UNIVERSIDADE FEDERAL DO CEARÁ

TECHNICAL REPORT

Aluno: Tauã Lima e Vitor Castro

1. Introdução

O dataset Stellar Classification Dataset tem como uma das maiores pesquisas astronômicas e em seu principal conceito ele apresenta informações importantes sobres as pesquisas espaciais, mais especificamente estrelas, galáxias e quasares cada qual em sua classe específica. Então seu objetivo é identificar objetos e construir modelos de classificação nas características físicas observadas .

Suas principais características são as coordenadas dos objetos no céu, magnitude fotométrica, deslocamento que fornece uma ideia de distância e velocidade relativa do objeto e a classe alvo definindo o tipo de objeto estelar.

Com base na característica apresentada ele tenta prever a classe do objeto.

procura identificar os atributos que terão maior impacto na classificação.

explora também o redshift ou as magnitudes fotométricas em classes diversas.

Já o dataset Ferrari and Tesla Share Prices (2015-2023) apresenta o histórico de preços das ações entre duas marcas muito famosas como Ferrari e Tesla. Dentro dele é apresentado preços de abertura, fechamento, volume de negociação e baixa e alta. Seu intuito é transparecer modelos de previsão de preços futuros do mercado e juntamente analisar também, possíveis tendências de mercado.

suas principais características são a data de observação, preço de abertura, maior preço já registrado, menor preço registrado, preço de fechamento, número de ações negociadas e diferença de dados entre as duas empresas.

Assim, busca apresentar a previsão de preços futuros de ações com base em dados, estudar como os preços das duas empresas evoluíram com o passar dos anos, identificar o desempenho de qual empresa segundo sua valorização ou estabilidade e verificar a relação entre volume de negociações e flexibilidade de preços.

2. Observações

Uma observação é que a prova em si foi bastante complicada, não só apenas para minha dupla e eu, porém mesmo tentando e tentando, ainda não foi possível uma boa compreensão dos códigos, mas no fim conseguimos resolver, mas alguns pontos não conseguimos entender e desenvolver .

3. Resultados e discussão

Nesta seção deve-se descrever como foram as resoluções de cada questão. Crie sessões indicando a questão e discuta a implementação e resultados obtidos nesta. Explique o fluxograma do processo de cada questão, indicando quais processamentos são realizados nos dados. Sempre que possível, faça gráficos, mostre imagens, diagramas de blocos para que sua solução seja a mais completa possível. Discuta sempre sobre os números obtidos em busca de motivos de erros e acerto.

1 questão

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv(r"C:\Users\vitor\Downloads\IA.BLACK\IA--SI\AV1\star_classification.csv")

print("Valores faltantes por coluna:")
print(df.isnull().sum())

df_cleaned = df.dropna()

print("Valores faltantes após limpeza:")
print(df_cleaned.isnull().sum())

colunas_relevantes = ['obj_ID', 'alpha', 'delta', 'u', 'g', 'r', 'i', 'z',
                      'run_ID', 'rerun_ID', 'cam_col', 'field_ID', 'spec_obj_ID',
                      'class', 'redshift', 'plate', 'MJD', 'fiber_ID']

colunas_existentes = [col for col in colunas_relevantes if col in df_cleaned.columns]
df_final = df_cleaned[colunas_existentes]

print("DataFrame final:")
print(df_final.head())

print("Distribuição das classes:")
print(df_final['class'].value_counts())

plt.figure(figsize=(8, 6))
sns.countplot(x='class', data=df_final)
plt.title('Distribuição das Classes')
```

Essa questão necessita de quatro bibliotecas diferentes para que possamos resolvê-la

sendo essas Pandas, Numpy, Matplotlib e Seaborn.

Importar essa bibliotecas é necessário para que ocorra a manipulação dos dados e visualização dos mesmos e também ainda garante que o ambiente está pronto para que o criador do código no caso, eu, possa chegar ao devido comando de processamento do dataset.

A primeira etapa foi o carregamento do dataset pelo seguinte caminho: ***pd.read_csv()***

A segunda etapa foi a identificação dos valores que faltavam sendo usado o comando ***df.isnull().sum()*** para essa identificação citada anteriormente.

Então foi necessário remover algumas linhas com o seguinte comando ***df.dropna()***.

Na terceira etapa foi definida as colunas relevantes e feito a filtragem, sendo as relevantes aquelas que poderiam ser úteis para a análise da classificação e a filtragem apenas para as colunas relevantes existentes no dataset oficial resultando em novo dataframe ***df_final***.

Na quarta etapa foram feitas a inspeção de distribuição de classes usando o comando ***value_counts()*** verificando a quantidade de registros em suas determinadas classes e por último a visualização criando um gráfico de barras usando o comando ***seaborn.countplot*** para visualizar a respectiva distribuição resultando em classes balanceadas mesmo com algumas tendo um pouco mais de registros que outras e um gráfico de distribuição das classes.

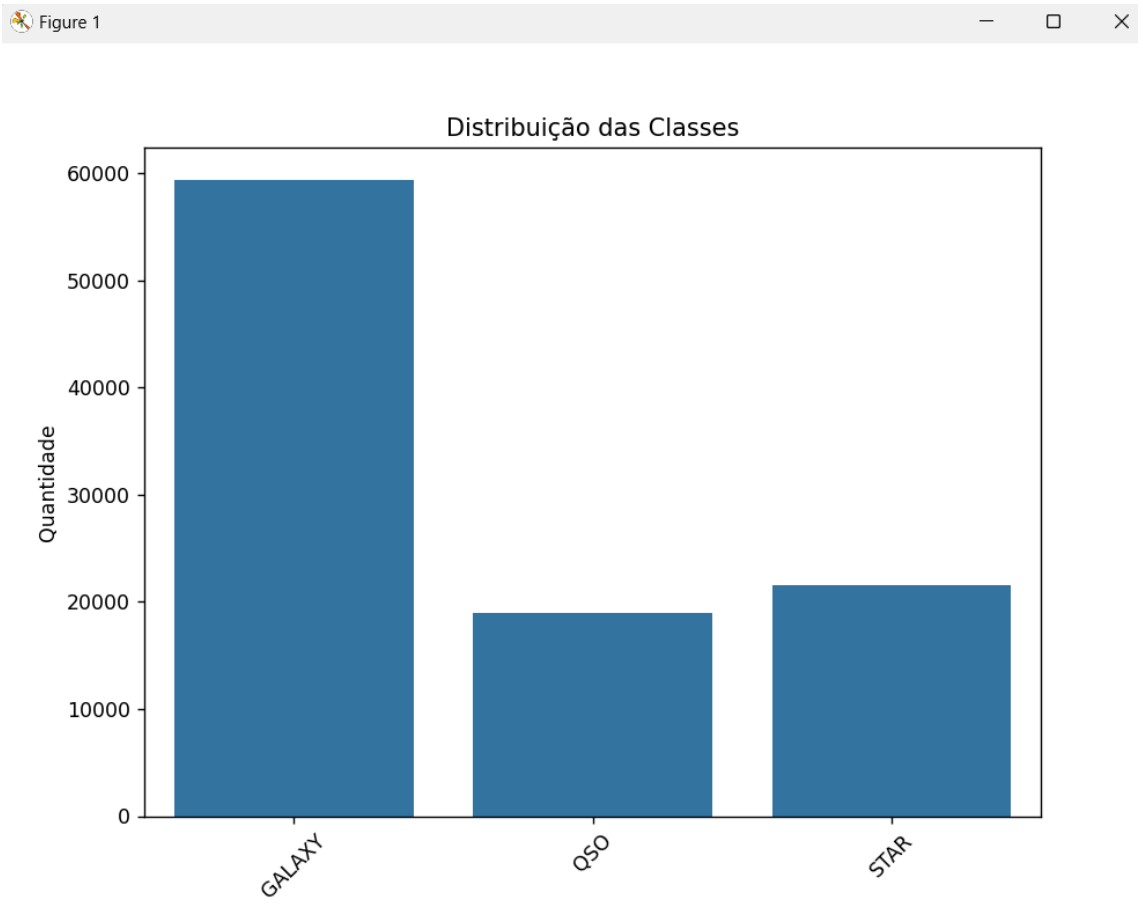
Na quinta etapa realizamos a verificação dos tipos de dados e a conversão, verificando a coluna com o comando ***class*** sendo do tipo ***object*** e convertendo os valores numéricos usando o comando ***astype('category').cat.codes*** mapeando cada categoria.

No que resulta no final na coluna ***class*** contendo valores numéricos, permitindo o uso em modelos de machine learning

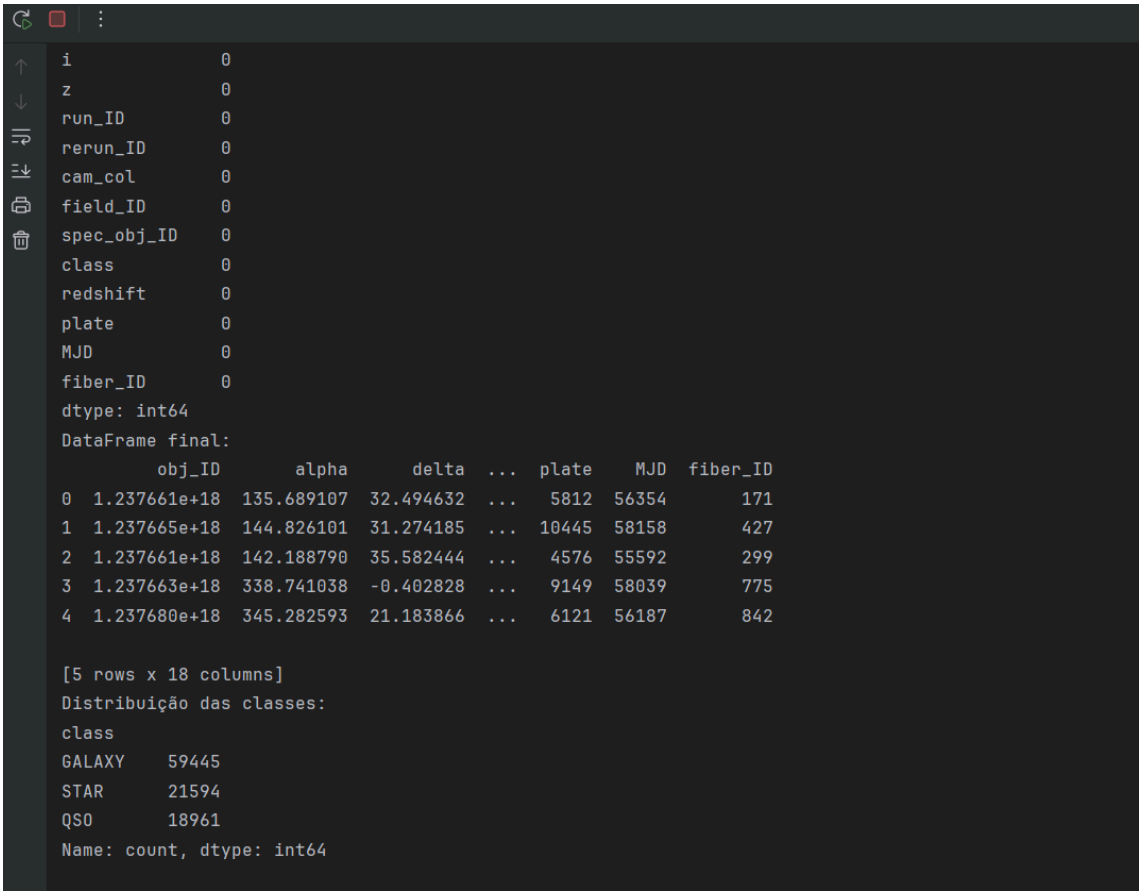
Na etapa 6 foram feita as verificações de modificação e exportação, havendo uma remoção de valores ausentes e transformações de classes o novo arquivo foi exportado e salvo com o novo nome ***star_classification_ajustado.csv*** usando ***to_csv()***

Carregar Dataset → Identificar Valores Faltantes → Remover Valores Faltantes → Selecionar Colunas Relevantes → Analisar Distribuição de Classes → Converter Classes para Numérico → Salvar Dataset Ajustado

Portanto os pontos fortes apresentados foi a limpeza do dataset e o preparo para análises e modelagem, processo de conversão de classes para valores numéricos facilitando o uso de algoritmos machine learning e por último a distribuição de gráficos fornecendo insigths sobre a necessidade de tecnicas para lhe dar coms os dados desbalanceados



As melhorias apresentadas como escalonamentos que, dependendo do modelo de utilização, normalização ou padronização das variáveis pode ser necessárias.



```
i              0
z              0
run_ID         0
rerun_ID       0
cam_col        0
field_ID       0
spec_obj_ID    0
class          0
redshift       0
plate          0
MJD            0
fiber_ID       0
dtype: int64
DataFrame final:
         obj_ID       alpha      delta  ...  plate    MJD  fiber_ID
0  1.237661e+18  135.689107  32.494632  ...   5812  56354       171
1  1.237665e+18  144.826101  31.274185  ...  10445  58158       427
2  1.237661e+18  142.188790  35.582444  ...   4576  55592       299
3  1.237663e+18  338.741038  -0.402828  ...   9149  58039       775
4  1.237680e+18  345.282593  21.183866  ...   6121  56187       842

[5 rows x 18 columns]
Distribuição das classes:
class
GALAXY    59445
STAR      21594
QSO       18961
Name: count, dtype: int64
```

2 questão

tem como objetivo

realizar o knn implementado manualmente avaliando a acurácia em diferentes métricas de distância.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from scipy.spatial.distance import cdist
from collections import Counter

# Carregar os dados
df = pd.read_csv(r"C:\Users\vitor\Downloads\IA.BLACK\IA--SI\AV1\star_classification.csv")

# Seleção das colunas relevantes
colunas_relevantes = ['alpha', 'delta', 'u', 'g', 'r', 'i', 'z', 'redshift', 'class']
df_final = df[colunas_relevantes]

X = df_final.drop(columns=['class'])
y = df_final['class']
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)

def knn_blockwise(X_train, y_train, X_test, k, distance_metric, VI=None, block_size=1000):  1 usage  new *
```

primeiro se importa as bibliotecas para a necessária manipulação de dados.

segundo faze-se o carregamento e seleção de dados de dados carregando o dataset apartir do arquivo CSV e selecionando as colunas relevantes.

terceira etapa é fazer a divisão em conjunto de treinos e teste

dividindo o data set em 30% oara teste e 70% para treino.

na quarta etapa deve-se implementar uma funcão customizada para o KNN coma biblioteca ***scipy*** para o cálculos de distancia

e por ultimo, o calculo da acurácia comparando as acurácias do modelo para cada metrica de distancia utilizando o valos fixo para k sendo o valor 7.

depois se processa os dados na seguiinte ordem

1-dataset

dados astrofísicos, ***alpha, delta, u, g, r, i, z, redshift***

2-divisão dos dados

train_test_split

3- pre-processamento

não houve normalização

4- cálculos de distancia

implementação utilizando a funcaão ***cdist*** da biblioteca ***scipy***

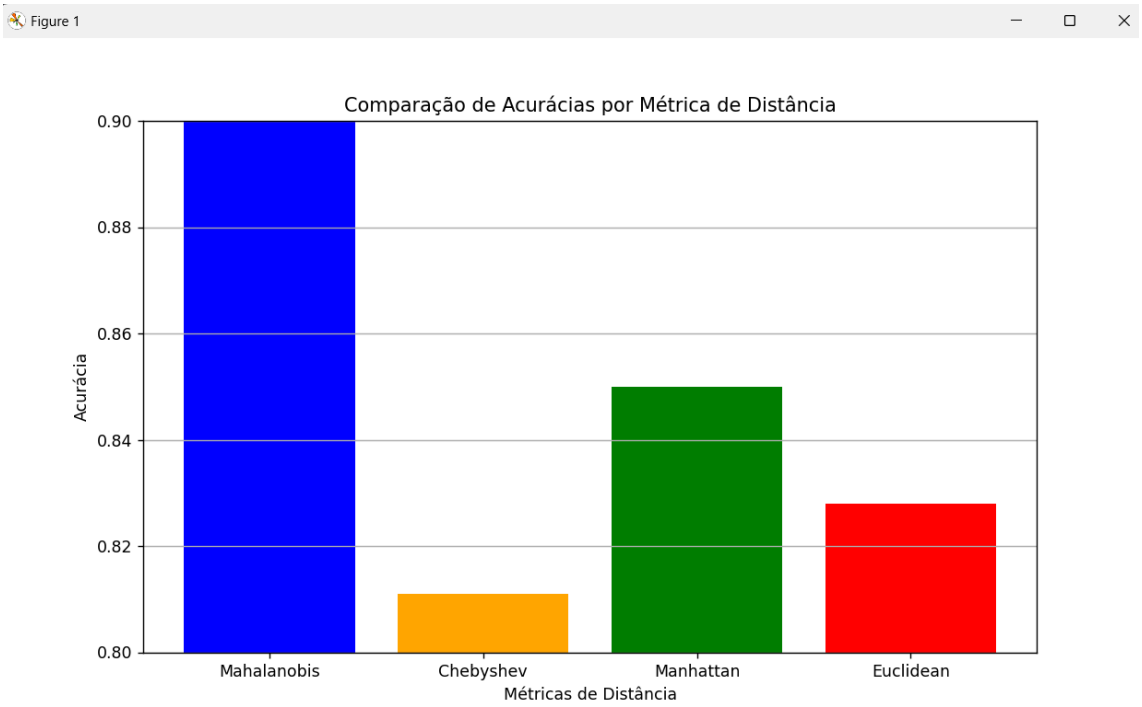
Depois desse relatório, quais os próximos passos você sugere para este projeto?

Início └——> Importar bibliotecas └——> Carregar dataset └——> Selecionar colunas relevantes

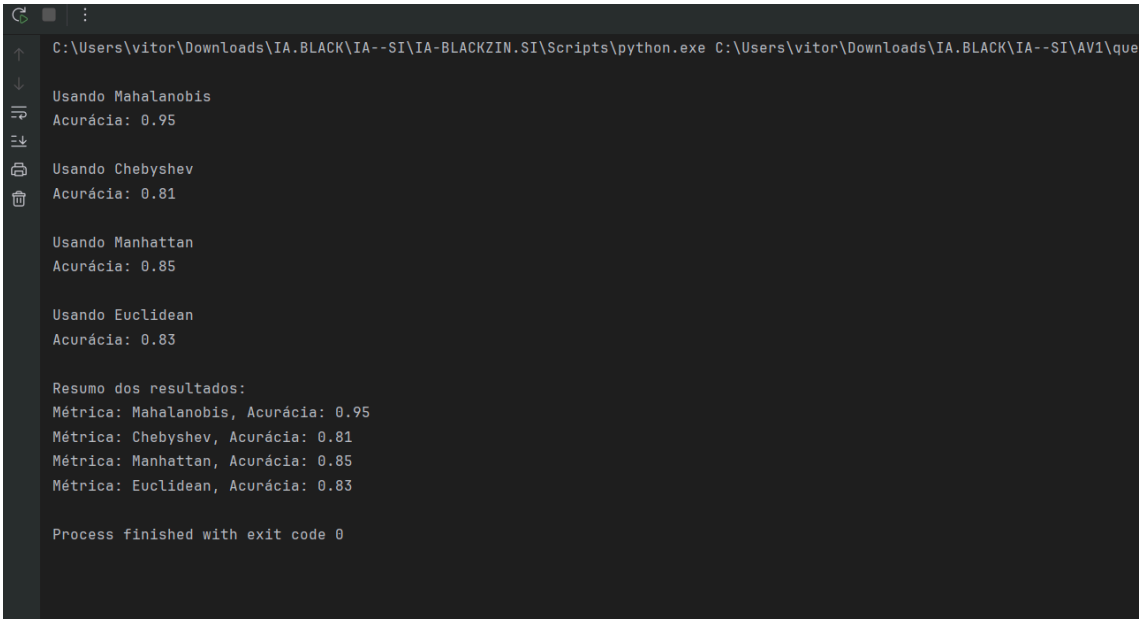
└——> Dividir dados em treino e teste └——> Implementar função KNN

└——> Calcular distâncias └——> Identificar vizinhos mais próximos

└——> Predizer classe └——> Avaliar acurácia └——> Comparar métricas



O KNN foi implementado com sucesso e mostrou bons níveis de acurácia e também apresentou que a métrica Mahalanobis foi mais eficaz para este dataset, mas também a Euclidiana como uma escolha mais robusta e padrão para diversos cenários.



As metricas ***Mahalanobis e Euclidiana*** obtiveram maior acuracia, ja a metrica **Manhattan** *apresentou desempenho intermediário e a* **Chebyshev** um desempenho mais baixo.

3. questão

Nessa questão o objetivo é verificar a normalização dos dados e seus impactos na acurácia do modelo KNN com as tecnicas de normalização logaritmica e normalização de média zero e variancia unitaria.

primeira etapa é a preparação dos dados carregando o dataset e suas respectivas colunas necessárias e substituir os dados negativos por zero e valores ausentes serem preenchidos com a mediana.

na segunda etapa normaliza

na terceira etapa divide em treino e teste

e por ultimo treina e avalia o modelo KNN resultando na normalização logaritmica com valor de 85% e na normalização media zero e variância unitária acurácia 87%. sendo que a normalização media zero e variancia unitaria obteve o melhor resultaado


Figure 1
Comparação de Acurácias com Diferentes Técnicas de Normalização
1.0
0.8
0.6
0.4
0.2
0.0
Acurácia
Logarítmica
Média Zero e Variância Unitária
Técnica de Normalização
(x, y) = (Média Zero e Variância Unitária, 0.649)

4 questão

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
from sklearn.model_selection import train_test_split
from collections import Counter
from joblib import Parallel, delayed

# Carregar o DataFrame
df = pd.read_csv(r"C:\Users\vitor\Downloads\IA.BLACK\IA--SI\AV1\star_classification.csv")

# Definir as colunas relevantes
colunas_relevantes = ['alpha', 'delta', 'u', 'g', 'r', 'i', 'z', 'redshift', 'class']
df_final = df[colunas_relevantes]

# Preprocessamento: corrigir valores inválidos
X = df_final.drop(columns=['class'])
X = X.apply(lambda x: np.where(x < 0, 0, x) if np.issubdtype(x.dtype, np.number) else x)  # Substituir valores negativos por 0
X = X.fillna(X.median(numeric_only=True))  # Preencher valores ausentes com a mediana

y = df_final['class']

# Normalização escolhida (logarítmica)
X_normalized = X.apply(lambda x: np.log1p(x) if np.issubdtype(x.dtype, np.number) else x)

# Divisão treino-teste
X_train, X_test, y_train, y_test = train_test_split( *arrays: X_normalized, y, test_size=0.3, random_state=42)

# Função KNN com cálculo paralelizado e otimizado
def knn_parallel_optimized(X_train, y_train, X_test, k, distance_metric, block_size=500, VI=None, n_jobs=-1):  1 usage  new *
    distance_metrics = {
        "mahalanobis": lambda x, y: cdist(x, y, metric="mahalanobis", VI=VI),
```

De inicio foram importadas as bibliotecas ***pandas, numpy, matplotlib e sklearn.***

essa questão tem como principal objetivo

encontrar a melhor configuração para o KNN em termos de escolha de numeros vizinhos k e da metrica de distancia utilizada.

primeira etapa se carrega os dados e prepara os dados.

carregamento do dataset com o comando ***star_classification.csv*** analizando as colunas relevantes

logo após o carregamento do dataset se inicia o pré
-processamento de dados substituindo valores negativos para 0 e o preenchimento de valores ausentes.

logo após, normalizam-se os dados utilizando o comando ***np.log1p(x).***

Por ultimo dividem-se os dados em treino e teste.

Na segunda etapa, faz-se a implementação do KNN primeiramente com o calculo paralelizado do KNN com o comando ***knn_parallel_optimized*** incluindo as metricas de distancia **Euclidiana**, **Chebyshev**, **Manhattan**, **Mahalanobis.**

No passo dois avalia-se os diferentes valores de k:

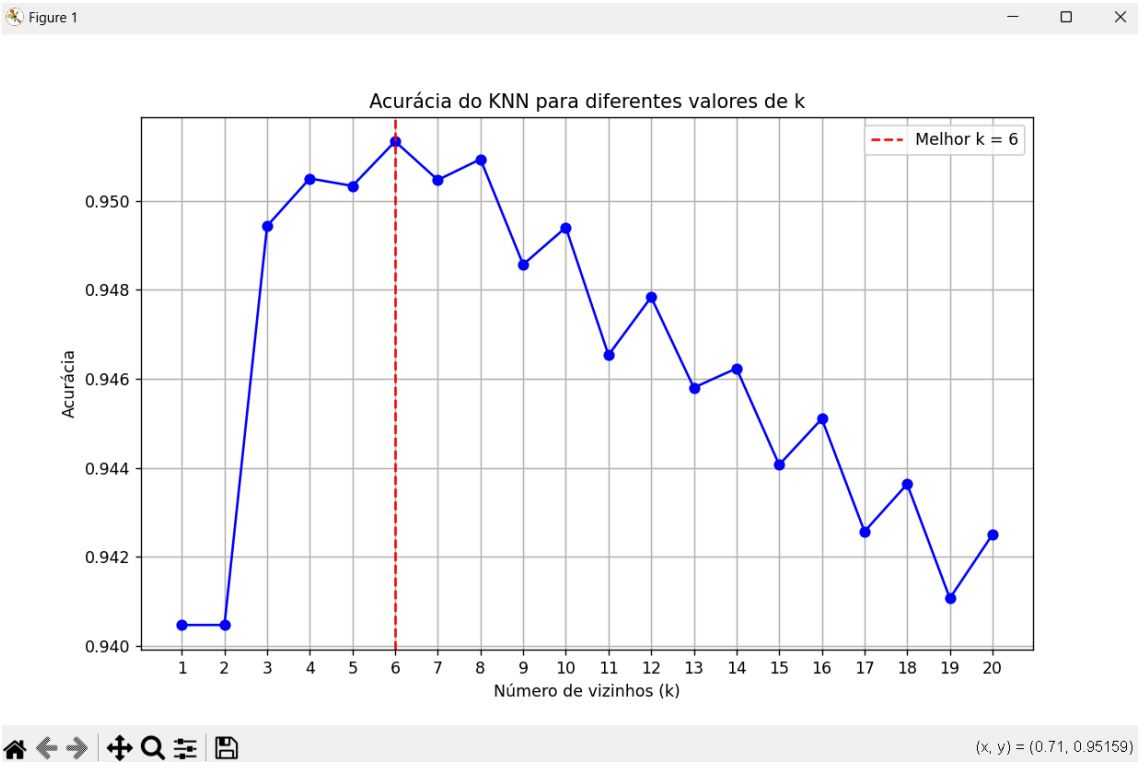
números de vizinhos variando entre 1 a 20 com a previsão calculada em valores e por fim, a escolha da melhor metrica de distancia sendo ela a **Euclidiana** .

Na terceira etapa segue o passo do fluxograma de processo sendo ele carregar, processar, normalizar, dividir, calcular e exibir.

```
C:\Users\vitor\Downloads\IA.BLACK\IA--SI\IA-BLACKZIN.SI\Scripts\python.exe C:\Users\vitor\Downloads\IA.BLACK\IA--SI\AV1\questao4.py
Melhor k: 6
Acurácia com o melhor k: 0.95

Process finished with exit code 0
```

portanto o KNN foi eficaz para identificar melhor valor de K e a metrica de distancia ideal. A análise é possivel ser aprimorada ao experimentar diferentes técnicas e formas de normalização, ajuste fino das métricas de distância e ainda a introdução de técnicas de redução de dimensionalidade, como a Análise de Componentes Principais.



5 questão

4. Conclusões

Os resultados esperados foram satisfeitos? Se não, qual o motivo? Qual a sua análise?

5. Próximos passos