Aula 3 - aprendizado de máquina com scikit-learn

Jayme Anchante

24 de fevereiro de 2021

motivação

carregando os dados

```
1  from sklearn.datasets import load_iris
2  iris = load_iris()
```

descrição dos dados

```
1  print(iris.DESCR)
```

montando base única

Crie um objeto chamado `df` que seja um pd.DataFrame com os atributos e as espécies de iris.

Dica: qual o tipo do objeto iris? veja as chaves.

montando base

```
1  import pandas as pd
2  df = pd.DataFrame(iris.data, columns=iris.feature_names)
3  df.loc[:, "species"] = iris.target
4  def f(x):
5      return iris.target_names[x]
6
7  df.loc[:, "species"] = df["species"].apply(f)
```

desafio

Dada a iris

```
vals = [[2, 3, 2.5, 0.7]]
uma_iris = pd.DataFrame(vals, columns=iris.feature_names)
```

A qual espécie de íris esta flor pertence?

algoritmo

Crie um algoritmo/função que receba como argumento `uma_iris` e retorno um texto com a espécie de iris que ela seria?

Dica: Comece escrevendo em linguagem natural (português) as etapas que você gostaria de seguir, depois tente implementar as etapas.

Dica: explore os dados, medidas de centralidade, dispersão, agrupamento etc.

regras de negócios

```
col = "petal length (cm)"
print(df.groupby("species")[col].mean())
if uma_iris.loc[0, col] < 3:
    print("setosa")
elif uma_iris.loc[0, col] >= 3 and uma_iris.loc[0, col] < 4
    print("versicolor")
elif uma_iris.loc[0, col] >= 4.5:
    print("virginica")
```

algoritmo das médias

Regressão linear

```
medias = df.groupby("species").mean()
diff = medias - uma_iris.loc[0, ]
diff_abs = diff.abs()
s = diff_abs.idxmin()
vc = s.value_counts()
vc.index[0]
```

vizinhos

```
df = df.set_index("species")
diff = df - uma_iris.loc[0]
diff_abs = diff.abs()
soma = diff_abs.sum(axis=1)
soma.idxmin()
# n é a quantidade de vizinhos
n = 3
soma.sort_values().head(n)
```

teoria e conceitos

bibliografia

Conteúdo baseado no capítulo 5 do livro Python Data Science Handbook de Jake VanderPlas

aprendizado de máquina

O que é:

- Subcategoria da inteligência artificial
- Aprender as correlações de um fenômeno a apartir de dados históricos
- Série de heurísticas para entender o padrão dos dados

O que não é:

- Solução mágica para todos os problemas
- Sempre efetiva, a melhor solução, fácil de implementar
- Díficil de aplicar

tipos de aprendizado de máquina

- Supervisionado: modela a relação entre dados de entrada (conhecidos como `features` ou `características`) e os dados de saída (conhecidos como `target` ou `alvo`)
- Não supervisionado: modela os dados de entrada sem o conhecimento de seu rótulo (`label` em inglês, o alvo)

aprendizado supervisionado

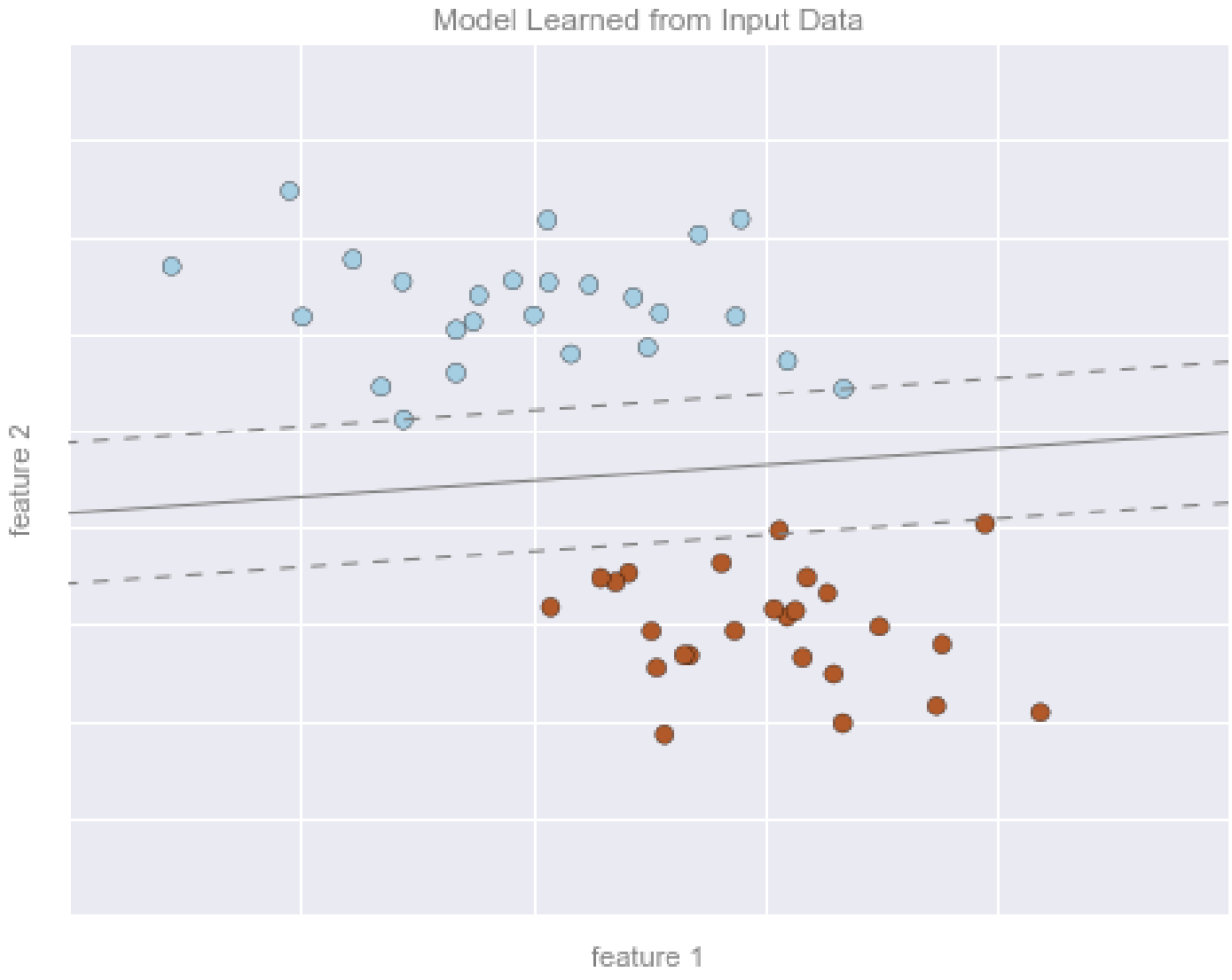
- Regressão: quando o problema envolve alvos contínuos ou numéricos. E.g. prever a idade dos usuários a partir de sua foto de perfil ou outras características; prever a receita de uma empresa a partir do balanço contábil do ano anterior; prever a renda de um indivíduo a partir de suas características demográficas e profissionais.
- Classificação: quando o problema envolve alvos categóricos ou textuais. E.g. prever se uma transação é uma fraude ou não; recomendar uma música com base nas músicas anteriores escutadas; detectar a presença ou não de um cachorro em uma foto.

aprendizado não supervisionado

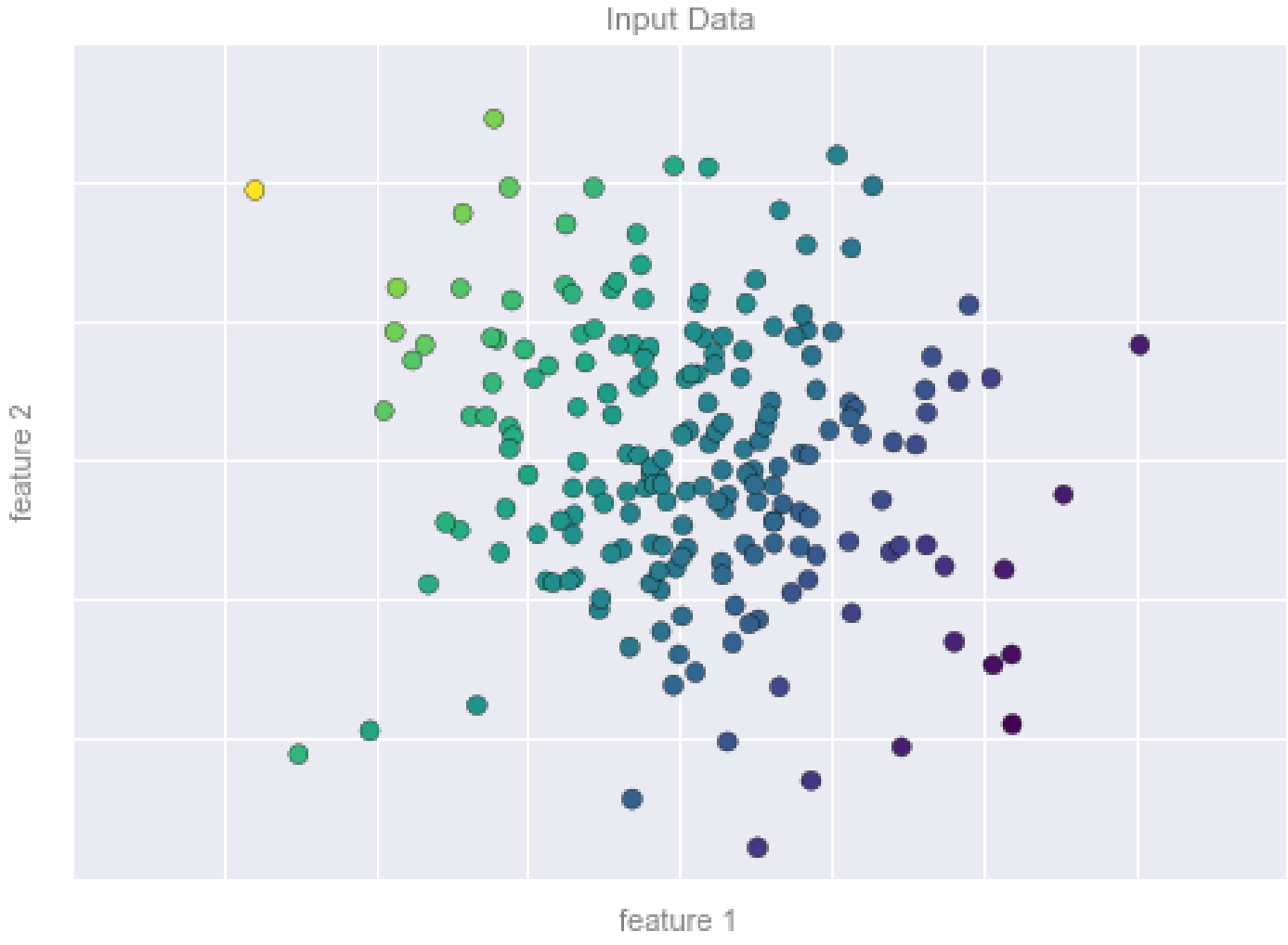
- Clusterização: agrupamento das amostras por afinidade. E.g. encontrar ocorrências anômalas caso pertençam a um certo grupo; segmentar clientes automaticamente; separar textos por tópicos.
- Redução de dimensionalidade: representar os dados de maneira mais sucinta. E.g. reduzir a quantidade de features a serem utilizadas; encontrar representações mais eficientes dos dados (embedding)

visualização de um problema de classificação

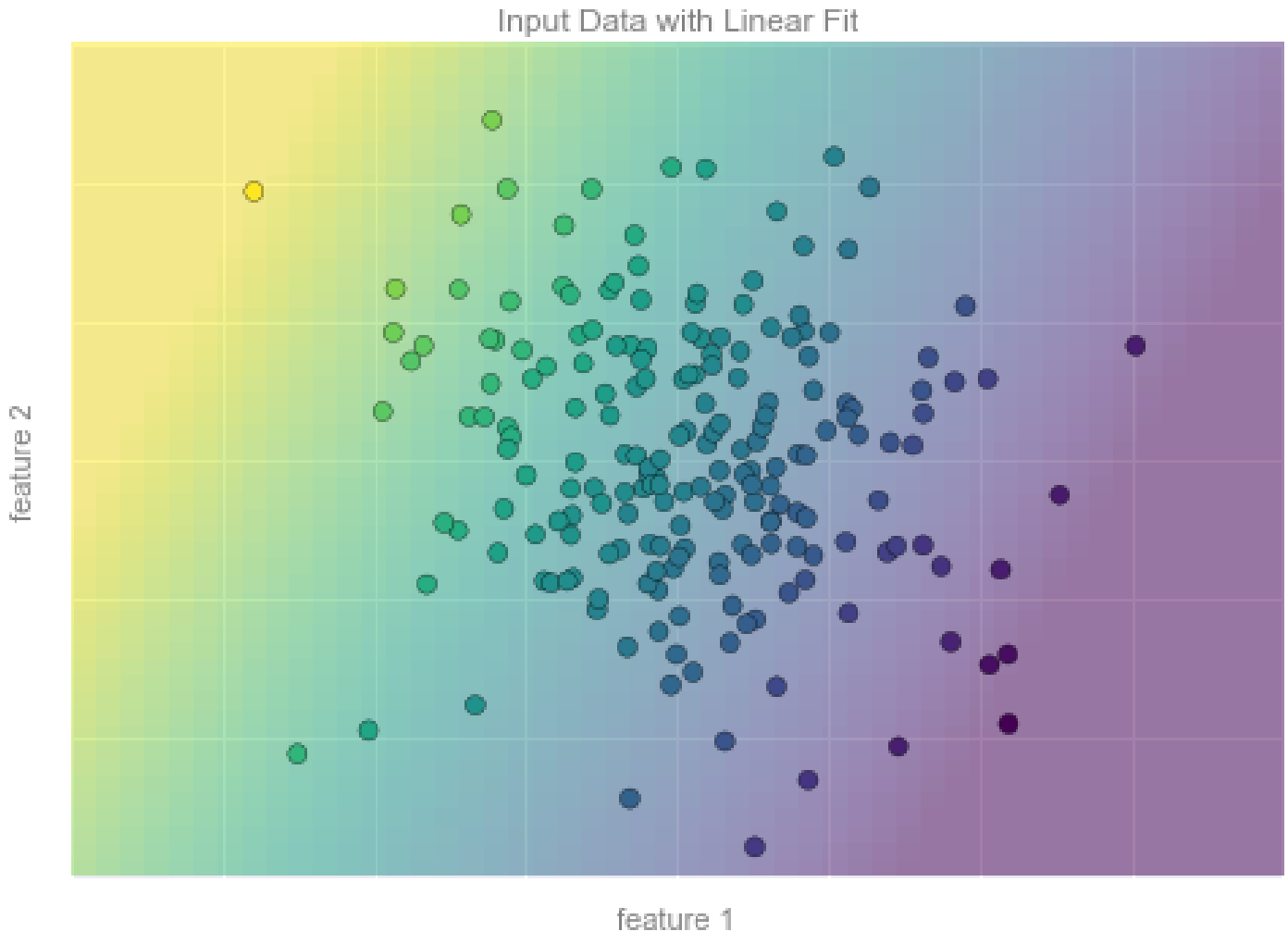
proposição



visualização de um problema de regressão

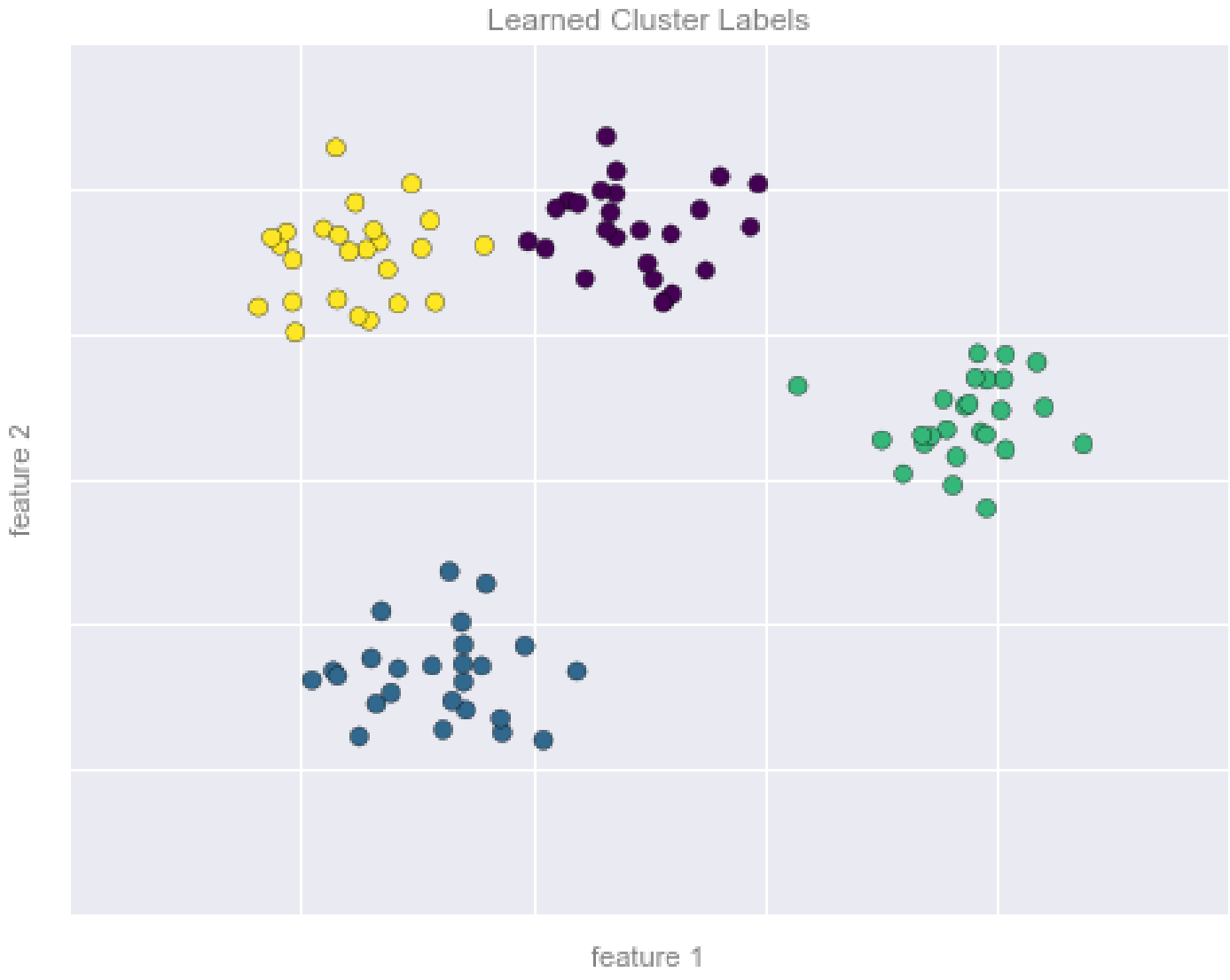


proposição

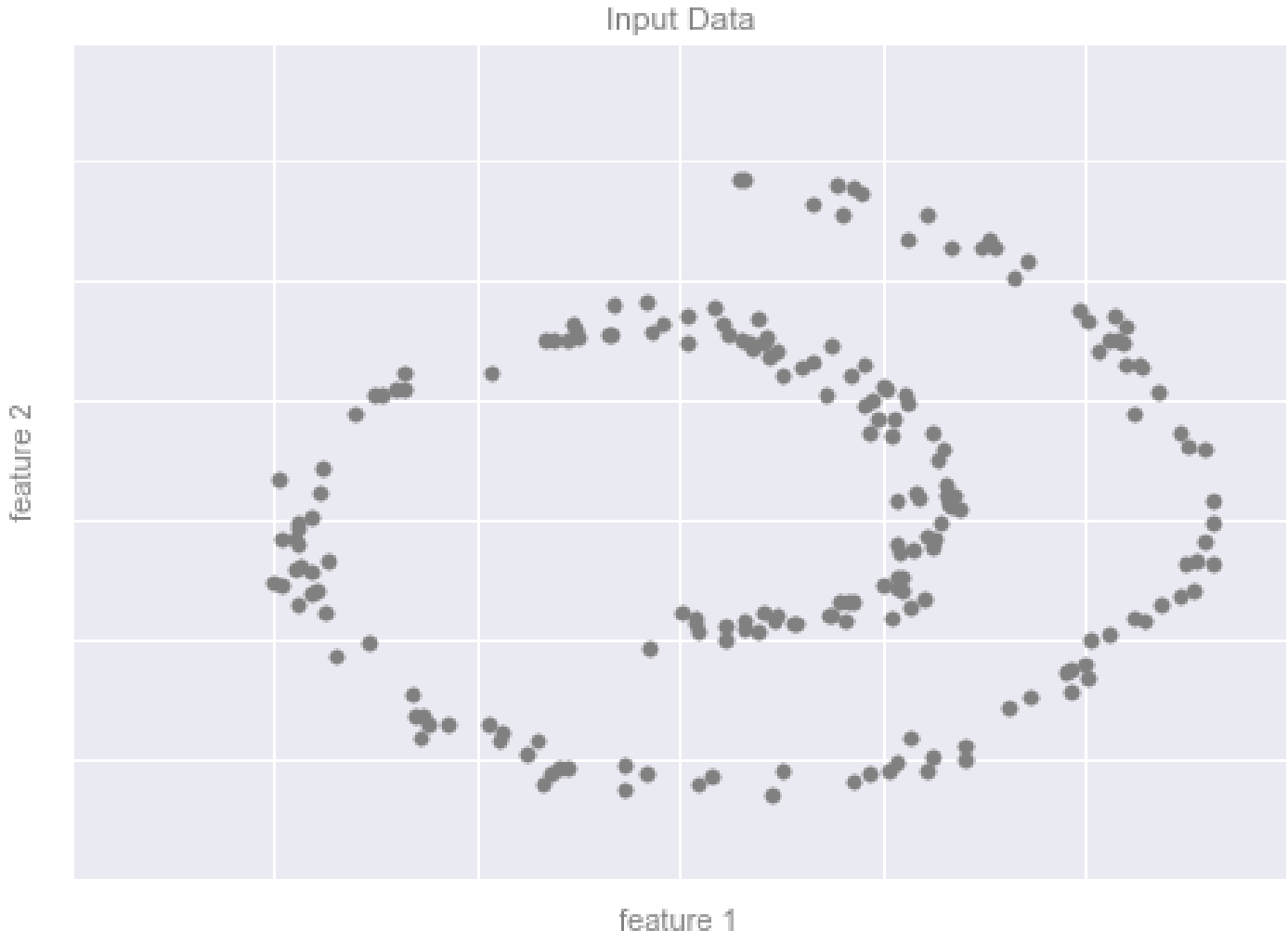


visualização de um problema de clusterização

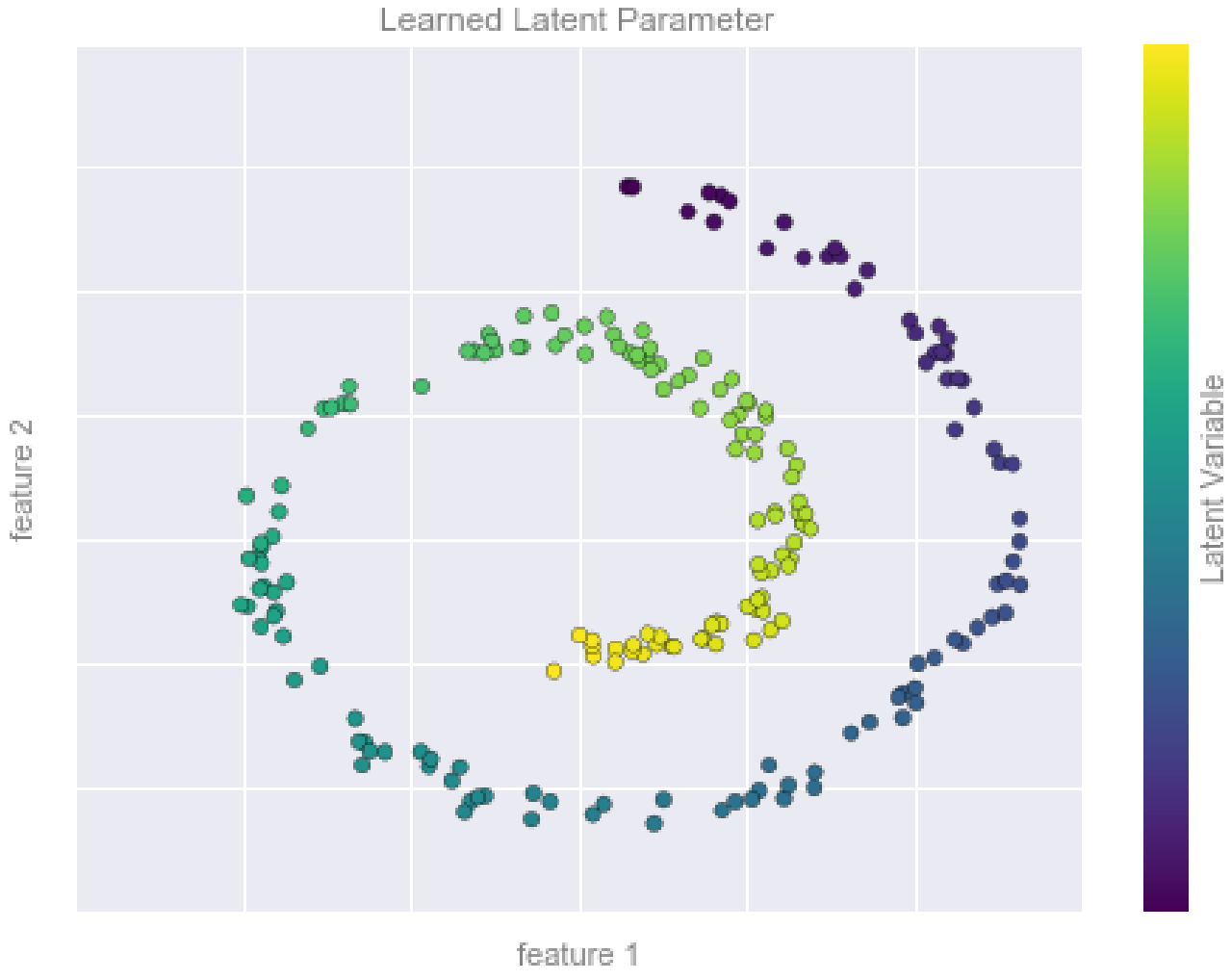
proposição



visualização de um problema de redução de dimensionalidade



proposição


Learned Latent Parameter
feature 2
feature 1
Latent Variable

a arte do aprendizado de máquina

método da redução

O reducionismo é uma forma de descrever fenômenos complexos em outros menores e simplificados

Em vez de

- Fazer um modelo para aumentar as vendas do comércio eletrônico da empresa

Uma redução seria

- Melhorar a experiência do usuário ofertando produtos condizentes com seu perfil (recomendação de produtos)
- Prover uma melhor estimativa de quando o produto chegará (previsão de tempo de envio)
- Mandar um cupom de desconto apenas para aqueles usuários que não converteriam sem ele (algoritmo de retenção)

modelo

Modelo é uma forma simplificada de representar a realidade em termos matemáticos

No caso de uma previsão de renovação de uma subscrição (churn), para sermos 100% efetivos precisaríamos saber exatamente os planos futuros de uma pessoa quanto ao serviço.

Um modelo poderia levar em conta atrasos no pagamento, data da última entrada no sistema, quantidade de ações realizadas na plataforma, atualizações do aplicativo etc.

na prática

As empresas querem aumentar vendas, entender e melhorar a experiênca do usuário, tomar decisões baseadas em dados.

Cabe a você cientista reduzir esses problemas genéricos em um problema de dados e explicitá-lo por meio de um modelo que possa ser utilizado e trazer retornos tangíveis.

1. Queremos tomar decisões baseadas em dados.
2. Mesmo email promocional é enviado para a base toda.
3. Segmentar a base em um atributo simples (idade ou data do último login)
4. Usar os passos anteriores como benchmark e tentar introduzir um algoritmo que traga um resultado melhor

scikit-learn: modelos

o “atalho”

```
1  from sklearn.linear_model import LogisticRegression
2  model = LogisticRegression(max_iter=1000)
3  model.fit(df.iloc[:, :4], df.iloc[:, 4])
4  model.predict(uma_iris)
```

a biblioteca

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Fonte: site do scikit-learn

componentes

- Classificação
- Regressão
- Clusterização
- Redução de dimensionalidade
- Seleção de modelos
- Pré processamento

Fonte: site do scikit-learn

empresas que utilizam

- J.P.Morgan
- Spotify
- Evernote
- Booking.com

Fonte: testemunhos no site do scikit-learn

módulos

Podem ser visualizados em na API

Lógica básica consiste em:

1. Importar o estimador apropriado
2. Escolher os parâmetros
3. Organizar os dados como características `X` e opcionalmente o alvo `y`
4. Aplicar o método `.fit` do estimador nos dados históricos
5. Aplicar o método `.predict` do estimador ajustado em dados novos

começando em scikit-learn

```
1  from sklearn.ensemble import RandomForestClassifier
2  clf = RandomForestClassifier(random_state=0)
3  X = [[ 1,  2,  3],  # 2 samples, 3 features
4       [11, 12, 13]]
5  y = [0, 1]  # classes of each sample
6  clf.fit(X, y)
7  clf.predict([[4, 5, 6]])  # predict classes of new data
```

aviso legal

Este não é um curso completo de estatística e o tempo limitado, a base teórica de cada modelo não será muito discutida nem aprofundada.

Ainda, para o praticante de aprendizado de máquina não é estritamente necessário (ainda que recomendável) conhecer o funcionamento de cada algoritmo.

Para entender o funcionamento de cada algoritmo, consultar o livro canônico: The Elements of Statistical Learning: Data Mining, Inference, and Prediction de Trevor Hastie, Robert Tibshirani e Jerome Friedman

modelos lineares

Muito utilizando como baseline para um problema por (quase) não terem parâmetros

Oferecem grandes vantagens para interpretação e testes de hipótese

Fonte

```
sklearn.linear_model.LinearRegression(
    fit_intercept=True,
    normalize=False,
    copy_X=True,
    n_jobs=None,
    positive=False
)
```

modelos de vizinhança

Não há aprendizado no estrito senso, mas uma previsão é realizada utilizando a interpolação do vizinhos da base de treinamento

Fonte

```
sklearn.neighbors.KNeighborsRegressor(
    n_neighbors=5,
    weights='uniform',
    algorithm='auto',
    leaf_size=30,
    p=2,
    metric='minkowski',
    metric_params=None,
    n_jobs=None
```

modelos de árvore

Fácil de extrair regras lógicas e visualização destas regras

Fonte

```
sklearn.tree.DecisionTreeRegressor(
    criterion='mse',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    min_impurity_split=None,
    ccp_alpha=0.0
)
```

modelos conjuntos: bagging

Diversas árvores de decisão cada uma recebendo uma pequena porção dos dados

Grande poder preditivo, fácil paralelização

Fonte

```
sklearn.ensemble.RandomForestRegressor(
    n_estimators=100,
    criterion='mse',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features='auto',
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    min_impurity_split=None,
    # ...
```

modelos conjuntos: boosting

Várias árvores de decisão em sequencia consertando os erros das árvores anteriores

Fonte

```
sklearn.ensemble.GradientBoostingRegressor(
    loss='ls',
    learning_rate=0.1,
    n_estimators=100,
    subsample=1.0,
    criterion='friedman_mse',
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_depth=3,
    #...
    validation_fraction=0.1,
    n_iter_no_change=None,
    tol=0.0001
```

redes neurais

Algoritmo de aprendizado profundo (todos os anteriores pertencem a categoria de aprendizado raso) muito utilizado para problemas envolvendo dados não tabulares (image, áudio, vídeo, texto)

Fonte

```
sklearn.neural_network.MLPRegressor(
    hidden_layer_sizes=100,
    activation='relu',
    solver='adam',
    alpha=0.0001,
    batch_size='auto',
    learning_rate='constant',
    learning_rate_init=0.001,
    max_iter=200,
    tol=0.0001,
    early_stopping=False,
    validation_fraction=0.1,
    n_iter_no_change=10,
```

clustering com kmeans

Escolha de grupos acontece definição posições para os centróides e tentando melhorar a posição deles de com a disperção de cada grupo.

Fonte

```
sklearn.cluster.KMeans(
    n_clusters=8,
    init='k-means++',
    n_init=10,
    max_iter=300,
    tol=0.0001,
    random_state=None,
    algorithm='auto'
    )
```

redução de dimensionalidade com pca

Redução linear de dimensionalidade usando SVD para projeção em um espaço dimensional reduzido.

Fonte

```
sklearn.decomposition.PCA(
    n_components=None,
    copy=True,
    whiten=False,
    svd_solver='auto',
    tol=0.0,
    iterated_power='auto',
    random_state=None
```

exercícios

1. Aplique kmeans a base de dados de iris com `n_clusters` igual a 3 (pois sabemos de antemão que a base possui 3 espécies) e com `random_state` igual a 42.
2. Tente dar nome a cada um dos clusters encontrados.
3. Utilizando a base de dados boston, ajuste uma rede neural mlp com duas camadas de 25 neurônios cada uma (`hidden_layer_sizes=(25,25)`), com uma máximo de 1000 iterações (`max_iter=1000`) e com `early_stopping` como verdadeiro.
4. Aplique o método `.predict` da mlp ajustada nos dados, qual a diferença entre os valores previstos e os valores reais? Faça a média e a soma absoluta do erro.
5. Reduza para apenas duas dimensões a base de dados boston e ajuste novamente a rede neural, sua resposta mudou na pergunta anterior?

scikit-learn: demais ferramentas

dados

Para os seguintes slides iremos utilizar a base iris e boston

```
from sklearn.datasets import load_iris
from sklearn.datasets import load_boston
X_reg, y_reg = load_boston(True)
X_clf, y_clf = load_iris(True)
```

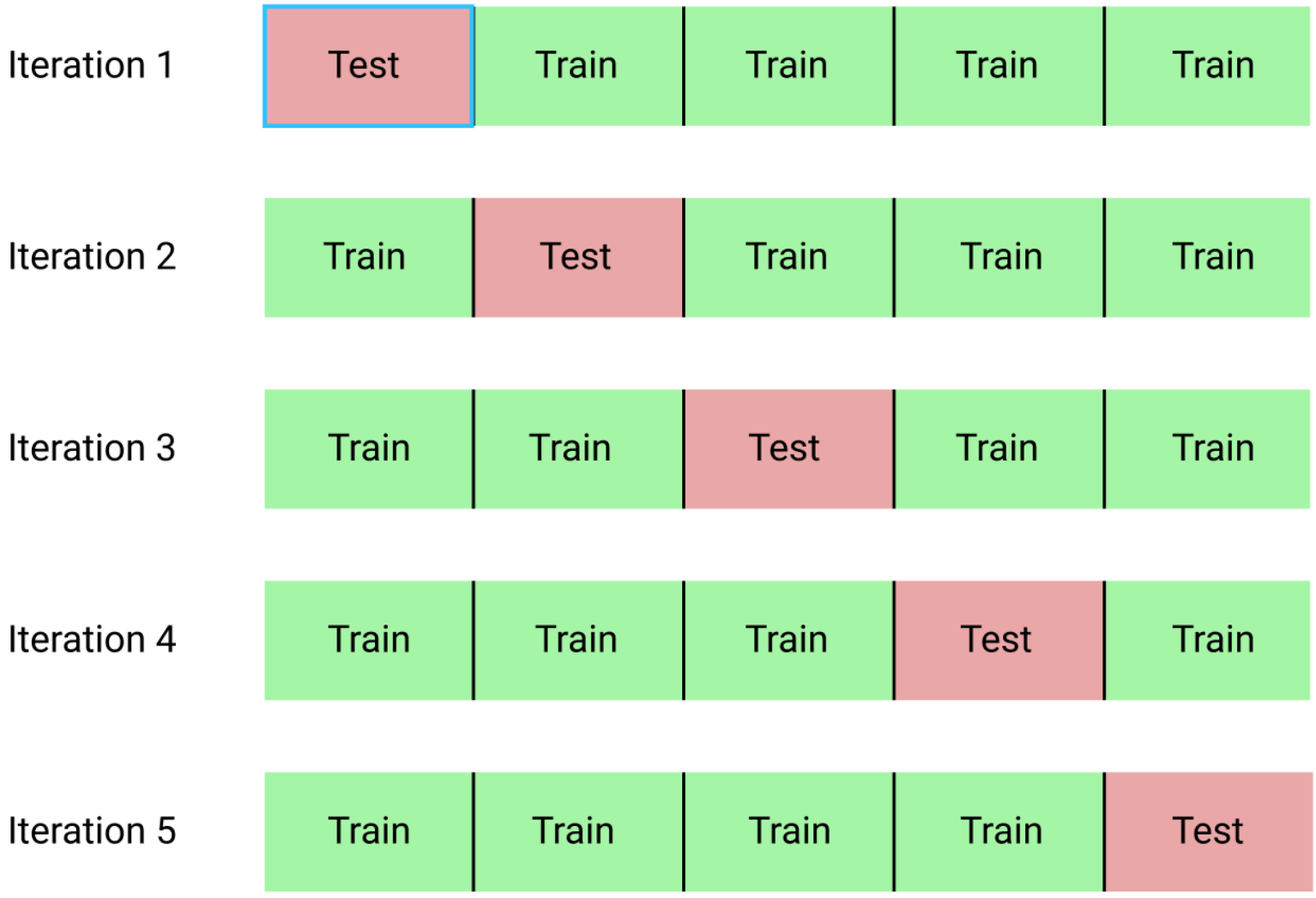
seleção de modelos: teste

É uma boa prática separar uma porção do teste para podermos aferir se estamos sendo efetivos com nosso modelo

```
sklearn.model_selection.train_test_split(
    test_size=None, train_size=None,
    random_state=None,
    shuffle=True,
    stratify=None
    )
```

```
from sklearn.model_selection import train_test_split
reg_train_test = train_test_split(
    X_reg, y_reg, test_size=0.2, random_state=42
    )
clf_train_test = train_test_split(
    X_clf, y_clf, test_size=0.2, random_state=42
    )
print(reg_train_test[0].shape, reg_train_test[1].shape)
```

seleção de modelos: visualizando validação cruzada



seleção de modelos: dobras para descoberta de parâmetros

```
1  from sklearn.model_selection import KFold
2  kf = KFold(n_splits=3)
3  splits = kf.split(X_clf)
4  for train, test in splits:
5      X_train, y_train = X_clf[train], y_clf[train]
6      X_test, y_test = X_clf[test], y_clf[train]
7      # do something
```

seleção de modelos: busca exaustiva

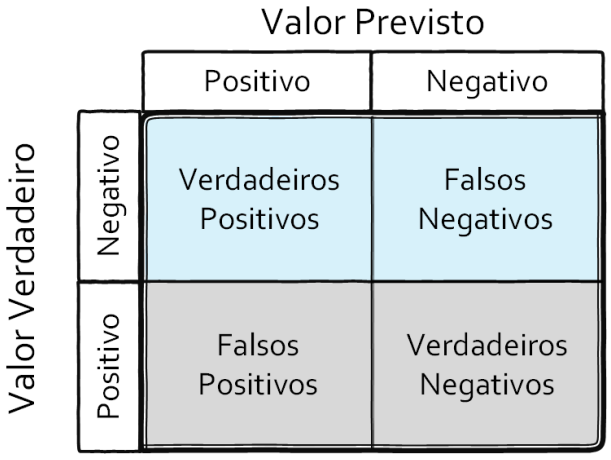
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
parameters = {"max_depth": [2, 4], "splitter": ["best", "ra
grid = GridSearchCV(dt, parameters)
grid.fit(clf_train_test[0], clf_train_test[2])
```

métricas: classificação

- Acurácia é a quantidade de classificações corretas sobre o total de predições
- Top-k é quando acertamos a classificação nas k classificações com maior probabilidade

```
1 from sklearn.metrics import (
2     accuracy_score,
3     top_k_accuracy_score,
4     confusion_matrix
5 )
```

visualização da matriz de confusão




```
1  y_true = clf_train_test[3]
2  y_pred = grid.predict(clf_train_test[1])
3  confusion_matrix(y_true, y_pred)
```

métricas: regressão

- Erro absoluto médio: valor previsto menos valor efetivo de todas as linhas, dividido pelo número de linhas
- Erro quadrático médio: valor previsto menos valor efetivo ao quadrado, dividido pelo número de linhas. Penaliza mais os grandes erros. É a medida utilizada pela regressão linear para otimização.
- R2 (coeficiente de determinação): variância do alvo que é explicado pelas características de entrada.

```
1  from sklearn.metrics import (
2      mean_absolute_error,
3      mean_squared_error,
4      r2_score
5  )
```

pré processamento

- Normalização: tirar a média e dividir pelo desvio padrão
- Codificador: "transformar" variáveis categóricas em numéricas

```
1  from sklearn.preprocessing import (
2      StandardScaler,
3      OneHotEncoder
4  )
5  X = [["a"], ["b"], ["c"], ["a"]]
6  ohe = OneHotEncoder()
7  ohe.fit_transform(X).toarray()
```

pipeline

Vamos encadenar uma série de transformações até chegar no modelo final

Tudo será armazenado em um único objeto, para ser fácil reproduzir todos as etapas

Seguiremos este passo a passo

A base de dados do titanic pode ser encontrada aqui

exercícios

```
X_train = reg_train_test[0]
X_test = reg_train_test[1]
y_train = reg_train_test[2]
y_test = reg_train_test[3]
```

1. Ajuste um modelo linear na base de treino.
2. Veja o erro quadrático médio na base de teste com o modelo anterior.
3. Faça uma busca exaustiva na base de treino usando um modelo de florestas aleatórias e variando pelo menos 2 parâmetros.
4. Você conseguiu melhorar o erro quadrático médio no teste?

exercício

dados

Use a base de dados Adult da UCI

A parte “Data Set Description” contém toda descrição da base de dados

O “Data Folder” contêm os dados de treinamento e teste

lista

1. Remova os NaN e treine uma regressão logística.
2. Substitua os NaN pela média da coluna (dê uma olhada no imputer) e faça uma busca exaustiva por pelo menos 4 parâmetros de um algoritmo de sua escolha.
3. Escreva um pipeline de processamento de dados e treinamento de modelo utilizando a classe `Pipeline` do scikit-learn.