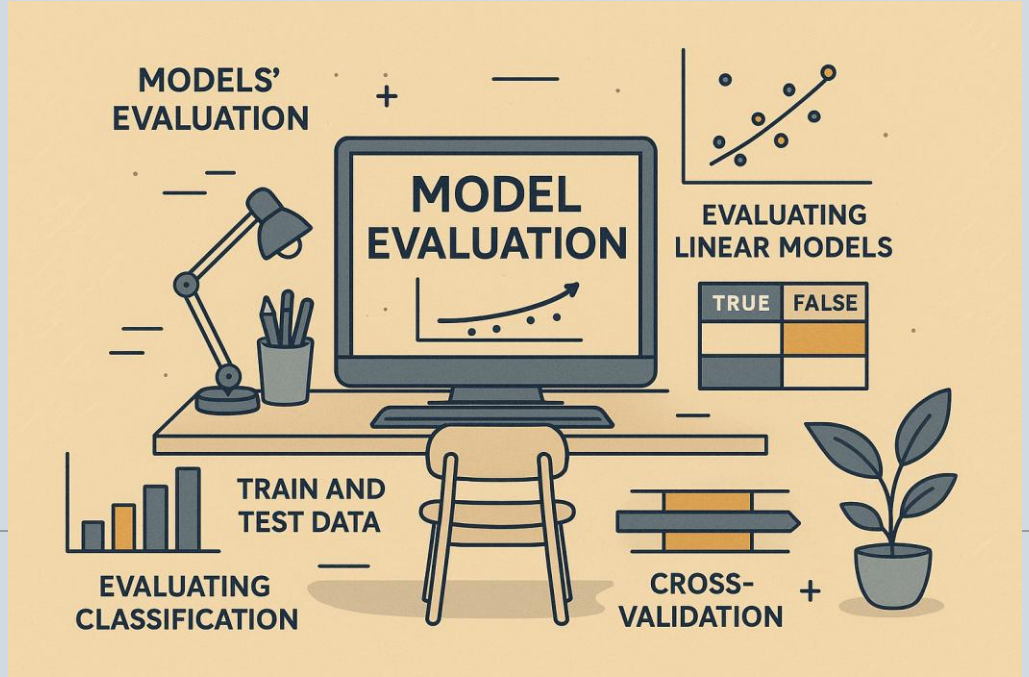
Model Evaluation



DR. ZVI BEN AMI

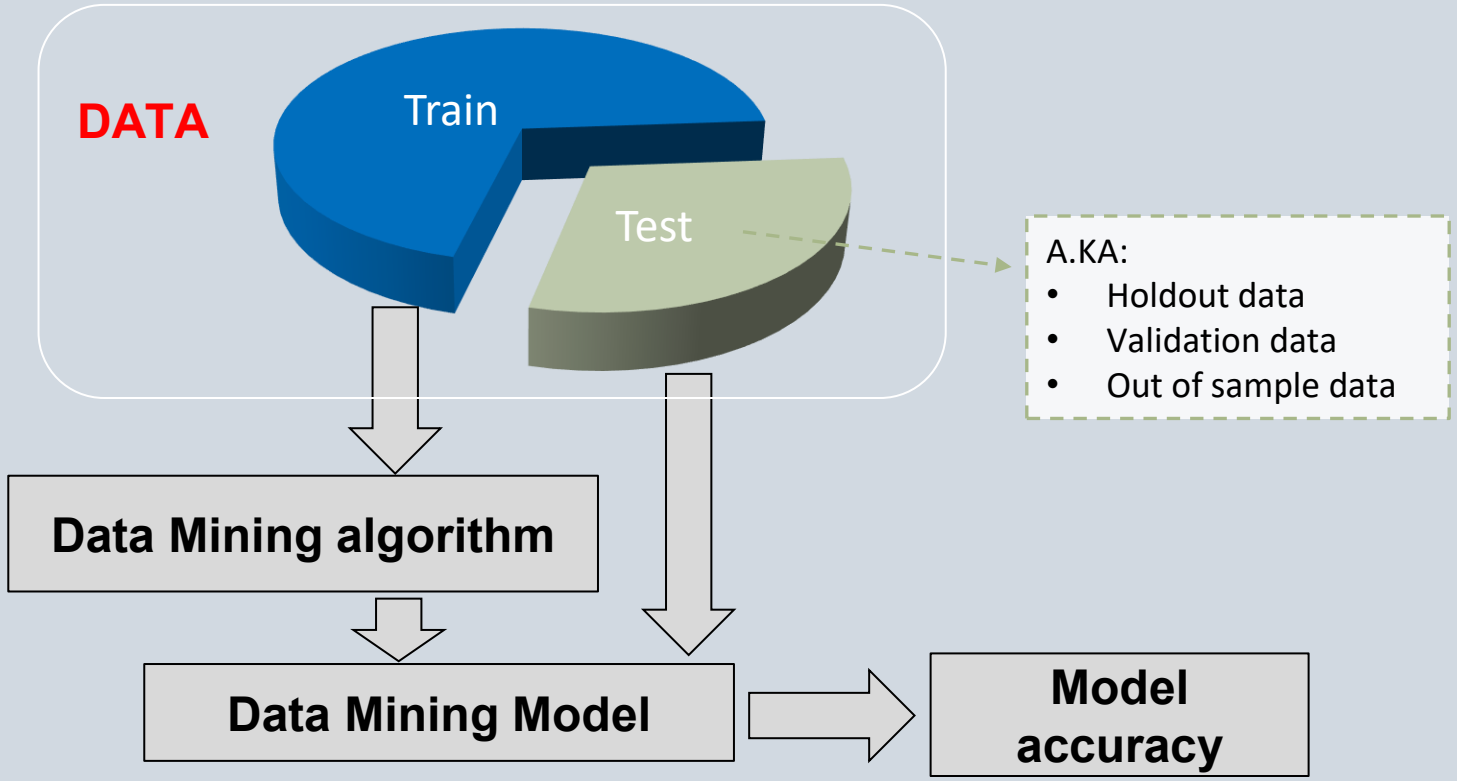
Agenda

- Models’ evaluation
- Train and test data
- Evaluating linear models
- Evaluating classification models
- Cross-validation
- Model overfitting and underfitting
- Model Bias and Variance

Model Evaluation

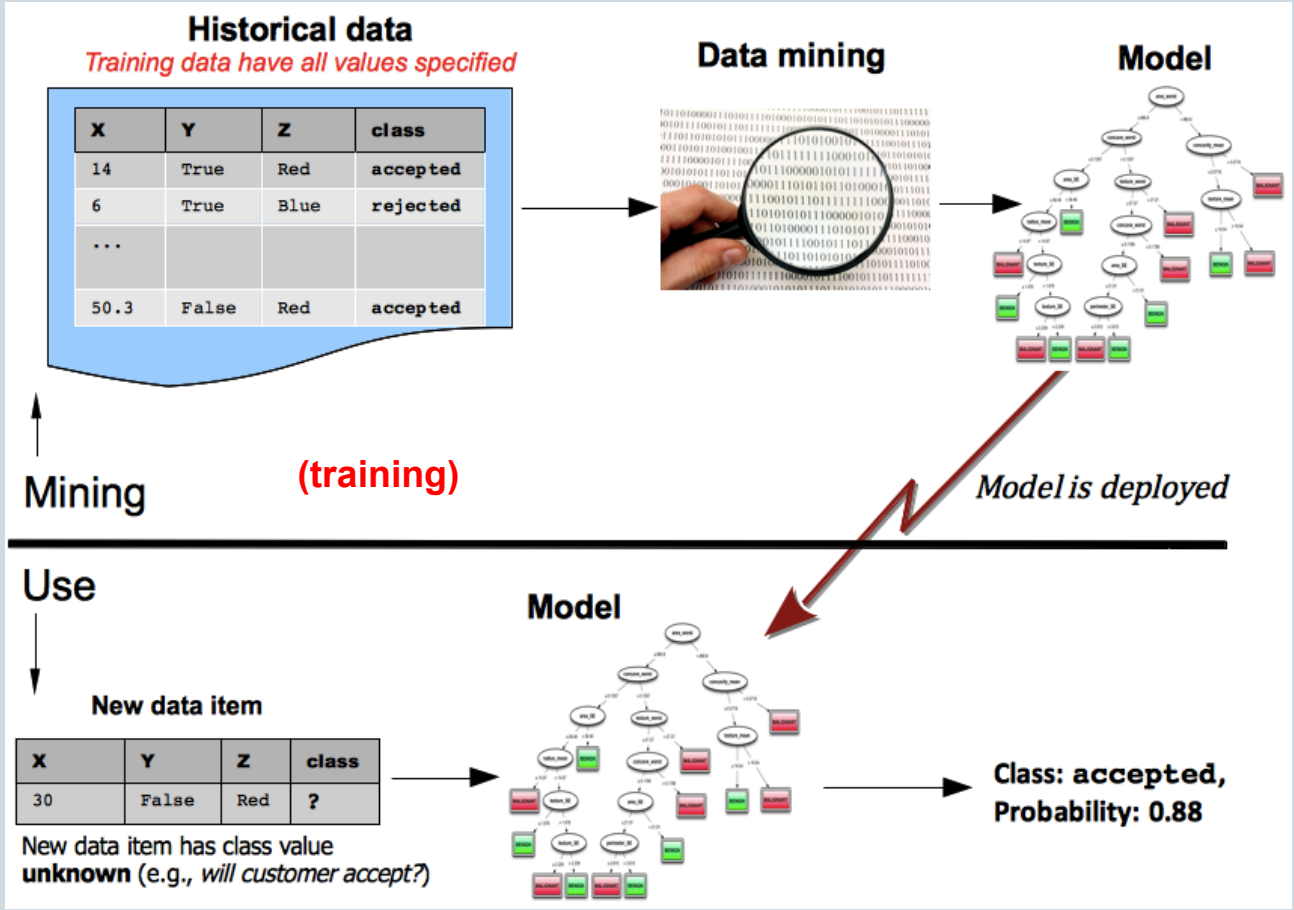
- Evaluation on training data provides no assessment of how well the model generalizes to unseen cases.
- Since we want to evaluate models performance before going to production we need to create a “Lab test” for model performance.
- We can “Hold out” some data for which we know the value of the target variable, but which will not be used to build the model.
- Evaluate model performance using the holdout data.

Training and Testing Data



Assumption: The distribution of a training set is identical to the distribution of a test set (including future unseen examples).

Data Mining Process



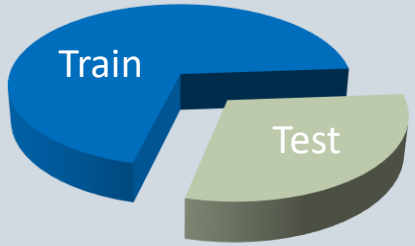


Image from “Data Science for Business”, Provost, Fawcett, 2013

A Typical Fitting Graph

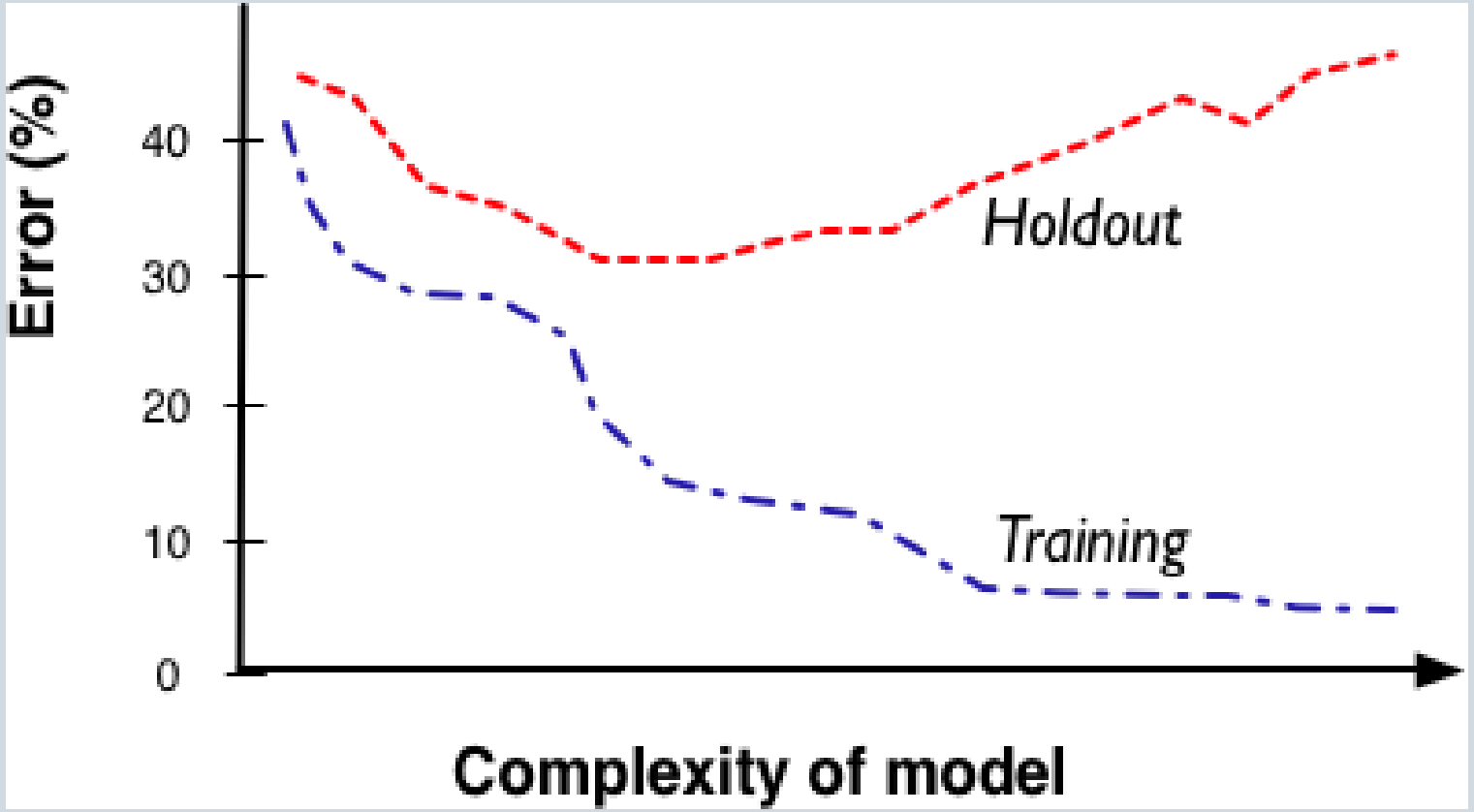


Image from “Data Science for Business”, Provost, Fawcett, 2013

Linear Regression Evaluation

- Linear regression can be used for prediction and estimation
- Prediction: given some input, provide best guess for the output
- In linear regression, output depends linearly on each input parameter
- y_i – actual value for instance i
- $\hat{y}_i$ – predicted value for instance i
- n – number of instances
- Mean Squared Error (MSE) = $\frac{1}{n}\sum_i^n (y_i - \hat{y}_i)^2$
- Root Mean Squared Error (RMSE) = $\sqrt{\frac{1}{n}\sum_i^n (y_i - \hat{y}_i)^2}$
- Mean Absolute Error (MAE) = $\frac{1}{n}\sum_i^n |y_i - \hat{y}_i|$
- Mean Absolute Percentile Error (MAPE) = $\frac{1}{n}\sum_i^n \left|\frac{y_i - \hat{y}_i}{y_i}\right|$

Model_evaluation.ipynb

Regression Metrics

Regression metrics

See the Regression metrics section of the user guide for further details.

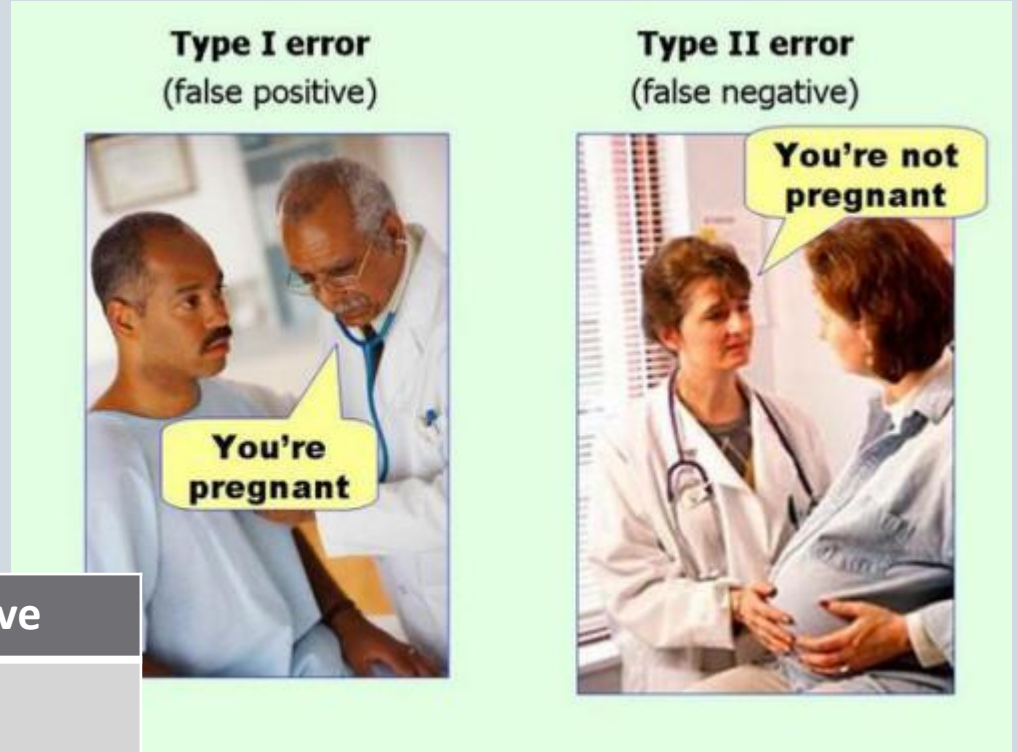
`metrics.explained_variance_score`(y_true, ...)	Explained variance regression score function.
`metrics.max_error`(y_true, y_pred)	The max_error metric calculates the maximum residual error.
`metrics.mean_absolute_error`(y_true, y_pred, *)	Mean absolute error regression loss.
`metrics.mean_squared_error`(y_true, y_pred, *)	Mean squared error regression loss.
`metrics.mean_squared_log_error`(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
`metrics.median_absolute_error`(y_true, y_pred, *)	Median absolute error regression loss.
`metrics.mean_absolute_percentage_error`(...)	Mean absolute percentage error (MAPE) regression loss.
`metrics.r2_score`(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.
`metrics.mean_poisson_deviance`(y_true, y_pred, *)	Mean Poisson deviance regression loss.
`metrics.mean_gamma_deviance`(y_true, y_pred, *)	Mean Gamma deviance regression loss.
`metrics.mean_tweedie_deviance`(y_true, y_pred, *)	Mean Tweedie deviance regression loss.
`metrics.d2_tweedie_score`(y_true, y_pred, *)	D^2 regression score function, percentage of Tweedie deviance explained.
`metrics.mean_pinball_loss`(y_true, y_pred, *)	Pinball loss for quantile regression.

https://scikit-learn.org/stable/api/sklearn.metrics.html#module-sklearn.metrics
https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

Evaluating Hypotheses: Classification Error Rate

Type 1 error / False Positive:
- When you see things that are not there

Type 2 error / False Negative:
- When you don't see things that are there



Classification Model Performance measures

	Classified as positive	Classified as negative
Success (correct prediction)	**True Positive (TP)** (hit!)	**True Negative (TN)** (correct rejection)!
Error (wrong prediction)	**False Positive (FP)** (false alarm: type 1 error)	**False Negative (FN)** (miss: type 2 error)

Confusion Matrix

Confusion matrix provides insights into (mis)classification of any classifier.

		Predicted	
		Positive	Negative
Actual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

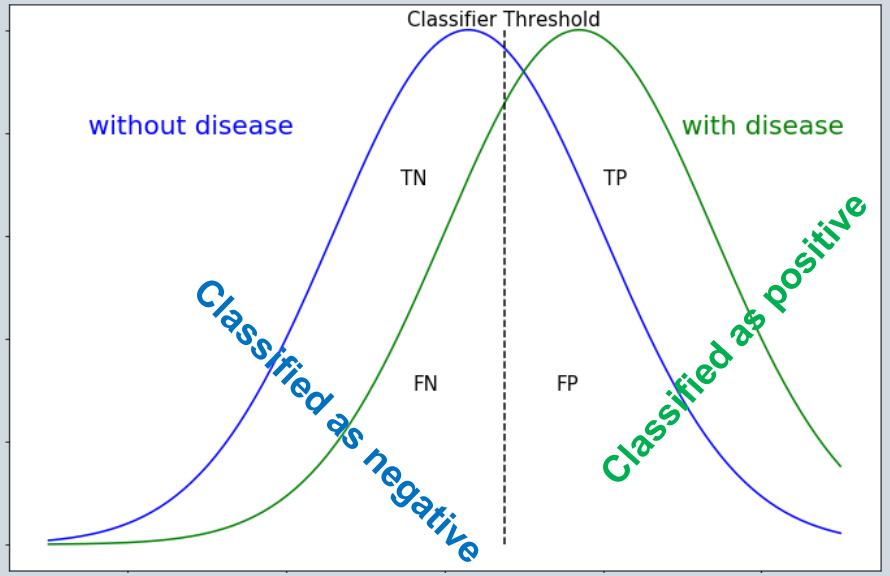
Evaluating Hypotheses: Classification Error Rate

Type 1 error / False Positive:

- When you see things that are not there

Type 2 error / False Negative:

- When you don't see things that are there



Classification Model Performance measures

(Confusion matrix)

Model Predictions

Ground Truth

	Classified as positive	Classified as negative
Success (correct prediction)	**True Positive (TP)** (hit!)	**True Negative (TN)** (correct rejection)!
Error (wrong prediction)	**False Positive (FP)** (false alarm: type 1 error)	**False Negative (FN)** (miss: type 2 error)

Classification error rate (accuracy score):

$$e = \frac{TN+TP}{TP+TN+FN+FP}$$

Classification Errors: Recall And Precision

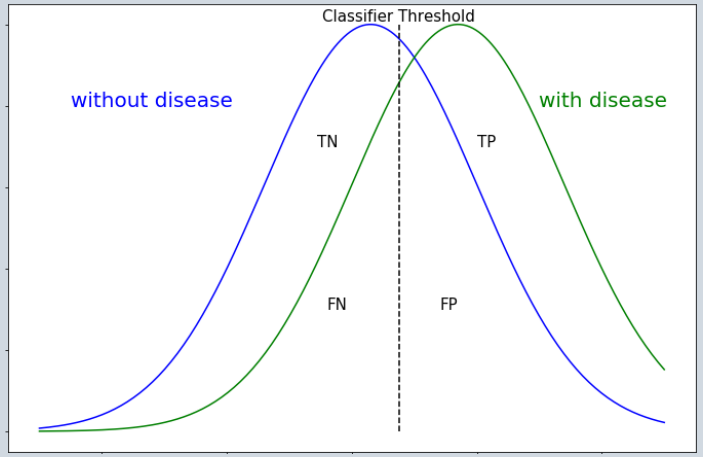
Classification Model Performance measures

(Confusion matrix)

Model Predictions

Ground Truth

	Classified as positive	Classified as negative
Success (correct prediction)	**True Positive (TP)** (hit!)	**True Negative (TN)** (correct rejection)!
Error (wrong prediction)	**False Positive (FP)** (false alarm: type 1 error)	**False Negative (FN)** (miss: type 2 error)



$Precision = \frac{TP}{TP+FP}$ - probability that the disease is present if test result is positive

$Recall = \frac{TP}{TP+FN}$ - probability that a test result will be positive when the disease is present. Also called Sensitivity

complementary

$Specificity = \frac{TN}{FP+TN}$ - probability that a test result will be negative when the disease is not present

Precision and Recall

Precision and recall are widely used to evaluate data mining models .

Precision is the fraction of correctly predicted classes

- i.e., precision is the number of correct results out of the total number of results returned.

$$\boldsymbol{Precision} = \frac{\#of\ correct\ preditions\ (for\ a\ given\ class)}{\#of\ predictions\ (for\ a\ given\ class)} \qquad Precision = \frac{TP}{TP + FP}$$

Recall is the fraction of classes which are relevant to the query and were successfully retrieved.

- i.e. recall is the number of correctly extracted results out of the total number of results that should have been extracted.

$$\boldsymbol{Recall} = \frac{\#of\ correct\ preditions\ (for\ a\ given\ class)}{\#of\ \text{items}(for\ a\ given\ class)} \qquad Recall = \frac{TP}{TP + FN}$$

Precision and Recall Tradeoff

There is a tradeoff between precision and recall

- Increase in precision can lower overall recall, while an increase in recall can lower precision.
- Easy to achieve 100% precision or 100% recall.
- Harder to obtain high precision and recall simultaneously.

Classification Errors: Recall and Precision

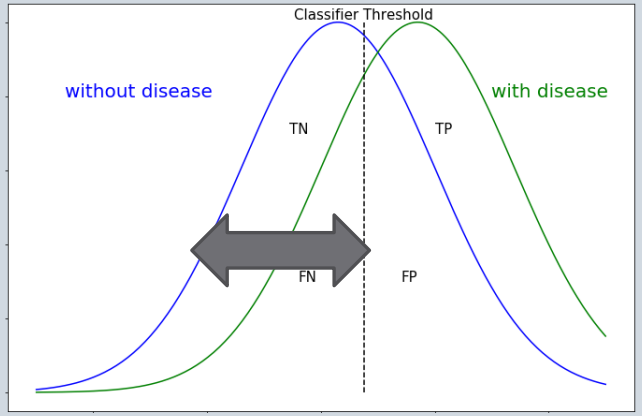
Classification Model Performance measures

(Confusion matrix)

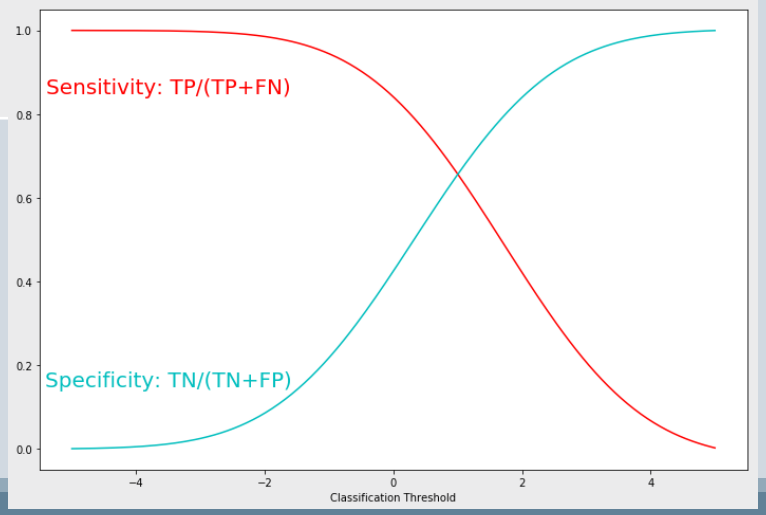
Model Predictions

Ground Truth

	Classified as positive	Classified as negative
Success (correct prediction)	**True Positive (TP)** (hit!)	**True Negative (TN)** (correct rejection)!
Error (wrong prediction)	**False Positive (FP)** (false alarm: type 1 error)	**False Negative (FN)** (miss: type 2 error)



Sensitivity-Specificity tradeoff



$$\textbf{\textit{Semsitivity}} = \frac{TP}{TP + FN}$$

$$\textbf{\textit{Specificity}} = \frac{TN}{TN + FP}$$

Model Evaluation – Accuracy

$$A = \frac{TP + TN}{Total} = \frac{547 + 7650}{8487} = 96.58\%$$

		Actual	
		Pos.	Neg.
Prediction	Pos.	**547**	**114**
	Neg.	**176**	**7650**

Model Evaluation – Precision

$$Pr = \frac{TP}{TP + FP} = \frac{547}{661} = 82.75\%$$

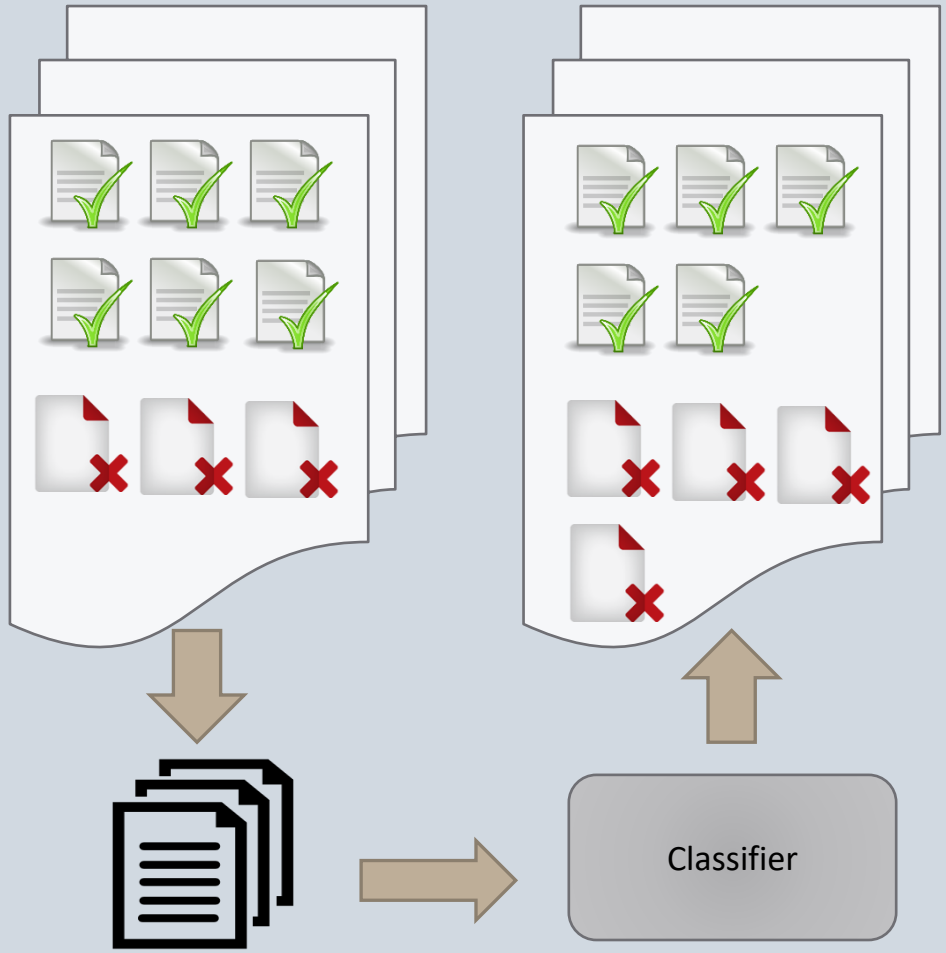
		Actual	
		Pos.	Neg.
Prediction	Pos.	**547**	**114**
	Neg.	**176**	**7650**

Model Evaluation – Recall

$$Rc = \frac{TP}{TP + FN} = \frac{547}{723} = 75.66\%$$

		Actual	
		Pos.	Neg.
Prediction	Pos.	**547**	**114**
	Neg.	**176**	**7650**

Is Accuracy a Good Measure?



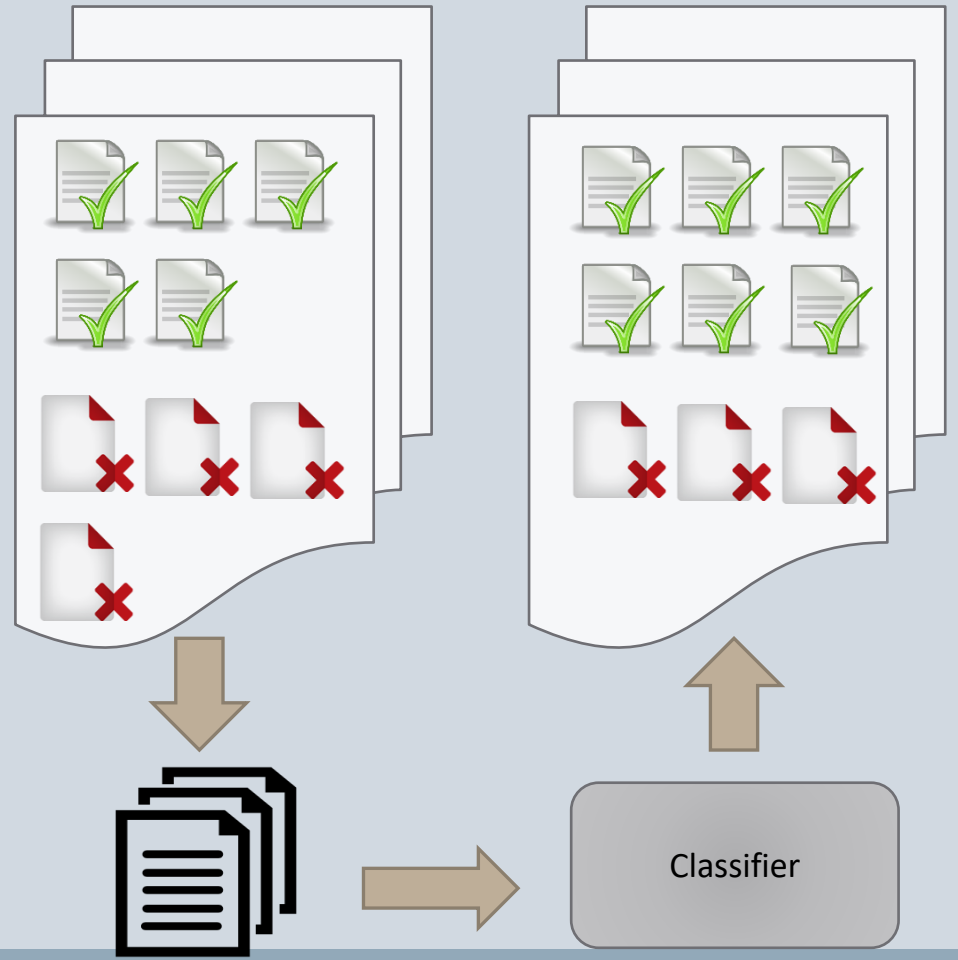
		Actual	
		Pos.	Neg.
Prediction	Pos.	3	1
	Neg.	0	5

Classifier Accuracy: 8/9=88.88%

Precision in finding fraud : 3/4=75%

Recall in finding fraud: 3/3=100%

Is Accuracy a Good Measure?



		Actual	
		Pos.	Neg.
Prediction	Pos.	3	0
	Neg.	1	5

Classifier Accuracy: 8/9=88.88%

Precision in finding fraud : 3/3=100%

Recall in finding fraud: 3/4=75%

Sklearn Confusion Matrix

- A visualization tool used to present the results attained by a learner.
- Helpful when the classifier commonly mislabeling one class as another

		PREDICTED		
		A	B	C
TRUE	A	5	3	0
	B	2	3	1
	C	0	2	11

Recall	5/8	3/6	11/13
Precision	5/7	3/8	11/12

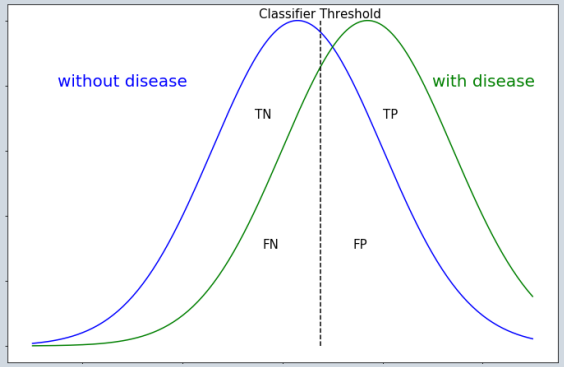
sklearn.metrics.confusion_matrix

Classification Errors: Recall and Precision → F_1-measure

Classification Model Performance measures

	Classified as positive	Classified as negative
Success (correct prediction)	**True Positive (TP)** (hit!)	**True Negative (TN)** (correct rejection)!
Error (wrong prediction)	**False Positive (FP)** (false alarm: type 1 error)	**False Negative (FN)** (miss: type 2 error)

Classification error rate: $e = \frac{FN+FP}{TP+TN+FN+FP}$



$Precision = \frac{TP}{TP+FP}$ - fraction of 'sick' among the ones marked as 'sick' by the classifier.

$Recall = \frac{TP}{TP+FN}$ - fraction of identified 'sick' among all 'sick' (=Specificity)

sklearn.metrics.f1_score(*y_true*, *y_pred)*

F-measure is a harmonic mean of precision and recall

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

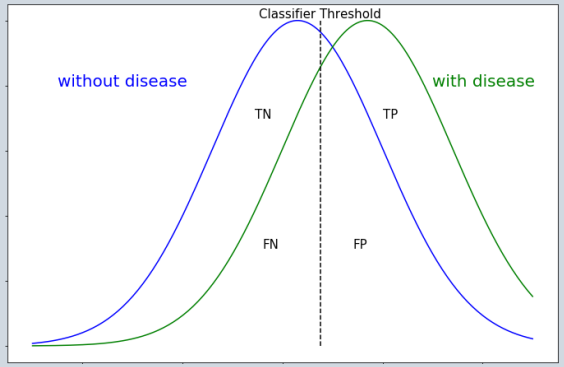
$$\begin{cases} if \quad Precision \approx Recall \quad , F_1 \approx \frac{1}{2}(Precision + Recall) \\ if \quad Precision \neq Recall \quad , F_1 < min(Precision, Recall) \end{cases}$$

Classification Errors: Recall and Precision → F_1-measure

Classification Model Performance measures

	Classified as positive	Classified as negative
Success (correct prediction)	**True Positive (TP)** (hit!)	**True Negative (TN)** (correct rejection)!
Error (wrong prediction)	**False Positive (FP)** (false alarm: type 1 error)	**False Negative (FN)** (miss: type 2 error)

Classification error rate: $e = \frac{FN+FP}{TP+TN+FN+FP}$



$Precision = \frac{TP}{TP+FP}$ - fraction of 'sick' among the ones marked as 'sick' by the classifier.

$Recall = \frac{TP}{TP+FN}$ - fraction of identified 'sick' among all 'sick' (=Specificity)

sklearn.metrics.f1_score(*y_true*, *y_pred)*

Recall is β times more important than precision

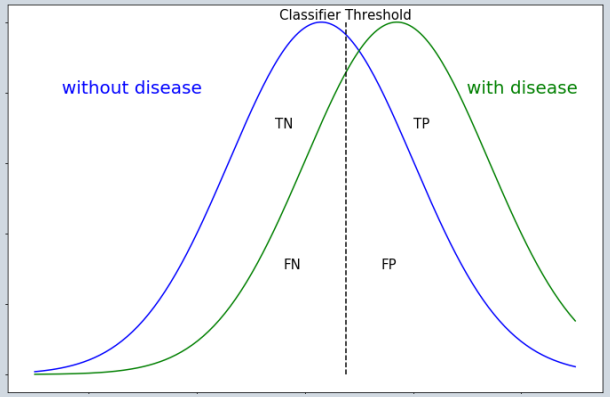
F2-score: weights recall higher than precision.
F0.5-score: weights precision higher than recall.

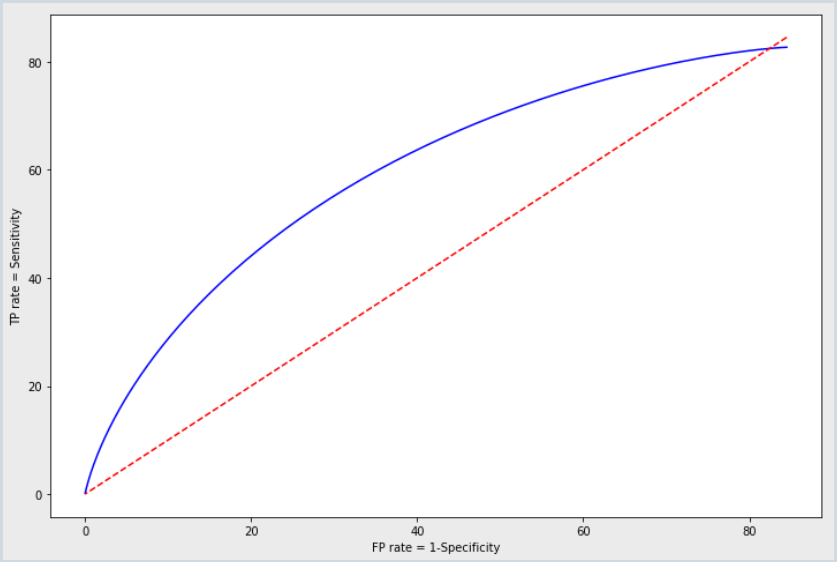
$$F_\beta = \frac{(1+\beta^2) \cdot Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

Receiver Operator Characteristic: Roc Curves

Classification Model Performance measures

	Classified as positive	Classified as negative
Success (correct prediction)	**True Positive (TP)** (hit!)	**True Negative (TN)** (correct rejection)!
Error (wrong prediction)	**False Positive (FP)** (false alarm: type 1 error)	**False Negative (FN)** (miss: type 2 error)



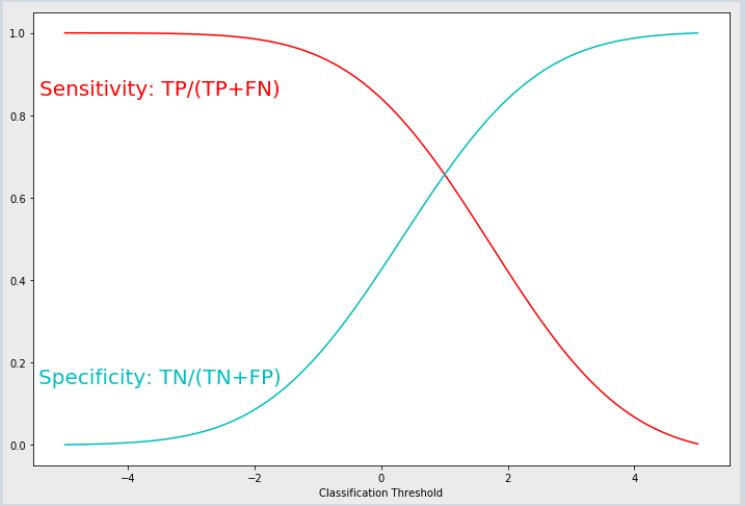


ROC curves: how TP rate changes with the change in FP rate

AUC: Area Under the Curve

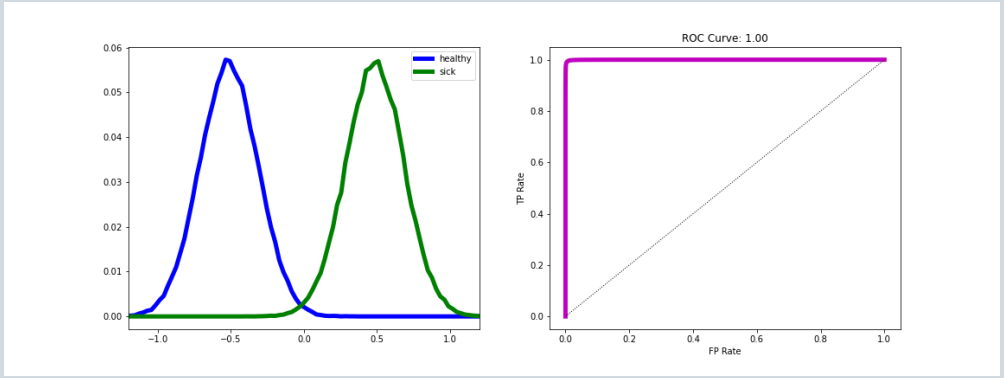
$$\text{TP-Rate}=Semsitivity = \frac{TP}{TP+FN}$$

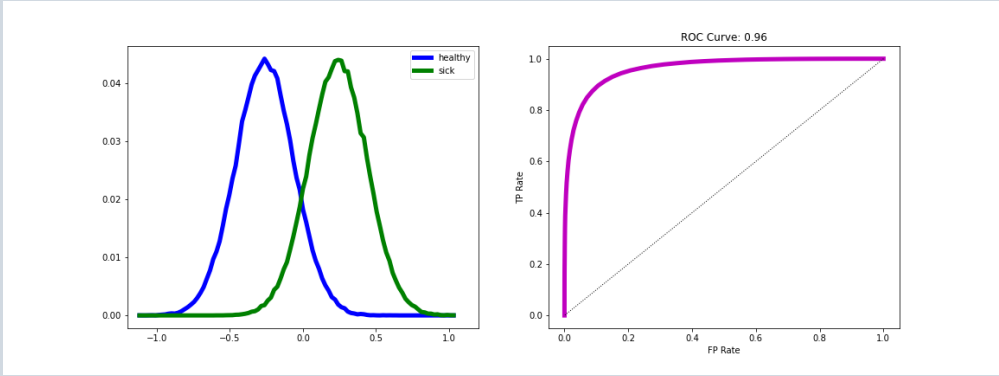
$$FP - Rate = 1 - Specificity = \frac{FP}{TN + FP}$$

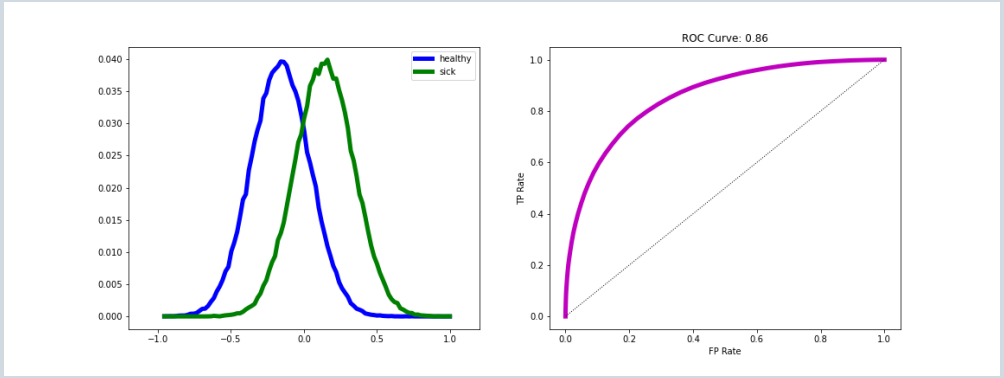


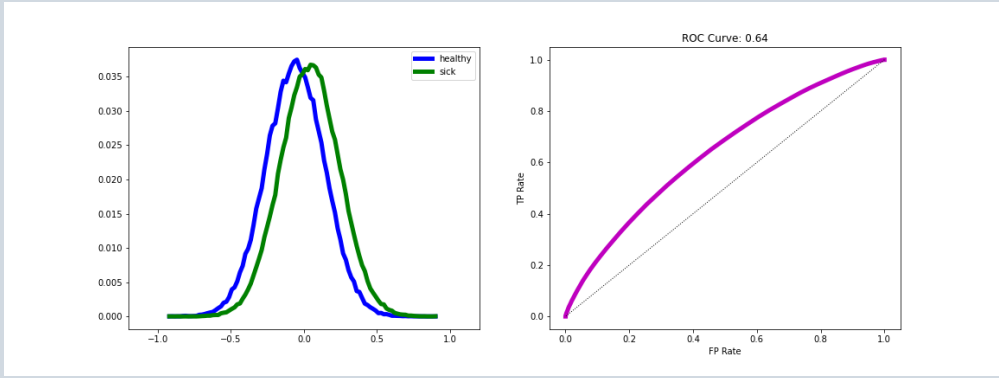
Receiver Operator Characteristic: Roc Curve Area Under The Curve: Auc

Model_evaluation.ipynb









Classification Metrics

metrics.accuracy_score(y_true, y_pred, *[, …])	Accuracy classification score.
metrics.auc(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule.
metrics.average_precision_score(y_true, …)	Compute average precision (AP) from prediction scores.
metrics.balanced_accuracy_score(y_true, …)	Compute the balanced accuracy.
metrics.brier_score_loss(y_true, y_prob, *)	Compute the Brier score loss.
metrics.classification_report(y_true, y_pred, *)	Build a text report showing the main classification metrics.
metrics.cohen_kappa_score(y1, y2, *[, …])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.recall_score(y_true, y_pred, *[, …])	Compute the recall.
metrics.roc_auc_score(y_true, y_score, *[, …])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score, *[, …])	Compute Receiver operating characteristic (ROC).
metrics.top_k_accuracy_score(y_true, y_score, *)	Top-k Accuracy classification score.
metrics.zero_one_loss(y_true, y_pred, *[, …])	Zero-one classification loss.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics
https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics

Classification Model Evaluation

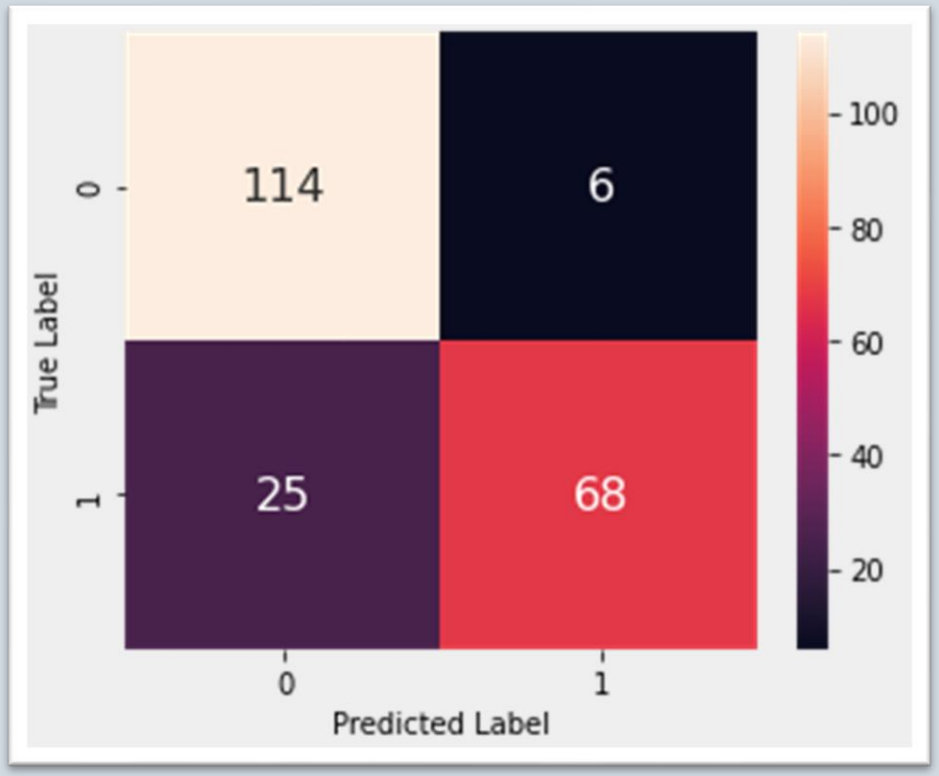
Classified as

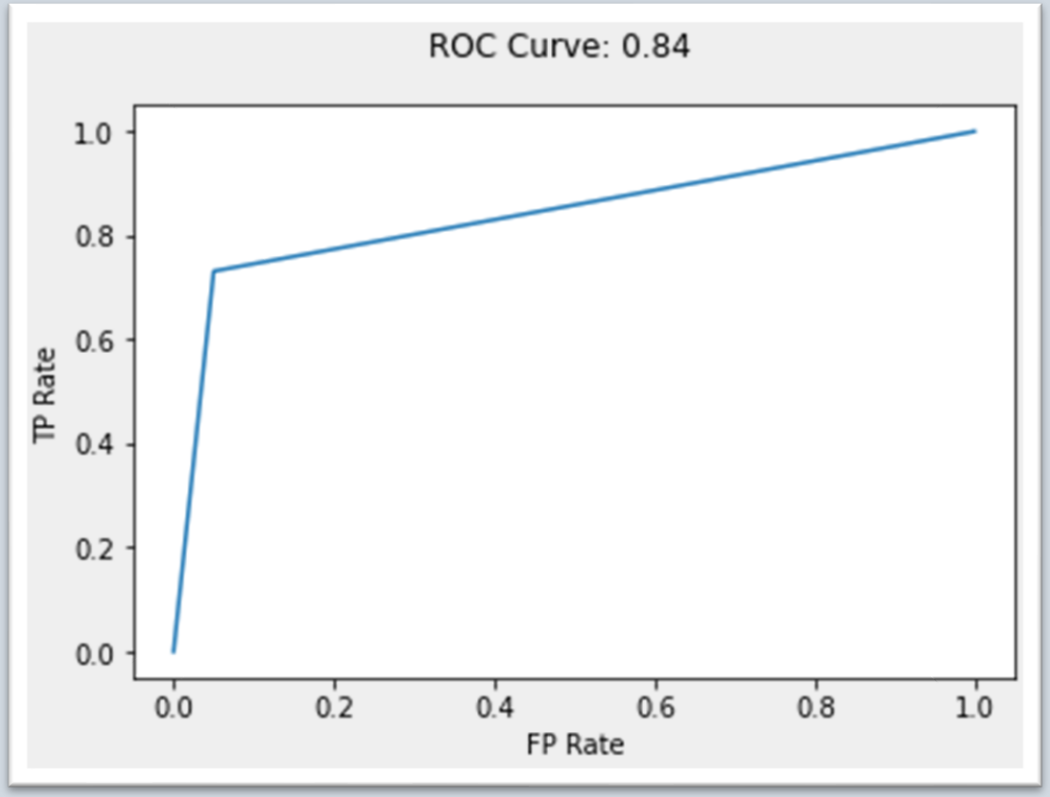
False Negatives: ***miss***

Model_evaluation.pynb

True Value

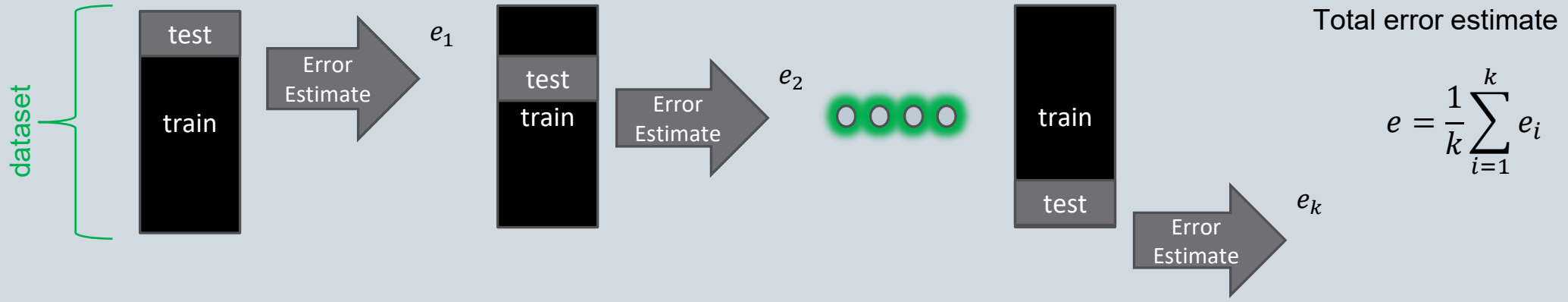
False Positives: false alarm





K-fold Cross-validation

- Cross validation is used to assess error by repeatedly splitting data into test set and training set
- Training and test set data in the same fold should never overlap



Make sure to retain some true test data for final evaluation!

5-fold Cross-validation

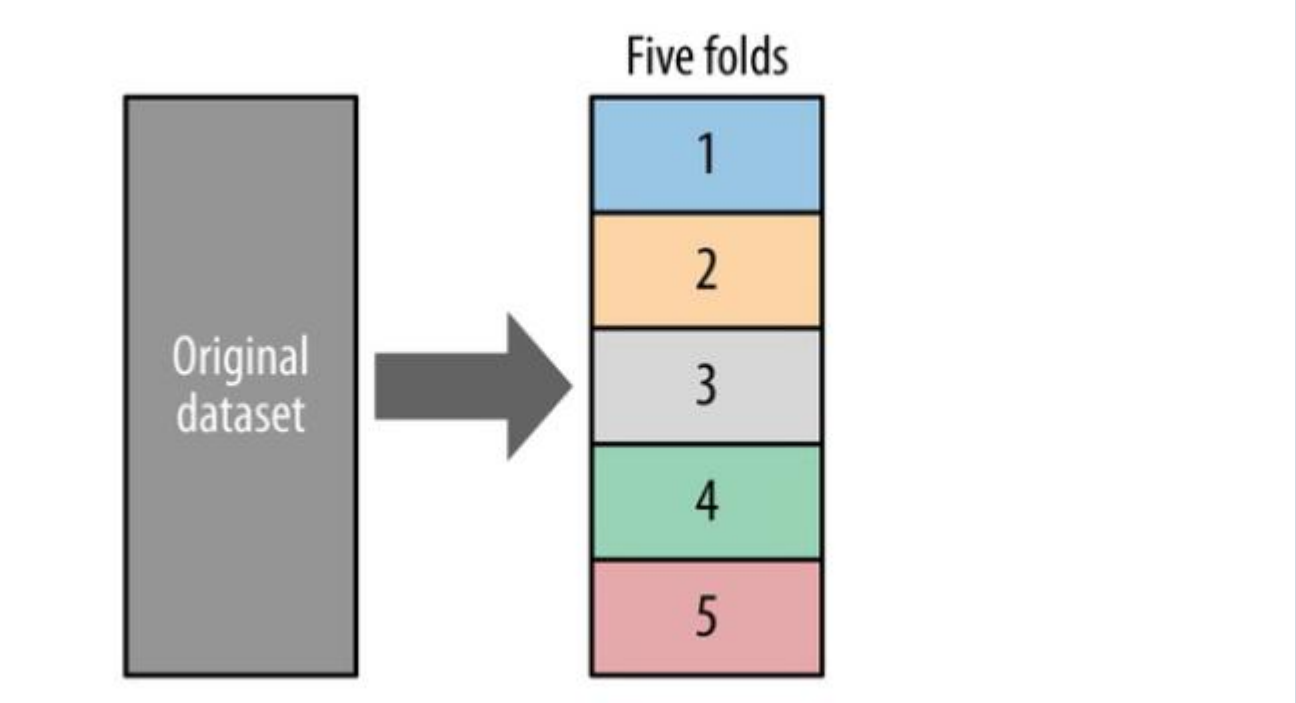


Image from “Data Science for Business”, Provost, Fawcett, 2013

5-fold Cross-validation

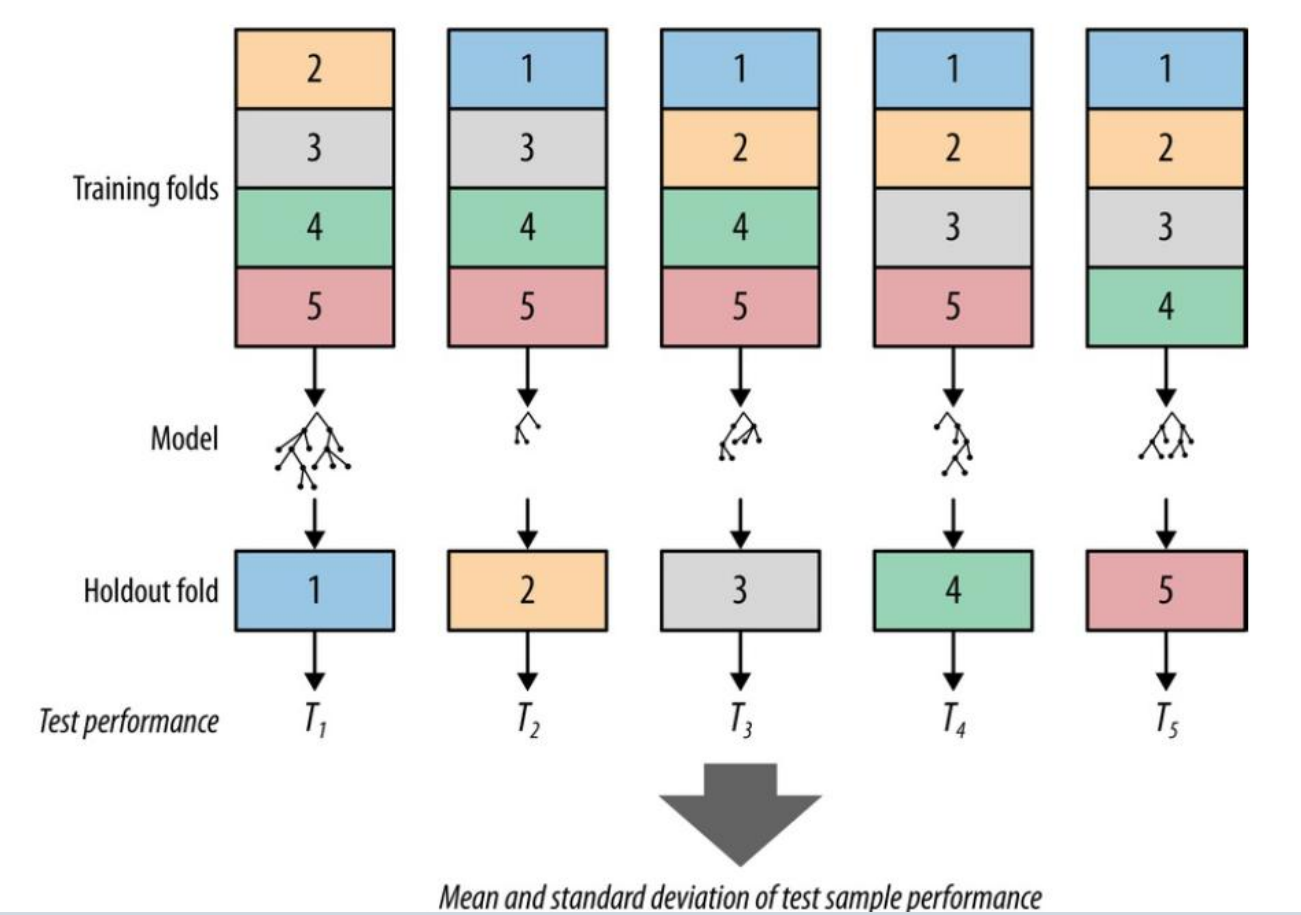
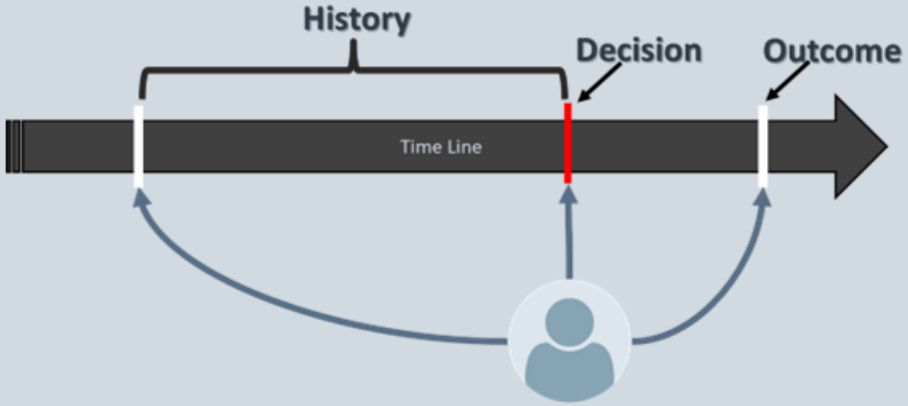


Image from “Data Science for Business”, Provost, Fawcett, 2013

Data leakage

- When information from outside the training dataset (especially the target or future info) slips into the model.
- Leads to overly optimistic performance during training, but poor generalization.

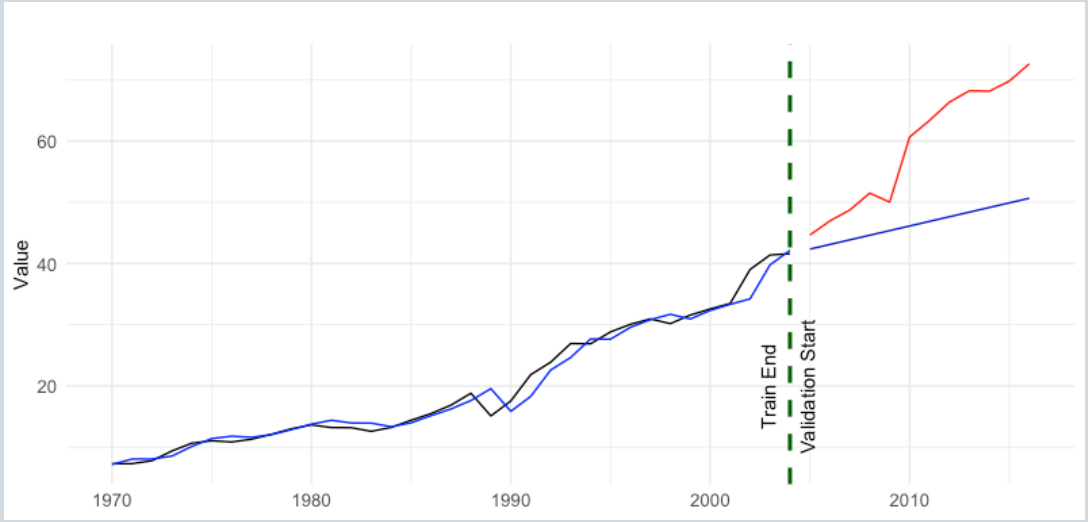
- **Common Causes:**
  - Imputation with full dataset statistics (mean/median including test data).
  - Encoding with the target variable (target encoding without CV).
  - Temporal leakage: using future values (e.g., “days until churn” at signup).
  - Data prep done before train-test split.



- **How to Prevent:**
  - Always split train/test first, then fit imputers/encoders only on train.
  - Use Pipelines/ColumnTransformer to keep preprocessing inside CV.
  - Check features for suspicious correlation with the target.

Time Series Train/Test Split

- In time series, temporal order matters — future values should never influence past predictions.
- Standard practice:
  - Train on the earlier portion of the data.
  - Validate/test on the later portion.



https://towardsdatascience.com/4-things-to-do-when-applying-cross-validation-with-time-series-c6a5674ebf3a/

TimeSeriesSplit

- Standard k-fold CV randomly splits the data into folds.
  - Random shuffling breaks temporal dependencies and leads to data leakage:
  - Solution: use TimeSeriesSplit (walk-forward or rolling validation) instead of random k-fold.
- TimeSeriesSplit in scikit-learn :
  - Successively expands the training window.
  - Always tests on later (future) data.

https://towardsdatascience.com/4-things-to-do-when-applying-cross-validation-with-time-series-c6a5674ebf3a/

```python
from sklearn.model_selection import TimeSeriesSplit


tscv = TimeSeriesSplit(n_splits=5)
for train_idx, test_idx in tscv.split(X):
    print("Train:", train_idx, "Test:", test_idx)
```

```
Split 1 Train range: 2015-01-02 00:00:00 → 2015-11-04 00:00:00 Test range : 2015-11-05 00:00:00 → 2016-09-02 00:00:00
Split 2 Train range: 2015-01-02 00:00:00 → 2016-09-02 00:00:00 Test range : 2016-09-06 00:00:00 → 2017-07-05 00:00:00
Split 3 Train range: 2015-01-02 00:00:00 → 2017-07-05 00:00:00 Test range : 2017-07-06 00:00:00 → 2018-05-03 00:00:00
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

Generalization and Overfitting

- **Generalization** is the property of a model or modeling process, whereby the model applies to data that were not used to build the model.

- **Overfitting** is the tendency of data mining procedures to tailor models to the training data, at the expense of generalization to previously unseen data points.
  - Model is too closely (or exactly) to the data
  - All of the model parameters can be justified (statistically) on the training data
  - Model is not general and will fail when used with unseen data point

Example – Buyers Classifier

X2 (gender)	X1 (age)	X3 (city)	X4 (children)	X4 (Income)	….	X30	Buyer
Male	24.3	Tel-Aviv	0	12,000	….	….	No
Female	45.2	Jerusalem	3	24,000	….	….	Yes
Female	31.1	Ashkelon	2	20,000	….	….	Yes
Male	28.0	Jerusalem	3	19,000	….	….	No
….	….	….	….	….	….	….	…..
Male	35.4	Haifa	4	17,000	….	….	Yes

If (

(Gender = Female AND 45.1<Age<45.3 AND City=Jerusalem AND #Childern=3 AND 23,999<Income<24,001 AND...)

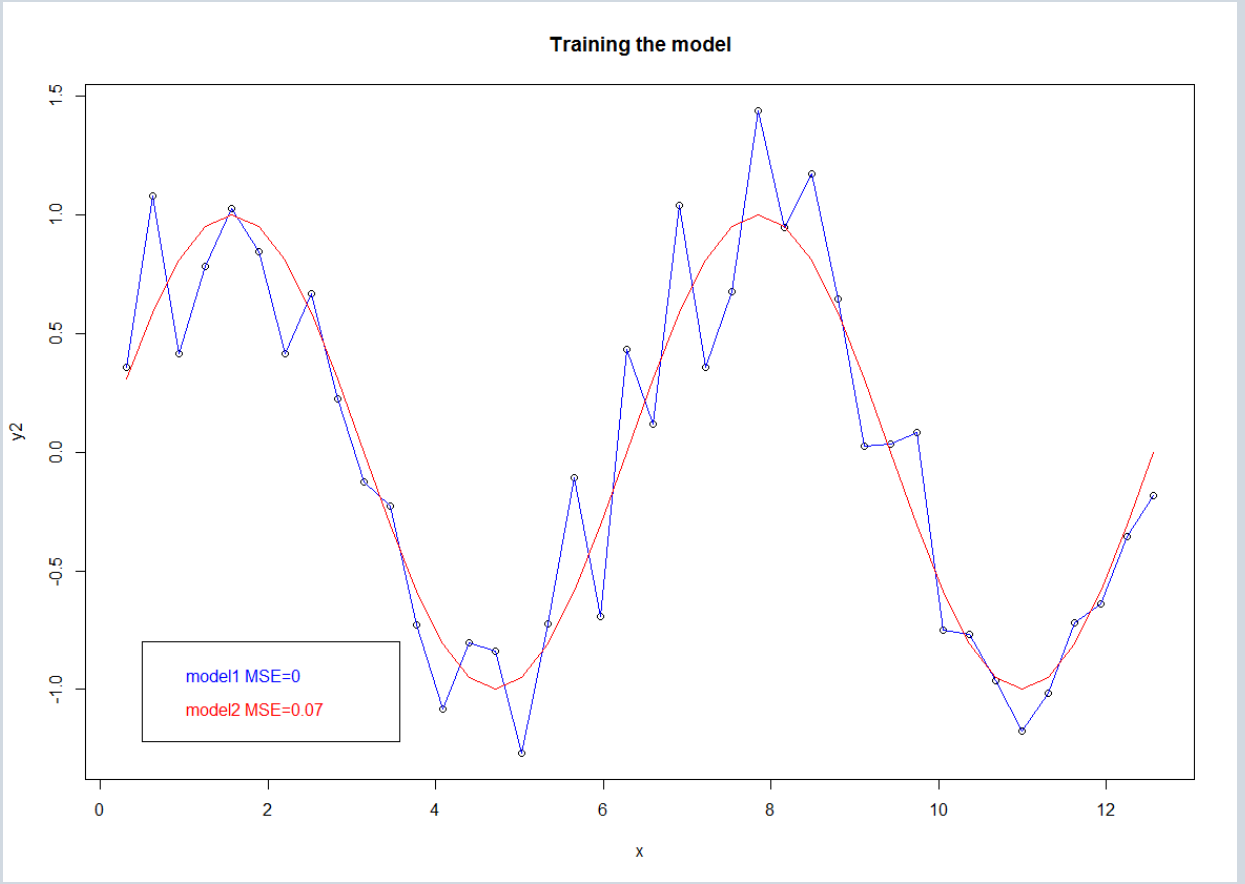
Or (Gender = Female AND 31.0<Age<31.2 AND City=Ashkelon AND #Childern=2 AND 19,999<Income<20,001 AND...)

Or (Gender = Male AND 27.9<Age<28.1 AND City=Haifa AND #Childern=4 AND 16,999<Income<17,001 AND...)

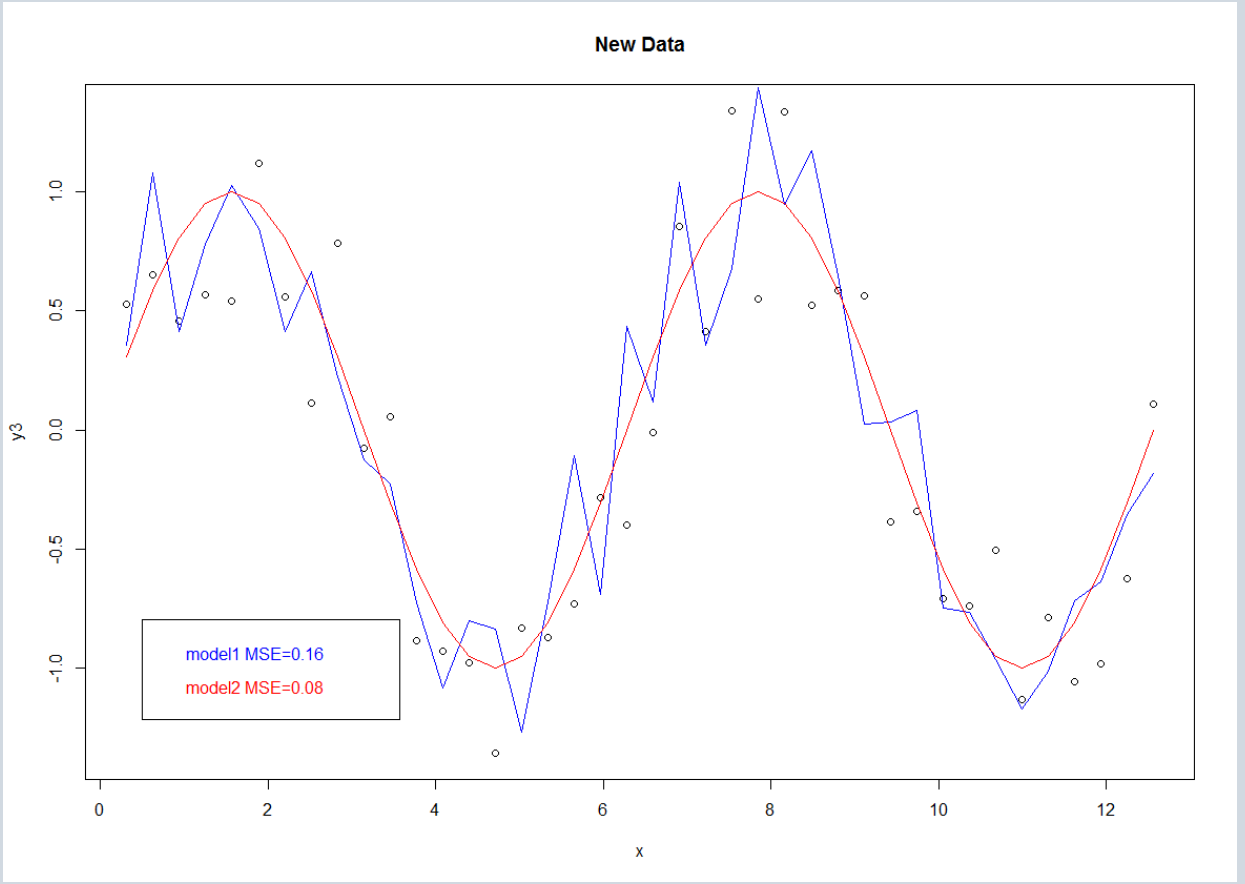
Or (…..)) => Yes

ELSE => NO

Which Model is Better?

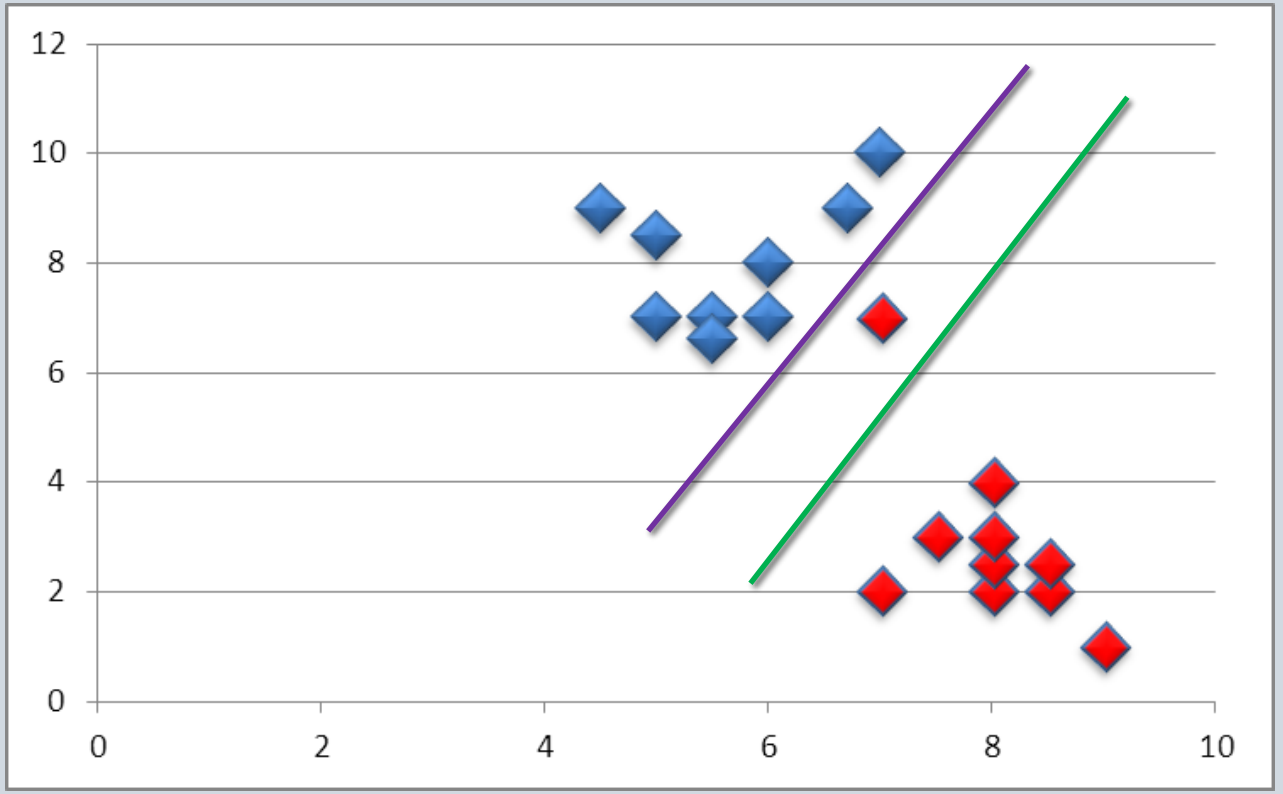


Which Model is Better?



Which Model is Better?

Income

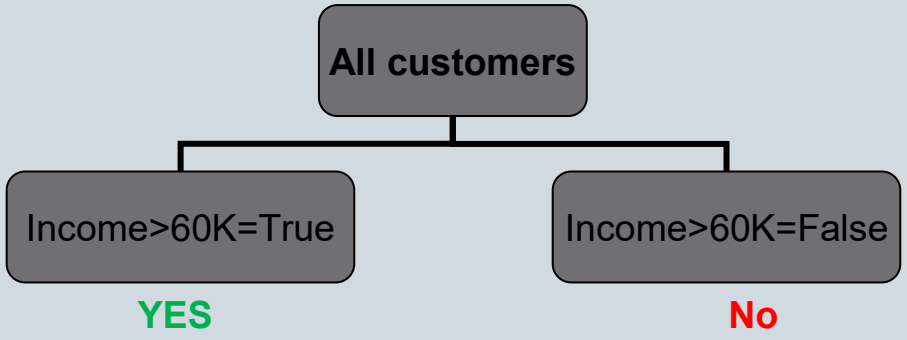


Ads click

Overfitting - Decision Trees

#	Income>60K	Click on Ad
1	0	No
2	0	No
3	0	No
4	0	Yes
5	0	Yes
6	1	Yes
7	1	Yes
8	1	Yes
9	1	Yes
10	1	Yes

Correctly classified instances for training data – 80%



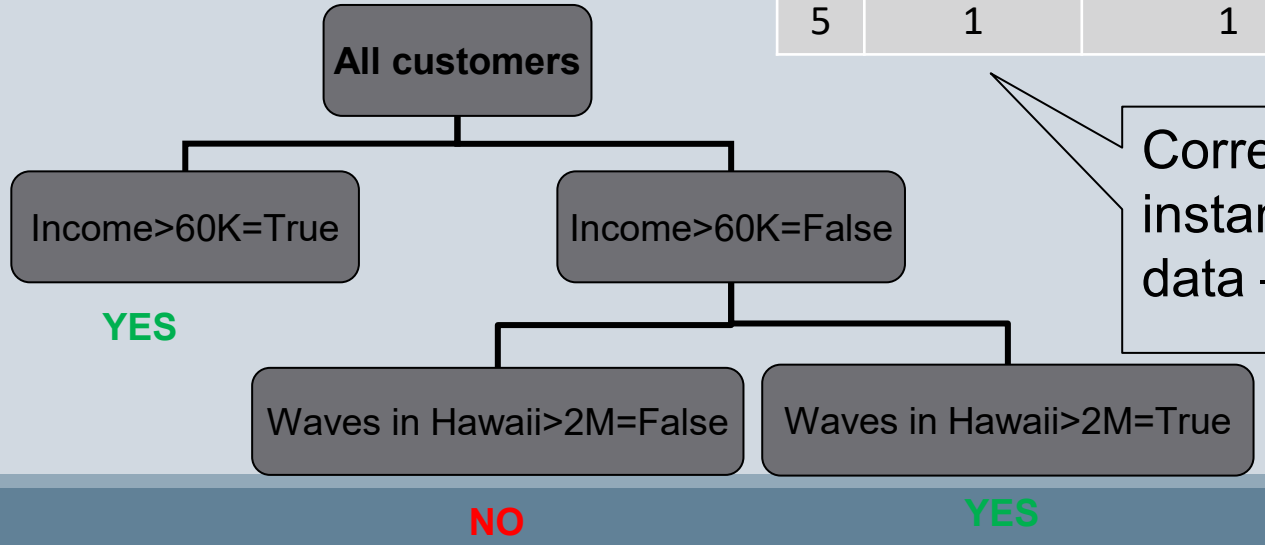
Correctly classified instances for test data – 80%

#	Income>60K	Click on Ad
11	0	No
12	0	No
13	0	Yes
14	1	Yes
15	1	Yes

Overfitting - Decision Trees

#	Income> 60K	Waves in Hawaii>2M	Click on Ad
1	0	0	No
2	0	0	No
3	0	0	No
4	0	1	Yes
5	0	1	Yes
6	1	1	Yes
7	1	1	Yes
8	1	0	Yes
9	1	1	Yes
10	1	0	Yes

Correctly classified instances for training data – 100%



#	Income> 60K	Waves in Hawaii>2M	Click on Ad
1	0	0	No
2	0	1	No
3	0	0	Yes
4	1	1	Yes
5	1	1	Yes

Correctly classified instances for test data – 60%

Causes of Overfitting

Overfitting Due to Presence of Noise

- Mislabeled instances may contradict the class labels of other similar records.

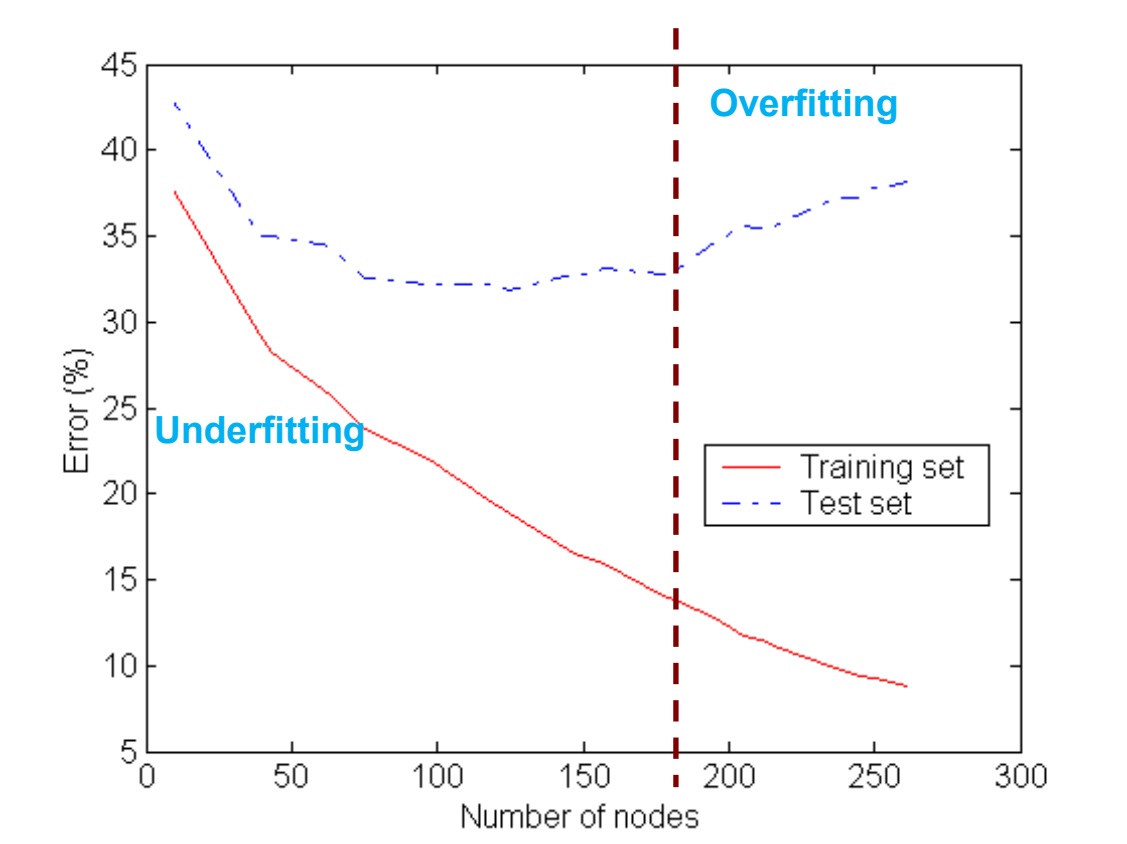
Overfitting Due to Lack of Representative Instances

- Lack of representative instances in the training data can prevent refinement of the learning algorithm.

Overfitting Due to Too-Complex Models

- Failure to compensate for algorithms that explore a large number of alternatives can result in spurious fitting.

Underfitting and Overfitting



Model Overfitting.ipynb

Overfitting:
two samples of the same data will result in very different models

Underfitting: when model is too simple, both training and test errors are large

Overfitting: when the model is too complex, training error is small, whereas test error is large

Possible Remedies*

- Use simpler (less flexible) models
- Use fewer number of features in final model (feature selection)
- Enrich data with influential predictors
- Enrich data with more observations
- Tip: Also, check compatibility of training and validation sets

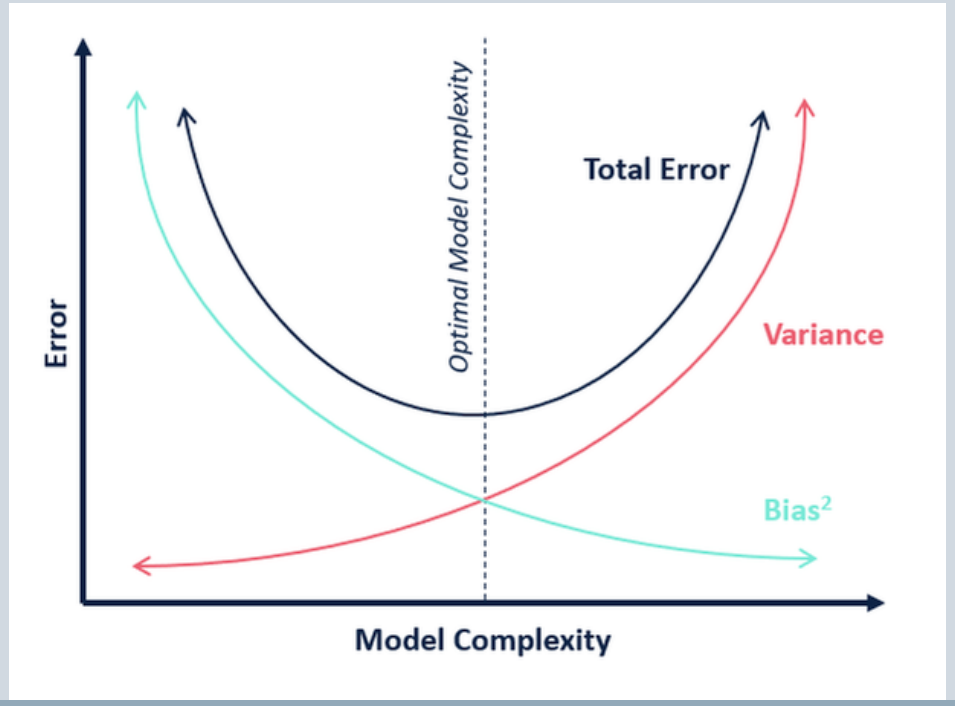
*May or may not work – depending on the root of the problem

Bias and Variance Trade Off

Bias is the difference between the average prediction of our model and the correct target value which model is trying to predict.

Variance is the amount that the estimate of the target function will change if different training data was used.

- **Bias-Variance Trade-Off**
  - High Variance-High Bias — The model is inconsistent and also inaccurate on prediction
  - Low Variance-High Bias — Models is consistent but low on prediction
  - High Variance-Low Bias — Somewhat accurate but inconsistent on prediction
  - Low Variance-Low Bias — ideal scenario, the model is consistent and highly accurate on prediction



https://medium.com/@itbodhi/bias-and-variance-trade-off-542b57ac7ff4

Bias and Variance Trade Off

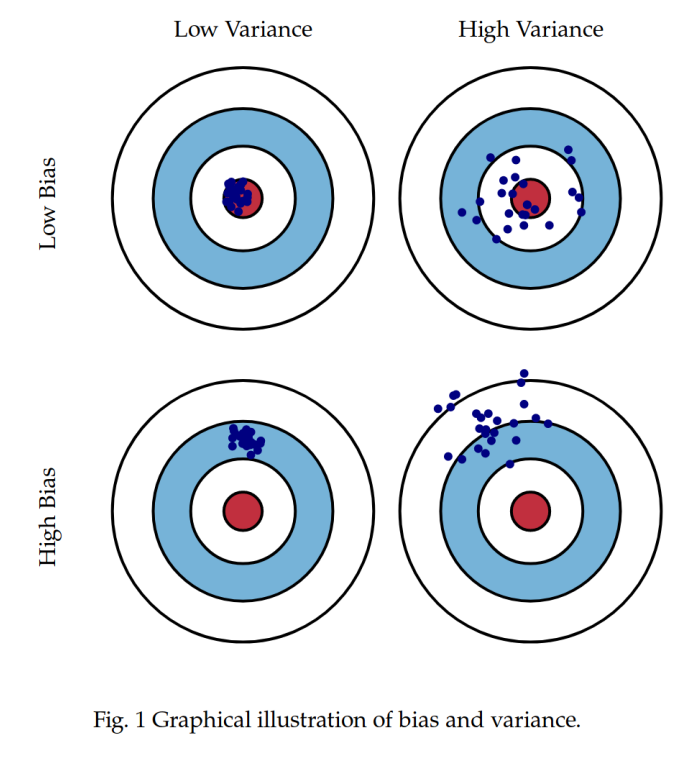
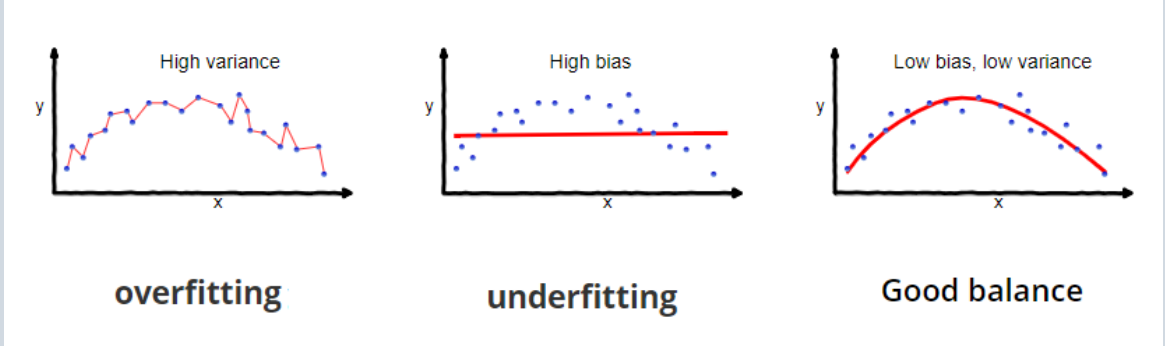
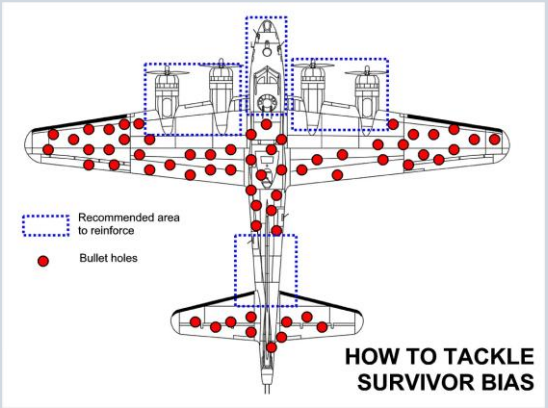


Fig. 1 Graphical illustration of bias and variance.

https://medium.com/@itbodhi/bias-and-variance-trade-off-542b57ac7ff4





https://medium.com/@ruer98/wwii-bombers-and-selection-bias-in-data-science-1b85eb25a672

THANK YOU FOR LISTENING

ZVI.BENAMI@MAIL.HUJI.AC.IL