Relatório LAMIA - 4
Prática: Principais Bibliotecas e Ferramentas Python para Aprendizado de Máquina (I)

Kaique Medeiros Lima

Introdução

O conteúdo desse relatório refere-se ao CARD 4, que são seções do curso na plataforma Udemy, '*Python para Data Science e Machine Learning*'. As seções requeridas pelo enunciado do card foram as seguintes:

- **Seção 3: Jupyter Notebook**
- **Seção 5: Python para Análise de dados - Numpy**
- **Seção 6: Python para análise de dados - Pandas**
- **Seção 7: Python para análise de dados - Exercícios Pandas**

Descrição da atividade

Seção 3: Jupyter Notebook

A primeira seção foi curta, contendo apenas a apresentação da IDE. Seu jeito diferente de espaçamento de código, por blocos, podendo rodar blocos específicos, e não o arquivo inteiro. A existência do MarkDown serve para a organização do código, é como um comentário, mas não é in-code.

Também foi apresentado alguns atalhos de teclas, como:

- **Alt** + **Enter**: Roda o bloco de código e cria outro bloco em seguida.
- **Shift** + **Enter**: Roda o bloco, mas só cria outro caso seja o último.

Seção 5: Python para Análise de dados - Numpy

Seção de apresentação da biblioteca. Utilizada no manipulamento de arrays e matrizes uni e multidimensionais, além de possuir várias funções matemáticas *built-in* necessárias futuramente no aprendizado de máquinas. As principais funções que foram cobertas foram:

- **np.array(lista_exemplo)**: Transforma uma lista ou matriz existente em uma array Numpy.
- **np.arange(x, y, z)**: Cria um array novo, caso especifique o z, é de quanto em quanto os números sobem de x até y
- **np.zeros(x)**: Cria um array composto de zeros, com tamanho x.
- **np.ones(x)**: Cria um array composto de uns, com tamanho x.
- **np.eye(x)**: Cria uma matriz identidade com tamanho x.
- **np.linspace(x, y, z)**: Cria uma array de x até y, z define o tamanho e consequentemente de quanto em quanto aumenta.

- **np.random.rand(x)**: Array de tamanho x com números aleatórios em distribuição uniforme.
- **np.random.randn(x)**: Array de tamanho x com números aleatórios em distribuição normal.
- **np.random.randint(x)**: Array de tamanho x com números inteiros aleatórios.
- **np.round(x, y)**: Arredonda o valor x para y casas decimais.
- **reshape()**: Transforma, apenas em sua apresentação, um vetor em uma matriz, não altera fisicamente.
- **shape**: Parâmetro, não função. Retorna o tamanho.
- **max()**: Retorna o maior valor.
- **min()**: Retorna o menor valor.
- **argmax()**: Posição do maior valor.
- **argmin()**: Posição do menor valor.
- **copy()**: Apenas copia a array, mas não altera a original.

Houve a apresentação do fatiamento de arrays, que é com o uso do (:), do seguinte modo: (assuma todos os cálculos com 0, 1, 2...)

- **array[2:5]**: Imprime os elementos 2, 3 e 4.
- **array[2:]**: Imprimirá a partir do elemento 3 até o final, sem o 2.
- **array[:5]**: 0 ao 4.

Finalizando a seção, foi apresentado operações matemáticas em arrays pelo Numpy, como a soma de arrays, subtração, multiplicação, divisão, exponenciação, radiciação, média, desvio padrão, seno e valores máximos e mínimos.

Seção 6: Python para análise de dados - Pandas

Seção que apresenta a biblioteca Pandas, que é essencial na manipulação e análise de dados. Ferramenta eficiente, consegue lidar com alto volume de dados com boa performance. Utilizada para a construção de tabelas de dados. O modo de achar dados em DataFrames são os nomes das colunas/linhas dentro de colchetes, como: dataframe[‘colunaX’].

Os tutoriais dessa seção foram aprofundados na manipulação e seleção dos dataframes, como procurar valores específicos, selecionar seções específicas de tabelas, separar linhas e colunas. O parâmetro inplace=True é importantíssimo, pois é o que deixa alterações feitas nas tabelas permanentes.

As principais funções que foram cobertas foram:

- **pd.Series()**: Cria uma série, array rotulado.
- **pd.DataFrame()**: Cria uma estrutura de tabela.
- **df.reset_index()**: Transforma o índice em uma coluna.

- **df.set_index()**: Torna uma coluna em índice.
- **list(zip())**: Mescla uma lista com outra, transformando em tuplas.
- **pd.MultiIndex.from_tuples()**: Cria um índice multinível, índice secundário.
- **df.loc[‘alvo’]**: Localiza o alvo em índices multiníveis
- **df.xs()**: Cross Section, puxa elementos internos sem precisar mencionar os externos.
- **df.drop()**: Remove linhas ou colunas.
- **df.dropna(thresh = x)**: Só vai dropar os NaN se tiver x quantidades por linha.
- **df.fillna(value = x)**: Substitui os NaN por x.
- **df.fillna(method = 'ffill')**: Forward Fill, preenche o valor com o valor anterior.
- **df.groupby(‘Coluna’)**: Agrupa a coluna. Como se selecionasse uma área no Excel.
- **pd.concat([df1, df2, df3])**: Junta tudo, mas tem que bater as linhas e colunas.
- **pd.merge(esquerda, direita, how='inner', on='key')**: Especifica os dois dataframes. how = método de união de tabelas SQL, o inner já é padrão do merge. on = mesclar por essa coluna, pois já possuem valores iguais.
- **df.join(dataframe)**: Pode ter diferenças nos índices, junta quando é igual, se forem diferentes, coloca valores. Usa os índices do df, pois foi aplicado a função join() nela.
- **df[‘coluna’].unique()**: Retorna um Numpy array com apenas valores únicos.
- **df[‘coluna’].nunique()**: Retorna o tamanho, quantidade de valores únicos.
- **df[‘coluna’].value_counts()**: Une o unique() e nunique().
- **df[‘coluna’].apply(função)**: Aplica a função em cada elemento de ‘coluna’.
- **df.sort_values(by=’coluna’)**: Ordena os valores de ‘coluna’.
- **df.head()**: Mostra os 5 primeiros índices.
- **df.pivot_table()**: Cria um índice multi-multinível.
- **pd.read_csv()**: Lê o arquivo csv e transforma em DataFrame.
- **pd.read_excel()**: Lê o arquivo do Excel (.xlsx) e transforma em DataFrame.
- **pd.read_html()**: Lê o arquivo HTML e transforma em DataFrame.

Seção 7: Python para análise de dados - Exercícios Pandas

Seção dedicada para dois testes de exercícios sobre a biblioteca Pandas, desafiando a absorção do conhecimento passado pela seção 6. O nível dos exercícios, em geral, não eram muito complexos, mas os rotulados como “Difíceis” foram um bom desafio, os testes me fizeram perceber que não possuía muito entendimento da função lambda, o que melhorou após encerrar os dois testes.

Conclusões

O estudo dessas ferramentas e bibliotecas é crucial para qualquer profissional que deseja atuar na área de IA e aprendizado de máquina. As seções completadas ofereceram um reforço sólido que será essencial para o avanço nos estudos e aplicações práticas no futuro.

Esse relatório reflete o aprendizado que tive nesse Card, já havia utilizado as duas bibliotecas na faculdade, mas com essas aulas, possuo uma compreensão mais aprofundada nos conteúdos.