UNINOVE – Campus Santo Amaro

Ana Carolina Carvalho Pereira

RA: 2222101992

Renato Roberto Reis Lopes

RA:2222104482

Pedro Henrique Meireles Nascimento de Oliveira

RA: 2222202321

Murilo Dias Dos Santos

RA:2322101573

PLANEJAMENTO EM INFORMÁTICA TURMA 40 – NOTURNO

Professor Felipe Santos de Jesus

SÃO PAULO

2024

ÍNDICE

Escopo do Projeto:

O presente documento descreve em detalhes o escopo do projeto proposto pela empresa TechCode.

O objetivo principal deste projeto é a criação de uma plataforma online inovadora e acessível para oferecer cursos relacionados à tecnologia. A plataforma será projetada visando proporcionar uma experiência de aprendizado envolvente e de alta qualidade, abrangendo uma variedade de tópicos e níveis de conhecimento. O foco será na entrega de conteúdo atualizado e relevante, aliado a recursos interativos e ferramentas de acompanhamento do progresso do aluno. O desenvolvimento da plataforma TechCode busca não apenas democratizar o acesso à educação tecnológica, mas também estimular o crescimento profissional e pessoal dos usuários, preparando-os para os desafios e oportunidades do mercado digital.

1. Serviços Oferecidos:

Na TechCode, estamos empenhados em fornecer a você as ferramentas necessárias para se destacar no mundo da tecnologia. Como pioneiros na área de educação online, oferecemos uma ampla gama de cursos projetados para impulsionar sua carreira e expandir seus conhecimentos no universo digital.

Nossos serviços abrangem uma variedade de áreas essenciais, incluindo:

Python Básico: Aprenda os fundamentos da linguagem de programação Python, construa uma base sólida para seus projetos futuros e mergulhe no emocionante mundo da codificação.

Data Science: Descubra como transformar dados em insights valiosos, dominando técnicas avançadas de análise e visualização de dados.

Machine Learning: Explore os princípios do aprendizado de máquina e descubra como criar modelos preditivos inteligentes para resolver problemas do mundo real.

Desenvolvimento Web: Aprofunde-se no desenvolvimento web e adquira habilidades essenciais em HTML, CSS, JavaScript e muito mais, capacitando-se para criar websites dinâmicos e responsivos.

SQL: Domine a linguagem SQL e aprenda a gerenciar e manipular bancos de dados de forma eficaz, preparando-se para trabalhar com sistemas de gerenciamento de banco de dados populares.

Rede de Computadores: Explore os conceitos fundamentais de redes de computadores, desde a arquitetura básica até a configuração e manutenção de redes empresariais.

Na TechCode, nosso compromisso é oferecer a você uma experiência de aprendizado enriquecedora e personalizada. Nossos instrutores especializados estão aqui para guiá-lo em cada passo do caminho, garantindo que você alcance seus objetivos com sucesso.

```python
import pandas as pd

# Dados fictícios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5],
    'Nome_Curso': ['Python Básico', 'Data Science', 'Machine Learning', 'Web Development', 'Python Básico'],
    'Data_Inscricao': ['2024-01-10', '2024-02-15', '2024-03-20', '2024-01-10', '2024-04-25'],
    'Avaliacao_Curso': [4.5, 4.7, 4.8, 4.6, None],  # Note que há um valor ausente
    'Valor_Pago': [500, 1500, 2000, 1000, 0]  # Note que há um valor inconsistente
}

# Criar um DataFrame
df = pd.DataFrame(dados_ficticios)

# Salvar em um arquivo CSV
df.to_csv('dados_empresa.csv', index=False)
```

```python
# Carregar dados do arquivo CSV
dados = pd.read_csv('dados_empresa.csv')
print(dados)
```

```
   ID_Aluno        Nome_Curso Data_Inscricao  Avaliacao_Curso  Valor_Pago
0         1     Python Básico     2024-01-10              4.5         500
1         2      Data Science     2024-02-15              4.7        1500
2         3  Machine Learning     2024-03-20              4.8        2000
3         4   Web Development     2024-01-10              4.6        1000
4         5     Python Básico     2024-04-25              NaN           0
```

```python
# Tratar valores ausentes, preenchendo com a média das avaliações
media_avaliacao = dados['Avaliacao_Curso'].mean()
dados['Avaliacao_Curso'].fillna(media_avaliacao, inplace=True)
```

```python
# Remover outliers e valores inconsistentes no campo 'Valor_Pago'
# Por exemplo, remover valores zero ou negativos
dados = dados[dados['Valor_Pago'] > 0]

# Outra forma de tratar valores baixos (assumindo que valores muito baixos sejam inválidos)
dados['Valor_Pago'] = dados['Valor_Pago'].apply(lambda x: x if x > 10 else media_avaliacao * 100)
```

```
<ipython-input-21-45fa45b10b16>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dados['Valor_Pago'] = dados['Valor_Pago'].apply(lambda x: x if x > 10 else media_avaliacao * 100)
```

```python
import pandas as pd

# Dados ficticios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5],
    'Nome_Curso': ['Python Básico', 'Data Science', 'Machine Learning', 'Web Development', 'Python Básico'],
    'Data_Inscricao': ['2024-01-10', '2024-02-15', '2024-03-20', '2024-01-10', '2024-04-25'],
    'Avaliacao_Curso': [4.5, 4.7, 4.8, 4.6, None],  # Note que há um valor ausente
    'Valor_Pago': [500, 1500, 2000, 1000, 0]  # Note que há um valor inconsistente
}

# Criar um DataFrame
df = pd.DataFrame(dados_ficticios)

# Salvar em um arquivo CSV
df.to_csv('dados_empresa.csv', index=False)
```

```python
# Carregar dados do arquivo CSV
dados = pd.read_csv('dados_empresa.csv')
print(dados)
```

```
   ID_Aluno        Nome_Curso Data_Inscricao  Avaliacao_Curso  Valor_Pago
0         1     Python Básico     2024-01-10              4.5         500
1         2      Data Science     2024-02-15              4.7        1500
2         3  Machine Learning     2024-03-20              4.8        2000
3         4   Web Development     2024-01-10              4.6        1000
4         5     Python Básico     2024-04-25              NaN           0
```

```python
# Tratar valores ausentes, preenchendo com a média das avaliações
media_avaliacao = dados['Avaliacao_Curso'].mean()
dados['Avaliacao_Curso'].fillna(media_avaliacao, inplace=True)
```

```python
# Remover outliers e valores inconsistentes no campo 'Valor_Pago'
# Por exemplo, remover valores zero ou negativos
dados = dados[dados['Valor_Pago'] > 0]

# Outra forma de tratar valores baixos (assumindo que valores muito baixos sejam inválidos)
dados['Valor_Pago'] = dados['Valor_Pago'].apply(lambda x: x if x > 10 else media_avaliacao * 100)
```

```
<ipython-input-21-45fa45b10b16>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dados['Valor_Pago'] = dados['Valor_Pago'].apply(lambda x: x if x > 10 else media_avaliacao * 100)
```

```
Matriz de Confusão:
[[2 0]
 [1 3]]
Acurácia: 0.83
Precisão: 1.00
Recall: 0.75
F1-Score: 0.86
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
```

```python
# Dados fictícios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Horas_Estudo': [10, 20, 15, 30, 25, 10, 20, 30, 40, 5],
    'Participacao_Forum': [1, 2, 1, 3, 2, 1, 2, 3, 3, 0],
    'Passou': [0, 1, 0, 1, 1, 0, 1, 1, 1, 0]  # Classe binária
}

# Criar DataFrame
df = pd.DataFrame(dados_ficticios)

# Separar características (X) e rótulos (y)
X = df[['Horas_Estudo', 'Participacao_Forum']]
y = df['Passou']

# Dividir 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)
# Escolher e treinar o modelo
modelo = RandomForestClassifier(random_state=42)
modelo.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = modelo.predict(X_test)

# Calcular a matriz de confusão
matriz_confusao = confusion_matrix(y_test, y_pred)
print("Matriz de Confusão:")
print(matriz_confusao)

# Calcular precisão, recall e F1-score
precisao = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Precisão: {precisao:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
```

```
   ID_Aluno        Nome_Curso Data_Inscricao  Avaliacao_Curso  Valor_Pago
0         1     Python Básico     2024-01-10              4.5      500000
1         2      Data Science     2024-02-15              4.7        1500
2         3  Machine Learning     2024-03-20              4.8        2000
3         4   Web Development     2024-01-10              4.6        1000
```

```python
from sklearn.metrics import confusion_matrix

# Valores reais
y_true = [1, 0, 1, 1, 0, 1]

# Valores previstos pelo classificador
y_pred = [1, 0, 1, 0, 0, 1]

# Calcular a matriz de confusão
matriz_confusao = confusion_matrix(y_true, y_pred)

# Exibir a matriz de confusão
print("Matriz de Confusão:")
print(matriz_confusao)

# Interpretação da Matriz de Confusão
# True Negative (TN): 2 (y_true=0 e y_pred=0)
# False Positive (FP): 0 (y_true=0 e y_pred=1)
# False Negative (FN): 1 (y_true=1 e y_pred=0)
# True Positive (TP): 3 (y_true=1 e y_pred=1)

# Calcular métricas a partir da matriz de confusão
tn, fp, fn, tp = matriz_confusao.ravel()

# Acurácia
acuracia = (tp + tn) / (tp + tn + fp + fn)
print(f"Acurácia: {acuracia:.2f}")

# Precisão
precisao = tp / (tp + fp) if (tp + fp) != 0 else 0
print(f"Precisão: {precisao:.2f}")

# Recall (Sensibilidade)
recall = tp / (tp + fn) if (tp + fn) != 0 else 0
print(f"Recall: {recall:.2f}")

# F1-Score
f1_score = 2 * (precisao * recall) / (precisao + recall) if (precisao + recall) != 0 else 0
print(f"F1-Score: {f1_score:.2f}")
```

```python
matriz_confusao = confusion_matrix(y_test, y_pred)
print("Matriz de Confusão:")
print(matriz_confusao)

# Calcular precisão, recall e F1-score
precisao = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Precisão: {precisao:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

# Dados fictícios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Horas_Estudo': [10, 20, 15, 30, 25, 10, 20, 30, 40, 5],
    'Participacao_Forum': [1, 2, 1, 3, 2, 1, 2, 3, 3, 0],
    'Passou': [0, 1, 0, 1, 1, 0, 1, 1, 1, 0]  # Classe binária
}

# Criar DataFrame
df = pd.DataFrame(dados_ficticios)

# Separar características (X) e rótulos (y)
X = df[['Horas_Estudo', 'Participacao_Forum']]
y = df['Passou']

# Dividir 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)

# Escolher e treinar o modelo
modelo = RandomForestClassifier(random_state=42)
modelo.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = modelo.predict(X_test)

# Calcular a matriz de confusão
matriz_confusao = confusion_matrix(y_test, y_pred)
print("Matriz de Confusão:")
print(matriz_confusao)

# Calcular precisão, recall e F1-score
precisao = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Precisão: {precisao:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Matriz de Confusão:
[[2]]
Precisão: 1.00
Recall: 1.00
F1-Score: 1.00
Matriz de Confusão:
[[2]]
Precisão: 1.00
Recall: 1.00
F1-Score: 1.00
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score

# Dados ficticios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Horas_Estudo': [10, 20, 15, 30, 25, 10, 20, 30, 40, 5],
    'Participacao_Forum': [1, 2, 1, 3, 2, 1, 2, 3, 3, 0],
    'Passou': [0, 1, 0, 1, 1, 0, 1, 1, 1, 0]  # Classe binária
}

# Criar DataFrame
df = pd.DataFrame(dados_ficticios)

# Separar caracteristicas (X) e rótulos (y)
X = df[['Horas_Estudo', 'Participacao_Forum']]
y = df['Passou']

# Dividir 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, stratify=y)

# Definir os hiperparâmetros para otimização
parametros = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Configurar o GridSearchCV com StratifiedKFold
cv = StratifiedKFold(n_splits=3)  # Ajuste o número de splits conforme necessário
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42), param_grid=parametros, cv=cv, n_jobs=-1, verbose=2)
```

```python
# Configurar o GridSearchCV com StratifiedKFold
cv = StratifiedKFold(n_splits=3)  # Ajuste o número de splits conforme necessário
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42), param_grid=parametros, cv=cv, n_jobs=-1, verbose=2)

# Executar o Grid Search no conjunto de treino
grid_search.fit(X_train, y_train)

# Melhor conjunto de hiperparâmetros
best_params = grid_search.best_params_
print(f"Melhores hiperparâmetros: {best_params}")

# Treinar o modelo com os melhores hiperparâmetros
modelo_otimizado = RandomForestClassifier(**best_params, random_state=42)
modelo_otimizado.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred_otimizado = modelo_otimizado.predict(X_test)

# Calcular a matriz de confusão
matriz_confusao_otimizada = confusion_matrix(y_test, y_pred_otimizado)
print("Matriz de Confusão após Otimização:")
print(matriz_confusao_otimizada)

# Calcular precisão, recall e F1-score
precisao_otimizada = precision_score(y_test, y_pred_otimizado)
recall_otimizada = recall_score(y_test, y_pred_otimizado)
f1_otimizada = f1_score(y_test, y_pred_otimizado)

print(f"Precisão após Otimização: {precisao_otimizada:.2f}")
print(f"Recall após Otimização: {recall_otimizada:.2f}")
print(f"F1-Score após Otimização: {f1_otimizada:.2f}")

# Validar o modelo usando validação cruzada
pontuacoes = cross_val_score(modelo_otimizado, X, y, cv=cv)

print(f"Pontos da Validação Cruzada: {pontuacoes}")
print(f"Média da Validação Cruzada: {pontuacoes.mean():.2f}")
```

```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
Melhores hiperparâmetros: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Matriz de Confusão após Otimização:
[[1 0]
 [0 1]]
Precisão após Otimização: 1.00
Recall após Otimização: 1.00
```

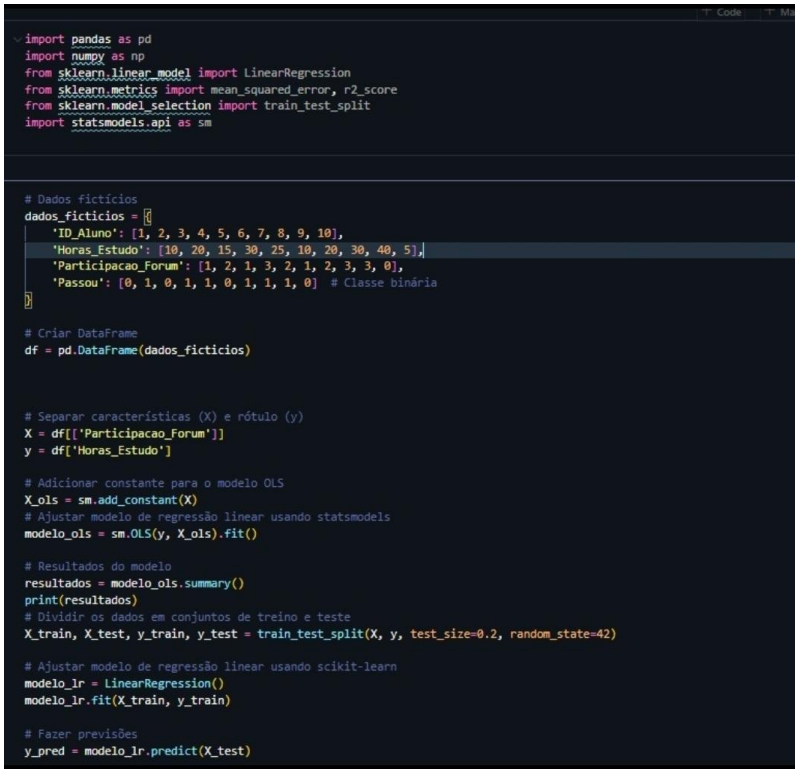
```python
# Calcular a matriz de confusão
matriz_confusao_otimizada = confusion_matrix(y_test, y_pred_otimizado)
print("Matriz de Confusão após Otimização:")
print(matriz_confusao_otimizada)

# Calcular precisão, recall e F1-score
precisao_otimizada = precision_score(y_test, y_pred_otimizado)
recall_otimizada = recall_score(y_test, y_pred_otimizado)
f1_otimizada = f1_score(y_test, y_pred_otimizado)

print(f"Precisão após Otimização: {precisao_otimizada:.2f}")
print(f"Recall após Otimização: {recall_otimizada:.2f}")
print(f"F1-Score após Otimização: {f1_otimizada:.2f}")

# Validar o modelo usando validação cruzada
pontuacoes = cross_val_score(modelo_otimizado, X, y, cv=cv)

print(f"Pontos da Validação Cruzada: {pontuacoes}")
print(f"Média da Validação Cruzada: {pontuacoes.mean():.2f}")
```

```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
Melhores hiperparâmetros: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Matriz de Confusão após Otimização:
[[1 0]
 [0 1]]
Precisão após Otimização: 1.00
Recall após Otimização: 1.00
F1-Score após Otimização: 1.00
Pontos da Validação Cruzada: [1. 1. 1.]
Média da Validação Cruzada: 1.00
```

```python
# Padronizar a coluna 'Valor_Pago' em mil unidades se estiver abaixo de 1000
dados['Valor_Pago'] = dados['Valor_Pago'].apply(lambda x: x * 1000 if x < 1000 else x)

# Converter a coluna 'Data_Inscricao' para o formato de data
dados['Data_Inscricao'] = pd.to_datetime(dados['Data_Inscricao'])

# Exibir os dados limpos
print(dados)
```

```
   ID_Aluno        Nome_Curso Data_Inscricao  Avaliacao_Curso  Valor_Pago
0         1     Python Básico     2024-01-10              4.5      500000
1         2      Data Science     2024-02-15              4.7        1500
2         3  Machine Learning     2024-03-20              4.8        2000
3         4   Web Development     2024-01-10              4.6        1000
```

```python
import pandas as pd

# Dados fictícios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5],
    'Nome_Curso': ['Python Básico', 'Data Science', 'Machine Learning', 'Web Development', 'Python Básico'],
    'Data_Inscricao': ['2024-01-10', '2024-02-15', '2024-03-20', '2024-01-10', '2024-04-25'],
    'Avaliacao_Curso': [4.5, 4.7, 4.8, 4.6, None],  # Note que há um valor ausente
    'Valor_Pago': [500, 1500, 2000, 1000, 0]  # Note que há um valor inconsistente
}

# Criar um DataFrame
df = pd.DataFrame(dados_ficticios)

# Salvar em um arquivo CSV
df.to_csv('dados_empresa.csv', index=False)

# Carregar dados do arquivo CSV
dados = pd.read_csv('dados_empresa.csv')

# Tratar valores ausentes, preenchendo com a média das avaliações
media_avaliacao = dados['Avaliacao_Curso'].mean()
dados['Avaliacao_Curso'].fillna(media_avaliacao, inplace=True)

# Remover outliers e valores inconsistentes no campo 'Valor_Pago'
# Por exemplo, remover valores zero ou negativos
dados = dados[dados['Valor_Pago'] > 0]

# Outra forma de tratar valores baixos (assumindo que valores muito baixos sejam inválidos)
dados['Valor_Pago'] = dados['Valor_Pago'].apply(lambda x: x if x > 10 else media_avaliacao * 100)

# Padronizar a coluna 'Valor_Pago' em mil unidades se estiver abaixo de 1000
dados['Valor_Pago'] = dados['Valor_Pago'].apply(lambda x: x * 1000 if x < 1000 else x)

# Converter a coluna 'Data_Inscricao' para o formato de data
dados['Data_Inscricao'] = pd.to_datetime(dados['Data_Inscricao'])
```



```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import statsmodels.api as sm


# Dados fictícios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Horas_Estudo': [10, 20, 15, 30, 25, 10, 20, 30, 40, 5],
    'Participacao_Forum': [1, 2, 1, 3, 2, 1, 2, 3, 3, 0],
    'Passou': [0, 1, 0, 1, 1, 0, 1, 1, 1, 0]  # Classe binária
}

# Criar DataFrame
df = pd.DataFrame(dados_ficticios)



# Separar características (X) e rótulo (y)
X = df[['Participacao_Forum']]
y = df['Horas_Estudo']

# Adicionar constante para o modelo OLS
X_ols = sm.add_constant(X)
# Ajustar modelo de regressão linear usando statsmodels
modelo_ols = sm.OLS(y, X_ols).fit()

# Resultados do modelo
resultados = modelo_ols.summary()
print(resultados)
# Dividir os dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Ajustar modelo de regressão linear usando scikit-learn
modelo_lr = LinearRegression()
modelo_lr.fit(X_train, y_train)

# Fazer previsões
y_pred = modelo_lr.predict(X_test)
```

```python
resultados = modelo_ols.summary()
print(resultados)
# Dividir os dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Ajustar modelo de regressão linear usando scikit-learn
modelo_lr = LinearRegression()
modelo_lr.fit(X_train, y_train)

# Fazer previsões
y_pred = modelo_lr.predict(X_test)

# Avaliar modelo
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print(f'R-squared: {r2}')
print(f'Mean Squared Error: {mse}')


# Separar características (X) e rótulo (y)
X = df[['Horas_Estudo', 'Participacao_Forum']]
y = df['Passou']

# Dividir os dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.ensemble import RandomForestClassifier

# Ajustar modelo de classificação usando RandomForest
modelo_rf = RandomForestClassifier(random_state=42)
modelo_rf.fit(X_train, y_train)

# Fazer previsões
y_pred_rf = modelo_rf.predict(X_test)

# Avaliar modelo
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(y_test, y_pred_rf)
precision = precision_score(y_test, y_pred_rf)
recall = recall_score(y_test, y_pred_rf)
f1 = f1_score(y_test, y_pred_rf)
```

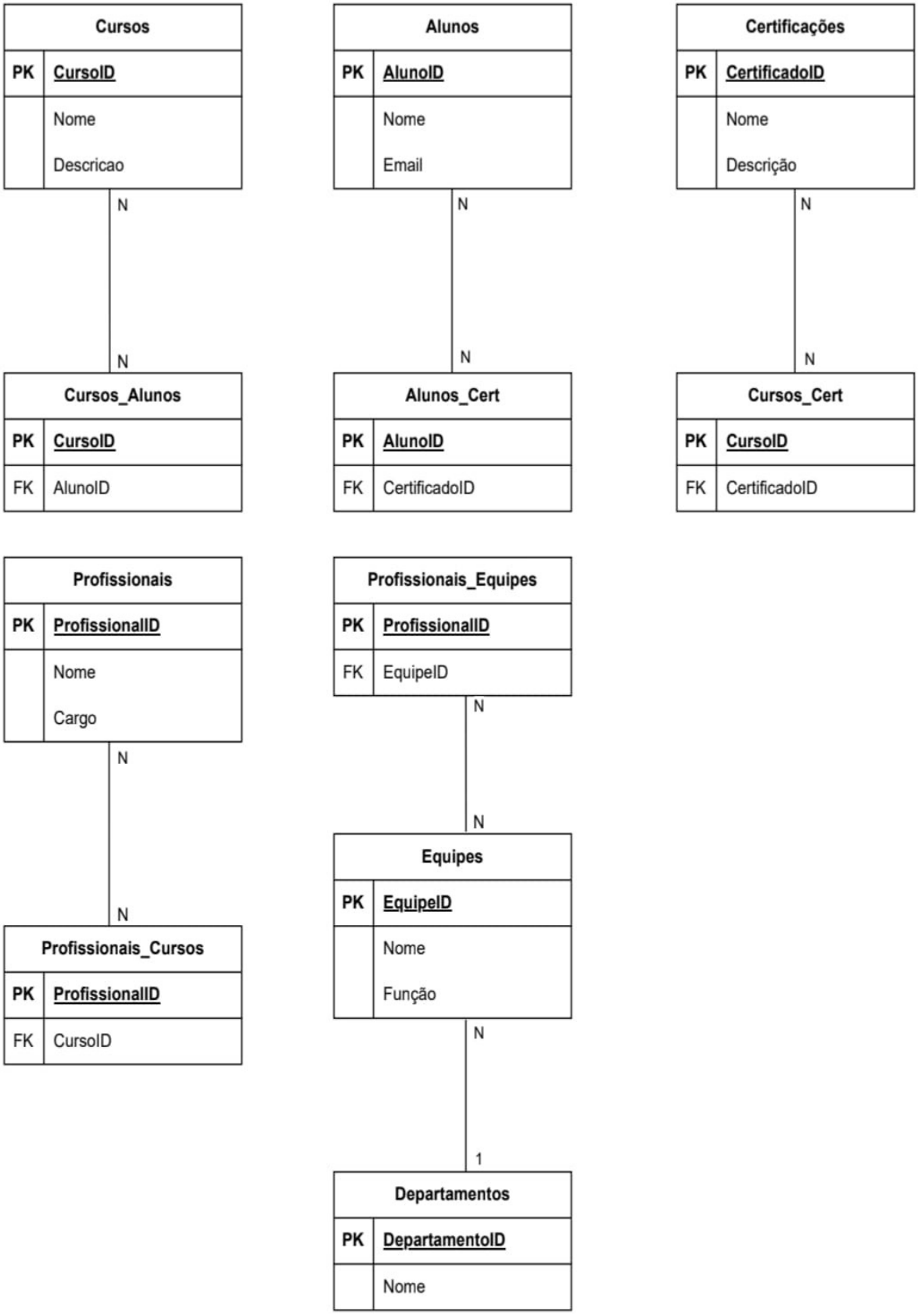
```
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

# Comparar R-squared e MSE da Regressão Linear
print(f'Regressão Linear - R-squared: {r2}, Mean Squared Error: {mse}')

# Comparar métricas da RandomForest
print(f'RandomForest Classificação - Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}, F1 Score: {f1}')

# Conclusão
if r2 > 0.7 and mse < 10:  # Supomos um limiar arbitrário para avaliar a adequação da regressão linear
    print('A Regressão Linear é adequada para prever as Horas de Estudo com base na Participação no Fórum.')
else:
    print('A Regressão Linear não é adequada para prever as Horas de Estudo com base na Participação no Fórum.')

if f1 > 0.7:  # Supomos um limiar arbitrário para avaliar a adequação da classificação
    print('A Classificação RandomForest é adequada para prever se um aluno passará ou não.')
else:
    print('A Classificação RandomForest não é adequada para prever se um aluno passará ou não.')

import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.ensemble import RandomForestClassifier

# Dados fictícios
dados_ficticios = {
    'ID_Aluno': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Horas_Estudo': [10, 20, 15, 30, 25, 10, 20, 30, 40, 5],
    'Participacao_Forum': [1, 2, 1, 3, 2, 1, 2, 3, 3, 0],
    'Passou': [0, 1, 0, 1, 1, 0, 1, 1, 1, 0]  # Classe binária
}

# Criar DataFrame
df = pd.DataFrame(dados_ficticios)

# Regressão Linear
# Separar características (X) e rótulo (y)
X = df[['Participacao_Forum']]
y = df['Horas_Estudo']
```

```
    # Separar características (X) e rótulo (y)
    X = df[['Participacao_Forum']]
    y = df['Horas_Estudo']

    # Adicionar constante para o modelo OLS
    X_ols = sm.add_constant(X)

    # Ajustar modelo de regressão linear usando statsmodels
    modelo_ols = sm.OLS(y, X_ols).fit()

    # Resultados do modelo
    resultados = modelo_ols.summary()
    print(resultados)

    # Dividir os dados em conjuntos de treino e teste
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Ajustar modelo de regressão linear usando scikit-learn
    modelo_lr = LinearRegression()
    modelo_lr.fit(X_train, y_train)

    # Fazer previsões
    y_pred = modelo_lr.predict(X_test)

    # Avaliar modelo
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)

    print(f'R-squared: {r2}')
    print(f'Mean Squared Error: {mse}')

    # Classificação
    # Separar características (X) e rótulo (y)
    X = df[['Horas_Estudo', 'Participacao_Forum']]
    y = df['Passou']

    # Dividir os dados em conjuntos de treino e teste
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Ajustar modelo de classificação usando RandomForest
    modelo_rf = RandomForestClassifier(random_state=42)
    modelo_rf.fit(X_train, y_train)

    # Fazer previsões
    y_pred_rf = modelo_rf.predict(X_test)
```

```python
# Ajustar modelo de classificação usando RandomForest
modelo_rf = RandomForestClassifier(random_state=42)
modelo_rf.fit(X_train, y_train)

# Fazer previsões
y_pred_rf = modelo_rf.predict(X_test)

# Avaliar modelo
accuracy = accuracy_score(y_test, y_pred_rf)
precision = precision_score(y_test, y_pred_rf)
recall = recall_score(y_test, y_pred_rf)
f1 = f1_score(y_test, y_pred_rf)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

# Comparação dos Resultados
print(f'Regressão Linear - R-squared: {r2}, Mean Squared Error: {mse}')
print(f'RandomForest Classificação - Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}, F1 Score: {f1}')

if r2 > 0.7 and mse < 10:
    print('A Regressão Linear é adequada para prever as Horas de Estudo com base na Participação no Fórum.')
else:
    print('A Regressão Linear não é adequada para prever as Horas de Estudo com base na Participação no Fórum')
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
                            OLS Regression Results
==============================================================================
Dep. Variable:          Horas_Estudo   R-squared:                       0.895
Model:                          OLS   Adj. R-squared:                  0.882
Method:               Least Squares   F-statistic:                     68.27
Date:              Mon, 27 May 2024   Prob (F-statistic):           3.46e-05
Time:                      00:48:52   Log-Likelihood:                -26.291
No. Observations:                10   AIC:                             56.58
Df Residuals:                     8   BIC:                             57.19
Df Model:                         1
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                 2.5000      2.480      1.008      0.343      -3.220       8.220
Participacao_Forum   10.0000      1.210      8.262      0.000       7.209      12.791
==============================================================================
Omnibus:                        3.332   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.189   Jarque-Bera (JB):                1.667
Skew:                           0.994   Prob(JB):                        0.435
Kurtosis:                       2.778   Cond. No.                         5.11
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
...
```

MODELAGEM DE BANCO DE DADOS



REDES DE COMPUTADORES

1. Montagem da Planta Baixa da Rede da Empresa:

Equipamentos e Departamentos:

1. Professores:

- **Computadores desktop/laptop**
- **Projetor (opcional, se necessário para apresentações online)**
- **Impressora (opcional, se necessário para impressão em casa)**

2. Gestores:

- **Computadores desktop/laptop**
- **Impressora (opcional, se necessário para impressão em casa)**

3. Desenvolvedores:

- **Estações de trabalho de alto desempenho**
- **Servidores de desenvolvimento (acessíveis remotamente)**
- **Dispositivos de armazenamento em rede (NAS ou serviços de armazenamento em nuvem)**
- **Impressora (opcional, se necessário para impressão em casa)**

4. Alunos:

- **Computadores desktop/laptop**
- **Tablets (opcional, para maior flexibilidade de acesso)**
- **Impressora (opcional, se necessário para impressão em casa)**

5. Financeiro/RH:

- **Computadores desktop/laptop**
- **Máquina de pagamento eletrônico (se aplicável)**
- **Impressora (opcional, se necessário para impressão em casa)**

6. Suporte/Segurança:

- **Computadores desktop/laptop**
- **Acesso remoto a switches, roteadores e servidores de segurança**
- **Ferramentas de gerenciamento remoto (para oferecer suporte técnico aos funcionários remotos)**

- **Câmeras de segurança (opcional, se necessário para monitoramento remoto)**

7. **Marketing:**

- **Computadores desktop/laptop**
- **Equipamentos de design gráfico (acessíveis remotamente, se necessário)**
- **Impressora (opcional, se necessário para impressão em casa)**

8. **Diretoria:**

- **Computadores desktop/laptop**
- **Videoconferência (software de videoconferência para reuniões virtuais)**
- **Servidor para plataforma de cursos online (acessível remotamente)**

2. Configuração de IP de Todos os Equipamentos:

Classe de Rede:
Uma Classe C ainda seria adequada para as necessidades de uma empresa de cursos online com funcionários em home office.
Padrão de Rede por Departamento:
Dada a natureza do home office, cada funcionário terá sua própria rede doméstica, geralmente gerenciada pelo roteador de sua residência. No entanto, para fins de organização e suporte técnico remoto, é possível estabelecer padrões para o endereçamento IP dos dispositivos:

- **Os endereços IP podem ser atribuídos dinamicamente pelos roteadores domésticos, utilizando o protocolo DHCP.**
- **A equipe de suporte/segurança pode configurar VPNs (Redes Privadas Virtuais) para permitir acesso seguro aos recursos da empresa a partir de redes domésticas dos funcionários.**
- **É importante implementar medidas de segurança, como autenticação forte e criptografia, para proteger o acesso remoto aos recursos da empresa.**

SEGURANÇA DA INFORMAÇÃO

Análise de Riscos

Identificação e avaliação dos riscos de segurança para a empresa:

- **Liste possíveis ameaças e vulnerabilidades**
- **Liste 20 possíveis ameaças e/ou vulnerabilidades**

Análise de vulnerabilidades e ameaças potenciais:

- **Avalie o impacto e a probabilidade de ocorrência de cada risco.**

Riscos Identificados para a Empresa Tech Code

1. Ameaça: Ataque de Phishing

- **Vulnerabilidade: Falta de treinamento em segurança para os funcionários**
- **Impacto: Médio**
- **Probabilidade: Alta**

2. Ameaça: Malware

- **Vulnerabilidade: Falta de atualização regular de software de segurança**
- **Impacto: Alto**
- **Probabilidade: Média**

3. Ameaça: Roubo de Dados Pessoais

- **Vulnerabilidade: Falta de criptografia adequada dos dados dos clientes**
- **Impacto: Alto**
- **Probabilidade: Alta**

4. Ameaça: Ataques DDoS

- **Vulnerabilidade: Infraestrutura de rede insuficiente**
- **Impacto: Alto**
- **Probabilidade: Média**

5. Ameaça: Violação de Privacidade

- **Vulnerabilidade: Políticas de privacidade fracas ou inexistentes**
- **Impacto: Alto**
- **Probabilidade: Média**

6. Ameaça: Acesso Não Autorizado

- **Vulnerabilidade: Uso de senhas fracas pelos usuários**
- **Impacto: Médio**
- **Probabilidade: Alta**

7. Ameaça: Engenharia Social

- **Vulnerabilidade: Falta de conscientização dos funcionários sobre engenharia social**
- **Impacto: Médio**
- **Probabilidade: Alta**

8. Ameaça: Perda de Dados

- **Vulnerabilidade: Ausência de backup regular e seguro**
- **Impacto: Alto**
- **Probabilidade: Média**

9. Ameaça: Ransomware

- **Vulnerabilidade: Falta de ferramentas de detecção e resposta a incidentes**
- **Impacto: Alto**
- **Probabilidade: Média**

10. Ameaça: Exploração de Vulnerabilidades de Software

- **Vulnerabilidade: Uso de software desatualizado**
- **Impacto: Alto**
- **Probabilidade: Média**

11. Ameaça: Interceptação de Comunicações

- **Vulnerabilidade: Uso de conexões não seguras (HTTP ao invés de HTTPS)**
- **Impacto: Médio**
- **Probabilidade: Alta**

12. Ameaça: Fraude Financeira

- **Vulnerabilidade: Sistemas de pagamento com baixa segurança**
- **Impacto: Alto**
- **Probabilidade: Média**

13. Ameaça: Exfiltração de Dados Internos

- **Vulnerabilidade: Falta de controle de acesso adequado aos dados sensíveis**
- **Impacto: Alto**
- **Probabilidade: Média**

14. Ameaça: Configurações de Segurança Impróprias

- **Vulnerabilidade: Falta de revisões regulares das configurações de segurança**
- **Impacto: Médio**
- **Probabilidade: Média**

15. Ameaça: Insider Threat

- **Vulnerabilidade: Falta de monitoramento de atividades internas**
- **Impacto: Alto**
- **Probabilidade: Baixa**

16. Ameaça: Ataque de Força Bruta

- **Vulnerabilidade: Implementação inadequada de medidas anti-brute force**
- **Impacto: Médio**
- **Probabilidade: Média**

17. Ameaça: Comprometimento de API

- **Vulnerabilidade: APIs mal projetadas ou inseguras**
- **Impacto: Alto**
- **Probabilidade: Média**

18. Ameaça: Vulnerabilidades Zero-Day

- **Vulnerabilidade: Falta de resposta rápida a novas vulnerabilidades descobertas**
- **Impacto: Alto**
- **Probabilidade: Baixa**

19. Ameaça: Perda de Confidencialidade

- **Vulnerabilidade: Exposição inadequada de informações em ambientes de desenvolvimento**
- **Impacto: Médio**
- **Probabilidade: Média**

20. Ameaça: Falha de Conformidade Regulamentar

- **Vulnerabilidade: Desconhecimento ou negligência das normas e regulamentos aplicáveis**
- **Impacto: Alto**
- **Probabilidade: Média**

Implementação de Medidas de Segurança

1. Implementação de políticas de controle de acesso aos sistemas e dados:

- **Estabeleça políticas de controle de acesso baseadas nos riscos identificados.**
- **Forneça no mínimo10 políticas de controle de acessos.**

Políticas de Controle de Acesso para a Empresa Tech Code

1. Acesso aos Dados Financeiros

- **Política: O acesso aos dados financeiros é restrito exclusivamente aos membros do departamento financeiro.**
- **Justificativa: Minimizar o risco de fraude financeira e garantir a confidencialidade dos dados financeiros.**

2. Acesso aos Dados Pessoais dos Clientes

- **Política: Apenas funcionários autorizados do departamento de atendimento ao cliente e da equipe de TI podem acessar os dados pessoais dos clientes.**
- **Justificativa: Proteger a privacidade dos clientes e evitar violações de dados.**

3. Acesso às APIs Críticas

- **Política: As APIs críticas são acessíveis apenas por desenvolvedores autorizados e pela equipe de operações.**
- **Justificativa: Prevenir o comprometimento de API e a exploração de vulnerabilidades.**

4. Acesso a Ambientes de Desenvolvimento e Teste

- **Política: O acesso aos ambientes de desenvolvimento e teste é restrito aos desenvolvedores e testadores autorizados.**
- **Justificativa: Proteger informações sensíveis e garantir a integridade dos processos de desenvolvimento.**

5. Acesso aos Sistemas de Pagamento

- **Política: O acesso aos sistemas de pagamento é limitado aos membros da equipe financeira e aos administradores de TI.**
- **Justificativa: Garantir a segurança das transações financeiras e reduzir o risco de fraude.**

6. Acesso a Configurações de Segurança

- **Política: Somente os administradores de segurança de TI têm permissão para alterar configurações de segurança.**
- **Justificativa: Prevenir configurações de segurança impróprias e manter a integridade do sistema.**

7. Acesso a Ferramentas de Backup

- **Política: O acesso às ferramentas de backup é restrito aos administradores de sistemas e à equipe de TI responsável pelo backup.**
- **Justificativa: Assegurar a proteção e a disponibilidade dos dados.**

8. Acesso a Logs de Auditoria

- **Política: Apenas auditores internos e administradores de segurança têm acesso aos logs de auditoria.**

- **Justificativa: Monitorar e investigar atividades suspeitas e manter a integridade dos registros de auditoria.**

9. Acesso a Dados Sensíveis

- **Política: O acesso a dados sensíveis, como informações de pagamento e dados pessoais, é concedido apenas com base na necessidade de conhecimento e mediante autorização.**
- **Justificativa: Minimizar o risco de exfiltração de dados internos e proteger informações confidenciais.**

10. Acesso ao Sistema de Gestão de Cursos

- **Política: O acesso ao sistema de gestão de cursos é restrito aos administradores do sistema, desenvolvedores e instrutores autorizados.**
- **Justificativa: Assegurar a integridade e a disponibilidade dos conteúdos dos cursos online.**

11. Acesso a Recursos de Rede

- **Política: O acesso a recursos de rede críticos, como servidores e dispositivos de rede, é restrito aos administradores de rede.**
- **Justificativa: Prevenir ataques DDoS e proteger a infraestrutura de rede.**

12. Acesso Remoto Seguro

- **Política: O acesso remoto aos sistemas da empresa é permitido apenas através de conexões VPN seguras e após autenticação de dois fatores.**
- **Justificativa: Proteger contra interceptação de comunicações e acessos não autorizados.**

13. Acesso a Ferramentas de Monitoramento de Segurança

- **Política: Somente a equipe de segurança tem permissão para acessar e operar ferramentas de monitoramento de segurança.**
- **Justificativa: Detectar e responder rapidamente a incidentes de segurança, como malware e ransomware.**

14. Acesso Baseado em Função (RBAC)

- **Política: Implementar um modelo de controle de acesso baseado em função (RBAC) para assegurar que os funcionários tenham apenas os acessos necessários para desempenhar suas funções.**
- **Justificativa: Reduzir o risco de acesso não autorizado e manter o princípio do menor privilégio.**

15. Revisão Regular de Acessos

- **Política: Realizar revisões regulares dos direitos de acesso dos funcionários para garantir que as permissões estejam atualizadas e adequadas.**
- **Justificativa: Identificar e revogar acessos desnecessários ou inadequados, prevenindo ameaças internas.**

2. Configuração de sistemas de detecção de intrusão e prevenção de ataques:

- **Instale e configure sistemas para detectar e prevenir intrusões.**
- **Informar pelo menos 10 medidas de detecção e prevenção de ataques na sua empresa**

Medidas de Detecção e Prevenção de Ataques para a Plataforma da Empresa Tech Code
Para proteger a plataforma de cursos online da Tech Code contra ameaças e vulnerabilidades, é fundamental implementar e configurar sistemas de detecção e prevenção de intrusões (IDS/IPS), juntamente com outras medidas de segurança. Segue medidas recomendadas:

1. Implementação de um Sistema IDS/IPS

- **Medida: Instalar e configurar sistemas de detecção (IDS) e prevenção (IPS) de intrusões na infraestrutura de rede que suporta a plataforma de cursos online.**
- **Justificativa: Detectar e bloquear atividades maliciosas, como tentativas de invasão e exploração de vulnerabilidades, antes que causem danos significativos.**

2. Monitoramento Contínuo de Redes

- **Medida: Utilizar ferramentas de monitoramento contínuo de rede para analisar o tráfego em tempo real, identificando padrões anômalos e comportamentos suspeitos.**
- **Justificativa: Detectar e responder rapidamente a atividades maliciosas ou incomuns na rede que possam comprometer a plataforma de cursos.**

3. Uso de Firewalls de Próxima Geração (NGFW)

- **Medida: Implementar firewalls de próxima geração (NGFW) para inspecionar profundamente o tráfego de rede, aplicando políticas de segurança avançadas e bloqueando ameaças sofisticadas.**
- **Justificativa: Proteger a infraestrutura de rede contra ataques complexos e garantir o controle sobre o tráfego de entrada e saída.**

4. Implementação de Sistemas de Detecção de Malware

- **Medida: Utilizar soluções avançadas de detecção de malware que incluam análise comportamental e sandboxing para identificar e mitigar ameaças de malware.**
- **Justificativa: Prevenir infecções por malware que possam comprometer a plataforma de cursos e os dados dos usuários.**

5. Autenticação de Dois Fatores (2FA)

- **Medida: Implementar autenticação de dois fatores (2FA) para todos os acessos aos sistemas críticos da plataforma de cursos online.**
- **Justificativa: Reduzir o risco de acesso não autorizado por meio de credenciais comprometidas, fortalecendo a segurança das contas dos usuários.**

6. Segmentação de Rede

- **Medida: Segmentar a rede em diferentes zonas de segurança, isolando sistemas críticos, dados sensíveis e serviços da plataforma de cursos online.**
- **Justificativa: Limitar a propagação de ameaças e facilitar o controle e monitoramento do tráfego de rede em segmentos distintos.**

7. Análise de Logs e SIEM

- **Medida: Configurar um sistema de gerenciamento e análise de informações e eventos de segurança (SIEM) para consolidar logs de segurança e detectar padrões de ataque.**
- **Justificativa: Fornecer uma visão centralizada das atividades de segurança, permitindo respostas rápidas e eficazes a incidentes de segurança.**

8. Implementação de Honeypots

- **Medida: Utilizar honeypots para atrair e monitorar atividades maliciosas, coletando informações sobre métodos de ataque e comportamentos dos invasores.**
- **Justificativa: Obter informações valiosas sobre ameaças emergentes e melhorar as estratégias de defesa contra invasores.**

9. Controles de Acesso Baseados em Função (RBAC)

- **Medida: Implementar controles de acesso baseados em função (RBAC) para garantir que os funcionários e usuários tenham apenas os acessos necessários para suas funções.**
- **Justificativa: Minimizar o risco de acessos não autorizados e garantir o princípio do menor privilégio, protegendo dados sensíveis e sistemas críticos.**

10. Auditorias e Testes de Penetração Regulares

- **Medida: Realizar auditorias de segurança e testes de penetração periodicamente para identificar e corrigir vulnerabilidades na plataforma de cursos online.**
- **Justificativa: Manter a segurança do sistema atualizada e robusta contra ameaças novas e emergentes, prevenindo explorações de vulnerabilidades.**

11. Atualizações e Patches Regulares

- **Medida: Implementar um processo rigoroso de gerenciamento de patches para garantir que todos os sistemas e softwares da plataforma de cursos estejam atualizados.**
- **Justificativa: Corrigir vulnerabilidades conhecidas e proteger contra exploits zero-day, mantendo a segurança da plataforma em dia.**

12. Treinamento de Conscientização em Segurança

- **Medida: Realizar treinamentos regulares de conscientização em segurança para todos os funcionários e instrutores, focando em boas práticas de segurança e como identificar ameaças.**
- **Justificativa: Aumentar a resiliência da organização contra-ataques de engenharia social e outros vetores baseados em humanos, fortalecendo a cultura de segurança.**