Geração de features através da exploração de conhecimento de domínio.

INTRODUÇÃO

Chegámos à era do Big Data, onde não só a quantidade de dados colecionados aumentou exponencialmente, como procurá-los e aproveitá-los traz as maiores vantagens ao nível profissional, comercial e industrial. No entanto, processar estes dados manualmente já não é possível, daí que tenha havido um “boom” em Machine Learning. O processo de extrair informação dos dados, isto é, Ciência de Dados, tem expandido para todos os domínios, o que levou ao problema de não haver cientistas de dados suficientes (e ainda menos cientistas experientes), que consigam lidar com esta quantidade de dados. Devido a isto, a investigação virou-se para a automatização do processo de ciência de dados de forma a reduzir a necessidade de intervenção humana.

Nos últimos anos, têm aparecido soluções de AutoML que já são capazes de lidar com partes importantes do processo, principalmente na seleção e otimização de modelos. Para serem o mais geral possível, usam métodos independentes ao domínio, funcionando como caixas-negras onde não se conseguem ver as operações que aplicam aos dados para os processar. Por outro lado, as soluções ainda não estão muito focadas numa das áreas onde cientistas demoram mais tempo, que é a área de feature engineering, uma das partes mais demoradas e importantes do processo KDD ou processo de descoberta de conhecimento.

Para aliviar este problema, a nossa proposta é melhorar a área de automatização do processo KDD, principalmente na área de geração de variáveis aproveitando o conhecimento de domínio, que consegue tornar os modelos mais robustos. Queremos aproveitar o uso deste conhecimento ao representá-lo num formalismo, que é um modelo entidade-relação e principalmente criar um algoritmo que interpreta o conhecimento no modelo e usa-o para expandir conjuntos de dados. Para isto, também desenvolvemos um conjunto de operadores que o algoritmo consegue entender e que podem ser usados para expandir os datasets automaticamente.

Para avaliar a proposta, criámos um caso de estudo adequado ao problema que estamos a tentar resolver, composto por vários datasets de vários domínios, cada um com um problema de classificação e um modelo ER, avaliando o algoritmo em termos de eficácia e eficiência, como vai ser explicado mais tarde.

BACKGROUND

Antes de apresentar a proposta, mostro um bocado do background do trabalho com os tópicos fundamentais para o trabalho. Primeiro, explicar o processo KDD, um termo que antecede ciência de dados, mostrando passos bem definidos para extraír e aproveitar informação destes. O processo é composto por 9 passos: entender o domínio, colecionar dados, preprocessá-los, engenharia de variáveis, definir um objetivo (classificação, regressão…), análise exploratória, data mining (uso de algoritmos de machine learning para criar modelos), interpretar os modelos e colocar em produção. A metodologia definida no processo KDD foi usada durante este trabalho, com maior foco em feature engineering.

Feature engineering é o passo onde normalmente os cientistas demoram mais tempo, por ser subjetivo e precisar de intervenção manual. Está dividido em três categorias, em que cada uma influencia a dimensão do conjunto de dados de maneira diferente. Seleção e extração de variáveis normalmente diminuí a dimensionalidade das variáveis, eliminando as redundantes ou irrelevantes ou fazendo uma combinação de variáveis para alterar o seu espaço, dependendo do método.

Por outro lado, feature engineering pode ser ainda a geração de variáveis, com ou sem conhecimento de domínio. Sem conhecimento de domínio, ainda pode ser dividida em baseada em dados, e baseada em hipóteses, dependendo do processo usado para as gerar. Este é o maior foco do trabalho.

Por fim, também é preciso estudar formas de representação de conhecimento. Dos formalismos estudados durante o trabalho, os mais importantes são Production Rule Systems (ou sistemas if-then), ontologias que são um corpo de conhecimento que descreve um facto, possuem axiomas e são úteis a todo o processo KDD. E no domínio de bases de dados, o modelo entidade-relação ou ER , que funciona como uma ontologia conceptual que expressa entidades e como estas estão relacionadas. Estes dois formalismos vão ser comparados com mais detalhe a seguir.

RELATED WORK

Resumidamente, o estado da arte em geração de variáveis pode ser dividido em três vertentes, a geração de variáveis sem conhecimento de domínio, que pode seguir uma abordagem guiada pelos dados, aplicando operações a variáveis ou conjuntos de variáveis, ou métodos guiados por hipóteses, em que usam as hipóteses geradas pelos algoritmos, principalmente decision trees. Outros trabalhos usam metodologia “expand-reduce”, em que geram um conjunto elevado de variáveis e fazem seleção de variáveis no final.

As frameworks de AutoML, como foi dito, ainda não se focam muito em gerar variáveis. Como tentam ser agnósticas ao domínio, não exploram nenhum conhecimento, aplicando apenas métodos simples como embeddings, one-hot encoding ou combinações polinomiais.

Outra vertente são métodos que usam conhecimento de domínio, onde os trabalhos mais importantes são métodos guiados por hipóteses semelhantes aos anteriores, ou os que trabalham em expandir bases de dados relacionais, ou a expansão de conjuntos de dados textuais através de fontes como a Wikipédia, que possui muito conhecimento disponível, mas ao mesmo tempo é preciso lidar com muito ruído.

A última vertente são métodos que já usam formalismos de representação de conhecimento, como grafos e linked data, que seguem um esquema ou ontologia e extraem informação de bases de conhecimento ricas, ou os métodos que usam ontologias, em que usam estas representações de conhecimento como forma de relacionar termos entre elas e dados e assim aumentar a quantidade de informação presente.

SOLUTION PROPOSAL

Como dito anteriormente, os métodos atuais de Machine Learning automático não se focam em geração de variáveis, nem exploram conhecimento de domínio.­

Assim, a nossa proposta é a geração de variáveis usando o conhecimento de domínio, representado com um formalismo de representação de conhecimento, ou seja, um modelo. O conhecimento é representado por conceitos e as relações entre estes, o que pode levar à criação de novas variáveis tendo em conta as suas propriedades e estas relações. Para a geração, é preciso criar um algoritmo que consiga interpretar o modelo e gere as variáveis. É importante que o algoritmo consiga funcionar independentemente do domínio, para poder ser usado como as frameworks de AutoML, daí que queremos que o conhecimento seja externo ao algoritmo.

A abordagem ideal para representar o conhecimento seria uma ontologia, pois permite definir os conceitos e relações, e usando os axiomas que mapeiam o conhecimento e as variáveis, seria possível gerar variáveis. No entanto, há problemas com esta abordagem, como a necessidade de definir claramente este mapeamento, e principalmente o facto de ainda não se encontrarem muitos conjuntos de dados semanticamente ricos e com ontologias prontas a usar.

A alternativa encontrada foi usar um modelo mais simples e com mais disponibilidade, representando o conhecimento num modelo ER, que é amplamente usado no contexto de bases de dados. Estes modelos podem ser vistos como quase ontologias, apenas faltando os axiomas. Mesmo assim, permitem mapear os dados a conceitos e entender as relações entre estes, que é o fundamental para a geração de variáveis. É preciso agora resolver o problema da reificação, decidindo o que queremos representar como objeto ou como atributo para um objeto. Como queremos trabalhar e deliberar sobre o conjunto de dados inteiro, considerámos as variáveis todas como objetos ou entidades, não representando atributos.

O modelo ER para representar o conhecimento tem as entidades mapeadas às variáveis do conjunto de dados a expandir e está representado em JSON, mas podia estar representado em outras como XML ou OWL. No entanto, tem de haver uma definição clara: cada entidade tem nome igual à variável, tipo e descrição adicional. Relações precisam de uma definição maior, pois são as relações que definem as novas variáveis a serem geradas. Cada relação tem o nome da nova variável, os inputs que são as entidades usadas para gerar variáveis, as operações que são usadas nos inputs para criar a variável, e possíveis restrições a seguir. Para as variáveis que precisem de agregações (que vão ser vistas em detalhe a seguir), existe ainda parâmetro de groupby para definir a variável usada como agregação, e alguma condição adicional como o número de registos a agregar.

Com o esquema do modelo definido, é preciso agora definir como perceber as operações. Para ser possível definir todas as operações imagináveis, seria preciso que o algoritmo conseguisse descodificar a operação descrita no modelo e transformar numa função. No entanto, isto acrescenta uma complexidade enorme à solução. A alternativa foi definir um conjunto de operações ou “templates” de relação que o algoritmo consegue entender e calcular, mantendo assim um número elevado de possibilidades. Definimos estas operações (decomposição, algébricas, mapeamento, agregação e composição), em que as primeiras três são consideradas *baseadas em registos*, porque apenas precisam da informação do próprio registo, e *baseadas em agregações*, que necessitam de informação de registos para lá do que está a ser processado (e precisam assim de um groupby).

Explicando em mais detalhe o procedimento com operações baseadas em registos, para um certo registo na variável a gerar, caso algum dos inputs não passe na restrição, é colocado null. Caso contrário, a operação definida na relação é calculada para os inputs definidos, e o resultado é inserido na nova variável desse registo.

O procedimento com operações baseadas em agregações é semelhante, mas os registos a agregar são guardados temporariamente para ser calculada a agregação, tendo em conta a variável definida na relação como groupby. Apenas são guardados os registos que passam a condição. Após a agregação estar concluída, o valor é inserido na nova variável desse registo.

Agora que temos as operações e o problema explicado, passamos à definição do algoritmo em si e como funciona. O algoritmo é o DANKFE, que conforme dito tem a função de interpretar um modelo ER com conhecimento de domínio e assim expandir um conjunto de dados. O algoritmo coloca as relações do modelo numa fila, processando uma de cada vez. Caso os inputs já estejam disponíveis, o algoritmo aplica a lista de operações (que estão presentes no algoritmo como um Production Rule System ou um sistema de if e then), e aplica os cálculos necessários para todos os registos que passem a restrição imposta, até não haver mais variáveis a processar. Esta primeira versão do DANKFE consegue processar variáveis cujos valores apenas dependem de valores do mesmo registo. É a versão mais rápida mas com menor funcionalidade.

Por outro lado, a segunda iteração do algoritmo permite gerar variáveis com operações baseadas em agregações. O funcionamento do algoritmo é semelhante, mas quando existe uma variável com *groupby* definido, os registos relevantes são guardados e passados ao Production Rule System do algoritmo, para que a agregação seja feita. O valor calculado é inserido na nova variável para o registo que está a ser processado, se passar as possíveis restrições. Esta versão permite mais funcionalidades fazendo um tradeoff com complexidade temporal.

Um lado que nenhuma das versões do DANKFE explora é o resto do processo de engenharia de variáveis e preparação de dados. Está provado que esta fase é a mais dispendiosa, mas também a que mais contribui para melhores resultados, daí que se possa acoplar o algoritmo no resto do processo de preparação de dados, permitindo fazer limpeza de dados, outras operações de preprocessamento e até geração de variáveis automaticamente, com pouco conhecimento de domínio, explorando apenas os tipos das variáveis. Assim, o DANKFE-III permite preencher missing values, faz a geração automática de certas variáveis, como a decomposição automática de datas e cria o “resumo de 5 números” de variáveis numéricas (média, mediana, máximo, mínimo e desvio padrão), ao estender o modelo ER com estas novas variáveis, seguindo um template que é completado automaticamente. Por fim, também permite escalar e balancear os dados caso o utilizador queira, para que a versão retornada do conjunto de dados tenha a melhor preparação possível.

Para isto, é preciso adicionar ao algoritmo um ficheiro de configurações, como se pode ver à direita, onde o utilizador pode escolher que operações de preparação são efetuadas, tanto de processamento como de geração automática. Para a geração automática de variáveis, existem variáveis de template, como se pode ver à esquerda, que são adicionadas automaticamente ao modelo ER, por exemplo, para extrair o dia de uma variável data.

CASE STUDY

Para testar a nossa proposta, foi necessário criar um caso de estudo relevante ao problema, seguindo o processo KDD. Daí, foram usados 9 conjuntos de dados de naturezas diferentes para testar o impacto do uso de conhecimento de domínio na geração de variáveis. Para todos os conjuntos, foi criado um modelo ER com conhecimento e uma classe binária para fazer um problema de classificação. Também foi feita uma análise exploratória.

Um exemplo de modelo ER usado para o conjunto de dados de Air Quality é este, onde se veem as variáveis originais como entidades ou retângulos brancos, as variáveis geradas como retângulos azuis, e as relações para as criar como losangos. A cor do losango depende do tipo de operação (laranja para decomposição, azul para algébrica, verde para mapeamento e amarelo para agregação).

A metodologia para avaliar o algoritmo testa a sua eficácia e eficiência. Foi testada a performance de vários modelos de machine learning nos conjuntos de dados originais e em cada versão estendida, medindo a importância dada às variáveis geradas. O tempo gasto em gerar e treinar modelos também foi medido, seguindo um critério semelhante ao das ferramentas de AutoML, fazendo alguma otimização de hiperparâmetros. Também foi medido o tempo por operação e a escalabilidade do algoritmo. Para comparar com as ferramentas de AutoML que fazem todo o processo KDD, o mesmo problema foi testado com o Auto-Sklearn, uma das frameworks mais usadas.

Podemos ver os resultados de cada versão do algoritmo. O DANKFE-I melhorou a performance de todos os modelos em comparação com o conjunto de dados original, atingindo resultados superiores ao auto-sklearn no KNN e Gradient Boosting. Isto demorando apenas 10 a 20 milissegundos por registo, comparado com 300. Os modelos dão maior importância às variáveis geradas em 5 dos 9 datasets, e como o DANKFE-I apenas consegue lidar com operações baseadas em registos, todas elas ocupam tempos aproximadamente semelhantes.

Para o DANKFE-II, que consegue processar operações baseadas em agregações, os resultados são melhores, atingindo melhor performance que o Auto-Sklearn para todos menos Naive Bayes. Como agora é preciso gerar variáveis que precisam de guardar e agrupar valores, o algoritmo demora mais a correr, mas em média abaixo dos 50 milissegundos por registo. As variáveis geradas já são as mais importantes em 7 ou 8 conjuntos de dados.

Para o DANKFE-III, como é feito todo o processo de preparação de dados, foi preciso também fazer as mesmas operações aos dados originais para comparar (a rosa). Mesmo com o aumento causado pela preparação, a geração de variáveis melhorou sempre a performance dos modelos, aumentando ligeiramente o tempo de processamento para acima dos 50 millisegundos, principalmente devido ao maior número de variáveis a gerar. Apenas um dataset não deu maior importância às variáveis geradas com conhecimento de domínio pelo DANKFE-III.

Estudando a escalabilidade do algoritmo, podemos ver que quase toda a complexidade temporal vem das operações baseadas em agregações, mantendo um comportamento linear com a variação do tamanho do conjunto de dados.

CONCLUSION

Em suma, o uso de conhecimento de domínio ainda está muito preso a domínios específicos e à disponibilidade de bases de conhecimento para esses domínios. Por outro lado, o aumento da quantidade de dados que necessitam de ser processados diariamente levou a um aumento na procura por cientistas qualificados, o que leva a uma necessidade de facilitar ou automatizar o processo KDD. Estes dois fatores fazem com haja um menor aproveitamento do conhecimento de domínio no processo, principalmente em geração de variáveis. Assim, o nosso trabalho propõe esta melhoria de machine learning automático para aproveitar o conhecimento, que é sabido que melhora os resultados, através de um algoritmo que automaticamente processa o conhecimento e estende os conjuntos de dados fornecidos, permitindo um grande conjunto de operações. As várias versões do algoritmo permitem trocar velocidade de processamento (o que pode ser importante no meio profissional), com as várias funcionalidades desejadas, com resultados que em média superam a performance do auto-sklearn, uma das ferramentas mais utilizadas para AutoML, demorando no pior caso 6x menos tempo por registo, o que mostra a importância do uso de conhecimento de domínio, que também pode melhorar a confiança nos sistemas AutoML.

Por outro lado, a nossa proposta ainda tem algumas limitações, como no conjunto limitado de operações, onde para se conseguir gerar qualquer variável possível seria preciso desenvolver um descodificador que interpretasse qualquer conhecimento no modelo ER e traduzisse para uma função. Outras operações poderiam ser desenvolvidas à volta do DANKFE, como a melhoria do preprocessamento, ou o uso de seleção ou extração de features para retirar variáveis irrelevantes ou redundantes, e diminuir a complexidade dos conjuntos de dados finais. O DANKFE também pode ser otimizado de outra maneira, permitindo gerar mais que uma variável ao mesmo tempo, tendo em conta as restrições de cada. Outro rumo a estudar é continuar o estudo feito no background, experimentando aproveitar formalismos de representação de conhecimento mais complexos, como ontologias ou modelos ER estendidos, que pecam por ter menor disponibilidade.