ML - 9 Logistic Regression (24 Aug 22)

Wednesday, August 10, 2022 1:23 PM

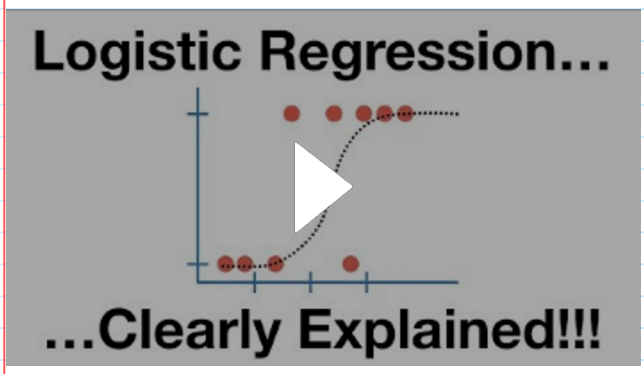
Pre-Class

Logistic Regression

Introduction

The linear regression fits a line and is used to predict a continuous value. For example, we estimate the price of a house using linear regression. However, the logistic regression fits a **s-shaped** (sigmoid) function and predicts whether something is true or false. It is mainly used in **binary classification problems**. In the video below, you will see the logic behind logistic regression and examine a binary classification problem related to mice being obese or not obese.

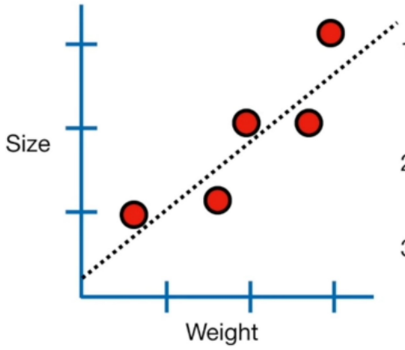
Watch:
StatQuest: Logistic Regression



Interview Q&A

> Q: Why can't linear regression be used in place of logistic regression for binary classification?
> A: In linear regression, the output is continuous. In case of binary classification, an output of a continuous value does not make sense. For binary classification problems, linear regression may predict values that can go beyond 0 and 1. If we want the output in the form of probabilities, which can be mapped to two different classes, then its range should be restricted to 0 and 1. As the logistic regression model can output probabilities with logistic/sigmoid function, it is preferred over linear regression.

Linear Regression:



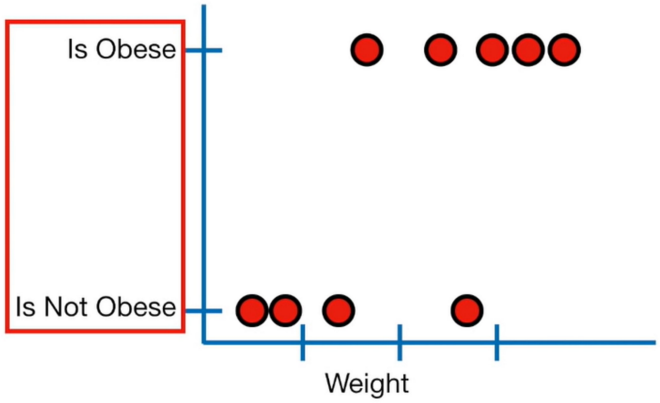
…and with that line, we could do a lot of things:

1) Calculate R^2 and determine if ***weight*** and ***size*** are correlated. Large values imply a large effect.

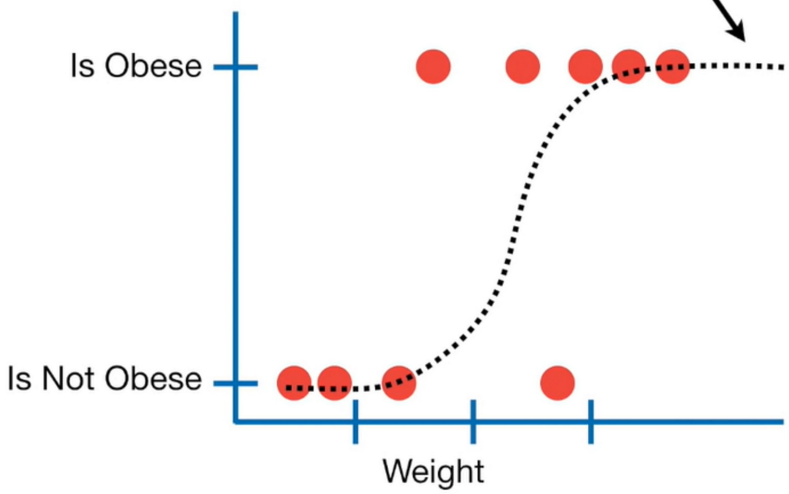
2) Calculate a p-value to determine if the R^2 value is statistically significant.

3) Use the line to predict ***size*** given ***weight***.

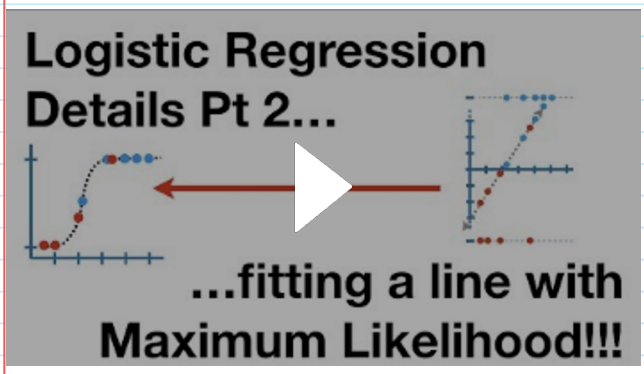
Logistic regression predicts whether something is ***True*** or ***False***, instead of predicting something continuous like ***size***.



…also, instead of fitting a line to the data, logistic regression fits an "S" shaped "logistic function".

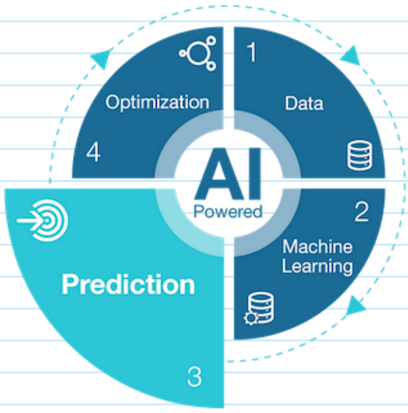


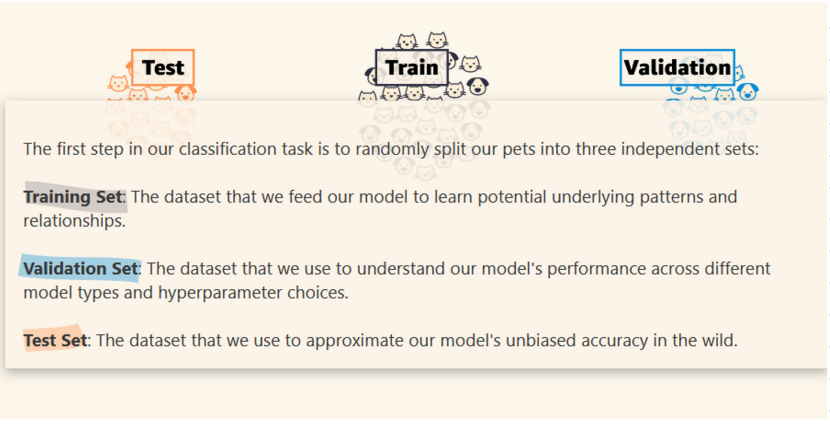
Watch:
[Logistic Regression Details Pt 2: Maximum Likelihood]



Evaluating Performance-Classification Error Metrics

In a binary classification model, our prediction can only get two results: **correct** or **incorrect**. We use classification error metrics to measure and evaluate the performance of our predictions.





The major classification metrics:

- **Accuracy**: The proportion of correct predictions (both true positives and true negatives) among the total number of cases examined.
- **Recall**: Also known as sensitivity. It is the fraction of relevant instances that were retrieved.
- **Precision**: Also called positive predictive value. It is the fraction of relevant instances among the retrieved instances
- **F1-Score**: The F1 score is the harmonic mean of the precision and recall.

★ Watch:
Never Forget Again! // Precision vs Recall with a Clear Example of Precision and Recall

Confusion Matrix:

The classification metrics mentioned above (accuracy, precision, etc.) are generated from the confusion matrix. A confusion matrix is a frequently used table to describe the performance of a classification model on a series of test data where real values are known.

💡 **Tips:**

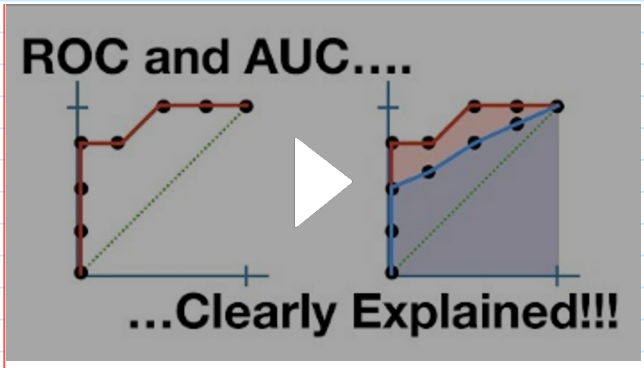
- Keep in mind that understanding the confusion matrix is critical in the performance measurement of classification models.

★ Watch:
The NO CONFUSION matrix! // What is the confusion matrix? // Confusion matrix visual explanation

Receiver operating characteristic (ROC) curve:

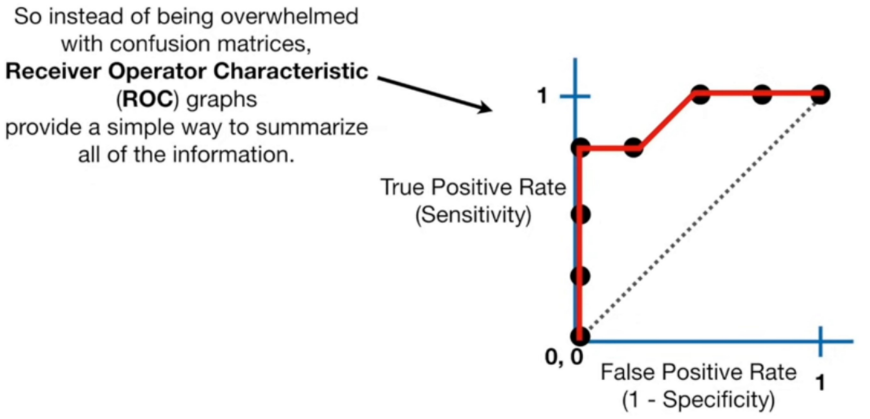
This is a graphical plot that illustrates the performance of a binary classifier as its discriminant threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold values.

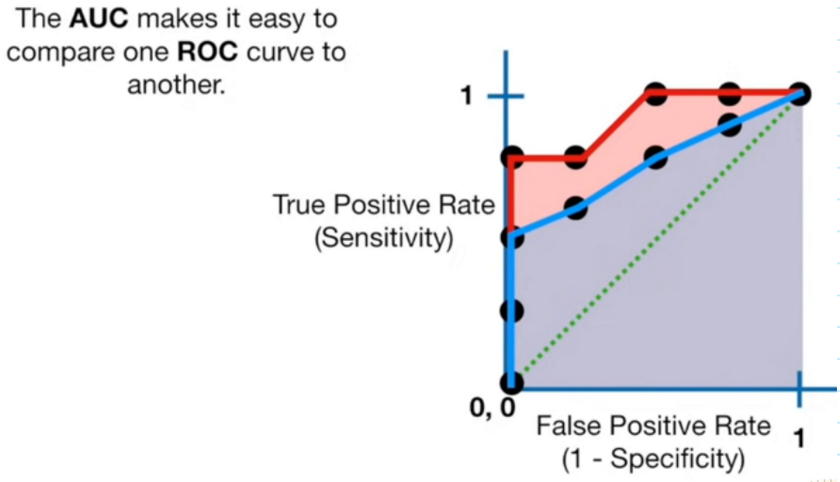
★ Watch:
ROC and AUC, Clearly Explained!



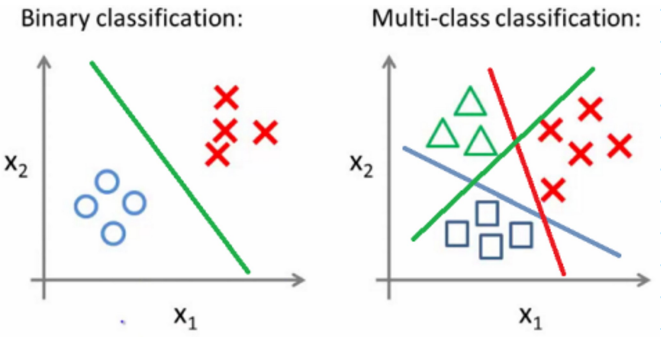
Interview Q&A

> Q: Why is accuracy not a good measure for classification problems?
> A: Accuracy is not a good measure for classification problems because it gives equal importance to both false positives and false negatives. However, this may not be the case in most business problems. For example, in case of cancer prediction, declaring cancer as benign is more serious than wrongly informing the patient that he is suffering from cancer. Accuracy gives equal importance to both cases and cannot differentiate between them.

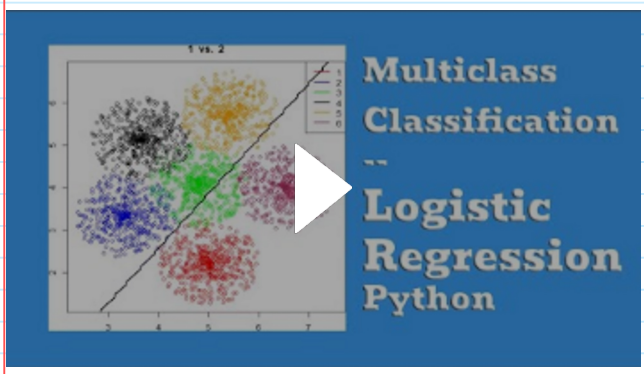




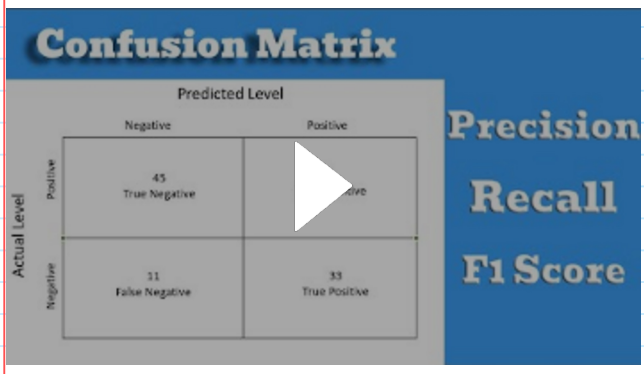
Logistic Regression - Multi-class Classification with Python



★ Watch:
[Multiclass Classification With Logistic regression in Python | Sklearn]



[Precision, Recall and F1 Score for Multiclass Classification - sklearn | Python]



★ Watch:

Interview Q&A

//

In-Class

Stratify = aynı dağılımda bölmek

Official documentation:
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html?highlight=train_test_split

stratify array-like, default=None
If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

Some classification problems can exhibit a large imbalance in the distribution of the target classes: for instance there could be several times more negative samples than positive samples. In such cases it is recommended to use stratified sampling as implemented in StratifiedKFold and StratifiedShuffleSplit to ensure that relative class frequencies is approximately preserved in each train and validation fold.

This stratify parameter makes a split so that the proportion of values in the sample produced will be the same as the proportion of values provided to parameter stratify.
For example, if variable y is a binary categorical variable with values 0 and 1 and there are 25% of zeros and 75% of ones, stratify=y will make sure that your random split has 25% of 0's and 75% of 1's.

Michael Gd-Tech.Coord.:
stratify target değerlerinin iki tarafa aynı oranda dağıtılmasını sağlıyor. yani target value(0,1) oranları mesela yüzde 30 (0) yüzde 70(1) olsun. train test split random olarak bölme yaptığı için birler bi tarafa sıfırlar bi tarafa toplanmasın diye stratify kullanılır. sonuçta bölme işlemi nihayetlenince train set yüzde 30 (0) yüzde 70(1) target value sahip olur aynı şekilde test set yüzde 30 (0) yüzde 70(1) target value sahip olur. target value unbalanced olduğunda kullanılır. diğer durumlarda da kullanılmasının sakıncası yoktur.

****

Soru:
burada neden pred_proba[: , 1] yaptık yani sadece 1. sütunu aldık? 0. sütunda de ne vardı pred_proba nın

Michael Gd-Tech.Coord.:
- model her zaman default 1 leri hedef olarak görür ve hesaplamayı ona göre yapar. o yüzden 1. sütuna bakıyoruz
- Sınıfın 0 olma olasılığı

****
Soru:
0 ve 1 için metrikler çok farklı çıktı neden?

Orion - ML Instructor:
- 0 sınıfından fazla olduğundan ondan daha çok öğrenmiş
- imbalance data setlerinde modelin yanlılığı fazla olan classa yönelir bu yüzdende scoreları genelde iyi çıklar.

****
Orion - ML Instructor:
parametre = modelin bulduğu
hyper parametre = bizim ayarladığınız

****

Limited-memory BFGS (**L-BFGS** or LM-BFGS) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) using a limited amount of computer memory. It is a popular algorithm for parameter estimation in machine learning.

****

Orion - ML Instructor:

```
fp_rate, tp_rate, thresholds = roc_curve(y_train, y_pred_proba[:,1])
```

bu şekilde yazmak gerekli.

****

John - ML Instructor:
modele verdiğimiz yeni verinin feature sırası modeldeki ile aynı olmali ve scaled olmali

Only Label / target may have categorical value. Not true for features, be careful!

****
ROC is used in binary classification scenarios, not in multi-class.

ROC, AUC, precision, and recall visually explained – https://paulvanderlaken.com/2019/08/16/roc-auc-precision-and-recall-visually-explained/

****

Orion - ML Instructor:

- modeliniz hangi sample range inde eğitilmiş ise o rangede en iyi çalışır
- linearde alpha büyüdükçe regülarizasyon artıyordu logisticte ise küçüldükçe artıyor

★ **In-Class Slides:**
ML - S8/S9 - (Logistic Regression and Classification Error Metrics)
https://app.peardeck.com/review/student/tbytsetdb

★ **In-Class Record:**
ML - S9 - 24/08/22 John (Logistic Regression with Phyton)
https://lms.clarusway.com/mod/page/view.php?id=24040

//

Post-Class

Scikit-Learn: https://scikit-learn.org/stable/index.html
Yellowbrick: https://www.scikit-yb.org/en/latest/quickstart.html

Orion - ML Instructor:
Drop first

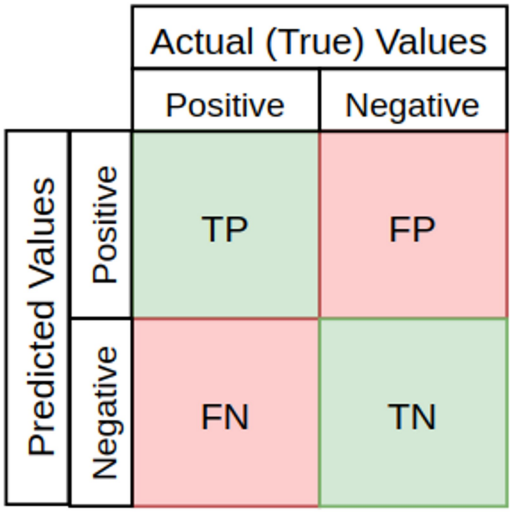
Although the category you choose to drop won't affect the model's performance, it can have a significant impact on the interpretability of the model.

If you plan to use the coefficients in your model to make accurate inferences about:

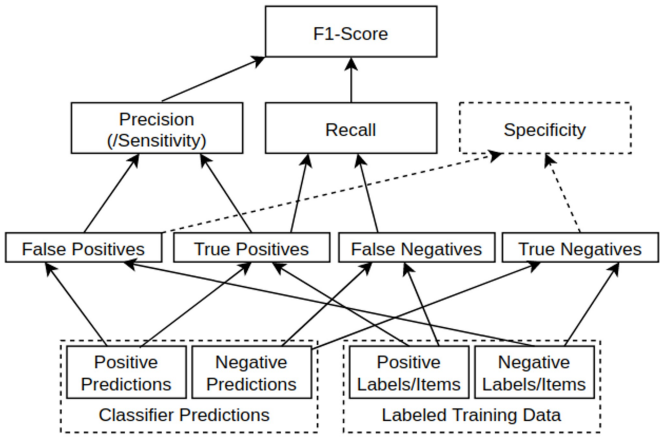
- impact on the target per predictor unit,
- or the magnitude of impact each predictor has in relation to others,

then you should carefully consider which categorical values you drop from your model.

A linear regression model's coefficients are interpreted in the context of a baseline model. For continuous variables, the baseline uses a reference point of 0. But **for categorical variables, whichever column is dropped becomes the reference point, which has a significant impact on how coefficients are interpreted.**



Confusion Matrix. Image by Author.



Hierarchy of Metrics from raw measurements / labeled data to F1-Score. Image by Author.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

ML-9 Recap

ML-9 Logistic Regression

Phases/steps build on top of each other. The output of previous step is the input of next step.

Phases

1) Data preparing / EDA
2) Divide data → features
   → label (target)

```
X = df.drop("target", axis=1)
y = df["target"]
```

3) Train / Test split:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, ...)
```

4) Scale (X) (we don't scale "y", since it's single column)

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- we fit_trans only X_train
- we transform X_test just

5) Select model (LinearRegression(), LogisticRegression() ...)

6) Train ("Fit") model with train data

```
log_model.fit(X_train_scaled, y_train)
```

7) Make predictions (by using X_test_scaled and "trained" model)

```
y_pred = log_model.predict(X_test_scaled)
```

8) Get probability estimates with "predict_proba"

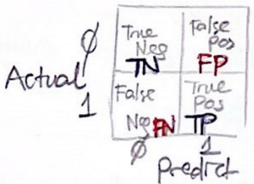
```
y_pred_proba = log_model.predict_proba(X_test_scaled)
```

9) Compare test data with prediction and probability.

```
test_data = pd.concat([X_test, y_test])
test_data['pred'] = y_pred
test_data['pred_proba'] = y_pred_proba[:, 1]
```

→ prob. estimates of outcomes (1)
- if we put 0 we get prob. of outcomes "0"

10) Get metrics and evaluate
- confusion matrix
- classification report



11) Cross validate with train data
- get average of scores.

12) Grid Search CV → to find best hyper-parameters

```
define parameters
grid_model.fit(X_train_scaled, y_train)
grid_model.best_params_
```

best "C", "penalty", "solver", "class_weight"

13) Roc and AUC review
14) Find best threshold
15) Final Model and deployment
train with all X data