MLE and Regularization in LoR

Maximum Likelihood Estimation (MLE)

- Used to find the best fit line.
- MLE is a method to estimate the parameters of a statistical model by finding the values that **maximize the probability** (likelihood) of observing the given data.
- In simple terms, it chooses the parameter values that make the observed data **most probable**.
- MLE is maximized, while Loss Function is minimized.

MLE in Machine Learning

- MLE is widely used only in Parametric models.
  - Not applicable to KNN, DT, etc.
- **Logistic Regression** → Finds the best parameters for classifying data.
- **Naive Bayes Classifier** → Estimates probabilities.
- **Neural Networks** → Optimized using cross-entropy loss (derived from MLE).

Assumptions of Logistic Regression

1. Binary Logistic Regression requires the dependent variable to be binary
2. Independence of observations
3. Linearity of independent variables and log odds (***logit***)

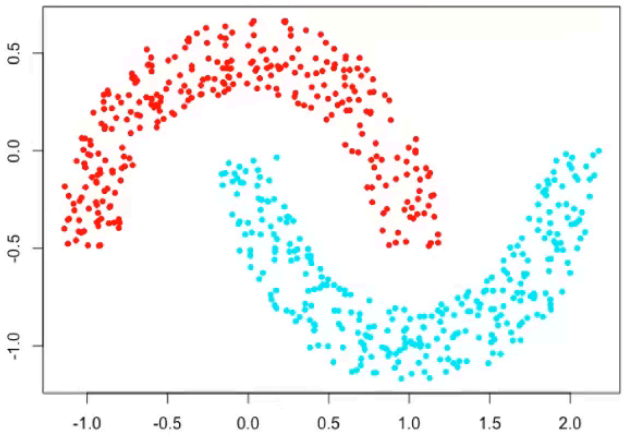
$$\log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

1. Absence of multicollinearity
2. Large sample size

Note that violating these assumptions doesn't mean you can't or shouldn't use logistic regression, but it may impact the validity of the results and you should proceed with caution.

Polynomial Features

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=3,include_bias=False)
```



- Draws non-linear decision boundaries.
- Other algorithms perform better

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=3,include_bias=False)

X_trf = poly.fit_transform(X)

clf1 = LogisticRegression()
np.mean(cross_val_score(clf1,X_trf,y,scoring='accuracy',cv=10))
```

Regularization in Logistic Regression

- Regularization used to prevent overfitting.
- In the context of linear models like linear regression and logistic regression, regularization works by adding a penalty term to the loss function that the model tries to minimize.

L1 regularization (Lasso Regression)

L2 regularization (Ridge Regression)

`penalty='l2'` ***(default)***

- `None`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

`C` ***=1.0 (default)***

- Small λ→ Underfitting
- Big λ (`C`) → Overfitting

`l1_ratio` ***=None (Only Applicable in*** `Elasticnet` ***)***

- 1 → L1

- 0.5 → 50:50
- 0→ L2