Aprendizado de Máquina (Algoritmos de ML)

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, precision_score
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import os
```

```
Registro_pagamento = pd.read_csv('Registro_pagamento.csv')
registro_pagamento.head()
```

	ID_pagamento	ID_cliente	ID_forma_pagamento	data	valor	status
0	1	1	1	2024-07-22	649.8	concluido
1	2	2	2	2024-07-23	629.7	concluido
2	3	3	3	2024-07-24	649.6	concluido
3	4	4	4	2024-07-24	529.8	concluido
4	5	5	5	2024-07-28	1139.9	concluido

```
Y = registro_pagamento['valor']
X = registro_pagamento.drop(['valor'], axis=1)

x_train, x_test, y_train, y_test = model_selection.train_test_split(X, Y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(x_train, y_train)
```

```
X = registro_pagamento.drop('valor', axis=1)
y = registro_pagamento['valor']

x_train, x_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=42)

predictions = registro_pagamento(x_train)

mse= mean_squared_error(y_train, predictions)
mae= mean_absolute_error(y_train, predictions)
rmse = mean_squared_error(y_train, predictions, squared=False)
print('MSE:', mse)
print('MAE:', mae)
print('RMSE:', rmse)
```

```
Mean Absolute Error: 84.16250001592562
Mean Squared Error: 8796.251565250057
R² Score: -86.0894439766356
```

Entrega 1: Exploração de Dados e Pré-processamento:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from google.colab import files
```

```python
Registro_pagamento = pd.read_csv('Registro_pagamento.csv')
registro_pagamento.head()
```

	ID_pagamento	ID_cliente	ID_forma_pagamento	data	valor	status
0	1	1	1	2024-07-22	649.8	concluido
1	2	2	2	2024-07-23	629.7	concluido
2	3	3	3	2024-07-24	649.6	concluido
3	4	4	4	2024-07-24	529.8	concluido
4	5	5	5	2024-07-28	1139.9	concluido

```python
def train_and_evaluate_model(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = LogisticRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print("Acurácia:", accuracy_score(y_test, y_pred))
    print("Precisão:", precision_score(y_test, y_pred))
    print("Recall:", recall_score(y_test, y_pred))
    print("F1-Score:", f1_score(y_test, y_pred))

    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
                xticklabels=['Pendente', 'Concluído'], yticklabels=['Pendente', 'Concluído'])
    plt.title('Matriz de Confusão')
    plt.xlabel('Previsto')
    plt.ylabel('Real')
    plt.show()
```

```
Acurácia: 1.0
Precisão: 1.0
Recall: 1.0
F1-Score: 1.0
```

Entrega 2: Implementação de Modelos de Aprendizado de Máquina:

```python
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import files
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

```python
Registro_pagamento = pd.read_csv('Registro_pagamento.csv')
registro_pagamento.head()
```

	ID_pagamento	ID_cliente	ID_forma_pagamento	data	valor	status
0	1	1	1	2024-07-22	649.8	concluido
1	2	2	2	2024-07-23	629.7	concluido
2	3	3	3	2024-07-24	649.6	concluido
3	4	4	4	2024-07-24	529.8	concluido
4	5	5	5	2024-07-28	1139.9	concluido

```python
def kmeans_clustering(df, n_clusters=3):
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(df)

    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(scaled_data)
    labels = kmeans.labels_

    silhouette_avg = silhouette_score(scaled_data, labels)
    print("Silhouette Score:", silhouette_avg)

    return labels
```

Cluster	valor	status
0	614.725	1.0
1	3500.000	0.0
2	1139.900	1.0

Entrega 3: Otimização e Validação do Modelo: