Logistic Regression

Logistic Regression

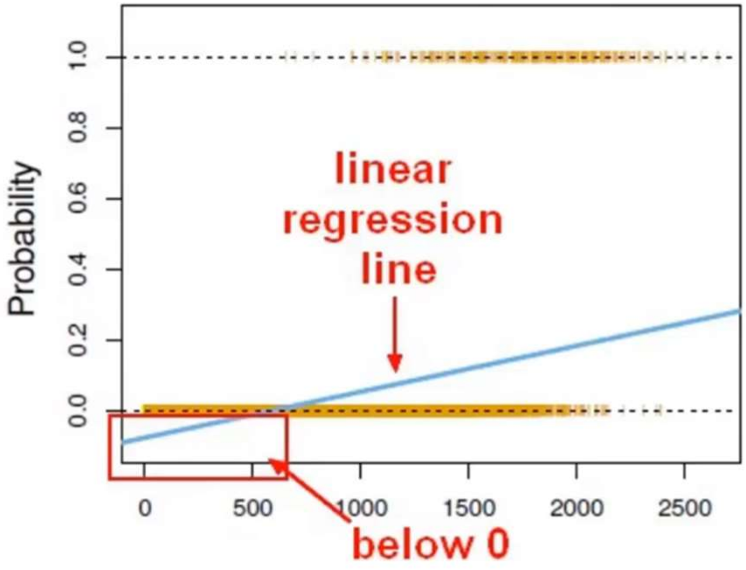
- We need to learn Logistic Regression as a method for Classification
- Some examples of classification problems:
  - Spam Vs Regular mails
  - Loan Default (Yes/No)
  - Disease Diagnosis (Present/Not Present)
- Above were all examples of **Binary Classification**

Logistic Regression

- Linear Regression tries to predict a continuous value
- Although the name may be confusing at first, logistic regression allows us to solve classification problems, where we are trying to predict discrete categories
- The convention for binary classification is to have only two classes represented by 0 and 1

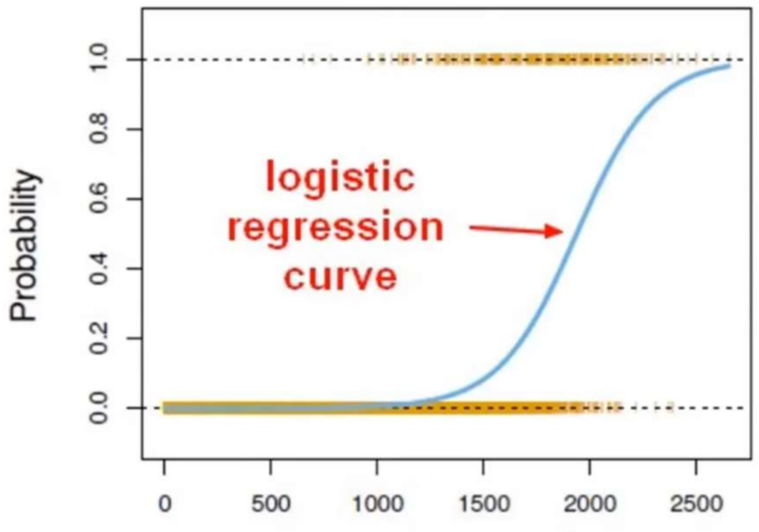
Background

- We cannot use normal linear regression model on binary groups. It won't lead to a good fit:



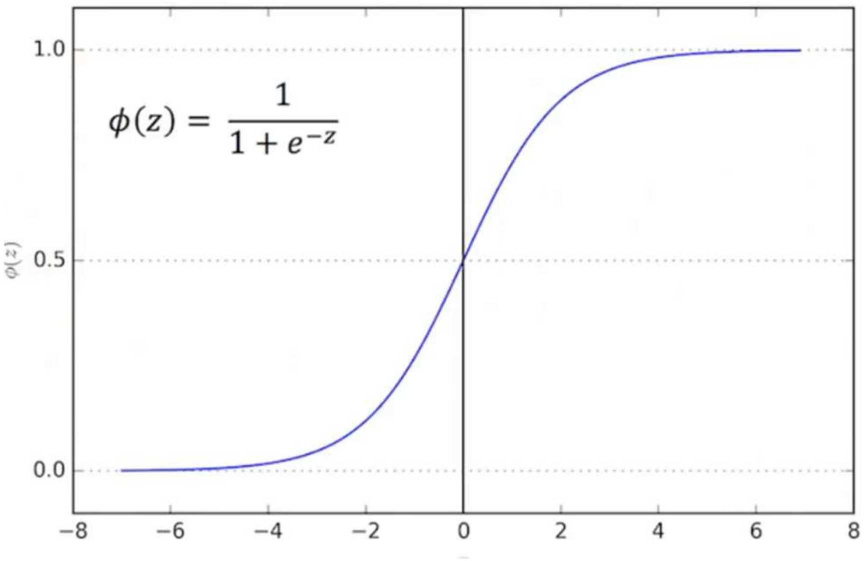
Background

- Instead we can transform our linear regression to a logistic regression curve:



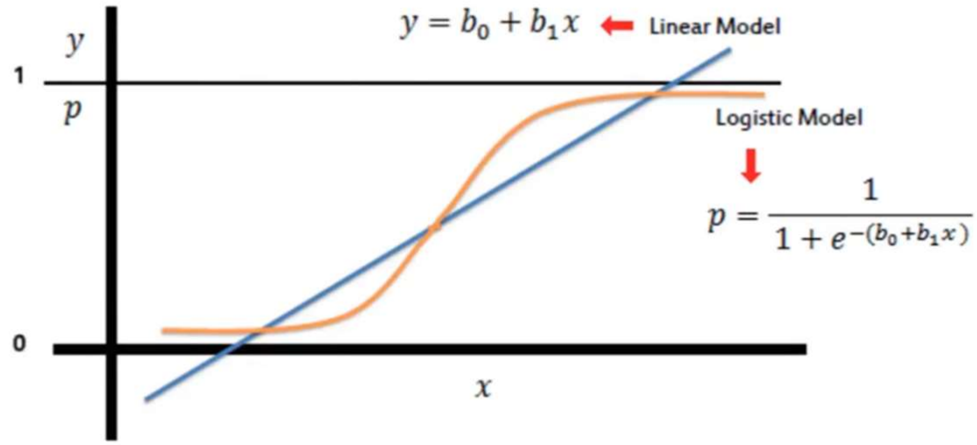
Sigmoid Function

- The Sigmoid (aka Logistic) Function takes in any value and outputs it to be between 0 and 1



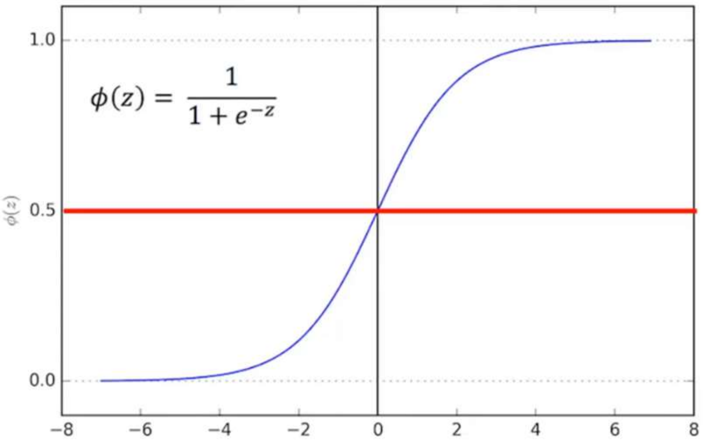
Sigmoid Function

- This means that we can take our Linear Regression Solution and place it into the Sigmoid Function



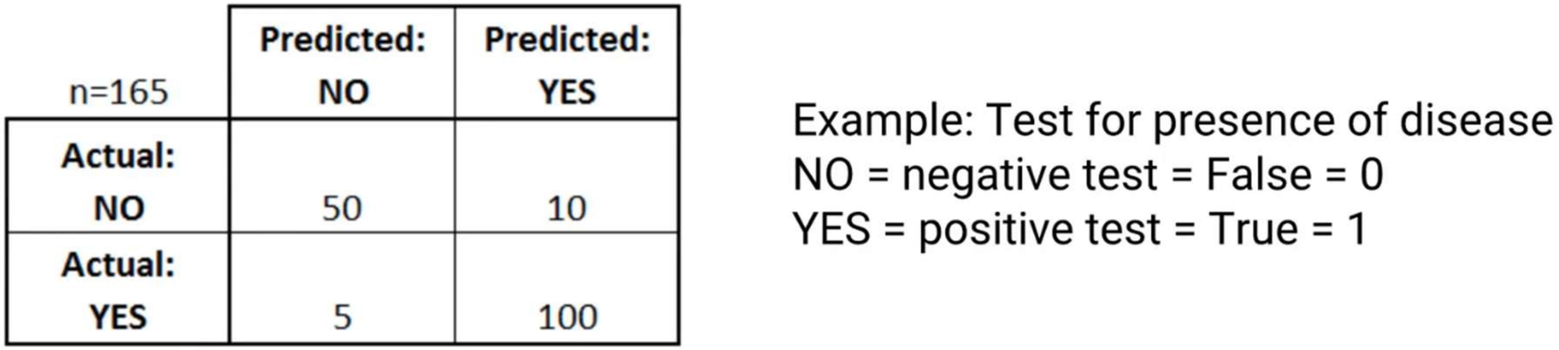
Sigmoid Function

- We can set a cut-off point at 0.5, anything below it results in class 0 and anything above is class 1
- We use the logistic function to output a value ranging from 0 to 1. Based out of this probability we assign a class



Model Evaluation

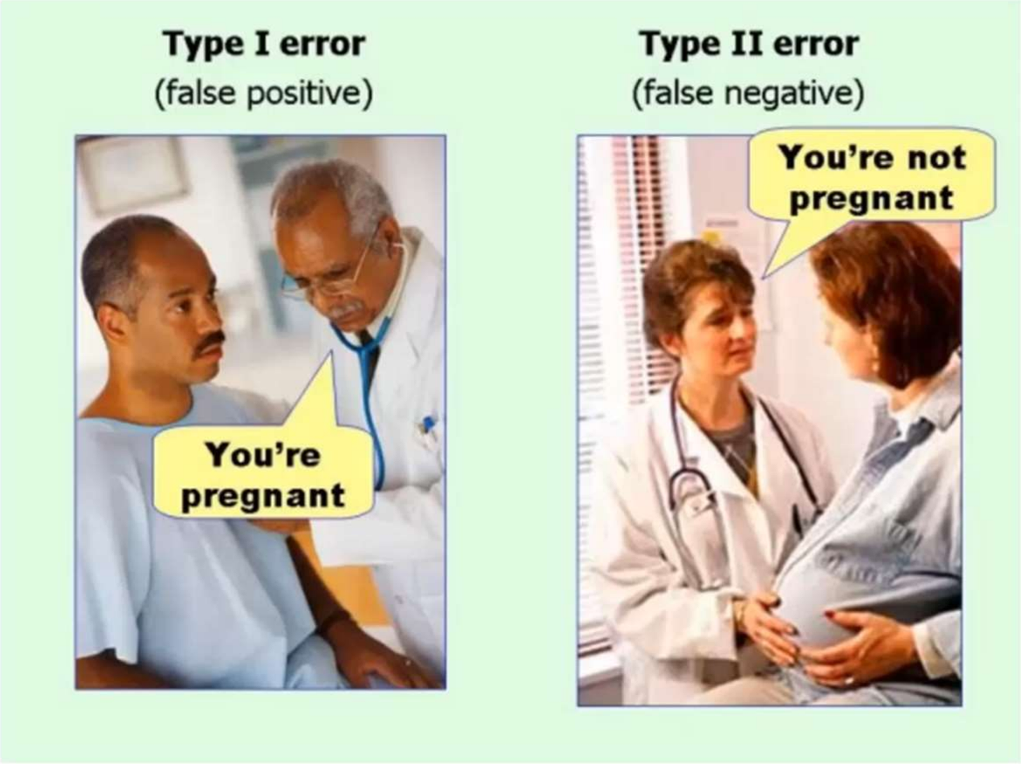
- After we train the logistic regression model on some training data, we can evaluate the model’s performance on some test data
- We can use a confusion matrix to evaluate classification models
- Example: Imagine testing for a disease



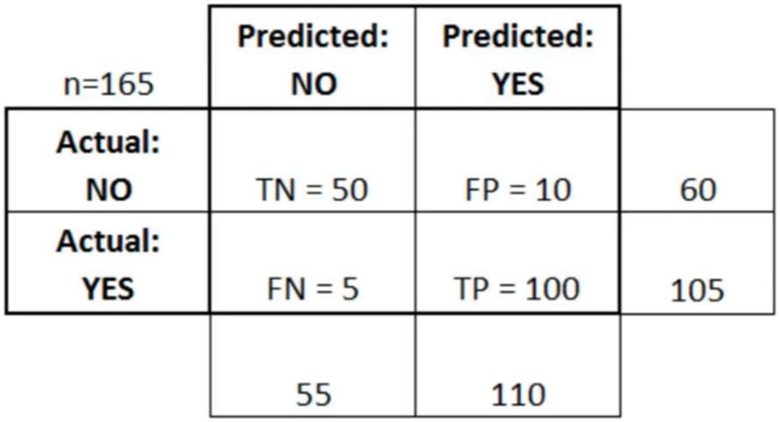
n=165	**Predicted:** **NO**	**Predicted:** **YES**
Actual: **NO**	50	10
Actual: **YES**	5	100

Example: Test for presence of disease
NO = negative test = False = 0
YES = positive test = True = 1

Confusion Matrix



Confusion Matrix



n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Basic Terminology:

- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

Accuracy:

- Overall, how often is it **correct**?
- (TP + TN) / total = 150/165 = 0.91

Misclassification Rate (Error Rate):

- Overall, how often is it **wrong**?
- (FP + FN) / total = 15/165 = 0.09

Classification Threshold

- It is tempting to assume that the classification threshold should always be 0.5, but thresholds are problem-dependent, and are therefore values that you must tune
- A logistic regression model that returns 0.9995 for a particular email message is predicting that it is very likely to be spam. Conversely, another email message with a prediction score of 0.0003 on that same logistic regression model is very likely not spam.
- However, what about an email message with a prediction score of 0.6? In order to map a logistic regression value to a binary category, you must define a **classification threshold** (also called the **decision threshold**).
- A value above that threshold indicates "spam"; a value below indicates "not spam."

Accuracy

- Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\mathbf{accuracy=Number\ of\ correct\ predictions/Total\ number\ of\ predictions}$$

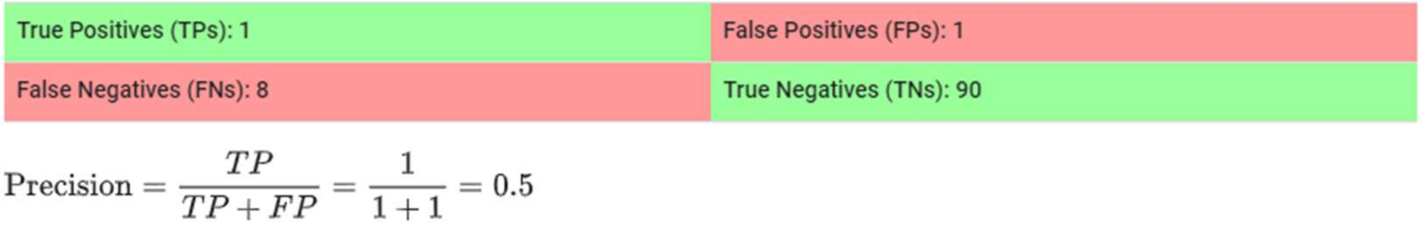
- For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\mathbf{accuracy=TP+TN/TP+TN+FP+FN}$$

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

Precision

- Precision tries to answer the question: **What proportion of positive identifications was actually correct?**

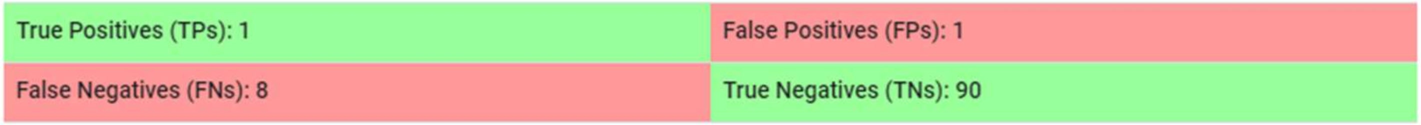


True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{1}{1+1} = 0.5$$

Recall

- Recall tries to answer the question: **What proportion of actual positives was identified correctly?**

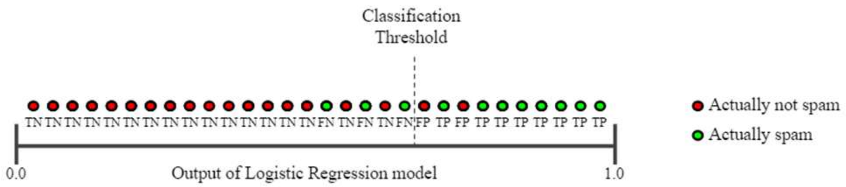


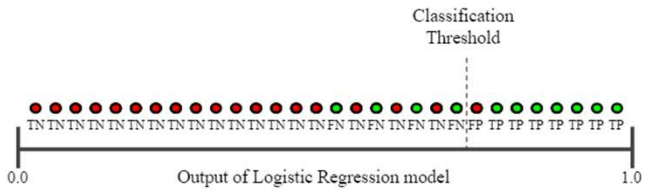
True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{1}{1+8} = 0.11$$

Precision Vs Recall

- To fully evaluate the effectiveness of a model, you must examine **both** precision and recall. Unfortunately, precision and recall are often in tension. That is, improving precision typically reduces recall and vice versa.


Classification
Threshold
TN TN TN TN TN TN TN TN TN TN TN TN TN TN TN FN TN FN TN FN FP TP FP TP TP TP TP TP TP TP
0.0
Output of Logistic Regression model
1.0
Actually not spam
Actually spam


Classification
Threshold
TN TN TN TN TN TN TN TN TN TN TN TN TN TN TN FN TN FN TN FN TN FN FP TP TP TP TP TP TP TP
0.0
Output of Logistic Regression model
1.0

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{8}{8+2} = 0.8$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{8}{8+3} = 0.73$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{7}{7+1} = 0.88$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{7}{7+4} = 0.64$$

F1-Score

- In statistical analysis of binary classification, the F-score or F-measure is a measure of a test's accuracy.
- It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive.
- Precision is also known as **positive predictive value**, and recall is also known as **sensitivity** in diagnostic binary classification.

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

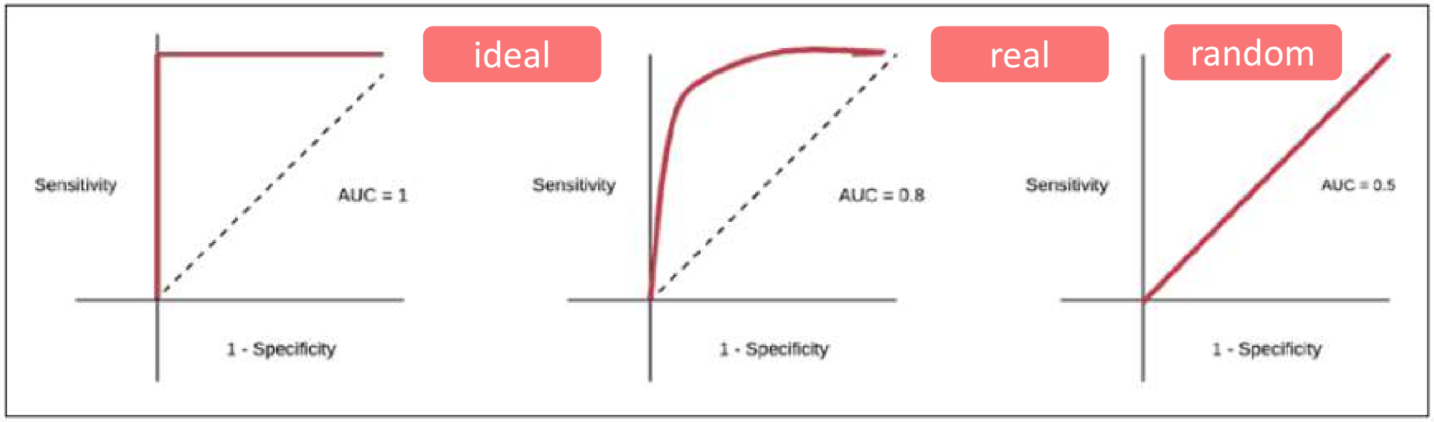
ROC

- ROC stands for Receiver Operating Characteristic.
- It’s is a type of curve. We draw the ROC curve to visualize the performance of the binary classifier.
- The ROC curve is a 2-D curve. It’s x axis represents the False Positive Rate (FPR) and its y axis represents the True Positive Rate (TPR).
- TPR is also known as **sensitivity**, and FPR is also known as **specificity** (SPC). You can refer to the following equations for FPR and TPR.

TPR = True Positive / Number of positive samples = TP / P
FPR = False Positive / Number of negative samples = FP / N = 1 — SPC

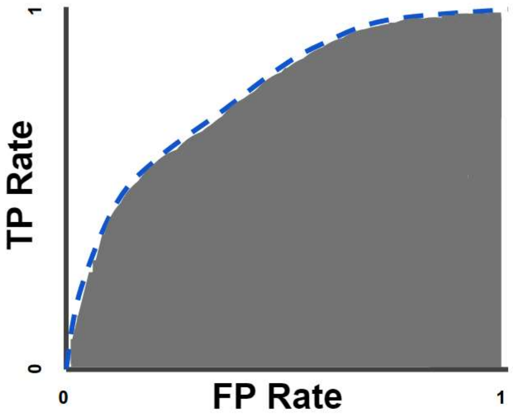
ROC Curve

- An ROC curve plots TPR vs. FPR at different classification thresholds.
- Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.



AUC

- AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1)
- A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0
- AUC is desirable because:
  - AUC is **scale-invariant**. It measures how well predictions are ranked, rather than their absolute values.
  - AUC is **classification-threshold-invariant**. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.



Prediction Bias

- Logistic regression predictions should be unbiased. That is:

  "average of predictions" should ≈ "average of observations“

- **Prediction bias** is a quantity that measures how far apart those two averages are. That is:

  prediction bias=average of predictions−average of labels in data set

- A significant nonzero prediction bias tells you there is a bug somewhere in your model, as it indicates that the model is wrong about how frequently positive labels occur.

Prediction Bias

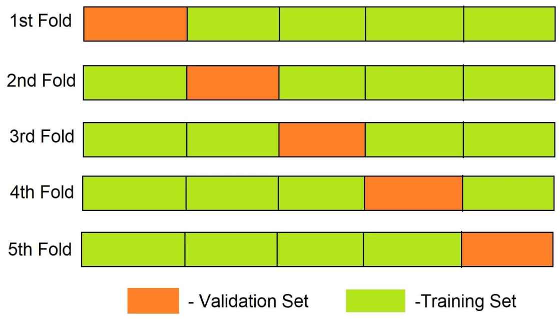
- Possible root causes of prediction bias are:
  - Incomplete feature set
  - Noisy data set
  - Buggy pipeline
  - Biased training sample
  - Overly strong regularization

K-Fold Cross Validation

- When evaluating different settings (“hyperparameters”) for estimators, there is still a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally.
- The knowledge about the test set can “leak” into the model and evaluation metrics no longer report on generalization performance.
- Validation set: training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.
- By partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.
- A solution to this problem is called Cross Validation
- It can be considered as a resampling procedure used to evaluate machine learning models on a limited data sample.

K-Fold Cross Validation

- The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation.
- The general procedure is as follows:
  - Shuffle the dataset randomly.
  - Split the dataset into k groups
  - For each unique group:
    - Take the group as a hold out or test data set
    - Take the remaining groups as a training data set
    - Fit a model on the training set and evaluate it on the test set
    - Retain the evaluation score and discard the model
  - Summarize the skill of the model using the sample of model evaluation scores



K-Fold Cross Validation

- This process results in k estimates of the test error, MSE1, MSE2, . . . , MSEk. The k-fold CV estimate, also called **Test Error** is computed by averaging these values:

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i.$$

K-Fold Cross Validation

- A poorly chosen value for k may result in a mis-representative idea of the skill of the model, such as a score with a high variance (that may change a lot based on the data used to fit the model), or a high bias, (such as an overestimate of the skill of the model).
- Three common tactics for choosing a value for k are as follows:
  - Representative: The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.
  - k=10: The value for k is fixed to 10, a value that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.
  - k=n: The value for k is fixed to n, where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.

Hyperparameters

- A model hyperparameter is a characteristic of a model that is external to the model and whose value cannot be estimated from data.
- The value of the hyperparameter has to be set before the learning process begins.
- For example, alpha in Lasso, c in Support Vector Machines, k in k-Nearest Neighbors, the number of hidden layers in Neural Networks.

Grid Search

- Grid-search is used to find the optimal hyperparameters of a model which results in the most ‘accurate’ predictions.