Análise Exploratória de Dados para Remuneração e Orçamento Estratégico

# Introdução

Este estudo aprofundado tem como objetivo realizar uma análise exploratória completa dos conjuntos de dados "df\_salary" e "df\_mart", utilizando técnicas avançadas e de fácil interpretação em Python e suas bibliotecas. Através da análise, serão identificados insights valiosos para auxiliar na tomada de decisões estratégicas de remuneração e orçamento.

## Conjuntos de Dados

### 1. df\_salary:

* Funcionários: 500 registros únicos (identificadores de funcionários)
* Departamento: Categorias discretas: RH, TI, Vendas, Marketing, Financeiro
* Nível do Cargo: Níveis hierárquicos: Sênior, Diretor, Pleno, Gerente, Junior
* Experiência (anos): Valores inteiros entre 1 e 30
* Salário Atual (R$): Valores entre 3.062,00 e 19.907,00
* Aumento Salarial (%): Valores percentuais entre 0,0% e 15,0%

### 2. df\_mart:

* Departamento: Categorias discretas: RH, TI, Vendas, Marketing, Financeiro
* Nível do Cargo: Níveis hierárquicos: Sênior, Diretor, Pleno, Gerente, Junior
* Salário de Mercado (R$): Valores entre 4.656,00 e 23.258,00

## Metodologia

* Importação de bibliotecas Python: numpy, matplotlib.pyplot, pandas, seaborn, statsmodels, plotly.express
* Leitura e pré-processamento dos dataframes "df\_salary" e "df\_mart"
* Análise descritiva abrangente de cada variável:
  + Distribuições de frequência
  + Medidas estatísticas (média, mediana, desvio padrão, quartis, etc.)
  + Visualizações avançadas com gráficos e heatmaps
* Análise comparativa entre diferentes departamentos e níveis de cargos:
  + Testes estatísticos (ANOVA, Kruskal-Wallis)
  + Gráficos de comparação de médias e distribuições
* Modelagem de regressão linear para identificar fatores que influenciam o salário:
  + Seleção de variáveis relevantes
  + Estimação de coeficientes e intercepto
  + Avaliação da qualidade do modelo (R², AIC, BIC)
* Visualização interativa dos resultados com dashboards em Plotly Express

## Resultados Esperados

* Insights valiosos sobre a estrutura salarial da empresa:
  + Diferenças salariais entre departamentos e níveis de cargos
  + Relação entre experiência e salário
  + Fatores que impactam o aumento salarial
* Recomendações estratégicas para otimização da remuneração:
  + Ajustes salariais para alcançar equidade interna
  + Estratégias de retenção de talentos
  + Planejamento orçamentário preciso para custos com pessoal

## Conclusão

A análise exploratória avançada dos dados de "df\_salary" e "df\_mart" fornecerá insights acionáveis para auxiliar na tomada de decisões estratégicas de remuneração e orçamento. A combinação de técnicas estatísticas robustas, visualizações interativas e modelagem de regressão permitirá uma compreensão profunda da estrutura salarial da empresa e a identificação de oportunidades para otimização.

Lógica de Programação Detalhada para Análise de Dados

# Importaçã o e Pré-processamento dos Dados:

## Carregamento e Inspeção Inicial:

* O conjunto de dados é importado para o Google Colab utilizando a biblioteca Pandas, atribuindo-se a ele um nome descritivo como df\_original.
* As primeiras linhas do dataframe são exibidas para verificar sua estrutura e o conteúdo das variáveis, permitindo uma compreensão inicial dos dados.

## Identificação e Correção de Inconsistências:

* A presença de valores nulos (NaN) e inconsistentes em cada coluna é verificada com as funções df.isnull().sum() e df.describe(), fornecendo um diagnóstico preciso da qualidade dos dados.
* Os valores nulos são tratados de acordo com o contexto:
* Imputação com a média, mediana ou outro valor estratégico, como a moda para variáveis categóricas, garantindo a preservação da integridade dos dados.
* Exclusão de linhas com muitos valores nulos se a quantidade for significativa, minimizando o impacto de informações incompletas na análise.
* Valores inconsistentes são identificados e corrigidos manualmente ou através de filtros e algoritmos específicos, assegurando a confiabilidade dos dados.

## Organização e Limpeza Detalhada:

* As colunas são renomeadas para nomes mais descritivos e consistentes, utilizando letras minúsculas e "\_" como separador, facilitando a compreensão e manipulação dos dados.
* Datas são convertidas para o formato datetime se necessário, garantindo a padronização e coerência temporal.
* A formatação de strings e valores é padronizada em todas as colunas, assegurando uniformidade e legibilidade.
* Novas colunas podem ser criadas a partir de cálculos ou combinações de outras colunas, expandindo a riqueza da informação disponível para análise.
* O dataframe limpo e organizado é salvo com um novo nome como df\_tratado, preservando o estado original dos dados e facilitando o acesso à versão processada.

# análise exploratória detalhada dos dados: insights abrangentes e aprofundados

## Métricas Salariais por Departamento: Desvendando Disparidades e Tendências

* Para cada departamento, o salário médio é calculado com a função df.groupby("Departamento")["Salario\_Atual"].mean(), revelando a remuneração média por área.
* O salário mediano é determinado com a função df.groupby("Departamento")["Salario\_Atual"].median(), oferecendo uma visão complementar da distribuição salarial dentro de cada departamento.
* A amplitude salarial, calculada como a diferença entre o maior e o menor salário em cada departamento com a função df.groupby("Departamento")["Salario\_Atual"].max() - df.groupby("Departamento")["Salario\_Atual"].min(), fornece insights sobre a desigualdade salarial entre os funcionários.
* Os resultados são armazenados em um dataframe ou dicionário para facilitar a análise e visualização, permitindo uma exploração mais profunda dos dados.

## Desvendando os Maiores Aumentos Salariais: Reconhecimento e Retenção de Talentos

* A variação percentual do salário de cada funcionário em relação ao ano anterior é calculada com a fórmula ((Salario\_Atual - Salario\_Anterior) / Salario\_Anterior) \* 100, identificando os indivíduos com maior crescimento salarial.
* Os 5 funcionários com maior aumento percentual são identificados e ordenados por essa métrica, destacando os casos de maior reconhecimento e potencial de retenção.
* Informações adicionais sobre esses funcionários, como cargo, departamento, tempo de empresa etc., são extraídas, fornecendo um contexto mais rico para a análise.
* Um dataframe ou tabela é criado para resumir os perfis dos 5 funcionários com maior aumento salarial, facilitando a comparação e a tomada de decisões estratégicas.

# Integração dos Dataframes:

Combinação de Dataframes:

Utilizar a função pd.merge() para combinar os dataframes df\_tratado e df\_salarios\_mercado na coluna Nivel\_Cargo, garantindo a correspondência entre os registros.

Atribuir um nome descritivo ao dataframe resultante, como df\_completo.

Verificar se a integração foi realizada com sucesso, verificando a quantidade de linhas e colunas no dataframe final.

4. Categorização dos Funcionários:

Criação da Coluna "Categoria":

Utilizar a função np.where() para criar uma nova coluna Categoria no dataframe df\_completo.

Definir as categorias como:

"Acima do mercado": se Salario\_Atual > Salario\_Mercado.

"No mercado": se Salario\_Atual ≈ Salario\_Mercado.

"Abaixo do mercado": se Salario\_Atual < Salario\_Mercado.

Definir a tolerância para a comparação de "No mercado" de acordo com o contexto (ex: 5%).

5. Visualização e Validação dos Resultados:

Exibição das Primeiras Linhas do Dataframe:

Exibir as primeiras linhas do dataframe df\_completo para verificar a integração dos dados e a categorização dos funcionários.

Confirmar se a categorização está de acordo com as definições e expectativas.

6. Análise Detalhada por Categoria:

Filtragem por Categoria:

Filtrar o dataframe df\_completo por cada categoria

7. Criação de Gráficos Informativos:

7.1 Distribuição dos Salários por Departamento:

Histograma por Departamento:

Criar um histograma para cada departamento, utilizando a função df.groupby("Departamento")["Salario\_Atual"].hist().

Ajustar os parâmetros do histograma, como número de bins e cor, para melhor visualização.

Adicionar títulos, legendas e eixos rotulados de forma clara e informativa.

Comparar visualmente a distribuição dos salários entre os departamentos.

Boxplot por Departamento:

Criar um boxplot para cada departamento, utilizando a função df.groupby("Departamento")["Salario\_Atual"].boxplot().

Identificar outliers e analisar a dispersão dos salários em cada departamento.

Comparar a mediana, quartis e valores extremos entre os departamentos.

7.2 Evolução do Salário Médio:

Gráfico de Linha do Salário Médio:

Se dados históricos estiverem disponíveis:

Calcular o salário médio por período (mês, trimestre, ano) utilizando a função df.groupby("Data")["Salario\_Atual"].mean().

Criar um gráfico de linha com a evolução do salário médio ao longo do tempo.

Incluir títulos, legendas e eixos rotulados de forma clara e informativa.

Identificar tendências e mudanças no salário médio ao longo do tempo.

8. Modelagem Financeira Preditiva:

8.1 Projeção do Orçamento Salarial:

Simulação de Cenários de Aumento:

Para cada cenário de aumento salarial (3%, 5%, 7%):

Multiplicar o salário atual de cada funcionário pelo fator de aumento (1 + taxa de aumento).

Calcular o orçamento salarial total para o próximo ano somando os salários projetados de todos os funcionários.

Criar uma tabela que resuma os resultados da simulação para cada cenário.

8.2 Considerações Adicionais:

Fatores a serem considerados:

Incluir outros fatores na modelagem, como inflação, crescimento da empresa e desempenho individual dos funcionários.

Ajustar o modelo para diferentes cenários e realizar análises de sensibilidade.