Impact of Medication Adherence on Hospital Admissions among Medicaid patient populations with Heart Failure

Toral Shah

Model Project

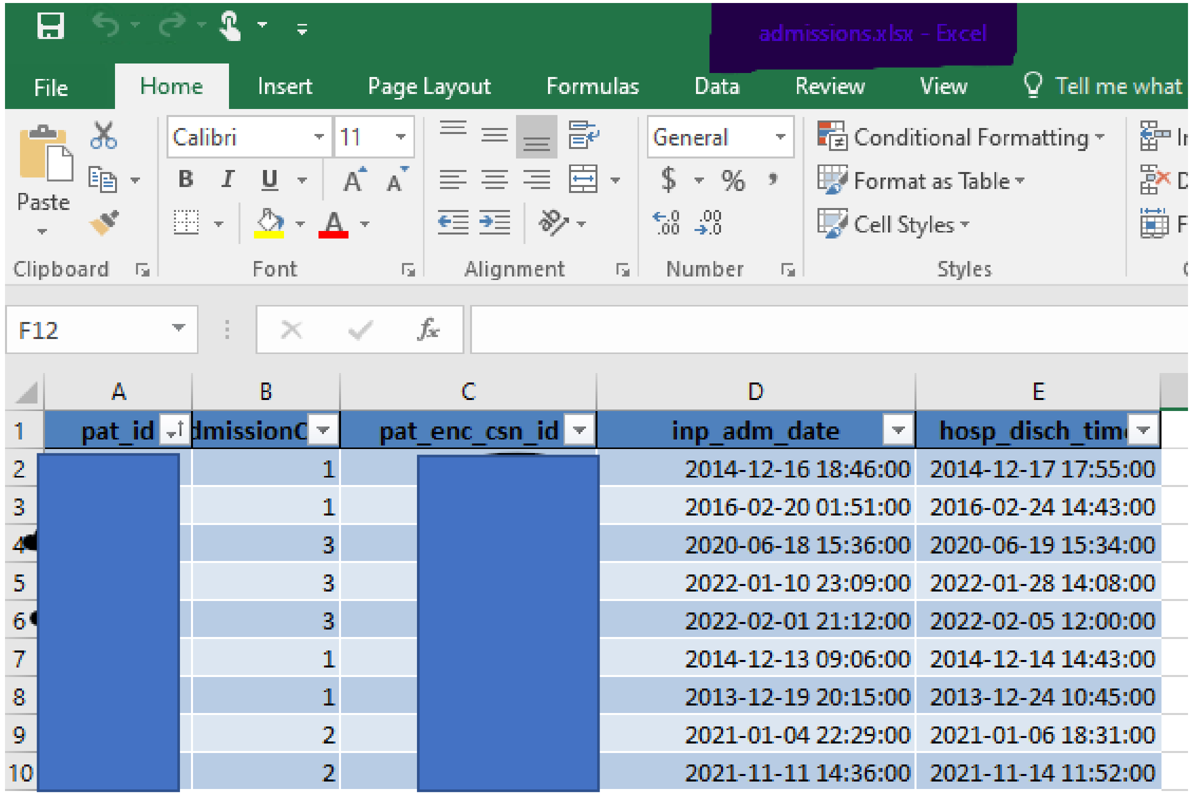
Impact of Medication Adherence on Health Services Utilization in Medicaid

Mark C. Roebuck, PhD, Robert J. Kaestner, PhD,† and Julia S. Dougherty, PhD‡*

Data Processing

- **Dataset criteria**
  - Age 18 -64
  - Health Insurance: Medicaid
  - Diagnosis: Heart failure and any of the chosen 7 comorbidity:
    - Congestive Heart Failure, Hypertension, Dyslipidemia, Diabetes, Asthma/COPD, Depression or Schizophrenia/Bipolar
  - At least one prescription order from defined given classes of medications
  - Inpatient hospital patients only
- **Challenges for Data Collection**
  - Required knowledge and expertise of SQL and Excel for preliminary data pull and cleaning
  - Required subject matter expertise on EPIC Clarity and Caboodle databases
  - Data was collected in four excel spreadsheets with different count numbers
    - Cohort1 : Admission records from 2017 to 2020
    - Cohort2: Diagnosis records with defined chronic conditions (based on charlson comorbidity Index)
    - Cohort3: Medication orders and pdc (proportion of days covered) with defined classes pulling from Pharmacy claims
    - Cohort4: Insurance – filter Medicaid

Data Processing



admissions.xlsx - Excel

	A	B	C	D	E
1	pat_id	dmissionC	pat_enc_csn_id	inp_adm_date	hosp_disch_tim
2	[redacted]	1	[redacted]	2014-12-16 18:46:00	2014-12-17 17:55:00
3		1		2016-02-20 01:51:00	2016-02-24 14:43:00
4		3		2020-06-18 15:36:00	2020-06-19 15:34:00
5		3		2022-01-10 23:09:00	2022-01-28 14:08:00
6		3		2022-02-01 21:12:00	2022-02-05 12:00:00
7		1		2014-12-13 09:06:00	2014-12-14 14:43:00
8		1		2013-12-19 20:15:00	2013-12-24 10:45:00
9		2		2021-01-04 22:29:00	2021-01-06 18:31:00
10		2		2021-11-11 14:36:00	2021-11-14 11:52:00

Cohort1: Admissions

Calculated Total count of admissions based on inpatient admission and discharge dates. Csn id is a defined id for encounters – here it is filtered for hospital encounters.

Data Processing

pat_i	dx
[redacted]	Hypertension
[redacted]	Hypertension
[redacted]	Hypertension
[redacted]	Heart Failure
[redacted]	Heart Failure
[redacted]	Hypertension
[redacted]	Hypertension
[redacted]	Hypertension
[redacted]	Heart Failure
[redacted]	Heart Failure

Pat_ID	COPD	Depression	Diabetes	Dyslipidemia	Heart Failure	Hypertension	Schizophrenia	Grand Total
[redacted]						1		1
[redacted]						1		1
[redacted]					2	1		3
[redacted]						2		2
[redacted]						1		1
[redacted]					2			2
[redacted]	3					1		4

Cohort2: Diagnosis

Calculated total number of comorbidities

Data Processing

	A	B	C
1	pat_i	hx_met_last_upd_dttr	pdc_scor
2	[redacted]	2020-11-20 21:05:00	69
3		2020-09-14 13:21:00	32
4		2020-07-20 15:55:00	26
5		2021-01-07 14:07:00	96
6		2021-10-15 15:56:00	69
7		2021-04-21 18:24:00	96
8		2021-09-09 13:50:00	66
9		2020-10-20 22:05:00	52
10		2021-11-04 06:16:00	81
11		2020-12-02 17:21:00	76
12		2020-02-13 03:43:00	100
13		2021-12-08 16:42:00	92
14		2021-05-12 16:14:00	84

A	B
[redacted]	pdc_avg
	73
	49
	81
	97
	47
	92
	0
	63
	40
	56

Cohort3: Medications

Medications orders data were merged with pdc score (Proportion of Days Covered) for each medication which then were averaged to calculate overall pdc score for each patient.

Data Processing

pat_id	payor_name	benefit_plan_name	prod_type
	FIDELIS CARE	FIDELIS CARE MEDICAID MANAGED CARE	Medicaid Managed Care
	HEALTHFIRST	HEALTHFIRST MEDICAID	Medicaid Managed Care
	EMPIRE BCBS HEALTH PLUS	BCBS HEALTHPLUS NY MEDICAID MANAGED CARE NY	Medicaid Managed Care
	MEDICAID	MEDICAID	Medicaid
	FIDELIS CARE	FIDELIS CARE MEDICAID MANAGED CARE	Medicaid Managed Care
	METROPLUS	METROPLUS MCD MANAGED CARE	Medicaid Managed Care

Cohort4: Insurance data

Patient Insurance data were filtered by Medicaid as their primary insurance

Analyzing Data

In [2]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [3]:
```python
df1 = pd.read_csv('H:/Toral/MLProject/Analysis/admissions.csv')
df1.head()
```

Out[3]:

	pat_id	AdmissionCount
0	[redacted]	1
1	[redacted]	1
2	[redacted]	3
3	[redacted]	1
4	[redacted]	1

In [4]:
```python
df2 = pd.read_csv('H:/Toral/MLProject/Analysis/dx.csv')
df2.head()
```

Out[4]:

	pat_id	COPD	Depression	Diabetes	Dyslipidemia	Heart Failure	Hypertension	Schizophrenia	TotalComorb
0	[redacted]	NaN	NaN	NaN	NaN	NaN	1.0	NaN	1
1	[redacted]	NaN	NaN	NaN	NaN	NaN	1.0	NaN	1
2	[redacted]	NaN	NaN	NaN	NaN	2.0	1.0	NaN	3
3	[redacted]	NaN	NaN	NaN	NaN	NaN	2.0	NaN	2
4	[redacted]	NaN	NaN	NaN	NaN	NaN	1.0	NaN	1

In [5]:
```python
df2=df2.drop(columns=['COPD', 'Depression', 'Diabetes','Dyslipidemia', 'Hypertension', 'Schizophrenia', 'Heart Failure']
```

Analyzing Data

In [6]:
```python
df_comorb = pd.merge(df1, df2, on="pat_id")
df_comorb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4810 entries, 0 to 4809
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   pat_id          4810 non-null   object
 1   AdmissionCount  4810 non-null   int64 
 2   TotalComorb     4810 non-null   int64 
dtypes: int64(2), object(1)
memory usage: 150.3+ KB
```

In [7]:
```python
df3 = pd.read_csv('H:/Toral/MLProject/Analysis/pdc.csv')
df3.head()
```

Out[7]:

	pat_id	pdc_avg
0	[redacted]	73
1	[redacted]	49
2	[redacted]	81
3	[redacted]	97
4	[redacted]	47

In [8]:
```python
df_pdc = pd.merge(df_comorb, df3, on="pat_id")
df_pdc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4176 entries, 0 to 4175
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   pat_id          4176 non-null   object
 1   AdmissionCount  4176 non-null   int64 
 2   TotalComorb     4176 non-null   int64 
 3   pdc_avg         4176 non-null   int64 
dtypes: int64(3), object(1)
memory usage: 163.1+ KB
```

Analyzing Data

```
In [9]: df4 = pd.read_csv('H:/Toral/MLProject/Analysis/insurance.csv')
        df4
```

	pat_id	prod_type
0	[redacted]	MedicaidManagedCare
1	[redacted]	MedicaidManagedCare
2	[redacted]	MedicaidManagedCare
3	[redacted]	Medicaid
4	[redacted]	MedicaidManagedCare
...	...	...
13010	[redacted]	MedicaidManagedCare
13011	[redacted]	MedicaidManagedCare
13012	[redacted]	Medicaid
13013	[redacted]	MedicaidManagedCare
13014	[redacted]	MedicaidManagedCare

13015 rows × 2 columns

```
In [10]: df_cohort = pd.merge(df_pdc, df4, on="pat_id")
         df_cohort.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 569 entries, 0 to 568
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   pat_id          569 non-null    object
 1   AdmissionCount  569 non-null    int64
 2   TotalComorb     569 non-null    int64
 3   pdc_avg         569 non-null    int64
 4   prod_type       569 non-null    object
dtypes: int64(3), object(2)
memory usage: 26.7+ KB
```

```
In [10]: df_cohort['Med_Adh'] = np.where(df_cohort['pdc_avg']<80, 0, 1)
         df_cohort.head()
```

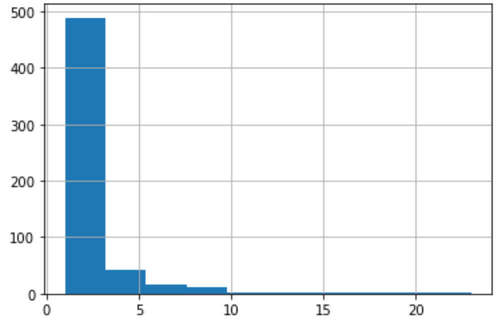
Analyzing Data

In [10]:
```
df_cohort['Med_Adh'] = np.where(df_cohort['pdc_avg']<80, 0, 1)
df_cohort.head()
```

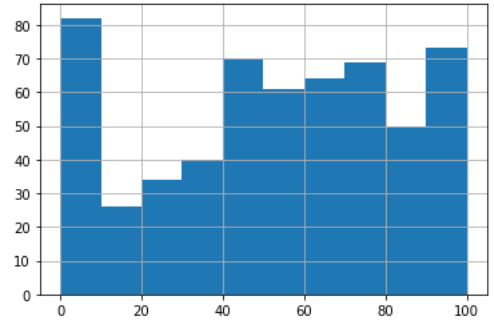
Out[10]:

	pat_id	AdmissionCount	TotalComorb	pdc_avg	prod_type	Med_Adh
0	[redacted]	2	2	83	Medicaid Managed Care	1
1	[redacted]	2	2	0	Medicaid	0
2	[redacted]	1	1	43	Medicaid	0
3	[redacted]	1	1	64	Medicaid Managed Care	0
4	[redacted]	4	9	81	Medicaid	1

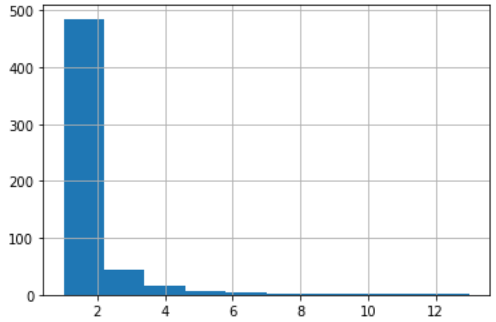
In [11]:
```
df_cohort['TotalComorb'].hist()
plt.show()
```



In [12]:
```
df_cohort['pdc_avg'].hist()
plt.show()
```



In [13]:
```
df_cohort['AdmissionCount'].hist()
plt.show()
```



Analyzing Data

In [14]:
```
one_hot_encoded_data = pd.get_dummies(df_cohort, columns=['prod_type'])
one_hot_encoded_data.head()
```

Out[14]:

	pat_id	AdmissionCount	TotalComorb	pdc_avg	Med_Adh	prod_type_Medicaid	prod_type_Medicaid Managed Care
0	[redacted]	2	2	83	1	0	1
1	[redacted]	2	2	0	0	1	0
2	[redacted]	1	1	43	0	1	0
3	[redacted]	1	1	64	0	0	1
4	[redacted]	4	9	81	1	1	0

In [15]:
```
one_hot_encoded_data.fillna(0)
```

Out[15]:

	pat_id	AdmissionCount	TotalComorb	pdc_avg	Med_Adh	prod_type_Medicaid	prod_type_Medicaid Managed Care
0	[redacted]	2	2	83	1	0	1
1	[redacted]	2	2	0	0	1	0
2	[redacted]	1	1	43	0	1	0
3	[redacted]	1	1	64	0	0	1
4	[redacted]	4	9	81	1	1	0
...	...	...	...	...	...	...	...
564	[redacted]	1	1	100	1	1	0
565	[redacted]	2	2	76	0	0	1
566	[redacted]	1	1	94	1	0	1
567	[redacted]	1	2	100	1	1	0
568	[redacted]	1	2	0	0	0	1

569 rows × 7 columns

Fitting of y to X

Observed Admission Counts Vector y

Regression variables matrix X

AdmissionCount	TotalComorb	pdc_avg	Med_Adh	prod_type_Medicaid	prod_type_Medicaid Managed Care
2	2	83	1	0	1
2	2	0	0	1	0
1	1	43	0	1	0
1	1	64	0	0	1
4	9	81	1	1	0

Analyzing Data

```
In [16]: X = one_hot_encoded_data.drop(columns=['AdmissionCount','pat_id'])
```

```
In [17]: X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 569 entries, 0 to 568
Data columns (total 5 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   TotalComorb                      569 non-null    int64
 1   pdc_avg                          569 non-null    int64
 2   Med_Adh                          569 non-null    int32
 3   prod_type_Medicaid               569 non-null    uint8
 4   prod_type_Medicaid Managed Care  569 non-null    uint8
dtypes: int32(1), int64(2), uint8(2)
memory usage: 16.7 KB
```

```
In [35]: #y = one_hot_encoded_data['Med_Adh']
```

```
In [18]: y = one_hot_encoded_data['AdmissionCount']
```

```
In [19]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =.20)
```

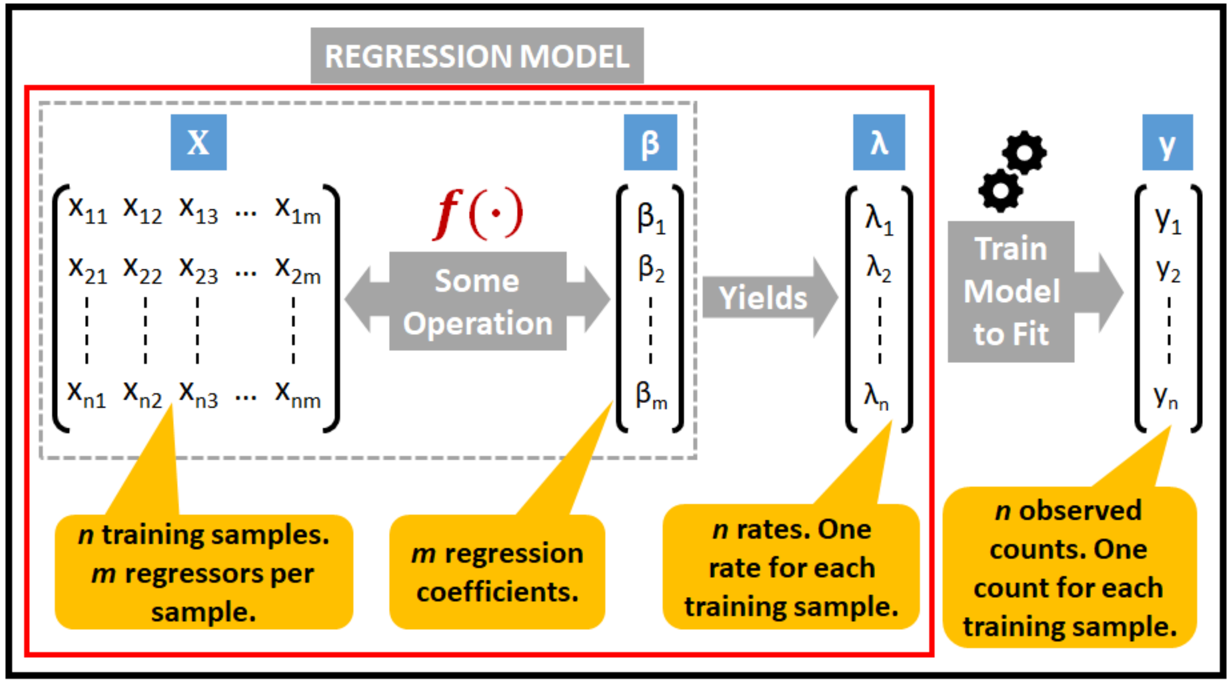
```
In [20]: print("size of X-train is", X_train.shape)
         print("size of y-train is", y_train.shape)
         print("size of X-test is", X_test.shape)
         print("size of y-test is", y_test.shape)
```

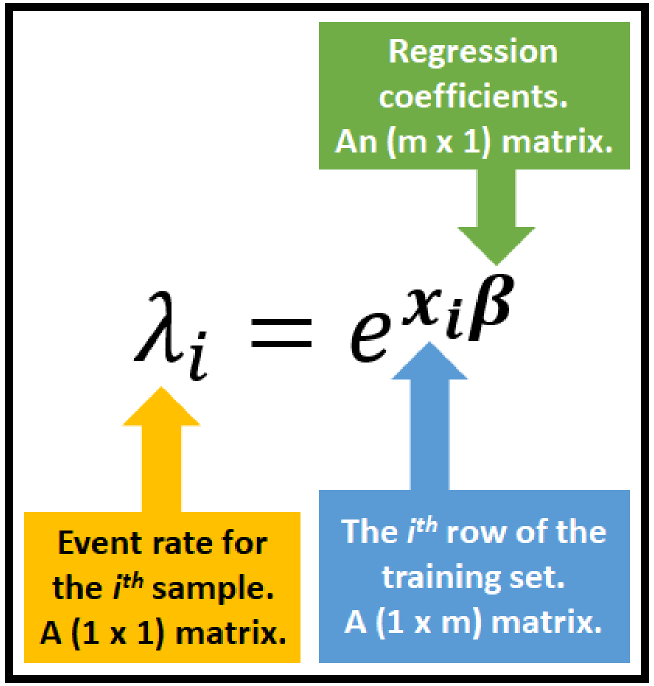
```
size of X-train is (455, 5)
size of y-train is (455,)
size of X-test is (114, 5)
size of y-test is (114,)
```

Analyzing Data

```
In [21]: from sklearn.linear_model import LogisticRegression
         LRClassifier = LogisticRegression(random_state = 0)
         LRClassifier.fit(X_train, y_train)

C:\Apps\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Out[21]: LogisticRegression(random_state=0)

In [22]: prediction = LRClassifier.predict(X_test)
         print(prediction)

[1 1 1 1 1 2 1 1 2 1 2 1 1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 2 1
 1 2 1 1 1 1 1 4 2 1 2 1 4 1 1 2 2 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 4
 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 4 1 2 1 1 2 1 1 1 1 1 4 1 1 1 1 1 1 1 1 1 1
 1 1 1]

In [23]: from sklearn.metrics import accuracy_score
         print("Accuracy is", accuracy_score(y_test, prediction))

Accuracy is 0.7631578947368421
```

Structure of Poisson Regression Model





Structure of Poisson Regression Model

Event rate for the i^{th} sample

$$PMF(y_i|\boldsymbol{x_i}) = \frac{e^{-\lambda_i} * \lambda_i^{y_i}}{y_i!}$$

Probability of seeing count y_i given the regression vector x_i

Regressors for the i^{th} sample

Regression coefficients vector

$$\lambda_i = e^{\boldsymbol{x_i\beta}}$$

Event rate for the i^{th} sample

Regressors for the p^{th} sample

$$y_p = \lambda_p = e^{\boldsymbol{x_p\beta}}$$

Predicted count y_p

Predicted event rate for p^{th} sample

Poisson Regression Model

In [60]:
```
# Create Training and Test data sets
mask = np.random.rand(len(one_hot_encoded_data)) < 0.8
df_train = one_hot_encoded_data[mask]
df_test = one_hot_encoded_data[~mask]
print('Training data set length='+str(len(df_train)))
print('Testing data set length='+str(len(df_test)))
```

```
Training data set length=458
Testing data set length=111
```

In [61]:
```
# Setup the regression expression in patsty notation. Basically we are telling patsy that Admissions Count is our
# dependent variable and it depends on the regreassion variables:
#Total comorb, pdc_avg, Med_adh, prod_type_medicaid, Prod_type_Medicaid Managed care

expr = """AdmissionCount ~ TotalComorb  + pdc_avg + Med_Adh + prod_type_Medicaid + prod_type_MedicaidManagedCare"""
```

In [62]:
```
# Setup X and y matrices for the training and testing data sets.

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

In [ ]:
```
# using the statsmodels GLM class, train the Poisson regresion model on the training data set
```

In [63]:
```
poisson_training_results = sm.GLM(y_train, X_train, family=sm.families.Poisson()).fit()
```

Poisson Regression Model

```
In [ ]: # using the statsmodels GLM class, train the Poisson regresion model on the training data set

In [63]: poisson_training_results = sm.GLM(y_train, X_train, family=sm.families.Poisson()).fit()

In [64]: print(poisson_training_results.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:         AdmissionCount   No. Observations:                  458
Model:                            GLM   Df Residuals:                      453
Model Family:                 Poisson   Df Model:                            4
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -582.02
Date:                Sun, 01 May 2022   Deviance:                       111.69
Time:                        11:41:24   Pearson chi2:                     125.
No. Iterations:                     5
Covariance Type:            nonrobust
=================================================================================================
                                    coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
Intercept                         0.0804      0.052      1.548      0.122      -0.021       0.182
TotalComorb                       0.1519      0.008     18.958      0.000       0.136       0.168
pdc_avg                          -0.0013      0.002     -0.796      0.426      -0.005       0.002
Med_Adh                           0.0160      0.118      0.135      0.892      -0.216       0.248
prod_type_Medicaid                0.0069      0.050      0.139      0.889      -0.090       0.104
prod_type_MedicaidManagedCare     0.0735      0.052      1.417      0.157      -0.028       0.175
=================================================================================================
```

95% Confidence interval

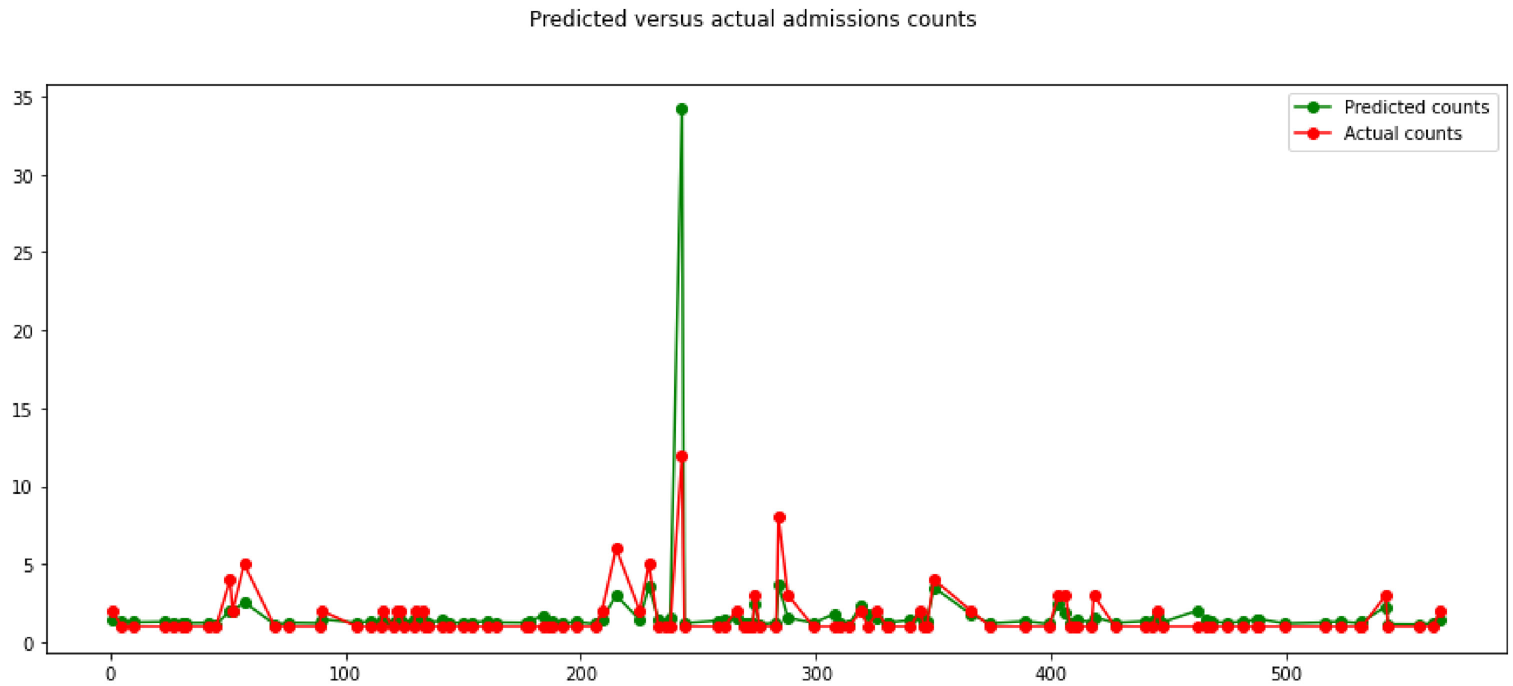
```
In [65]: poisson_predictions = poisson_training_results.get_prediction(X_test)
         #summary_frame() returns a pandas DataFrame
         predictions_summary_frame = poisson_predictions.summary_frame()
         print(predictions_summary_frame)
```

```
         mean   mean_se  mean_ci_lower  mean_ci_upper
1    1.478599  0.121418       1.258787       1.736795
5    1.352270  0.118791       1.138385       1.606340
10   1.276897  0.065698       1.154412       1.412378
23   1.307923  0.077435       1.164628       1.468850
27   1.225178  0.090338       1.060319       1.415670
..        ...       ...            ...            ...
542  2.279676  0.135268       2.029391       2.560827
543  1.158522  0.127555       0.933654       1.437550
556  1.144720  0.131638       0.913722       1.434115
562  1.212153  0.104927       1.023000       1.436282
565  1.428068  0.099830       1.245218       1.637768

[111 rows x 4 columns]
```

Prediction for the p^{th} sample in the test set

Poisson Regression Model

```
In [74]: predicted_counts=predictions_summary_frame['mean']
         actual_counts = y_test['AdmissionCount']
         fig = plt.figure(figsize=(15,6))
         fig.suptitle('Predicted versus actual admissions counts')
         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()
```



Poisson Regression Model

In [75]:
```
plt.clf()
fig = plt.figure(figsize=(15, 6))
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()
```

```
<Figure size 432x288 with 0 Axes>
```