Relatório 10 - Lidando com Dados do Mundo Real

Lucas Scheffer Hundsdorfer

Descrição da atividade

Aula 1 - K-Nearest-Neighbors Concepts

Nessa aula ele começa falando sobre K-Nearest-Neighbors(KNN), na tradução literal fica ‘K vizinhos mais próximos’, basicamente uma métrica utilizada para classificar novos pontos de dados baseados nas distâncias de pontos já conhecidos.

Aula 2 - Using KNN to predict a rating for a movie

Nessa aula vamos definir uma métrica de distância entre filmes utilizando os metadados. Primeiramente ele cria um Data Frame baseado em filmes e utilizando colunas de ‘movie_id’ e rating, após isso ele transforma o Data Frame em apenas o índice de ‘movie_id’ e utiliza uma métrica de 0 a 1 para definir a popularidade do filme, com isso ele cria um dicionário que contém todos os filmes, suas categorias e a métrica de popularidade e a média da avaliação do filme. Usando o dicionário ele cria uma função que consegue retornar os K vizinhos próximos, basta passar o número de K e o filme desejado, ele retorna os filmes que tem mais similaridade com o filme passado.

```
[73]: import operator

def getNeighbors(movieID, K):
    distances = []
    for movie in movieDict:
        if (movie != movieID):
            dist = ComputeDistance(movieDict[movieID], movieDict[movie])
            distances.append((movie, dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(K):
        neighbors.append(distances[x][0])
    return neighbors

K = 25
avgRating = 0
neighbors = getNeighbors(1, K)
for neighbor in neighbors:
    avgRating += movieDict[neighbor][3]
    print (movieDict[neighbor][0] + " " + str(movieDict[neighbor][3]))

avgRating /= K
```

Essa é a função, esse é o retorno dela:

```
Liar Liar (1997) 3.156701030927835
Aladdin (1992) 3.8127853881278537
Willy Wonka and the Chocolate Factory (1971) 3.6319018404907975
Monty Python and the Holy Grail (1974) 4.0664556962025316
Full Monty, The (1997) 3.926984126984127
George of the Jungle (1997) 2.685185185185185
Beavis and Butt-head Do America (1996) 2.7884615384615383
Birdcage, The (1996) 3.4436860068259385
Home Alone (1990) 3.0875912408759123
Aladdin and the King of Thieves (1996) 2.8461538461538463
Lion King, The (1994) 3.7818181818181817
Jungle2Jungle (1997) 2.4393939393939394
Babe (1995) 3.9954337899543377
Wrong Trousers, The (1993) 4.466101694915254
Raising Arizona (1987) 3.875
Beauty and the Beast (1991) 3.792079207920792
Back to the Future (1985) 3.834285714285714
101 Dalmatians (1996) 2.908256880733945
Fish Called Wanda, A (1988) 3.785425101214575
Pinocchio (1940) 3.6732673267326734
Matilda (1996) 3.210526315789474
Mary Poppins (1964) 3.7247191011235956
In & Out (1997) 3.3043478260869565
Fantasia (1940) 3.7701149425287355
Snow White and the Seven Dwarfs (1937) 3.7093023255813953
```

Aula 3 - Dimensionality Reduction; Principal Component Analysis (PCA)

Primeiramente ele começa a falar da maldição das dimensionalidades ‘curse of dimensionality’, muitos problemas podem ser considerados como tendo um grande número de ‘dimensões’ ele usa como exemplo os filmes da aula passada, querendo dizer que a avaliação do filme pode representar uma dimensão, e cada filme tem sua própria dimensão fazendo com que fique cada vez mais complicado analisar a situação, por isso o ‘Dimensionality Reduction’ existe, para tentar reduzir a quantidade de ‘dimensões’ de um conjunto de dados enquanto ainda preserva o máximo da variação dos dados. Uma forma de reduzir as dimensões é utilizando o conceito do vídeo, ‘Principal Component Analysis’, é uma técnica que visa simplificar essa análise ao transformar essas ‘dimensões’ em um novo conjunto de ‘dimensões’ não correlacionadas. Tudo isso é feito na intenção de facilitar a visualização e a compreensão dos dados.

Aula 4 - PCA Example with the Iris data set

Nessa aula ele explica um pouco mais sobre o PCA e aplica isso.

```
[1]: from sklearn.datasets import load_iris
     from sklearn.decomposition import PCA
     import pylab as pl
     from itertools import cycle

     iris = load_iris()

     numSamples, numFeatures = iris.data.shape
     print(numSamples)
     print(numFeatures)
     print(list(iris.target_names))

     150
     4
     ['setosa', 'versicolor', 'virginica']
```

Aqui os dados em 4 dimensões:

```
[14]: pca.components_

[14]: array([[ 0.36138659, -0.08452251,  0.85667061,  0.3582892 ],
             [ 0.65658877,  0.73016143, -0.17337266, -0.07548102]])
```

Após isso ele transforma em 2 e mostra o quão preservado foi o conjunto de dados:

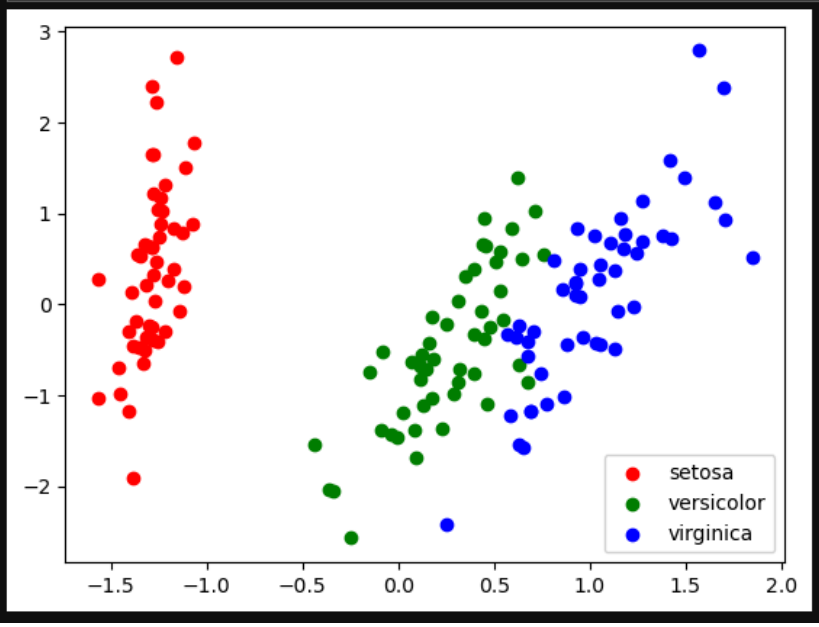
```
[16]: print(pca.explained_variance_ratio_)
      print(sum(pca.explained_variance_ratio_))

      [0.92461872 0.05306648]
      0.9776852063187924
```

E demonstra em um gráfico de dispersão como fica com os dados agora em 2D:

```
[19]: %matplotlib inline
      from pylab import *

      colors = cycle('rgb')
      target_ids = range(len(iris.target_names))
      pl.figure()
      for i, c, label in zip(target_ids, colors, iris.target_names):
          pl.scatter(X_pca[iris.target == i, 0], X_pca[iris.target == i, 1],
              c=c, label=label)
      pl.legend()
      pl.show()
```



Aula 5 - Data Warehousing Overview ETL and ELT

Data Warehousing é basicamente um banco de dados gigante que têm informações de diferentes fontes e as conecta para você, agora ETL, que significa extrair, transformar e carregar, é basicamente a forma em que você vai conectar esses dados dentro do Warehouse porém transformar esses dados em tabelas no SQL fica cada vez mais complexo principalmente trabalhando

com Big Data, aí que surgiu o ELT, funciona extraindo os dados porém agora você carrega eles no próprio sistema do Warehouse e usa o poder do Warehouse para poder transformar os dados.

Aula 6 - Reinforcement Learning

‘Reinforcement Learning’ na tradução literal é o aprendizado por reforço, uma das técnicas de machine learning, ele imita as ações humanas de tentativa e erro, funciona da seguinte forma, o agente de aprendizado interage com o ambiente e toma ações aprendendo com a experiência, ele é penalizado quando erra e recompensado quando acerta, com um certo tempo o agente aprende a tomar as decisões corretas para a situação, o exemplo do vídeo é o jogo do Pacman. Ele também fala sobre o Q-Learning, um algoritmo sem modelo baseado em valores de Q, onde Q significa qualidade, onde o agente aprende a tomar decisões baseadas em qual estado de Q tem o maior valor. Um dos problemas do aprendizado por reforço pode ser o ‘Exploration Problem’ que é a preocupação com a eficiência de explorar todos os estados possíveis, a melhor maneira de resolver é utilizando um termo epsilon que consiste em se um número aleatório for menor que epsilon não siga o maior ‘Q’, apenas faça um caminho aleatório, assim dessa forma a exploração nunca acaba

Aula 7 - Activity Reinforcement Learning & Q-Learning with Gym

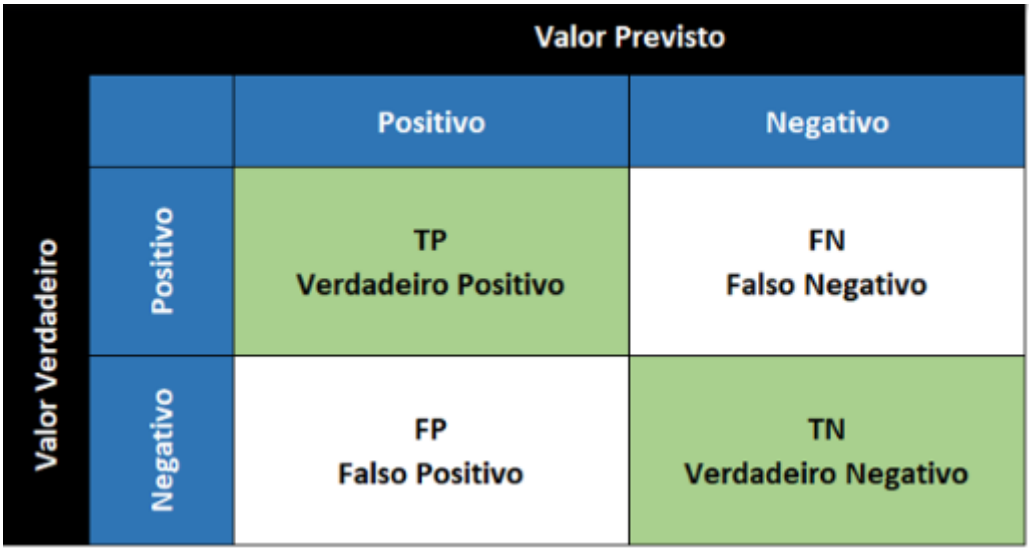
Nos é apresentado a biblioteca do python ‘Gym’ onde é possível simular ambientes de aprendizado por reforço (RF), e isso vai ser mostrado na prática com um ambiente chamado ‘Táxi-v3’, basicamente estamos modelando um táxi autônomo que pode pegar passageiros em um conjunto de locais fixos e deixá-los em outro local e tentar fazer tudo isso com o menor tempo e menor quantidade de obstáculos

R/G/B/Y são os locais onde os passageiros podem ser pegados ou deixados, agora a cor azul na letra significa que temos que pegar o passageiro e a cor magenta significa aonde temos que deixá-los.

Dentro dessa aula nos é mostrado como que é feito para simular um aprendizado por reforço, e após é mostrado o resultado, do táxi indo e pegando o passageiro e deixando ele no lugar esperado, sempre evitando a menor quantidade de movimentos e desviando dos obstáculos.

Aula 8 - Understanding a Confusion Matrix

Matriz de confusão é uma tabela usada para mostrar as previsões de algo com os resultados reais, pode ser usada tanto para um teste de doença ou para a visão computacional onde ele tem que dizer se tem algo ali ou não, um exemplo de matriz deixa as coisas mais claras:



Supondo que existe um teste de drogas e devemos usar a matriz de confusão para poder saber o quão preciso ele está, a matriz ajuda nisso, visando sempre que o melhor resultado é o da diagonal principal onde ele acerta se é verdadeiro ou falso.

Aula 9 - Measuring Classifiers (Precision, Recall, F1, ROC, AUC)

Recall é uma conta simples, uma divisão do verdadeiro positivo, pela soma do falso negativo com o verdadeiro positivo, também chamado de taxa positiva verdadeira uma forma de chegar na porcentagem de positivos previstos corretamente, é utilizada quando se importa muito com a métrica de falsos negativos.

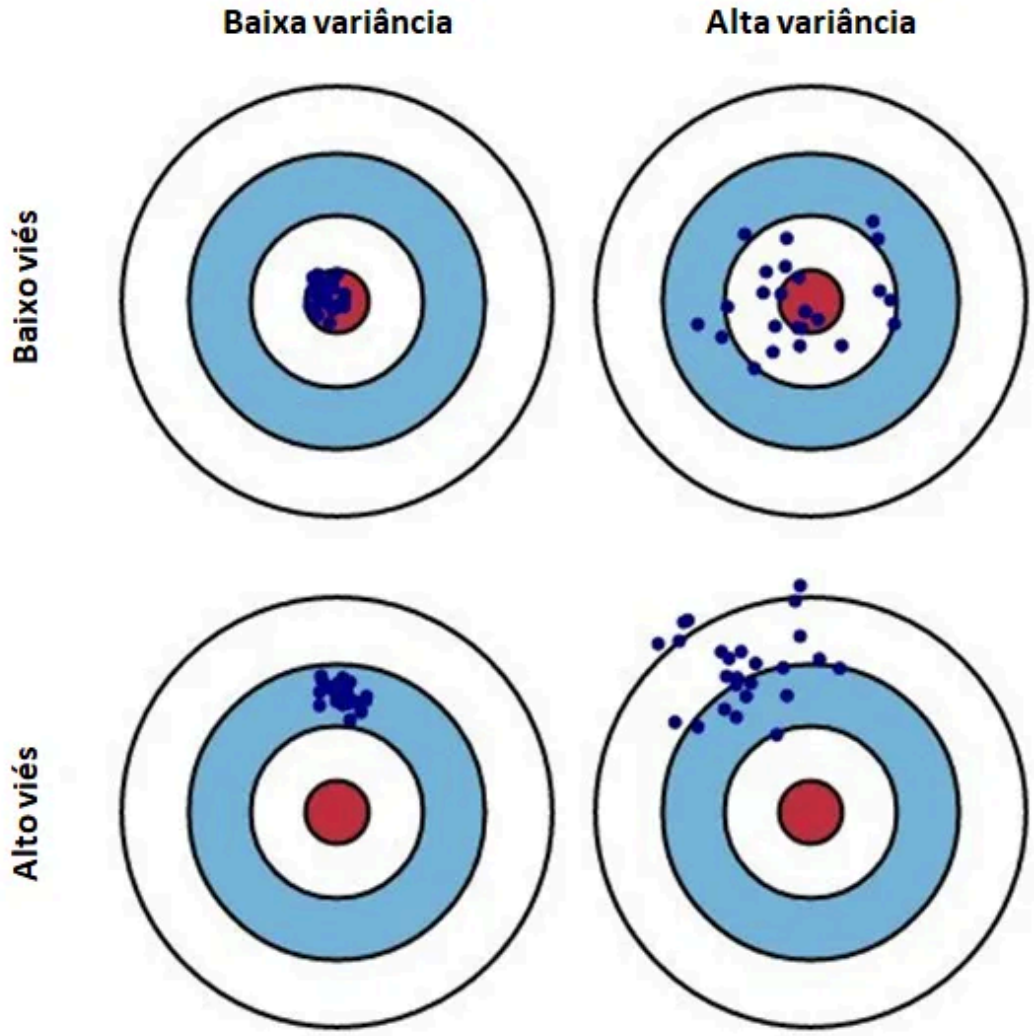
Precision é uma métrica da divisão do verdadeiro positivo pela soma do falso positivo com o verdadeiro positivo, mostra a porcentagem dos resultados relevantes e é útil quando se importa com os falsos positivos.

F1 é uma forma de calcular fazendo a divisão da multiplicação da Precision e do Recall pela soma dos dois, utilizada quando você se importa com os dois parâmetros e é a média harmônica entre eles.

Seção 7 - Dealing with Real-World Data

Aula 1 - Bias Variance TradeOff

Bias Variance ou viés e variância, viés é o quão longe a média da sua previsão está longe dos resultados reais, e a variância quão dispersos estão seus valores previstos da resposta real, exemplo:

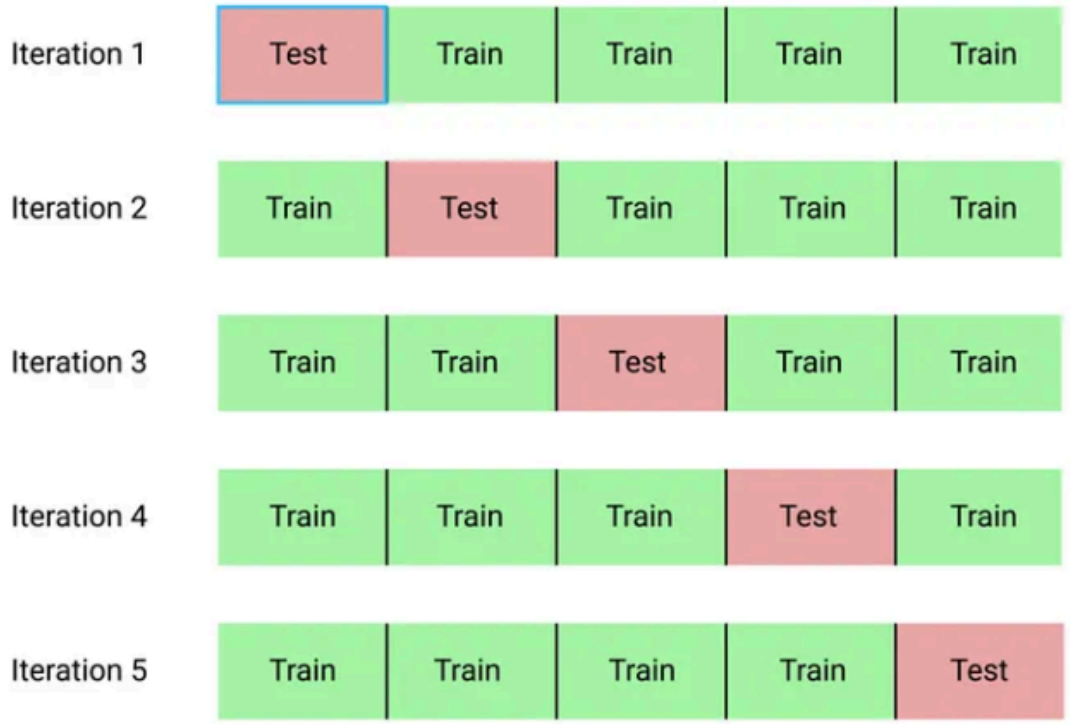


Mas com que nos importamos é o erro, e a taxa de erro pode ser medida pela soma de viés ao quadrado com a variância, porém é o erro que queremos minimizar não a taxa de variância ou o viés.

Aula 2 - K-Fold Cross-Validation to avoid overfitting

Nos é apresentado um dos problemas do machine learning, o overfitting, é um problema que ocorre quando um modelo é treinado muito de perto para um conjunto de dados, fazendo com que ele tenha um desempenho ruim em novos dados. Uma das maneiras de evitar esse problema é o K-fold cross-validation é uma técnica de validação onde os dados são divididos em K partes iguais. Em cada iteração, uma parte é usada para teste, enquanto as outras K-1 partes são usadas para treino. O processo é repetido K vezes, garantindo que cada parte seja usada como teste uma vez. No final, uma média das métricas de avaliação é calculada para medir o desempenho geral do modelo, exemplo dos

conjuntos.



Aula 3 - Data Cleaning and Normalization

A aula começa com ele dizendo que a maior parte do trabalho de um cientista de dados é limpar a entrada de dados, que é quase tão importante que o modelo que você vai usar, na entrada de dados pode aparecer tanto como dados discrepantes, dados faltando, dados maliciosos, dados irrelevantes, dados inconsistentes entre outros que podem acabar prejudicando o seu modelo e a forma que ele toma decisões.

Aula 4 - Cleaning web log data

Agora ele vai mostrar a prática da limpeza dos dados, é trazido um arquivo sobre os acesso de um site dele, e é mostrado alguns scripts filtrando e limpando os dados.

```
URLCounts = {}

with open(logPath, "r") as f:
    for line in (l.rstrip() for l in f):
        match= format_pat.match(line)
        if match:
            access = match.groupdict()
            request = access['request']
            fields = request.split()
            if (len(fields) == 3):
                URL = fields[1]
                if URL in URLCounts:
                    URLCounts[URL] = URLCounts[URL] + 1
                else:
                    URLCounts[URL] = 1

results = sorted(URLCounts, key=lambda i: int(URLCounts[i]), reverse=True)

for result in results[:20]:
    print(result + ": " + str(URLCounts[result]))
```

Ao longo da aula ele vai filtrando da maneira que mais lhe faz sentido e ele chega nesse código por último, tentando mostrar qual das notícias do site é a mais acessada.

```
URLCounts = {}

with open(logPath, "r") as f:
    for line in (l.rstrip() for l in f):
        match= format_pat.match(line)
        if match:
            access = match.groupdict()
            agent = access['user_agent']
            if (not('bot' in agent or 'spider' in agent or
                    'Bot' in agent or 'Spider' in agent or
                    'W3 Total Cache' in agent or agent =='-')):
                request = access['request']
                fields = request.split()
                if (len(fields) == 3):
                    (action, URL, protocol) = fields
                    if (URL.endswith("/")):
                        if (action == 'GET'):
                            if URL in URLCounts:
                                URLCounts[URL] = URLCounts[URL] + 1
                            else:
                                URLCounts[URL] = 1

results = sorted(URLCounts, key=lambda i: int(URLCounts[i]), reverse=True)

for result in results[:20]:
    print(result + ": " + str(URLCounts[result]))
```

Aula 5 - Normalizing numerical data

A importância de normalizar dados numéricos vêm de que se um modelo é baseado em vários atributos numéricos e eles não forem comparáveis, um dos exemplos passados é a análise de renda por idade, onde idade pode variar de 0 - 100, e renda de 0 - 1.000.000.000, e dependendo do modelo ele pode não trabalhar muito bem com isso. No final ele diz para ler a documentação pois lá deve estar escrito se precisa ou não dessa normalização.

Aula 6 - Detecting outliers

Os valores discrepantes às vezes vale a pena retirar algumas vezes não, ele deixa claro que tem que saber quando retirar esses valores discrepantes, os exemplos usados nos vídeos foram como, alguém num sistema de avaliação de filmes que avaliou todos os filmes existentes poderia afetar a métrica do sistema, porém não faria sentido você estar coletando a média de renda dos EUA e retirar os bilionários só por que é um valor discrepante com os demais.

É citado que o desvio padrão consegue ajudar a identificar os valores discrepantes, porém é destacado que não há uma maneira exata que é de bom tom olhar o histograma, os dados em si para se ter uma melhor certeza se há um valor discrepante.

```
def reject_outliers(data):
    u = np.median(data)
    s = np.std(data)
    filtered = [e for e in data if (u - 2 * s < e < u + 2 * s)]
    return filtered

filtered = reject_outliers(incomes)

plt.hist(filtered, 50)
plt.show()
```

Na aula é feito um código para retirar os outliers que é baseado em retirar todos os dados que são 2 vezes maior ou menor que o desvio padrão.

Aula 7 - Feature Engineering and the Curse of Dimensionality

A engenharia de recursos é aplicar o conhecimento em cima dos dados para criar melhores recursos para o seu modelo treinar, por exemplo o que fazer com dados faltando, quais recursos devo usar, eu deveria pegar vários recursos e combiná-los matematicamente para reduzir as dimensões. É basicamente isso, aqui é onde a expertise do machine learning ocorre, não é simplesmente jogar todos os dados e esperar resultados.

A engenharia de recursos é necessária para evitar um dos problemas que é a maldição das dimensionalidades, que consiste em que a cada recurso é uma nova dimensão, muitos recursos levam a dados esparsos o que pode dificultar cada vez mais para achar uma solução.

Aula 8 - Imputation Technique for Missing Data

Uma das maneiras mais fáceis de lidar com a falta de dados é substituindo eles pela média da coluna(considerando que a coluna seja a representação de um recurso), é simples e rápido é possível também trazer a mediana se estiver tratando com outliers. É rápido mas não é tão útil, já que funciona só com colunas, não é muito assertivo.

Outro método é o dropping, consiste simplesmente em eliminar os dados faltando, é fácil e rápido, porém tem que ter uma noção se a eliminação desses dados não vai afetar o conjunto de dados de alguma maneira. É uma opção mais rápida mas provavelmente a pior.

Outra maneira é utilizando o próprio Machine Learning, a primeira forma é usando o KNN onde K é um valor especificado e encontra os valores mais ‘próximos’ da linha e faz a média desses valores, porém só funciona com dados numéricos. Para dados categóricos o mais recomendado é utilizar do deep learning basicamente criar um modelo de machine learning para imputar dados para o seu modelo de machine learning.

E o último método apresentado para repor dados ausentes é conseguindo mais dados.

Aula 9 - Handling Unbalanced Data Oversampling, Undersampling, and SMOTE

Outro problema da Engenharia de recursos é os dados desbalanceados que é basicamente a discrepância entre positivos e falsos, o exemplo dado é o caso de um modelo treinado para descobrir fraudes, porém fraudes são muito raros e a maior parte dos dados vão ser não fraudulentos, o que pode acontecer com isso é que como no conjunto de dados que foi usado para treinar o modelo só tinha 0,01 de fraudes ele pode começar a entender que apenas 0,01 de todos os casos vão ser fraudes. Isso é o poder de dados desbalanceados podem fazer, a solução é o Oversampling na tradução sobre amostragem, que consiste em pegar o caso minoritário e multiplicá-lo para poder balancear o

conjunto de dados. Outra maneira é o Undersampling, sub amostragem que consiste em retirar alguns dos casos majoritários para poder equilibrar, porém só faz sentido você jogar dados fora caso você esteja evitando big data.

O melhor exemplo para balancear os dados é o SMOTE, Synthetic Minority Over-Sampling Technique ou técnica de sobreamostragem minoritária sintética que consiste em criar artificialmente novos casos minoritários baseados no KNN de cada caso minoritário já existente e criando novos exemplos através da média do resultado.

Aula 10 - Binning, Transforming, Encoding, Scaling, and Shuffling

Nos é apresentado mais algumas técnicas que podemos usar durante o processo de engenharia de recursos.

Binning é transformar dados numéricos em dados categóricos, exemplo, invés de utilizar exatamente a idade podemos agrupar por décadas, como 20 e poucos anos, 30 e poucos anos, a transformação você está perdendo dados, então só é recomendada quando não se tem certeza das informações.

Transforming é basicamente aplicar uma função para que o recurso seja melhor para o treinamento.

Encoding é transformar os dados numa representação que o modelo exige.

Scaling é o processo de transformar características numéricas em um conjunto de dados em uma escala ou intervalo comum, garantindo que todas as características contribuem igualmente para o modelo.

Shuffling é o processo de reorganização aleatória da ordem dos pontos de dados dentro de um conjunto de dados.

Conclusões

Durante o curso inteiro ele fala que ciência de dados está atribuída a nomes complicados e intimidadores porém a definição disso não é complexa, e realmente não é complexo é apenas desafiador entender algumas das técnicas do machine learning.

Referências

Análise de Componentes Principais (PCA) Explicada!

O que é bias-variance tradeoff e como isso impacta seu modelo | Data Hackers

Cross Validation: Avaliando seu modelo de Machine Learning | by Eduardo Braz Rabello | Medium