Machine Learning - Student Score Example

Quelle

https://stackabuse.com/linear-regression-in-python-with-scikit-learn/

Libraries

pandas

pandas ist eine Programmbibliothek zur Verarbeitung und Analyse von Daten in der Programmiersprache Python. Insbesondere enthält sie Datenstrukturen und Operatoren für den Zugriff auf numerische Tabellen und Zeitreihen.
https://pandas.pydata.org/

NumPy

NumPy ist eine Programmbibliothek für die Programmiersprache Python, die eine einfache Handhabung von Vektoren, Matrizen oder generell großen mehrdimensionalen Arrays ermöglicht. Neben den Datenstrukturen bietet NumPy auch effizient implementierte Funktionen für numerische Berechnungen an.
https://numpy.org/

Mathplotlib

Matplotlib ist eine Programmbibliothek für Python, die es erlaubt, mathematische Darstellungen aller Art anzufertigen.
https://matplotlib.org/

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Studenten-Datensatz

Datensatz mit Prüfungsergebnis in Abhängigkeit zum Lernaufwand

In [2]:
```
# Data Gathering
'''
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
import os
os.chdir("/content/drive/My Drive/ML")
!ls
'''
```

Out[2]:
```
'\nfrom google.colab import drive \ndrive.mount(\'/content/drive\',force_remount=Tru
e)\nimport os \nos.chdir("/content/drive/My Drive/ML")\n!ls\n'
```

In [3]:
```
dataset = pd.read_csv('student_scores.csv')

dataset.head()
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

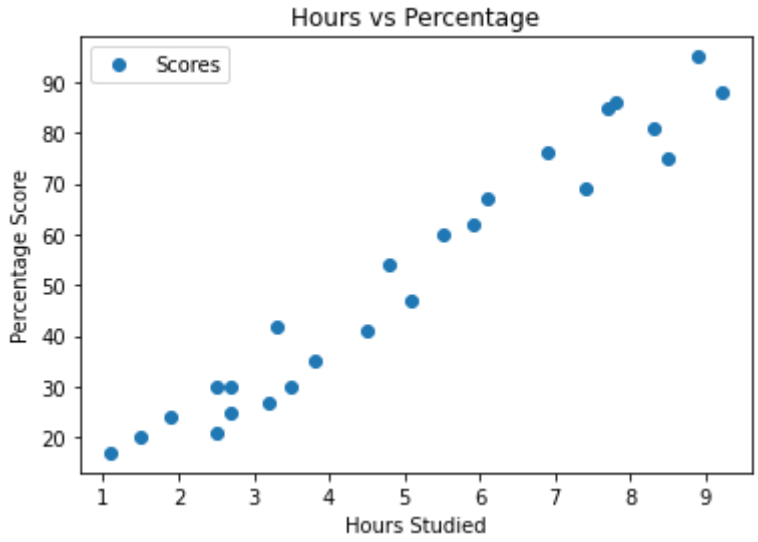
In [4]:
```
dataset.describe()
```

Out[4]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

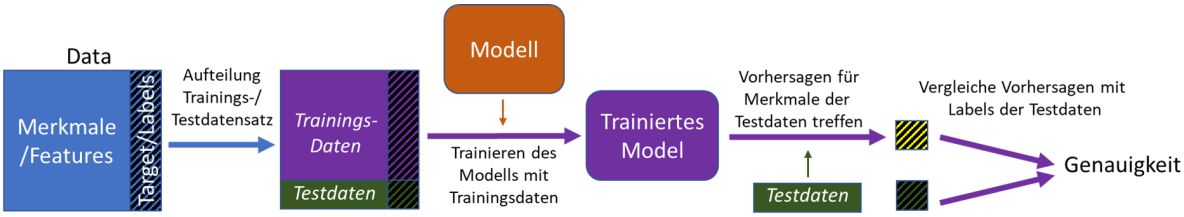
Mit Mathplotlib kann man schnell die Daten darstellen, um zu überlegen, welches Modell man anwenden könnte:

In [5]:
```
dataset.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



Machine Learning

Vorgehen



Scikit-learn

Scikit-learn ist eine freie Software-Bibliothek zum maschinellen Lernen für die Programmiersprache Python. Es bietet verschiedene Klassifikations-, Regressions- und Clustering-Algorithmen.
https://scikit-learn.org/stable/

Daten in Trainingsdaten und ein Testdaten aufspalten

Um ein Machine Learning Modell korrekt zu trainieren, wird ein Datensatz benötigt. Bei überwachten Lernverfahren wird dieser Datensatz in der Regel in mindestens zwei verschiedene Datensätze unterteilt: Training- und Testdaten. Oft gibt es noch einen weiteren Datensatz: Validierungsdaten.

Trainingsdaten
Ein Trainingsdatensatz ist ein Datensatz mit Beispielen (oder auch Zielvariablen genannt), die für das Lernen der Muster und Zusammenhänge in den Daten verwendet wird. Die Anpassung der Gewichte des Algorithmus wird über den Trainingsdatensatz antrainiert d.h. der Algorithmus lernt aus diesen Daten. Trainingsdaten mit Beispielen werden für Klassifikations- und Regressionsprobleme benötigt.

Testdaten
Die Testdaten werden bei dem Training nicht genutzt d.h. der Algorithmus kennt die Daten nicht

und kann diese nicht zum Lernen nutzen. Auch hier sind Beispiele bzw. Zielvariablen vorhanden, woran im Anschluss die Qualität des Modells gemessen werden kann.

In [6]:
```
X = dataset[['Hours']]
y = dataset['Scores']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

X_train
```

Out[6]:

	Hours
22	3.8
17	1.9
24	7.8
23	6.9
14	1.1
1	5.1
10	7.7
13	3.3
8	8.3
6	9.2
18	6.1
4	3.5
9	2.7
7	5.5
20	2.7
3	8.5
0	2.5
21	4.8
15	8.9
12	4.5

Lineare Regression

Die Lineare Regression beruht auf der Grundidee, einen Zusammenhang zwischen Variablen durch eine lineare Funktion zu beschreiben, im einfachsten Fall: y = m * x + b. Dabei wird die Summe der quadrierten Abstände zwischen den beobachteten Werten und der Regressionsgerade minimiert.
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Modell trainieren

In [7]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[7]:
```
▾ LinearRegression

LinearRegression()
```

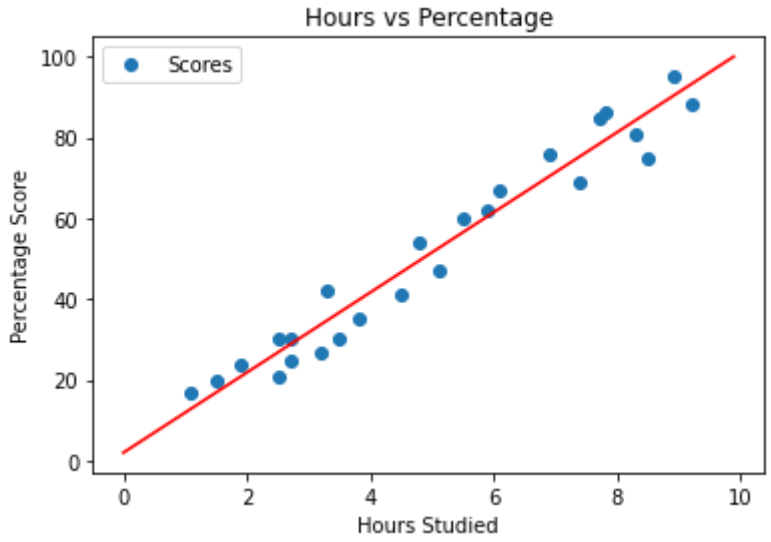
In [8]:
```
m = regressor.coef_[0]
b = regressor.intercept_

print("Scores =", "{:.2f}".format(m), "* Hours +", "{:.2f}".format(b))
```

```
Scores = 9.91 * Hours + 2.02
```

In [9]:
```
x = [0, (100 - b) / m ]
y = [b, 100]

dataset.plot(x='Hours', y='Scores', style='o')
plt.plot(x, y, color="red")
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



Vergleich Testdaten: Vorhersagen gegenüber tatsächlichen Werten

Nun kann man überprüfen, inwieweit die Daten im Testset mit den Vorhersagen übereinstimmen.

In [10]:
```
y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[10]:

	Actual	Predicted
5	20	16.884145
2	27	33.732261

	Actual	Predicted
19	69	75.357018
16	30	26.794801
11	62	60.491033

Evaluation des Modells

Messung der Genauigkeit des Modells.

In [11]:
```
from sklearn import metrics

print('Wurzel des mittleren Fehlerquadrats:', '{:.2f}'.format(np.sqrt(metrics.mean_s
print('Genauigkeit des Modells: ', '{:.2f}'.format(regressor.score(X_test, y_test)))
```

```
Wurzel des mittleren Fehlerquadrats: 4.65
Genauigkeit des Modells:  0.95
```