Introduction to Machine Learning

Лекции

Алексей Александрович Шпильман

alexey@shpilman.com

Telegram: @aashpilman

Практики

Олег Свидченко

svidchen-ko@mail.ru

Telegram: @ArgentumWalker

Оценка

3 элемента: квизы, домашки, экзамен. За каждый элемент – оценка от 0 до 10.

Результирующая оценка: **0.2 квизы + 0.4 домашки + 0.4 экзамен**.

Блокирующие элементы: домашки, экзамен.

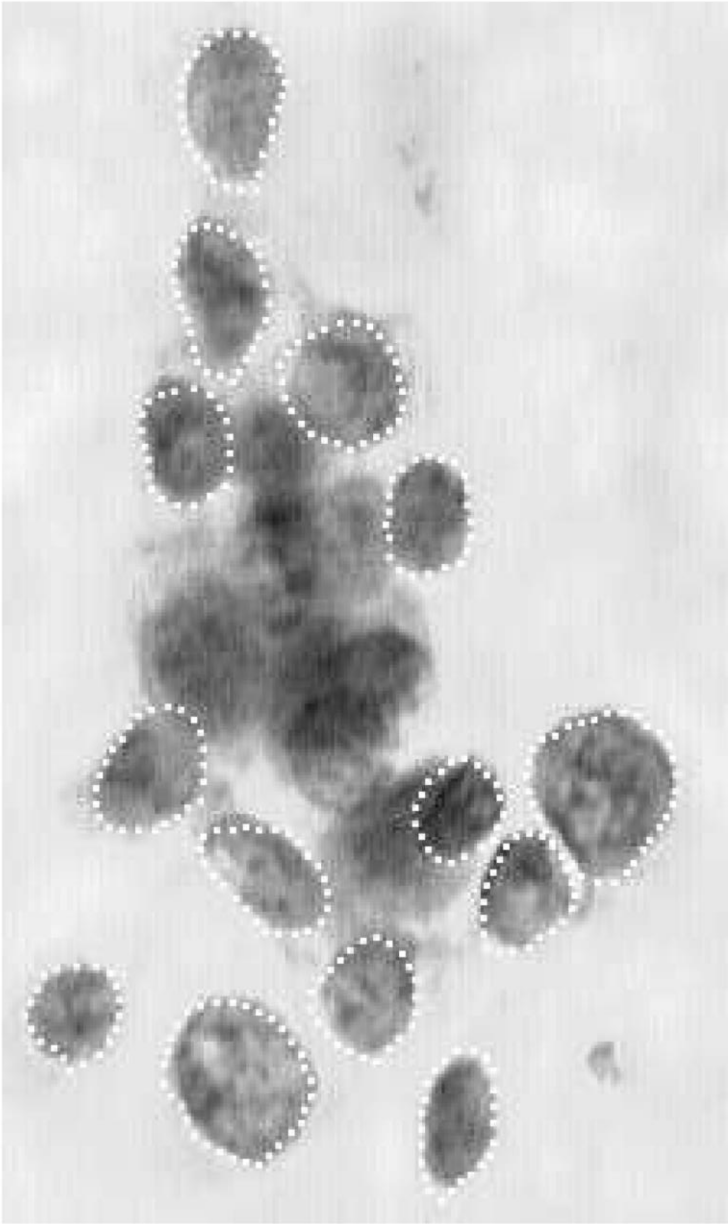
Два дедлайна по домашкам – мягкий и жесткий. После мягкого дедлайна баллы за домашку половинятся. После жесткого дедлайна домашку сдавать нельзя. Небольшие исправления – можно.

Benign/malignant tumor dataset

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus (mean, highest and standard error):

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)



Cell features

label	1	2	3	4	5	6	7	8	9	10
M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871
M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999
M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744
M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883
M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613
M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742
M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451
M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389
M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243
M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697
M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082
M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078
M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338
M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682
M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077
M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922
M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356
M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395
B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766
B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811
B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905
M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032
M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278
M	16.65	21.38	110	904.6	0.1121	0.1457	0.1525	0.0917	0.1995	0.0633

Supervised learning

- Input: $\mathbf{x}$
- Output (label): $\boldsymbol{y}$
- Target function: $\boldsymbol{f}: \boldsymbol{X} \rightarrow \boldsymbol{Y}$
- Data:
  $(\mathbf{x}_1, \boldsymbol{y}_1), (\mathbf{x}_2, \boldsymbol{y}_2), .., (\mathbf{x}_N, \boldsymbol{y}_N)$
- Hypothesis: $\boldsymbol{h}: \boldsymbol{X} \rightarrow \boldsymbol{Y}$

Classification problem - $\boldsymbol{y}$ belongs to a set (of classes).

Regression problem - $\boldsymbol{y}$ is a real valued number (or a vector).

Instance-based learning

Instance-based learning
Lazy learning

$$h(\mathrm{x}; D) = \arg\max_{y \in \mathrm{Y}} \underbrace{\sum_{\mathrm{x}_i \in D} [y_i = y] w(\mathrm{x}_i, \mathrm{x})}_{\Gamma_y(\mathrm{x})}$$

$w(\mathrm{x}_i, \mathrm{x})$ – weight of x_i for x

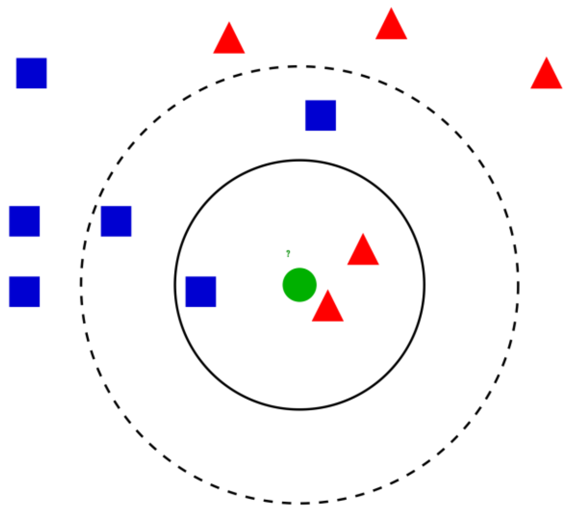
$\Gamma_y(\mathrm{x})$ – affinity of x to class y

kNN – k-nearest neighbors

$$h(\mathrm{x}; D) = \arg\max_{y \in \mathrm{Y}} \sum_{\mathrm{x}_i \in D} [y_i = y] w(\mathrm{x}_i, \mathrm{x})$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if } \mathrm{x}_i - \text{one of the k nearest neighbors of x}$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if distance } \rho(\mathrm{x_i}, \mathrm{x}) < \mathrm{R}$$ (Radius Neighbors)



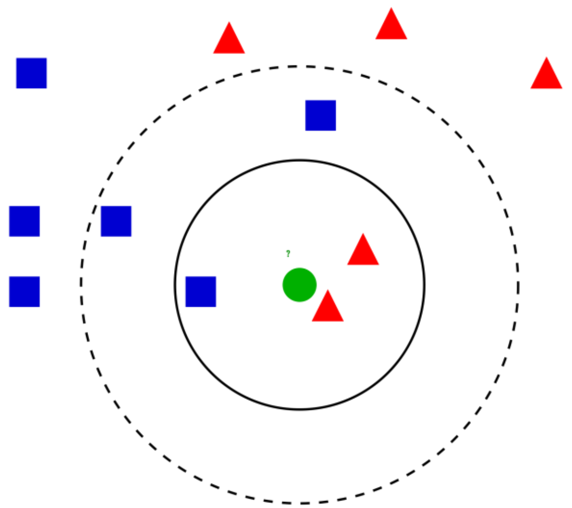
Important note: we don't need a vector space, we only need the distance $\rho(\mathrm{x_i}, \mathrm{x})$ to be defined.

kNN – k-nearest neighbors

$$h(\mathrm{x}; D) = \arg\max_{y \in \mathrm{Y}} \sum_{\mathrm{x}_i \in D} [y_i = y] w(\mathrm{x}_i, \mathrm{x})$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if } \mathrm{x}_i - \text{one of the k nearest neighbors of x}$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if distance } \rho(\mathrm{x_i}, \mathrm{x}) < \mathrm{R} \text{ (Radius Neighbors)}$$



Leave-one-out error:

$$\mathrm{LOO}(k, D) = \frac{\sum_{\mathrm{x}_i \in D} [h(\mathrm{x}_i; D \backslash \mathrm{x}_i; k) \neq y_i]}{|D|}$$

Train/Test

Put aside a part of a dataset ($\approx 10 - 20\%$) for testing (**test dataset**).

Train classifier on what is left (**training dataset**).

Optimize hyperparameters, such as the number of neighbors, looking at the accuracy on the test dataset.

If we have too many hyperparameters, there is a present danger of overfitting on **test dataset**. Put aside one more dataset (**validation dataset**).

Train/Validate/Test

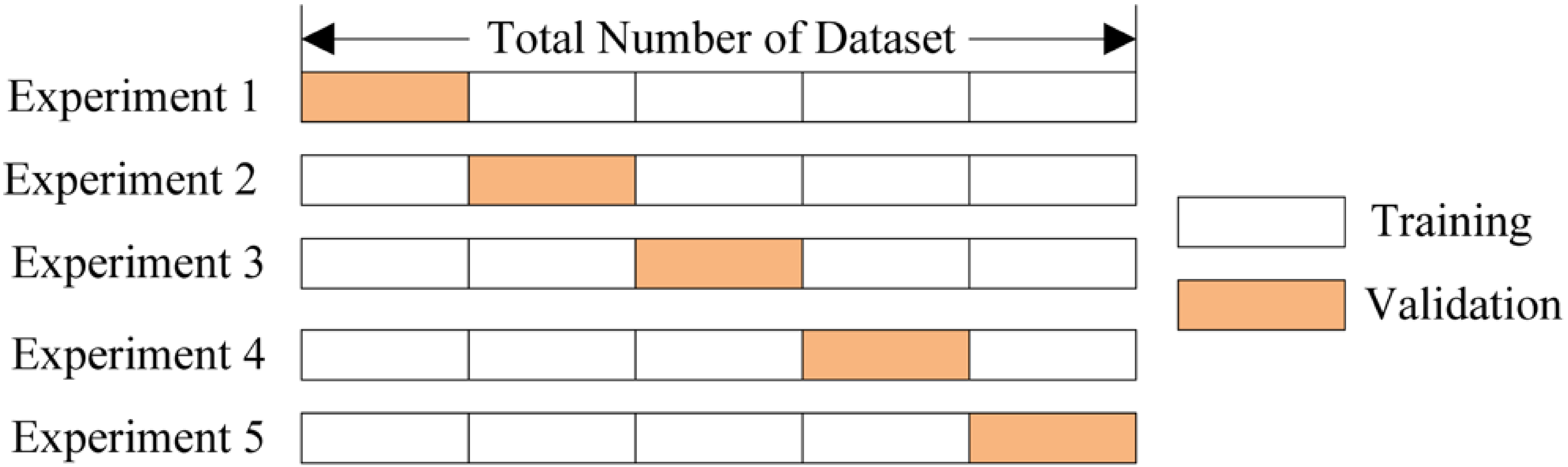
Put aside a part of a dataset ($\approx 10 - 20\%$) for validation (**validation dataset**) and a part for testing (**test dataset**).

Train classifier on what is left (**training dataset**).

Optimize hyperparameters, such as the number of neighbors, looking at the accuracy on the **validation dataset**.

Compare classifiers on the **test dataset**.

Cross-validation

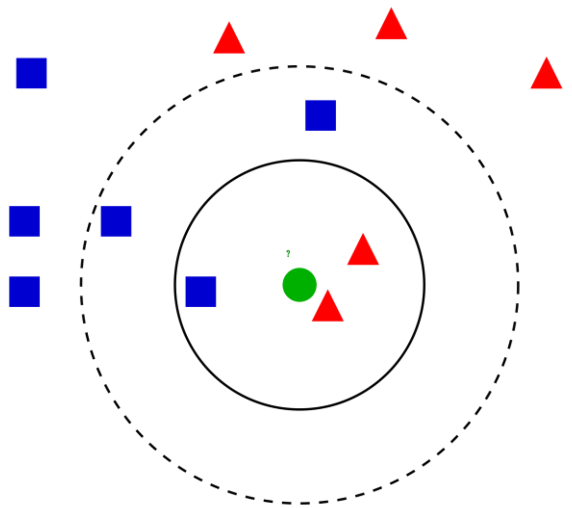


kNN – k-nearest neighbors

$$h(\mathrm{x}; D) = \arg\max_{y \in \mathrm{Y}} \sum_{\mathrm{x}_i \in D} [y_i = y] w(\mathrm{x}_i, \mathrm{x})$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if } \mathrm{x}_i - \text{one of the k nearest neighbors of x}$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if distance } \rho(\mathrm{x}_\mathrm{i}, \mathrm{x}) < \mathrm{R}$$ (Radius Neighbors)



WkNN – Weighted kNN

Different functions for *w*:

$$w_i = \left[\frac{r - \rho(\mathrm{x}, \mathrm{x}_i)}{r}\right]_+ - Triangle\ kernel$$

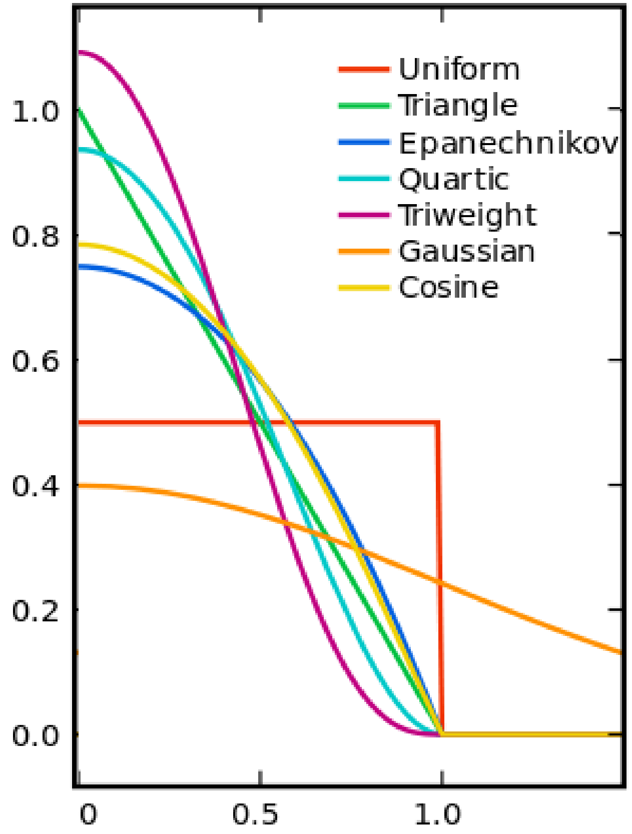
$$w_i = q^{-\rho(\mathrm{x}, \mathrm{x}_i)} - Gaussian\ kernel$$

WkNN – Weighted kNN

Different functions for *w*:

$$w_i = \left[\frac{r - \rho(\mathrm{x}, \mathrm{x}_i)}{r}\right]_+$$

$$w_i = q^{-\rho(\mathrm{x}, \mathrm{x}_i)}$$



Parzen window method:

$$w(\mathrm{x}, \mathrm{x}_i) = K\left(\frac{\rho(\mathrm{x}, \mathrm{x}_i)}{r}\right)$$

Fixed width

$$w(\mathrm{x}, \mathrm{x}_i) = K\left(\frac{\rho(\mathrm{x},\mathrm{x}_i)}{\rho(\mathrm{x},\mathrm{x}_j)}\right)$$, where x_j is the (k+1)st neighbor

kNN width

Cell features

label	1	2	3	4	5	6	7	8	9	10
M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871
M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999
M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744
M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883
M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613
M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742
M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451
M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389
M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243
M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697
M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082
M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078
M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338
M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682
M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077
M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922
M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356
M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395
B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766
B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811
B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905
M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032
M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278
M	16.65	21.38	110	904.6	0.1121	0.1457	0.1525	0.0917	0.1995	0.0633

Spam features

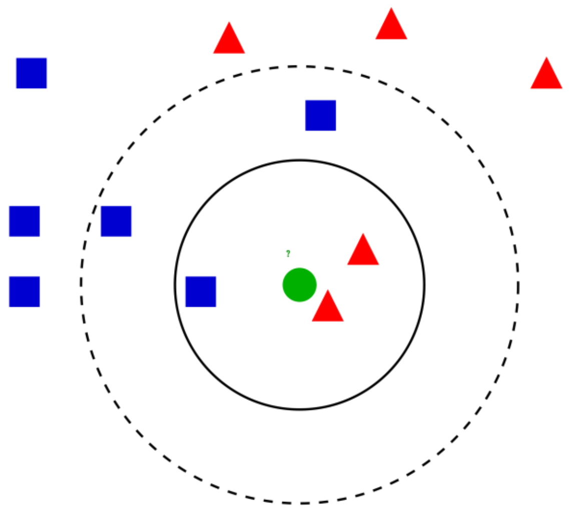
char_freq_[	char_freq_!	char_freq_$	char_freq_#	capital_run_length_	capital_run_length_	capital_run_length_	label
0	0.778	0	0	3.756	61	278	1
0	0.372	0.18	0.048	5.114	101	1028	1
0	0.276	0.184	0.01	9.821	485	2259	1
0	0.137	0	0	3.537	40	191	1
0	0.135	0	0	3.537	40	191	1
0	0	0	0	3	15	54	1
0	0.164	0.054	0	1.671	4	112	1
0	0	0	0	2.45	11	49	1
0	0.181	0.203	0.022	9.744	445	1257	1
0	0.244	0.081	0	1.729	43	749	1
0	0.462	0	0	1.312	6	21	1
0	0.663	0	0	1.243	11	184	1
0	0.786	0	0	3.728	61	261	1
0	0	0	0	2.083	7	25	1
0	0.357	0	0	1.971	24	205	1
0	0.572	0.063	0	5.659	55	249	1
0	0.428	0	0	4.652	31	107	1
0	1.975	0.37	0	35.461	95	461	1
0	0.455	0	0	1.32	4	70	1

kNN – k-nearest neighbors

$$h(\mathrm{x}; D) = \arg\max_{y \in Y} \sum_{\mathrm{x}_i \in D} [y_i = y] w(\mathrm{x}_i, \mathrm{x})$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if } \mathrm{x}_i - \text{one of the k nearest neighbors of x}$$

$$w(\mathrm{x}_i, \mathrm{x}) = 1, \text{if distance } \rho(\mathrm{x}_\mathrm{i}, \mathrm{x}) < \mathrm{R} \text{ (Radius Neighbors)}$$



Classifier evaluation

		$\boldsymbol{y_i}$		
		$= \boldsymbol{y}$	$\neq \boldsymbol{y}$	
$h(\mathbf{x_i})$	$= \boldsymbol{y}$	True positive (TP)	False positive (FP)	Positive predictive value, **Precision** $\frac{TP}{TP+FP}$
	$\neq \boldsymbol{y}$	False negative (FN)	True negative (TN)	
		True positive rate, **Recall** $\frac{TP}{TP+FN}$	False positive rate $\frac{FP}{FP+TN}$	**Accuracy** $\frac{TP+TN}{TP+FP+TN+FN}$

$$\boldsymbol{F_1 score} = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2\frac{precision * recall}{precision + recall} = \frac{2TP}{2TP + FP + FN}$$

Example #1

		y_i		
		$= y$	$\neq y$	
$h(\mathbf{x_i})$	$= y$	0 (TP)	0 (FP)	Positive predictive value, **Precision** 0%
	$\neq y$	50 (FN)	950 (TN)	
		True positive rate, **Recall** 0%	False positive rate 0%	**Accuracy** 95%

$$\boldsymbol{F_1 score} = 0$$

Example #2

		$\boldsymbol{y_i}$		
		$= \boldsymbol{y}$	$\neq \boldsymbol{y}$	
$h(\mathbf{x_i})$	$= \boldsymbol{y}$	50 (TP)	950 (FP)	Positive predictive value, **Precision** 5%
	$\neq \boldsymbol{y}$	0 (FN)	0 (TN)	
		True positive rate, **Recall** 100%	False positive rate 100%	**Accuracy** 5%

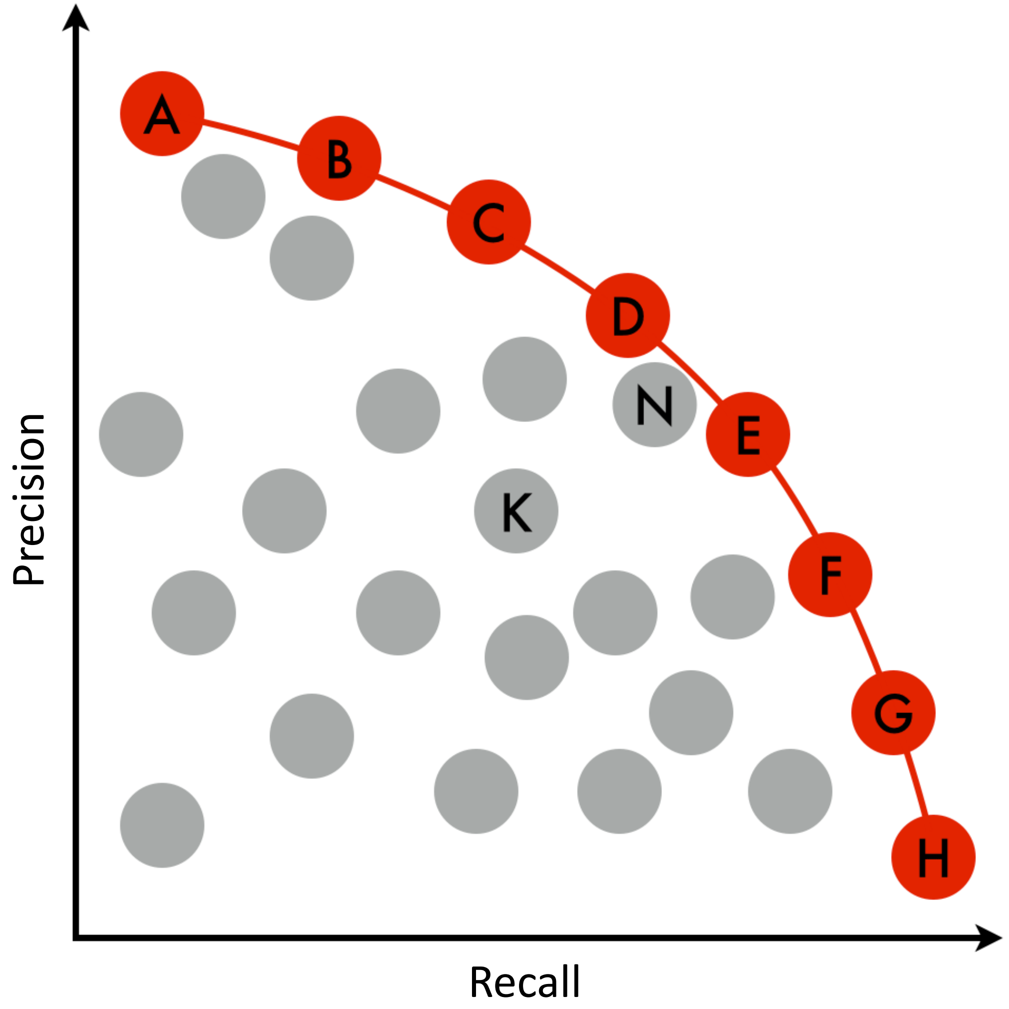
$$\boldsymbol{F_1 score} = 10\%$$

Example #3

		$\boldsymbol{y_i}$		
		$= \boldsymbol{y}$	$\neq \boldsymbol{y}$	
$h(\mathbf{x_i})$	$= \boldsymbol{y}$	25 (TP)	475 (FP)	Positive predictive value, **Precision** 5%
	$\neq \boldsymbol{y}$	25 (FN)	475 (TN)	
		True positive rate, **Recall** 50%	False positive rate 50%	**Accuracy** 50%

$$\boldsymbol{F_1 score} = 9\%$$

Pareto efficiency

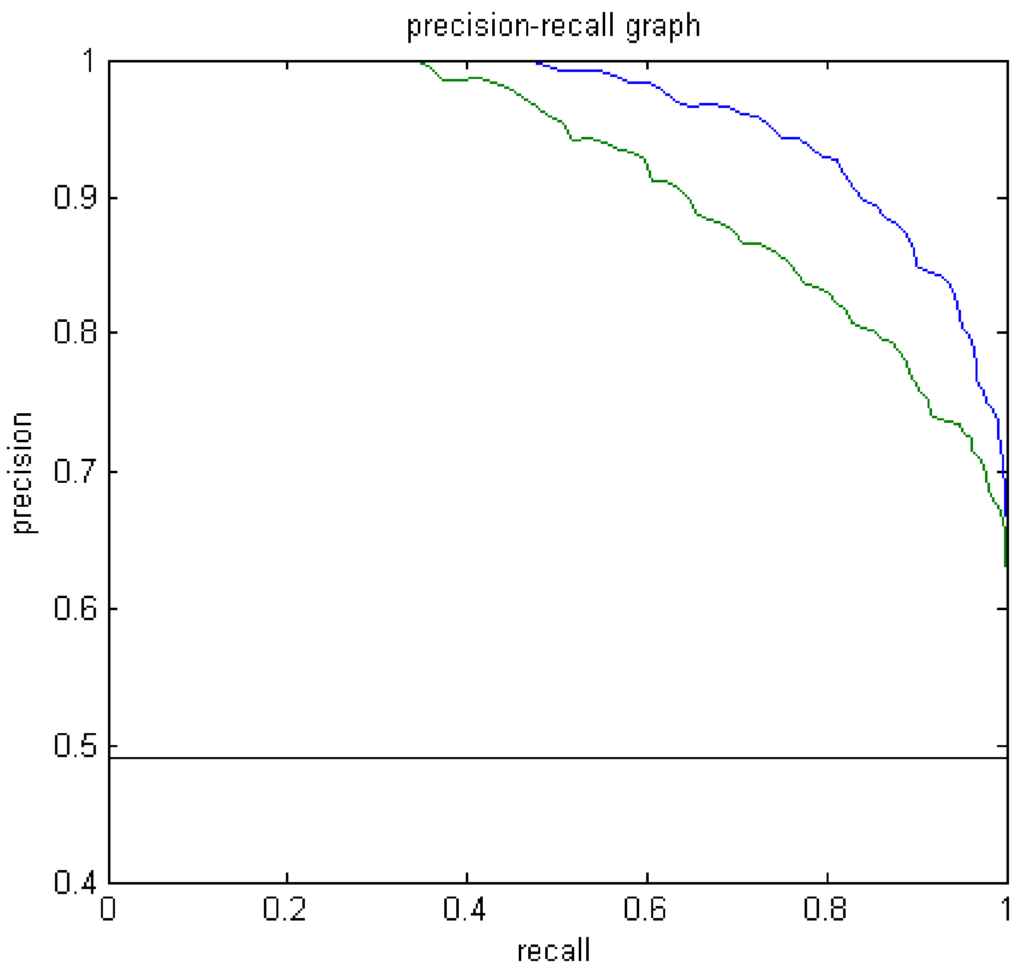


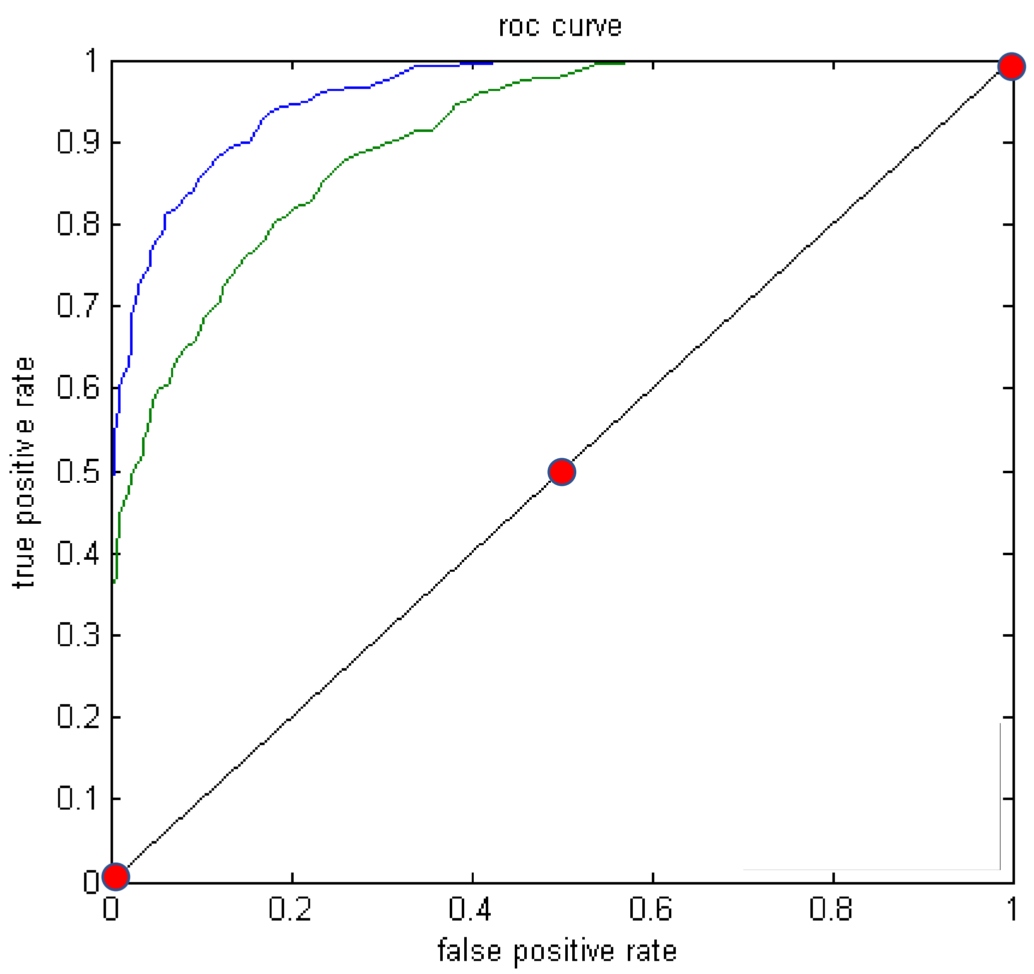
- 1^{st} Pareto frontier

N – 2^{nd} Pareto frontier, K – 3^{rd}

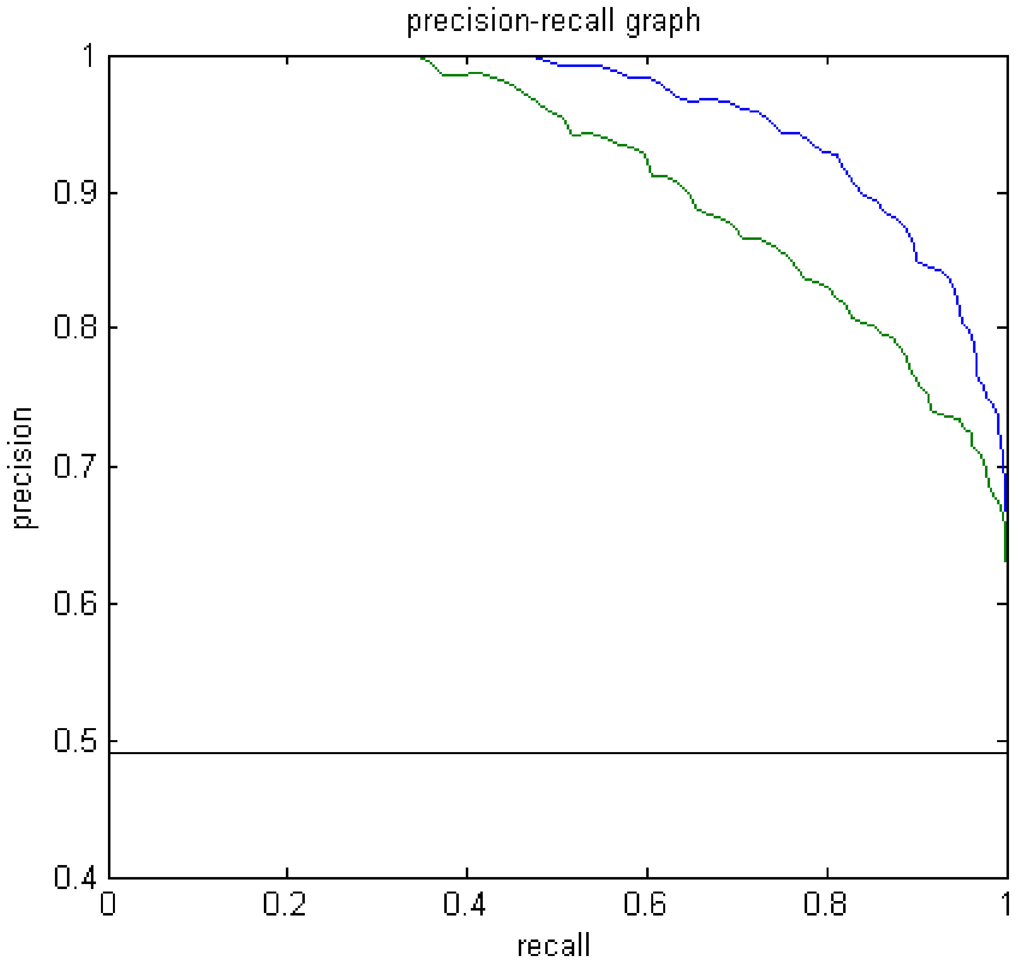
A classifier is Pareto efficient if there is no better classifier in terms of both precision and recall.

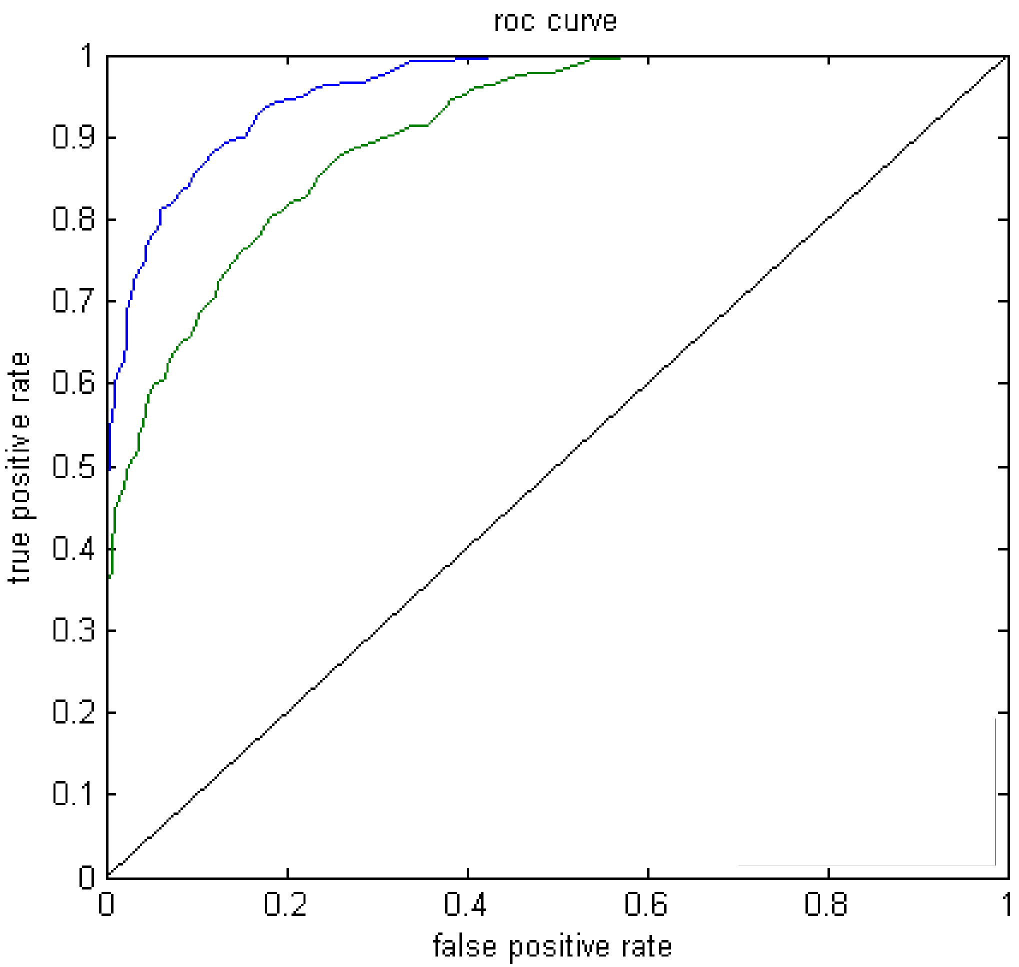
ROC (**R**eceiver **O**perating **C**haracteristic)





ROC (**R**eceiver **O**perating **C**haracteristic)





AUC – **A**rea **U**nder the ROC **C**urve