9 - 4 | 2021

# O meta-dataset AmTriangle para experiências com Supervised Machine Learning

***The AmTriangle meta-dataset for experimenting with Supervised Machine Learning***

***El metaconjunto de datos AmTriangle para experimentar con el aprendizaje automático supervisado***

**Artur Marques**

**Electronic version**
URL: https://revistas.rcaap.pt/uiips/ ISSN: 2182-9608

**Publisher**
Revista UI_IPSantarém

**Printed version**
Date of publication: 31st December 2021 Number of pages: 11
ISSN: 2182-9608

**Electronic reference**

Marques, A. (2021). *O meta-dataset AmTriangle para experiências com Supervised Machine Learning.* Revista da UI_IPSantarém. *Edição Temática: Ciências Exatas e das Engenharias*. Número especial: Conferência Internacional Cooperação Internacional, multiculturalidade, trabalho colaborativo e ambientes mais inclusivos, sustentáveis e resilientes. 9(4), 58-68. https://revistas.rcaap.pt/uiips/

**revistauiips@ipsantarem.pt**
Complexo Andaluz,
Apartado 279
2001-904 Santarém

*Revista da UI_IPSantarém – Unidade de Investigação do Instituto Politécnico de Santarém,*
Vol. 9, N. º 4, 2021, pp. 58-68, Santarém
ISSN: 2182-9608
https://revistas.rcaap.pt/uiips/

# O META-DATASET AMTRIANGLE PARA EXPERIÊNCIAS COM SUPERVISED MACHINE LEARNING

## The AmTriangle meta-dataset for experimenting with Supervised Machine Learning

## El metaconjunto de datos AmTriangle para experimentar con el aprendizaje automático supervisado

**Artur Marques**

Instituto Politécnico de Santarém

artur.marques@esg.ipsantarem.pt | ORCID: 0000-0002-1625-0341 | Ciência ID: 5114-3496-0D1F

## RESUMO

Este artigo apresenta a primeira versão de um meta-dataset, designado de "AmTriangle", e de um conjunto de ferramentas relacionadas, concebidas para a introdução e experimentação de situações relacionadas com "supervised machine learning"/aprendizagem-máquina supervisionada.

É um "meta-dataset", porque é um instrumento para a geração de conjuntos de amostras (datasets), tantas quantas se desejar, devidamente classificadas. Cada amostra é um triângulo, classificado como "acute", "obtuse" ou "right", consoante seja "agudo", "obtuso" ou "reto", conforme convencionado em trigonometria.

Um dataset gerado pode ser utilizado para "ensinar" a classificar novas amostras, por diferentes técnicas, como KNN ou redes neuronais; os resultados dependem de opções, discutidas. O artigo faz paralelos com outros datasets: os "triângulos" poderiam ser outros objetos, abordáveis por analogia.

**Palavras-chave**: AmTriangle, dataset, Python code, KNN, supervised learning

## ABSTRACT

This article presents the first version of a meta-dataset named "AmTriangle", and related tools, designed for introducing and experimenting with supervised machine learning.

It is a "meta-dataset", because it is an instrument for generating sets of samples (datasets), as many as desired, correctly classified. Each sample is a triangle, classified as "acute", "obtuse" or "right", according to trigonometry.

A generated dataset can be used to "teach-by-example" how to classify new samples, by different techniques, such as KNN or neural networks; the results depend on options, discussed. The article makes parallels with other datasets: the "triangles" could be other analogous objects.

**Keywords:** AmTriangle, dataset, Python code, KNN, supervised learning

## 1 INTRODUÇÃO

"Machine learning" é uma área de Inteligência Artificial em que os programas de computador, "máquinas", conseguem aprender autonomamente, não a partir de algoritmia taxativa que determina em absoluto todo os resultados, mas antes a partir de exemplos, que permitirão a construção de um modelo (Alzubi, Nayyar, & Kumar, 2018; Gangadhar & Shanta, 2018; Jaime, Ryszard, & Tom, 1983).

Neste contexto, um modelo é como que uma função que, recebendo entradas - inclusive entradas nunca antes vistas -, deverá conseguir classificá-las bem, com certo grau de confiança.

A precisão ou confiança na classificação feita, alude às técnicas estatísticas subjacentes. "Statistical learning" é outra designação para "machine learning".

O "AmTriangle" é para problemas de classificação discreta; mas, noutras situações, o desafio é o output de um valor "contínuo" - problemas de regressão.

As técnicas aplicadas aos datasets gerados, e adiante discutidas, tiram partido de estar disponível uma classificação correta para cada amostra que venha a ser utilizada para "ensinar" ou "treinar" a máquina. É esta disponibilidade de exemplos devidamente classificados que justifica a expressão "supervised learning" (Caruana & Niculescu-Mizil, 2006; Dridi, 2021). Noutras situações, as classificações certas poderão não estar disponíveis e a máquina terá de as descobrir - essa outra classe de problemas diz-se "unsupervised learning" (Barlow, 1989; Ghahramani, 2004).

Em cada dataset "AmTriangle" todas e cada uma das amostras consistem em vetores de três números reais, correspondentes aos três ângulos internos de um triângulo. Por exemplo, o vetor (90.0, 2.06, 87.94) é uma amostra válida correspondente a um triângulo cuja classificação certa é "right", ou "reto", por ter um ângulo interno de 90 graus - note-se que a soma dos três valores computa 180. Ângulos superiores a 90 graus dizem-se "obtusos"; inferiores a 90 graus dizem-se "agudos".

Assim, para um dataset com, por exemplo, 999 amostras, ter-se-á como que uma matriz 2D de 999 linhas por 3 colunas. Por haver três classificações possíveis, este é um "3-class classification problem". Neste contexto de discurso, outras designações habituais para classificações são "targets" e "classes".

Todas as técnicas e instrumentos foram implementados na linguagem de programação Python. Na versão correspondente a este artigo, a solução consiste nos programas "triangle_plural_datasets_generator.py" e "ml_classifier.py", que também serão referidas como o "gerador" e o "classificador".

O código em "triangle_plural_datasets_generator.py" produz datasets.

O código em "ml_classifier.py" gera um modelo de machine learning, capaz de aprender com um dataset gerado e classificar novas amostras, com um grau de confiança aferível.

Datasets com mais amostras deverão conduzir a modelos mais confiáveis, em média. Todavia, o número de amostras é apenas uma opção que influencia a precisão dos modelos: se houver um desequilíbrio no número de exemplos por classe/"target", a aprendizagem poderá resultar tendenciosa, ou "biased" nalgum sentido, por sub-representação de alternativas.

## 2 ARQUITETURA E TÉCNICAS APLICADAS NA SOLUÇÃO

O gerador de datasets oferece a possibilidade de produzir coleções absolutamente aleatórias, ou garantindo a representação por igual de exemplos de todas as classes: por exemplo, para uma amostra de 999 exemplos, contendo 333 triângulos "acute", 333 triângulos "obtuse" e 333 triângulos "right".

O modelo corrente é treinado pelo algoritmo de K-Nearest-Neighbors (KNN), na sua complexidade mais elevada; isto é, com k=1.

Em KNN (Zhang, 2016), cada amostra nova N - ou seja, cada triângulo nunca antes visto -, que a máquina tenha que classificar a partir do que tenha aprendido, vai ser medido em "distância" relativamente a todos os exemplos conhecidos.

Com k=1, encontrar-se-á o exemplo mais "próximo" e a sua classificação C será determinada como a classificação a atribuir a N.

Com k>1, encontrar-se-ão os top k exemplos mais "próximos" de N, e será escolhida a classificação maioritária no conjunto das classificações que lhes correspondam. Isto significa que, no caso extremo de k = número de elementos na amostra, a classificação mais frequente será sempre a atribuída a qualquer nova amostra, o que significa um modelo de baixa complexidade e, provavelmente, baixa precisão - este seria um caso de "underfitting" ou "undercomputing" (Dietterich, 1995), pois o modelo não se ajustaria aos valores corretos para os dados de treino, generalizando em excesso.

Com redes neuronais - que são grafos computacionais em que cada nó desempenha funções matemáticas de ponderação dos seus inputs, de forma a tentar chegar a um resultado, que poderá alimentar nós em camadas seguintes, até se chegar a uma camada de output cujas saídas ambicionam estar "próximas" do que os exemplos ensinaram como targets - a complexidade será tanto maior quanto maior a quantidade de nós e/ou camadas.

Modelos muito complexos podem gerar situações ditas de "overfitting" (Roelofs et al., 2019), em que o modelo desempenha muito bem com os dados de treino, mas não consegue generalizar como se desejaria, perante novas amostras.

Um algoritmo de mensuração de proximidade, adequado e adaptável a espaços dimensionais diversos, incluindo o espaço 2D em que os triângulos se inscrevem, é a distância de Minkowski.

Em "AmTriangle", o módulo "ml_classifier.py" utiliza KNN com k=1 e distância de Minkowski.

A distância de Minkowski de ordem n, entre amostras P e Q, define-se assim:

d(P,Q) = ( |p1-q1|^n + ... + |pn-qn|^n )^(1/n)

Ou seja, para n=2 é a distância Euclidiana, e para n=1 a distância de Manhattan, a soma do valor absoluto das diferenças entre todos os componentes dos vetores que se pretendam medir.

Concretizando para vetores P e Q, cada um com três números reais, que definem um triângulo pelos seus ângulos internos, assumindo que a soma dos três valores é exatamente 180, tem-se o seguinte:

P = (p1, p2, p3) #p1+p2+p3 = 180

Q = (q1, q2, q3) #q1+q2+q3 = 180

Distância de Manhattan(P,Q) = |p1-q1| + |p2-q2| + |p3-q3|

Distância Euclidiana(P,Q) = sqrt( |p1-q1|^2  + |p2-q2|^2 + |p3-q3|^2 )

## 3 FLUXO DE DADOS E RACIONAL

O módulo gerador de datasets produz ficheiros JSON (JavaScript Object Notation), que descrevem os triângulos computados aleatoriamente. O módulo aceita um argumento que estabelece quantas amostras se pretendem, por "target" ou classe de classificação possível.

O módulo classificador pode ser utilizado enviando-lhe um destes ficheiros.

Por exemplo, chamando pela linha de comandos a geração de um dataset com apenas uma amostra de cada tipo de triangulo:

python triangle_plural_datasets_generator.py 1

resulta a escrita do ficheiro "triangles_dataset_1_samples_per_target.JSON", inútil para aprendizagem, pela falta de pluralidade dos seus dados.

O propósito deste caso inútil é tão somente facilitar o entendimento do ficheiro gerado, visualizado na imagem seguinte.

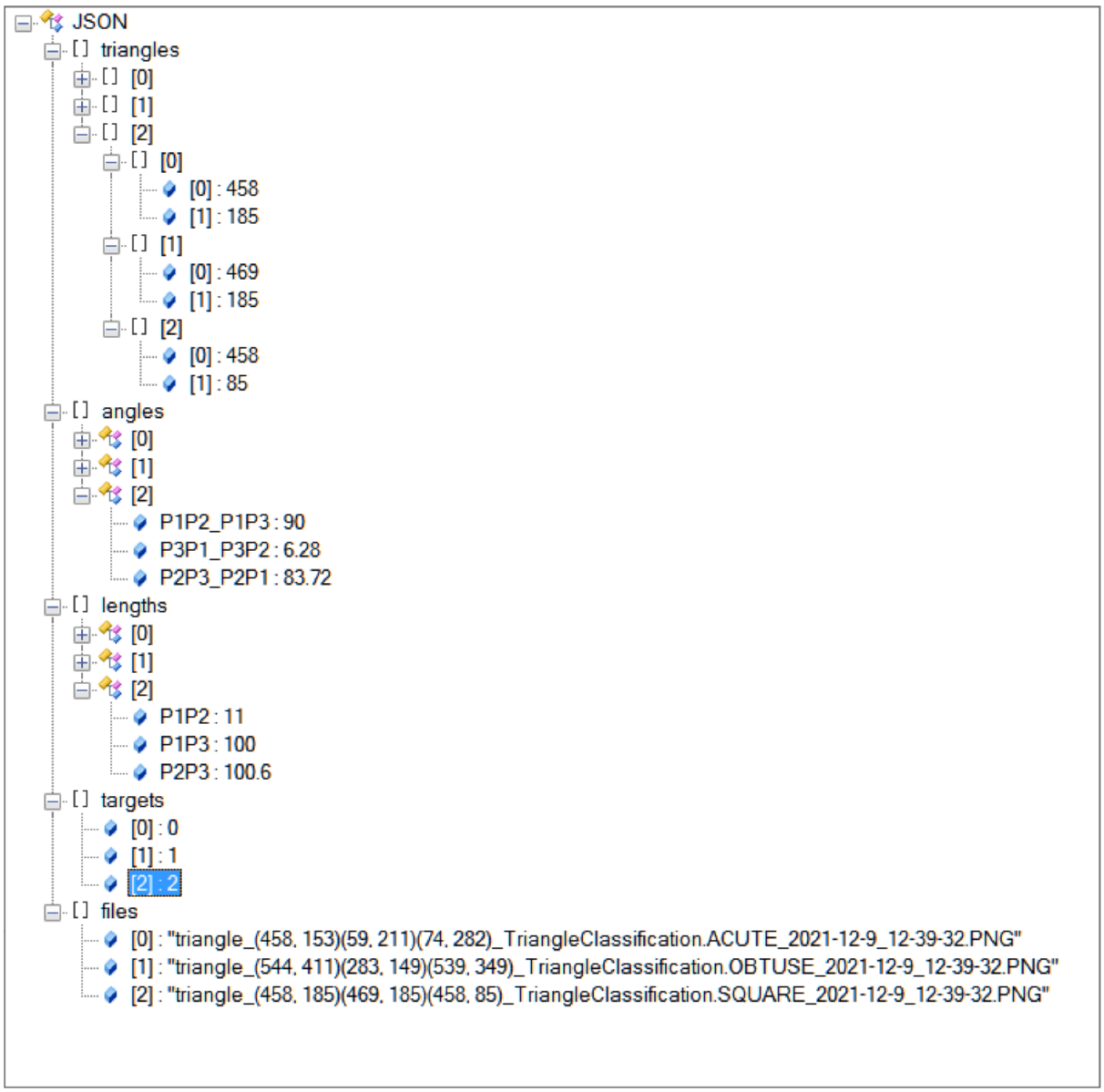


*Figura 1: Representação JSON de um dataset "AmTriangle"*

O dataset está organizado em secções ou "chaves" de nomes "triangles", "angles", "lengths", "targets" e "files".

Algumas destas chaves acedem a coleções de endereços baseados em zero:

- "triangles" é uma coleção de vetores, cada um com 3 números reais, correspondentes a 3 pontos que permitem definir uma representação gráfica de um triângulo. Na figura vê-se descrito o triângulo ((458, 185), (469, 185), (458,85)), que deve ser imaginado num referencial em que o ponto (0,0) é o canto superior esquerdo do plano.
- "angles" é uma coleção de vetores, cada um com 3 números reais, correspondentes aos ângulos internos estabelecidos pelos lados definidos pelos pontos do triângulo. Na figura vê-se expandido (90, 6.28, 83.72).

- "lengths" é uma coleção de vetores, cada um com 3 números reais, correspondentes aos comprimentos dos lados de um triângulo. Na figura vê-se (11, 100, 100.6).
- "targets" corresponde à classificação correta de cada amostra. Na estrutura JSON, lê-se que o primeiro triângulo tem target 0 ("acute"), o segundo target 1 ("obtuse") e o terceiro target 2 ("right").
- Por fim, a chave "files" corresponde ao nome de ficheiros com as imagens dos triângulos, se houver necessidade de visualizá-los.

Em muitos problemas de machine learning, a classificação de amostras é fornecida por peritos. Por exemplo, no estabelecido "Iris dataset" de Ronald Fisher (Fisher, 1936), o biólogo classifica 150 amostras de flores em 3 espécies/targets, em função de 4 números reais (comprimento e largura de sépalas, comprimento e largura de pétalas); e no "UCI ML Breast Cancer Wisconsin Diagnostic dataset" (Wolberg, Street, & Mangasarian, 1995) (https://goo.gl/U2Uwz2), os criadores classificam imagens de tecidos extraídos de nódulos mamários, em benignos ou malignos, em função de 10 valores.

No "AmTriangle" dataset, a classificação correta é determinada por métodos trigonométricos, que fazem a vez de "peritos". Estes métodos só atuam no fabrico do dataset. Durante a classificação, só operam as técnicas de machine learning, baseadas nos exemplos.

Na realidade, a classificação de triângulos não precisa de técnicas de machine learning - resolve-se com trigonometria secular. O racional que fundamenta o meta-dataset "AmTriangle" é o seguinte:

- Em primeiros contactos com machine learning, a falta de experiência com áreas periféricas, como botânica ou imagiologia médica, pode constituir um obstáculo ao entendimento do que significam as amostras e porque são classificadas como são. A utilização de amostras triviais, num contexto bem compreendido, minimiza essa barreira e liberta o praticante para os instrumentos e técnicas concretas que estiverem em aplicação.
- Um triângulo pode ser entendido como qualquer objeto com um trio de "features" ou características, como, por exemplo, um inseto voador visto pela perspetiva de (envergadura de asas em mm, comprimento em mm, massa em gramas). Isto significa um convite à imaginação do praticante.
- Sendo este um projeto com código free and open source (FOSS), o praticante pode, com liberdade, adaptar o módulo gerador para produzir entidades n-dimensionais, com n>3, ficando com um instrumento que lhe confere o poder de escrever datasets de tamanho variável (Alwosheel, van Cranenburgh, & Chorus, 2018), adequados a aferições que procurem relacionar o tamanho da coleção, e/ou a quantidade de features, com os resultados de diferentes técnicas que queira testar.

No workflow previsto, primeiro gera-se um dataset e, depois, o ficheiro JSON do dataset serve de input ao módulo de aprendizagem.

Eis um exemplo de chamada, pela linha de comandos, do treino de um modelo:

ml_classifier.py triangles_dataset_400_samples_per_target.JSON

O modelo utilizado na chamada consiste em 400 amostras de cada target.

Durante o funcionamento do gerador, parâmetros no código controlam aspetos acessórios, como se são gravadas as imagens dos triângulos.

Durante o funcionamento do classificador, acontece feedback diverso: visualizações do conjunto, listagem das amostras, listagem das classificações correspondentes, resumo do treino feito com as amostras, e qual a precisão do modelo a que o treino conduziu.

## 4 CONSTRUÇÃO E AFERIÇÃO DO MODELO

O que define realmente um triângulo, para a classificação ambicionada, são os seus ângulos internos. Assim, uma das primeiras operações no classificador, é a extração dos valores dos ângulos.

Como existe informação extra, nomeadamente designações para os lados que formam os ângulos, há que proceder-se a uma "limpeza" de redução, até se ter uma estrutura de dados que corresponda ao que será estritamente necessário para treino e aferição.

O excerto de código seguinte arruma no identificador "listOfListsEachATrioOfAngles" os dados apropriados.

```
datasetAsListOfDictsOfAngles = dictDatasetFromJSON['angles']
listOfListsEachATrioOfAngles = []
for dictTrio in datasetAsListOfDictsOfAngles:
    listCurrentTrio = auxConvertDictAsTrioOfAnglesToListOfAngles(dictTrio)
    listOfListsEachATrioOfAngles.append(listCurrentTrio)
#for every sample
```

Tais dados são uma lista de vetores de 3 valores, 3 ângulos.

Uma outra lista, paralela, fornece a sua correta classificação. No excerto de código seguinte, essa lista é o identificador "listOfTargetsForEachTrioOfAngles".

```
listOfTargetsForEachTrioOfAngles = dictDatasetFromJSON["targets"]
```

Segue-se um processo de reserva de uma fração dos dados para treino (75%, por defeito), e outra fração para aferição dos resultados do treino (25%, por defeito). Para esse efeito, utilizou-se o instrumento "model_selection.train_test_split" do módulo "sklearn".

O resultado do trabalho desse instrumento é a divisão ("split") da coleção recebida, em partes que serão utilizadas para "treino" ("train") e "teste" ("test").

O excerto de código seguinte corresponde a esse momento:

```
from sklearn import model_selection
tupleTrainAndTestsSets = model_selection.train_test_split(
    listOfListsEachATrioOfAngles,
    listOfTargetsForEachTrioOfAngles,
    shuffle=True
)
X_train, X_test, y_train, y_test = tupleTrainAndTestsSets
```

Em machine learning, é comum terminar este estágio, disponibilizando os seguintes identificadores:

- X_train - a lista dos vetores que serão utilizados como exemplos para aprendizagem;
- X_test - uma lista de vetores, que NÃO serão utilizados como exemplos, mas antes reservados para se medir a qualidade do treino, em estágio ulterior. A máquina terá que, autonomamente, classificar estas amostras, sem ter acesso à sua correta classificação,

permitindo assim ao praticante de machine learning comparar os resultados que vierem a ser atingidos pelo modelo, com os que são os resultados corretos.

- y_train - a lista de targets corretas; isto é, de classificações, para cada uma das amostras em X_train.
- y_test - a lista de targets, para a classificação das amostras em X_test. Esta lista é fundamental para o processo de aferição da precisão do modelo obtido.

A aprendizagem faz-se com o algoritmo KNN, utilizando a distância de Minkowski.

O processo de treino ocorre aquando da chamada do método "fit" do classificador.

```
knn = KNeighborsClassifier(
   algorithm="auto",
   metric="minkowski"
   n_neighbors=1
)
#treino!
knnResult = knn.fit(
   X_train,
   y_train
)
```

Após o "fitting" aos dados, que é a aprendizagem a partir dos exemplos, o objeto programático "knnResult" pode ser utilizado para fazer previsões sobre amostras novas.

Por exemplo, a execução do código seguinte deixa contido em "thePrediction" uma previsão de qual deverá ser a classificação de uma nova amostra.

```
thePrediction = knnResult.predict(someNewSample)
```

A previsão estará certa ou errada. Para determinar quanto se pode confiar nas previsões feitas pelo modelo, todos os elementos de X_test serão sujeitos ao método predict, e todas as previsões resultantes serão comparadas com as certas, disponíveis em y_test. Se, acaso, por exemplo, 70 em 75 estiverem certas, a precisão do modelo é 70/75 = 0.9(3) ~ 93%

## 5 Observação dos resultados (dataset gerado e precisão do modelo)

Cada triângulo consiste em 3 atributos/"features". A visualização de 3 ou mais atributos é menos fácil do que a visualização de pares 2D. Uma possibilidade de visualização é trabalhar com uma matriz de dispersão, que é um instrumento cuja diagonal consiste em histogramas de frequências absolutas de cada um dos atributos em análise - cada um dos ângulos, neste projeto. Os restantes slots da "scatter matrix", ditos de "pair plots", são representações 2D da interceção de pares de valores das outras características, duas a duas.

Na figura 2, tem-se a visualização de um dataset com 100 amostras de cada classe de triângulo, portanto um dataset de 300 amostras, 75% das quais serviram para treino (225) e 25% para aferição (75).

Os pontos de dados de triângulos "acute" aparecem a vermelho, os "obtuse" a verde, e os "right" a azul, ou outras cores que se decidam no código, conforme dicionário de correspondências, de acordo com o modelo matemático de cor "RGBA" (Red Green Blue Opacity).

```
dictColorsForEachTargetClassAsDesired=\
{0:"#FF0000", 1:"#00FF00", 2:"#0000FF"} #RGBA color model
```

Por limitações gráficas, poderão não ser visualizáveis todas as amostras nos histogramas.

Uma ferramenta auxiliar, facilitadora da observação das previsões, permite sujeitar qualquer amostra à sua classificação pelo modelo, e medir se o modelo acerta, ou erra, perante a nova submissão. A título meramente ilustrativo, eis o output da utilização dessa ferramenta, depois de duas chamadas em que as previsões foram corretas (OK!):

```
P1=(537, 200) P2=(247, 382) P3=(367, 350)
P1P2_P1P3=9.31 P3P1_P3P2=153.51 P2P3_P2P1=17.18
P1P2=342.38 P1P3=226.72 P2P3=124.19
TriangleClassification.OBTUSE

OK! [(537, 200), (247, 382), (367, 350)] angles [[  9.31 153.51  17.18]] shaped (1, 3) predicted as class [1] = Obtuse
```

---

```
P1=(191, 131) P2=(501, 131) P3=(191, 296)
P1P2_P1P3=90.0 P3P1_P3P2=61.98 P2P3_P2P1=28.02
P1P2=310.0 P1P3=165.0 P2P3=351.18
TriangleClassification.SQUARE

OK! [(191, 131), (501, 131), (191, 296)] angles [[90.   61.98 28.02]] shaped (1, 3) predicted as class [2] = Square
```

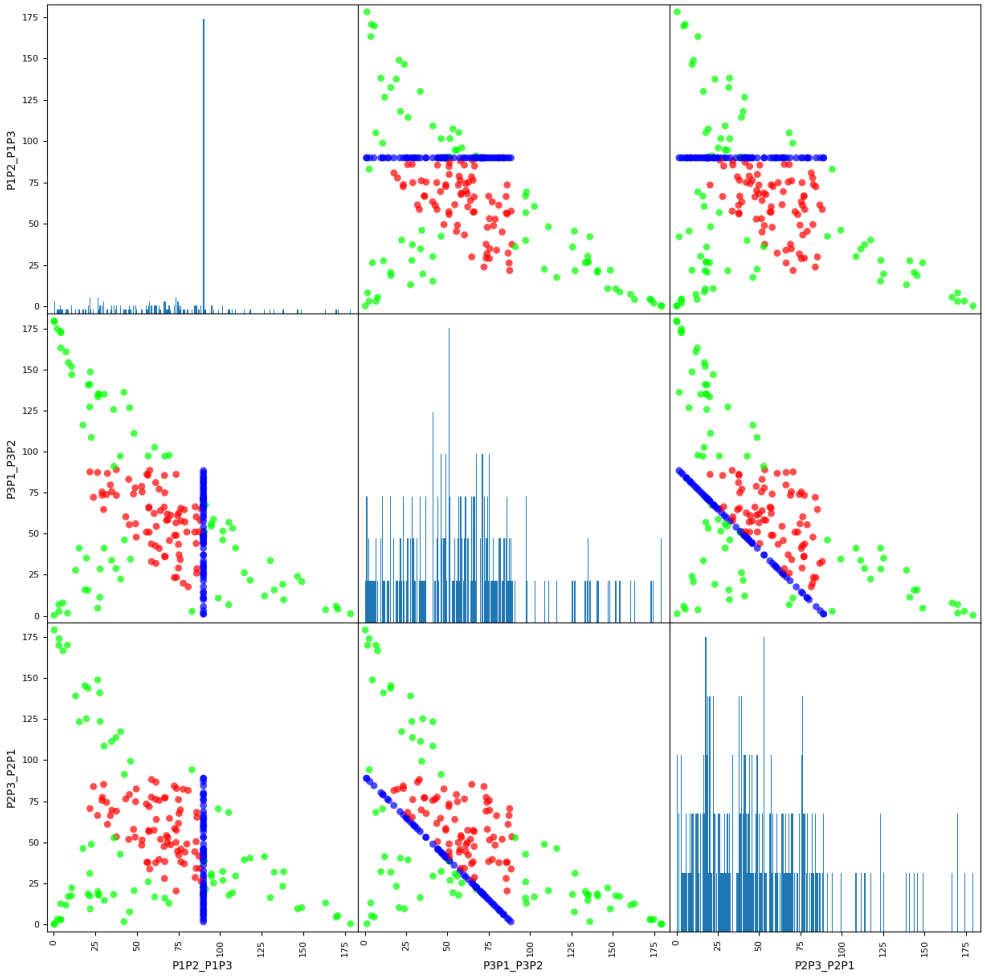


*Figura 2: Scatter matrix para um AmTriangle dataset com 100 amostras de cada tipo*

O módulo classificador termina o seu funcionamento com um cálculo da precisão do modelo, obtido de duas formas:

- utilizando o método score do objeto que fez o treino - knnResult.score(X_test, y_test),
- e confrontando cada previsão com o resultado correto, via operador relacional de comparação.

No dataset utilizado para este exemplo, com 100 amostras por cada tipo de triângulo, a precisão foi de ~93%:

Model accuracy / average matches: 0.9333333333333333

Model score by knn.score 0.9333333333333333.

## 6 Discussão dos resultados

"AmTriangle" é um par de instrumentos programáticos que objetivam a introdução e experimentação de situações relacionadas com "supervised machine learning" ou aprendizagem-máquina supervisionada.

Um dos instrumentos é um gerador de datasets com qualquer número de amostras, cada uma com 3 features, devidamente classificada como triângulo "acute", "obtuse" ou "right".

O outro instrumento é um classificador automático, que aprende com os exemplos fornecidos via datasets, utilizando o algoritmo KNN, com k=1, por defeito.

A flexibilidade quanto ao número de amostras, a adaptação fácil para espaços dimensionais diferentes, e o output de datasets em formato JSON, consumível por outros programas, deverão facilitar o entendimento da codificação de dados e convidar à experimentação.

Os instrumentos de observação e aferição, baseados em KNN com distância de Minkowski, utilizam conceitos familiares de "proximidade".

O utilizador pode pedir datasets pequenos e/ou biased, para testemunhar quebras na precisão do modelo. Ou pode pedir datasets gigantes e equilibrados, com diferentes valores/complexidades de k em KNN, para causar flutuações no esforço computacional e na precisão.

Assim, a modularidade e opções disponíveis nos instrumentos descritos, argumentam-se como facilitadores para a introdução e experimentação com aprendizagem-máquina supervisionada.

## 7 REFERÊNCIAS


Alwosheel, A., van Cranenburgh, S., & Chorus, C. G. (2018). Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis. *Journal of Choice Modelling, 28*, 167-182. doi:https://doi.org/10.1016/j.jocm.2018.07.002

Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series, 1142*, 012012. doi:10.1088/1742-6596/1142/1/012012

Barlow, H. B. (1989). Unsupervised Learning. *Neural Computation, 1*(3), 295-311. doi:10.1162/neco.1989.1.3.295

Caruana, R., & Niculescu-Mizil, A. (2006). An Empirical Comparison of Supervised Learning Algorithms. *Icml '06*, 161–168. doi:10.1145/1143844.1143865

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Comput. Surv., 27*(3), 326–327. doi:10.1145/212094.212114

Dridi, S. (2021). *Supervised Learning - A Systematic Literature Review*.

Fisher, R. (1936). *Fisher's Irises*. Retrieved from: https://dasl.datadescription.com/datafile/fishers-irises/

Gangadhar, S., & Shanta, R. (2018). Chapter 8 - Machine Learning. *Handbook of Statistics, 38*, 197-228. doi:https://doi.org/10.1016/bs.host.2018.07.004

Ghahramani, Z. (2004). Unsupervised Learning. In O. Bousquet, U. von Luxburg, & G. Rätsch (Eds.), *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures* (pp. 72-112). Berlin, Heidelberg: Springer Berlin Heidelberg.

Jaime, G. C., Ryszard, S. M., & Tom, M. M. (1983). 1 - an Overview of Machine Learning. 3-23. doi:https://doi.org/10.1016/B978-0-08-051054-5.50005-4

Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., & Schmidt, L. (2019). A Meta-Analysis of Overfitting in Machine Learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 32): Curran Associates, Inc. %U https://proceedings.neurips.cc/paper/2019/file/ee39e503b6bedf0c98c388b7e8589aca-Paper.pdf.

Wolberg, D. W. H., Street, W. N., & Mangasarian, O. L. (1995). *Breast Cancer Wisconsin (Diagnostic) Data Set*. Retrieved from: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Zhang, Z. (2016). Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine, 4*(11), 218-218. doi:10.21037/atm.2016.03.37