Aplicação de Modelo de Machine Learning no Projeto Integrador

Lucca Brandão RA: 23024740

Murilo Lopes RA:24026183

Rodrigo Cruz RA: 24026578

Vinicius Kingo RA: 24026141

Modelo: Random Forest Classifier
Objetivo: Prever se o cliente responderá à campanha.

Metodo Random Forest Classifier :

Neste projeto, utilizou-se o aprendizado de máquina para antecipar a probabilidade de um cliente reagir a uma campanha promocional, fundamentando-se em informações prévias da empresa. O modelo implementado foi um Classificador de Floresta Aleatória, um algoritmo de aprendizado supervisionado que integra várias árvores de decisão para aprimorar a exatidão das previsões. Após ser treinado com os dados contidos no arquivo CampaignQueue, o modelo demonstrou a habilidade de reconhecer padrões e efetuar previsões automáticas, ajudando a otimizar as escolhas de marketing e a alocar os recursos das campanhas de maneira mais eficaz.

Etapas Realizadas:

1. Limpeza e transformação dos dados
2. Criação da variável alvo (has_response)
3. Divisão em treino e teste (75/25)
4. Treinamento do modelo Random Forest

Resultados Obtidos:

Acurácia do modelo: 63.76%

Codigo Utilizado :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet
```

```
df = pd.read_csv("CampaignQueue_semicolon.csv", sep=';')
df["has_response"] = df["response"].apply(lambda x: 0 if pd.isna(x) or
str(x).strip() == "" else 1)
X = df[["jobId", "campaignId", "status", "customerId"]]
y = df["has_response"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
modelo = RandomForestClassifier(random_state=42)
modelo.fit(X_train, y_train)
y_pred = modelo.predict(X_test)

acc = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
matrix = confusion_matrix(y_test, y_pred)

doc = SimpleDocTemplate("relatorio_resultados.pdf", pagesize=A4)
styles = getSampleStyleSheet()
conteudo = []
conteudo.append(Paragraph("<b>Aplicação de Modelo de Machine Learning no
Projeto Integrador</b>", styles['Title']))
conteudo.append(Spacer(1, 12))
conteudo.append(Paragraph("Modelo: Random Forest Classifier",
styles['Normal']))
conteudo.append(Paragraph("Objetivo: Prever se o cliente responderá à
campanha.", styles['Normal']))
conteudo.append(Spacer(1, 12))
conteudo.append(Paragraph("<b>Etapas Realizadas:</b>", styles['Heading2']))
conteudo.append(Paragraph("1. Limpeza e transformação dos dados",
styles['Normal']))
conteudo.append(Paragraph("2. Criação da variável alvo (has_response)",
styles['Normal']))
conteudo.append(Paragraph("3. Divisão em treino e teste (75/25)",
styles['Normal']))
conteudo.append(Paragraph("4. Treinamento do modelo Random Forest",
styles['Normal']))
conteudo.append(Spacer(1, 12))
conteudo.append(Paragraph("<b>Resultados Obtidos:</b>", styles['Heading2']))
conteudo.append(Paragraph(f"Acurácia do modelo: {round(acc*100, 2)}%",
styles['Normal']))
conteudo.append(Spacer(1, 12))
conteudo.append(Paragraph("<b>Relatório de Classificação:</b>",
styles['Heading3']))
conteudo.append(Paragraph(f"<pre>{report}</pre>", styles['Code']))
conteudo.append(Spacer(1, 12))
conteudo.append(Paragraph("<b>Matriz de Confusão:</b>", styles['Heading3']))
conteudo.append(Paragraph(str(matrix), styles['Normal']))
conteudo.append(Spacer(1, 20))
conteudo.append(Paragraph("Conclusão: O modelo conseguiu prever com boa
precisão se um cliente tende a responder às campanhas. "
                          "A abordagem pode ser refinada com mais variáveis
e ajuste de hiperparâmetros.", styles['Normal']))
doc.build(conteudo)
```