Linear Models in Python

April 6, 2020

[1]:
```
import statsmodels.api as sm

from statsmodels.formula.api import ols, glm

import pandas as pd
```

[4]:
```
file_path = '/Users/MuhammadBilal/Desktop/Data Camp/Generalized linear models␣
↪in python/Data/salary.csv'
salary = pd.read_csv(file_path)
salary.head()
```

[4]:

	Experience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

[5]:
```
# Fitting a linear model
model_lm = ols(formula = 'Salary ~ Experience',
               data = salary).fit()

# Viewing model coefficients
print(model_lm.params)

# Fitting a GLM
model_glm = glm(formula = 'Salary ~ Experience',
                data = salary,
                family = sm.families.Gaussian()).fit()

# Viewing model coefficients
print(model_glm.params)
```

```
Intercept     25792.200199
Experience     9449.962321
dtype: float64
Intercept     25792.200199
```

```
Experience     9449.962321
dtype: float64
```

[ ]:
```
# Looking at the coefficient estimates notice how both models give the same␣
↪values.
```

[6]:
```
file_path1 = '/Users/MuhammadBilal/Desktop/Data Camp/Generalized linear models␣
↪in python/Data/crab.csv'
crab = pd.read_csv(file_path1)
print(crab)
```

```
     crab  sat  y  weight  width  color  spine         width_C
0       1    8  1   3.050   28.3      2      3  [28.25, 29.25)
1       2    0  0   1.550   22.5      3      3    [0.0, 23.25)
2       3    9  1   2.300   26.0      1      1  [25.25, 26.25)
3       4    0  0   2.100   24.8      3      3  [24.25, 25.25)
4       5    4  1   2.600   26.0      3      3  [25.25, 26.25)
..     …    … ..      …      …      …      …               …
168   169    3  1   2.750   26.1      3      3  [25.25, 26.25)
169   170    4  1   3.275   29.0      3      3  [28.25, 29.25)
170   171    0  0   2.625   28.0      1      1  [27.25, 28.25)
171   172    0  0   2.625   27.0      4      3  [26.25, 27.25)
172   173    0  0   2.000   24.5      2      2  [24.25, 25.25)

[173 rows x 8 columns]
```

[7]:
```
# Creating a linear model from the Gaussian family and a logistic regression␣
↪model from the Binomial family to fit to the dataset.

# Defining model formula
formula = 'y ~ width'
```

[8]:
```
# Defining probability distribution for the response variable for
# the linear (LM) and logistic (GLM) model
family_LM = sm.families.Gaussian()
family_GLM = sm.families.Binomial()
```

[9]:
```
# Defining and fitting a linear regression model
model_LM = glm(formula = formula, data = crab, family = family_LM).fit()
print(model_LM.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:                Gaussian   Df Model:                            1
Link Function:               identity   Scale:                         0.19515
Method:                          IRLS   Log-Likelihood:                -103.13
```

```
Date:                 Mon, 30 Mar 2020   Deviance:                       33.371
Time:                         23:36:53   Pearson chi2:                     33.4
No. Iterations:                      3
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -1.7655      0.421     -4.190      0.000      -2.591      -0.940
width          0.0915      0.016      5.731      0.000       0.060       0.123
==============================================================================
```

[10]:
```python
# Defining and fitting a logistic regression model
model_GLM = glm(formula = formula, data = crab, family = family_GLM).fit()
print(model_GLM.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                       y   No. Observations:                  173
Model:                             GLM   Df Residuals:                      171
Model Family:                 Binomial   Df Model:                            1
Link Function:                   logit   Scale:                          1.0000
Method:                           IRLS   Log-Likelihood:                -97.226
Date:                 Mon, 30 Mar 2020   Deviance:                       194.45
Time:                         23:37:30   Pearson chi2:                     165.
No. Iterations:                      4
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -12.3508      2.629     -4.698      0.000     -17.503      -7.199
width          0.4972      0.102      4.887      0.000       0.298       0.697
==============================================================================
```

[ ]:
```python
# Comparing predicted values
# In the above exercise, a linear and a GLM (logistic) regression model are␣
 ↪fitted using crab data, predicting y with width. In other words, the␣
 ↪probability that the female has a satellite crab nearby given her width is␣
 ↪predicted.
# In the following codes, the estimated probabilities (the output) from the two␣
 ↪models are examined and it will be deduced if the linear fit would be␣
 ↪suitable for the problem at hand.
```

[11]:
```python
test = crab.head()

# Viewing test set
print(test)
```

```
   crab  sat  y  weight  width  color  spine        width_C
```

```
0     1    8  1    3.05   28.3      2      3  [28.25, 29.25)
1     2    0  0    1.55   22.5      3      3    [0.0, 23.25)
2     3    9  1    2.30   26.0      1      1  [25.25, 26.25)
3     4    0  0    2.10   24.8      3      3  [24.25, 25.25)
4     5    4  1    2.60   26.0      3      3  [25.25, 26.25)
```

[12]:
```python
# Computing estimated probabilities for linear model: pred_lm
pred_lm = model_LM.predict(test)

# Computing estimated probabilities for GLM model: pred_glm
pred_glm = model_GLM.predict(test)

# Creating dataframe of predictions for linear and GLM model: predictions
predictions = pd.DataFrame({'Pred_LM': pred_lm, 'Pred_GLM': pred_glm})

# Concatenating test sample and predictions and viewing the results
all_data = pd.concat([test, predictions], axis = 1)
print(all_data.head())
```

```
   crab  sat  y  weight  width  color  spine         width_C   Pred_LM  \
0     1    8  1    3.05   28.3      2      3  [28.25, 29.25)  0.824786
1     2    0  0    1.55   22.5      3      3    [0.0, 23.25)  0.293907
2     3    9  1    2.30   26.0      1      1  [25.25, 26.25)  0.614265
3     4    0  0    2.10   24.8      3      3  [24.25, 25.25)  0.504428
4     5    4  1    2.60   26.0      3      3  [25.25, 26.25)  0.614265

   Pred_GLM
0  0.848233
1  0.238099
2  0.640418
3  0.495125
4  0.640418
```

[ ]:
```python
# Comparing the predicted values for both models, the GLM model provides values␣
↪within the (0,1) range as is required by the binary response variable.
```

[13]:
```python
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import glm

file_path2 = '/Users/MuhammadBilal/Desktop/Data Camp/Generalized linear models␣
↪in python/Data/wells.csv'

wells = pd.read_csv(file_path2)
```

[ ]:
```python
# Model fitting step-by-step
```

```
# In the following codes the components of the GLM will be defined by following␣
↪statsmodels package step by step and finally a model will be fitted by␣
↪calling the .fit() method.
```

[14]:
```
# Defining the formula for the logistic model
model_formula = 'switch ~ distance100'

# Defining the correct probability distribution and the link function of the␣
↪response variable
link_function = sm.families.links.logit
```

[15]:
```
model_family = sm.families.Binomial(link = link_function)

# Fitting the model
wells_fit = glm(formula = model_formula,
                data = wells,
                family = model_family).fit()

# Viewing the results of the wells_fit model
wells_fit.summary()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
DeprecationWarning: Calling Family(..) with a link class as argument is
deprecated.
Use an instance of a link class instead.
  """Entry point for launching an IPython kernel.
```

[15]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                 switch   No. Observations:                 3010
Model:                            GLM   Df Residuals:                     3008
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2030.6
Date:                Mon, 30 Mar 2020   Deviance:                       4061.3
Time:                        23:43:38   Pearson chi2:                 3.01e+03
No. Iterations:                     4
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept       0.6108      0.060     10.104      0.000       0.492       0.729
distance100    -0.6291      0.098     -6.446      0.000      -0.820      -0.438
===============================================================================
"""
```

[ ]:
```python
# Extracting parameter estimates
```

[16]:
```python
# Extract coefficients from the fitted model wells_fit
intercept, slope = wells_fit.params

# Print coefficients
print('Intercept =', intercept)
print('Slope =', slope)

# Extract and print confidence intervals
print(wells_fit.conf_int())
```

```
Intercept = 0.6108118803818955
Slope = -0.6290808479557684
                   0         1
Intercept    0.492327  0.729297
distance100 -0.820345 -0.437816
```

[17]:
```python
# Compute the odds
odds = 15/(60-15)

# Print the result
print('Odds are: ', round(odds,3))
```

```
Odds are:  0.333
```

[18]:
```python
# Load libraries and functions
import statsmodels.api as sm
from statsmodels.formula.api import glm

# Fit logistic regression model
model_GLM = glm(formula = 'switch ~ arsenic',
                data = wells,
                family = sm.families.Binomial()).fit()

# Print model summary
print(model_GLM.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                 switch   No. Observations:                 3010
Model:                            GLM   Df Residuals:                     3008
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -1997.3
Date:                Tue, 31 Mar 2020   Deviance:                       3994.6
Time:                        17:24:58   Pearson chi2:                 3.03e+03
No. Iterations:                     4
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.3058      0.070     -4.340      0.000      -0.444      -0.168
arsenic        0.3799      0.039      9.837      0.000       0.304       0.456
==============================================================================
```

[ ]:
```
# We will continue with the data from the study on the contamination of ground water with arsenic in Bangladesh where we want to model the probability of switching the current well given the level of arsenic present in the well.
```

[19]:
```
# Load libraries and functions
import statsmodels.api as sm
from statsmodels.formula.api import glm
import numpy as np

# Fit logistic regression model
model_GLM = glm(formula = 'switch ~ distance100',
                data = wells,
                family = sm.families.Binomial()).fit()

# Extract model coefficients
print('Model coefficients: \n', model_GLM.params)

# Compute the multiplicative effect on the odds
print('Odds: \n', np.exp(model_GLM.params))
```

```
Model coefficients:
 Intercept      0.610812
distance100   -0.629081
dtype: float64
Odds:
 Intercept      1.841926
distance100    0.533082
dtype: float64
```

[ ]:
```
# The odds of switching the well is 1/2 for a 1-unit (100m) increase in distance, so for every one switch (household switches to the nearest safe well) there would be 2 households who would not switch to the nearest safe well.
```

[21]:
```
model_GLM.summary()
```

[21]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
```

```
==============================================================================
Dep. Variable:                 switch   No. Observations:                 3010
Model:                            GLM   Df Residuals:                     3008
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2030.6
Date:                Tue, 31 Mar 2020   Deviance:                       4061.3
Time:                        21:52:18   Pearson chi2:                 3.01e+03
No. Iterations:                     4
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept       0.6108      0.060     10.104      0.000       0.492       0.729
distance100    -0.6291      0.098     -6.446      0.000      -0.820      -0.438
===============================================================================
"""
```

[ ]:
```
# With one-unit increase in distance100 the log odds decrease by -0.6219.␣
↪Implying that the probability of household switching the well decreases.
```

[23]:
```
# Defining x at 1.5
x = 1.5

# Extract intercept & slope from the fitted model
intercept, slope = model_GLM.params
```

[24]:
```
# Compute and print the estimated probability
est_prob = np.exp(intercept + slope*x)/(1+np.exp(intercept + slope*x))
print('Estimated probability at x = 1.5: ', round(est_prob, 4))
```

```
Estimated probability at x = 1.5:  0.4176
```

[25]:
```
# Compute the slope of the tangent line for parameter beta at x
slope_tan = slope * est_prob * (1 - est_prob)
print('The rate of change in probability: ', round(slope_tan,4))
```

```
The rate of change in probability:  -0.153
```

[ ]:
```
# So at the distance100 value of 1.5 the estimated probability is 0.419 with␣
↪the rate of change in the estimated probability of negative 0.1514. This␣
↪means that for every 1oo m increase in distance100 at the distance100 value␣
↪of 1.5 the probability of well switch decreases by 15,3%.
```

[26]:
```
# Import libraries and th glm function
import statsmodels.api as sm
from statsmodels.formula.api import glm
```

```python
# Fit logistic regression and save as crab_GLM
crab_GLM = glm('y ~ width', data = crab, family = sm.families.Binomial()).fit()

# Print model summary
print(crab_GLM.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:                Binomial   Df Model:                            1
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -97.226
Date:                Tue, 31 Mar 2020   Deviance:                       194.45
Time:                        23:41:28   Pearson chi2:                     165.
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -12.3508      2.629     -4.698      0.000     -17.503      -7.199
width          0.4972      0.102      4.887      0.000       0.298       0.697
==============================================================================
```

[ ]:
```python
# There is a positive significant relationship (width increases the chance of a satellite).
```

[27]:
```python
# Extract coefficients
intercept, slope = crab_GLM.params

# Estimated covariance matrix: crab_cov
crab_cov = crab_GLM.cov_params()
print(crab_cov)

# Compute standard error (SE): std_error
std_error = np.sqrt(crab_cov.loc['width', 'width'])
print('SE: ', round(std_error, 4))

# Compute Wald statistic
wald_stat = slope/std_error
print('Wald statistic: ', round(wald_stat,4))
```

```
           Intercept     width
Intercept   6.910158 -0.266848
width      -0.266848  0.010350
SE:  0.1017
```

```
Wald statistic:  4.8875
```

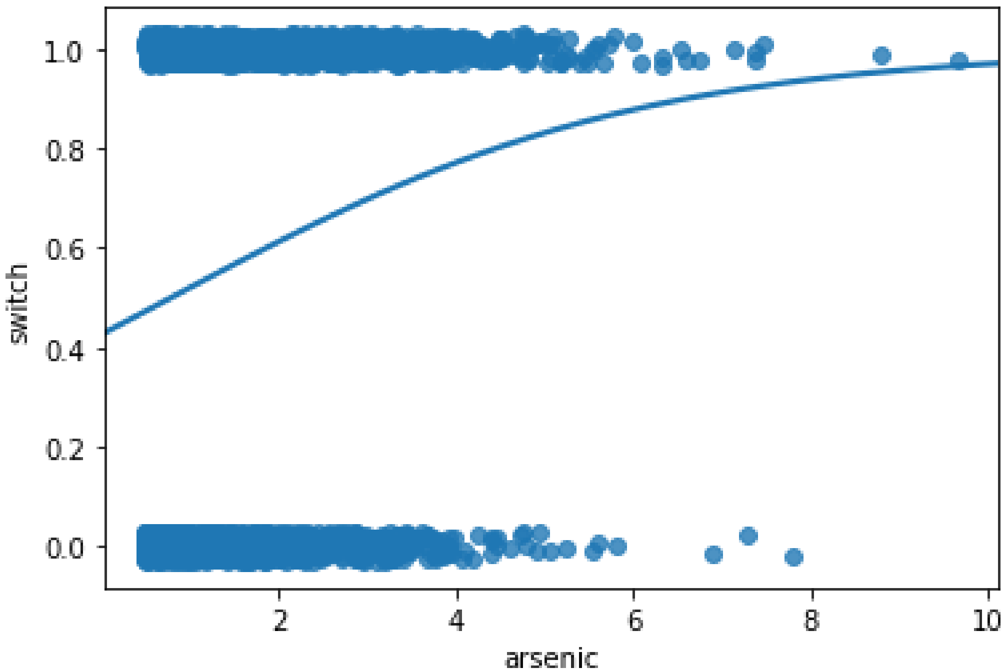
[ ]:
```
# With the Wald statistic at 4.887 we can conclude that the width variable is
↪statistically significant if we apply the rule of thumb cut-off value of 2.
```

[28]:
```
# Extract and print confidence intervals from the fitted model using .
↪conf_int() function.

# Extract and print confidence intervals
print(crab_GLM.conf_int())

# Compute and print confidence intervals for the odds.

# Compute confidence intervals for the odds
print(np.exp(crab_GLM.conf_int()))
```

```
                   0         1
Intercept -17.503010 -7.198625
width       0.297833  0.696629
                      0         1
Intercept  2.503452e-08  0.000748
width      1.346936e+00  2.006975
```

[ ]:
```
# We can conclude that a 1 cm increase in width of a female crab has at least
↪35% increase odds (from lower bound) and at most it doubles the odds (from
↪upper bound) that a satellite crab is present.
```

[32]:
```
# Plot distance and switch and add overlay with the logistic fit
import seaborn as sns
import matplotlib.pyplot as plt
sns.regplot(x = 'arsenic', y = 'switch',
            y_jitter = 0.03,
            data = wells,
            logistic = True,
            ci = None)

# Display the plot
plt.show()
```



[ ]:
```
# Computing predictions
```

[50]:
```
wells_test = wells.iloc[2610:3010,:]
```

[58]:
```
wells_test
```

[58]:

	switch	arsenic	distance	assoc	education	distance100	education4	\
2610	0	1.70	28.686001	0	5	0.28686	1	
2611	0	1.29	70.551003	0	5	0.70551	1	
2612	1	0.56	9.242000	0	0	0.09242	0	
2613	0	0.51	12.541000	0	5	0.12541	1	
2614	0	0.97	30.716000	0	5	0.30716	1	
…	…	…	… …	…	…	…		
3005	0	0.52	19.347000	1	5	0.19347	1	
3006	0	1.08	21.386000	1	3	0.21386	1	
3007	0	0.51	7.708000	0	4	0.07708	1	
3008	0	0.64	22.841999	0	3	0.22842	1	
3009	1	0.66	20.844000	1	5	0.20844	1	

	prediction
2610	0.605958
2611	0.541651
2612	0.634755
2613	0.629931
2614	0.602905

```
…          …
3005    0.619895
3006    0.616868
3007    0.636990
3008    0.614701
3009    0.617674

[400 rows x 8 columns]
```

[52]:

```
# Compute predictions for the test sample wells_test and save as prediction
prediction = wells_fit.predict(exog = wells_test)

# Add prediction to the existing data frame wells_test and assign column name␣
↪prediction
wells_test['prediction'] = prediction

# Examine the first 5 computed predictions
print(wells_test[['switch', 'arsenic', 'prediction']].head())
```

```
      switch  arsenic  prediction
2610       0     1.70    0.605958
2611       0     1.29    0.541651
2612       1     0.56    0.634755
2613       0     0.51    0.629931
2614       0     0.97    0.602905

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

[53]:

```
# Define the cutoff
cutoff = 0.5

# Compute class predictions: y_prediction
y_prediction = np.where(prediction > cutoff, 1, 0)
```

[54]:

```
# Compute class predictions y_pred
y_prediction = np.where(prediction > cutoff, 1, 0)

# Assign actual class labels from the test sample to y_actual
y_actual = wells_test['switch']

# Compute the confusion matrix using crosstab function
```

```
conf_mat = pd.crosstab(y_actual, y_prediction,
                       rownames=['Actual'],
                       colnames=['Predicted'],
                       margins = True)

# Print the confusion matrix
print(conf_mat)
```

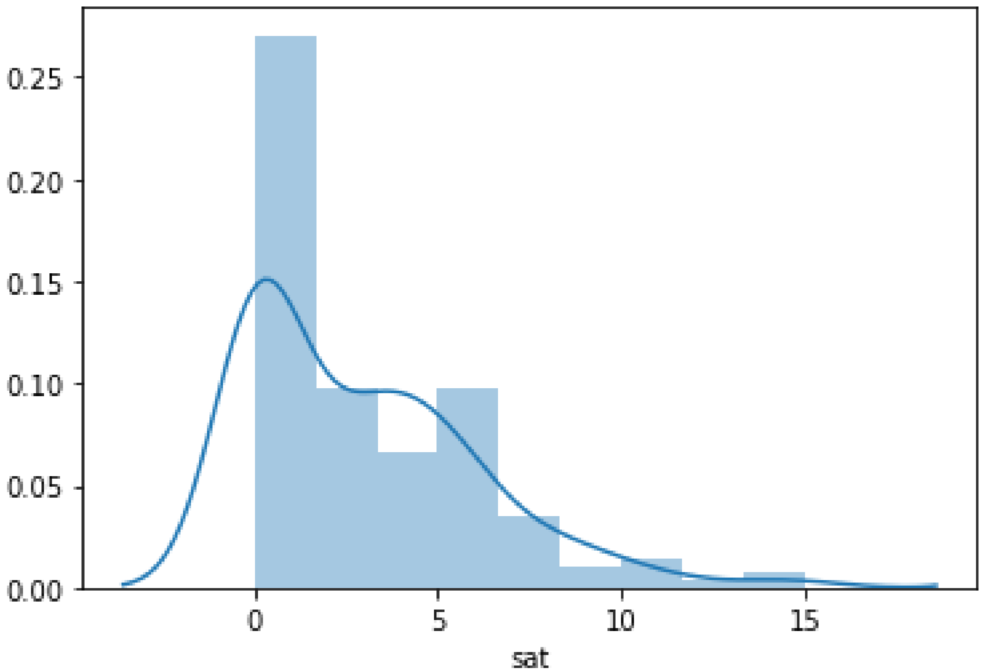
```
Predicted   0    1  All
Actual
0          25  203  228
1           9  163  172
All        34  366  400
```

[ ]:
```
# TP = 163
# TN = 25
# FP = 203
# FN = 9

# This simple model has 203 errors by inccorectly predicting switching of the␣
↪well and 9 errors by incorrectly predicting not switching of the well.
```

[ ]:
```
# So far I have modelled binary data to check the probability of the occurance␣
↪of an event.
# Now onward, I will still try to find probability of the occurance of an␣
↪event, this time occurance won't be a binary value, rather I will count the␣
↪number of occurances in a specified unit of time, distance, area or volume.
# Poisson random variable is used for such measurements.
```

[ ]:
```
# Visualizing the response
```

[61]:
```
# Import libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Plot sat variable
sns.distplot(crab['sat'])

# Display the plot
plt.show()
```




```
[ ]: # Fitting the Poisson model
```

```
[62]: # Import libraries
import statsmodels.api as sm
from statsmodels.formula.api import glm

# Fit Poisson regression of sat by width
model = glm('sat ~ weight', data = crab, family = sm.families.Poisson()).fit()

# Display model results
print(model.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    sat   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:                 Poisson   Df Model:                            1
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -458.08
Date:                Wed, 01 Apr 2020   Deviance:                       560.87
Time:                        22:49:34   Pearson chi2:                     536.
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

```
==============================================================================
Intercept     -0.4284      0.179     -2.394      0.017      -0.779      -0.078
weight         0.5893      0.065      9.064      0.000       0.462       0.717
==============================================================================
```

[64]:
```python
# Compute average crab width
mean_width = np.mean(crab['width'])

# Print the compute mean
print('Average width: ', round(mean_width, 3))

# Extract coefficients
intercept, slope = model.params

# Compute the estimated mean of y (lambda) at the average width
est_lambda = np.exp(intercept) * np.exp(slope * mean_width)

# Print estimated mean of y
print('Estimated mean of y at average width: ', round(est_lambda, 3))
```

```
Average width:  26.299
Estimated mean of y at average width:  2.744
```

[ ]:
```python
# The Poisson regression model states that at the mean value of female crab␣
↪width of 26.3 the expected mean number of satellite crabs present is 2.74.
```

[66]:
```python
model.summary()
```

[66]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    sat   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:                 Poisson   Df Model:                            1
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -461.59
Date:                Thu, 02 Apr 2020   Deviance:                       567.88
Time:                        00:18:52   Pearson chi2:                     544.
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -3.3048      0.542     -6.095      0.000      -4.368      -2.242
width          0.1640      0.020      8.216      0.000       0.125       0.203
==============================================================================
```

"""

[ ]:
```
# The estimate  1 is positive meaning that the effect on the mean of the␣
↪response will be exp( 1) times larger than if  x=0.
```

[67]:
```
# Extract coefficients
intercept, slope = model.params

# Compute and print he multiplicative effect
print(np.exp(slope))
```

```
1.17826743864523
```

[ ]:
```
# To conclude a 1-unit increase in female crab width the number of satellite␣
↪crabs will increase, which will be multiplied by 1.18.
```

[68]:
```
# Compute confidence intervals for the coefficients
model_ci = model.conf_int()

# Compute and print the confidence intervals for the multiplicative effect on␣
↪the mean
print(np.exp(model_ci))
```

```
                  0         1
Intercept  0.012683  0.106248
width      1.133051  1.225289
```

[ ]:
```
# The multiplicative effect on the mean response for a 1-unit increase in width␣
↪is at least 1.13 and at most 1.22.
```

[ ]:
```

```