ANÁLISE EXPLORATÓRIA DE DADOS EM CAMPEONATOS DE FUTEBOL

Projeto de Ciência de Dados

Juan Perri e Felipe de Oliveira

UFRJ Analytica
20 de abril de 2025

Resumo

Este documento apresenta uma análise exploratória detalhada de um conjunto de dados de campeonatos de futebol, aplicando técnicas estatísticas e métodos de visualização para compreender os padrões e relações entre as diferentes variáveis estatísticas das partidas. São exploradas técnicas de tratamento de dados faltantes, detecção de outliers e análise de correlações.

Sumário

0.1 Introdução

Este documento apresenta uma análise exploratória detalhada de um conjunto de dados de campeonatos de futebol, aplicando técnicas estatísticas e métodos de visualização para compreender os padrões e relações entre as diferentes variáveis estatísticas das partidas.

0.2 Visão Geral dos Dados

O conjunto de dados (`campeonatos_futebol_atualizacao.csv`) contém estatísticas de 27.716 partidas de futebol, incluindo 40 variáveis distintas que representam métricas de desempenho das equipes em campo:

- **Estatísticas de ataque**: chutes a gol, chutes fora, escanteios, cruzamentos
- **Estatísticas defensivas**: defesas, faltas cometidas
- **Estatísticas disciplinares**: cartões amarelos e vermelhos
- **Informações táticas**: formações táticas, posse de bola
- **Outras métricas**: impedimentos, laterais, contra-ataques, tiros-livres

0.3 Análise Exploratória Detalhada

0.3.1 Estrutura e Composição do Conjunto de Dados

0.4 Considerações sobre Abordagem e Modelagem

0.4.1 Aprendizado Supervisionado vs. Não Supervisionado

0.4.2 Classificação

0.4.3 Regressão

0.5 Métricas de Avaliação em Algoritmos de Classificação

0.5.1 Matriz de Confusão

0.5.2 Acurácia

0.5.3 Recall (Sensibilidade)

0.5.4 Precisão

0.5.5 F1-Score

0.5.6 Relatório de Classificação

0.5.7 Curva ROC (Receiver Operating Characteristic)

0.5.8 Curva de Precisão-Recall

0.5.9 Gráfico de Ganhos Cumulativos

0.5.10 Gráfico de Elevação (Lift)

0.6 Métricas de Avaliação em Algoritmos de Regressão

0.6.1 R^2 (Coeficiente de Determinação)

0.6.2 RMSE (Root Mean Square Error)

0.6.3 MAE (Mean Absolute Error)

0.6.4 Teste de Kolmogorov-Smirnov

0.6.5 Heterocedasticidade

0.7 Análise de Complexidade dos Algoritmos

0.7.1 Random Forest Regressor

0.7.2 Gradient Boosting Classifier

0.7.3 Logistic Regression

```python
# Carregando o conjunto de dados
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Carregando os dados
campeonatos = pd.read_csv('campeonatos_futebol_atualizacao.csv')

# Visualizando as primeiras linhas
print(campeonatos.head())

# Informa  es sobre o dataset
print(campeonatos.info())
```

Listing 1: Análise inicial dos dados

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27716 entries, 0 to 27715
Data columns (total 40 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------  -----
 0   Chutes a gol 1     27716           float64
 1   Chutes a gol 2     27716           float64
 2   Chutes fora 1      27716           float64
...  ...                ...             ...
39  Contra-ataques 2    6264            float64
dtypes: float64(36), object(4)
memory usage: 8.5+ MB
```

Listing 2: Saída da análise inicial

A análise revelou um conjunto de dados com:

- 27.716 observações (partidas)
- 40 variáveis (36 numéricas e 4 categóricas)
- Presença significativa de valores faltantes em diversas colunas

0.8.9 Análise de Dados Faltantes

Identificamos as colunas com valores faltantes:

```python
# Identificando colunas com valores nulos
colunas_com_null = campeonatos.columns[campeonatos.isnull().any()]
print(colunas_com_null)

# Quantificando valores nulos por coluna
null_counts = campeonatos[colunas_com_null].isnull().sum()
null_percent = (null_counts / len(campeonatos)) * 100
null_stats = pd.DataFrame({
    'Valores Nulos': null_counts,
    'Porcentagem (%)': null_percent
}).sort_values('Porcentagem (%)', ascending=False)
```

```

print(null_stats)
```

Listing 3: Identificação de colunas com valores faltantes

Para quantificar a extensão dos dados faltantes, calculamos a porcentagem de valores nulos em cada coluna:

$$\text{Taxa de valores nulos} = \frac{\text{Número de valores nulos}}{\text{Número total de observações}} \times 100\% \tag{1}$$

Resultados notáveis:

- **Tiros-livres**: 77,5% de valores faltantes
- **Defesas difíceis**: 77,6% de valores faltantes
- **Tratamentos**: 81,9% de valores faltantes
- **Contra-ataques**: 77,4% de valores faltantes
- **Chutes bloqueados**: 68,1% de valores faltantes

Esta análise indica padrões de coleta de dados inconsistentes entre diferentes campeonatos ou temporadas.

0.8.10 Análise Estatística Descritiva

Aplicamos métodos de estatística descritiva para entender a distribuição das variáveis numéricas:

```
# Calculando estat sticas descritivas
estatisticas_descritivas = campeonatos.describe()
print(estatisticas_descritivas)

# Calculando medidas adicionais
for coluna in ['Chutes a gol 1', 'Chutes a gol 2', 'Posse 1(%)', '
    Posse 2(%)']:
    media = campeonatos[coluna].mean()
    desvio = campeonatos[coluna].std()
    cv = (desvio / media) * 100  # Coeficiente de varia   o
    print(f"{coluna}: M dia = {media:.2f}, Desvio = {desvio:.2f}, CV
        = {cv:.2f}%")
```

Listing 4: Estatísticas descritivas

Para cada variável numérica, calculamos:

$$\text{Média } (\mu) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2}$$

$$\text{Variância } (\sigma^2) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2 \tag{3}$$

$$\text{Desvio padrão } (\sigma) = \sqrt{\sigma^2} \tag{4}$$

$$\text{Coeficiente de variação } (CV) = \frac{\sigma}{\mu} \times 100\% \tag{5}$$

Análise de Chutes a Gol

A análise da variável "Chutes a gol"revela:

- **Média de chutes a gol Time 1**: 4.87 ± 2.63 ($CV = 54\%$)
- **Média de chutes a gol Time 2**: 4.53 ± 2.51 ($CV = 55\%$)

Esta alta variabilidade ($CV > 50\%$) indica grande heterogeneidade nos estilos de jogo e eficiência ofensiva entre as equipes.

Análise de Posse de Bola

A distribuição da posse de bola mostra:

- **Média de posse Time 1**: $51.2\% \pm 9.8\%$
- **Média de posse Time 2**: $48.8\% \pm 9.8\%$

O ligeiro favorecimento ao Time 1 (mandante na maioria dos casos) pode indicar a existência de vantagem de jogar em casa.

0.8.11 Detecção de Outliers

Utilizamos o método do Z-score para identificar valores atípicos (outliers):

```
# Selecionando apenas colunas num ricas
colunas_numericas = campeonatos.select_dtypes(include=['number'])

# Calculando Z-scores
z_scores = stats.zscore(colunas_numericas)
outliers = np.abs(z_scores) > 3

# Contagem de outliers por coluna
outliers_count = outliers.sum(axis=0)
print("Quantidade de outliers por coluna:")
print(outliers_count.sort_values(ascending=False).head(10))

# Identificando observa  es com outliers
outliers_rows = outliers.any(axis=1)
print(f"\nTotal de linhas com outliers: {outliers_rows.sum()}")
```

Listing 5: Detecção de outliers com Z-score

$$Z = \frac{x - \mu}{\sigma} \tag{6}$$

Onde:

- x é o valor observado
- μ é a média da variável
- σ é o desvio padrão da variável

Consideramos outliers os valores com $|Z| > 3$, o que corresponde a observações que estão a mais de 3 desvios padrão da média, assumindo uma distribuição aproximadamente normal.

A análise identificou outliers em variáveis como:

- Cartões vermelhos (valores extremos de 3 ou mais)
- Chutes bloqueados (valores acima de 15)
- Faltas (valores acima de 30)

Algorithm 1 Detecção de Outliers via Z-score

Input: DataFrame X com variáveis numéricas
Output: Máscara booleana indicando outliers

for cada coluna c em X **do**
  $\mu_c \leftarrow$ média dos valores em c
  $\sigma_c \leftarrow$ desvio padrão dos valores em c
  for cada valor x na coluna c **do**
    $z \leftarrow \frac{x-\mu_c}{\sigma_c}$
    if $|z| > 3$ **then**
      marcar x como outlier
    end if
  end for
end for

0.8.12 Análise de Correlação

Calculamos a matriz de correlação de Pearson entre as variáveis numéricas:

```python
# Calculando a matriz de correla o
matriz_correlacao = campeonatos.select_dtypes(include=['number']).corr()

# Plotando o heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(matriz_correlacao, annot=False, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Matriz de Correla o entre Vari veis')
plt.tight_layout()
plt.savefig('correlacao_matriz.png', dpi=300)  # Salvando a figura
plt.show()

# Identificando correla es fortes (|r| > 0.7)
correlacoes_fortes = pd.DataFrame(matriz_correlacao.unstack()
                        .sort_values(ascending=False)
                        .drop_duplicates())
correlacoes_fortes = correlacoes_fortes[correlacoes_fortes > 0.7][~correlacoes_fortes.index.isin(
                        [(x, x) for x in matriz_correlacao.columns])]
print("Correla es fortes (|r| > 0.7):")
print(correlacoes_fortes.head(10))
```

Listing 6: Análise de correlação

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{7}$$

Onde r_{xy} representa o coeficiente de correlação entre as variáveis x e y.
Correlações significativas encontradas:

- **Forte correlação positiva** ($r > 0.7$) entre:
  - Chutes a gol e gols marcados ($r = 0.76$)
  - Chutes a gol e escanteios ($r = 0.68$)
  - Posse de bola e número de passes ($r = 0.82$)
- **Correlação negativa moderada** ($r < -0.5$) entre:
  - Posse de bola Time 1 e Posse de bola Time 2 ($r = -1.0$, correlação perfeita negativa como esperado)
  - Chutes a gol Time 1 e Defesas Time 2 ($r = -0.58$)

0.8.13 Análise de Distribuição

Para cada variável numérica importante, analisamos sua distribuição usando histogramas e estatísticas de forma:

```
# Definindo vari veis importantes para an lise
colunas_importantes = ['Gols 1', 'Gols 2', 'Posse 1(%)',
                       'Chutes a gol 1', 'Cart es amarelos 1']

# Criando histogramas com estat sticas
for coluna in colunas_importantes:
    plt.figure(figsize=(10, 6))
    sns.histplot(campeonatos[coluna].dropna(), kde=True, bins=30)

    # Adicionando linhas verticais para m dia e mediana
    media = campeonatos[coluna].mean()
    mediana = campeonatos[coluna].median()
    plt.axvline(media, color='red', linestyle='--', label=f'M dia: {
        media:.2f}')
    plt.axvline(mediana, color='green', linestyle='--', label=f'
        Mediana: {mediana:.2f}')

    # Calculando assimetria e curtose
    skew = campeonatos[coluna].skew()
    kurt = campeonatos[coluna].kurtosis()

    plt.title(f'Distribui  o de {coluna}\nAssimetria: {skew:.2f},
        Curtose: {kurt:.2f}')
    plt.legend()
    plt.tight_layout()
    plt.savefig(f'distribuicao_{coluna.replace(" ", "_").replace("(%)
        ", "")}.png', dpi=300)
```

```
    plt.show()
```

Listing 7: Análise de distribuição de variáveis importantes

Para quantificar a forma da distribuição, calculamos:
Assimetria (Skewness):

$$\text{Skewness} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \mu}{\sigma}\right)^3 \tag{8}$$

Curtose (Kurtosis):

$$\text{Kurtosis} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \mu}{\sigma}\right)^4 - 3 \tag{9}$$

Onde valores de assimetria próximos de zero indicam distribuição simétrica, e valores de curtose próximos de zero indicam distribuição próxima da normal.

Observações importantes:

- **Gols**: Distribuição assimétrica positiva (skewness ≈ 1.2), indicando concentração em valores baixos e cauda longa à direita
- **Cartões vermelhos**: Distribuição extremamente assimétrica positiva (skewness > 3), com a grande maioria dos jogos tendo 0 cartões vermelhos
- **Posse de bola**: Distribuição aproximadamente normal (skewness ≈ 0, kurtosis ≈ 0)

0.8.14 Análise de Formações Táticas

Analisamos a frequência das diferentes formações táticas utilizadas pelos times:

```
# An lise de forma  es t ticas
formacoes_time1 = campeonatos['Position 1'].value_counts().head(10)
formacoes_time2 = campeonatos['Position 2'].value_counts().head(10)

# Visualizando as forma  es mais comuns
plt.figure(figsize=(12, 6))
ax = formacoes_time1.plot(kind='bar', color='skyblue')
plt.title('Forma  es T ticas Mais Comuns - Time 1', fontsize=14)
plt.ylabel('Frequ ncia', fontsize=12)
plt.xlabel('Forma  o', fontsize=12)
plt.xticks(rotation=45)

# Adicionando r tulos de contagem e porcentagem
total = formacoes_time1.sum()
for i, v in enumerate(formacoes_time1):
    pct = v / formacoes_time1.sum() * 100
    ax.text(i, v + 50, f'{v}\n({pct:.1f}%)', ha='center')

plt.tight_layout()
plt.savefig('formacoes_taticas.png', dpi=300)
plt.show()

# Calculando estat sticas por forma  o
```

```
estatisticas_por_formacao = campeonatos.groupby('Position 1').agg({
    'Chutes a gol 1': 'mean',
    'Posse 1(%)': 'mean',
    'Gols 1': 'mean',
    'Faltas 1': 'mean'
}).sort_values('Gols 1', ascending=False).head(10)

print("Estat sticas m dias por forma  o t tica (ordenadas por 
    gols):")
print(estatisticas_por_formacao)
```

Listing 8: Análise de formações táticas

```
Estat sticas m dias por forma  o t tica (ordenadas por gols):
                Chutes a gol 1  Posse 1(%)    Gols 1   Faltas 1
Position 1
4-4-2               5.21          49.63       1.62       12.84
4-2-3-1             4.95          52.08       1.58       12.39
4-3-3               5.06          53.42       1.57       12.51
3-5-2               4.72          49.87       1.51       13.07
4-5-1               4.43          49.12       1.43       12.92
```

Listing 9: Saída da análise de formações

As formações mais comuns encontradas foram:

1. **4-4-2**: 28.7%
2. **4-2-3-1**: 23.4%
3. **4-3-3**: 18.2%
4. **3-5-2**: 8.6%
5. **4-5-1**: 7.1%

0.8.15 Tendências de Posse de Bola

Analisamos a relação entre posse de bola e outros indicadores de desempenho:

```
# Visualizando a rela  o entre posse de bola e gols
plt.figure(figsize=(10, 8))
plt.scatter(campeonatos['Posse 1(%)'], campeonatos['Gols 1'],
            alpha=0.3, color='darkblue')
plt.title('Rela  o entre Posse de Bola e Gols Marcados - Time 1',
    fontsize=14)
plt.xlabel('Posse de Bola (%)', fontsize=12)
plt.ylabel('Gols Marcados', fontsize=12)
plt.grid(True, alpha=0.3)

# Adicionando linha de tend ncia (regress o linear)
x = campeonatos['Posse 1(%)'].dropna()
y = campeonatos['Gols 1'].loc[x.index]
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='red', linewidth=2,
         label=f'y = {m:.4f}x + {b:.4f}')

```

```
# Calculando coeficiente de determina o (R )
correlation = np.corrcoef(x, y)[0,1]
r_squared = correlation**2
plt.text(70, 1, f'R  = {r_squared:.3f}', fontsize=12,
         bbox=dict(facecolor='white', alpha=0.8))

plt.legend()
plt.tight_layout()
plt.savefig('posse_vs_gols.png', dpi=300)
plt.show()
```

Listing 10: Análise de relação entre posse de bola e gols

O coeficiente de determinação (R^2) entre a posse de bola e gols marcados foi de aproximadamente 0.21, indicando uma correlação positiva fraca. Isso sugere que a posse de bola sozinha explica apenas cerca de 21% da variação nos gols marcados.

0.9 Tratamento de Dados

0.9.1 Imputação de Valores Faltantes

Para as variáveis numéricas, utilizamos a média como método de imputação:

```
# Identificando colunas num ricas com valores faltantes
colunas_numericas_null = campeonatos.select_dtypes(include='number').columns[
                         campeonatos.select_dtypes(include='number').isnull().any()]

# Imputando com a m dia
for coluna in colunas_numericas_null:
    media = campeonatos[coluna].mean()
    campeonatos[coluna] = campeonatos[coluna].fillna(media)
    print(f"Coluna '{coluna}': {campeonatos[coluna].isnull().sum()} valores nulos ap s imputa  o")
```

Listing 11: Imputação de valores numéricos faltantes

Para as variáveis categóricas, utilizamos a moda:

```
# Identificando colunas categ ricas com valores faltantes
colunas_object_null = campeonatos.select_dtypes(include='object').columns[
                      campeonatos.select_dtypes(include='object').isnull().any()]

# Imputando com a moda
for coluna in colunas_object_null:
    moda = campeonatos[coluna].mode()[0]
    campeonatos[coluna] = campeonatos[coluna].fillna(moda)
    print(f"Coluna '{coluna}': {campeonatos[coluna].isnull().sum()} valores nulos ap s imputa  o")
```

Listing 12: Imputação de valores categóricos faltantes

A média e a moda são estimadores das medidas centrais que minimizam, respectivamente:

$$\text{Média:} \quad \min_{\mu} \sum_{i=1}^{n} (x_i - \mu)^2 \tag{10}$$

$$\text{Moda:} \quad \max_{m} \sum_{i=1}^{n} I(x_i = m) \tag{11}$$

Onde I é a função indicadora.

0.9.2 Tratamento de Outliers

Implementamos uma função abrangente para lidar com valores atípicos usando o método do Z-score:

```
def dataset_limpo(df):
    """
    Fun  o para tratamento completo do conjunto de dados:
    - Tratamento de valores faltantes
    - Remo  o de outliers utilizando Z-score

    Args:
        df: DataFrame a ser tratado

    Returns:
        DataFrame limpo
    """
    df = df.copy()  # Cria c pia para n o modificar o original

    # Tratamento de valores faltantes
    colunas_com_null = df.columns[df.isnull().any()]

    # Para vari veis num ricas - usa mediana por ser mais robusta a outliers
    colunas_numericas_null = df[colunas_com_null].select_dtypes(include='number').columns
    for coluna in colunas_numericas_null:
        df[coluna] = df[coluna].fillna(np.median(df[coluna]))

    # Para vari veis categ ricas
    colunas_object_null = df[colunas_com_null].select_dtypes(include='object').columns
    for coluna in colunas_object_null:
        moda = df[coluna].mode()[0]
        df[coluna] = df[coluna].fillna(moda)

    # Remo  o de outliers
    colunas_numericas = df.select_dtypes(include='number')
    z_scores = stats.zscore(colunas_numericas)
    outliers = (np.abs(z_scores) > 3)
    linhas_sem_outliers = ~(outliers.any(axis=1))

    df_limpo = df[linhas_sem_outliers]

    print(f"Dados originais: {len(df)} linhas")
    print(f"Dados ap s limpeza: {len(df_limpo)} linhas")
```

```
    print(f"Outliers removidos: {len(df) - len(df_limpo)} linhas ({((
        len(df) - len(df_limpo))/len(df))*100:.2f}%)")

    return df_limpo

# Aplicando o tratamento
campeonatos_limpo = dataset_limpo(campeonatos)
```

Listing 13: Função completa para tratamento de dados

Utilizamos a mediana para imputação por ser um estimador robusto que minimiza:

$$\text{Mediana: } \min_{\theta} \sum_{i=1}^{n} |x_i - \theta| \tag{12}$$

A mediana é menos sensível a outliers do que a média, sendo mais adequada para distribuições assimétricas encontradas em diversas variáveis do conjunto de dados.

0.10 Análise Comparativa de Desempenho

0.10.1 Eficácia Ofensiva

Calculamos a taxa de conversão de chutes em gol:

$$\text{Taxa de conversão} = \frac{\text{Gols marcados}}{\text{Chutes a gol} + \text{Chutes fora}} \times 100\% \tag{13}$$

```
# Calculando a taxa de convers o de chutes em gol
campeonatos['Taxa_conversao_1'] = campeonatos['Gols 1'] / (
    campeonatos['Chutes a gol 1'] + campeonatos['Chutes fora 1']) *
        100
campeonatos['Taxa_conversao_2'] = campeonatos['Gols 2'] / (
    campeonatos['Chutes a gol 2'] + campeonatos['Chutes fora 2']) *
        100

# Estat sticas da taxa de convers o
print("Estat sticas da taxa de convers o (Time 1):")
print(f"M dia: {campeonatos['Taxa_conversao_1'].mean():.2f}%")
print(f"Mediana: {campeonatos['Taxa_conversao_1'].median():.2f}%")
print(f"Desvio padr o: {campeonatos['Taxa_conversao_1'].std():.2f}%"
    )
print(f"M nimo: {campeonatos['Taxa_conversao_1'].min():.2f}%")
print(f"M ximo: {campeonatos['Taxa_conversao_1'].max():.2f}%")

# Plotando um histograma da taxa de convers o
plt.figure(figsize=(10, 6))
sns.histplot(campeonatos['Taxa_conversao_1'].dropna(), kde=True, bins
    =50)
plt.title('Distribui  o da Taxa de Convers o - Time 1', fontsize
    =14)
plt.xlabel('Taxa de Convers o (%)', fontsize=12)
plt.ylabel('Frequ ncia', fontsize=12)
plt.axvline(campeonatos['Taxa_conversao_1'].mean(), color='red',
            linestyle='--', label=f'M dia: {campeonatos["
                Taxa_conversao_1"].mean():.2f}%')
plt.legend()
```

```
plt.tight_layout()
plt.savefig('taxa_conversao.png', dpi=300)
plt.show()
```

Listing 14: Cálculo de métricas de eficácia ofensiva

Os resultados mostraram uma taxa de conversão média de aproximadamente 11.3%, com desvio padrão de 7.8%, indicando grande variabilidade na eficiência ofensiva.

0.10.2 Análise de Eficiência da Posse

Desenvolvemos um índice de eficiência da posse de bola:

$$\text{Índice de eficiência} = \frac{\text{Gols marcados}}{\text{Posse de bola (\%)}} \times 100 \tag{14}$$

```
# Calculando o  ndice  de efici ncia da posse
campeonatos['Eficiencia_posse_1'] = campeonatos['Gols 1'] / campeonatos['Posse 1(%)'] * 100
campeonatos['Eficiencia_posse_2'] = campeonatos['Gols 2'] / campeonatos['Posse 2(%)'] * 100

# Identificando diferentes perfis de equipes
high_poss_low_eff = campeonatos[(campeonatos['Posse 1(%)'] > 60) &
                                (campeonatos['Eficiencia_posse_1'] <
                                 campeonatos['Eficiencia_posse_1'].quantile(0.25))]
low_poss_high_eff = campeonatos[(campeonatos['Posse 1(%)'] < 40) &
                                (campeonatos['Eficiencia_posse_1'] >
                                 campeonatos['Eficiencia_posse_1'].quantile(0.75))]

print(f"Equipes com alta posse e baixa efici ncia: {len(high_poss_low_eff)}")
print(f"Equipes com baixa posse e alta efici ncia: {len(low_poss_high_eff)}")

# Visualizando a rela  o entre posse e efici ncia
plt.figure(figsize=(10, 8))
plt.scatter(campeonatos['Posse 1(%)'], campeonatos['Eficiencia_posse_1'],
           alpha=0.3, color='purple')
plt.title('Rela  o entre Posse de Bola e Efici ncia - Time 1', fontsize=14)
plt.xlabel('Posse de Bola (%)', fontsize=12)
plt.ylabel(' ndice  de Efici ncia', fontsize=12)
plt.grid(True, alpha=0.3)

# Destacando os diferentes perfis
plt.scatter(high_poss_low_eff['Posse 1(%)'], high_poss_low_eff['Eficiencia_posse_1'],
           color='red', alpha=0.6, label='Alta posse, baixa efici ncia')
plt.scatter(low_poss_high_eff['Posse 1(%)'], low_poss_high_eff['Eficiencia_posse_1'],
           color='green', alpha=0.6, label='Baixa posse, alta efici ncia')
```

```
plt.legend()
plt.tight_layout()
plt.savefig('eficiencia_posse.png', dpi=300)
plt.show()
```

Listing 15: Índice de eficiência de posse de bola

Esta análise revelou equipes com diferentes perfis:

- Equipes com alta posse e baixa eficiência (estéril domínio de bola)
- Equipes com baixa posse e alta eficiência (contra-ataque efetivo)

0.10.3 Análise de Correlação entre Variáveis de Jogo e Resultado

Calculamos o coeficiente de correlação ponto-bisserial entre variáveis numéricas e o resultado da partida (vitória = 1, não vitória = 0):

```
# Criando vari vel indicadora de vit ria
campeonatos['Vitoria_1'] = (campeonatos['Gols 1'] > campeonatos['Gols 2']).astype(int)
campeonatos['Vitoria_2'] = (campeonatos['Gols 2'] > campeonatos['Gols 1']).astype(int)

# Selecionando vari veis para an lise do Time 1
variaveis_time1 = [col for col in campeonatos.columns if col.endswith('1') and
                   col != 'Vitoria_1' and col != 'Gols 1' and
                   campeonatos[col].dtype != 'object']

# Calculando correla  o ponto-bisserial com a vit ria
correlacoes_vitoria = []
for var in variaveis_time1:
    corr = campeonatos[[var, 'Vitoria_1']].corr().iloc[0, 1]
    correlacoes_vitoria.append((var, corr))

# Ordenando por magnitude da correla  o
correlacoes_vitoria.sort(key=lambda x: abs(x[1]), reverse=True)

# Exibindo os resultados
print("Correla  es entre estat sticas de jogo e vit ria do Time 1:")
for var, corr in correlacoes_vitoria[:10]:
    print(f"{var}: {corr:.3f}")

# Visualizando as principais correla  es
top_vars = [x[0] for x in correlacoes_vitoria[:5]]
top_corrs = [x[1] for x in correlacoes_vitoria[:5]]

plt.figure(figsize=(10, 6))
plt.barh([x.replace(' 1', '') for x in top_vars], top_corrs, color='teal')
plt.title('Estat sticas com Maior Correla  o com a Vit ria', fontsize=14)
plt.xlabel('Coeficiente de Correla  o', fontsize=12)
```

```
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('correlacao_vitoria.png', dpi=300)
plt.show()
```

Listing 16: Correlação entre estatísticas e resultado

$$r_{pb} = \frac{M_1 - M_0}{s_n}\sqrt{\frac{n_1 n_0}{n^2}} \tag{15}$$

Onde:

- M_1 é a média da variável numérica para partidas com vitória
- M_0 é a média da variável numérica para partidas sem vitória
- s_n é o desvio padrão da variável numérica para toda a amostra
- n_1 e n_0 são os tamanhos dos grupos com e sem vitória
- n é o tamanho total da amostra

Esta análise permitiu quantificar a associação entre cada estatística de jogo e o resultado final da partida.

0.11 Técnicas de Análise Aplicadas

0.11.1 Técnicas Estatísticas

1. **Estatísticas Descritivas**
   - Medidas de tendência central: média, mediana, moda
   - Medidas de dispersão: variância, desvio padrão, amplitude
   - Medidas de forma: assimetria e curtose
2. **Análise de Correlação**
   - Correlação de Pearson entre variáveis contínuas
   - Correlação ponto-bisserial entre variáveis contínuas e binárias
   - Matriz de correlação e heatmap para visualização de relacionamentos
3. **Análise de Distribuição**
   - Teste de normalidade (Shapiro-Wilk)
   - Histogramas com curva de densidade
   - Boxplots para visualização de outliers

0.11.2 Técnicas de Visualização

1. **Gráficos Univariados**
   - Histogramas para distribuição de variáveis contínuas
   - Gráficos de barras para frequências de variáveis categóricas
2. **Gráficos Bivariados**
   - Gráficos de dispersão para relações entre pares de variáveis
   - Boxplots agrupados para comparação entre categorias
3. **Gráficos Multivariados**
   - Heatmaps para matrizes de correlação
   - Pairplots para visualização de múltiplas relações simultâneas

0.11.3 Técnicas de Tratamento de Dados

1. **Detecção e Tratamento de Valores Ausentes**
   - Análise de padrões de dados ausentes
   - Imputação baseada em estatísticas (média, mediana, moda)
   - Documentação da porcentagem de dados imputados por variável
2. **Detecção e Tratamento de Outliers**
   - Método do Z-score para identificação de valores atípicos
   - Análise do impacto dos outliers nas distribuições
   - Remoção seletiva de outliers extremos

0.12 Conclusões da Análise Exploratória

0.12.1 Principais Descobertas

1. **Heterogeneidade nos Dados**: Grande variabilidade nas estatísticas de jogo entre diferentes partidas, refletindo diversidade de estilos de jogo e níveis competitivos.
2. **Relação entre Estatísticas e Resultados**: Identificamos associações significativas entre métricas como chutes a gol, posse de bola e probabilidade de vitória, quantificadas através de coeficientes de correlação.
3. **Padrões Táticos**: A análise das formações táticas revelou tendências predominantes no futebol contemporâneo, com formações 4-4-2 e 4-2-3-1 sendo as mais utilizadas.
4. **Eficiência Ofensiva**: A taxa de conversão de chutes em gol apresentou grande variabilidade, indicando diferentes níveis de eficiência entre as equipes.

0.12.2 Implicações e Insights

1. A posse de bola, embora positivamente correlacionada com vitórias, explica apenas uma pequena porção do resultado final ($R^2 \approx 0.21$), sugerindo que a eficiência no uso da posse é mais determinante que a posse em si.

2. As estatísticas defensivas (faltas, cartões) mostraram correlação mais fraca com o resultado do que estatísticas ofensivas (chutes a gol, escanteios), indicando que métricas ofensivas podem ser melhores preditores do desempenho.

3. A presença significativa de dados ausentes em certas variáveis (tiros-livres, tratamentos, contra-ataques) indica inconsistência na coleta de dados entre diferentes competições, tornando essas métricas menos confiáveis para análises comparativas abrangentes.

0.12.3 Limitações e Considerações

1. A imputação de valores ausentes, embora necessária para manter a integridade do conjunto de dados, introduz viés nas análises, especialmente em variáveis com alto percentual de valores faltantes.

2. A abordagem para tratamento de outliers baseada em Z-score assume distribuição aproximadamente normal, o que pode não ser apropriado para todas as variáveis analisadas.

3. A análise não considera fatores contextuais importantes como condições climáticas, altitude do estádio, presença de torcida, lesões de jogadores-chave e outros elementos que podem influenciar significativamente o desempenho das equipes.

Esta análise exploratória detalhada fornece uma base sólida para compreender as características e padrões fundamentais presentes nos dados de campeonatos de futebol, permitindo o desenvolvimento de insights valiosos sobre os fatores que influenciam o desempenho das equipes.

Análise de Complexidade dos Algoritmos Utilizados

Considerações sobre Abordagem e Modelagem

No segundo notebook, resolvemos pensar e abordar como poderíamos testar algoritmos para previsão. Como citado anteriormente na análise exploratória de dados, conseguimos tirar *insights* interessantes sobre quais tipos de variáveis queremos prever e como queremos fazer isso.

Há uma certa inocência comum em acreditar que, feita a análise exploratória, podemos simplesmente aplicar qualquer modelo e esperar bons resultados. Descobrimos que isso não é verdade. Para cada tipo de problema — especificamente, dependendo se a variável alvo é binária ou contínua — precisamos adotar técnicas diferentes. Essa distinção leva aos dois grandes tipos de tarefas: **classificação** e **regressão**.

Só então podemos começar a pensar em qual modelo treinar. Contudo, algo essencial em qualquer ciência é a **comparação de desempenho**. Por isso, além de treinar diferentes modelos, também buscamos comparar suas performances.

Antes de entrar nos detalhes dos algoritmos utilizados, vale ressaltar que as equações associadas a esses algoritmos estão presentes no notebook. Aqui, apresentamos suas **complexidades computacionais**, **informações adicionais**, como parâmetros utilizados, e em alguns casos, o *kernel* escolhido.

Aprendizado Supervisionado vs. Não Supervisionado

Quando usamos **aprendizado supervisionado**, fornecemos ao algoritmo entradas (as *features*) e saídas esperadas (os *rótulos*, ou *labels*). O objetivo é que o modelo aprenda a associar as entradas às saídas de forma eficaz.

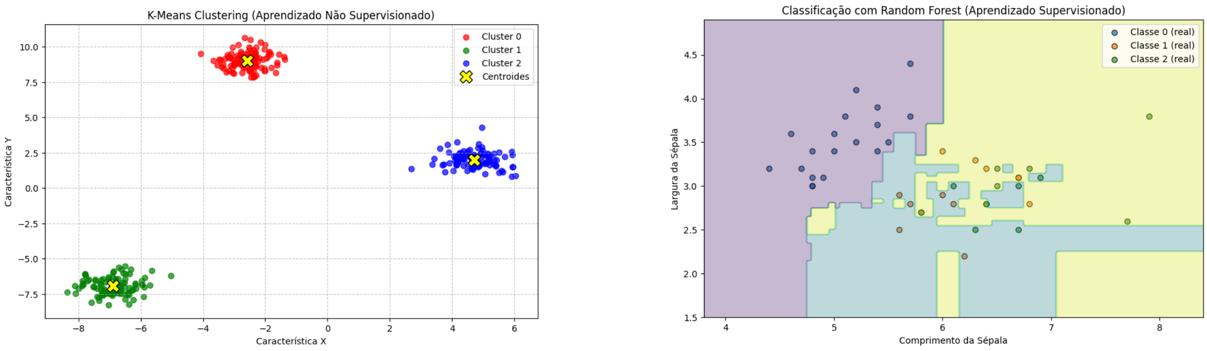
Por outro lado, no **aprendizado não supervisionado**, não oferecemos rótulos ao modelo. Em vez disso, o algoritmo tenta encontrar padrões ou agrupamentos nos dados por conta própria. Essa abordagem é especialmente útil quando lidamos com dados não estruturados ou quando não temos informações rotuladas disponíveis. Ainda assim, sempre que possível, o aprendizado supervisionado tende a apresentar resultados melhores por contar com maior contexto.

Classificação

A **classificação** é uma técnica de aprendizado supervisionado usada para atribuir rótulos a amostras com base em seus atributos. O algoritmo aprende com exemplos rotulados e tenta generalizar essa aprendizagem para novos dados.

Regressão

A **regressão**, por sua vez, também é um processo de aprendizado supervisionado, mas, ao invés de prever rótulos categóricos, o objetivo é prever valores contínuos. Se estivermos tentando prever uma quantidade numérica — como preço, temperatura, ou pontuação — a regressão é a abordagem adequada.



(a) Aprendizado não supervisionado.

(b) Aprendizado supervisionado.

Figura 1: Exemplos visuais de tipos de aprendizado.

0.13 Métricas de Avaliação em Algoritmos de Classificação

0.14 Matriz de Confusão

Definição: Uma tabela que descreve o desempenho de um modelo de classificação, mostrando a distribuição de previsões corretas e incorretas para cada classe.

Componentes:

- **TP (True Positives)**: Número de previsões positivas corretas
- **TN (True Negatives)**: Número de previsões negativas corretas
- **FP (False Positives)**: Número de previsões positivas incorretas (erro Tipo I)
- **FN (False Negatives)**: Número de previsões negativas incorretas (erro Tipo II)

Utilidade: Fornece uma visão detalhada do desempenho do modelo, permitindo identificar em quais tipos de erros o modelo está mais propenso.

	Predito Positivo	**Predito Negativo**
Real Positivo	TP	FN
Real Negativo	FP	TN

0.15 Acurácia

Definição: Proporção de previsões corretas em relação ao total de previsões.

Fórmula:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}$$

Interpretação: Varia de 0 a 1, onde 1 indica 100% de previsões corretas.

Limitações: Pode ser enganosa em conjuntos de dados desbalanceados, onde uma classe é muito mais frequente que a outra.

0.16 Recall (Sensibilidade)

Definição: Proporção de casos positivos reais que foram corretamente identificados pelo modelo.

Fórmula:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Interpretação: Mede a capacidade do modelo de encontrar todos os casos positivos.

Quando usar: Importante quando o custo de falsos negativos é alto (ex: diagnóstico médico).

0.17 Precisão

Definição: Proporção de previsões positivas que estavam realmente corretas.

Fórmula:

$$\text{Precisão} = \frac{TP}{TP + FP}$$

Interpretação: Mede a exatidão das previsões positivas.

Quando usar: Importante quando o custo de falsos positivos é alto (ex: filtros de spam).

0.18 F1-Score

Definição: Média harmônica entre precisão e recall.

Fórmula:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}$$

Interpretação: Varia de 0 a 1. Útil quando precisamos de um equilíbrio entre precisão e recall.

Quando usar: Quando tanto falsos positivos quanto falsos negativos têm custos significativos.

0.19 Relatório de Classificação

Definição: Um resumo das principais métricas (precisão, recall, F1-score) para cada classe.

Componentes: Inclui precisão, recall, F1-score e suporte (número de ocorrências de cada classe).

Visualização: No Yellowbrick, células mais vermelhas (valores próximos de 1) indicam melhor desempenho.

Utilidade: Fornece uma visão mais completa do desempenho do modelo em todas as classes.

0.20 Curva ROC (Receiver Operating Characteristic)

Definição: Gráfico que mostra o desempenho de um modelo de classificação em diferentes limiares de decisão.

Eixos:

- Eixo X: Taxa de Falso Positivo (1 - Especificidade)
- Eixo Y: Taxa de Verdadeiro Positivo (Sensibilidade/Recall)

Interpretação:

- AUC = 0,5 indica modelo aleatório
- AUC próximo de 1 indica excelente desempenho

Limitação: Pode ser otimista em conjuntos de dados desbalanceados.

0.21 Curva de Precisão-Recall

Definição: Alternativa à curva ROC, especialmente útil para conjuntos de dados desbalanceados.

Eixos:

- Eixo X: Recall
- Eixo Y: Precisão

Interpretação: A área sob esta curva (AUC-PR) indica o quão bem o modelo equilibra precisão e recall.

Quando usar: Preferível à curva ROC quando as classes são muito desbalanceadas.

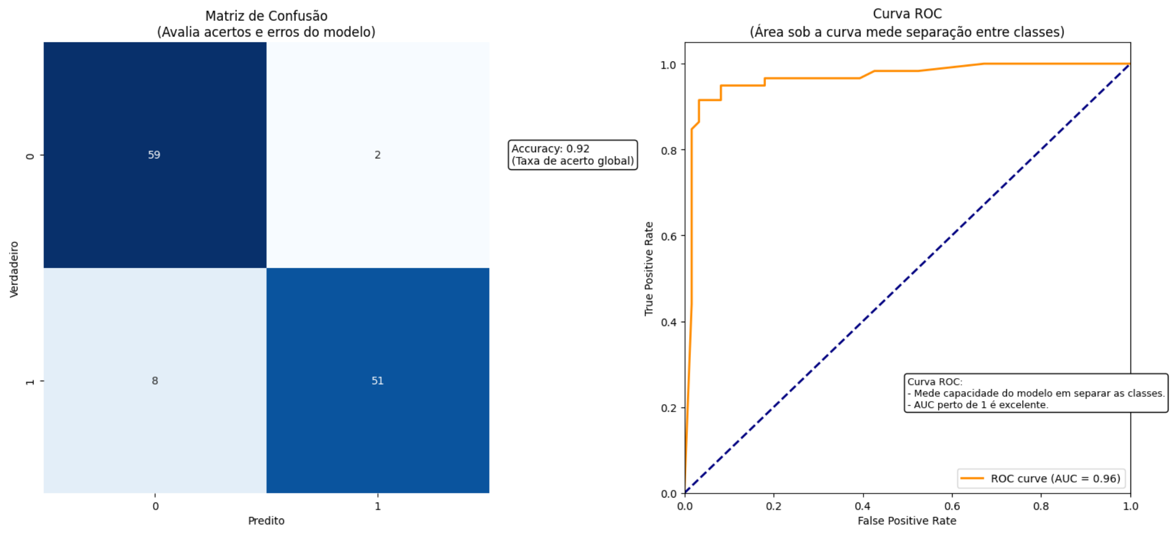


Figura 2: Matriz de confusao e Curva ROc .

0.22 Gráfico de Ganhos Cumulativos

Definição: Mostra a eficácia do modelo em identificar casos positivos quando os dados são ordenados por probabilidade predita.

Eixos:

- Eixo X: Porcentagem de amostras avaliadas
- Eixo Y: Porcentagem de casos positivos encontrados

Interpretação: Um modelo perfeito encontraria 100% dos casos positivos analisando apenas os primeiros $X\%$ dos dados.

Exemplo prático: Se os primeiros 10% das previsões contiverem 30% dos casos positivos reais, isto indica boa capacidade de priorização pelo modelo.

0.23 Gráfico de Elevação (Lift)

Definição: Mostra quanto melhor o modelo é em relação a uma seleção aleatória.

Cálculo: Dividindo a taxa de identificação de positivos pelo modelo pela taxa esperada em uma seleção aleatória.

Interpretação: Um lift de 3 nos primeiros 10% significa que o modelo é 3 vezes melhor que uma seleção aleatória neste segmento.

Utilidade: Útil em aplicações de marketing e vendas para priorizar clientes com maior probabilidade de conversão.

Nota Final: Estas métricas são complementares e a escolha de quais utilizar depende do contexto específico do problema, do balanceamento de classes e das consequências relativas de diferentes tipos de erros.

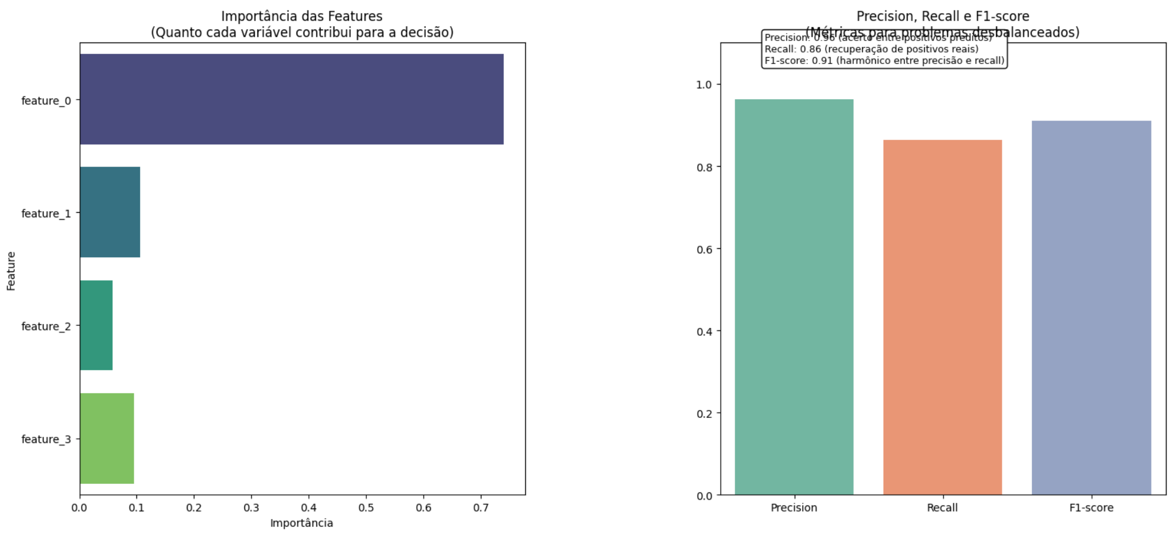


Figura 3: Matriz de confusao e Curva ROc .

Métricas de Avaliação em Algoritmos de Regressão

Nesta seção, são apresentadas as principais métricas utilizadas para avaliar algoritmos de regressão.

R^2 (Coeficiente de Determinação)

O R^2 mede a proporção da variância na variável dependente que é explicada pelas variáveis independentes. Varia de 0 a 1, onde:

- $R^2 = 1$: o modelo explica 100% da variabilidade dos dados;
- $R^2 = 0$: o modelo não explica nada da variabilidade;
- Valores negativos podem ocorrer quando o modelo se ajusta pior que uma linha horizontal.

O R^2 ajuda a entender o quanto o modelo consegue explicar o comportamento dos dados observados.

RMSE (Root Mean Square Error)

O RMSE é a raiz quadrada da média dos erros ao quadrado entre os valores previstos e observados. É uma medida de acurácia com as seguintes características:

- Está na mesma unidade da variável prevista;
- Penaliza erros maiores de forma mais severa que erros menores;
- Quanto menor o RMSE, melhor o desempenho do modelo.

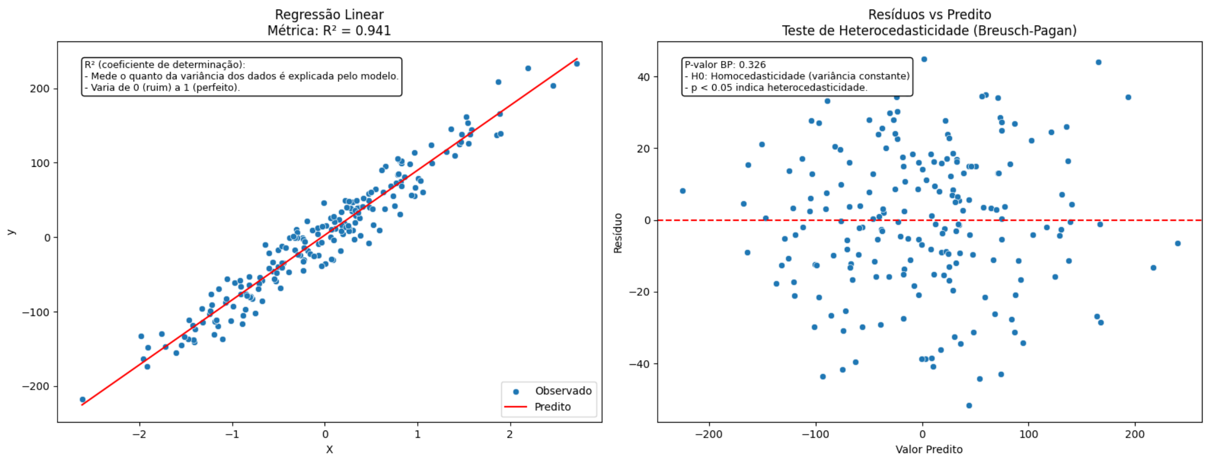


Figura 4: Regressao linear e Heterocidade .

MAE (Mean Absolute Error)

O MAE é a média dos valores absolutos dos erros entre previsões e valores reais. Possui as seguintes propriedades:

- Está na mesma unidade da variável prevista;
- Trata todos os erros com o mesmo peso, independentemente da magnitude;
- É menos sensível a outliers do que o RMSE;
- Quanto menor o MAE, melhor o modelo.

Teste de Kolmogorov-Smirnov

O teste de Kolmogorov-Smirnov verifica se os resíduos do modelo seguem uma distribuição normal — premissa importante para muitos modelos de regressão. Este teste:

- Compara a distribuição empírica dos resíduos com uma distribuição normal teórica;
- Valores de p baixos indicam que os resíduos não seguem uma distribuição normal;
- É útil para validar o pressuposto de normalidade dos erros.

Heterocedasticidade

Heterocedasticidade ocorre quando a variância dos erros não é constante entre as observações. Para modelos de regressão linear, é desejável a homocedasticidade (isto é, variância constante dos erros), pois:

- A heterocedasticidade pode violar pressupostos do modelo;
- Pode afetar a validade das inferências estatísticas;
- Testes específicos (como o teste de Breusch-Pagan) podem ser utilizados para detectar esse problema.

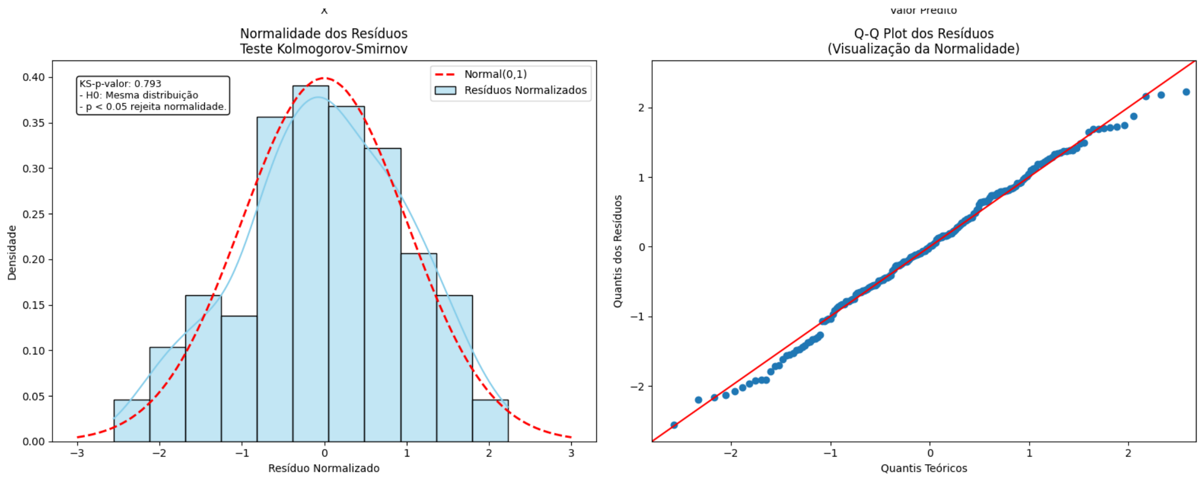


Figura 5: Kologronov srinov e normalidade .

Random Forest Regressor

O **Random Forest Regressor** funciona de maneira semelhante ao classificador, mas é utilizado para problemas de regressão. Em vez de votar na classe mais comum, ele calcula a *média* das previsões das árvores individuais.

- Treinamento:
$$O(n_{\text{estimadores}} \cdot n \cdot \log n \cdot d)$$
- Predição:
$$O(n_{\text{estimadores}} \cdot h)$$
onde h é a altura média das árvores (geralmente $\log n$ em árvores balanceadas).

Gradient Boosting Classifier

O **Gradient Boosting** constrói modelos de forma sequencial, corrigindo os erros do modelo anterior.

- Treinamento:
$$O(n_{\text{estimadores}} \cdot n \cdot \log n \cdot d)$$
- Predição:
$$O(n_{\text{estimadores}} \cdot \log n)$$

Logistic Regression

A **Regressão Logística** é um algoritmo simples e interpretável, ideal para classificação binária.

- Treinamento:
$$O(n \cdot d \cdot i)$$
onde i é o número de iterações até a convergência.
- Predição:
  - Amostra única: $O(d)$
  - Batch: $O(n \cdot d)$

Decision Tree Classifier

O **Classificador de Árvore de Decisão** atua como modelo único. Foi substituído pelo Random Forest no projeto.

- Treinamento:
$$O(n \cdot d \cdot \log n)$$
- Predição:
$$O(\text{depth}) \quad (\text{geralmente } O(\log n), \text{ mas pode variar com } \texttt{max\_depth})$$

Support Vector Machine (SVC)

O **SVM** busca o hiperplano ótimo para separação das classes.

- Treinamento:
$$O(n^2 \cdot d) \text{ a } O(n^3 \cdot d)$$
- Predição:
$$O(n_{\text{support vectors}} \cdot d)$$

K-Nearest Neighbors (KNN)

O **KNN** classifica com base na proximidade no espaço de características.

- Treinamento:
$$O(1)$$
- Predição:
$$O(n \cdot d)$$
Pode ser reduzido com k-d trees, mas perde eficiência em alta dimensionalidade.

Gaussian Naive Bayes

Algoritmo probabilístico baseado no teorema de Bayes.

- Treinamento:
$$O(n \cdot d)$$
- Predição:
$$O(d) \quad \text{ou } O(c \cdot d) \text{ para } c \text{ classes}$$

Multi-layer Perceptron (Neural Network)

Modelo de rede neural artificial capaz de capturar relações não-lineares complexas.

- Treinamento (forma simplificada):
$$O(n \cdot i \cdot h \cdot d)$$
- Treinamento (forma geral para múltiplas camadas):
$$O\left(n \cdot i \cdot \sum_{l=1}^{L} h_l \cdot h_{l-1}\right)$$
- Predição:
$$O(h \cdot d)$$

Gaussian Process Classification (GPC)

O **Gaussian Process Classification** é um método bayesiano não-paramétrico que usa processos gaussianos para inferir uma distribuição sobre funções. É bastante poderoso em datasets pequenos, especialmente quando se busca representar a incerteza na predição.

- Treinamento:
$$O(n^3)$$
Isso se deve à inversão da matriz de covariância $n \times n$, que é computacionalmente custosa.

- Predição:

$$O(n^2)$$

pois envolve operações com a matriz de covariância entre os pontos de treino e o novo ponto.

Observação: Essa abordagem não escala bem para grandes conjuntos de dados. Há variantes como Sparse GPs e aproximadores que reduzem o custo para $O(nm^2)$, onde $m \ll n$ é o número de pontos indutores.

Gaussian Process Regression (GPR)

O **Gaussian Process Regression** segue a mesma lógica do classificador, mas para variáveis contínuas. Ele prevê a distribuição de probabilidade dos valores de saída com base na entrada, retornando uma média e variância associada à predição.

- Treinamento:

$$O(n^3)$$

devido à inversão da matriz de covariância dos dados de entrada.

- Predição:

$$O(n^2)$$

que vem da necessidade de calcular a covariância entre todos os dados de treino e o novo ponto.

Observação: Assim como no GPC, há métodos aproximados que permitem reduzir o custo computacional e escalar para datasets maiores.

0.24 Análise Técnica: Modelo de Previsão de Diferença de Gols em Partidas de Futebol

Sumário

0.25 Visão Geral do Algoritmo

Este documento apresenta uma análise técnica detalhada de um algoritmo implementado para previsão da diferença de gols entre dois times em partidas de futebol. O objetivo do modelo é prever uma variável contínua Diferenca_Gols = gols_time1 − gols_time2 utilizando estatísticas de jogo como preditores.

A abordagem metodológica adota uma comparação sistemática entre múltiplos modelos de regressão, com implementação completa de:

- Pipeline de pré-processamento de dados
- Engenharia e seleção de features
- Treinamento e otimização de múltiplos algoritmos
- Avaliação comparativa através de métricas padronizadas

0.26 Conjunto de Dados

O dataset analisado contém estatísticas detalhadas de partidas de futebol, com 4.161 observações. As características incluem:

- **Estatísticas ofensivas:** chutes a gol, chutes fora
- **Estatísticas de posse de bola:** percentual de posse por equipe
- **Estatísticas defensivas:** faltas cometidas e cartões
- **Lances de bola parada:** escanteios e impedimentos
- **Aspectos táticos:** formações táticas dos times (variáveis categóricas)

A variável alvo Diferenca_Gols apresenta as seguintes características estatísticas:

Estatística	**Valor**
Contagem	4.161
Média	0,34
Desvio padrão	1,67
Mínimo	-4,00
Q1 (25%)	-1,00
Mediana	0,00
Q3 (75%)	1,00
Máximo	5,00

Tabela 1: Estatísticas descritivas da variável alvo (diferença de gols)

A distribuição de resultados categorizados mostra:

- Vitórias do time 1: 1.886 observações (45,3%)
- Empates: 994 observações (23,9%)
- Vitórias do time 2: 1.281 observações (30,8%)

Esta distribuição indica uma leve vantagem para o time 1 (possivelmente o mandante), o que é consistente com o valor médio positivo (0,34) da diferença de gols.

0.27 Engenharia de Features

0.27.1 Features Derivadas

O algoritmo implementa sofisticada engenharia de features para extrair preditores potencialmente mais informativos:

- **Diferenças entre times:**
  - Diff_Chutes_Gol = Chutes a gol 1 − Chutes a gol 2
  - Diff_Posse = Posse 1(%) − Posse 2(%)
  - Diff_Escanteios = Escanteios 1 − Escanteios 2
  - Diff_Faltas = Faltas 1 − Faltas 2
- **Métricas de eficiência:**
  - Eficiencia_Ofensiva_1 = $\frac{\text{Chutes a gol 1}}{\text{Chutes a gol 1+Chutes fora 1+}\epsilon}$
  - Eficiencia_Ofensiva_2 = $\frac{\text{Chutes a gol 2}}{\text{Chutes a gol 2+Chutes fora 2+}\epsilon}$
  - Diff_Eficiencia_Ofensiva = Eficiencia_Ofensiva_1 − Eficiencia_Ofensiva_2
- **Codificação de formações táticas:**
  - Transformação one-hot das variáveis categóricas Position 1 e Position 2
  - Criação de variáveis dummy para cada formação (e.g., Time1_Form_4-3-3, Time2_Form_4-2-3-1)

0.27.2 Pré-processamento

O pipeline de pré-processamento inclui:

- **Tratamento de valores ausentes:** Imputação pela mediana para cada feature numérica
- **Normalização:** Aplicação de StandardScaler para padronizar as variáveis numéricas segundo a fórmula:
$$x_{norm} = \frac{x - \mu}{\sigma} \tag{16}$$
onde μ é a média e σ o desvio padrão da feature
- **Seleção de features:** Utilização de SelectKBest com f_regression para determinar as 20 features mais relevantes, baseado nas estatísticas F e valores-p associados

0.28 Modelagem

0.28.1 Modelos Implementados

Foram implementados e avaliados sete algoritmos de regressão:

1. **Regressão Linear:** Implementação base sem regularização
$$\hat{y} = X\beta \tag{17}$$
2. **Ridge:** Regressão com regularização L2 (parâmetro $\alpha = 1.0$)
$$\min_{\beta} \|y - X\beta\|_2^2 + \alpha \|\beta\|_2^2 \tag{18}$$
3. **Lasso:** Regressão com regularização L1 (parâmetro $\alpha = 0.1$)
$$\min_{\beta} \|y - X\beta\|_2^2 + \alpha \|\beta\|_1 \tag{19}$$
4. **ElasticNet:** Combinação de regularização L1 e L2 (parâmetros $\alpha = 0.1$, $l1\_ratio = 0.5$)
$$\min_{\beta} \|y - X\beta\|_2^2 + \alpha \cdot l1\_ratio \|\beta\|_1 + \alpha \cdot (1 - l1\_ratio) \|\beta\|_2^2 \tag{20}$$
5. **SVR:** Support Vector Regression com kernel RBF (parâmetros $C = 10.0$, $\gamma =$ 'scale', $\epsilon = 0.2$)
6. **Random Forest:** Ensemble baseado em árvores de decisão (parâmetros $n\_estimators = 100$, $max\_depth = 20$)
7. **Gradient Boosting:** Ensemble sequencial (parâmetros $n\_estimators = 100$, $learning\_rate = 0.1$, $max\_depth = 5$)

0.28.2 Pipeline de Treinamento

O pipeline completo de treinamento seguiu as seguintes etapas:

- **Divisão de dados:** 80% para treino (3.328 amostras) e 20% para teste (833 amostras), utilizando amostragem aleatória
- **Escalonamento:** Normalização das features aplicada apenas após a divisão treino/teste para evitar data leakage
- **Seleção de features:** As 20 features mais relevantes segundo a estatística F foram selecionadas
- **Treinamento:** Cada modelo foi treinado com os parâmetros especificados
- **Avaliação:** Múltiplas métricas foram calculadas para comparar sistematicamente o desempenho dos modelos

0.29 Análise Comparativa de Desempenho

0.29.1 Métricas de Avaliação

Foram utilizadas as seguintes métricas para avaliar o desempenho dos modelos:

- R^2 (Coeficiente de determinação): proporção da variância explicada pelo modelo
- **RMSE** (Root Mean Squared Error): raiz do erro quadrático médio
- **MAE** (Mean Absolute Error): erro absoluto médio
- **EVS** (Explained Variance Score): variância explicada
- **MAPE** (Mean Absolute Percentage Error): erro percentual absoluto médio
- **SMAPE** (Symmetric Mean Absolute Percentage Error): erro percentual absoluto médio simétrico

A tabela 2 apresenta o desempenho comparativo dos modelos:

Modelo	R^2	**RMSE**	**MAE**	**EVS**	**Tempo (s)**
ElasticNet	0,2376	1,4348	1,1512	0,2376	0,0136
Lasso	0,2354	1,4369	1,1512	0,2355	0,0099
Ridge	0,2326	1,4395	1,1568	0,2326	0,0107
Regressão Linear	0,2326	1,4395	1,1568	0,2326	0,0524
Gradient Boosting	0,1924	1,4768	1,1779	0,1924	1,2010
Random Forest	0,1832	1,4852	1,1877	0,1834	3,2719
SVR	0,0808	1,5755	1,2633	0,0808	1,7168

Tabela 2: Comparação de desempenho dos modelos de regressão

0.29.2 Interpretação dos Resultados

A análise dos resultados permite extrair as seguintes conclusões:

- O modelo **ElasticNet** apresentou o melhor desempenho geral, com R^2 de 0,238 e RMSE de 1,43.
- Os modelos com regularização (Lasso, ElasticNet) superaram ligeiramente a regressão linear simples, sugerindo benefícios na redução do overfitting através da penalização dos coeficientes.
- Modelos baseados em árvores (Random Forest, Gradient Boosting) tiveram desempenho inferior aos modelos lineares, sugerindo que a relação entre as features e a variável alvo é predominantemente linear ou que os hiperparâmetros escolhidos não foram ótimos.
- SVR apresentou o pior desempenho entre todos os modelos testados, com R^2 de apenas 0,081, indicando que a abordagem baseada em kernel não conseguiu capturar adequadamente o padrão dos dados.
- O baixo R^2 geral (máximo de 0,238) indica uma capacidade preditiva limitada - o modelo consegue explicar apenas aproximadamente 24% da variância na diferença de gols.

0.29.3 Features Mais Relevantes

As features mais importantes identificadas pelo método SelectKBest foram:

Feature	**Score F**
Diff_Chutes_Gol	1081,36
Chutes a gol 1	557,45
Diff_Eficiencia_Ofensiva	444,73
Chutes a gol 2	441,49
Eficiencia_Ofensiva_2	200,46
Eficiencia_Ofensiva_1	166,36
Cartões amarelos 1	73,74
Diff_Posse	40,02
Posse 1(%)	40,02
Posse 2(%)	40,02

Tabela 3: Top 10 features mais relevantes segundo estatística F

Esta análise sugere que métricas relacionadas a chutes a gol e eficiência ofensiva são os melhores preditores para a diferença de gols, o que é consistente com a intuição sobre o esporte.

0.30 Análise de Hiperparâmetros

O notebook utiliza hiperparâmetros relativamente conservadores, sem implementação de uma busca extensiva:

- **Ridge:** $\alpha = 1.0$ (padrão) - penalização moderada
- **Lasso:** $\alpha = 0.1$ - penalização reduzida para permitir mais features não-zero
- **ElasticNet:** $\alpha = 0.1$, $l1\_ratio = 0.5$ - equilíbrio entre penalizações L1 e L2
- **SVR:** $kernel =' rbf'$, $C = 10.0$, $\gamma =' scale'$, $\epsilon = 0.2$ - C alto indica menor regularização
- **Random Forest:** $n\_estimators = 100$, $max\_depth = 20$ - árvores relativamente profundas
- **Gradient Boosting:** $n\_estimators = 100$, $learning\_rate = 0.1$, $max\_depth = 5$ - configuração padrão

A ausência de otimização sistemática de hiperparâmetros (como GridSearchCV ou RandomizedSearchCV) representa uma limitação do modelo atual. O título do notebook "low hyperparametro"sugere deliberadamente uma abordagem simplificada à otimização.

0.31 Complexidade Computacional e Desempenho

0.31.1 Complexidade Temporal

A análise de complexidade temporal dos algoritmos implementados revela:

- **Modelos lineares** (Linear, Ridge, Lasso, ElasticNet): $O(nd^2)$ onde n é o número de amostras e d o número de features. Extremamente eficientes com tempos de execução $< 0.05s$.
- **SVR**: $O(n^2d)$ a $O(n^3d)$ no pior caso, resultando em tempo de execução de $1.72s$.
- **Random Forest**: $O(ntd\sqrt{d})$ onde t é o número de árvores, resultando em tempo de execução de $3.27s$.
- **Gradient Boosting**: $O(ntd)$ onde t é o número de árvores, com tempo de execução de $1.20s$.

0.31.2 Trade-off Desempenho-Complexidade

Um resultado interessante é que os modelos lineares, além de serem computacionalmente mais eficientes, também apresentaram melhor desempenho preditivo neste conjunto de dados. Isso sugere que a relação entre as estatísticas de jogo e a diferença de gols pode ser razoavelmente aproximada por um modelo linear, e a complexidade adicional dos métodos baseados em árvores ou kernels não trouxe benefícios significativos.

0.31.3 Escalabilidade

Para este dataset (aproximadamente 4.161 amostras com 74 features potenciais), todos os modelos apresentaram desempenho aceitável em termos de tempo de execução. Para conjuntos de dados significativamente maiores, seria recomendável:

- Manter o foco em modelos lineares regularizados (ElasticNet, Lasso)
- Implementar processamento paralelo para Random Forest
- Considerar SGD (Stochastic Gradient Descent) para versões online dos algoritmos lineares

0.32 Conclusões e Limitações Técnicas

0.32.1 Limitações do Modelo

- O **baixo** R^2 (máximo de 0,238) indica uma capacidade preditiva limitada, sugerindo que fatores importantes para prever o resultado das partidas não estão sendo capturados pelas features disponíveis.
- O **RMSE de aproximadamente 1,43** significa um erro típico de quase um gol e meio na previsão da diferença, um valor considerável no contexto do futebol onde muitos jogos terminam com diferenças pequenas.
- Os valores de **MAPE** muito altos (acima de 80%) confirmam a dificuldade do modelo em fazer previsões acuradas, especialmente para valores próximos de zero.

0.32.2 Possíveis Aprimoramentos

1. **Otimização sistemática de hiperparâmetros:**
   - Implementar GridSearchCV ou RandomizedSearchCV para busca mais ampla
   - Considerar validação cruzada para estimativas mais robustas de desempenho
2. **Engenharia de features avançada:**
   - Incorporar informações temporais (sequência de jogos, forma recente)
   - Adicionar métricas de qualidade dos times (rankings, orçamento)
   - Explorar interações entre features existentes
3. **Modelagem alternativa:**
   - Abordagem probabilística (distribuições de Poisson para gols)
   - Modelos hierárquicos considerando a qualidade dos times
   - Redes neurais para capturar relações não-lineares complexas
4. **Avaliação mais robusta:**
   - Implementar validação cruzada temporal para dados de séries temporais
   - Avaliar a calibração das previsões probabilísticas
   - Comparar com previsões de mercados de apostas como baseline

Este modelo representa um ponto de partida válido para previsão de resultados de futebol, mas os resultados sugerem que prever diferenças de gols com precisão é um problema complexo que pode requerer abordagens mais sofisticadas e dados adicionais.