**Конспект эконометрика.**

**Условные обозначения:**

То, что нужно скопировать, не выделяю. То, что нужно заменить на другие данные выделяю желтым.

Серым выделяю редко нужную фигню.

**Питон для всех моделей:**

**import** numpy **as** np

**import** pandas **as** pd

**import** statsmodels.formula.api **as** smf

**import** statsmodels.api **as** sm

**from** statsmodels.tools **import** add\_constant

**from** statsmodels.iolib.summary2 **import** summary\_col, summary\_params

**from** scipy.stats **import** t *# t-распределение*

**from** scipy.stats **import** norm *# нормальное распределение*

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** statsmodels.formula.api as smf

**from** statsmodels.iolib.summary2 import summary\_params

**from** scipy.stats import t # t-распределение

**from** scipy.stats import f # F-распределение

**from** scipy.stats import chi2 # критические значения chi2

***# подключение датасета.***

df **=** pd**.**read\_csv('НАЗВАНИЕ ФАЙЛА.csv', na\_values**=**(' ', '', ' '))

*# посмотрим на первые пять строк датасета*

df**.**head(n**=**5)

*# размеры датасета*

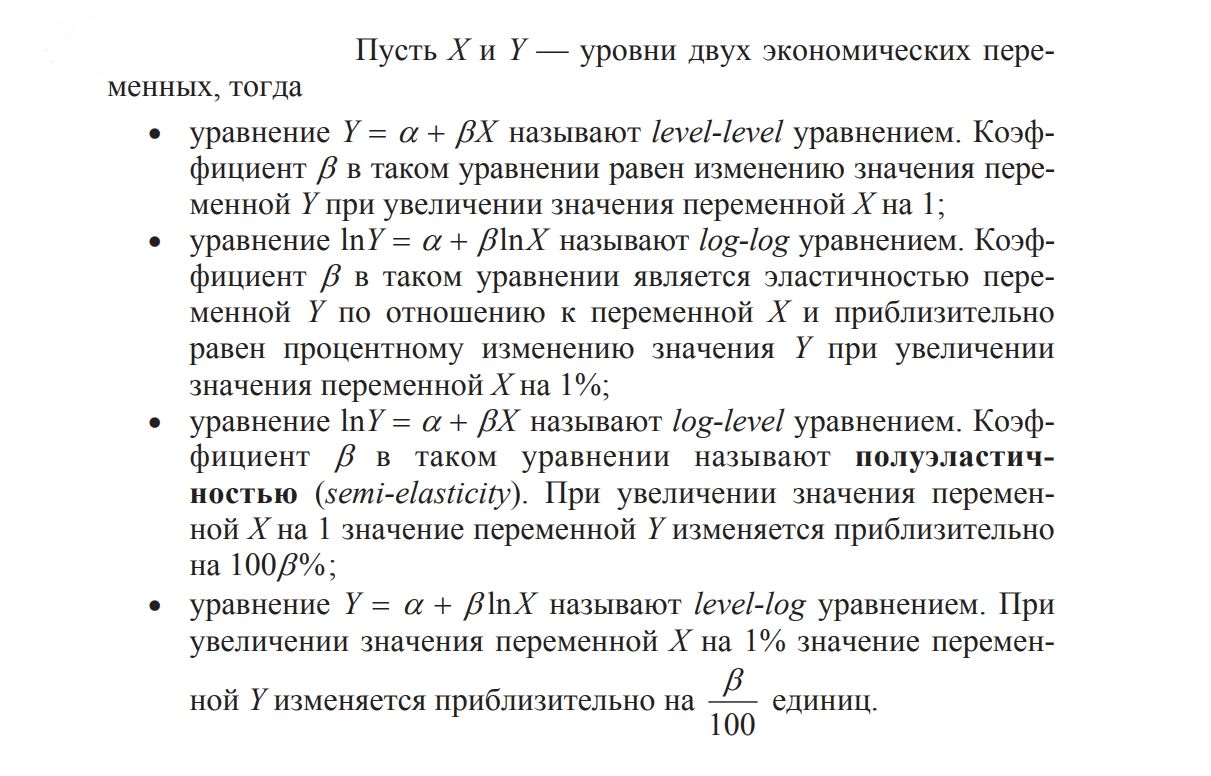
df**.**shape

*# информация о датасете*

df**.**info()

*#* Если нужно записать квадрат параметра (потом в модели вписываем I)

I=df.ПАРАМЕТР\*\*2



**LPM модель (линейная регрессия)**

**Запись модели:**   
*approve=β0+β1mortno+β2unem+β3dep+β4male+β5married+β6yjob+β7self+u  
ЛИБО: P(approve=1)=β0+β1mortno+β2unem+β3dep+β4male+β5married+β6yjob+β7self*

**Питон:**

**Включаем LPM модель**

mod\_lpm **=** smf**.**ols(formula**=**'approve~параметр+…+параметр', data**=**df)

***# подгонка модели***

res\_lpm\_hc **=** mod\_lpm**.**fit(cov\_type**=**'HC3')

***# результаты регрессии***  
print(res\_lpm\_hc**.**summary(slim**=True**))

*# Сравнение моделей (Опционально)*

res\_lpm\_ols **=** mod\_lpm**.**fit(cov\_type**=**'nonrobust')

print(summary\_col([res\_lpm\_hc, res\_lpm\_ols], model\_names**=**['Robust', 'Non-robust'], stars**=True**))

***# Коэфициенты модели с округлением до 3-х десятичных знаков***

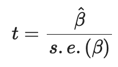
res\_lpm\_hc**.**params**.**round(3)

**ИНТЕРПРЕТАЦИЯ**

**- Если бинарная** пишем так: для регрессора \_\_\_\_\_\_\_\_\_ разница вероятностей между 0 и 1 равна \_\_

**- Если количественная** пишем так: при увеличении регрессора \_\_\_\_\_\_\_\_\_ на единицу, вероятность успеха уменьшится или увеличится на \_\_\_\_\_\_\_\_\_

**T-test (тест значимости коэф. LPM модели)**

*Запись теста:  
  
Вариант 1: – Т-статистика - делим бету (Coef.) на стандартную ошибку (Std.Err).   
  
 - Т-критическая – находится по таблице распределения Стьюдента, по пересечению* ***уровня значимости* α** (обычно равно 1%,5% или 10%) и степеней несвободы df (**df=n** *(number of observations)* **– k** *(количество коэфициентов без constant или intersept)* **- 1**)  
***Сравниваем. Если* |tst| > tcr, то Х при этом регресоре значим.** (tst берём по МОДУЛЮ!!!)  
  
Вариант 2. Через P – значения. Если P< **α,** то коэфициент значим.

**Питон:**

***Считаем отдельно каждый параметр для варианта 1:***

***# t-статистика для каждого коэффициента с округлением до 3-х десятичных знаков***

np**.**round(res\_lpm\_hc**.**tvalues, 3)

***# Число наблюдений в модели*** *(number of observations)*

res\_lpm\_hc**.**nobs

***# число регрессоров*** *(количество коэфициентов без constant или intersept)*

res\_lpm\_hc**.**df\_model

***# степени свободы (df)***

res\_lpm\_hc**.**df\_resid

***# Вывести tcr с округлением до 3-х десятичных знаков при* α = 1%**

np**.**round(t**.**ppf(q**=**1**-**0.01**/**2, df**=**res\_lpm\_hc**.**df\_resid), 3)

***# P-значения для t-статистик с округленим до 3-х десятичных знаков***

res.pvalues.round(3)

**# Сначала обязательно должна быть прогнана LPM модель**

**# Показываем результаты *t-теста для коэффициентов (неробастные s.e.* ( \_ols)*)***

res\_lpm\_ols = mod\_lpm.fit(cov\_type='nonrobust')

summary\_params(res\_lpm\_ols, alpha**=**0.01)

***# Выводим значимость коэффициентов***

df\_ols **=** np**.**round(summary\_params(res\_lpm\_ols, alpha**=**0.01), 3)

df\_ols['significance'] **=** df\_ols**.**apply(**lambda** x: 'Значим' **if** x['P>|t|']**<**0.01 **else** 'Незначим', axis**=**1)

df\_ols

**# Показываем результаты *t-теста для коэффициентов (робастные s.e.* ( \_hc)*)***

summary\_params(res\_lpm\_hc, alpha**=**0.01)

***# Выводим значимость коэффициентов***

df\_hc **=** np**.**round(summary\_params(res\_lpm\_hc, alpha**=**0.01), 3)

df\_hc['significance'] **=** df\_hc**.**apply(**lambda** x: 'Значим' **if** x['P>|t|']**<**0.01 **else** 'Незначим', axis**=**1)

df\_hc

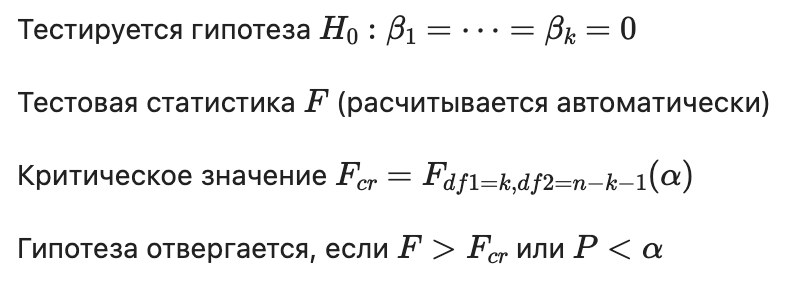
***# Можем вывести значимость отдельных коэффициентов (робастные s.e.)***

res\_lpm\_hc.t\_test('**Коэфициент**=0, **Коэфициент2**=0')

**ИНТЕРПРЕТАЦИЯ (Значимость)**

*Если***|tst| > tcr,** то **Х** при этом регресоре **значим.** (tst берём по МОДУЛЮ!!!)  
  
Через P – значения. Если P< **α,** то коэфициент значим.

**F-test (тест значимости всей LPM модели)**

**Запись модели:**   
***Два варианта, как и в t-test.******Вариант 1:*** *Через сравнение F-статистики и F-критического. Прогоняем LPM модель, там есть* ***F-statistic.*** *Затем считаем F-критическое одной строчкой и сравнивааем. Если Fst>Fcr, регрессия значима.****Вариант 2:*** *Через P, как и в t-тесте. Когда прогоняем LPM модель, мы получаем* ***Prob (по F-statistic)****, затем сравниваем с* ***уровнем значимости* α** (обычно равно 1%,5% или 10%). Если P< **α**, регрессия значима.

**Питон:**

**Сначала нужно прогнать LPM модель:**

mod\_lpm **=** smf**.**ols(formula**=**'approve~параметр+…+параметр', data**=**df)

***# подгонка модели***

res\_lpm\_hc **=** mod\_lpm**.**fit(cov\_type**=**'HC3')

***# результаты регрессии (тут смотрим F-statistic и Prob (по F-statistic))***  
print(res\_lpm\_hc**.**summary(slim**=True**))

**# F - статистика и P-значение (РОБАСТНЫЕ)**

np.round(res\_lpm\_hc.fvalue, 3), np.round(res\_lpm\_hc.f\_pvalue, 3)

**# тестовая статистика и P-значение (НЕ РОБАСТНЫЕ)**

res\_lpm\_ols = mod\_lpm.fit(cov\_type='nonrobust')

res\_lpm\_ols.fvalue.round(3), res\_lpm\_ols.f\_pvalue.round(3)

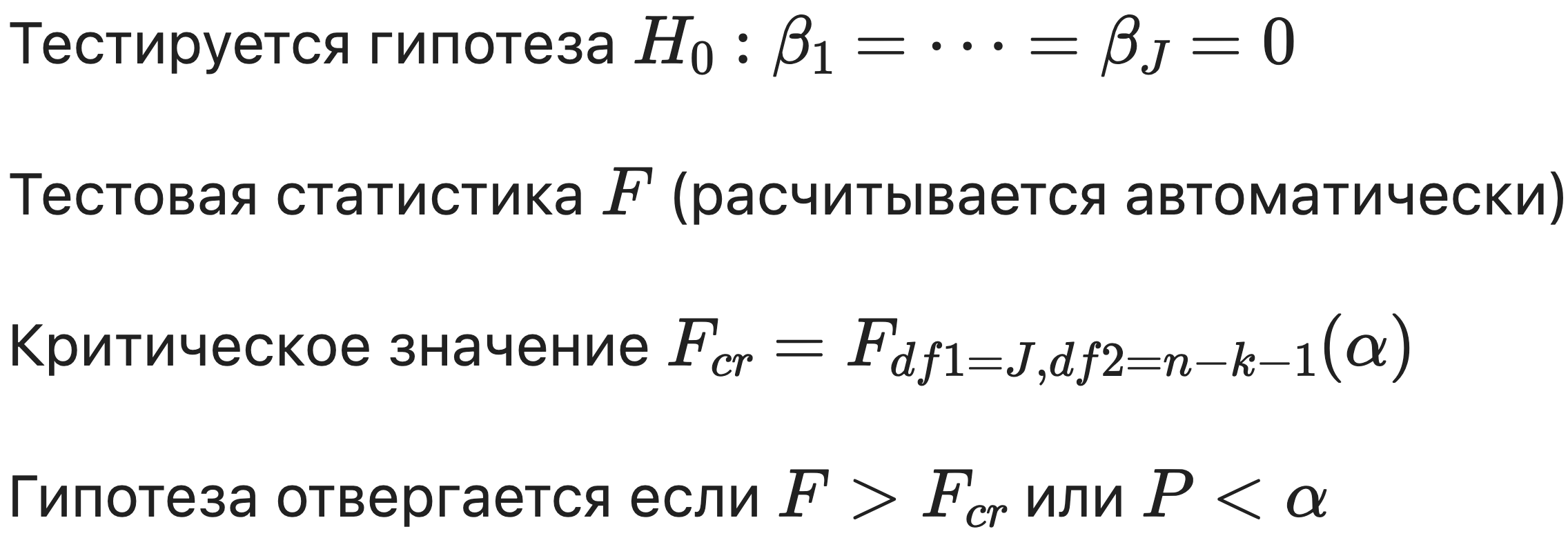
**# Выводи F-критическое, при α = 10%**

f.ppf(q=1-0.1, dfn=res\_lpm\_hc.df\_model, dfd=res\_lpm\_hc.df\_resid).round(3)

**ИНТЕРПРЕТАЦИЯ**

*Если***Fst> Fcr,** торегрессия **значима.**  
  
Через P – значения. Если P***(по F-statistic)***< **α,** то регрессия значима.

**F-test (тест значимости отдельных коэф. LPM модели)**

**Запись модели:**   
***Разница*** *только в* ***df1=J (количество выбранных коэфициентов)  
Два варианта, как и в t-test.******Вариант 1:*** *Через сравнение F-статистики и F-критического. Прогоняем LPM модель, там есть* ***F-statistic.*** *Затем считаем F-критическое одной строчкой и сравнивааем. Если Fst>Fcr, регрессия значима.****Вариант 2:*** *Через P, как и в t-тесте. Когда прогоняем LPM модель, мы получаем* ***Prob (по F-statistic)****, затем сравниваем с* ***уровнем значимости* α** (обычно равно 1%,5% или 10%). Если P< **α**, регрессия значима.

**Питон:**

**Сначала нужно прогнать LPM модель:**

mod\_lpm **=** smf**.**ols(formula**=**'approve~параметр+…+параметр', data**=**df)

***# подгонка модели***

res\_lpm\_hc **=** mod\_lpm**.**fit(cov\_type**=**'HC3')

***# результаты регрессии (тут смотрим F-statistic и Prob (по F-statistic))***  
print(res\_lpm\_hc**.**summary(slim**=True**))  
**# F-статистика и P-значение**

f.ppf(q=1-**0.05,** dfn=2, dfd=res\_lpm\_hc.df\_resid).round(3)

**# F-критическое и P-значение (РОБАСТНЫЕ)**

I=df.appinc\*\*2

**# тестовая статистика, P-значение и степени свободы**

print(res\_lpm\_hc.f\_test('appinc=I=0'))

**# тестовая статистика, P-значение и степени свободы**

print(res\_lpm\_hc.wald\_test('appinc=I=0', use\_f=True))

**# F-критическое и P-значение (НЕ РОБАСТНЫЕ)**

**# подгонка модели**

res\_lpm\_ols = mod\_lpm.fit(cov\_type='nonrobust')

I=df.appinc\*\*2

**# тестовая статистика, P-значение и степени свободы**

print(res\_lpm\_ols.f\_test('appinc=I=0'))

**ИНТЕРПРЕТАЦИЯ**

*Если***Fst> Fcr,** торегрессия **значима.**  
  
Через P – значения. Если P< **α,** то регрессия значима.

**Probit модель (Φ)**

Запись модели:  
P(approve=1)=Φ(β0+β1appinc+β2mortno+β3unem+β4dep+β5male+β6married+β7yjob+β8self)  
ЛИБО:  
probit(P(approve=1))=β0+β1appinc+β2mortno+β3unem+β4dep+β5male+β6married+β7yjob+β8self

**Питон:**

**Включаем Probit модель**

mod = smf.probit(formula='approve~ параметр+…+параметр', data=df)

**# подгонка модели**

res = mod.fit()

***# результаты регрессии***

*res.summary()*

***# коэффициенты подогнанной модели с округлением***

*res.params.round(3)*

**ИНТЕРПРЕТАЦИЯ (Только знак при регрессоре)**

**- Если бинарная** пишем так: для людей с регрессором (женатых, трудоустроенных) \_\_\_\_\_\_\_\_\_ вероятность успеха увеличивается (уменьшается), чем для людей с обратным значением (неженатых, нетрудоустроенных).

**- Если количественная** пишем так: при увеличении регрессора \_\_\_\_\_\_\_\_\_ на единицу, вероятность успеха умешается или увеличивается (смотрим на знак).

**Logit модель (Λ)**

Запись модели:  
P(approve=1)=Λ(β0+β1appinc+β2mortno+β3unem+β4dep+β5male+β6married+β7yjob+β8self)  
ЛИБО  
logit(P(approve=1))=β0+β1appinc+β2mortno+β3unem+β4dep+β5male+β6married+β7yjob+β8self  
ЛИБО   
logit(P(approve=1))=logP(approve=1)/(1−P(approve=1))=logP(approve=1)/(P(approve=0))

**Питон:**

**Включаем Probit модель**

mod = smf.logit(formula='approve~параметр+…+параметр', data=df)

**# подгонка модели**

res = mod.fit()

***# результаты регрессии***

*res.summary()*

***# коэффициенты подогнанной модели с округлением***

*res.params.round(3)*

**ИНТЕРПРЕТАЦИЯ (Отношение шансов)**

**- Если бинарная** пишем так: для людей с регрессором (женатых, трудоустроенных) \_\_\_\_\_\_\_\_\_ отношение шансов больше (меньше) на \_\_%, чем для людей с обратным значением (неженатых, нетрудоустроенных).

**- Если количественная** пишем так: при увеличении регрессора \_\_\_\_\_\_\_\_\_ на единицу, отношение шансов уменьшается (увеличивается) на \_\_%.

**Z-test (тест значимости коэф. Probit и Logit моделей)**

*Полностью идентичен T-тесту, только Z-критическая – находится по таблице* ***нормального распределения.***

**Питон:**

***Считаем отдельно каждый параметр для варианта 1:***

***# t-статистика для каждого коэффициента с округлением до 3-х десятичных знаков***

np**.**round(res\_lpm\_hc**.**tvalues, 3)

***# Число наблюдений в модели*** *(number of observations)*

res\_lpm\_hc**.**nobs

***# число регрессоров*** *(количество коэфициентов без constant или intersept)*

res\_lpm\_hc**.**df\_model

***# степени свободы (df)***

res\_lpm\_hc**.**df\_resid

***# Вывести zcr с округлением до 3-х десятичных знаков при* α = 1%**

sign\_level = 0.01

norm.ppf(q=1-sign\_level/2)

***# P-значения для t-статистик с округленим до 3-х десятичных знаков***

res.pvalues.round(3)

**# Показываем результаты *z-теста для коэффициентов (неробастные s.e.* ( \_ols)*)***

res\_lpm\_ols = mod\_lpm.fit(cov\_type='nonrobust')

summary\_params(res\_lpm\_ols, alpha**=**0.01)

***# Выводим значимость коэффициентов***

df\_ols **=** np**.**round(summary\_params(res\_lpm\_ols, alpha**=**0.01), 3)

df\_ols['significance'] **=** df\_ols**.**apply(**lambda** x: 'Значим' **if** x['P>|t|']**<**0.01 **else** 'Незначим', axis**=**1)

df\_ols

**# Показываем результаты *t-теста для коэффициентов (робастные s.e.* ( \_hc)*)***

summary\_params(res\_lpm\_hc, alpha**=**0.01)

***# Выводим значимость коэффициентов***

df\_hc **=** np**.**round(summary\_params(res\_lpm\_hc, alpha**=**0.01), 3)

df\_hc['significance'] **=** df\_hc**.**apply(**lambda** x: 'Значим' **if** x['P>|t|']**<**0.01 **else** 'Незначим', axis**=**1)

df\_hc

***# Можем вывести значимость отдельных коэффициентов (робастные s.e.)***

res\_lpm\_hc.t\_test('**Коэфициент**=0, **Коэфициент2**=0')

**ИНТЕРПРЕТАЦИЯ (Значимость)**

*Если***|zst| > zcr,** то **Х** при этом регресоре **значим.** (tst берём по МОДУЛЮ!!!)  
  
Через P – значения. Если P< **α,** то коэфициент значим.

**LR-test (тест значимости всей Probit или Logit модели)**

*LR = 2(logLlikelihood – logLnull)  
  
Если LR>x2df(***α),** то регрессия **значима**Если P< **α**, регрессия значима.

**Питон:**

***Считаем отдельно каждый параметр:***

**# Число наблюдений, по которым была оценена модель**

res.nobs

**# Loglikelihood**

res.llf.round(3)

**# Lognull**

res.llnull.round(3)

**# Тестовая статистика LR-теста и её P-значение с округленим**

res.llr.round(3), res.llr\_pvalue.round(3)

**# Степени свободы для х2 распределения**

res.df\_model

**# Критические значения x2df(α)**

chi2.ppf(q=1-0.05, df=res.df\_model).round(3)

**ИНТЕРПРЕТАЦИЯ (Значимость)**

*Если LR>x2df(***α),** то регрессия **значима**

Если P< **α**, регрессия **значима**.

**LR-test (тест значимости отдельных коэф. Probit или Logit модели)**

*LR = 2(logLlikelihood – logLr)  
  
Если LR>x2df(***α),** то регрессия **значима**Если P< **α**, регрессия значима.

**Питон:**

***Считаем отдельно каждый параметр:***

**# Число наблюдений, по которым была оценена модель**

res.nobs

**# Loglikelihood**

res.llf.round(3)

**# Lognull**

res.llnull.round(3)

**# подгонка модели с ограничениями**

df\_mod = df[['approve','appinc','mortno','unem','dep', 'male', 'married', 'yjob', 'self']].dropna()

mod\_r = smf.logit(formula ='approve~mortno+unem+dep+male+married+yjob+self', data = df\_mod)

res\_r = mod\_r.fit()

res\_r.nobs # число наблюдений, на которых была подогнана модель

**# Тестовая статистика LR-теста с округленим (Значение LR)**

lr\_stat=2\*(res.llf-res\_r.llf)

lr\_stat.round(3)

**# Степени свободы для х2 распределения**

res.df\_model-res\_r.df\_model

**# P-значение тестовой статистики LR-теста с округленим**

lr\_pvalue = chi2.sf(lr\_stat, df=res.df\_model-res\_r.df\_model)

lr\_pvalue.round(3)

**# Критические значения x2df(α)**

sign\_level = 0.01 # уровень значимости

chi2.ppf(q=1-sign\_level, df=res.df\_model-res\_r.df\_model).round(3)

**ИНТЕРПРЕТАЦИЯ (Значимость)**

*Если LR>x2df(***α),** то регрессия **значима**

Если P< **α**, регрессия значима.