Introdução

Esta apostila tem por objetivo apresentar conceito introdutórios à ciência de dados e inteligência artificial, servindo como um guia para o aprendizado dos conceitos mais importantes para iniciar os estudos no assunto, de forma que o estudante, ao final da apostila, esteja apto a produzir, compreender e criticar modelos de análise de dados e de técnicas de inteligência artificial, como Machine Learning e Deep learning.

Para os códigos que serão praticados neste manual, será utilizada a linguagem Python 3.4+, e como IDE, o Jupyter Notebook. Para a instalação local de um ambiente de desenvolvimento é recomendado que se utilize o pacote Anaconda que fornecerá todas as plataformas e bibliotecas necessárias para o desenvolvimento dos estudos iniciais apresentados nesta apostila.

Contudo, é recomendado que se utilize a plataforma Kaggle como IDE hosteada em nuvem. Por ela, também é utilizado o Jupyter notebook como IDE, mas, a partir desta plataforma, se torna mais fácil o processo de importação de banco de dados e utilização das bibliotecas abordadas nesta apostila. Mais detalhes sobre o funcionamento do site e utilização dos Notebooks, chamados de “Kernels” no Kaggle, serão mostrados mais adiante no manual.

Unidade 1 - Exploração de Dados e Modelagem Inicial

1 - Introdução ao kaggle

O kaggle é uma plataforma desenvolvida para competições envolvendo ciência de dados, além disso existem diversos minicursos dedicados ao aprendizado básico deste assunto, integrando com técnicas de machine learning e deep learning.

Para acessar a plataforma será necessário o cadastro do usuário, iremos utilizar o e-mail institucional para a realização dos cursos e submissão dos códigos para as competições. Após o cadastro, o usuário é direcionado para a página inicial da plataforma, em que podem ser visualizados os seguintes ítens: Home, Compete, Data, Code, Communities e Courses.

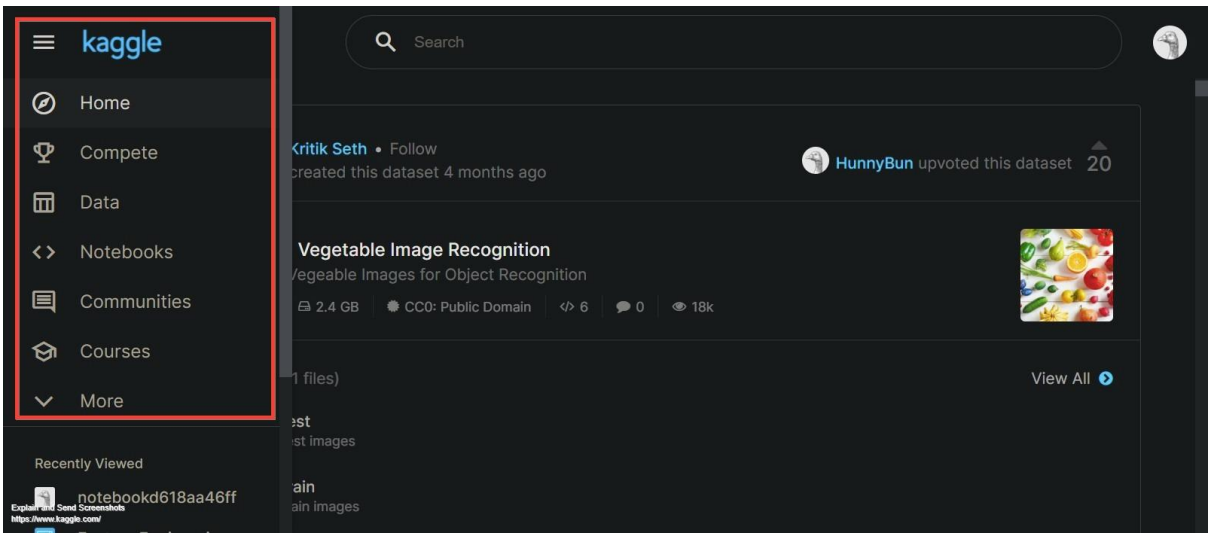

Figura 1: página inicial do kaggle

Na página “Home” são apresentadas as últimas notícias e discussões dentro do site, também podem ser visualizadas as últimas competições abertas ao público.

1.1 - Compete

O item “Compete” apresenta dois segmentos, o “Your Competitions” em que podemos localizar as competições que estamos participando, as que já completamos, as favoritadas, ou seja, as que foram marcadas para mais fácil localização mas o usuário ainda não está participando, e as competições que foram publicadas e são administradas por este usuário, na aba “Hosted”. O segmento “All competitions” apresenta todas as competições ativas na plataforma, em que não estamos participando, também apresenta as que já foram completas e as destinadas a aprendizado na aba “InClass”.

1.2 - Data

O terceiro item que temos acesso é o “Data”, como uma plataforma de ciência de dados o kaggle conta com inúmeros bancos de dados para fornecer aos seus colaboradores, estes datasets são utilizados como objetos de estudo para as competições. Os bancos de dados podem ser em formato CSV, JSON, BigQuery e outros tipos de dados semelhantes.

O mais utilizado é o formato CSV, é nele que são representadas as tabelas, com linhas indexadas, geralmente por números, e colunas que descrevem cada dado, que são nomeadas para facilitar a compreensão. Os outros tipos de dados funcionam da mesma forma, contudo, possuem uma maior complexidade, como o JSON, que apresenta arquivos com mais de uma camada de tabelas, ou BigQuery, que é uma ferramenta do google para facilitar a manipulação de bancos de dados com muitas informações.

Pode-se procurar datasets já disponíveis na plataforma por seu nome ou assunto que abordam, por meio de palavras chave. Também é possível realizar o upload de arquivos para o kaggle, os tornando públicos para que todos os usuários tenham acesso, e permitindo que se tenha a possibilidade de utilizá-los em competições.

1.3 - Code

Os Notebooks são ambientes de programação online, IDEs onde são utilizados servidores remotos para a interpretação e execução das linhas de comando construídas nas linguagens disponíveis da plataforma. No kaggle, as duas linguagens permitidas para seus notebooks são Python e R, é por meio deles que são submetidos os códigos para as competições disponíveis no kaggle e para exercícios de aprendizado nos minicursos ofertados.

Ao acessar este item será possível procurar por notebooks publicados na plataforma, por meio de seus nomes ou das palavras chave que indicam o assunto abordado. É interessante explorar notebooks dos assuntos que estão sendo estudados pelo usuário, para se ter uma base de quais caminhos seguir durante o desenvolvimento do código.

Neste item também pode-se criar notebooks, é neles que iremos construir as análises de dados. Ao criar um notebooks seremos apresentados à seguinte página.

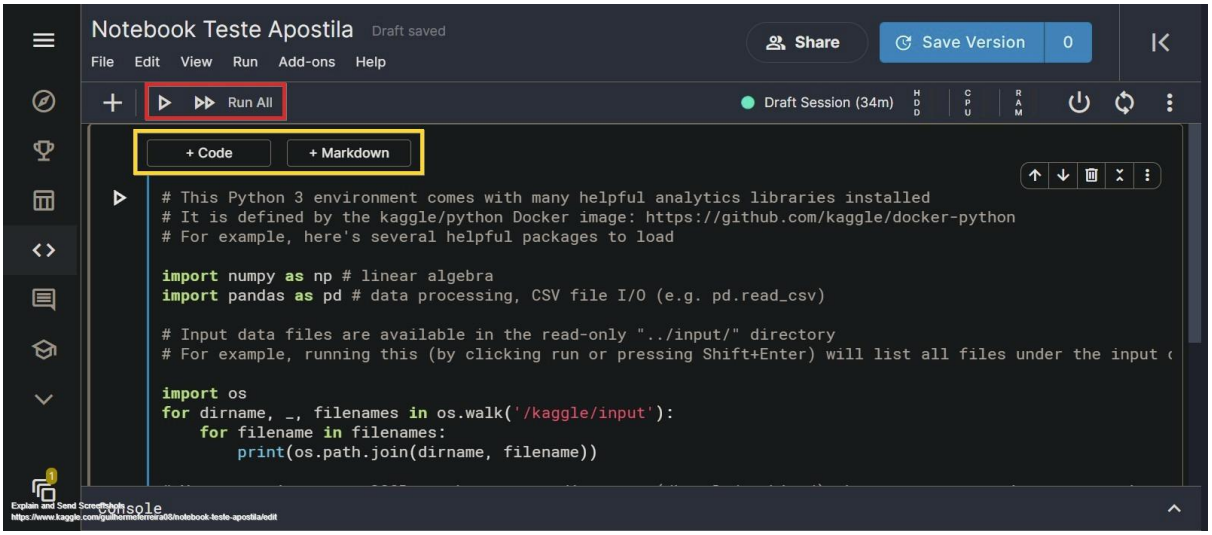

Figura 2: Tela de criação de notebook no Kaggle

Primeiramente deve-se adicionar um nome ao notebook, este ambiente funciona por meio de células que podem ser interpretadas individualmente, por meio do botão de play destacado em vermelho, ou todas de uma vez, por meio do botão “run all”. Também pode-se adicionar novas células com os botões “+Code”, que adiciona executáveis, ou “+Markdown”, que adiciona células de texto que podem ser utilizadas para deixar o código mais claro e organizado, ambos botões estão destacados em amarelo na figura 2.

Inicialmente já existirá uma célula criada para importar as principais bibliotecas utilizadas na ciência de dados, Numpy e pandas, além de setar o notebook para utilizar o sistema operacional do kaggle.

Na aba “file” é possível realizar upload de arquivos, internos do computador do usuário, por meio de link, ou pesquisar por um repositório existente no github, desde que estes arquivos sejam em linguagem Python ou R, desta maneira pode-se criar notebooks no kaggle a partir de códigos já construídos em outras IDEs.

1.4 - Communities

É a página onde os usuários do kaggle podem discutir sobre algum assunto específico, está separada por temas, como machine learning e data visualization, que discutem as ferramentas utilizadas na ciência de dados e questões que estão sendo resolvidas nas competições, como o COVID-19.

Alguns desses assuntos são específicos para iniciantes, é interessante visitá-los para ter acesso a dicas de pessoas experientes na área sobre como aprender o necessário para começar a ser um cientista de dados.

1.5 - Courses

O kaggle também disponibiliza minicursos para aprendizado básico de ciência de dados, são cursos em que existe uma explicação teórica inicial de cada assunto e também utilizam os notebooks para a realização de exercícios práticos para melhor aprendizado dos conceitos.

Inicialmente, para que se possa construir um modelo de análise de dados utilizando técnicas de inteligência artificial, será necessário a realização de 4 cursos: o de Python, que apresentará quais as ferramentas dessa linguagem serão necessárias para a construção de modelos; Intro to Machine Learning, que ensinará como criar, treinar e avaliar um modelo básico de inteligência artificial utilizando a técnica de machine learning; Pandas, que descreve como se deve utilizar a biblioteca “Pandas”, que é extremamente útil na manipulação de bancos de dados; Data Visualization, que fornece ferramentas para a criação e manipulação de diversos tipos de gráficos, muito úteis durante a análise de dados.

2 - Git e GitHub

1 dia para completar capítulo

O git é uma ferramenta que permite o fácil acesso a diferentes versões de um projeto. Através dele é possível revisar mudanças, editar partes de um código e retomar versões antigas com facilidade, além de permitir que diversos desenvolvedores interfiram na linha de produção do projeto, comparem suas mudanças e discutam, a fim de definir quais alterações devem ser incorporadas à linha principal do projeto.

Alguns conceitos devem ser bem claros para a utilização correta do git dentro de um projeto. Para o aprendizado dos comandos do git em um terminal, como Powershell ou Ubuntu, deve-se assistir o curso ministrado pelo professor José de Assis, pelo menos até a aula 10. Os conceitos teóricos apresentados no curso estão mais detalhadamente explicados no decorrer deste capítulo.

Também é necessário criar uma conta no GitHub e seguir o tutorial de como criar um repositório, editar arquivos, realizar o commit, criar um pull request e realizar o merge na plataforma. É aconselhável que este tutorial seja realizado antes da aula 6 do curso no youtube.

2.1 - Repositório

É onde são guardados todos os arquivos de um projeto. Esta unidade pode conter qualquer formato de arquivo, além de outras pastas para a melhor organização. Um projeto deve ter apenas um repositório para facilitar a organização, também é aconselhável que exista um arquivo inicial “README” que contenha as informações básicas sobre o projeto, como a sua descrição, objetivos e quais etapas já foram concluídas.

2.2 - Branch

Define o meio com o qual o git permite a fácil navegação entre versões de um projeto. A imagem a seguir mostra a base do funcionamento deste conceito.

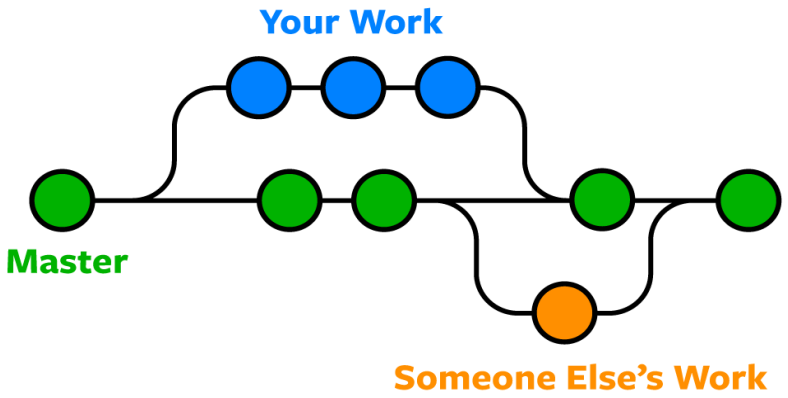

Figura 3: Esquema Branch

A figura 3 apresenta o funcionamento de um projeto utilizando git, cada ramo é uma branch, em cor verde tem-se o ramo “master”, que é a branch criada ao início do projeto, o qual irá conter as principais mudanças, nele devem ser adicionadas as versões apenas após revisão e discussão.

Qualquer outra branch criada será uma cópia da master branch (ou main branch). É importante notar que essa cópia será da última versão da master branch existente no momento da criação do novo ramo.

Cada ramo criado pode ser editado independentemente dos outros e da master branch, ou seja, cada desenvolvedor poderá editar sua versão, corrigir problemas e, só então, solicitar que suas alterações sejam incorporadas ao projeto, desta forma o controle de versões fica muito facilitado, assim como a detecção de erros.

2.3 - Commit

Ao realizar qualquer alteração no projeto, seja na master ou em qualquer outra branch, é necessário realizar o commit para que essa mudança seja registrada pelo git, ao realizar o commit deve-se acrescentar um comentário descrevendo a alteração que foi realizada.

É por meio deste comando que pode-se realizar o controle de versões, já que cada commit é identificado pelo git, sendo assim, caso ocorra um erro, em que seja necessário retornar a uma versão anterior do projeto, é necessário apenas identificar qual commit designa o ponto ao qual se deseja retornar.

2.4 - Pull Requests

Após realizados a edição e commit de uma branch diferente da main, esta versão terá algum acréscimo ou correção no projeto que está em progresso. Antes de incorporá-la à linha principal de produção, é necessário realizar um **Pull Request**, que nada mais é que a solicitação para que as edições feitas em uma branch paralela sejam incorporadas na main.

É aconselhável que se utilize o github para este processo, já que lá é possível comparar claramente as diferenças entre as versões, além de possibilitar a discussão entre os desenvolvedores. localmente também é possível realizar esta comparação dentro da IDE que está sendo utilizada

2.5 - Merge

Para terminar o processo de incorporação de outras versões na principal, é necessário realizar o **Merge.** Após a comparação e discussão das diferenças entre a versão em que foi realizado o Pull Request e a main, se essa nova versão foi aprovada, resta agora incorporá-la ao projeto, isso é feito pela ação de Merge.

Com isto feito, as correções e atualizações feitas na branch paralela farão parte da linha principal do projeto. Após o merge haverá a opção de deletar a branch que foi incorporada, isso pode ser uma boa prática para que o projeto não fique poluído, contudo há momentos em que é interessante manter outras versões, como quando está sendo realizado um trabalho em equipe, ou quando é desejável testar e validar mais de uma estratégia para a resolução de algum problema.

2.6 - Objetivos de aprendizado

Ao final da leitura dos capítulo, da décima aula do curso do professor José de Assis e da criação de uma conta e término do tutorial no github, é necessário que o aluno tenha domínio dos seguintes conceitos:

- Como criar um repositório localmente e no github
- O funcionamento das edições e realização de commits
- Navegar pelas versões do projeto por meio dos commits
- Criar e transitar entre branches, localmente e pelo github
- Comparar versões, fazer comentários e realizar o merge, localmente e no github

3 - Python

2 dias para completar capítulo

A linguagem que será utilizada nas competições que iremos participar é o python, será necessário ter o domínio de algumas de suas ferramentas, para que seja possível a

manipulação de dados e a elaboração dos modelos, de inteligência artificial, que serão submetidos às competições.

Para o aprendizado necessário é importante que o aluno aluno termine o curso de Python da plataforma Kaggle. A seguir serão mostrados quais os conceitos mais importantes que iremos precisar deste curso, e exemplos de como aplicá-los.

A seguir será apresentado um exercício para testar se os conhecimentos necessários foram compreendidos durante o minicurso, poderão ser utilizados qualquer notebook ou IDE que o aluno tenha preferência para a resolução dos exercícios. Esta atividade foi retirada do curso Data Analysis with Python: Zero to Pandas, da plataforma Jovian, e tem por objetivo a realização de uma análise simples de dados sem a utilização de bibliotecas, A solução será mostrada em um apêndice desta apostila.

3.1 - Exercício - Twitter Sentiment Analysis

A atividade consistirá em encontrar dados, dentro de uma lista, que contenham o valor desejado.

A lista “Tweets” apresenta diversos tweets que podem ser separados em felizes ou tristes, o dever do aluno é, através da identificação de palavras contidas nos tweets, classificar entre tweets felizes e tweets tristes.

A seguir é apresentada a lista 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. Quantos tweets existem nesse banco de dados?

2. A partir das listas de palavras apresentadas abaixo, tome o primeiro tweet como amostra e identifique se é feliz ou não.

   ```
   happy_words = ['great', 'excited', 'happy', 'nice', 'wonderful', 'amazing', 'good', 'best']
   sad_words = ['sad', 'bad', 'tragic', 'unhappy', 'worst']
   ```

   dica: 1. A partir de uma variável inicialmente falsa (“is_happy = False”, por exemplo), mude-a para verdadeiro caso o tweet seja feliz.

   2. Utilize um loop para analisar a frase completa, lembre-se que strings se comportam como listas no python, então utilizar o comando “in” é útil para encontrar uma determinada palavra em uma frase

3. A partir do item “b)”, crie uma nova variável que conte quantos tweets felizes estão contidos na lista “tweets”, crie também outra variável que conte o número de tweets tristes. Exiba os número encontrados com um texto que os defina

   Dica: Serão utilizados loops dentro de loops para a realização deste item.

4 - Pandas

2 dias para completar capítulo

A biblioteca pandas é a principal ferramenta que é utilizada para a manipulação de bancos de dados, com ela é possível retirar informações já existentes de forma inteligente, realizar operações matemáticas dentro do banco de dados, editar valores, adicionar novas informações, dentre outras diversas opções possíveis.

Normalmente esta ferramenta é utilizada para ler bancos de dados em CSV, mas também é útil para qualquer tipo de banco de dados que tenha formato de tabela. Para o aprendizado dos comandos básicos dessa biblioteca será necessário que seja realizado o curso de pandas da plataforma Kaggle.

Após o término do curso será necessário realizar uma atividade que irá colocar em prática os conhecimentos adquiridos sobre a biblioteca Pandas, neste exercício o aluno terá que utilizar as ferramentas da biblioteca para responder a algumas perguntas de um banco de dados disponível no Kaggle. É aconselhável que o aluno utilize um Kaggle kernel para facilitar a importação de dados e a utilização das bibliotecas.

4.1 - Explorando o “COVID-19 Dataset”

O banco de dados que será utilizado contém informações sobre a progressão do vírus COVID-19 em todo o mundo, sua última atualização ocorreu em setembro de 2020 e a partir dele serão feitas algumas perguntas que, com a utilização da biblioteca Pandas, poderão ser respondidas a partir da manipulação de dados.

É importante notar que este dataset tem diversos arquivos em CSV. Para esta atividade serão utilizados primeiramente o "covid_19_clean_complete", o qual apresenta as informações diárias sobre a evolução do vírus nos países estudados, e depois o “worldmeter_data”, que contém informações adicionais, como a população de cada país.

Uma boa prática é ler a descrição do dataset e navegar pelas features dos arquivos CSV que ele contém, para entender quais informações podem ser adquiridas dentro dos arquivos.

Os itens abaixo contem tarefas que devem ser cumprimas utilizando a biblioteca Pandas, abaixo da atividade em si, contém um passo a passo para a construção do item, este passo a passo é opcional para facilitar a construção do código, se o aluno preferir fazer da sua forma, é permitido desde que realize o que se pede no item.

1. O primeiro passo da exploração de dados é obter informações gerais, deve-se importar o arquivo “covid_19_clean_complete.csv” e obter as seguintes informações.

a. Quantidade de linhas e colunas
b. Quais são as colunas
c. O tipo de dado de cada coluna
d. A coluna de datas deve ser transformada de “Object” para “datetime64”, utilizando o comando “pd.todatetime( )”
e. Informações estatísticas sobre o banco de dados
f. Se necessário, transformar uma coluna pertinente para formato de data
g. Quais colunas apresentam NaN

2. Crie um dataframe que contenha apenas as 5 províncias da China que mais registraram casos.

    a. Descubra qual o nome das províncias chinesas contidas no dataset
    b. Retire do banco de dados apenas as informações dessas províncias
    c. Tome apenas as informações das features: “Confirmed”, “Active”, “Deaths”, “Recovered”
    d. Agrupe o novo dataframe a partir dessas províncias. Qual a função de agrupamento que deve ser usada?
    e. Produza um dataframe contendo as 5 regiões com o maior número de casos confirmados

3. A coluna “Province/State” contém muitos valores faltando. Para descartá-la perdendo o mínimo de informação, inclua na coluna “Country/Region” o nome das regiões junto ao nome dos países em que existem mais de uma província registrada no banco de dados, utilizando o método “.apply”. Como por exemplo, juntar o nome China com a região pertinente: “China_Hubei”, “China_Guangdong”.

    a. Elabore uma função que tenha uma linha do banco de dados como argumento e, se a coluna “Province/State” não for valor faltante, concatenar seu valor com coluna “Country/Region”
        i. Utilize o método “pandas.notna” para averiguar de “Province/State” é valor faltante

    b. Faça uma cópia do banco de dados para poder retirar informações sem perder o banco de dados original
    c. Aplique, por meio do método “apply”, a função criada no item a
    d. Exclua a coluna “Province/State” do novo dataframe

4. Importe o arquivo "world meter data.csv" e, a partir de seus dados de população e continentes, faça um ranking de maior número de mortes por milhão de habitantes entre os continentes.

    a. Importar o banco de dados "world meter data.csv"
    b. Tome apenas as informações de população, país e continentes deste banco de dados
    c. Agrupe o “covid_19_clean_complete” por países. Deve-se usar o dataframe do item “c)” ou o original?

d. Mescle os dois dataframes, a partir dos países, contendo apenas as informações de mortes, população e continente
e. Agrupe o novo dataset agora pelo continente
f. Crie uma nova coluna contendo o número de mortes por milhão dentro dos continentes (Utilize a fórmula: (mortes/população)*10^6)
g. Faça um ranking do número de mortes por milhão de habitantes entre os continentes

5 - Visualização de dados

2 dias para completar capítulo

Uma das ferramentas fundamentais de um cientista de dados são os gráficos, a partir deles é possível analisar tendências e relações entre variáveis que podem ser úteis ao interpretar um banco de dados, além de ser fundamental para a interpretação dos padrões que um modelo de inteligência artificial detectou em seu treinamento.

As principais bibliotecas utilizadas na geração de gráficos para a interpretação de dados são a Seaborne e MatPlotLib. A partir delas é possível criar diversos gráficos que são úteis para diferentes tipos de análise, para compreender os comandos básicos dessas bibliotecas o aluno deverá realizar o Curso de Data Visualization do kaggle, que demonstrará como construir diversos tipos de gráficos a partir de um banco de dados.

Os códigos dos exercícios deste capítulo devem ser realizados em um notebook do kaggle, para facilitar a utilização dos bancos de dados que serão estudados. Estes Notebooks devem conter explicações para o programa construído, seja por comentários nas células de código ou, preferencialmente, utilizando as células de markdown.

Continuando com o mesmo COVID - 19 Dataset, agora produziremos gráficos a partir das manipulações já feitas no exercício 4.1 além de algumas outras que serão necessárias para o exercício.

5.1 - Line Chart

Um dos tipos de gráficos mais simples é o line chart, com ele é possível observar relações entre duas variáveis, geralmente contínuas, a partir dos pontos que as relacionam ligados por uma linha contínua. Uma das melhores aplicações para este tipo de gráfico são as progressões temporais, já que, a partir delas é possível analisar como certa variável se comporta no tempo, assim como suas tendências.

Neste exercício serão comparadas as progressões no tempo dos números de mortes das regiões "Eastern Mediterran", "Americas", "Europe". Após a geração destas distribuições deve-se responder às seguintes perguntas:

1. Quais das regiões tiveram mais mortes?
2. Qual o momento em que cada progressão começa a subir?
3. Como está a taxa de crescimento de cada progressão?

1. Criar gráfico linechart que indique a progressão no tempo das mortes nas regiões: Américas, Eastern Mediterran e Europe (OBS: Utilize apenas o arquivo “covid_19_clean_complete.csv”)

    a. Importar bibliotecas matplotlib e seaborn, para retirar a necessidade de criar arquivos para os gráficos, digitar comando “%matplotlib inline”
    b. Realizar agrupamento utilizando as colunas “Date” e “WHO Region”, utilizando “sum” como função de agregação
    c. Plotar gráficos que contenham as progressões das regiões exigidas
    d. adicionar nomes para eixos, título do gráfico, grid e legendas para as linhas
    e. Analisar gráfico de acordo com as perguntas feitas

5.2 - Bar Charts

Este tipo de gráfico é geralmente utilizado para comparar quantidades entre diferentes grupos. Um dos agrupamentos importantes do data frame são os continentes, a partir deles pode-se comparar a quantidade de mortes para entender melhor como a pandemia afeta cada parte do mundo e assim formular hipóteses.

Neste exercício será produzido um gráfico bar chart que compare a quantidade de mortes nos continentes do mundo, após isso deve-se responder às seguintes perguntas:

1. Quais os 2 continentes com maior número de mortes?
2. Qual o continente com menor número de mortes?
3. Qual sua hipótese para responder às perguntas anteriores?

1. Agora deve-se produzir um gráfico bar chart, indicando o número de mortes em cada continente (OBS: utilize o dataframe criado no exercício 4.1 - 4)

    a. Editar o tamanho do gráfico, como sugestão, coloque o tamanho (10,6)
    b. Plotar gráfico bar chart das mortes nos continentes
    c. editar título e nome dos eixos
    d. Responder perguntas com base no gráfico

5.3 - Scatter Plots

É geralmente utilizado para a análise de relações entre diferentes colunas, a partir dos plots feitos de cada interação, é possível notar padrões antes ocultos no banco de dados. No próximo exercício, serão escolhidos os continentes com maior número de mortes por milhão e relacionar esta coluna com o número da população, para descobrir se há relação entre estas features do banco de dados.

Existe a chance de elas não se relacionarem o suficiente para que se possa fazer uma análise adequada. Após a construção do gráfico devem ser respondidas as seguintes perguntas:

1. Onde se concentram a maioria dos dados?
2. No segundo continentes por mais mortes por milhão, é possível perceber alguma relação entre as features
3. Pelo gráfico, é possível enxergar relação entre essas colunas do banco de dados?

1. Descobrir se há relação entre o número de mortes por milhão e a quantidade de habitantes, nos 3 continentes com maior média de mortes por milhão de habitantes, por meio de scatter plot (OBS: Utilizar somente arquivo “worldometer_data.csv”)

    a. Agrupar o dataframe pela coluna “Continent”, retirar apenas os dados de "Deaths/1 M pop", utilizar “mean” como função de agregação e ordenar para descobrir os 3 continentes com maior valor
    b. Obter apenas os dados do dataframe, correspondentes a esses continentes
    c. Criar Scatterplot sendo o eixo x mostrando as mortes por milhão, o eixo y, a população total e representar os 3 continentes com cores diferentes
    d. Responder as perguntas com base no gráfico

5.4 - Heatmap de Correlação

Um dos principais conceitos na ciência de dados é a correlação. Este valor mede o quanto uma variável está relacionada, direta ou inversamente, com outra. A partir dessa métrica, podem ser feitas diversas hipóteses sobre o banco de dados que serão futuramente exploradas, especialmente em bancos de dados com muitas features, os valores vão de -1 até 1, em que 1 indica que duas variáveis são diretamente proporcionais, -1 que são inversamente proporcionais e 0 que são independentes, para mais informações acesse este artigo do site towards data science.

Para a visualização simultânea de das correlações entre todas as features de uma banco de dados, o heatmap é frequentemente utilizado, portanto, o objetivo do próximo exercício é produzir e analisar um gráfico que mostre todos os valores de correlação entre as colunas do banco de dados “worldometer_data.csv”

1. Será utilizado o método DataFrame.corr() do pandas para gerar os valores de correlação, então o aluno deverá produzir um heatmap para a visualização dos valores gerados

    a. Criar uma variável para armazenar os valores de correlação
    b. Criar um heatmap, com título, utilizando a variável criada no item anterior
    c. Escolher duas features para analisar seus valores de correlação (Sugestão: “Population”, “Teste/1M pop”)

6 - Introdução ao Machine Learning

2 dias para completar capítulo

Neste capítulo será apresentada uma introdução de como produzir modelos, a partir de inteligência artificial utilizando o Machine Learning. Esta estratégia baseia-se em, a partir de um banco de dados, extrair informações e identificar padrões que possibilitem a criação de um modelo que, a partir de uma nova entrada, preveja a saída que ocorreria na realidade, com o menor erro possível.

Inicialmente iremos trabalhar com bancos de dados tabulares, ou seja, que são identificados pelo índice da linha e pelo nome da coluna, como os que foram analisados nos capítulos 4 e 5. A partir destes dataframes, podemos separar as colunas em 2 grupos, as Features, que serão usadas para treinar o modelo, e o Target, que será o valor que pretende-se prever a partir do treinamento.

Para o aprendizado de como utilizar as bibliotecas necessárias à construção de modelos de ML, será necessário fazer o curso Intro to Machine Learning da plataforma Kaggle. A partir dos conhecimentos adquiridos, o aluno será capaz de produzir o primeiro modelo de ML e submetê-lo a uma competição no Kaggle, esta competição será a Titanic - Machine Learning from Disaster, que contém um banco de dados simples e ideal para iniciantes em ciência de dados e inteligência artificial. Lembre-se de se inscrever nela antes de começar o exercício.

No curso do Kaggle os exemplos foram feitos com algoritmos de regressão, que utilizam as decision trees para estimar uma resposta aproximada a partir das features com as quais treinou, esta abordagem é utilizada quando se pretende prever um target com valores contínuos. A métrica utilizada no curso para validação de um regressor é a “MAE”, que toma a média dos erros de todas as predições.

Na competição Titanic, o objetivo será prever se um passageiro sobreviveu ou não a partir das suas características, ou seja, há apenas dois valores possíveis, sim ou não, neste caso, o modelo deverá indicar qual é a resposta certa, e não aproximada. Para este tipo de problema, são utilizados os classifiers, que funcionam de forma semelhante aos regressors, mas suas respostas são dadas em valores discretos. A métrica para avaliar este tipo de modelo é “accuracy”, que é a porcentagem das predições que foram acertadas pelo modelo, já que “MAE” não seria efetivo (Por que?). Para mais informações sobre diferenças entre classificação e regressão em problemas de predição, acesse este artigo.

No exercício a seguir será construído o primeiro modelo de IA, além de sua submissão a uma competição do Kaggle. É importante esclarecer que não será necessário um modelo bem otimizado e com uma predição muito próxima da realidade, já que o objetivo aqui é aprender como criar um modelo por conta própria, gerar um arquivo que contenha as predições geradas e submetê-las a competição.

Este exercício foi baseado no Kernel publicado por Nadim Tamer. Os alunos são incentivados a procurar outros notebooks para melhorar o que irão produzir neste Capítulo.

6.1 - Importação e visualização do banco de dados

Os primeiros passos são: importar as bibliotecas que serão utilizadas, os bancos de dados de treino e teste, e explorar as colunas dos bancos de dados que serão utilizados. Para o primeiro modelo serão manipulados apenas as features numéricas, qualquer outra que esteja na forma de string será retirada. Em relação aos valores nulos (NaN), as linhas que os contém também serão retiradas do dataset.

1. Importar as bibliotecas que irão ser utilizadas:

    a. Numpy as np
    b. Pandas as pd
    c. Matplotlib.pyplot
    d. seaborne as sns
    e. matplotlib inline, para gerar gráficos no próprio notebook
    f. Configurar o random seed para 0 (np.random.seed(0))

2. Transformar arquivos em dataframe e gerar momentos estatísticos

    a. Importar os arquivos “kaggle/input/titanic/test.csv” e “kaggle/input/titanic/train.csv”
    b. Gerar momentos estatísticos sobre o dataframe de treino, utilizando “.describe( )”

3. Análise dos dados

    a. Identificar as Features do banco de dados que são categóricas (não numéricas), e as que são numéricas. As numéricas, deve-se separá-las em discretas e contínuas.
        i. Pode-se usar o método “.info( )” para descobrir o tipo de dado de cada Feature.

    b. Descobrir quais as Features contém NaN, além da quantidade desse tipo de dado em cada Feature.
        i. Pode-se utilizar o comando “DataFrame.isnull.sum( )”

    c. Deve-se utilizar apenas Features numéricas e retirar as categóricas. Portanto, faça dois novos dataframes, apenas com as features numéricas do banco de dados de treino e de teste.

    d. Retire as linhas que contenham NaN das features do banco de dados de treino.
        i. Para isso será útil o método DataFrame.dropna( )

    e. no banco de dados de teste, substitua os NaN pelo valor 0, para que o arquivo de predição tenha o número de linhas exigido pela competição
        i. Utilizar o método DataFrame.fillna( )

6.2 - Exploração do banco de dados

A próxima etapa é a visualização de dados. Para cada Feature serão gerados gráficos que indiquem a probabilidade de sobrevivência do passageiro de acordo com o valor de sua característica. Será utilizado o bar plot do seaborn, já que possibilita este tipo de visualização.

1. Para a Feature “Age” será necessário criar uma nova Feature que a separe em 8 grupos para que se possa utilizar “sns.barplot”. (DICA: Crie um novo dataframe apenas com as informações necessárias para a criação deste gráfico)
    a. Criar uma lista com os nomes que serão dados aos grupos em que as idades serão repartidas (Exemplo: “(0-10)”, “(10-20)”, ...)
    b. utilize pandas.cut, para separar as idades em 8 grupos e utilize o argumento “labels” para utilizar a lista dos nomes dos grupos
    c. Criar a nova coluna com os grupos, perceba que esta nova coluna será categórica
    d. Criar um gráfico que indique a chance de sobrevivência por grupo de idade
    e. Faça uma análise a partir da visualização deste gráfico

2. Crie gráficos para todas as outras features numéricas discretas, e faça uma análise dos resultados obtidos. Não será necessário produzir um gráfico para a feature “Fare”. Após a criação de cada gráfico será necessário realizar uma breve análise para avaliação do impacto de cada feature na predição

6.3 - Treinamento do modelo e submissão das predições

O próximo passo é preparar os dados para treinar o modelo. Deve-se definir quais colunas do banco de dados de treino são Features e qual é o target, após isso é necessário separar o banco de dados de treino em treino e validação. (OBS: Lembre-se de tirar a coluna que contém o ID dos passageiros antes de treinar o modelo).

Deve-se criar e treinar um modelo a partir das manipulações feitas nos dados. Será necessário utilizar um Classifier para a geração do modelo, como já dito anteriormente no capítulo. O “MAE” também não é indicado neste caso, para medir a qualidade de um classifier utilizasse o accuracy score.

1. Separação de banco de dados de treino e validação.

    a. Criar variável X, contendo o banco de dados apenas das Features
    b. Criar variável y, contendo o target da predição
    c. Usar Train Test Split para separar treino e validação (DICA: utilizar validação com 20% do tamanho do treino)

2. Treinamento e validação do modelo

a. Importar Random Forest Classifier da biblioteca sklearn.ensemble
b. Importar accuracy score da biblioteca sklearn.metrics
c. Treinar random forest com as features e o target de treino
d. Realizar a predição
e. Validar a predição, comparando com o target da validação, utilizando accuracy score
f. Avalie se a pontuação é satisfatória

3. Gerar predição, salvar a versão e submeter à competição

a. Treinar novamente o modelo, utilizando o banco de dados completo
b. Salvar o índice do banco de dados de teste
c. Gerar uma predição a partir do modelo, sendo a entrar o banco de dados de teste, sem seu índice
d. Criar um dataframe com o pandas que contenha: os mesmos índices do banco de dados de teste, as predições realizadas (OBS: Olhar arquivo modelo "gender_submission.csv")
e. Transformar este dataframe em um arquivo csv utilizando pandas
f. Salvar a versão clicando em “Save Version”, selecionar a versão que foi salva e submeter à competição

Unidade 2 - Machine Learning Clássico

7 - Data Cleaning

2 dias para completar capítulo

Quando se começa a análise de um banco de dados, normalmente não será possível fazer uma análise detalhada logo de início. Isso se deve a vários fatores, como a inclusão de valores nulos (NaN) em algumas células, o formato dos dados, quando existe uma coluna de data e não está identificada como tal para o sistema, ou em colunas de texto, em que geralmente pode-se tirar informações importantes se esses valores forem manipulados.

Data cleaning, é a etapa de organizar o banco de dados para que possamos fazer análises, além de tornar possível que os algoritmos de inteligência artificial produzam modelos. Para entender algumas estratégias de limpeza de dados, além dos APIs utilizados nestes processos, é importante que o aluno realize o minicurso de Data Cleaning do Kaggle.

Em alguns dos exercícios anteriores desta apostila, foram realizadas algumas limpezas de dados, como a utilização de datas no exercício 5.1- D, e a análise do dataframe realizada em 6.1 - C e D. Após a realização do minicurso, é recomendado que o aluno revise estas etapas para um melhor entendimento do processo.

Neste capítulo será utilizada a mesma competição do exercício 6.1, mas agora o aluno deverá criar uma nova versão para a aplicação dos novos conhecimentos que serão adquiridos durante esta unidade, possibilitando o aprofundamento da análise, otimização do código e melhoria de performance do modelo. Será útil utilizar algumas partes da primeira versão, então é vantajoso editá-la ao invés de criar um novo notebook do zero.

A partir do próximo exercício ocorrerá um aprofundamento na análise do banco de dados da competição Titanic, o primeiro passo será a limpeza e criação de algumas novas features que facilitem a interpretação e percepção de como o modelo irá se comportar. A construção deste exercício foi feita com base no Notebook de LD Freeman.

Algo que foi deixado de lado durante o exercício do capítulo 6, foi a manipulação do banco de dados de teste. A partir de agora ele também será analisado e manipulado de forma que o modelo possa ser construído com melhor precisão, desta forma é necessário que as análises já realizadas durante a primeira versão sejam refeitas.

1. Análise inicial:

    a. Quais as colunas do banco de dados de teste e de treino
    b. Qual o tipo de dado de cada coluna nos dataframes de teste e de treino
    c. Qual a quantidade de valores nulos (NaN) em cada feature
    d. Realizar um cópia do banco de dados de teste e de treino para que se possa fazer a manipulação sem perder informações

2. Para lidar com valores nulos, podemos preencher estes valores de alguma forma ou descartar a informação. Neste item utilizaremos algumas estratégias para tal.

    a. A feature “Cabin” contém muitos valores nulos e, assim como “Ticket”, não fornece informações relevantes ao modelo à primeira vista. Então pode-se retirá-las.
    b. A coluna “Age” também contém muitos valores nulos, mas dessa vez iremos preenchê-los. Utilizar a mediana dos valores de idade para isso, é uma boa estratégia inicial.
    c. A feature “Fare” contém poucos valores nulos, e pode ser útil para análises futuras. Pode-se preenchê-la com a mesma estratégia que “Age”, já que ambas são contínuas.
    d. Em “Embarked” também tem-se pouca ocorrência de valores nulos, e como é uma feature categórica, preencher com a moda parece menos impactante na construção do modelo.

3. Para as Features contínuas será útil a criação de grupos para facilitar a análise. Dois métodos do pandas são úteis para esta tarefa, pd.cut e pd.qcut, também é útil visitar esta referência para uma melhor entendimento destes métodos.

    a. Criar Feature que separe a Feature “Age” em 5 intervalos de mesma extensão.
    b. Criar Feature que separe “Fare” em 6 intervalos que contenham o mesmo número de dados (Não precisam ter a mesma extensão).

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, 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 predição 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, ou seja, transformar seus valores em números que as representem. Existem algumas estratégias para realizar o encoding, neste capítulo utilizaremos 3, one-hot encoding, ordinal encoding e label encoding.

1. Será necessário realizar o encoding das variáveis categóricas, no momento, 3 estratégias que serão utilizadas são o one-hot, label encoding e ordinal encoding. Para fazer isso será necessário relembrar quais são as features categóricas, definir a estratégia que será utilizada, criar as features codificadas e retirar as categóricas:

    a. Utilizar o método ".info( )" para relembrar quais são as features categóricas.

    b. Utilizar 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. Temos algumas features categóricas que contém uma ordem clara, ou seja, existe o primeiro valor, segundo valor, terceiro valor, assim por diante, já em outras, isso não acontece.

i. Para as features que possuem ordem, o label encoder é mais indicado como primeira abordagem.
ii. Para features sem ordem definida, one-hot encoding pode ser a melhor opção.

d. Criar uma função que realize o one-hot encode e, como saída, retorna um novo dataframe com as colunas que resultam da codificação, devidamente nomeadas, ao invés das features categóricas:
   i. Os argumentos deverão ser: o dataframe que será manipulado e as colunas que serão codificadas.
   ii. Realizar uma cópia do banco de dados que foi dado como argumento.
   iii. Criar um loop que realize o encoding das colunas presentes no argumento da função, as features resultantes devem conter o nome do valor que representam (OBS: utilizar o método ".get_feature_names").
   iv. Ainda dentro do loop, retirar do dataframe copiado as features categóricas e mesclar as features criadas pelo one-hot encoder.
e. Criar novas features para realizar o label encoding das features que contém um ordenamento claro, e depois retirar as features categóricas que foram codificadas.

   OBS: é importante lembrar que esse encoding deve ser realizado nos dataframes de treino e teste

2. Agora será realizado um novo treinamento e avaliação de modelo, os passos serão os mesmo realizados no item 6.1 g), mas agora com o banco de dados que é resultado do encoding das variáveis categóricas. Após essa avaliação, será possível notar a evolução da precisão do modelo produzido utilizando essas novas técnicas.

8.2 - Pipelines

Esta estratégia serve para tornar o código mais otimizado e direto, com a utilização das pipelines é possível realizar várias operações, de limpeza e modelagem de dados, com poucos comandos, sendo que as bibliotecas construídas para pipelines já são otimizadas da melhor forma possível, tornando o código menos custoso em termos de processamento, além de mais rápido.

No capítulo a seguir serão utilizadas as pipelines para realizar parte dos processo que já foi feito anteriormente no banco de dados do Titanic, além do teste de diferentes estratégias de encoding e limpeza de dados, para que sejam avaliadas e comparadas, a fim de encontrarmos as melhores

1. Deverá ser criada uma função que produz e avalia pipelines que utilizam diversas estratégias de imputing de valores nulos e encoding de variáveis categóricas.

   a. Colocar de volta os valores nulos das colunas "Age" e "Embarked" do dataframe de treino, utilizando as colunas do banco de dados original.

Lembrando que, ao iniciar este notebook, foi feita uma cópia deste banco de dados.

b. Criar uma função que produza e avalie pipelines que utilizam estratégias indicadas nos argumentos:
    i. Argumentos:
        - **data:** Banco de dados a partir do qual será produzido o modelo.
        - **encoder:** a estratégia de encoding de variáveis categóricas, como one-hot ou label encoder.
        - **model:** o algoritmo que irá produzir o modelo, inicialmente será o random forest classifier.
        - **numerical_imputer:** a estratégia usada para substituir valores nulos nas features numéricas, como "mean" ou "median".
        - **categorical_imputer:** estratégia para preencher valores nulos nas features categóricas, aqui será usada apenas "most frequent", que é substituir pelo valor que mais aparece no banco de dados.

    ii. Definir as features e o target que serão usados para treinar o modelo e separar o banco de dados utilizando o train-test split.

    iii. Criar um preprocessor que: para as features numéricas, utilize o imputer indicado por “numerical_imputer”; para as categóricas, utilize o imputer fornecido em “categorical_imputer” e como encoder utilize o argumento “encoder”.

    iv. produzir pipeline utilizando o preprocessor previamente construído e o modelo do argumento “model”.

    v. Realizar o treinamento da pipeline, gerar predições e avaliar utilizando accuracy score.

    vi. A função deve retornar o resultado da avaliação em porcentagem.

3. Aqui serão feitas duas listas, uma com algumas estratégias de imputing de valores nulos e outra com os encoders que foram utilizados até agora. O label encoder deve ser substituído pelo ordinal encoder, ambos têm o mesmo resultado, contudo o ordinal é feito para decodificar features, e o label, para o target. Acesse estas referências para entender sobre a utilização do Simple Imputer e Ordinal encoder

    a. Definir as 4 seguintes estratégias para imputing:
        i. Mean: substitui a o valor nulo pela média
        ii. Most_frequent: substitui pelo valor mais frequente na feature
        iii. Median: substitui pela mediana
        iv. Zero: substitui todos os valores nulos por 0

    b. Definir as 2 estratégias para encoding:

i. One-Hot encoder
ii. Ordinal encoder

4. A partir da função de criação de pipelines e das listas de encoders e imputers, criar um loop que produza todas as combinações possíveis dessas estratégias, e mostre o resultado da avaliação de precisão de cada pipeline produzida no loop. (OBS: terá que ser usado um “for” dentro de outro).

5. Avalie se alguma dessas estratégias usadas separadamente, gerou uma precisão do modelo maior que suas utilizações mescladas, que foi a estratégia realizada para produzir o modelo antes das pipelines.

8.3 - Cross Validation e Gradient Boosting

Após a criação de um modelo, uma parte fundamental do processo de ciência de dados é a avaliação do modelo, a partir dela pode-se averiguar a generalidade e precisão do modelo que está sendo construído, quanto melhor é a estratégia de avaliação, mais certeza teremos sobre a qualidade do modelo. Um método robusto, que pode ser usado para indicar como o modelo se sairá com dados reais é o cross validation, que será utilizado neste capítulo.

Um Algoritmo que é utilizado para aumentar a generalidade e precisão do modelo é o gradient boosting. Assim como o random forest, este algoritmo utiliza diversas decision trees para gerar predições e então as avalia, contudo, neste algoritmo, a cada modelo realizado, seu erro é calculado, então os modelos seguintes são feitos para que esse erro seja diminuído, dessa forma o algoritmo se torna mais robusto.

Neste capítulo iremos avaliar o modelo utilizando o cross validation, entender se é suficientemente preciso e robusto, para que então possa-se enxergar o impacto positivo que o algoritmo de gradient boosting pode causar.

1. O cross validation é uma maneira melhor para avaliar os modelos com os quais estamos lidando. Portanto, o modelo que melhor se saiu nos teste até agora, deverá ser avaliado utilizando este método. Como resultado será utilizada a média entre as avaliações, lembre-se de utilizar o scoring = “accuracy”.

2. Assim como o Random Forest, o XGBoost também tem seu equivalente como classifier. O Gradient Boosting Classifier será utilizado agora para gerar um novo modelo, a partir do dataframe que foi manipulado com as melhores estratégias até agora.

    a. Importar o Gradient Boosting Classifier, utilizando random state = 0 e n_iter_no_change = 100 (Entender o que são esses parâmetros a partir da documentação do algoritmo).

b. Determinar as features e o target que serão utilizados para gerar o modelo.
c. Treinar e avaliar o modelo utilizando cross validation.
d. A partir da nota gerada pela avaliação, determinar qual algoritmo se saiu melhor, Random Forest ou Gradient Boosting

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 que serão utilizados, e devem ser compreendidos, 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, e, a partir delas, o modelo poderá ser ajustado para aumentar sua precisão. Já os algoritmos de unsupervised learning, não utilizam dados rotulados, apenas encontram relações entre as features, sem a comparação com respostas previamente fornecidas (sem o target), para mais informações sobre o assunto acesse o artigo da IBM.

No contexto de Feature Engineering, os algoritmos de unsupervised learning serão usados para encontrar novas relações anteriormente ocultas entre as features, sem a utilização do target, contudo, o modelo que está sendo construído é considerado supervised learning, já que as predições devem ser ajustadas de acordo com a coluna “Survived”, já previamente preenchida no banco de dados de treino.

Feature Engineering é composto de 3 áreas, feature extraction, feature construction e feature selection. 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 e feature selection diminui a quantidade de features detectando redundâncias e selecionando as feature mais importantes

9.1 - Feature Selection

Os objetivos dessa área são, selecionar as features do banco de dados que sejam menos relacionadas possível entre si, para evitar a redundância, e o mais correlacionadas possível com o target, para que, a partir da variação dessas features, o modelo consiga identificar a variação do target, a fim de predizer-lo.

Neste capítulo utilizaremos uma ferramenta muito útil para a averiguação da correlação entre as features e o target. O Mutual Information avalia qual a interferência de cada feature para o target, ou seja, o quanto a variação de uma feature específica pode explicar a variação da coluna que é o alvo da produção. Após a avaliação, as features que tiveram as melhores pontuações serão analisadas mais detalhadamente

1. O primeiro passo é a avaliação da importância de cada feature que já está no banco de dados de teste, para isso será criada uma função que produza um gráfico indicado o Mutual Information score de cada feature existente no banco de dados d treino, em relação ao target

    a. Determinar quais são as features discretas (todas menos “Age” e “Fare”)
    b. Criar a função que produza o gráfico de MI scores
        i. Argumentos:
            1. Features
            2. Target
            3. Features discretas
        ii. Gerar MI scores e produzir uma série do pandas com estes valores ordenados de forma descendente.
        iii. Criar um gráfico de barras que indique a pontuação de cada features.
        iv. A função deve retornar a série que contém os MI scores.
    c. Determinar quais são as features que serão utilizadas na função e o target.
    d. Produzir gráfico a partir da função e dos argumentos necessários.

2. Após a observação do gráfico gerado é possível concluir que as features que as features com maior influência no target são “Fare”, “Sex” e “Age”, por este motivo, elas serão analisadas mais a fundo por meio da utilização de gráficos. Para facilitar a visualização dos dados, é útil que seja usado um banco de dados que ainda contenha as features categóricas originais

    a. Construir dois gráficos do tipo sns.histplot para a análise de “Fare”.

        i. A distribuição total das taxas.
        ii. A distribuição relacionada com o target
        iii. Realize uma análise em relação à distribuição total, se a curva é normal ou não, por exemplo. Qual a influência da feature “Fare” no target que pode ser percebida a partir do segundo gráfico?

    b. Agora serão criados 4 gráficos também utilizando histogramas com seaborn. Tais gráficos relacionam a Feature “Age” com “Sex”. Para facilitar a visualização é indicada a utilização de plt.subplots, para a visualização de mais de um gráfico por output.

        i. Distribuição da idade dos passageiros utilizando o sexo como camada.
        ii. A distribuição da idade dos passageiros, apenas entre os sobreviventes, utilizando o sexo como camada.
        iii. Distribuição da idade dos homens, utilizando “Survived” como camada.
        iv. Distribuição da idade das mulheres, utilizando “Survived” como
        v. Faça uma análise comparando a distribuição dos sobreviventes, comparando homens e mulheres e qual e como a idade influencia na chance de sobrevivência

9.2 - Feature Construction

Nessa área, o cientista de dados procura extrair informações relevantes que não estão explícitas nos dados, essas informações surgem a partir de operações entre as colunas, como por exemplo, a soma ou subtração de features numéricas, ou conjunção de features booleanas, dessa forma são criadas novas colunas que contém informações melhor relacionadas com o target.

Este passo do feature engineering vai para o lado oposto em relação ao feature selection, já que aumenta o número de features. Em uma interpretação espacial, podemos dizer que cada feature do banco de dados é uma dimensão, feature construction aumenta este número de eixos do espaço do banco de dados, enquanto feature selection diminui.

Neste capítulo iremos manipular algumas colunas para conseguir informações que podem ser relevantes para identificar a chance de cada passageiro sobreviver ao naufrágio.

1. Na feature “Name” existe, para cada passageiro, um título, como “Mr” ou “Miss”. Esta informação pode ser útil, então podemos criar uma nova coluna para obter estes dados.

    a. A partir de “Name” criar a feature “Title” para separar os valores, primeiramente, pela vírgula, e depois pelo ponto. Para isso deve-se utilizar o método str.split, lembre-se de criar uma nova coluna para armazenar estas informações, após isso, pode-se excluir a feature “Names”

2. Criar novas Features para resumir informações de outras já existentes, uma para indicar o número total de familiares de cada passageiro, sejam eles primos, filhos ou acompanhantes, e outra para informar se o passageiro está sozinho ou não.

    a. Criar feature que indique o tamanho da família, somando “SibSp” e “Parch”.
    b. Criar feature booleana indicando se o passageiro está sozinho, sendo, verdadeiro caso não tenha nenhum familiar a bordo, e falso caso tenha.

9.3 - Feature Extraction

Assim como feature selection, feature extraction tem o objetivo de diminuir as dimensões (Features) do banco de dados, mas aqui há uma importante diferença, nesta área ocorrerá a manipulação das features originais, ou seja, elas serão modificadas a partir da utilização de operações de mapeamento. São criadas novas colunas que podem resumir as informações das colunas originais, ou seja, ter um nível similar de informações sobre o target, com um número menor de features.

Dois algoritmos não supervisionados serão utilizados para mapeamento, ambos serão aplicados nas features “Fare” e “Age”, com o objetivo de reduzi-las a apenas uma coluna que resuma suas informações. Estes métodos são o Clustering e o Principal Component Analysis (PCA). No Clustering são criadas categorias que agrupam uma determinada distribuição, no caso, será a distribuição dos valores de “Fare” em relação a “Age” (“Fare” no eixo y e “Age” no eixo x). O PCA, por sua vez tem o objetivo de criar um novo eixo (Feature) em que, ao se

projetar os pontos dos outros dois, “Fare” e “Age”, pode-se obter as informações de ambas as colunas resumidas em apenas uma, para mais informações acesse este artigo.

É possível notar que estas estratégias são usadas para reduzir duas features em apenas uma, mas é necessário ter cuidado, nem sempre será possível fazer isso, há a possibilidade das novas colunas, criadas com essas técnicas, não serem capazes de resumir as informações de forma satisfatória. Para avaliar as features criadas serão utilizados, mutual information no caso do clustering e razão de variância explicada para o PCA. O valor da razão de variância explicada é dado em porcentagem e representa o quanto a variação de um valor pode explicar a variação de outro, no nosso exemplo, este teste indicaria o quanto a nova feature foi capaz de representar as informações das features originais.

1. No banco de dados em que estamos trabalhando existem duas features com valores contínuos, são "Fare" e “Age”, portanto, a partir da interação entre elas pode-se criar grupos utilizando K-means como forma de clustering.

    a. O primeiro passo é analisar a distribuição dos passageiros levando em consideração “Fare” e “Age”

        i. Produzir um gráfico, do tipo scatter plot, colocando “Age” e “Fare” como eixos.
        ii. Nesta distribuição é possível identificar outliers, esses valores devem ser retirados para que o algoritmo k-means, encontre os grupos da maneira correta. Como esses valores podem ser retirados?

    b. Agora será necessário criar os grupos utilizando a biblioteca K-means do sklearn

        i. Crie um dataframe que contenha apenas as features “Age” e “Fare”, para remover os outliers, selecione apenas os passageiros que tenham as taxas menores que 500.
        ii. Crie um algoritmo k-means que produza 6 grupos.
        iii. Utilizando este algoritmo, produza uma nova coluna no dataframe criado, contendo os grupos de cada passageiro

    c. Agora será útil entender quais as características de cada grupo criado pelo algoritmo, para isso serão utilizados dois gráficos, um para identificar os grupos na distribuição “Age” x “Fare”, outro para avaliá-los em relação à chance de sobrevivência.

        i. Adicionar a nova feature ao banco de dados de treino.
        ii. Construir um gráfico scatter plot que mostra a distribuição “Fare” x “Age”, tendo os grupos como camadas.
        iii. Um gráfico bar chart que indique a chance de sobrevivência, em média, de cada grupo.
        iv. Faça uma análise qualitativa do que cada grupo representa e discuta sobre as chances de sobrevivência médias

2. Ao analisarmos a nova feature, é fácil notar que existem valores nulos, estes valores correspondem aos outliers que foram retirados para a construção dos grupos. Para substituir os NaN, deve-se identificar quem são esses outliers e designar, manualmente, seu grupo

    a. Identifique quem são os outliers

        i. Utilize o método .loc do pandas para localizar as linhas onde ocorrem os valores nulos da nova feature
        ii. Analise as características destes passageiros e defina qual grupo melhor encaixam

    b. Substitua os valores nulos da nova feature pelo grupo em que melhor se encaixam

        i. Obtenha novamente as linhas que contém NaN e substitua apenas o valor da nova feature
        ii. Observe se os valores foram substituídos de forma adequada realizando o print das linhas editadas

3. Agora avalie, utilizando mutual information, se a nova feature é significativa para a previsão do target (Utilize os passos mostrados em 9.1 - 1)

4. As features originais não representam a mesma grandeza, “Age” conta a idade em anos e “Fare” a taxa de embarque em dólares, portanto, para que seja possível utilizar o PCA, deve-se primeiramente realizar os scaling das variáveis. para esta etapa será utilizado o MinMax Scaler do scikit learn

    a. Scaling das features “Age” e “Fare”, entre 0 e 1

        i. Criar um scaler utilizando minmax scaler
        ii. Criar um dataframe para armazenar as features após o scaling
        iii. Criar um loop que passe pelas duas features que serão escaladas
            1. produzir uma coluna no novo dataframe que contenha a respectiva coluna do banco de dados de treino.
            2. Ajustar e transformar a nova coluna utilizando o minmax scaler

    b. Para garantir que o processo foi realizado com sucesso agora deve-se visualizar as duas colunas, elas devem ter seus valores entre 0 e 1 e manter sua forma original

        i. Fazer um histograma com a coluna “Age” do novo dataframe.
        ii. Fazer um histograma com a coluna “Fare” do novo dataframe.

5. Agora já é possível realizar o PCA, contudo, não há garantias que esta estratégia trará novas informações úteis para o modelo, portanto, é necessário, após a criação dos novos eixos, analisar criticamente se vale a pena introduzi-los ao banco de dados.

a. Realizar o PCA no dataframe com features escaladas

   i. Criar um PCA utilizando a biblioteca do sklearn
   ii. Ajustar e transformar o dataframe com features escaladas utilizando PCA.
   iii. Criar um dataframe com os novos dados após a transformação com PCA.
   iv. Criar um dataframe com os valores que foram multiplicados os eixos das features “Age” e “Fare”.
   v. A partir dos valores do item iv, analise se os eixos foram significamente alterados, se sim, descreva qual a nova relação que se pode perceber entre as duas features

6. Agora, o último passo será avaliar a nova feature criada pelo PCA, utilizando o mesmo processo encontrado no curso de feature engineering do kaggle, no capítulo Principal Component Analysis. É viável, com essas técnicas, a substituição das features “Fare” e “Age” por apenas uma?

9.4 - Target Encoding

Até agora, quase todas as features categóricas já foram codificadas, apenas uma ainda resta. No item 9.2 foi criada a feature “Title”, a partir de “Name”, essa coluna contém diversos valores únicos, de forma que é promissora a utilização do target encoding para codificá-la. Esta estratégia utiliza a relação da feature categórica com o target para substituir seus valores, ou seja, é uma estratégia considerada de supervised learning. Para o target encoding será utilizado o algoritmo MEstimateEncoder.

1. Deve-se então realizar o encoding da feature “Title”, após isso será interessante analisar a distribuição dos valores resultantes e compará-los com o target

   a. Encoding da feature “Title”

      i. Criar um encoder utilizando o M-estimators indicando qual feature será codificada e utilizando m = 5
      ii. Criar um dataframe com os valores codificados da feature “Title”, utilizando o encoder, indicando a feature que será codificada e o target que será utilizado para codificação

   b. Plot da distribuição dos valores codificados

      i. Criar um histograma com os valores da feature “Survived”
      ii. Criar um gráfico do tipo kdeplot, sobreposto ao primeiro, que indique a distribuição dos valores da feature “Title” codificada, não esqueça de adicionar legenda

Unidade 3 - Deep Learning

10 - Redes Neurais Simples

2 dias para completar capítulo

A partir deste capítulo será introduzida uma nova estratégia para machine learning. Conhecida como deep learning, esta maneira de se treinar um modelo se baseia na utilização de redes neurais para capturar os padrões que relacionam os dados de treinamento. Será visto a seguir que esta é uma ferramenta muito poderosa, com ela será possível capturar relações não lineares entre os dados, trabalhar com bancos de dados muito maiores e trabalhar com dados não estruturados, ou seja, que não são em forma de tabela, como áudios, textos e imagens.

O framework que será utilizado para os exercícios envolvendo deep learning é o keras. No curso Intro to deep learning do Kaggle estão exemplos de como esta biblioteca funciona, assim como exercícios para a construção, treinamento e avaliação de um modelo de deep learning. As redes neurais tem suas vantagens e desvantagens em relação ao ML clássico, apesar de poderem identificar padrões mais complexos, necessitam de um maior poder computacional para utilizá-las.

Inicialmente serão utilizados bancos de dados tabulares, contudo, com muito mais informações do que os já vistos até aqui. O dataset Weather in Szeged 2006-2016 será estudado como exercício de aplicação para as redes neurais, nele estão contidas informações sobre o clima em Szeged, na Hungria, desde 2006 até 2016, dessa, pode-se trabalhar com um banco de dados relativamente grande. O objetivo será prever o sumário do clima no dia (“Daily Summary”), a partir das informações das outras colunas, mas antes de treinar a rede neural será necessário uma breve exploração e limpeza do banco de dados. Para a criação deste estudo de caso foi utilizado o notebook weather prediction ,regression , neural model de salma maamouri

10.1 - Exploração do banco de dados

Assim como foi feito na competição Titanic, é necessário conhecer os banco de dados antes que se possa limpar os dados. Nesta sessão será realizada uma exploração a fim de entender melhor o banco de dados e identificar possíveis falhas que possam afetar o treinamento do modelo.

1. O primeiro passo é importar e visualizar o banco de dados, rastrear os valores faltantes e identificar os momentos estatísticos existentes

a. Importar o banco de dados utilizando o pandas

b. Identificar o número de valores faltantes em cada colunas

    i. Qual sua hipótese para a existência desses valores faltantes?

c. Gerar momentos estatísticos do banco de dados

    i. Qual coluna parece fora do comum? porque?

d. Utilizando o método ".hist" do pandas realize plots de distribuição dos valores das colunas

    i. É possível identificar algo de estranho com a coluna "Pressure(millibars)". Identifique e explique o que há de errado
    ii. Utilizando ".loc" do bandas, identifique a quantidade de valores errados

2. Deve-se agora definir quais são as features categóricas e quais são as numéricas, para isso existe o método ".select_dtypes", aqui será necessário identificar apenas os nomes das features

    a. Identifique as features numéricas, aquelas que são do tipo "int64" ou "float64", utilizando ".select_dtypes"

    b. identifique as features categóricas, aquelas que são do tipo "object"

        i. Uma das features é do tipo object mas não é categórica. Qual é ela?

    c. Inspecione os valores únicos de cada feature categórica, para garantir que não há nenhum valor que ocorra apenas uma vez, o que iria interferir na separação do banco de dados em treino e validação (Por que?). Utilize o método ".value_counts" para isso.

        i. Há uma feature em que existem valores que ocorrem mais de uma vez. esse valores serão retirados do banco de dados futuramente

10.2 - Limpeza do banco de dados

A partir das informações obtidas durante a exploração do banco de dados será agora necessário realizar sua limpeza. Aqui serão usadas pipelines que, após configuradas, realizarão a substituição dos valores faltantes, encoding das variáveis categóricas e, como saída, gerarão um objeto do tipo np.array, necessário para o treino da rede neural.

1. Uma das features do tipo "object" é "Formatted Date" que, na verdade, indica a data do registro climático. Essa coluna deverá então ser transformada para o formato de data para que seja possível extrair informações úteis para o treinamento do modelo

a. Existe um valor nessa feature que indica o fuso-horário (“+0200”). Para que a coluna seja corretamente transformada este valor desse ser retirado

    i. Deve ser realizado um processo semelhante ao item 9.2-1, em que será utilizado o método “str.split” para retirar o texto “+0200” de todos os valores da feature.
    ii. Use o método "to_datetime" para transformar a coluna no formato desejado

b. Utilizando os métodos existentes em “.dt” crie novas colunas retirando as informações de hora, dia, mês e ano de cada valor de data na feature

    i. Utilize os métodos “dt.hour ”, “dt.day ”, “dt.month ”, “dt.year ”
    ii. retire a coluna original de data do banco de dados

2. Agora será criada a pipeline para completar a limpeza dos dados, com ela será realizada a substituição dos valores faltantes nas colunas “Pressure (millibars)” e “Precip Type” e o encoding, utilizando ordinal encoder, das variáveis categóricas. Para isso será utilizada a biblioteca Column Transformer do sklearn, mas antes deverão ser retirados os dados que não servirão para o treinamento do modelo.

    a. Retirar a coluna “Loud Cover”, já que contém apenas valores nulos, e linhas cujo valor da coluna “Summary” aparecem apenas uma vez

        i. Para localizar as linhas que serão retiradas utilize o método “.loc”

    b. Definir imputers para variáveis numéricas e para categóricas, e o encoder das variáveis categóricas

        i. Configurar imputer das variáveis numéricas com simple imputer, utilizando a média como substituta do valor faltante e indicando que o valor 0 significa valor faltante (valores iguais a 0 em “Pressure (millibars)” representam valores faltantes).
        ii. Criar uma pipeline para tratamento das variáveis categóricas, primeiro passo deve ser a substituição dos valores faltantes utilizando simple imputer, substituindo o valor faltante por um texto, o segundo passo é realizar o encoding utilizando ordinal encoder

    c. Criar um objeto do tipo Column Transformer para aplicar os passos criados anteriormente

        i. Configurar column transformer para fazer o tratamento das variáveis numéricas e categóricas

3. O último passo da limpeza dos dados será a separação em treino e validação, aplicação da pipeline de limpeza e o encoding da coluna que será o target (“Daily summary”)

a. Definir quais são as features, o que é o target e separar o banco de dados utilizando train-test split, deve-se garantir que todos os valores do target estão distribuídos de forma equilibrada na reparação do banco de dados.

    i. Indicar o target como sendo “Daily Summary” e as features como as colunas restantes.
    ii. Utilizar train-test split, configurado para que o tamanho do banco de dados de teste seja 30% do banco de dados original e utilizando o target como parâmetro de “stratify”.

b. Aplicar pipeline de limpeza do banco de dados

    i. Utilize apenas as features dentro da pipeline, ajuste utilizando as features de treino e apenas transforme as de teste.

c. Utilize label encoder para realizar o encoding do target

    i. Da mesma forma, utilize o target de treino para ajustar e apenas transforme o de teste

10.3 - Treinamento e Avaliação da rede neural

Após realizada a limpeza de dados, o último passo será construir a rede neural, treiná-la e avaliá-la. Para isso será utilizado o framework Keras, ao final do treinamento serão mostrados gráficos que indiquem a evolução da acurácia e do erro, calculado por cross entropy, em relação ao número de épocas utilizadas para o treinamento. Outro passo importante é a indicação para o modelo de que o target é um valor discreto, para isso será utilizada uma ferramenta nativa do keras que transforma o target de forma semelhante ao one-hot encoding.

1. Primeiramente deve-se criar a arquitetura da rede neural, ela terá duas camadas escondidas, sendo cada uma com uma saída de dimensão 256, entre elas serão utilizados Batch normalization e Dropout. Como funções de ativação serão usados “relu” para as camadas escondidas e o “softmax” na camada de saída, esta função serve para que o modelo realize predições na forma de classificação não binária, ou seja, que seja capaz de prever mais de duas classes.

    a. Configurar o tamanho da entrada, que é o número de features para treinamento, e o tamanho da saída, que será a quantidade de classes do target

        i. utilizar o método “.shape” para descobrir quais as dimensões das features e “.unique” para descobrir quantas classes existem no target

    b. Construir arquitetura do modelo utilizando a descrição já informada

        i. Camada de entrada utilizando batch normalization configurando seu tamanho a partir das features do banco de dados

ii. Primeira camada escondida utilizando “relu” como função de ativação, saída de dimensão 256, batch normalization e dropout de 0.3
iii. Segunda camada escondida igual à primeira
iv. Camada de saída com tamanho igual ao número de classes do target e função de ativação “softmax”

c. Realizar tratamento do target utilizando o método “utils.to_categorical” do keras

i. Transformar target de treino e de teste utilizando o método do keras
ii. Veja as dimensões da matriz de target e responda, o target é composto de quantas classes (Possíveis previsões)?

2. Configurar o otimizador, a função de perda que será utilizada e a métrica para avaliação do modelo

a. Como otimizador será utilizado “adam”, como função de perda “categorical_crossentropy” e como métrica para avaliação “categorical_accuracy”

3. Por fim, será realizado o treinamento do modelo e plotagem dos resultados das avaliações, será utilizado o early stopping para evitar o overfitting. Os hiperparâmetros que serão utilizados, como o número de épocas e tamanho de cada batch durante o treinamento do modelo podem ser alterados pelo aluno para teste e possível aprimoramento.

a. Configurar early stopping

i. Utilizar a biblioteca do keras para configurar early stopping, com uma tolerância de 10 épocas e uma variação mínima no resultado da função de perda de 0.001

b. Realizar o treino e armazenar os resultados da avaliação do modelo

i. Utilizar as features e target de treino para treinar o modelo
ii. Utilizar os dados de teste para validar o modelo
iii. Utilizar um tamanho de batch de 3000
iv. Utilizar 100 épocas para treino
v. Indicar utilização de early stopping

c. Para a plotagem deve-se transformar os resultados do treino em um dataframe do pandas e plotar os valores de perda e acurácia em função das épocas

i. Utilizar o método “.DataFrame” para transformar resultados em data frame do pandas
ii. Utilizar gráfico de linha para plotar os resultados da perda e da acurácia em função das épocas de treino

11 - Redes Convolucionais para Visão Computacional

2 dias para completar capítulo

Existe um outro tipo de rede neural que é largamente utilizada nas aplicações de ciência de dados, sobretudo quando se trata de análise de bancos de dados de imagem, esta é a chamada rede neural convolucional (CNN). As imagens são consideradas dados não estruturados, já que não estão em forma de tabela, elas podem ser vistas como matrizes de duas dimensões, no caso de imagens em preto e branco, ou de três dimensões, para imagens coloridas, em que cada pixel é um valor da matriz, que vai de 0 (preto) a 255 (Branco). As imagens coloridas geralmente contém 3 camadas, uma com as cores vermelhas (R), outra com as verdes (G) e a última com as azuis (B), que, quando sobrepostas, formam as cores da imagem.

Até aqui, com a utilização de dados estruturados, foi uma tarefa simples identificar as características dos bancos de dados, sendo elas as colunas das tabelas que não fossem o target, já para os dados não estruturados, como imagens, áudios e textos, isso não é tão simples. No exemplo da foto de um rosto humano, as características da imagem podem ir de coisas simples como linhas horizontais, verticais e círculos, até algo mais complexo, como os olhos, a boca e o nariz. Para a extração dessas características são utilizadas as CNNs, que são filtros capazes de identificar as diferentes características das imagens, sendo que, quanto mais camadas a rede possui (quanto mais profunda) mais complexas as características que podem ser captadas por esses filtros. Para mais informações acesse este artigo.

Nesta apostila será utilizada a biblioteca de imagens CIFAR 10, que apresenta 10 classes de imagens com 5000 exemplares cada, exemplos dessas classes são: avião, cachorro, gato. O objetivo então será criar uma CNN que irá identificar as características das imagens e classificá-las de acordo com as 10 classes do banco de dados, para isso será utilizado o framework keras, assim como no capítulo anterior. Para o aprendizado de como utilizar o Keras para a construção de uma CNN será necessário realizar o curso de visualização de dados do Kaggle.

Como serão utilizadas imagens para o treinamento da rede neural, é aconselhável que o aluno produza seu código no Google Colab. nesta plataforma será possível a utilização de uma GPU, que é uma unidade de processamento específico para imagens, ou TPU, que é feita para processar tensores. Para conectar o Colab ao drive e realizar as configurações de processamento necessárias, acesse este link.

11.1 - Importação e visualização do banco de dados

Após a montagem do notebook no drive será necessário agora importar as bibliotecas iniciais e banco de dados para a construção do modelo. As bibliotecas mais importantes para a importação são keras, matplotlib, numpy e pandas, utilize outros notebooks já produzidos durante a apostila como modelo de como importar essas bibliotecas. Copie a função “set_seed” de um dos notebooks do curso de computer vision do keras.

1. O banco de dados CIFAR 10 será importado diretamente de “Keras.datasets”, desta maneira os dados já estarão separados em treino e teste, facilitando a produção do código. Será necessário escalar os pixels de cada imagem, para que se transformem

em valores de 0 a 1, além disso, o target também deve ser transformado da mesma forma que no capítulo anterior.

a. Utilizando keras.datasets, importe o banco de dados separando em imagens e labels de teste e imagens e labels de treino

    i. Utilize o comando “cifar10.load_data()”. Este comando gera duas tuplas como saída, correspondentes a banco de dados de treino e de teste, cada tupla contém imagens e seus respectivos rótulos

b. Realize o scaling das imagens e o one-hot encoding do target

    i. Divida as variáveis correspondentes às imagens por 255, dessa forma os valores de cada bit ficaram entre 0 e 1

    ii. Utilizando a biblioteca “keras.utils.np_utils”, através do método “.to_categorical”, realize o one-hot encoding dos rótulos das imagens. OBS: armazene os rótulos após o encoding em outras variáveis

2. Para um melhor entendimento do banco de dados, principalmente quando é de imagens, a visualização é fundamental, para isso será criada uma função que gere exemplos das imagens do dataset, com seus respectivos rótulos.

    a. Os argumentos da função serão imagens e os rótulos originais, sem one-hot encoding

    b. Será necessário transformar os números das categorias para o rótulo que elas representam. Utilize este link para identificar qual número representa qual rótulo

        i. Transforme o argumento de categorias em uma série do Pandas, para isso utilize também o método “.flatten()” do numpy para diminuir a dimensão dos rótulos

        ii. Utilize o método “.map” para transformar cada número em sua respectiva categoria

    c. Crie uma figura que tenha 3 linhas e 7 colunas de imagens do banco de dados CIFAR 10, cada uma com seu respectivo rótulo

        i. Criar figura com “plt.figure” e configure seu tamanho para (14, 6)

        ii. Faça um loop com um range de 3 * 7 = 21, dentro dele deverão estar contidos os seguintes itens:

            1. Utilizar “plt.subplot” configurando o número de linhas para 3, de colunas para 7 e o índice sendo o número em que o loop se encontra mais 1.

2. Executar os métodos “plt.xticks” e “plt.yticks” vazios

3. Com o método “plt.imshow”, selecione a imagem que será mostrada no índice correspondente da figura (Utilize o número em que o loop se encontra)

4. Utilizando o método “plt.xlabel()” selecione o rótulo correspondente à imagem do item anterior

iii. Utilize “plt.show” fora do loop para mostrar a imagem construída

iv. Execute a função de visualização utilizando as imagens de treino e seus rótulos como argumento.

11.2 - Criação da Rede Neural Convolucional

A criação das CNNs para este capítulo irá ocorrer por meio de blocos para a maior clareza do código, além de maior facilidade para o teste de arquiteturas e hiperparâmetros diferentes. Serão criadas duas arquiteturas inicialmente idênticas, a diferença será a utilização ou não de data augmentation durante o treinamento, para que, ao final, os dois tipos de treinamento sejam avaliados e comparados.

A construção das CNNs será ligeiramente diferente da utilizada no curso de computer vision do Kaggle, o método “Sequencial” não será utilizado, ao invés disso a sintaxe do Keras Layers API oferece um modo de conectar o input de uma camada ao output de outra com a utilização de parênteses ao lado do instanciamento do layer, para identificação do input.

1. Serão criados dois blocos, um para as camadas escondidas da rede neural, base, e outro para a classificação e geração do output, head. Estes blocos serão construídos por meio de funções, além disso, será adicionada a opção da utilização ou não de drop out e batch normalization em cada camada.

    a. Criar função para construção de bloco de base para rede neural. Este bloco deverá conter uma camada convolucional e uma de max pooling, ambas com ativação ReLu, além da opção de drop out e batch normalization

        i. Argumentos da função:
            1. Input layer: são os dados de input do bloco
            2. Filters: a quantidade de filtros da camada convolucional
            3. Kernel size: o tamanho da matriz de kernel da camada convolucional
            4. Activation: a função de ativação das duas camadas, inicializada com ReLU
            5. Padding: modo de padding das duas camadas, inicializado com “same”

6. Drop out: booleano para indicar presença ou não de drop out no bloco, inicializado como verdadeiro
7. Normalization: booleano para indicar presença ou não de batch normalization no bloco, inicializado como verdadeiro
8. Drop out ratio: decimal indicando a razão de funcionamento do drop out caso tenha no bloco

ii. Criar camada convolucional, utilizando a sintaxe do Keras Layers API, instanciando Conv2D com os parâmetros pertinentes presentes nos argumentos da função. O input deve ser o argumento Input Layer

iii. Criar camada max pooling, utilizar padding como argumento, o input deve ser o output da camada convolucional

iv. Criar condicionais para drop out e batch normalization, a entrada dessas camadas deve ser a saída do max pooling

v. A função deve retornar o output das camadas de drop out e batch normalization, caso existam, ou do max pooling

b. Criar função para a classificação das imagens, este bloco deve diminuir a dimensão das imagens, classificá-las de acordo com o target e gerar saídas em probabilidade para cada classe

i. Argumentos da função
1. input layer: dados de input do bloco
2. neurons: a quantidade de neurônios na camada de classificação
3. dense activation: função de ativação da camada de classificação, iniciada com ReLu
4. out activation: função de ativação da camada de saída, iniciada com softmax, para transformar valores em probabilidades
5. num classes: número de classes do output, iniciado com valor 10

ii. Criar camada para achatamento das imagens (torná-las arrays de apenas uma dimensão) utilizando “Flatten()”, a entrada devem ser os dados de entrada do bloco

iii. Criar camada para classificação com a quantidade de neurônios do argumento, a entrada deve ser o output da camada de achatamento

iv. Criar camada de saída com a função de ativação softmax e número de classes dos argumentos, a entrada deve ser o output da camada de classificação

v. A função deve retornar o output da camada de saída

2. A partir dos blocos já criados, agora serão elaboradas as arquiteturas das redes neurais. Inicialmente serão utilizadas 3 camadas convolucionais, uma de entrada e o head para a saída. O otimizador será Adam, cross entropy como função de perda e acurácia como métrica. Early stopping também será usado para evitar o overfitting do modelo.

    a. Elaborar a arquitetura do modelo que será utilizado para treinamento sem data augmentation

        i. Utilizar Keras Layer “Input()” como entrada, especificando dimensões (32, 32, 3) das imágens do banco de dados

        ii. Executar função de construção de bloco base, com input sendo a saída da camada de input, número de filtros igual a 32 e kernel size igual a 3

        iii. Criar outros dois blocos, um com 64 e outro com 128 filtros, kernel size de 3, sendo suas entradas os outputs das camadas anteriores

        iv. Criar bloco de saída, com 128 neurônios na camada de classificação e entrada sendo o output do bloco anterior

        v. Para compilar o modelo, Utilize a classe “Model” do keras, sendo o input a camada de entrada e o output a camada de saída

        vi. Instancie Early Stopping, utilizando min_delta = 0, patience = 10 e indique a restauração dos melhores pesos

        vii. Compile o modelo, utilizando o otimizador Adam, binary cross entropy como perda e acurácia como métrica. Por fim, gerar o sumário do modelo

    b. Crie uma segunda rede neural, utilizando a mesma arquitetura e parâmetros da anterior, apenas mudando seus nomes. Essa segunda rede será utilizada para treinamento com data augmentation.

11.3 - Treinamento e Avaliação da CNN

Após a criação da rede neural será necessário treiná-la, para uma das redes será utilizado o data augmentation através do Image Data Generator do keras, para a outra, o treinamento será simples. Ao final, serão gerados gráficos da evolução da perda e da acurácia dos modelos com o passar das épocas, para que seja possível avaliar e comparar qual modelo obteve a melhor performance.

1. A partir do método “.fit”, treine o modelo sem data augmentation, utilizando as imagens e rótulos de treino para treino e os de teste para validação, configure o batch size para 128 e o número de épocas para 50, como callback utilize o early stopping

2. Agora será utilizado o data augmentation para tentar melhorar a performance e generalização do modelo. Primeiramente, devemos criar um gerador de dados aumentados, ajustá-lo ao modelo e demonstrar exemplos de seu funcionamento, após isso ele será implementado na rotina de treinamento

    a. Com a biblioteca image data generator, crie, ajuste e gere imagens de um gerador de dados aumentados, com giro horizontal e alteração de 0.3 na largura e altura.

        i. instanciar objeto ImageDataGenerator com, horizontal flip = True, width e height shift iguais a 0.3

        ii. Utilizar método “.fit” para ajustar gerador às imagens do treino

        iii. Gerar batch de banco de dados, com imagens e rótulos, utilizando método “.flow”

        iv. Utilizar a função “next()” do python para executar uma geração de dados. Esta função terá duas saídas, uma sendo as imagens e outra os rótulos

        v. Com os dados obtidos pela função anterior, gere uma figura com imagens como exemplo por meio da função visualize data, criada em 11.1

    b. Com a utilização do data augmentation, o ideal é que a cada época sejam gerados novos dados aumentados, então o gerador será colocado dentro do treinamento do modelo

        i. Treinar modelo criado para data augmentation

        ii. O método “.flow” deverá ser utilizado ao invés dos bancos de dados de treinamento
            1. Imagens e rótulos de treinamento devem ser utilizados no método
            2. O batch size é 128

        iii. Ao invés de configurar o batch size, com o data augmentation deve ser utilizado "steps_per_epoch", e seu valor deverá ser a quantidade de imagens dividido pelo batch size (Funciona de forma semelhante à configuração do batch size)

        iv. Utilize 50 épocas para o treinamento, os dados de teste para validação e early stopping como callback

3. Para a avaliação serão gerados dois gráficos de linha, um representado o valor da acurácia e outro o da função de perda, ambos em função das épocas de treinamento

    a. Utilizar “plt.subplots” com 1 linha e duas colunas, tamanho da figura (24,6)

    b. O primeiro gráfico deve conter os valores de acurácia em relação ao banco de dados de treino e de validação do treinamento de ambos os modelos. Devem ser adicionadas legendas que representam cada gráfico.

        i. Plotar na cor laranja e linha contínua a acurácia em relação ao banco de dados de treino (history[‘accuracy’]) do modelo sem aumento de dados

        ii. Plotar na cor laranja e linha tracejada a acurácia em relação ao banco de dados de teste (history[‘val_accuracy’]) do modelo sem aumento de dados

        iii. Plotar na cor azul e linha contínua a acurácia em relação ao banco de dados de treino (history[‘accuracy’]) do modelo com aumento de dados

        iv. Plotar na cor azul e linha tracejada a acurácia em relação ao banco de dados de teste (history[‘val_accuracy’]) do modelo com aumento de dados

    c. Agora deve ser gerado um novo gráfico semelhante ao anterior, contudo seus valores devem ser correspondentes à função de perda (cross entropy)

    d. Por fim, utilize o método “model.evaluate” para gerar valores finais de ambos os modelos, compará-los e definir qual obteve o melhor desempenho