**Algoritmos de Machine Learning - Aplicações Práticas**

**Dionisio Gause Junior**

*Centro Universitário Leonardo da Vinci – UNIASSELVI*

*dionisio@informatyk.com.br; dgausejr@gmail.com*

O Aprendizado de Máquina ou “*Machine Learning”,* têm se destacado nas mais variadas áreas, ou seja, treinar o computador para poder responder rapidamente a uma questão, com base em dados fornecidos, comparando-os, e desta forma também “aprendendo” conforme mais dados são acrescentados a base de dados, utilizando-se de Métodos Supervisionados.

Desta forma, escolhemos dois métodos para demonstrar o quanto a área de Inteligência Artificial está se tornando cada vez mais essencial ao nosso dia a dia.

Para configurar o ambiente de desenvolvimento utilizou-se a versão do *Python* 3.7.2 e a IDE *JetBrains PyCharm Community Edition 2018.2.3 x64 e* as bibliotecas: *Numpy*, *Pandas, Matplotlib, Seaborn e* *Sklearn*.

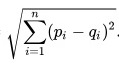
Algumas dicas importantes são mencionadas por diversos autores quanto a padronização, ou seja, “*...utilize a regra de convenção. Isso faz com que pessoas que lerem seu código no futuro — incluindo você mesmo — possa identificar a biblioteca mais facilmente.*”[[1]](#footnote-1)

A proposição é a de criar no Python, algoritmos de *Machine Learning ML*, utilizando o método Supervisionado de Classificação - KNN (*K Neareast Neighbor*) ou K Vizinhos mais Próximos e o Método de Regressão Linear - método Supervisionado de Predição muito utilizado quando do uso de duas variáveis.

Para tal intento abordamos um pouco dos conceitos utilizados para estes desenvolvimentos iniciando pelo k-NN e após a Regressão Linear.

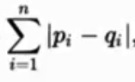
“In the absence of prior knowledge, most kNN classifiers use simple Euclidean distances to measure the dissimilarities between examples represented as vector inputs.”[[2]](#footnote-2)

“Na ausência de conhecimento prévio, a maioria dos classificadores de kNN usa distâncias euclidianas simples para medir as diferenças entre exemplos representados como entradas de vetores.”



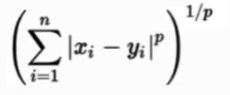
*Fig1. Fórmula Euclidiana[[3]](#footnote-3)*

Outra métrica para a classificação é a de Manhattan.



*Fig2. Fórmula de Manhattan[[4]](#footnote-4)*

A apresentação destas duas métricas são importantes, pois, são delas que a forma generalista de Minkowski se baseia, ou seja, ao informarmos o valor de *p = 2* utilizaremos a forma Euclidiana e/ou ao informar o valor de *p = 1* utilizamos a métrica de Manhattan.



*Fig3. Fórmula Minkowski[[5]](#footnote-5)*

Esta abordagem se faz necessária, pois ao desenvolvermos o algoritmo proposto, utilizamos pacotes ou *Packages* necessários para a codificação na linguagem Python, onde as funções utilizadas nos ajudarão a chegar às soluções propostas.

Um problema comum em várias áreas do conhecimento é a análise da relação linear entre duas variáveis.

E no método da regressão linear, o método tenta definir a dependência entre as variáveis, ou seja, quanto mais dados, melhor os modelos serão.

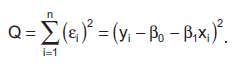
Cabe destacar um ponto chamado Correlação, que é a forma com as variáveis x e y se relacionam, podendo ser positiva, quando as variáveis são diretamente proporcionais, ou seja, quando uma aumenta a outra também aumenta. Pode ser negativa, quando as variáveis são inversamente proporcionais ou o aumento de uma acarreta na diminuição de outra ou ainda quando não há correlação, sendo os dados muito dispersos para utilizar o método de regressão linear, optando desta forma por outra forma de classificação.

Em outras palavras o objetivo em uma regressão sempre será prever o resultado de variável contínua.



*Fig.4 Fórmula Regressão Linear Simples*

O Estimação por Regressão Linear por Mínimos Quadrados Ordinários(MQO) é um método de análise estatística que permite estimar o valor de uma determinada variável resposta (variável dependente) como função de outras variáveis preditoras (variáveis independentes).



*Fig. 5 Fórmula MQO*

A utilização dos dois modelos está descrito nos códigos Python nos Anexo1 e Anexo2 deste artigo.

Podemos perceber ainda que a principal diferença está na utilização dos dados disponíveis e da quantidade de variáveis (atributos) a serem utilizados.

Desse modo observamos que a utilização de um ou outro método está ligado aos dados disponíveis, ou seja, a utilização de um dos métodos será determinada pela observação por parte do pesquisador ou analista, qual método é o apropriado a ser utilizado.

**Pré-Processamento**

Uma das fases no aprendizado de máquina ou (*Machine Learning*) é o Pré-Processamento dos dados, que consomem um tempo extraordinário na Mineração de Dados e pode ser entendida como uma das fases críticas utilizadas para a Descoberta de Conhecimento em Banco de Dados e “engloba uma análise inicial dos dados para se ter sólidas definições dos mesmos.”[[6]](#footnote-6)

Neves afirma que “a fase de Pré-processamento é composta pelas seguintes subfases: entendimento, seleção, limpeza e transformação de dados.”[[7]](#footnote-7)

“*A fase de entendimento do Domínio do Problema envolve entrevistas com os especialistas do domínio do problema (usuário) a fim de identificar os objetivos do mesmo, verificação de qualquer conhecimento prévio que possa contribuir ao alcance dos objetivos.*”[[8]](#footnote-8)

A autora ressalta que “*não existe uma sequência obrigatória quanto a ocorrência das subfases de pré-processamento, pois dependendo da situação pode-se por exemplo, preferir realizar a limpeza antes de um determinado tipo de seleção.*”[[9]](#footnote-9)

“*A limpeza de dados refere-se a garantia da qualidade dos dados que pode ser obtida através de algumas operações tais como: padronização de dados, tratamento de valores ausentes, eliminação de dados errôneos e de duplicatas.*

*Quanto a Transformação de dados esta corresponde a operações que tornem a apresentação de dados apropriada a técnica de mineração de dados a ser utilizada, assim encontram-se descritas operações do tipo normalização de dados, conversões de valores simbólicos para valores numéricos, discretização e composição de atributos.*”[[10]](#footnote-10)

*“Antes de ser iniciada qualquer operação de seleção, limpeza e transformação de dados é de fundamental importância que se tenha o conhecimento ou entendimento dos dados, pois isto orienta as tomadas de decisões do que se investigar e o que será necessário ser realizado em termos de preparação dos dados para a mineração.”*[[11]](#footnote-11)

O Professor Alexandre Chiavegatto da Faculdade de Saúde pública de SP aponta o Pré-Processamento como a “chave para a boa performance preditiva” e enumera as técnicas utilizadas como:

A *Seleção de Variáveis* é importante para incluir as variáveis plausíveis a serem adotadas nos modelos. O cuidado com o *Vazamento de Dados* deve-se ao fato de evitar que os modelos aprendam padrões desinteressantes. Através da *Padronização* alguns atributos são tratados para ter uma mesma escala de valores a serem analisados e utilizados. Na *Redução de Dimensão* observa-se que quanto maior a dimensão dos dados, maior o risco de sobreajuste do modelo. Observando a *Colinearidade* a ser tratada, pois variáveis colineares podem trazer informação redundante, aumentando a instabilidade do modelo. A técnica de *Valores Missing* trata dos atributos ausentes ou faltantes refletindo no modelo. A técnica *one-hot encoding* é utilizada para o ajuste de variáveis que apresentam categorias diferentes, transformando cada categoria em uma variável diferente de valor 0 e 1, evitando desse modo trazer problemas em alguns modelos como na regressão linear.

A Dissertação de Neves (2013) nos apresenta os Métodos de Seleção de Atributos do tipo Filtro, W*rapper* e Híbrida com abordagens supervisionadas e não-supervisionadas, os Métodos de Seleção de Instâncias do tipo Filtro e Wrapper e os Métodos de Discretização por Divisão e Combinação.

**Métricas de avaliação de Modelos**

Conforme Rodrigo Santana(2017) publicou em seu artigo, *“as métricas de avaliação nos mostram se o modelo está com bons resultados e como podemos melhorar.” [[12]](#footnote-12)*

*“As métricas Precision (precisão), recall (revocação) e F-1(medida F) temos uma visão clara dos resultados, mas antes precisamos saber sobre a terminologia para classificação, onde:*

*True Positive(TP): classificação correta da classe Positive.*

*True Negative(TN) classificação correta da classe Negative.*

*False Positive(FP) Classificação errada da classe Positive.*

*False Negative(FN) Classificação errada da classe Negative.*

*Onde a Precision(precisão) é calculada da seguinte forma:*

*Precision = TP / (TP + FP)*

*Significa que o número de vezes que uma classe foi predita corretamente dividida pelo número de vezes que a classe foi predita.*

*Onde Recall(revocação) é calculada da seguinte forma:*

*Recall = TP / (TP + FN)*

*Significa o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).*

*A F1-Score é calculada da seguinte forma:*

*F1 = 2\*((Precision \* Recall) / (Precision + Recall))*

*Essa medida é a média harmônica entre precisão e revocação.*

*Para calcular a acurácia, use a fórmula:*

*Accuracy = (TP + TN) / (TP + FP + TN + FN)*

*A acurácia nos mostra como o classificador se saiu de uma maneira geral, pois, esta mede a quantidade de acertos sobre o todo.”*[[13]](#footnote-13)

Sobre a acurácia Neves (2013) define que *“constitui um fator importante na avaliação do sucesso da mineração de dados. Quando aplicada a dados, a acurácia refere-se a taxa de valores corretos no dado. Quando aplicada a modelos, a acurácia consiste no grau de ajuste entre o modelo e o dado, ela verifica então quão as predições dos modelos estão livres de erros.”*[[14]](#footnote-14)

***Referências:***

*<http://papers.nips.cc/paper/2795-distance-metric-learning-for-large-margin-nearest-neighbor-classification.pdf acessado em 12/09/2018 18:42h>*

*<https://paulovasconcellos.com.br/como-criar-seu-primeiro-aplicativo-de-machine-learning-7b6af291ba11 - acessado em 12/09/2018 21:35h>*

*<https://www.youtube.com/watch?v=zvmbB3315Ko acessado em 19/09/2018 10:00h>*

*<https://www.youtube.com/watch?v=cCMi1fjfAUY - 12 passos para a classificação do K vizinhos mais próximos em Python - acessado em 20/10/2018 10:00h>*

*<https://www.youtube.com/watch?v=pTf4Vmwp6Vw - 3 super técnicas em data Science - acessado em 22/10/2018 - 12:00>*

*<https://www.youtube.com/watch?v=5Nmzd9slQk0 - correlação e regressão linear simples - acessado em 22/10/2018 - 14:00>*

*Estatística Aplicada – Luiz Aparecido Milan – 2014 – 161 páginas – disponível em<http://livresaber.sead.ufscar.br:8080/jspui/bitstream/123456789/2696/1/EA\_Milan\_EstatisticaAplicada.pdf>*

*<http://neylsoncrepalde.github.io/2018-02-25-regressao-linear-python/ - acessado em 09/10/2018 – 10:06h>*

*<https://www.youtube.com/watch?v=MgtIdBrf0v8 - acessado em 09/10/2018 - 09:51h>*

*<https://www.youtube.com/watch?v=jh4m0WN-n48 – acessado em 09/10/2018 – 09:58h>*

*<https://www.youtube.com/watch?v=H5Efih9wEfo - acessado em 20/10/2018 - 11:33h>*

*<https://www.youtube.com/watch?v=AD4sfopB-Uk - acessado em 04/11/2018 - 16:00 - Alexandre Chiavegatto Faculdade Saúde Publica SP.>*

*<http://minerandodados.com.br/index.php/2017/10/10/cafe-com-codigo-09-metricas-de-avaliacao-de-modelos/ - acessado em:04/11/2018 - 20:00>*

*<Neves, Rita de Cássia David das – Dissertação Mestrado – 2003 – 13p. - Pré-Processamento no Processo de Descoberta de Conhecimento em Banco de Dados – Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/2701/000375412.pdf - acessado em 30/10/2018>*

***Códigos e arquivos disponíveis em:***

*https://github.com/dgausejr/BigData/blob/master/knn\_iris.py - commited in 22/10/2018*

*https://github.com/dgausejr/BigData/blob/master/iris.csv - commited in 22/10/2018*

*https://github.com/dgausejr/BigData/blob/master/reg\_linear.py - commited in 22/10/2018*

***Anexo 1***

***Código em Python – KNN Iris – arquivo: knn\_iris.py***

*# Projeto Big Data - Uniasselvi - outubro / 2018  
# Dionisio Gause Junior  
# Missão: Implementar o metodo kNN para a flor IRIS  
# Desenvolimento: Baseado na aula do professor Sanderson Macedo  
# Disponível em: https://www.youtube.com/watch?v=cCMi1fjfAUY*

*#Importando bibliotecas necessárias  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sb  
  
# carrega dados em dataframe df  
df = pd.read\_csv('iris.csv')  
# usado para identificar o nome das colunas  
print(df.columns)  
# usado para visualizar os dados  
print(df)  
# print usado para descrição dos dados  
print(df.describe())  
# Usado para visualizar a dispersão dos dados  
sb.pairplot(df, hue='Species')  
plt.show()  
# selecionando as caracteristicas para a classificação em um array  
X = np.array(df.drop('Species', 1))  
# visualizar os dados do array criado (X)  
print(X)  
# selecionando a classes para classificação em array  
y = np.array(df.Species)  
# visualizar os dados do array criado(y)  
print(y)  
# importando a funcao KNeighborsClassifier da biblioteca sklearn.neighbors  
from sklearn.neighbors import KNeighborsClassifier  
# informando o n\_neighbors a verificar - quanto maior, maior a probabilidade de acerto  
# sempre em valores impares (evitando empate na classificação)  
knn = KNeighborsClassifier(n\_neighbors=5)  
# Treinando um classificador com base nas caracteristicas (X) e classe (y)  
knn.fit(X, y)  
comp\_sepala = float(input('Digite o comprimento da Sépala: '))  
larg\_sepala = float(input('Digite a largura da Sépala....: '))  
comp\_petala = float(input('Digite o comprimento da Pétala: '))  
larg\_petala = float(input('Digite a largura da Pétala....: '))  
p = knn.predict([[comp\_sepala, larg\_sepala, comp\_petala, larg\_petala]])  
print(f'Com base nos dados informados e utilizando o método K-NN, é provável que seja uma Iris {p}\n'  
 f'Obrigado por utilizar nosso APP, volte sempre !')*

***Anexo 2***

***Código em Python - Regressão linear – arquivo: Reg\_linear.py***

*# Projeto Big Data - Uniasselvi - outubro / 2018  
# Dionisio Gause Junior  
# Desafio: Implementar o metodo de regressão linear simples  
# Referências:  
# https://www.youtube.com/watch?v=gzyq\_6wdtSk  
# https://medium.com/@felipebormann/aprendendo-scikit-learn-e-um-pouco-mais-de-python-6b27025f9d5b  
# http://neylsoncrepalde.github.io/2018-02-25-regressao-linear-python/  
# https://imasters.com.br/back-end/data-science-regressoes-com-python*

*# Importando bibliotecas necessárias*

*import numpy as np  
import matplotlib.pyplot as plt  
# Criando os arrays com as variáveis x e y a serem utilizados  
km\_percorridos = [8, 10, 12,16,20]  
x = np.array(km\_percorridos)  
calorias\_queimadas = [90, 112, 148, 189, 212]  
y = np.array(calorias\_queimadas)  
# chama a função polyfit informando os vetores e o grau do polinomio  
# onde a regressão linear o grau é 1 - equação da reta  
p1 = np.polyfit(x, y, 1)  
# a partir dos coeficientes linear e angular calcula-se o y predito  
yfit = p1[0] \* x + p1[1]  
# Calcula-se o resíduo (valor de y menos o predito)  
yresid = y - yfit  
# Calcula-se o somatório dos quadrados dos residuos  
SQresid = sum(pow(yresid,2))  
# Calcula-se o somatório da diferença entre o valor real e a média dos valores de y  
# quantidade de elementos de y vezes a variança  
SQtotal = len(y) \* np.var(y)  
# com base nos valores SQresid e SQtotal calcula-se o coeficiente de determinação  
R2 = 1 - SQresid/SQtotal  
# mostra o intercepto e inclinação  
print(f'\nCoeficiente angular = {p1[0]}\n')  
print(f'Coeficiente linear = {p1[1]}\n')  
# mostra o coeficiente de determinação  
print(f'Coeficiente de Determinação = {R2}')  
# determina-se as variáveis para apresentção do gráfico (plt)  
plt.title('Quantidade de km percorridos por calorias queimadas') #título do gráfico  
plt.xlabel('“Km percorridos') #Legenda na horizontal  
plt.ylabel('Calorias Queimadas') #Legenda do eixo Y  
plt.plot(x, y, 'o') # os dados nos dois eixos  
# valores preditos de y (linha)  
plt.plot(x, np.polyval(p1,x),'g--')  
plt.grid(True) # Grid = grades, zonas do gráfico  
plt.show() # mostra o gráfico*

1. *<https://paulovasconcellos.com.br/28-comandos-%C3%BAteis-de-pandas-que-talvez-voc%C3%AA-n%C3%A3o-conhe%C3%A7a-6ab64beefa93 – acessado em 20/09/2018 19:19h>* [↑](#footnote-ref-1)
2. *<http://papers.nips.cc/paper/2795-distance-metric-learning-for-large-margin-nearest-neighbor-classification.pdf>* [↑](#footnote-ref-2)
3. *<https://www.youtube.com/watch?v=zvmbB3315Ko acessado em 19/09/2018 10:00h>* [↑](#footnote-ref-3)
4. *<https://www.youtube.com/watch?v=zvmbB3315Ko acessado em 19/09/2018 10:00h>* [↑](#footnote-ref-4)
5. *<https://www.youtube.com/watch?v=zvmbB3315Ko acessado em 19/09/2018 10:00h>* [↑](#footnote-ref-5)
6. *Neves, Rita de Cássia David das - 2013, pg.30* [↑](#footnote-ref-6)
7. *Neves, Rita de Cássia David das - 2013, pg.30* [↑](#footnote-ref-7)
8. *Neves, Rita de Cássia David das - 2013, pg.30* [↑](#footnote-ref-8)
9. *Neves, Rita de Cássia David das - 2013, pg.31* [↑](#footnote-ref-9)
10. *Neves, Rita de Cássia David das - 2013, pg.32* [↑](#footnote-ref-10)
11. *Neves, Rita de Cássia David das - 2013, pg.32* [↑](#footnote-ref-11)
12. *http://minerandodados.com.br/index.php/2017/10/10/cafe-com-codigo-09-metricas-de-avaliacao-de-modelos/* [↑](#footnote-ref-12)
13. *http://minerandodados.com.br/index.php/2017/10/10/cafe-com-codigo-09-metricas-de-avaliacao-de-modelos/* [↑](#footnote-ref-13)
14. *Neves, Rita de Cássia David das - 2013, pg.129* [↑](#footnote-ref-14)