

ANÁLISE DE DADOS

DATA SCIENCE FOR BUSINESS

DESCREVENDO ETAPAS DA ANÁLISE DE DADOS



PROF. GUILHERMINO GOMES

ARQUITETO E URBANISTA, EM 2017 - PLINIO LEITE;
ENG. SEG. DO TRABALHO, EM 2018 - SILVA E SOUZA;

--

ESPECIALIZANDO EM MACHINE LEARNING E ARTIFICIAL INTELLIGENCE - 2023;
MBA EM ANÁLISE DE DADOS - 2022;
MBA EM BUSINESS INTELLIGENCE - 2022;
FORMAÇÃO DNC DEX3 - 2022;
FORMAÇÃO DSACADEMY - 2023;

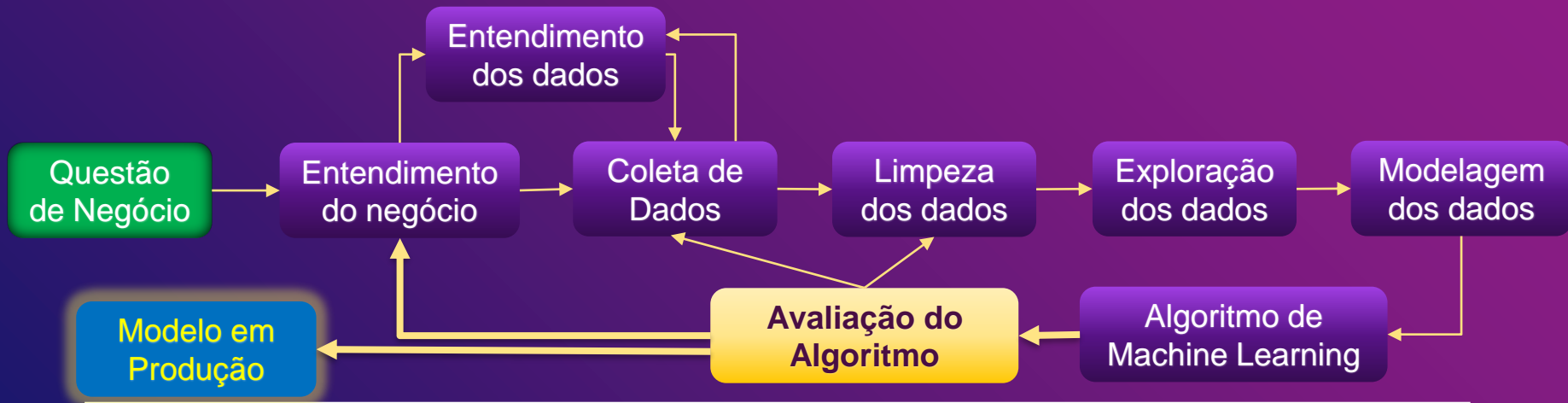
HOJE SOU MONITOR DE DATA SCIENCE NA DNC;
E ATUO COMO ENGENHEIRO DE DADOS EM UM STARTUP DE GAMES MOBILE.



CRISP-DM: Cross-Industry Standard Process for Data Mining

O que é o CRISP-D:

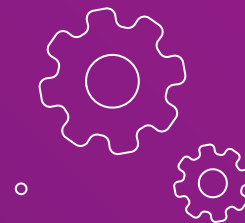
O CRISP-DM um modelo padrão para gerenciamento de projetos de mineração de dados. Ele foi desenvolvido pelo CRISP-DM “*Special Interest Group*” em 1996 e é atualmente um dos modelos mais populares e amplamente utilizados para gerenciamento de projetos de mineração de dados e análise de dados.



CRISP – DM: QUESTÃO DE NEGÓCIO

Nessa fase conhecemos a empresa, e é onde a empresa entra nos trás as duas demandas:

- Procure conhecer os setores e subsetores;
- Estude a empresa;
- Crie um plano de ação;
- Procura interpretar o problema;
- Indetificado o problema, inicia com os planos de ação.
- E por último, pergunte, pergunte, pergunte..





CRISP – DM: ENTENDIMENTO DO NEGÓCIO

Nesta etapa de "Entendimento do Negócio" dentro do CRISP-DM é fundamental identificar as questões de negócio que serão abordadas pelo projeto de mineração de dados. Nessa fase, é importante definir claramente os objetivos do projeto, as necessidades e requisitos do usuário, bem como as restrições e limitações do projeto.



Quais perguntas devemos fazer nesse momento?

- Quais são as necessidades de negócio que o projeto de mineração de dados visa atender?
- Quais são os objetivos do projeto. Aumentar as vendas, reduzir os custos ou melhorar a eficiência dos processos?
- Quais são as restrições do projeto. Os recursos são limitados ou prazos são apertados?
- Quais são os requisitos do usuário, quais tipos de análises ou relatórios são necessários?
- Como o projeto de mineração de dados se integra à estratégia de negócios mais ampla da organização?



CRISP – DM: ENTENDIMENTO DOS DADOS¹



Na etapa de Entendimento dos Dados é uma das etapas do processo de mineração de dados e tem como objetivo coletar e explorar os dados disponíveis para a identificação de padrões, tendências e distribuições nos dados e entender os metadados também tem uma importância nessa etapa pois são informações descritivas sobre os dados que descrevem o contexto, o conteúdo e a estrutura dos dados. Eles fornecem informações adicionais para ajudar a entender e interpretar os dados de maneira mais eficiente e precisa. Alguns exemplos de metadados incluem:



Durante essa fase, são realizadas atividades como:

- Coleta dos dados necessários para o projeto de mineração de dados;
- Descrição dos dados, identificando a qualidade e integridade dos mesmos;
- Exploração dos dados, identificando padrões, tendências e distribuições nos dados;
- Identificação de variáveis relevantes para a análise;
- Verificação da qualidade dos dados e consistência das variáveis;
- Identificação de valores ausentes, discrepantes ou extremos;
- Análise da correlação entre as variáveis e as possíveis relações entre elas.

Metadados ou Meta informação:

- Nome da variável;
- Tipo de dado;
- Descrição da variável;
- Formato dos dados;
- Data de criação;
- Autor do arquivo;
- Método de coleta de dados;
- Nome do conjunto de dados;
- Número de registros;
- Frequência de atualização.



¹Metadados são informação contidas na tabela



CRISP – DM: COLETA DOS DADOS

A coleta de dados é o processo de obtenção de dados relevantes e precisos para o projeto de mineração de dados. Envolve a identificação das fontes de dados, a verificação da qualidade e integridade dos dados, e a garantia de que a coleta de dados esteja de acordo com as regulamentações e políticas de privacidade. É um processo fundamental para garantir a eficácia do projeto de mineração de dados.



Algumas das principais considerações a serem levadas em conta na coleta de dados incluem:

- Identificação dos tipos de dados necessários para o projeto de mineração de dados;
- Identificação das fontes de dados relevantes e disponíveis;
- Definição do método de coleta de dados a ser utilizado;
- Verificação da qualidade e integridade dos dados;
- Garantia de que a coleta de dados esteja de acordo com as regulamentações e políticas de privacidade.





CRISP – DM: LIMPEZA DOS DADOS

A limpeza de dados é uma etapa importante do processo de mineração de dados que envolve a identificação e correção de problemas nos dados, como valores ausentes, inconsistentes ou incorretos. Esses problemas podem afetar negativamente a qualidade dos dados e prejudicar a eficácia da análise exploratória de dados e dos modelos de mineração de dados.

Alguns exemplos de técnicas de limpeza de dados incluem:

- **Identificação de valores ausentes** e preenchimento desses valores com um valor apropriado;
- **Identificação e remoção de valores discrepantes e extremos;**
- Identificação e correção de valores incorretos ou inconsistentes;
- Normalização dos dados para garantir que os valores estejam na mesma escala;
- Remoção de dados duplicados ou irrelevantes.

```
import pandas as pd

# Carregando o arquivo CSV em um DataFrame
df = pd.read_csv('dados.csv')

# Verificando os valores ausentes
print(df.isnull().sum())

# Preenchendo os valores ausentes com a média
df.fillna(df.mean(), inplace=True)

# Removendo as linhas com valores discrepantes
df = df[df['coluna'] < valor_maximo]

# Salvando o DataFrame limpo em um novo arquivo CSV
df.to_csv('dados_limpos.csv', index=False)
```





CRISP – DM: EXPLORAÇÃO DE DADOS

Quanto a exploração de dados, também conhecida como Análise Exploratória de Dados (AED), é uma etapa importante do processo de mineração de dados que tem como objetivo identificar padrões, tendências e relações entre as variáveis dos dados.

Algumas das técnicas utilizadas na exploração de dados incluem:

- Análise descritiva: estatísticas básicas, como média, mediana, desvio padrão, etc., são usadas para resumir as características dos dados;
- Análise de correlação: a relação entre duas ou mais variáveis é analisada para determinar se existe uma relação entre elas;
- Visualização de dados: gráficos, histogramas e outros métodos visuais são usados para ilustrar os padrões e relações nos dados;
- Análise de cluster: os dados são agrupados com base em suas características semelhantes.

```
import pandas as pd
import matplotlib.pyplot as plt

# Carregando o arquivo CSV em um DataFrame
df = pd.read_csv('dados.csv')

# Verificando as estatísticas básicas dos dados
print(df.describe())

# Plotando um histograma da variável 'idade'
df['idade'].plot(kind='hist', bins=20)
plt.title('Histograma da Idade')
plt.xlabel('Idade')
plt.ylabel('Frequência')
plt.show()

# Verificando a correlação entre as variáveis
corr_matrix = df.corr()
print(corr_matrix)
```



CRISP – DM: EXPLORAÇÃO DE DADOS

Neste exemplo eu estou usando a base de dados Iris para demonstrar a exploração de dados em Python.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregando o conjunto de dados Iris em um DataFrame
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
nomes_colunas = ['comprimento_sepala', 'largura_sepala', 'comprimento_petala', 'largura_petala', 'classe']
df = pd.read_csv(url, names=nomes_colunas)

# Exibindo as primeiras linhas do DataFrame
print(df.head())

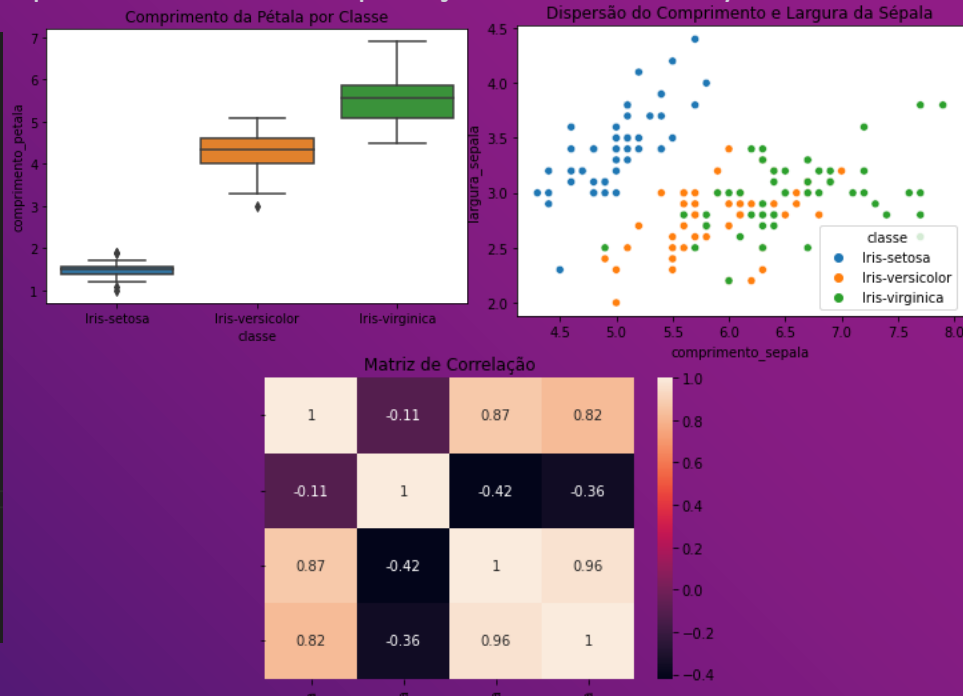
# Exibindo informações básicas do DataFrame
print(df.info())

# Exibindo estatísticas descritivas do DataFrame
print(df.describe())

# Plotando um gráfico de dispersão para comparar o comprimento e a largura da sépala
sns.scatterplot(data=df, x="comprimento_sepala", y="largura_sepala", hue="classe")
plt.title('Dispersão do Comprimento e Largura da Sépala')
plt.show()

# Plotando um gráfico de caixa para comparar o comprimento da pétala por classe
sns.boxplot(data=df, x="classe", y="comprimento_petala")
plt.title('Comprimento da Pétala por Classe')
plt.show()

# Verificando a correlação entre as variáveis do DataFrame
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True)
plt.title('Matriz de Correlação')
plt.show()
```



Plotei um gráfico de dispersão para comparar o comprimento e largura da sépala, um gráfico de caixa para comparar o comprimento da pétala por classe e uma matriz de correlação para verificar as relações entre as variáveis do DataFrame.



CRISP – DM: MODELOS DE MACHINE LEARNING

A modelagem dos dados é uma etapa importante do processo de mineração de dados que envolve a construção de modelos de aprendizado de máquina.

Algumas técnicas de modelagem de dados:



- **Regressão linear:** técnica de aprendizado supervisionado usada para prever um valor contínuo com base em variáveis independentes;
- **Regressão logística:** técnica de aprendizado supervisionado usada para prever uma classe binária com base em variáveis independentes;
- **Árvores de decisão:** técnica de aprendizado supervisionado usada para classificar ou prever dados com base em uma série de decisões baseadas em regras;
- **Random Forest:** técnica de aprendizado supervisionado que usa múltiplas árvores de decisão para prever ou classificar dados;
- **K-Means:** técnica de aprendizado não supervisionado usada para identificar clusters ou grupos de dados semelhantes;
- **Análise de componentes principais (PCA):** técnica de aprendizado não supervisionado usada para reduzir a dimensionalidade dos dados, mantendo as informações relevantes;
- **Redes neurais:** técnica de aprendizado de máquina que simula o funcionamento do cérebro humano para prever ou classificar dados.





CRISP – DM: MODELAGEM - REGRESSÃO LINEAR

Regressão linear: técnica de aprendizado supervisionado usada para prever um valor contínuo com base em variáveis independentes. É um modelo matemático que representa uma linha reta que melhor se ajusta aos dados. Essa função é usada para fazer previsões sobre a variável dependente com base nos valores das variáveis independentes.

Neste código utilizei a biblioteca Scikit-Learn para fazer o Modelo de Regressão Linear e como retorno obtemos o MSE – Mean Squared Error ou Erro Médio Quadrático. É uma métrica usada para avaliar a qualidade de um modelo de regressão. Ela mede a média dos erros quadrados entre as previsões do modelo e os valores reais. Quanto menor o valor do MSE, melhor é o modelo, isso significa que as previsões estão próximas da realidade.

A fórmula do MSE é: $MSE = (1/n) * \sum (y_{true} - y_{pred})^2$

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Carregando o arquivo CSV em um DataFrame
url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/anscombe.csv'
df = pd.read_csv(url)

# Definindo as variáveis independentes e dependentes
X = df[['x']]
y = df['y']

# Dividindo os dados em conjunto de treinamento e conjunto de teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Treinando o modelo de regressão linear
modelo = LinearRegression()
modelo.fit(X_train, y_train)

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


# Calculando o erro médio quadrático
mse = mean_squared_error(y_test, y_pred)
print('Erro médio quadrático: ', mse)
```

N = é o número de amostras nos dados de teste;

y_true = é o valor real da variável dependente para a i-ésima amostra nos dados de teste;

y_pred = é o valor previsto da variável dependente para a i-ésima amostra nos dados de teste.

CRISP – DM: MODELAGEM - REGRESSÃO LOGÍSTICA



Regressão logística: É um modelo de regressão utilizado para problemas de classificação binária, ou seja, para prever a probabilidade de um evento ocorrer ou não. O modelo é baseado na função logística, que é uma função sigmoide que produz um valor de saída entre 0 e 1. A partir dos dados de entrada, o modelo estima a probabilidade de uma determinada classe (0 ou 1) e, em seguida, classifica a amostra com base nessa probabilidade.

O modelo é treinado com um conjunto de dados rotulados, ou seja, para cada amostra do conjunto de treinamento, já é conhecido o rótulo ou classe correta. Durante o treinamento, o modelo ajusta seus parâmetros para minimizar a diferença entre as probabilidades previstas e os rótulos verdadeiros.

Após o treinamento, o modelo é testado em um conjunto de dados de teste para avaliar sua capacidade de generalização. A acurácia é uma métrica comum para avaliar a qualidade do modelo, mas outras métricas, como precisão, recall e F1-score, também podem ser utilizadas dependendo do problema em questão.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

# Carregando o arquivo CSV em um DataFrame
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
colunas = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
df = pd.read_csv(url, header=None, names=colunas)

# Definindo as variáveis independentes e dependentes
X = df.drop('class', axis=1)
y = df['class']

# Dividindo os dados em conjunto de treinamento e conjunto de teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Treinando o modelo de regressão logística
modelo = LogisticRegression()
modelo.fit(X_train, y_train)

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

# Calculando a acurácia do modelo
acuracia = accuracy_score(y_test, y_pred)
print('Acurácia: ', acuracia)

# Calculando a matriz de confusão do modelo
matriz_confusao = confusion_matrix(y_test, y_pred)
print('Matriz de Confusão: \n', matriz_confusao)
```



CRISP – DM: MODELAGEM - ÁRVORE DE DECISÃO

Árvores de decisão: É um método de aprendizado de máquina supervisionado utilizado para problemas de classificação ou regressão. O modelo consiste em criar uma árvore onde cada nó interno representa uma decisão baseada em uma das variáveis preditoras, e cada folha representa o resultado final da classificação ou regressão. A árvore é construída a partir de um conjunto de dados rotulados, utilizando algoritmos de divisão de dados em cada nó, até que as amostras dentro de cada nó sejam suficientemente homogêneas.

A técnica de árvores de decisão é conhecida por sua simplicidade e interpretabilidade, já que as decisões são tomadas de forma transparente e intuitiva. Além disso, a técnica pode lidar com dados categóricos e numéricos e pode ser utilizada em problemas com grande número de variáveis.

No entanto, a técnica de árvores de decisão pode sofrer de **overfitting**, ou seja, a árvore pode se ajustar demais aos dados de treinamento e não generalizar bem para novos dados. Para lidar com esse problema, podem ser utilizadas técnicas de poda de árvore ou a combinação de múltiplas árvores em uma técnica chamada de Random Forest.

```
# Importando as bibliotecas necessárias
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Carregando o conjunto de dados
iris = load_iris()
X = iris.data
y = iris.target

# Dividindo os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Criando uma árvore de decisão
clf = DecisionTreeClassifier(max_depth=3)

# Treinando a árvore de decisão
clf.fit(X_train, y_train)

# Fazendo previsões com a árvore de decisão
y_pred = clf.predict(X_test)

# Avaliando a acurácia da árvore de decisão
accuracy = accuracy_score(y_test, y_pred)
print("Acurácia da árvore de decisão:", accuracy)
```





CRISP – DM: MODELAGEM – RANDOM FOREST

Random Forest: É uma técnica de aprendizado supervisionado que usa múltiplas árvores de decisão para realizar a classificação ou regressão de dados.

Cada árvore é construída usando um subconjunto dos dados e variáveis preditoras, e a classificação ou regressão final é feita pela média das previsões de cada árvore. É útil para problemas com muitas variáveis e amostras, e pode ser usado para selecionar variáveis importantes.

Reduz o overfitting em relação à árvore de decisão única, mas pode ser mais lento e consumir mais recursos computacionais.

Essa técnica é capaz de gerar modelos preditivos robustos e precisos, mesmo em grandes conjuntos de dados com muitas variáveis.

Uma das **principais vantagens do Random Forest é a sua capacidade de lidar com problemas de overfitting**. Isso ocorre porque, ao contrário da árvore de decisão única, que pode se ajustar demais aos dados de treinamento, o Random Forest combina as previsões de várias árvores de decisão, o que pode reduzir a variância e melhorar a generalização do modelo para novos dados.

```
from sklearn.datasets import load_wine
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Carregando o dataset de vinhos do Scikit-learn
wine = load_wine()

# Dividindo o dataset em treino e teste
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.3, random_state=42)

# Criando um modelo Random Forest com 100 estimadores
rf_model = RandomForestClassifier(n_estimators=100)

# Treinando o modelo
rf_model.fit(X_train, y_train)

# Fazendo previsões no conjunto de teste
y_pred = rf_model.predict(X_test)

# Avaliando o desempenho do modelo
score = rf_model.score(X_test, y_test)

print("Acurácia:", score)
```





CRISP – DM: MODELAGEM – K-MEANS

K-Means: É um método de aprendizado de máquina não supervisionado que é comumente usado para agrupar conjuntos de dados em clusters. O número de clusters k é especificado pelo usuário e o algoritmo funciona tentando minimizar a soma das distâncias entre cada amostra e o centroide do seu respectivo cluster.

O algoritmo funciona inicialmente selecionando aleatoriamente k amostras como os centros iniciais dos clusters. Em seguida, cada amostra no conjunto de dados é atribuída ao centroide mais próximo. Em seguida, a posição dos centroides é atualizada e o processo de atribuição é repetido até que a posição dos centroides não mude significativamente ou até que um número máximo de iterações seja alcançado.

Após a conclusão do algoritmo, as amostras são divididas em k clusters, onde cada cluster é caracterizado pelo seu centroide. A técnica de K-Means é amplamente utilizada em áreas como análise de mercado, processamento de imagem e reconhecimento de padrões, onde a identificação de grupos de dados semelhantes é fundamental para a tomada de decisões.

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

# Carregando o conjunto de dados Iris do Scikit-learn
iris = load_iris()

# Criando o modelo de K-Means com 3 clusters
kmeans_model = KMeans(n_clusters=3)

# Treinando o modelo com os dados do conjunto Iris
kmeans_model.fit(iris.data)

# Obtendo as previsões de cluster para os dados
y_pred = kmeans_model.predict(iris.data)

# Imprimindo os resultados
print("Previsões de cluster:")
print(y_pred)
```



CRISP – DM: MODELAGEM – PCA

Análise de componentes principais (PCA): É uma técnica de aprendizado de máquina não supervisionada usada para reduzir a dimensionalidade dos dados, mantendo as informações mais relevantes.

A principal aplicação da PCA é reduzir o número de variáveis em conjuntos de dados complexos. Em outras palavras, a PCA é usada para transformar conjuntos de dados com muitas variáveis em um conjunto menor de variáveis chamadas de componentes principais. Esses componentes principais são selecionados de forma que retenham a maior parte da variação dos dados originais.

A redução da dimensionalidade dos dados pode ajudar a simplificar a análise de dados, tornar a visualização de dados mais fácil e reduzir o tempo de processamento necessário para executar algoritmos de aprendizado de máquina. A PCA também é frequentemente usada em problemas de reconhecimento de padrões, análise de imagem, bioinformática e outras áreas onde a redução da dimensionalidade dos dados é importante para a realização de análises eficientes e eficazes.

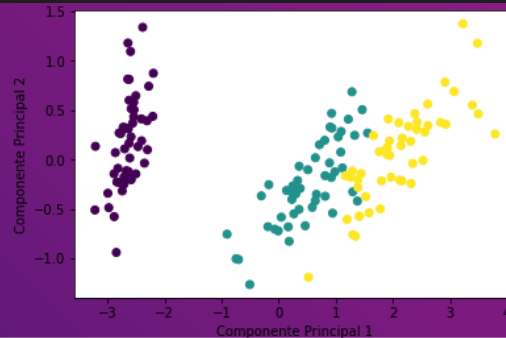
```
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Carregando o conjunto de dados Iris do Scikit-learn
iris = load_iris()

# Criando um objeto PCA com 2 componentes principais
pca = PCA(n_components=2)

# Transformando os dados originais em 2 componentes principais
iris_pca = pca.fit_transform(iris.data)

# Plotando o resultado da transformação PCA
plt.scatter(iris_pca[:, 0], iris_pca[:, 1], c=iris.target)
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.show()
```



Obs.: a técnica de PCA também tem algumas limitações. Ela pode ser sensível a valores extremos e outliers, e pode ser difícil de interpretar, uma vez que as novas variáveis não correspondem diretamente às variáveis originais.



CRISP – DM: MODELAGEM – SÉRIES TEMPORAIS

1. **Modelos de média móvel:** são modelos que utilizam a média móvel dos dados históricos para prever valores futuros.
2. **Modelos de regressão linear:** são modelos que usam uma regressão linear para encontrar padrões de tendência nos dados e prever valores futuros.
3. **Modelos autorregressivos (AR):** são modelos que usam uma regressão dos valores anteriores da série temporal para prever valores futuros.
4. **Modelos de média móvel autorregressiva (ARMA):** são modelos que combinam a média móvel e modelos autorregressivos para prever valores futuros.
5. **Modelos de média móvel autorregressiva integrada (ARIMA):** são modelos que combinam a média móvel, modelos autorregressivos e técnicas de diferenciação para prever valores futuros.
6. **Redes neurais recorrentes (RNN):** são modelos de redes neurais que podem modelar padrões temporais complexos em séries temporais.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

# Carregando os dados
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv', parse_dates=['date'])
df = df.set_index('date')

# Visualizando a série temporal
plt.plot(df)
plt.title('Série Temporal')
plt.xlabel('Ano')
plt.ylabel('Valor')
plt.show()

# Dividindo a série temporal em treinamento e teste
train_data = df['1994']
test_data = df['1995']

# Treinando o modelo ARIMA
model = ARIMA(train_data, order=(2,1,1))
model_fit = model.fit()

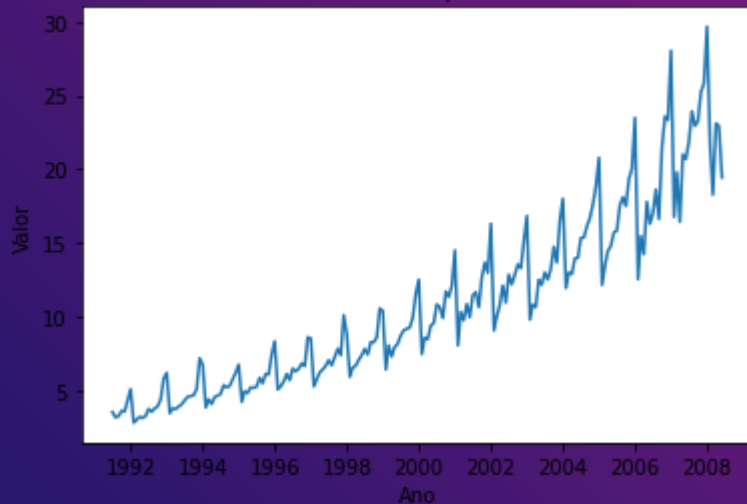
# Realizando previsões com o modelo treinado
predictions = model_fit.predict(start='1995-01-01', end='1996-12-01', typ='levels')

# Visualizando as previsões em comparação com os dados de teste
plt.plot(test_data)
plt.plot(predictions, color='red')
plt.title('Previsões de Série Temporal com ARIMA')
plt.xlabel('Ano')
plt.ylabel('Valor')
plt.legend(['Dados de Teste', 'Previsões'])
plt.show()
```

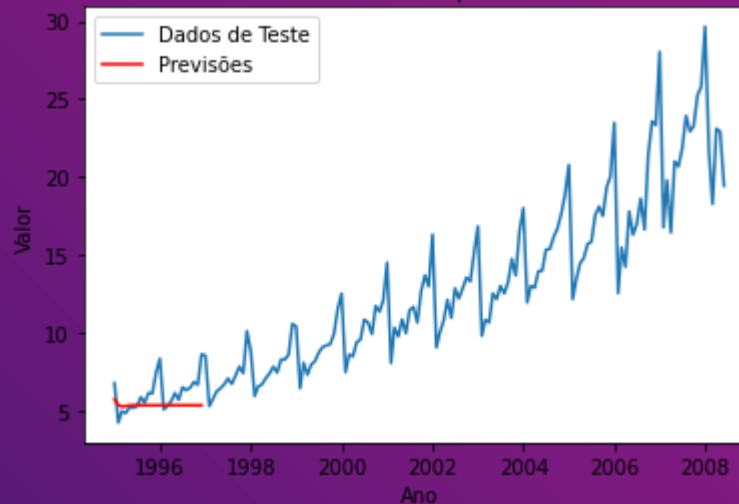


CRISP – DM: MODELAGEM – SÉRIES TEMPORAIS

Série Temporal



Previsões de Série Temporal com ARIMA





CRISP – DM: MODELAGEM – SÉRIES TEMPORAIS

```
[3] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

[5] url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/monthly-car-sales.csv'
df = pd.read_csv(url, header=0, index_col=0, parse_dates=True, squeeze=True)

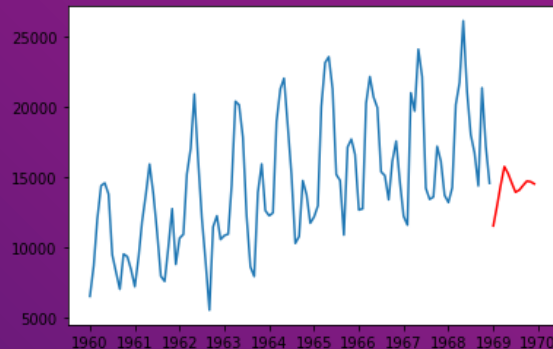
[18] order = (3, 1, 0) # (p, d, q)

model = ARIMA(df, order=order)
model_fit = model.fit()

/usr/local/lib/python3.9/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
self._init_dates(dates, freq)
/usr/local/lib/python3.9/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
self._init_dates(dates, freq)
/usr/local/lib/python3.9/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided, so inferred frequency MS will be used.
self._init_dates(dates, freq)

[20] forecast = model_fit.forecast(steps=12)

[21] plt.plot(df)
plt.plot(forecast, color='red')
plt.show()
```





CRISP – DM: MODELAGEM – REDES NEURAIS

Redes Neurais Artificiais (RNAs): é baseada em modelos computacionais inspirados no funcionamento do cérebro humano. Ela é usada para problemas de classificação ou regressão e consiste em várias camadas de neurônios interconectados, onde cada neurônio recebe entradas dos neurônios da camada anterior, realiza um cálculo matemático e envia uma saída para os neurônios da camada posterior.

Durante o treinamento, os pesos das conexões entre os neurônios são ajustados para minimizar a diferença entre as previsões do modelo e os rótulos verdadeiros. Isso é feito usando técnicas de otimização, como o gradiente descendente, que ajustam os pesos em direção à solução ótima do problema.

A técnica de RNAs é conhecida por apresentar boa performance em problemas complexos, como reconhecimento de fala, detecção de fraudes, reconhecimento de imagens e outras aplicações em que os dados possuem uma alta complexidade. No entanto, a técnica pode exigir grandes quantidades de dados e recursos computacionais para o treinamento, e o processo de ajuste dos pesos pode ser lento e requerer ajustes cuidadosos.

Alguns exemplos são:

- **Feedforward Neural Networks:** Redes neurais simples, a informação flui apenas em uma direção, da camada de entrada para a camada de saída. São usadas para problemas de classificação e regressão.
- **Redes neurais convolucionais (CNNs):** Redes neurais especializadas em processamento de imagens, onde as camadas aprendem a detectar padrões em diferentes regiões da imagem.
- **Redes neurais recorrentes (RNNs):** Redes neurais que permitem a informação fluir em loop, permitindo que a rede possa manter informações de estados anteriores. São usadas em problemas de previsão temporal, como, séries temporais.
- **Autoencoder:** é uma rede neural que aprende a representar os dados de entrada em um espaço de menor dimensão. É usada para reduzir a dimensionalidade dos dados e para fins de compressão de dados.
- **Redes neurais generativas adversariais (GANs):** são redes neurais compostas por duas redes, uma geradora e outra discriminadora, que são treinadas juntas para gerar dados novos que são semelhantes aos dados reais.





CRISP – DM: MODELAGEM – ALGORITMOS DE ML

Aprendizado Supervisionado:

Este tipo de algoritmo é utilizado quando se tem dados rotulados, ou seja, quando o resultado esperado já é conhecido. O algoritmo é treinado com esses dados e, a partir disso, consegue prever resultados para novos dados.

- Regressão Linear
- Regressão Logística
- Árvores de Decisão
- Random Forest
- Redes Neurais

Aprendizado Não Supervisionado:

Este tipo de algoritmo é utilizado quando não se tem dados rotulados, ou seja, quando não se sabe qual é o resultado esperado. O objetivo é encontrar padrões e estruturas nos dados.

- K-Means
- Análise de Componentes Principais (PCA)
- Algoritmos de Clusterização

Aprendizado por Reforço:

Este tipo de algoritmo é utilizado em situações em que o agente precisa aprender a partir de sua interação com o ambiente. Ele recebe feedback em forma de recompensas ou penalidades, que são utilizadas para ajustar seu comportamento.

- Q-Learning
- Aprendizado Profundo por Reforço





CRISP – DM: AVALIAÇÃO DO ALGORITMO

A avaliação do desempenho do algoritmo é uma etapa importante em qualquer projeto de Machine Learning. Existem diversas métricas que podem ser utilizadas para avaliar o desempenho do algoritmo, e a escolha da métrica adequada depende do tipo de problema que está sendo abordado. Algumas das métricas mais comuns incluem:

- **Acurácia:** medida de quantas previsões foram corretas em relação ao total de previsões feitas.
- **Precisão:** medida de quantas previsões positivas foram corretas em relação ao total de previsões positivas feitas.
- **Recall:** medida de quantos valores positivos foram corretamente previstos em relação ao total de valores positivos.
- **F1-score:** média harmônica entre precisão e recall, que fornece uma medida geral do desempenho do algoritmo.
- **Área sob a curva ROC (AUC-ROC):** medida de quão bem o modelo é capaz de distinguir entre as classes positivas e negativas.
- **Erro quadrático médio (MSE):** medida de quão próximas as previsões estão dos valores verdadeiros em um problema de regressão.





CRISP – DM: AVALIAÇÃO DO ALGORITMO

Exemplos de códigos para cada uma das métricas de avaliação de algoritmos de Machine Learning:

Acurácia:

```
from sklearn.metrics import accuracy_score

# Y_true: valores verdadeiros
# Y_pred: valores previstos pelo modelo
acc = accuracy_score(Y_true, Y_pred)
print('Acurácia:', acc)
```

Recall:

```
from sklearn.metrics import recall_score

# Y_true: valores verdadeiros
# Y_pred: valores previstos pelo modelo
# average: tipo de média utilizada (macro, micro ou weighted)
rec = recall_score(Y_true, Y_pred, average='macro')
print('Recall:', rec)
```

Curva ROC (AUC - ROC):

```
from sklearn.metrics import roc_auc_score

# Y_true: valores verdadeiros
# Y_score: scores de probabilidade do modelo
auc = roc_auc_score(Y_true, Y_score)
print('AUC-ROC:', auc)
```

Precisão:

```
from sklearn.metrics import precision_score

# Y_true: valores verdadeiros
# Y_pred: valores previstos pelo modelo
# average: tipo de média utilizada (macro, micro ou weighted)
prec = precision_score(Y_true, Y_pred, average='macro')
print('Precisão:', prec)
```

F1-score:

```
from sklearn.metrics import f1_score

# Y_true: valores verdadeiros
# Y_pred: valores previstos pelo modelo
# average: tipo de média utilizada (macro, micro ou weighted)
f1 = f1_score(Y_true, Y_pred, average='macro')
print('F1-score:', f1)
```

Erro quadrático médio (MSE)

```
from sklearn.metrics import mean_squared_error

# Y_true: valores verdadeiros
# Y_pred: valores previstos pelo modelo
mse = mean_squared_error(Y_true, Y_pred)
print('MSE:', mse)
```





CRISP – DM: DETALHADO E O KDD

- CRISP-DM:

1. **Entendimento do negócio:** identificar o problema de negócio e definir os objetivos da mineração de dados.
2. **Entendimento dos dados:** coletar, avaliar e explorar os dados disponíveis.
3. **Preparação dos dados:** limpar, transformar e preparar os dados para a modelagem.
4. **Modelagem:** selecionar e aplicar técnicas de mineração de dados para construir modelos que atendam aos objetivos do negócio.
5. **Avaliação:** avaliar e validar os modelos criados, verificando se eles atendem aos critérios de sucesso definidos.
6. **Implantação:** implementar os modelos em um ambiente de produção.

- KDD - Descoberta de Conhecimento em Bancos de Dados:

1. Seleção de dados: identificar os dados relevantes para o problema de negócio.
2. Pré-processamento: limpar, transformar e integrar os dados selecionados.
3. Transformação: criar novas variáveis e reduzir a dimensionalidade dos dados.
4. Mineração de dados: aplicar técnicas de mineração de dados para identificar padrões e relacionamentos nos dados.
5. Interpretação e avaliação: interpretar e avaliar os padrões e relacionamentos encontrados e identificar seu valor para o negócio.
6. Consolidação: integrar os resultados da mineração de dados com o conhecimento do domínio e do negócio para produzir um resultado final.



Links:

Arquivo 1 -- <https://colab.research.google.com/drive/1dTLZJvOqILSJOiBlRcGzc1fQ1SufGdKw?usp=sharing>

Arquivo 2 -- <https://colab.research.google.com/drive/1QK23hijYHbQhMqt--Go4EUwz-qSq09hy?usp=sharing>

Bibliografia:

<https://web.archive.org/web/20051030083454/http://www.crisp-dm.org/>

<https://web.archive.org/web/20110515013442/http://crispdm.wordpress.com/>

https://s2.smu.edu/tfomby/eco5385_eco6380/data/SPSS/CRISP_Wikipedia.pdf

[https://www.professores.uff.br/fcbernardini/wp-content/uploads/sites/68/2017/08/01-](https://www.professores.uff.br/fcbernardini/wp-content/uploads/sites/68/2017/08/01-Introdu%C3%A7%C3%A3o-a-KDD-e-DM.pdf)

[Introdu%C3%A7%C3%A3o-a-KDD-e-DM.pdf](https://www.professores.uff.br/fcbernardini/wp-content/uploads/sites/68/2017/08/01-Introdu%C3%A7%C3%A3o-a-KDD-e-DM.pdf)



“FAZER!!! OU NÃO FAZER,
TENTAR NÃO HÁ!”

MESTRE YODA