pesikj Rok 2024 last month

2727 lines (2727 loc) · 870 KB

Preview Code Blame Raw

Lekce 4

Korelace

V reálném světě se často stává, že nějaká skutečnost má vliv na něco jiného. Například čas strávený studiem má vliv na body. V souboru Student_Marks.csv jsou data o průměrné době, kterou uživatelé strávili studiem on-line kurzu, a průměrným počtem bodů, které z kurzů dostali. Podívejme se, jak silný je vliv doby strávené studiem na bodový výsledek.

In [2]:
```
import pandas as pd
import seaborn as sns
from scipy import stats
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt


data = pd.read_csv("Student_Marks.csv")
data.head()
```

Out[2]:

	number_courses	time_study	Marks
0	3	4.508	19.202
1	4	0.096	7.734
2	4	3.133	13.811
3	6	7.909	53.018
4	8	7.811	55.299

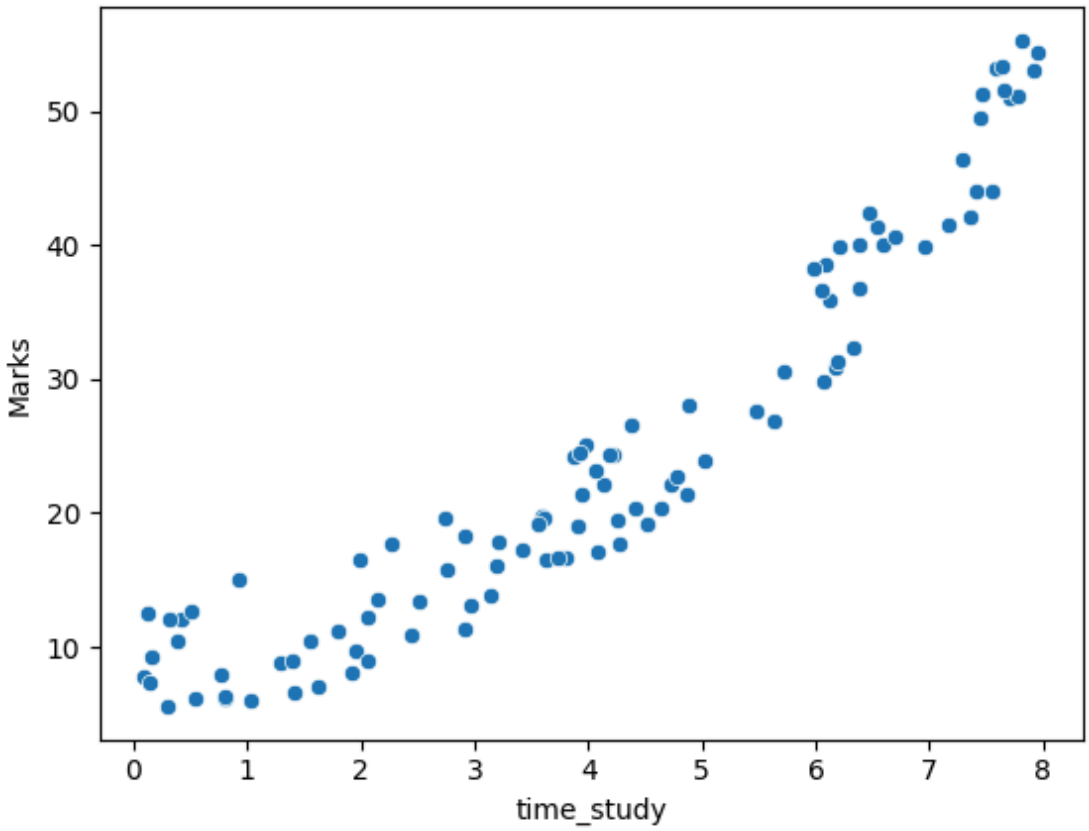
K zobrazení použijeme bodový graf (*scatter plot*). Na vodorovné ose máme průměrný čas strávený studiem a na svislé ose průměrný počet bodů. Vidíme, že známka má tendenci růst s tím, jak roste čas strávený studiem. Současně je patrný jistý vliv náhody. Takové závislosti se říká stochastická závislost (*stochastic dependence*). Dále platí, že závislost je lineární (*linear*), tj. kdybychom ho chtěly popsat pomocí matematické funkce, mohli bychom použít přímku.

In [3]:
```
sns.scatterplot(data=data, x="time_study", y="Marks")
```

```
c:\Users\jiri.pesik.HULD\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is depr
ecated and will be removed in a future version. Use isinstance(dtype, Catego
ricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\jiri.pesik.HULD\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is depr
ecated and will be removed in a future version. Use isinstance(dtype, Catego
ricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
if pd.api.types.is_categorical_dtype(vector):
```

Out[3]: <Axes: xlabel='time_study', ylabel='Marks'>



Takové závislosti říkáme **korelace** (*correlation*) a to, jak je závislost silná, můžeme popsat pomocí **korelačního koeficientu** (*correlation coefficient*). Pro jeho hodnoty platí následující:

- Hodnoty blízko +1 znamenají silnou přímou lineární závislost, tj. hodnoty v obou sloupcích rostou současně.
- Hodnoty blízko 0 znamenají lineární nezávislost.
- Hodnoty blízko -1 znamenají silnou nepřímou lineární závislost, tj. jedna hodnota roste a současně druhá klesá.

Příklady přímé závislosti: čas strávený studiem a výsledek v testu, délka tréninku a výsledek v závodě, délka praxe a výše mzdy atd. Příklady nepřímé závislosti: množství vypitého alkoholu a kognitivní schopnosti, zimní teploty a spotřeba energie na vytápění.

Hodnotu korelace zjistíme pomocí metody `corr()` pro zvolenou tabulku.

In [4]:
```
data.corr()
```

Out[4]:

	number_courses	time_study	Marks
number_courses	1.000000	0.204844	0.417335
time_study	0.204844	1.000000	0.942254
Marks	0.417335	0.942254	1.000000

Korelace automaticky neznamená, že obě veličiny se vzájemně ovlivňují.

Uvažujme například počet turistů a počet komárů v kempu u přehrady. Obě veličiny jsou sice korelované, ale v důsledku počasí jako třetího vlivu. Pokud bychom například kemp zavřeli a počet turistů by klesl na nulu, počet komárů to nijak neovlivní.

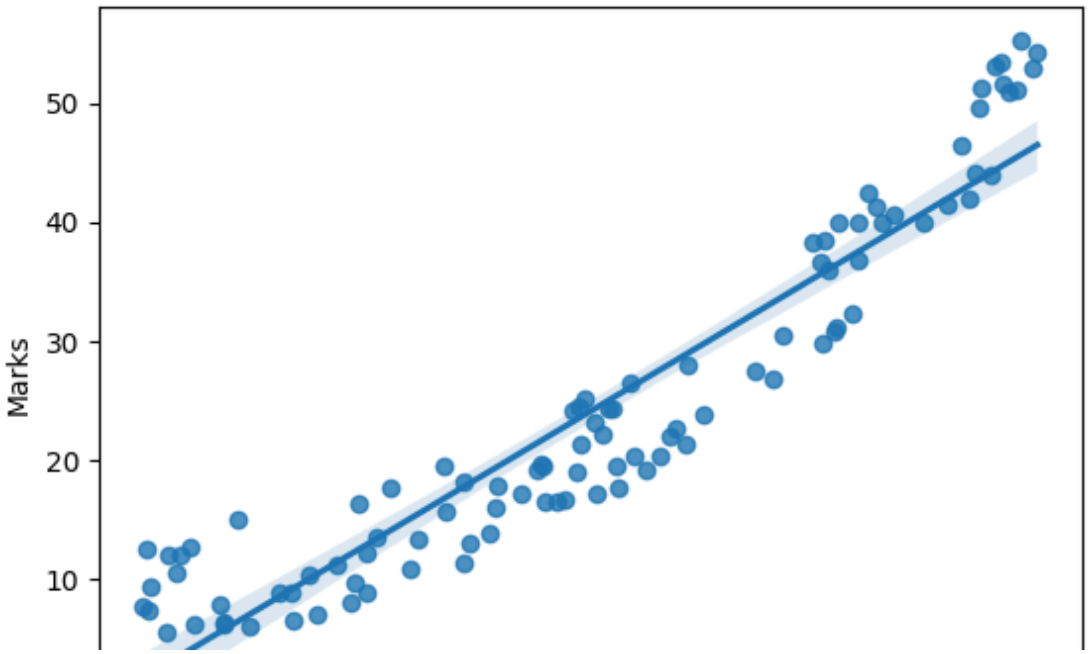
Podobně můžeme "odhalit" korelaci, která je ve skutečnosti náhodná. Doslova legendárním příkladem se stala korelace mezi počtem lidí, kteří se utopili v důsledku pádu do bazénu, a počtem filmů, ve kterých hrál Nicolas Cage. Z logiky věci je jasné, že tyto veličiny se vzájemně nijak neovlivňují, přesto bychom mezi nimi našli vysokou korelaci.

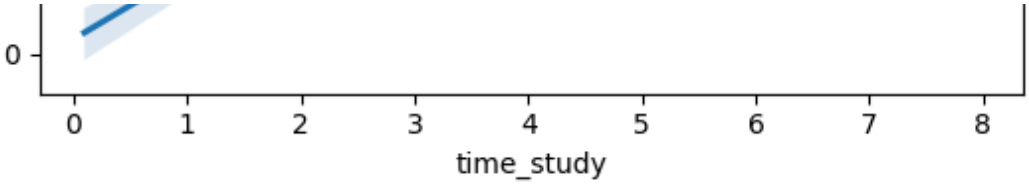
Regrese

Samotná informace o tom, že existuje statisticky významný vztah mezi obytnou plochou domu a jeho cenou, sice může být zajímavá, ale můžeme zjistit více. K tomu můžeme využít regresi. Regrese je nástroj, který umí vztah mezi dvěma proměnnými popsat. Abychom si pod slovem "popsat" dokázali něco představit, využijeme graf. Využijeme opět modul `seaborn`, tentokrát vygenerujeme graf pomocí funkce `regplot()`. U regrese vždy rozlušujeme mezi **závislou** (**vysvětlovanou**) a **nezávislou** (**vysvětlující**) proměnnou. Závislou proměnnou umísťujeme na svislou osu (*y*) a nezávislou vodorovnou osu (*x*). V našem případě je nezávislou proměnnou čas strávený studiem a závislou počet bodů. Tvrdíme totiž, že čas strávený studiem ovlivňuje počet bodů, tj. počet bodů vysvětlujeme pomocí času stráveného studiem.

Musíme si uvědomit, že výslednou známku ovlivňují i další vlivy - například únava studentů a studentek v době psaní testu, štěstí na otázky atd. Z toho důvodu neleží všechny body na regresní křivce, ale pohybují se kolem ní. Pokud jsou nad ní, pak byl skutečný výsledek studenta nebo studentky lepší, než kolik by predikoval náš model. Pokud je bod pod ní, pak je skutečný výsledek horší.

In [5]:
```
g = sns.regplot(data, x="time_study", y="Marks")
```





Pomocí této funkce dokážeme predikovat, kolik bodů student nebo studentka kurzu získá, a to na základě počtu hodin, které strávil(a) studiem.

Tato funkce je označovaná jako "lineární" a k jejímu vykreslení potřebujeme znát dvě hodnoty:

- První je hodnota, která určuje, kde leží průsečík s osou *y*. V našem případě jde o hodnotu, která udává počet bodů, který by získal(a) potenciální student(ka), který se na test vůbec nepřipravoval(a) (např. kolik otázek je možné si tipnout).
- Druhá je hodnota, která udává sklon funkce. Čím bude hodnota vyšší, tím více skloněná funkce bude. V našem případě toto číslo určí, kolik dodatečných bodů je možné získat hodinou studia navíc.

K zobrazení těchto hodnot můžeme použít modul *statmodels*. Ten zobrazí velkou tabulku se spoustou čísel, nás však budou zajímat pouze některá.

In [6]:
```
formula = "Marks ~ time_study"
mod = smf.ols(formula=formula, data=data)
res = mod.fit()
res.summary()
```

Out[6]: OLS Regression Results

Dep. Variable:	Marks	**R-squared:**	0.888
Model:	OLS	**Adj. R-squared:**	0.887
Method:	Least Squares	**F-statistic:**	775.8
Date:	Thu, 26 Oct 2023	**Prob (F-statistic):**	2.36e-48
Time:	12:46:18	**Log-Likelihood:**	-298.21
No. Observations:	100	**AIC:**	600.4
Df Residuals:	98	**BIC:**	605.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.2239	0.962	1.272	0.206	-0.686	3.133
time_study	5.6888	0.204	27.853	0.000	5.283	6.094

Omnibus:	7.504	**Durbin-Watson:**	1.757
Prob(Omnibus):	0.023	**Jarque-Bera (JB):**	4.633
Skew:	0.357	**Prob(JB):**	0.0986
Kurtosis:	2.225	**Cond. No.**	9.73

Kurtosis:	2.225	**Cond. No.**	9.72

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Podívejme se nejprve na dvě čísla (koeficienty - *coefficients*), která potřebujeme k nakreslení naší funkce:

- V řádku `intercept` máme hodnotu, která určuje, kde funkce protne se svislou osou.
- V řádku `time_study` máme hodnotu, která udává sklon funkce. Hodnota 5.6888 nám říká, že jedna hodina studia navíc přinese v průměru 5.6888 bodů navíc.

Pokud bychom chtěli odhadnout počet bodů na základě hodin strávených studiem, můžeme použít metodu `predict()`. K tomu si načteme data z tabulky `Study_Marks_to_estimate.csv`, kde nám chybí výsledná známka. Tu my odhadneme a odhady vložíme do dat jako nový sloupec.

In [7]:
```
data_to_predict = pd.read_csv("Study_Marks_to_estimate.csv")
data_to_predict["MarksPredicted"] = res.predict(data_to_predict)
data_to_predict.head()
```

Out[7]:

	number_courses	time_study	MarksPredicted
0	3	5.5	32.511985
1	4	8.0	46.733861
2	1	3.0	18.290109

Vyhodnocení kvality modelu

Regresní model nevysvětlí data dokonale. Jak jsme si již řekli, na vysvětlovanou proměnnou působí další vlivy, které v datech nemáme. Regresní modely se mezi sebou liší podle toho, jak dobře vysvětlovanou proměnnou dokážou vysvětlit. Abychom tuto skutečnost dokázali vysvětlit, vznikl ukazatel označovaný jako koeficient determinace (*coefficient of determination*, *R-squared*). Ten říká, kolik procent variability (různorodosti) vysvětlované proměnné (v našem případě známky z testu) dokážeme pomocí našeho modelu vysvětlit. Koeficient determinace je číslo mezi 0 a 1 a platí, že čím vyšší koeficient determinace je, tím lépe náš model naše data popisuje.

Model se v matematických vzorcích často značí R^2, v naší tabulce je označen jako `R-squared`. Vidíme, že náš model má koeficient determinace 0.88, dokáže tedy vysvětlit přibližně 88 % variability známek z online kurzů.

Test hypotézy o statistické významnosti koeficientu

S regresí souvisí řada testů statistických hypotéz. Jedním z nich je test statistické významnosti regresního koeficientu. Ten se hodí hlavně v případě, kdy máme koeficientů více a zajímá nás, které má smysl v modelu používat.

Test má následující hypotézy:

- H0: Koeficient je statisticky nevýznamný.
- H1: Koeficient je statisticky významný.

Pokud je p-hodnota testu méně než 0.05, můžeme tedy koeficient označit jako statisticky významný. p-hodnotu testu najdeme ve sloupci `P>|t|` . p-hodnotu máme pro každý koeficient zvlášť. V našem případě platí, že koeficient `time_study` je statisticky významný a koeficient `Intercept` je statisticky nevýznamný.

Odlehlá pozorování

V případě regrese často můžeme narazit na problémy, které souvisí s daty, které máme k dispozici. Uvažujme například dataset o cenách nemovitostí ze souboru house_prices.csv. V nich máíme data o různých domech, které byly nabízeny k prodeji na americkém trhu.

Na začátku nás budou zajímat dva sloupce:

- `GrLivArea` - obytná plocha domu.
- `SalePrice` - cena domu v dolarech.

In [8]:
```
data = pd.read_csv("house_prices.csv")
data.head()
```

Out[8]:

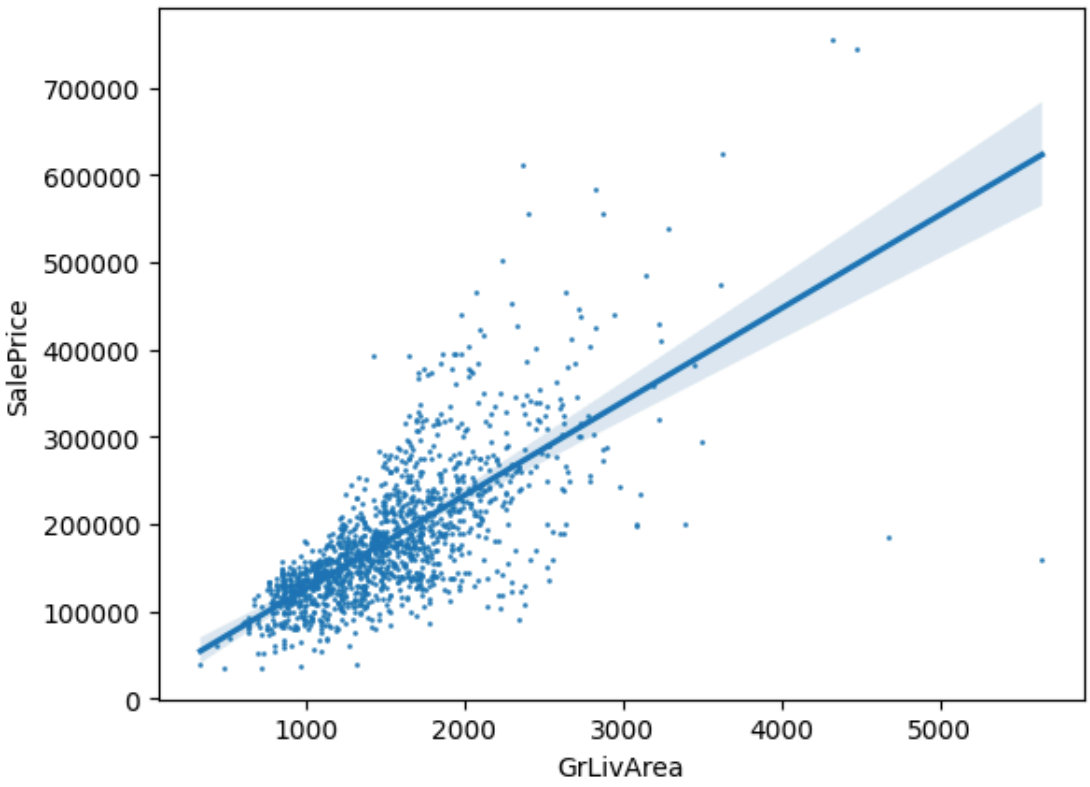
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lan
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	

5 rows × 81 columns

Vytvoříme si regresní graf. Protože je dat hodně, zmenšíme velikost bodů na 1 s využitím parametru `scatter_kws` .

Na grafu jsou vidět body, které označujeme jako odlehlá pozorování. Jedná se o domy, které jsou výrazně levnější nebo výrazně dražší ve srovnání s podobně velkými domy.

In [9]:
```
g = sns.regplot(data, x="GrLivArea", y="SalePrice", scatter_kws={"s": 1})
```

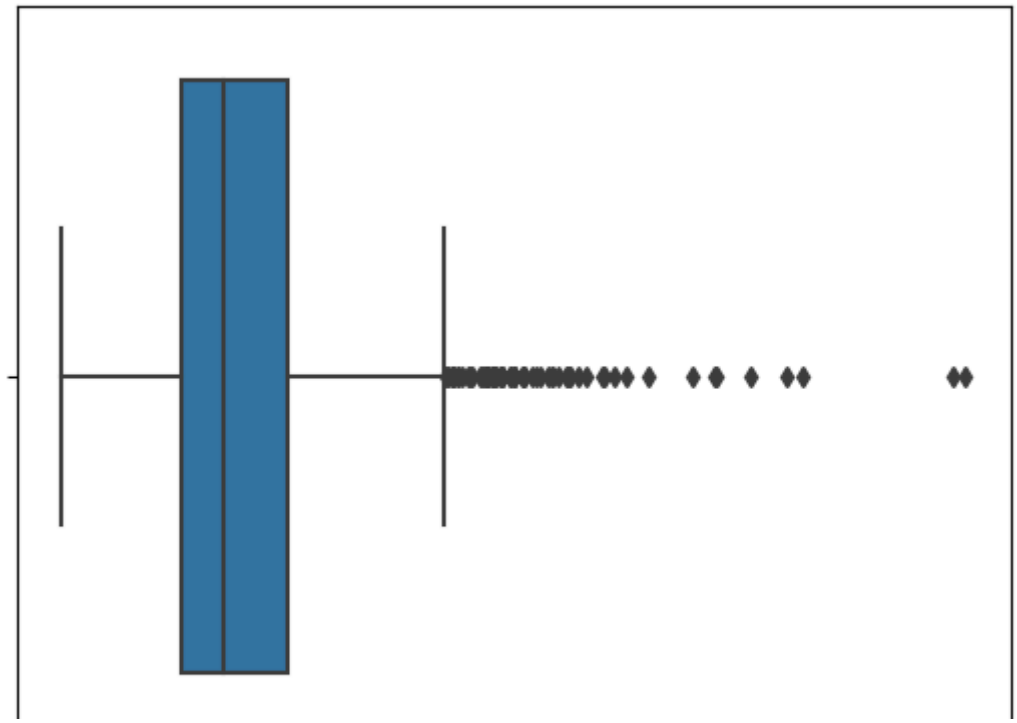


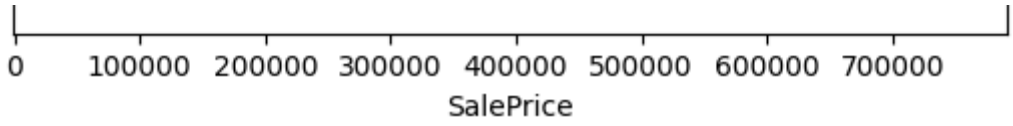
Můžeme použít i krabicový graf, abychom se podívali, v jaké cenové relaci se nechází cena domu.

In [10]:
```
sns.boxplot(data, x="SalePrice")
```

```
c:\Users\jiri.pesik.HULD\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[10]:
```
<Axes: xlabel='SalePrice'>
```





Odlehlá pozorování vadí především z důvodu, že mají tendenci "přitahovat" si k sobě regresní funkci. To pak může regresní funkci vychýlit a ona bude zobrazovat chybné výsledky.

Teoreticky je možné, že bychom zdánlivě velmi vysokou nebo nízkou cenu domu dokázali vysvětlit pomocí dalších sloupců - například může být nový, ve vynikajícím stavu, má velký pozemek, je na dobrém místě atd. Pokud ale do dat zařadíme více sloupců, nemůžeme už si data zobrazit graficky.

V rámci regrese existuje měřítko, které říká, jak moc jeden konkrétní bod ovlivňuje regresní funkci. Takovýmu měřítku říkáme Cookova vzdálenost (*Cook's distance*). Cookova vzdálenost jednoho uvažuje, o kolik by se regresní funkce "posunula", pokud bychom tento bod vynechali (anglicky toto ozančujeme jako *leverage*) a vzdálenost bodu od regresní funkce.

In [11]:

```
from statsmodels.stats.outliers_influence import OLSInfluence

formula = "SalePrice ~ OverallQual + GrLivArea + TotalBsmtSF + YearBuilt +
mod = smf.ols(formula=formula, data=data)
results = mod.fit()
influence = OLSInfluence(results)
data['Cook Distance'] = influence.cooks_distance[0]
data.head()
```

Out[11]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lar
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	

5 rows × 82 columns

Jednoduché pravidlo (*rule of the thumb*) doporučuje odstranit hodnoty s Cookovou vzdáleností vyšší než 1.

In [12]:

```
data = data[data['Cook Distance'] < 1]
mod = smf.ols(formula=formula, data=data)
```

```
res = mod.fit()
res.summary()
```

Out[12]:

OLS Regression Results

Dep. Variable:	SalePrice	**R-squared:**	0.801
Model:	OLS	**Adj. R-squared:**	0.800
Method:	Least Squares	**F-statistic:**	833.0
Date:	Thu, 26 Oct 2023	**Prob (F-statistic):**	0.00
Time:	12:46:19	**Log-Likelihood:**	-17355.
No. Observations:	1459	**AIC:**	3.473e+04
Df Residuals:	1451	**BIC:**	3.477e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-8.629e+05	9.08e+04	-9.508	0.000	-1.04e+06	-6.85e+05
OverallQual	1.905e+04	1073.847	17.740	0.000	1.69e+04	2.12e+04
GrLivArea	63.3108	3.420	18.513	0.000	56.602	70.019
TotalBsmtSF	37.7036	3.047	12.372	0.000	31.726	43.681
YearBuilt	399.3179	47.693	8.373	0.000	305.764	492.872
FullBath	-9607.9940	2674.084	-3.593	0.000	-1.49e+04	-4362.511
HalfBath	-1784.6415	2422.955	-0.737	0.462	-6537.511	2968.228
GarageArea	43.9541	5.742	7.654	0.000	32.690	55.218

Omnibus:	413.680	**Durbin-Watson:**	1.981
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	14644.507
Skew:	0.622	**Prob(JB):**	0.00
Kurtosis:	18.471	**Cond. No.**	2.70e+05

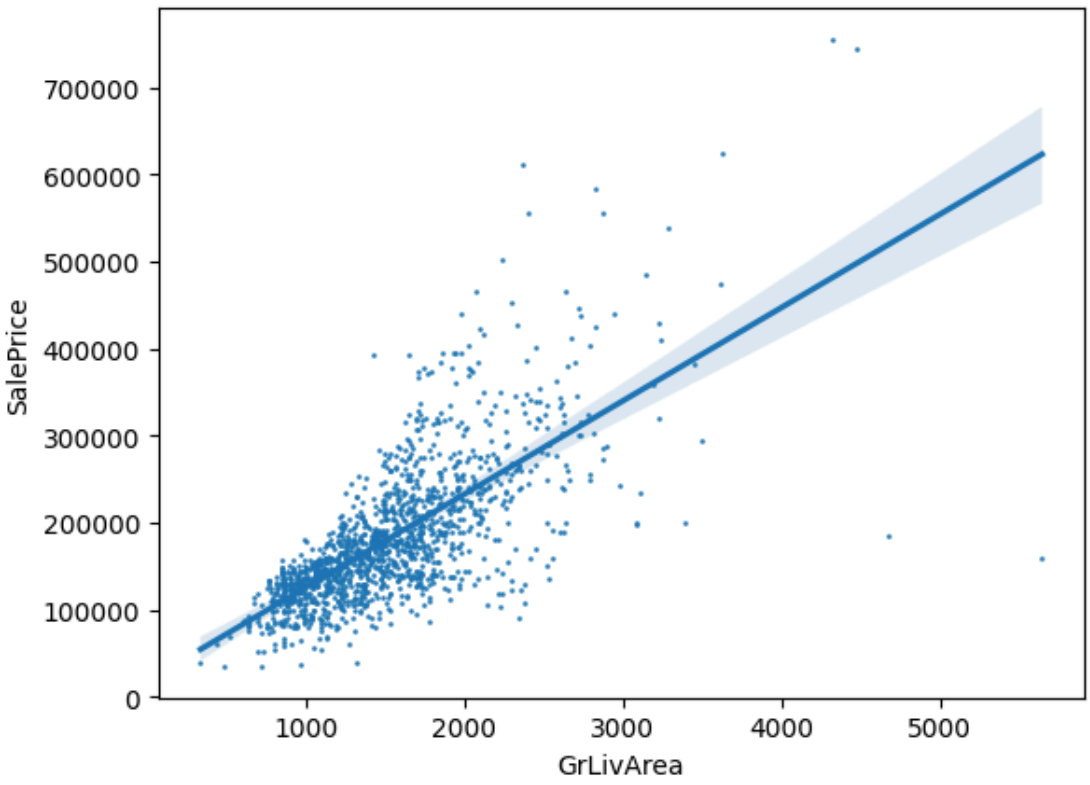
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.7e+05. This might indicate that there are strong multicollinearity or other numerical problems.

Čtení na doma: Robustní regrese

Regresním modelům vadí i další věc, která se označuje strašidelným pojmem heteroskedasticita (*heteroscedasticity*). Tím je myšleno, že data mají stejnou variabilitu. Vraťme se k regresnímu grafu. Zde vidíme, že čím je dům větší, tím jsou

ceny více rozptýlené. Opět platí, že se díváme pouze na dvě proměnné, i když ve výsledném modelu jich máme více.

In [13]:
```
data = pd.read_csv("house_prices.csv")
g = sns.regplot(data, x="GrLivArea", y="SalePrice", scatter_kws={"s": 1})
```



Pokud máme podezření na heteroskedasticitu, můžeme použít jiný typ regrese, která je označovaná jako robustní regrese. Ta přiřazuje jednotlivým pozorováním různé váhy, čímž vykompenzuje vliv měnící se variability dat. Robustní regrese je alternativním řešením i pro odlehlá pozorování, protože dává menší váhu odlehlejším pozorováním a tím zeslabí jejich vliv na regresní funkci.

In [14]:
```
formula = "SalePrice ~ OverallQual + GrLivArea + TotalBsmtSF + YearBuilt +
mod = smf.rlm(formula=formula, data=data)
res = mod.fit()
res.summary()
```

Out[14]: Robust linear Model Regression Results

Dep. Variable:	SalePrice	**No. Observations:**	1460
Model:	RLM	**Df Residuals:**	1452
Method:	IRLS	**Df Model:**	7
Norm:	HuberT		
Scale Est.:	mad		
Cov Type:	H1		
Date:	Thu, 26 Oct 2023		
Time:	12:46:20		

No. Iterations:	6					

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-8.516e+05	6.52e+04	-13.054	0.000	-9.8e+05	-7.24e+05
OverallQual	1.68e+04	766.452	21.916	0.000	1.53e+04	1.83e+04
GrLivArea	58.1283	2.431	23.913	0.000	53.364	62.893
TotalBsmtSF	31.4140	2.127	14.766	0.000	27.244	35.584
YearBuilt	405.2442	34.289	11.818	0.000	338.039	472.450
FullBath	-7314.6164	1910.839	-3.828	0.000	-1.11e+04	-3569.440
HalfBath	-3183.1509	1742.865	-1.826	0.068	-6599.103	232.801
GarageArea	43.4179	4.130	10.513	0.000	35.323	51.513

If the model instance has been used for another fit with different fit parameters, then the fit options might not be the correct ones anymore .

Přehled různých metod, jak snížit váhu koeficientů, jsou zde.

Transformace dat

Podívejme se ještě na další dataset. Ten obsahuje data o střední délce života (*life expectancy*) v různých zemích.

In [15]:
```
data = pd.read_csv("Life-Expectancy-Data-Updated.csv")
data.head()
```

Out[15]:

	Country	Region	Year	Infant_deaths	Under_five_deaths	Adult_mortality	Alc
0	Turkiye	Middle East	2015	11.1	13.0	105.8240	
1	Spain	European Union	2015	2.7	3.3	57.9025	
2	India	Asia	2007	51.5	67.9	201.0765	
3	Guyana	South America	2006	32.8	40.5	222.1965	
4	Israel	Middle East	2012	3.4	4.3	57.9510	

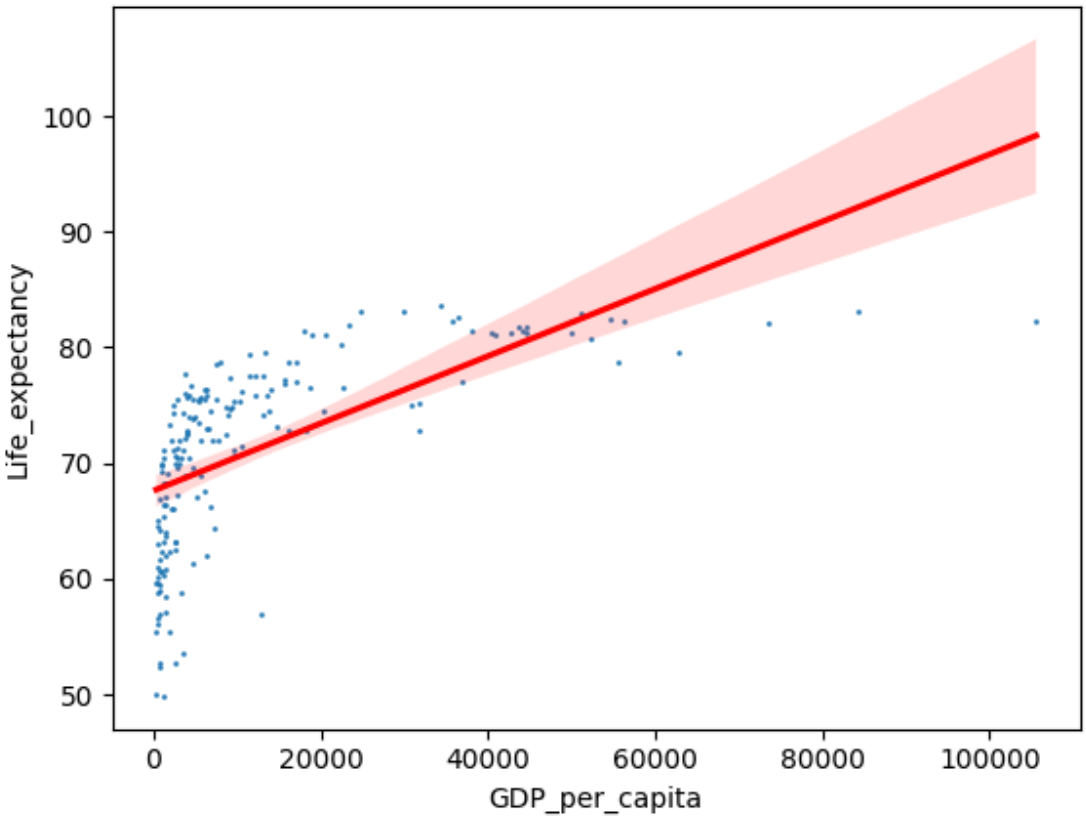
5 rows × 22 columns

Uvažujme, že nás zajímá, jaký je vliv bohatství země (měřeno pomocí hrubého domácího produktu na hlavu - *GDP per capita*, sloupec `GDP_per_capita`) na střední délku života. Pravděpodobně bude pozitivní, protože lidé v bohatých zemích

žijí v průměru déle.

In [16]:
```
data = data[data["Year"] == 2014]
sns.regplot(data, x="GDP_per_capita", y="Life_expectancy", scatter_kws={"s'
```

Out[16]: <Axes: xlabel='GDP_per_capita', ylabel='Life_expectancy'>



V grafu vidíme jednu podivnost. Délka života s bohatstvím sice roste, ale naše lineární funkce to nepopisuje moc dobře. Proč?

Zkusme se nad tím zamyslet. V nejchudších zemích je obrovské množství problémů (nedostatek potravin, pitné vody, léků atd.). I malé zvýšení HDP na hlavu povede k velkému nárůstu střední délky života. Pokud země bohatne dál, situace se zlepšuje (např. je dostupná základní zdravotní péče), ale každých tisíc dolarů k HDP na hlavu zvyšuje délku života stále méně. Lineární vztah by naopak předpokládal, že každý tisíc dolarů prodlouží průměrnou délku života o stejný počet let.

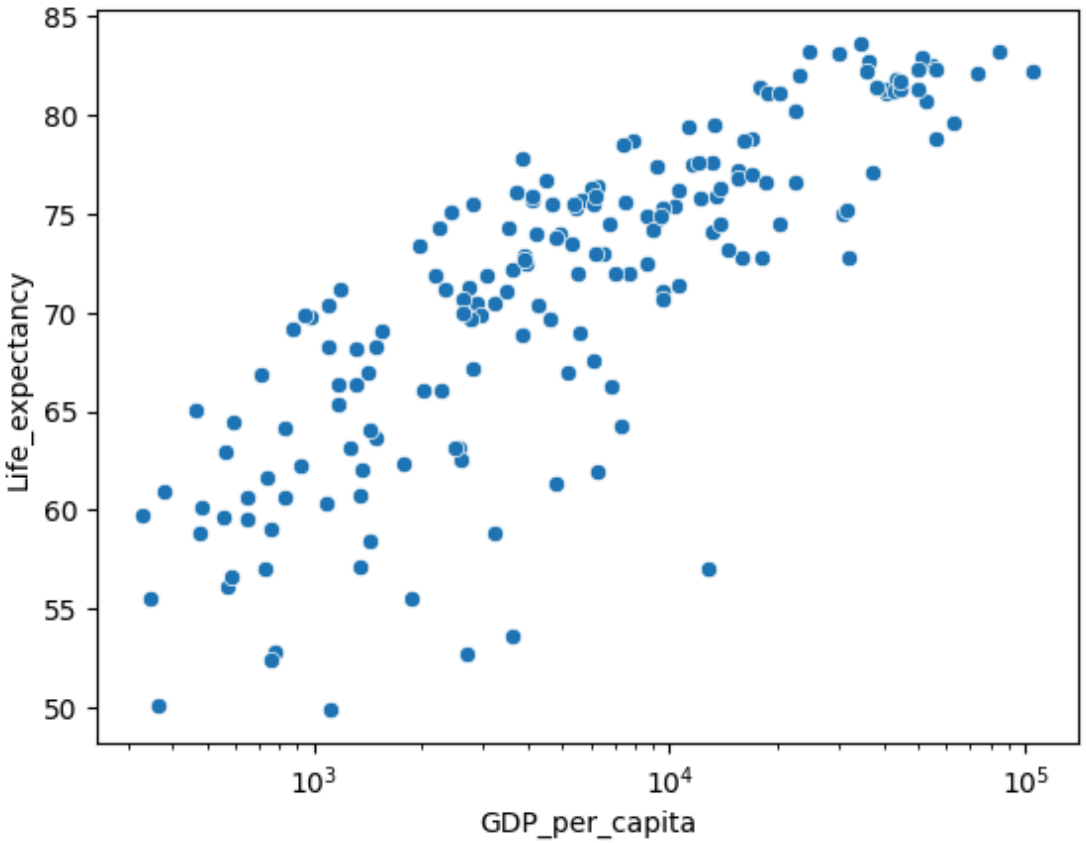
Pokud se podíváte na to, jak jsou rozptélené body v obrázku, možná vám to připomene jednu matematickou funkci - logaritmus. Nejprve si zkusme upravit naše zobrazení. U vodorovné osy *x* můžeme přepnout měřítko osy na "logaritmické" (*logarithmic scale*). To způsobí, že hodnoty na vodorovné ose budou "růst stále rychleji".

In [17]:
```
plt.xscale('log')
sns.scatterplot(data, x="GDP_per_capita", y="Life_expectancy")
```

```
c:\Users\jiri.pesik.HULD\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is depr
ecated and will be removed in a future version. Use isinstance(dtype, Catego
ricalDtype) instead
```

```
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\jiri.pesik.HULD\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is depr
ecated and will be removed in a future version. Use isinstance(dtype, Catego
ricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

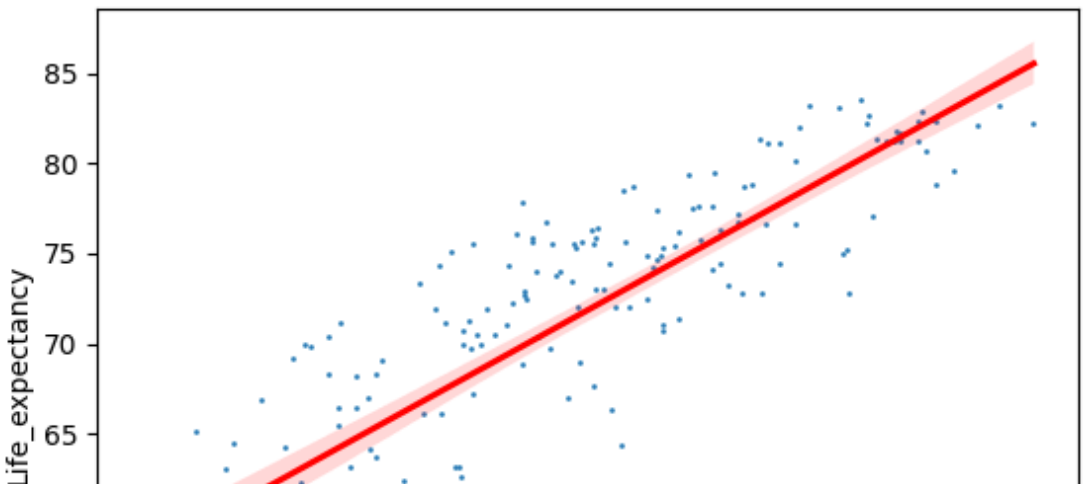
Out[17]: <Axes: xlabel='GDP_per_capita', ylabel='Life_expectancy'>

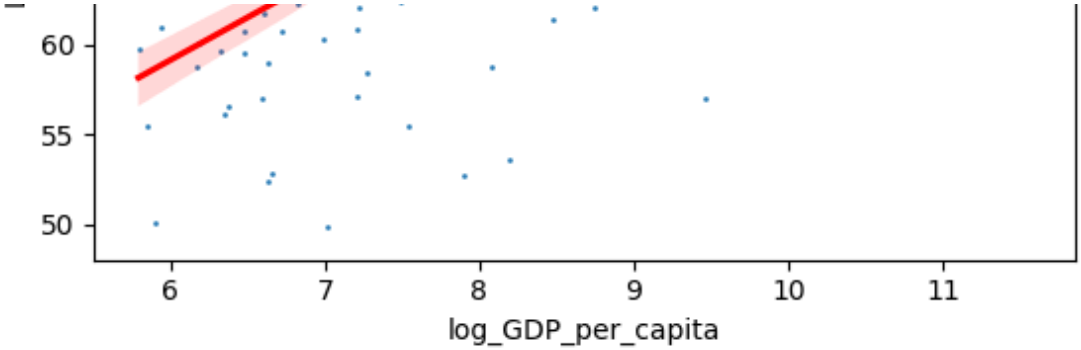


Tím jsme vyřešili problém zobrazení, ale rádi bychom použili naši myšlenku i v regresním modelu. V takovém případě je možné použít tzv. logaritmickou transformaci (*logarithmic transformation*). Vytvoříme nový sloupec, do kterého uložíme logaritmované hodnoty ze sloupce `GDP_per_capita` .

In [18]:
```
data["log_GDP_per_capita"] = np.log(data["GDP_per_capita"])
sns.regplot(data, x="log_GDP_per_capita", y="Life_expectancy", scatter_kws=
```

Out[18]: <Axes: xlabel='log_GDP_per_capita', ylabel='Life_expectancy'>





Regresní funkce již nyní popisuje data celkem dobře. Velký rozdíl oproti předchozímu grafu je nyní na vodorovné ose - namísto HDP na hlavu v dolarech tam vidíme **logaritmus** HDP na hlavu v dolarech.

In [19]:
```
formula = "Life_expectancy ~ log_GDP_per_capita"
mod = smf.ols(formula=formula, data=data)
res = mod.fit()
res.summary()
```

Out[19]: OLS Regression Results

Dep. Variable:	Life_expectancy	**R-squared:**	0.672
Model:	OLS	**Adj. R-squared:**	0.670
Method:	Least Squares	**F-statistic:**	362.8
Date:	Thu, 26 Oct 2023	**Prob (F-statistic):**	1.01e-44
Time:	12:46:22	**Log-Likelihood:**	-527.14
No. Observations:	179	**AIC:**	1058.
Df Residuals:	177	**BIC:**	1065.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	30.6788	2.156	14.228	0.000	26.424	34.934
log_GDP_per_capita	4.7435	0.249	19.047	0.000	4.252	5.235

Omnibus:	42.394	**Durbin-Watson:**	2.188
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	74.513
Skew:	-1.177	**Prob(JB):**	6.60e-17
Kurtosis:	5.109	**Cond. No.**	54.7

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Náš model má index determinace 0.67. Zkusme do něj přidat další sloupce

Náš model má index determinace 0.67. Zkusme do něj přidat další sloupce.

In [20]:
```
formula = "Life_expectancy ~ log_GDP_per_capita + Schooling + Incidents_HIV
mod = smf.ols(formula=formula, data=data)
res = mod.fit()
res.summary()
```

Out[20]: OLS Regression Results

Dep. Variable:	Life_expectancy	**R-squared:**	0.840
Model:	OLS	**Adj. R-squared:**	0.836
Method:	Least Squares	**F-statistic:**	181.9
Date:	Thu, 26 Oct 2023	**Prob (F-statistic):**	6.22e-67
Time:	12:46:22	**Log-Likelihood:**	-462.83
No. Observations:	179	**AIC:**	937.7
Df Residuals:	173	**BIC:**	956.8
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	29.0217	3.693	7.859	0.000	21.733	36.311
log_GDP_per_capita	2.8762	0.305	9.424	0.000	2.274	3.479
Schooling	0.4516	0.144	3.131	0.002	0.167	0.736
Incidents_HIV	-1.5116	0.143	-10.581	0.000	-1.794	-1.230
Polio	0.1328	0.021	6.315	0.000	0.091	0.174
BMI	0.1228	0.144	0.852	0.395	-0.162	0.407

Omnibus:	5.821	**Durbin-Watson:**	2.213
Prob(Omnibus):	0.054	**Jarque-Bera (JB):**	5.424
Skew:	-0.400	**Prob(JB):**	0.0664
Kurtosis:	3.294	**Cond. No.**	1.42e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.42e+03. This might indicate that there are strong multicollinearity or other numerical problems.

U koeficientů je nyní zajímavé, že jeden z nich je záporný, a to `Incidents_HIV`. Celkem logicky totiž platí, že vyšší výskyt HIV v populaci průměrnou délku života snižuje.

Čtení na doma

K logaritmické transformaci je třeba dodat, že ji lze použít pouze pro kladné hodnoty. Pokud jsou v datech nuly, je možné použít například funkci `log1p()`.

Další možnou transformací, kterou lze na data použít, je například transformace označovaná jako Winsorizing. Ta zbaví čísla odlehlých hodnot tak, že je nahradí určitým percentilem.

Jak vlastně regrese funguje? Přidejme si do tabulky `data` dva sloupce:

- `fittedvalues` (vyrovnané hodnoty) jsou hodnoty odhadované modelem, tj. cena domu, kterou by predikoval náš model na základě jeho velikosti,
- `resid` (rezidua) je rozdíl mezi skutečnou cenou domu a predikovanou cenou.

V tabulce níže například vidíme, že v Saudské Arábii byla střední délka dožití 74.5 let a náš model predikovat 78.14 let, tj. zmýlil se o cca tří a půl roku.

In [21]:
```
data["residuals"] = res.resid
data["predictions"] = res.fittedvalues
data[["Country", "Life_expectancy", "residuals", "predictions"]].head()
```

Out[21]:

	Country	Life_expectancy	residuals	predictions
36	Saudi Arabia	74.5	-3.635767	78.135767
40	Singapore	82.5	1.427102	81.072898
44	Costa Rica	79.4	4.500963	74.899037
47	Austria	81.5	0.221308	81.278692
72	Argentina	75.9	-0.313423	76.213423

Regrese funguje na principu minimalizace druhé mocniny součtu reziduí. Tj. vypočtené koeficienti minimalizují druhou mocninu součtu reziduí. Proto je tato metoda často označována jako metoda nejmenších čtverců (*OLS - Ordinary Least Squares*).

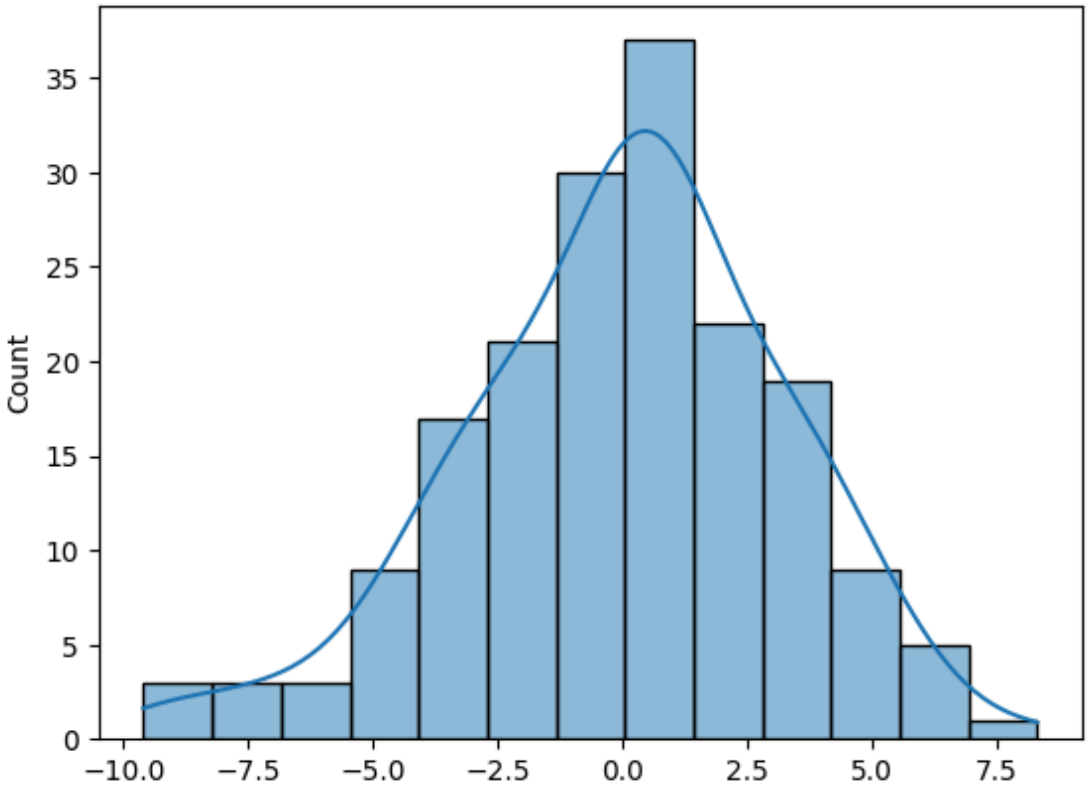
Pro regresi je dále důležité, aby rezidua měla normální rozdělení. Pro testování normality je možné využít standardní testy, modul `statsmodels` nám nabízí výsledky dvou testů: Omnibus testu a Jarque-Bera testu. Oba fungují na základě tvaru distribuční funice - šikmosti (jak moc je hustota symetrická) a špičatosti (jak rychle hustota od středu klesá). Poskytnuté p-hodnoty (ty jsou vždy označené jako `Prob.`) jsou v obou případech větší než 0.05, nezamítáme tedy hypotézu normality reziduí.

In [22]:
```
sns.histplot(res.resid, kde=True)
```

```
c:\Users\jiri.pesik.HULD\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\seaborn\_oldcore.py:1498: FutureWarning: is_categorical_dtype is depr
ecated and will be removed in a future version. Use isinstance(dtype, Catego
```

```
ecated and will be removed in a future version. Use isinstance(dtype, catego
ricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
c:\Users\jiri.pesik.HULD\AppData\Local\Programs\Python\Python311\Lib\site-pa
ckages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN be
fore operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[22]: <Axes: ylabel='Count'>



Normalita reziudí nám umožní použít řady údajů, které nám modul vypočítal. Dvě z nich jsou následující: