

توضیح کد پایتون



```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor

# Download and prepare the data
data_root = "https://github.com/ageron/data/raw/main/"
lifesat = pd.read_csv(data_root + "lifesat/lifesat.csv")

X = lifesat[["GDP per capita (USD)"]].values
y = lifesat[["Life satisfaction"]].values

# Visualize the data
lifesat.plot(kind='scatter', grid=True,
x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])

# Select a linear model
model = LinearRegression()
# linear regression model with k-nearest neighbors
# model = KNeighborsRegressor(n_neighbors=3)

# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[37_655.2]] # Cyprus' GDP per capita in 2020
print(f"Predicted life satisfaction for Cyprus = {model.predict(X_new)}") # output: [[6.30165767]]
X_new = np.linspace(23_500, 62_500, 100).reshape(-1, 1)
y_pred = model.predict(X_new)

plt.plot(X_new, y_pred, color="green", linewidth=2, label="Linear Regression")
plt.legend()
plt.show()
```

. برای من تمام سطر های این برنامه با جزئیات کامل توضیح بده

خیلی عالی، بیا خط به خط کد رو با جزئیات بررسی کنیم:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
```

- `matplotlib.pyplot` → برای رسم نمودار (اسکاتریلات، خطوط و غیره).
- `numpy` → برای محاسبات عددی و ساخت آرایه‌ها (مثل `linspace`).
- `pandas` → برای کار با داده‌ها به صورت DataFrame (خواندن CSV، دستکاری داده).
- `LinearRegression` → یک مدل رگرسیون خطی از کتابخانه `scikit-learn`.
- `KNeighborsRegressor` → یک مدل یادگیری ماشین دیگر بر پایه K-نزدیک‌ترین همسایه‌ها.

```
data_root = "https://github.com/ageron/data/raw/main/"
lifesat = pd.read_csv(data_root + "lifesat/lifesat.csv")
```

- `data_root` → آدرس ریشه‌ی داده‌ها در GitHub.
- `pd.read_csv(...)` → فایل `lifesat.csv` را از اینترنت می‌خواند و در یک DataFrame ذخیره می‌کند.
- نتیجه: متغیر `lifesat` شامل داده‌های کشورهای گسورهاست (ستون‌هایی مثل GDP و Life Satisfaction).

```
X = lifesat[["GDP per capita (USD)"]].values
y = lifesat[["Life satisfaction"]].values
```

- اینجا داده‌ها را به `X` (ویژگی) و `y` (هدف) جدا می‌کنیم.
- `X`: تولید ناخالص داخلی (GDP per capita).
- `y`: میزان رضایت از زندگی.
- `[[]]` باعث می‌شود خروجی به شکل آرایه ۲ بعدی `((n, 1))` باشد، چون مدل‌های `sklearn` به ورودی دوبعدی نیاز دارند.

```
lifesat.plot(kind='scatter', grid=True,
             x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])
```

- رسم نمودار پراکندگی (scatter plot) → هر نقطه یک کشور است.
- محور `x`: GDP per capita، محور `y`: Life satisfaction.
- `plt.axis(...)`: محدوده‌ی محورها را مشخص می‌کند →
 - `x` از 23,500 تا 62,500
 - `y` از 4 تا 9

```
model = LinearRegression()
# model = KNeighborsRegressor(n_neighbors=3)
```

- اینجا یک **مدل خطی** ساخته می‌شود.
- خط بعدی کامنت است → اگر آن را فعال کنیم، به جای رگرسیون خطی، از **KNN** با 3 همسایه استفاده خواهد شد.
- تفاوت:
- LinearRegression → یک خط مستقیم یاد می‌گیرد.
- KNN → براساس میانگین همسایه‌ها پیش‌بینی می‌کند (پیش‌بینی پله‌ای).

```
model.fit(X, y)
```

- داده‌ها را به مدل می‌دهیم تا آموزش ببیند.
- برای رگرسیون خطی: مدل ضریب خط (slope) و عرض از مبدأ (intercept) را پیدا می‌کند.
- برای KNN: مدل داده‌ها را ذخیره می‌کند و آماده می‌شود تا در زمان پیش‌بینی همسایه‌ها را پیدا کند.

```
X_new = [[37_655.2]] # Cyprus' GDP per capita in 2020
print(f"Predicted life satisfaction for Cyprus = {model.predict(X_new)}")
```

- X_new: یک مقدار جدید از GDP برای قبرس (سال 2020).
- model.predict(X_new): پیش‌بینی میزان رضایت از زندگی بر اساس مدل.
- خروجی → حدوداً `[[6.30165767]]`.
- چون sklearn خروجی را هم به صورت آرایه ۲ بعدی برمی‌گرداند.

```
X_new = np.linspace(23_500, 62_500, 100).reshape(-1, 1)
y_pred = model.predict(X_new)
```

- np.linspace(23_500, 62_500, 100) → تولید 100 عدد بین 23,500 تا 62,500 (برای ترسیم خط).
- reshape(-1, 1) → به آرایه ۲ بعدی با شکل (100, 1) تبدیل می‌شود (مناسب برای ورودی مدل).
- model.predict(X_new) → پیش‌بینی مقادیر y برای این نقاط.

```
plt.plot(X_new, y_pred, color="green", linewidth=2, label="Linear Regression")
plt.legend()
plt.show()
```

- `plt.plot` → خط سبزی روی نمودار رسم می‌کند که پیش‌بینی مدل خطی را نشان می‌دهد.
- `linewidth=2` → خط ضخیم‌تر.
- `label="Linear Regression"` → برای نمایش در `legend`.
- `plt.legend()` → توضیح مدل (لیبل) را به نمودار اضافه می‌کند.
- `plt.show()` → نمودار کامل را نمایش می‌دهد.

🔑 خلاصه‌ی اجرا:

1. داده‌ها (GDP و Life Satisfaction) از GitHub خوانده می‌شوند.
2. داده‌ها روی نمودار پراکندگی رسم می‌شوند.
3. مدل رگرسیون خطی آموزش داده می‌شود.
4. رضایت از زندگی برای قبرس پیش‌بینی می‌شود (~6.3).
5. خط پیش‌بینی روی نمودار رسم می‌شود.

می‌خواهی من همین برنامه را با مدل **KNeighborsRegressor (3 همسایه)** هم برای توضیح بدهم تا تفاوت خروجی‌ها را ببینی؟



تولید 100 عدد بین 23,500 تا 62,500 (برای ترسیم خط) → `np.linspace(23_500, 62_500, 100)`

چرا این کارو باید انجام بدهیم؟

سؤال خیلی خوبی پرسیدی 🔥

بذار با یک مثال ساده توضیح بدم:

چرا از `np.linspace` استفاده می‌کنیم؟

وقتی یک مدل مثل **رگرسیون خطی** یاد می‌گیرد، در اصل یک **تابع** (یا رابطه ریاضی) بین GDP و Life satisfaction پیدا می‌کند.

حالا برای اینکه این خط (یا منحنی) را روی نمودار رسم کنیم، باید از مدل بخواهیم برای چندین نقطه‌ی مختلف از GDP مقدار پیش‌بینی‌شده‌ی Life satisfaction را حساب کند.

- اگر فقط برای یک نقطه (مثلاً قبرس) پیش‌بینی کنیم → فقط یک نقطه به دست می‌آید، نه یک خط.
- برای کشیدن یک خط صاف، نیاز داریم چندین نقطه (x و y) داشته باشیم و آن‌ها را به هم وصل کنیم.

پس نقش `np.linspace` چیه؟

```
X_new = np.linspace(23_500, 62_500, 100).reshape(-1, 1)
```

- `np.linspace(23_500, 62_500, 100)` → **صد مقدار** بین 23,500 و 62,500 تولید می‌کند (به طور یکنواخت).
- این مقادیر، همان مقادیر GDP هستند که می‌خواهیم خط مدل را روی آن‌ها محاسبه کنیم.
- بعد با `model.predict(X_new)`، برای هر کدام یک مقدار Life satisfaction پیش‌بینی می‌کنیم.
- در نهایت 100 نقطه‌ای (x, y) داریم که اگر به هم وصل شوند، خط پیش‌بینی مدل را تشکیل می‌دهند.

اگر این کار را نکنیم چی میشه؟

- اگر فقط برای چند مقدار محدود مثل `X = [[30_000], [40_000], [50_000]]` پیش‌بینی کنیم → فقط 3 نقطه خواهیم داشت و خط درست کشیده نمی‌شود.
- با `linspace` ما یک محدوده‌ی پیوسته (از 23,500 تا 62,500) را پوشش می‌دهیم، پس خط صاف و کامل روی نمودار رسم می‌شود.

🔑 خلاصه:

`linspace` برای این استفاده می‌شود که:

1. مقادیر پیوسته‌ی GDP در یک بازه تولید کنیم.
2. مدل برای آن‌ها پیش‌بینی کند.
3. پیش‌بینی‌ها را به صورت یک خط (یا منحنی) رسم کنیم.

می‌خوای من همین قسمت رو با **یه نمودار کوچک آزمایشی** برات رسم کنم تا فرق بین پیش‌بینی با 3 نقطه و پیش‌بینی با 100 نقطه رو ببینی؟



به آرایه ۲ بعدی با شکل (1, 100) تبدیل می‌شود (مناسب برای ورودی مدل) → `reshape(-1, 1)`.

اینم کامل توضیح بده با استدلال ؟

خیلی خوب، بذار این موضوع رو قدم به قدم باز کنم 🙌

♦ مفهوم شکل (shape) آرایه در NumPy

در NumPy هر آرایه یک **شکل (shape)** دارد:

- `(, n)` → آرایه‌ی یک بعدی با `n` عنصر.
- `(n, 1)` → آرایه‌ی دوبعدی با `n` سطر و ۱ ستون.

مثال:

```
import numpy as np

a = np.array([10, 20, 30, 40])
print(a.shape)    # (4,)

b = a.reshape(-1, 1)
print(b.shape)    # (4, 1)
```

- **a** فقط یک بردار یک بعدی است.
- **b** یک ماتریس 4×1 است (۴ نمونه، هرکدام ۱ ویژگی).

♦ چرا **scikit-learn** به $(n, 1)$ نیاز دارد؟

مدل‌های **sklearn** همیشه انتظار دارند داده‌ها این‌طور باشند:

scss

```
(n_samples, n_features)
```

- **n_samples** → تعداد نمونه‌ها (مثلاً 100 کشور).
- **n_features** → تعداد ویژگی‌ها (مثلاً GDP).

پس اگر GDP را فقط به صورت $(100,)$ بدهیم، مدل نمی‌فهمد این 100 تا مقدار:

- 100 نمونه با 1 ویژگی هستند
یا
- یک نمونه با 100 ویژگی؟

برای همین **sklearn** خطا می‌دهد:

pgsql

```
Expected 2D array, got 1D array instead
```

♦ نقش **reshape(-1, 1)**

وقتی از **reshape(-1, 1)** استفاده می‌کنیم:

- **1** → به NumPy می‌گوید «خودت تعداد سطرها را حساب کن».
- **1** → یعنی یک ستون (یک ویژگی).

مثلاً:

python

```
X = np.linspace(23500, 62500, 100)
print(X.shape)    # (100,)
```

```
X_resaped = X.reshape(-1, 1)
print(X_resaped.shape) # (100, 1)
```

حالا:

- $(100,)$ → فقط یک آرایه‌ی تخت (flat).
- $(1, 100)$ → مناسب برای ورودی مدل (100 کشور، 1 ویژگی).

جمع‌بندی 🔑

- `reshape(-1, 1)` داده‌ها رو از یک بُعدی → به شکل استاندارد دوبعدی تبدیل می‌کنه.
- بدونش، `sklearn` نمی‌دونه داده‌ها رو چطور تفسیر کنه.
- باهاش، هر سطر یک **نمونه** و هر ستون یک **ویژگی** است.

می‌خوای من به **نمودار مقایسه‌ای** بکشم که نشون بده داده به شکل $(100,)$ چه شکلیه و به شکل $(100, 1)$ چه تفاوتی پیدا می‌کنه؟