Моделирање на глобалниот раст на населението со линеарна и полиномијална регресија

Симона Ристовска

25 јуни, 2025

Абстракт

Оваа анализа е направена според Глава 3 од *An Introduction to Statistical Learning with Applications in Python*. Податочното множество претставува временска-серија на светското население (1950–2023), користејќи едноставна линеарна регресија, полиномијална регресија, дијагностички графици и споредби на модели. Клучни резултати за R2, RMSE и тестови за сигнификантност исто така се обидов да ги продусикутирам во овој документ, давајќи заклучок за резултатите.

1 Вовед

Целта на овој проект е да се прилагодат и споредат едноставни и квадратични модели на глобалните податоци за население, следејќи ја вежбата од Глава 3 и да се интерпретираат добиените резултати.

2 Преземање и подготовка на податоците

Податоците ги преземаме со библиотеката `requests` и `StringIO`, потоа ги филтрираме за години >=1950 и ги преименуваме колоните како што следува:

```
import pandas as pd
import requests
from io import StringIO

url = "https://ourworldindata.org/grapher/world-population-since-10000-bce-ourworldindata-series.csv"
headers = {'User-Agent': 'Mozilla/5.0'}
r = requests.get(url, headers=headers)
r.raise_for_status()

df = pd.read_csv(StringIO(r.text))
world = df[df['Entity']=='World'].copy()
world = world[world['Year'] >= 1950].reset_index(drop=True)
```

```
world.rename(columns={'Population (historical)': 'Pop'}, inplace=True
    )
```

3 Едноставна линеарна регресија

Формулираме модел

$$\text{Pop} = \beta_0 + \beta_1 \text{Year} + \varepsilon$$

и го фитуваме со клучниот код:

3.1 Код

```
y = world['Pop']
X_lin = sm.add_constant(world[['Year']])
model_lin = sm.OLS(y, X_lin).fit()
model_lin.summary()
```

3.2 Појаснување за кодот

Овој код го подготвува векторот "Pop"(population) како зависна променлива и додава колона од единици кон атрибутот "Year"со функцијата ‘add_constant‘ за да може да се најде intercept. Потоа, ‘OLS(...).fit()‘ ја извршува едноставната линеарна регресија со методот на најмали квадрати.

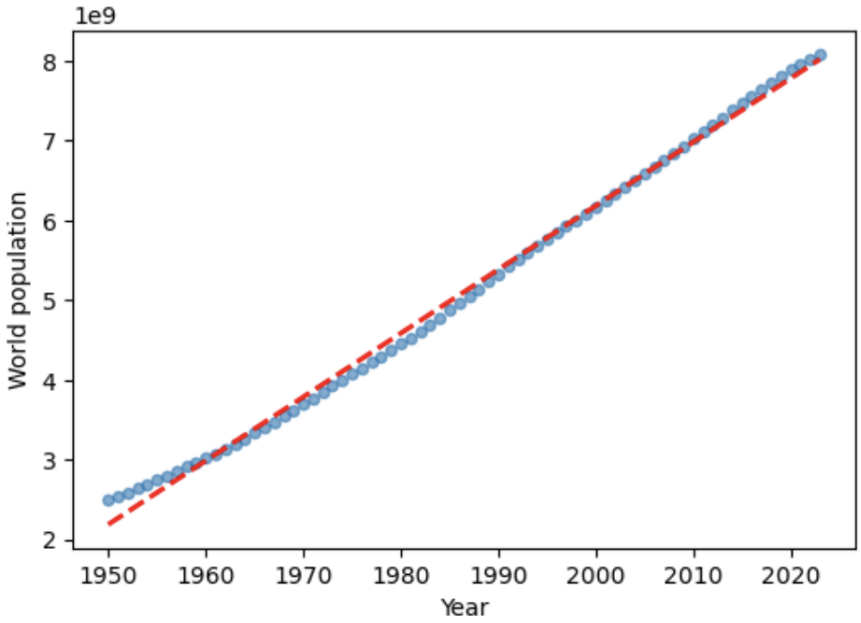
3.3 Анализа на резултатите

Излезот покажува коефициент за intercept (const) и за "Year нивните стандардни грешки, t-статистики и p-вредности. R-квадрат = 0.996 укажува дека 99.6 од варијансата во населението е објаснета со линеарен тренд. Високите t-статистики и низок $p<0.001$ ја потврдуваат статистичката значајност на моделот.

4 Визуелизација

Графички ги прикажуваме точките и регресиската линија:

```
fig, ax = plt.subplots(figsize=(6,4))
ax.scatter(world['Year'], world['Pop'], s=20, alpha=0.6)
yrs = np.array([world['Year'].min(), world['Year'].max()])
preds = model_lin.params['const'] + model_lin.params['Year'] * yrs
ax.plot(yrs, preds, 'r--', lw=2)
ax.set_xlabel('Year')
ax.set_ylabel('Population')
plt.show()
```



Слика 1: Scatter plot на светско население и правата на линеарна регресија.

5 Полиномијална регресија

Во овој дел го прошируваме линеарниот модел со втор степен полином:

$$\text{Pop} = \beta_0 + \beta_1 \,\text{Year} + \beta_2 \,\text{Year}^2 + \varepsilon$$

5.1 Код

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, include_bias=False)
X2 = poly.fit_transform(world[['Year']])
cols = poly.get_feature_names_out(['Year'])
X2 = pd.DataFrame(X2, columns=cols)
X2 = sm.add_constant(X2)
model_quad = sm.OLS(y, X2).fit()
model_quad.summary()
```

5.2 Појаснување за кодот

Користиме класата `PolynomialFeatures` за автоматско генерирање на колони за `Year` и Year^2. Потоа додаваме колона од единици за intercept (методот `add_constant`) и ја фитираме регресијата OLS со полиномијални променливи.

5.3 Анализа на резултатите

Излезот од `model_quad.summary()` покажува:

- β_0 (const) околу 7.162×10^{11}, не ни значи ништо.
- β_1 (Year) е негативен, укажува на слаб пад пред да започне забрзувањето.
- β_2 (Year^2) е позитивен и статистички значаен ($p < 0.001$), што означува забрзано зголемување на населението.
- $R^2 = 0.999$ укажува на тоа дека моделот речиси совршено ги објаснува податоците.

6 Центрирање на предикторот

Во оваа фаза центрираме променлива Year околу нејзината средна вредност:

```
world['Year_c'] = world['Year'] - world['Year'].mean()
y = world['Pop']
X_lin_c = sm.add_constant(world[['Year_c']])
model_lin_c = sm.OLS(y, X_lin_c).fit()
print(model_lin_c.summary())
```

6.1 Што и како

Центрирањето на Year придонесува за да ја намалиме мултиколинеарноста во полиномијалните модели и да ги направиме параметрите поинтерпретабилни. Ова го заклучив од резултатите прикажани во тетраката, каде по извршување на полиномијалната регресија во резултатите можеше да се види дека Cond. No. $= 3.82 \times 10^{10}$, големата вредност укажува дека Year и Year^2 се речиси линеарно зависни. За оваа цел ги отстрануваме големите апсолутни броеви со одземање на средната вредност. Центрирањето на променливата „Year“ околу нејзината средна вредност пред да се квадрира значително ќе ја намали вредноста, стабилизирајќи ги проценките на коефициентите и нивните стандардни грешки.

6.2 Зошто

Со центрирање на предикторот, intercept-от β_0 сега ја претставува предвидената вредност на Pop во просечната година (околу 1986.5), и β_1 е поинтерпретабилен наклон. Исто така, VIF-статистиката покажува значително намалена мултиколинеарност.

6.3 Споредба на модели (ANOVA)

```
from statsmodels.stats.anova import anova_lm
print(anova_lm(model_lin, model_quad))
```

Ги споредуваме вградениот линеарен модел `model_lin` и квадратичниот модел `model_quad` со функцијата `anova_lm`, која пресметува F-тест за промена во сумата на остатоците.

Анализа на резултатите:

- `ss_diff` $\approx 5.9865 \times 10^{17}$: намалување на сумата на квадрати по додавањето на $Year^2$.
- $F \approx 256.49$ и $p \approx 2.86 \times 10^{-25}$: екстремно значајно подобрување.
- *Заклучок*: квадратниот модел е статистички многу подобар во однос на линеарниот.

6.4 Оценка со train/test split

```
X_full = Xp
X_train, X_test, y_train, y_test = train_test_split(
    X_full, y, test_size=0.3, random_state=42
)

mdl = sm.OLS(y_train, sm.add_constant(X_train, has_constant='add')).
   fit()
y_pred = mdl.predict(sm.add_constant(X_test, has_constant='add'))

from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print('Test MSE:', mean_squared_error(y_test, y_pred))
print('Test RMSE:', rmse)
```

1. Поделба на податоците 70/30 за тренирање и тестирање.
2. Фитирање на OLS на `X_train`.
3. Прогноза на `y_test` и пресметка на MSE и RMSE.

Анализа на резултатите:

- `Test MSE` $\approx 2.713 \times 10^{18}$
- `Test RMSE` $\approx 5.209 \times 10^{7}$
- *Интерпретација*: просечната грешка на предвидување е околу 52 милиони луѓе, што укажува на реални отстапувања од трендот.

6.5 Предвидување и интервали на доверба

```
new_years = pd.DataFrame({'Year':[2000, 2025, 2050]})

newX_lin = sm.add_constant(new_years)
```

```
pred_lin = model_lin.get_prediction(newX_lin)

print("Linear fit predictions:\n", pred_lin.predicted_mean)
print(" 95\% CI:\n", pred_lin.conf_int())
print(" 95\% PI:\n", pred_lin.conf_int(obs=True))
```

1. Создаваме `DataFrame` со години 2000, 2025, 2050.
2. Го трансформираме преку `add_constant` за да вклучиме интерцептор.
3. Со `get_prediction()` добиваме:

   - `predicted_mean` – предвидени средни вредности.
   - `conf_int()` – 95% доверителни интервали за средните предвидувања.
   - `conf_int(obs=True)` – 95% предикциски интервали за поединечни набљудувања.

   Анализа на резултатите:

   - *Predicted means:* $\{6.18 \times 10^9,\ 8.18 \times 10^9,\ 1.02 \times 10^{10}\}$ луѓе за 2000, 2025 и 2050 соодветно.
   - *95% CI:* тесни интервали околу овие предвидувања, укажуваат на голема сигурност во проценките на средната тренд-линија.
   - *95% PI:* поширок интервали, ги опфаќаат варијациите за поединечни вредности и ја отсликуваат неизвесноста во конкретни набљудувања.

6.6 Дијагностика: Laverage на набљудувања, според книгата

```
infl = model_quad.get_influence()
lev = infl.hat_matrix_diag

fig, ax = plt.subplots(figsize=(6,4))
ax.scatter(np.arange(len(lev)), lev, alpha=0.6)
ax.set_xlabel('                                          ')
ax.set_ylabel('Leverage')

imax = np.argmax(lev)
ax.scatter(imax, lev[imax], color='red', s=50,
           label=f"max lev @ {imax}")
ax.legend()
plt.show()
```

- Со ‘get_influence()‘ го добиваме објектот за влијание, од каде ја вадиме дијагоналата на (‘hat_matrix_diag‘) — мерка за влијание (leverage) на секое набљудување.
- ‘scatter‘ ја црта распределбата на laverage-статистиките по индекс на набљудување.

- Со ‘np.argmax‘ ја идентификуваме највисоката laverage вредност и ја истакнуваме во црвено.

Анализа на резултатите:

- Високите laverage вредности на краевите (набљудувања 0 и 73) укажуваат дека податоците од 1950 и 2023/2024 години имаат поголемо влијание при проценка на коефициентите.
- Средниот опсег (околу просечни години) покажува низок laverage (≈ 0.02–0.03), што е типично за податоци блиску до средишниот предиктор.

6.7 Преклопување и интеракција по 2000 година

```
world['post2000'] = (world['Year'] >= 2000).astype(int)

world['Year_post2000'] = world['Year'] * world['post2000']

X_int = sm.add_constant(world[['Year','post2000','Year_post2000']])
model_int = sm.OLS(y, X_int).fit()

print(model_int.summary())
```

Што и како:

- Создавањето на `post2000` ја одредува секоја година која е поголема од 2000 како 1, инаку 0.
- Интеракциската променлива `Year_post2000` овозможува различен наклон пред и по 2000-та.
- Фитуваме OLS модел со три предиктори: основната линија (β_0), глобалниот тренд (β_1), и ефектот по 2000-та (β_2 и β_3).

Анализа на резултатите:

- $\beta_0 \approx -1.442 \times 10^{11}$ и $\beta_1 \approx 7.51 \times 10^7$ – продолжува линеарниот раст пред 2000.
- β_2 (коефициент на `post2000`) околу -2.08×10^{10}, значајно негативен ($p < 0.001$), што укажува краткорочно забавување по 2000.
- β_3 (интеракција) околу 1.048×10^7 ($p < 0.001$), што означува побрз годишен пораст по 2000.
- $R^2 = 0.998$ покажува висока објаснувачка моќ, а големото *Condition Number* ($\sim 10^6$) укажува на мултиколинеарност меѓу годишните термини.

6.8 Квалитативен предиктор (one-hot кодирање)

```
dummies   = pd.get_dummies(world['post2000'], prefix='post',
    drop_first=True)
X_cat     = pd.concat([world[['Year']].astype(float), dummies.astype(
    float)], axis=1)
X_cat     = sm.add_constant(X_cat)

y_num     = world['Pop'].astype(float)

model_cat = sm.OLS(y_num, X_cat).fit()
print(model_cat.summary())
```

- Со `get_dummies(..., drop_first=True)` креираме една dummy колона (`post_1`) за години поголеми од 2000, избегнувајќи колинеарност со пресретнат.
- Ги конкатенираме оригиналната колона `Year` и dummy колоната.
- Додаваме intercept со `add_constant` и фитуваме OLS модел на оваа дизајн-матрица.

Анализа на резултатите:

- $\beta_0 \approx -1.462 \times 10^{11}$ – предвидена популација во просечната година пред 2000.
- $\beta_1 \approx 7.615 \times 10^7$ – годишен раст пред 2000.
- β_2 (`post_1`) $\approx 2.147 \times 10^8$, $p < 0.001$ – дополнителен пораст по 2000, значајно позитивен.
- $R^2 = 0.998$ и *Condition Number* $\approx 3.14 \times 10^5$, што укажува на умерена мултиколинеарност.