Capítulo 3

Explainability e *auditability*: interpretando e validando modelos de *machine learning*

Wendley Souza da Silva, Victória Tomé Oliveira, Sayonara Santos Araújo, Dario Vieira e Miguel Franklin de Castro

Abstract

Machine learning (ML) models and solutions have many applications in our daily lives, with the real possibility to improve products, processes, and techniques; however, in some situations, the systems do not sufficiently or convincingly explain their predictions. Understanding decision-making in highly sensitive areas such as healthcare or finance is of paramount importance to provide transparency, security, and trust to the users of intelligent systems. This short course introduces the participant, in theory and practice, to strategies and techniques for making machine learning models and their decisions interpretable and auditable.

Resumo

Os modelos e soluções de aprendizado de máquina (ML) possuem diversas aplicações no nosso cotidiano, com real possibilidade de melhorar produtos, processos e técnicas; entretanto, em algumas situações, os sistemas não explicam de forma suficiente ou convincente as suas previsões. Entender a tomada de decisões em áreas altamente sensíveis, como saúde ou finanças, é de suma importância para oferecer transparência, segurança e confiança aos usuários dos sistemas inteligentes. Este minicurso apresenta ao participante, na teoria e prática, as estratégias e técnicas para tornar os modelos de aprendizado de máquina e suas decisões interpretáveis e auditáveis.

3.1. Introdução

A precisão dos modelos atuais de Inteligência Artificial (IA) é notável, mas tal atributo não é o único aspecto de significativa importância. Para domínios sensíveis, uma compreensão detalhada do modelo e das saídas também é importante. Os algoritmos de aprendizado de máquina e aprendizado profundo constroem modelos complexos que são opacos

para humanos burkart2021survey. Holzinger et al. (2019) afirmam que o domínio médico está entre os maiores desafios para a IA. Para áreas como saúde, onde uma compreensão profunda da aplicação de IA é crucial, a necessidade de Inteligência Artificial Explicável (XAI, em inglês, *explainable artificial intelligence*) torna-se notável.

A explicabilidade é importante em muitos domínios, mas não exatamente em todos os domínios. Já mencionamos áreas em que a explicabilidade é importante, como a saúde. Por outro lado, em outros domínios como o de colisões de aeronaves, os algoritmos operam sem interação humana e sem dar explicações há anos. A explicabilidade é necessária quando há algum grau de completude (Burkart, Huber, 2021).

De acordo com Lipton (2018), a explicabilidade é exigida sempre que o objetivo para o qual o modelo de previsão foi construído difere do uso real quando o modelo está sendo implantado. Em outras palavras, a necessidade de explicabilidade surge devido ao descompasso entre o que um modelo pode explicar e o que um tomador de decisão deseja saber. De acordo com Martens et al. (2008), a explicabilidade é essencial sempre que um modelo precisa ser validado antes de ser implementado e implantado. Os domínios que demandam explicabilidade são caracterizados pela tomada de decisões críticas que envolvem, por exemplo, uma vida humana, mais especificamente, na área da saúde.

Nesse contexto, o renovado Regulamento Geral de Proteção de Dados da UE (GDPR) (Europa.eu, 2017) poderá exigir que os provedores de IA forneçam aos seus usuários explicações sobre os resultados da tomada de decisão automatizada com base em seus dados pessoais. O GDPR substituiu a Diretiva de Proteção de Dados de 1995 e esse novo requisito afeta grande parte da indústria. O Parlamento Europeu revisou os regulamentos que dizem respeito à coleta, armazenamento e uso de informações pessoais. O GDPR pode restringir ou até mesmo levar à proibição do uso de modelos opacos que são usados para determinadas aplicações, por exemplo, para sistemas de recomendação que funcionam com base em dados pessoais. Goodman, Flaxman (2017) chamam isso de direito de explicação para cada sujeito (pessoa). Isso provavelmente afetará instituições financeiras, redes sociais e o setor de saúde. A tomada de decisão automatizada usada por instituições financeiras para monitorar o risco de crédito ou lavagem de dinheiro precisa ser transparente, interpretável e responsável.

Em maio de 2017, a DARPA (Gunning, 2017) lançou o programa XAI que visa fornecer modelos explicáveis e altamente precisos. XAI é um termo genérico para qualquer pesquisa que tente resolver o problema da caixa preta para a IA. Uma vez que existem muitas abordagens diferentes para resolver este problema, cada uma com suas próprias necessidades e objetivos individuais, não há uma única definição comum do termo XAI. A ideia-chave, no entanto, é permitir que os usuários entendam a tomada de decisão de qualquer modelo.

Explicabilidade e interpretabilidade também são aspectos importantes para modelos de aprendizado profundo, onde uma decisão depende de uma enorme quantidade de pesos e parâmetros. Aqui, os parâmetros são muitas vezes abstratos e desconectados do mundo real, o que dificulta a interpretação e explicação dos resultados dos modelos profundos (Angelov, Soares, 2020). Por muito tempo, os modelos de ML eram universalmente vistos como caixas pretas porque humanos não podiam explicar o que acontecia com os dados entre a entrada e a saída. A partir disto, surgiu a explicabilidade, que será

detalhada a seguir. Dessa forma, este minicurso apresenta ao participante, na teoria e prática, as estratégias e técnicas para tornar os modelos de aprendizado de máquina e suas decisões explicáveis e interpretáveis, alcançando a auditabilidade.

3.2. Conceitos e técnicas de *explainability* e *auditability*

3.2.1. Explainability

Para explicar as previsões de um modelo de aprendizado de máquina, pode-se utilizar algum método de explicação, que na prática é um algoritmo que gera explicações. Uma explicação geralmente relaciona os valores de recursos de uma instância à previsão do modelo de uma maneira humanamente compreensível. A explicabilidade no aprendizado de máquina significa que você pode explicar o que acontece em seu modelo da entrada à saída, tornando os modelos mais transparentes e resolvendo, ou minimizando, o problema da caixa preta (Burkart, Huber, 2021).

A IA explicável (XAI) é a maneira mais formal de descrever isso e se aplica a toda a inteligência artificial. XAI significa métodos que ajudam especialistas humanos a entender as soluções desenvolvidas pela IA. "Explicabilidade" e "interpretabilidade" são frequentemente usados de forma intercambiável, apesar de terem o mesmo objetivo que é entender o modelo (Burkart, Huber, 2021).

Em seu livro, "*Interpretable Machine Learning*", Christoph Molnar (Molnar, 2020) define interpretabilidade como o grau em que um ser humano pode entender a causa de uma decisão ou o grau em que um ser humano pode prever consistentemente os resultados do modelo de ML.

A IA explicável trata de entender melhor os modelos de ML. Como eles tomam decisões e por quê. Os três aspectos mais importantes da explicabilidade do modelo são:

- Transparência;
- Capacidade de questionar;
- Facilidade de entendimento.

Pode-se abordar a explicabilidade de duas maneiras:

Globalmente Esta é a explicação geral do comportamento do modelo. Ele nos mostra uma visão geral do modelo e como os recursos nos dados afetam coletivamente o resultado;

Localmente Isso nos informa sobre cada instância e recurso nos dados individualmente (como explicar as observações vistas em determinados pontos do modelo) e como os recursos afetam individualmente o resultado.

3.2.1.1. Razões para a Explicação

Os sistemas automatizados de tomada de decisão não são amplamente aceitos, pois os humanos querem entender uma decisão ou pelo menos querem obter uma explicação

para certas decisões. Sendo assim, a confiança, então, é um dos aspectos motivadores da explicabilidade. Outros aspectos motivadores são causalidade, transferibilidade, informatividade, tomada de decisão justa e ética (Lipton, 2018), prestação de contas, ajustes e funcionalidade de proxy.

Confiança Confiança e aceitação do modelo de previsão são necessários para a implantação do modelo de previsão. Compreender e conhecer os pontos fortes e fracos do modelo de previsão é um pré-requisito para a confiança humana e, portanto, para a implantação do modelo.

Causalidade Explicação, por exemplo na forma de importância do atributo, transmite uma sensação de causalidade ao grupo-alvo do sistema. Este conceito de causalidade só pode ser apreendido quando o sistema aponta a relação insumo-produto subjacente.

Transferibilidade O modelo de previsão precisa transmitir uma compreensão do comportamento futuro para um tomador de decisão humano para usar o modelo de previsão com dados não vistos. Somente quando o decisor souber que o modelo generaliza bem ou quando souber em que contexto ele generaliza bem, o modelo de previsão será encarregado de tomar decisões.

Informatividade Para ser implantado como um sistema, é necessário saber se o sistema realmente atende aos propósitos do mundo real para o qual foi projetado, em vez de apenas servir aos propósitos para os quais foi treinado. Se essas informações forem fornecidas, o sistema poderá ser implantado.

Tomada de decisão justa e ética conhecer as razões de uma determinada decisão é uma necessidade da sociedade e provavelmente será um direito oficial dos cidadãos da UE (Goodman, Flaxman, 2016). Esse direito à explicação exige que os tomadores de decisão apresentem seus resultados de forma compreensível para perceber a conformidade com os padrões éticos. Cada pessoa afetada por uma decisão automatizada pode fazer uso desse direito à explicação.

Responsabilidade Um objetivo de incorporar explicabilidade no processo de tomada de decisão é tornar um algoritmo responsável por suas ações. Para que um sistema seja responsável, ele deve ser capaz de explicar e justificar suas decisões. Além disso, o problema do deslocamento de dados pode ser direcionado com sistemas interpretáveis, tornando-os mais responsáveis por suas ações (Freitas, 2014).

Ajustes Compreender o modelo de previsão e os fatores subjacentes permite que especialistas de domínio comparem o modelo de previsão com o conhecimento de domínio existente. A explicabilidade é um pré-requisito para a capacidade de ajustar o modelo de previsão incorporando o conhecimento do domínio. De acordo com (Selvaraju et al., 2017), a explicabilidade dos modelos de previsão pode ensinar os humanos, especialmente os especialistas de domínio que usam esses modelos de previsão, como tomar melhores decisões. Além disso, quando observada de um ponto de vista algorítmico, a explicabilidade permite que os projetistas de sistemas façam alterações no modelo de previsão, por exemplo, ajustando parâmetros.

A explicabilidade também é útil para desenvolvedores, pois pode ser usada para identificar modos de falha.

Funcionalidade de proxy quando a explicabilidade é fornecida por um sistema, ela também pode ser examinada com base em outros critérios que não podem ser facilmente quantificados, como segurança, não discriminação, privacidade, robustez, confiabilidade, usabilidade, justiça, verificação e causalidade (Doshi-Velez, Kim, 2017). Nesse caso, a explicabilidade serve como proxy.

3.2.1.2. Modelos explicáveis

Alguns modelos em ML têm a propriedade característica de explicabilidade, ou seja, transparência, facilidade de compreensão e capacidade de questionar. Sendo eles:

1. **Modelos lineares** - Modelos lineares como regressão linear, SVMs com kernel linear, etc seguem o princípio da linearidade de que duas ou mais variáveis podem ser somadas para que sua soma também seja uma solução. Por exemplo, y = mx + c.

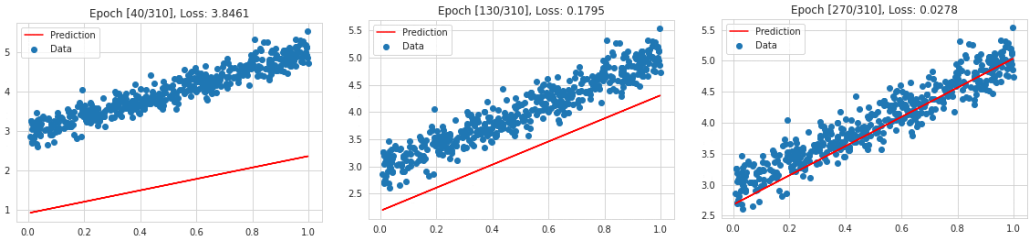

Figura 3.1. Modelo linear [Autor].

2. **Algoritmos de Árvore de Decisão** - Os modelos que usam árvores de decisão são treinados aprendendo regras de decisão simples obtidas de dados anteriores. Como eles seguem um conjunto específico de regras, entender o resultado depende simplesmente de aprender e entender as regras que levaram ao resultado.

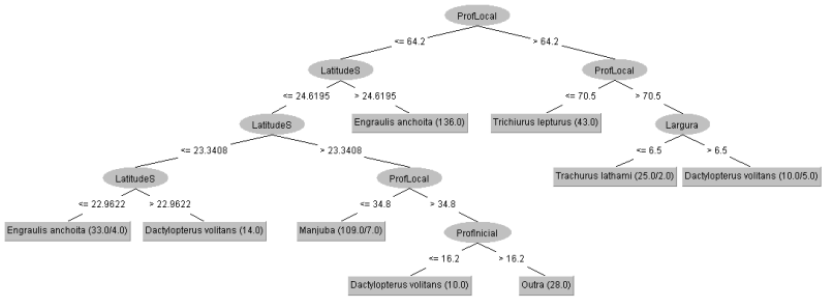

Figura 3.2. Algoritmos de Árvore de Decisão (Costa, 2019).

3. **Modelos Aditivos Generalizados (GAM)** - GAMs são modelos onde a relação usual entre variáveis preditivas e variável dependente (resposta) é substituída por funções suaves lineares e não lineares para modelar e capturar a não linearidade nos dados. GAMs são modelos lineares generalizados com função de suavização devido à sua natureza viciante, cada variável contribui para a saída. Assim, podemos explicar a saída de um GAM simplesmente entendendo as variáveis preditivas.

O problema com os modelos mais explicáveis é que eles na maioria das vezes não capturam a complexidade de alguns problemas do mundo real e podem ser inadequados. Além disso, porque um modelo é simples ou linear, isso não garante explicabilidade. Redes neurais ou modelos de conjunto, etc, são modelos complexos. Assim, para modelos complexos, usamos técnicas e ferramentas para torná-los explicáveis. Existem duas abordagens principais:

Abordagem Modelo-Agnóstica Técnicas/ferramentas independentes de modelo podem ser usadas em qualquer modelo de aprendizado de máquina, não importa o quão complicado seja. Esses métodos agnósticos geralmente funcionam analisando os pares de entrada e saída de recursos. Um bom exemplo é o LIME.

Abordagem Específica do Modelo As técnicas/ferramentas específicas do modelo são específicas para um único tipo de modelo ou um grupo de modelos. Eles dependem da natureza e das funções do modelo específico, por exemplo, intérpretes de árvore.

Existem inúmeras técnicas explicabilidade em ML, sendo elas: Parcelas de Dependência Parcial (PDP), Gráficos de expectativas de condição individual (ICE), Deixar uma coluna de fora (LOCO), Efeitos Locais Acumulados (ALE), Explicações agnósticas do modelo interpretável local (LIME), Âncoras, SHapley Additive exPlanations (SHAP), Recursos importantes de aprendizado profundo (DeepLIFT), Propagação de relevância em camadas (LRP), Método de Explicações Contrastivas (CEM) entre outros.

3.2.2. Auditability

Ao mesmo tempo que a Inteligência Artificial traz inúmeras facilidades às tarefas humanas, ela também acarreta desafios referentes à segurança, robustez e confiabilidade. Uma das abordagens relevantes para enfrentar esses desafios é utilizar mecanismos que permitam auditar sistemas de IA, como ferramentas, padrões e estratégias de avaliação (Berghoff et al., 2021).

As complexas interligações entre os componentes dos sistemas, os diversos fluxos de dados e muitas vezes, o grande número de pessoas envolvidas no processo de treinamento, implantação e manutenção dos modelos, faz com que seja necessário a utilização de alguma forma de auditoria para garantir a integridade das operações (Senft, Gallegos, 2008). Auditoria é basicamente uma análise imparcial que compara o produto desejado e o produto obtido, cujo objetivo é averiguar se o planejamento desse produto foi realizado e como foi desempenhado (Araújo, 2001).

A auditoria fornece garantia às várias partes interessadas de que o objeto alvo da auditoria está de acordo com sua missão e objetivo. Em Aprendizado de Máquina, a

auditabilidade fornece uma maior transparência aos modelos, enquanto que a explicabilidade interpreta os resultados obtidos (Toreini et al., 2020). Dessa forma, as duas técnicas contribuem para que os usuários deixem de enxergar as soluções de ML como uma caixa-preta misteriosa.

Mais especificamente, a auditabilidade avalia a influência dos dados de entrada na saída do modelo. Em outras palavras, auditabilidade diz respeito a qualquer informação que possa ser utilizada para descrever os dados e processos, como por exemplo: a natureza dos dados, o que inclui sua origem, destino e as dependências associadas a eles; as etapas de processamento; e outras informações contextuais, como aspectos de proteção de dados, configuração do sistema, ações da equipe, etc (Singh et al., 2018).

Auditar a IA não é muito diferente de auditar qualquer outra tecnologia emergente (e.g. computação em nuvem), exceto que a IA tem o potencial de impactar desproporcionalmente grupos já marginalizados, com por exemplo, o viés inerente aos conjuntos de dados utilizados no treinamento de novos modelos. Uma das ferramentas que pode ser utilizada no processo de auditabilidade é o *framework* proposto por Hummer et. al. chamado de ModelOps, que visa fornecer as ferramentas necessárias para o *deploy*, governança e monitoramento de modelos de IA (Hummer et al., 2019).

Embora a auditabilidade não seja um conceito novo, ela possui um potencial enorme por ser capaz de permitir que as soluções de IA sejam vistas como confiáveis pelos usuários. Assim, é importante desenvolver técnicas e ferramentas que visem melhorar tal área, pois só com modelos mais confiáveis de IA são a chave para o crescimento dessa tecnologia.

3.3. *Local Interpretable Model-Agnostic Explanations* (LIME)

O *Local Interpretable Model-Agnostic Explanations* (LIME) foi desenvolvido por pesquisadores da Universidade de Washington para ver o que acontece dentro de um algoritmo capturando interações de recursos. O LIME é uma estrutura mais geral que visa tornar as previsões de "qualquer" modelo de aprendizado de máquina mais interpretável (Ribeiro et al., 2016a).

O LIME explica as previsões do modelo no nível da amostra de dados. Ele permite que os usuários finais interpretem essas previsões e executem ações com base nelas, como mostra a Figura 3.3.

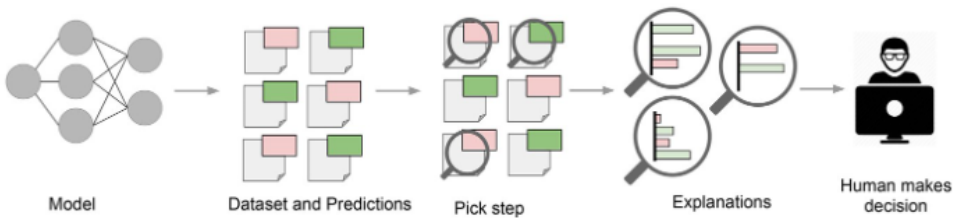

Figura 3.3. Explicando um modelo para um decisor humano (Ribeiro et al., 2016b).

O LIME é independente de modelo, o que significa que pode ser aplicado a qualquer modelo de aprendizado de máquina. A técnica tenta entender o modelo perturbando

a entrada de amostras de dados e entendendo como as previsões mudam. O LIME executa várias perturbações de vários recursos em torno de uma previsão específica e mede os resultados, ele também lida com entradas irregulares. Isso acaba sendo um benefício em termos de interpretabilidade, porque pode-se perturbar a entrada alterando componentes que fazem sentido para humanos, por exemplo, palavras ou partes de uma imagem, mesmo que o modelo esteja usando componentes muito mais complicados como recursos (ex., incorporação de palavras) (Ribeiro et al., 2016a).

O LIME assume um modelo de aprendizado de máquina de caixa preta e investiga a relação entre entrada e saída, representada pelo modelo, como pode-se observar na Figura 3.4.

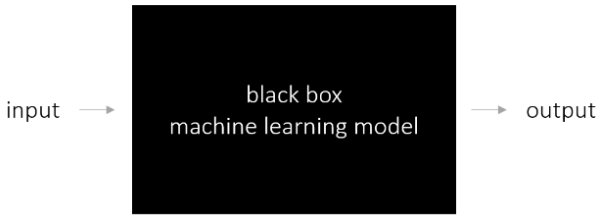

Figura 3.4. Modelo de aprendizado de máquina de caixa preta [Autor].

As abordagens específicas do modelo visam entender o modelo de aprendizado de máquina do modelo preto analisando os componentes internos e como eles interagem. Em modelos de aprendizado profundo, é possível, por exemplo, investigar unidades de ativação e vincular ativações internas de volta à entrada. Isso requer uma compreensão completa da rede e não se adapta a outros modelos (Hulstaert, 2018).

O LIME fornece interpretabilidade do modelo local. O LIME modifica uma única amostra de dados ajustando os valores do recurso e observa o impacto resultante na saída. Muitas vezes, isso também está relacionado ao que os humanos estão interessados ao observar a saída de um modelo. A pergunta mais comum é provavelmente: por que essa previsão foi feita ou quais variáveis causaram a previsão? (Hulstaert, 2018).

Outras técnicas de interpretabilidade do modelo apenas respondem à pergunta acima da perspectiva de todo o conjunto de dados. A importância do recurso explica em um nível de conjunto de dados quais recursos são importantes. Ele permite verificar hipóteses e se o modelo está superajustado ao ruído, mas é difícil diagnosticar previsões específicas do modelo (Hulstaert, 2018). O LIME tenta desempenhar o papel de "explicador", explicando as previsões para cada amostra de dados, como mostra a Figura 3.5.

3.3.1. Intuição por trás do LIME

Um requisito fundamental para a LIME é trabalhar com uma representação interpretável do input, que seja compreensível para os humanos. Exemplos de representações interpretáveis são, por exemplo, um vector BoW para PNL, ou uma imagem para visão por computador. Por outro lado, se o LIME trabalhar com incrustações densas ou não interpretáveis, provavelmente não melhorará a interpretabilidade (Ribeiro, 2021).

O output do LIME é uma lista de explicações, refletindo a contribuição de cada

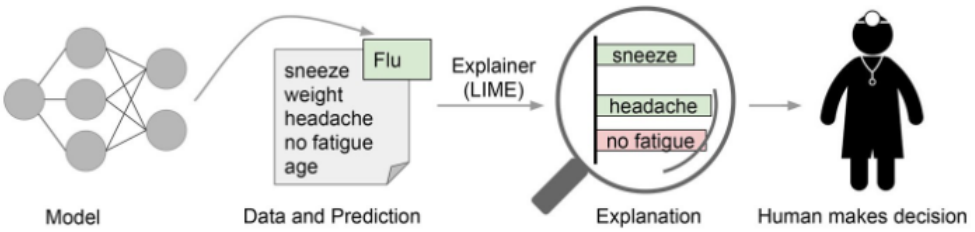

Figura 3.5. Explicando previsões individuais para um decisor humano (Ribeiro et al., 2016b).

recurso para a previsão de uma amostra de dados. Isso fornece interpretabilidade local e também permite determinar quais alterações de recursos terão mais impacto na prediction.output do LIME, é uma lista de explicações, refletindo a contribuição de cada recurso para a previsão de uma amostra de dados. Isso fornece interpretabilidade local e também permite determinar quais alterações de recursos terão mais impacto na previsão (Ribeiro, 2021).

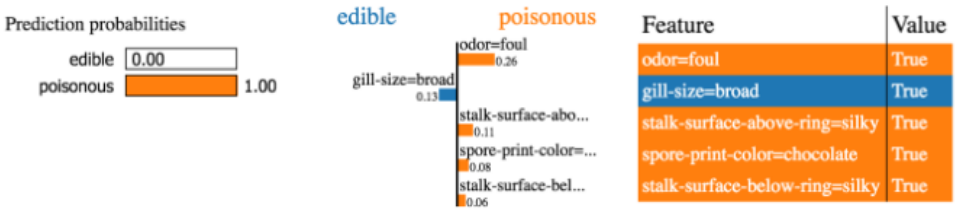

Figura 3.6. Um exemplo de LIME aplicado a um problema clássico de classificação (Ribeiro et al., 2016c).

Uma explicação é criada aproximando o modelo subjacente localmente por um modelo interpretável. Modelos interpretáveis são, por exemplo, modelos lineares com forte regularização, árvores de decisão, etc. Os modelos interpretáveis são treinados em pequenas perturbações da instância original e devem fornecer apenas uma boa aproximação local. O "conjunto de dados" é criado, por exemplo, adicionando ruído a recursos contínuos, removendo palavras ou ocultando partes da imagem. Ao aproximar apenas a caixa preta localmente (na vizinhança da amostra de dados), a tarefa é significativamente simplificada (Ribeiro, 2021).

3.3.2. Implementações de LIME

Usa-se o LIME para explicar uma infinidade de classificadores, como florestas aleatórias, máquinas de vetor de suporte (SVM) e redes neurais, nos domínios de texto e imagem. Implementações dessas técnicas tornaram-se disponíveis recentemente à medida que bibliotecas de programação específicas foram desenvolvidas (Ribeiro et al., 2016a; Alber et al., 2019).

Para explicações de imagem, uma determinada imagem é segmentada algoritmicamente em superpixels, e a relevância de cada superpixel para uma determinada classificação é determinada usando um modelo linear. O algoritmo é independente de modelo, pois requer apenas as saídas do classificador para imagens diferentes. De fato,

o LIME pode ser usado para qualquer sistema de classificação de imagens, não apenas redes neurais, pois não emprega procedimentos específicos de retropropagação ou etapas específicas para qualquer tipo de modelo individual (Ribeiro et al., 2016b).

A Figura 3.7 mostra um exemplo de como o LIME funciona para classificação de imagens. Imagine que queremos explicar um classificador que prevê a probabilidade de a imagem conter um sapo. Pegamos a imagem à esquerda e a dividimos em componentes interpretáveis, ou seja, superpixels contíguos (Ribeiro et al., 2016b).

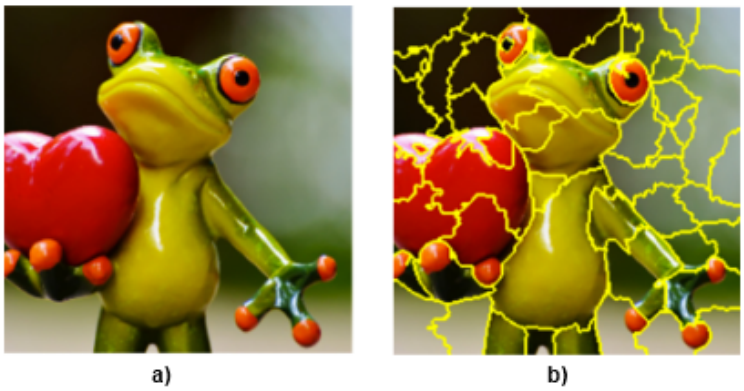

Figura 3.7. Transformando uma imagem em componentes interpretáveis. a) Imagem Original. b) Componentes Interpretáveis (Ribeiro et al., 2016b).

Gera-se um conjunto de dados de instâncias perturbadas desligando alguns dos componentes interpretáveis, neste caso, tornando-os cinza, como mostra a Figura 3.8. Para cada instância perturbada, obtem-se a probabilidade de que uma perereca esteja na imagem de acordo com o modelo. Em seguida, aprendem-se um modelo simples, linear, nesse conjunto de dados, que é ponderado localmente, ou seja, preocupa-se mais em cometer erros em instâncias perturbadas que são mais semelhantes à imagem original. No final, apresenta-se os superpixels com maiores pesos positivos como explicação, esmaecendo todo o resto (Ribeiro et al., 2016b).

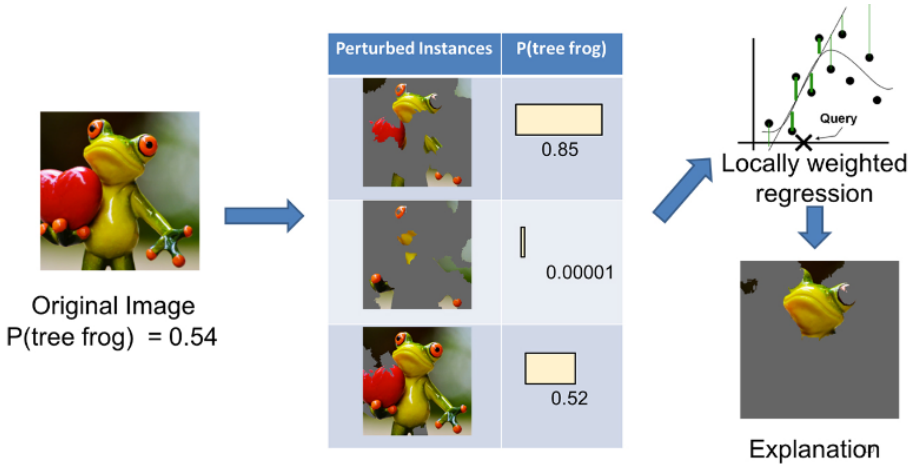

Figura 3.8. Previsão com LIME (Ribeiro et al., 2016b).

No contexto de classificação de texto, isso significa que algumas das palavras

são, por exemplo, substituídas, para determinar quais elementos da entrada afetam as previsões.

3.3.3. Desvantagens do LIME

Embora a ideia geral do LIME pareça fácil, existem algumas desvantagens em potencial. Na implementação atual, apenas modelos lineares são usados para aproximar o comportamento local. Até certo ponto, essa suposição está correta quando se observa uma região muito pequena ao redor da amostra de dados. Ao expandir essa região, no entanto, é possível que um modelo linear não seja poderoso o suficiente para explicar o comportamento do modelo original. A não linearidade em regiões locais acontece para os conjuntos de dados que exigem modelos complexos e não interpretáveis. Não ser capaz de aplicar o LIME nestes cenários é uma armadilha significativa (Hulstaert, 2018).

Em segundo lugar, o tipo de modificações que precisam ser realizadas nos dados para obter as explicações adequadas geralmente são específicos do caso de uso. Segundo Ribeiro et al. (2016a) um modelo que prevê imagens com tom sépia a serem retro não pode ser explicado pela presença ou ausência de super pixels.

Muitas vezes, simples perturbações não são suficientes. Idealmente, as perturbações seriam impulsionadas pela variação que é observada no conjunto de dados. Dirigir manualmente as perturbações do outro provavelmente não é uma ótima ideia, pois provavelmente introduziria viés nas explicações do modelo (Hulstaert, 2018).

3.4. *SHapley Additive exPlanations* (SHAP)

O SHAP é um método desenvolvido por Lundberg e Lee, em 2017, para explicar predições individuais, baseando-se nos valores de Shapley da teoria dos jogos. O método tenta interpretar de maneira mais direta as decisões dos modelos de inteligência artificial (IA), diferente das soluções caixa preta. Para isso, SHAP explica uma saída específica calculando a contribuição de cada característica (atributo) da entrada para predição (instância) (Burkart, Huber, 2021). Assim, SHAP indica as influências negativas ou positivas das características para um resultado do modelo. A Figura 3.9 ilustra o método, onde os atributos idade, gênero, pressão arterial e índice de massa corporal (IMC) têm valores de contribuições +0.4, -0.3, +0.1 e +0.1 para a saída apresentada.

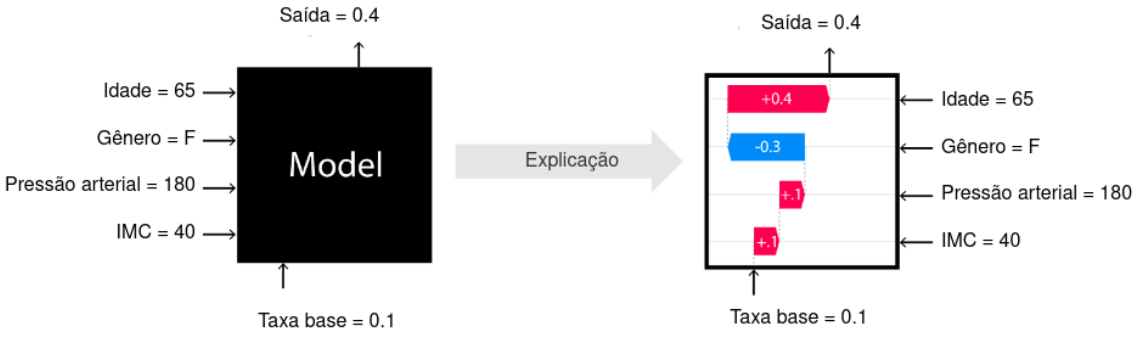

Figura 3.9. Ilustração da explicação com SHAP. Adaptação de Mitchell et al. (2020).

3.4.1. Base do SHAP: valores de Shapley

Para entender SHAP, é interessante observar como funciona o cálculo dos valores de Shapley. Esse método determina a contribuição de uma característica verificando a diferença entre a predição de um modelo com e sem essa característica. Como a ordem que o modelo analisa as características pode variar as contribuições dos atributos, faz-se necessário verificar todas as combinações (coalizões) possíveis. Então, um valor de Shapley é a contribuição marginal média de um atributo de uma instância entre todas as coalizões possíveis (López, 2021).

Na Figura 3.10, mostramos um exemplo prático do cálculo do valor de Shapley para o atributo A1. Nesse exemplo, organizamos seis coalizões e agrupamos essas coalizões sempre variando a presença de A1. Com a diferença das coalizões, obtemos a contribuição marginal (δ) de cada grupo. Por fim, calculamos a contribuição marginal média (ϕ), o valor de Shapley de A1.

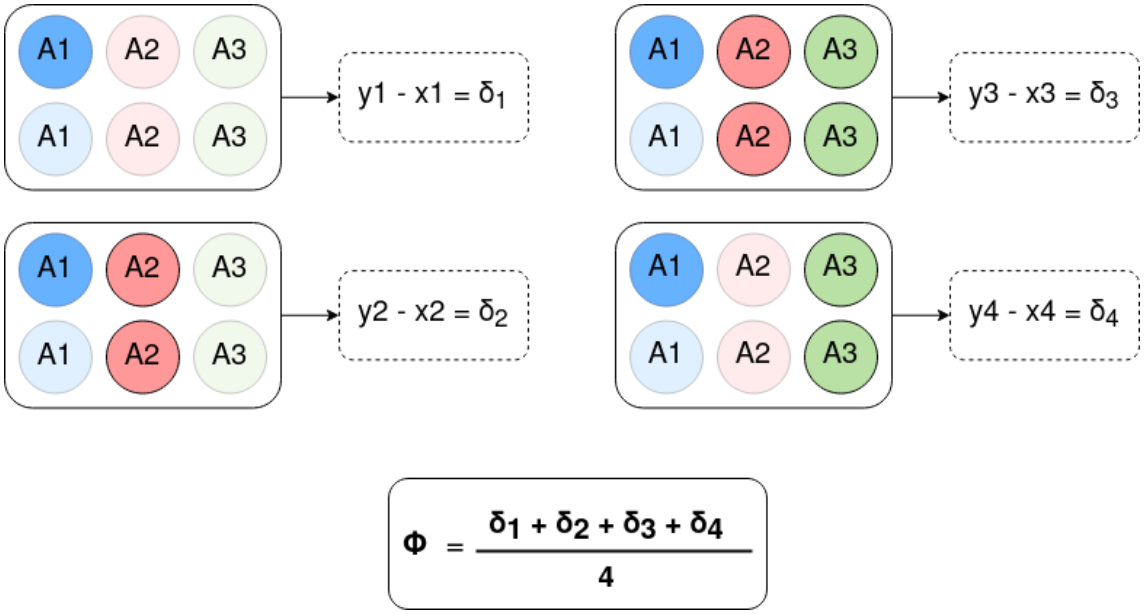

Figura 3.10. Exemplo: obtenção do valor de Shapley para o atributo A1.

A desvantagem dessa abordagem é calcular o valor exato de cada atributo, pois seu custo cresce exponencialmente de acordo com a quantidade de atributos (Bhatt et al., 2019). Uma das maneiras de otimizar esse cálculo é utilizando o SHAP.

3.4.2. Funcionamento do SHAP

O intuito do SHAP é explicar uma previsão feita a partir de uma instância, descobrindo o valor de Shapley de cada característica que resultou na predição. Essas características podem ser, por exemplo, um valor individual para dados tabulares ou um grupo de valores de característica para uma imagem. Além disso, SHAP cumpre três propriedades, já presente no método clássico de valores de Shapley, desejáveis em métodos de atribuição aditiva de atributos (Lundberg, Lee, 2017):

- Precisão local: determina a mesma saída para o modelo aproximado e o modelo

original;

- Ausência: garante nenhum impacto na saída para recursos ausentes;
- Consistência: afirma que, se um recurso do modelo aumenta ou permanece o mesmo independentemente dos outros recursos, o impacto desse recurso não deve diminuir.

Como vimos anteriormente, encontrar os valores de Shapley para um grande número de atributos pode ser um trabalho oneroso. Para evitar isso, Lundberg e Lee propuseram o Kernel SHAP, um método adaptado do LIME linear para calcular os valores de Shapley (Lundberg, Lee, 2017). O Kernel SAHP consegue obter esses valores com algumas amostras de coalizão, utilizando dados de treinamento do modelo e uma regressão linear ponderada onde os coeficientes da solução são os valores de Shapley.

Para cada amostra de coalizão, adquire-se sua predição e calculam-se os pesos W_C com a forma do *kernel* descrita na equação 1, onde C é índice da coalizão (López, 2021). Com essa equação, obtemos os pesos de uma regressão linear, os valores de Shapley.

$$W_C = \frac{N^o \text{ de atributos}}{\binom{N^o \text{ de coalizões}}{\text{do tamanho de } C} \times \left(N^o \text{ de atributos de } C\right) \times \binom{N^o \text{ de atributos}}{\text{ausentes em } C}} \tag{1}$$

Com esses pesos, entendemos a influência dos atributos. Quanto maior seu valor absoluto, mais importantes para predição. Para ter uma visão geral global dos resultados, calcula-se a média dos valores absolutos de Shapley.

Por fim, é importante acrescentar que o Kernel SHAP é agnóstico para o cálculo dos valores de Shapley, podendo ser aplicado a qualquer modelo de Aprendizado de Máquina. Contudo, existem outras versões de SHAP que possuem otimizações específicas para certos modelos, como os métodos Tree SHAP, Deep SHAP, Low-Order SHAP, Linear SHAP e Max SHAP (López, 2021).

3.5. Outras técnicas XAI

Atualmente, existem inúmeras técnicas para explicabilidade em Machine Learning, como: Anchors (Ribeiro et al., 2018), Partial Dependence Plots (PDP), Individual Condition Expectations plots (ICE) (Goldstein et al., 2015), Leave One Covariate Out (LOCO) (Molnar et al., 2020), Accumulated Local Effects (ALE) (Galkin et al., 2018), Deep Learning Important Features (DeepLIFT) (Shrikumar et al., 2017), Layer-wise relevance propagation (LRP) (Montavon et al., 2019), Contrastive Explanations Method (CEM) (Dhurandhar et al., 2018a), ProfWeight (Dhurandhar et al., 2018b), Permutation Feature Importance, entre outras. A seguir, uma visão geral de algumas dessas técnicas.

3.5.1. Anchors

A abordagem anchors foi construída pelos mesmos criadores do LIME. O método anchors explica as previsões individuais de um modelo usando regras IF-THEN facilmente compreensíveis - ”âncoras- que suportam (ancoram) as previsões suficientemente bem. Para

encontrar âncoras, os autores usam técnicas de reforço em combinação com um algoritmo de busca em grafo para explorar os conjuntos de perturbações em torno dos dados e seus efeitos nas previsões. Este método é agnóstico de modelo.

No artigo original (Ribeiro et al., 2018), os autores compararam o LIME com o Anchors e visualizaram como eles processam um modelo de classificador binário complexo (+ ou –) para chegar a um resultado. Conforme a Figura 3.11, a explicação do LIME funciona aprendendo um limite de decisão linear que melhor se aproxima do modelo, com alguma ponderação local, enquanto Âncoras adapta sua cobertura ao comportamento do modelo e deixa seus limites claros.

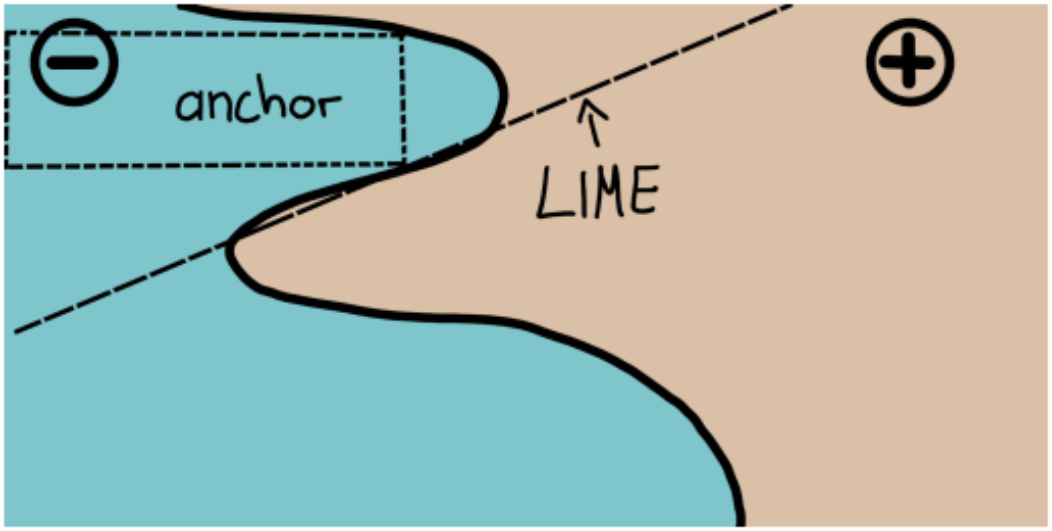

Figura 3.11. Comparação do método Anchors com o método LIME (Ribeiro et al., 2016c).

As âncoras também foram testadas em uma variedade de tarefas de Machine Learning, como classificação, geração de texto e previsões estruturadas.

3.5.2. Partial Dependence Plots (PDP)

O PDP obtém uma representação visual global de como um ou dois recursos influenciam o resultado previsto do modelo, com outros recursos mantidos constantes. O PDP informa se a relação entre o destino e o recurso escolhido é linear ou complexo. O PDP é independente de modelo.

3.5.3. Individual Conditional Expectations plots (ICE)

O ICE fornece uma representação visual local do efeito de um recurso no modelo em relação ao recurso de destino. Ao contrário do PDP, o ICE mostra previsões separadas da dependência do recurso com uma linha por amostra. Também é agnóstico de modelo (Goldstein et al., 2015).

3.5.4. Leave One Covariate Out (LOCO)

O Leave One Covariate Out é uma abordagem muito simplista. O LOCO deixa uma coluna de fora, treina novamente o modelo e, em seguida, calcula as diferenças de cada

modelo LOCO para a pontuação de previsão do modelo original. Se a pontuação mudar muito, a variável que ficou de fora deve ser importante. Dependendo da largura do modelo (quantidade de recursos), essa abordagem pode ser demorada (Molnar et al., 2020).

Existem algumas desvantagens que PDP, ICE e LOCO compartilham:

- Capturam diretamente as interações dos recursos;
- Podem ser muito aproximados, o que é potencialmente problemático para dados categóricos e codificação one-hot que é frequentemente usada no processamento de linguagem natural.

3.5.5. Layer-wise relevance propagation (LRP)

A propagação de relevância em camadas é semelhante ao DeepLIFT, ele faz a propagação para trás usando um conjunto de regras de propagação projetadas propositadamente a partir da saída, identificando os neurônios mais relevantes dentro da rede neural até você retornar à entrada. Assim, você obtém todos os neurônios (por exemplo, pixels que realmente contribuem para a saída. LRP funciona bem em CNNs e pode ser usado para explicar LSTMs (Montavon et al., 2019).

Confira esta demonstração interativa para ver como o LRP funciona.

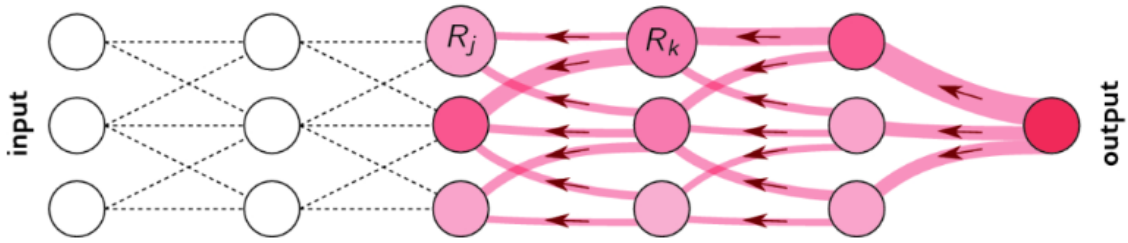

Figura 3.12. Representação visual de como o LRP faz retropropagação do nó de saída através dos neurônios da camada oculta até a entrada, identificando os neurônios que tiveram impacto na saída do modelo (Montavon et al., 2019).

3.5.6. ProfWeight

Em 2018, (Dhurandhar et al., 2018b) fez um artigo "Improving Simple Models with Confidence Profiles". O artigo propôs o método ProfWeight para explicabilidade do modelo. ProfWeight transfere a alta precisão de teste de uma rede neural profunda pré-treinada para uma rede rasa com baixa precisão de teste, como é visto na Figura 3.13.

Assim como um professor transferindo conhecimento para um aluno, ProfWeight utiliza sondas (pesos na amostra de acordo com a dificuldade da rede) para transferir conhecimento.

ProfWeight pode ser resumido em quatro etapas principais:

1. Anexe e treine sondas em representações intermediárias de uma rede neural de alto desempenho;

2. Treine um modelo simples no conjunto de dados original;

3. Aprenda pesos para exemplos no conjunto de dados em função do modelo simples e das sondas;

4. Treine novamente o modelo simples no conjunto de dados ponderado final.

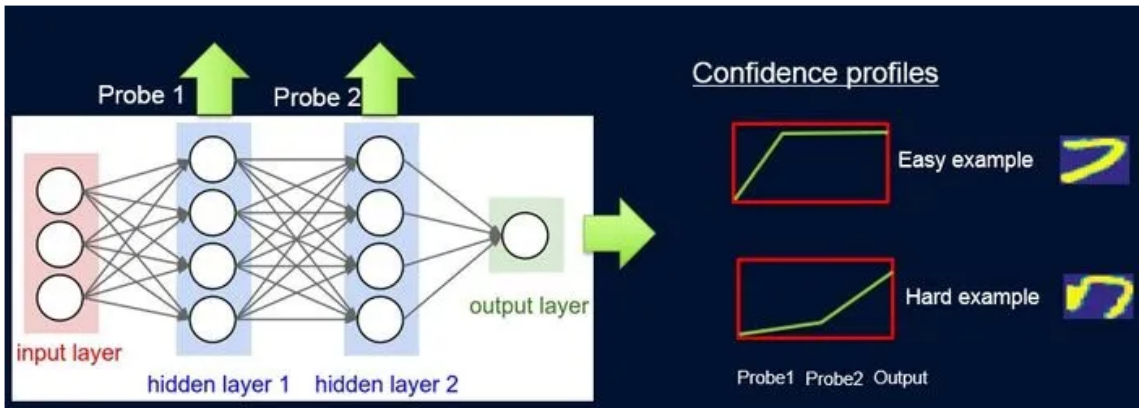

Figura 3.13. Representação visual do ProfWeight (Dhurandhar et al., 2018b).

3.6. Ferramentas e práticas

Nesta seção serão apresentadas as principais abordagens e trechos de código-fonte necessários para algumas implementações de técnicas XAI. Os trechos de código foram baseados nos algoritmos apresentados em [1] e [2]. Os autores criaram um *notebook* público no ambiente Google Colaboratory com todos os códigos abaixo exibidos, além de outros não listados neste documento. Para a execução, basta executar as células em sequência. A permissão do *notebook* é de visualização, mas podem ser realizadas cópias e, a partir de então, essas cópias poderão ser editadas e modificadas. Os autores recomendam que façam esse procedimento, pois é de suma importância que realizem diversas modificações nos códigos para compreender como que os algoritmos atuam à medida que trocamos parâmetros, valores *etc*. O link para o notebook está em [3].

3.6.1. Partial Dependence Plots - PDP

Gráficos de dependência parcial (*Partial Dependence Plots* - PDP) e gráficos de expectativa condicional individual (*Individual Conditional Expectation* - ICE) são imagens que podem ser utilizadas para facilitar a compreensão dos dados e analisar a interação entre a resposta alvo e um conjunto de recursos de entrada de interesse.

[1] https://scikit-learn.org/stable/modules/partial_dependence.html

[2] https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html

[3] https://colab.research.google.com/drive/1Fh8xmXzqMRR6aDR5lWLztC63VCuFH1t6?usp=sharing

Dessa forma, pode-se dizer que o PDP é uma técnica que fornece uma representação visual global de como um ou dois recursos influenciam o resultado previsto do modelo. A técnica também informa se a relação entre o destino e o recurso escolhido é linear ou complexo.

Os módulos de inspeção de aprendizado do Scikit fornecem uma função para gráfico de dependência parcial chamada *plot_partial_dependence* que cria um gráfico de dependência parcial unidirecional e bidirecional. No trecho de código a seguir é criado uma grade de gráficos de dependência parcial: dois PDPs unidirecionais para os recursos 0 e 1 e um PDP bidirecional entre os dois recursos. Como base de dados, são utilizados os dados de habitação da Califórnia-EUA. Esse conjunto de dados consiste em 20.640 quarteirões de casas em toda a Califórnia em 1990, e o objetivo é prever o logaritmo natural do preço médio das casas a partir de 8 recursos diferentes:

- MedInc - renda mediana no grupo de blocos
- HouseAge - idade média da casa no grupo de quarteirões
- AveRooms - número médio de quartos por família
- AveBedrms - número médio de quartos por domicílio
- Population - população do grupo de blocos
- AveOccup - número médio de membros da família
- Latitude - latitude do grupo de blocos
- Longitude - longitude do grupo de blocos

O código adiante carrega o banco de dados do conjunto habitacional de California-EUA:

```
import pandas as pd
import shap
import sklearn

# a classic housing price dataset
X,y = shap.datasets.california(n_points=1000)

X100 = shap.utils.sample(X, 100) # 100 instances for use as the
    ↪ background distribution

# a simple linear model
model = sklearn.linear_model.LinearRegression()
model.fit(X, y)
```

Em seguida, podemos examinar alguns coeficientes dos dados. A maneira mais comum de entender um modelo linear é examinar os coeficientes aprendidos para cada característica. Esses coeficientes nos dizem o quanto a saída do modelo muda quando alteramos cada um dos recursos de entrada:

```
print("Model coefficients:\n")
for i in range(X.shape[1]):
    print(X.columns[i], "=", model.coef_[i].round(5))
```

```
OUTPUT:

Model coefficients:

MedInc = 0.45769
HouseAge = 0.01153
AveRooms = -0.12529
AveBedrms = 1.04053
Population = 5e-05
AveOccup = -0.29795
Latitude = -0.41204
Longitude = -0.40125
```

Para entender a importância de um determinado recurso em um modelo, é necessário compreender como a alteração desse recurso afeta a saída do modelo e também a distribuição dos valores desse recurso. Para visualizar isso para um modelo linear, podemos construir um gráfico de dependência parcial clássico e mostrar a distribuição de valores de recursos como um histograma no eixo x. O código adiante faz esse processo:

```
shap.partial_dependence_plot(
    "MedInc", model.predict, X100, ice=False,
    model_expected_value=True, feature_expected_value=True
)
```

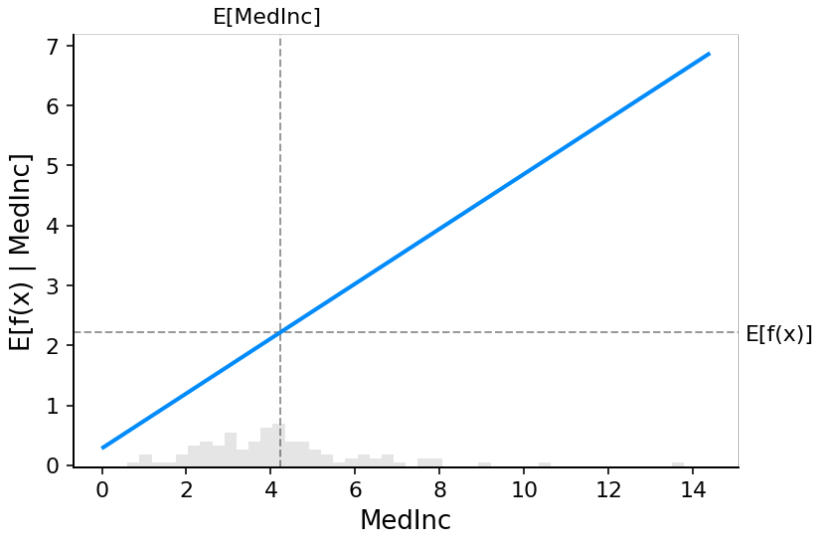

Figura 3.14. PDP do modelo em análise

3.6.2. SHAP

Para manipularmos na prática a técnica SHAP, inicialmente precisamos instalar o SHAP. O modo padrão para realizar tal procedimento no Python é conforme abaixo:

```
pip install shap
```

Uma das propriedades fundamentais dos valores de Shapley é que eles sempre se resumem à diferença entre o resultado do "jogo" quando todos os jogadores estão presentes e o resultado do jogo quando nenhum jogador está presente. Para modelos de aprendizado de máquina, isso significa que os valores SHAP de todos os recursos de entrada sempre somarão a diferença entre a saída do modelo de linha de base (esperada) e a saída do modelo atual para a previsão explicada. A maneira mais fácil de ver isso é por meio de um gráfico em cascata que começa na expectativa anterior de fundo para um preço de casa e, em seguida, adiciona recursos um de cada vez até atingirmos a saída do modelo atual. O trecho de código adiante produz a imagem esperada (Figura 3.15).

```
# the waterfall_plot shows how we get from shap_values.base_values to
    ↪ model.predict(X)[sample_ind]
shap.plots.waterfall(shap_values[sample_ind], max_display=14)
```

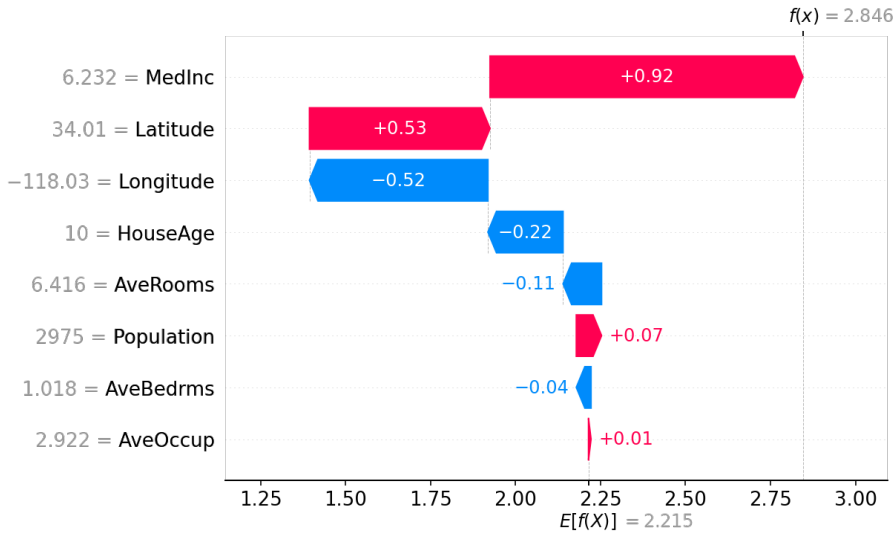

Figura 3.15. SHAP do modelo de casas - Califórnia

A partir de agora, usaremos o conhecido conjunto de dados de espécies Iris para ilustrar como o SHAP pode explicar a saída de muitos tipos de modelos diferentes. O conjunto de dados possui apenas 150 amostras e foram utilizados 130 de um conjunto aleatório para treinamento e 20 para teste dos modelos. Esta prática foi adaptada de [4]. Inicialmente, será realizada a coleta do banco de dados para subsequente preparo.

```
import sklearn
from sklearn.model_selection import train_test_split
import numpy as np
import shap
import time
```

[4] https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/model_agnostic/Iris%20classification%20with%20scikit-learn.html

```
X_train,X_test,Y_train,Y_test = train_test_split(*shap.datasets.iris(),
    ↪  test_size=0.2, random_state=0)

# rather than use the whole training set to estimate expected values,
    ↪ we could summarize with
# a set of weighted kmeans, each weighted by the number of points they
    ↪ represent. But this dataset
# is so small we don't worry about it
#X_train_summary = shap.kmeans(X_train, 50)

def print_accuracy(f):
    print("Accuracy = {0}%".format(100*np.sum(f(X_test) == Y_test)/len(
        ↪ Y_test)))
    time.sleep(0.5) # to let the print get out before any progress bars

shap.initjs()
```

Em seguida, será apresentado o resultado da técnica SHAP para o modelo K-nearest neighbors aplicado aos dados da íris.

```
knn = sklearn.neighbors.KNeighborsClassifier()
knn.fit(X_train, Y_train)

print_accuracy(knn.predict)
```

Na Figura 3.16 é apresentado visualmente o efeito que a previsão do KNN gerou no modelo.

```
explainer = shap.KernelExplainer(knn.predict_proba, X_train)
shap_values = explainer.shap_values(X_test.iloc[0,:])
shap.initjs()
shap.force_plot(explainer.expected_value[0], shap_values[0], X_test.
    ↪ iloc[0,:])
```

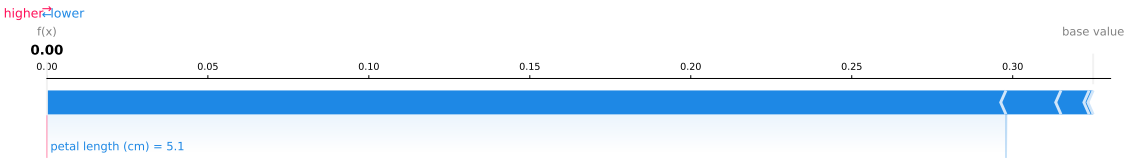

Figura 3.16. SHAP do modelo Íris - explicando uma única previsão do conjunto

Para identificarmos os efeitos combinados dos atributos, com resultados dinâmicos à medida que movemos o mouse sobre a imagem gerada pelo SHAP (imagem obtida diretamente no terminal), podemos fazer uso do método `shap.force_plot`. Na Figura 3.18 é apresentada uma imagem estática do que seria essa interação dinâmica que a técnica proporciona.

```
shap_values = explainer.shap_values(X_test)
shap.initjs()
shap.force_plot(explainer.expected_value[0], shap_values[0], X_test)
```

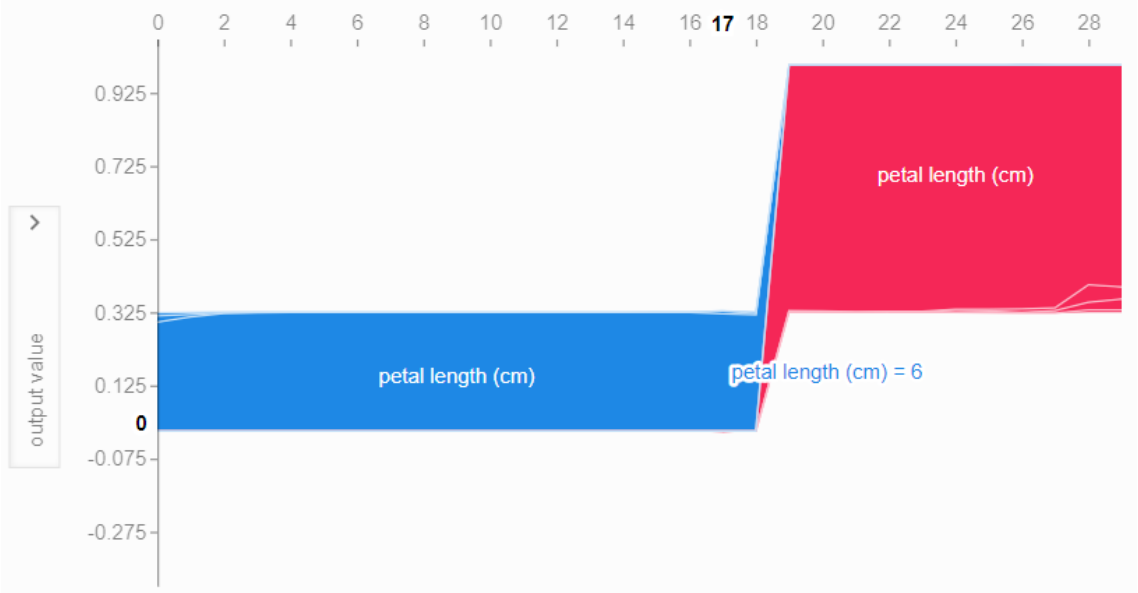

Figura 3.17. SHAP do modelo Íris - KNN

Agora veremos o efeito do modelo Support vector machine com um kernel linear sobre os dados da íris. Para tal, podemos utilizar o seguinte trecho de código:

```
svc_linear = sklearn.svm.SVC(kernel='linear', probability=True)
svc_linear.fit(X_train, Y_train)
print_accuracy(svc_linear.predict)

# explain all the predictions in the test set
explainer = shap.KernelExplainer(svc_linear.predict_proba, X_train)
shap_values = explainer.shap_values(X_test)
shap.force_plot(explainer.expected_value[0], shap_values[0], X_test)
```

3.7. Conclusão

O aprendizado de máquina tem um grande potencial para melhorar produtos, processos e pesquisas, entretanto, os sistemas computacionais por si geralmente não explicam suas previsões, o que é uma barreira para a adoção do aprendizado de máquina por diversos setores. Este minicurso tratou de forma introdutória sobre como tornar os modelos de aprendizado de máquina e suas decisões interpretáveis.

Depois de explorar os conceitos de interpretabilidade, são discutidos modelos simples e interpretáveis, como árvores de decisão, regras de decisão e estratégias para interpretar modelos de caixa preta, como a importância de recursos e efeitos locais acumulados, e explicar previsões individuais com valores de Shapley e LIME.

Referências

Alber Maximilian, Lapuschkin Sebastian, Seegerer Philipp, Hägele Miriam, Schütt Kristof T, Montavon Grégoire, Samek Wojciech, Müller Klaus-Robert, Dähne Sven, Kindermans Pieter-Jan. iNNvestigate neural networks! // J. Mach. Learn. Res. 2019. 20,

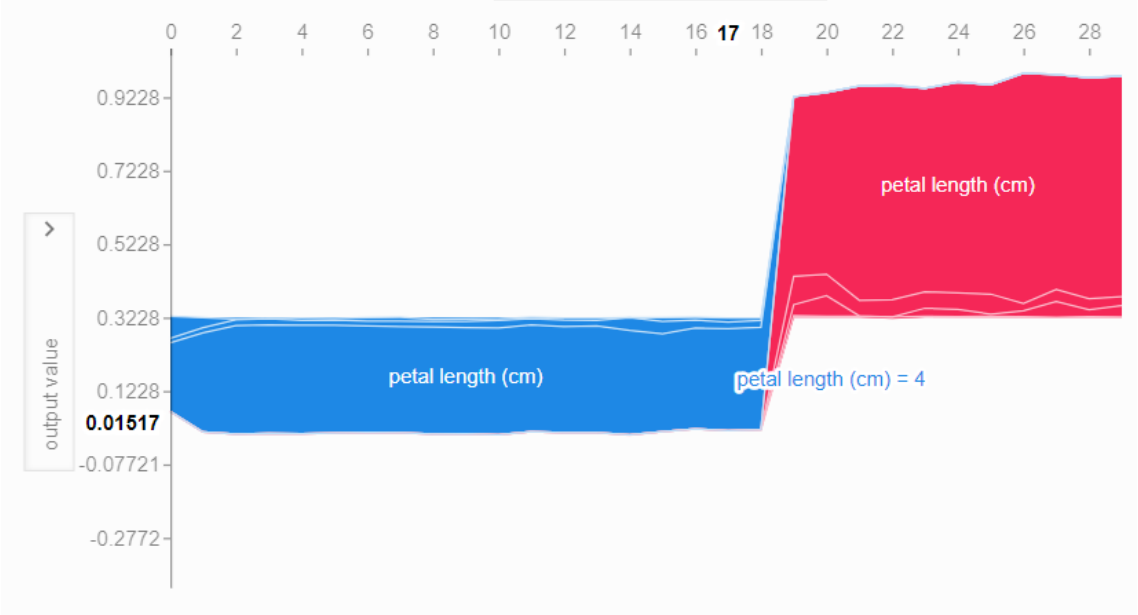

Figura 3.18. SHAP do modelo Íris com SVM

93. 1–8.

Angelov Plamen, Soares Eduardo. Towards explainable deep neural networks (xDNN) // Neural Networks. 2020. 130. 185–194.

Araújo Inaldo da Paixão Santos. Introdução à auditoria operacional. 2001.

Berghoff Christian, Biggio Battista, Brummel Elisa, Danos Vasilios, Doms Thomas, Ehrich Heiko, Gantevoort Thorsten, Hammer Barbara, Iden Joachim, Jacob Sven, Khlaaf Heidy, Komrowski Lars, Kröwing Robert, Metzen Jan, Neu Matthias, Petsch Fabian, Poretschkin Maximilian, Samek Wojciech, Schäbe Hendrik, Twickel Arndt, Vechev Martin, Wiegand Thomas. Towards Auditable AI Systems: Current status and future directions [White paper]. May 2021.

Bhatt Umang, Xiang Alice, Sharma Shubham, Weller Adrian, Taly Ankur, Jia Yunhan, Ghosh Joydeep, Puri Ruchir, Moura José M. F., Eckersley Peter. Explainable Machine Learning in Deployment. 2019.

Burkart Nadia, Huber Marco F. A survey on the explainability of supervised machine learning // Journal of Artificial Intelligence Research. 2021. 70. 245–317.

Costa Lucas Tubino Bonifácio Costa. Análise de modelos explicáveis de sistemas de classificação para registros hidroacústicos em ambientes marítimos. 2019.

Dhurandhar Amit, Chen Pin-Yu, Luss Ronny, Tu Chun-Chen, Ting Paishun, Shanmugam Karthikeyan, Das Payel. Explanations based on the missing: Towards contrastive explanations with pertinent negatives // Advances in neural information processing systems. 2018a. 31.

Dhurandhar Amit, Shanmugam Karthikeyan, Luss Ronny, Olsen Peder A. Improving simple models with confidence profiles // Advances in Neural Information Processing Systems. 2018b. 31.

Doshi-Velez Finale, Kim Been. Towards a rigorous science of interpretable machine learning // arXiv preprint arXiv:1702.08608. 2017.

Europa.eu . Official journal of the european union:Regulations. 2017.

Freitas Alex A. Comprehensible classification models: a position paper // ACM SIGKDD explorations newsletter. 2014. 15, 1. 1–10.

Galkin Fedor, Aliper Aleksandr, Putin Evgeny, Kuznetsov Igor, Gladyshev Vadim N, Zhavoronkov Alex. Human microbiome aging clocks based on deep learning and tandem of permutation feature importance and accumulated local effects // BioRxiv. 2018. 507780.

Goldstein Alex, Kapelner Adam, Bleich Justin, Pitkin Emil. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation // journal of Computational and Graphical Statistics. 2015. 24, 1. 44–65.

Goodman Bryce, Flaxman Seth. Eu regulations on algorithmic decision-making and a" right to explanation" // arXiv preprint arxiv:1606.08813. 2016.

Goodman Bryce, Flaxman Seth. European Union regulations on algorithmic decision-making and a "right to explanation" // AI magazine. 2017. 38, 3. 50–57.

Gunning David. Explainable artificial intelligence (XAI), Defense Advanced Research Projects Agency (DARPA), 2017. 2017.

Holzinger Andreas, Langs Georg, Denk Helmut, Zatloukal Kurt, Müller Heimo. Causability and explainability of artificial intelligence in medicine // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2019. 9, 4. e1312.

Hulstaert Lars. Understanding model predictions with LIME. VII 2018.

Hummer Waldemar, Muthusamy Vinod, Rausch Thomas, Dube Parijat, El Maghraoui Kaoutar, Murthi Anupama, Oum Punleuk. Modelops: Cloud-based lifecycle management for reliable and trusted ai // 2019 IEEE International Conference on Cloud Engineering (IC2E). 2019. 113–120.

Lipton Zachary C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. // Queue. 2018. 16, 3. 31–57.

Lundberg Scott M., Lee Su-In. A Unified Approach to Interpreting Model Predictions // Proceedings of the 31st International Conference on Neural Information Processing Systems. Red Hook, NY, USA: Curran Associates Inc., 2017. 4768–4777. (NIPS'17).

López Fernando. SHAP: Shapley Additive Explanations. 2021. Online; acessado em Jul 2022.

Martens David, Baesens BB, Van Gestel Tony. Decompositional rule extraction from support vector machines by active learning // IEEE Transactions on Knowledge and Data Engineering. 2008. 21, 2. 178–191.

Mitchell Rory, Frank Eibe, Holmes Geoffrey. GPUTreeShap: Massively Parallel Exact Calculation of SHAP Scores for Tree Ensembles. 2020.

Molnar C, Gruber S, Kopper P. Limitations of interpretable machine learning methods. 2020.

Molnar Christoph. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. 2020.

Montavon Grégoire, Binder Alexander, Lapuschkin Sebastian, Samek Wojciech, Müller Klaus-Robert. Layer-wise relevance propagation: an overview // Explainable AI: interpreting, explaining and visualizing deep learning. 2019. 193–209.

Ribeiro Marco Tulio. LIME. VII 2021.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. "Why should i trust you?"Explaining the predictions of any classifier // Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016a. 1135–1144.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. Local Interpretable Model-Agnostic Explanations (LIME): An Introduction. VIII 2016b.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. Model-agnostic interpretability of machine learning // arXiv preprint arXiv:1606.05386. 2016c.

Ribeiro Marco Tulio, Singh Sameer, Guestrin Carlos. Anchors: High-precision model-agnostic explanations // Proceedings of the AAAI conference on artificial intelligence. 32, 1. 2018.

Selvaraju Ramprasaath R, Cogswell Michael, Das Abhishek, Vedantam Ramakrishna, Parikh Devi, Batra Dhruv. Grad-cam: Visual explanations from deep networks via gradient-based localization // Proceedings of the IEEE international conference on computer vision. 2017. 618–626.

Senft Sandra, Gallegos Frederick. Information technology control and audit. 2008.

Shrikumar Avanti, Greenside Peyton, Kundaje Anshul. Learning important features through propagating activation differences // International conference on machine learning. 2017. 3145–3153.

Singh Jatinder, Cobbe Jennifer, Norval Chris. Decision provenance: Harnessing data flow for accountable systems // IEEE Access. 2018. 7. 6562–6574.

Toreini Ehsan, Aitken Mhairi, Coopamootoo Kovila, Elliott Karen, Zelaya Carlos Gonzalez, Van Moorsel Aad. The relationship between trust in AI and trustworthy machine learning technologies // Proceedings of the 2020 conference on fairness, accountability, and transparency. 2020. 272–283.