Material:

Q: Onde posso encontrar os materiais das aulas do professor Seiji e professor Mangan?

R: Ao fazer o download do material de apoio que consta no ícone de folha a4 ao lado do título da disciplina das aulas do professor Seiji é possível encontrar todos os links para os na apresentação PPT disponível.

A exceção consta próximo ao minuto 06 da parte 2 da aula 2, onde o professor Seiji abre o link do artigo sobre ensembles, segue o link do artigo: https://www.datasciencecentral.com/profiles/blogs/want-to-win-at-kaggle-pay-attention-to-your-ensembles.

Sobre o link do Scipy Lecture é exatamente o que o professor utilizou para guiar a aula em questão de conceitos do Python.

O professor Mangan utilizou a documentação da Scipy para lecionar suas aulas. O material está disponível em:

https://scipy-lectures.org

O professor Mangan utilizou a tradução automática gerada pelo programa Tradutor da Google:

https://translate.google.com/translate?sl=en&tl=pt&u=https://scipy-lectures.org

Ferramentas:

Q: Nos instantes 37:40 e 49:50 da parte 2 da aula 2 o professor cita duas bibliotecas, uma para a área de teste e outra para área de reconhecimento de voz. Há algum link que forneça mais informações sobre estas duas ferramentas?

R: Quanto a qualidade de código, é possível que o professor tenha se referido à biblioteca *Structure 101* disponível em: https://structure101.com/products/studio/.

O professor também se referia à biblioteca *Kaldi* disponível em: https://github.com/kaldi-asr/kaldi para transcrição de voz.

Quanto à biblioteca *Understand* não encontramos nenhuma referência.

Conceitos e Exercícios:

Q: Tem algum material para indicar para estudar um pouco sobre gráficos, tipos e suas aplicações? Não necessariamente relacionada a alguma tecnologia como Python, para um estudo mais teórico.

R: O curso oferece uma ótima disciplina sobre visualização de dados com a profª Isabel e há uma demonstração de alguns com o professor Renan Xavier na disciplina de Introdução à Ciência de Dados e à Inteligência Artificial.

Quanto à tecnologia Python, há algumas bibliotecas que permitem algumas explorações visuais como:

Seaborn: https://seaborn.pydata.org/

Matplotlib: https://matplotlib.org/

Uma recomendação do professor Mangan para esse tópico também é um livro que trata sobre como usar gráficos:

Knaflic, Cole Nussbaumer (2015) Storytelling with data: a data visualization guide for business professionals.

Na biblioteca Central da PUCRS: http://primo-pmtna01.hosted.exlibrisgroup.com/PUC01:PUC01:oclcwiley_e000xww_923807870

Q: Onde consigo acesso aos links do drive com o material, por favor?

Os links encontram-se no material de apoio, ao lado do nome da disciplina possui um ícone A4, ele levará aos materiais utilizados pelo professor, na aula 1 há um PDF, nele você encontra os links dos arquivos notebooks do professor com as respostas.

Olá. Eu estava executando o exercício presente no minuto 12 da aula e recebi uma mensagem de erro. Ao pesquisar no *stackoverflow*, descobri o seguinte: o módulo *sklearn.datasets.samples*\generator_ foi substituído por *sklearn.datasets*. Sem essa substituição, o exercício dá erro.

O comando em funciona normalmente dependendo da versão que se está utilizando, talvez seja o caso de utilizar ou esse comando que você indicado, ou instalar o pacote de samples do *scikit learning*.

Q: Boa tarde! Não consegui descobrir como importar o *logistic regression* para o Python, quero dizer para o *Google Colaborate* .

R: O *logistic regression* é uma *feature* do *scikit-learn*, a princípio, se você executar o comando "*from sklearn.linear_model import LogisticRegression*" que tem no *Google Colab* ele deve executar e ficar com um "*checkzinho verde*" ao lado da célula.

Caso queira instalar o *scikit learn* você pode executar o comando "*pip install scikit-learn*" que ele irá instalar os pacotes do *scikit* no seu ambiente.

Q: Conforme imagem pcdd01.jpg, foi questionado sobre como executar a soma dos valores de um vetor a. Eu respondi *a.sum()* e foi considerado como errado. Porém, de acordo com a documentação na imagem pcdd02.jpg (ou link http://scipy-lectures.org/intro/numpy/operations.html#computing-sums), minha resposta está correta e a alternativa marcada como a certa é que está errada. Para fazer a soma utilizando a estrutura sum(a) seria necessário referenciar ao pacote *numpy* , com a seguinte estrutura: *numpy.sum(a).*

O gabarito está correto mesmo. Segue:

A premissa da questão é tratar das funções primitivas do Python, ou seja, as que não precisam de uma biblioteca como a *numpy*.

No seu próprio comentário, você indica soma de valores de um VETOR, mas a questão indica LISTA.

Para que a opção escolhida por você fosse correta, você deveria transformar a lista em um vetor de valores *numpy* "a" e depois sim, executar o comando *a.sum()*.

Por último, mas não menos importante, ao usar o *numpy* você deixa de atender ao texto-base. O *numpy* é uma biblioteca adicional.

Q: Quando o professor explica sobre a separação de dados de treino e teste, se refere a *features* (representados por uma matriz) e labels (array). Em suma uma matriz não é um array, fiquei um pouco confuso com isso.Entendi que features são os atributos descritivos (que conhecemos) e labels é o que estamos tentando prever, correto?

R: Quando você separa entre treino e teste, nada mais é que separar o seu conjunto de dados em 2 (e.g. 2 dataframes). Para treinar o seu modelo de IA, terá os dados de treinamento, onde você fornece todos os atributos (colunas do seu dataframe\matriz) e para cada registro (linha) terá um label e durante o treinamento, o algoritmo irá aprender a relação entre o conjunto de variáveis e esse label. Quando tratamos de matriz é por que são várias linhas (registros) x várias colunas (variáveis\atributos). Quando tratamos de labels é um array por que é uma lista de valores a serem previstos mesmo. Espero que tenha te ajudado, caso contrário me avise, bons estudos.

Q: Fiquei com um dúvida na parte do Kmeans. No exemplo passado tanto o eixo X quanto o Y do scatter plot são numéricos. Mas muitas vezes o eixo X é um texto. Por exemplo, agrupar países em 4 clusters pelo seu IDH. Pesquisei e encontrei que eu deveria transformar o que é texto em numérico mas que isso poderia gerar diferenças nos resultados. Nesse exemplo que eu dei do país, o Kmeans ainda seria o método recomendado?

R: O KMeans pode ser utilizado sim dentre outros para agrupamento. A questão é que devemos passar as variáveis texto para variáveis categóricas\numéricas. Há diversas formas de fazer isso como o one-hot encode entre outras. A questão do seu exemplo seria facilmente resolvida ao gerar categorias ou valores ao invés de usar o nome do país. Caso fosse utilizar algum dashboard\gráfico depois, você poderia fazer apenas um de -> para para aparecer os labels depois.

Q: Podes, mesmo que de forma muito empírica, explicar o que são "pequenos volumes de dados" e o que são "grandes volumes de dados" - e claro, especificar que capacidade de processamento é necessário em cada caso.

R: Na literatura não há nada que delimite o que é um grande volume de dado e o que é um pequeno volume de dado, pois mesmo que você conte com um pequeno número de registros (linhas) seu dataset pode ter um grande número de colunas (variáveis), e conforme for você vai necessitar de um grande poder de processamento. Mas é comum vermos datasets menores (e.g. 10 mil linhas) darem um bom resultado em alguns processos de modelagem estatística, mesmo assim esse dataset pode ser de um pequeno volume de dados comparado a uma tabela de um banco de dados por exemplo do ramo financeiro que pode ter milhões ou até bilhões de registros. É comum também termos datasets de comentários de vídeos de youtube que terão 150 mil registros por categoria, e ai você tem um grande volume de dado. Da mesma forma, para processar nem sempre você conseguirá processar esse volume de dados em um tempo de resposta rápida em um computador doméstico por exemplo. Quando trabalhamos com a GPU dedicada, é bastante utilizada para treinar modelos de deep learning que fazem muitos cálculos, independente do volume dos seus dados. Por isso a Google oferece o Colab Pro, a AWS possui o SageMaker entre outras.

Q: Na aula 4 parte 3 aos 14 min o professor está exemplificando como utilizar o LinearRegression com os dados do tópico 3.6.2.1. O import do LinearRegrssion e do numpy ocorre bem, consigo criar os arrays x e y, mas ao treinar o modelo com "model.fit(X, y) o colab apresentar um erro, diz que algo foi descontinuado e não consigo prosseguir no exemplo.

R: Nesse caso, você deve estar utilizando a versão 1.0.2, que é a última versão estável do scikit-learn, conforme a documentação da biblioteca: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression . Também como visto na documentação, esse é um Warning (aviso) de que esse parâmetro foi deprecado e que será REMOVIDO nas próximas versões 1.1.x. Assim, a mensagem que você recebeu é o warning apenas e seu código funciona conforme a imagem que anexo neste comentário do código que repliquei no Google Colab. Podes reparar que há uns checkzinhos verdes de execução com sucesso, mesmo tendo o warning. Caso queira que a mensagem não apareça, você pode forçar a instalação de versões anteriores do scikit-learn, que irá funcionar também.

Q: Vendo a aula sobre arvores de decisão e radom forest o professor diz que é uma maneira de explicar a decisão tomada, certo? Então não seria possível utilizar arvores de decisão para a explicação no caso do tribunal ao final

da aula?

R: Sim, as árvores de decisão têm um bom poder de explicação para nos indicar quais parâmetros foram cruciais para aquela tomada de decisão. Nesse caso do tribunal, é possível sim que as árvores possam auxiliar no entendimento do algoritmo, dependendo das features utilizadas para o treinamento do modelo.

Q: O Numpy é uma biblioteca para armazenamento de dados do mesmo tipo, certo? Já o pandas aceita tipos diferentes de dados a cada coluna. Caso a tabela em que os dados estejam armazenados forem todas do mesmo tipo de dado, por exemplo inteiro, utilizaria-se o Numpy no lugar do pandas? ou ainda sim seria melhor utilizar o pandas?

R: O Numpy na verdade foi concebido para manipulação de arrays (multidimensionais (ndarrays) ou não), vetores, listas e etc.. Já o pandas ele vai te permitir trabalhar com dataframes, ou seja, conjuntos de dados de diferentes formatos (string, boolean, numérico entre outros). O Numpy vai permitir por exemplo operações rápidas para tratamento e limpeza de dados dentro de dataframes Pandas.

**FAQ gerado com base em comentários até o dia 21/04/2022.*