Autor: Diego Vieira
6) Análise de Dados

a) Caso Supermercado

O problema relatado no enunciado se trata de um problema de sistema de recomendação. Para gerar um produto que satisfaça ao cliente sem correr o risco de gerar insatisfação é preciso avaliar situações chamadas **heurísticas de recomendação**.

Considerando conforme o enunciado do problema considera que todos os dados necessários estão disponíveis, os passos são:

1. Montar um conjunto de dados usando um ETL.
2. Determinar heurísticas de recomendações.
3. Escolher um modelo estatístico ou de aprendizado de máquina para treinamento.
4. Separar os dados em um grupo de treinamento e grupo de validação.
5. Verificar métricas de desempenho.
6. Realizar predições em ambiente real ou de produção.
7. Melhorar o modelo com base nas predições.

Explicando cada passo:

1. Montar um dataset usando um ETL. Geralmente o Pandas, mas para especificamente problemas de recomendação, é mais desejável usar o Surprise.
2. Nesse dataset queremos enfatizar as métricas como avaliações de produtos por usuário, número de produtos comprados por categoria, quais produtos um cliente específico comprou, quantos outros clientes compraram o mesmo produto. Parâmetros que queremos saber: quantidade de avaliações individuais por produto, valor médio das avaliações por produto, produtos comprados por usuário, avaliações dos produtos comprados por usuário.
3. Escolher um modelo de aprendizado de máquina que seja usável para a situação. Consideremos que a classificação seja binária para um modelo mais simples: o usuário gostará ou não do produto, então esperamos que um modelo linear seja capaz de resolver o problema com um bom score.
4. O próximo passo é separar os dados presentes em um grupo de treinamento e um grupo de teste. É comum também utilizar algum método de **validação cruzada** para isso. O mais comum é o método de **k-fold cross-validation**, em que se separa os dados em um conjunto de pastas, e se utiliza uma certa proporção, geralmente 80%/20% para treinamento e posteriormente teste. O objetivo dessa divisão em pastas é fazer com que todos os dados passem pelas fases de treinamento e teste, otimizando a quantidade de dados que se possui.
5. Ao treinar o modelo, geralmente se faz sob uma **métrica**. A mais comum é o **accuracy score.** Podemos calcular o accuracy score tanto para treinamento quanto para validação. É também interessante plotar a evolução dessas duas métricas ao longo das épocas em que o algoritmo é treinado, assim podemos avaliar se o modelo está ótimo, ou se está sofrendo com **overfitting** ou **underfitting**. Esses dois estágios são perigosos, pois um modelo em overfitting está “decorando” as predições, então ao ver dados novos, ele não se comportará com o mesmo desempenho. Um modelo em underfitting

não é capaz o suficiente de aprender com eficiência as relações entre dado ou saída, caso isso ocorra, o algoritmo utilizado não é adequado o suficiente.

6. Caso ocorra o underfitting, é necessário escolher outro modelo e repetir os processos de 1 a 5.
7. Caso ocorra o overfitting, é possível melhorar a dinâmica de aprendizado utilizando técnicas de *call-backs*. Pensando em frameworks como scikit learn e o keras, as técnicas de call-backs mais utilizadas para combater o overfitting são o **EarlyStopping** e o **ModelCheckpoint SaveBestOnly**. Esses call-backs têm por objetivo monitorar a atualização dos vieses e dos pesos em cada época de treinamento, parando a atualização dos pesos quando necessário e reiniciando o treinamento a partir do estado atual dos pesos. Essas técnicas servem para modelos complexos e com grande quantidade de dados, conforme assumo que seja o estado no case apresentado. Para modelos e dados mais simples, é possível aplicar a métrica R2 ou então diminuir a quantidade de épocas de treinamento, ou até mesmo a proporção entre a divisão entre treino e teste.
8. Considerando que o modelo esteja treinado sem overfitting, é hora de realizar as predições com os dados reais. Novamente monitora-se métricas para verificar se o modelo não sofrerá queda de rendimento. E se tudo ocorrer bem, o usuário receberá boas predições de produtos que receberá recomendações.
9. Caso o desempenho de acerto caia num nível que não seja aceito, isso determinado pelas regras de negócio, é necessário reavaliar as heurísticas de recomendação, bem como usar os dados das situações reais para melhorar o dataset e refazer o treinamento. Geralmente em um processo envolvendo aprendizado de máquina por técnica supervisionada, os passos descritos nesse algoritmo são feitos de forma recursiva de acordo com as regras de negócio.

B) Técnicas de Pré-Processamento

Antes de aplicar técnicas de Machine Learning em dados, é preciso trabalhá-los de forma que o computador os entenda. As técnicas de pré-processamento são aplicadas em três fases, conhecidas como rotinas ETL (Extract, Transform and Load). De forma mais simples, essas fases de pré-processamento são a limpeza, a transformação e a redução de dados.

Na limpeza, geralmente verifica-se os dados são suficientes ou se os dados apresentam ruídos. Caso isso ocorra, geralmente remove-se os dados inúteis, como atributos nulos, atributos que não podem ser convertidos para um valor numérico. Caso haja uma quantidade insuficiente, completa-se os dados por meio de técnicas de **data-augmentation**. Por exemplo, considerando um algoritmo que detecta padrões em imagens, ao realizar-se o **data augmentation**, cria-se imagens por meio da mudança da resolução, rotação, translação e segmentação das imagens já existentes. Em dados gerais, também é possível utilizar o aumento de dados com técnicas de suavização, regressão e agrupamento. No caso deste último, tem-se por objetivo eliminar dados chamados de *outliers*, que são ocorrências que fogem muito do desvio padrão do conjunto observado. Caso esses outliers não sejam tratados corretamente, isso pode direcionar o treinamento para uma análise equivocada.

Na transformação de dados, normalmente se coloca os mesmos em uma base que seja possível ser reconhecida pelos algoritmos de Machine Learning. Por exemplo, verifica-se se os dados são categóricos ou não. Sendo os mesmos categóricos, aplica-se técnicas de codificação como *one-hot-encoder*, que transforma os dados em uma codificação vetorial, que remove uma possível interpretação numérica que o algoritmo poderia ter sobre eles. Aplica-se a normalização, que

pode ser entendida como a readequação de intervalo de observação. Por exemplo, em imagens de nível de cinza, a normalização transforma os valores de intensidade dos pixels do intervalo de codificação de 0 a 255 para 0 a 1.

Seleciona-se os atributos desejados, que são aqueles os quais se quer obter uma classificação. Voltando ao case do supermercado, estes atributos seriam a quantidade de itens comprados por cliente, a nota do produto, por exemplo. Em seguida aplica-se a discretização, principalmente se os dados forem dispostos em séries temporais. Nesse processo, é determinado um intervalo de amostragem, em que os dados são seccionados. Por último, avalia-se os atributos estão dispostos em uma ordem hierarquizada, o que permite fazer assunções prévias sobre os dados.

Normalmente, os passos descritos até aqui são suficientes para a resolução da maioria dos problemas em Machine Learning. Mas em alguns casos, a redução de dimensionalidade se dá no propósito de diminuir a complexidade do problema. Esse processo pode ser tanto para diminuir a quantidade de dados treinados por vez, os *batches sizes*, quanto na dimensão de classes que o problema pode identificar. Neste caso, por exemplo, um conjunto de n-atividades humanas possíveis de serem detectadas ou de n-palavras que possuem conotação semântica dentro de um analisador de sentimento podem ser reduzidas para uma ordem inferior, como por exemplo um problema binário, com classes POSITIVO E FALSO. Para realizar a redução de dimensionalidade, emprega-se técnicas estatísticas como a Análise de Componentes Principais.

C) Outliers ou Anomalias

Outliers ou anomalias são dados que estão fora do intervalo esperado que o conjunto geral se comporte. Por exemplo, em uma observação de candidatos que participam do Exame Nacional do Ensino Médio, o ENEM, observa-se que a maioria dos indivíduos inscritos possuem idade entre 16 e 22 anos, porém há outros que possuem idades como 56, 40, 65 anos. Estes últimos indivíduos são chamados de outliers quanto à idade. A presença de outliers em um conjunto de dados pode induzir a classificações equivocadas em relação ao atributo analisado.

A melhor forma de identificar outliers em um conjunto de dados é por meio de visualização por meio de gráficos. O método mais adequado é calcular os quartis do conjunto de dados e plotar os boxplot por meio dos quartis. Então, identificar-se-á os elementos em torno da mediana e os elementos acima dos quartis dos ¾ e abaixo dos quartis do ¼. Estes elementos externos podem ser tratados como outliers e então removidos do conjunto.

D) Algoritmo de Machine Learning: Perceptron

O modelo perceptron de camada simples, ou somente Perceptron, ou até neurônio é o modelo mais simples de rede neural e a estrutura básica de todos os algoritmos de Deep Learning conhecidos até hoje. O perceptron funciona por sua regra de aprendizado, chamada de Regra de Aprendizado do Perceptron.

Ele possui um conjunto de entrada e um conjunto de saída, e dois itens de atualização conhecidos como bias, ou viés, e pesos. O conjunto de saída pode ser definido como uma multiplicação dos pesos pela entrada, subtraído pela função de ativação.

A função de ativação é um mecanismo que considera o quanto o estado atual da predição está desviando do resultado esperado por meio de um erro, que é calculado pelo resultado esperado subtraído do viés. Essa função de erro determina a função de ativação somando-se ao estado anterior da observação, ou seja, é um processo que se repete ao longo do tempo. Assim, a

resposta, ou conjunto de saída liberado pelo perceptron é corrigido iteração a iteração, convergindo na direção da saída esperada. Como esse algoritmo é de aprendizado supervisionado, é necessário conhecer a saída esperada. Portanto, partindo de uma predição arbitrária, o perceptron por meio de seus pesos e atualização de funções de ativação vai corrigindo a trajetória em direção a saída esperada. O processo é recursivo, ou seja, se repete até atingir uma predição adequada, e cada repetição é denominada iteração, que pode ser detalhada a seguir:

1. Recebe-se os dados de entrada.
2. Prediz-se uma saída arbitrária.
3. Multiplica-se os dados da entrada pela saída arbitrada anterior (pesos).
4. Obtém-se o erro
5. Calcula-se novos pesos subtraindo a função de ativação atual pelo novo erro.
6. Repete-se o processo.