Avinash Navlani
27 de diciembre de 2019

SCIKIT-LEARN + 1

Soporta máquinas vectoriales con Scikit-learn

En este tutorial, aprenderá sobre Support Vector Machines, uno de los algoritmos de aprendizaje automático supervisado más populares y ampliamente utilizados.

SVM ofrece una precisión muy alta en comparación con otros clasificadores, como la regresión logística y los árboles de decisión. Es conocido por su truco del kernel para manejar espacios de entrada no lineales. Se utiliza en una variedad de aplicaciones, como detección de rostros, detección de intrusos, clasificación de correos electrónicos, artículos de noticias y páginas web, clasificación de genes y reconocimiento de escritura a mano.

En este tutorial, usará scikit-learn en Python. Si desea obtener más información sobre este paquete de Python, le recomiendo que eche un vistazo a nuestro curso de Aprendizaje supervisado con scikit-learn .

SVM es un algoritmo interesante y los conceptos son relativamente simples. El clasificador separa los puntos de datos utilizando un hiperplano con la mayor cantidad de margen. Es por eso que un clasificador SVM también se conoce como clasificador discriminativo. SVM encuentra un hiperplano óptimo que ayuda a clasificar nuevos puntos de datos.

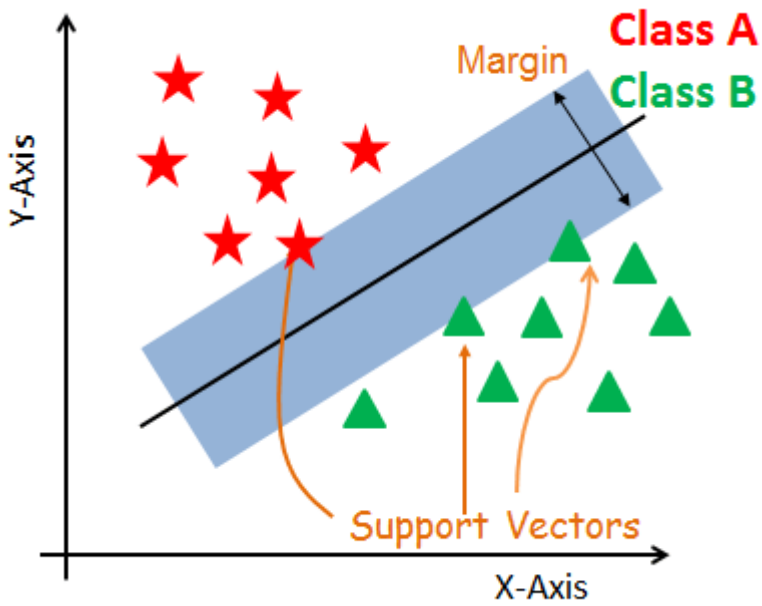
En este tutorial, cubrirá los siguientes temas:

- Máquinas de vectores de soporte
- ¿Como funciona?

- Construcción de clasificadores en Scikit-learn
- Ajuste de hiperparámetros
- Ventajas y desventajas

Máquinas de vectores de soporte

Generalmente, Support Vector Machines se considera un enfoque de clasificación, pero puede emplearse en ambos tipos de problemas de clasificación y regresión. Puede manejar fácilmente múltiples variables continuas y categóricas. SVM construye un hiperplano en un espacio multidimensional para separar diferentes clases. SVM genera un hiperplano óptimo de manera iterativa, que se utiliza para minimizar un error. La idea central de SVM es encontrar un hiperplano marginal máximo (MMH) que divida mejor el conjunto de datos en clases.



Vectores de apoyo

Los vectores de soporte son los puntos de datos más cercanos al hiperplano. Estos puntos definirán mejor la línea de separación al calcular los márgenes. Estos puntos son más relevantes para la construcción del clasificador.

Hiperplano

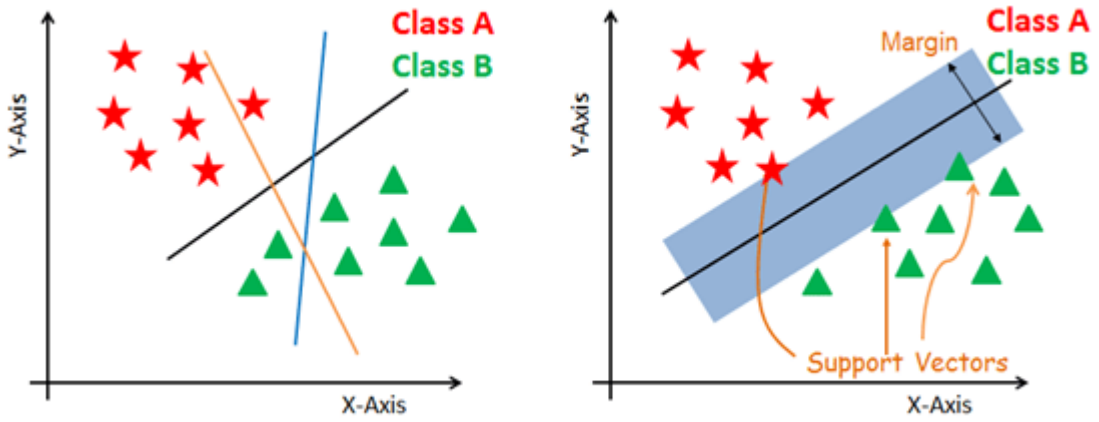
Un hiperplano es un plano de decisión que separa un conjunto de objetos que tienen diferentes membresías de clase.

Un margen es un espacio entre las dos líneas en los puntos de clase más cercanos. Esto se calcula como la distancia perpendicular desde la línea hasta los vectores de soporte o los puntos más cercanos. Si el margen es mayor entre las clases, entonces se considera un buen margen, un margen más pequeño es un mal margen.

¿Cómo funciona SVM?

El objetivo principal es segregar el conjunto de datos dado de la mejor manera posible. La distancia entre los puntos más cercanos se conoce como margen. El objetivo es seleccionar un hiperplano con el máximo margen posible entre los vectores de soporte en el conjunto de datos dado. SVM busca el hiperplano marginal máximo en los siguientes pasos:

1. Genere hiperplanos que segreguen las clases de la mejor manera. Figura del lado izquierdo que muestra tres hiperplanos negro, azul y naranja. Aquí, el azul y el naranja tienen un error de clasificación más alto, pero el negro separa las dos clases correctamente.
2. Seleccione el hiperplano derecho con la máxima segregación de los puntos de datos más cercanos como se muestra en la figura del lado derecho.

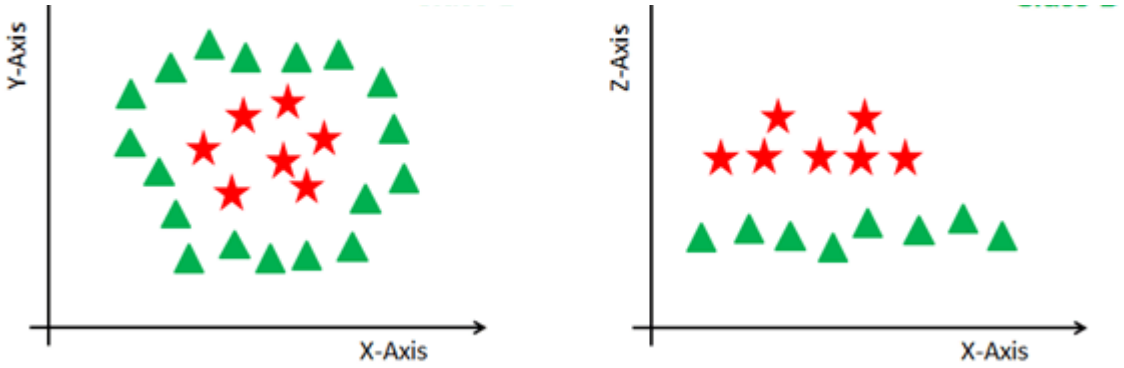


Lidiando con planos no lineales e inseparables

Algunos problemas no se pueden resolver utilizando un hiperplano lineal, como se muestra en la siguiente figura (lado izquierdo).

En tal situación, SVM usa un truco del kernel para transformar el espacio de entrada en un espacio de mayor dimensión como se muestra a la derecha. Los puntos de datos se grafican en el eje xy en el

eje z (Z es la suma al cuadrado de xey: z = x ^ 2 = y ^ 2). Ahora puede segregar fácilmente estos puntos mediante la separación lineal.



Núcleos SVM

El algoritmo SVM se implementa en la práctica utilizando un kernel. Un kernel transforma un espacio de datos de entrada en la forma requerida. SVM usa una técnica llamada truco del kernel. Aquí, el kernel toma un espacio de entrada de baja dimensión y lo transforma en un espacio de mayor dimensión. En otras palabras, puede decir que convierte un problema no separable en problemas separables añadiéndole más dimensión. Es más útil en problemas de separación no lineal. El truco del kernel te ayuda a construir un clasificador más preciso.

- **Kernel** lineal Un kernel lineal se puede utilizar como producto escalar normal dos observaciones dadas cualesquiera. El producto entre dos vectores es la suma de la multiplicación de cada par de valores de entrada.

```
K(x, xi) = sum(x * xi)
```

- **Núcleo** polinómico Un núcleo polinómico es una forma más generalizada del núcleo lineal. El núcleo polinomial puede distinguir el espacio de entrada curvo o no lineal.

```
K(x,xi) = 1 + sum(x * xi)^d
```

Donde d es el grado del polinomio. d = 1 es similar a la transformación lineal. El título debe especificarse manualmente en el algoritmo de aprendizaje.

- **Núcleo de** función de base radial El núcleo de función de base radial es una función de núcleo popular que se usa comúnmente en la clasificación de máquinas vectoriales de soporte. RBF

puede mapear un espacio de entrada en un espacio dimensional infinito.

```
K(x, xi) = exp( gamma * sum((x - xi^2))
```

Aquí gamma es un parámetro, que varía de 0 a 1. Un valor más alto de gamma se ajustará perfectamente al conjunto de datos de entrenamiento, lo que provoca un ajuste excesivo. Gamma = 0,1 se considera un buen valor predeterminado. El valor de gamma debe especificarse manualmente en el algoritmo de aprendizaje.

Construcción de clasificadores en Scikit-learn

Hasta ahora, ha aprendido sobre los antecedentes teóricos de SVM. Ahora aprenderá sobre su implementación en Python usando scikit-learn.

Por ejemplo, [illegible] problema de clasificación de clases múltiples muy famoso. Este conjunto de datos se calcula a partir de una imagen digitalizada de un aspirado con aguja fina (FNA) de una masa mamaria. Describen las características de los núcleos celulares presentes en la imagen.

El conjunto de datos comprende 30 características (radio medio, textura media, perímetro medio, área media, suavidad media, compacidad media, concavidad media, puntos cóncavos medios, simetría media, dimensión fractal media, error de radio, error de textura, error de perímetro, error de área, error de suavidad, error de compacidad, error de concavidad, error de puntos cóncavos, error de simetría, error de dimensión fractal, peor radio, peor textura, peor perímetro, peor área, peor suavidad, peor compacidad, peor concavidad, peores puntos cóncavos, peor simetría y peor dimensión fractal) y un objetivo (tipo de cáncer).

Estos datos tienen dos tipos de clases de cáncer: maligno (dañino) y benigno (no dañino). Aquí, puede construir un modelo para clasificar el tipo de cáncer. El conjunto de datos está disponible en la biblioteca scikit-learn o también puede descargarlo de la biblioteca de aprendizaje automático de UCI.

Loading Data

Let's first load the required dataset you will use.

```
#Import scikit-learn dataset library
from sklearn import datasets
```

```
#Load dataset
cancer = datasets.load_breast_cancer()
```

After you have loaded the dataset, you might want to know a little bit more about it. You can check feature and target names.

```
# print the names of the 13 features
print("Features: ", cancer.feature_names)

# print the label type of cancer('malignant' 'benign')
print("Labels: ", cancer.target_names)
```

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
```

Let's explore it for a bit more. You can also check the shape of the dataset using shape.

```
# print data(feature)shape
cancer.data.shape
```

```
(569, 30)
```

Let's check top 5 records of the feature set.

```
# print the cancer data features (top 5 records)
print(cancer.data[0:5])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
```

```
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]



  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```

Let's take a look at the target set.

```
# print the cancer labels (0:malignant, 1:benign)
print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
```

```
0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function `train_test_split()` . you need to pass 3 parameters features, target, and test_set size. Additionally, you can use random_state to select records randomly.

```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,random_s
```

Generating Model

Let's build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in `SVC()` function.

Then, fit your model on train set using `fit()` and perform prediction on the test set using `predict()` .

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluating the Model

Let's estimate how accurately the classifier or model can predict the breast cancer of patients.

Accuracy can be computed by comparing actual test set values and predicted values.

```
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
```

Well, you got a classification rate of 96.49%, considered as very good accuracy.

For further evaluation, you can also check precision and recall of model.

```
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Precision: 0.9811320754716981
Recall: 0.9629629629629629
```

Well, you got a precision of 98% and recall of 96%, which are considered as very good values.

Tuning Hyperparameters

- **Kernel**: The main function of the kernel is to transform the given dataset input data into the required form. There are various types of functions such as linear, polynomial, and radial basis function (RBF). Polynomial and RBF are useful for non-linear hyperplane. Polynomial and RBF kernels compute the separation line in the higher dimension. In some of the applications, it is suggested to use a more complex kernel to separate the classes that are curved or nonlinear. This transformation can lead to more accurate classifiers.
- **Regularization**: Regularization parameter in python's Scikit-learn C parameter used to maintain regularization. Here C is the penalty parameter, which represents misclassification or error term.

The misclassification or error term tells the SVM optimization how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. A smaller value of C creates a small-margin hyperplane and a larger value of C creates a larger-

- Gamma: A lower value of Gamma will loosely fit the training dataset, whereas a higher value of gamma will exactly fit the training dataset, which causes over-fitting. In other words, you can say a low value of gamma considers only nearby points in calculating the separation line, while the a value of gamma considers all the data points in the calculation of the separation line.

Advantages

SVM Classifiers offer good accuracy and perform faster prediction compared to Naïve Bayes algorithm. They also use less memory because they use a subset of training points in the decision phase. SVM works well with a clear margin of separation and with high dimensional space.

Disadvantages

SVM is not suitable for large datasets because of its high training time and it also takes more time in training compared to Naïve Bayes. It works poorly with overlapping classes and is also sensitive to the type of kernel used.

Conclusion

Congratulations, you have made it to the end of this tutorial!

In this tutorial, you covered a lot of ground about Support vector machine algorithm, its working, kernels, hyperparameter tuning, model building and evaluation on breast cancer dataset using the Scikit-learn package. You have also covered its advantages and disadvantages. I hope you have learned something valuable!

To learn more about this type of classifiers, you should take a look at our Linear Classifiers in Python course. It introduces other types of regression and loss functions, as well as Support Vector Machines.

I look forward to hearing any feedback or questions. You can ask the question by leaving a comment and I will try my best to answer it.

Rohit Jagannath
15/10/2018 12:14 a.m.

Can you cross check the expression(Z is the squared sum of both x and y: z=x^2=y^2) for correction?

Also, Can you try to do the same with Train, Test and Validate split?

▲ 6

Avinash Navlani
17/10/2018 08:20 p.m.

Thanks for the feedback and spotting the mistake. It should be z=x^2+y^2.

Yes, we can do this. Already, SVM performs analysis in multidimensional dataset to classify.

▲ 4

Saad Munir
15/11/2018 08:44 a.m.

How to develop a data-set yourself for the SVM classifier? Also is there any pre defined data set of word documents that can be used for Microsoft word document carving?

▲ 2

Avinash Navlani
17/11/2018 02:47 p.m.

You can take any dataset and try out SVM classifier, tune your hyperparameters. I have no idea about MS word document carving.

▲ 1

Joachim Rosenberger
12/07/2018 05:21 p.m.

```
K(x,xi) = exp(-gamma * sum((x – xi^2)) should be
```

```
K(x,xi) = exp(-gamma * sum((x – xi)^2)
```

▲ 4

Avinash Navlani
12/08/2018 04:53 p.m.

▲ 1

Rohit Kumar
16/12/2018 07:00 p.m.

Can you make an article of using SVM with Autoencoder. That'll be very useful on how to use svm as a classifier with autoencoder

▲ 3

Avinash Navlani
20/12/2018 07:42 a.m.

Thanks for your feedback!

I will try when I get time. Right now i am busy with other articles.

▲ 1

Guillermo Viñas
26/12/2018 09:19 p.m.

Dear, you have an example of multivariate analysis type MIMO, to detect models and predict variables based on neural networks in python or some alternative in python that allows to solve the indicated?

▲ 1

---

haider ali
31/03/2019 10:33 a.m.

can you show how to apply this SVM code om multi-class classification. thank you

▲ 5

Paian Simarmata
04/06/2019 02:40 p.m.

hi avinash .

i have a project to predict the maturity of roasted coffee bean from its color.

since the coffee bean roasting have three types of roasting ==> light, medium and dark with also produce different color maturity.

i have to take the image of the roasted coffee bean realtime using webcam and sent the image to be processed and determine the color of maturity so the machine will stop roasting.

my question is can i use SVM to classify the image of the coffee bean color ?

i hope you can help me.

▲ 2

Avinash Navlani
04/07/2019 07:02 p.m.
SVM can use for image classification but CNN are more effective in image classification.

▲ 1

Paulo Oliveira
06/03/2019 12:50 a.m.
Thanks for the post!
I have implemented and concluded your tutorial! Now my model is trained right? but how can i test with a new input, and see what the svm think it is?

▲ 5

Bahare Samadi
13/06/2019 08:56 p.m.
Hi, I have 27 samples that I would like to do binary classification. Each sample is a 2D- matrice. (100 x 96). I try to use the SVM, but it was not possible, since I have arrays with dim 3. ValueError: Found array with dim 3. Estimator expected <= 2. Could you please advice ?

▲ 2

Yogi Pratama
14/06/2019 08:12 a.m.
Bagaimana saya membuat 3 klasfifikasi dlam contoh kasus adalah, klasifikasi (Normal, Kurang Normal, dan Tidak Normal). Apakah bisa? Terimakasih sebelumnya

▲ 2

Elisio Armando
04/03/2020 06:49 AM
Could you please provide link to **exact** data set you are using? Thanks

▲ 1

footballbet footballbet
07/03/2020 01:46 PM
เทคนิค ก็ คือ ยาม คับ ด้วย ฝากฝัง เพื่อ ฟุตบอล ว เดิมพัน กัน ด้วย. ใน ขณะ ที่ หลาย แฟน คลับ ชอบ พนัน พวก เขา เลือก กลุ่ม หรือ อยู่ ใกล้ ๆ กำหนดการ ใช้ มัน มัน ส่ง คุ้ม ที่ จะ ลอง ดู ที่ สูง จะ ไป เอา

เพิ่มเติม ความ รู้ อยู่ บน 10Apostar บ ไซต์ ที่ คุณ จะ เป็นเป็น เป็น ออนไลน์ hechos ข้อมูล สำหรับ แต่ละ ตรง กับ ใน เรื่อง ที่ สำคัญ กลุ่ม และ สิ่ง เล็กน้อย rivalidades.

▲ 1