Classificação de tipos de produtos utilizando reviews

Introdução

Neste relatório, serão abordadas diferentes etapas de análise de texto e treinamento de modelos, visando a classificação de produtos com base em comentários de clientes. A análise envolve desde o pré-processamento dos dados textuais até a aplicação de modelos de classificação, considerando abordagens variadas e técnicas específicas para cada etapa.

Carregamento dos pacotes

Primeiramente, foram carregados os pacotes necessários para a análise. O pacote *tm* é utilizado para realizar tarefas de mineração de texto, como pré-processamento e criação de Document-Term Matrices (DTM). Por sua vez, o pacote *caret* é uma ferramenta poderosa para treinamento e avaliação de modelos de machine learning.

```
# Carregar pacotes necessários
library(tm)
library(caret)
```

Extração e preparação dos dados

Nesta seção, foi realizada a extração e preparação das informações contidas nos comentários dos produtos. Inicialmente, os dados foram carregados a partir dos arquivos CSV "train.csv" e "test_unlabeled.csv". Além disso, os rótulos das classes foram ajustados, substituindo espaços por sublinhados para melhor tratamento posterior.

```
# Ler o arquivo CSV para um dataframe
train <- read.csv('train.csv', header = T)
test_unlabeled <- read.csv('test_unlabeled.csv', header = T)
train$Product <- gsub(" ", "_", train$Product)
```

Em seguida, um VCorpus foi criado para armazenar os textos dos comentários. A técnica TF-IDF (Term Frequency-Inverse Document Frequency) foi utilizada para a criação da Document-Term Matrix (DTM). Durante o processo, foram aplicadas transformações como a conversão para letras minúsculas, remoção de números, pontuações e stopwords, além da aplicação do peso TF-IDF para ponderação das frequências dos termos.

```
# Criar um VCorpus
corp <- VCorpus(VectorSource(train$Text))

# Criar DTM com TF-IDF
dtm <- DocumentTermMatrix(corp,
```

```
                          list(tolower = TRUE,
                               stemming = FALSE,
                               removeNumbers = TRUE,
                               removePunctuation = TRUE,
                               stopwords = stopwords("en"),
                               weighting = weightTfIdf,  # Usar TF-IDF
                               bounds = list(global = c(3, Inf))))
```

Após a criação da DTM, termos pouco frequentes foram removidos utilizando o critério de que os termos devem aparecer em pelo menos 5% dos documentos para serem considerados relevantes. Isso é útil para reduzir a dimensionalidade da DTM, concentrando-se nos termos mais relevantes e eliminando termos que ocorrem com baixíssima frequência.

```
dtm <- removeSparseTerms(dtm, 0.95)
```

Finalmente, a DTM foi convertida em um dataframe, e os rótulos das classes foram adicionados como uma coluna. Esses passos constituem a preparação inicial dos dados para a etapa de treinamento e avaliação dos modelos de classificação.

```
# Converter a DTM para um dataframe e adicionar rótulos
dtm_data <- as.data.frame(as.matrix(dtm))
dtm_data$Product <- train$Product
```

Ajuste dos modelos

Nesta etapa, diversos modelos de classificação foram treinados e avaliados utilizando os dados previamente preparados. Cada modelo foi ajustado para realizar a tarefa de classificação dos produtos com base nos comentários associados a eles, utilizando como dados de entrada a DTM.

XGBoost

Para o modelo XGBoost, foi configurado um controle de treinamento utilizando validação cruzada com 3 folds. Diversos hiperparâmetros foram ajustados, incluindo o número de rounds (nrounds), a profundidade máxima da árvore (max_depth), a taxa de aprendizado (eta), o parâmetro de regularização (gamma), entre outros. O modelo foi ajustado utilizando o método "xgbTree" e a métrica de avaliação escolhida foi a acurácia.

```
# Configurar o controle de treinamento
ctrl <- trainControl(method = "cv", number = 3,
                     verboseIter = FALSE, classProbs = TRUE)

# Definir a grade de parâmetros
paramGrid <- expand.grid(nrounds = 150, max_depth = 1,
                         eta = 0.3, gamma = 0, colsample_bytree = 0.6,
                         min_child_weight = 1, subsample = 1)

# Treinar o modelo xgboost
set.seed(2023)
model_xgb <- train(Product ~ ., data = dtm_data,
                   method = "xgbTree",
```

```
                    trControl = ctrl,
                    metric = "Accuracy",
                    tuneGrid = paramGrid)
```

AdaBoost

No caso do modelo AdaBoost, também foi utilizado um controle de treinamento com validação cruzada de 3 folds. O modelo foi ajustado através do método "AdaBoost.M1". Apenas um conjunto de hiperparâmetros finais foi considerado, incluindo o número máximo de classificadores (mfinal), a profundidade máxima (maxdepth) e o coeficiente de aprendizado (coeflearn). A métrica de avaliação escolhida também foi a acurácia.

```
# Configurar o controle de treinamento
ctrl <- trainControl(method = "cv", number = 3,
                     verboseIter = FALSE, classProbs = TRUE)

# Definir uma grade de parâmetros com a única combinação de hiperparâmetros finais
paramGrid <- expand.grid(mfinal = 50, maxdepth = 3, coeflearn = "Zhu")

# Treinar o modelo AdaBoost
set.seed(2023)
model_adaboost <- train(Product ~ ., data = dtm_data,
                        method = "AdaBoost.M1",
                        trControl = ctrl,
                        metric = "Accuracy",
                        tuneGrid = paramGrid)
```

Regressão Logística Multiclasse

O modelo de regressão logística multiclasse foi ajustado utilizando o método "multinom". Novamente, um controle de treinamento com validação cruzada de 3 folds foi configurado. Hiperparâmetros relacionados ao processo de otimização foram especificados, como o parâmetro de decaimento (decay) e o limite máximo de pesos (MaxNWts). A métrica de avaliação considerada foi a acurácia.

```
# Configurar o controle de treinamento
ctrl <- trainControl(method = "cv", number = 3,
                     summaryFunction = multiClassSummary,
                     classProbs = TRUE, verboseIter = FALSE)

# Treinar o modelo de regressão logística multiclasse
model_logreg <- train(Product ~ ., data = dtm_data,
                      method = "multinom",
                      trControl = ctrl,
                      tuneGrid = expand.grid(decay = 0.1),
                      MaxNWts = 100000)
```

Seleção do melhor modelo

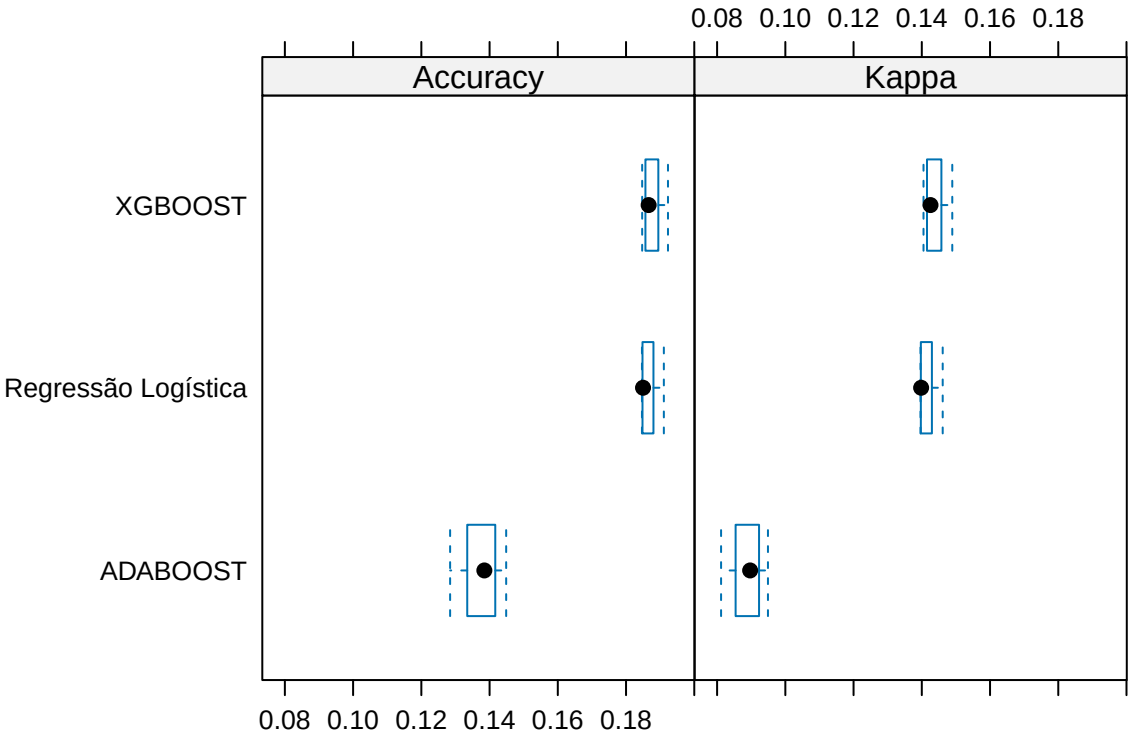
Após o treinamento e avaliação dos modelos, seleciona-se o melhor modelo com base nas métricas de desempenho obtidas. Para isso, os resultados dos modelos previamente treinados em foram organizados em

uma lista denominada “models”. Em seguida, a função “resamples” foi utilizada para calcular as métricas resumidas de cada modelo, como a acurácia. Além disso, um gráfico boxplot mostra o desempenho dos modelos na validação cruzada.

```
# Juntar os resultados dos modelos em uma lista
models <- list(model_xgb,
               model_adaboost,
               model_logreg)

# Calcular as métricas resumidas
res <- resamples(models, modelNames = c('XGBOOST', 'ADABOOST',
                                        'Regressão Logística'))

# Visualizar as métricas resumidas (por exemplo, acurácia)
bwplot(res)
```



Com base na análise das métricas resumidas, foi selecionado o melhor modelo, que apresentou o desempenho mais promissor para a tarefa em questão, considerando a métrica de acurácia (XGBoost).

```
# Definir o melhor modelo selecionado
best_model <- model_xgb
```

Em uma tarefa desafiadora como essa, onde há milhares de amostras e possibilidades de composição dos dados e ajuste dos modelos, um dos passos fundamentais é a exploração de diversos modelos de machine learning. Permitindo assim avaliar diferentes abordagens e algoritmos, visando encontrar aquele que melhor se adapta aos dados e maximiza o desempenho de classificação.

Neste trabalho, foram treinados diversos modelos diferentes, abrangendo uma variedade de algoritmos e configurações de hiperparâmetros. Os modelos foram ajustados utilizando técnicas como XGBoost, AdaBoost e Regressão Logística Multiclasse. Para cada modelo, diferentes combinações de hiperparâmetros foram exploradas, resultando em uma grande quantidade de modelos (mais de 10), a fim de encontrar a configuração que proporcionasse o melhor resultado.

A seleção das configurações de hiperparâmetros de cada algoritmo utilizado foi embasada nos desempenhos observados durante o treinamento a avaliação dos modelos (grid search). Não foram, portanto, adicionados ao relatório tão somente para ganho de tempo de treinamento, uma vez que o treinamento desses modelos demanda muito tempo.

Classificação de novos dados

Para avaliar o desempenho dos modelos treinados em dados não classificados, uma etapa importante é aplicar esses modelos a novos conjuntos de dados. Isso permite mensurar como os modelos se comportam em dados que não foram utilizados durante o treinamento, oferecendo uma avaliação realista da capacidade de generalização dos modelos.

Assim, o melhor o modelo encontrado foi utilizado para a classificação dos dados de teste.

Um VCorpus foi criado a partir dos comentários presentes nos dados não classificados. O vocabulário extraído da Document-Term Matrix (DTM) dos dados de treinamento foi utilizado para criar uma DTM com a técnica TF-IDF para os dados não rotulados. Isso garante a coerência na representação dos termos entre os conjuntos de treinamento e teste.

As classes preditas pelo modelo foram transformadas, substituindo os underlines (_) por espaços em branco nos nomes das classes, a fim de tornar a apresentação mais legível e compreensível e coerente com a forma inicial com que foram apresentadas. Um novo dataframe foi criado, contendo as colunas "Text" (com os comentários) e a coluna de classes preditas, agora com nomes mais claros.

Por fim, as classificações foram salvas em um arquivo CSV chamado "predictions.csv", com as colunas de cabeçalho e aspas delimitadoras removidas, conforme orientação inicial no trabalho.

```r
# Criar um VCorpus
corp_unlabeled <- VCorpus(VectorSource(test_unlabeled$Text))

# Extrair o vocabulário da DTM dos dados de treinamento
vocab <- colnames(dtm_data)

# Criar DTM com TF-IDF para dados não rotulados usando o mesmo vocabulário
dtm_unlabeled <- DocumentTermMatrix(corp_unlabeled,
                                    control = list(dictionary = vocab,
                                                   tolower = TRUE,
                                                   stemming = FALSE,
                                                   removeNumbers = TRUE,
                                                   removePunctuation = TRUE,
                                                   stopwords = stopwords("en"),
                                                   weighting = weightTfIdf,
                                                   bounds = list(global = c(3, Inf))))

# Converter a DTM para um dataframe
dtm_data_unlabeled <- as.data.frame(as.matrix(dtm_unlabeled))

# Fazer previsões
predictions_unlabeled <- predict(best_model, newdata = dtm_data_unlabeled)
```

```r
# Substituir os underlines (_) por espaços em branco nos nomes das classes
predicted_classes <- gsub("_", " ", as.character(predictions_unlabeled))

# Criar um novo dataframe com as colunas Text e Predited_Class
output_data <- data.frame(Text = test_unlabeled$Text, Predicted_Class = predicted_classes)
names(output_data) <- c('Text', '')

# Salvar o dataframe em um arquivo CSV
write.csv(output_data, "predictions.csv", row.names = FALSE, quote = FALSE)
```

Função de perda

Dada a seguinte função:

$\text{Loss} = \sum_c \frac{1}{P(Y=c)} \cdot I(Y \neq g(X), Y = c)$

Essa função de perda é uma forma de avaliar o desempenho dos modelos de classificação considerando não apenas a acurácia geral, mas também a penalização por erros específicos.

$\sum_c$ é um somatório sobre todas as classes c

$\sum_c \frac{1}{P(Y=c)}$ é um termo depeso inversamente proporcional à probabilidade da classe c nos dados. Isso é útil para tratar classes desbalanceadas, dando mais peso às classes menos frequentes.

$I(Y \neq g(X), Y = c)$ é uma função indicadora que é igual a 1 se a classe verdadeira Y é c e a classe prevista $g(x)$ é diferente de c. Assim, ela é 1 quando o modelo faz uma previsão incorreta para a classe c.

Então, a função de perda é uma soma ponderada dos erros de classificação. Para cada classe c, ela verifica se a classe verdadeira era c e a classe prevista era diferente de c. Se isso acontecer, ela adiciona um termo de penalidade à função de perda, ponderado pelo inverso da probabilidade da classe c nos dados.

Isso pode ser útil em cenários onde as classes estão desbalanceadas e se deseja dar mais importância a erros cometidos nas classes menos frequentes.

Conclusão

A análise realizada revelou desafios significativos na tarefa de classificação de produtos com base em comentários de clientes. Mesmo após explorar uma variedade de modelos de machine learning e otimizar seus hiperparâmetros, observou-se que o melhor modelo alcançou uma acurácia apenas de aproximadamente 20%. Esse resultado indica que a tarefa é altamente complexa e desafiadora, provavelmente devido a uma combinação de fatores, como a natureza diversificada dos comentários e o grande número de classes.

Um dos principais desafios encontrados foi o treinamento dos modelos. Dada a grande quantidade de classes, a tarefa de ajustar os hiperparâmetros e treinar cada modelo se mostrou extremamente demorada, demandando muito tempo de processamento em um computador pessoal padrão. O número elevado de classes também dificulta a tarefa de identificar padrões relevantes nos dados, o que impacta diretamente na capacidade de generalização dos modelos.

Entretanto, embora os resultados tenham sido abaixo das expectativas, essa situação é relativamente comum em problemas de classificação de textos com muitas classes. A complexidade e a variabilidade dos dados podem desafiar a capacidade dos modelos de generalizar com eficácia. Portanto, soluções mais avançadas, como o uso de redes neurais mais complexas (deep learning) ou técnicas de pré-processamento mais sofisticadas, podem ser necessárias para melhorar o desempenho.

No entanto, apesar dos resultados modestos, este projeto serviu como uma oportunidade para explorar a complexidade dos problemas de classificação de textos e compreender as dificuldades envolvidas na tarefa.

Além disso, a experiência de ajustar e avaliar diversos modelos permitiu uma compreensão mais profunda das nuances e desafios da aplicação de machine learning em cenários práticos do mundo real.