APRENDIZAGEM DE MÁQUINA

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br
https://github.com/josenalde/machinelearning

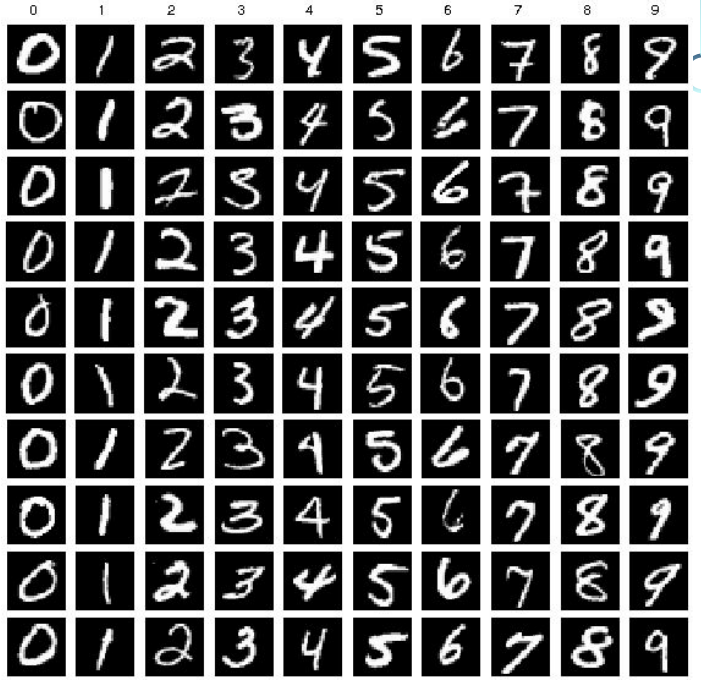
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

AVALIANDO MODELOS DE CLASSIFICAÇÃO

MATRIZ DE CONFUSÃO

[1] Ferramenta comum para avaliar desempenho de classificador

[2] Ideia geral: contar quantas vezes as instâncias de classe A são classificadas como classe B. Por exemplo, um classificador **multiclasse** (ou multinomial) como SGD, randomForest ou naiveBayes (que tratam multiclasse nativamente) teríamos uma matriz 10 x 10 para o dataset MNIST (0-9) e para saber quantas imagens de 3s foram confundidas com imagens de 5s olharíamos na quarta linha e na sexta coluna



Exemplo: SGD@MNIST para dígito 5

[3] Quando se tem a coluna target disponível no conjunto de teste final, pode-se predizer as saídas com o modelo final escolhido e sintonizado e gerar as métricas a partir da matriz de confusão. **Contudo, para comparar modelos, pode-se usar cross_val_predict para gerar predições nos conjuntos de validação em cada fold, selecionar N 'melhores' e aplicar no conjunto de teste**

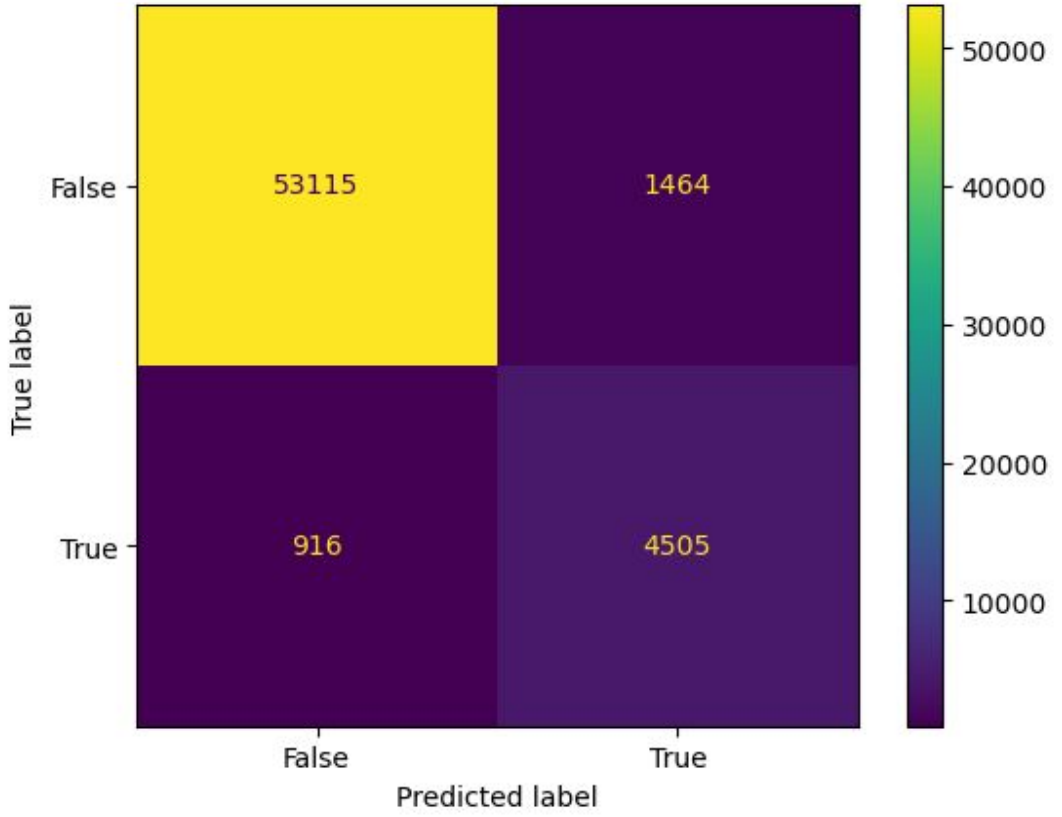
AVALIANDO MODELOS DE CLASSIFICAÇÃO

MATRIZ DE CONFUSÃO

[4] Linhas: classes reais, Colunas: classes previstas (preditas)

- 53115 imagens não-5 (classe negativa) foram classificadas corretamente como não-5 (verdadeiro negativo: TN/VN) enquanto as 1464 restantes classificadas erroneamente como 5s (classe positiva) (falso positivo: FP - ERRO I (**alarme falso**))
- 916 imagens de 5s classificadas erroneamente como não-5 (falso negativo: FN - ERRO II (**falha**)) e as 4505 restantes corretamente classificadas como 5s (verdadeiro positivo).

Problema desbalanceado (comum em doenças raras, detecção de fraude etc.)

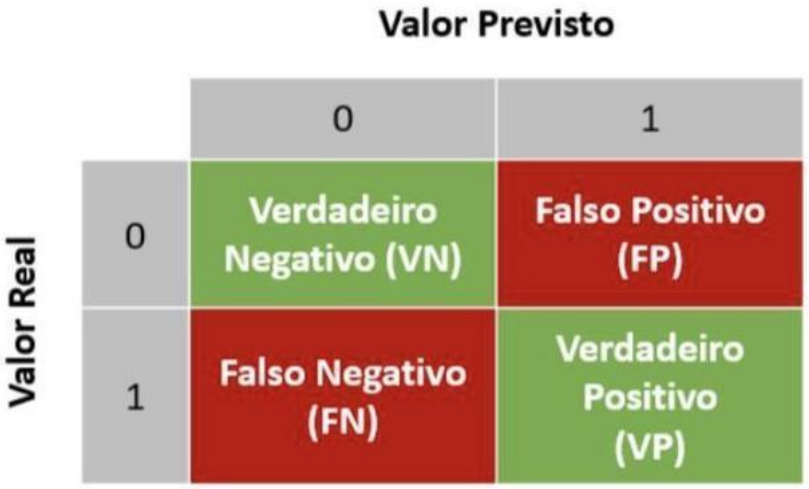


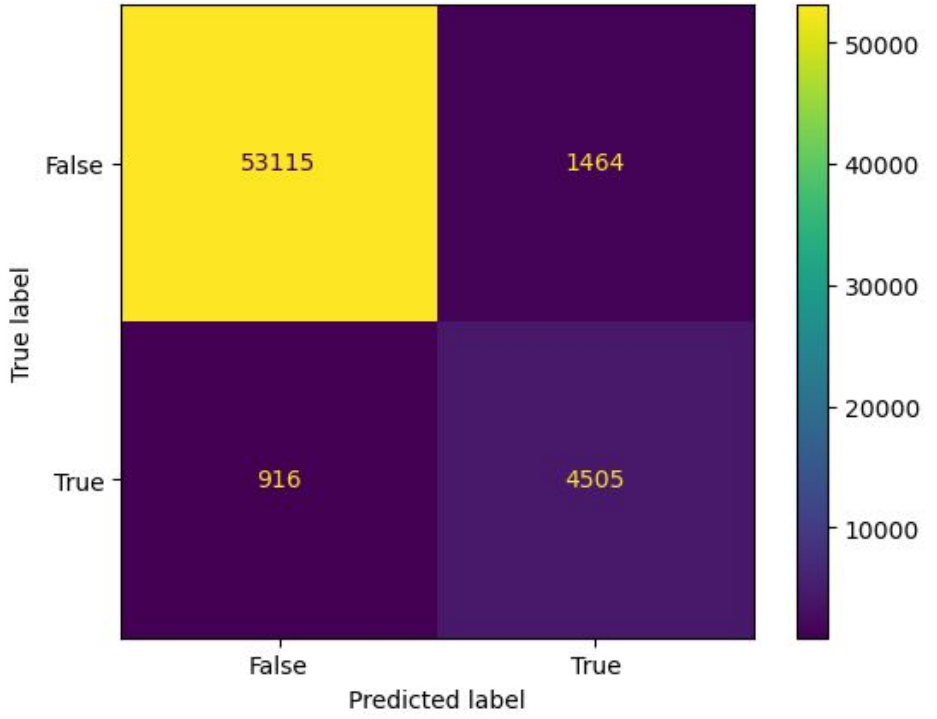
AVALIANDO MODELOS DE CLASSIFICAÇÃO

MATRIZ DE CONFUSÃO – métricas para além da acurácia

A escolha da(s) métrica(s) mais significativa(s) depende do problema, do balanceamento do conjunto em avaliação, do custo das avaliações incorretas

[1] **Acurácia**: o quanto o classificador acertou, seja da classe positiva, seja da classe negativa.
No caso do SPAM, quantos e-mails foram classificados corretamente?
Um modelo perfeito teria 0FN e 0FP com acurácia 100%
É contudo métrica bem geral, com suposição de balanceamento
Quando o dataset é desbalanceado ou um dos erros de FN/FP são mais graves (doenças por exemplo), melhor outras métricas



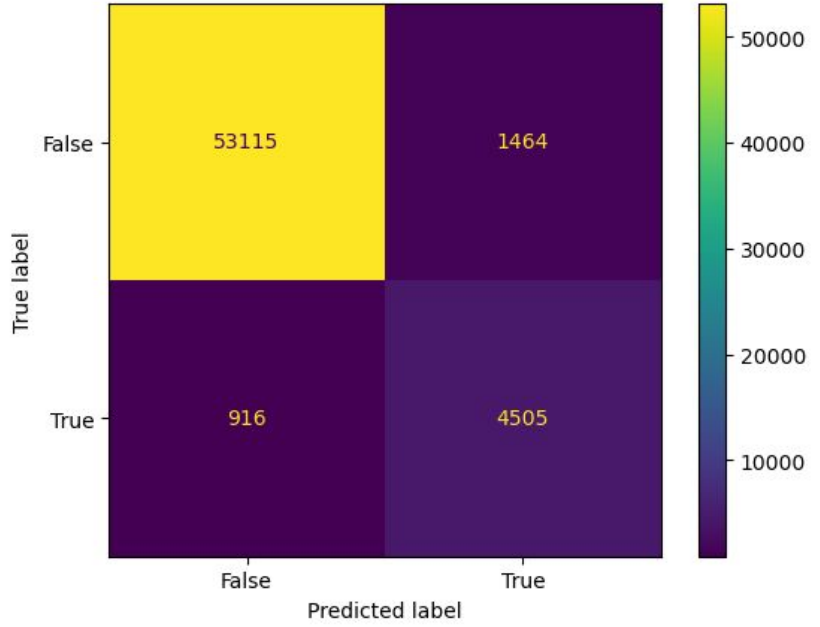


AVALIANDO MODELOS DE CLASSIFICAÇÃO

MATRIZ DE CONFUSÃO – métricas para além da acurácia

[2] **Acurácia das predições POSITIVAS (precisão/precision**): proporção de classificações positivas que são realmente positivas. No exemplo, proporção de imagens classificadas como 5 que realmente são imagens de 5s: P = Vp/(Vp+Fp)

Fração de e-mails **classificados**
como spam que eram realmente spam



MAXIMIZAR precisão: reduzir alarmes falsos, ou seja, falsos positivos

AVALIANDO MODELOS DE CLASSIFICAÇÃO

MATRIZ DE CONFUSÃO – métricas para além da acurácia

[3] **Revocação (recall, sensibilidade, taxa de verdadeiros positivos (TPR), taxa de acerto)**: Em geral usada junto com a precisão, pois é a proporção de instâncias positivas detectadas corretamente pelo classificador. FN são positivos reais classificados incorretamente (falhas): R = Vp/(Vp+Fn)

Mede a fração de spams que foram corretamente classificados como spam
Também chamada **probabilidade de detecção. Maximizar R significa Fn baixo, poucas falhas, o custo de falhas é relevante**

[4] **Taxa de falso positivo (TFP) ou taxa de queda**: proporção de todos negativos reais que foram classificados incorretamente como positivos, conhecida como **probabilidade de alarme falso**
TFP = 1 - Especificidade: Fp / (Fp+Vn)**,** uma TFP baixa significa Vn alto, ou seja, poucos alarmes falsos

AVALIANDO MODELOS DE CLASSIFICAÇÃO

MATRIZ DE CONFUSÃO – métricas para além da acurácia

[5] **Especificidade, seletividade ou Taxa de Verdadeiros Negativos**:
Avalia a capacidade do modelo detectar resultados negativos,
de modo oposto à sensibilidade: E = Vn/(Vn+Fp)

[6] **F1-score**: média harmônica entre precisão e revocação, dá mais importância a valores mais baixos e só será alta se a revocação e precisão forem altas. Bom para comparar classificadores. Para P e R semelhante, F1 semelhante. Senão F1 se aproxima da pior métrica.

F1 = 2. (P R) / (P + R)

A **média aritmética** dá mais peso aos valores grandes.

A **média harmônica** dá mais peso aos valores pequenos.
É por isso que o **F1-score** usa a média harmônica: se **Precisão** ou **Recall** for baixo, o F1 também fica baixo (não deixa o valor alto ser mascarado).

AVALIANDO MODELOS DE CLASSIFICAÇÃO

E como escolher?

Ocorre que há um trade-off entre precisão e revocação/sensibilidade. Aumentar um reduz o outro!

Exemplo 1: classificador para detectar videos seguros (**classe positiva**) para crianças

Objetivo: rejeitar muitos videos bons (baixa sensibilidade: Fn alto) e manter apenas os seguros (alta precisão: Fp baixo) OU uma revocação muito alta (Fn baixo), mas permita que alguns vídeos ruins sejam exibidos em seu produto (Fp alto)?

Métrica	Orientação
Acurácia	Indicador bem geral sobre a performance do modelo em bases balanceadas
Precisão	Use quando é muito importante que predições POSITIVAS sejam acuradas
Revocação (Taxa de Verdadeiro Positivo)	Use quando o custo de FALSOS NEGATIVOS seja maior que de FALSOS POSITIVOS
Taxa de Falso Positivo	Use quando o custo de FALSOS POSITIVOS seja maior que de FALSOS NEGATIVOS

AVALIANDO MODELOS DE CLASSIFICAÇÃO

E como escolher?

Ocorre que há um trade-off entre precisão e revocação/sensibilidade. Aumentar um reduz o outro!

Exemplo 2: detectar ladrões de lojas em imagens de vigilância
Objetivo: pode ser bom que o classificador tenha somente 30% de precisão, desde que tenha 99% de recall (pode ter poucos alarmes falsos, mas não admite falhas (falsos negativos))

EXERCÍCIOS

A) Seja uma matriz de confusão com 5VP, 6VN, 3FP e 2FN. Calcule a sensibilidade: (0.455, 0.714, 0.625) ?

B) Seja uma matriz de confusão com 3VP, 4VN, 2FP e 1FN. Calcule a precisão.

C) Você está desenvolvendo um classificador binário que verifica se determinado inseto nocivo está presente em fotos de armadilhas inteligentes (smart traps). Se o modelo detecta o inseto, o entomologista é notificado. **A detecção antecipada deste inseto é crítica para prevenir uma infestação.** Um alarme falso (falso positivo) é fácil de tratar: o entomologista vê a foto que foi classificada incorretamente e a marca como tal. Presumindo um nível aceitável de acurácia, para qual métrica este modelo deve ser otimizada?

1. Revocação
2. Precisão
3. Taxa de Falso Positivo

EXERCÍCIOS

1. Seja uma matriz de confusão com 5VP, 6VN, 3FP e 2FN. Calcule a sensibilidade (0.455, 0.714, 0.625)

6	3
2	5

2. Seja uma matriz de confusão com 3VP, 4VN, 2FP e 1FN. Calcule a precisão.

4	2
1	3

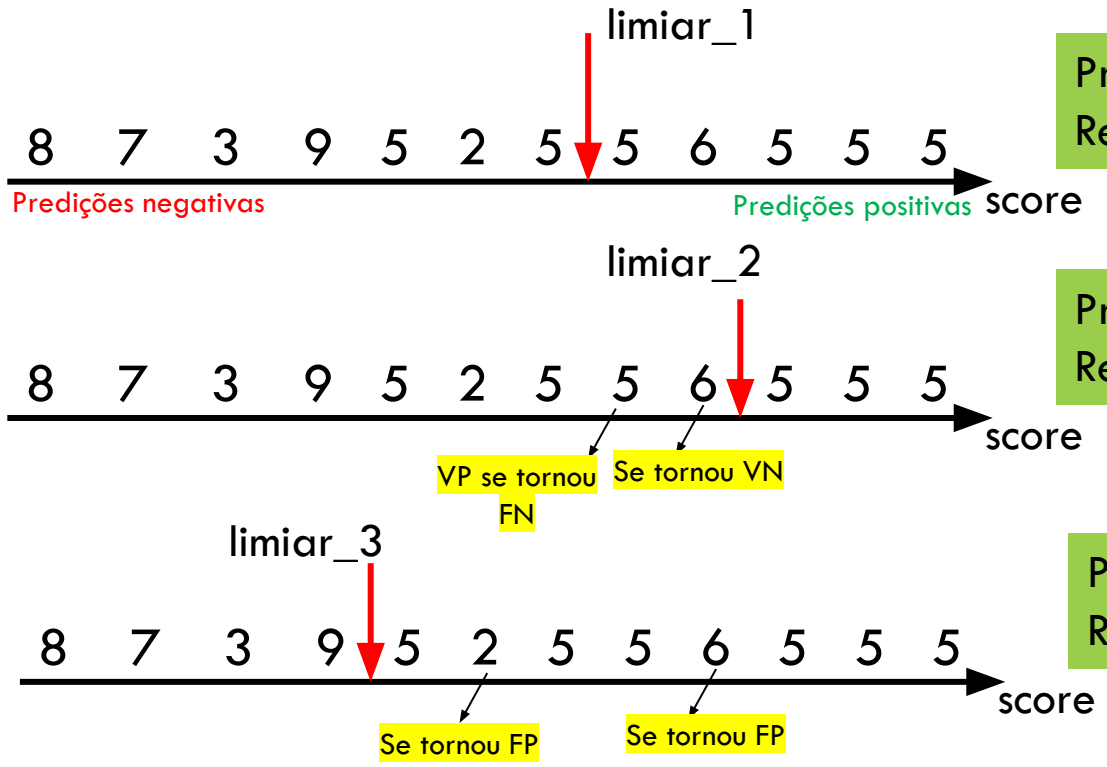
3. Você está desenvolvendo um classificador binário que verifica se determinado inseto nocivo está presente em fotos de armadilhas inteligentes (smart traps). Se o modelo detecta o inseto, o entomologista é notificado. A detecção antecipada deste inseto é crítica para prevenir uma infestação. Um alarme falso (falso positivo) é fácil de tratar: o entomologista vê a foto que foi classificada incorretamente e a marca como tal. Presumindo um nível aceitável de acurácia, para qual métrica este modelo deve ser otimizado?
    1. Revocação
    2. Precisão
    3. Taxa de Falso Positivo

Neste cenário, alarmes falsos (falsos positivos) tem custo baixo e falsos negativos tem custo muito alto (infestação). Faz sentido maximizar Revocação ou a probabilidade de detecção

4. Leia a documentação do classification_report do sklearn.metrics

A RELAÇÃO PRECISÃO/REVOCAÇÃO

[1] Como um algoritmo de classificação "decide"? Intuitivamente nos parece que é gerada alguma medida interna da relação entre as features (score) e, com base em algum limiar (threshold) a classificação é atribuída à classe X, Y, Z etc.
[2] O SGDClassifier, por exemplo, para cada instância calcula um score baseado em uma função de decisão e, se esse score for maior que um limiar, ele atribui a instância à classe positiva, ou então a atribui à classe negativa.



Precisão: 4/5 (4VP e 1FP) = 80%
Revocação: 4/6 (pois existem 6 5s reais) = 67%

Precisão: 3/3 (3VP e 0FP) = 100%
Revocação: 3/6 = 50%

Precisão: 6/8 (6VP e 2FP) = 75%
Revocação: 6/6 = 100%

A RELAÇÃO PRECISÃO/REVOCAÇÃO

Uma forma de verificar é usar o método decision_function() do modelo ao invés do predict(), a qual retorna um score e pode-se fazer a predição com base neste score, definindo-se um limiar. No caso do classificador de 5:

```
y_scores = model_sgd.decision_function([some_digit])
print(y_scores)

# definindo um threshold = 0 (padrão do SGDClassifier com predict())
thresh = 0

y_some_digit_pred = (y_scores > thresh)
print(y_some_digit_pred) # TRUE

# aumentando o threshold para 8000 dará False, diminuindo a revocação (o classificador perde esta imagem
thresh = 3000
y_some_digit_pred = (y_scores > thresh)
print(y_some_digit_pred) # FALSE
```

✓ 0.0s

```
2164.22030239]
 True]
False]
```

A RELAÇÃO PRECISÃO/REVOCAÇÃO

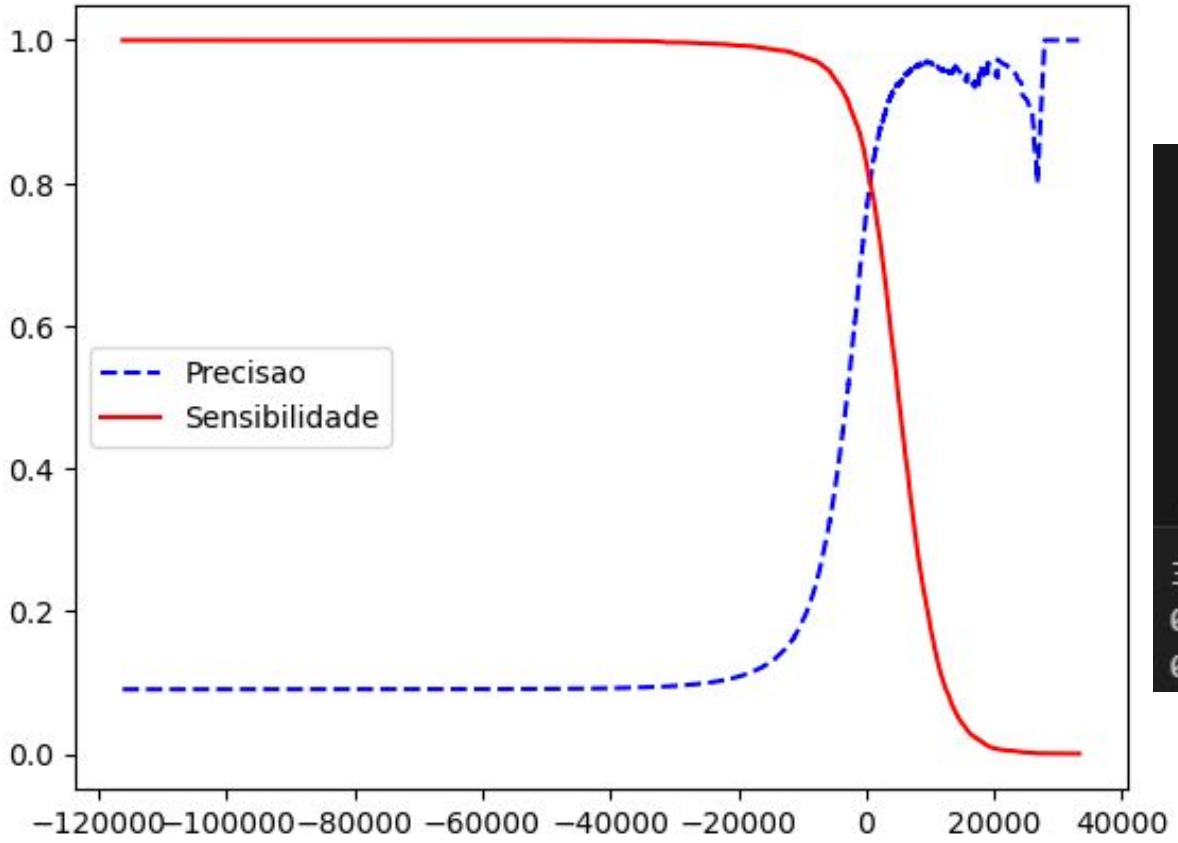
E qual limiar escolher? (para entendimento do pano de fundo...)

1. Usar cross_val_predict() para obter os scores de todas as instâncias de treinamento, com o método “decision_function”
2. Usar precision_recall_curve() para calcular precisão e revocação em todos os limiares possíveis
3. Plotar o gráfico P-R

A RELAÇÃO PRECISÃO/REVOCAÇÃO

```
1 y_scores = cross_val_predict(model_sgd, X_train, y_train_5, cv=5, method='decision_function')

  from sklearn.metrics import precision_recall_curve
2 precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)

  def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):
      plt.plot(thresholds, precisions[:-1], 'b--', label='Precisao')
      plt.plot(thresholds, recalls[:-1], 'r-', label='Sensibilidade')
      plt.legend()

3 plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
  plt.show()
```

A RELAÇÃO PRECISÃO/REVOCAÇÃO



Para o limiar que dá uma precisão de 90%, qual a revocação associada?

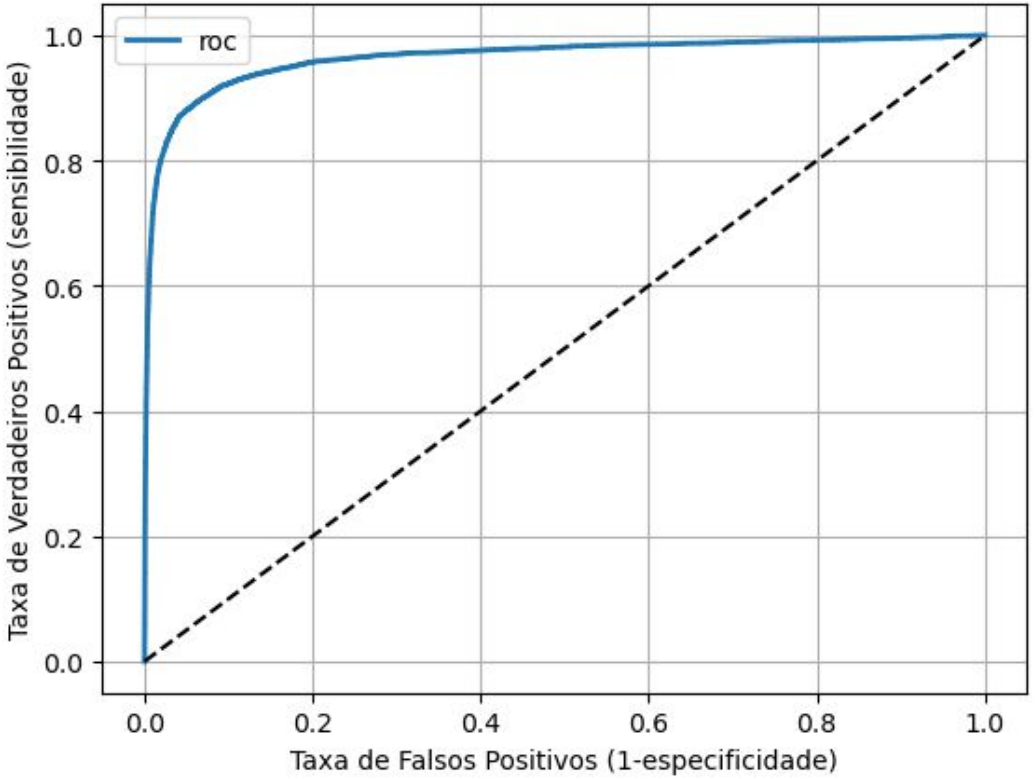
```
threshold_90_precision = thresholds[np.argmax(precisions >= 0.90)]
print(threshold_90_precision) #3045

y_train_pred_90 = (y_scores >= threshold_90_precision)
from sklearn.metrics import precision_score, recall_score
print(precision_score(y_train_5, y_train_pred_90)) #90%
print(recall_score(y_train_5, y_train_pred_90)) #65%
```

✓ 0.0s

```
3045.9258227053647
0.9002016129032258
0.6589190186312488
```

A CURVA ROC (CARACTERÍSTICA DE OPERAÇÃO)

- Muito usada para avaliar classificadores binários
- É a curva entre a Taxa de Verdadeiros Positivos X Taxa de Falsos Positivos (TFP, taxa de instâncias negativas classificadas como positivas)
- TFP = 1 – TVN ou 1 - especificidade
- A AUC-ROC não leva em conta a distribuição das classes
- Não há preocupação com limiares, com “custos” dos FP e FN




```
from sklearn.metrics import roc_curve
tfp, tvp, thresholds = roc_curve(y_train_5, y_scores)

def plot_roc_curve(tfp, tvp, label='roc'):
    plt.plot(tfp,tvp,linewidth=2,label=label)
    plt.plot([0,1],[0,1],'k--')
    plt.legend()
    plt.grid()
    plt.xlabel('Taxa de Falsos Positivos (1-especificidade)')
    plt.ylabel('Taxa de Verdadeiros Positivos (sensibilidade)')

plot_roc_curve(tfp, tvp)
plt.show()
```

A CURVA ROC (CARACTERÍSTICA DE OPERAÇÃO)

- A área sob a curva (AUC) é uma forma de comparar classificadores: (quanto mais próximo de 1, melhor)
- Problema: quando a classe negativa é majoritária e há pouco valor em predições verdadeiras negativas (Vn). Neste caso pode oferecer medida muito otimista do desempenho do modelo

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_train_5, y_scores)
```

✓ 0.0s

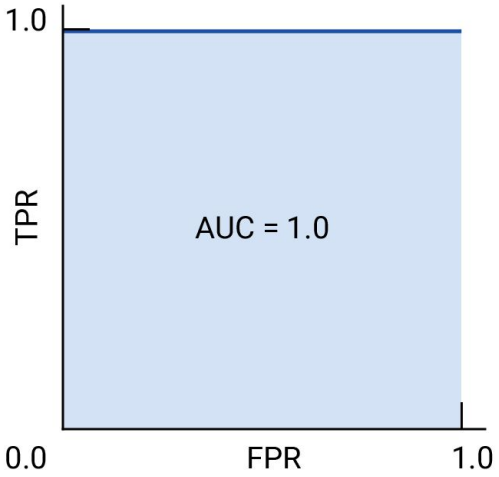
```
0.9648211175804801
```

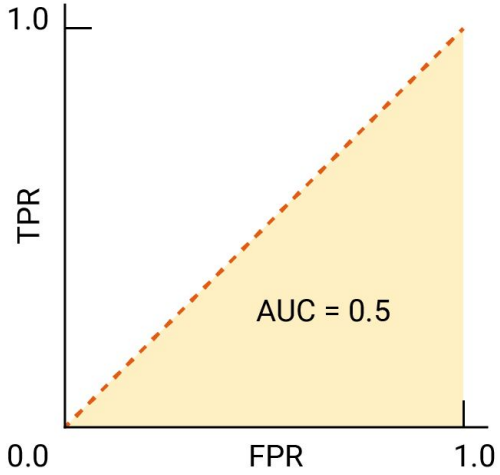
Exemplo: dataset com 10 positivos e 100000 negativos (relação 1/10000)
Seja o Modelo A que classifica 900 positivos, sendo 9 verdadeiros positivos: Pa = 9/900 = 0.01 e o Modelo B que classifica 90 positivos, sendo 9 verdadeiros positivos: Pb = 9/90 = 0.1. B portanto é mais “preciso”. Agora vamos olhar as métricas ROC:
TVPa = 9/10=0.1, TFPa=(900-9)/100000 = 0.00891
TVPb = 9/10=0.1, TFPb=(90-9)/100000 = 0.00081

TVPa=TVPb, mas como o número de negativos é bem maior que positivos, a diferença TFPa-TFPb=0.0081 é perdida (arredondada para 0). Uma grande mudança no número de FP resultou em mínima mudança no TFP, escondendo quanto melhor é o modelo B no trato com falso positivos. No caso de detecção de fraudes, não há interesse em transações corretas!

A CURVA ROC (CARACTERÍSTICA DE OPERAÇÃO)

- Em (0,1) temos FPR (TFP=0, ou seja, 1-especificidade=0 implica especificidade=1). Mas lembre que especificidade é a taxa de verdadeiros negativos. Se ela é igual a 1, significa que não há falsos positivos.
- Em (TPR, TVP=1=sensibilidade) não há falsos negativos.
- **Logo é o melhor cenário pontos próximos a (0,1) pois estão os limiares que geram os melhores desempenhos**



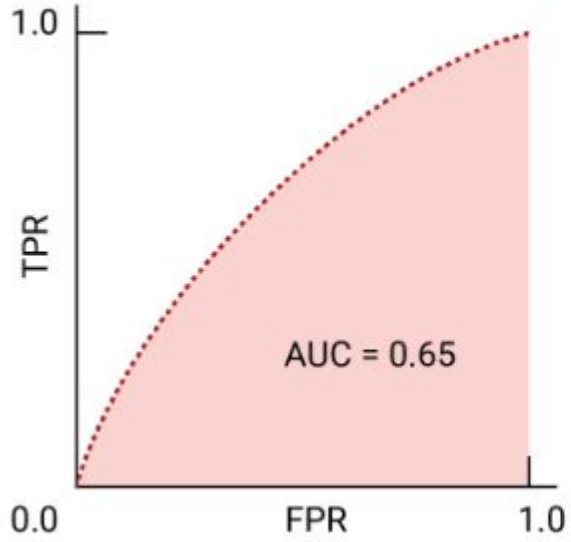


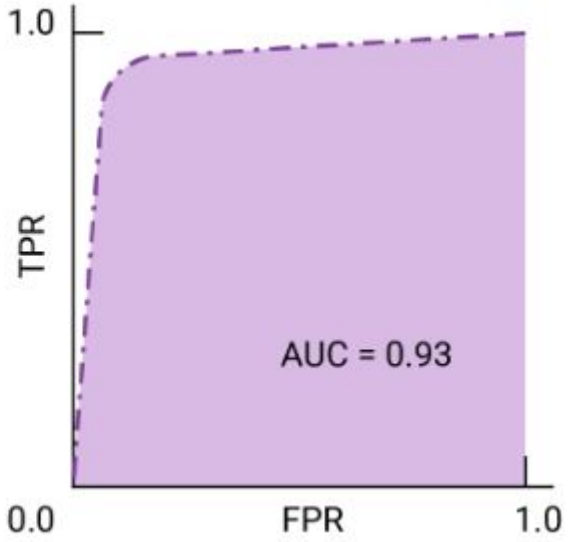
Já neste caso, um AUC de 0.5 equivale a predições aleatórias (como cara-coroa etc.), ou seja, 50% de probabilidade de ser positivo e 50% de ser negativo (1 pra 1)

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

A CURVA ROC (CARACTERÍSTICA DE OPERAÇÃO)

- 0.93>0.65, logo modelo à direita tem melhor desempenho, maior probabilidade de classificar uma amostra aleatória como classe positiva

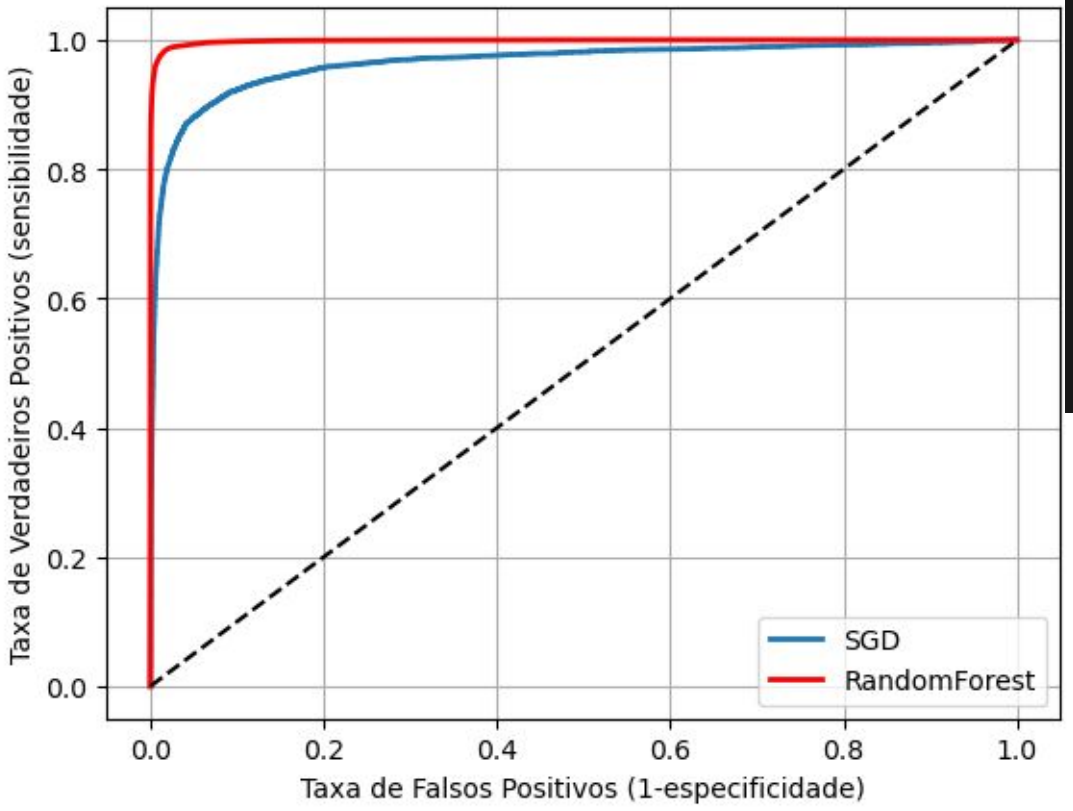




A CURVA ROC (CARACTERÍSTICA DE OPERAÇÃO)

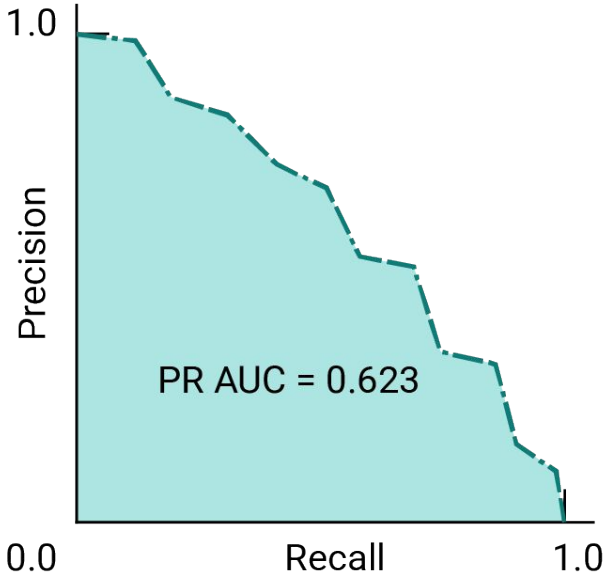
- Comparando SGD X RandomForest para o classificador binário de 5s no MNIST

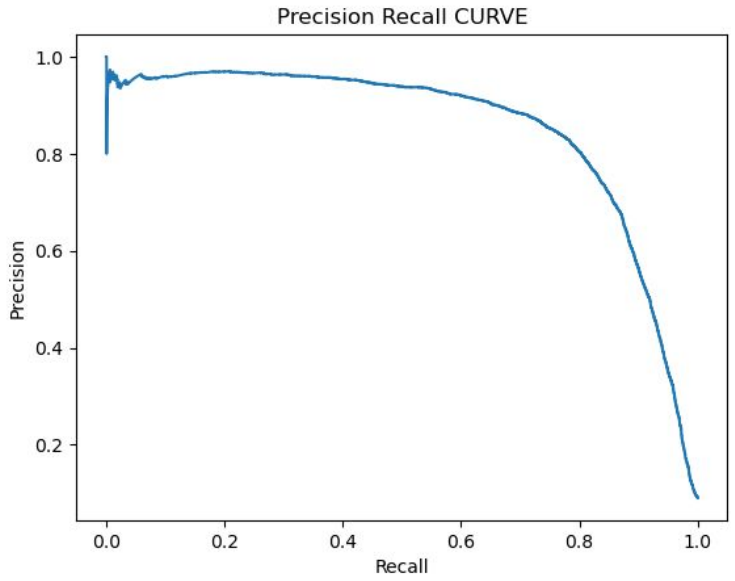
```python
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(random_state=42)

y_probas_rf = cross_val_predict(model_rf, X_train, y_train_5, cv=5, method='predict_proba')

y_scores_rf = y_probas_rf[:,1] # ou seja as probabilidades da classe positiva (5)
tfp_rf, tvp_rf, thresholds_rf = roc_curve(y_train_5, y_scores_rf)

def plot_roc_curve(tfp_1, tvp_1, tfp_2, tvp_2, label_1, label_2):
    plt.plot(tfp_1,tvp_1, linewidth=2, label=label_1)
    plt.plot(tfp_2,tvp_2, 'r', linewidth=2, label=label_2)
    plt.plot([0,1],[0,1],'k--')
    plt.legend()
    plt.grid()
    plt.xlabel('Taxa de Falsos Positivos (1-especificidade)')
    plt.ylabel('Taxa de Verdadeiros Positivos (sensibilidade)')

plot_roc_curve(tfp, tvp, tfp_rf, tvp_rf, 'SGD', 'RandomForest')
plt.show()
```



ROC AUC_RF = 0.9984

A RELAÇÃO PRECISÃO/REVOCAÇÃO (CURVA PR)



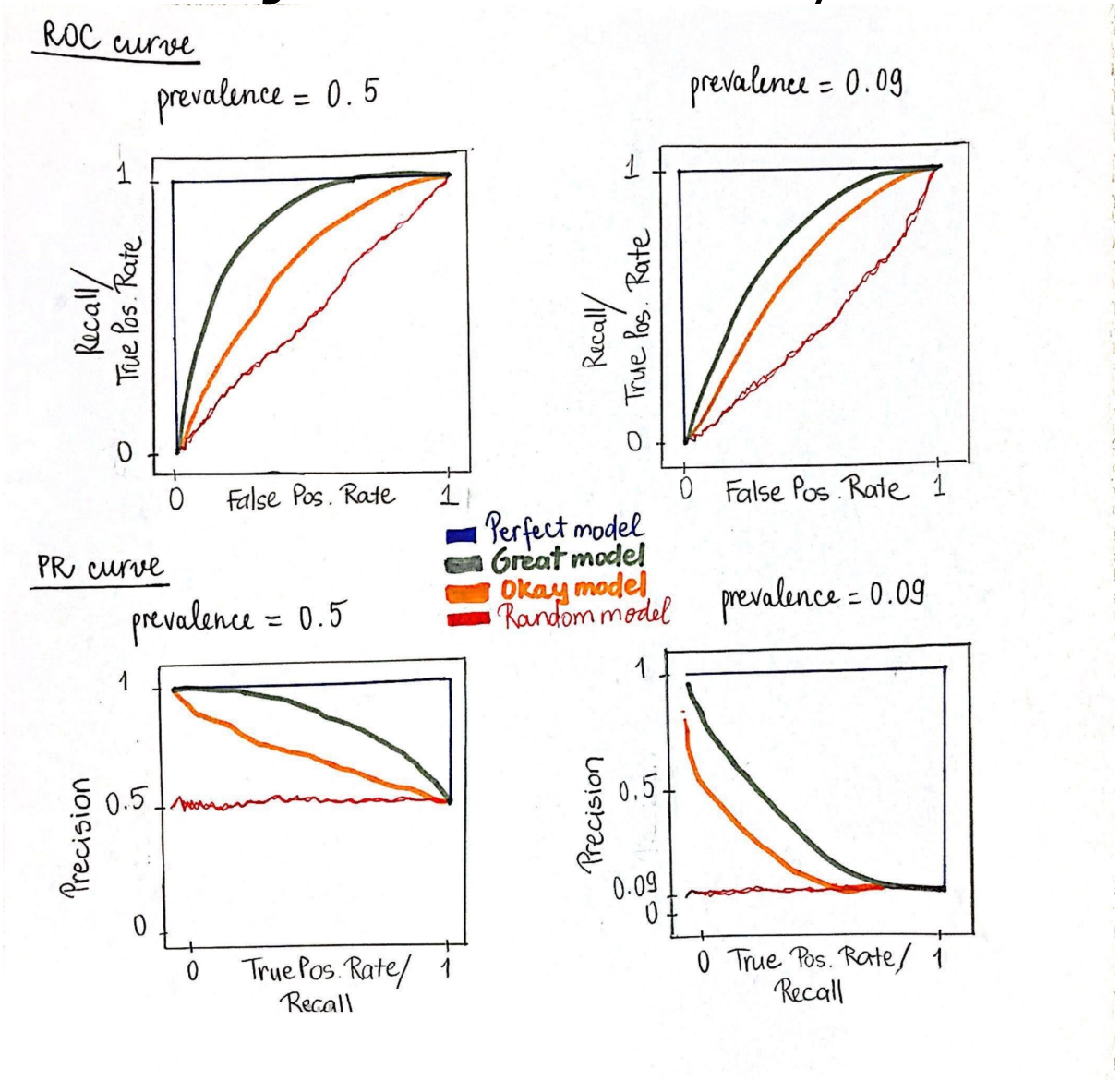


- A curva PR e a respectiva área sobre esta curva (AUCPR) é mais indicada para bases desbalanceadas onde o maior interesse está na classe positiva (casos raros etc.) - é mais informativo!
- Como Precisão e Recall não são afetados por verdadeiros negativos, não é afetado pelo desbalanceamento

A RELAÇÃO PRECISÃO/REVOCAÇÃO (CURVA PR)

- Modelo A: recall = TVP = 0.9 e precisão = 9/900 = 0.01
  Modelo B: recall = TVP = 0.9 e precisão = 9/90 = 0.1
- Na AUC-ROC a base (prevalência) é fixa em 0.5. Na AUC-PR, a área é calculada em relação à base variável da prevalência da classe positiva: P/(P+N). Por exemplo, uma base de 0,5 significa que há balanceamento. Já uma base de 0,09 significa uma relação de 1 P para 10 N. **É importante que a base de teste reflita a distribuição de dados da população real. A AUC-PR é sensível à relação P/N**

A RELAÇÃO PRECISÃO/REVOCAÇÃO (CURVA PR)



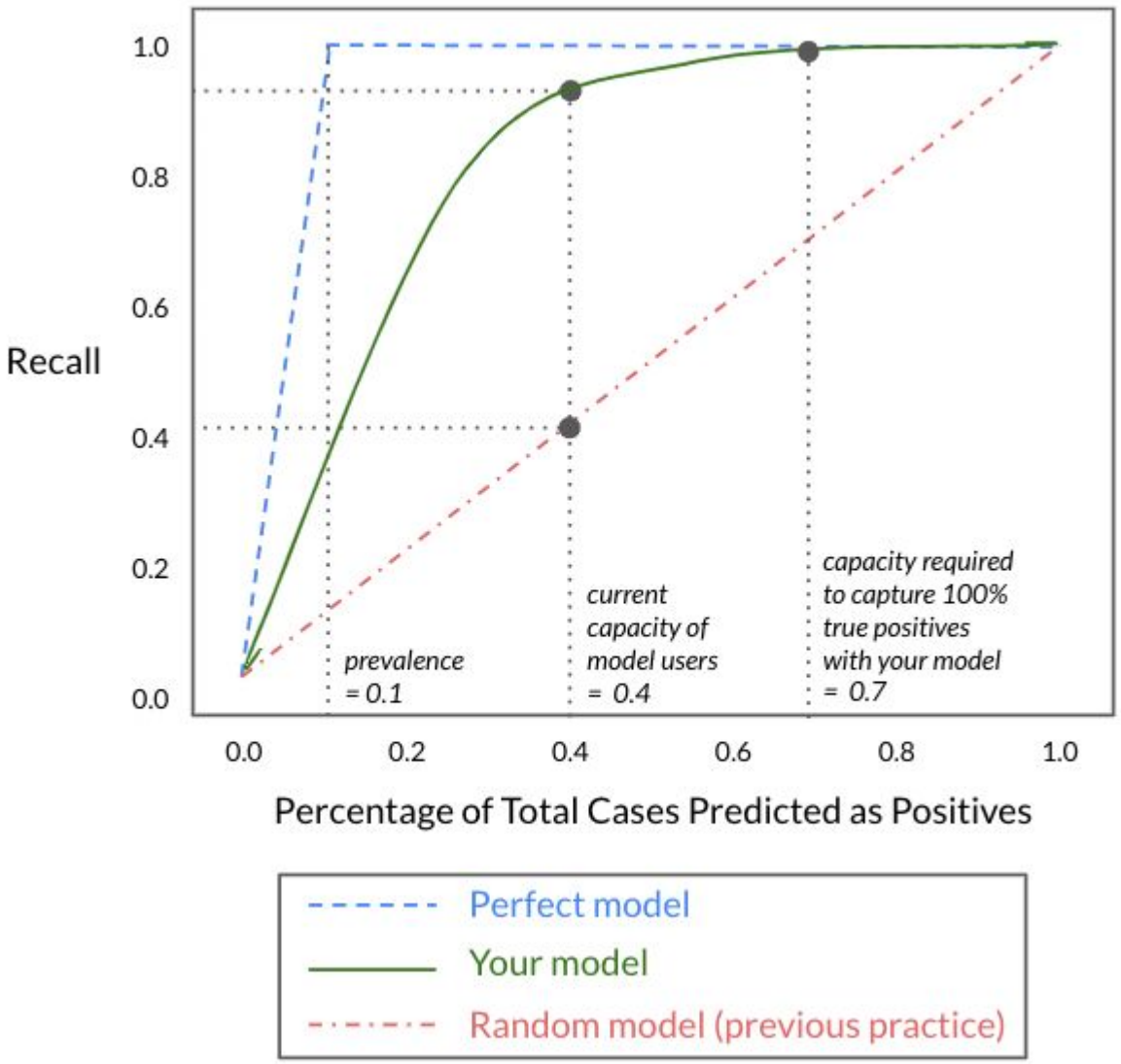
Métricas baseadas em limiar x baseadas em ranking

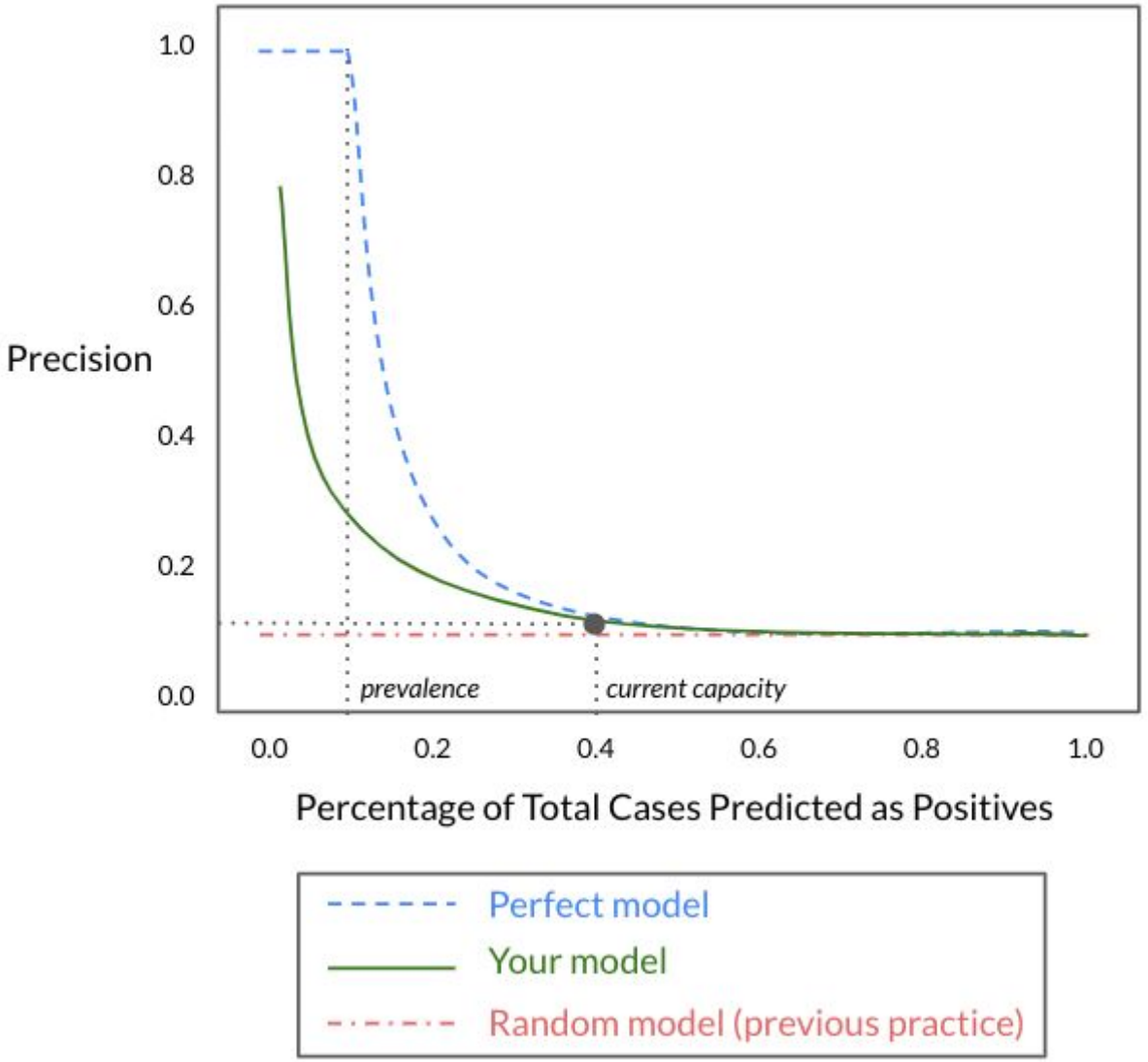
	Threshold metrics (e.g. accuracy, precision, recall)	Ranking metrics (e.g. ROC curve, PR curve)
Pros	Provide means to compare models against each other, as well as with perfect and random classifiers	
	Intuitive to non-technical audience	No assumption about relative cost of two types of errors
Cons	Require knowledge of outcome distribution and quantifiable costs of two types of errors	Don't provide any actionable insights

Curvas de performance X prevalência

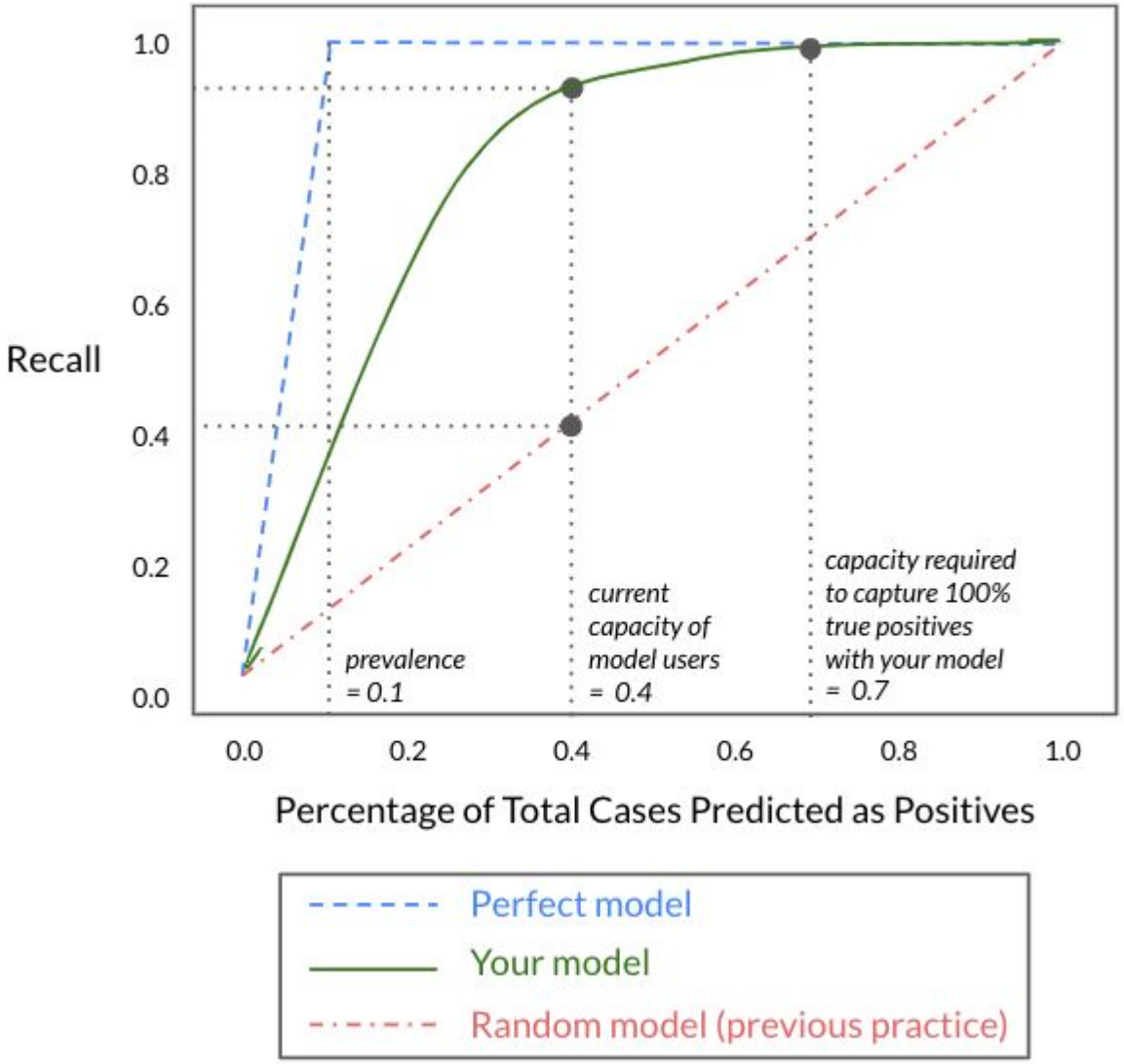
- Cenário: empresa de seguros, com o objetivo de detectar fraudes
- A taxa de prevalência atual é de 0.1 (pode-se imaginar uma relação 1:9, 1/1+9 = VP/(VP+FP))
- A cada semana entram 25 novos casos. O auditor seleciona aleatoriamente 10 casos pois é sua capacidade de trabalho, ou seja, 10/25=0.4 (40%) a capacidade de determinar VP
- Você desenvolveu um modelo de ML para ranquear o top-10 casos e agora o auditor seleciona os casos que seu modelo indica. Como determinar que seu modelo melhora a prática atual? Se melhora, quanto melhora? Você então plota o gráfico de performance de sensibilidade X prevalências e o gráfico de performance de precisão X prevalências
- A equipe deseja detectar os casos fraudulentos o mais rápido possível. Como provar que seu modelo alcança este objetivo de forma mais eficiente e quantificar esta eficiência?

Curvas de performance X prevalência



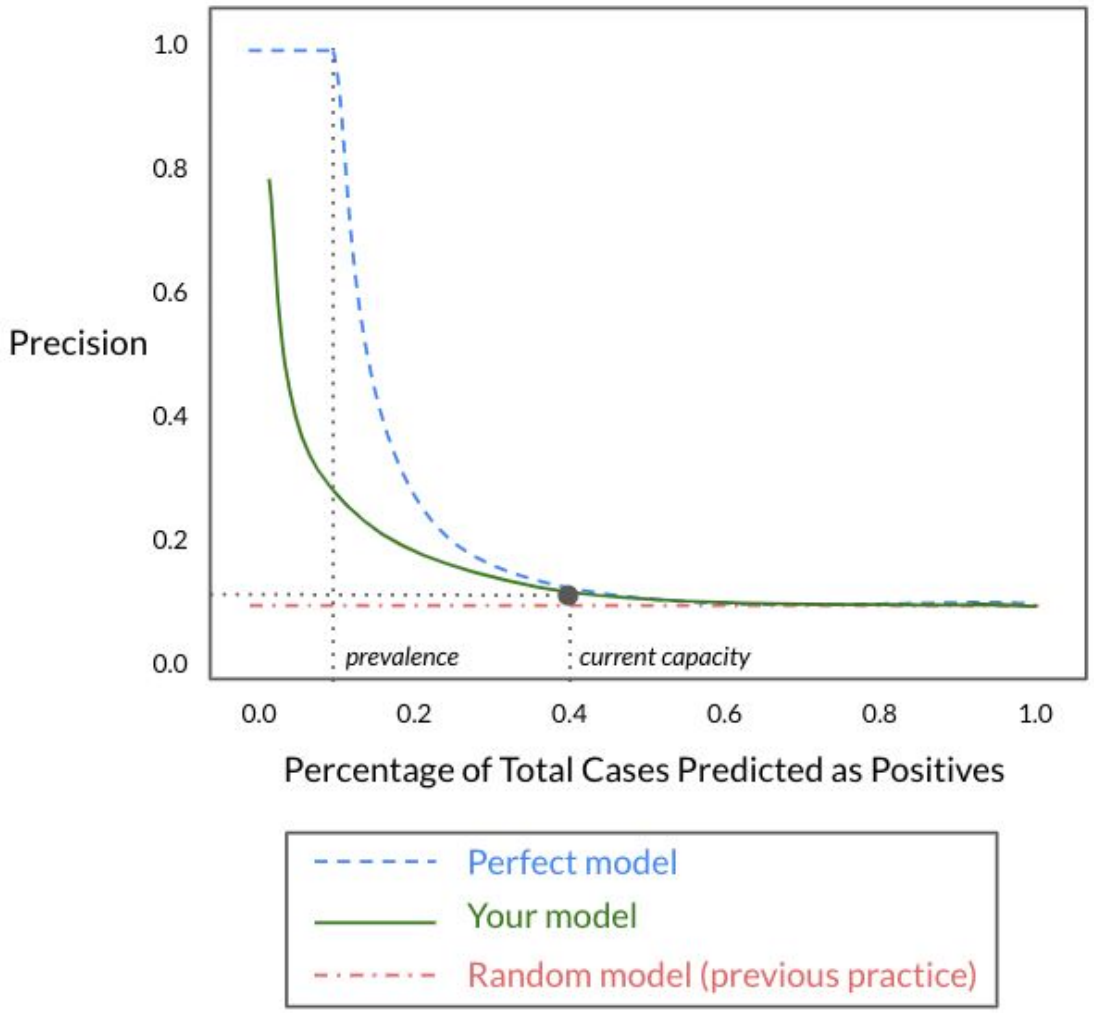


Curvas de performance X prevalência



- Seu modelo, para prevalência de 0.4, captura ~95% de VP enquanto atualmente é ~41%, e isto sem comprometer precisão, pois atualmente é 10% e seu modelo entrega ~12%
- Para obter 100% de VP, o modelo atual exige 100% de casos auditados. Já seu modelo atinge 100% de VP com 70% de casos ranqueados auditados, melhorando 30% o processo

Curvas de performance X prevalência



https://github.com/josenalde/machinelearning/blob/main/src/prcurve_prevalence.ipynb

APERFEIÇOANDO O MODELO APÓS SELEÇÃO

- A escolha manual de hiperparâmetros é tediosa (encontrar uma combinação)
- No Scikit-learn pode-se usar o GridSearchCV ou o RandomizedSearchCV
- No GridSearch ele avalia TODAS as combinações por meio de validação cruzada
- Exemplo para o RandomForest

```
from sklearn.model_selection import GridSearchCV
hyperparam_grid = [{'n_estimators':[3,10,30],
                    'max_features':[2,4,6,8]},
                   {'bootstrap':[False], 'n_estimators':[3,10], 'max_features':[2,3,4]},]
model_rf = RandomForestRegressor()
model_rf_gs = GridSearchCV(model_rf, hyperparam_grid, cv=5,scoring='f1',return_train_score=True)
model_rf_gs.fit(X,y)
```

Testa no primeiro dict 3x4=12 combinações com o parâmetro bootstrap:True (default)
Depois testa no segundo dict 2x3=6 combinações com bootstrap:False
Testa portanto 12+6 combinações e cada rodada de 5 treinos com validação cruzada, ou seja, 18 x 5 = 90 rodadas de treinamento. Ao fim, os melhores parâmetros estão aqui:

APERFEIÇOANDO O MODELO APÓS SELEÇÃO

```
# se os melhores resultados forem com os limites superiores, pense em
aumentá-los

model_rf_gs.best_params_
model_rf_gs.best_estimator_ #(melhor estimador, aplicar no teste, exportar)
cvres = model_rf_gs.cv_results_
for mean_f1, params in zip(cvres['f1'], cvres['params']):
    print(mean_f1, params)
```

APERFEIÇOANDO O MODELO APÓS SELEÇÃO

- Outro exemplo com KNN

```
from sklearn.model_selection import GridSearchCV

param_grid = {'n_neighbors': range(1,40,2), 'weights': ['uniform', 'distance'], 'p': [1, 2, 3]}
grid = GridSearchCV(KNeighborsClassifier(),param_grid, verbose = 3)#verbose indica a quantidade de detalhame
grid.fit(X_train,y_train)
```

APERFEIÇOANDO O MODELO APÓS SELEÇÃO

- Se o espaço de pesquisa for grande é melhor usar busca randomizada
- Avalia determinado número de combinações aleatórias, selecionando um valor aleatório para cada hiperparâmetro por iteração (1000 iterações no Randomized explorar mais do que no GridSearch) e pode-se definir o número de iterações desejado para controlar melhor o custo computacional

```
from sklearn.model_selection import RandomizedSearchCV
hyperparam_grid = [{'n_estimators':[3,10,30],
                    'max_features':[2,4,6,8]},
                   {'bootstrap':[False], 'n_estimators':[3,10],
'max_features':[2,3,4]},]
model_rf = RandomForestRegressor()
model_rf_gs = RandomizedSearchCV(model_rf, hyperparam_grid, n_iter=10,
scoring='f1', cv=5, return_train_score=True)
model_rf_gs.fit(X,y)
```

https://optuna.org/ e muitos outros sintonizadores
Hyperopt, Hyperas, Talos, Kopt, Keras Tuner,
Scikit-Optimize (skopt), Spearmint, Hyperband,
Sklearn-Deap, Arimo, SigOpt, Oscar etc.