Nome: _____Matrícula:_____
Data:__/__/____

EXERCÍCIO PRÁTICO DE MINERAÇÃO DE DADOS
CLASSIFICAÇÃO

Considere a seguinte situação:

- Uma agência de venda de veículos pretende estimar se um determinado veículo será vendido ou não com base em 3 características: - Preço, - Quilometragem, - Ano de Fabricação.

Os dados para treino e testes do modelo encontram-se no arquivo CSV: ***car-prices.csv***

mileage_per_year	**model_year**	**price**	**sold**
21801	2000	30941.02	yes

- **mileage_per_year:** indica a quantidade de milhas que o carro rodou por ano desde sua fabricação.
- **model_year:** indica o ano de fabricação.
- **price**: indica o preço.
- **sold**: Indica se o carro foi vendido ou não (yes / no).

EXERCÍCIO:

1) Crie um programa em Python que possa ler os dados do arquivo CSV, treinar e aprender com base nos registros e testar possíveis classificações de novos carros que poderão ser vendidos ou não.

Atividades:

- Pré-processando dados com *dataframe* do Pandas:
  - Utilize a biblioteca ***Pandas*** para leitura do arquivo CSV.
  - Renomeie as colunas para português utilizando o *rename* do *Pandas*.
  - *Sold*: Mapeie a coluna *sold* para uma nova coluna com valores 1(yes) ou 0(no).
  - *Model_year*: Ao invés de usar a dimensão ano do modelo, crie uma nova coluna que armazene a idade do veículo (Ano atual - ano do modelo).
  - Mileage_per_year: Crie uma nova coluna que armazene a quilometragem do veículo (em quilômetros) → Milhas por ano * 1.60934 * idade do modelo.
  - Deixe o dataframa somente com as colunas (quilometragem, idade, preco, vendido). Delete as demais colunas.

- Explore as bibliotecas ***Matplotlib*** e ***Seaborn*** para criar um gráfico de 3 dimensões para visualizar o modelo.

- Utilize a biblioteca ***SKLearn*** para treinar e testar a taxa de acertos de veículos que serão vendidos ou não.
  - Para a divisão das bases de treino e teste utilize a função ***train_test_split***
  - Utilize 25% da massa de dados para testes.
  - Utilize um *random_test* padrão com SEED = 20.
  - Estratifique a porcentagem de Y para as amostras de treino e teste.
- Encontre o *baseline*, ou seja, a classificação boba, para usar como parâmetro de avaliação de teste de precisão do algoritmo. Utilize a classe *DummyClassifier* com constante 1 para teste.
- Dentre os algoritmos abaixo, verifique qual algoritmo de classificação apresentará melhor taxa de acerto:

from *sklearn.dummy* ***import*** *DummyClassifier*

from *sklearn.svm* ***import*** *LinearSVC*

from *sklearn.svm* ***import*** *SVC*

from *sklearn.ensemble* ***import*** *RandomForestClassifier*

from *sklearn.linear_model* ***import*** *LinearRegression*

from *sklearn.linear_model* ***import*** *LogisticRegression*

from *sklearn.neighbors* ***import*** *KNeighborsClassifier*

from *sklearn.naive_bayes* ***import*** *MultinomialNB*

from *sklearn.naive_bayes* ***import*** *GaussianNB*

from *sklearn.tree* ***import*** *DecisionTreeClassifier*