Fundamentos de Machine Learning com Python

Machine Learning é uma subárea da Inteligência Artificial que foca no desenvolvimento de sistemas capazes de aprender e melhorar com a experiência sem serem explicitamente programados.

O processo de aprendizado em Machine Learning inicia com a alimentação de dados para os modelos de aprendizado. Esses dados podem ser imagens, textos, registros de áudio

Com base na identificação desses padrões, o modelo é capaz de fazer previsões ou tomar decisões quando confrontado com novos conjuntos de dados.

As abordagens para Aprendizado de Máquina (Machine Learning) podem ser categorizadas em três principais métodos: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

Aprendizado Supervisionado
O algoritmo é treinado em um conjunto de dados que já contém as respostas desejadas, chamadas de “labels” ou etiquetas. O objetivo é que, após o treinamento, o modelo seja capaz de prever a etiqueta correta para novos conjuntos de dados não vistos anteriormente. Esse tipo de aprendizado é utilizado em tarefas como classificação (onde a saída é discreta, como identificar se um email é spam ou não) e regressão (onde a saída é contínua, como prever o preço de uma casa).

Aprendizado Não Supervisionado
Diferentemente do aprendizado supervisionado, o aprendizado não supervisionado lida com dados que não possuem etiquetas. O objetivo aqui é explorar a estrutura dos dados para extrair padrões, agrupamentos ou correlações sem a orientação de uma variável de saída específica. Essa abordagem é útil em situações onde se sabe pouco sobre os dados ou quando é difícil ou inviável etiquetar os dados manualmente. Exemplos de uso do aprendizado não supervisionado incluem a

segmentação de clientes em marketing, detecção de anomalias e redução de dimensionalidade.

Aprendizado por Reforço
Abordagem dinâmica em que o modelo, ou agente, aprende a tomar decisões através de tentativa e erro, interagindo com um ambiente. Em cada ação, o agente recebe uma recompensa ou penalidade, com o objetivo de maximizar as recompensas ao longo do tempo. Esse método é particularmente útil em situações que requerem uma sequência de decisões, como jogos ou navegação de robôs.

Diferenças
Enquanto o aprendizado supervisionado requer um grande volume de dados etiquetados, o que pode ser um desafio em si, o aprendizado não supervisionado e por reforço oferecem alternativas quando tais dados não estão disponíveis ou são difíceis de obter.

Machine Learning Aprendizado Supervisionado vs Aprendizado não Supervisionado

Aprendizado Supervisionado

Usa **conjuntos de dados rotulados durante o treinamento**, permitindo que algoritmos classifiquem dados com precisão ou prevejam resultados, se divide em tarefas de classificação e regressão ao analisar dados:

Problemas de classificação: envolvem algoritmos que atribuem dados de teste a categorias específicas com precisão, como separar maçãs de laranjas. Esse enfoque é útil em filtros de spam. **Algoritmos** comuns usados incluem **classificadores lineares, máquinas de vetor de suporte, árvores de decisão e florestas aleatórias.**

Tarefas de regressão: compreendem as relações entre variáveis independentes e uma variável dependente. Modelos preveem valores

numéricos baseados em pontos de dados, tornando-os ideais para prever projeções de receita de vendas para empresas. **Algoritmos** de **regressão populares incluem regressão linear, regressão logística e regressão polinomial.**

Aprendizado Não Supervisionado

O aprendizado não supervisionado analisa e agrupa **conjuntos de dados não rotulados de forma independente**, revelando padrões ocultos sem intervenção humana. Os modelos de aprendizado não supervisionado realizam tarefas de agrupamento, associação e redução de dimensionalidade:

Agrupamento: agrupa dados não rotulados de acordo com a semelhança ou diferença. Algoritmos de agrupamento como K-means atribuem pontos de dados semelhantes a grupos, determinados pelos valores K que representam tamanho e granularidade. As aplicações variam de segmentação de mercado a compressão de imagens.

Associação: descobre conexões entre variáveis dentro de um conjunto de dados usando regras.

Redução de dimensionalidade: reduz conjuntos de dados com muitos recursos para tamanhos gerenciáveis enquanto preserva a integridade.

Principais Diferenças Entre Aprendizado Supervisionado e Não Supervisionado

No aprendizado supervisionado, são usados dados de entrada e saída rotulados, enquanto modelos de aprendizado não supervisionado não requerem rótulos. Modelos de aprendizado supervisionado tendem a ser mais precisos porque aprendem de conjuntos de dados rotulados, mas exigem intervenção humana inicial para rotulagem adequada. Modelos de aprendizado não supervisionado descobrem estruturas inerentes aos dados

de forma independente, porém necessitam de validação para interpretação das variáveis de saída.

Escolhendo Entre Aprendizado Supervisionado e Não Supervisionado

Considere se seus dados de entrada são rotulados ou não, se especialistas podem apoiar rotulagem adicional, se há problemas recorrentes a serem resolvidos ou se é necessário prever problemas desconhecidos. Reveja as opções de algoritmos em relação aos requisitos de dimensionalidade, capacidades de manuseio de volume de dados e adequação estrutural.

Aprendizado por Reforço

Esta técnica não depende de dados rotulados ou não rotulados. Em vez disso, baseia-se na ideia de recompensa e penalidade, um agente aprende a tomar decisões por meio da interação com um ambiente. Cada ação tomada pelo agente resulta em uma recompensa ou uma penalidade, e o objetivo é **maximizar as recompensas** ao longo do tempo. Esse método é amplamente utilizado em sistemas que requerem uma sequência de decisões, como jogos, navegação de robôs e automação de veículos autônomos.

- **O Agente**: o modelo ou algoritmo que toma decisões.
- **O Ambiente**: o mundo com o qual o agente interage e onde ele realiza suas ações.
- **A Recompensa**: o feedback que o agente recebe após cada ação

Oferece uma opção dinâmica e adaptativa para problemas que envolvem decisões sequenciais e interações com ambientes que estão em constante mudança.

Desvendando o Ciclo de Vida de um Projeto de Machine Learning

1. Definição do Problema

Essa fase é crítica porque orienta a direção estratégica do projeto, definindo o escopo e os objetivos que o modelo de ML deve alcançar.

Entender o Contexto de Negócio: Esta etapa começa com uma imersão profunda nos objetivos estratégicos da empresa ou do setor. É crucial alinhar o problema de ML com as metas do negócio e os resultados desejados. A equipe deve fazer perguntas como: “Quais são os desafios de negócios que estamos tentando superar?” ou “Que processo ou resultado específico queremos melhorar com ML?”.

Formulação do Problema de ML: Após compreender as metas do negócio, o próximo passo é traduzir esses objetivos em um problema de ML bem definido. A problemática deve ser formulada de tal forma que seja clara, mensurável e viável.

Estabelecendo Metas e Métricas Claras: A definição do problema deve ser acompanhada pela determinação de métricas específicas que serão utilizadas para avaliar o sucesso do projeto.

Identificação dos Stakeholders: Nesta fase, é importante identificar todas as partes interessadas – internas e externas – que serão afetadas pelo projeto de ML ou que terão influência sobre ele.

Viabilidade Técnica e de Dados: Antes de prosseguir, a equipe deve avaliar se o problema definido é tecnicamente viável com os dados disponíveis ou se é possível adquirir os dados necessários. É crucial verificar se os dados

existentes são suficientemente informativos e se existe a infraestrutura técnica e a expertise necessária para desenvolver a solução de ML.

2. Coleta de Dados

Aqui, a atenção meticulosa é dada a reunir informações que serão a base para o treinamento do algoritmo.

Identificação das Fontes de Dados: A busca pelos dados corretos começa com a identificação das fontes apropriadas. Essas fontes podem variar de bancos de dados internos e arquivos de log, a conjuntos de dados públicos ou dados adquiridos de terceiros.

Avaliação da Qualidade dos Dados: Nem todos os dados são criados iguais. A equipe precisa avaliar se os dados coletados são de alta qualidade, o que envolve checar a precisão, a completude e a consistência das informações.

Volume e Variedade: O volume de dados necessário pode variar de acordo com o problema específico e a complexidade do modelo. Modelos mais sofisticados normalmente exigem quantidades maiores de dados.

Aspectos Legais e Éticos: Ao coletar dados, especialmente dados sensíveis ou pessoais, é fundamental considerar aspectos legais, como conformidade com a General Data Protection Regulation (GDPR) ou outras regulamentações de privacidade.

Coleta e Agregação: Uma vez identificadas as fontes de dados e avaliada sua qualidade, começa o processo de coleta e agregação. Ferramentas e sistemas de ETL (Extract, Transform, Load) são comumente utilizados para extrair dados das fontes, transformá-los conforme necessário e carregá-los em um repositório centralizado onde eles podem ser acessados e utilizados pelos modelos de ML.

Pré-Processamento Inicial: Embora o pré-processamento em profundidade ocorra na próxima fase do ciclo de vida, é importante realizar uma limpeza inicial dos dados durante a coleta. Isso pode incluir a remoção de duplicatas, o tratamento de valores ausentes e a identificação de outliers.

3. Preparação e Análise dos Dados

Aqui, os dados coletados são transformados e refinados para garantir que estão em um formato ideal para extração de padrões relevantes pelo algoritmo de aprendizado.

Limpeza de Dados: Esta subetapa envolve a correção ou remoção de dados incorretos, incompletos, duplicados ou irrelevantes.

Transformação dos Dados: Os dados podem precisar ser transformados para se ajustarem melhor aos requisitos do algoritmo. Isso inclui a normalização ou padronização para que os dados estejam na mesma escala, a conversão de variáveis categóricas em numéricas, ou a engenharia de features, que é o processo de criar novos atributos preditivos a partir dos dados existentes.

Análise Exploratória de Dados (EDA): A EDA é um componente crucial desta fase e envolve a exploração visual e quantitativa para identificar padrões, detectar outliers e testar hipóteses.

Seleção de Características: Nem todas as features dos dados são igualmente importantes para o modelo. A seleção de características é o processo de identificar as variáveis mais relevantes para o problema. Isso não só melhora o desempenho do modelo, mas também reduz a complexidade computacional e o risco de overfitting.

Divisão dos Dados: Geralmente, os dados são divididos em conjuntos de treino e teste (e às vezes um conjunto de validação). Essa prática é essencial para avaliar a capacidade do modelo de generalizar para novos dados.

Documentação: A documentação rigorosa durante a preparação e análise dos dados é fundamental. Isso inclui registrar as decisões tomadas, as técnicas utilizadas e as descobertas feitas.

4. Treinamento do Modelo

Esta fase é repleta de complexidades técnicas e requer uma estratégia metodológica para assegurar que o modelo final possa fazer previsões precisas.

Escolha do Algoritmo Apropriado: O ponto de partida do treinamento é selecionar um algoritmo adequado para o problema em mãos. O algoritmo escolhido depende da natureza dos dados, do tipo de problema (classificação, regressão, agrupamento, etc.) e dos objetivos específicos estabelecidos durante a definição do problema. Modelos comuns incluem regressão linear, árvores de decisão, redes neurais e máquinas de vetor de suporte

Configuração dos Hiperparâmetros: Antes de iniciar o treinamento, é necessário configurar os hiperparâmetros do modelo. Hiperparâmetros são configurações que governam o processo de aprendizado e podem ter um impacto significativo no desempenho do modelo.

Treinamento e Validação Cruzada: Com os dados preparados e os hiperparâmetros definidos, o modelo é treinado utilizando os conjuntos de treino. Durante este processo, técnicas como a validação cruzada são frequentemente empregadas para avaliar como o modelo generalizará para um conjunto de dados independente.

Minimização da Função de Perda: O objetivo do treinamento é minimizar a função de perda, uma métrica que quantifica a diferença entre as previsões do modelo e os verdadeiros resultados.

Evitar Sobreajuste: Um dos desafios cruciais durante o treinamento é evitar o sobreajuste, que ocorre quando o modelo aprende padrões específicos para o conjunto de treino, mas não generaliza bem para novos dados.

Monitoramento do Processo de Treinamento: Ao longo do treinamento, é importante monitorar o desempenho do modelo. Isso normalmente envolve acompanhar a função de perda e outras métricas de desempenho, como acurácia, ao longo do tempo.

5. Avaliação do Modelo

É crucial para entender a eficácia do modelo de Machine Learning após o treinamento. Ela serve como um indicativo de como o modelo irá se comportar na prática, quando exposto a dados que nunca viu.

Utilização de Conjuntos de Teste: Para avaliar a performance, utilizamos o conjunto de teste que foi mantido separado durante a etapa de treinamento do modelo.

Métricas de Avaliação: Dependendo do tipo de problema de ML (por exemplo, classificação ou regressão), diferentes métricas são utilizadas para avaliar o modelo. Para problemas de classificação, métricas como precisão, recall e a pontuação F1 são comuns, enquanto que para regressão, métricas como o erro quadrático médio (MSE) ou o coeficiente de determinação (R^2) são utilizados.

Análise de Erros: Um componente essencial da avaliação é a análise dos erros cometidos pelo modelo. Ao examinar os casos em que o modelo errou, podemos obter insights sobre possíveis melhorias tanto nos dados quanto no algoritmo utilizado.

Curvas de Aprendizado: As curvas de aprendizado são gráficos que mostram a evolução do desempenho do modelo em função da quantidade de dados de treinamento.

Matriz de Confusão: Em classificação, a matriz de confusão é uma ferramenta poderosa que fornece uma visão detalhada do desempenho do modelo.

Validação Cruzada Robusta: Diferente da validação cruzada usada durante o treinamento, que visava ajustar os parâmetros do modelo, aqui a validação cruzada é usada para confirmar a robustez do desempenho do modelo.

Feedback de Stakeholders: Eles podem oferecer uma perspectiva diferente sobre o desempenho do modelo, especialmente em relação a como ele atende aos objetivos do negócio definidos na primeira etapa.

6. Ajuste Fino e Otimização

Aqui, iteramos sobre o modelo com o intuito de melhorar seu desempenho, fazendo ajustes baseados nas informações coletadas durante a avaliação.

Tuning de Hiperparâmetros: Um dos primeiros passos para otimizar o modelo é o tuning, ou ajuste fino, dos hiperparâmetros. Esta é uma etapa delicada, pois envolve encontrar o equilíbrio correto entre o poder de generalização e a capacidade de capturar padrões nos dados. Técnicas como pesquisa em grade (grid search), pesquisa aleatória (random search) ou métodos Bayesianos são frequentemente usadas para automatizar e otimizar este processo.

Reengenharia de Características: Isso pode envolver a combinação de características existentes, a transformação de variáveis, ou a eliminação de características que não estão contribuindo para o desempenho do modelo.

Ensemble Learning: Isso pode aumentar a precisão e a robustez do sistema de Machine Learning, pois reduz o risco de erros devido a variações nos dados de treino.

Pruning (Poda) de Modelos: A técnica de pruning envolve a remoção seletiva de partes do modelo que têm pouco ou nenhum valor.

Validação em Diferentes Subconjuntos de Dados: O modelo otimizado é frequentemente validado em diferentes subconjuntos de dados. Isso assegura que o modelo mantém uma boa performance independente das variações dos dados.

Análise de Custo-Benefício: Alguns ajustes podem trazer melhorias mínimas no desempenho com custos computacionais ou temporais significativamente mais altos. Uma análise cuidadosa deve ser feita para evitar a otimização excessiva.

Automatização do Ajuste Fino: Ferramentas e plataformas de AutoML têm ganhado popularidade por sua capacidade de automatizar muitos dos processos de ajuste fino e otimização

7. Implantação em Produção

A última milha no desenvolvimento de um modelo de Machine Learning é a **Implantação em Produção**, onde o modelo bem testado e otimizado é finalmente colocado em uso real.

Seleção da Infraestrutura: A decisão sobre onde e como o modelo será implantado é crucial. Opções incluem servidores locais, na nuvem, ou até mesmo edge computing.

Integração com Sistemas Existentes: A implementação deve ser feita de forma que o modelo possa se integrar sem problemas com os sistemas já em uso na empresa. Isso pode envolver o desenvolvimento de APIs, microserviços, ou adaptações em sistemas de banco de dados.

Monitoramento Contínuo: Uma vez implantado, o modelo deve ser monitorado continuamente para garantir que mantenha seu desempenho e para identificar rapidamente qualquer degradação ou falha.

Atualização e Manutenção: A implantação do modelo não é o final da jornada. Modelos podem precisar ser reajustados ou re-treinados ao longo do tempo devido a mudanças nos padrões de dados.

Governança e Compliance: Uma vez implantado, o modelo deve ser monitorado continuamente para garantir que mantenha seu desempenho e para identificar rapidamente qualquer degradação ou falha. Métricas de desempenho são frequentemente usadas, além de logs de sistema e alertas automatizados.

Atualização e Manutenção: Modelos podem precisar ser reajustados ou re-treinados ao longo do tempo devido a mudanças nos padrões de dados.

Governança e Compliance: O cumprimento das regulamentações de dados é um aspecto significativo, especialmente para modelos que lidam com informações sensíveis.

Feedback Loop para Melhorias Contínuas: A implantação bem-sucedida de um modelo oferece a oportunidade para um feedback loop onde os dados coletados na produção podem ser utilizados para refinar ainda mais o modelo e sua performance.

Estratégias de Implantação: Existem diferentes estratégias para a implantação, como a implantação direta (ou “big bang”), implantação paralela, onde o modelo antigo e o novo funcionam simultaneamente, e a implantação em fases (ou “canary release”), onde o modelo é exposto a uma parte dos usuários antes de uma liberação completa.

8. Monitoramento e Manutenção

O monitoramento ajuda a detectar mudanças nos padrões dos dados que podem afetar o desempenho do modelo, enquanto a manutenção regular garante que o modelo continua a performar adequadamente ao longo do tempo.

9. Feedback e Iteração

A fase de **Feedback e Iteração** é vital para refinamentos contínuos e garantia de que o modelo permanece alinhado com as necessidades dos usuários e as dinâmicas do mercado.

Coleta de Feedback: O feedback dos usuários finais do modelo é uma mina de ouro para entender seu desempenho no mundo real. A

Análise do Feedback: O feedback coletado precisa ser sistematicamente analisado para identificar padrões, problemas recorrentes ou oportunidades de aprimoramento.

Iteração Contínua: Com base nesses insights, o modelo pode necessitar de iterações.

Aprendizado com os Dados: À medida que o modelo está em produção, ele gera novos dados que podem ser usados para aprendizado adicional.

Retreinamento do Modelo: Quando são identificadas mudanças significativas nos dados ou no desempenho do modelo, pode-se retreinar o modelo utilizando os dados mais recentes.

A/B Testing e Experimentação: A implementação de testes A/B, onde diferentes versões de um modelo são testadas em paralelo

Gestão do Ciclo de Vida do Modelo: Isso inclui saber quando é hora de aposentar um modelo ou quando investir em uma nova geração de soluções.

Documentação e Comunicação: Documentar cada iteração, os motivos para as mudanças e os resultados obtidos é importante tanto para o processo de aprendizado quanto para a comunicação com as partes interessadas.

NumPy

Conversão de Lista para ndarray: Você pode transformar uma lista Python em um ndarray usando a função `np.array()`
Para verificar o tipo de dados com `print(meu_array.dtype)`

Criação usando Funções do NumPy: O NumPy também oferece várias funções para criar arrays com conteúdo inicial específico. Por exemplo, `np.zeros()` ou `np.ones()` para criar arrays preenchidos com zeros ou uns

Quando você executa o comando `print(meu_array.shape)` no contexto da biblioteca NumPy e recebe a saída `(5,)`, isso indica que `meu_array` é um array unidimensional (1D) com 5 elementos.

- A **primeira parte** da saída, que é o número `5`, representa o número de elementos ao longo da primeira (e única, nesse caso) dimensão do array.
- O uso da **vírgula** seguida de um **parêntese fechado** é a notação do Python para indicar que se trata de uma **tupla** de um único elemento.

A Função linspace(): Outra função poderosa é `np.linspace(start, stop, num)`, que cria arrays com valores igualmente espaçados dentro de um intervalo especificado.

Seleção e Manipulação de Dados

Acessar Elementos Individuais: Para acessar um único elemento, utilize índices que correspondam à sua posição na matriz.

```
elemento_200 = array_2d[0, 1]
print(elemento_200)
```

Slicing: Para acessar um SUBCONJUNTO do array, utilizamos a técnica conhecida como slicing. O slicing pode ser aplicado em arrays 1D, 2D.

```
# Acessar a primeira linha inteira
```

```
primeira_linha = array_2d[0, :]
# Acessar a terceira coluna inteira
terceira_coluna = array_2d[:, 2]
```

antes do : é o início depois dele é o fim

por exemplo

```
teste = array_2d[2:, 1:4]
```

resultado => `[[7 8 9]]`

Quando não especificamos antes do : o python entende que é para começar do início do array, ou seja, indice 0. Quando não especificamos o depois do :, o python entende que é para ir até o fim.

- **array_2d[:, 2]** esse aqui indica que vamos pegar todas as linhas da coluna no índice 2, ou seja, a terceira coluna

Adição de Step ao Slice

O 'step' pode ser especialmente útil em arrays maiores, onde talvez você queira acessar elementos saltando um certo intervalo.

```
# Acessar elementos na segunda linha, saltando de dois em dois
linha_com_step = array_2d[1, ::2]
```

antes do : é o início, o meio dele é o fim, e último valor é o passo. Então:

inicio:fim:passo => é sintaxe do slice + step

Exemplo mais detalhado:

```
slice_step = segundo_array_2d[1, 0:4:2]
```

- **1**: Especificamos que queremos acessar a segunda linha do array.
- **0:4:2** é a expressão de fatiamento onde:
  - **0** é o índice inicial do slice.
  - **4** é o índice final do slice, não inclusivo, então fazemos 4 - 1, assim, indo até o índice 3.

- **2** é o ‘step', que nos diz que queremos selecionar elementos saltando de dois em dois.

Outro exemplo:

```
slice_2d = array_2d[1:3, ::2]
```

O primeiro par de índices `1:3` define quais linhas do array serão incluídas no slice. Neste caso, ele começa no índice 1 e vai até o índice 3.

A segunda parte, `::2`, é aplicada nas colunas. Aqui, o `::` informa que você deseja incluir todas as colunas, mas o `2` adicionado ao final indica que você quer fazer isso em passos de dois.

Explorando Métodos ndarray

métodos `sum()`, `mean()` e `reshape()`

O Método sum(): O método `sum()` é usado para somar todos os elementos de um array ou somar elementos ao longo de um determinado eixo.

```
array_2d = np.array([[1, 2], [3, 4], [5, 6]])

# Somando elementos de cada coluna (axis=0)
soma_colunas = array_2d.sum(axis=0)

# Somando elementos de cada linha (axis=1)
soma_linhas = array_2d.sum(axis=1)
```

axis=0 => é as linhas| **axis=1** => é as colunas

`soma_colunas` resultará em `[9, 12]` porque soma os elementos de cada coluna individualmente (1+3+5 e 2+4+6), `soma_linhas` dará `[3, 7, 11]`, somando os elementos linha por linha.

Método mean(): Já o método `mean()` calcula a média aritmética de um array. Quando não especificado, ele calcula a média de todos os elementos.

O Método reshape(): Com `reshape()`, você pode alterar a estrutura de um array sem alterar seus dados.

```
array_2x5 = array_1d.reshape(2, 5)
```

Passar `-1` em uma das dimensões instrui o NumPy a calcular automaticamente o tamanho dessa dimensão.

```
# Redimensionar para 2 linhas e calcular o número necessário de colunas
array_2d_auto = array_1d.reshape(2, -1)
# Redimensionar para um número desconhecido de linhas e 5 colunas
array_autox5 = array_1d.reshape(-1, 5)
```

No NumPy, `reshape()` pode ser utilizado tanto como **função** quanto como **método** de um objeto array NumPy.

Como função:

```
import numpy as np

# Cria um array com 6 elementos
array_original = np.array([1, 2, 3, 4, 5, 6])

# Usa a função np.reshape para alterar a forma do array
array_reshaped = np.reshape(array_original, (2, 3))
```

parei em **Funções Essenciais do NumPy**