Linear Regression Models in Python

April 25, 2020

I have created this document by following the course 'Generalized Linear Models in Python' on DataCamp. The course equipped me with the skills to handle data where the response variable is either binary, count, or approximately normal, all under one single framework.

By taking the course my regression tool box is extended with the logistic and Poisson models, by learning how to fit, understand, assess model performance and finally use the model to make predictions on new data. The course helped me significantly to understand and interpret Regression models.

I used data from real world studies such as the largest population poisoning in world's history and nesting of horseshoe crabs.

[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
```

[ ]:
```
Salary ~ Experience suggests that salary is predicted by experience. Salary is␣
↪also considered as response variable and output and experience is input.
```

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]
```

Creating a linear model from the Gaussian family and a logistic regression model from the Binomial family to fit to the dataset.

[7]:
```
# 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]:
```
# 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
==============================================================================
```

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]:
```
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]:
```
# 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
```

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. How to arrive at the specific class, 0 or 1, will be covered next.

[13]:
```
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)
```

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
==============================================================================
"""
```

Extracting parameter estimates

[16]:
```
# Extracting 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
```

Computing odds and probabilities

Here I will review the concept of odds and their relationship to probabilities.

odds=event occurring /event NOT occurring

and odds in terms of probabilities

odds=probability/1−probability

An athlete competes in 60 races and wins 15 times.

Computing and printing the odds of winning.

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

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

```
Odds are:  0.333
```

An athlete competes in 60 races and wins 15 times.

Computing and printing the probability of winning.

[149]:
```
# Computing the probability
probability = 15/60

# Printing the result
print('Probability is: ', round(probability,3))
```

```
Probability is:  0.25
```

Computing and printing the odds using the computed probabilities from previous exercise.

[135]:
```
# Probability calculation
probability = 15/60

# Computing odds using probability calculation
odds_from_probs = probability/(1 - probability)

# Printing the results
print(round(odds_from_probs, 3))
```

```
0.333
```

Predicting whether people switch to another well based on the level of arsenic.

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

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

# Printing 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 and the distance of other 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

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

[24]:
```
# Computing and printing 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]:
```
# Computing 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
```

At the distance100 value of 1.5 the estimated probability is 0.417 with the rate of change in the estimated probability of negative 0.153. 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%.

Applying Regression models on Crab data.

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

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

# Printing 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
==============================================================================
```

There is a positive significant relationship (width increases the chance of a satellite).

Now I will assess the significance of the width variable by computing the Wald statistic. Also note that in the model summary the Wald statistic is presented by the letter z which means that the value of a statistic follows a standard normal distribution. The formula for the Wald statistic: z=ˆ/SE where ˆ is the estimated coefficient and SE its standard error.

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

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

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

# Computing 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
```

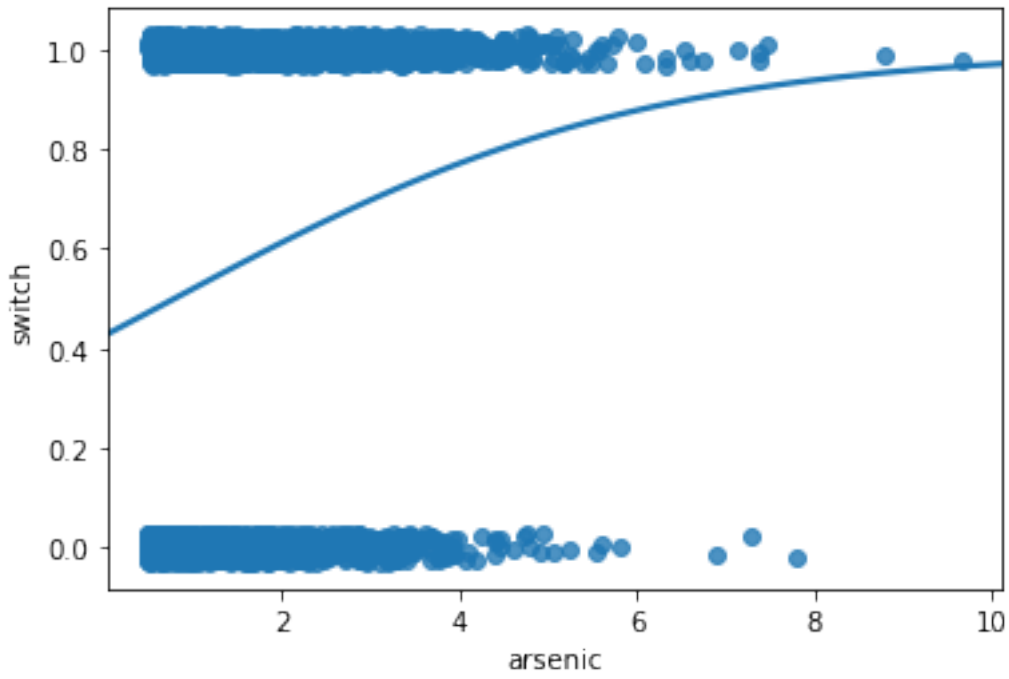
With the Wald statistic at 4.887 we can conclude that the width variable is statistically significant if we apply the rule of thumb of cut-off value of 2.

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

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

# Computing and print confidence intervals for the odds.

# Computing 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]:
```
# Plotting distance and switch and adding 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)

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



I have plotted training data and added a logistic regression fit to it.

Computing predictions

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]:
```
# Computing predictions for the test sample wells_test and saving as prediction
prediction = wells_fit.predict(exog = wells_test)

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

# Examining 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
  """
```

I have computed predictions using the fitted model by computing estimated probabilities. This is a very useful tool as many research questions relate to prediction models.

Computing confusion matrix

The logistic regression model generates two types of predictions, a continuous valued prediction, in the form of a probability, and a class prediction which in the example of the wells dataset is a discrete category with two classes.

Above I computed the continuous values prediction in the form of a probability. Next I will use those values to assign a class to each observation in the wells_test sample. Finally I will describe the model using the confusion matrix.

[ ]:
```
# Defining the cutoff
cutoff = 0.5

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

[54]:
```
# Computing class predictions y_pred
y_prediction = np.where(prediction > cutoff, 1, 0)

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

# Computing the confusion matrix using crosstab function
conf_mat = pd.crosstab(y_actual, y_prediction,
                       rownames=['Actual'],
                       colnames=['Predicted'],
                       margins = True)

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

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

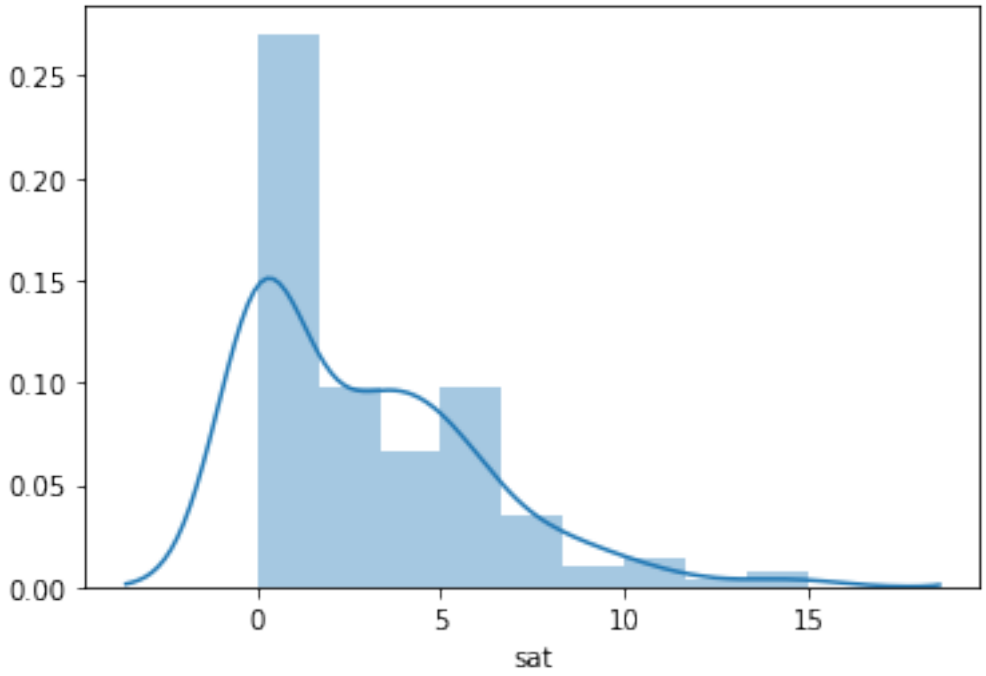
[ ]:
```
TP = 163
TN = 25
FP = 203
FN = 9

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]:
```
# Importing libraries
import seaborn as sns
import matplotlib.pyplot as plt

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

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



Visualizing the response variable there is apparent skewness of the distribution.

[71]:
```
y_variance = crab['sat'].var()
y_mean = crab['sat'].mean()
print(y_variance)
print(y_mean)
```

```
9.912017744320465
2.9190751445086707
```

Fitting the Poisson model

Estimating parameter lambda

The log link function provides for the linear combination in the parameters defining the Poisson regression model of the form

log( )= 0+ 1x1

To obtain the response function in terms of lambda we exponentiated the model function to obtain

=E(y)=exp(0+ 1x1) =E(y)=exp(0)×exp(1x1)

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

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

# Displaying 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
==============================================================================
```

I have now fitted a Poisson regression model. Notice how easy it is to switch from logistic regression using the same glm function. Even the output looks very similar.

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

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

# Extracting coefficients
intercept, slope = model.params

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

# Printing 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
```

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]:
```
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]:
```
# Extracting 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.

Is the mean equal to the variance?

Under the Poisson model one of the assumptions is that the mean should be the same as the variance. If this assumption is violated then there is overdispersion. Without adjusting for overdispersion you would wrongly interpret standard errors of the given model.

[74]:
```
# Computing and printing sample mean of the number of satellites: sat_mean
sat_mean = np.mean(crab.sat)
print('Sample mean:', round(sat_mean, 3))

# Computing and printing sample variance of the number of satellites: sat_var
sat_var = np.var(crab.sat)
print('Sample variance:', round(sat_var, 3))

# Computing ratio of variance to mean
print('Ratio:', round(sat_var/sat_mean, 3))
```

```
Sample mean: 2.919
Sample variance: 9.855
Ratio: 3.376
```

The variance is 3.37 times the mean. This gives an indication that Poisson GLM will not provide the most accurate fit to the data.

Computing expected number of counts

I computed the mean and variance of the crab data and determined they are not equal. I will further practice another analysis for overdispersion by using the already computed mean and calculating the expected number of counts per certain value of counts, for example zero counts. In other words, what count of zero satellites should we expect in the sample given the computed sample mean.

[76]:
```
import math

# Expected number of zero counts
exp_zero_cnt = ((sat_mean**0)*np.exp(-sat_mean))/math.factorial(0)

# Print exp_zero_counts
print('Expected zero counts given mean of ', round(sat_mean,3),
      'is ', round(exp_zero_cnt,3)*100)

# Number of zero counts in sat variable
actual_zero_ant = sum(crab['sat'] == 0)

# Number of observations in crab dataset
num_obs = len(crab)
```

```
# Print the percentage of zero count observations in the sample
print('Actual zero counts in the sample: ', round(actual_zero_ant /␣
 ↪num_obs,3)*100)
```

```
Expected zero counts given mean of  2.919 is  5.4
Actual zero counts in the sample:  35.8
```

The mean parametar should be 5.4% observations with zero count, but in the crab sample there are 35.8% observations with zero count, indicating the presence of overdispersion.

Checking for overdispersion

I will check for overdispersion in the model I fitted previously, namely the horseshoe crab Poisson model where I fitted sat given width. In order to check for potential overdispersion in the fit we would compute the following:

model.pearson_chi2 / model.df_resid

where .pearson_chi2 represents Pearson statistic and .df_resid represents the degrees of freedom of the residuals.

[87]:
```
# Computing and printing the overdispersion approximation
model.pearson_chi2 / model.df_resid
```

[87]:
```
3.182204743877359
```

There is overdispersion present since the ratio is greater than 1, meaning that the coefficient estimates should not be interpreted directly. This problem will be solved next.

Fitting negative binomial

The negative binomial allows for the variance to exceed the mean, which is what has been in the data crab. In the below code I will recall the previous fit of the Poisson regression using the log link function and additionally fit negative binomial model also using the log link function.

I will analyze and see how the statistical measures have changed.

[88]:
```
# Define the formula for the model fit
formula = 'sat ~ width'

# Fit the GLM negative binomial model using log link function
crab_NB = smf.glm(formula = formula, data = crab,
                  family = sm.families.NegativeBinomial()).fit()

# Print Poisson model's summary
print(model.summary())

# Print the negative binomial model's summary
print(crab_NB.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:                Sat, 18 Apr 2020   Deviance:                       567.88
Time:                        23:12:12   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
==============================================================================
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                    sat   No. Observations:                  173
Model:                            GLM   Df Residuals:                      171
Model Family:        NegativeBinomial   Df Model:                            1
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -375.80
Date:                Sat, 18 Apr 2020   Deviance:                       206.41
Time:                        23:12:12   Pearson chi2:                     155.
No. Iterations:                     6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.0323      1.129     -3.572      0.000      -6.245      -1.820
width          0.1913      0.042      4.509      0.000       0.108       0.274
==============================================================================
```

[ ]:
```
Standard error has increased to 0.042, reflecting overdispersion which was not␣
↪captured with the Poisson model.
```

Confidence intervals for negative Binomial model

Continuing with the previously fitted crab model I will compare the confidence intervals for the negative Binomial model with the Poisson regression model.

[89]:
```
# Compute confidence intervals for crab_Pois model
print('Confidence intervals for the Poisson model')
print(model.conf_int())

# Compute confidence intervals for crab_NB model
print('Confidence intervals for the Negative Binomial model')
print(crab_NB.conf_int())
```

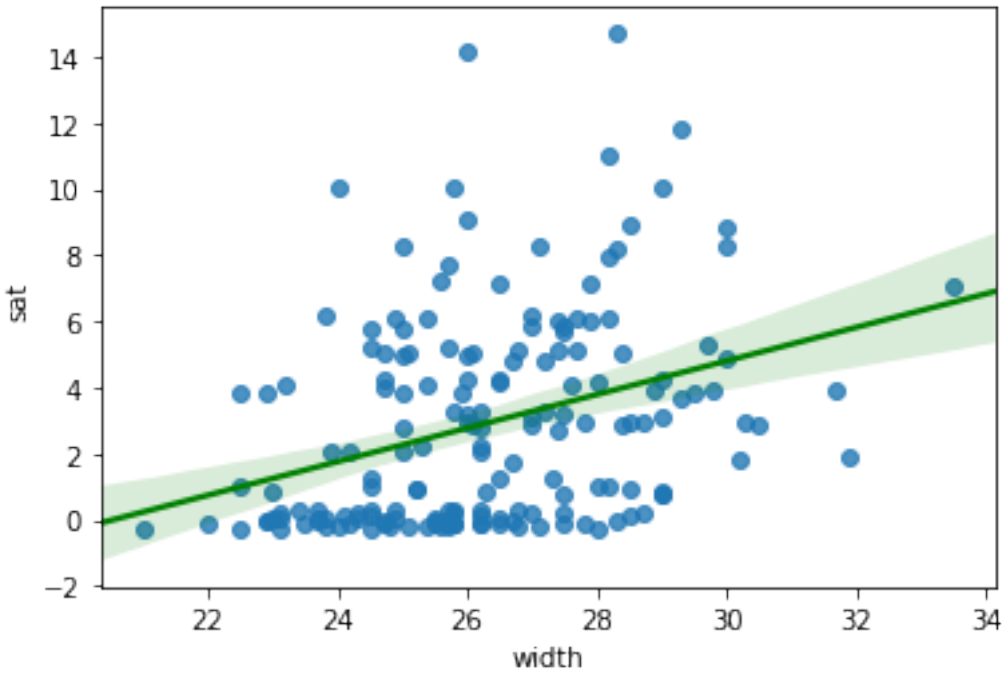
```
Confidence intervals for the Poisson model
                  0         1
Intercept -4.367531 -2.241983
width      0.124914  0.203176
Confidence intervals for the Negative Binomial model
                  0         1
Intercept -6.244509 -1.820000
width      0.108155  0.274472
```

The confidence intervals are wider for the negative Binomial model compared to quite narrow confidence intervals for the Poisson model since it did not account for overdispersion.

Plotting data and linear model fit

Below I have visually analyzed the crab data and then the model fit.

[90]:
```
# Importing libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Plotting the data points and linear model fit
sns.regplot('width', 'sat', data = crab,
            y_jitter = 0.3,
            fit_reg = True,
            line_kws = {'color':'green',
                        'label':'LM fit'})

# Print plot
plt.show()
```



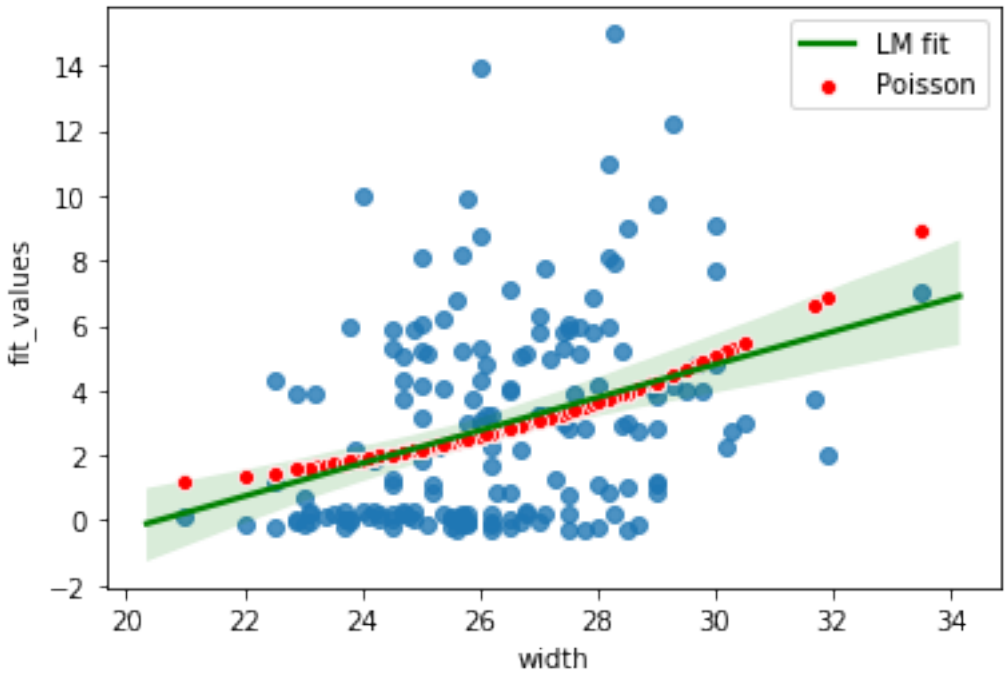
Let's see how this fit compares to the Poisson regression model.

Plotting fitted values

Using the previously fitted Poisson regression model of horseshoe crab data with sat as the response and width as the explanatory variable I will visually assess the model fit and compare to the previously visualized linear model.

[91]:
```
# Adding fitted values to the fit_values column of crab dataframe
crab['fit_values'] = model.fittedvalues
```

[92]:
```
# Plotting data points
sns.regplot('width', 'sat', data = crab,
            y_jitter = 0.3,
            fit_reg = True,
            line_kws = {'color':'green',
                        'label':'LM fit'})

# Poisson regression fitted values
sns.scatterplot('width','fit_values', data = crab,
           color = 'red', label = 'Poisson')

# Printing plot
plt.show()
```



We can compare both fits on one graph! Similary as for the model with weight variable the linear and Poisson fits are close in the mid range of width values, but diverge on smaller and larger values.

Fitting a multivariable logistic regression

The crab dataset is revisited to fit a multivariate logistic regression model. Previously I fitted a logistic regression with width as explanatory variable. Now I will analyze the effects of adding color as additional variable.

The color variable has a natural ordering from medium light, medium, medium dark and dark. As such color is an ordinal variable which here I will treat as a quantitative variable.

The only difference in the code from the univariate case is in the formula argument, where now I will add structure to incorporate the new variable.

[96]:
```
# Importing statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import glm

# Defining model formula
formula = 'y ~ width + color'

# Fitting GLM
model = glm(formula, data = crab, family = sm.families.Binomial()).fit()

# Printing model summary
print(model.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      170
Model Family:                Binomial   Df Model:                            2
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -94.561
Date:                Sun, 19 Apr 2020   Deviance:                       189.12
Time:                        18:07:54   Pearson chi2:                     170.
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -10.0708      2.807     -3.588      0.000     -15.572      -4.569
width          0.4583      0.104      4.406      0.000       0.254       0.662
color         -0.5090      0.224     -2.276      0.023      -0.947      -0.071
==============================================================================
```

[102]:
```
np.exp(-0.5090)
```

[102]: 0.6010963747389753

From model summary it can be seen that for each one-level increase in color of the female crab, the estimated odds multiply by exp(-0.509)=0.6, i.e. the odds for dark crabs are 60% than those for medium crabs.

The effect of multicollinearity

Using the crab dataset I will analyze the effects of multicollinearity. Multicollinearity can have the following effects:

Coefficient is not significant, but variable is highly correlated with y. Adding/removing a variable significantly changes coefficients. Not logical sign of the coefficient. Variables have high pairwise correlation.

[104]:
```
# Importing statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import glm

# Defining model formula
formula = 'y ~ weight + width'

# Fitting GLM
model = glm(formula, data = crab, family = sm.families.Binomial()).fit()

# Printing model summary
print(model.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      170
Model Family:                Binomial   Df Model:                            2
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -96.446
Date:                Sun, 19 Apr 2020   Deviance:                       192.89
Time:                        20:07:06   Pearson chi2:                     167.
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -9.3547      3.528     -2.652      0.008     -16.270      -2.440
weight         0.8338      0.672      1.241      0.214      -0.483       2.150
width          0.3068      0.182      1.686      0.092      -0.050       0.663
==============================================================================
```

It can be noticed that neither weight nor width are statistically significant. When I fitted univariate logistic regressions for each variable, both variables where statistically significant. There is evident presence of multicollinearity! Let's measure it in the next exercise.

Computing VIF

One of the most widely used diagnostic for multicollinearity is the variance inflation factor or VIF, which is computed for each explanatory variable.

The rule of thumb threshold is VIF at the level of 2.5, meaning if the VIF is above 2.5 one should consider there is effect of multicollinearity on the fitted model.

[105]:
```
# Import functions
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Get variables for which to compute VIF and add intercept term
X = crab[['weight', 'width', 'color']]
X['Intercept'] = 1

# Compute and view VIF
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

# View results using print
print(vif)
```

```
   variables         VIF
0     weight    4.691018
1      width    4.726378
2      color    1.076594
3  Intercept  414.163343
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6:
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
```

With VIF well above 2.5 for weight and width means that there is multicollinearity present in the model and we can not use both variables in the model.

Checking the model fit

There can be an improvement in the model fit by adding additional variable on the wells data. Continuing with this data set I will see how further increase in model complexity effects deviance and model fit.

[109]:
```
# Importing statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import glm
```

```
# Defining model formula
formula = 'switch ~ distance100 + arsenic'

# Fitting GLM
model_dist_ars = glm(formula, data = wells, family = sm.families.Binomial()).
 ↪fit()

# Comparing deviance of null and residual model
diff_deviance = model_dist_ars.null_deviance - model_dist_ars.deviance

# Printing the computed difference in deviance
print(diff_deviance)
```

```
188.76305963384857
```

[138]:
```
model_dist_ars.summary()
```

[138]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                 switch   No. Observations:                 3010
Model:                            GLM   Df Residuals:                     3007
Model Family:                Binomial   Df Model:                            2
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -1957.6
Date:                Tue, 21 Apr 2020   Deviance:                       3915.2
Time:                        19:49:42   Pearson chi2:                 3.05e+03
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0055      0.080      0.070      0.945      -0.151       0.162
distance100   -0.9059      0.105     -8.662      0.000      -1.111      -0.701
arsenic        0.4627      0.042     11.146      0.000       0.381       0.544
==============================================================================
"""
```

Having both distance100 and arsenic in the model reduces deviance by 187 compared to the intercept only model. But what is the actual impact of additional variable arsenic lets find out.

Comparing two models

I have fitted a model with distance100 and arsenic as explanatory variables. Below I will analyze the impact on the model fit for each of the added variables.

[112]:
```
# Compute the difference in adding distance100 variable
diff_deviance_distance = wells_fit.null_deviance - wells_fit.deviance

# Print the computed difference in deviance
print('Adding distance100 to the null model reduces deviance by: ',
      round(diff_deviance_distance,3))

# Compute the difference in adding arsenic variable
diff_deviance_arsenic = wells_fit.deviance - model_dist_ars.deviance

# Print the computed difference in deviance
print('Adding arsenic to the distance model reduced deviance further by: ',
      round(diff_deviance_arsenic,3))
```

```
Adding distance100 to the null model reduces deviance by:  42.726
Adding arsenic to the distance model reduced deviance further by:  146.037
```

Adding distance100 to the null model reduces deviance by 41.9 and with an addition of arsenic the deviance further reduces by 145. Having such large reduction than expected reduction by 1 we can conclude that the multivariate model has improved the model fit.

Deviance and linear transformation

In the above codes the deviance decreased as I added a variable that improves the model fit. Now I will consider the well switch data example and the model I fitted with distance variable, but I will assess what happens when there is a linear transformation of the variable.

The variable distance100 is the original variable distance divided by 100 to make for more meaningful representation and interpretation of the results.

[113]:
```
# Importing functions
import statsmodels.api as sm
from statsmodels.formula.api import glm

# Fitting logistic regression model saved as model_dist_1
model_dist_1 = glm('switch ~ distance', data = wells, family = sm.families.
 ↪Binomial()).fit()

# Checking the difference in deviance of model_dist_1 and model_dist
print('Difference in deviance is: ', round(model_dist_1.deviance - wells_fit.
 ↪deviance,3))
```

```
Difference in deviance is:  0.0
```

The linear transformations do not change the model error and hence the deviance remains the same. The reason being since linear transformation does not add new data information to the model.

Model matrix for continuous variables

Dmatrix() is used to obtain the model matrix. The input to dmatrix() is the right hand side of the glm() formula argument. In case the variables are part of the dataframe, then we should also

specify the data source via the data argument.

dmatrix('y ~ x1 + x2', data = my_data)

In the below codes I will analyze and confirm the structure of the model before model fit.

[ ]:
```
# Importing dmatrix from patsy, using it to construct the model_matrix with␣
↪'arsenic', and printing the first 5 rows.

# Importing function dmatrix()
from patsy import dmatrix

# Constructing model matrix with arsenic
model_matrix = dmatrix('arsenic', data = wells, return_type = 'dataframe')
print(model_matrix.head())
```

[115]:
```
# Importing dmatrix from patsy, constructing the model_matrix with 'arsenic'␣
↪and 'distance100', and printing the first 5 rows.

# Importing function dmatrix()
from patsy import dmatrix

# Constructing model matrix with arsenic and distance100
model_matrix = dmatrix('arsenic + distance100', data = wells, return_type =␣
↪'dataframe')
print(model_matrix.head())
```

```
   Intercept  arsenic  distance100
0        1.0     2.36      0.16826
1        1.0     0.71      0.47322
2        1.0     2.07      0.20967
3        1.0     1.15      0.21486
4        1.0     1.10      0.40874
```

Dmatrix() silently includes an intercept for each model matrix without we specifying it. Analyzing the output from dmatrix() one can be sure that the inputs are correctly structured.

Variable transformation

Continuing with the wells I will practice applying variable transformation directly in the formula and model matrix setting without the need to add the transformed data to the data frame first. I will also revisit the computation of model error or deviance to see if the transformation improved the model fit. The structure of dmatrix() function is the right hand side of the glm() formula argument in addition to the data argument.

[116]:
```
# Import function dmatrix
import numpy as np
from patsy import dmatrix

# Construct model matrix for arsenic with log transformation
```

```
dmatrix('np.log(arsenic)', data = wells,
       return_type = 'dataframe').head()
```

[116]:

	Intercept	np.log(arsenic)
0	1.0	0.858662
1	1.0	-0.342490
2	1.0	0.727549
3	1.0	0.139762
4	1.0	0.095310

[117]:
```
# Importing statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import glm
import numpy as np

# Defining model formula
formula = 'switch ~ np.log(arsenic)'

# Fitting GLM
model_log_ars = glm(formula, data = wells,
                    family = sm.families.Binomial()).fit()

# Printing model summary
print(model_log_ars.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:                -1987.6
Date:                Mon, 20 Apr 2020   Deviance:                       3975.3
Time:                        22:34:54   Pearson chi2:                 3.01e+03
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
===
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---
Intercept           0.0966      0.041      2.331      0.020       0.015
0.178
np.log(arsenic)     0.7089      0.064     11.046      0.000       0.583
0.835
==============================================================================
```

===

[120]:
```
# Checking the difference in deviance of model_dist_1 and model_dist
print('Difference in deviance is: ', round(model_GLM.deviance - model_log_ars.
 ↪deviance,3))
```

```
Difference in deviance is:  86.017
```

The model with log(arsenic) improves model fit with a reduction in deviance of 86.017.

Yes, comparing the deviance of the model with log(arsenic) and deviance of the model with arsenic there is a reduction in deviance of 86.017, which is larger than expected 1 and hence it does improve the model fit.

Coding categorical variables

Above I created a model matrices for continuous variables and applied variable transformation. Now I will practice the ways of coding a categorical variable.

Categorical data provides a way to analyze and compare relationships given different groups or factors. Hence, choosing a reference group is important and often, depending on the study at hand, one might want to change the reference group, from the default one. One frequently used reason for changing the reference group is that the interpretation of coefficient estimates is more applicable and interesting given the study.

Crab dataset will be revisited where color and spine are categorical variables.

[121]:
```
# Importing function dmatrix
from patsy import dmatrix

# Constructing and print model matrix for color as categorical variable
print(dmatrix('C(color)', data = crab,
              return_type = 'dataframe').head())
```

```
   Intercept  C(color)[T.2]  C(color)[T.3]  C(color)[T.4]
0        1.0            1.0            0.0            0.0
1        1.0            0.0            1.0            0.0
2        1.0            0.0            0.0            0.0
3        1.0            0.0            1.0            0.0
4        1.0            0.0            1.0            0.0
```

[122]:
```
# Construct and print the model matrix for color with reference group 3
print(dmatrix('C(color, Treatment(3))',
        data = crab,
        return_type = 'dataframe').head())
```

```
   Intercept  C(color, Treatment(3))[T.1]  C(color, Treatment(3))[T.2]  \
0        1.0                          0.0                          1.0
1        1.0                          0.0                          0.0
2        1.0                          1.0                          0.0
3        1.0                          0.0                          0.0
```

```
4        1.0                          0.0                          0.0

   C(color, Treatment(3))[T.4]
0                          0.0
1                          0.0
2                          0.0
3                          0.0
4                          0.0
```

[ ]:
```
Notice the change in columns where now the medium dark category is the␣
↪reference group, where its mean behavior is represented by the intercept.
```

Modeling with categorical variable

Previously I have fitted a logistic regression model with color as explanatory variable along with width where the color was treated as quantitative variable. Now I will treat color as a categorical variable which when the model is constructed matrix will encode the color into 3 variables with 0/1 encoding.

The default encoding in dmatrix() uses the first group as a reference group. To view model matrix as a dataframe an additional argument in dmatrix(), namely, return_type will be set to 'dataframe'.

The color variable has a natural ordering as follows: 1. medium light 2. medium 3. medium dark 4. dark

[142]:
```
# Constructing model matrix
model_matrix = dmatrix('C(color, Treatment(4))' , data = crab,
                       return_type = 'dataframe')

# Printing first 5 rows of model matrix
print(model_matrix.head())

# Fitting and print the results of a glm model with the above model matrix␣
↪configuration
model = glm('y ~ C(color, Treatment(4))', data = crab,
            family = sm.families.Binomial()).fit()

print(model.summary())
```

```
   Intercept  C(color, Treatment(4))[T.1]  C(color, Treatment(4))[T.2]  \
0        1.0                          0.0                          1.0
1        1.0                          0.0                          0.0
2        1.0                          1.0                          0.0
3        1.0                          0.0                          0.0
4        1.0                          0.0                          0.0

   C(color, Treatment(4))[T.3]
0                          0.0
1                          1.0
2                          0.0
```

```
3                          1.0
4                          1.0
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      169
Model Family:                Binomial   Df Model:                            3
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -106.03
Date:                Thu, 23 Apr 2020   Deviance:                       212.06
Time:                        21:32:53   Pearson chi2:                     173.
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================
                                 coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
Intercept                     -0.7621      0.458     -1.665      0.096
-1.659       0.135
C(color, Treatment(4))[T.1]    1.8608      0.809      2.301      0.021
0.276       3.446
C(color, Treatment(4))[T.2]    1.7382      0.512      3.393      0.001
0.734       2.742
C(color, Treatment(4))[T.3]    1.1299      0.551      2.051      0.040
0.050       2.210
==============================================================================
```

The estimated odds that a crab with color_2 (medium) has a satellite nearby are 5.687 times the estimated odds that a crab with color_4 (dark) has a satellite present.

[ ]:
```
logit=−0.7621 + 1.8608 × color1 + 1.7382 × color2 + 1.1299 × color3
```

[147]:
```
np.exp(1.7382)
```

[147]:
```
5.6870974286722
```

To the previous model matrix and logistic regression model adding width as additional explanatory variable. Viewing model matrix before fitting the model and then viewing model results.

[133]:
```
# Construct model matrix
model_matrix = dmatrix('C(color, Treatment(4)) + width' , data = crab,
                       return_type = 'dataframe')

# Print first 5 rows of model matrix
print(model_matrix.head())
```

```
# Fit and print the results of a glm model with the above model matrix␣
 ↪configuration
model = glm('y ~ C(color, Treatment(4)) + width', data = crab,
            family = sm.families.Binomial()).fit()

print(model.summary())
```

```
   Intercept  C(color, Treatment(4))[T.1]  C(color, Treatment(4))[T.2]  \
0        1.0                          0.0                          1.0
1        1.0                          0.0                          0.0
2        1.0                          1.0                          0.0
3        1.0                          0.0                          0.0
4        1.0                          0.0                          0.0

   C(color, Treatment(4))[T.3]  width
0                          0.0   28.3
1                          1.0   22.5
2                          0.0   26.0
3                          1.0   24.8
4                          1.0   26.0
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  173
Model:                            GLM   Df Residuals:                      168
Model Family:                Binomial   Df Model:                            4
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -93.729
Date:                Tue, 21 Apr 2020   Deviance:                       187.46
Time:                        14:56:23   Pearson chi2:                     169.
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
===============
                                 coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
---------------
Intercept                    -12.7151      2.762     -4.604      0.000
-18.128      -7.302
C(color, Treatment(4))[T.1]     1.3299      0.853      1.560      0.119
-0.341       3.001
C(color, Treatment(4))[T.2]     1.4023      0.548      2.557      0.011
0.327       2.477
C(color, Treatment(4))[T.3]     1.1061      0.592      1.868      0.062
-0.054       2.267
width                           0.4680      0.106      4.434      0.000
```

```
0.261       0.675
==============================================================================
===============
```

[136]:
```
# Computing the multiplicative effect on the odds
print('Odds: \n', np.exp(model.params))
```

```
Odds:
 Intercept                      0.000003
C(color, Treatment(4))[T.1]    3.780738
C(color, Treatment(4))[T.2]    4.064684
C(color, Treatment(4))[T.3]    3.022612
width                          1.596727
dtype: float64
```

A one-unit increase in width has multiplicative effect of 1.5967 on the odds that the satellite is nearby for all color groups.

Interaction terms

Now I will analyze the effects of interaction between two continuous variables.

I will use centered variables instead of original values to be able to interpret the coefficient effects more easily, i.e. from the level of the mean values rather than 0 which may not be logical for the study at hand. In other words we don't want to interpret the model by assuming 0 for arsenic or distance100 variables.

The model 'switch ~ distance100 + arsenic' is already there as model_dist_ars in the workspace.

[137]:
```
# Importing libraries
import statsmodels.api as sm
from statsmodels.formula.api import glm

# Fitting GLM and print model summary
model_int = glm('switch ~ center(distance100) + center(arsenic) +␣
 ↪center(distance100):center(arsenic)',
                data = wells, family = sm.families.Binomial()).fit()

# Viewing model results
print(model_int.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                 switch   No. Observations:                 3010
Model:                            GLM   Df Residuals:                     3006
Model Family:                Binomial   Df Model:                            3
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -1956.2
Date:                Tue, 21 Apr 2020   Deviance:                       3912.4
Time:                        19:36:19   Pearson chi2:                 3.08e+03
```

```
No. Iterations:                     4
Covariance Type:            nonrobust
==============================================================================================
                                        coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Intercept                              0.3521      0.040      8.813      0.000       0.274       0.430
center(distance100)                   -0.8834      0.105     -8.410      0.000      -1.089      -0.678
center(arsenic)                        0.4713      0.042     11.168      0.000       0.389       0.554
center(distance100):center(arsenic)   -0.1744      0.103     -1.700      0.089      -0.376       0.027
==============================================================================================
```

The value of interaction term of -0.18 is added to the coefficient value of distance100 of -0.8834 thereby increasing the importance of the coefficient for distance100.

The value of interaction term of -0.18 is added to the coefficient value of arsenic of 0.4713 thereby decreasing the importance of the coefficient for arsenic.

At average value of distance100 and arsenic the probability of switching from the current well is equal to 0.59.

[139]:
```
np.exp(0.35)/(1+np.exp(0.35))
```

[139]:
```
0.58661757891733
```

The interaction term increases the importance of distance100 as explanatory variable given one unit increase in arsenic levels.

The interaction term decreases the importance of arsenic as explanatory variable given one unit increase in distance100 values.

[ ]:
```

```