

Aprendizado de Máquina: Regressão e Classificação

Aplicando Algoritmos para Previsão e Categorização de Dados

Márcio Nicolau

2025-11-06

Table of contents

Introdução	2
Objetivo de Aprendizagem	2
Revisão Rápida: Aprendizado Supervisionado	2
Regressão	2
Regressão Linear Simples	3
Função de Custo para Regressão: Erro Quadrático Médio (MSE)	3
Métricas de Avaliação de Regressão	3
Exemplo em Python: Regressão Linear com Scikit-learn	4
Classificação	7
Regressão Logística	7
Função de Custo para Classificação: Entropia Cruzada	7
Métricas de Avaliação de Classificação	7
Exemplo em Python: Regressão Logística com Scikit-learn	8
Escolhendo entre Regressão e Classificação	11
Considerações Finais	11
Verificação de Aprendizagem	11
Referências Bibliográficas	12

List of Figures

1	Matriz de Confusão para Classificação Binária	8
---	---	---

Introdução

Na aula anterior, estabelecemos os fundamentos do Aprendizado de Máquina (ML), definindo-o como a capacidade de sistemas aprenderem a partir de dados e explorando seus principais paradigmas: supervisionado, não supervisionado e por reforço. Concluímos que o aprendizado supervisionado é o mais comum, onde um modelo aprende a mapear entradas para saídas corretas.

Nesta aula, aprofundaremos no Aprendizado Supervisionado, focando em suas duas tarefas mais cruciais: **Regressão** e **Classificação**. Apresentaremos algoritmos fundamentais para cada uma, demonstraremos sua aplicação prática em Python com a biblioteca `scikit-learn`, e discutiremos as métricas de avaliação essenciais para entender o desempenho de nossos modelos.

Objetivo de Aprendizagem

Ao final desta aula, você será capaz de:

- Distinguir problemas de regressão e classificação.
- Compreender os princípios básicos de algoritmos de regressão como a Regressão Linear.
- Compreender os princípios básicos de algoritmos de classificação como a Regressão Logística.
- Aplicar `scikit-learn` para implementar e treinar modelos de regressão e classificação em Python.
- Avaliar o desempenho de modelos de regressão usando métricas como Erro Médio Absoluto (MAE) e Erro Quadrático Médio (MSE).
- Avaliar o desempenho de modelos de classificação usando métricas como Acurácia, Precisão, Recall e F1-Score.

Revisão Rápida: Aprendizado Supervisionado

No aprendizado supervisionado, o objetivo é que o modelo aprenda uma função que mapeia um conjunto de entradas para uma saída, com base em um conjunto de dados de treinamento rotulados. Esses dados consistem em pares (x, y) , onde x são as features (atributos) de entrada e y é o rótulo (saída desejada). O agente aprende essa função para ser capaz de prever a saída para novos dados não vistos ([Russell; Norvig, 2004, p. 700](#)).

As duas categorias principais de problemas de aprendizado supervisionado são:

- **Regressão:** Previsão de valores de saída **contínuos** (numéricos).
- **Classificação:** Previsão de valores de saída **discretos** (categorias).

Regressão

A **regressão** é uma tarefa de aprendizado supervisionado cujo objetivo é prever um valor de saída contínuo (numérico) com base nas features de entrada. Por exemplo, prever o preço de uma casa, a temperatura do próximo dia ou as vendas de um produto.

Regressão Linear Simples

A Regressão Linear é um dos algoritmos de regressão mais simples e fundamentais. Ela assume que existe uma relação linear entre as features de entrada (variáveis independentes) e a variável de saída (variável dependente).

Para um caso de uma única feature de entrada x , o modelo linear é expresso como:

$$y = mx + b$$

onde:

- y é o valor previsto.
- x é a feature de entrada.
- m é o coeficiente angular (inclinação da linha).
- b é o intercepto (onde a linha cruza o eixo y).

O processo de aprendizado envolve encontrar os valores ótimos de m e b que minimizem o erro entre as previsões do modelo e os valores reais dos dados de treinamento.

Função de Custo para Regressão: Erro Quadrático Médio (MSE)

Uma função de custo comum para regressão é o **Erro Quadrático Médio (Mean Squared Error - MSE)**. Ele calcula a média dos quadrados das diferenças entre os valores previstos ($\hat{y}_i$) e os valores reais (y_i):

$$MSE = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2$$

O objetivo do algoritmo de aprendizado (como o Gradiente Descendente) é minimizar o MSE.

Métricas de Avaliação de Regressão

- **Erro Médio Absoluto (MAE - Mean Absolute Error):** Média dos valores absolutos dos erros. Menos sensível a *outliers* do que o MSE.

$$MAE = \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i|$$

- **Erro Quadrático Médio (MSE - Mean Squared Error):** Já definido, penaliza erros maiores de forma mais severa.
- **Raiz do Erro Quadrático Médio (RMSE - Root Mean Squared Error):** É a raiz quadrada do MSE, tornando a métrica na mesma unidade da variável de saída, o que facilita a interpretação.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2}$$

- **Coefficiente de Determinação (R-squared - R^2):** Indica a proporção da variância na variável dependente que é previsível a partir das variáveis independentes. Varia de 0 a 1, onde 1 indica um ajuste perfeito.

Exemplo em Python: Regressão Linear com Scikit-learn

Vamos usar `scikit-learn` para implementar uma Regressão Linear simples.

Exemplo de Regressão Linear em Python

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# 1. Gerar dados de exemplo
np.random.seed(0)
X = np.random.rand(100, 1) * 10 # Variável independente (features)
y = 2 * X + 3 + np.random.randn(100, 1) * 2 # Variável dependente (target) com ruído

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

print("Dados de treinamento (X_train.shape, y_train.shape):", X_train.shape, y_train.shape)
print("Dados de teste (X_test.shape, y_test.shape):", X_test.shape, y_test.shape)

# 3. Criar e treinar o modelo de Regressão Linear
model = LinearRegression()
model.fit(X_train, y_train)

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

# 5. Avaliar o modelo
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"\nCoeficiente (m): {model.coef_:.2f}")
print(f"Intercepto (b): {model.intercept_:.2f}")
print(f"MAE (Mean Absolute Error): {mae:.2f}")
print(f"MSE (Mean Squared Error): {mse:.2f}")
print(f"RMSE (Root Mean Squared Error): {rmse:.2f}")
print(f"R-squared (R2): {r2:.2f}")

# 6. Visualizar os resultados
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Valores Reais (Teste)')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Previsões do Modelo')
plt.title('Regressão Linear Simples')
plt.xlabel('X (Feature)')
plt.ylabel('Y (Target)')
plt.legend()
plt.grid(True)
plt.show()

```

Classificação

A **classificação** é uma tarefa de aprendizado supervisionado cujo objetivo é prever uma categoria ou classe discreta para uma dada entrada. Por exemplo, classificar um e-mail como “spam” ou “não spam”, um cliente como “inadimplente” ou “adimplente”, ou uma imagem como “gato” ou “cachorro”.

Regressão Logística

A Regressão Logística, apesar do nome “regressão”, é um algoritmo fundamental para problemas de **classificação binária** (duas classes). Em vez de prever um valor contínuo, ela estima a **probabilidade** de uma instância pertencer a uma determinada classe.

A função principal usada na Regressão Logística é a **função sigmoide (ou logística)**, que “espreme” qualquer valor real entre 0 e 1:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

Se a probabilidade estimada for maior que 0.5 (um limiar comum), o modelo classifica a instância como pertencente à classe 1; caso contrário, à classe 0.

Função de Custo para Classificação: Entropia Cruzada

Para a Regressão Logística, a função de custo mais comum é a **Entropia Cruzada (Cross-Entropy Loss)**. Ela penaliza o modelo quando ele prevê uma alta probabilidade para a classe errada. Minimizar a entropia cruzada é equivalente a maximizar a verossimilhança dos dados.

Métricas de Avaliação de Classificação

Para problemas de classificação, as métricas são mais diversas e dependem do balanço entre os tipos de erro.

- **Acurácia (Accuracy):** A proporção de previsões corretas sobre o total de previsões.

$$Acurcia = \frac{\text{Número de Previsões Corretas}}{\text{Número Total de Previsões}}$$

- *Limitação:* Pode ser enganosa em conjuntos de dados desbalanceados.
- **Matriz de Confusão (Confusion Matrix):** Uma tabela que resume o desempenho de um algoritmo de classificação.
 - **Verdadeiro Positivo (VP):** Previu Positivo, é Positivo.
 - **Verdadeiro Negativo (VN):** Previu Negativo, é Negativo.
 - **Falso Positivo (FP):** Previu Positivo, é Negativo (Erro Tipo I).
 - **Falso Negativo (FN):** Previu Negativo, é Positivo (Erro Tipo II).
- **Precisão (Precision):** Dos que o modelo previu como positivos, quantos eram realmente positivos. Importante quando o custo de um Falso Positivo é alto.

$$Preciso = \frac{VP}{VP + FP}$$

- **Recall (Revocação/Sensibilidade):** Dos que eram realmente positivos, quantos o modelo conseguiu identificar. Importante quando o custo de um Falso Negativo é alto.

$$Recall = \frac{VP}{VP + FN}$$

- **F1-Score:** A média harmônica da Precisão e do Recall. Útil quando se busca um equilíbrio entre Precisão e Recall, especialmente em classes desbalanceadas.

$$F1-Score = 2 \times \frac{Preciso \times Recall}{Preciso + Recall}$$

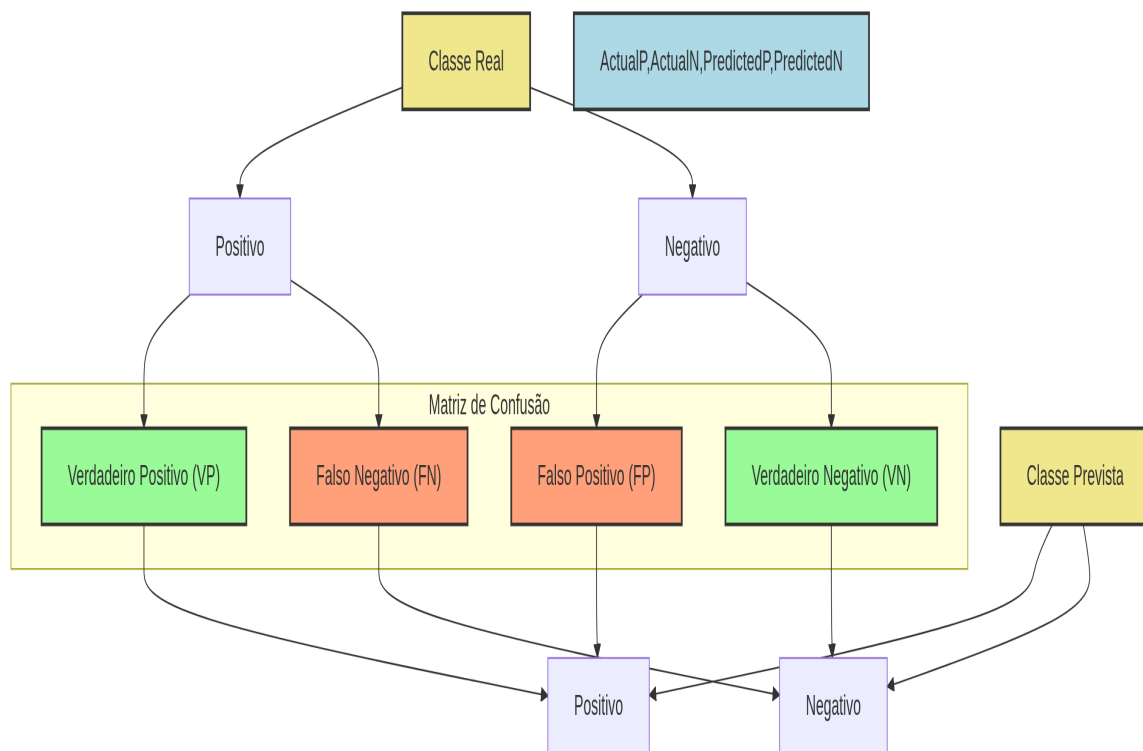


Figure 1: Matriz de Confusão para Classificação Binária

Exemplo em Python: Regressão Logística com Scikit-learn

Vamos usar o famoso dataset Iris para um problema de classificação multiclasse (embora a Regressão Logística seja binária por natureza, `scikit-learn` a estende para multiclasse via estratégias One-vs-Rest por padrão).

Exemplo de Classificação com Regressão Logística em Python

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns

# 1. Carregar dados de exemplo (Dataset Iris)
iris = load_iris()
X, y = iris.data, iris.target
feature_names = iris.feature_names
target_names = iris.target_names

print(f"Features: {feature_names}")
print(f"Classes: {target_names}")
print(f"Formato dos dados X: {X.shape}, y: {y.shape}")

# 2. Dividir 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, stratify=y)

print("\nDados de treinamento (X_train.shape, y_train.shape):", X_train.shape, y_train.shape)
print("Dados de teste (X_test.shape, y_test.shape):", X_test.shape, y_test.shape)

# 3. Criar e treinar o modelo de Regressão Logística
# solver='liblinear' é bom para pequenos datasets e classificação binária/multiclasse simples
# multi_class='ovr' (One-vs-Rest) é o padrão para multiclasse
model = LogisticRegression(solver='liblinear', random_state=42, multi_class='ovr')
model.fit(X_train, y_train)

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

# 5. Avaliar o modelo
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred, target_names=target_names)

print(f"\nAcurácia do modelo: {accuracy:.2f}")

print("\nMatriz de Confusão:")
print(conf_matrix)

print("\nRelatório de Classificação:")
print(class_report)

# 6. Visualizar Matriz de Confusão (opcional, mas muito útil)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=target_names, yticklabels=target_names)
plt.xlabel('Previsto')
plt.ylabel('Real')
plt.title('Matriz de Confusão')
plt.show()

```

Escolhendo entre Regressão e Classificação

A escolha entre um problema de regressão e um de classificação é determinada pela natureza da variável de saída que se deseja prever:

- **Use Regressão:** Quando a variável de saída é um **valor numérico contínuo**.
 - *Exemplos:* Preço de ações, temperatura, tempo de espera, consumo de energia.
- **Use Classificação:** Quando a variável de saída é uma **categoria discreta**.
 - *Exemplos:* Sim/Não, A/B/C, Tipo de objeto (carro, caminhão, moto), Diagnóstico médico (doente, saudável).

A escolha do algoritmo específico dentro de cada categoria dependerá de fatores como a complexidade da relação nos dados, o volume de dados, a interpretabilidade desejada e os recursos computacionais disponíveis.

Considerações Finais

Regressão e Classificação são as espinhas dorsais do aprendizado supervisionado, capacitando os sistemas de IA a fazer previsões e tomar decisões em uma miríade de aplicações do mundo real. Desde a previsão do tempo até o diagnóstico médico, esses algoritmos formam a base de soluções inteligentes. É crucial não apenas entender o funcionamento desses algoritmos, mas também saber como avaliar seu desempenho usando as métricas apropriadas para cada tipo de problema. Nas próximas aulas, exploraremos outros algoritmos e conceitos mais avançados de Machine Learning.

Verificação de Aprendizagem

Responda às seguintes questões e implemente as tarefas propostas para solidificar seu entendimento.

1. **Regressão vs. Classificação:**
 - a) Qual é a principal diferença entre um problema de regressão e um problema de classificação? Dê um exemplo de um cenário real para cada um, diferente dos mencionados na aula.
 - b) Por que a Regressão Logística, apesar do nome, é considerada um algoritmo de classificação?
2. **Métricas de Regressão:** Você treinou um modelo de regressão para prever o tempo de entrega de um pacote. Qual das métricas MAE, MSE ou R-squared você usaria para:
 - a) Entender o erro médio absoluto de previsão em unidades de tempo?
 - b) Penalizar fortemente previsões muito distantes do valor real?
 - c) Entender a proporção da variância no tempo de entrega que seu modelo consegue explicar?
3. **Métricas de Classificação:** Imagine que você está desenvolvendo um modelo de classificação para detectar uma doença rara. O custo de um **falso negativo** (não detectar a doença em alguém que realmente a tem) é muito alto.
 - a) Qual métrica (Acurácia, Precisão, Recall, F1-Score) você priorizaria para avaliar o desempenho do seu modelo neste cenário? Justifique sua escolha.
 - b) O que significaria um alto valor para essa métrica no contexto do problema?
4. **Implementação Prática (Conceitual):** Considere um dataset com duas features (idade, renda) e uma variável alvo.
 - **Problema A:** Prever o *valor de compra* que um cliente fará na próxima visita (um número real).

- **Problema B:** Prever se um cliente *comprará* ou *não comprará* um produto na próxima visita (Sim/Não).
- a) Para o **Problema A**, qual tipo de tarefa de ML seria? Sugira um algoritmo (`scikit-learn`) e uma métrica de avaliação.
- b) Para o **Problema B**, qual tipo de tarefa de ML seria? Sugira um algoritmo (`scikit-learn`) e uma métrica de avaliação.
- c) Escreva um pseudocódigo (ou descreva os passos com código simplificado) em Python para o Problema B, incluindo:
 - Geração ou carregamento de dados fictícios.
 - Divisão em treino/teste.
 - Criação e treinamento do modelo.
 - Previsão e avaliação (com a métrica que você priorizou).

Referências Bibliográficas

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência Artificial: Um Enfoque Moderno**. 2. ed. Rio de Janeiro: Prentice Hall, 2004.