دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

آمار مهندسی
به کارگیری پایتون در آمار مهندسی

دکتر امیر احمدی جاوید

تدریسیار:
پدرام پیرو اصفیا   مهدی محمدی

فهرست

- **آزمون های فرض**
  - آشنایی با دیتاست استفاده شده در این بخش
  - آزمون های مربوط به میانگین ها
  - آزمون های مربوط به تفاضل دو میانگین
  - آزمون های درباره واریانس ها
  - آزمون های مربوط به نسبت ها
  - آزمون های مربوط به تفاضل های بین k نسبت
  - تحلیل جدول $r \times c$
  - نیکویی برازش
- **محاسبه بازه اطمینان**
  - بازه اطمینان میانگین ها
  - بازه اطمینان تفاضل بین میانگین ها
  - بازه اطمینان نسبت ها

فهرست

آزمون های فرض

❖ آشنایی با دیتاست استفاده شده:

دیتاستی که در این بخش برای انجام تست های آماری استفاده کرده ایم در زیر قابل مشاهده است که مربوط به چرخ و فلک زدن کارکنان دانشگاه میشیگان(!) است:

	ID	Age	Gender	GenderGroup	Glasses	GlassesGroup	Height	Wingspan	CWDistance	Complete	CompleteGroup	Score
0	1	56	F	1	Y	1	62.0	61.0	79	Y	1	7
1	2	26	F	1	Y	1	62.0	60.0	70	Y	1	8
2	3	33	F	1	Y	1	66.0	64.0	85	Y	1	7
3	4	39	F	1	N	0	64.0	63.0	87	Y	1	10
4	5	27	M	2	N	0	73.0	75.0	72	N	0	4

اطلاعاتی که در این دیتاست مشهود است، به شرح زیر است:

- Age
- Gender
- Glass-wearing or not
- Height
- Weight
- Wingspan (arm length)
- Completion
- Cartwheel distance
- Overall cartwheel score

آزمون های فرض

همانطور که میدانید، آزمون های فرض مختلفی وجود دارد که در این بخش قرار است با نحوه پیاده سازی آن ها در پایتون آشنا شویم.

❖ آزمون های مربوط به میانگین ها:

آزمون های مربوط به میانگین ها (با فرض نرمال بودن داده ها)، به دو بخش کلی تقسیم میشدند:

۱- واریانس جامعه را داریم.   ۲- واریانس جامعه را نداریم.

که در حالت اول آماره ما از Z و در حالت دوم از T_n پیروی میکند.

میخواهیم میانگین فاصله چرخ و فلک زدن این نمونه کارکنان را (با فرض اینکه سیگمای جامعه معلوم است و برابر با انحراف معیار خود نمونه است) در تست های زیر بررسی کنیم و نتایج را تحلیل کنیم: (فاصله چرخ و فلک زدن، ستون CWDistance است)

1. $\begin{cases} H_0: \mu \geq 80 \\ H_1: \mu < 80 \end{cases}$ 2. $\begin{cases} H_0: \mu = 82 \\ H_1: \mu \neq 82 \end{cases}$ 3. $\begin{cases} H_0: \mu \leq 75 \\ H_1: \mu > 75 \end{cases}$

بدین منظور باید از کتابخانه جدیدی به نام statsmodels استفاده کنیم؛

```
from statsmodels.stats.weightstats import ztest
```

آزمون های فرض

اول از همه برای اینکه ذهنیت کلی از این ستون داشته باشیم:

```
df['CWDistance'].describe()
```

output

```
count     25.000000
mean      82.480000
std       15.058552
min       63.000000
25%       70.000000
50%       81.000000
75%       92.000000
max      115.000000
Name: CWDistance, dtype: float64
```

1) فرض ۱ در تست اول، $\mu < 80$ است:

```
ztest(df["CWDistance"], value = 80, alternative = "smaller")
```

output

$\underbrace{(0.8234523266982029,}_{\text{آماره آزمون}}\ \underbrace{0.7948745915460473)}_{\text{p-value}}$

که چون p-value خیلی بزرگ است، فرض صفر را نمیتوانیم رد کنیم.

2) فرض ۱ در تست دوم، $\mu \neq 82$ است:

```
ztest(df["CWDistance"], value = 82, alternative = "two-sided")
```

output

$\underbrace{(0.15937786968352421,}_{\text{آماره آزمون}}\ \underbrace{0.8733711734744286)}_{\text{p-value}}$

و فرض صفر را نمیتوانیم رد کنیم.

آزمون های فرض

3) فرض ۱ در تست سوم، $\mu > 75$ است:

```
ztest(df["CWDistance"], value = 75, alternative = "larger")
```

output

```
(2.4836384692348994, 0.006502388157178615)
```

$\underbrace{2.4836384692348994}_{\text{آماره آزمون}}$, $\underbrace{0.006502388157178615}_{\text{p-value}}$

و چون p-value مقدار کوچکی است، فرض صفر را رد میکنیم.

اما همانطور که میدانید، این فرض که سیگمای جامعه ای که از آن نمونه گیری کرده ایم معلوم است و برابر با سیگمای نمونه است، فرض بعیدی است و معمولا این تست ها را با فرض مجهول بودن سیگما انجام میدهیم:

```
from scipy import stats
```

```
stats.ttest_1samp(df["CWDistance"], popmean=80 , alternative = "less")
stats.ttest_1samp(df["CWDistance"], popmean=82, alternative = "two-sided")
stats.ttest_1samp(df["CWDistance"], popmean=75, alternative = "greater")
```

output

```
Ttest_1sampResult(statistic=0.823452326698202 9, pvalue=0.7908206671466147)
Ttest_1sampResult(statistic=0.1593778696835242 1, pvalue=0.8747048534781223)
Ttest_1sampResult(statistic=2.4836384692348994, pvalue=0.010191116983652436)
```

آزمون های فرض

❖ **آزمون های مربوط به تفاضل دو میانگین:**

برای این بخش باید فرض های زیر را حتما در نظر داشته باشید:

✓ دو جامعه از هم مستقل هستند.

✓ انحراف معیار دو جامعه را نداریم ولی میدانیم با هم برابر هستند.

✓ دو جامعه از توزیع نرمال پیروی میکنند.

حال فرض کنید میخواهیم آزمون فرض برابری میانگین فاصله چرخ و فلک زدن را برای زنان و مردان بسنجیم:

$$\begin{cases} H_0: \mu_{male} - \mu_{female} = 0 \\ H_1: \mu_{male} - \mu_{female} \neq 0 \end{cases}$$

```
male_CWD=df[df['GenderGroup']==2]['CWDistance']
female_CWD=df[df['GenderGroup']==1]['CWDistance']

stats.ttest_ind(male_CWD ,female_CWD , equal_var=True , alternative='two-sided')
```

output

```
Ttest_indResult(statistic=0.7038770203471643, pvalue=0.48857493382089734)
```

پس با این حساب، نمیتوانیم فرض صفر را رد کنیم.

❖ آزمون های درباره واریانس ها:

برای این بخش باید فرض های زیر را حتما در نظر داشته باشید:

- ✓ دو جامعه از هم مستقل هستند.
- ✓ دو جامعه از توزیع نرمال پیروی میکنند.

توجه داشته باشید که تست های استفاده شده در این بخش از لحاظ تئوری از سطح درس فراتر هستند.

حال فرض کنید میخواهیم آزمون فرض برابری واریانس فاصله چرخ و فلک زدن را برای زنان و مردان بسنجیم:

$$\begin{cases} H_0: \sigma^2_{male} = \sigma^2_{female} \\ H_1: \sigma^2_{male} \neq \sigma^2_{female} \end{cases}$$

برای این موضوع از تست bartlett استفاده میکنیم. همچنین به داکیومنتیشن این تست نیز میتوانید رجوع کنید. برای استفاده از این تست، محدودیت تعداد نمونه ندارید و میتوانید برابری واریانس چندین نمونه را بررسی کنید.

```
stats.bartlett(male_CWD ,female_CWD)
```

output

```
BartlettResult(statistic=2.534074437179493, pvalue=0.11141219941918763)
```

که یعنی واریانس ها برابر نیستند و دلیل آن این است که واریانس مردان برابر ۳۳۰ و زنان برابر ۱۲۳ میباشد.

هرچند این تست، تنها تست برابری واریانس ها نیست و میتوان به تست هایی مثل levene و ... نیز اشاره کرد.

❖ آزمون های مربوط به نسبت ها:

فرض کنید میخواهیم درباره نسبت مردان در این جامعه آزمونی انجام دهیم: (فرض کنید تعداد نمونه بزرگتر از ۳۰ است که شرایط تخمین نرمال برقرار باشد)

$$\begin{cases} H_0: p \geq 0.6 \\ H_1: p < 0.6 \end{cases}$$

همچنین تعداد مردان و زنان در این نمونه را میتوانیم از طریق زیر ببینیم:

```
df['Gender'].value_counts()
```

output

```
M     13
F     12
Name: Gender, dtype: int64
```

برای انجام این تست از کتابخانه statsmodels استفاده میکنیم (داکیومنتیشن تست):

```
from statsmodels.stats.proportion import proportions_ztest
proportions_ztest(count = len(df[df['Gender']=='M']), nobs=len(df), value=0.6, alternative='smaller')
```

output

```
(-0.8006407690254352, 0.21166982079122193)
```

که به این معناست نمیتوانیم فرض صفر را رد کنیم.

❖ آزمون های مربوط به تفاضل های بین k نسبت

برای این بخش از یکی از مثال های کتاب استفاده میکنیم (ص ۵۴۸ pdf جان فروند):

مثال ۱۰.۱۳

برمبنای داده‌های نمونه‌ای که در جدول زیر نشان داده‌شده، تعیین کنید که آیا نسبت واقعی مشتریانی که مادهٔ شویندهٔ A را به مادهٔ شویندهٔ B ترجیح می‌دهند، در هر سه شهر یکسان است یا نه.

	عده‌ای که مادهٔ شویندهٔ A را ترجیح می‌دهند	عده‌ای که مادهٔ شویندهٔ B را ترجیح می‌دهند	
شهر الف	۲۳۲	۱۶۸	۴۰۰
شهر ب	۲۶۰	۲۴۰	۵۰۰
شهر ج	۱۹۷	۲۰۳	۴۰۰

برای انجام این تست از کتابخانه statsmodels استفاده میکنیم (داکیومنتیشن تست):

آزمون های فرض

```python
from statsmodels.stats.proportion import proportions_chisquare
proportions_chisquare(count = [232,260,197] , nobs=[400,500,400] )
```

output

```
(6.473303894018467,
 0.03929523764331261,
 (array([[232, 168],
         [260, 240],
         [197, 203]]),
  array([[212., 188.],
         [265., 235.],
         [212., 188.]])))
```

در بخش count، تعداد موفقیت ها را وارد میکنیم (میتوانیم اینگونه در نظر بگیریم که هرکس که A را انتخاب کند برنده و افرادی که B انتخاب کرده اند بازنده هستند). در بخش nobs (number of observations) نیز تعداد نمونه های هر کدام از شهر ها را مینویسیم. خروجی ها به ترتیب اینگونه هستند:

1. آماره آزمون
2. پی ولیو
3. آرایه مربوط به فراوانی های نمونه
4. آرایه مربوط به امید فراوانی ها

❖ تحلیل یک جدول $r \times c$

برای این بخش از یکی از مثال های کتاب استفاده میکنیم و استقلال نحوه کار را از بهره هوشی میسنجیم (ص ۵۵۱ pdf جان فروند):

		نحوهٔ کار			
		ضعیف	متوسط	خوب	
بهرهٔ هوشی	کمتر از متوسط	۶۷	۶۴	۲۵	۱۵۶
	متوسط	۴۲	۷۶	۵۶	۱۷۴
	بالاتر از متوسط	۱۰	۲۳	۳۷	۷۰
		۱۱۹	۱۶۳	۱۱۸	۴۰۰

برای انجام این تست از کتابخانه scipy استفاده میکنیم (داکیومنتیشن تست):

```
obs = np.array([[67,64,17],[42,76,56],[10,23,37]])
stats.chi2_contingency(obs)
```

output →

```
(51.45978838023338,
 1.789147611060362e-10,
 4,
 array([[44.92857143, 61.54081633, 41.53061224],
        [52.82142857, 72.35204082, 48.82653061],
        [21.25      , 29.10714286, 19.64285714]]))
```

آزمون های فرض

خروجی ها از قرار زیر است:

1. آماره آزمون
2. پی ولیو
3. درجه آزادی (برای آماره آزمون χ^2)
4. فراوانی های مورد انتظار

❖ **نیکویی برازش**

به علت استفاده زیاد از دو تابع نرمال و نمایی، توابع زیادی برای این تست نیکویی برازش برای این دو توزیع وجود دارد، این در حالیست که برای دیگر توابع به این صورت نیست. میتوانید لیستی از توابع مناسب برای نیکویی برازش را در اینجا ببینید.

همچنین کتابخانه scipy نیز متُدی دارد که میتوانید فراوانی مشاهده شده و فراوانی مورد انتظار را به آن بدهید و آماره آزمون و پی ولیو را به شما بدهد (که همین محاسبه فراوانی مورد انتظار مشکل است).

آزمون های فرض

توجه کنید که تست های مربوط به نرمال بودن دیتا **خیلی زیاد** هستند و شما باید بهترین را بر اساس نوع کارتان و تشخیص خودتان انتخاب کنید. (خیلی از تست ها در کتابخانه scipy هستند)

```
normal_sample=np.random.normal(loc=10 , scale=2 , size=1000)
exp_sample = np.random.exponential(scale=20 , size=1000)
```

دو نمونه رندوم از جوامع نرمال و نمایی میگیریم و تست ها را روی آن ها پیاده میکنیم:

```
from statsmodels.stats.diagnostic import kstest_normal
from statsmodels.stats.diagnostic import kstest_exponential

print('normality test for normal population:',kstest_normal(normal_sample))
print('normality test for exponential population:',kstest_normal(exp_sample))

print('exponential test for exponential population:',kstest_exponential(exp_sample))
print('exponential test for normal population:',kstest_exponential(normal_sample))
```

output

```
normality test for normal population: (0.019233716976817306, 0.5724557897411924)
normality test for exponential population: (0.1587866880711949, 0.0009999999999998899)
exponential test for exponential population: (0.021355063428022958, 0.6079974609799691)
exponential test for normal population: (0.44467894274467484, 0.0009999999999998899)
```

❖ بازه اطمینان میانگین ها:

با فرض اینکه دیتایی که در اختیار داریم از یک جامعه نرمال با سیگمای معلوم است، داریم:

```
data=np.random.randint(1,51,size=100)
print(data[:10])
stats.norm.interval(alpha=0.95 , loc=np.mean(data) , scale=stats.sem(data))
```

میانگین دیتا

انحراف معیار $\bar{X}$

output

دیتای رندوم (۱۰ تای اول)

بازه اطمینان ۹۵٪

```
[43 36  8 35 20 43 43 36 47  9]

(23.3049413305260 5, 29.09505866947395)
```

اگر سیگما معلوم نباشد و $n \leq 30$ باشد:

درجه آزادی

```
stats.t.interval(alpha=0.95 , df=len(data)-1 ,loc=np.mean(data) , scale=stats.sem(data))
```

میانگین دیتا

انحراف معیار $\bar{X}$

output

بازه اطمینان ۹۵٪

```
(23.269117324147956, 29.130882675852042)
```

❖ بازه اطمینان تفاضل بین میانگین ها:

با فرض اینکه جوامع کاربردی نرمال با سیگمای معلوم هستند، داریم:

```
import statsmodels.stats.api as sms
A=[164.4,169.7 , 169.2,169.5,161.8,168.7,169.5,163.9]
B=[163.5 , 162.8,163,163.2,160.7,161.5,160.9,162.1]

cm = sms.CompareMeans(sms.DescrStatsW(A), sms.DescrStatsW(B))
cm.zconfint_diff(alpha=0.05,alternative='two-sided')
```

output

```
(2.547119130488207З, 7.202880869511793)
```

چند نکته:

- در اینجا از API کتابخانه statsmodels استفاده کردیم، بین کتابخانه و api اش یک سری تفاوت ها وجود دارد که به بررسی آن نمی پردازیم.
- DescrStatsW از دیتاتایپ های ساخته شده در کتابخانه statsmodels میباشد (مثل آرایه در نامپای) و برای محاسبه برآورد فاصله ای نیاز به همچنین دیتاتایپی است.

محاسبه بازه اطمینان

با فرض اینکه جوامع کاربردی نرمال هستند ولی واریانس آن ها معلوم نیست و $n \leq 30$ برای هر دو میباشد:

```python
import statsmodels.stats.api as sms
A=[164.4,169.7 , 169.2,169.5,161.8,168.7,169.5,163.9]
B=[163.5 , 162.8,163,163.2,160.7,161.5,160.9,162.1]

cm = sms.CompareMeans(sms.DescrStatsW(A), sms.DescrStatsW(B))
cm.tconfint_diff(alpha=0.05,alternative='two-sided',usevar='pooled')
```

output

```
(2.32760222158774, 7.42239777841226)
```

❖ **بازه اطمینان نسبت ها:**

نمونه ای ۴۰۰ نفره را در نظر بگیرید که ۱۳۶ نفر در آن سیگار میشکند، یک فاصله اطمینان ٪۹۵ به صورت زیر است:

```python
sms.proportion_confint(count=136,nobs=400,alpha=.05)
```

output

```
(0.2935773934552420З, 0.386422606544758)
```

❖ بازه اطمینان تفاضل بین نسبت ها:

مثال ۹.۱۱

اگر ۱۳۲ نفر از ۲۰۰ رأی‌دهندهٔ مذکر و ۹۰ نفر از ۱۵۰ رأی‌دهندهٔ مؤنث موافق کاندیدای خاصی برای انتخاب ریاست جمهوری باشند، یک بازهٔ اطمینان ٪۹۹ برای تفاضل بین نسبتهای واقعی رأی‌دهندگان مرد و زن که موافق این کاندیدا هستند، به‌دست آورید.

```
import statsmodels.stats as sm
sm.proportion.confint_proportions_2indep(count1=132, nobs1=200,count2=90,nobs2=150,alpha=0.01)
```

output

```
(-0.07220812460932052, 0.1920655801928588)
```

❖ **بازه اطمینان واریانس ها:**

مثال ۱۰.۱۱

در ۱۶ بارکار آزمایشی یک موتور تحت آزمایش، مصرف بنزین آن دارای انحراف معیار ۲٫۲ گالن بوده است. یک بازهٔ اطمینان ٪۹۹ برای σ^2 بسازید که میزان تغییرپذیری مصرف بنزین این موتور را بسنجد.

۱) محاسبه $[\chi^2_{1-\frac{\alpha}{2},n-1},\chi^2_{\frac{\alpha}{2},n-1}]$:

```
stats.chi2.interval(alpha=0.99 , df=15)
```

output

```
(4.600915571727341, 32.80132064579183)
```

۲) محاسبه $\left[\frac{(n-1)S^2}{\chi^2_{\frac{\alpha}{2},n-1}},\frac{(n-1)S^2}{\chi^2_{1-\frac{\alpha}{2},n-1}}\right]$:

```
chi_int=stats.chi2.interval(alpha=0.99 , df=15)
(15*2.2**2/chi_int[1] , 15*2.2**2/chi_int[0])
```

output

```
(2.213325517712472, 15.779467992442099)
```

❖ **بازه اطمینان نسبت دو واریانس:**

مثال ۶.۱۱

مطالعه‌ای برای مقایسهٔ محتوای نیکوتین دو نوع سیگار به‌عمل آمده است. متوسط محتوای نیکوتین ۱۰ سیگار نوع (الف) ۳٫۱ میلی‌گرم با انحراف معیار ۰٫۵ میلی‌گرم بوده است، در حالی‌که ۸ سیگار نوع (ب) دارای محتوای نیکوتین متوسط ۲٫۷ میلی‌گرم با انحراف معیار ۰٫۷ میلی‌گرم بوده‌اند.

یک بازهٔ اطمینان ۹۸٪ برای $\frac{\sigma_1^2}{\sigma_2^2}$ پیدا کنید.

1) محاسبه $\left[f_{1-\frac{\alpha}{2},n_1-1,n_2-1}, f_{\frac{\alpha}{2},n_1-1,n_2-1}\right]$:

```
stats.f.interval(alpha=0.98,dfn=10-1,dfd=8-1)
```

output

```
(0.17816211773303609, 6.718752481824472)
```

۲) محاسبه $\left[\frac{S_1^2}{S_2^2} f_{\frac{\alpha}{2},n_1-1,n_2-1}, \frac{S_1^2}{S_2^2} f_{1-\frac{\alpha}{2},n_1-1,n_2-1}\right]$:

```
f_int=stats.f.interval(alpha=0.98,dfn=10-1,dfd=8-1)
(0.25/0.49*1/f_int[1] , 0.25/0.49*1/f_int[0])
```

output

```
(0.07593732363453692, 2.863706876212369)
```

❖ آشنایی با دیتاست استفاده شده:

دیتاستی که در این بخش از آن استفاده شده است، از دیتاست های از پیش قرار داده شده در یکی از کتابخانه های معروف برای ماشین لرنینگ و دیتاساینس میباشد (scikit-learn).

دیتاست را فراخوانی میکنیم و آن را تبدیل به یک دیتافریم میکنیم (نحوه وارد کردن دیتاست و دیگر عملیات های صورت گرفته روی آن برای تبدیلش به دیتافریم مهم نیست).

```
from sklearn.datasets import load_boston
boston_dataset = load_boston()
boston = pd.DataFrame(data=boston_dataset.data, columns=boston_dataset.feature_names)
boston["MEDV"] = boston_dataset.target
boston.head()
```

output

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

اطلاعات ستون ها از قرار زیر است:

- **CRIM:** Per capita crime rate by town
- **ZN:** Proportion of residential land zoned for lots over 25,000 sq. ft
- **INDUS:** Proportion of non-retail business acres per town
- **CHAS:** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- **NOX:** Nitric oxide concentration (parts per 10 million)
- **RM:** Average number of rooms per dwelling
- **AGE:** Proportion of owner-occupied units built prior to 1940
- **DIS**: Weighted distances to five Boston employment centers
- **RAD**: Index of accessibility to radial highways
- **TAX:** Full-value property tax rate per $\$10,000$
- **PTRATIO:** Pupil-teacher ratio by town
- **B:** $1000(Bk — 0.63)^2$, where Bk is the proportion of [people of African American descent] by town
- **LSTAT:** Percentage of lower status of the population
- **MEDV:** Median value of owner-occupied homes in 1000$s

❖ پیاده سازی مدل رگرسیون خطی

توجه داشته باشید که کتابخانه های زیادی برای فیت کردن مدل های رگرسیون و ... است، ولی برای اینکه کار را خیلی پیچیده نکنیم و درک بهتری از مدل بدست آمده داشته باشیم، از کتابخانه statsmodels استفاده میکنیم. همانطور که میدانید روش های مختلفی برای بدست آوردن خط رگرسیون وجود دارد که معروف ترین آن، روش **خط کمترین مربعات (OLS)** است.

حال به سراغ فیت کردن خط رگرسیون میرویم. فرض کنید میخواهیم متغیر پیشبینی کننده را RM در نظر بگیریم. (**پس به صورت خلاصه: متغیر پیشبینی کننده (مستقل): RM متغیر هدف (وابسته): MEDV**)

```python
import statsmodels as sm
model = sm.regression.linear_model.OLS.from_formula("MEDV ~ RM", data=boston)
result = model.fit()
result.summary()
```

در اسلاید بعد، گام هایی که طی کردیم را بررسی میکنیم.

```python
import statsmodels as sm
model = sm.regression.linear_model.OLS.from_formula("MEDV ~ RM", data=boston)
result = model.fit()
result.summary()
```

از کتابخانه statsmodels که با عنوان sm (برای راحتی) وارد شده است، به بخش regression/linear_model میرویم و چون میخواهیم پارامتر ها را به صورت کمترین مربعات برآورد کنیم، به OLS میرویم. حال فرمول رگرسیونی را به آن میدهیم:

$$MEDV = \beta_0 + \beta_1 . RM + \epsilon \quad \xRightarrow{\text{بعد از تخمین پارامتر}} \quad \widehat{MEDV} = \widehat{\beta_0} + \widehat{\beta_1} . RM \Rightarrow MEDV \sim RM$$

سپس مدلی که ساختیم را فیت میکنیم و در متغیر result، این آبجکت را ذخیره میکنیم.

در نهایت خلاصه ای از مدل فیت شده میبینیم.

خروجی را در اسلاید بعد میتوانید ببینید.

متغیر مستقل	Dep. Variable:	MEDV	R-squared:	0.484	معیار R^2
نوع مدل	Model:	OLS	Adj. R-squared:	0.483	معیار $adjusted\ R^2$
متد محاسبه پارامترها	Method:	Least Squares	F-statistic:	471.8	
تاریخ فیت کردن مدل	Date:	Wed, 13 Apr 2022	Prob (F-statistic):	2.49e-74	
زمان فیت کردن مدل	Time:	14:14:52	Log-Likelihood:	-1673.1	
تعداد مشاهدات	No. Observations:	506	AIC:	3350.	
درجه آزادی باقی مانده ها (۱)	Df Residuals:	504	BIC:	3359.	
درجه آزادی مدل (۲)	Df Model:	1			
	Covariance Type:	nonrobust			

درجه آزادی باقی مانده ها: برابر با تعداد مشاهدات منهای تعداد متغیر هایی است که برآورد میشود. (β_0,β_1)
درجه آزادی مدل: برابر با تعداد متغیر هایی است که در پیشبینی مورد استفاده قرار میگیرد.

<table>
<tr><td></td><td></td><td>ضریب</td><td>انحراف معیار</td><td>آماره آزمون</td><td>پی ولیو</td><td>کران پایین فاصله اطمینان ۹۵٪</td><td>کران بالای فاصله اطمینان ۹۵٪</td></tr>
<tr><td></td><td></td><td>coef</td><td>std err</td><td>t</td><td>P>|t|</td><td>[0.025</td><td>0.975]</td></tr>
<tr><td>عرض از مبدا</td><td>Intercept</td><td>-34.6706</td><td>2.650</td><td>-13.084</td><td>0.000</td><td>-39.877</td><td>-29.465</td></tr>
<tr><td>متغیر مستقل</td><td>RM</td><td>9.1021</td><td>0.419</td><td>21.722</td><td>0.000</td><td>8.279</td><td>9.925</td></tr>
</table>

توجه کنید که آماره آزمون و پی ولیو بدست آمده مربوط به آزمون زیر است:

$$\begin{cases} H_0: \beta_i = 0 \\ H_1: \beta_i \neq 0 \end{cases}$$

که چون پی ولیو برای هر دو بتا صفر است، به این معناست که فرض صفر رد میشود.

خروجی آخر نیز فراتر از سطح درس است و از توضیح آن خودداری میکنیم.

همچنین مدل های با تعداد متغیر پیشبینی بیشتر نیز میتوان ساخت، به طور مثال:

متغیر پیشبینی کننده: RM, CRIM, LSTAT   متغیر هدف: MEDV

همیشه به خاطر داشته باشید که برای پیشبینی به دنبال متغیر هایی هستیم که از هم مستقل باشند و یا همبستگی کمی داشته باشند (چرا که مدلی که نتیجه میدهد قابلیت تفسیر بهتری داشته باشد)، در عین حال همبستگی زیادی با متغیر هدفمان داشته باشند (چرا که بتواند به خوبی مقادیر را پیشبینی کند و نزدیک به مقدار واقعی باشد).

با تحلیل همبستگی این سه متغیر با متغیر هدف داریم:

```
boston[['RM','CRIM' , 'LSTAT','MEDV']].corr()
```

output

	RM	CRIM	LSTAT	MEDV
RM	1.000000	-0.219247	-0.613808	0.695360
CRIM	-0.219247	1.000000	0.455621	-0.388305
LSTAT	-0.613808	0.455621	1.000000	-0.737663
MEDV	0.695360	-0.388305	-0.737663	1.000000

	RM	CRIM	LSTAT	MEDV
RM	1.000000	-0.219247	-0.613808	0.695360
CRIM	-0.219247	1.000000	0.455621	-0.388305
LSTAT	-0.613808	0.455621	1.000000	-0.737663
MEDV	0.695360	-0.388305	-0.737663	1.000000

دو متغیر RM و LSTAT همبستگی خیلی خوبی با MEDV دارند، با وجود اینکه همبستگی CRIM کم است، ولی به علل آموزشی آن را نیز در مدل میاوریم. توجه داشته باشید که همبستگی بین متغیر های پیشبینی RM و LSTAT زیاد است (‎-۰.۶۱‎) و شاید این موضوع که هر دو را در مدل میاوریم خوب نباشد.

مدل رگرسیونی (با برآورد پارامترها):

$$\widehat{MEDV} = \widehat{\beta_0} + \widehat{\beta_1}.RM + \widehat{\beta_2}.CRIM + \widehat{\beta_3}.LSTAT \Rightarrow MEDV \sim RM + CRIM + LSTAT$$

```python
model = sm.regression.linear_model.OLS.from_formula("MEDV ~ RM + CRIM + LSTAT", data=boston)
result = model.fit()
result.summary()
```

output

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.5623	3.166	-0.809	0.419	-8.783	3.658
RM	5.2170	0.442	11.802	0.000	4.348	6.085
CRIM	-0.1029	0.032	-3.215	0.001	-0.166	-0.040
LSTAT	-0.5785	0.048	-12.135	0.000	-0.672	-0.485

با توجه به نتایج بدست آمده:

$$\widehat{MEDV} = -2.562 + 5.217 \times RM - 0.102 \times CRIM - 0.5785 \times LSTAT$$

حال میخواهیم متغیر باینری CHAS را نیز وارد مدل کنیم. برای این کار باید توجه کنیم که ۱ بودن این متغیر به معنای مقدار بیشتر از ۰ نیست. بلکه ۱ یا صفر بودن مشخص کننده کتگوری میباشد:

CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

که مقدار "یک" است اگر مسیر خانه توسط رودخانه محدود شود، و "صفر" در غیر این صورت.

برای این که پایتون هم متوجه این موضوع شود که با یک ستون کتگوریکال سر و کار دارد، تایپ ستون را به category تغییر میدهیم:

```
boston['CHAS']=boston['CHAS'].astype("category")
boston['CHAS'].dtype
```

output

```
CategoricalDtype(categories=[0.0, 1.0], ordered=False)
```

حال میخواهیم مدل را میسازیم،

توجه داشته باشید در ساخت مدل با داشتن یک متغیر کتگوریکال، به تعداد کتگوری منهای یک باید متغیر باینری بسازیم. برای روشن شدن این موضوع یک مثال میزنیم (مثال زده شده در ادامه ربطی به دیتاست ندارد)

مثلا اگر ۳ سطح درآمد «خوب، بد، متوسط» داشته باشیم و بخواهیم در مدل بیاوریم، فقط دو تای آن ها را باید بیاوریم.

$$X_{high} = \begin{cases} 1 & if\ income\ is\ high \\ 0 & ow \end{cases} ;\ X_{low} = \begin{cases} 1 & if\ income\ is\ low \\ 0 & ow \end{cases}$$

حال مدلی میسازیم که شامل وضعیت درآمد باشد:

$$Y = \beta_0 + \beta_1 . X_{high} + \beta_2 . X_{low} + \epsilon$$

حال اگر بعد از فیت کردن مدل و بدست آوردن ضرایب، مقادیر β_2 و β_1 برابر صفر شد (به علت پی ولیو) میتوانیم این نتیجه را بگیریم که درآمد متوسط در پیشبینی تاثیر گذار است (عملا اگر درآمد نه بالا $(\beta_1 = 0)$ باشد نه پایین $(\beta_2 = 0)$، متوسط است.)

به ساخت مدل بر روی دیتاست میپردازیم:

$$MEDV \sim CHAS + CRIM + LSTAT$$

```
model = sm.regression.linear_model.OLS.from_formula("MEDV ~ CHAS+ CRIM + LSTAT", data=boston)
result = model.fit()
result.summary()
```

خروجی در اسلاید بعد؛

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	33.8848	0.571	59.352	0.000	32.763	35.006
CHAS[T.1.0]	4.8511	1.068	4.544	0.000	2.754	6.948
CRIM	-0.0648	0.035	-1.832	0.068	-0.134	0.005
LSTAT	-0.9052	0.043	-21.254	0.000	-0.989	-0.822

میبینیم که با تغییر تایپ ستون CHAS به کتگوریکال، خود statsmodels تشخیص داد که باید با آن به صورت یک متغیر کتگوریکال رفتار کند و یک متغیر باینری بسازد (به صورت خودکار وضعیت ۱ را در نظر گرفت). از آنجایی که ضریب CHAS حدود ۴.۸۵ است و پی ولیو آن صفر است پس میبینیم که این که مسیر خانه در محدوده رودخانه باشد، در پیشبینی ما تاثیر گذار است. (اگر پی ولیو بزرگ بود، به این معنا بود اینکه خانه ای در محدوده رودخانه باشد بر پیشبینی ما تاثیر گذار نیست)

$$\widehat{MEDV} = 33.88 + 4.85 \times CHAS - 0.0.6 \times CRIM - 0.90 \times LSTAT$$

❖ آزمون همبستگی:

گذر خیلی کوتاهی به کتابخانه seaborn میزنیم. در حد اینکه همبستگی بین متغیر ها را بهتر درک کنیم:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(10,7))
corr = boston.corr()
sns.heatmap(corr, annot=True)
plt.show()
```

output

آزمونی که میخواهیم انجام دهیم از قرار زیر است:

$$\begin{cases} H_0\text{: the two samples are independent } (\rho = 0) \\ H_1\text{: there is a dependency between the samples } (\rho \neq 0) \end{cases}$$

در گام اول این آزمون را بین دو داده MEDV و LSTAT انجام میدهیم:

```
stats.pearsonr(boston.MEDV , boston.LSTAT)
```

output

$\underbrace{\texttt{(-0.7376627261740147,}}_{\textbf{correlation}}\ \underbrace{\texttt{5.08110339438754 7e-88)}}_{\textbf{pval}}$

که به وضوح میگوید که همبستگی غیر صفر است.

در گام دوم این آزمون را برای دو داده رندوم امتحان میکنیم:

```
stats.pearsonr(np.random.randint(1,51,size=50) , np.random.randint(1,51,size=50))
```

output

$\underbrace{\texttt{(0.05410914732757818,}}_{\textbf{correlation}}\ \underbrace{\texttt{0.7089953013928426)}}_{\textbf{pval}}$

چون پی ولیو بزرگ است، دلیلی برای رد فرض صفر نداریم.

❖ طرح های یک طرفه

توجه داشته باشید ۳ فرض اساسی برای استفاده از آزمون ANOVA یک طرفه از قرار زیر است:

1. نرمال بودن – هر نمونه باید از یک جامعه نرمال گرفته شود.
2. واریانس های برابر – واریانس های جوامعی که نمونه ها از آن ها می آیند باید برابر باشند.
3. استقلال – مشاهدات هر گروه مستقل از یکدیگر بوده و مشاهدات درون گروهی به صورت تصادفی به دست آمده باشند.

همچنین آزمونی که میخواهیم انجام دهیم:

$$\begin{cases} H_0: \mu_1 = \mu_2 = \cdots = \mu_k \\ H_1: ow \end{cases}$$

در اسلاید بعد، به بررسی یک مثال میپردازیم.

۱۹.۱۵ اعداد زیر تعداد کلماتی را که یک منشی در هر دقیقه در زمانهای مختلف با چهار ماشین تحریر مختلف تایپ کرده است، نشان می‌دهند.

ماشین تحریر C: ۷۱، ۷۵، ۶۹، ۷۷، ۶۱، ۷۲، ۷۱، ۷۸
ماشین تحریر D: ۶۸، ۷۱، ۷۴، ۶۶، ۶۹، ۶۷، ۷۰، ۶۲
ماشین تحریر E: ۷۵، ۷۰، ۸۱، ۷۳، ۷۸، ۷۲
ماشین تحریر F: ۶۲، ۵۹، ۷۱، ۶۸، ۶۳، ۶۵، ۷۲، ۶۰، ۶۴

با استفاده از فرمولهای محاسباتی تمرین ۴.۱۵ برای محاسبهٔ مجموعهای مربعات، در سطح معنی‌دار بودن ۰۵ر۰، آزمون کنید که آیا اختلاف بین چهار میانگین نمونه‌ای را می‌توان به‌تصادف نسبت داد یا نه؟

با فرض اینکه جوامع نرمال هستند و از هم مستقل میباشند (شرط یک و سه)، به بررسی شرط دوم میرسیم:

```
C=[71,75,69,77,61,72,71,78]
D=[68,71,74,66,69,67,70,62]
E=[75,70,81,73,78,72]
F=[62,59,71,68,63,65,72,60,64]

print(np.std(C))
print(np.std(D))
print(np.std(E))
print(np.std(F))
```

output

```
5.018714974971183
3.3517719194479807
3.7155828016013253
4.331908597692873
```

برای اینکه مطمئن شویم که واریانس ها برابر هستند یا نه، از آزمون bartlett استفاده میکنیم (که قبلا نیز به آن اشاره کردیم):

$$\begin{cases} H_0: \sigma_C^2 = \sigma_D^2 = \sigma_E^2 = \sigma_F^2 \\ H_1: ow \end{cases}$$

```
stats.bartlett(C,D,E,F)
```

output

```
BartlettResult(statistic=1.144176592674825, pvalue=0.7664217809102014)
```

پس شرط دوم نیز برای one-way ANOVA برقرار است؛ آزمون را انجام میدهیم:

```
stats.f_oneway(C,D,E,F)
```

output

```
F_onewayResult(statistic=6.838190783150868, pvalue=0.0014184643266147632)
```

که چون پی ولیو کم است، یعنی میتوان فرض صفر را رد کرد، پس اختلاف بین میانگین ها معنا دار است (به اصطلاح significant است) و شانسی نیست.