Most data scientists who do not have strong statistics background may think of logistic regression as a machine learning model. That’s true. But it has been around for a long time in statistics.

Statistics is a very important part of data science and machine learning. Because it is essential for any type of exploratory data analysis or machine learning algorithm to learn the features of a dataset, how they relate to each other, how one feature affects the other features and the overall output.

Luckily python has this amazing library that is ‘statsmodels’ library. This library has great functionalities to understand the dataset and also we can use this library to make predictions. Statsmodels library already has models in-built that can be fitted to the data to find the correlation between the features, learn the coefficients, p-value, test-statistic, standard error, and confidence interval.

This article will explain a statistical modeling technique with an example. I will explain a logistic regression modeling for binary outcome variables here. That means the outcome variable can have only two values, 0 or 1.

We will also analyze:

1. the correlation amongst the predictor variables (the input variables that will be used to predict the outcome variable),

2. how to extract useful information from the model results,

3. the visualization techniques to better present and understand the data and

4. the prediction of the outcome. **I am assuming that you have the basic knowledge of statistics and python.**

The sequence of discussion:

1. A basic logistic regression model fitting with one variable
2. Understanding odds and log-odds with examples
3. Logistic regression model fitting with two variables
4. Logistic regression model fitting with three variables
5. Visualization of the fitted model
6. Prediction

If you need a refresher on confidence interval and hypothesis testing, please check out these articles to clearly understand those topics:

<https://towardsdatascience.com/confidence-interval-calculation-and-characteristics-1a60fd724e1d>

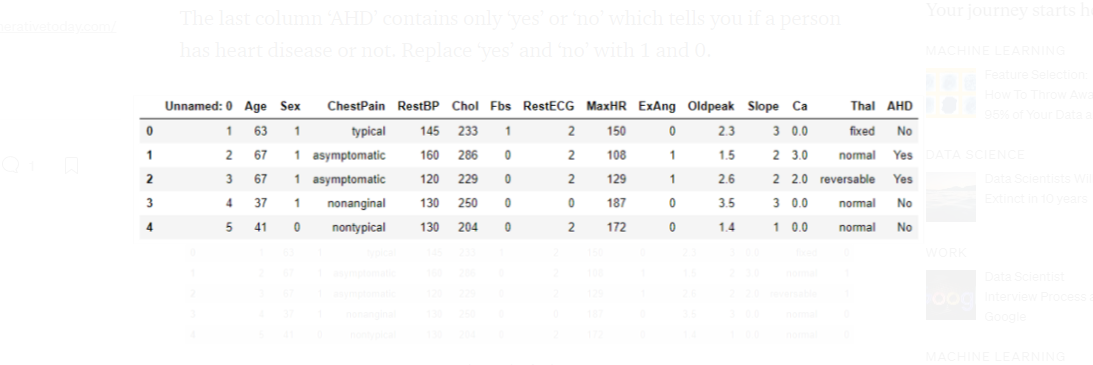
# **The Tools Used**

For this tutorial, we will use:

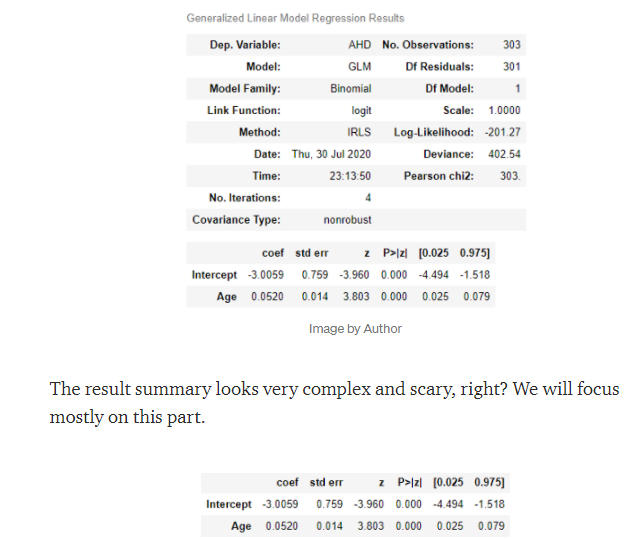
1. [Numpy Library](https://numpy.org/)
2. [Pandas Library](https://pandas.pydata.org/)
3. [Matplotlib Library](https://matplotlib.org/)
4. [Seaborn Library](https://seaborn.pydata.org/)
5. [Statsmodels Library](https://www.statsmodels.org/v0.10.1/)
6. [Jupyter Notebook](https://jupyter.org/) environment.

Let’s import the necessary packages and the dataset.

%matplotlib inline  
import matplotlib.pyplot as plt  
import seaborn as sns  
import pandas as pd  
import statsmodels.api as sm  
import numpy as npdf = pd.read\_csv('Heart.csv')  
df.head()



model = sm.GLM.from\_formula("AHD ~ Age", family = sm.families.Binomial(), data=df)result = model.fit()  
result.summary()



Now, let’s understand all the terms above.

**coef**

First, we have the coefficients where -3.0059 is the B, and 0.0520 is our A (think of the linear regression formula Y = AX + B).

If a person’s age is 1 unit more s/he will have a 0.052 (coefficient with age in the table above) unit more chance of having heart disease based on the p-value in the table.

**std err**

There is a standard error of 0.014 that indicates the distance of the estimated slope from the true slope.

**z**

z-statistic of 3.803 means that the predicted slope is going to be 3.803 unit above the zero.

**Confidence intervals**

And the last two columns are the confidence intervals (95%). Here the confidence interval is 0.025 and 0.079. Later we will visualize the confidence intervals throughout the length of the data.

# **Odds And Log Odds**

A logistic regression model provides the ‘odds’ of an event. Remember that, ‘odds’ are the probability on a different scale. Here is the formula:

If an event has a probability of p,

the odds of that event are p/(1-p)

Odds are the transformation of the probability. **Based on this formula, if the probability is 1/2, the ‘odds’ is 1.**

To understand the odds and log-odds clearly, let’s work on an example. We will use the gender variable.

**Because a categorical variable is appropriate for this.** Check the proportion of males and females having heart disease in the dataset.

df["Sex1"] = df.Sex.replace({1: "Male", 0:"Female"})  
c = pd.crosstab(df.Sex1, df.AHD)  
c = c.apply(lambda x: x/x.sum(), axis=1)

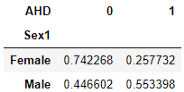


Image by Author

Let’s calculate the ‘odds’ of heart disease for males and females.

c["odds"] = c.loc[:, 1] / c.loc[:, 0]

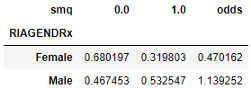


Image by Author

The ‘odds’ show that the probability of a female having heart disease is substantially lower than a male(32% vs 53%) that reflects very well in the odds. Odds ratios are common to use while working with two population groups.

c.odds.Male / c.odds.Female

**The ratio comes out to be 3.587 which indicates a man has a 3.587 times greater chance of having a heart disease**.

Remember that, an individual probability cannot be calculated from an odd ratio

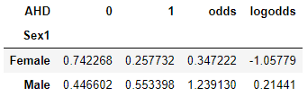
Another important convention is to work with log-odds which are odds in a logarithmic scale.

Recall that the neutral point of the probability is 0.5. Using the formula for ‘odds’, odds for 0.5 is 1 and ‘log-odds’ is 0 (log of 1 is 0).

*In our exercise where men have a greater chance of having heart disease, have ‘odds’ between 1 and infinity. At the same time, the ‘odds’ of women having a greater chance of having heart disease is 0 to 1.*

Here is the log odds calculation:

c['logodds'] = np.log(c.odds)



Here, the log-odds of the female population are negative which indicates that less than 50% of females have heart disease.

Log-odds of males are positive and a little more than 0 which means more than half of the males have heart disease.

**The relationship between log odds and logistic regression will be more clear from the model summary below.**Let’s see the model summary using the gender variable only:

model = sm.GLM.from\_formula("AHD ~ Sex1", family = sm.families.Binomial(), data=df)result = model.fit()  
result.summary()

https://miro.medium.com/max/60/0*XjUnlfaQy0viXUab.png?q=20

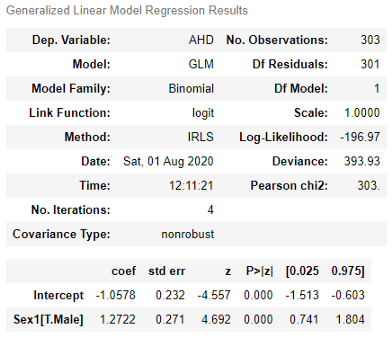


Image by Author

Look at the coefficients above.

The logistic regression coefficient of males is 1.2722 which should be the same as the log-odds of males minus the log-odds of females.

c.logodds.Male - c.logodds.Female

This difference is exactly 1.2722.

# **A logistic regression Model With Three Covariates**

We can use multiple covariates. I am using both ‘Age’ and ‘Sex1’ variables here. Before we dive into the model, we can conduct an initial analysis with the categorical variables. Check the proportion of males and females having heart disease in the dataset.

df["Sex1"] = df.Sex.replace({1: "Male", 0:"Female"})  
c = pd.crosstab(df.Sex1, df.AHD)  
c = c.apply(lambda x: x/x.sum(), axis=1)

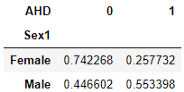


Image by Author

Now, generate a model using both the ‘Age’ and ‘Sex’ variable.

model = sm.GLM.from\_formula("AHD ~ Age + Sex1", family = sm.families.Binomial(), data=df)  
result = model.fit()  
result.summary()

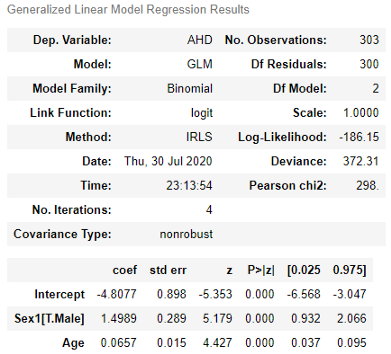


Image by Author

**Understand the coefficients better**. Adding gender to the model changed the coefficient of the ‘Age’ parameter a little(0.0520 to 0.0657).

According to this fitted model, older people are more likely to have heart disease than younger people. **The log odds for heart disease increases by 0.0657 units for each year.**

*If a person is 10 years older his or her chance of having heart disease increases by 0.0657 \* 10 = 0.657 units.*

In the case of the gender variable, the female is the reference as it does not appear in the output.

While comparing a male and a female of the same age, the male has a 1.4989 units higher chance of having a heart disease.

Now, let’s see the effect of both gender and age. If a 40 years old female is compared to 50 years old male, the log odds for the male having heart disease is 1.4989 + 0.0657 \* 10 = 2.15559 units greater than the female.

*All the coefficients are in log-odds scale. You can exponentiate the values to convert them to the odds*

# **A logistic regression Model With Three Covariates**

Now, we will fit a logistic regression with three covariates. This time we will add ‘Chol’ or cholesterol variables with ‘Age’ and ‘Sex1’.

model = sm.GLM.from\_formula("AHD ~ Age + Sex1 + Chol", family = sm.families.Binomial(), data=df)  
result = model.fit()  
result.summary()

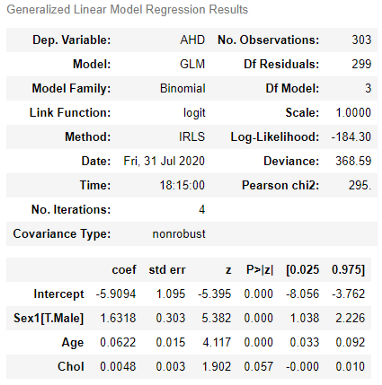
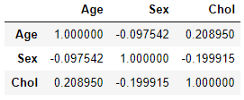


Image by Author

As you can see, after adding the ‘Chol’ variable, the coefficient of the ‘Age’ variable reduced a little bit and the coefficient of the ‘Sex1’ variable went up a little.

The change is more in ‘Sex1’ coefficients than the ‘Age’ coefficient. This is because ‘Chol’ is better correlated to the ‘Sex1’ covariate than the ‘Age’ covariate. Let’s check the correlations:

df[['Age', 'Sex', 'Chol']].corr()



# **Visualization of the Fitted Model**

We will begin by plotting the fitted proportion of the population that have heart disease for different subpopulations defined by the regression model. We will plot how the heart disease rate varies with the age.

We will fix some values that we want to focus on in the visualization. We will visualize the effect of ‘Age’ on the female population having a cholesterol level of 250.

from statsmodels.sandbox.predict\_functional import predict\_functionalvalues = {"Sex1": "Female", "Sex":0, "AHD": 1, "Chol": 250}pr, cb, fv = predict\_functional(result, "Age", values=values, ci\_method="simultaneous")ax = sns.lineplot(fv, pr, lw=4)  
ax.fill\_between(fv, cb[:, 0], cb[:, 1], color='grey', alpha=0.4)  
ax.set\_xlabel("Age")  
ax.set\_ylabel("Heart Disease")

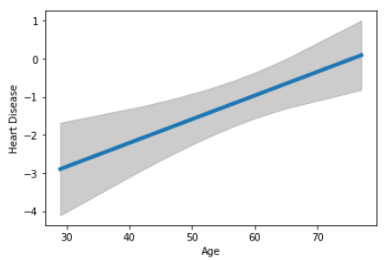


Image by Author

We just plotted the fitted log-odds probability of having heart disease and the 95% confidence intervals. The confidence band is more appropriate. The confidence band looks curvy which means that it’s not uniform throughout the age range.

We can visualize in terms of probability instead of log-odds. The probability can be calculated from the log odds using the formula 1 / (1 + exp(-lo)), where lo is the log-odds.

pr1 = 1 / (1 + np.exp(-pr))  
cb1 = 1 / (1 + np.exp(-cb))  
ax = sns.lineplot(fv, pr1, lw=4)  
ax.fill\_between(fv, cb1[:, 0], cb[:, 1], color='grey', alpha=0.4)  
ax.set\_xlabel("Age", size=15)  
ax.set\_ylabel("Heart Disease")

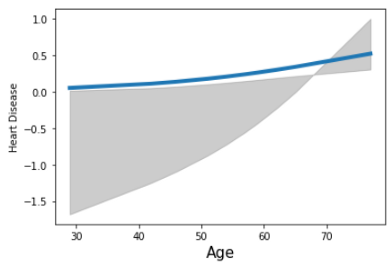


Image by Author

Here is the problem with the probability scale sometimes.

While the probability values are limited to 0 and 1, the confidence intervals are not.

The plots above plotted the average. On average that was the probability of a female having heart disease given the cholesterol level of 250. Next, we will visualize in a different way that is called a partial residual plot.

In this plot, it will show the effect of one covariate only while the other covariates are fixed. This shows even the smaller discrepancies. So, the plot will not be as smooth as before. Remember, t**he small discrepancies are not reliable if the sample size is not very large.**

from statsmodels.graphics.regressionplots import add\_lowess  
fig = result.plot\_partial\_residuals("Age")  
ax = fig.get\_axes()[0]  
ax.lines[0].set\_alpha(0.5)  
\_ = add\_lowess(ax)

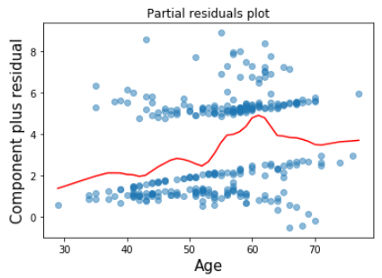


Image by Author

This plot shows that the heart disease rate rises rapidly from the age of 53 to 60.

# **Prediction**

Using the results from the model, we can predict if a person has heart disease or not. The models we fitted before were to explain the model parameters. For the prediction purpose, I will use all the variables in the DataFrame. Because we do not have too many variables. Let’s check the correlations amongst the variables.

df['ChestPain'] = df.ChestPain.replace({"typical":1, "asymptomatic": 2, 'nonanginal': 3, 'nontypical':4})df['Thal'] = df.Thal.replace({'fixed': 1, 'normal': 2, 'reversable': 3})  
df[['Age', 'Sex1', 'Chol','RestBP', 'Fbs', 'RestECG', 'Slope', 'Oldpeak', 'Ca', 'ExAng', 'ChestPain', 'Thal']].corr()



Image by Author

We can see that each variable has some correlations with other variables. I will use all the variables to get a better prediction.

model = sm.GLM.from\_formula("AHD ~ Age + Sex1 + Chol + RestBP+ Fbs + RestECG + Slope + Oldpeak + Ca + ExAng + ChestPain + Thal", family = sm.families.Binomial(), data=df)  
result = model.fit()  
result.summary()

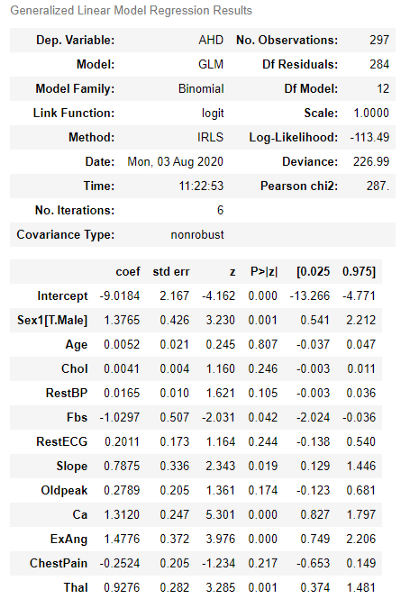


Image by Author

We can use the predict function to predict the outcome. But the predict function uses only the DataFrame. So, let’s prepare a DataFrame with the variables and then use the predict function.

X = df[['Age', 'Sex1', 'Chol','RestBP', 'Fbs', 'RestECG', 'Slope', 'Oldpeak', 'Ca', 'ExAng', 'ChestPain', 'Thal']]  
predicted\_output = result.predict(X)

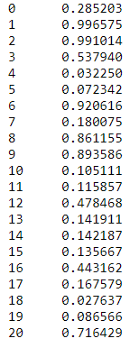


Image by Author

The predicted output should be either 0 or 1. It’s 1 when the output is greater than or equal to 0.5 and 0 otherwise.

for i in range(0, len(predicted\_output)):  
 predicted\_output = predicted\_output.replace()  
 if predicted\_output[i] >= 0.5:  
 predicted\_output = predicted\_output.replace(predicted\_output[i], 1)  
 else:  
 predicted\_output = predicted\_output.replace(predicted\_output[i], 0)

Now, compare this predicted\_output to the ‘AHD’ column of the DataFrame which indicates the heart disease to find the accuracy:

accuracy = 0  
for i in range(0, len(predicted\_output)):  
 if df['AHD'][i] == predicted\_output[i]:  
 accuracy += 1  
accuracy/len(df)

The accuracy comes out to be 0.81 or 81% which is very good.