Métricas de Avaliação

1. Introdução

Desenvolver modelos de *Data Science* e *Machine Learning* por si só não agregam valor se não houver alguma forma de quantificar a qualidade dos resultados. Dessa forma, as **métricas de avaliação** funcionam como ferramentas que auxiliam na avaliação da qualidade do modelo e o possível erro cometido durante as predições.

Existem métricas de avaliação específicas para cada tipo de modelo, seja de classificação ou regressão, onde serão detalhadas nos tópicos a seguir.

2. Métricas de Classificação

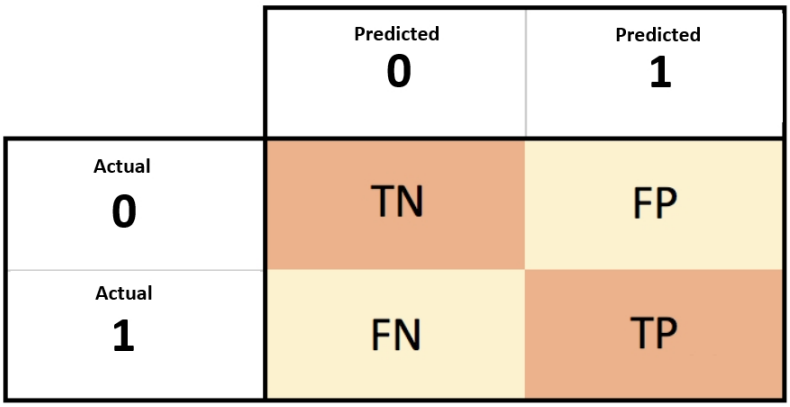
O desenvolvimento das métricas de avaliação para modelos de classificação serão descritas para o caso de classificações binárias, ou seja, entre duas classes, mas todas as métricas podem ser estendidas para o caso de modelos de multiclasses (3 ou mais classes). O objetivo de um modelo de classificação é decidir se determinada observação pertence a uma das duas classes, podendo essas classes serem denominadas como positivas (P) ou negativas (N) no caso das classes indicarem a possibilidade de ocorrência de uma determinado evento.

Um exemplo seria classificar se um determinado cliente pode vir a dever no cartão de crédito, portanto se ele será inadimplente (positivo) ou não (negativo). A forma de se avaliar o desempenho de um modelo de classificação é feita a partir de uma comparação pelos resultados preditos pelo o modelo em comparação às classes verdadeiras daqueles respectivos dados, avaliando o quão distante o modelo possa estar de um resultado real.

Uma maneira fácil e intuitiva de entender e construir as principais métricas de classificação seria utilizando como referência a **matriz de confusão**.

2.1 Matriz de Confusão

A matriz de confusão é uma forma tabular de comparação entre os valores preditos pelo modelo em comparação aos valores reais, permitindo visualizar rapidamente a quantidade de elementos classificados corretamente e erroneamente em cada classe:



fonte: Static Packt Cdn

Cada um dos elementos da matriz de confusão são definidos de acordo com o comparativo dos valores preditos com os reais, sendo definidos nos itens abaixo:

- **Verdadeiros Positivos (VP)**: classificação correta da classe positiva;
- **Verdadeiros Negativos (VN)**: classificação correta da classe negativa;
- **Falsos Positivos (FP, erro tipo I)**: classficação como da classe positiva pelo modelo, onde na verdade seria da classe negativa;
- **Falsos Negativos (FN, erro tipo II)**: classficação como da classe negativa pelo modelo, onde na verdade seria da classe positiva.

Definida as componentes da matriz de confusão, o próximo passo será teorizar as principais métricas de avaliação a partir destes componentes.

Existem algumas formas de implementar tanto o cálculo como também a ilustração da matriz de confusão em *Python*, conforme o exemplo em código a seguir:

```
# Instalar a bibliote mlxtend
! pip install mlxtend

# Bibliotecas para o calculo e gráfico da matriz de confusão
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix

# Cria a matriz de confusão
cm = confusion_matrix(y_test, y_pred)

# Gera a ilustração da Matriz de Confusão
plot_confusion_matrix(conf_mat = cm)
```

```
# Mostra o gráfico
plt.show()
```

2.2 Acurácia

A acurácia (*accuracy* em inglês) indica o percentual de classificações corretas, independente se pertencer a classe positiva (P) ou negativa (N). Por exemplo, dado um conjunto de de 1000 observações e 700 destas foram classificadas corretamente, o valor da acurácia para este modelo seria de 70%. A fórmula para o cálculo da acurácia a partir das componentes da matriz de confusão é definida por:

$$Acurácia = \frac{TP + TN}{TP + TN + FP + FN}$$

A acurácia é uma métrica bem simples e direta, mas que requer bastante cautela a ser utilizada, pois pode ser facilmente influenciada pelo balanceamento das classes, ou seja, quantos elementos têm na classe positiva (P) e quantos têm na classe negativa (N). Por exemplo, dado que têm 1000 observações onde 900 destas são positivas e 100 negativas, se o modelo por qualquer motivo classificar que todas as 1000 observações sejam positivas porque erre todas as observações negativas, ainda assim a acurácia para este caso seria de 90%.

A implementação em *Python* utilizando da função para acurácia do *Scikit-Learn* está indicada no código a seguir:

```
# biblioteca para calcular a acurácia
from sklearn.metrics import accuracy_score

# Cálculo da acurácia
print("Valor da Acurácia: ", accuracy_score(y_test, y_pred))
```

2.3 Precisão

A precisão (*precision* em inglês) é definida pela razão entre a quantidade de exemplos classificados corretamente como positivos e o total de observações classificadas como positivas, conforme a fórmula abaixo:

$$Precision = \frac{TP}{TP + FP}$$

A precisão tem como objetivo avaliar o **erro tipo I**, ou seja, a quantidade de **Falso Positivos** (FP) cometidos pelo modelo.

A implementação em *Python* utilizando da função para acurácia do *Scikit-Learn* está indicada no código a seguir:

```
# biblioteca para calcular a precisão
from sklearn.metrics import precision_score

# Cálculo da precisão
print("Valor da Precisão: ", precision_score(y_test, y_pred))
```

2.4 Revocação

A revocação (*recall* em inglês), normalmente também chamada como sensibilidade, é a métrica que avalia a relação de todos as observações classificadas corretamente pelo modelo como positivas e todas as observações que realmente são positivas, conforme a fórmula abaixo:

$$Recall = \frac{TP}{TP + FN}$$

A revocação tem como objetivo avaliar o **erro tipo II**, ou seja a quantidade de **Falso Negativos** (FN) cometidos pelo modelo.

A implementação em *Python* utilizando da função para revocação do *Scikit-Learn* está indicada no código a seguir:

```
# biblioteca para calcular a revocação
from sklearn.metrics import recall_score

# Cálculo da revocação
print("Valor do Recall: ", recall_score(y_test, y_pred))
```

2.5 F1-Score

O *F1-Score* é uma métrica que leva em consideração a **precisão** e a **revocação** e a sua fórmula é definida pela **média harmônica** entre estas duas outras métricas conforme indicado a seguir:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

A principal característica da média harmônica é ser bem sensível a alteração de qualquer uma das métricas, por exemplo se a precisão ou revocação for muito baixo ou próximo de zero, o valor para o *F1-Score* também será. Ou seja, um modelo que apresenta um *F1-score* bom é um modelo capaz de equilibrar uma precisão e revocação alta. Portanto, esta métrica tende a ser um bom indicativo sobre a qualidade do modelo.

A implementação em *Python* utilizando da função para *F1-Score* do *Scikit-Learn* está indicada no código a seguir:

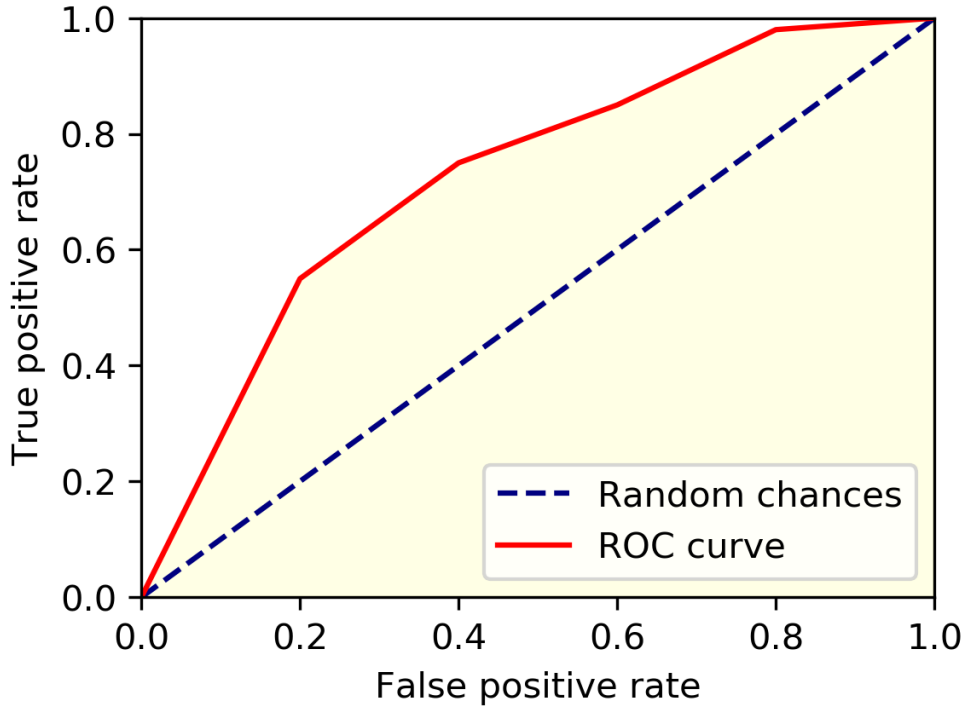
```
# biblioteca para calcular a F1
from sklearn.metrics import f1_score

# Cálculo do F1
print("Valor do F1 Score: ", f1_score(y_test, y_pred))
```

2.6 Curva ROC-AUC

A **curva ROC** (*Receiver Operating Chracteristics*) é uma curva de probabilidade do modelo, onde a área embaixo da curva é denominada como **AUC** (*Area under the Curve*). O valor do AUC representa o **grau de separabilidade atingido pelo modelo**, ou seja está medindo a capacidade do modelo em distinguir entre duas classes.

A curva ROC é construída com a **taxa de falsos positivos** no eixo x, e a **taxa de verdadeiros positivos** no eixo y, para diferentes **thresholds de classificação**, ou seja diferentes limiares de corte na separação das classes:



Fonte Medium

Os valores de AUC variam entre 0 e 1, onde quanto maior o valor do AUC, melhor e a capacidade do modelo em classificar as observações. Um modelo simples que indique aleatoriamente qual classe cada observação pertence vai ter um valor de AUC igual a 0.5, Ou seja, para que um modelo de classificação esteja acertando mais predições do que errando, o valor de AUX precisa ser maior que 0.5.

A implementação para o cálculo do AUC e o cálculo dos parâmetros para construir uma curva ROC utilizando *Python*, estão indicadas no código abaixo:

```
# Carrega as funções para a ROC e o AUC do Scikit-Learn
from sklearn.metrics import roc_auc_score, roc_curve

# Taxa de Falsos positivos, Verdadeiros Positivos e Thresholds para a
construção da curva ROC
fpr, tpr, thresholds = metrics.roc_curve(y_test, model.predict_proba(X_test)
[:,1])

# Cálculo do AUC
auc = metrics.roc_auc_score(y_test, model.predict(X_test))
```

3. Métricas de Regressão

Para os modelos de regressão, existem métricas para **avaliação dos erros** e métricas para a **determinação do ajuste da regressão aos dados**. As métricas de avaliação dos erros conseguem levantar o quanto de erro está sendo cometido pelo modelo nas predições e comumente utilizadas para **comparar regressões diferentes**, ou seja avaliando entre duas ou mais regressões qual delas estão cometendo menos erros. As principais métricas de avaliação nas Regressões são:

- **Erro Quadrático Médio** (*Mean Squared Error* ou MSE): Calculado pela a soma das diferenças entre o valor predito e as observações, isto elevado ao quadrado:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

A implementação em *Python* utilizando da função do *Scikit-Learn* está indicada no código a seguir:

```
# biblioteca para calcular a MSE
from sklearn.metrics import mean_squared_error

# Cálculo do MSE
print("Valor do MSE: ", mean_squared_error(y_test, y_pred))
```

- **Raiz Quadrada da Média dos Erros Quadráticos** (*Root Mean Squared Error* ou RMSE): Cálculo de forma análoga ao MSE, mas tirando a raiz quadrada da soma resultante das diferenças:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

A implementação em *Python* utilizando da função do *Scikit-Learn* está indicada no código a seguir:

```
# biblioteca para calcular a MSE
from sklearn.metrics import mean_squared_error

# Cálculo do RMSE - Apenas aplica a raiz no MSE
print("Valor do RMSE: ", mean_squared_error(y_test, y_pred, squared = False))
```

- **Erro Absoluto Médio** (*Mean Absolute Error* ou MAE): Calculado pela soma da diferença absoluta entre o valor predito e as observações:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

A implementação em *Python* utilizando da função para *Erro Absoluto Médio* do *Scikit-Learn* está indicada no código a seguir:

```
# biblioteca para calcular a MAE
from sklearn.metrics import mean_absolute_error

# Cálculo do MAE
print("Valor do MAE: ", mean_absolute_error(y_test, y_pred))
```

No caso da **métrica de determinação**, utiliza-se um importante coeficiente chamado R^{2}, que mede a proporção da variabilidade em Y que pode ser explicada a partir de X:

$$ R^2 = 1-\frac{\sum_{i=1}^n (y_i-\hat{y}*i)^2}{\sum*{i=1}^n(y_i-\bar{y})^2}, \quad 0\leq R^2\leq 1 $$

Por exemplo, se para uma regressão qualquer o valor de R^{2} é de 80%, significa que esta regressão consegue representar 80% deste conjunto de dados.

A implementação em *Python* utilizando da função para R^{2} do *Scikit-Learn* está indicada no código a seguir:

```
# biblioteca para calcular o R2
from sklearn.metrics import r2_score

# Cálculo do R2
print("Valor do R2: ", r2_score(y_test, y_pred))
```

Materiais Complementares

Documentação no Scikit-Learn sobre o metrics;

Artigo publicado pela KUNUMI no *Medium* - *Métricas de Avaliação em Machine Learning: Classificação*.;

Referências

James, Gareth, et al. An Introduction to Statistical Learning: With Applications in R. Alemanha, Springer New York, 2013;

Bruce A., Bruce P. Estatística Prática para Cientistas de Dados. Segunda Edição, Alta books, 2019;

Tópico anterior

Próximo Tópico