Foundations of Logistic Regresion

#Advanced_Data_Analytics #Regression/Logistic #Regression/Logistic/Logit #Regression/Logistic/Beta_Coefficients #Regression/Logistic/MLE #Regression/Logistic/Confusion_Matrix #Exercise #Programming/Python

Recall that as we navigate the first two stages of pace, we can figure out if a logistic regression model is the best choice to address the question we're working on.

Binomial Logistic regression

A binomial logistic regression (often referred to simply as logistic regression), predicts the probability that an observation falls into one of two categories of a dichotomous dependent variable based on one or more independent variables that can be either continuous or categorical.

What are the Main assumptions for Logistic Regression?

1. Linearity
2. Independent observations
3. No multicollinearity
4. No extreme outliers

Assumptions for logistic regression : Linearity

There should be a linear relationship between each X variable and the `logit` of the probability that $Y = 1$

The linearity assumption is the KEY assumption that explains how we can estimate a logistic model that fits the data best. To do this **we must define the `Odds`.

$$Odds = \frac{p}{1-p}$$

The mathematical definition is the probability of P occurring divided by the probability of P NOT occurring.

Lets do an example : Cookie package

Let's imagine that in a package of cookies with different flavors:

- You know that about 60% are chocolate. You'll represent this as 0.6.
- Then the probability of a cookie not being chocolate is 0.4, because 1 -0.60 is 0.4,
  - the odds a given cookies, chocolate is 0.6, divided by 0.4, which is 1.5. The `logit` or `log-odds function` is the logarithms of the odds of a given probability. So the logit of probability P is equal to the logarithms of P divided by 1 -P.

$$Odds = \frac{0.6}{0.4} = 1.5$$

$$Log.odds = log(1.5) = 0.1760$$

What is the Logit (log-odds function) ?

The logarithm of the odds of a given probability. So the logit of probability p is equals to the logarithm of p divided by 1 minus p

$$Logit(p) = log(\frac{p}{1-p})$$

- *Logit is the **most common** `link function` used to linearly relate the X variables to the probability of Y.*

Logit in terms of X variables

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

Assumptions for logistic regression : Independent Observation

This assumption relates to how the data was collected.

- Because the observations are assumed to be independent, we can say that the probability of observing data point A and observing data point B, is equal to the probability of observing A times the probability of observing B.

Because the observations are independent

$$P(A \text{ AND } B) = P(A) * P(B)$$

The best logistic regression model estimates the set of `beta coefficients` *that maximizes the likelihood of observing all of the sample data*

Figuring out the best model for *logistic regression*

When figuring out the best `Beta Coefficients` (B_0 , B_1 , B_n) lets remember what we know:

Linear Regression (OLS Model)

When we performed `Linear regression` models we used the *minimum squared residuals (SSR, SSE)* or also known as `Ordinary least squares` as our method to find the best fit B_0 coefficients.

> Note : SSR is a measure of error , residuals means error.

Logistic Regression (MLE Model)

In logistic regression, we'll often use maximum likelihood estimation to find the best logistic regression model.

What is Maximum likelihood estimation ... ?

MLE, is a technique for estimating the `beta parameters` that maximize the likelihood of the model producing the observed data.

Common metrics to assess logistic regression results

#Advanced_Data_Analytics #Regression/Logistic/metrics/ROC #Regression/Logistic/metrics/accuracy #Regression/Logistic/metrics/recall #Regression/Logistic/metrics/AUC #Regression/Logistic/metrics/precision #Programming/Python/sklearn #Regression/Logistic/Classifier #Regression/Logistic/Confusion_Matrix

Evaluation metrics

- Precision
- Recall
- Accuracy
- ROC curve
- AUC

[You can access the functions used in this video through scikit-learn metrics module]

Precision

The proportion of positive predictions that were true positive

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Using `sklearn.metrics` as `metrics`
  - `metrics.precision_score(y_test, y_pred)`

Recall

The proportion of positives the model was able to identify correctly.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- Using `sklearn.metrics` as `metrics`
  - `metrics.recall_score(y_test, y_pred)`

Accuracy

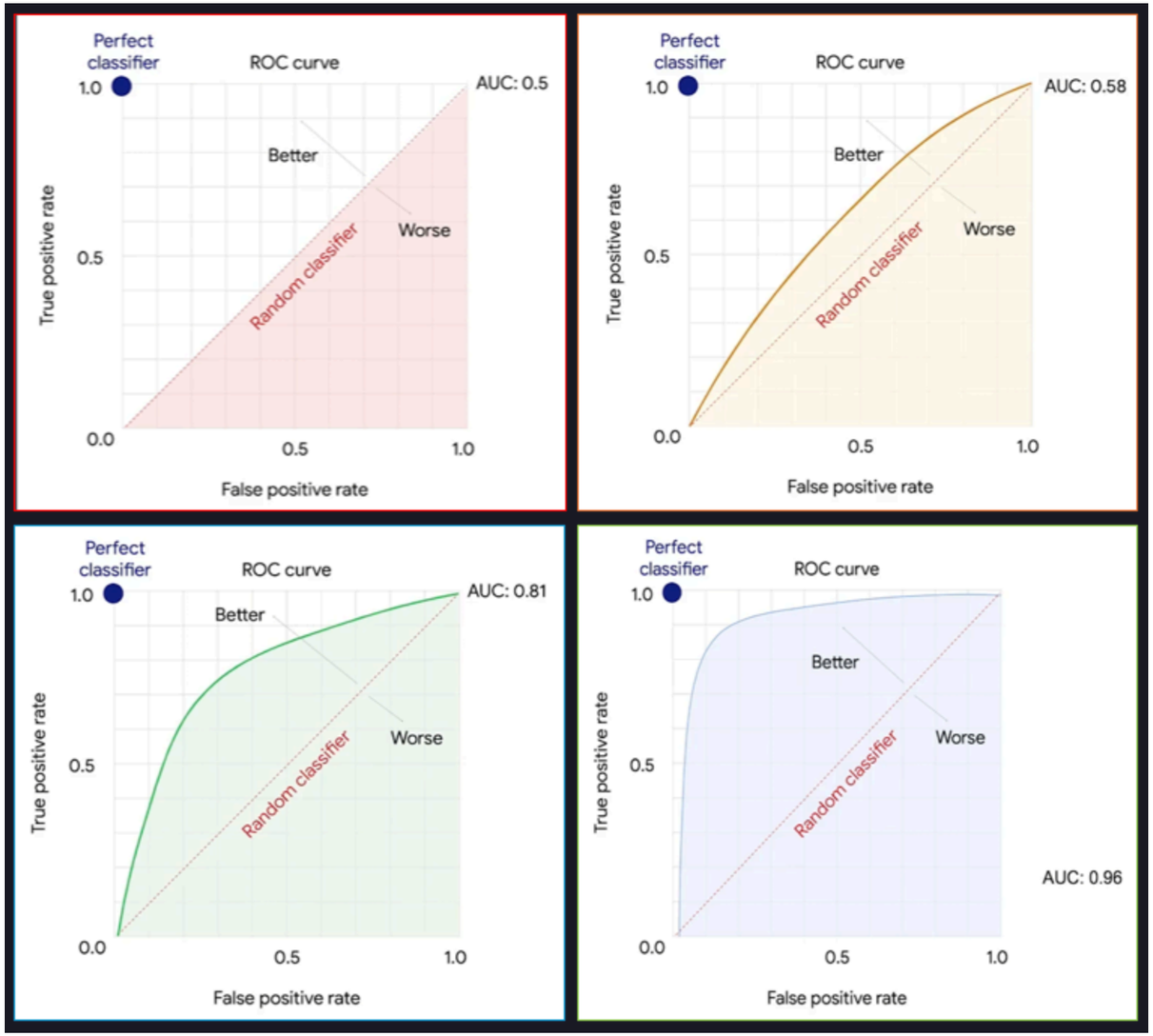
The proportion of data points that were correctly categorized

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}}$$

- Using `sklearn.metrics` as `metrics`
  - `metrics.accuracy_score(y_test, y_pred)`

ROC Curves AUC

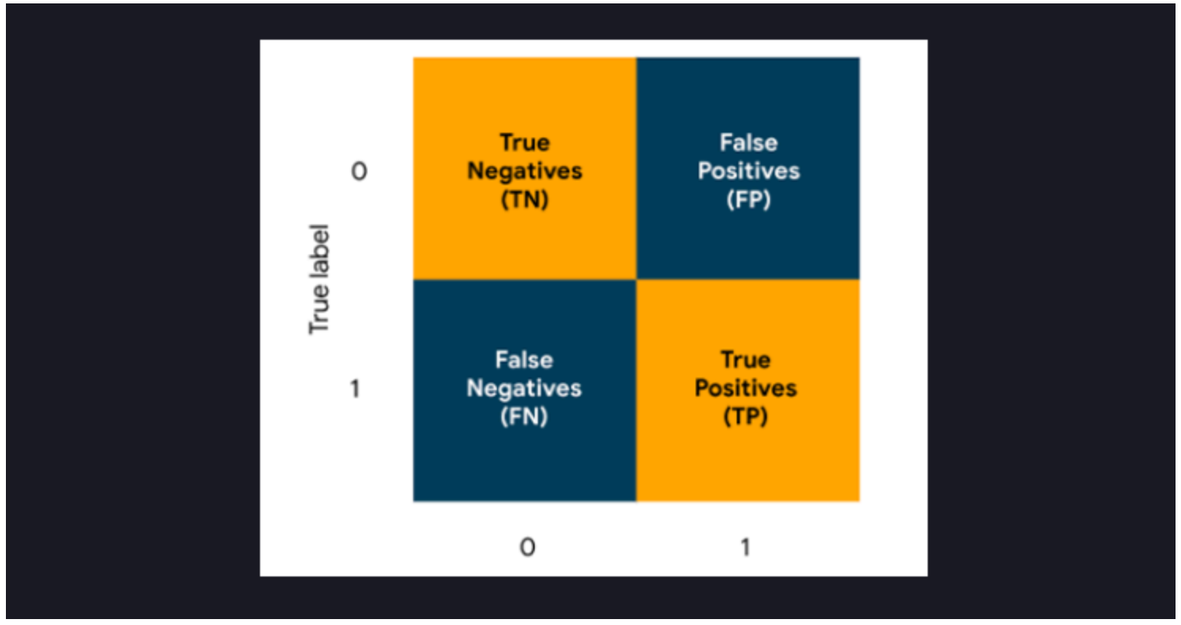
Receiver operating characteristic curve , also known as ROC curve is a graphical plot that illustrates the *performance of a binary classifier model* (although it can be generalized to multiple classes) *at varying threshold values*.



We can use an ROC curve in AUC to examine how the true positive and false positive rate change together at every threshold. Data professionals may use ROC curves and AUC when comparing classification models, so we'll explore them later in this program.

Elements of a Confusion Matrix

A confusion matrix helps summarize the performance of a `classifier`. The components of a confusion matrix are used to compute metrics for evaluating logistic regression classifiers.



The 4 key elements of a confusion matrix (for a binary classification model)

True Negatives:

The count of observations that a classifier correctly predicted as False (0)

True Positives:

The count of observations that a classifier correctly predicted as True (1)

False Positives:

The count of observations that a classifier incorrectly predicted as True (1)

False Negatives:

The count of observations that a classifier incorrectly predicted as False (0)

More theory on...

We already touched this in the *Course 4. Power of Statistics* , you can further explore the theory for this.

- Conditional Probability and Bayes Theorem

Understanding better ROC curves

What is the main purpose?

ROC curves helps visualizing the `performance` of a logistic regression classifier.

To visualize the `performance` of a classifier at different classification `thresholds`, you can graph an ROC curve.

- In the context of `binary classification`, a `classification threshold` is a *cutoff for differentiating the positive class from the negative class.*

Two Key concepts of ROC curves : TPR and FPR

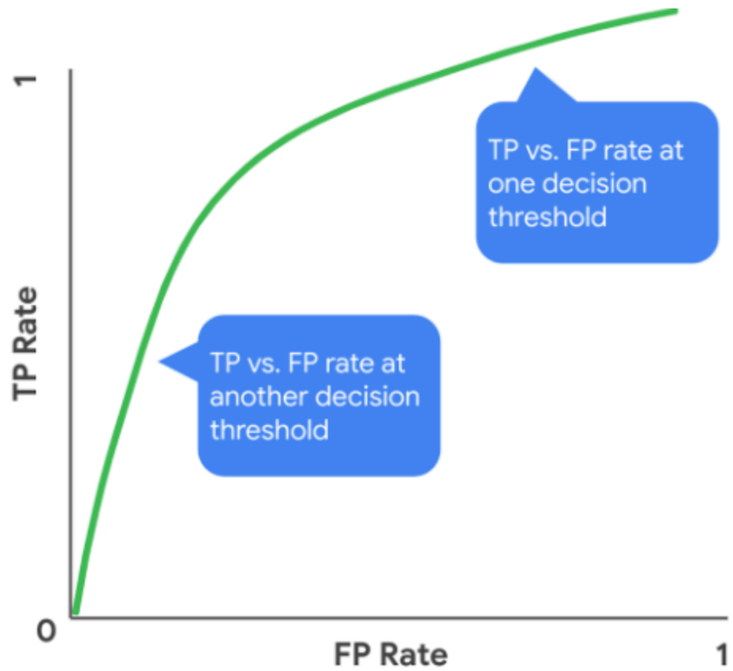
- `TPR` (**True Positive Rate**) : Is the equivalent to `Recall`. The formula for True Positive rate is as follows:

$$True\ Positive\ Rate = \frac{True\ Positives}{True\ Positives\ +\ False\ Negatives}$$

- `FPR` (**False Positive Rate**) : The ratio between the False positives and the total count of observations that should be predicted as False. The formula is as follows:

$$False\ Positive\ Rate = \frac{False\ Positives}{False\ Positives\ +\ True\ Negatives}$$

For each point on the curve, the x and y coordinates represent the False Positive Rate and the True Positive Rate respectively at the corresponding threshold.



In the ROC curve for an ideal model, there would exist a threshold at which the True Positive Rate is high and the False Positive Rate is low.

- The more that the ROC curve hugs the top left corner of the plot, *the better the model does at classifying the data.*

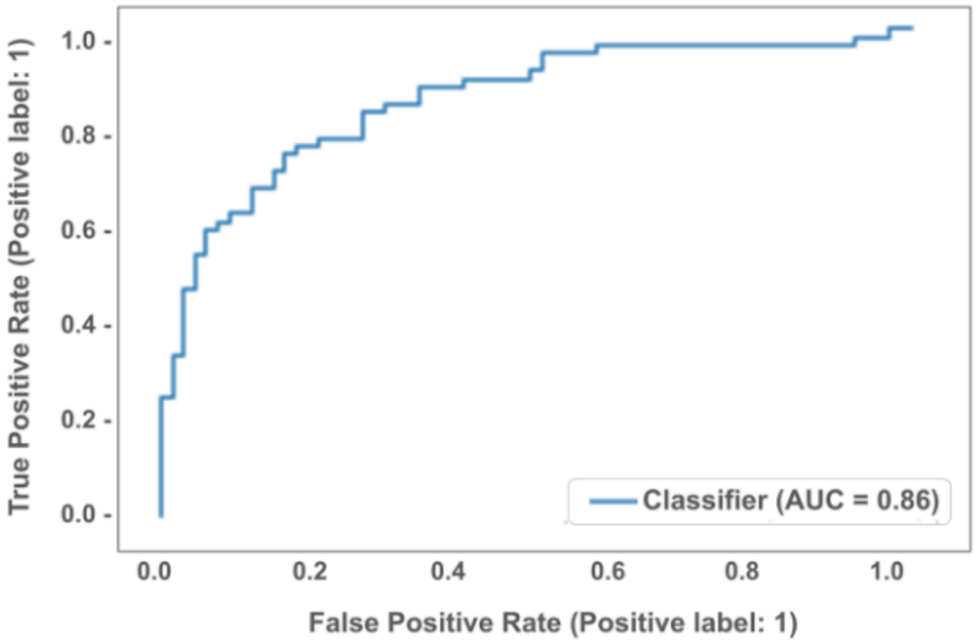
Compute a ROC curve in Python:

Imports Modules

```
1. import matplotlib.pyplot as plt
2. from sklearn.metrics import RocCurveDisplay
```

Then use the following code to plot the ROC curve.

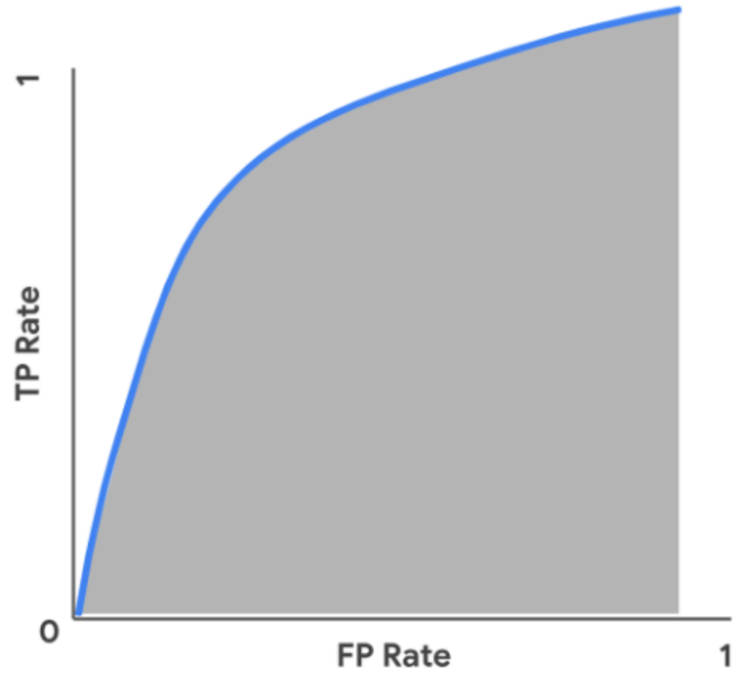
```
3. RocCurveDisplay.from_predictions(y_test, y_pred)
4. plt.show()
```



What is the AUC of a ROC curve?

AUC stands for `Area under the curve`. AUC provides an aggregate measure of performance across all possible classification thresholds. AUC *ranges in value from 0.0 to 1.0. A model whose predictions are 100% wrong has an AUC of 0.0*, and a *model whose predictions are 100% correct has an AUC of 1.0*.

- *Note that* : An AUC *smaller than 0.5 indicates* that the *model performs worse than a random classifier* (i.e. a classifier that randomly assigns each example to True or False), and an AUC larger than 0.5 indicates that the model performs better than a random classifier.



Compute a AUC Plot

To compute AUC in Python, you can use the `roc_auc_score()` function from the metrics module. The function takes in true values and predicted values as arguments and returns the accuracy score. You can use the following code to compute the AUC.

- `metrics.roc_auc_score(y_test,y_pred)`

Resources for more information

- precision_score: Documentation on the precision_score() function
- recall_score: Documentation on the recall_score() function
- accuracy_score: Documentation on the accuracy_score() function
- RocCurveDisplay.from_predictions: Documentation on the RocCurveDisplay.from_predictions() function
- roc_auc_score: Documentation on the roc_auc_score() function

---

Interpret the results of a logistic regression

#Advanced_Data_Analytics #Regression/Logistic/metrics/accuracy #Regression/Logistic/metrics/recall #Regression/Logistic/metrics/precision #Programming/Python/sklearn #Regression/Logistic/Classifier #Regression/Logistic/Confusion_Matrix

Lets understand the interpretation from its origin

Lets being with the formula of our logistic regression.

$$logit(p) = log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1$$

Going back to our example of vertical acceleration and knowing if a person is lying down or not.

What do the β_0 *coefficient* mean in the equation?

A *One-Unit INCREASE* in `vertical acceleration` is associated with a β_1 *INCREASE* in the `logit of p`.

- But if we *exponentiate* the `log odds`, then we can determine how much the odds change as a percentage based on changes in vertical acceleration.

$$e^{\beta}$$

e^{β} is how many times the `odds of p` will increase or decrease for every one-unit increase in vertical acceleration.

Lets go back to our exercise example

We know our coefficient is *negative 0.118* from the output (Check the Python exercise)

```
clf.coef_ = -0.1177
```

Get coefficients and visualize model

We can use the `coef_` and `intercept_` attributes of the `clf` object to get the coefficient and intercept of our model.

```
#Show coef and intercept from the clf model
print(clf.coef_)  , print(clf.intercept_)
```

```
[[-0.1177471]]
[6.10180958]

(None, None)
```

So, based on what we've found, our model has:

- an intercept or β_0 of 6.10 and a β_1 of -0.12.

So if we compute :

$$\beta_1 = -0.118$$

$$e^{-0.118} = 0.89$$

KEY Interpretation

For every *One-Unit INCREASE* in the vertical acceleration, we except that the odds of the person lying down *DECREASES* by 11%.

What if our β_1 is positive?

Lets explore the following scenario:

$$\beta_1 = 0.25$$

Then

$$e^{0.25} = 1.28$$

KEY Interpretation

For every *One-Unit INCREASE* in the vertical acceleration, we except that the odds of the person lying down *INCREASES* by 28%.

What if we multiple independent variables (X_1 , X_2 , X_n)

For every *One-Unit INCREASE* in X , *holding other variables X constant*, we explect the odds of Y being 1 to increase by 28%.

What is Important for us as data professionals to communicate about a logistic regression?

Moving beyond the `Beta coefficient`, it is always helpful to state the `P-value` and `confidence interval` to give additional information about how likely the result is just by chance.

Keep in mind that...

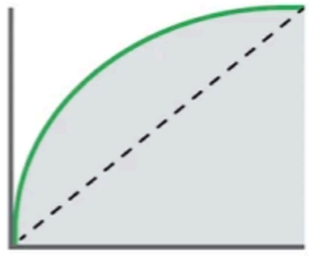
- *Scikit-learn does NOT have a built-in way of getting p-values or confidence intervals*, but `statsmodels does`. This is a great example of how different tools and packages can give you different information.
- As a data practitioner, you have to choose your tools depending on your priorities. When presenting the results, it can be helpful to include a confusion matrix and some statistics on precision, recall accuracy and or ROC, AUC.
- However, depending on the situation, you might want to include all of them or focus on a subset of metrics. Consider that certain industries or organizations may have preferred metrics given how they manage their modeling process. It's important to check with your team about this as you get started as a data analytics professional.

Confusion Matrix

Accuracy

Precision

Recall



ROC/AUC

Understand that your criteria is -key-

Take for example, the case of *detecting spam texts or unsolicited messages* sent to many recipients. Only a small fraction of text messages that you receive are probably spam.

- But *if your goal is to accurately classify spam text messages* so you don't give away private information or click on a bad link, then you're really *only focusing on how well you can detect spam messages.*

In fact, in this case, `accuracy` *is **NOT** a good metric*.

- Let's say *only three percent of text messages are spam*. That means 97 percent are text message you want to receive from friends and family or automated messages from service providers. So if a model predicts that 100 percent of messages are not spam, that model would have a 97 percent accuracy rate, which seems great. But it's not that good in this context because the model will not have detected any spam at all, even though three percent of messages are in fact spam. In the case of spam messages, recall is probably a more meaningful metric.

`Recall` will tell us the proportion of spam messages that the model was actually *able to detect.*

`Precision`, on the other hand, would tell us the *proportion of data points labeled spam, that were actually spam.*

- `Precision` essentially *measures the quality of our spam labeling.* There are other metrics that you can explore as well, such as *AIC and BIC*, which can help determine how good a model is while factoring in how complex the model is.

Resources for more information

- LogisticRegression: Documentation for implementing Logistic Regression models using sklearn and accessing intercept and coefficients from a model.