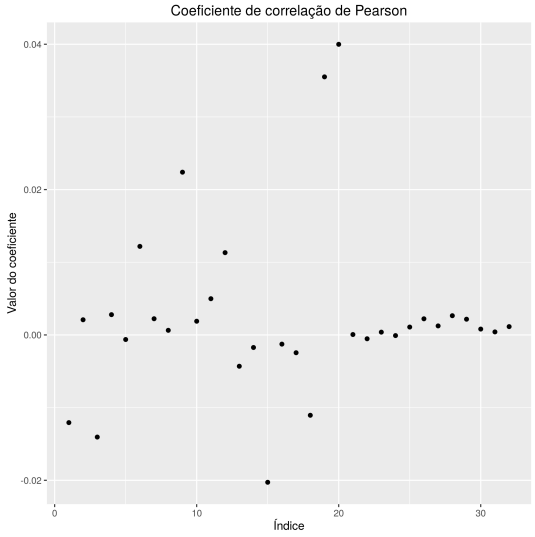

Figure 9. Coeficiente de correlação de Pearson

support vector machines, naive bayes, redes neurais, entre outras.

Referências

Gareth James, Daniela Witten, Trevor Hastie and Tibshirani, Robert. An introduction to statistical learning. Srpinger, 2009.

Spencer Aiello, Tom Kraljevic and Maj, Petr. h2o package, 2016. URL https://cran.r-project.org/web/packages/h2o/h2o.pdf.

Tianqi Chen, Tong He and Benesty, Michael. xgboost package, 2015. URL https://cran.r-project.org/web/packages/h2o/h2o.pdf.