UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
CENTRO DE CIÊNCIAS FÍSICAS E MATEMÁTICAS
DEPARTAMENTO DE MATEMÁTICA
LICENCIATURA EM MATEMÁTICA
AUGUSTO PACHECO DA ROCHA

PROJETO DE FILTRO DE SPAM ATRAVÉS DO ALGORITMO DE NAIVE BAYES

FLORIANÓPOLIS, 2024

RESUMO DO PROJETO

Usando o arquivo "Bayesian_Inference.ipynb" foi elaborado esse projeto do filtro de SPAM. Neste arquivo foi feito um resumo de como o projeto foi feito, assim como texto respondendo perguntas complementares solicitadas na atividade da criação do filtro de SPAM usando Naive Bayes.

PASSO A PASSO CRIAÇÃO DO MODELO

Seguindo o arquivo "Bayesian_Inference.ipynb", nos parágrafos posteriores será feito um breve resumo do que foi feito em cada passo.

Step 0: Introduction to the Naive Bayes Theorem

Lido o texto que faz uma breve introdução ao teorema de Naive Bayes.

Step 1.1: Understanding our dataset

Aqui foi baixado o dataset que será usado no projeto. Todos os códigos foram escritos no Replit através da linguagem Python.

Step 1.2: Data Preprocessing

Aqui foi importado a biblioteca pandas e o dataset num arquivo do replit. A sintaxe dos códigos utilizados pode ser vista no arquivo "step_1.2_data_preprocessing.py".

Step 2.1: Bag of words

Lido o texto que faz uma breve explicação do que é "bag of words"

Step 2.2: Implementing Bag of Words from scratch

Aqui foi implementado o bag of words de uma maneira mais manual para se compreender melhor o processo. Todo feito nesta seção está no arquivo "step_2.2_bag_of_words.py".

Step 2.3: Implementing Bag of Words in scikit-learn

Depois de feito o bag of word de uma maneira mais manual agora se refez o processo usando recurso do scikit-learn. Tudo feito nesta seção está disponível no arquivo "step_2.3_bag_of_words.py".

Step 3.1: Training and testing sets

Aqui foi dividido o dataset em um conjunto para treino e outro para teste. Tudo feito nesta seção está no arquivo "step_3.1_training_and_testing.py"

Step 3.2: Applying Bag of Words processing to our dataset

Aqui se dividiu o dataset em conjunto de treino e conjunto de teste, depois se aplicou o bag of words. Tudo feito nesta seção está no arquivo “step_3.2_applying_bow.py”.

Step 4.1: Bayes Theorem implementation from scratch

Lido o texto que explica como o teorema de Bayes funciona tomando como exemplo um exame para diabetes.

Step 4.2: Naive Bayes implementation from scratch

Lido o texto que explica como o teorema de Naive Bayes funciona usando como exemplo a probabilidade de certos candidatos falarem certas palavras se fizerem um discurso.

Step 5: Naive Bayes implementation using scikit-learn

Aqui foi implementado o Naive Bayes usando o dataset já mencionado para criar um filtro de spam. Tudo feito nesta seção está no arquivo “Step_5_and_6_NB.py”.

Step 6: Evaluating our model

Aqui foi calculado a acurácia, precisão, recall e f1 score do modelo que são:
Acurácia: 0.99
Precisão: 0.99
Recall: 0.94
F1-score: 0.96
Tudo feito nesta seção está disponível no arquivo “Step_5_and_6_NB.py”

Step 7: Conclusion

Lido o texto acerca das vantagens de se usar o Naive Bayes.

O QUE É BAG OF WORDS?

Numa tradução literal podemos dizer que “bag of words” é “saco de palavras”. A ideia por trás dessa frase é transformar texto em dados numéricos através de uma tabela de frequência. Por exemplos tomemos essa três mensagens abaixo como nosso dataset:

1. "Eu adoro pizza"
2. "Pizza é deliciosa"
3. "Eu adoro comida deliciosa"

Nosso saco (bag) conteria as palavras [‘eu’, ‘adoro’, ‘pizza’, ‘e’, ‘deliciosa’, ‘comida’], importante fazer um tratamento nos dados antes para retirar letras maiúsculas e pontuação. Esse tratamento é importante pois uma mesma palavra pode ser entendida como uma palavra diferente. Exemplo ‘win’, ‘Win’, ‘WIN’, ‘Win!’,... Embora seja a mesma palavra sem esse tratamento pode ser entendida como palavras diferentes o que pode gerar imprecisão nos modelos criados a partir desses dados.

Dado o esclarecimento da importância do tratamento dos dados, após isso feito basta montar uma tabela ou matriz de frequência em que a linha é frase e cada coluna as palavras do saco. No nosso exemplo a matriz seria dessa forma:

Palavras / Frase	Eu	adoro	pizza	e	deliciosa	comida
frase 1	1	1	1	0	0	0
frase 2	0	0	1	1	1	0
frase 3	1	1	0	0	1	1

Ou seja, conseguimos transformar texto em uma forma numérica. Uma possível aplicação é que está sendo feito nesse projeto de filtro de SPAM.

DIFERENÇA ENTRE ESPECIFICIDADE E SENSITIVIDADE

Dizemos que a especificidade é a razão entre os verdadeiros negativos (VN) e a soma dos verdadeiros negativos (VN) com falsos positivos (FP). Podemos dizer então que a especificidade responde a pergunta, *"Dentre os casos que realmente são negativos, quantos o modelo conseguiu identificar corretamente?"* Um exemplo de quando a especificidade é mais importante que a sensibilidade são em questões de segurança pois falsos positivos podem gerar pânico e transtornos logísticos mesmo que isso signifique "deixar passar" alguns casos positivos como por exemplo objetos cortantes em uma bolsa.

Já a sensibilidade é a razão entre verdadeiro positivismo (VP) e a soma dos verdadeiros positivos (VP) com falsos negativos (FN). Podemos dizer então que a sensibilidade responde a pergunta *"Dentre os casos que realmente são positivos, quantos o modelo conseguiu identificar corretamente?"* Um exemplo de quando é a sensibilidade é importante que especificidade são nas questões de saúde como detectar uma doença é menos danoso um falso positivo do que um falso negativo, já que o atraso no tratamento pode desencadear complicações maiores ou até mesmo o óbito.

DECISION TREE VS. NAIVE BAYES

Foi escolhido o algoritmo de Decision Tree para ser comparado ao Naive Bayes. Toda a implementação desse modelo está no arquivo "Spam_DT.py". Abaixo segue um resumo geral dos dois modelos.

Dados do Naive Bayes:

Acurácia: 0.99
Precisão: 0.99
Recall: 0.94
F1-score: 0.96

Dados da Árvore de decisão:

Acurácia: 0.97
Precisão: 0.88
Recall: 0.89
F1-score: 0.89

Ao olhar os dados apresentados acima podemos ver que o Naive Bayes pontuou melhor em todos os indicadores, vale ressaltar que a acurácia foi muito parecida apenas um ponto percentual de diferença, já nos outro indicadores houve uma diferença significativa. Dado o apresentado o Naive Bayes se mostrou mais eficiente para um filtro de spam do que decision tree. No entanto, se usássemos o decision tree seria mais fácil entender porque o modelo escolheu classificar o email como SPAM. Ou seja, para ser apenas um filtro de SPAM o Naive Bayes tem melhor resultado, mas se o objetivo é entender quais palavras aumentam a chance de um email ser classificado como spam o modelo de decision tree seria mais interessante.