Time Series Analysis, Regression and Forecasting

With tutorials in Python

The Poisson Regression Model

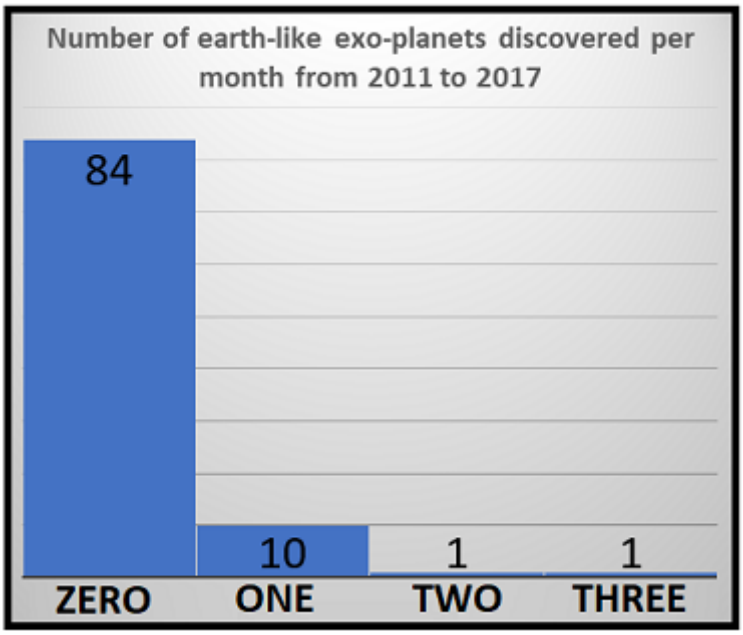
And a tutorial on Poisson regression using Python

In this section, we'll cover the following topics:

1. **Features of count based data:** Count data sets are some of the most commonly occurring data in the world. We'll look at what makes count based data different.
2. **Regression models for forecasting counts:** We'll look at **Poisson regression model** in detail. The **Negative Binomial (NB) regression model** is another commonly used model for count based data. I'll cover that in a later section.
3. **Python tutorial on Poisson regression:** I will take you through a step-by-step tutorial on how to create a Poisson regression model in Python using the **GLM class** of **statsmodels**, and how to train it on a real world data set.

What is count based data?

Count based data contains events that occur at a certain rate. The rate of occurrence may change over time or from one observation to next. Here are some examples of count based data:



Data source: Wikipedia: potentially habitable exoplanets (https://en.wikipedia.org/wiki/List_of_potentially_habitable_exoplanets) (Image by Author (https://www.linkedin.com/in/sachindate/))

- Number of vehicles crossing an intersection per hour,
- Number of people visiting a doctor's office per month,
- Number of earth-like planets spotted per month.

A data set of counts has the following characteristics:

- **Whole number data:** The data consists of non-negative integers: [0… ∞] Regression techniques such as Ordinary Least Squares Regression may not be appropriate for modeling such data as OLSR works best on real numbers such as -656.0, -0.00000345, 13786.1 etc.
- **Skewed Distribution:** The data may contain a large number of data points for just a few values, thereby making the frequency distribution quite skewed. See for example above histogram.
- **Sparsity:** The data may reflect the occurrence of a rare event such as a gamma ray burst, thereby making the data sparse.
- **Rate of occurrence:** For the sake of creating a model, it can be assumed that there is a certain rate of occurrence of events λ that drives the generation of such data. The event rate may drift over time.

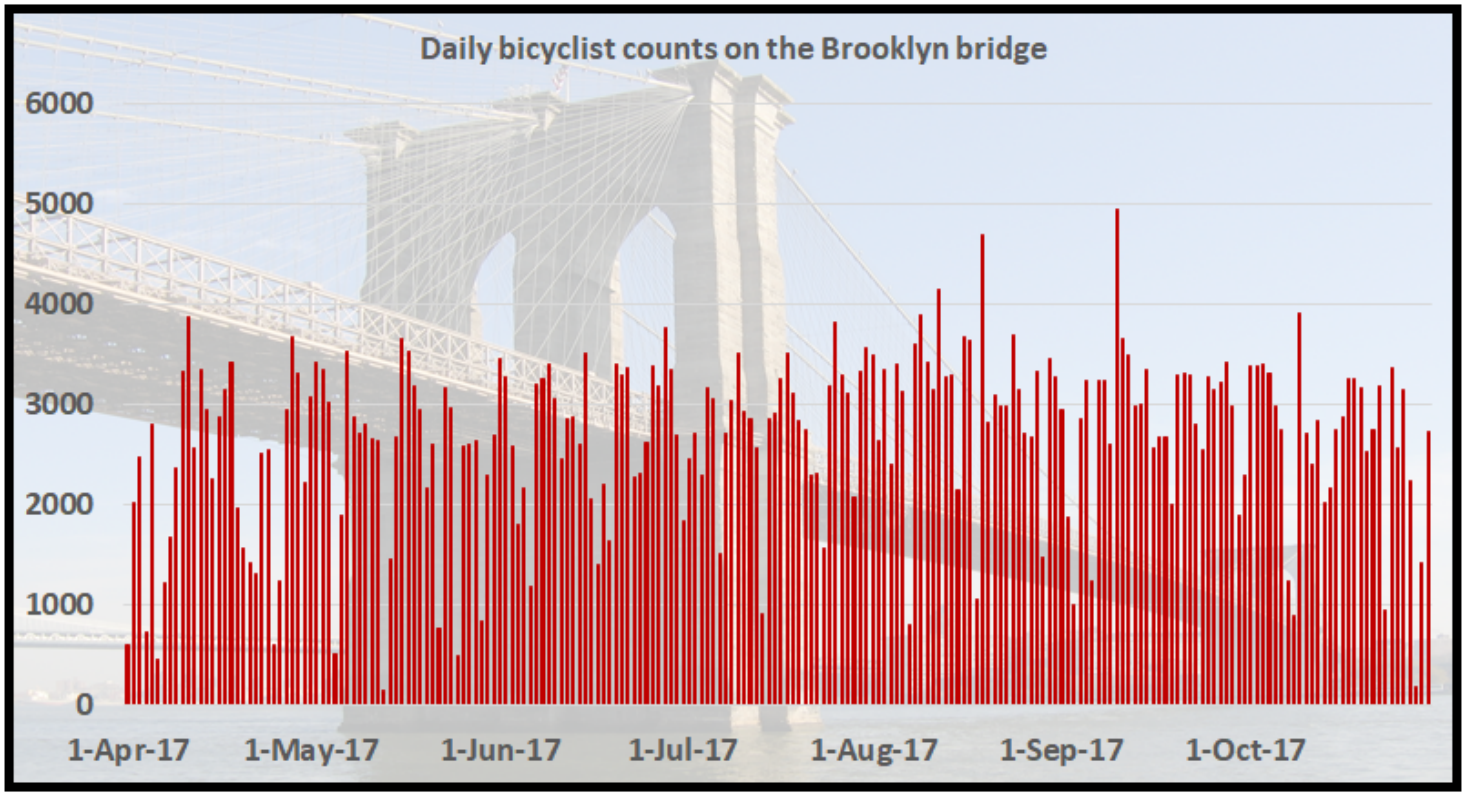
A real world data set of counts

The following table contains counts of bicyclists traveling over various NYC bridges. The counts were measured daily from 01 April 2017 to 31 October 2017.

Date	Day	High Temp (°F)	Low Temp (°F)	Precipitation	Brooklyn Bridge	Manhattan Bridge	Williamsburg Bridge	Queensboro Bridge	Total
6/1	Thursday	78.1	62.1	0.00	3,468	7,328	8,461	6,184	25,441
6/2	Friday	73.9	60.1	0.01	3,271	7,007	7,968	5,293	23,539
6/3	Saturday	72.0	55	0.01	2,589	4,510	6,210	4,084	17,393
6/4	Sunday	68.0	60.1	0.09	1,805	3,127	4,023	3,023	11,978
6/5	Monday	66.9	60.1	0.02	2,171	4,552	5,276	3,359	15,358
6/6	Tuesday	55.9	53.1	0.06	1,193	3,021	3,807	2,572	10,593
6/7	Wednesday	66.9	54	0.00	3,211	7,180	7,632	5,072	23,095
6/8	Thursday	68.0	59	0.00	3,253	7,083	7,778	5,288	23,402
6/9	Friday	80.1	59	0.00	3,401	6,859	7,744	5,155	23,159
6/10	Saturday	84.0	68	0.00	3,066	5,193	6,391	4,425	19,075
6/11	Sunday	90.0	73	0.00	2,465	4,388	5,153	3,178	15,184
6/12	Monday	91.9	77	0.00	2,854	6,265	7,049	5,032	21,200
10/26	Thursday	57.0	53.1	0.00	2,565	5,217	5,958	5,011	18,751
10/27	Friday	62.1	48.0	0.00	3,150	5,610	6,450	5,181	20,391
10/28	Saturday	68.0	55.9	0.00	2,245	4,520	5,104	4,069	15,938
10/29	Sunday	64.9	61.0	3.03	183	661	1,026	965	2,835
10/30	Monday	55.0	46.0	0.25	1,428	2,966	3,547	2,924	10,865
10/31	Tuesday	54.0	44.0	0.00	2,727	5,597	5,894	4,883	19,101

Source: Bicycle Counts for East River Bridges (https://data.cityofnewyork.us/Transportation/Bicycle-Counts-for-East-River-Bridges/gua4-p9wg) (Data source: NYC OpenData) (Image by Author (https://www.linkedin.com/in/sachindate/))

Here is a time sequenced plot of the bicyclist counts on the Brooklyn bridge:



Background image: The Brooklyn bridge as seen from Manhattan island (https://en.wikipedia.org/wiki/Brooklyn_Bridge#/media/File:Brooklyn_Bridge_Postdlf.jpg)

Regression models for counts

The **Poisson regression model** and the **Negative Binomial regression model** are two popular techniques for developing regression models for counts. Other possibilities are Ordered Logit (https://en.wikipedia.org/wiki/Ordered_logit), Ordered Probit (https://en.wikipedia.org/wiki/Ordered_probit) and Nonlinear Least Squares (https://timeseriesreasoning.com/contents/nonlinear-least-squares-nls-regression/) models.

Regression strategy

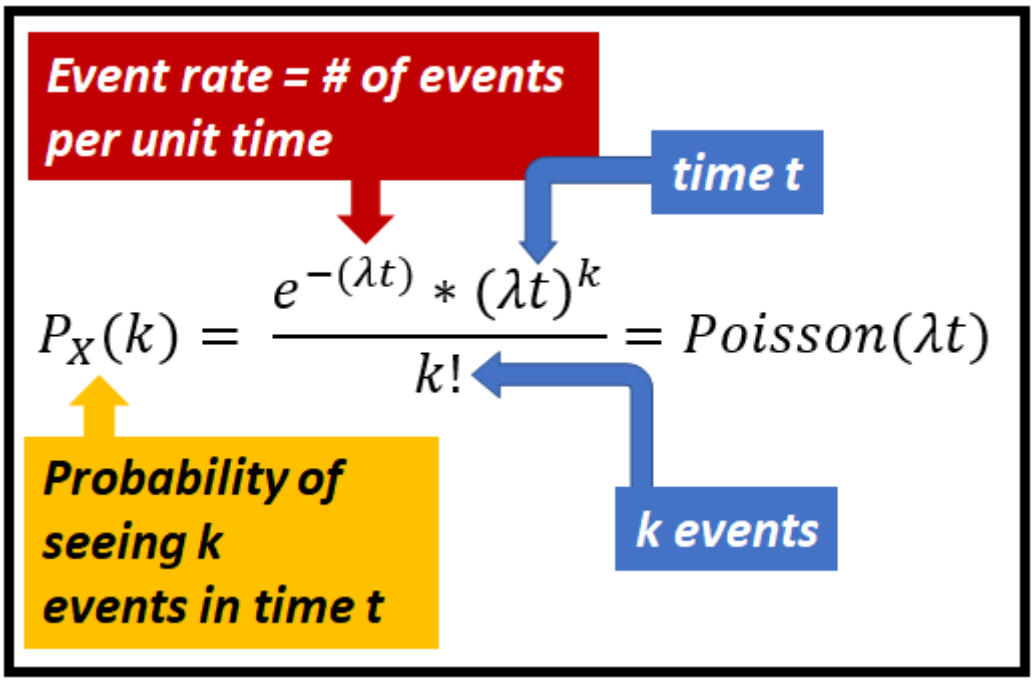
It's good practice to start with the Poisson regression model and use it as the "control" for either more complex, or less constrained models. In their book Regression Analysis of Count Data (http://faculty.econ.ucdavis.edu/faculty/cameron/racd2/), Cameron and Trivedi say the following:

> "A sound practice is to estimate both Poisson and negative binomial models."

In this section, we'll use the Poisson regression model for regressing the bicyclist counts observed on the Brooklyn bridge, and in a following section, we'll train the Negative Binomial model (https://timeseriesreasoning.com/contents/negative-binomial-regression-model/) on the same data set.

Introducing the Poisson model

The Poisson distribution has the following **P**robability **M**ass **F**unction.



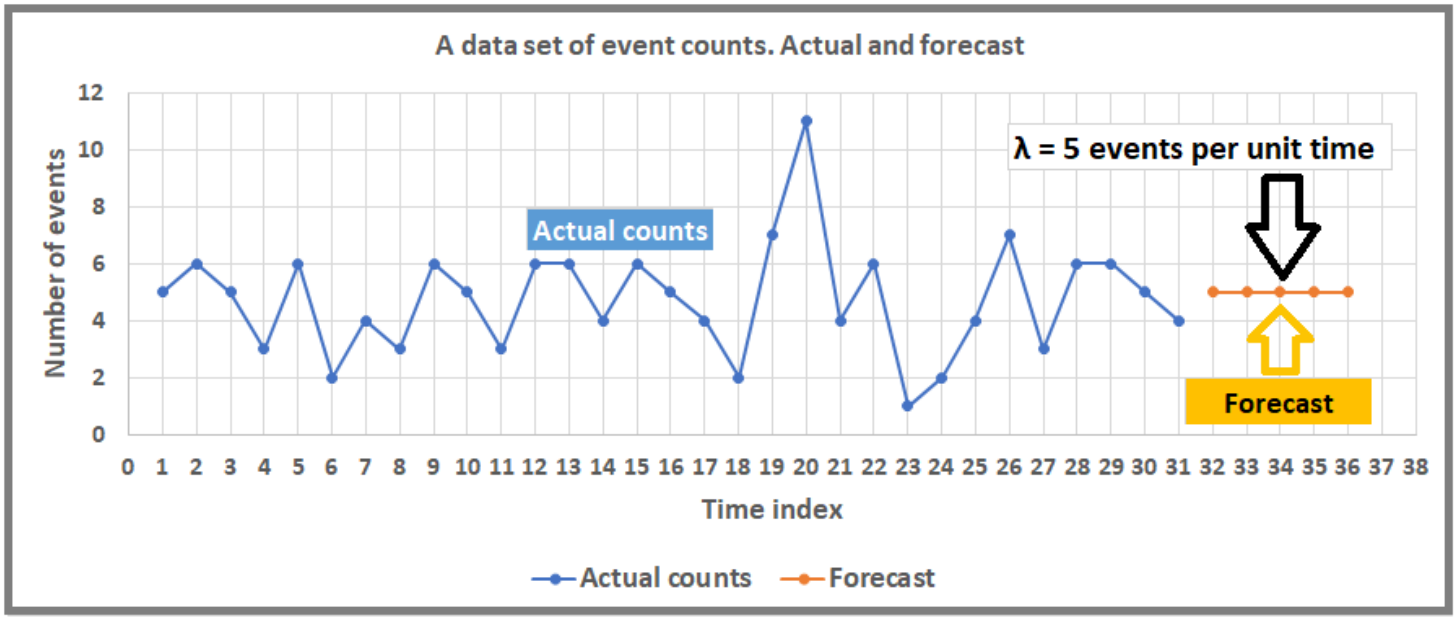
Probability of seeing ***k*** events in time **t**, given ***λ*** events occurring per unit time (Image by Author (https://www.linkedin.com/in/sachindate/))

The expected value (mean) for a Poisson distribution is ***λ***. Thus in the absence of other information, one should expect to see ***λ*** events in any unit time interval such as 1 hour, 1 day, etc. For any interval ***t***, one would expect to see ***λt*** events.

A Poisson regression model *for* ***constant λ***

If the event rate λ is constant, one can simply use a modified Mean Model for predicting future counts of events. In this case, one would set all predicted values of counts to this constant value λ.

The following figure illustrates the constant λ scenario:



Actual and predicted counts for a constant rate model (Image by Author (https://www.linkedin.com/in/sachindate/))

The following Python code was used to generate the blue dots (actual counts in the past time steps) using a Poisson process with λ=5. The orange dots (predictions) are all set to the same value **5**:

```
1  import random
2  import math
3
4  _lambda = 5
5  _num_total_arrivals = 150
6  _num_arrivals = 0
7  _arrival_time = 0
8  _num_arrivals_in_unit_time = []
9  _time_tick = 1
10
11 print('RANDOM_N,INTER_ARRIVAL_TIME,EVENT_ARRIVAL_TIME')
12
13 for i in range(_num_total_arrivals):
14       #Get the next probability value from Uniform(0,1)
15       p = random.random()
16
```

```
17      #Plug it into the inverse of the CDF of Exponential(_lamnbda)
18      _inter_arrival_time = –math.log(1.0 – p)/_lambda
19
20      #Add the inter-arrival time to the running sum
21      _arrival_time = _arrival_time + _inter_arrival_time
22
23      #Increment the number of arrival per unit time
24      _num_arrivals = _num_arrivals + 1
25      if _arrival_time > _time_tick:
26          _num_arrivals_in_unit_time.append(_num_arrivals)
27          _num_arrivals = 0
28          _time_tick = _time_tick + 1
29
30      #print it all out
31      print(str(p)+','+str(_inter_arrival_time)+','+str(_arrival_time))
32
33  print('\nNumber of arrivals in successive unit length intervals ===>')
34  print(_num_arrivals_in_unit_time)
35
36  print('Mean arrival rate for sample:' + str(sum(_num_arrivals_in_unit_time)/len(_num_arrivals_ir
```

view raw poisson_counts_generator.py hosted with ❤ by **GitHub**

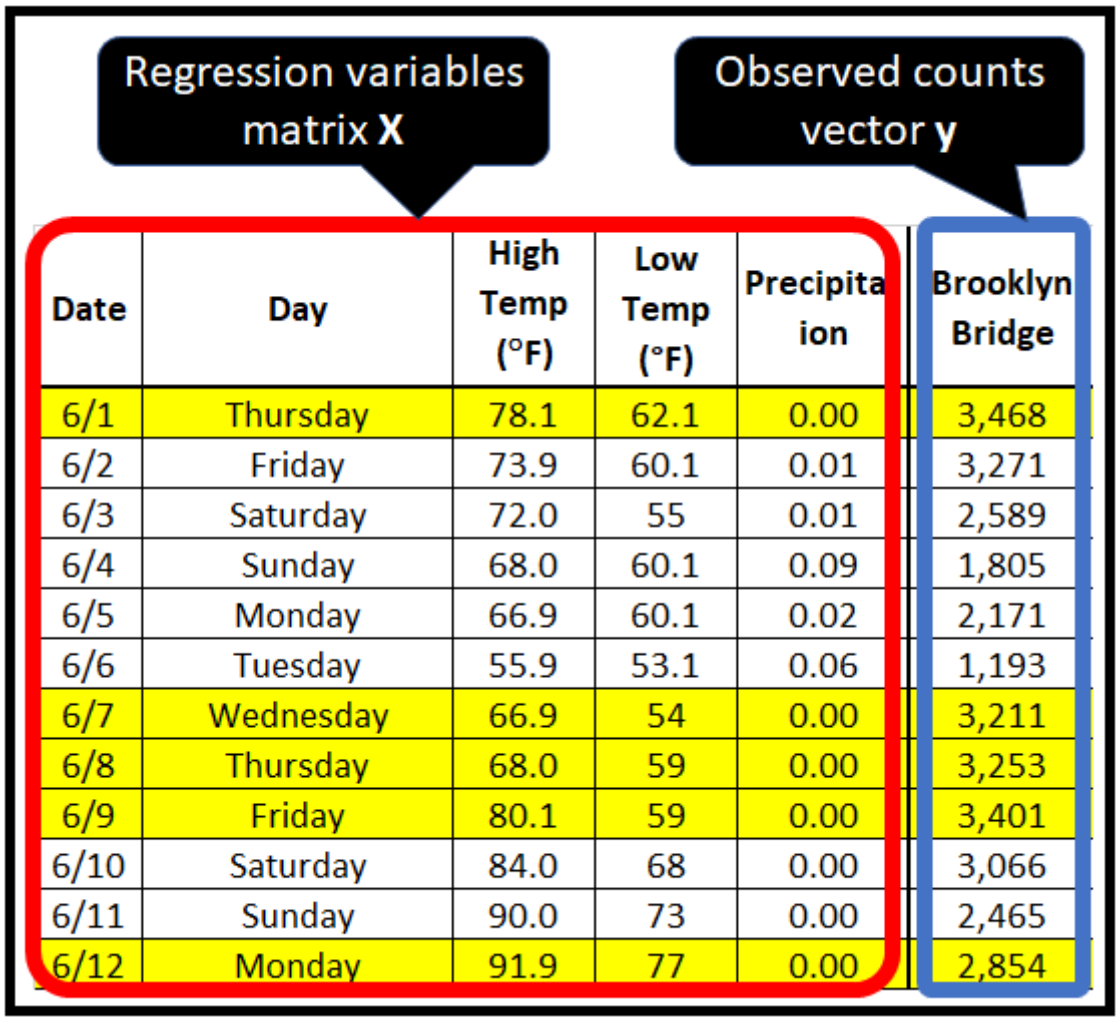
A Python program to generate event counts using a Poisson process

A Poisson regression model for a **non-constant λ**

Now we get to the fun part. Let us examine a more common situation, one where **λ** can change from one observation to the next. In this case, we assume that the value of **λ** is influenced by a vector of **explanatory variables**, **also known as predictors, regression variables**, or **regressors**. We'll call this matrix of regression variables, **X**.

The job of the regression model is to fit the observed counts **y** to the matrix of regression values **X** .

In the NYC bicyclist counts data set, the regression variables are *Date*, *Day of Week*, *High Temp*, *Low Temp* and *Precipitation*. We can also introduce additional regressors such as *Month* and *Day of Month* that are derived from *Date*, and we have the liberty to drop existing regressors such as *Date*.



Date	Day	High Temp (°F)	Low Temp (°F)	Precipitation	Brooklyn Bridge
6/1	Thursday	78.1	62.1	0.00	3,468
6/2	Friday	73.9	60.1	0.01	3,271
6/3	Saturday	72.0	55	0.01	2,589
6/4	Sunday	68.0	60.1	0.09	1,805
6/5	Monday	66.9	60.1	0.02	2,171
6/6	Tuesday	55.9	53.1	0.06	1,193
6/7	Wednesday	66.9	54	0.00	3,211
6/8	Thursday	68.0	59	0.00	3,253
6/9	Friday	80.1	59	0.00	3,401
6/10	Saturday	84.0	68	0.00	3,066
6/11	Sunday	90.0	73	0.00	2,465
6/12	Monday	91.9	77	0.00	2,854

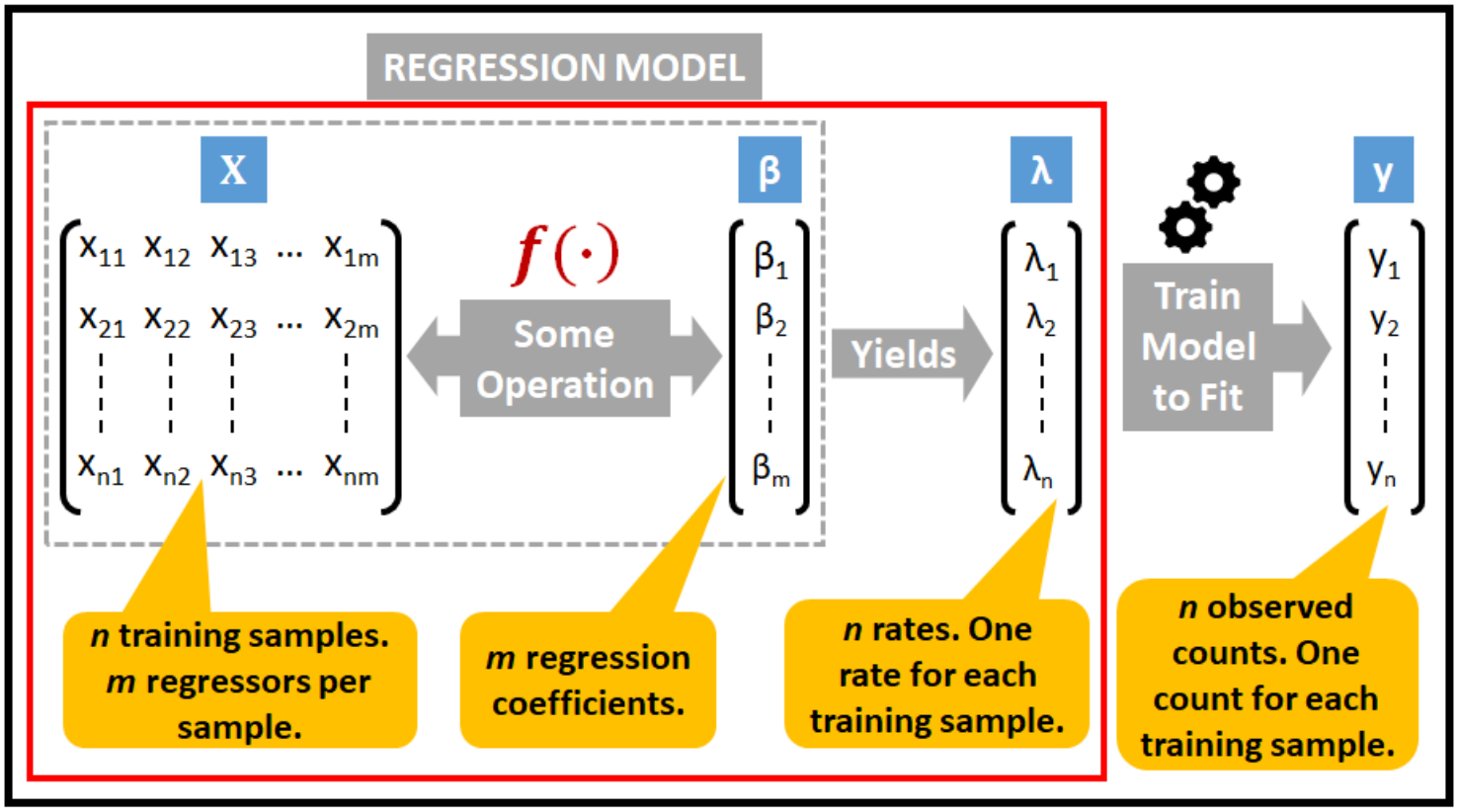
(Image by Author (https://www.linkedin.com/in/sachindate/))

The fitting of $\boldsymbol{y}$ to $\mathbf{X}$ happens by fixing the values of a vector of regression coefficients $\boldsymbol{\beta}$.

In a Poisson Regression model, the event counts $\boldsymbol{y}$ are assumed to be Poisson distributed, which means the probability of observing $\boldsymbol{y}$ is a function of the event rate vector $\boldsymbol{\lambda}$.

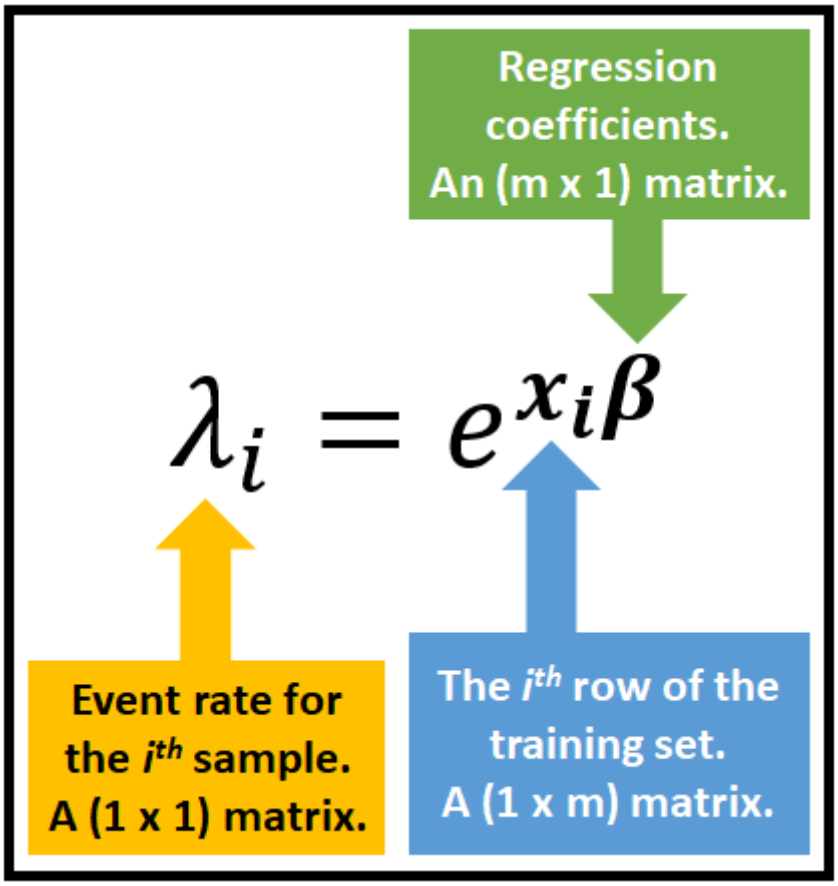
> The job of the Poisson Regression model is to fit the observed counts $\mathbf{y}$ to the regression matrix $\mathbf{X}$ via a link-function that expresses the rate vector $\boldsymbol{\lambda}$ as a function of, 1) the regression coefficients $\boldsymbol{\beta}$ and 2) the regression matrix $\mathbf{X}$.

The following figure illustrates the structure of the Poisson regression model.



Scan from left to right: The structure of the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

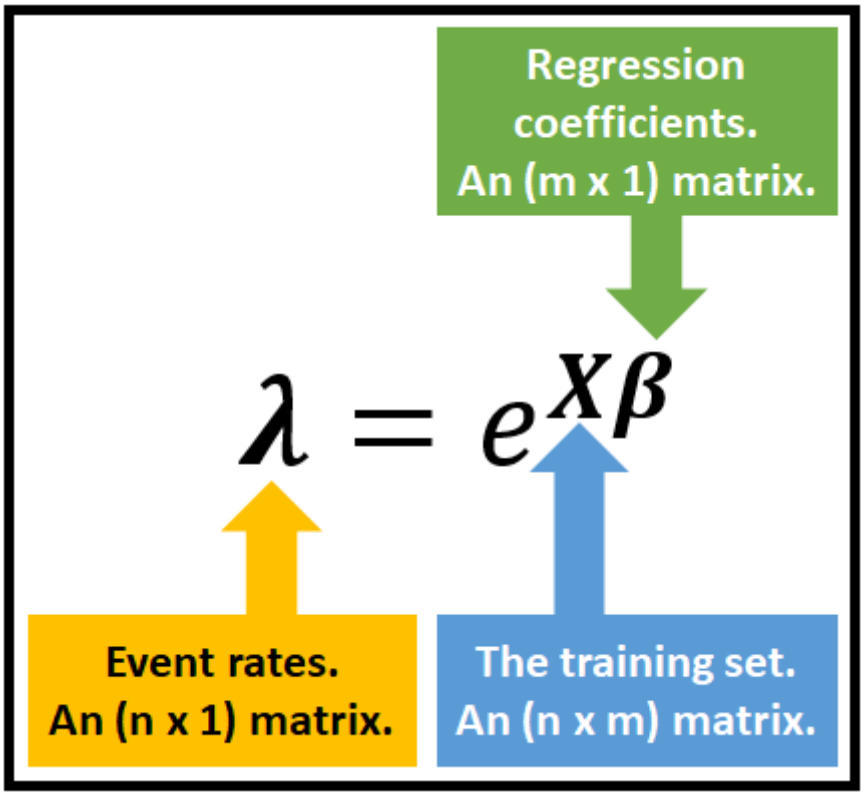
What might be a good link function $f(.)$ connecting $\boldsymbol{\lambda}$ with $\mathbf{X}$? It turns out the following exponential link-function works great:



The exponential link function of the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

This link function keeps λ non-negative even when the regressors $\mathbf{X}$ or the regression coefficients β have negative values. This is a requirement for count based data.

In general, we have:

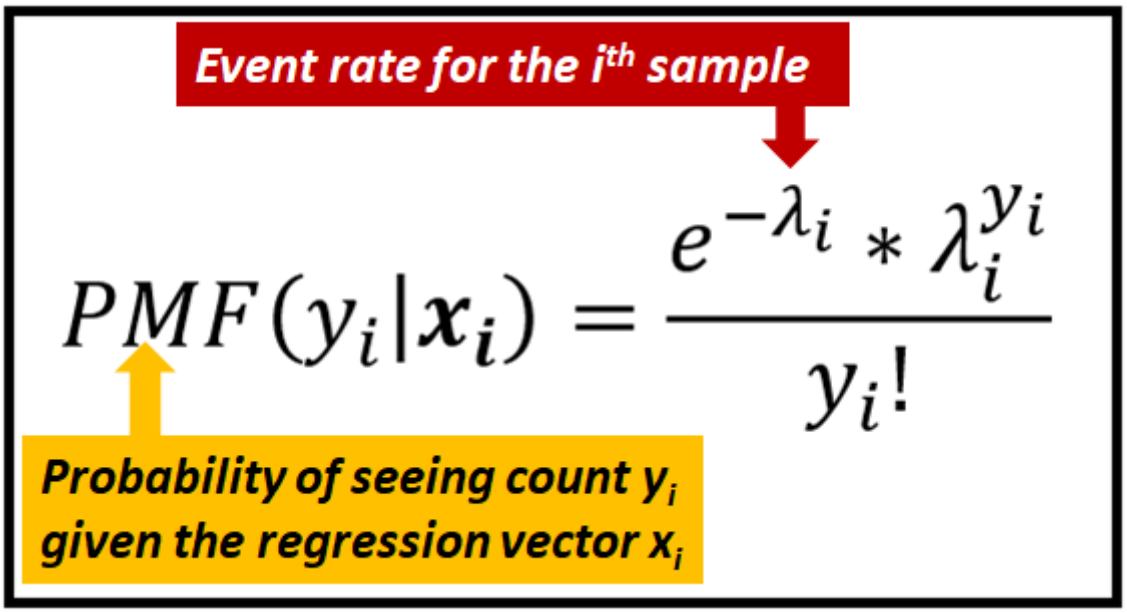


The exponential link function of the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

A Formal Specification of the Poisson Regression Model

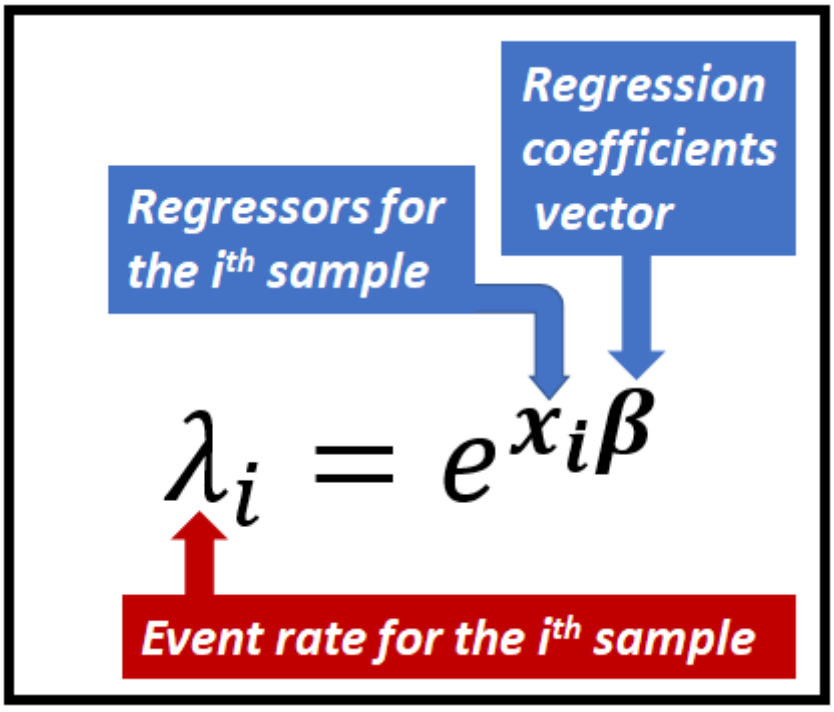
The complete specification of the Poisson regression model for count based data is given as follows:

For the *ith* observation in the data set denoted by y_i corresponding to the row of regression variables $\boldsymbol{x\_i}$, the probability of observing the count y_i is Poisson distributed as per the following PMF:



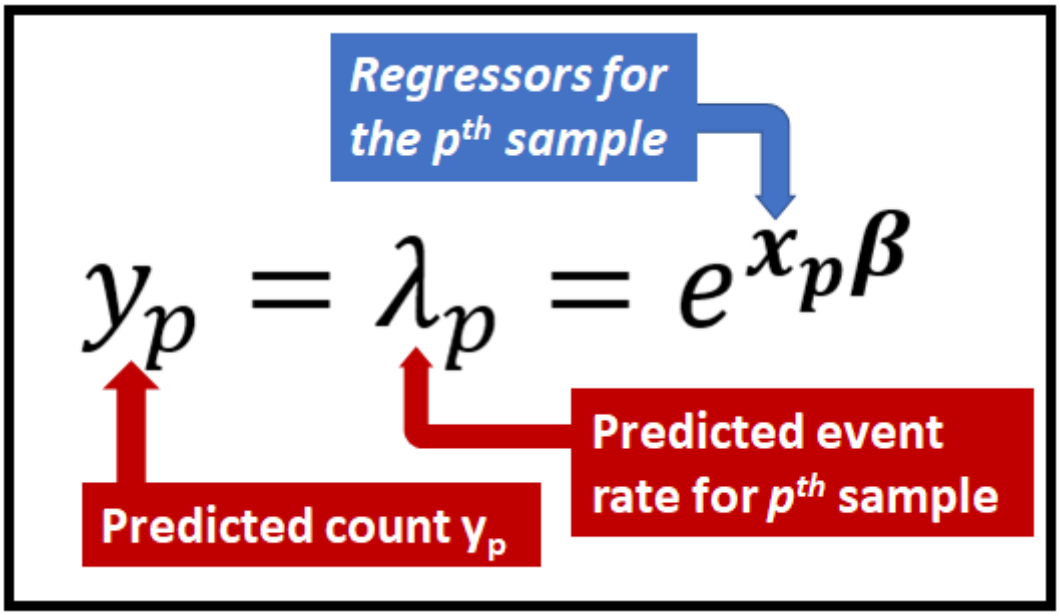
Probability of observing count y_i given **x_i** *(as per the Poisson PMF formula)* (Image by Author (https://www.linkedin.com/in/sachindate/))

Where the mean rate λ_i for the ith sample is given by the exponential link function shown earlier. We reproduce it here:



The exponential link function of the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

Once the model is fully trained on the data set, the regression coefficients $\boldsymbol{\beta}$ are known, and the model is ready to make predictions. To predict the event count y_p corresponding to an input row of regressors $\boldsymbol{x\_p}$ that one has observed, one uses this formula:



The prediction equation for the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

All of this hinges on our ability to train the model successfully so that the regression coefficients vector $\boldsymbol{\beta}$ is known.

Let's look at how this training takes place.

Training the Poisson regression model

Training a Poisson regression model involves finding the values of the regression coefficients $\boldsymbol{\beta}$ which would make the vector of observed counts $\mathbf{y}$ most likely.

The technique for identifying the coefficients $\boldsymbol{\beta}$ is called **M**aximum **L**ikelihood **E**stimation (MLE).

Let's get acquainted with the technique of **MLE.**

Understanding Maximum Likelihood Estimation (MLE)

I'll illustrate the **MLE** technique using the bicyclist counts data set. Take a look at the first few rows of this data set:

Date	Day	High Temp (°F)	Low Temp (°F)	Precipitation	Brooklyn Bridge
6/1	Thursday	78.1	62.1	0.00	3,468
6/2	Friday	73.9	60.1	0.01	3,271
6/3	Saturday	72.0	55	0.01	2,589
6/4	Sunday	68.0	60.1	0.09	1,805
6/5	Monday	66.9	60.1	0.02	2,171
6/6	Tuesday	55.9	53.1	0.06	1,193
6/7	Wednesday	66.9	54	0.00	

Daily counts of bicyclists on the Brooklyn bridge (Image by Author (https://www.linkedin.com/in/sachindate/))

Our assumption is that the bicyclist counts shown in the red box arise from a Poisson process. Hence we can say that their probabilities of occurrence is given by the Poisson PMF. Here are the probabilities for the first 4 occurrences:

$$P(3468|\boldsymbol{x_1}) = \frac{e^{-\lambda_1} * \lambda_1^{3468}}{3468!}$$

$$P(3271|\boldsymbol{x_2}) = \frac{e^{-\lambda_2} * \lambda_2^{3271}}{3271!}$$

$$P(2589|\boldsymbol{x_3}) = \frac{e^{-\lambda_3} * \lambda_3^{2589}}{2589!}$$

$$P(1805|\boldsymbol{x_4}) = \frac{e^{-\lambda_4} * \lambda_4^{1805}}{1805!}$$

Probabilities of observing the bicyclist counts for the first few occurrences given corresponding regression vectors (Image by Author

(https://www.linkedin.com/in/sachindate/)

We can similarly calculate the probabilities for all ***n*** counts observed in the training set.

Note that in the above formulae, *λ_1, λ_2, λ_3,…,λ_n* are calculated using the link function as follows:

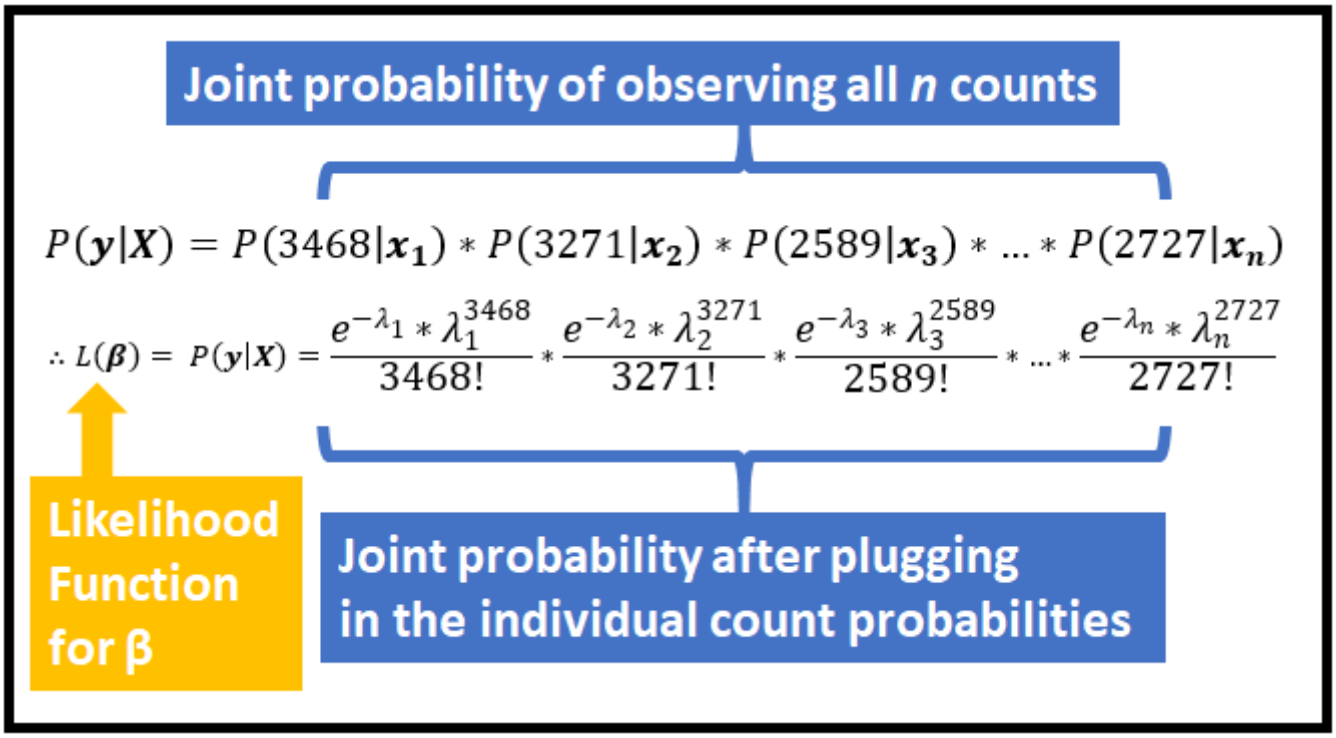
$$\lambda_1 = e^{x_1\beta}$$
$$\lambda_2 = e^{x_2\beta}$$
$$\lambda_3 = e^{x_3\beta}$$
$$\lambda_4 = e^{x_4\beta}$$

The event rates corresponding to the counts for the first few days (Image by Author (https://www.linkedin.com/in/sachindate/))

Where ***x_1, x_2, x_3, x_4*** are the first 4 rows of the regression matrix.

The probability of occurrence of the entire set of *n* counts *y_1, y_2,…,y_n* in the training set is the joint probability of occurrence of the individual counts.

The counts ***y*** are Poisson distributed, *y_1, y_2,…,y_n* are independent random variables, given correspondingly ***x_1, x_2,…,x_n***. Hence the joint probability of occurrence of *y_1, y_2,…,y_n* can be expressed as a simple multiplication of the individual probabilities. Here is how the joint probability looks like for the entire training set:



The **likelihood function** L(***β***) *expressed as a joint probability mass function* (Image by Author (https://www.linkedin.com/in/sachindate/))

Let's recollect that *λ_1, λ_2, λ_3,…,λ_n* are linked to the regression vectors ***x_1, x_2,x_3,…,x_n*** via the regression coefficients ***β.***

What value of ***β*** will make the given set of observed counts ***y*** most likely? It is the value of ***β*** for which the joint probability shown in the above equation achieves the maximum value. In other words, it is the value of ***β*** for which the rate of change of the joint probability function w.r.t. ***β*** is **0.** Put another way, it is the solution of the equation obtained from differentiating the joint probability equation w.r.t. ***β*** and setting this differential equation to **0.**

It is easier to differentiate the **logarithm** of the joint probability equation than the original equation. The solution to the logged equation yields the same optimal value of ***β***.

This logarithmic equation is called the **log-likelihood function**. For the Poisson regression, the log-likelihood function is given by the following equation:

$$\ln L(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i \boldsymbol{x_i \beta} - e^{\boldsymbol{x_i \beta}} - ln y_i!)$$

log-likelihood function for the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

The above equation is obtained by taking the natural logarithm of both sides of the joint probability function shown earlier, after substituting the λ_i with $exp(\mathbf{x\_i}^{*}\boldsymbol{\beta})$.

As mentioned earlier, we differentiate this log-likelihood equation w.r.t. $\boldsymbol{\beta}$, and set it to zero. This operation gives us the following equation:

$$\sum_{i=1}^{n} \left(y_i - e^{\boldsymbol{x_i \beta}}\right) \boldsymbol{x_i} = \mathbf{0}$$

The Poisson MLE for $\boldsymbol{\beta}$ *is the solution to this equation* (Image by Author (https://www.linkedin.com/in/sachindate/))

Solving this equation for the regression coefficients $\boldsymbol{\beta}$ will yield the **Maximum Likelihood Estimate (MLE)** for $\boldsymbol{\beta}$.

To solve the above equation one uses an iterative method such as Iteratively Reweighted Least Squares (IRLS) (https://en.wikipedia.org/wiki/Iteratively_reweighted_least_squares). In practice, one does not solve this equation by hand. Instead, you use statistical software such as the Python **statsmodels**

package which will do all the calculations for you while training the Poisson regression model on your data set.

Summary of steps for performing Poisson Regression

In summary, here are the steps for performing a Poisson Regression on a count based data set:

1. First, make sure that your data set contains counts. One way to tell is that it contains only non-negative integer values that represent the number of occurrences of some event during some interval. In the bicyclist counts data set, it is the count of bicyclists crossing the Brooklyn bridge per day.
2. Find out (or guess) the regression variables that will influence the observed counts. In the bicyclist counts data set the regression variables are *Day of Week, Min Temp, Max Temp, Precipitation* etc.
3. Carve out a training data set that your regression model will train on, and a test data set that should keep aside. Do *not* train the model on the test data.
4. Use a suitable statistical software such as the **Python statsmodels package** to configure and fit the Poisson Regression model on the training data set.
5. Test the performance of the model by running it on the test data set so as to generate predicted counts. Compare them with the actual counts in the test data set.
6. Use a goodness-of-fit measure to determine how well your model has trained on the training data set.

How to train a Poisson Regression Model in Python

Let's put into practice what we have learnt. The Python **statsmodels (https://www.statsmodels.org/stable/index.html)**package has excellent support for doing Poisson regression.

Let's use the Brooklyn bridge bicyclist counts data set. You can pick up the data set from **here** (https://gist.github.com/sachinsdate/c17931a3f000492c1c42cf78bf4ce9fe)**.**

Our goal is to build a Poisson regression model for the observed bicyclist counts ***y*.** We will use the trained model to predict daily counts of bicyclists on the Brooklyn bridge that the model has not seen during training.

Regression variables matrix **X**

Observed counts vector **y**

Date	Day	High Temp (°F)	Low Temp (°F)	Precipitation	Brooklyn Bridge
6/1	Thursday	78.1	62.1	0.00	3,468
6/2	Friday	73.9	60.1	0.01	3,271
6/3	Saturday	72.0	55	0.01	2,589
6/4	Sunday	68.0	60.1	0.09	1,805
6/5	Monday	66.9	60.1	0.02	2,171
6/6	Tuesday	55.9	53.1	0.06	1,193
6/7	Wednesday	66.9	54	0.00	3,211
6/8	Thursday	68.0	59	0.00	3,253
6/9	Friday	80.1	59	0.00	3,401
6/10	Saturday	84.0	68	0.00	3,066
6/11	Sunday	90.0	73	0.00	2,465
6/12	Monday	91.9	77	0.00	2,854
10/26	Thursday	57.0	53.1	0.00	2,565
10/27	Friday	62.1	48.0	0.00	3,150
10/28	Saturday	68.0	55.9	0.00	2,245
10/29	Sunday	64.9	61.0	3.03	183
10/30	Monday	55.0	46.0	0.25	1,428
10/31	Tuesday	54.0	44.0	0.00	2,727

Daily count of bicyclists on the Brooklyn bridge (Image by Author (https://www.linkedin.com/in/sachindate/))

We'll start by importing all the required packages.

```
import pandas as pd
from patsy import dmatrices
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

Create a pandas DataFrame for the counts data set.

```
df = pd.read_csv('nyc_bb_bicyclist_counts.csv', header=0,
infer_datetime_format=True, parse_dates=[0], index_col=[0])
```

We'll add a few derived regression variables to the **X** matrix.

```
ds = df.index.to_series()
df['MONTH'] = ds.dt.month
df['DAY_OF_WEEK'] = ds.dt.dayofweek
df['DAY'] = ds.dt.day
```

We will not use the *Date* variable as a regressor since it contains an absolute date value but we don't need to do anything special to drop *Date* as it is already consumed as the index of the pandas DataFrame. So it will not be available to us in the **X** matrix.

Let's create the training and testing data sets.

```
mask = np.random.rand(len(df)) < 0.8
df_train = df[mask]
df_test = df[~mask]
print('Training data set length='+str(len(df_train)))
print('Testing data set length='+str(len(df_test)))
```

Setup the regression expression in patsy notation. We are telling patsy that BB_COUNT is our dependent variable and it depends on the regression variables: DAY, DAY_OF_WEEK, MONTH, HIGH_T, LOW_T and PRECIP.

```
expr = """BB_COUNT ~ DAY  + DAY_OF_WEEK + MONTH + HIGH_T + LOW_T +
PRECIP"""
```

Set up the **X** and **y** matrices for the training and testing data sets. patsy makes this really simple.

```
y_train, X_train = dmatrices(expr, df_train, return_type='dataframe')
y_test, X_test = dmatrices(expr, df_test, return_type='dataframe')
```

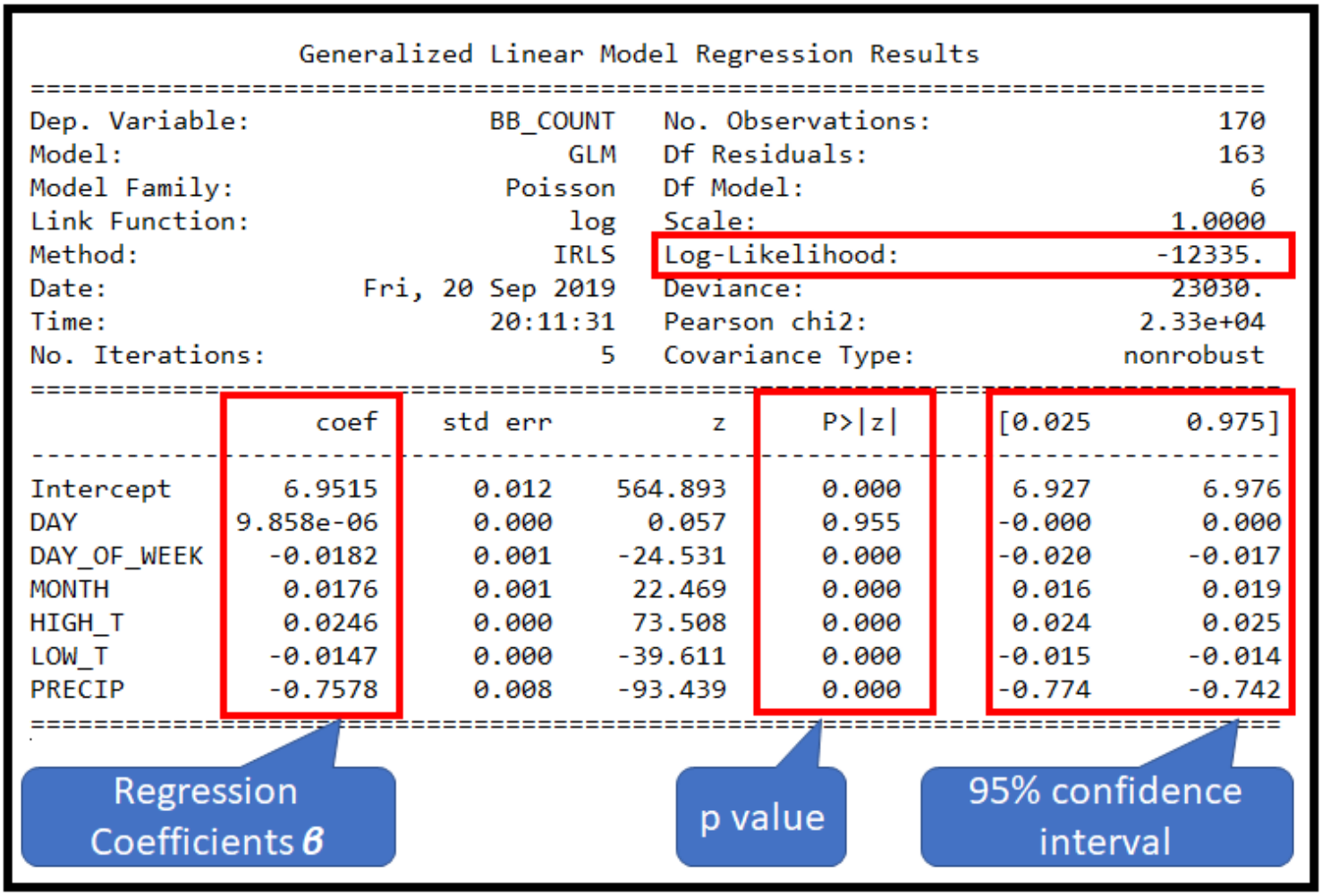
Using the statsmodels GLM class, train the Poisson regression model on the training data set.

```
poisson_training_results = sm.GLM(y_train, X_train,
family=sm.families.Poisson()).fit()
```

Print the training summary.

```
print(poisson_training_results.summary())
```

This prints out the following:

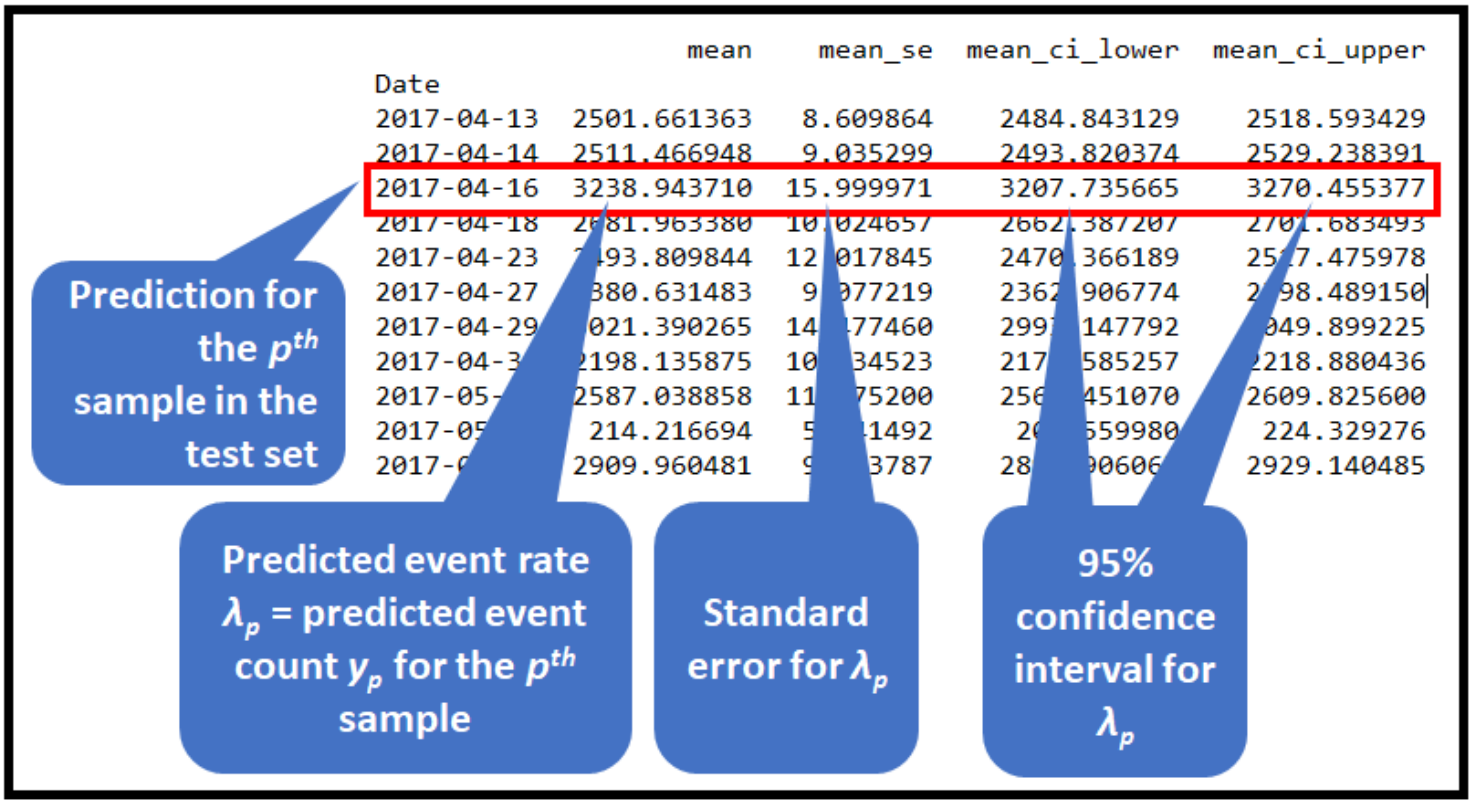


Training summary for the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

So how well did our model do? Let's make some **predictions** on the test data set.

```
poisson_predictions = poisson_training_results.get_prediction(X_test)
#summary_frame() returns a pandas DataFrame
predictions_summary_frame = poisson_predictions.summary_frame()
print(predictions_summary_frame)
```

Here are the first few rows of the output:



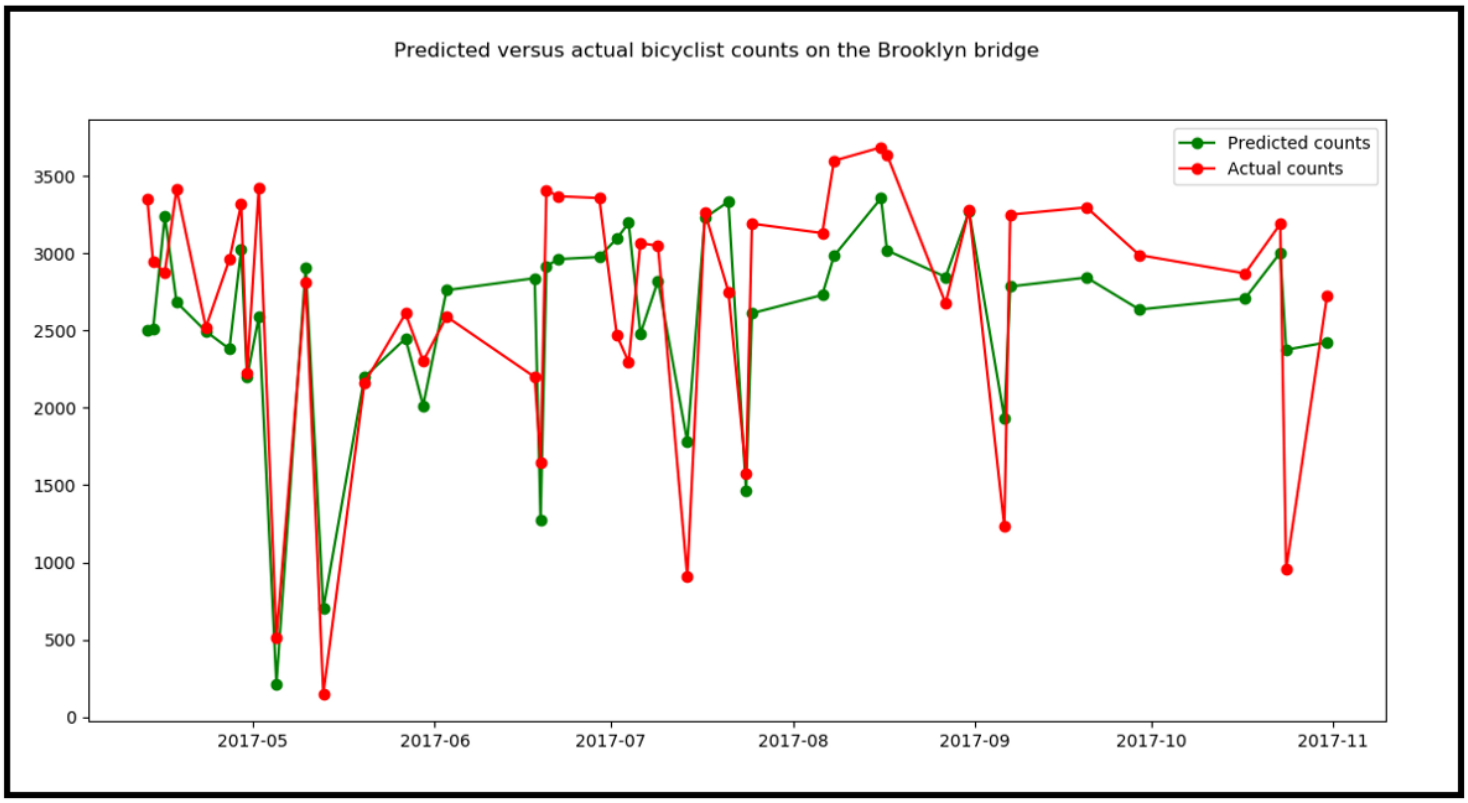
Date	mean	mean_se	mean_ci_lower	mean_ci_upper
2017-04-13	2501.661363	8.609864	2484.843129	2518.593429
2017-04-14	2511.466948	9.035299	2493.820374	2529.238391
2017-04-16	3238.943710	15.999971	3207.735665	3270.455377
2017-04-18	2681.963380	10.024657	2662.387207	2701.683493
2017-04-23	2493.809844	12.017845	2470.366189	2517.475978
2017-04-27	2380.631483	9.077219	2362.906774	2398.489150
2017-04-29	2021.390265	14.477460	2993.147792	3049.899225
2017-04-3	2198.135875	10.534523	2177.585257	2218.880436
2017-05-	2587.038858	11.075200	2565.451070	2609.825600
2017-05	214.216694	5.041492	204.559980	224.329276
2017-	2909.960481	9.583787	2891.90606	2929.140485

First few rows of output from **poisson_predictions.summary_frame()** (Image by Author (https://www.linkedin.com/in/sachindate/))

Let's plot the predicted counts versus the actual counts for the test data.

```
predicted_counts=predictions_summary_frame['mean']
actual_counts = y_test['BB_COUNT']
fig = plt.figure()
fig.suptitle('Predicted versus actual bicyclist counts on the Brooklyn bridge')
predicted, = plt.plot(X_test.index, predicted_counts, 'go-', label='Predicted counts')
actual, = plt.plot(X_test.index, actual_counts, 'ro-', label='Actual counts')
plt.legend(handles=[predicted, actual])
plt.show()
```

Here's the output:



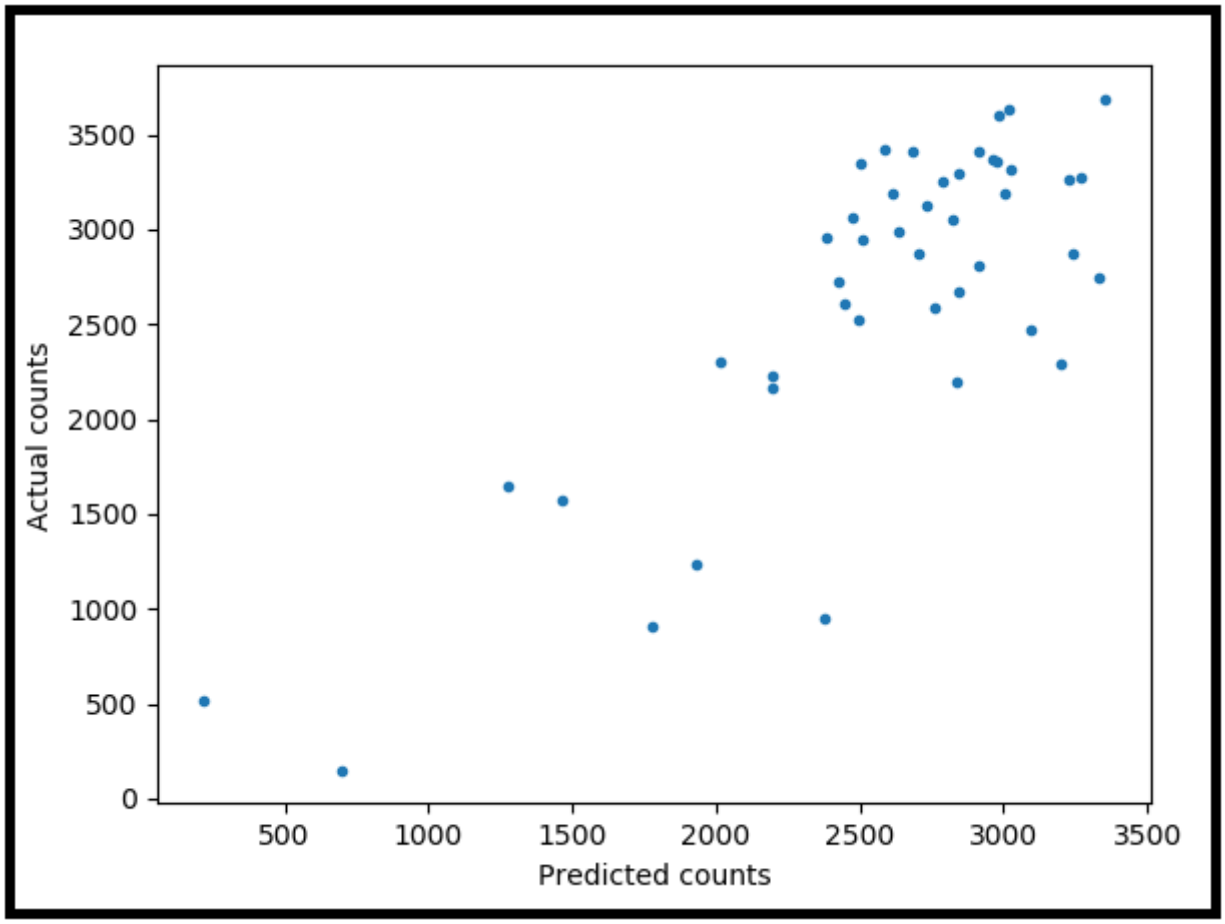
Predicted versus actual bicyclist counts on the Brooklyn bridge (Image by Author (https://www.linkedin.com/in/sachindate/))

The model seems to be more or less tracking the trend in the actual counts although **in many cases its predictions are way off the actual value**.

Let's also plot Actual versus Predicted counts.

```
plt.clf()
fig = plt.figure()
fig.suptitle('Scatter plot of Actual versus Predicted counts')
plt.scatter(x=predicted_counts, y=actual_counts, marker='.')
plt.xlabel('Predicted counts')
plt.ylabel('Actual counts')
plt.show()
```

Here is the plot:



Scatter plot of Actual versus Predicted counts
(Image by Author
(https://www.linkedin.com/in/sachindate/)
)

Here is the complete source code for doing Poisson regression using Python:

```
1  import pandas as pd
2  from patsy import dmatrices
3  import numpy as np
4  import statsmodels.api as sm
5  import matplotlib.pyplot as plt
6
7
8  #Create a pandas DataFrame for the counts data set.
9  df = pd.read_csv('nyc_bb_bicyclist_counts.csv', header=0, infer_datetime_format=True, parse_date
10
11 #Add a few derived regression variables.
12 ds = df.index.to_series()
13 df['MONTH'] = ds.dt.month
14 df['DAY_OF_WEEK'] = ds.dt.dayofweek
```

```
15  df['DAY'] = ds.dt.day
16
17  #Create the training and testing data sets.
18  mask = np.random.rand(len(df)) < 0.8
19  df_train = df[mask]
20  df_test = df[~mask]
21  print('Training data set length='+str(len(df_train)))
22  print('Testing data set length='+str(len(df_test)))
23
24  #Setup the regression expression in patsy notation. We are telling patsy that BB_COUNT is our de
25  # it depends on the regression variables: DAY, DAY_OF_WEEK, MONTH, HIGH_T, LOW_T and F
26  expr = """BB_COUNT ~ DAY  + DAY_OF_WEEK + MONTH + HIGH_T + LOW_T + PRECIP"""
27
28  #Set up the X and y matrices
29  y_train, X_train = dmatrices(expr, df_train, return_type='dataframe')
30  y_test, X_test = dmatrices(expr, df_test, return_type='dataframe')
31
32  #Using the statsmodels GLM class, train the Poisson regression model on the training data set.
33  poisson_training_results = sm.GLM(y_train, X_train, family=sm.families.Poisson()).fit()
34
35  #Print the training summary.
36  print(poisson_training_results.summary())
37
38  #Make some predictions on the test data set.
39  poisson_predictions = poisson_training_results.get_prediction(X_test)
40  #.summary_frame() returns a pandas DataFrame
41  predictions_summary_frame = poisson_predictions.summary_frame()
42  print(predictions_summary_frame)
43
44  predicted_counts=predictions_summary_frame['mean']
45  actual_counts = y_test['BB_COUNT']
46
47  #Mlot the predicted counts versus the actual counts for the test data.
48  fig = plt.figure()
49  fig.suptitle('Predicted versus actual bicyclist counts on the Brooklyn bridge')
50  predicted, = plt.plot(X_test.index, predicted_counts, 'go-', label='Predicted counts')
51  actual, = plt.plot(X_test.index, actual_counts, 'ro-', label='Actual counts')
52  plt.legend(handles=[predicted, actual])
53  plt.show()
54
55  #Show scatter plot of Actual versus Predicted counts
56  plt.clf()
```

```
57  fig = plt.figure()
58  fig.suptitle('Scatter plot of Actual versus Predicted counts')
59  plt.scatter(x=predicted_counts, y=actual_counts, marker='.')
60  plt.xlabel('Predicted counts')
61  plt.ylabel('Actual counts')
62  plt.show()
```

view raw poisson_regression.py hosted with ❤ by **GitHub**

Poisson Regression Model

Goodness-of-fit of the Poisson regression model

Recollect that both the expected value (i.e. mean) and the variance, of the Poisson distribution is λ. This rather strict condition is violated by most real-world data.

> A common source of failure of the Poisson regression model is that the data does not satisfy the *mean = variance* criterion imposed by the Poisson distribution.

The *summary()* method on the statsmodels *GLMResults* (https://www.statsmodels.org/devel/generated/statsmodels.genmod.generalized_linear_model.GLMResults.html) class shows a couple of useful goodness-of-fit statistics to help you evaluate whether your Poisson regression model was able to successfully fit the training data. Let's look at their values:

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                BB_COUNT   No. Observations:                  170
Model:                            GLM   Df Residuals:                      163
Model Family:                 Poisson   Df Model:                            6
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -12335.
Date:                Fri, 20 Sep 2019   Deviance:                       23030.
Time:                        20:11:31   Pearson chi2:                 2.33e+04
No. Iterations:                     5   Covariance Type:             nonrobust
```

Training summary for the Poisson regression model (Image by Author (https://www.linkedin.com/in/sachindate/))

The reported values of Deviance and Pearson chi-squared are very large. A good fit is virtually impossible given these values. To make a quantitative determination of the goodness-of-fit at some confidence level, say 95% (p=0.05), we look up the value in the $\chi 2$ table for p=0.05 and Degrees of freedom of residuals=163. (DF Residuals = No. Observations *minus* DF model]). We compare this Chi-Squared value with the observed statistic, in this case, the Deviance or the Pearson's chi-squared value reported in GLMResults. We find that at p=0.05 and DF Residuals = 163, the chi-squared value from a standard Chi-Squared table (https://www.medcalc.org/manual/chi-square-table.php) is 193.791 which is much smaller than the reported statistic of 23030 and 23300. Hence as per this test, the Poisson regression model, in spite of demonstrating an 'okay' *visual* fit for the test data set, has fit the training data rather poorly.

Conclusion and next steps

For counts based data, a useful starting point is the Poisson regression model. One can then compare its performance with other popular counts based models, such as:

1. The **The Negative Binomial regression model** (https://timeseriesreasoning.com/contents/negative-binomial-regression-model/) which does not make the *mean = variance* assumption about the data.
2. **Generalized Poisson regression models** (https://timeseriesreasoning.com/contents/generalized-poisson-regression-model/) which also work well with over-dispersed or under-dispersed data sets.
3. A **Zero Inflated Poisson model** (https://timeseriesreasoning.com/contents/zero-inflated-poisson-regression-model/) if you suspect that your data contains excess zeros i.e. many more zeroes than what the regular Poisson model can explain

Happy modeling!

Related

Getting to Know The Poisson Process And The Poisson Probability Distribution (https://timeseriesreasoning.com/contents/poisson-process/)

Citations and Copyrights

Data set

Bicycle Counts for East River Bridges. (https://data.cityofnewyork.us/Transportation/Bicycle-Counts-for-East-River-Bridges/gua4-p9wg) Daily total of bike counts conducted monthly on the Brooklyn Bridge, Manhattan Bridge, Williamsburg Bridge, and Queensboro Bridge. From NYC Open Data under Terms of Use. **Curated data set for download. (https://gist.github.com/sachinsdate/c17931a3f000492c1c42cf78bf4ce9fe)**

Book

Cameron A. C. and Trivedi P. K., Regression Analysis of Count Data (http://faculty.econ.ucdavis.edu/faculty/cameron/racd2/), Second Edition, Econometric Society Monograph No. 53, Cambridge University Press, Cambridge, May 2013.

Images

PREVIOUS: The Quantile Regression Model (https://timeseriesreasoning.com/contents/introduction-to-the-quantile-regression-model/)

NEXT: The Negative Binomial Regression Model (https://timeseriesreasoning.com/contents/negative-binomial-regression-model/)

UP: Table of Contents (https://timeseriesreasoning.com/)