Relatório <11> - <Prática: Predição e a Base de Aprendizado de Máquina (II))>

<Jonas Correia>

1. [Activity] Linear Regression

Análise de regressão é um conceito central em ciência de dados e estatística, representando qualquer método que tente ajustar uma função a um conjunto de observações para realizar previsões. Em sua forma mais básica, a regressão linear encaixa uma linha reta nos dados, permitindo fazer previsões sobre valores futuros. Por exemplo, ao medir altura e peso de um grupo de pessoas, nota-se que esses pontos geralmente exibem uma relação linear, o que sugere que podemos ajustar uma linha para predizer a altura de uma pessoa com base no seu peso.

Para calcular a linha de melhor ajuste, utiliza-se a técnica de mínimos quadrados ordinários (OLS), que minimiza a soma dos erros quadráticos entre cada ponto de dados e a linha estimada. Esse método proporciona a linha que melhor se ajusta aos dados observados. Calcula-se a inclinação da linha como a correlação entre as variáveis multiplicada pelo desvio padrão de y dividido pelo de x. A interseção é obtida pela média de y menos o produto da inclinação pela média de x. Em Python, pacotes como o SciPy simplificam essas operações.

O valor R^2 ou coeficiente de determinação, mede a qualidade do ajuste, variando de 0 a 1, onde valores próximos de 1 indicam um bom ajuste, capturando a maior parte da variação dos dados. Essa medida auxilia a avaliar a precisão de um modelo e permite compará-lo com outras abordagens.

Técnicas alternativas como descida de gradiente também podem ajustar modelos lineares, especialmente em dados tridimensionais ou de alta dimensão, embora o OLS permaneça eficiente para maioria dos casos unidimensionais. A regressão linear é uma introdução simples, porém poderosa, às análises de predição, preparando terreno para técnicas mais complexas em modelagem preditiva.

2. [Activity] Polynomial Regression

O vídeo aborda a transição da regressão linear para a regressão polinomial, explicando que, embora a linear seja útil para modelar dados que seguem uma relação reta, há muitos casos onde os dados exibem uma curva. Nesse contexto, a regressão polinomial se torna relevante, ao permitir a modelagem com polinômios de graus superiores. Na regressão linear, a fórmula é $y = mx + b$, onde m e b são os parâmetros estimados. Porém, ao adotar uma regressão polinomial de segundo grau, a fórmula se torna $y = ax^2 + bx + c$, o que permite modelar dados com uma leve curvatura. Para graus ainda mais altos,

como o terceiro, a fórmula incluiria termos cúbicos, expandindo as possibilidades de curvatura.

A escolha do grau do polinômio, no entanto, deve ser cuidadosa. Um polinômio de grau muito alto pode levar a um ajuste exagerado, ou "overfitting", o que significa que a curva se adapta demais aos pontos de dados observados, capturando flutuações irrelevantes e prejudicando a capacidade de prever novos dados. Este fenômeno é ilustrado com exemplos práticos onde um polinômio de grau muito alto pode gerar uma curva que "ziguezagueia" para ajustar a todos os pontos, mesmo quando esses não seguem uma tendência real. Nesse sentido, é importante balancear o grau do polinômio para representar a tendência dos dados sem exagero.

O vídeo destaca o uso da métrica R-quadrado como uma ferramenta para avaliar a qualidade do ajuste, embora ela apenas indique quão bem o modelo se adapta aos dados de treinamento. A visualização e uma análise intuitiva do gráfico também são recomendadas para verificar se o ajuste reflete a tendência real dos dados, ao invés de aberrações. Para facilitar a experimentação, o código sugerido utiliza a função `polyfit` do NumPy, que permite ajustar polinômios de diferentes graus aos dados e visualizar os efeitos do ajuste com um gráfico de dispersão e a curva prevista.

3. [Activity] Multiple Regression, and Predicting Car

Na videoaula, foi abordado o conceito de regressão múltipla, sendo uma extensão da regressão linear simples para incluir mais de uma variável independente. O instrutor explicou que esse tipo de regressão é utilizado quando se deseja prever um valor com base em múltiplas características, em vez de apenas uma. Um exemplo prático é a previsão do preço de um carro usando diferentes atributos como quilometragem, idade, número de cilindros e portas. Cada um desses atributos representa uma variável independente, que, ao ser combinada, fornece uma previsão mais detalhada do valor de venda.

A distinção entre regressão múltipla e regressão multivariada foi mencionada: enquanto a regressão múltipla utiliza várias variáveis independentes para prever um único valor, a regressão multivariada prevê mais de um valor simultaneamente. Por exemplo, além do preço, poderíamos tentar prever o tempo estimado para vender o carro. Esse tipo de análise pode tornar o modelo mais complexo, ao requerer múltiplos resultados simultâneos, usando um conjunto de variáveis independentes.

A técnica comum para treinar o modelo de regressão múltipla, chamada de Ordinary Least Squares (OLS) ou mínimos quadrados ordinários, ajusta a linha de melhor ajuste minimizando o erro quadrático médio entre os valores reais e os previstos. Na implementação prática, o pacote statsmodels oferece uma função OLS que facilita a construção do modelo, inclusive normalizando as variáveis

para que todas as características estejam em uma escala comparável, melhorando a precisão e estabilidade do modelo. Uma análise de cada coeficiente também foi sugerida para entender a importância relativa de cada característica e auxiliar na seleção de variáveis mais relevantes para o modelo.

4. Multi-Level Models

Os modelos multiníveis (também conhecidos como modelos hierárquicos ou de efeitos mistos) são uma abordagem usada para lidar com dados que possuem uma estrutura hierárquica ou agrupada. Isso significa que os dados podem ser organizados em vários níveis de análise, onde os fatores de um nível afetam os de outro, criando interdependências complexas.

Por exemplo, imagine que você está analisando o desempenho de alunos em uma prova. Esses alunos estão agrupados em turmas, que por sua vez estão em escolas, em distritos escolares. Nesse caso, o desempenho individual de um aluno pode ser afetado por fatores específicos de sua própria experiência (nível individual), fatores relacionados à turma em que ele está inserido (nível da turma), fatores da escola (nível da escola) e até fatores maiores, como políticas educacionais de um distrito (nível distrital).

Em um modelo simples, talvez você tentasse prever o desempenho de um aluno apenas com base nas características pessoais dele, como horas de estudo ou QI. Mas um modelo multinível reconhece que existem influências tanto no nível do aluno quanto em níveis mais altos (por exemplo, a qualidade do ensino na escola ou o ambiente familiar) que precisam ser consideradas.

5. Supervised vs. Unsupervised Learning, and Train/Test

A aprendizagem de máquina envolve algoritmos que aprendem a partir de dados observacionais para fazer previsões. Esse processo pode ser dividido em dois tipos principais: a aprendizagem supervisionada e a não supervisionada. Na primeira, o modelo é treinado com dados rotulados, ou seja, acompanhados de respostas corretas, para que possa fazer previsões sobre novos dados. Já na aprendizagem não supervisionada, o modelo tenta encontrar padrões ou grupos nos dados sem ter acesso a essas respostas previamente. Um exemplo é quando um algoritmo agrupa objetos com base em suas semelhanças, sem uma categorização definida previamente.

Um conceito essencial para avaliar o desempenho de modelos supervisionados é o "Train/Test", onde os dados são divididos em dois grupos: um para treinar o modelo e outro para testá-lo. Dessa forma, é possível verificar se o modelo consegue generalizar seu aprendizado para novos dados, evitando o problema de overfitting, que ocorre quando o modelo se ajusta muito aos dados de treinamento e perde sua capacidade de fazer previsões precisas em novos dados.

Para tornar a avaliação mais robusta, existe a validação cruzada K-fold, uma técnica que divide os dados em várias partes, treinando e testando o modelo repetidamente em diferentes subconjuntos dos dados. Isso melhora a precisão da avaliação, garantindo que o modelo não esteja se ajustando apenas a um conjunto específico de dados. Assim, a aprendizagem de máquina abrange desde métodos simples de regressão até abordagens mais complexas para assegurar que os modelos possam prever com precisão novos resultados.

6. [Activity] Using Train/Test to Prevent Overfitting a Polynomial Regression

Na aula foi discutido o uso de um conjunto de dados para realizar uma regressão polinomial, aplicando o "train-test split" para encontrar o grau de polinômio que melhor se ajusta aos dados. A regressão foi apresentada como uma forma de aprendizado supervisionado, e os dados simulados representaram uma relação exponencial entre velocidades pagas e valores de compra, gerados aleatoriamente. Após dividir os dados, 80% foram usados para treinamento e 20% para teste, garantindo que ambos os conjuntos mantivessem a representatividade do total.

Em um exemplo de ajuste de modelo, foi aplicado um polinômio de oitava ordem aos dados de treinamento. Esse ajuste resultou em um "overfitting" evidente: o modelo ajustou-se excessivamente aos dados de treinamento, mas demonstrou baixa precisão para prever novos valores. A métrica R^2 para o conjunto de dados de teste apresentou um valor baixo, indicando um desempenho insatisfatório fora do conjunto de treinamento.

Essa prática do "train-test split" é essencial para garantir que um modelo consiga generalizar adequadamente para dados novos, evitando que ele seja altamente específico apenas para os dados de treinamento.

7. Bayesian Methods: Concepts

O classificador de spam em e-mails é uma aplicação interessante de técnicas de aprendizado de máquina, especificamente do método conhecido como Naive Bayes, que se fundamenta no Teorema de Bayes. Essa abordagem permite determinar a probabilidade de um e-mail ser classificado como spam com base em características observadas em mensagens anteriores. O Teorema de Bayes calcula a probabilidade de um evento, considerando a informação prévia disponível.

Para construir um classificador de spam, o modelo é treinado utilizando um conjunto de e-mails já identificados como spam ou não spam. Por exemplo, ao analisar um e-mail que contém a palavra "grátis", o classificador utiliza o Teorema de Bayes para calcular a probabilidade de que esse e-mail seja spam, considerando quantas vezes a palavra aparece em mensagens de spam comparado ao total de e-mails analisados.

No entanto, para o classificador ser eficaz, ele deve considerar não apenas uma única palavra, mas todas as palavras presentes no e-mail. O modelo deve conseguir analisar cada termo e determinar sua contribuição para a probabilidade geral de o e-mail ser spam. Essa análise é facilitada por ferramentas como o Scikit-learn em Python, que permite dividir e processar as palavras eficientemente. A técnica é chamada de "Naive" porque assume que as palavras são independentes entre si, simplificando o cálculo, embora essa suposição nem sempre reflita a realidade.

A implementação do classificador de spam com essas técnicas torna o processo acessível e prático, permitindo filtrar mensagens indesejadas e aprimorar a experiência do usuário na gestão de e-mails. Assim, o uso do Naive Bayes não apenas exemplifica a aplicação de conceitos matemáticos na tecnologia, mas também mostra como a ciência de dados pode impactar a comunicação digital.

8. [Activity] Implementing a Spam Classifier with Naive Bayes

O vídeo detalha a implementação de um classificador de spam utilizando o algoritmo de Naive Bayes, destacando a simplicidade e eficiência desse modelo. Inicialmente, enfatiza que o maior esforço reside na leitura e manipulação dos dados de entrada, onde dados de spam e de e-mails normais ("presunto") são organizados para treinar o classificador.

Para isso, cria-se um DataFrame usando a biblioteca pandas, com uma coluna para o texto dos e-mails e outra para a classificação (spam ou presunto). Em seguida, com uma função de leitura de arquivos, cada e-mail é lido, excluindo-se os cabeçalhos, para se obter o conteúdo relevante.

O modelo de Naive Bayes, importado da biblioteca scikit-learn, utiliza o CountVectorizer para transformar as mensagens em vetores de contagem de palavras, permitindo que cada palavra seja representada como um índice em uma matriz esparsa. Isso facilita o treinamento do modelo ao codificar as palavras numericamente. Assim, o MultinomialNB é aplicado, e o modelo é ajustado usando os dados processados, treinando-o para prever se um e-mail é spam.

Para testar o modelo, duas mensagens de exemplo são usadas: uma com o conteúdo "Viagra grátis agora", identificada corretamente como spam, e outra mensagem comum, "Olá Bob, que tal um jogo de golfe amanhã?", que é classificada como presunto.

9. K-Means Clustering

O K-means é uma técnica de aprendizado de máquina não supervisionado, amplamente utilizada para agrupar dados com base em suas características. O princípio fundamental dessa abordagem é a divisão de um conjunto de dados em K grupos ou aglomerados, onde K é um número definido pelo usuário. O

processo envolve a identificação de K centróides, que são os pontos centrais de cada grupo. Cada ponto de dado é associado ao centróide mais próximo, e esse agrupamento é ajustado iterativamente para melhor refletir a distribuição dos dados.

O algoritmo começa escolhendo aleatoriamente K centróides em um gráfico de dispersão. Em seguida, para cada ponto de dado, calcula-se a distância até cada centróide e o ponto é atribuído ao centróide mais próximo. Após essa atribuição, os centróides são recalculados com base nas novas associações de pontos, movendo-os para o centro dos pontos que pertencem a cada grupo. Este processo é repetido até que os centróides não se movam significativamente, indicando que uma convergência foi alcançada.

Embora o K-means seja uma técnica poderosa, existem desafios associados à sua aplicação. Um dos principais é a escolha do valor de K, que não é uma tarefa simples. Uma abordagem comum para determinar K é começar com valores baixos e aumentá-lo gradualmente, observando a redução no erro quadrático até que se alcance um ponto de estagnação, indicando que não se obtém mais informações relevantes ao adicionar mais grupos. Além disso, o algoritmo pode convergir para mínimos locais, dependendo da escolha inicial dos centróides. Por isso, é aconselhável executar o K-means várias vezes com diferentes inicializações e combinar os resultados.

Outro ponto a ser considerado é que o K-means não fornece rótulos ou interpretações explícitas para os agrupamentos formados. Embora o algoritmo consiga identificar relações entre os dados, cabe ao usuário analisar e nomear esses grupos com base nas características observadas. Assim, a interpretação dos resultados pode ser um desafio, mas também uma oportunidade de explorar padrões ocultos nos dados.

O uso de ferramentas como o Scikit-learn facilita a implementação do K-means, permitindo que os usuários explorem essa técnica de maneira acessível e prática, enquanto descobrem agrupamentos e padrões que podem não ser imediatamente evidentes.

10. [Activity] Clustering people based on income and age

No vídeo, foi discutido como implementar o algoritmo de agrupamento k-means com o Scikit-Learn em Python. Para começar, o instrutor cria dados fictícios que representam grupos predefinidos, com uma função `createClusteredData` que simula grupos naturais em uma relação entre idade e renda. Essa função gera grupos em torno de centróides aleatórios com desvio padrão controlado, oferecendo uma distribuição próxima de uma dispersão realista. Cada centróide é definido aleatoriamente dentro dos intervalos

especificados para as variáveis de renda (entre 20.000 e 200.000) e idade (entre 20 e 70 anos).

Para aplicar o algoritmo k-means, é necessário primeiro importar KMeans da biblioteca sklearn.cluster e usar a função createClusteredData para gerar um conjunto de 100 pontos em cinco clusters. Esse número de grupos (k) é previamente conhecido, facilitando a aplicação do modelo para fins didáticos, mas em situações reais, o valor ideal de k precisaria ser ajustado iterativamente.

Como o algoritmo k-means requer que os dados estejam em uma escala comparável, o instrutor aplica uma normalização, já que renda e idade variam em ordens de magnitude diferentes. O uso de StandardScaler ou MinMaxScaler ajuda a colocar os dados em uma escala comum, o que melhora a eficácia do algoritmo.

Após ajustar o modelo com fit, são obtidos os rótulos atribuídos aos dados, que representam os grupos identificados. Esses rótulos podem ser visualizados em um gráfico com `matplotlib`, onde cada ponto recebe uma cor conforme seu rótulo. A análise visual sugere que, enquanto alguns clusters são claramente identificados, outros apresentam sobreposição, dificultando uma separação ideal – uma indicação de que talvez um valor menor de k fosse mais apropriado.

Esse processo demonstra a simplicidade de uso do k-means no Scikit-Learn e a importância da normalização dos dados. Esse método é útil para descobrir padrões ocultos em dados não rotulados, sendo uma ferramenta relevante para análises exploratórias e insights sobre estruturas subjacentes em conjuntos de dados.

11. Measuring Entropy

Entropia, no contexto de ciência de dados, é uma medida que quantifica a desordem ou incerteza em um conjunto de dados. Esse conceito, emprestado da física e termodinâmica, nos ajuda a entender quão uniformes ou variados são os elementos em um conjunto de informações. Imagine que você possui um conjunto de animais classificados por espécie. Se todos os animais forem da mesma espécie, a entropia será muito baixa, pois há pouca variação — todos são iguais. No entanto, se cada animal for de uma espécie diferente, a entropia será alta, já que há muita diversidade, ou desordem, no conjunto.

Basicamente, a entropia nos diz o quanto um conjunto de dados é homogêneo ou heterogêneo. Quando a entropia é zero, significa que todas as classes são iguais. Se a entropia é alta, as classes são muito diferentes entre si. Em um nível intuitivo, o conceito é simples: a entropia descreve quão uniformes ou diversas são as coisas em um conjunto de dados.

Matematicamente, a entropia é calculada considerando a proporção de elementos de cada classe em relação ao conjunto total de dados. Para cada classe, você calcula a probabilidade (P) de um dado pertencer àquela classe e, em seguida, usa essa probabilidade para calcular a contribuição de cada classe para a entropia total. A fórmula envolve o produto de P pelo seu logaritmo natural, e o resultado é somado para todas as classes. Se a proporção de uma classe for zero ou cem por cento, sua contribuição para a entropia é nula, pois todos pertencem àquela classe ou nenhum pertence.

12. Decision Trees: Concepts

Uma árvore de decisão é uma ferramenta amplamente utilizada no aprendizado de máquina, que funciona como um fluxograma para auxiliar na tomada de decisões com base em vários atributos. Seu objetivo é prever resultados futuros com base em dados históricos. Ao fornecer ao algoritmo um conjunto de dados de treinamento, ele pode construir uma árvore que segmenta os dados em diferentes níveis, cada um representando uma decisão com base em um atributo específico. Esses atributos podem ser fatores como a umidade, a temperatura ou a presença de sol, no exemplo de decidir se uma pessoa deve ou não sair para brincar.

As árvores de decisão utilizam o conceito de entropia para orientar suas escolhas. A entropia é uma medida de desordem ou incerteza nos dados, e o objetivo do algoritmo é escolher atributos que a minimizem em cada divisão. Ao longo do processo, o algoritmo analisa qual atributo melhor separa os dados de forma que, em cada etapa, a árvore se aproxime de uma decisão mais clara, como “sim” ou “não” em relação à questão sendo analisada.

Um exemplo prático mencionado foi o uso de árvores de decisão para filtrar currículos, avaliando características como a experiência de trabalho, nível de educação, estágio e se o candidato frequentou uma universidade de prestígio. O modelo, ao ser treinado com dados de candidatos anteriores e os resultados de suas contratações, pode construir um fluxograma que ajudará a determinar se um novo candidato tem uma alta probabilidade de ser contratado.

O algoritmo responsável por construir essas árvores é chamado ID3, que segue uma abordagem gananciosa, ou seja, em cada etapa ele escolhe a melhor opção para minimizar a entropia naquele momento. No entanto, um dos desafios das árvores de decisão é o overfitting, um fenômeno no qual o modelo se ajusta muito bem aos dados de treinamento, mas não se generaliza bem para novos dados. Para resolver esse problema, uma técnica chamada floresta aleatória é frequentemente utilizada. As florestas aleatórias criam várias árvores de decisão, cada uma treinada em uma amostra aleatória dos dados e, em seguida, todas as árvores votam no resultado final. Esse processo de votação ajuda a reduzir o risco de overfitting e torna o modelo mais robusto.

13. [Activity] Decision Trees: Predicting Hiring Decisions

Na vídeo, foi falado sobre a simplicidade e poder das Árvores de Decisão para resolver problemas de classificação com poucos passos em Python. O instrutor utiliza dados fictícios sobre decisões de contratação para demonstrar como criar uma árvore de decisão e visualizar seu fluxo lógico. Começa com a leitura de um arquivo CSV usando pandas e transforma os dados para um formato adequado ao modelo: valores categóricos, como "Sim" e "Não", são convertidos em valores numéricos, e as informações sobre o nível educacional são codificadas.

Depois de organizar os dados, as colunas de atributos (ou características) são separadas da coluna de destino (contratação) para servir como variáveis X e Y, respectivamente, no processo de classificação. Com essas variáveis preparadas, é possível instanciar o classificador da árvore de decisão (DecisionTreeClassifier) e ajustar os dados, o que se resume a poucas linhas de código. Para visualizar a árvore de decisão resultante, ele utiliza um código gráfico que transforma os critérios de decisão em um fluxograma. Cada nó da árvore representa uma decisão baseada em atributos dos candidatos, como experiência e se já tiveram emprego.

Em seguida, o instrutor introduz o conceito de Floresta Aleatória, onde múltiplas árvores de decisão são usadas para aprimorar a precisão da classificação. A ideia é treinar várias árvores independentes e combinar suas previsões para reduzir o risco de superajuste, típico de uma única árvore de decisão. Com um componente de aleatoriedade, a Floresta Aleatória utiliza o método de "bagging" (Bootstrap Aggregating) para produzir diferentes combinações de amostras e, assim, gerar árvores ligeiramente distintas, melhorando a generalização do modelo.

14. Ensemble Learning

O aprendizado em conjunto é uma abordagem poderosa na aprendizagem de máquina que envolve a combinação de vários modelos para melhorar a precisão e a robustez das previsões. Em vez de confiar em um único modelo, utilizam-se vários modelos que trabalham juntos para produzir um resultado final mais eficaz. Essa técnica, como vimos no caso da Random Forest, utiliza múltiplas árvores de decisão, onde cada uma é treinada em subamostras dos dados e em diferentes conjuntos de atributos. Ao final, todas votam para decidir a classificação.

Existem diferentes métodos de aprendizado em conjunto. O ensacamento (ou bootstrap aggregating) é uma técnica onde múltiplos modelos são treinados com amostras aleatórias dos dados. No caso da Random Forest, cada árvore de decisão é construída com uma amostra aleatória dos dados de treinamento e, em

seguida, todas elas contribuem para o resultado final. Essa abordagem ajuda a evitar o overfitting, proporcionando previsões mais generalizáveis.

Outro método de aprendizado em conjunto é o reforço, que se diferencia do ensacamento. No reforço, cada novo modelo tenta corrigir os erros cometidos pelo modelo anterior, concentrando-se nas áreas onde a classificação falhou. A ideia é aumentar a precisão progressivamente, construindo modelos que aprimoram os erros anteriores.

Um terceiro método, conhecido como balde de modelos, permite o uso de diferentes tipos de modelos (como k-means, árvores de decisão e regressão) para fazer previsões. Nesse caso, os modelos "competem" entre si, e o modelo com o melhor desempenho é escolhido para fazer a previsão final. Já no empilhamento, os resultados de vários modelos são combinados de alguma forma para melhorar o resultado final, em vez de escolher apenas um modelo vencedor.

O aprendizado em conjunto mostrou-se extremamente eficaz em diversas aplicações práticas. Um exemplo famoso é o concurso da Netflix, onde pesquisadores utilizaram várias abordagens de conjunto para superar o algoritmo de recomendação de filmes da empresa, conseguindo resultados significativamente melhores ao combinar diferentes algoritmos.

Embora existam técnicas avançadas de aprendizado em conjunto, como o Classificador Ideal Bayes e o Bayesian Model Combination, essas abordagens são geralmente complexas e pouco práticas. Muitas vezes, técnicas mais simples, como o ensacamento e o empilhamento, são suficientes para alcançar excelentes resultados. Por isso, a recomendação é começar com as soluções mais simples e, apenas se necessário, explorar alternativas mais sofisticadas.

15. [Activity] XGBoost

Na videoaula, é discutido o funcionamento e as vantagens do XGBoost, que se destaca como um dos algoritmos mais eficientes para tarefas de aprendizado de máquina. Inicialmente, o conceito de boosting é abordado, explicando que se trata de um método de ensemble onde várias árvores de decisão são criadas, cada uma corrigindo os erros da anterior. Esse processo cria um modelo robusto que, segundo o instrutor, é rotineiramente o vencedor em competições como Kaggle devido à sua alta precisão e versatilidade.

O XGBoost, ou "eXtreme Gradient Boosted trees", é especialmente poderoso devido a funcionalidades avançadas, como o uso de regularização L1 e L2 para evitar o overfitting e sua capacidade de lidar automaticamente com valores faltantes, um desafio comum na ciência de dados. Além disso, ele é altamente escalável e aproveita a computação paralela, podendo processar grandes conjuntos de dados em clusters de computadores, o que o torna adequado para problemas tanto de classificação quanto de regressão.

Outras funcionalidades notáveis incluem a validação cruzada em cada iteração, permitindo ao usuário monitorar a performance do modelo durante o treinamento e interrompê-lo quando o ponto ideal é atingido. O XGBoost também permite retomar o treinamento em um modelo previamente treinado, facilitando o trabalho em lotes. Além disso, o algoritmo permite o uso de objetivos de otimização personalizados, tornando-o altamente adaptável a diferentes problemas.

Uma das técnicas discutidas para otimizar o desempenho do XGBoost envolve o ajuste de hiperparâmetros, como a taxa de aprendizado (eta), a profundidade máxima da árvore e o peso mínimo das folhas. O instrutor menciona que esses ajustes podem ser realizados por meio de ferramentas como GridSearchCV para identificar as melhores combinações, sendo a experimentação essencial nesse processo.

16. Support Vector Machines (SVM) Overview

As Máquinas de Vetores de Suporte (SVM) são uma técnica poderosa para classificar e agrupar dados em dimensões elevadas, ou seja, quando há muitas características ou atributos envolvidos. Elas são particularmente eficazes quando estamos lidando com dados complexos e várias variáveis ao mesmo tempo. O objetivo principal de uma SVM é encontrar um "hiperplano" que melhor separe os dados em diferentes classes. A matemática por trás das SVMs pode ser complexa, mas ferramentas como o scikit-learn facilitam a implementação prática sem a necessidade de entrar nos detalhes técnicos.

O conceito fundamental das SVMs gira em torno dos "vetores de suporte", que são pontos de dados críticos que definem o hiperplano de separação. A técnica utiliza um "truque do kernel" para lidar com dados não-linearmente separáveis, permitindo que sejam encontrados hiperplanos em dimensões superiores. Isso significa que, em vez de simplesmente dividir dados em duas dimensões, uma SVM pode trabalhar em espaços multidimensionais, o que a torna muito útil para dados complexos.

Os kernels (núcleos) são funções que ajudam a projetar os dados em espaços de maior dimensão. Existem diferentes tipos de kernels, como o linear, polinomial e radial, e a escolha do kernel adequado é importante para a eficácia do modelo. Cada kernel tem um custo computacional diferente, então, escolher o correto é essencial para balancear precisão e eficiência.

Outro ponto importante sobre as SVMs é que elas são uma técnica de aprendizagem supervisionada, o que significa que precisam de um conjunto de dados de treinamento com rótulos corretos para aprender. Isso as diferença de técnicas não supervisionadas, como o k-means, que não exige rótulos durante o treinamento.

Um exemplo clássico da aplicação de SVMs é a classificação do conjunto de dados Iris, que contém medições de flores de diferentes espécies. Com base no comprimento e largura das pétalas e sépalas, as SVMs podem prever a espécie da flor. Dependendo do kernel utilizado, os resultados podem variar, e há um equilíbrio entre a complexidade do modelo e sua capacidade de generalização.

Ao usar SVMs, é crucial realizar validação cruzada para encontrar os melhores parâmetros e evitar o sobreajuste (quando o modelo se ajusta demais aos dados de treinamento e não generaliza bem para novos dados). Portanto, o teste e ajuste dos parâmetros é uma parte essencial para garantir que o modelo tenha bom desempenho em dados não vistos

17. [Activity] Using SVM to cluster people using scikit-learn

Na videoaula, foi apresentada a prática com Máquinas de Vetores de Suporte (SVM), uma técnica de classificação amplamente utilizada em aprendizado de máquina. Inicialmente, foi destacada a facilidade de implementação das SVMs, utilizando-se um exemplo prático com o Scikit-learn para ilustrar a criação e o treinamento de um modelo de classificação baseado em SVM.

O primeiro passo consistiu em criar dados simulados para serem utilizados no treinamento do modelo. Foram gerados pontos de dados que representam clusters, onde cada ponto possui características bidimensionais (idade e renda) e rótulos associados ao cluster a que pertencem. Os centróides de cada cluster foram selecionados aleatoriamente em intervalos específicos, e os pontos foram distribuídos em torno deles.

Para melhorar o desempenho do modelo SVM, foi necessário escalar os dados para um intervalo entre -1 e 1, usando o `MinMaxScaler`, pois isso ajuda a garantir uma convergência mais estável, especialmente ao utilizar kernels complexos, como o polinomial.

Após preparar os dados, o modelo SVM foi configurado com um kernel linear, e o hiperparâmetro `C` foi mantido em 1.0. O ajuste do modelo foi rápido, e uma visualização foi criada para mostrar os clusters e as regiões de classificação geradas pelo SVM. Na visualização, foram destacadas áreas onde o modelo classificou os dados e sobrepôs as previsões de cluster com os rótulos reais. Em casos onde os clusters se sobrepunham, era esperado um certo nível de erro, já que a separação perfeita dos clusters se torna inviável nessas situações.

A aula também destacou a importância de transformar novos dados de entrada para o mesmo intervalo antes de realizar previsões, garantindo que o modelo treinado com dados escalonados receba novas entradas coerentemente.