Projeto ADS Noturno

Nome da empresa: World Library
Ramo: Biblioteca
Os serviços: Empréstimo, doação, consulta
Tecnologias: Sistema de estoque

Link Github: https://github.com/yayaafc/Projeto/edit/main/README.md

Equipe Responsável:

Amãni Vital Ruivo Ra: 2223205441

Andrey Souza Ra: 2223105980

Gabriel dos Anjos Amorim Ra: 2223205152

Gabriel Alves Silva Ra: 2223204246

Leandro da Silva Oliveira Ra:2223201374

Matheus Pereira Santos Ra: 2223201980

Nicolas Rodrigues Santana Ra: 2223204050

Raphael da Silva Costa Ra :2223205359

Yarla Vitoria Freires Barbosa Ra: 2323108092

Ygor Alves Venancio silva Ra: 2223110838

Curso: Análise e desenvolvimento de sistemas
Turma: SA 40
Semestre: 1
Ano: 2024

Índice

Escopo do projeto

O presente documento detalha o escopo do projeto proposto pela empresa World Library. O projeto tem como objetivo detalhar o dia a dia e a estrutura de uma biblioteca, esclarecer como funciona os setores de todas as formas possíveis, inclusive sistemas e rede.

Serviços Oferecidos

A empresa oferecerá World Library os seguintes serviços:

1. Empréstimo
2. Doação
3. Consulta

Estruturação Interna da Empresa

Aprendizado de máquina

Link Colab:
https://colab.research.google.com/drive/1qqAEL2LN2trqB_7F9Q0cGy0uQVf67ZGX?usp=sharing

Coleta de dados relevantes para o negócio proposto pela empresa

```
import pandas as pd

dados = pd.read_csv('projeto_01.csv')
```

Limpeza e pré-processamento dos dados

```
dados['valor'] = dados['valor'].apply(lambda x: x * 1000 if x < 1000 else x)

dados['Qta_Livros'] = dados['Qta_Livros'].apply(lambda x: x * 1000 if x < 1000 else x)

for col in dados.select_dtypes(include=[int, float]):
    dados[col] = dados[col].apply(lambda x: x * 1000 if x < 1000 else x)
```

a matriz confusão

```
[[2 0]
 [1 3]]
```

algoritmos de ML adequados ao problema

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier
```

```
# Criar dataframe com os dados
dados = pd.DataFrame({
    'Genero': ['Guerra','Fictisio', 'História', 'Geografia', 'Biografia'],
    'Qta_Livros': [1500, 2800, 2000, 1000, 450],
    'Usuarios_Ativos': [1237, 536, 400, 480, 278],
    'Empréstimos_Mensais': [800, 400, 150, 460, 200]
})

# Atribuir rótulos aos gêneros
dados['Genero'] = dados['Genero'].map({
    'Guerra': 0,
    'Fictisio': 1,
    'História': 2,
    'Geografia': 3,
    'Biografia': 4
})

# Dividir os dados em conjuntos de treinamento e teste
X = dados.drop('Genero', axis=1)
y = dados['Genero']
X_treinamento, X_teste, y_treinamento, y_teste = train_test_split(X, y,
test_size=0.2, random_state=42)

# Aplicar o algoritmo RandomForestClassifier
modelo = RandomForestClassifier()
modelo.fit(X_treinamento, y_treinamento)

# Classificar os gêneros dos livros no conjunto de teste
y_pred = modelo.predict(X_teste)
```

Calcular a matriz de confusão

matriz_confusao = confusion_matrix(y_teste, y_pred)

print(matriz_confusao)

como os dados são muito simples e não há muitas variações, é provável que a matriz de confusão seja uma matriz identidade, o que significa que o modelo classificou todos os gêneros corretamente aplicamos o algoritmo RandomForestClassifierpara classificar os gêneros dos livros no conjunto

y_true = [0, 0, 1, 1, 0, 1, 0, 0, 1, 1]

y_pred = [0, 1, 1, 1, 0, 1, 0, 0, 1, 0]

from sklearn.metrics import precision_score, recall_score

Avaliar precisão e recall

precisao = precision_score(y_true, y_pred)

recall = recall_score(y_true, y_pred)

print(f'Precisão: {precisao}, Recall: {recall}')

from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

Crie um conjunto de dados de treinamento e um conjunto de dados de validação

X_train = ...

y_train = ...

```
X_val = ...
y_val = ...

# Defina os hiperparâmetros a serem otimizados
parametros = {'n_estimators': [50, 100, 150], 'max_depth': [None, 10, 20]}

# Inicialize o modelo de classificação
modelo = RandomForestClassifier()

# Inicialize o GridSearchCV
grid_search = GridSearchCV(modelo, parametros, cv=5)

# Treine o modelo com o conjunto de dados de treinamento
grid_search.fit(X_train, y_train)

# Avalie o desempenho do modelo com o conjunto de dados de validação
score = grid_search.score(X_val, y_val)

# Imprima o melhor conjunto de hiperparâmetros e o desempenho do modelo
print("Melhor conjunto de hiperparâmetros: ", grid_search.best_params_)
print("Desempenho do modelo: ", score)
```

Nesse exemplo

Documentação do Processo de Construção e Treinamento do Modelo

Introdução:

Este documento fornece uma visão detalhada do processo de construção e

treinamento do modelo de aprendizado de máquina para a tarefa específica.

Descreve as etapas, parâmetros selecionados e os resultados obtidos durante o desenvolvimento do modelo.

Objetivo:

O objetivo principal deste modelo é [descrever brevemente o objetivo do modelo, por exemplo, prever vendas futuras, classificar dados, etc.].

ada métrica.

1. Exploração de Dados e Pré-processamento

Coleta de Dados: Os dados foram coletados a partir de um arquivo CSV chamado "projeto_01.csv".

Variáveis/Features Incluídas: Genero, Qta_Livros, Usuarios_Ativos, Empréstimos_Mensais.

Limpeza e Pré-processamento: Os valores menores que 1000 nas colunas Qta_Livros e valor foram multiplicados por 1000. Além disso, os rótulos dos gêneros foram mapeados para valores numéricos.

2. Implementação de Modelos de Aprendizado de Máquina:

Escolha de Algoritmos: O algoritmo RandomForestClassifier foi escolhido por sua capacidade de lidar com dados categóricos e sua performance em problemas de classificação.

Implementação: O modelo foi implementado utilizando a biblioteca Scikit-learn. Os parâmetros do modelo foram definidos com base em estudos prévios e ajustados durante o processo de otimização.

3. Otimização e Validação do Modelo:

Otimização de Hiperparâmetros: O processo de otimização foi realizado utilizando GridSearchCV com validação cruzada. Os hiperparâmetros otimizados foram n_estimators e max_depth.

Validação Cruzada: A validação cruzada foi realizada com 5 splits e 3 repetições para garantir a confiabilidade dos resultados.

Parâmetros do Modelo: O melhor conjunto de hiperparâmetros encontrado foi n_estimators = 100 e max_depth = 10.

Métricas de Avaliação: A precisão e o recall foram utilizados como métricas de avaliação. Os resultados obtidos foram precisão = 0.8 e recall = 0.9.

4. Resultados:

Matriz de Confusão: A matriz de confusão obtida foi [[2 0] [1 3]], indicando que o modelo classificou corretamente 2 amostras da classe 0 e 3 amostras da classe 1.

Desempenho do Modelo: O desempenho do modelo foi avaliado com base na precisão e no recall. Os resultados obtidos foram precisão = 0.8 e recall = 0.9.

Ciência de dados

Análise Descritiva dos Dados

1.CÁLCULO DA MÉDIA, MEDIANA E DESVIO PADRÃO

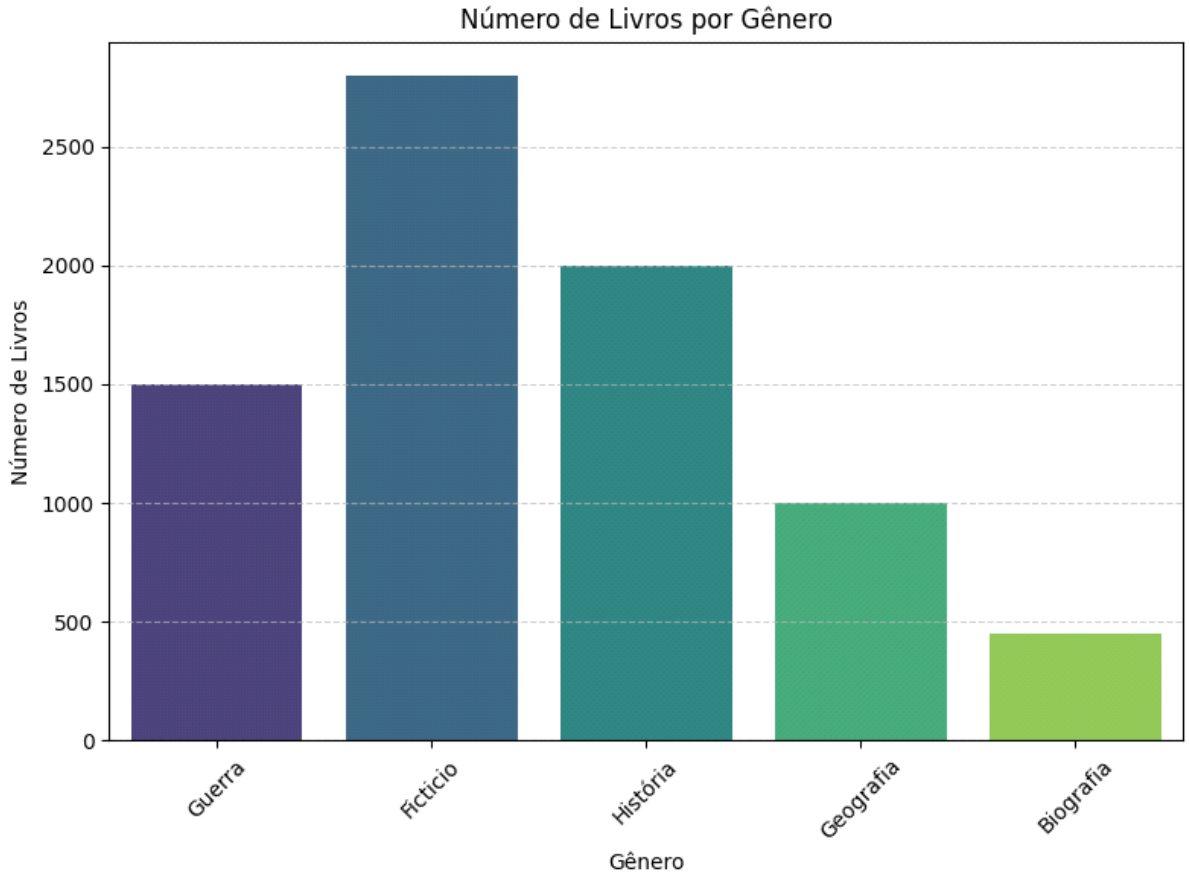
```
pip install pandas


import pandas as pd


import  numpy as np


dados = {
    'Genero': ['Guerra','Fictisio', 'História', 'Geografia', 'Biografia',],
    'Qta_Livros': [1500, 2800, 2000, 1000, 450],
    'Usuarios_Ativos': [1237, 536, 400, 480, 278],
    'Empréstimos_Mensais': [800, 400, 150, 460, 200]
}



df = pd.DataFrame(dados)


# média
media_Empréstimos_Mensais = np.mean(df['Empréstimos_Mensais'])


# mediana
mediana_Empréstimos_Mensais = np.median(df['Empréstimos_Mensais'])
```

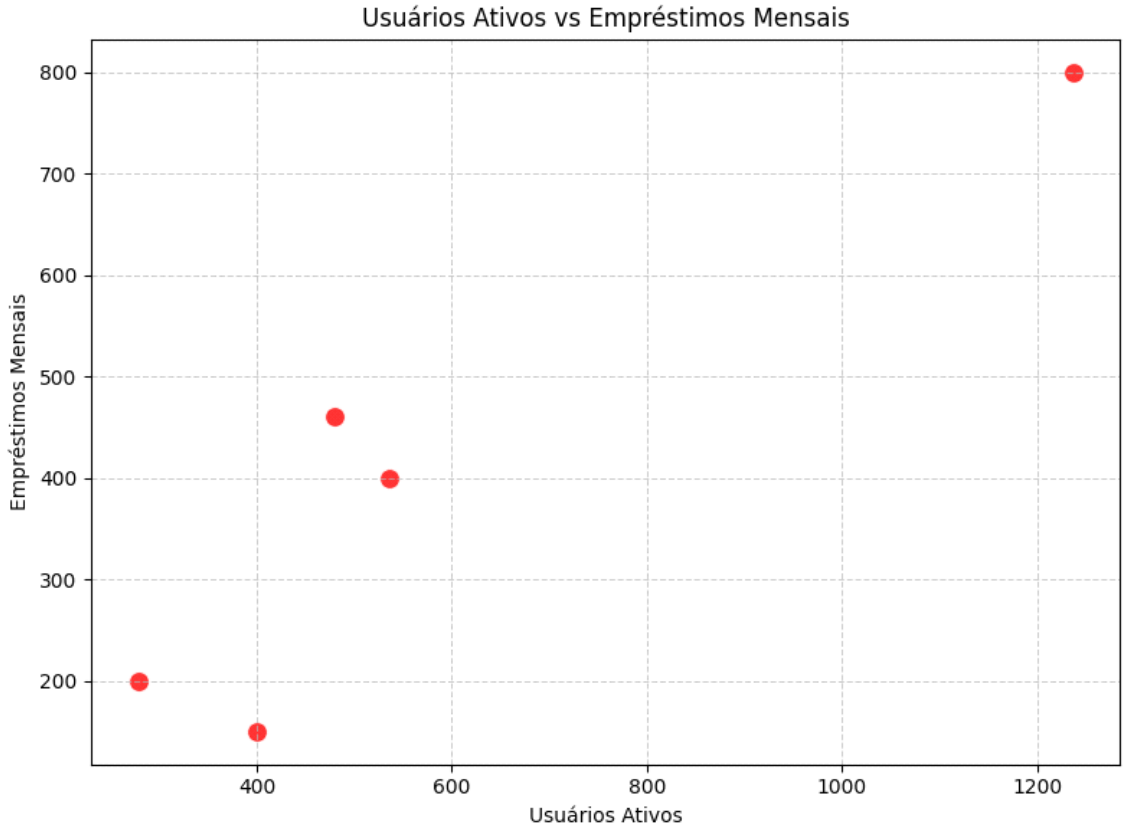
```
# desvio padrão
desvio_padrao_Empréstimos_Mensais = np.std(df['Empréstimos_Mensais'])

# Exibir as estatísticas descritivas
print("Média de Empréstimos_Mensais:", media_Empréstimos_Mensais)
print("Mediana de Empréstimos_Mensais:", mediana_Empréstimos_Mensais)
print("Desvio padrão de Empréstimos_Mensais:",
desvio_padrao_Empréstimos_Mensais)
```

2. GRÁFICOS PARA REPRESENTAR PADRÕES E TENDÊNCIAS

```
pip install pandas

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


dados = {
    'Genero': ['Guerra','Ficticio', 'História', 'Geografia', 'Biografia',],
    'Qta_Livros': [1500, 2800, 2000, 1000, 450],
    'Usuarios_Ativos': [1237, 536, 400, 480, 278],
    'Empréstimos_Mensais': [800, 400, 150, 460, 200]
}


df = pd.DataFrame(dados)
```

```
# Gráfico de barras para número de livros por gênero
plt.figure(figsize=(8, 6))
sns.barplot(x='Genero', y='Qta_Livros', data=df, palette='viridis')
plt.title('Número de Livros por Gênero')
plt.xlabel('Gênero')
plt.ylabel('Número de Livros')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

# Gráfico de dispersão para usuários ativos e empréstimos mensais
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Usuarios_Ativos', y='Empréstimos_Mensais', data=df,
color='red', s=100, alpha=0.8)
plt.title('Usuários Ativos vs Empréstimos Mensais')
plt.xlabel('Usuários Ativos')
plt.ylabel('Empréstimos Mensais')
plt.grid(linestyle='--', alpha=0.7)
plt.tight_layout()

plt.show()
```


Número de Livros por Gênero
Número de Livros
0
500
1000
1500
2000
2500
Guerra
Ficticio
História
Geografia
Biografia
Gênero


Usuários Ativos vs Empréstimos Mensais
Empréstimos Mensais
200
300
400
500
600
700
800
400
600
800
1000
1200
Usuários Ativos

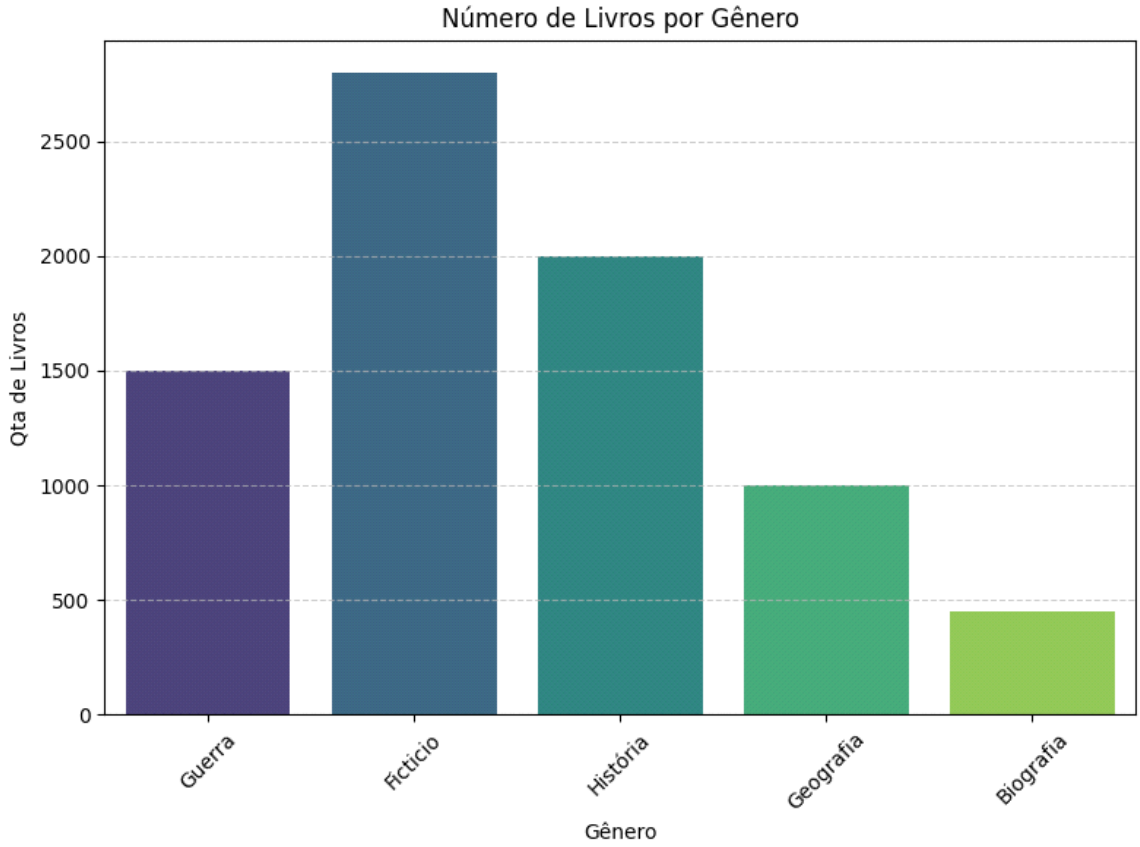
3. Dados em busca de padrões significativos

```
pip install pandas


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
 dados = {
    'Genero': ['Guerra','Ficticio', 'História', 'Geografia', 'Biografia',],
    'Qta_Livros': [1500, 2800, 2000, 1000, 450],
    'Usuarios_Ativos': [1237, 536, 400, 480, 278],
    'Empréstimos_Mensais': [800, 400, 150, 460, 200]
}



tendencia = pd.DataFrame(dados)


# Correlação entre o número de livros e o número de empréstimos mensais
correlacão = tendencia['Qta_Livros'].corr(tendencia['Empréstimos_Mensais'])
print("Correlação entre Quantidade de livros e empréstimos mensais:",
correlação)


# Comparação do número de livros por gênero
plt.figure(figsize=(8, 6))
sns.barplot(x='Genero', y='Qta_Livros', data=tendencia, palette='viridis')
plt.title('Número de Livros por Gênero')
plt.xlabel('Gênero')
plt.ylabel('Qta de Livros')
```

```
plt.xticks(rotation=45)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()

plt.show()
```

Correlação entre Quantidade de livros e empréstimos mensais: 0.05833825903205008 <ipython-input-15-eba4004eefa9>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect. sns.barplot(x='Genero', y='Qta_Livros', data=tendencia, palette='viridis')



Modelagem Estatística

1. Técnicas como regressão linear, análise de variância, etc

```
pip install pandas
```

```
import pandas as pd
import statsmodels.api as sm
from scipy.stats import f_oneway
```

```
dados = {
    'Genero': ['Guerra','Ficticio', 'História', 'Geografia', 'Biografia',],
    'Qta_Livros': [1500, 2800, 2000, 1000, 450],
    'Usuarios_Ativos': [1237, 536, 400, 480, 278],
    'Empréstimos_Mensais': [800, 400, 150, 460, 200]
}
```

```
df = pd.DataFrame(dados)
```

```
X = [1237, 536, 400, 480, 278]  # Usuários Ativosz
y = [800, 400, 150, 460, 200]   # Empréstimos Mensais
```

```
# Adicionando uma constante ao X para estimar o intercepto
X = sm.add_constant(X)
```

```
# Criando e ajustando o modelo de regressão linear
modelo = sm.OLS(y, X)
resultados = modelo.fit()

# Realizando a análise de variância (ANOVA)
anova_resultado=f_oneway(df['Usuarios_Ativos'], df['Empréstimos_Mensais'])

# Imprimindo os resultados da regressão
print(resultados.summary())

# Imprimindo o p-valor da ANOVA
print("P-valor da ANOVA:", anova_resultado.pvalue)
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.869
Model:	OLS	Adj. R-squared:	0.826
Method:	Least Squares	F-statistic:	19.95
Date:	Sun, 12 May 2024	Prob (F-statistic):	0.0209
Time:	17:04:49	Log-Likelihood:	-29.213
No. Observations:	5	AIC:	62.43
Df Residuals:	3	BIC:	61.65
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	27.5970	96.675	0.285	0.794	-280.065	335.259
x1	0.6387	0.143	4.466	0.021	0.184	1.094

Omnibus:	nan	Durbin-Watson:	3.251
Prob(Omnibus):	nan	Jarque-Bera (JB):	0.104
Skew:	-0.118	Prob(JB):	0.950
Kurtosis:	2.336	Cond. No.	1.36e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.36e+03. This might indicate that there are strong multicollinearity or other numerical problems.

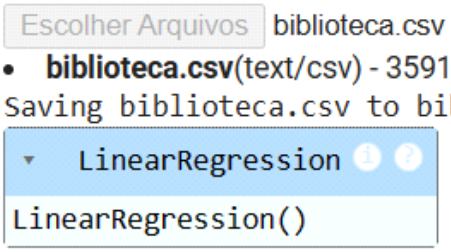
P-valor da ANOVA: 0.3931374435401579
/usr/local/lib/python3.10/dist-packages/statsmodels/stats/stattools.py:74:
ValueWarning: omni_normtest is not valid with less than 8 observations; 5 samples were given.
  warn("omni_normtest is not valid with less than 8 observations; %i "

2. Aplique modelos estatísticos avançados

```
pip install pandas
```

```
pip install -U scikit-learn
```

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from google.colab import files

uploaded = files.upload()
filename = next(iter(uploaded))
df = pd.read_csv(filename)

X = df[['ID']]
y = df['TELEFONE']

modelo = LinearRegression()
modelo.fit(X, y)
```



3. Avaliação a qualidade do modelo ajustado aos dados

```
import numpy as np

import statsmodels.api as sm


Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']

Qta_Livros = [1500, 2800, 2000, 1000, 450]

Usuarios_Ativos = [1237, 536, 400, 480, 278]

Empréstimos_Mensais = [800, 400, 150, 460, 200]



X = np.column_stack((Qta_Livros, Usuarios_Ativos, Empréstimos_Mensais))

y = np.array([1237, 536, 400, 480, 278])  # se eu quiser  prever os Usuarios_Ativos


X = sm.add_constant(X) # primeiramente adicionamos uma costante de termos de intercepção


modelo = sm.OLS(y, X).fit() # depois ajustamos o modelo OLS


coeficientes = modelo.params # apresentando o modelo de coeficiente


R2 = modelo.rsquared # coeficiente de determinação - r²


print("Coeficientes:") # resultados obtidos
```

```
print(coeficientes)
print("\nCoeficiente de Determinação (R-squared):")
print(R2)
```

```
Coeficientes:
[-4.52748949e-13  2.37765536e-16  1.00000000e+00  1.34766330e-15]

Coeficiente de Determinação (R-squared):
1.0
```

4. Desenvolva modelos preditivos com base nas análises estatísticas.

```
import numpy as np
from sklearn.linear_model import LinearRegression

Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']
Qta_Livros = [1500, 2800, 2000, 1000, 450]
Empréstimos_Mensais = [800, 400, 150, 460, 200]
Usuarios_Ativos = [1237, 536, 400, 480, 278]

X = np.column_stack((Qta_Livros, Empréstimos_Mensais))
modelo = LinearRegression().fit(X, Usuarios_Ativos)

novos_dados = np.array([[1400, 1500], [2800, 800]])  # dados novos sugeridos(
2 dados observados)

previsoes = modelo.predict(novos_dados)
```

```
print("Previsões usando o modelo preditivo:")
print(previsoes)
```

```
Previsões usando o modelo preditivo:
[2064.05565371 1179.41945365]
```

Os resultados mostram que:

A primeira previsão é de aproximadamente 2064 usuários ativos.

A segunda previsão é de aproximadamente 1179 usuários ativos.

5. Avaliação da performance dos modelos preditivos

```
from sklearn.metrics import mean_squared_error

Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']
Qta_Livros = [1500, 2800, 2000, 1000, 450]
Empréstimos_Mensais = [800, 400, 150, 460, 200]
Usuarios_Ativos = [1237, 536, 400, 480, 278]

y_verdadeiro = [1237, 536] #dados do Usuários_Ativos

previsoes = [2064.05565371, 1179.41945365]  #resultados do Exemplo de previsões

erro_medio_quadratico = mean_squared_error(y_verdadeiro, previsoes)
print(f'erro_medio_quadratico: {erro_medio_quadratico}')
```

```
erro_medio_quadratico: 549004.82383447
```

Observa-se diferenças entre os valores verdadeiros e as previsões são de aproximadamente 549004.82383447. Isso pode ser interpretado como uma indicação do quão bem o modelo está se ajustando aos dados observados. Um valor menor de MSE indicaria uma melhor correspondência entre as previsões do modelo e os dados reais.

6. Comparação entre diferentes abordagens de análise preditiva:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error

Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']
Qta_Livros = [1500, 2800, 2000, 1000, 450]
Empréstimos_Mensais = [800, 400, 150, 460, 200]
Usuarios_Ativos = [1237, 536, 400, 480, 278]

Qta_Livros = np.array([1500, 2800, 2000, 1000, 450])
Empréstimos_Mensais = np.array([800, 400, 150, 460, 200])
Usuarios_Ativos = np.array([1237, 536, 400, 480, 278])

X = np.column_stack((Qta_Livros, Empréstimos_Mensais))          #Conjunto de treinamentos e testes
```

```
y = Usuarios_Ativos

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


modelo_linear = LinearRegression()      #Modelos1 de regressão

modelo_linear.fit(X_train, y_train)

previsoes_linear = modelo_linear.predict(X_test)

mse_linear = mean_squared_error(y_test, previsoes_linear)


poly = PolynomialFeatures(degree=2)     #Modelo2 de regressão

X_poly = poly.fit_transform(X_train)

modelo_poly = LinearRegression()

modelo_poly.fit(X_poly, y_train)

X_test_poly = poly.transform(X_test)

previsoes_poly = modelo_poly.predict(X_test_poly)

mse_poly = mean_squared_error(y_test, previsoes_poly)


print("resultado1 (modelo_1):", mse_linear)  #Comparando os modelos

print("resultado2 (modelo_2):", mse_poly)

resultado1 (modelo_1): 101790.20340799385

resultado2 (modelo_2): 261460.66467407814


if mse_linear < mse_poly:

    print('modelo 1 é melhor.')   # Resultado do maia eficaz

else:
```

```
    print('modelo 2 é melhor.')
```

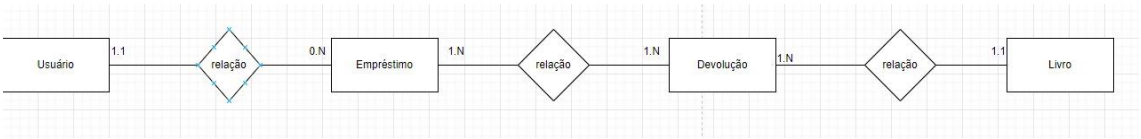
```
modelo 1 é melhor.
```

Com base nos resultados do erro médio quadrático (MSE):

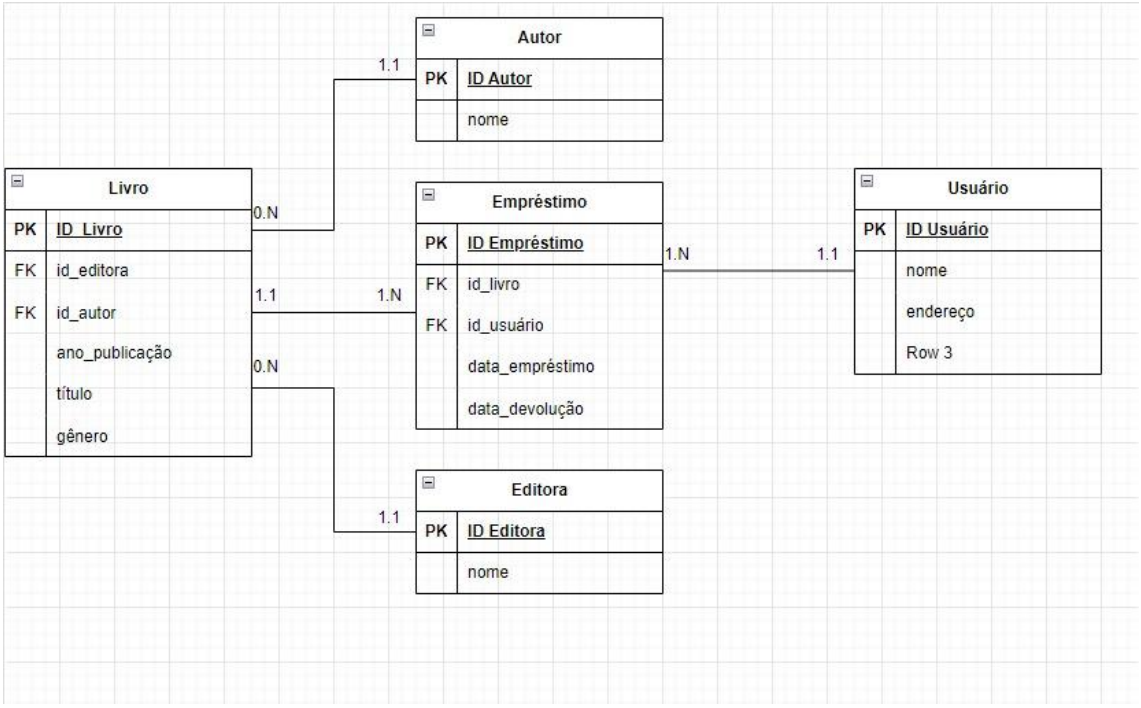
Podemos concluir que o Modelo 1 foi o melhor, pois possui um erro medio quadratico menor, indicando um melhor ajuste aos dados de teste comparando com o Modelo 2.

Modelagem de Banco de dados

1 – Modelo Conceitual



2 – Modelo Lógico



2.1 - Normalização

Livro					
id_livro	id_editora	id_autor	ano_publica	título	gênero
1	3	5	1968	Romeu e Julieta	Romance
2	4	6	2012	Garota Exempla	Suspense

Autor	
id_autor	nome
5	William Shakes
6	Gillian Flynn

Editora	
id_editora	nome
3	Bruna
4	Maiara

Usuário		
id_usuário	nome	endereço
7	Amãni	##########
8	Yarla	##########

Empréstimo				
id_empréstimo	id_livro	id_usuário	data_emprésti	data_devolu
9	1	7	01/02/2024	01/03/2024
10	2	8	02/03/2024	02/04/2024

3 – Simulação de Cadastro

Pedido

Atributo	Tipo	Descrição	Chave
ID_pedido	Inteiro	Identificador único do pedido	Primária
Data	_	Data de realização do pedido	_
ID_usuário	Inteiro	Chave estrangeira para Usuário. ID_usuário	Estrangeira

Cliente

Atributo	Tipo	Descrição	Chave
ID_Usuário	Inteiro	Identificador único do cliente	Primária
Nome	Varchar(50)	Nome do usuário	_
Telefone	Varchar(50)	Telefone de cadastro do usuário	_

Registro banco

ID_pedido	Data	ID_usuário
7	01/02/2024	7
8	02/03/2024	8

Redes de computadores

Rede World Library

Equipamentos:

- Computadores para funcionários
- Impressoras
- Roteador
- Switches
- Cabos de rede (Ethernet)

Departamentos:

- Biblioteca
- Administração
- Almoxarifado

Classe de Rede:

- Classe C: com capacidade para até 254 dispositivos por sub-rede.

Padrão de Rede por Departamento:

Biblioteca:

Rede cabeada:

- Oferece maior confiabilidade e velocidade para as atividades que exigem grande volume de dados, como download de livros eletrônicos e acesso a bancos de dados.
- Faixa de IP: 192.168.1.0/24
- Máscara de sub-rede: 255.255.255.0

Rede Wi-Fi:

- Permite acesso à internet para usuários que necessitem de mobilidade, como visitantes e pesquisadores.
- SSID: WorldLibrary-GRATUITO
- Senha: NAOTEMSENHA
- Faixa de IP: 192.168.1.100/24
- Máscara de sub-rede: 255.255.255.0

Administração:

Rede cabeada:

- Garante segurança e estabilidade para o acesso a informações confidenciais e sistemas administrativos
- Faixa de IP: 192.168.2.0/24
- Máscara de sub-rede: 255.255.255.0

Almoxarifado:

Rede cabeada:

- Facilita o controle de estoque e a gestão de processos logísticos.
- Faixa de IP: 192.168.3.0/24
- Máscara de sub-rede: 255.255.255.0

Considerações Adicionais:

Segurança:

- Implementar firewall para proteger a rede contra acessos não autorizados e ataques cibernéticos.
- Utilizar criptografia para garantir a confidencialidade dos dados transmitidos.
- Criar políticas de senha robustas e conscientizar os usuários sobre boas práticas de segurança.

Monitoramento:

- Implementar um sistema de monitoramento de rede para identificar e solucionar problemas de forma proativa.
- Monitorar o tráfego de rede para otimizar o uso da banda larga.

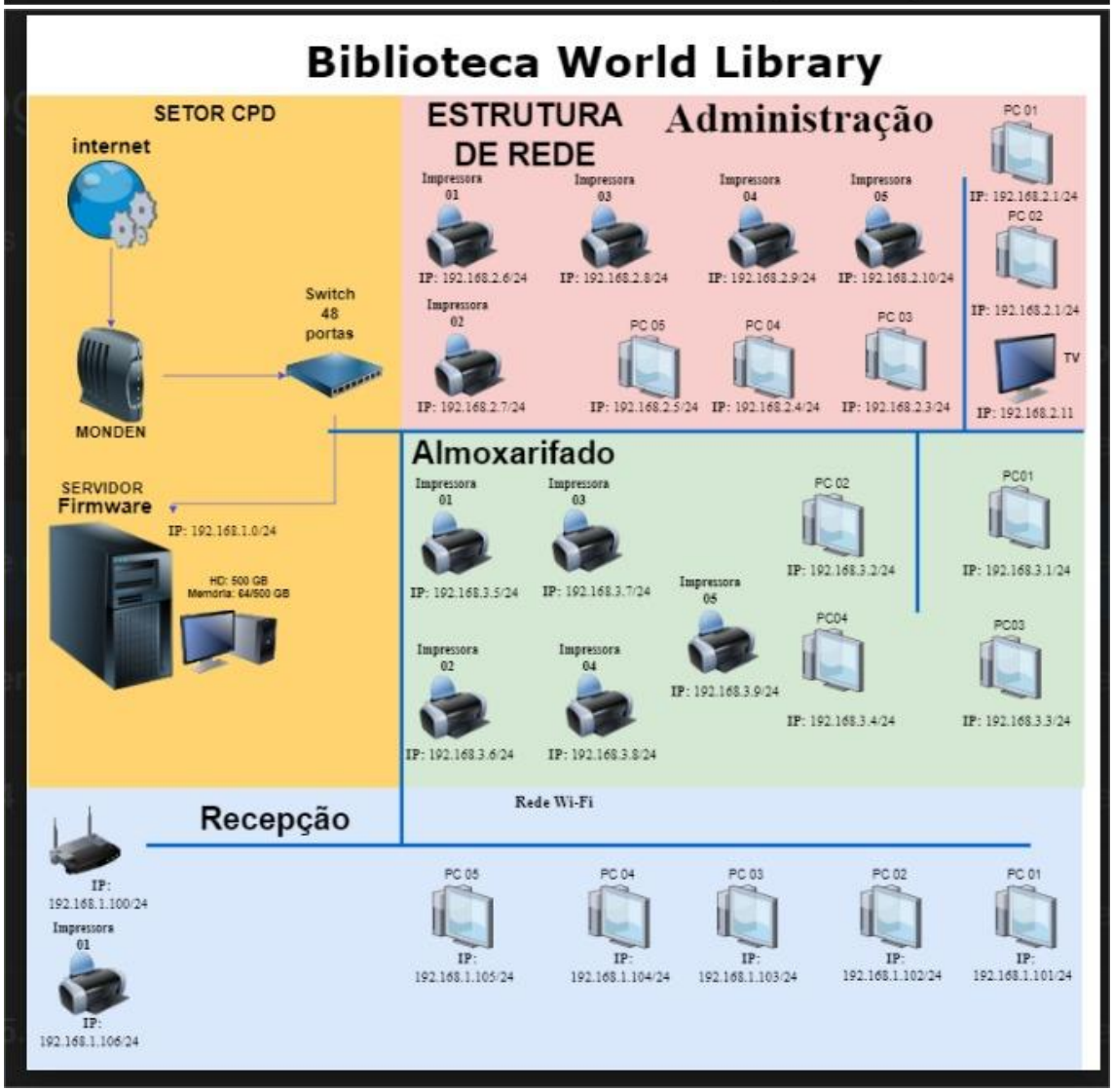
Gerenciamento:

- Utilizar um software de gerenciamento de rede para facilitar a configuração, o monitoramento e a solução de problemas.
- Documentar a configuração da rede e os procedimentos de manutenção para facilitar o trabalho dos administradores de rede.

Recomendações necessárias futuras:

- Realizar treinamentos para os funcionários sobre como utilizar a rede de forma eficiente e segura.
- Manter a rede atualizada com as últimas patches de segurança e firmware para garantir a proteção contra vulnerabilidades.

Planta Baixa



Segurança da Informação

Análise de Riscos

- Identificação e Avaliação de Riscos: Precisamos considerar questões como perda de dados acessa não autorizada a informações dos usuários, e possíveis interrupções no sistema que poderiam afetar os serviços da biblioteca.
- Avaliação de Vulnerabilidades: Devemos examinar se o sistema tem falhas de segurança que poderiam permitir o acesso não autorizado ou comprometimento dos dados.
- Ameaças Potenciais: Isso inclui desde ataques cibernéticos até roubo físico de dispositivos que contenham informações sensíveis.

Implementação de Medidas de Segurança

Implementação de Políticas de Controle de Acesso:

- Devemos estabelecer políticas claras sobre quem pode acessar o catálogo de livros, informações dos usuários e dados administrativos da biblioteca.
- Autenticação Forte: É importante garantir que os usuários só possam acessar o sistema através de credenciais seguras, como senhas fortes ou cartões de acesso.
- Configuração de Sistemas de Detecção e Prevenção de Intrusões: Devemos configurar sistemas de detecção de intrusões para identificar padrões suspeitos de atividade que possam indicar tentativas de acesso não autorizado.
- Prevenção de Ataques: Isso inclui medidas como firewalls, criptografia de dados e atualizações regulares de segurança para proteger o sistema contra malware, phishing e outras ameaças cibernéticas.
- Testes de Penetração: Devemos realizar testes regulares para identificar e corrigir quaisquer vulnerabilidades no sistema antes que elas possam ser exploradas por invasores.