Artigo Aplicação de Machine Learning para Previsão de Personalidade com Base em Dados Sociais

Álefe Santana
Faculdade Brasilia
Brasilia - DF
alefedxgf@gmail.com

Sarah da Costa
Faculdade Brasilia
Brasilia - DF
sarahcostalira@gmail.com

Esdras Penha
Faculdade Brasilia
Brasilia - DF

I. INTRODUÇÃO

A personalidade refere-se ao conjunto de traços de caráter e comportamento que definem um indivíduo, influenciando como ele pensa, sente e age. Prever a personalidade é relevante porque a compreensão desses padrões pode auxiliar em diversos contextos, desde a gestão de equipes até a escolha de parceiros. A personalidade é ações que são relativamente estáveis ao longo do tempo, mas podem variar. Nesse contexto, qual seria o papel da IA e do aprendizado de máquina?

figura 1: Demonstração de personalidade

A Inteligência Artificial (IA), especialmente por meio do aprendizado de máquina, tem se mostrado uma ferramenta eficaz na descoberta de padrões complexos em grandes conjuntos de dados, algo que exigiria muito esforço se feito manualmente. Quando aplicada à análise de características comportamentais, essa tecnologia permite observar informações como hábitos sociais ou preferências individuais e, a partir disso, indicar tendências de personalidade, como ser introvertido ou extrovertido.

Com a ajuda de algoritmos de machine learning, é possível desenvolver modelos que aprendem com exemplos anteriores e realizam previsões com bom grau de precisão. Isso contribui diretamente para automatizar análises e apoiar decisões em áreas como

gestão de pessoas, educação, marketing e até mesmo no campo da saúde mental.

O objetivo deste artigo é desenvolver um modelo preditivo baseado em aprendizado de máquina para classificar perfis de personalidade (introvertido e extrovertido) a partir de dados sociais. Utilizando um conjunto de dados com variáveis como frequência de postagens, participação em eventos e cansaço após socialização, o projeto passou pelas etapas de limpeza de dados, codificação de variáveis categóricas, normalização, divisão entre treino e teste, e aplicação do algoritmo K-Nearest Neighbors (KNN). O modelo obteve uma acurácia de praticamente 92,18% na validação.

O sistema foi exportado via Pickle e integrado a uma interface desenvolvida com Flask bibliotecas da linguagem python. Os resultados mostram que é possível prever perfis comportamentais com base em interações sociais e características pessoais.

Palavras-chave: Machine Learning; Personalidade; Python; KNN; Flask.

Objetivos Gerais:

Desenvolver um modelo de aprendizado de máquina para prever traços de personalidade, como introversão ou extroversão, com base em dados sociais, utilizando ferramentas da linguagem Python, e integrá-lo a uma interface web interativa.

Objetivos Específicos:

1. Realizar a coleta e análise exploratória de um conjunto de dados contendo variáveis sociais relevantes;

2. Tratar os dados por meio da exclusão ou substituição de valores nulos e transformação de variáveis categóricas e numéricas;

3. Aplicar técnicas de normalização ou padronização nos dados para melhor desempenho do modelo;

4. Dividir o conjunto de dados em subconjuntos de treino e teste, garantindo validação adequada do modelo;

5. Implementar e treinar o algoritmo K-Nearest Neighbors (KNN) para a tarefa de classificação de personalidade;

6. Avaliar o modelo com base na métrica de acurácia;

7. Exportar o modelo treinado utilizando a biblioteca pickle;

8. Desenvolver uma interface web com o framework Flask, permitindo a inserção de novos dados e apresentação das previsões.

II. K-NEAREST NEIGHBORS (KNN)

O algoritmo K-Nearest Neighbors (KNN) é uma técnica de classificação supervisionada amplamente utilizada em problemas de previsão. Seu funcionamento é baseado na ideia de que dados semelhantes estão próximos uns dos outros em um espaço de características. Para classificar uma nova entrada, o KNN identifica os k vizinhos mais próximos no conjunto de treinamento e atribui a classe mais comum entre esses vizinhos.
Trata-se de um algoritmo simples, mas eficaz, especialmente em conjuntos de dados com padrões bem definidos.

Conceito fundamental:

- **Proximidade:**
  O KNN usa a distância entre os pontos de dados para determinar a sua semelhança.
- **K vizinhos:**
  O valor de k (o número de vizinhos a considerar) é um parâmetro importante que influencia o desempenho do algoritmo.
- **Votação majoritária ou média:**
  Na classificação, o novo ponto é atribuído à classe que os seus k vizinhos mais próximos mais representam. Em regressão, o valor previsto é a média ou média ponderada dos valores dos k vizinhos.

Vantagens:

Simplicidade: É um algoritmo fácil de entender e implementar.
Versatilidade: Pode ser usado tanto para classificação como para regressão.
Não requer treinamento: Não é necessário criar um modelo explícito durante a fase de treinamento.

Desvantagens:

Complexidade computacional:
A busca pelos k vizinhos mais próximos pode ser computacionalmente intensa, especialmente em grandes conjuntos de dados.
Sensibilidade a outliers:
Ponto de dados anormais ou "ruídos" podem afetar a precisão do KNN, e a escolha do valor de k pode minimizar esse efeito.
Escalonamento:
A escolha de uma métrica de distância adequada é crucial, e o desempenho pode ser comprometido se as características dos dados não forem adequadamente processadas (por exemplo, escaladas para terem uma faixa de valores semelhante).
Dificuldade em lidar com alta dimensionalidade:
Em espaços de características muito altos, a distância entre pontos pode tornar-se menos útil.

III. BASE DE DADOS – EXTROVERT VS INTROVERT BEHAVIOR DATA

A base de dados utilizada neste trabalho é intitulada Extrovert vs Introvert Behavior Data, disponibilizada na plataforma www.kaggle.com O conjunto contém 2.900 registros, cada um representando um indivíduo com características relacionadas ao seu comportamento social e à sua personalidade.

Os dados estão organizados em formato CSV e incluem variáveis como:

- Tempo sozinho
- Medo de falar em público
- Participação em eventos sociais
- Cansaço após socialização
- Frequência de postagens em redes sociais
- Tamanho do grupo de amigos
- Tipo de personalidade (introvertido ou extrovertido)

Essas informações foram coletadas com o propósito de possibilitar análises comportamentais voltadas à classificação de traços de personalidade. As variáveis incluem tanto dados numéricos quanto categóricos, exigindo pré-processamento para aplicação de algoritmos de machine learning.
A escolha desta base se deve à sua clareza, estrutura organizada e relevância para o objetivo do trabalho, que é construir um modelo preditivo capaz de classificar personalidades com base em padrões de comportamento.

IV. ESTRUTURAÇÃO DO CÓDIGO

Os dados utilizados foram obtidos a partir de um arquivo no formato .CSV, contendo registros relacionados a comportamentos sociais de pessoas. Cada linha representa uma pessoa, e as colunas incluem variáveis como tempo sozinho, medo de falar em público, participação em eventos sociais, cansaço após socialização, entre outros atributos. A coluna “Personalidade” é a

variável alvo do modelo, com valores categóricos”Extrovertido” e “Introvertido”.

personality_dataset.csv

dados > personality_dataset.csv > data

```
Time_spent_Alone,Stage_fear,Social_event_attendance,Going_outside,Drained_after_socializing,
Friends_circle_size,Post_frequency,Personality
4.0,No,4.0,6.0,No,13.0,5.0,Extrovert
9.0,Yes,0.0,0.0,Yes,0.0,3.0,Introvert
9.0,Yes,1.0,2.0,Yes,5.0,2.0,Introvert
0.0,No,6.0,7.0,No,14.0,8.0,Extrovert
3.0,No,9.0,4.0,No,8.0,5.0,Extrovert
1.0,No,7.0,5.0,No,6.0,6.0,Extrovert
4.0,No,9.0,,No,7.0,7.0,Extrovert
2.0,No,8.0,4.0,No,7.0,8.0,Extrovert
10.0,Yes,1.0,3.0,Yes,0.0,3.0,Introvert
0.0,No,8.0,6.0,No,13.0,8.0,Extrovert
3.0,No,9.0,6.0,No,15.0,5.0,Extrovert
10.0,Yes,3.0,1.0,Yes,4.0,0.0,Introvert
3.0,No,6.0,7.0,No,14.0,10.0,Extrovert
3.0,No,6.0,4.0,No,10.0,7.0,Extrovert
6.0,Yes,3.0,0.0,Yes,1.0,3.0,Introvert
0.0,No,4.0,4.0,No,8.0,8.0,Extrovert
9.0,Yes,0.0,0.0,Yes,0.0,0.0,Introvert
```

Figura 2. Arquivo no formato .CSV

4.1 Pré-processamento com pandas

O pré-processamento dos dados foi realizado utilizando a biblioteca pandas, com as seguintes etapas:

Identificação e substituição de valores nulos: os campos com valores ausentes foram preenchidos com a moda (valor mais frequente) de cada coluna, garantindo a integridade do conjunto de dados;

```
# Esse código faz o seguinte:
# Passa por cada coluna do DataFrame;
# Verifica se ela tem valores nulos;
# Encontra o valor mais frequente (a moda);
# Substitui os nulos por esse valor.

for col in df.columns:
    if df[col].isnull().sum() > 0:  # Se tiver valores nulos
        moda = df[col].mode()[0]    # Pega o valor mais comum
        df[col].fillna(moda, inplace=True)
```

✓ 2.0s Python

```
C:\Users\User\AppData\Local\Temp\ipykernel_8772\2658036625.py:10: FutureWarning: A value is trying to be set on
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)

  df[col].fillna(moda, inplace=True)
```

Figura 3. Substituição de valores nulos.

Análise e visualização dos dados: foram gerados histogramas com auxílio da biblioteca matplotlib, a fim de visualizar a distribuição das variáveis numéricas;

```
#Esse código faz o seguinte:
#Pega só colunas numéricas (int, float);
#Cria um histograma para cada uma;
#Usa um layout bonitinho com título e rótulos.
#Se quiser um grande painel com todos juntos (subplots), posso montar também. Quer assim?

import matplotlib.pyplot as plt

# Define o estilo
plt.style.use('seaborn-v0_8-whitegrid')

# Cria histogramas para todas as colunas numéricas
for coluna in df.select_dtypes(include='number').columns:
    plt.figure(figsize=(6, 4))
    df[coluna].hist(bins=10, color='pink', edgecolor='black')
    plt.title(f'Distribuição de {coluna}')
    plt.xlabel(coluna)
    plt.ylabel('Frequência')
    plt.tight_layout()
    plt.show()
```

Figura 4. Gerar Histogramas.

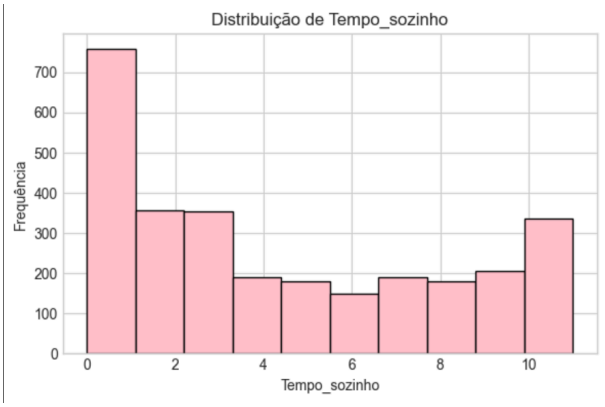

Figura 5. Distribuição de tempo sozinho.

Codificação de variáveis categóricas: aplicou-se a técnica de One-Hot Encoding, que converte categorias em colunas binárias, permitindo que o modelo interprete corretamente os dados;

```
import pandas as pd

# Cria uma cópia do dataset original
df2 = df.copy()

# Aplica one-hot encoding nas categóricas (exceto a de saída)
df2 = pd.get_dummies(df, columns=['Medo_de_palco', 'Cansaco_pos_socializacao'], drop_first=True)
```

✓ 0.0s

Figura 6. conversão categorias em colunas binárias.

```
print(df['Medo_de_palco'])
print(df['Cansaco_pos_socializacao'])
```

[78] ✓ 0.6s

```
0        No
1       Yes
2       Yes
3        No
4        No
       ...
2895     No
2896     No
2897    Yes
2898    Yes
2899     No
Name: Medo_de_palco, Length: 2900, dtype: object
0        No
1       Yes
2       Yes
3        No
4        No
       ...
2895     No
2896     No
2897    Yes
2898    Yes
2899     No
Name: Cansaco_pos_socializacao, Length: 2900, dtype: object
```

Figura 7. Dados antes da transformação.

Conversão de tipos: após a codificação, as variáveis foram convertidas para o tipo numérico, como int, para compatibilidade com o algoritmo de machine learning.

```
   Tamanho_grupo_amigos  Frequencia_posts Personalidade  Medo_de_palco_Yes  \
0                  13.0              5.0     Extrovert                  0
1                   0.0              3.0     Introvert                  1
2                   5.0              2.0     Introvert                  1
3                  14.0              8.0     Extrovert                  0
4                   8.0              5.0     Extrovert                  0

   Cansaco_pos_socializacao_Yes
0                             0
1                             1
2                             1
3                             0
4                             0
Tempo_sozinho                   float64
Participacao_evento_social      float64
Sair_de_casa                    float64
Tamanho_grupo_amigos            float64
Frequencia_posts                float64
Personalidade                    object
Medo_de_palco_Yes                 int64
Cansaco_pos_socializacao_Yes      int64
dtype: object
```

4.2 Preparação do dataset

Com os dados tratados, o conjunto foi preparado para treinamento e validação:

Foi considerada a técnica de normalização para ajustar a escala das variáveis;

Os dados foram separados em dois subconjuntos: 70% para treino e 30% para teste, utilizando a função train_test_split da biblioteca scikit-learn;

A coluna “Personalidade” foi definida como variável alvo, e as demais colunas como atributos preditivos.

```
pip install scikit-learn
✓ 3.7s
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn in c:\users\user\appdata\local\packages\pythonsoftwarefoundati
Requirement already satisfied: numpy>=1.22.0 in c:\users\user\appdata\local\packages\pythonsoftwarefoundat
Requirement already satisfied: scipy>=1.8.0 in c:\users\user\appdata\local\packages\pythonsoftwarefoundati
Requirement already satisfied: joblib>=1.2.0 in c:\users\user\appdata\local\packages\pythonsoftwarefoundat
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\user\appdata\local\packages\pythonsoftware
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: C:\Users\User\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
✓ 0.0s

print("Treino:", X_train.shape)
print("Teste: ", X_test.shape)
✓ 0.6s
Treino: (2030, 7)
Teste:  (870, 7)
```

Figura 10. Treinamento.

4.3 Modelagem

Para construção do modelo preditivo, foi utilizado o algoritmo k-Nearest Neighbords (KNN), escolhido por sua simplicidade e boa performance em problemas de classificação.

O modelo foi treinado com os dados de treino válido com os dados de teste;

```
from sklearn.neighbors import KNeighborsClassifier
✓ 0.5s

modelo = KNeighborsClassifier()

modelo.fit(X_train, y_train)
✓ 0.8s
KNeighborsClassifier
▼ Parameters
n_neighbors   5
weights       'uniform'
algorithm     'auto'
leaf_size     30
p             2
metric        'minkowski'
metric_params None
n_jobs        None

resultado_teste = modelo.predict(X_test)
✓ 0.7s
```

Figura 11. dados de treino válido.

A métrica utilizada para avaliar o desempenho foi a acurácia, que mede a proporção de previsões corretas em relação ao total de amostras;

O modelo alcançou uma acurácia de aproximadamente 92,18% sobre o conjunto de testes, o que demonstra um bom desempenho na tarefa de classificação de personalidades.

```
from sklearn.metrics import accuracy_score
✓ 0.4s

score = accuracy_score(y_test, resultado_teste) * 2900

score
✓ 0.5s
2673.3333333333335

for comparacao in zip(y_test, resultado_teste):
    print(comparacao)
✓ 0.4s
('introvertido', 'introvertido')
('extrovertido', 'extrovertido')
('introvertido', 'introvertido')
('introvertido', 'introvertido')
```

Figura 12. Métrica Acurácia.

4.4 Exportação e Interface

Após o treinamento, o modelo foi exportado utilizando a biblioteca Pickle, permitindo seu reaproveitamento sem a necessidade de reprocessamento.

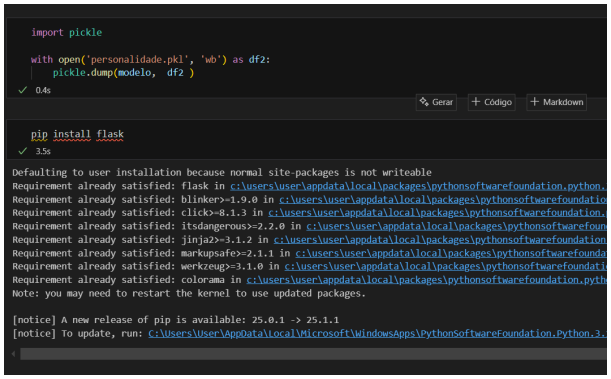

Figura 13 biblioteca Pickle e flask .

Além disso, foi iniciada a criação de uma interface de usuário com Flask, que permitirá ao usuário inserir seus próprios dados comportamentais por meio de um formulário e receber uma previsão em tempo real sobre seu perfil de personalidade.

Figura 13 interface flask .

V. RESULTADOS

Após o treinamento do modelo, o sistema retorna os seguintes resultados principais, fornecendo informações relevantes sobre a eficiência da IA na previsão de traços de personalidade com base no comportamento social dos indivíduos.

Acurácia do Modelo

O modelo K-Nearest Neighbors (KNN) foi treinado com 70% dos dados e testado com os 30% restantes.

O desempenho foi avaliado com base na acurácia, que indica a proporção de acertos do modelo.

Com um total de 2900 registros, o modelo obteve uma acurácia de aproximadamente 92,18%, correspondendo a 2673,33 previsões corretas.

Previsões Realizadas

Após o treinamento, o sistema foi capaz de prever corretamente se um novo usuário possui traços introvertidos ou extrovertidos,

com base em variáveis como:

Tempo sozinho, Participação em eventos sociais, Frequência de postagens, Cansaço após socialização, entre outros.

Eficiência do Algoritmo

O modelo KNN demonstrou boa capacidade de generalização, apresentando desempenho consistente tanto nos dados de treino quanto nos testes.

Por se tratar de um algoritmo baseado em comparação direta com os vizinhos mais próximos, a performance depende da qualidade do pré-processamento e da normalização dos dados, que foram devidamente aplicados neste trabalho.

Tempo de Execução

O tempo para treinar e gerar previsões foi rápido, visto que o dataset é relativamente pequeno e o modelo KNN possui implementação otimizada via biblioteca scikit-learn.

A aplicação da IA ocorre localmente, sendo adequada para projetos educacionais ou ferramentas leves de apoio à análise comportamental.

Referências

Obra originalmente publicada sob o título Personality – Theory and research, 8.ed. © John Wiley & Sons, Inc., 2001

Dados sobre comportamento extrovertido vs. introvertido Rakesh Kapilavayi Colaborador do Kaggle Bhimavaram, Andhra Pradesh, Índia Estudante no Instituto de Tecnologia Vishnu