Projeto Prático 1
Operação Clarividência

Adlla Katarine Aragão Cruz Passos, Daniel Alves Costa, Ramon de Cerqueira Silva

1

1. Descrição dos métodos propostos e sistemas desenvolvidos

1.1. Sistema de Busca de Personagens por Similaridade

A primeira parte do projeto consiste em um sistema de busca de personagens por similaridade, ou seja, o produto deverá disponibilizar a escolha de um personagem, uma distância para o cálculo de similaridade entre personagens (sendo que pelo menos três devem estar disponíveis) e a escolha de quantas *features* o usuário desejar.

Houve uma supervisão sobre toda a base de dados em busca de valores faltantes e/ou *NaN* nos atributos, porém todas as informações estavam completas.

1.2. Predição de super-poderes

A segunda parte do projeto apresenta uma avaliação experimental sobre o uso de Árvore de Decisão, método de aprendizagem de máquina, para prever características ou super-poderes. Utilizando ambas as base de dados fornecidas, ela determinaria se um herói tem um devido poder ou não, levando em conta o restante de suas características e poderes.

Dentre eles, devemos prever cinco atributos, Alignment, Flight, Super Strenght, Accelerated Healing, e Teleporting (atributo extra). E para um bom resultado, deve-se construir um preditor ou classificador com a melhor qualidade, que será avaliada por métodos experimentais. Assim, escolhemos a linguagem Python, a mais popular no meio da Ciência de Dados, e a IDE Spyder, facilitando os calculos estatisticos e visualização dos dados.

2. Algoritmos e Ferramentas Utilizadas

Inicialmente, para a manipulação das base de dados, foi utilizado a biblioteca *Pandas*, que fornece ferramentas de análise de dados e estruturas de dados de alta performance e fáceis de usar. Ela cria um DataFrame com arquivos *.csv*, por exemplo, como foi o caso do projeto. E a partir disso, foi possível trabalhar facilmente com os dados e fazer o uso de outras bibliotecas do *Python*.[Pandas]

Na primeira parte do projeto (Sistema de Busca de Personagens por Similaridade), sendo necessário o uso de métodos para cálculo de distância, foi escolhida a biblioteca *Scipy*. Ela contém várias ferramentas que correspondem a diferentes aplicações, tais como interpolação, integração, otimização, processamento de imagens, estatísticas, funções especiais, etc. Para esse problema foi usada a sua classe *distance* que trabalha com atributos booleanos, representado por toda a base de dados de super-poderes.[Scipy]

A interface foi feita utilizando ferramentas do *Qt*, com o ramo voltado para a criação de aplicações em *Python*, o *PyQt* em sua versão 5. Com ele foi possível a criação e manipulação de itens aplicados nas telas da interface. Assim, obtendo a possibilidade de uma visualização para o sistema de busca de super-heróis semelhantes.[Documentation]

A biblioteca *Numpy* permite trabalhar com arranjos, vetores e matrizes de N dimensões de forma eficiente. Foi usada durante o pré-processamento dos dados, para o tratamento de valores *NaN*.[Numpy]

Para a maior parte do desenvolvimento da segunda parte do projeto (Predição de super-poderes), foi utilizado uma biblioteca de aprendizagem de máquina, *Sklearn*[scikit learn]. Nela contém vários algoritmos de classificação, regressão e agrupamento. Dela, usamos algoritmos para o pré-processamento, árvore de decisão, avaliação do modelo de predição, métodos de distancias para similaridade, entre outros.

Para avaliação da qualidade do modelo de predição, houve o uso da curva ”Receiver Operating Characteristic”, ou simplesmente curva ROC, é uma representação gráfica da sensibilidade ou verdadeiros

positivos, calculando a taxa positiva verdadeira, assim como a taxa de falsos positivos gerando um gráfico que pode ser visualizada pela biblioteca **matplolib.pyplot**.

3. Descrição dos experimentos realizados e seus resultados

Como qualquer experimento físico ou químico, eles contém etapas e mais etapas antes de uma conclusão, na ciência de dados não é diferente. Para criar um classificador ou modelo de classificação precisamos coletar os dados, limpar os dados, tratar, modificar, treinar, testar e por fim, validar, como uma reação química, caso a reação ou modelo de classificação não contenha a qualidade ideal, repete-se o processo até a qualidade estiver de acordo com seus desejos. Cada atributo que será previsto tem seu pre-processamento, usar o mesmo tratamento para atributos diferentes podem influenciar diretamente no desempenho, havendo a divisão de cinco arquivos cada um para com sua devida previsão. Assim, foi subdivido cada parte experimental em três tópicos: Pre-Processamento, Treinamento e Validação.

3.1. Pre-processamento

Antes de qualquer treinamento com aprendizagem de máquina, os dados são limpos, tratados e transformados. A primeira observação quando olhar uma base de dados é verificar se ele contém valores, nulos, vagos, negativos ou repetitivos. Assim, observou-se uma quantidade de heróis duplicados, que foram excluídos, ou com o mesmo nome, onde foram renomeados com um numero para diferenciar.

Depois, houve o agrupamento das bases de dados, heróis e superpoderes, com o metodo **merge** que gerou alguns problemas. Um desses problemas foi a quantidade de nomes de heróis de uma base serem maior que a outra. Para isso fez-se uma mescla dos dados de heróis com os mesmos nomes, e exclusão dos heróis que não estavam contidos em ambas base, não havendo características ou poderes. Agrupamentos podem ajudar na predição, pois diminui-se a quantidade de classes a serem previstas.

Com isso, passou-se a tratar os atributos numerais, *Height* e *Weight*, que continham valores negativos, tratando-os com a media geral do atributo, e apos isso o agrupamento de intervalos, simplificando os atributos. Para *Height* dividimos em 0 e 1, respectivamente baixo e alto, e *Weight* em 0,1,2, respectivamente leve, médio e pesado.

Os atributos categóricos *Gender*, *Eye color*, *Race*, *Hair color* e *Skin color*, foram agrupados, substituindo valores '-', pela negação do atributo (no-color, no-race). O atributo Publisher foi dividido em Marvel Comics, por caracterizar mais que 50% dos herois, e Outhers, apenas em algumas bases. No entanto para um base ser bem treinada, transformamos os atributos categóricos em numéricos, usando o método **LabelEncoder**.O restante dos valores nulos e contidos com '-' foram todos atribuídos pela classe mais frequente do atributo.

E por fim, a exclusão dos atributos a serem previstos, **Alignment**, **Accelerated Healing**,**Flight**, **Super Strenght**, **Teleportation**, transformados em **classe**.

3.2. Treinamento

Com os dados pre-processados, agora está na hora de treina-los. Existe varias maneiras de treinar os dados, mas até aqui usou-se o método simples *train_test_split*, para separar os dados aleatoriamente entre teste e treinamento, utilizando 30% para teste e 70% para treinamento. O ideal é sempre deixa uma parte menor para os teste e outra maior para o treinamento, por que haverá um quantidade de dados valida para confiar em um teste.

A árvore de decisão é criada, e instancia-se parâmetros para definir uma melhor predição, como a profundidade da árvore ou quantidade de atributos que ela usa. Em primeira mão, houve um teste repetitivo para encontrar a melhor profundidade para a melhora ou maximização da predição, o mesmo com a quantidade de atributos, mas depois de alguns pesquisas, implementou-se a *GridSearchCV*, uma árvore de decisão que, a partir de hiperparametros, entra a melhor configuração para a árvore. O método divide os dados por validação cruzada, evitando *Overfitting* (Adaptação entre dados), testa configuração por configuração, retornando a que apresente a melhor configuração, executando a predição e depois criando um classificador com o método *fit*.

3.3. Validação

E agora para validar, e saber realmente se o classificador criado atende as suas necessidades, usamos métodos de medidas, *accuracy*, *precison*, *recall*, etc.

Como utilizou-se *GridSearchCV* para classificar, ele também mede o score do classificador, exatamente o da sua melhor configuração. Pode-se também medir o eficácia do classificador usando o método *cross_validate*. Ele cruza os dados utilizando o método de validação cruzada criando uma lista de *precision* e *recall*.

Outras métricas também que são *accuracy* e a matriz de confusão, o primeiro compara o teste com os dados previstos retornando um porcentagem de acerto, e a matriz divide-se entre *false/true* negativos e *false/true* positivos, detalhando a ocorrência dos atributos previstos. E por fim, usando a mesma ideia de *false/true* negativos e *false/true*, a curva ROC, utilizado no projeto para calcular a AUC, área embaixo da curva.

3.4. Sistema de Busca

Durante o desenvolvimento da primeira parte do projeto, foi usado a biblioteca *Pandas* para a manipulação do arquivo dos super-poderes. Para o cálculo de distância, foi usado 3, toda das da biblioteca *Scipy*. Apesar de serem cáculo de similaridade, duas das distâncias tendem a 0.0 quanto mais similar.

3.5. Resultados

Realizado todo processo, cada previsão teve um desempenho diferente. O atributo Alignment chegou a ter 66% de *accuracy*, aumentando 10%, 61% de *recall*, e 58% de *AUC*, área embaixo da curva. Accelerated Healing, 80% de *accuracy*, aumentando 5%, e 70 % de *recall*, com *AUC* igual a 78%. Flight, 80% de *accuracy*, aumentando 7%, e 72 % de *recall*, com *AUC* igual a 72%. Super Strength, 78% de *accuracy*, aumentando 4%, e 77 % de *recall*, com *AUC* igual a 80%. Finalizando com o Teleportation, houve 93% de *accuracy*, aumentando 15%, e 76 % de *recall*, com *AUC* igual a 73%. Em geral, todo processo aumentou 5% a 10% de precisão. Como usado métodos avaliativos, estes seriam as porcentagem de eficiência em um ambiente real.

4. Conclusão

Os produtos desenvolvidos atendem a todos os requisitos determinados. Implementações ou upgrades poderiam dar solidez ao projeto geral, utilizando outros métodos de avaliação de classificadores. Maior utilização de bibliotecas para visualização de dados, como a utilização de outras ferramentas.E também, uma predição no pre-processamento melhorando a taxa de *accuracy*. Aumentar a quantidade de distancias por similaridade para dar maior liberdade ao cliente.

5. Dificuldades encontradas

Mesmo as pessoas falando que Python é fácil, exigiu uma boa busca sobre o uso dessa linguagem de programação. Houve dificultou por ter sido as primeiras aplicações do grupo feitas em Python. Como a procura e uso dos cálculos de similaridade, em que a biblioteca apresentava a maioria como calculo de dissimilaridade. E como pre-processar os atributos das bases de dados e uni-las, teve um certo grau de dificuldade. Fora que esse foi o primeiro projeto com aprendizagem de máquina do grupo.

Referências

Documentation, P. https://doc.qt.io/qtforpython/.

Numpy. https://numpy.org.

Pandas. https://pandas.pydata.org.

scikit learn, M. L. i. P. https://scikit-learn.org/stable/.

Scipy, D. https://docs.scipy.org/doc/scipy/reference/spatial.distance.html.