Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas
Departamento de Engenharia Elétrica

Apostila de treinamento

Equipe NIAS-IA

Núcleo Interdisciplinar de Análise de Sinais

"Fail fast, learn faster."

Brasil
2025

A Equipe NIAS-IA agradece à Fapemig pelo apoio financeiro. Esta apostila foi desenvolvida no âmbito do Projeto APQ-00748-19, Chamada Fapemig 06/2019, Programa Santos Dumont.

Parte I

Exploração de Dados e Modelagem Inicial

Apresentação

Esta apostila tem como objetivo oferecer uma introdução sólida e prática aos fundamentos da Ciência de Dados e da Inteligência Artificial, com ênfase especial nas técnicas de Aprendizado de Máquina (Machine Learning) e Aprendizado Profundo (Deep Learning). Ao longo do material, o(a) estudante será guiado(a) desde conceitos básicos até a construção, treinamento e avaliação de modelos capazes de extrair padrões complexos de dados estruturados e não estruturados.

O conteúdo foi estruturado para que o(a) aluno(a) compreenda não apenas os aspectos teóricos, mas também desenvolva habilidades práticas para manipulação, exploração e limpeza de dados, além da implementação de modelos de redes neurais usando a biblioteca Keras, reconhecida pela sua simplicidade e eficiência no desenvolvimento de Deep Learning.

Para a implementação dos exemplos, utiliza-se a linguagem Python, versão 3.4 ou superior, em conjunto com o ambiente Jupyter Notebook, que propicia uma interface interativa ideal para aprendizagem e experimentação. Para instalações locais, recomenda-se o uso do pacote Anaconda [1], que facilita a configuração completa do ambiente com todas as bibliotecas necessárias.

Como alternativa, destaca-se a plataforma em nuvem Kaggle [2], que oferece um ambiente pronto para uso com suporte integrado a diversas bases de dados e ferramentas avançadas de Machine Learning e Deep Learning, incluindo suporte nativo ao Jupyter Notebook (chamado de "Kernel"). Essa plataforma é indicada para facilitar o acesso, execução e compartilhamento dos projetos abordados.

Ao final da apostila, espera-se que o(a) estudante esteja apto(a) a interpretar resultados, ajustar modelos, identificar limitações e potencialidades das técnicas estudadas, preparando-se para aprofundar seus estudos e aplicações em Ciência de Dados e Inteligência Artificial.

Capítulo 1

Introdução ao Kaggle

O *Kaggle* é uma plataforma amplamente utilizada para competições em Ciência de Dados, além de oferecer uma série de minicursos voltados ao aprendizado introdutório de técnicas de *Machine Learning* e *Deep Learning*.

Para acessar a plataforma, é necessário realizar um cadastro de usuário. Recomenda-se utilizar o e-mail institucional, tanto para a inscrição nos cursos quanto para a submissão de códigos em competições. Após o cadastro, o usuário será redirecionado à página inicial da plataforma, onde é possível visualizar os seguintes menus principais: *Home*, *Compete*, *Data*, *Code*, *Communities* e *Courses*.

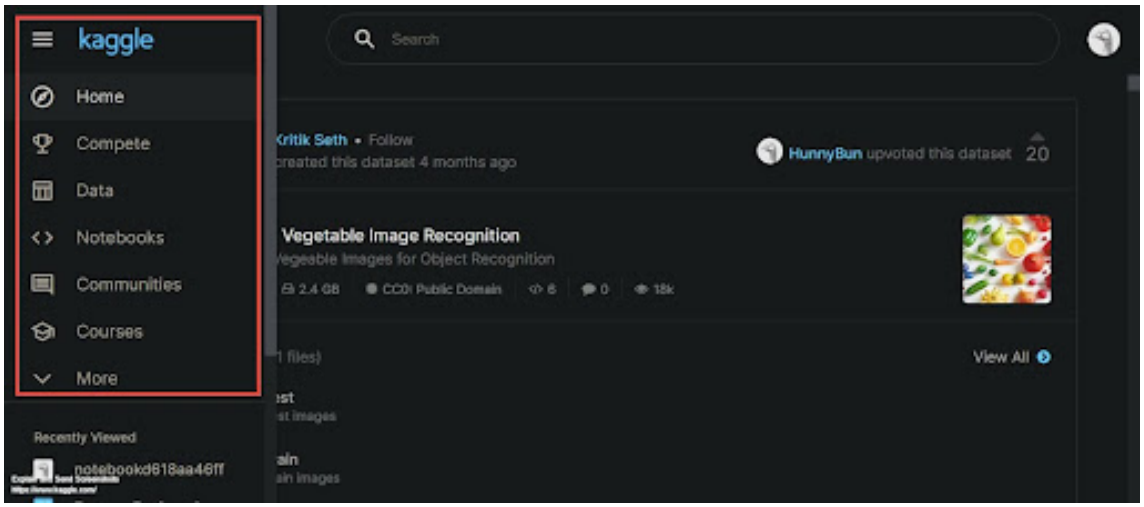

Figura 1.1: Página inicial do Kaggle.

Na seção *Home*, são apresentadas as notícias mais recentes, discussões em destaque e competições ativas na plataforma.

1.1 Compete

A aba *Compete* é dedicada às competições de Ciência de Dados. Ela se divide em dois segmentos principais:

- **Your Competitions:** exibe as competições nas quais o usuário está participando, bem como aquelas que já foram concluídas, marcadas como favoritas ou hospedadas pelo próprio usuário (na aba *Hosted*).

- **All Competitions:** apresenta todas as competições ativas, as que já foram encerradas, bem como competições voltadas ao aprendizado prático, disponíveis na aba *InClass*.

1.2 Data

A seção *Data* disponibiliza uma vasta coleção de conjuntos de dados (datasets) que podem ser utilizados tanto em competições quanto em estudos individuais. Esses datasets estão disponíveis em formatos variados, como CSV, JSON e BigQuery, entre outros.

O formato mais comumente utilizado é o CSV, que representa dados em forma tabular, com linhas geralmente indexadas por números e colunas nomeadas. Formatos como JSON e BigQuery apresentam estruturas mais complexas, sendo o JSON caracterizado por dados hierárquicos e o BigQuery por ser uma ferramenta do Google voltada à manipulação de grandes volumes de dados.

É possível pesquisar datasets por nome ou por palavras-chave relacionadas ao conteúdo. Além disso, usuários podem realizar o upload de seus próprios arquivos, tornando-os públicos para a comunidade da plataforma e, eventualmente, utilizá-los em competições.

1.3 Code

A aba *Code* permite o acesso aos Notebooks do Kaggle, ambientes de desenvolvimento integrados (IDEs) que utilizam servidores em nuvem para a execução de código. As linguagens atualmente suportadas são Python e R. Os Notebooks são utilizados tanto para a submissão de soluções em competições quanto para a realização dos exercícios propostos nos minicursos.

Nesta seção, é possível pesquisar notebooks públicos por título ou por palavras-chave. Recomenda-se explorar códigos relacionados aos temas em estudo, o que pode auxiliar na definição de estratégias para a construção dos próprios modelos.

O usuário também pode criar seus próprios notebooks. Ao iniciar um novo notebook, será apresentada uma interface como a ilustrada a seguir:

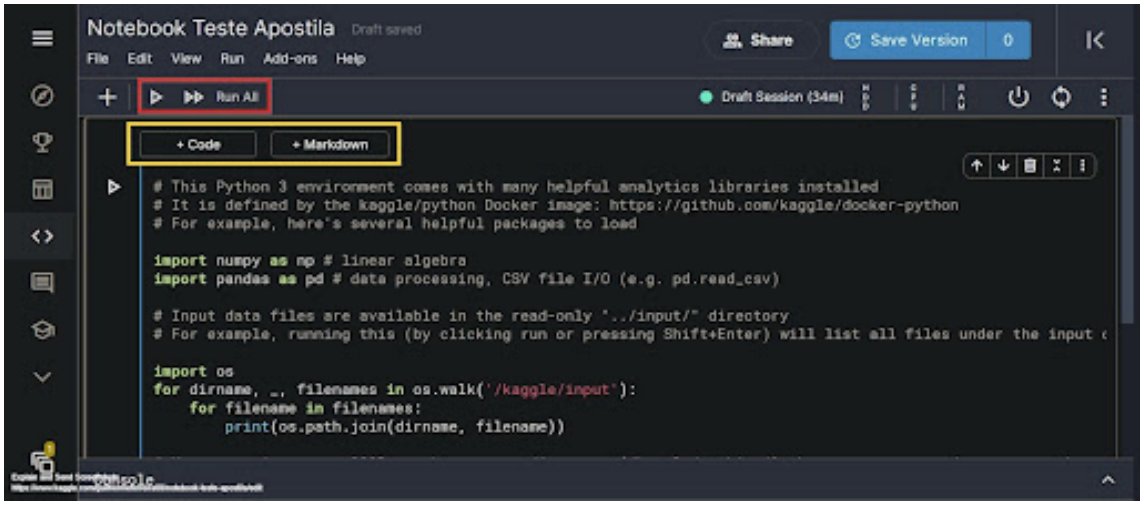

Figura 1.2: Tela de criação de notebook no Kaggle.

Inicialmente, é necessário nomear o notebook. O ambiente opera por meio de células, que podem ser executadas individualmente (botão *play*) ou todas de uma vez (botão *Run All*). O botão *+Code* adiciona novas células de código, enquanto o botão *+Markdown* permite inserir textos explicativos, facilitando a organização e documentação do projeto.

A célula inicial do notebook geralmente contém a importação de bibliotecas essenciais como `NumPy` e `Pandas`, além de configurações básicas para o ambiente de execução no Kaggle.

Na aba *File*, é possível realizar o upload de arquivos locais, importar conteúdos via URL ou diretamente de repositórios no GitHub, desde que estejam escritos em Python ou R.

1.4 Communities

A aba *Communities* é um espaço dedicado à interação entre os usuários da plataforma. Os fóruns são organizados por temas, como *Machine Learning* e *Data Visualization*, e servem para o compartilhamento de dúvidas, soluções e discussões sobre problemas enfrentados em competições ou aplicações reais, incluindo tópicos específicos como análises relacionadas à COVID-19.

Há também seções voltadas para iniciantes, onde é possível encontrar dicas, tutoriais e sugestões de usuários mais experientes sobre como iniciar uma carreira na área de Ciência de Dados.

1.5 Courses

O Kaggle oferece minicursos introdutórios gratuitos sobre temas centrais da Ciência de Dados. Cada curso apresenta uma fundamentação teórica, seguida por exercícios práticos desenvolvidos em notebooks interativos, promovendo a fixação dos conceitos abordados.

Para construir um modelo de análise de dados utilizando técnicas de Inteligência Artificial, recomenda-se a conclusão dos seguintes cursos introdutórios:

- **Python:** introduz os principais recursos da linguagem necessários ao desenvolvimento de modelos.
- **Intro to Machine Learning:** apresenta os fundamentos da criação, treinamento e avaliação de modelos de aprendizado de máquina.
- **Pandas:** ensina o uso da biblioteca `Pandas`, essencial para manipulação e análise de dados tabulares.
- **Data Visualization:** explora ferramentas para construção de gráficos e visualizações, indispensáveis durante a fase de análise exploratória.

Capítulo 2

Git e GitHub

Tempo estimado para conclusão: 1 dia

O *Git* é uma ferramenta de controle de versões amplamente utilizada no desenvolvimento de software. Com ela, é possível gerenciar de forma eficiente diferentes versões de um projeto, revisar alterações realizadas no código, restaurar versões anteriores e facilitar o trabalho colaborativo entre diversos desenvolvedores. Essa abordagem permite comparar modificações e decidir, em equipe, quais mudanças devem ser incorporadas à linha principal de desenvolvimento.

Para o uso efetivo do Git, alguns conceitos fundamentais devem ser compreendidos. Recomenda-se a realização do curso introdutório [3] do professor José de Assis (YouTube), especialmente até a aula 10, onde são apresentados os principais comandos em terminais como PowerShell ou Ubuntu. Os conceitos abordados nas videoaulas serão aprofundados ao longo deste capítulo.

Adicionalmente, é necessário criar uma conta na plataforma *GitHub*, onde será possível criar repositórios, editar arquivos, realizar *commits*, abrir *pull requests* e efetuar operações de *merge*. Recomenda-se a realização do tutorial introdutório do GitHub antes da aula 6 do curso mencionado.

2.1 Repositório

O repositório é a unidade fundamental de armazenamento de um projeto no Git. Ele contém todos os arquivos relacionados, incluindo subpastas e arquivos de diferentes formatos. Recomenda-se que cada projeto possua um único repositório central, com um arquivo inicial chamado `README`, que deve conter informações essenciais, como a descrição do projeto, seus objetivos e o progresso atual.

2.2 Branch

O conceito de *branch* refere-se à ramificação do projeto em diferentes linhas de desenvolvimento. A figura a seguir ilustra esse funcionamento:

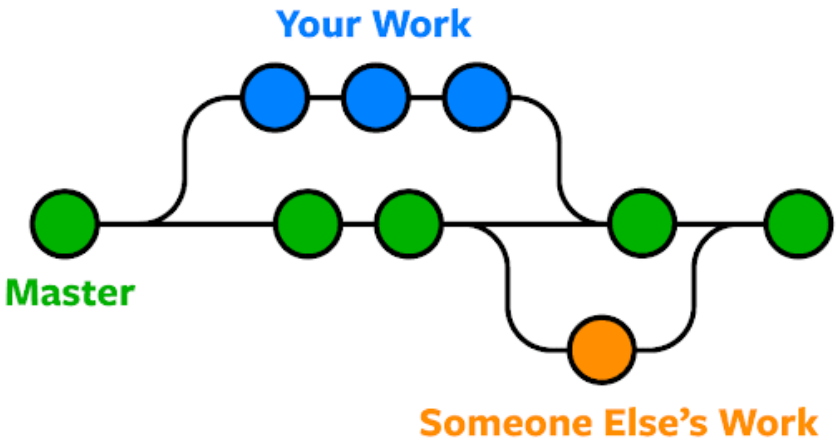

Figura 2.1: Esquema de funcionamento de *branches* no Git.

Na Figura 2.1, cada linha representa uma *branch*. A *master* (ou *main*), destacada em verde, é

a branch principal do projeto, onde são integradas as versões estáveis, geralmente após revisão e validação.

Demais *branches* são cópias da *main* no momento de sua criação, podendo ser modificadas de forma independente. Isso permite que diferentes desenvolvedores realizem experimentações ou correções sem comprometer a integridade do projeto principal.

2.3 Commit

Toda modificação realizada em um projeto versionado com Git deve ser registrada por meio de um *commit*. Ao efetuar um *commit*, o(a) desenvolvedor(a) deve adicionar uma mensagem descritiva, indicando o que foi alterado.

O histórico de *commits* permite a rastreabilidade das modificações, facilitando o retorno a versões anteriores caso haja necessidade de correção ou rollback. Cada *commit* é identificado por um código único, tornando a navegação entre versões eficiente e segura.

2.4 Pull Request

Após realizar modificações em uma branch secundária, é necessário solicitar sua integração à *main* por meio de um *pull request*. Esta ação representa a intenção de incorporar as alterações ao repositório principal, permitindo que outros membros da equipe revisem, comentem e aprovem as modificações.

O GitHub oferece uma interface intuitiva para esse processo, com funcionalidades que facilitam a comparação entre versões e o registro de discussões. Ferramentas integradas em IDEs também podem ser utilizadas localmente para esse fim.

2.5 Merge

O *merge* é a operação final de incorporação de uma branch à linha principal de desenvolvimento. Após a aprovação de um *pull request*, o *merge* consolida as alterações na *main*.

Uma vez concluído, é possível excluir a branch integrada, prática recomendada para manter o repositório limpo e organizado. No entanto, em casos de desenvolvimento paralelo ou experimentações simultâneas, pode ser interessante manter múltiplas branches ativas.

2.6 Objetivos de Aprendizado

Ao final deste capítulo, da décima aula do curso do professor José de Assis e da realização do tutorial introdutório no GitHub, espera-se que o(a) estudante seja capaz de:

- Criar um repositório localmente e na plataforma GitHub;
- Entender o funcionamento do Git e realizar *commits* corretamente;
- Navegar entre diferentes versões do projeto por meio do histórico de *commits*;
- Criar e alternar entre *branches*, tanto localmente quanto no GitHub;
- Comparar versões, inserir comentários e realizar o *merge* entre branches, de forma local ou pela interface do GitHub.

Capítulo 3

Python

Tempo estimado para conclusão: 2 dias

A linguagem de programação utilizada nas competições propostas neste curso será o *Python*. Para participar de forma eficiente dessas atividades, é fundamental o domínio de algumas funcionalidades básicas da linguagem, especialmente aquelas voltadas à manipulação de dados e à construção de modelos de Inteligência Artificial.

Como parte do processo formativo, recomenda-se a conclusão do [4]minicurso de Python oferecido pela plataforma Kaggle. Os principais conceitos abordados nesse curso serão retomados a seguir, acompanhados de exemplos práticos de aplicação.

Ao final desta seção, será proposto um exercício com o objetivo de verificar a compreensão dos conteúdos essenciais. A atividade pode ser realizada em qualquer ambiente de desenvolvimento com suporte à linguagem Python, como notebooks locais ou a plataforma Kaggle. O exercício foi adaptado do curso [5]Data Analysis with Python: Zero to Pandas, disponível na plataforma Jovian, e tem como objetivo conduzir uma análise exploratória simples de dados sem o uso de bibliotecas auxiliares.

3.1 Exercício — Análise de Sentimentos no Twitter

O objetivo da atividade é identificar, em uma lista de mensagens, aquelas que expressam sentimentos positivos (felizes) ou negativos (tristes), com base na presença de palavras-chave.

A seguir, é apresentada a lista de *tweets* que será analisada:

```
tweets = [
    "Wow, what a great day today!! #sunshine",
    "I feel sad about the things going on around us. #covid19",
    "I'm really excited to learn Python with @JovianML #zerotopandas",
    "This is a really nice song. #linkinpark",
    "The Python programming language is useful for data science",
    "Why do bad things happen to me?",
    "Apple announces the release of the new iPhone 12. Fans are excited.",
    "Spent my day with family!! #happy",
    "Check out my blog post on common string operations in Python. #zerotopandas",
    "Freecodecamp has great coding tutorials. #skillup"
]
```

1. Determine quantos *tweets* existem na lista apresentada.

2. Considere as listas de palavras abaixo, que identificam sentimentos felizes e tristes, respectivamente. Utilize o primeiro *tweet* como amostra e classifique-o como positivo ou negativo com base na presença dessas palavras.

   ```
   happy_words = ['great', 'excited', 'happy', 'nice', 'wonderful',
                  'amazing', 'good', 'best']

   sad_words = ['sad', 'bad', 'tragic', 'unhappy', 'worst']
   ```

   - **Dicas:**

   - Utilize uma variável booleana inicializada como `False` (por exemplo, `is_happy = False`) e altere seu valor para `True` caso alguma palavra feliz seja identificada.
   - Para percorrer o conteúdo do *tweet*, use um laço de repetição e o operador `in`, útil para verificar se uma determinada substring está presente em um texto.

3. A partir da lógica construída no item anterior, elabore um código que percorra todos os *tweets* da lista e contabilize a quantidade de mensagens felizes e tristes. Exiba os resultados ao final, acompanhados de mensagens explicativas.

   - **Dica:** Será necessário utilizar laços aninhados para verificar, em cada *tweet*, a ocorrência de palavras de ambas as listas.

Capítulo 4

Pandas

Tempo estimado para conclusão: 2 dias

A biblioteca `Pandas` é uma das principais ferramentas utilizadas na Ciência de Dados para manipulação e análise de dados estruturados. Por meio dela, é possível realizar operações como extração de informações relevantes, aplicação de funções estatísticas, edição de valores, inserção de novas colunas e outras transformações diretamente sobre os dados, de forma simples e eficiente.

Essa biblioteca é comumente aplicada à leitura de arquivos em formato CSV, embora também possa ser utilizada com outros tipos de dados tabulares. Para a familiarização com seus principais recursos, recomenda-se a conclusão do curso de Pandas [6] disponível na plataforma Kaggle.

Após a conclusão do curso, será proposta uma atividade prática com o objetivo de aplicar os conhecimentos adquiridos. Nesta atividade, o(a) estudante deverá utilizar as funcionalidades da biblioteca `Pandas` para responder a questões baseadas em um conjunto de dados real, disponível na plataforma Kaggle. Recomenda-se a utilização de um *Kaggle Kernel* para facilitar a importação de dados e a utilização das bibliotecas.

4.1 Explorando o Dataset "COVID-19 Dataset"

O conjunto de dados utilizado nesta atividade contém informações sobre a evolução da pandemia de COVID-19 em diversos países ao redor do mundo. Sua última atualização ocorreu em setembro de 2020. A análise será conduzida com base em dois arquivos principais:

- `covid_19_clean_complete.csv`: contém registros diários sobre a situação da pandemia em cada país ou província.
- `worldmeter_data.csv`: inclui informações demográficas, como a população de cada país e o continente correspondente.

Antes de iniciar a análise, é recomendado explorar a descrição do dataset, verificar o dicionário de variáveis e examinar as principais colunas dos arquivos CSV para compreender o escopo e as possibilidades de extração de conhecimento a partir dos dados.

Os itens a seguir propõem tarefas específicas que devem ser realizadas utilizando a biblioteca `Pandas`. Para cada item, são oferecidas instruções passo a passo que podem ser seguidas ou adaptadas conforme a preferência do(a) estudante.

1. **Análise exploratória inicial:** importar o arquivo `covid_19_clean_complete.csv` e responder:
   (a) Qual é a dimensão da base de dados (número de linhas e colunas)?;
   (b) Quais são os nomes das colunas?;
   (c) Qual o tipo de dado de cada coluna?;
   (d) Transformar a coluna de datas (caso esteja como tipo `object`) para o tipo `datetime64` utilizando `pd.to_datetime()`;
   (e) Obter um resumo estatístico dos dados com `.describe()`;
   (f) Verificar a presença de colunas com valores nulos (`NaN`).
2. **Análise por províncias da China:** identificar as 5 províncias chinesas com maior número de casos confirmados.

(a) Listar as províncias da China presentes no dataset;

(b) Filtrar o dataframe contendo apenas os registros dessas províncias;

(c) Selecionar apenas as colunas: `Confirmed`, `Active`, `Deaths`, `Recovered`;

(d) Agrupar os dados por província utilizando a função `groupby()`;

(e) Gerar um novo dataframe contendo as 5 províncias com o maior número de casos confirmados.

3. **Tratamento de valores nulos na coluna `Province/State`:** muitas entradas desta coluna estão ausentes. Para minimizar a perda de informação, deve-se unir os nomes das províncias com os respectivos países (quando houver mais de uma província) na coluna `Country/Region`. Exemplo: `"China_Hubei"`, `"China_Guangdong"`.

   (a) Escreva uma função que receba uma linha do dataframe como argumento e concatene os valores das colunas `Country/Region` e `Province/State`, caso esta última não seja nula;

   (b) Utilize a função `pandas.notna()` para verificar se a coluna `Province/State` contém valor válido;

   (c) Crie uma cópia do dataframe original;

   (d) Aplique a função anterior com o método `.apply()` sobre o dataframe;

   (e) Remova a coluna `Province/State` do novo dataframe.

4. **Ranking de mortes por milhão de habitantes por continente:** com base no arquivo `worldmeter_data.csv`, estime o número relativo de mortes por continente.

   (a) Importe o arquivo `worldmeter_data.csv`;

   (b) Selecione apenas as colunas relativas à população, países e continentes;

   (c) Agrupe os dados de COVID-19 por país (utilize o dataframe modificado no item anterior);

   (d) Faça a junção dos dois dataframes com base no nome dos países, mantendo apenas as colunas: número de mortes, população e continente;

   (e) Agrupe o novo dataframe por continente;

   (f) Crie uma nova coluna com o número de mortes por milhão de habitantes, utilizando a fórmula: `(mortes / população) * 10`$\hat{6}$;

   (g) Ordene os dados do novo dataframe, gerando um ranking de continentes com base na taxa de mortalidade ajustada pela população.

Capítulo 5

Visualização de Dados

Tempo estimado para conclusão: 2 dias

A visualização de dados é uma ferramenta essencial para cientistas de dados, pois permite interpretar tendências, identificar padrões e compreender relações entre variáveis de forma mais clara e intuitiva. Além disso, gráficos bem elaborados facilitam a comunicação de resultados para públicos diversos, incluindo aqueles que não possuem familiaridade com análise estatística ou algoritmos de inteligência artificial.

As bibliotecas mais utilizadas para visualização gráfica em Python são a `Matplotlib` e a `Seaborn`. Ambas oferecem grande flexibilidade para a criação de diversos tipos de gráficos, sendo fundamentais na etapa de análise exploratória de dados. Para o domínio das funcionalidades básicas dessas bibliotecas, recomenda-se a conclusão do curso "Data Visualization" [7] da plataforma Kaggle.

Os exercícios propostos neste capítulo deverão ser implementados em Notebooks do Kaggle. Recomenda-se que as células de código sejam acompanhadas de explicações, preferencialmente utilizando células de texto em `Markdown`, o que facilita a organização e a clareza do raciocínio.

A análise será feita com base no mesmo conjunto de dados do capítulo anterior — o `COVID-19 Dataset` — e utilizará as manipulações desenvolvidas previamente.

5.1 Gráficos de Linha (Line Charts)

Os gráficos de linha são particularmente úteis para visualizar séries temporais e observar o comportamento de variáveis contínuas ao longo do tempo. Neste exercício, será analisada a evolução temporal do número de mortes causadas pela COVID-19 em três regiões da Organização Mundial da Saúde (OMS): *Eastern Mediterranean*, *Americas* e *Europe*.

Após a construção do gráfico, responda:

1. Qual região apresentou maior número de mortes acumuladas?
2. Em que período a curva de mortes começou a crescer significativamente em cada região?
3. Como se comporta a taxa de crescimento das curvas?

1. Produzir um gráfico de linhas com a progressão temporal das mortes por região (usar `covid_19_clean_complete.csv`):
   (a) Importar as bibliotecas `matplotlib` e `seaborn`. Utilizar o comando `%matplotlib inline` para visualização direta no notebook;
   (b) Agrupar os dados pelas colunas `Date` e `WHO Region`, somando os valores de mortes;
   (c) Plotar as curvas para as regiões especificadas;
   (d) Adicionar título, rótulos de eixos, grade (`grid`) e legenda (`legend`);
   (e) Realizar a análise conforme as perguntas propostas.

5.2 Gráficos de Barras (Bar Charts)

Os gráficos de barras são indicados para comparações entre categorias discretas. Neste exercício, o objetivo é comparar o número total de mortes entre os continentes. A partir do gráfico, responda:

   1. Quais são os dois continentes com maior número de mortes?
   2. Qual continente apresentou o menor número de mortes?
   3. Qual sua hipótese para explicar os resultados observados?

1. Construir um gráfico de barras com os dados de mortes por continente (usar o dataframe do exercício 4.1-4):
   (a) Ajustar o tamanho do gráfico para (10,6) utilizando `plt.figure(figsize=(10,6))`;
   (b) Gerar o gráfico com `seaborn.barplot` ou equivalente;
   (c) Adicionar título e rótulos de eixos;
   (d) Responder às perguntas com base no gráfico produzido.

5.3 Gráficos de Dispersão (Scatter Plots)

Gráficos de dispersão são utilizados para explorar relações entre duas variáveis quantitativas. No exercício a seguir, investigaremos a relação entre população total e mortes por milhão de habitantes, nas três regiões com maiores médias de mortalidade proporcional.

Questões a responder com base no gráfico:

   1. Há concentração de dados em alguma região do gráfico?
   2. É possível observar uma tendência entre o número de mortes por milhão e a população total?
   3. Existe alguma relação perceptível entre as variáveis analisadas?

1. Produzir um gráfico de dispersão (usar `worldometer_data.csv`):
   (a) Agrupar o dataframe por continente e calcular a média da coluna `"Deaths/1M pop"`, ordenando para identificar os três continentes com maior valor;
   (b) Filtrar os dados do dataframe original para manter apenas os registros correspondentes a esses continentes;
   (c) Criar o gráfico de dispersão com:
      - Eixo X: `Deaths/1M pop`
      - Eixo Y: `Population`
      - Cores distintas para os três continentes
   (d) Interpretar o gráfico a partir das perguntas propostas.

5.4 Mapa de Calor de Correlação (Correlation Heatmap)

A correlação estatística entre variáveis indica o grau de associação entre elas, podendo ser direta (positiva), inversa (negativa) ou inexistente. A métrica de correlação de Pearson é frequentemente utilizada para esse fim, com valores variando de -1 (correlação inversa perfeita) a +1 (correlação direta perfeita), sendo 0 a ausência de relação linear.

Para visualizar as correlações entre todas as variáveis numéricas de um conjunto de dados, é comum a utilização de um *heatmap* (mapa de calor).

Mais informações podem ser obtidas neste artigo introdutório sobre correlação [8] e neste guia sobre o coeficiente de Pearson [9].

1. Gerar um mapa de calor das correlações com base no arquivo `worldometer_data.csv`:
   (a) Utilizar o método `.corr()` do `Pandas` para calcular a matriz de correlação;
   (b) Armazenar os valores em uma variável auxiliar;
   (c) Criar o *heatmap* utilizando a biblioteca `Seaborn`, com título e escala de cores adequada;
   (d) Selecionar ao menos duas variáveis e interpretar seus valores de correlação (sugestão: `"Population"` e `"Tests/1M pop"`).

Capítulo 6

Introdução ao Machine Learning

Tempo estimado para conclusão: 2 dias

Neste capítulo, será apresentada uma introdução à criação de modelos de inteligência artificial utilizando Machine Learning (ML). Essa abordagem consiste em treinar um modelo computacional a partir de um conjunto de dados, permitindo que ele reconheça padrões e faça previsões sobre novas entradas com o menor erro possível.

Inicialmente, trabalharemos com conjuntos de dados tabulares, ou seja, organizados em tabelas com linhas (observações) e colunas (variáveis). A partir desses *dataframes*, podemos separar as colunas em dois grupos:

- **Features (entradas)**: variáveis utilizadas para treinar o modelo;
- **Target (saída)**: variável que se deseja prever.

Para adquirir o conhecimento necessário sobre as ferramentas e bibliotecas usadas na construção de modelos, o aluno deve concluir o curso Intro to Machine Learning [10], disponível na plataforma Kaggle. Ao final do curso, o aluno será capaz de construir seu primeiro modelo preditivo e submetê-lo a uma competição real de ML.

A competição utilizada como exercício será a clássica Titanic - Machine Learning from Disaster [11]. Ela propõe prever, com base em dados dos passageiros, quem sobreviveu ao naufrágio do Titanic. É um problema de classificação binária e ideal para iniciantes. Certifique-se de se inscrever na competição antes de iniciar os exercícios.

Observação: no curso do Kaggle, os exemplos utilizam algoritmos de regressão, como árvores de decisão (*decision trees*), que são apropriados para prever valores contínuos. A métrica utilizada nesse contexto é a MAE (Mean Absolute Error), que calcula a média dos erros absolutos nas previsões.

Entretanto, no problema do Titanic, a saída é categórica (sobreviveu ou não), ou seja, um problema de *classificação*. Nestes casos, usamos classificadores (como o Random Forest Classifier) e a métrica recomendada é a **accuracy** — a proporção de previsões corretas. Para entender melhor a diferença entre regressão e classificação, consulte este artigo [12].

Este exercício foi inspirado no notebook [13] publicado por Nadim Tamer. Recomendamos também a leitura de outros notebooks disponíveis na competição para comparar estratégias e aperfeiçoar sua solução.

6.1 Importação e visualização do banco de dados

1. **Importar as bibliotecas necessárias:**
   (a) `numpy` como `np`;
   (b) `pandas` como `pd`;
   (c) `matplotlib.pyplot`;
   (d) `seaborn` como `sns`;
   (e) Comando `%matplotlib inline` para exibir gráficos no notebook;
   (f) Fixar a semente aleatória com `np.random.seed(0)`.
2. **Importar os dados e realizar uma análise inicial:**

(a) Carregar os arquivos `train.csv` e `test.csv` localizados em `kaggle/input/titanic/`;
(b) Gerar estatísticas descritivas do conjunto de treino com `.describe()`.

3. **Análise exploratória preliminar:**

(a) Identificar quais colunas são categóricas e quais são numéricas (diferenciando entre contínuas e discretas). Use o método `.info()`;
(b) Verificar a presença de valores ausentes (NaN) com `DataFrame.isnull().sum()`;
(c) Criar novos dataframes contendo apenas as features numéricas (para treino e teste);
(d) Remover linhas com valores faltantes no conjunto de treino usando `DataFrame.dropna()`;
(e) Substituir os NaN por 0 no conjunto de teste, utilizando `DataFrame.fillna(0)`.

6.2 Exploração do banco de dados

Objetivo: Visualizar a influência das features nas taxas de sobrevivência. Para isso, serão criados gráficos com `sns.barplot`, que comparam médias de sobrevivência para diferentes faixas ou categorias.

1. **Análise da feature "Age":**

(a) Criar uma lista com rótulos (ex.: "(0–10)", "(10–20)", etc.);
(b) Agrupar a coluna "Age" em 8 faixas utilizando `pandas.cut()`, com rótulos apropriados;
(c) Criar uma nova coluna com os grupos de idade;
(d) Utilizar `sns.barplot()` para visualizar a taxa de sobrevivência por faixa etária;
(e) Interpretar os resultados obtidos no gráfico.

2. **Analisar graficamente as demais features numéricas discretas:**

(a) Criar gráficos individuais para cada feature;
(b) Produzir interpretações sobre como cada variável pode impactar a previsão de sobrevivência;
(c) A feature "Fare" pode ser omitida nesta etapa.

6.3 Treinamento do modelo e submissão das predições

Agora, será construído o primeiro modelo preditivo. A etapa inclui a separação entre treino e validação, treinamento com Random Forest Classifier e geração de um arquivo para submissão na competição.

1. **Separação entre treino e validação:**

(a) Criar a variável `X` com apenas as features numéricas (remover a coluna "PassengerId");
(b) Criar a variável `y` contendo os valores da coluna `Survived`;
(c) Utilizar `train_test_split` com validação de 20%;

2. **Treinamento e validação do modelo:**

(a) Importar `RandomForestClassifier` da biblioteca `sklearn.ensemble`;
(b) Importar `accuracy_score` de `sklearn.metrics`;
(c) Treinar o classificador com os dados de treino;
(d) Realizar predições com os dados de validação;
(e) Avaliar a acurácia do modelo utilizando `accuracy_score`;
(f) Discutir a qualidade da predição obtida.

3. **Geração e submissão da predição:**

(a) Treinar o modelo novamente, agora com todos os dados de treino disponíveis;
(b) Armazenar os valores da coluna "PassengerId" do conjunto de teste;
(c) Realizar as predições com os dados de teste;
(d) Criar um `DataFrame` com os campos `PassengerId` e `Survived`;
(e) Exportar o dataframe para CSV com `DataFrame.to_csv()`;
(f) Salvar a versão no Kaggle ("Save Version") e submetê-la à competição.

Parte II

Machine Learning Clássico

Capítulo 7

Data Cleaning

Tempo estimado para conclusão: 2 dias

Antes de realizar análises aprofundadas ou treinar modelos de inteligência artificial, é essencial preparar adequadamente o banco de dados. Muitas vezes, os dados brutos vêm com inconsistências, valores ausentes (NaN), formatos inadequados ou colunas pouco informativas. Essa etapa de preparação é chamada de **Data Cleaning** (ou limpeza de dados).

O processo de limpeza tem como objetivo organizar e transformar os dados para que eles estejam prontos para análise e modelagem. É uma etapa fundamental tanto para análise estatística quanto para aplicações com modelos de Machine Learning.

Para conhecer as principais técnicas, ferramentas e métodos utilizados em limpeza de dados, recomenda-se que o aluno complete o minicurso Data Cleaning [14] da plataforma Kaggle.

Durante os capítulos anteriores, especialmente nos exercícios 5.1-D e 6.1-C/D, já realizamos algumas operações básicas de limpeza. Após a conclusão do minicurso, sugerimos revisar esses exercícios à luz dos novos conhecimentos adquiridos, para consolidar os conceitos de forma prática.

Objetivo do Capítulo

Neste capítulo, iremos aplicar os conceitos aprendidos para reestruturar e melhorar a versão inicial do modelo de IA desenvolvida no Capítulo 6 (Titanic). Em vez de criar um novo notebook do zero, sugerimos duplicar e editar a versão anterior, aproveitando os códigos já desenvolvidos e incorporando melhorias.

A partir de agora, o banco de dados de `teste` também será analisado e limpo, assim como o banco de `treino`. Isso permitirá que o modelo final seja construído com maior qualidade e precisão. Utilizaremos como base o notebook [15] de LD Freeman, que propõe uma abordagem organizada e eficiente para análise do Titanic Dataset.

Exercício: Limpeza e transformação de dados

1. **Análise inicial:**
   (a) Liste todas as colunas presentes nos dataframes de treino e teste;
   (b) Identifique os tipos de dados de cada coluna (numérico, texto, data, etc);
   (c) Verifique quantos valores nulos existem em cada coluna de ambos os dataframes;
   (d) Faça uma cópia dos dataframes originais para preservar os dados brutos:
   - `train_clean = train.copy()`
   - `test_clean = test.copy()`

2. **Tratamento de valores ausentes:**

   Em geral, existem duas estratégias principais para lidar com valores ausentes:

   - *Descartar a coluna ou linha afetada;*
   - *Preencher os valores ausentes com algum valor calculado.*

   Vamos aplicar diferentes abordagens dependendo das características de cada coluna:

(a) Remova as colunas `"Cabin"` e `"Ticket"`, que contêm muitos valores nulos e pouca utilidade informacional para o modelo;

(b) Preencha os valores nulos da coluna `"Age"` com a **mediana** dos dados, pois é uma variável contínua com muitos NaNs;

(c) Preencha os valores ausentes em `"Fare"` também com a mediana, visto que é outra variável contínua;

(d) Para a coluna categórica `"Embarked"`, preencha os valores ausentes com a **moda** (valor mais frequente).

3. **Criação de grupos a partir de variáveis contínuas:**

   Agrupar variáveis contínuas pode facilitar a visualização e ajudar modelos baseados em árvores de decisão. Vamos criar duas novas features:

   (a) Use o método `pd.cut()` para dividir a coluna `"Age"` em 5 intervalos de mesma largura (ex.: 0–16, 16–32, ...);

   (b) Use o método `pd.qcut()` para dividir a coluna `"Fare"` em 6 intervalos com quantidades iguais de dados, mesmo que os intervalos tenham tamanhos diferentes;

   (c) Para entender melhor as diferenças entre `cut()` e `qcut()`, consulte esta discussão [16].

Próximos passos

Após realizar a limpeza e a criação de novas variáveis, os dados estarão mais prontos para análises, visualizações e treinamento de modelos. No próximo capítulo, trabalharemos com a codificação de variáveis categóricas (feature encoding) e faremos a construção de uma nova versão do modelo, buscando melhorar sua performance.

Exercício Final: Reflexão sobre as escolhas de limpeza

Para consolidar os aprendizados deste capítulo, responda às perguntas abaixo com base nas ações que você tomou ao limpar o banco de dados do Titanic. As respostas devem ser registradas em uma célula de Markdown no seu notebook.

1. **Por que você optou por remover as colunas “Cabin” e “Ticket”? Há algum cenário em que essas colunas poderiam ser úteis?**

2. **Você escolheu preencher os valores ausentes de “Age” e “Fare” com a mediana. Em quais situações a média ou moda seriam mais apropriadas? Justifique.**

3. **A substituição dos NaNs da coluna “Embarked” foi feita com a moda. Quais os riscos de aplicar esse tipo de preenchimento em variáveis categóricas?**

4. **Você criou faixas para as variáveis contínuas “Age” e “Fare”. Quais são as vantagens de agrupar variáveis contínuas? Há desvantagens?**

5. **Durante a limpeza, quais decisões você acredita que mais impactaram positivamente a preparação do banco de dados para modelagem? Justifique.**

6. **Se o dataset fosse muito maior e mais complexo, como você adaptaria seu processo de data cleaning para lidar com múltiplas fontes de dados e diferentes tipos de ruídos?**

Capítulo 8

Aperfeiçoamento da modelagem

2 dias para completar capítulo

Até agora, foram utilizados apenas algoritmos simples para a geração de modelo, além de que só foram previstos resultados por meio da utilização de features categóricas. Esses modelos foram úteis para criar o que podemos chamar de **Baseline Model**, que são modelos criados com as estratégias mais básicas para servir de comparação para novos modelos utilizando estratégias mais complexas, de modo a facilitar a definição de quais métodos utilizar.

Para este capítulo, é necessário que o aluno realize o curso Intermediate Machine Learning [17], por completo. Dessa forma, se tornará capaz de manipular e realizar **encoding** de variáveis categóricas, produzir modelos utilizando **pipeline**, utilizar o **cross validation** para avaliar predições, aprender como utilizar o **gradient boosting** para criar modelos de predições e, finalmente, detectar se existe vazamento de dados na produção do modelo.

8.1 Encoding de Variáveis Categóricas

Nesta etapa da produção do notebook serão postas em prática novas estratégias da análise e manipulação de dados. Para facilitar a produção do código, algumas funções serão criadas, então vale a pena revisar este conteúdo se ainda não for dominado pelo aluno. Também é indicado que outros notebooks já publicados para esta competição sejam utilizados como referência, além disso, o aluno será livre para testar outras estratégias que conhecer, para melhorar as previsões do modelo.

Até aqui não foram utilizadas as variáveis categóricas, não numéricas, o que acarreta em uma grande perda de informações, já que algumas delas dizem muito sobre a variável que deve ser prevista para o modelo. Para podermos utilizar essas features teremos que realizar o encoding [18], ou seja, transformar seus valores em números que as representem. Existem algumas estratégias para realizar o encoding, neste capítulo utilizaremos três: **one-hot encoding**, **ordinal encoding** e **label encoding**.

1. Será necessário realizar o encoding das variáveis categóricas, no momento, três estratégias que serão utilizadas são o one-hot, label encoding e ordinal encoding. Para isso, siga:
   (a) Utilize o método `.info()` para relembrar quais são as features categóricas;
   (b) Utilize o método .select_dtypes para identificar o nome das colunas numéricas e categóricas. Para numéricas identifique formatos `int64` e `float64`, para categóricas identifique `object`;
   (c) Identifique quais features categóricas possuem uma ordem clara e quais não:
      i. Para as features com ordem, o **label encoder** é mais indicado como primeira abordagem;
      ii. Para features sem ordem definida, o **one-hot encoding** pode ser a melhor opção;
   (d) Crie uma função que realize o one-hot encode e, como saída, retorne um novo dataframe com as colunas que resultam da codificação, devidamente nomeadas, ao invés das features categóricas:
      i. Os argumentos deverão ser: o dataframe a ser manipulado e as colunas que serão codificadas;
      ii. Faça uma cópia do dataframe original;

iii. Crie um loop que realize o encoding das colunas passadas como argumento, as features resultantes devem conter o nome do valor que representam (use o método .get_feature_names_out);

iv. Ainda dentro do loop, remova do dataframe copiado as features categóricas originais e mescle as novas features criadas pelo one-hot encoder;

(e) Crie novas features para realizar o label encoding das features que contêm ordenamento claro, e depois remova as features categóricas originais que foram codificadas.

- **Observação:** é importante lembrar que esse encoding deve ser realizado tanto nos dataframes de treino quanto nos de teste.

2. Realize um novo treinamento e avaliação do modelo, seguindo os passos do item 6.1 g), mas agora utilizando o dataframe resultante do encoding das variáveis categóricas. Compare a precisão obtida com o modelo anterior e analise a evolução.

8.2 Pipelines

Essa estratégia serve para tornar o código mais otimizado e direto. Com o uso de pipelines, é possível realizar várias operações — de limpeza a modelagem — com poucos comandos. As bibliotecas criadas para pipelines são otimizadas para performance, tornando o código mais rápido e menos custoso computacionalmente. Caso haja dúvidas, consulte este artigo [19] sobre pipelines usando Scikit-Learn.

No capítulo a seguir, utilizaremos pipelines para realizar parte dos processos já realizados anteriormente no banco de dados do Titanic, além de testar diferentes estratégias de encoding e limpeza para avaliar e comparar os resultados, visando encontrar as melhores abordagens.

1. Crie uma função que produza e avalie pipelines com diversas estratégias de imputação de valores nulos e encoding de variáveis categóricas.

   (a) Restaure os valores nulos das colunas “Age” e “Embarked” no dataframe de treino, usando os dados originais (lembre-se que uma cópia do banco original foi criada no início do notebook).

   (b) Crie uma função que produza e avalie pipelines conforme os argumentos:

   - **data**: dataframe a partir do qual será produzido o modelo;
   - **encoder**: estratégia de encoding para variáveis categóricas (one-hot ou ordinal);
   - **model**: algoritmo que irá gerar o modelo (inicialmente, Random Forest Classifier);
   - **numerical_imputer**: estratégia para substituir valores nulos nas features numéricas (como “mean” ou “median”);
   - **categorical_imputer**: estratégia para preencher valores nulos nas features categóricas (use “most frequent”);

   (c) Defina as features e o target para treinamento e faça a separação treino/validação (train-test split).

   (d) Crie um preprocessor que:

   - para as features numéricas, utilize o imputador indicado por `numerical_imputer`;
   - para as features categóricas, utilize o imputador indicado por `categorical_imputer` e o encoder indicado por `encoder`.

   (e) Monte uma pipeline usando o preprocessor e o modelo fornecido.

   (f) Treine a pipeline, gere predições e avalie utilizando `accuracy_score`.

   (g) A função deve retornar o valor de acurácia (em porcentagem).

2. Defina listas com estratégias para imputação e encoding:

   (a) Estratégias para imputing:

   - **mean**: substitui valor nulo pela média;
   - **most_frequent**: substitui pelo valor mais frequente;
   - **median**: substitui pela mediana;
   - **zero**: substitui todos os valores nulos por 0;

(b) Estratégias para encoding:

- **one-hot encoder**;
- **ordinal encoder** (recomenda-se usar o OrdinalEncoder no lugar do LabelEncoder para features categóricas);

3. Crie um loop aninhado que teste todas as combinações possíveis dessas estratégias e imprima os resultados de avaliação de cada pipeline produzida.

4. Analise se alguma das estratégias usadas isoladamente gerou precisão maior que as estratégias combinadas utilizadas antes das pipelines.

8.3 Cross Validation e Gradient Boosting

Após criar um modelo, uma etapa fundamental em ciência de dados é avaliá-lo para entender sua generalização e precisão. Quanto melhor a estratégia de avaliação, maior a certeza sobre a qualidade do modelo. Uma técnica robusta para isso é o **cross validation**, que será utilizado neste capítulo.

Outro algoritmo amplamente usado para melhorar precisão e generalização é o **Gradient Boosting**. Assim como o Random Forest, ele utiliza múltiplas decision trees, mas com a diferença de que cada modelo sucessivo corrige os erros do anterior, tornando o conjunto final mais robusto.

1. Utilize o cross validation para avaliar o modelo que obteve o melhor desempenho nos testes anteriores. Utilize a métrica `scoring="accuracy"`. A avaliação deve ser a média das scores nas divisões.

2. Use o Gradient Boosting Classifier para gerar um novo modelo, usando o dataframe tratado com as melhores estratégias.

   (a) Importe o GradientBoostingClassifier, definindo `random_state=0` e `n_iter_no_change=100` (pesquise o significado desses parâmetros na documentação);

   (b) Defina as features e o target para treinamento;

   (c) Treine e avalie o modelo usando cross validation;

   (d) Compare a pontuação obtida com a do Random Forest e determine qual algoritmo teve melhor desempenho.

Capítulo 9

Feature Engineering

3 dias para completar capítulo

Até o momento, foram criadas apenas features a partir do encoding das variáveis categóricas e agrupamento das contínuas. Neste capítulo, serão criadas novas colunas a partir da utilização de estratégias de análise estatística e operações matemáticas entre as features já existentes. Para a compreensão dessa parte da criação do modelo, será necessário realizar o curso de Feature Engineering do Kaggle.

Dois conceitos importantes são "Supervised Learning" e "Unsupervised Learning". No **Supervised Learning**, os algoritmos trabalham com dados rotulados, ou seja, existe um banco de dados de treino que contém as respostas corretas das predições. A partir das características de cada dado, o modelo é ajustado para aumentar sua precisão. Já nos algoritmos de **Unsupervised Learning**, não há dados rotulados; os algoritmos encontram relações entre as features sem a comparação com respostas previamente fornecidas (sem o target). Para mais informações, consulte o artigo da IBM [20].

No contexto de Feature Engineering, os algoritmos de Unsupervised Learning serão usados para encontrar novas relações ocultas entre as features, sem utilizar o target, contudo, o modelo que está sendo construído é considerado Supervised Learning, pois as predições devem ser ajustadas de acordo com a coluna "Survived" já previamente preenchida no banco de dados de treino.

Feature Engineering é composto por três áreas: **feature extraction**, **feature construction** e **feature selection**. A feature extraction tem o objetivo de reduzir a dimensionalidade das features aplicando operações de mapeamento. Feature construction expande o espaço das features, criando novas dimensões a partir de operações entre as originais. Já feature selection diminui a quantidade de features detectando redundâncias e selecionando as mais importantes.

9.1 Feature Selection

O objetivo é selecionar features que sejam pouco correlacionadas entre si, para evitar redundância, e que tenham alta correlação com o target, para que o modelo possa predizer melhor o alvo.

Neste capítulo utilizaremos o **Mutual Information** para avaliar a importância das features. Ele mede quanto a variação de cada feature explica a variação do target.

1. Avaliação da importância das features usando Mutual Information Score:
   (a) Identifique quais features são discretas (note que "Age" e "Fare" não são discretas);
   (b) Crie uma função que produza um gráfico com o Mutual Information Score para cada feature, em relação ao target:
      i. Argumentos da função:
         - features;
         - target;
         - features discretas;
      ii. Calcule os scores e crie uma série pandas ordenada de forma decrescente;
      iii. Produza um gráfico de barras com os scores de cada feature;
      iv. A função deve retornar a série com os scores.
   (c) Defina as features e o target para a função;

(d) Produza o gráfico chamando a função com os argumentos necessários.

2. Análise aprofundada das features mais influentes (“Fare”, “Sex” e “Age”):

   (a) Crie dois gráficos do tipo sns.histplot para “Fare”:

      i. Distribuição total das taxas;
      ii. Distribuição relacionada com o target;
      iii. Analise se a curva é normal ou não e discuta a influência de “Fare” no target.

   (b) Crie quatro histogramas para analisar “Age” em relação a “Sex” (utilize plt.subplots para visualizar múltiplos gráficos):

      i. Distribuição da idade dos passageiros por sexo;
      ii. Distribuição da idade dos sobreviventes por sexo;
      iii. Distribuição da idade dos homens com “Survived” como camada;
      iv. Distribuição da idade das mulheres com “Survived” como camada;
      v. Analise a influência da idade e do sexo nas chances de sobrevivência.

9.2 Feature Construction

Aqui, criamos novas colunas a partir de operações entre as features existentes, buscando extrair informações relevantes não explícitas nos dados.

1. Extração de título a partir da feature “Name”:

   (a) Separe o título de cada passageiro extraindo a substring entre a vírgula e o ponto usando o método str.split, crie uma nova coluna “Title” e remova a coluna “Name”.

2. Crie duas novas features relacionadas à família:

   (a) “FamilySize”: soma das features “SibSp” e “Parch”;

   (b) “IsAlone”: booleano que indica se o passageiro está sozinho (True se não houver familiares a bordo, False caso contrário).

9.3 Feature Extraction

Feature Extraction visa reduzir a dimensionalidade do banco, criando novas features que resumem as informações das originais.

Aqui serão aplicados dois métodos não supervisionados em “Fare” e “Age”: **Clustering** (com K-means) e **Principal Component Analysis (PCA)**.

1. Agrupamento via K-means em “Fare” e “Age”:

   (a) Visualize a distribuição com um scatter plot (“Age” no eixo x e “Fare” no eixo y);

   (b) Identifique e remova outliers (exemplo: valores de “Fare” acima de 500);

   (c) Crie um dataframe apenas com passageiros com “Fare” ¡ 500;

   (d) Execute K-means com 6 clusters;

   (e) Adicione a coluna de clusters ao dataframe original;

   (f) Faça dois gráficos:

      i. Scatter plot de “Fare” x “Age” com os clusters;
      ii. Gráfico de barras da média da chance de sobrevivência por cluster;

   (g) Analise qualitativamente cada grupo criado e suas chances médias de sobrevivência.

2. Trate os valores nulos da nova feature de cluster:

   (a) Use `.loc` para localizar linhas com valores nulos na coluna de clusters;

   (b) Analise as características desses passageiros;

   (c) Atribua manualmente o grupo mais adequado a esses outliers;

   (d) Verifique se as substituições foram feitas corretamente.

3. Avalie a relevância da feature cluster usando Mutual Information (como em 9.1);
4. PCA nas features “Fare” e “Age”:
   (a) Faça o scaling das features usando MinMaxScaler entre 0 e 1;
   (b) Visualize histogramas das features escaladas para confirmar o processo;
   (c) Aplique PCA para criar uma nova feature que combine “Fare” e “Age”;
   (d) Analise os componentes principais e interprete o que eles representam;
   (e) Avalie a nova feature criada pelo PCA para verificar se é viável substituir “Fare” e “Age” por ela.

9.4 Target Encoding

A feature “Title” criada na etapa de feature construction possui muitos valores únicos, por isso é adequada para **target encoding**, uma estratégia de Supervised Learning que codifica a feature categórica baseada na relação com o target.

Utilizaremos o algoritmo **MEstimateEncoder** para isso.

1. Realize o encoding da feature “Title”:
   (a) Crie um encoder MEstimateEncoder com parâmetro $m = 5$;
   (b) Gere um dataframe com os valores codificados da feature “Title”, usando o target “Survived”.
2. Analise a distribuição dos valores codificados:
   (a) Plote um histograma da feature “Survived”;
   (b) Sobreponha um `kdeplot` da distribuição dos valores da feature “Title” codificada;
   (c) Adicione legenda para melhor interpretação.

Parte III

Deep Learning

Capítulo 10

Redes Neurais Simples

2 dias para completar capítulo

A partir deste capítulo será introduzida uma nova estratégia para Machine Learning: **Deep Learning**. Essa abordagem utiliza redes neurais para capturar padrões complexos nos dados de treinamento. Redes neurais são ferramentas poderosas que permitem modelar relações não lineares, trabalhar com grandes volumes de dados e lidar com dados não estruturados, como áudios, textos e imagens.

O framework utilizado nos exercícios será o `keras`. Para uma introdução prática, recomendo o curso Intro to Deep Learning [21] do Kaggle, que traz exemplos, exercícios e a construção de modelos com Keras.

Inicialmente, serão usados dados tabulares, porém com muito mais informações do que nos exemplos anteriores. O dataset Weather in Szeged 2006-2016 [22] contém dados climáticos de Szeged, Hungria, de 2006 a 2016. O objetivo será prever o sumário diário do clima ("Daily Summary") a partir das demais colunas.

O estudo de caso baseia-se no notebook Weather prediction, regression, neural model [23], de Salma Maamouri.

10.1 Exploração do banco de dados

Antes de treinar o modelo, é fundamental explorar e entender os dados para identificar possíveis problemas.

1. Importe e visualize o banco de dados; identifique valores faltantes e estatísticas básicas:
   (a) Importe o dataset usando `pandas`;
   (b) Conte os valores faltantes em cada coluna;
   - Qual sua hipótese para a existência desses valores faltantes?

   (c) Gere estatísticas descritivas (momentos estatísticos);
   - Alguma coluna parece fora do comum? Por quê?

   (d) Plote histogramas das colunas usando o método `.hist`;
   - Identifique anomalias, como na coluna "Pressure (millibars)". Explique o que está errado;
   - Use `.loc` para encontrar a quantidade de valores errados.

2. Classifique as features em categóricas e numéricas:
   (a) Use `.select_dtypes` para identificar features numéricas (tipos "int64" e "float64");
   (b) Identifique features categóricas (tipo "object");
   - Há uma feature do tipo "object" que não é categórica. Qual?

   (c) Analise os valores únicos de cada feature categórica com `.value_counts` para garantir que não há valores únicos que atrapalhem a divisão treino/validação;
   - Identifique e liste valores que ocorrem mais de uma vez e que serão removidos futuramente.

10.2 Limpeza do banco de dados

Com base na exploração, vamos limpar os dados, substituindo valores faltantes, aplicando encoding em categóricas e preparando os dados para o treinamento.

1. A feature "Formatted Date" indica a data e está no formato texto com fuso horário ("+0200"). Para usá-la:
   (a) Remova o texto "+0200" usando `str.split`;
   (b) Converta a coluna para o tipo datetime usando `pd.to_datetime`.
2. Extraia novas features temporais da coluna convertida:
   (a) Use os métodos `.dt.hour`, `.dt.day`, `.dt.month` e `.dt.year`;
   (b) Remova a coluna original de data.
3. Crie uma pipeline para limpeza dos dados, incluindo imputação e encoding:
   (a) Remova a coluna "Loud Cover" (apenas valores nulos) e linhas cujos valores na coluna "Summary" ocorrem apenas uma vez;
      - Use `.loc` para localizar essas linhas.
   (b) Configure imputers:
      - Para variáveis numéricas, use SimpleImputer substituindo valores faltantes pela média, considerando 0 como valor faltante para "Pressure (millibars)";
      - Para variáveis categóricas, crie pipeline que primeiro substitui valores faltantes por texto com SimpleImputer e depois aplica OrdinalEncoder.
   (c) Use ColumnTransformer para aplicar os passos acima separadamente para numéricas e categóricas.
4. Separe os dados em treino e validação, aplique a pipeline e encode o target:
   (a) Defina features (todas as colunas menos "Daily Summary") e target ("Daily Summary");
   (b) Separe dados com train_test_split, com 30% para teste e usando stratify para manter a distribuição do target;
   (c) Ajuste a pipeline nos dados de treino e transforme os dados de treino e teste;
   (d) Use LabelEncoder para codificar o target, ajustando com treino e transformando teste.

10.3 Treinamento e Avaliação da rede neural

Com os dados limpos, construiremos a rede neural com Keras, treinaremos e avaliaremos seu desempenho.

1. Defina a arquitetura da rede:
   (a) Determine o tamanho da entrada (número de features) e saída (número de classes no target);
      - Use `.shape` para as features e `.unique` para contar as classes do target.
   (b) Construa a rede com:
      - Camada de entrada com Batch Normalization;
      - Primeira camada oculta: 256 neurônios, ativação "relu", Batch Normalization e Dropout de 0.3;
      - Segunda camada oculta idêntica à primeira;
      - Camada de saída: número de neurônios igual ao número de classes, ativação "softmax".
   (c) Prepare o target para classificação multi-classe:
      - Use `keras.utils.to_categorical` para transformar os targets de treino e teste;
      - Observe as dimensões resultantes e conte o número de classes.
2. Configure o compilador do modelo:

- Otimizador: “adam”;
- Função de perda: “categorical_crossentropy”;
- Métrica: “categorical_accuracy”.

3. Treine o modelo com early stopping para evitar overfitting:
   (a) Configure early stopping com paciência de 10 épocas e variação mínima de perda de 0.001;
   (b) Execute o treino usando:
      - Features e target de treino para treino;
      - Dados de teste para validação;
      - Batch size = 3000;
      - Até 100 épocas;
      - Early stopping habilitado.
   (c) Após o treino, converta o histórico para um `pandas.DataFrame` e plote gráficos de perda e acurácia em função das épocas usando gráficos de linha.

Referências Bibliográficas

[1] A. Inc., "Download anaconda." https://www.anaconda.com/download, 2025. Acessado em: 15 de julho de 2025.

[2] Kaggle, "Kaggle: Your machine learning and data science community." https://www.kaggle.com/, 2025. Acessado em: 15 de julho de 2025.

[3] J. de Assis, "Curso de git e github - o que é git? - o que é github? - definições e conceitos importantes." https://www.youtube.com/watch?v=FF1f4bKYhoo&list=PLbEOwbQR9lqzK14I7OOeREEIE4k6rjgIj, 2025. Acessado em: 15 de julho de 2025.

[4] C. Morris, "Learn python — kaggle." https://www.kaggle.com/learn/python, 2017. Acessado em: 15 de julho de 2025.

[5] A. N. S, "Data analysis with python: Zero to pandas." https://jovian.ai/learn/data-analysis-with-python-zero-to-pandas, 2021. Acessado em: 15 de julho de 2025.

[6] A. Bilogur, "Learn pandas — kaggle." https://www.kaggle.com/learn/pandas, 2017. Acessado em: 15 de julho de 2025.

[7] A. Cook, "Learn data visualization — kaggle." https://www.kaggle.com/learn/data-visualization, 2017. Acessado em: 15 de julho de 2025.

[8] R. Nagy, "Correlation explained: What is it, how to use and why it differs from causation." https://medium.com/@rolandnagydata/correlation-explained-what-is-it-how-to-use-and-why-it-differs-from-breakcausation-0e80734b78 2024. Acessado em: 15 de julho de 2025.

[9] GeeksforGeeks, "Pearson correlation coefficient." https://www.geeksforgeeks.org/pearson-correlation-coefficient, 2025. Acessado em: 15 de julho de 2025.

[10] D. Becker, "Intro to machine learning." https://www.kaggle.com/learn/intro-to-machine-learning, 2018. Kaggle.

[11] W. Cukierski, "Titanic - machine learning from disaster." https://kaggle.com/competitions/titanic, 2012. Kaggle.

[12] GeeksforGeeks, "Classification vs regression in machine learning." https://www.geeksforgeeks.org/ml-classification-vs-regression/, 2025. Acessado em: 15 de julho de 2025.

[13] N. Tamer, "Titanic survival predictions (beginner)." https://www.kaggle.com/nadintamer/titanic-survival-predictions-beginner, 2018. Kaggle.

[14] R. Tatman, "Data cleaning." https://www.kaggle.com/learn/data-cleaning, 2018. Kaggle.

[15] L. Freeman, "A data science framework: To achieve 99% accuracy." https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy, 2017. Kaggle.

[16] WillZ, "What is the difference between pandas.qcut and pandas.cut?." https://stackoverflow.com/questions/30211923/what-is-the-difference-between-pandas-qcut-and-pandas-cut, 2015. Acessado em: 15 de julho de 2025.

[17] A. Cook, "Intermediate machine learning." https://www.kaggle.com/learn/intermediate-machine-learning, 2018. Acessado em: 15 de julho de 2025.

[18] B. Roy, "All about categorical variable encoding." https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02, 2018. Acessado em: 15 de julho de 2025.

[19] J. P. Nogueira, "Como usar pipelines no scikit-learn." https://medium.com/data-hackers/como-usar-pipelines-no-scikit-learn-1398a4cc6ae9, 2019. Acessado em: 15 de julho de 2025.

[20] J. Delua, "Supervised versus unsupervised learning: What's the difference?." https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning, 2021. Acessado em: 15 de julho de 2025.

[21] R. Holbrook, "Intro to deep learning." https://www.kaggle.com/learn/intro-to-deep-learning, 2018. Acessado em: 15 de julho de 2025.

[22] N. Budincsevity, "Weather in szeged 2006-2016." https://www.kaggle.com/budincsevity/szeged-weather, 2016. Kaggle.

[23] salma maamouri, "weather prediction ,regression , neural model." https://www.kaggle.com/salmamaamouri/weather-prediction-regression-neural-model, 2018. Kaggle.