SUMÁRIO

O QUE VEM POR AÍ?

Chegamos no momento final da nossa disciplina: nesta aula, você aprenderá a integrar um modelo de Machine Learning treinado em Python dentro de uma API Flask funcional. Pense em todo o potencial que isso traz: em vez de manter o modelo isolado em um notebook, agora ele estará disponível para ser consumido via HTTP, tornando suas previsões acessíveis a qualquer sistema ou cliente. E não para por aí: você também verá como proteger essas rotas com autenticação via tokens JWT, armazenar o histórico de predições em um banco de dados e lidar com problemas comuns de produção.

Imagine poder fornecer recomendações, classificações ou análises preditivas diretamente a partir de um endpoint. Ao final desta aula, você não apenas terá um modelo disponível como serviço, mas entenderá todo o fluxo que envolve treinar, salvar, carregar o modelo, incorporá-lo em uma API Flask, adicionar segurança e até persistir predições para auditoria futura.

Não deixe de conferir as videoaulas e retornar a este material sempre que precisar. Mãos à obra!

HANDS ON

Nas videoaulas desta seção, você acompanhará o processo completo de implantação do modelo em um serviço Flask. Partindo de um modelo treinado e salvo localmente, veremos como carregá-lo na inicialização da aplicação, criar endpoints para realizar predições e salvá-las em um banco de dados. Além disso, será demonstrado como incorporar autenticação JWT, garantindo que apenas usuários autorizados possam acessar as rotas de previsão.

Este é o momento de “colocar a mão na massa”: abra seu terminal, siga as instruções do vídeo e rode a aplicação localmente. Você poderá enviar requisições de teste usando ferramentas como cURL, Postman ou o próprio navegador. Experimente alterar parâmetros, consulte o histórico de predições e observe a resposta da API.

Aperte o play e acompanhe; volte quando precisar: teste e compreenda na prática!

SAIBA MAIS

Agora, vamos nos aprofundar no raciocínio por trás da integração do modelo ao Flask. Você entenderá como o ciclo completo funciona: desde o treinamento e salvamento do modelo, passando pelo carregamento na aplicação, até a utilização em endpoints seguros e a persistência das previsões em um banco de dados.

Carregando e Servindo um Modelo

Ao treinar um modelo de Machine Learning (por exemplo, um classificador Logistic Regression), é comum salvá-lo em disco usando bibliotecas como **joblib** ou **pickle**. Ao iniciar a API, você carrega esse modelo na memória, deixando-o pronto para prever respostas a qualquer requisição recebida.

Fluxo Típico:

1. Treine o modelo offline e salve-o em um arquivo (modelo_iris.pkl).
2. Na API, carregue o modelo ao iniciar (start da aplicação Flask).
3. Ao receber uma requisição de previsão (POST /predict), extraia os parâmetros, formate-os e chame model.predict().
4. Retorne o resultado ao cliente.

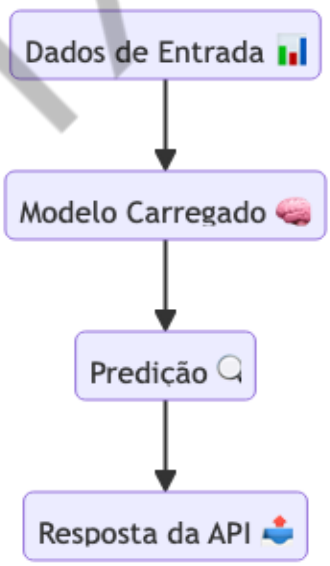

Figura 1 - Fluxo de integração do modelo
Fonte: Elaborado pelo Autor (2025)

Experimente alterar o modelo ou treinar outro modelo para outra tarefa e integre-o da mesma forma!

Autenticação com JWT

Para evitar que qualquer pessoa acesse a rota de previsão, é comum proteger a API com autenticação. O **JWT (JSON Web Token)** é um método eficaz para criar tokens autocontidos que carregam informações do usuário e prazo de expiração. A lógica é simples:

- O cliente envia credenciais para /login.
- Se corretas, a API responde com um token JWT.
- Em requisições subsequentes, o cliente envia o token no cabeçalho Authorization: Bearer <token>.

O servidor valida a assinatura e a validade do token antes de atender a rota protegida. Assim, apenas usuários autenticados podem fazer predições ou listar histórico.

Teste a expiração do token, aguarde uma hora (ou ajuste o tempo) e tente chamar o endpoint novamente!

Armazenando Predições no Banco de Dados

Salvar o histórico de previsões é importante por vários motivos: auditoria, análise de uso, re-treino do modelo ou simples rastreamento do que o sistema está produzindo. Usando SQLAlchemy, você pode criar modelos que representam tabelas e salvar cada requisição atendida. Assim, cada predição gera um registro com dados de entrada, classe predita e timestamp.

Passos para Persistência

- Definir um modelo SQLAlchemy (por ex. Prediction) com colunas para parâmetros e resultado.
- Após cada previsão, criar um objeto Prediction e salvar (commit) no banco.
- Expor rotas para consultar o histórico, filtrando por limit e offset por exemplo.

Estude o banco de dados e veja quais previsões foram armazenadas, tente implementar filtros adicionais!

Cache e Desempenho

Para reduzir a latência em cenários de alta demanda, um cache simples em memória pode ser empregado. Se o mesmo conjunto de parâmetros for requisitado várias vezes, o modelo não precisa recalcular a previsão toda hora, basta retornar a resposta cacheada. Claro que isso depende do tipo de predição, e nem sempre cache é útil, mas esta é uma opção para otimizar a performance.

Desafios do Cache:

- Dados em cache podem ficar desatualizados se o modelo mudar.
- Pode ser útil apenas para conjuntos de dados estáticos ou cenários de alta repetição.

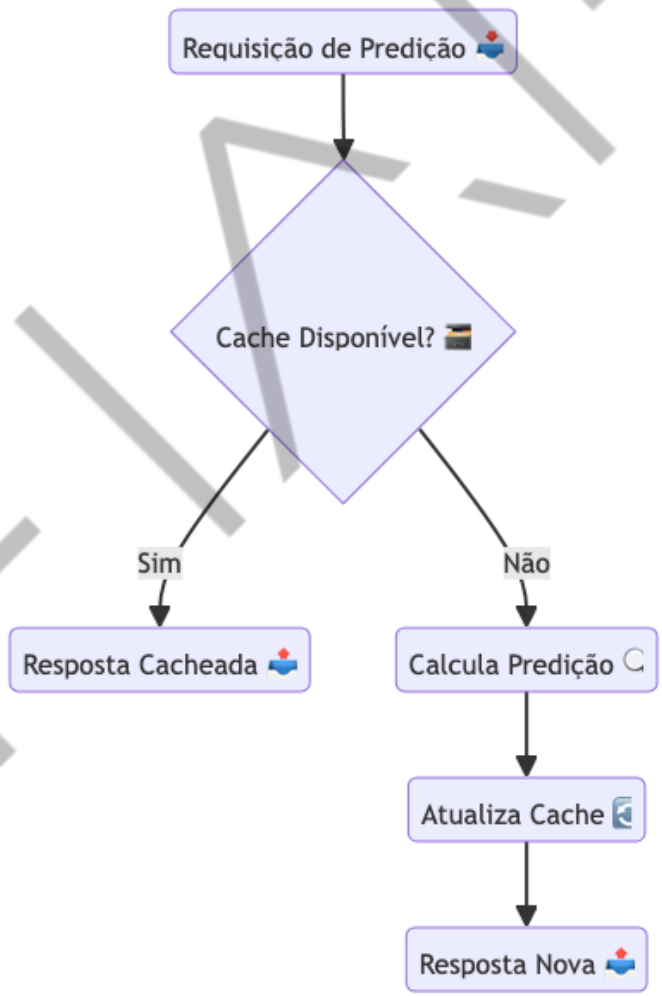

Figura 2 - Fluxo com Cache
Fonte: Elaborado pelo Autor (2025)

Teste remover o cache e medir o tempo de resposta; depois, reintroduza o cache e compare!

Segurança e Boas Práticas

Além do token JWT, outras boas práticas incluem:

- **Criptografar senhas**: mesmo neste exemplo simples, em produção deve-se usar hash seguro.
- **Variáveis de ambiente**: não coloque segredos (chaves JWT, credenciais) no código. Use variáveis de ambiente.
- **Logs e Monitoramento:** mantenha registros de erros, latência e número de requisições.

Ferramentas Recomendadas

- Alembic: para gerenciar migrações do banco de dados e acompanhar a evolução do schema.
- Gunicorn/Uvicorn: para rodar a aplicação Flask/FastAPI em produção.
- Docker: containerize a aplicação para facilitar o deploy.

Considere implementar migrações com Alembic, mesmo que apenas para entender o fluxo de evolução do schema!

Referências e Leituras

- Documentação Flask.
- SQLAlchemy.
- joblib.
- Documentação JWT.
- Alembic para migrações.

Livros e Materiais:

- Flask Web Development (Miguel Grinberg) - Práticas avançadas com Flask.
- Building Machine Learning Powered Applications (Emmanuel Ameisen) - Integração de ML com sistemas produtivos.

Veja as documentações, entenda cada componente e retorne a este material caso você tenha dúvidas!

O QUE VOCÊ VIU NESTA AULA?

Você aprendeu a pegar um modelo de Machine Learning treinado e disponibilizá-lo através de uma API Flask, adicionando autenticação JWT para proteger as rotas, cache para melhorar desempenho e um banco de dados para armazenar resultados.

Esse conjunto de técnicas leva seu projeto de um simples notebook a um serviço profissional, pronto para integrar-se a qualquer ecossistema.

Relembre os passos, assista aos vídeos novamente, teste novas abordagens e continue expandindo suas habilidades!

REFERÊNCIAS

AMEISEN, E. **Building Machine Learning Powered Applications**: Going from Idea to Product. [s.l.]: O'Reilly Media, 2020.

JOHNSON, D.; LEE, E. Scalable Machine Learning Deployments with APIs. **Journal of Applied AI**, v. 12, n. 3, 2019, p. 45-58.

SMITH, A.; JOHNSON, B.: WILLIAMS, C. **APIs in Machine Learning**: Bridging the Gap Between Models and Applications. [s.l.]: Tech Publishers, 2020.

PALAVRAS-CHAVE

Palavras-chave: APIs. Machine Learning. Escalabilidade.