UNIVERSIDADE FEDERAL DO MARANHÃO
DEPARTAMENTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO
Disciplina: Inteligência Artificial
Prof: Tiago Bonini Borchartt
Alunos: Giulia de Araujo Freulon, Isabelle Maria Borges, Juliana Gonçalves Camara e Valter Abreu Silva

Trabalho Final – Relatório

K-Nearest Neighbors (KNN)

- Etapas de Implementação do modelo:

1. Carregar os dados

```
# - CARREGAR OS DADOS -
df = pd.read_csv(r'database\Cartao de credito.csv', encoding='ISO-8859-1', delimiter=";")
df_unlabeled = pd.read_csv(r'database\Dataset Validacao.csv', encoding='ISO-8859-1', delimiter=";")
```

- Utilizamos o Pandas para carregar as bases csv.

2. Tratar os dados

```
# - TRATAR OS DADOS -
# Removendo a coluna de identificação da amostra
df.drop(columns=['Identificador da transação'], inplace=True)
df_unlabeled.drop(columns=['Identificador da transação'], inplace=True)

# Convertendo a coluna 'Bandeira' em variáveis numéricas usando codificação one-hot
df = pd.get_dummies(df, columns=['Bandeira do Cartão'], drop_first=True)
df_unlabeled = pd.get_dummies(df_unlabeled, columns=['Bandeira do Cartão'], drop_first=True)

# Trocando os valores SIM por 1 e NÃO por 0
df['Fraude'] = df['Fraude'].map({'SIM': 1, 'NÃO': 0})

# Convertendo as colunas numéricas para float
df['Distância de Casa'] = df['Distância de Casa'].str.replace(',', '.').astype(float)
df_unlabeled['Distância de Casa'] = df_unlabeled['Distância de Casa'].str.replace(',', '.').astype(float)
df['Distância da Última Transação'] = df['Distância da Última Transação'].str.replace(',', '.').astype(float)
df_unlabeled['Distância da Última Transação'] = df_unlabeled['Distância da Última Transação'].str.replace(',', '.').astype(float)
df['Razão entre o valor da compra e o valor médio'] = df['Razão entre o valor da compra e o valor médio'].str.replace(',', '.').astype(float)
df_unlabeled['Razão entre o valor da compra e o valor médio'] = df_unlabeled['Razão entre o valor da compra e o valor médio'].str.replace(',
', '.').astype(float)

# Separando atributos e rótulos
X = df.drop(columns=['Fraude'])
X_unlabeled = df_unlabeled.drop(columns=['Fraude'])
y = df['Fraude']
```

- No tratamento de dados:
  - ✓ removemos a coluna de identificação da amostra;
  - ✓ convertemos a coluna de ‘Bandeira’ em rótulos one-hot;
  - ✓ trocamos os valores da coluna ‘Fraude’ de nominais para numéricos;
  - ✓ convertemos as colunas numéricas para float;
  - ✓ separamos os atributos e rótulos;

3. Separar os dados entre treino e teste

```
# - SEPARAR OS DADOS EM TREINO E TESTE -
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Utilizamos o train_test_split para separar os dados em treino e teste.

4. Normalizar os dados

```
# - NORMALIZAR OS DADOS -
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_unlabeled = scaler.transform(X_unlabeled)
```

- Utilizamos o MinMaxScaler do Sklearn para normalizar os dados em valores entre 0 e 1.

5. Treinar o modelo

```
# - TREINAR O MODELO -
knn = KNeighborsClassifier(
    n_neighbors=5,
    weights='distance',
    algorithm='auto',
    metric='euclidean')

knn.fit(X_train, y_train)
```

- Treinamos o modelo utilizando o classificador KNeighborsClassifier do Sklearn.

6. Fazer previsões

```
# - FAZER PREVISÕES -
y_pred = knn.predict(X_test)

# Verificando acurácia
accuracy = accuracy_score(y_test, y_pred)
print(f'\nAcurácia do modelo: {accuracy * 100:.2f}%')

# Verificando a matriz de confusão
class_names = ['NÃO', 'SIM']
conf_matrix = confusion_matrix(y_test, y_pred)
print('\nMatriz de Confusão:')
print(pd.DataFrame(conf_matrix, columns=class_names, index=class_names))

# Verificando o relatório de classificação
print('\nRelatório de Classificação:')
print(classification_report(y_test, y_pred, target_names=class_names))
```

- Utilizamos as métricas de accuracy_score, confusion_matrix e classification_report do Sklearn para avaliar nosso modelo.

7. Classificar dados sem rótulo

```
# - CLASSIFICAR DADOS SEM RÓTULO -
print("CLASSIFICANDO DADOS SEM RÓTULO")
y_unlabeled_pred = knn.predict(X_unlabeled)

# Substituindo 0 por 'NÃO' e 1 por 'SIM'
y_pred_str = ['SIM' if x == 1 else 'NÃO' for x in y_unlabeled_pred]

# Adicionando a coluna de Fraude preenchida ao DataFrame
new_csv = pd.read_csv(r'database\Dataset Validacao.csv', encoding='ISO-8859-1', delimiter=';')
new_csv = new_csv.drop(columns=['Fraude'])
new_csv['Fraude'] = y_pred_str

# Salvando o resultado em um novo arquivo CSV
new_csv.to_csv(r'database\Dados classificados.csv', index=False, encoding='ISO-8859-1', sep=';')
```

- Fizemos a previsão da classificação dos dados sem rótulo e geramos um csv com esses resultados.

- Treino e Teste

  Separamos 80% dos dados para treino e 20% para teste.

- Parametrização

  Utilizamos os parâmetros de:

  ✓ n_neighbors = 5

  - Esse parâmetro vai permitir que se utilize 5 vizinhos para classificar um dado. Escolhemos 5, pois é um valor padrão utilizado no knn.

  ✓ weights = ‘distance’

  - Esse parâmetro vai permitir que vizinhos mais próximos tenham mais influência do que os mais distantes. Escolhemos ‘distance’, pois isso pode ajudar a capturar melhor os padrões locais nos dados.

  ✓ algorithm = ‘auto’

  - Esse parâmetro vai permitir que o scikit-learn escolha o melhor algoritmo para o nosso problema. Escolhemos ‘auto’, pois preferimos deixar que a escolha do algoritmo fosse feita com base nos valores passados no método fit.

  ✓ metric = ‘euclidean’

  - Esse parâmetro vai permitir que seja utilizada a distância euclidiana para calcular a distância entre vizinhos. Escolhemos ‘euclidean’, pois é comum e funciona bem na maioria dos casos.

- Resultados

  Nosso modelo conseguiu atingir as seguintes métricas:

  ✓ Acurácia

  ```
  Acurácia do modelo: 99.83%
  ```

- ✓ Matriz de Confusão

```
Matriz de Confusão:
        NÃO    SIM
NÃO  182238     66
SIM     279  17217
```

- ✓ Relatório de Classificação

```
Relatório de Classificação:
              precision    recall  f1-score   support

         NÃO       1.00      1.00      1.00    182304
         SIM       1.00      0.98      0.99     17496

    accuracy                           1.00    199800
   macro avg       1.00      0.99      0.99    199800
weighted avg       1.00      1.00      1.00    199800
```

- Classificação das instâncias desconhecidas
  Geramos um arquivo csv com as respectivas classificações das instâncias desconhecidas.