Apprentissage automatique

L'apprentissage automatique permet à l'ordinateur d'apprendre en étudiant des données et des statistiques.

L'apprentissage automatique est un pas dans la direction de l'intelligence artificielle (IA).

L'apprentissage automatique est un programme qui analyse les données et apprend à prédire le résultat.

Où commencer?

Dans ce didacticiel, nous reviendrons aux mathématiques et étudierons les statistiques, et comment calculer des nombres importants en fonction d'ensembles de données.

Nous allons également apprendre à utiliser divers modules Python pour obtenir les réponses dont nous avons besoin.

Et nous apprendrons à créer des fonctions capables de prédire le résultat en fonction de ce que nous avons appris.

Base de données

Dans l'esprit d'un ordinateur, un ensemble de données est une collection de données. Cela peut être n'importe quoi, d'un tableau à une base de données complète.

Exemple de tableau:

```
[99,86,87,88,111,86,103,87,94,78,77,85,86]
```

Exemple de base de données:

Nom de la voiture	Couleur	Âge
BMW	rouge	5

Volvo	noir	7
VW	gris	8
VW	blanc	7
Gué	blanc	2
VW	blanc	17
Tesla	rouge	2
BMW	noir	9
Volvo	gris	4
Gué	blanc	11
Toyota	gris	12
VW	blanc	9
Toyota	bleu	6

En regardant le tableau, nous pouvons deviner que la valeur moyenne est probablement autour de 80 ou 90, et nous sommes également en mesure de déterminer la valeur la plus élevée et la valeur la plus basse, mais que pouvons-nous faire d'autre?

Et en regardant la base de données, nous pouvons voir que la couleur la plus populaire est le blanc et que la voiture la plus ancienne a 17 ans, mais que se passerait-il si nous pouvions prédire si une voiture avait un AutoPass, simplement en regardant les autres valeurs?

C'est à cela que sert le Machine Learning! Analyser les données et prédire le résultat!

Dans le Machine Learning, il est courant de travailler avec de très grands ensembles de données. Dans ce didacticiel, nous allons essayer de rendre aussi facile que possible la compréhension des différents concepts de l'apprentissage automatique, et nous travaillerons avec de petits ensembles de données faciles à comprendre.

Types de données

Pour analyser les données, il est important de savoir à quel type de données nous avons affaire.

Nous pouvons diviser les types de données en trois catégories principales:

- **Numérique**
- **Catégorique**
- **Ordinal**

Les données **numériques** sont des nombres et peuvent être divisées en deux catégories numériques:

- Données discrètes
  - nombres limités aux entiers. Exemple: le nombre de voitures qui passent.
- Données continues
  - nombres qui ont une valeur infinie. Exemple: le prix d'un article ou la taille d'un article

Les données **catégorielles** sont des valeurs qui ne peuvent pas être comparées les unes aux autres. Exemple: une valeur de couleur ou toute valeur oui / non.

Les données **ordinales** sont comme des données catégorielles, mais peuvent être mesurées les unes par rapport aux autres. Exemple: notes scolaires où A est meilleur que B et ainsi de suite.

En connaissant le type de données de votre source de données, vous serez en mesure de savoir quelle technique utiliser lors de leur analyse.

Machine Learning - Mean Median Mode

Moyenne, médiane et mode

Que pouvons-nous apprendre en regardant un groupe de nombres?

En Machine Learning (et en mathématiques), il y a souvent trois valeurs qui nous intéressent:

- **Moyenne** - La valeur moyenne
- **Médiane** - La valeur **médiane**
- **Mode** - La valeur la plus courante

Exemple: Nous avons enregistré la vitesse de 13 voitures:

```
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]
```

Quelle est la valeur de vitesse moyenne, moyenne ou la plus courante?

Signifier

La valeur moyenne est la valeur moyenne.

Pour calculer la moyenne, trouvez la somme de toutes les valeurs et divisez la somme par le nombre de valeurs:

```
(99+86+87+88+111+86+103+87+94+78+77+85+86) / 13 = 89.77
```

Le module NumPy a une méthode pour cela. Découvrez le module NumPy dans notre didacticiel NumPy .

Exemple

Utilisez la `mean()`méthode NumPy pour trouver la vitesse moyenne:

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.mean(speed)

print(x)
```

Médian

La valeur médiane est la valeur au milieu, après avoir trié toutes les valeurs:

```
77, 78, 85, 86, 86, 86, 87, 87, 88, 94, 99, 103, 111
```

Il est important que les nombres soient triés avant de pouvoir trouver la médiane.

Le module NumPy a une méthode pour cela:

Exemple

Utilisez la `median()`méthode NumPy pour trouver la valeur du milieu:

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.median(speed)

print(x)
```

S'il y a deux nombres au milieu, divisez la somme de ces nombres par deux.

```
77, 78, 85, 86, 86, 86, 87, 87, 94, 98, 99, 103

(86 + 87) / 2 = 86.5
```

Exemple

Utilisation du module NumPy:

```
import numpy

speed = [99,86,87,88,86,103,87,94,78,77,85,86]

x = numpy.median(speed)

print(x)
```

Mode

La valeur Mode est la valeur qui apparaît le plus souvent:

```
99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86 = 86
```

Le module SciPy a une méthode pour cela. Découvrez le module SciPy dans notre didacticiel SciPy .

Exemple

Utilisez la `mode()`méthode SciPy pour trouver le numéro qui apparaît le plus:

```
from scipy import stats

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = stats.mode(speed)

print(x)
```

Résumé du chapitre

La moyenne, la médiane et le mode sont des techniques souvent utilisées dans l'apprentissage automatique, il est donc important de comprendre le concept qui les sous-tend.

Qu'est-ce que l'écart type?

L'écart type est un nombre qui décrit la répartition des valeurs.

Un faible écart type signifie que la plupart des nombres sont proches de la valeur moyenne (moyenne).

Un écart type élevé signifie que les valeurs sont réparties sur une plage plus large.

Exemple: Cette fois, nous avons enregistré la vitesse de 7 voitures:

```
speed = [86,87,88,86,87,85,86]
```

L'écart type est:

```
0.9
```

Cela signifie que la plupart des valeurs sont comprises entre 0,9 et la valeur moyenne, qui est 86,4.

Faisons de même avec une sélection de nombres avec une gamme plus large:

```
speed = [32,111,138,28,59,77,97]
```

L'écart type est:

```
37.85
```

Cela signifie que la plupart des valeurs se situent dans la plage de 37,85 à partir de la valeur moyenne, qui est de 77,4.

Comme vous pouvez le voir, un écart type plus élevé indique que les valeurs sont réparties sur une plage plus large.

Le module NumPy a une méthode pour calculer l'écart type:

Exemple

Utilisez la `std()`méthode NumPy pour trouver l'écart type:

```
import numpy

speed = [86,87,88,86,87,85,86]

x = numpy.std(speed)

print(x)
```

Exemple

```
import numpy

speed = [32,111,138,28,59,77,97]

x = numpy.std(speed)

print(x)
```

Variance

La variance est un autre nombre qui indique la répartition des valeurs.

En fait, si vous prenez la racine carrée de la variance, vous obtenez l'écart type!

Ou l'inverse, si vous multipliez l'écart type par lui-même, vous obtenez la variance!

Pour calculer la variance, vous devez procéder comme suit:

1. Trouvez la moyenne:

```
(32+111+138+28+59+77+97) / 7 = 77.4
```

2. Pour chaque valeur: trouvez la différence par rapport à la moyenne:

```
 32 - 77.4 = -45.4
111 - 77.4 =  33.6
138 - 77.4 =  60.6
 28 - 77.4 = -49.4
 59 - 77.4 = -18.4
 77 - 77.4 = - 0.4
 97 - 77.4 =  19.6
```

3. Pour chaque différence: trouvez la valeur carrée:

```
(-45.4)² = 2061.16
 (33.6)² = 1128.96
 (60.6)² = 3672.36
(-49.4)² = 2440.36
(-18.4)² =  338.56
(- 0.4)² =    0.16
 (19.6)² =  384.16
```

4. La variance est le nombre moyen de ces différences au carré:

```
(2061.16+1128.96+3672.36+2440.36+338.56+0.16+384.16) / 7 = 1432.2
```

Heureusement, NumPy a une méthode pour calculer la variance:

Exemple

Utilisez la `var()`méthode NumPy pour trouver la variance:

```
import numpy

speed = [32,111,138,28,59,77,97]

x = numpy.var(speed)

print(x)
```

Écart-type

Comme nous l'avons appris, la formule pour trouver l'écart type est la racine carrée de la variance:

```
√1432.25 = 37.85
```

Ou, comme dans l'exemple précédent, utilisez NumPy pour calculer l'écart type:

Exemple

Utilisez la `std()`méthode NumPy pour trouver l'écart type:

```
import numpy
```

```
speed = [32,111,138,28,59,77,97]

x = numpy.std(speed)

print(x)
```

Symboles

L'écart type est souvent représenté par le symbole Sigma: σ

La variance est souvent représentée par le symbole Sigma Square: σ^2

Résumé du chapitre

L'écart type et la variance sont des termes souvent utilisés dans le Machine Learning, il est donc important de comprendre comment les obtenir et le concept qui les sous-tend.

Apprentissage automatique - Train / Test

Évaluez votre modèle

Dans le Machine Learning, nous créons des modèles pour prédire l'issue de certains événements, comme dans le chapitre précédent où nous avons prédit les émissions de CO2 d'une voiture lorsque nous connaissions le poids et la taille du moteur.

Pour mesurer si le modèle est assez bon, nous pouvons utiliser une méthode appelée Train / Test.

Qu'est-ce que Train / Test

Train / Test est une méthode pour mesurer la précision de votre modèle.

Il s'appelle Train / Test car vous divisez l'ensemble de données en deux ensembles: un ensemble d'apprentissage et un ensemble de test.

80% pour la formation et 20% pour les tests.

Vous *entraînez* le modèle à l'aide de l'ensemble d'apprentissage.

Vous *testez* le modèle à l'aide de l'ensemble de test.

Former le modèle signifie *créer* le modèle.

Tester le modèle signifie tester la précision du modèle.

Commencez avec un ensemble de données

Commencez avec un ensemble de données que vous souhaitez tester.

Notre ensemble de données illustre 100 clients dans un magasin et leurs habitudes d'achat.

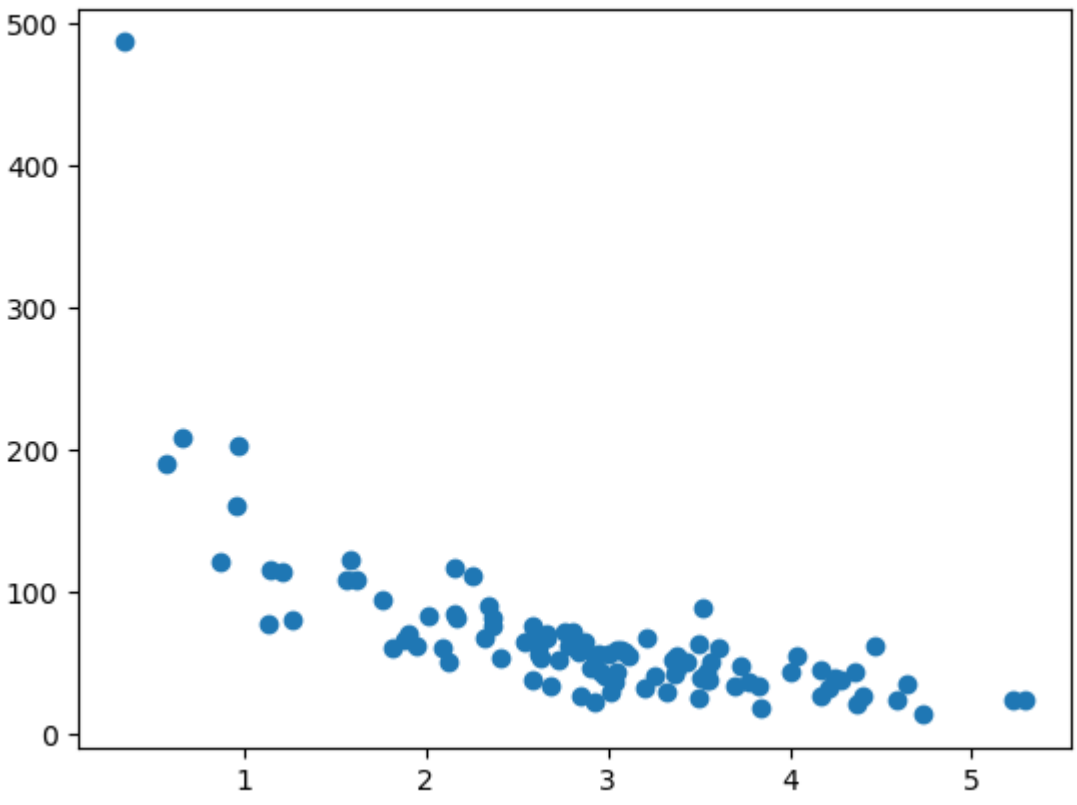
Exemple

```
import numpy
import matplotlib.pyplot as plt
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

plt.scatter(x, y)
plt.show()
```

Résultat:

L'axe des x représente le nombre de minutes avant d'effectuer un achat.

L'axe des y représente le montant d'argent dépensé pour l'achat.



Split en train / test

L' ensemble d' *apprentissage* doit être une sélection aléatoire de 80% des données d'origine.

L' ensemble de *test* doit être les 20% restants.

```
train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]
```

Afficher l'ensemble d'entraînement

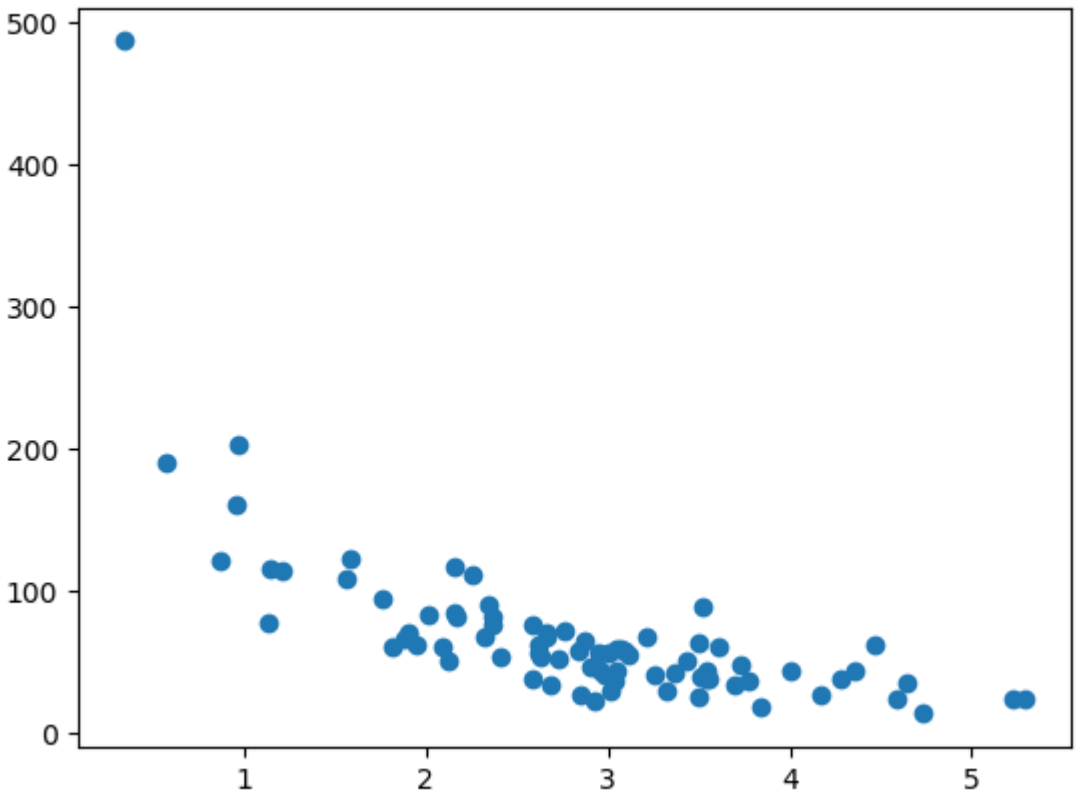
Affichez le même nuage de points avec l'ensemble d'apprentissage:

Exemple

```
plt.scatter(train_x, train_y)
plt.show()
```

Résultat:

Cela ressemble à l'ensemble de données d'origine, donc cela semble être une bonne sélection:



Afficher l'ensemble de test

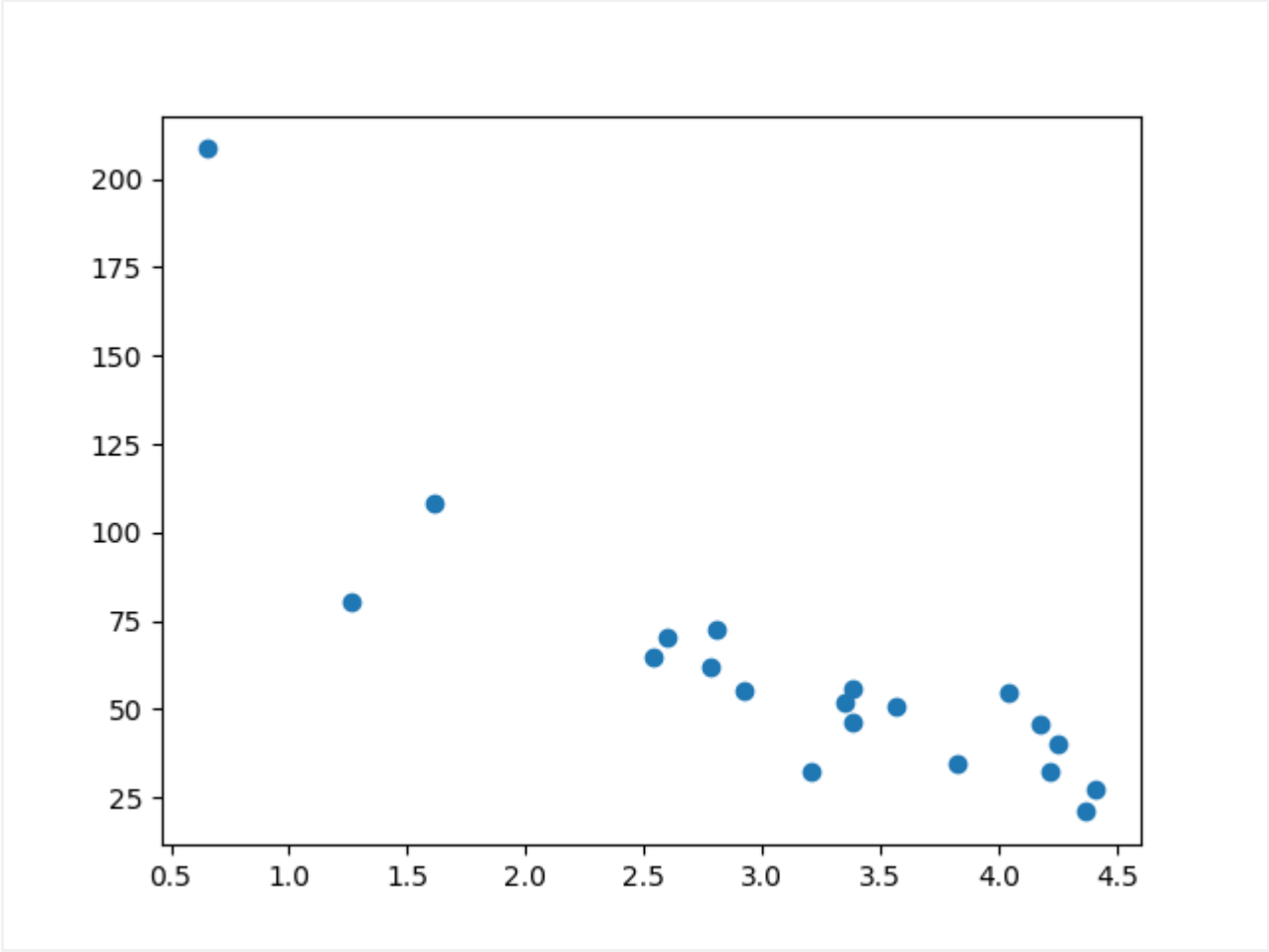
Pour nous assurer que l'ensemble de test n'est pas complètement différent, nous allons également examiner l'ensemble de test.

Exemple

```
plt.scatter(test_x, test_y)
plt.show()
```

Résultat:

L'ensemble de test ressemble également à l'ensemble de données d'origine:



Ajuster l'ensemble de données

À quoi ressemble l'ensemble de données? À mon avis, je pense que le meilleur ajustement serait une régression polynomiale , alors dessinons une ligne de régression polynomiale.

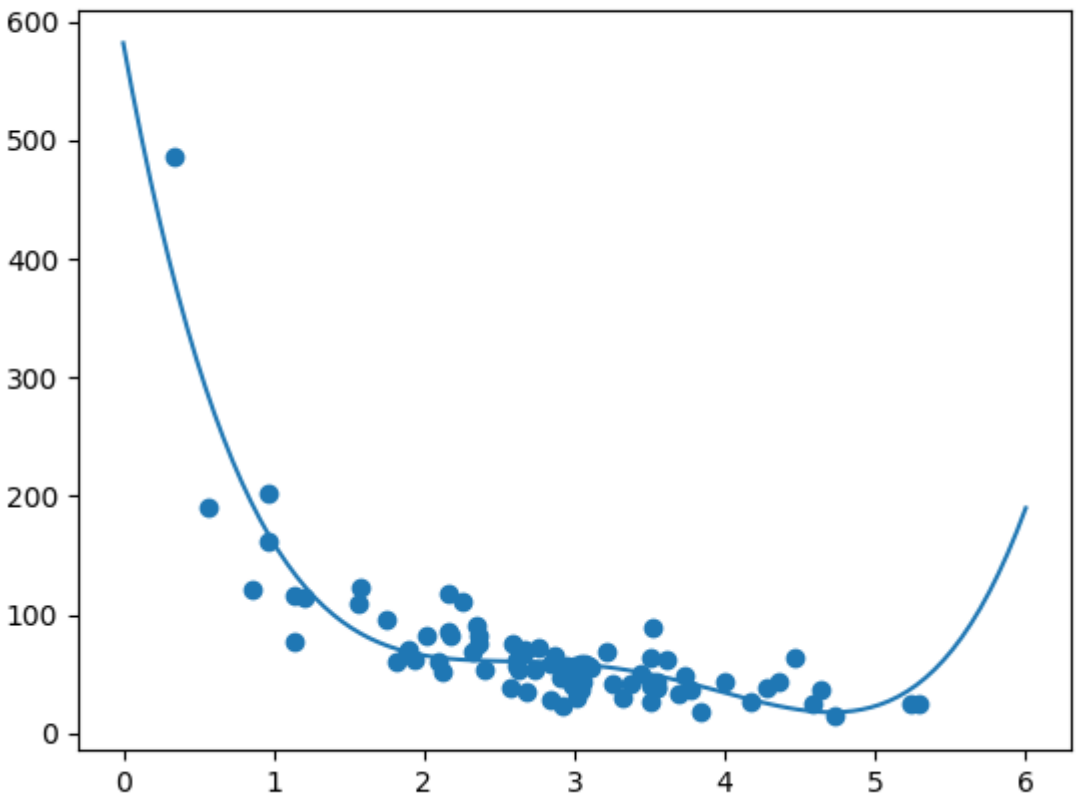
Pour tracer une ligne à travers les points de données, nous utilisons la `plot()`méthode du module matplotlib:

Exemple

Tracez une ligne de régression polynomiale à travers les points de données:

```
import numpy
import matplotlib.pyplot as plt
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]

mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

myline = numpy.linspace(0, 6, 100)

plt.scatter(train_x, train_y)
plt.plot(myline, mymodel(myline))
plt.show()
```

Résultat:



Le résultat peut étayer ma suggestion de l'ensemble de données ajustant une régression polynomiale, même si cela nous donnerait des résultats étranges si nous essayons de prédire des valeurs en dehors de l'ensemble de données. Exemple: la ligne indique qu'un client passant 6 minutes dans la boutique ferait un achat d'une valeur de 200. C'est probablement un signe de surapprentissage.

Mais qu'en est-il du score R-carré? Le score R-carré est un bon indicateur de l'adéquation de mon ensemble de données au modèle.

R2

Rappelez-vous R2, également connu sous le nom de R-carré?

Il mesure la relation entre l'axe des x et l'axe des y, et la valeur varie de 0 à 1, où 0 signifie aucune relation et 1 signifie totalement lié.

Le module sklearn a une méthode appelée `r2_score()` qui nous aidera à trouver cette relation.

Dans ce cas, nous aimerions mesurer la relation entre les minutes qu'un client reste dans la boutique et combien d'argent il dépense.

Exemple

Dans quelle mesure mes données d'entraînement s'intègrent-elles dans une régression polynomiale?

```
import numpy
from sklearn.metrics import r2_score
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]

mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

r2 = r2_score(train_y, mymodel(train_x))

print(r2)
```

Remarque: le résultat 0.799 montre qu'il existe une relation OK.

Apportez l'ensemble de test

Nous avons maintenant créé un modèle qui convient, du moins en ce qui concerne les données d'entraînement.

Maintenant, nous voulons tester le modèle avec les données de test également, pour voir si nous donne le même résultat.

Exemple

Trouvons le score R2 lors de l'utilisation des données de test:

```
import numpy
from sklearn.metrics import r2_score
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]

mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

r2 = r2_score(test_y, mymodel(test_x))

print(r2)
```

Remarque: Le résultat 0.809 montre que le modèle correspond également à l'ensemble de test, et nous sommes convaincus que nous pouvons utiliser le modèle pour prédire les valeurs futures.

Prédire les valeurs

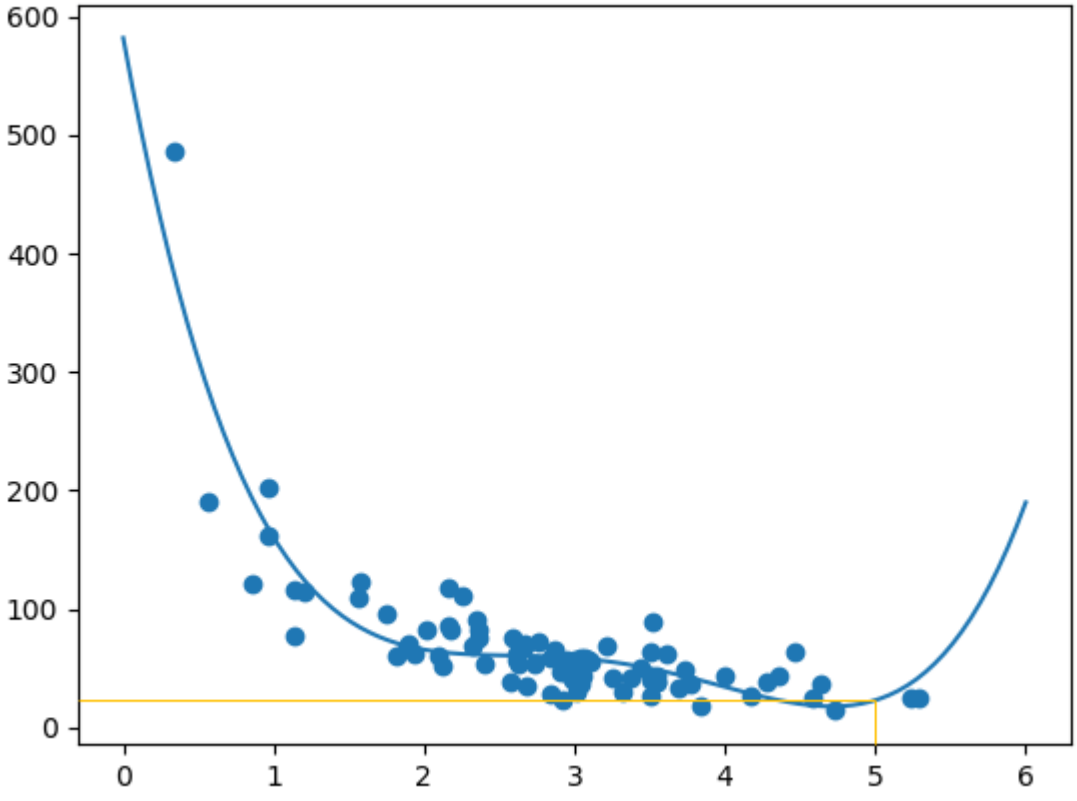
Maintenant que nous avons établi que notre modèle est OK, nous pouvons commencer à prédire de nouvelles valeurs.

Exemple

Combien d'argent un client acheteur dépensera-t-il s'il reste dans le magasin pendant 5 minutes?

```
print(mymodel(5))
```

L'exemple prédit que le client dépensera 22,88 dollars, comme semble correspondre au diagramme:


600
500
400
300
200
100
0
0
1
2
3
4
5
6