Algoritmos de Aprendizado Supervisionado

Prof. Edson Pedro Ferlin

O que é Aprendizado Supervisionado?

Definição

Aprendizado com dados rotulados (entrada e saída conhecida).

Objetivo

Fazer previsões ou classificações com base nos dados de entrada.

Exemplos

- Prever o preço de uma casa (Regressão Linear)
- Identificar se um e-mail é spam (Classificação)

Tipos de Algoritmos Supervisionados

Regressão

Prever valores contínuos (preço, temperatura).

Classificação

Classificar dados em categorias discretas.

Exemplo

- Prever preço de casas.
- Classificar e-mails como spam.

Regressão Linear

Objetivo

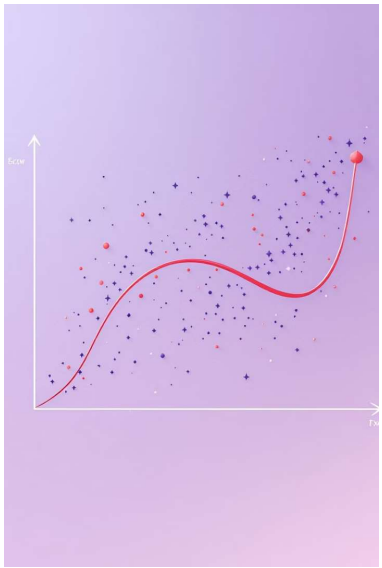
Relação entre variável dependente e independentes.

Modelo

$y=\beta 0+\beta 1x$

Variáveis

- y: variável dependente (target)
- β0: intercepto
- β1: coeficiente
- x: variável independente



Exemplo de Regressão Linear

1 Problema

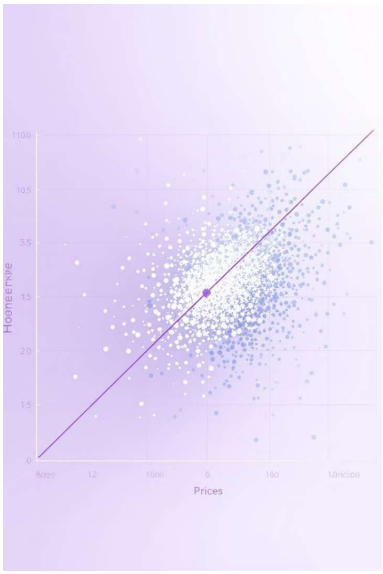
Prever o preço de uma casa com base em sua área.

2 Modelo

Preço=β0+β1(Área)

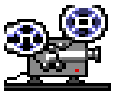
3 Objetivo

Encontrar a relação entre área da casa e o seu preço.



Regressão Linear

Assista o vídeo sobre Regressão Linear
(link: https://youtu.be/3k_a_NHGatM).



Classificação

1 Objetivo

Classificar dados em categorias ou classes.

2 Exemplo

- Spam ou Não Spam: Classificar e-mails.
- Doente ou Saudável: Classificar pacientes.

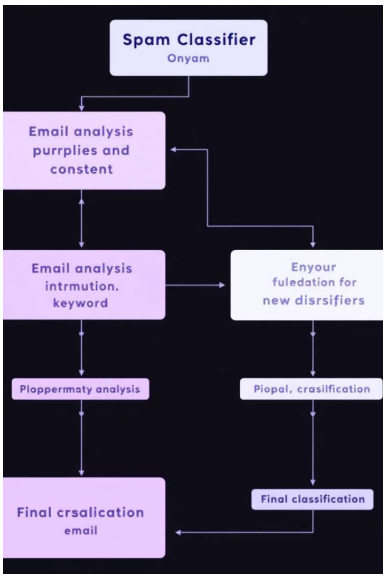
Exemplo de Classificação

1 Problema

Classificar e-mails como spam ou não spam.

2 Modelo

Árvore de Decisão, k-NN ou SVM para classificação.



Usando o Scikit-learn

1

O que é?

Biblioteca popular para aprendizado supervisionado em Python.

2

Contém

Algoritmos de regressão, classificação e agrupamento.

Passos para usar o Scikit-learn:

- **Importar as Bibliotecas**: `sklearn`, `pandas`, `matplotlib`.
- **Carregar e Preparar os Dados**.**Escolher e Treinar o Modelo**.
- **Avaliar o Modelo**.
- **Fazer Previsões**.

Exemplo Prático: Regressão Linear com Scikit-learn

Problema: Prever o preço de casas com base na área.

- **Importação**: `pandas` para manipulação de dados, `sklearn` para modelos e avaliação.
- **Divisão dos dados**: Dividimos os dados em **treinamento** e **teste**.
- **Modelo**: Usamos `LinearRegression()` para treinar o modelo.
- **Avaliação**: Calculamos o **erro quadrático médio** (MSE) para verificar a qualidade do modelo.

```
# Importando bibliotecas
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

# Carregar o dataset
data = pd.read_csv('house_data.csv')
X = data[['area']]  # Características (área)
y = data['price']   # Rótulo (preço)

# Dividir os 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)

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

# Fazer previsões
y_pred = model.predict(X_test)

# Avaliar o modelo
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

Exemplo Prático: Classificação com Scikit-learn

Problema: Classificar e-mails como **spam** ou **não spam**.

Transformação de Texto: Usamos **TF-IDF** para converter o texto em vetores numéricos.

Modelo: Utilizamos o **Naive Bayes** para classificação.

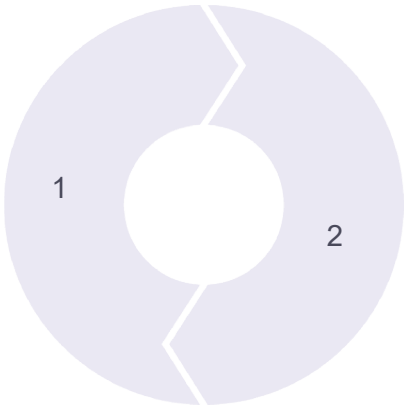
Avaliação: A acurácia do modelo é calculada com a função accuracy_score().

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# Carregar o dataset de e-mails
data = pd.read_csv('emails.csv')
X = data['email_text'] # Características (texto do e-mail)
y = data['label'] # Rótulo (spam ou não spam)

# Dividir os 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)

# Converter o texto para vetores numéricos usando TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)

# Criar o modelo de classificação
model = MultinomialNB()
model.fit(X_train_vect, y_train)

# Fazer previsões
y_pred = model.predict(X_test_vect)

# Avaliar o modelo
accuracy = accuracy_score(y_test, y_pred)
print(f'Acurácia: {accuracy}')
```


Avaliação de Modelos

Regressão

Erro Quadrático Médio (MSE).



Classificação

Acurácia, precisão, revocação e F1-Score.

Desafios Comuns

Overfitting

Modelo aprende demais os dados de treinamento.

Dados desbalanceados

Uma classe tem mais exemplos do que a outra.

Escolha do modelo

Determinar o algoritmo mais adequado.

Contato

eferlin@live.com

(BLOG) professorferlin.blogspot.com

(SITE) professorferlin.webnode.com.br

(YOUTUBE) ProfEdsonPedroFerlin