**MACHINE LEARNING PARA DETECÇÃO DE COLABORADORES COM SALÁRIO ACIMA DE 50 MIL/ANO – SUPORTE RH**

[https://lh3.googleusercontent.com/ogw/AF2bZyjpYUgveDvSLU28KCV6kSkcbM8PpcsoAmKVB9WpKw=s32-c-mo](https://accounts.google.com/SignOutOptions?hl=pt-BR&continue=https://colab.research.google.com/drive/1IRaaKpg4tXF6GXyyaamvE8mJgM7M4uyn?hl=pt-BR&ec=GBRAqQM) Project by Vladimir Gama

**Contexto**

Toda empresa precisa provisionar e ter um conhecimento claro sobre os gastos com os salários dos colaboradores. E a seguir seguem alguns motivos:

1. **Planejamento Financeiro:** Saber quanto será gasto com os salários dos colaboradores é essencial para o planejamento financeiro da empresa. Isso permite que a organização preveja com precisão seus custos operacionais e desenvolva estratégias para alocar recursos de forma eficiente.
2. **Controle Orçamentário:** O provisionamento adequado dos salários ajuda na manutenção do controle orçamentário. Ao conhecer antecipadamente os valores a serem desembolsados com os salários, a empresa pode comparar esses números com seu orçamento previsto, identificar discrepâncias e tomar medidas corretivas, se necessário.
3. **Evitar Surpresas Financeiras:** Ter uma previsão precisa dos gastos com salários ajuda a evitar surpresas financeiras desagradáveis. Isso inclui a possibilidade de não ter fundos suficientes para pagar os salários dos colaboradores, o que pode causar problemas de fluxo de caixa e até mesmo afetar a reputação da empresa.
4. **Garantir a Sustentabilidade Financeira:** Ao provisionar adequadamente os salários, a empresa está se preparando para garantir sua sustentabilidade financeira a longo prazo. Isso é especialmente importante em momentos de incerteza econômica ou quando ocorrem flutuações nos negócios.
5. **Transparência e Credibilidade:** Ter uma compreensão clara dos gastos com salários demonstra transparência e credibilidade para os stakeholders da empresa, incluindo funcionários, investidores, credores e órgãos reguladores.

Em resumo, provisionar e saber quanto será gasto com os salários dos colaboradores é fundamental para uma gestão financeira eficaz, garantindo o equilíbrio entre receitas e despesas e contribuindo para a sustentabilidade e o sucesso financeiro da empresa.

**1 PROBLEMA DE NEGÓCIO**

Esse script demonstra a criação de uma Máquina Preditiva que, a partir de dados históricos dos colaboradores, determine quais deles vão receber o valor acima de R$ 50.000,00 ao ano.

**2 ANÁLISE EXPLORATÓRIO DOS DADOS**

**2.1 IMPORTAÇÃO DAS BIBLIOTECAS DE ANÁLISE DE DADOS E VISUALIZAÇÃO**

Com essas bibliotecas e funções, nossa ML está pronta para realizar análise exploratória de dados, preparar seus dados para modelagem, criar e treinar modelos de machine learning, e avaliar o desempenho desses modelos. Essas são etapas comuns de análise de dados e modelagem de machine learning. Vou explicar a utilidade de cada uma delas:

**pandas (import pandas as pd):** Essa biblioteca é amplamente utilizada para manipulação e análise de dados. Você pode carregar conjuntos de dados, fazer limpeza e preparação de dados, realizar operações de filtragem e agregação, entre outras.

**ydata\_profiling (from ydata\_profiling import ProfileReport):** O ydata\_profiling é uma ferramenta que gera relatórios detalhados de perfil de dados. Isso inclui estatísticas descritivas, distribuições, correlações, e muito mais. É útil para explorar e entender rapidamente a estrutura e a qualidade dos seus dados.

**RandomForestClassifier (from sklearn.ensemble import RandomForestClassifier):** RandomForestClassifier é um algoritmo de aprendizado de máquina baseado em árvores de decisão. Ele é usado para problemas de classificação e é conhecido por sua robustez e capacidade de lidar bem com conjuntos de dados complexos.

**LabelEncoder (from sklearn.preprocessing import LabelEncoder):** LabelEncoder é usado para codificar variáveis categóricas em números inteiros. Isso é necessário para que os algoritmos de aprendizado de máquina possam processar essas variáveis.

**train\_test\_split (from sklearn.model\_selection import train\_test\_split):** Esta função é usada para dividir o conjunto de dados em conjuntos de treinamento e teste. Isso é essencial para avaliar a capacidade de generalização do modelo.

**confusion\_matrix (from sklearn.metrics import confusion\_matrix):** A matriz de confusão é uma ferramenta que permite a visualização do desempenho de um algoritmo de classificação. Ela mostra o número de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos.

**2.2 IMPORTAÇÃO DOS DADOS PARA ANÁLISE EXPLORATÓRIA**

A seguir vou explicar a função de cada elemento do código:

**pandas (import pandas as pd):** Importa a biblioteca pandas e a renomeia como **pd**, permitindo acessar suas funções usando **pd.nome\_da\_função**.

**read\_csv():** Uma função do pandas que lê dados de um arquivo CSV e os carrega em um DataFrame. Neste caso, ele está carregando os dados de um arquivo CSV hospedado online.

O URL que aponta para o arquivo CSV que contém os dados que você deseja carregar: **'**<https://raw.githubusercontent.com/llSourcell/Best-Programming-Languages-for-Machine-Learning/master/adults.txt>**'**

**sep=',':** Um parâmetro opcional da função **read\_csv()** que especifica o delimitador usado no arquivo CSV para separar os valores. Neste caso, o delimitador é uma vírgula.

**data:** O nome dado ao DataFrame que armazenará os dados carregados do arquivo CSV. Você pode usar esse nome para acessar e manipular os dados posteriormente.

**2.3 CRIAÇÃO DE UM RELATÓRIO PARA AJUDAR NA ANÁLISE EXPLORATÓRIA**

Esses são os elementos do código e suas respectivas funções que são usados para gerar um relatório de perfil de dados com o ydata\_profiling e exibi-lo tanto no notebook quanto salvá-lo como um arquivo HTML.

**ydata\_profiling (from ydata\_profiling import ProfileReport):** Importa a classe **ProfileReport** do módulo **ydata\_profiling**. Esta classe é responsável por gerar relatórios de perfil de dados.

**ProfileReport(data, title='Relatório Base de Dados', html={'style':{'full\_width':True}}):**

* + **data:** O DataFrame que será analisado para gerar o relatório de perfil.
  + **title='Relatório Base de Dados':** Título do relatório. Neste caso, está definido como 'Relatório Base de Dados'.
  + **html={'style':{'full\_width':True}}:** Este parâmetro controla o estilo do relatório HTML. Aqui, estamos configurando o estilo para que o relatório tenha largura total na visualização HTML.

**profile.to\_notebook\_iframe():** Método que exibe o relatório de perfil diretamente no notebook como um iframe. Isso permite que você visualize o relatório sem precisar abrir um arquivo HTML separado.

**profile.to\_file(output\_file="Relatório Base de Dados.html"):**

* + **output\_file="Relatório Base de Dados.html":** Especifica o nome do arquivo HTML no qual o relatório de perfil será salvo. Neste caso, o nome do arquivo será "Relatório Base de Dados.html".

**2.4 PRÉ-PROCESSAMENTO DE DADOS**

Este código realiza a codificação de variáveis categóricas em números inteiros usando LabelEncoder para cada variável presente na lista fornecida. Isso é útil para preparar os dados categóricos para serem utilizados em modelos de machine learning que requerem entradas numéricas. Segue a explicação de cada elemento do código:

**for variavel in [...]**: Esta é uma estrutura de loop em Python chamada de "loop for". Ele itera sobre cada elemento presente na lista fornecida. No caso deste código, a lista fornecida é **['sex', 'race', 'occupation', 'education', 'workclass', 'marital\_status','relationship', 'native\_country']**, que contém os nomes das variáveis que você deseja codificar.

**data[variavel] = LabelEncoder().fit\_transform(data[variavel])**:

* + **LabelEncoder():** Instancia um objeto da classe LabelEncoder do scikit-learn, que é usado para codificar variáveis categóricas em números inteiros.
  + **fit\_transform(data[variavel])**: Este método ajusta o codificador aos dados da variável (**data[variavel]**) e, em seguida, transforma esses dados para a forma codificada. Os valores codificados são atribuídos de volta à coluna correspondente no DataFrame **data**. Cada categoria na variável categórica é mapeada para um número inteiro único.

**variavel:** O nome da variável sendo iterada no loop atual. A cada iteração do loop, **variavel** assume o valor de uma das variáveis presentes na lista fornecida.

Parte superior do formulário

**3 MANIPULAÇÃO E TRATAMENTOS DE DADOS**

Esses são os elementos do código e suas respectivas funções. Eles são usados para instalar a biblioteca scikit-learn, importá-la para o ambiente Python e importar a função train\_test\_split para dividir os dados em conjuntos de treinamento e teste. Segue a explicação de cada elemento do código:

**!pip install scikit-learn**: Este comando é usado no ambiente Jupyter Notebook para instalar a biblioteca scikit-learn (também conhecida como sklearn) diretamente do Python Package Index (PyPI). Ele adiciona a biblioteca ao ambiente atual, permitindo que você a utilize em seu código.

**import sklearn**: Importa a biblioteca scikit-learn para o seu script Python. Esta biblioteca é uma das mais populares para aprendizado de máquina em Python e fornece uma variedade de algoritmos de aprendizado de máquina, ferramentas de pré-processamento de dados e métricas de avaliação de modelos.

**from sklearn.model\_selection import train\_test\_split**: Importa a função train\_test\_split do módulo model\_selection da scikit-learn. Esta função é usada para dividir os dados em conjuntos de treinamento e teste, o que é crucial para avaliar o desempenho do modelo de aprendizado de máquina.

**3.1 SELECIONA VARIÁVEIS INPUT E OUTPUT**

**3.1.1 INPUT**   
Essa operação é comum quando se está trabalhando com conjuntos de dados e deseja-se selecionar apenas um subconjunto das colunas para análise ou modelagem. Neste caso, as colunas selecionadas são consideradas relevantes para o problema em questão, e elas serão usadas como *features* para algum tipo de análise ou modelagem posteriormente. Segue a explicação de cada elemento do código:

**X = data[['sex', 'race', 'hours\_per\_week', 'occupation', 'education', 'workclass', 'marital\_status','relationship']]:**

* + **data[['sex', 'race', 'hours\_per\_week', 'occupation', 'education', 'workclass', 'marital\_status','relationship']]:** Esta parte do código seleciona um subconjunto específico das colunas do DataFrame **data**. No caso, estamos selecionando as colunas 'sex', 'race', 'hours\_per\_week', 'occupation', 'education', 'workclass', 'marital\_status' e 'relationship'. Isso cria um novo DataFrame **X** contendo apenas essas colunas.
  + **X = ...**: Atribui o novo DataFrame criado à variável **X**. Este novo DataFrame contém apenas as colunas selecionadas, que são usadas como features (variáveis independentes) em análises posteriores ou em modelos de machine learning.

**3.1.2 OUTPUT**

Dessa forma, a variável **Y** contém uma lista onde cada elemento representa o número de ocorrências de um valor único na coluna 'salary'. Essa lista pode ser útil para várias análises, como visualização ou modelagem de dados. Segue a explicação de cada elemento do código:

**Y = data['salary'].value\_counts().tolist()**:

* + **data['salary']**: Isso seleciona a coluna 'salary' do DataFrame **data**. Presumivelmente, esta coluna contém os salários dos indivíduos.
  + **value\_counts()**: Esta função do pandas conta o número de ocorrências únicas em uma série. No caso, está contando o número de ocorrências únicas de cada valor na coluna 'salary'.
  + **tolist()**: Este método converte a série resultante de contagens em uma lista Python. Cada elemento da lista corresponde ao número de ocorrências de um valor único na coluna 'salary'.

**3.2 PREPARA DADOS PARA MODELAGEM DE MACHINE LEARNING.**

Essas operações são comuns ao preparar dados para modelagem de machine learning. A variável **X** conterá as features (ou variáveis independentes) que serão usadas para prever a variável alvo **Y** (ou variável dependente), que contém os salários dos indivíduos neste caso. Segue a explicação de cada elemento do código:

**X = data.drop('salary', axis=1)**:

* + **data.drop('salary', axis=1)**: Esta expressão remove a coluna 'salary' do DataFrame **data**, produzindo um novo DataFrame **X** que contém todas as colunas exceto 'salary'.
  + **axis=1**: Este parâmetro indica que a operação deve ser realizada ao longo do eixo das colunas. Isso significa que o pandas irá remover a coluna 'salary' do DataFrame **data**.

**Y = data['salary']**:

* + **data['salary']**: Esta expressão seleciona a coluna 'salary' do DataFrame **data**, criando uma série **Y** que contém os salários dos indivíduos.

**3.3 SEPARANDO DADOS PARA TREINO E TESTE**

Essa função **train\_test\_split** é frequentemente usada para dividir os dados em conjuntos de treinamento e teste, permitindo avaliar o desempenho do modelo em dados não vistos. Isso é fundamental para evitar overfitting e avaliar a capacidade de generalização do modelo. Segue a explicação cada elemento do código fornecido:

* + 1. **X\_train, X\_test, y\_train, y\_test**:

As quatro variáveis à esquerda do sinal de igualdade recebem os conjuntos de dados resultantes da divisão:

* + **X\_train**: Conjunto de features de treinamento.
  + **X\_test**: Conjunto de features de teste.
  + **y\_train**: Conjunto de labels de treinamento correspondentes aos dados em X\_train.
  + **y\_test**: Conjunto de labels de teste correspondentes aos dados em X\_test.
    1. **train\_test\_split (X, Y, test\_size=0.2, random\_state=42)**:
  + **X**: O conjunto de features (variáveis independentes) que será dividido em conjuntos de treinamento e teste.
  + **Y**: O conjunto de labels (variável dependente) correspondente aos dados em X.
  + **test\_size=0.2**: Especifica a proporção do conjunto de dados que será reservada para o conjunto de teste. Neste caso, 20% dos dados serão usados para teste.
  + **random\_state=42**: Controla a aleatoriedade na divisão dos dados. Definindo um valor fixo para **random\_state**, garantimos que os dados serão divididos da mesma maneira sempre que o código for executado. Isso é útil para garantir a reprodutibilidade dos resultados.

1. **MÁQUINA PREDITIVA**

**4.1 CRIA O MODELO**

O termo **clf\_RF** é uma instância do RandomForestClassifier com 1000 árvores na floresta. Este classificador pode ser usado para treinar um modelo de classificação baseado em Random Forest nos dados fornecidos. Segue o detalhamento de cada elemento do código:

1. **clf\_RF = RandomForestClassifier(n\_estimators=1000)**:
   * **RandomForestClassifier**: Esta é uma classe do scikit-learn que implementa o algoritmo de Random Forest para classificação. Um Random Forest é um ensemble de árvores de decisão, onde várias árvores são treinadas em subconjuntos aleatórios dos dados e suas previsões são combinadas para produzir uma previsão final.
   * **n\_estimators=1000**: Este é um parâmetro do RandomForestClassifier que especifica o número de árvores na floresta. Neste caso, estamos configurando **n\_estimators** para 1000, o que significa que serão treinadas 1000 árvores na floresta.

**4.2 TREINA MODELO**

Este trecho do script significa que estamos treinando o classificador RandomForest (**clf\_RF**) nos dados de treinamento (**X\_train** e **y\_train**). Após o treinamento, o modelo está pronto para fazer previsões sobre novos dados. Segue explicação de cada termo do código:

**clf\_RF:** Esta é uma variável que representa o classificador RandomForest que você está usando. É comum nomear o modelo como **clf** (abreviação de classifier) seguido do nome do algoritmo. Neste caso, **clf\_RF** indica que estamos usando um classificador Random Forest.

**fit():** **fit()** é um método da classe do modelo de machine learning que treina o modelo nos dados de treinamento fornecidos. Durante o treinamento, o modelo ajusta seus parâmetros internos para encontrar padrões nos dados que permitam fazer previsões precisas.

**X\_train:** Este é o conjunto de features (variáveis independentes) de treinamento. Ele contém os dados que o modelo usará para aprender os padrões nos dados.

**y\_train:** Este é o conjunto de labels (variável dependente) de treinamento. Ele contém os valores reais que o modelo tentará prever com base nos dados de treinamento.

Após o treinamento, o modelo está pronto para fazer previsões sobre novos dados. Segue explicação de cada termo do código:

**5 AVALIAÇÃO DA MÁQUINA PREDITIVA**

Este trecho de código calcula a acurácia do modelo de RandomForest (**clf\_RF**) com base nos dados de teste e exibe a acurácia na tela. Isso permite que você avalie o desempenho do modelo em dados não vistos.

segue detalhamento de cada termo do código:

**accuracy = clf\_RF.score(X\_test, y\_test)**:

* + **clf\_RF.score(X\_test, y\_test)**: O método **score()** do classificador RandomForest (**clf\_RF**) calcula a acurácia do modelo com base nos dados de teste (**X\_test**) e nos rótulos reais correspondentes (**y\_test**). A acurácia é uma métrica comum usada para avaliar a precisão de um modelo de classificação, representando a proporção de previsões corretas sobre o total de previsões feitas pelo modelo.
  + **accuracy = ...**: A acurácia calculada é atribuída à variável **accuracy**.

**print('accuracy:', str(accuracy))**:

* + **print()**: A função **print()** é usada para exibir uma mensagem na saída padrão (geralmente, a tela do console).
  + **'accuracy:', str(accuracy)**: Esta é a mensagem que será exibida. **accuracy** é a acurácia do modelo calculada anteriormente. Usamos **str(accuracy)** para converter o valor numérico da acurácia em uma string antes de exibi-lo na tela.

**6 CONSIDERAÇÕES FINAIS**

Utilizando o método pré-estabelecido ao avaliar o problema a ser solucionado foi possível chegar a uma máquina preditiva com acurácia de 0.8622754491017964, o que é um resultado bastante satisfatório, já que o sistema detecta quem vai receber acima de 50 mil reais levando em consideração os parâmetros disponíveis, em números arredondados, de 86 pessoas a cada 100.