Learning “Learning to Rank”

Sophie Watson
@sophwats

Learning “Learning to Rank”

Sophie Watson
@sophwats

Search Wikipedia

NETFLIX

WolframAlpha computational intelligence

Enter what you want to calculate or know about

Browse Examples Surprise Me

Google Scholar

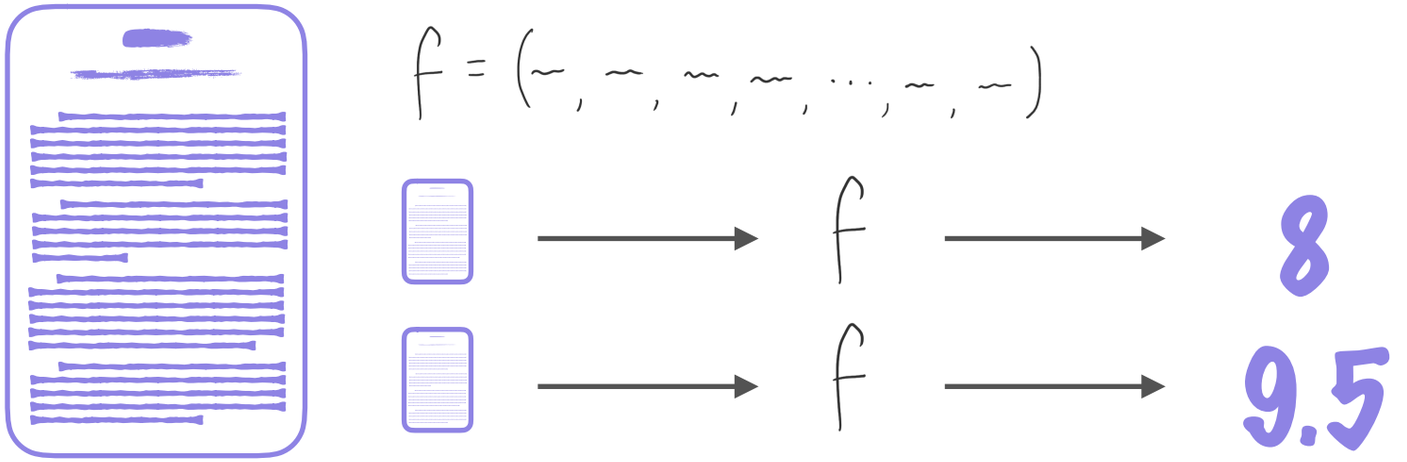
Articles Case law

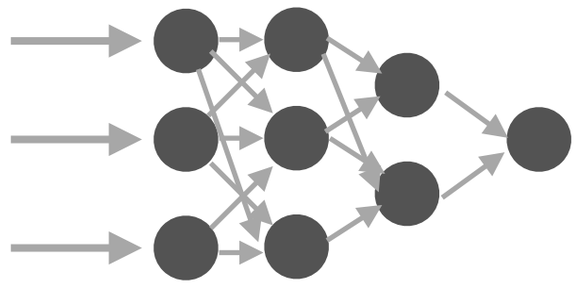
Bing

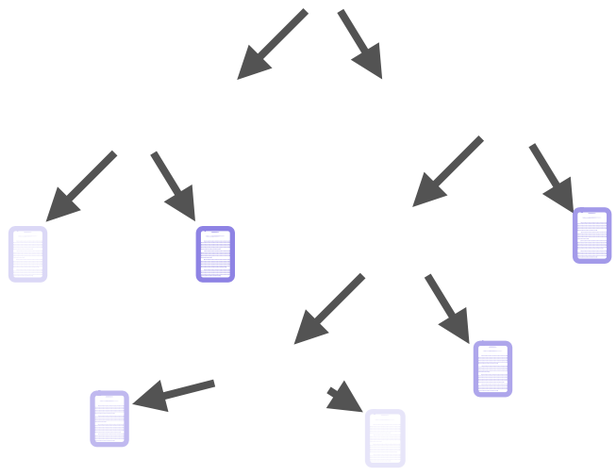
Ask

Search Twitter

amazon



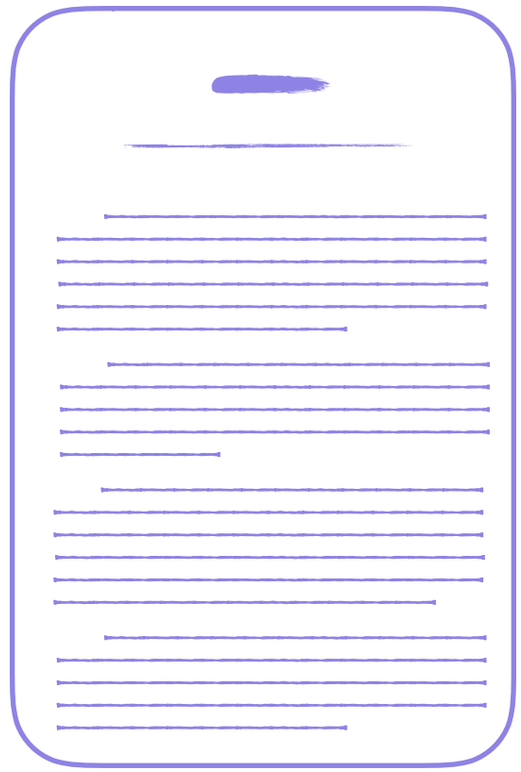




dmlc
XGBoost

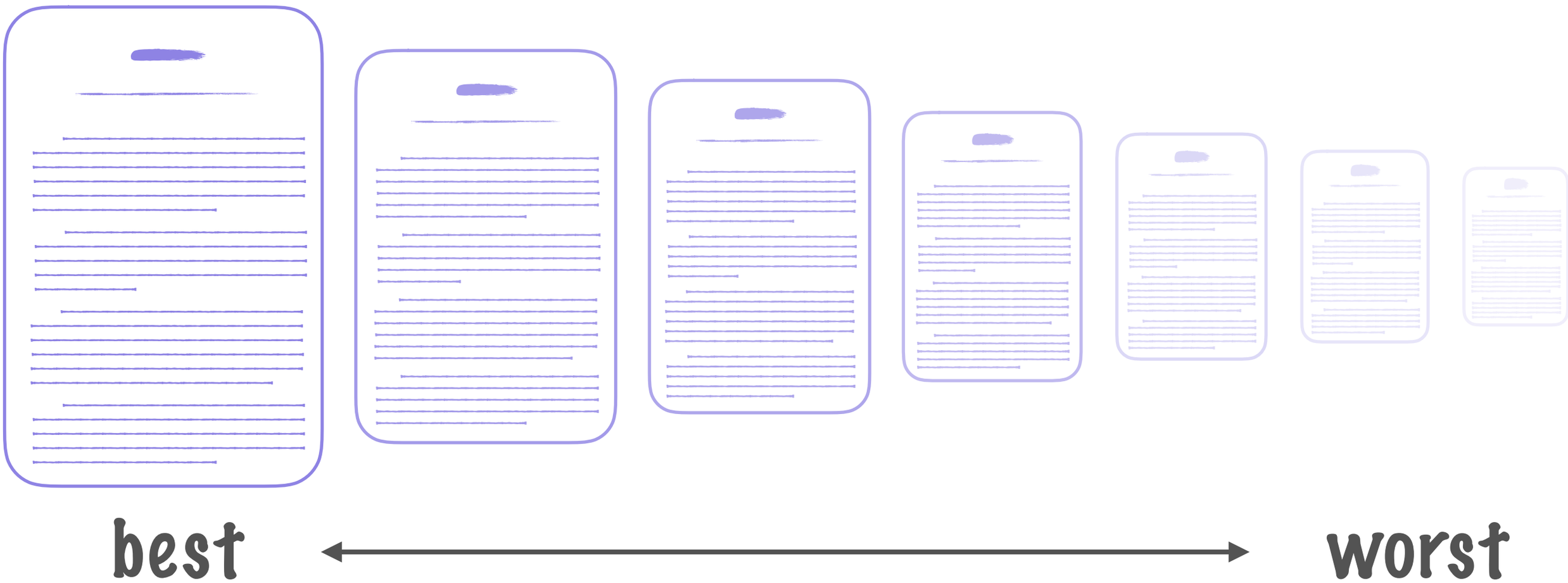
query

$q = (\sim\ \sim\ \sim\sim)$ **query**



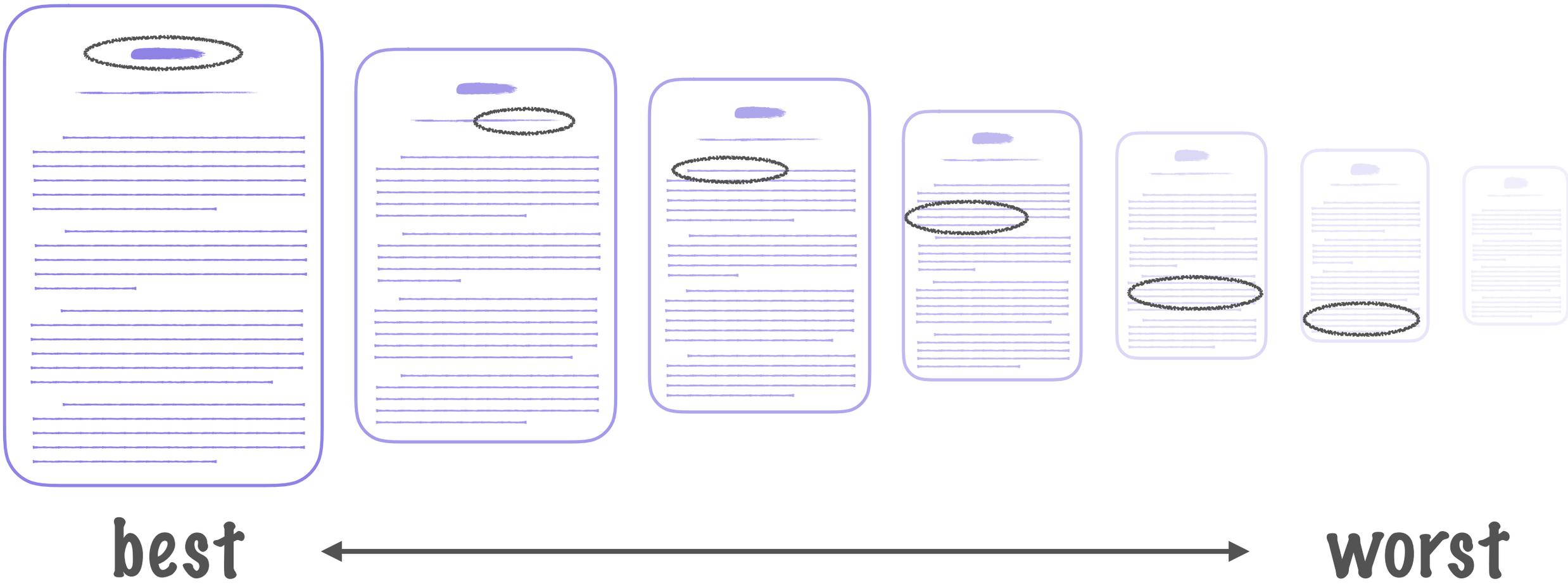
document

$q = (\ldots\ \ldots\ \ldots)$ **query**



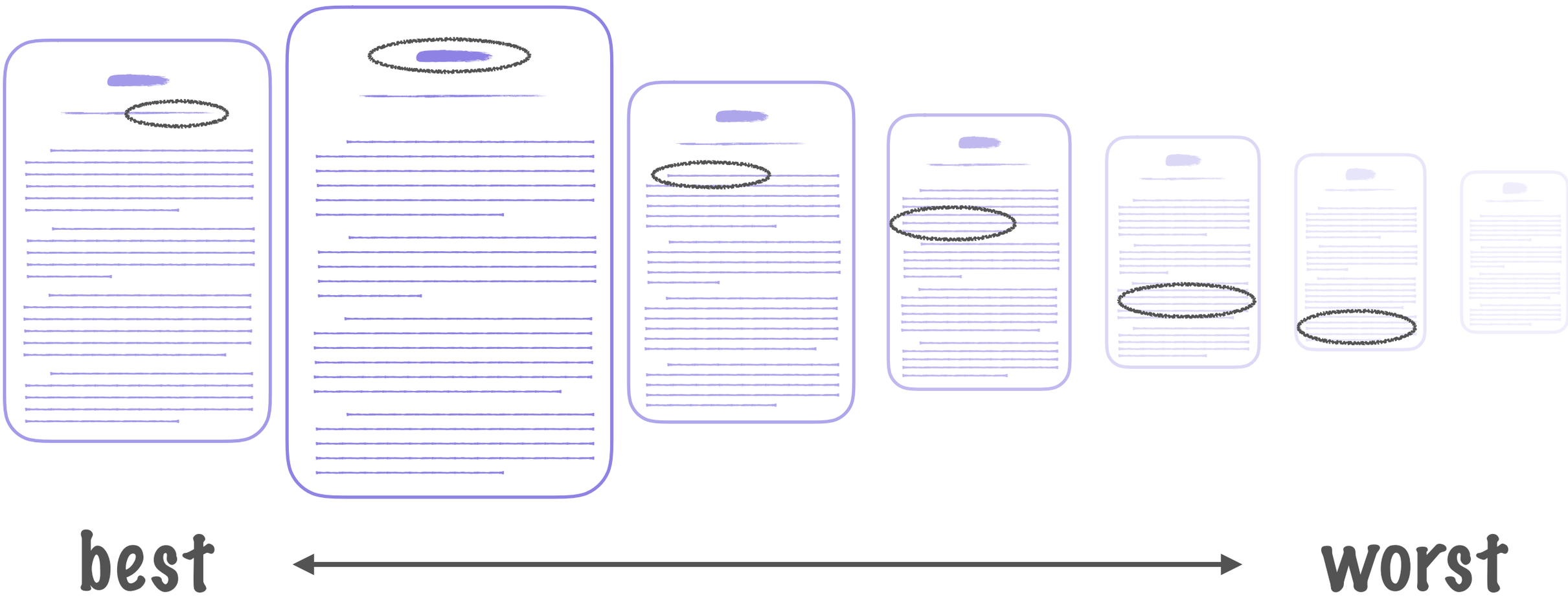
best ⟷ worst

q = (… … …) **query**



best ⟷ worst

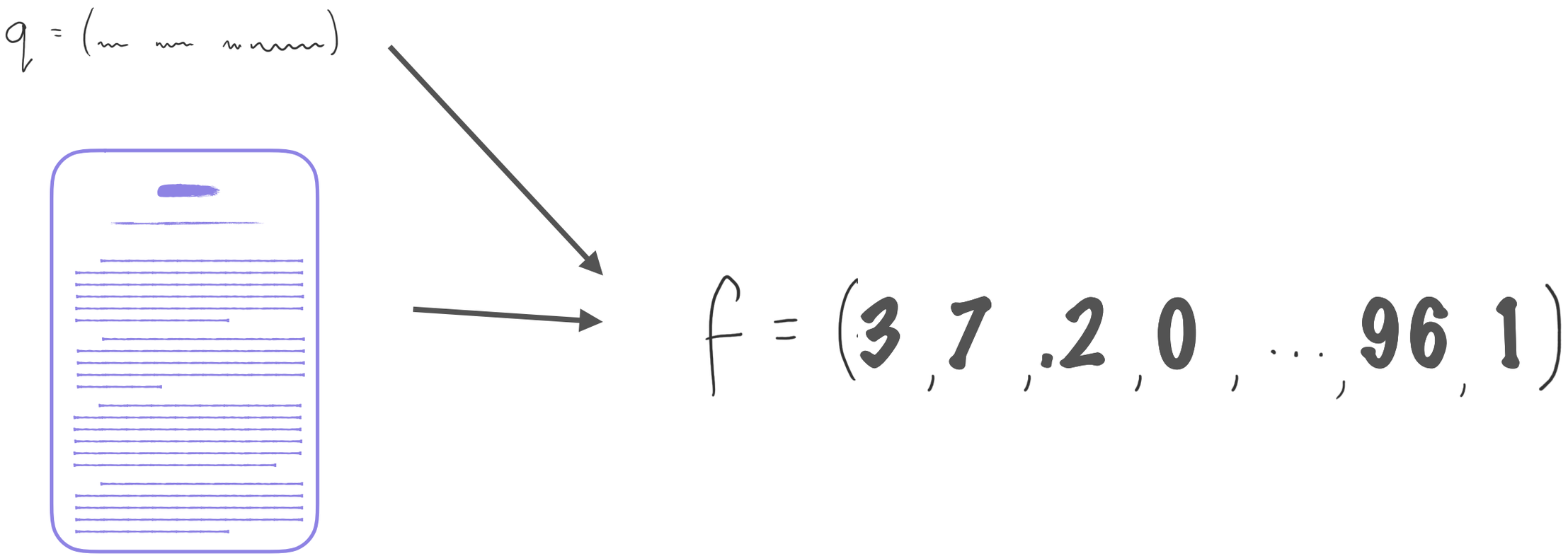
q = (… … …) **query**

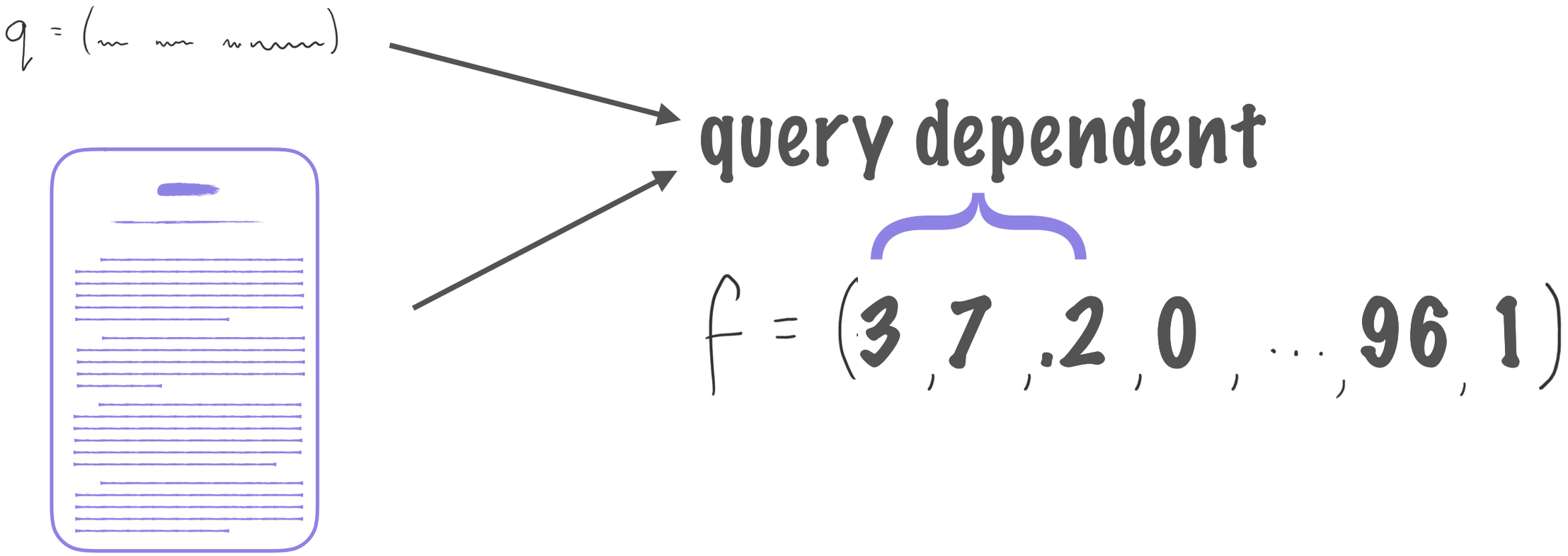


best ⟷ worst

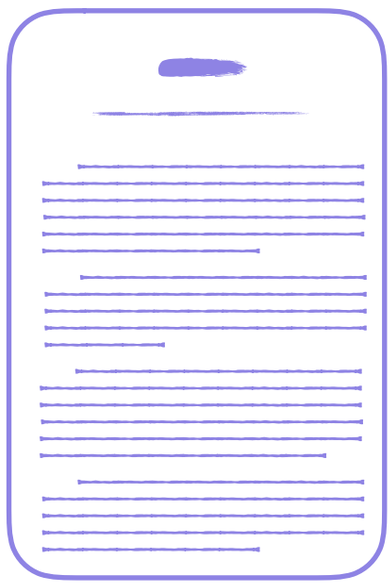
$q = (\_\_ \_\_ \_\_\_\_)$ **query**

$q = (\sim\sim\sim\sim)$


q = (... ... .)
f = (3, 7, .2, 0, ..., 96, 1)


q = (... ... ...)
query dependent
f = (3, 7, .2, 0, ..., 96, 1)

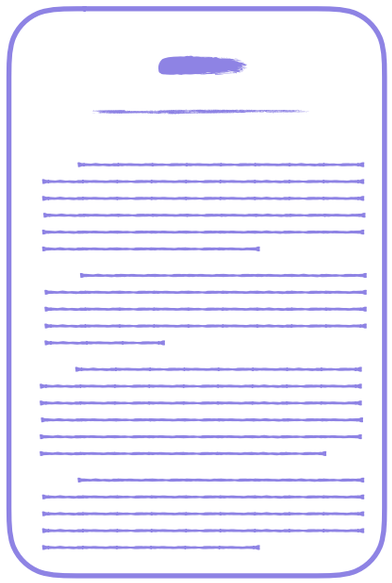
$q = (\_\_\_\_\_)$



query independent

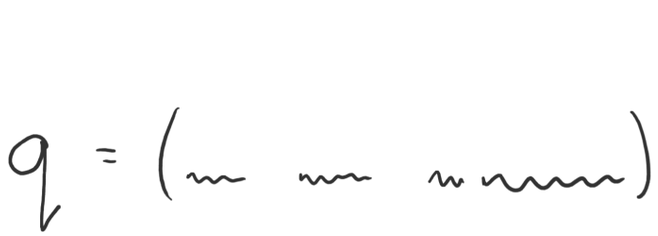
$f = (3, 7, .2, 0, \ldots, 96, 1)$

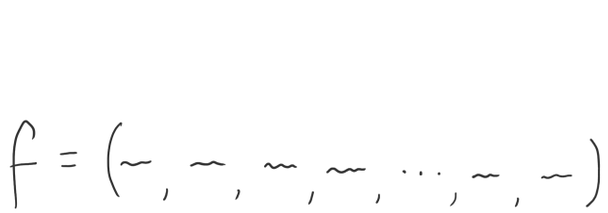
$q = (\ldots\ \ldots\ \ldots)$

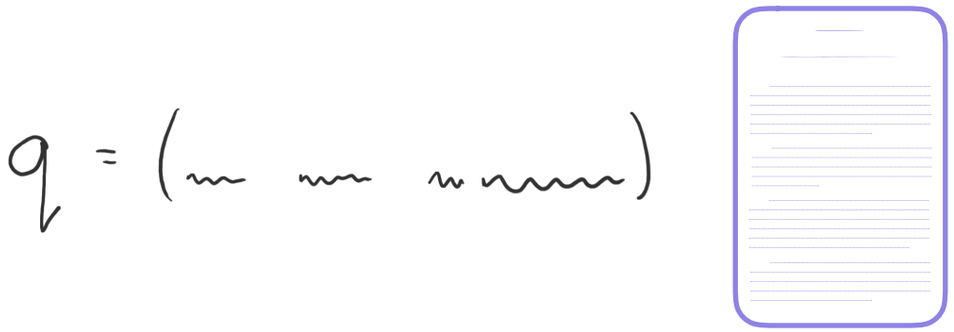


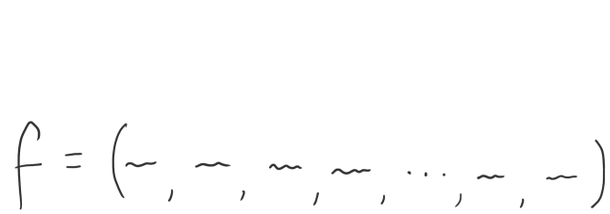
query level

$f = (3, 7, .2, 0, \ldots, 96, 1)$



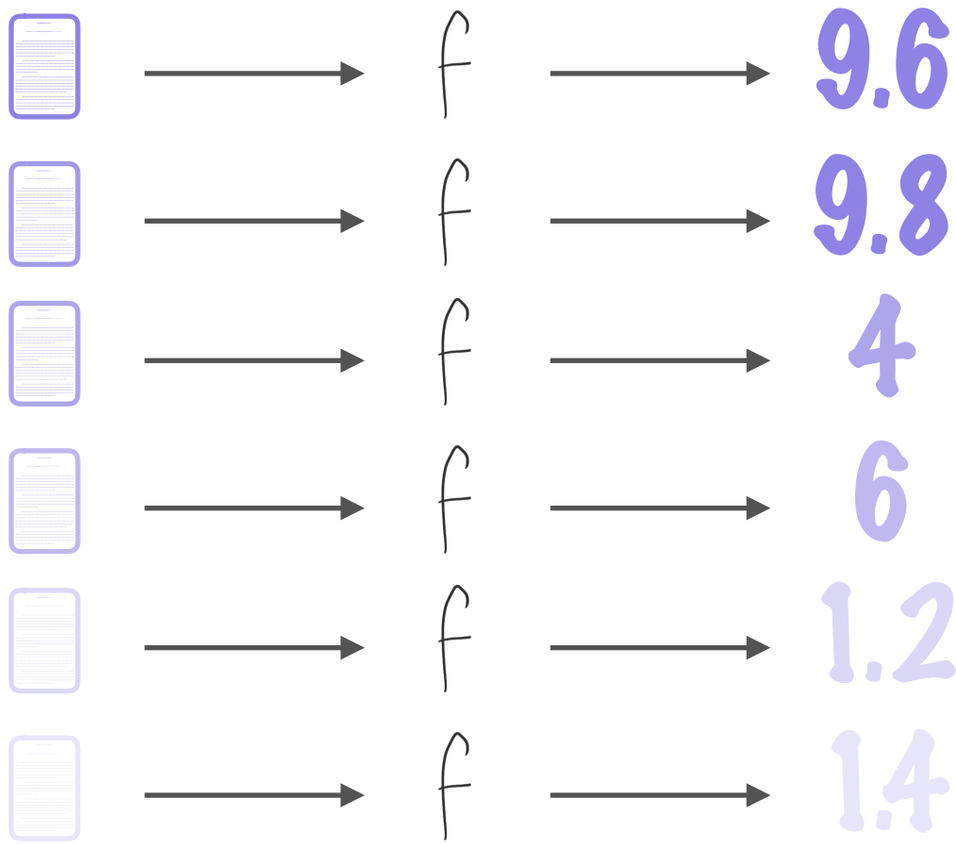




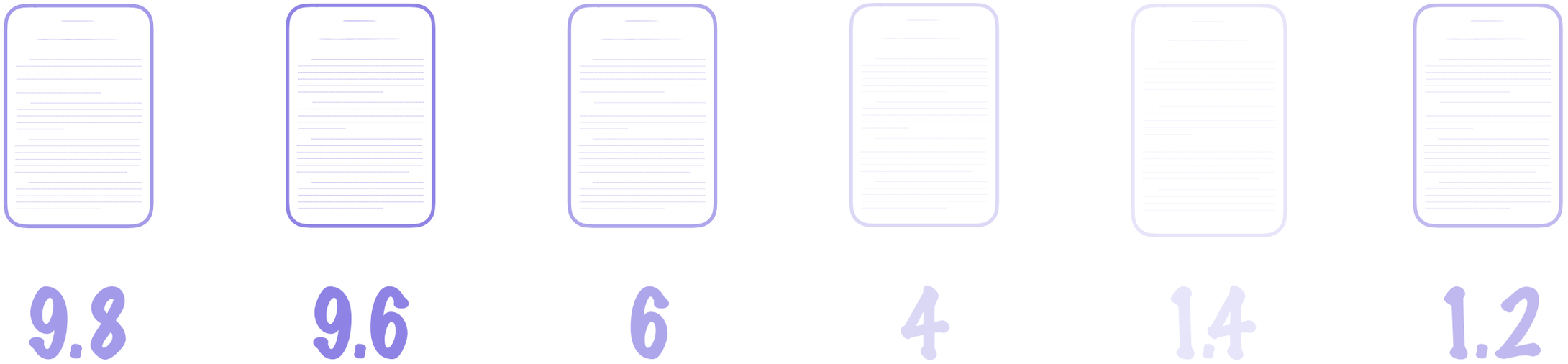


score

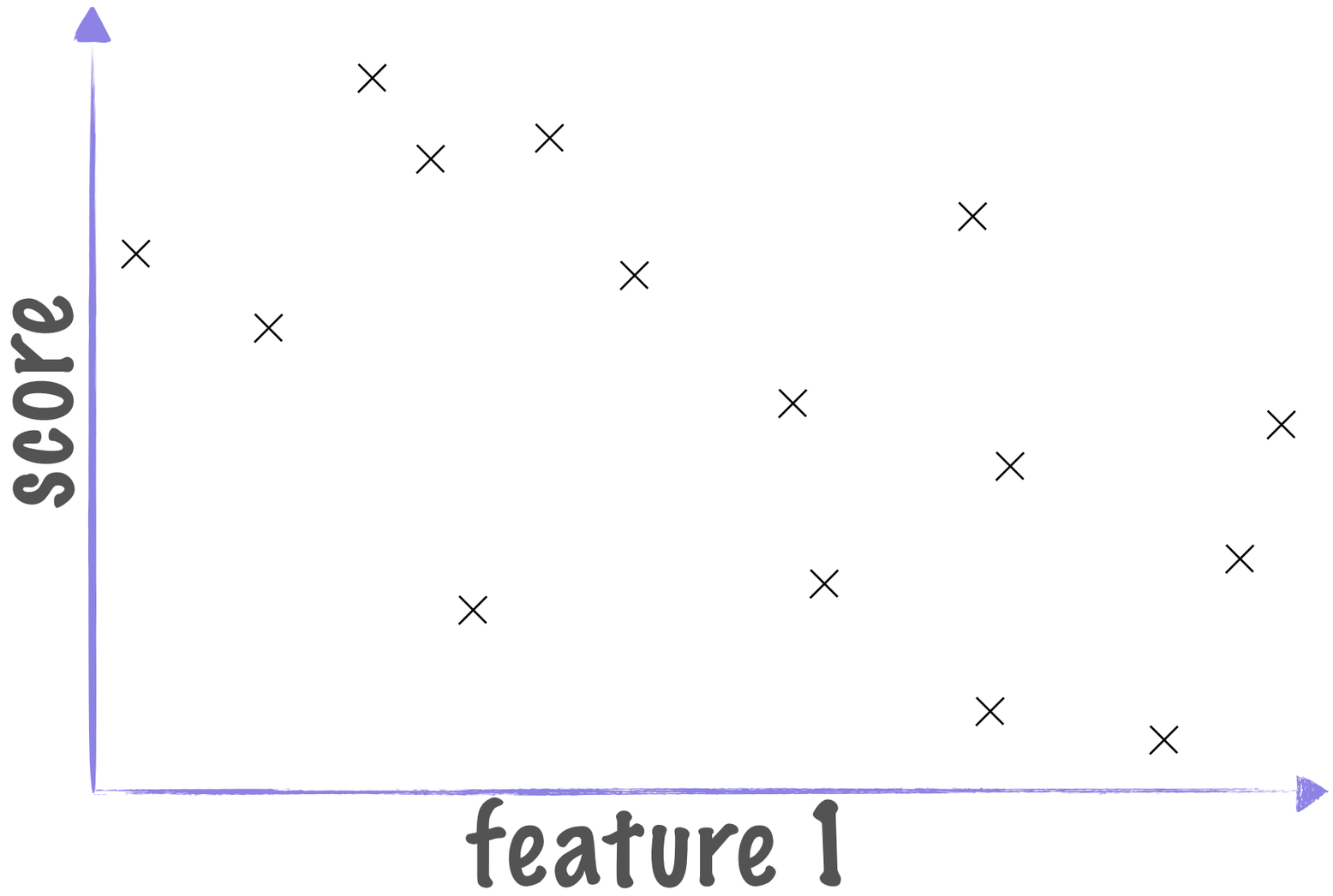
Pointwise Ranking

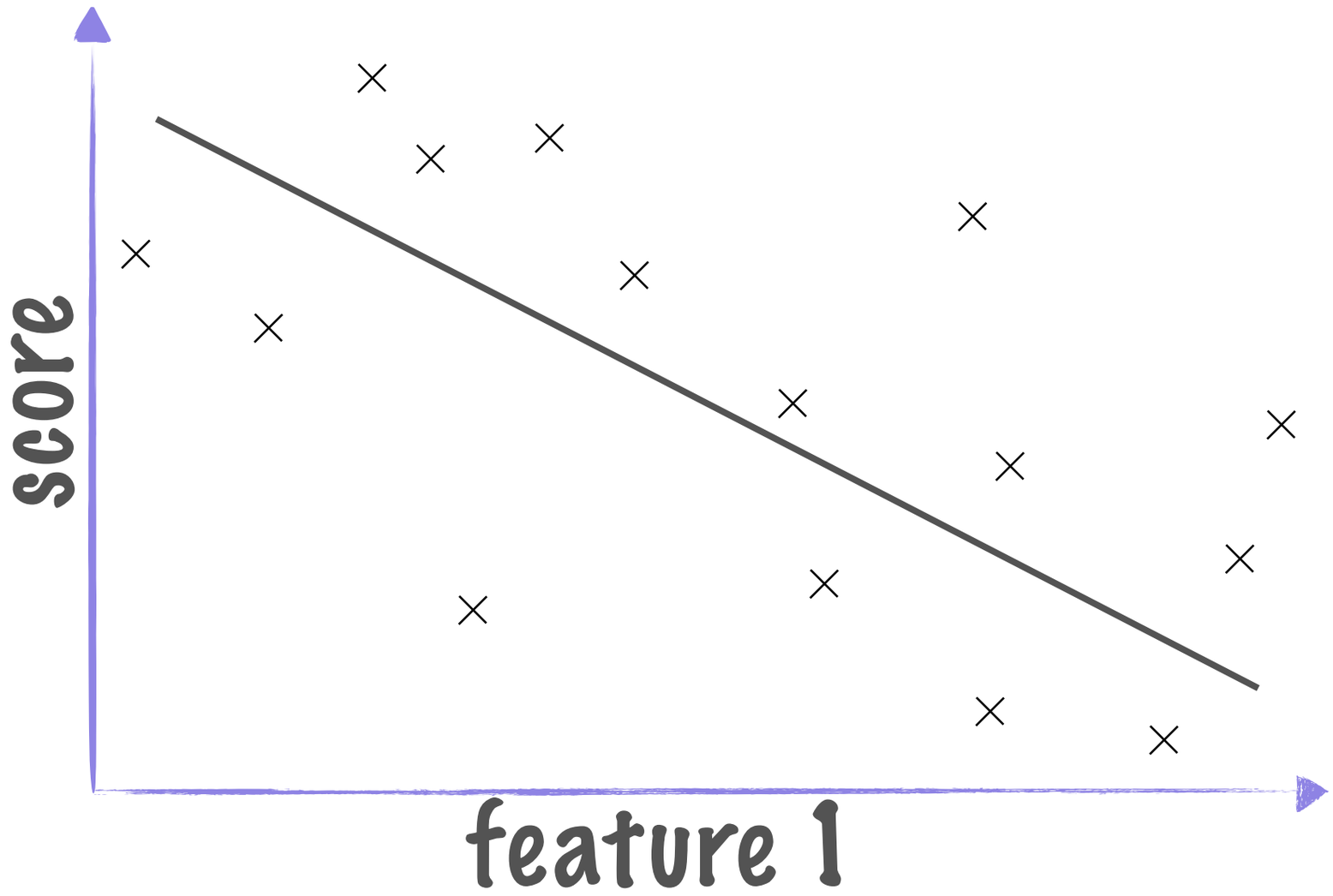


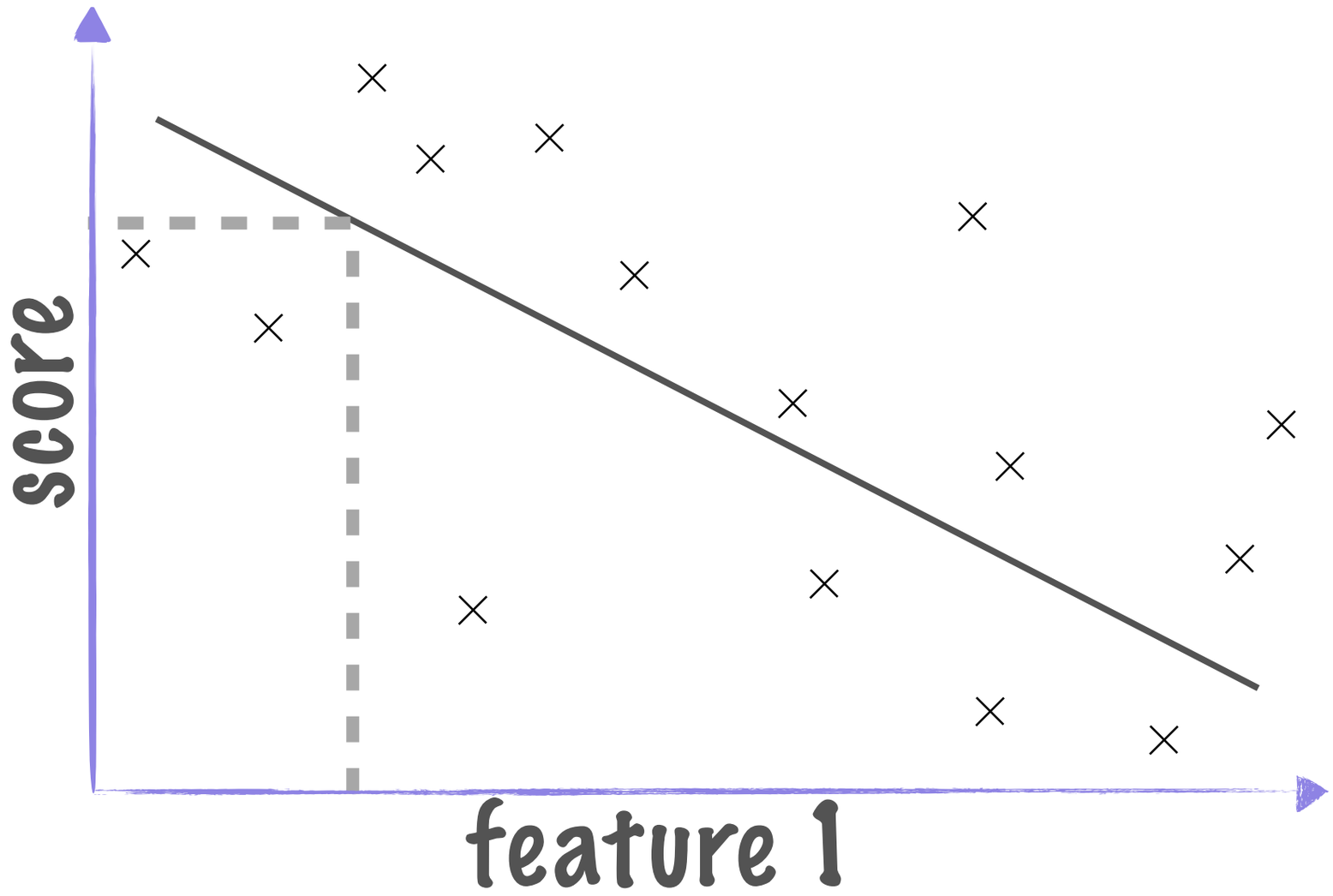
Pointwise Ranking

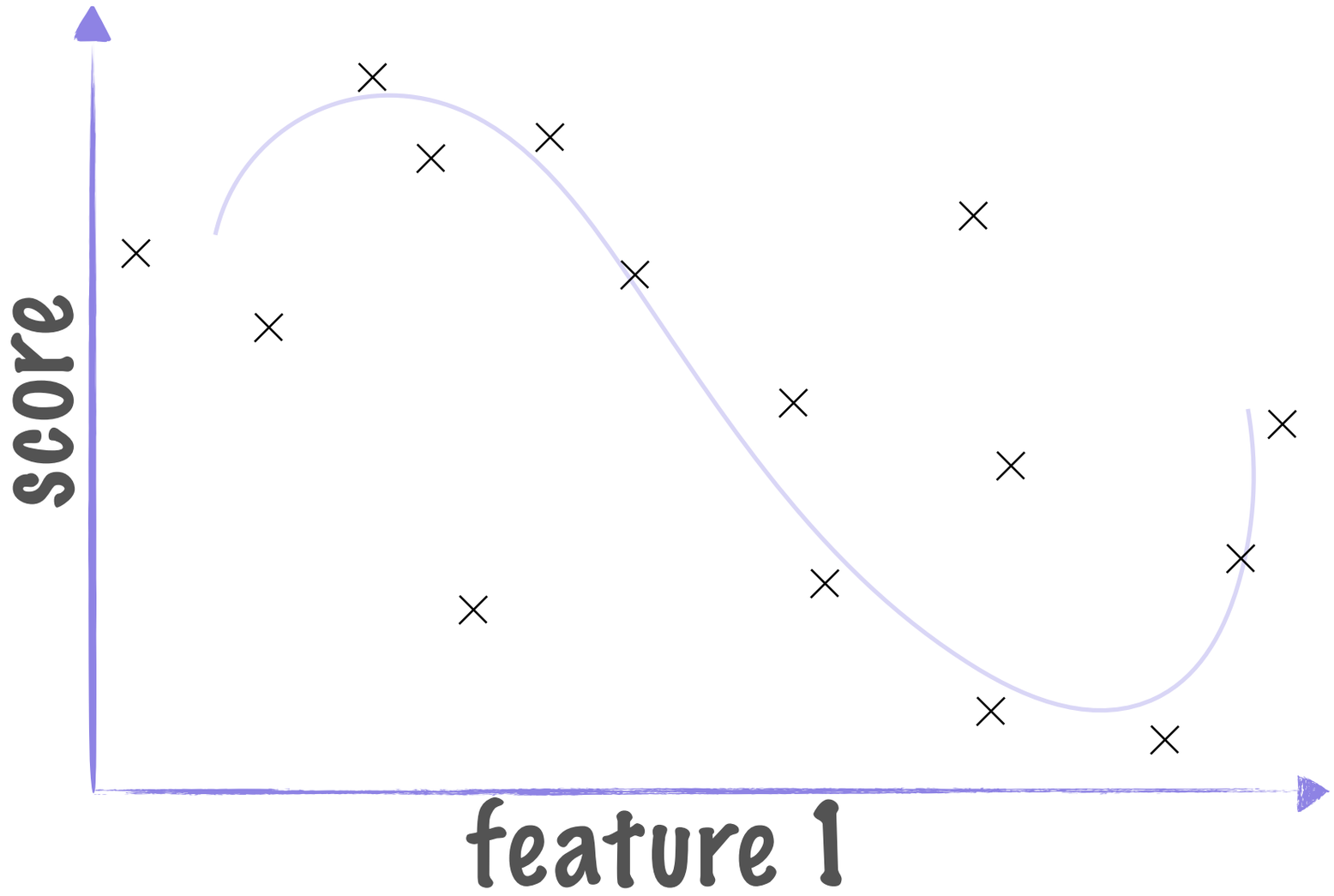


best ⟷ worst

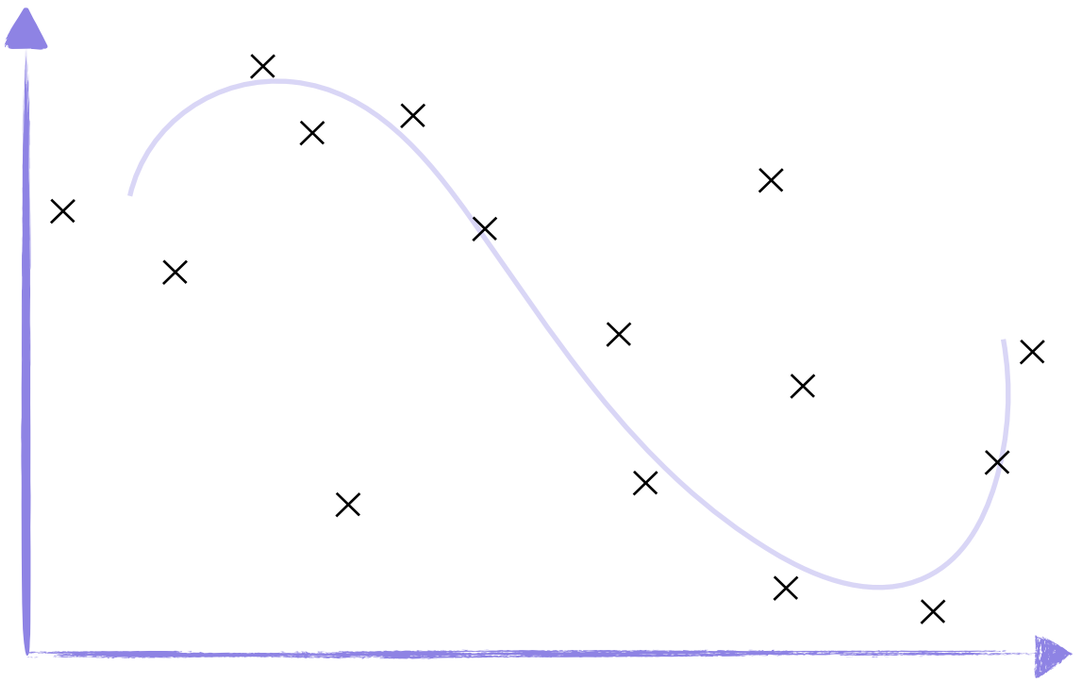

score
feature 1


score
feature 1


score
feature 1


score
feature 1

$$\sum_{\text{docs} \in \text{testing set}} \left(\frac{\text{true score} - \text{predicted score}}{|\text{testing set}|} \right)^2$$



9.8 9.6 6 4 1.4 1.2

predicted rank	1	2	3	4	5	6
true rank	2	1	3	5	6	4

predicted rank	1	2	3	4	5	6
true rank	2	1	3	5	6	4

predicted rank	2	1	3	4	5	6
true rank	1	2	3	5	6	4

swaps: 1

predicted rank 2 1 3 4 6 5

true rank 1 2 3 5 4 6

swaps:2

predicted rank 2 1 3 6 4 5

true rank 1 2 3 4 5 6

swaps:3

true relevance	9.8	9.6	6	4	1.4	1.2

Normalised Discounted Cumulative Gain

predicted rank	1	2	3	4	5	6
true rank	2	1	3	5	6	4
true score	9.6	9.8	6	1.4	1.2	4.0

~~Normalised~~ Discounted Cumulative Gain

predicted rank	1	2	3	4	5	6
true rank	2	1	3	5	6	4
true score	9.6	9.8	6	1.4	1.2	4.0

$$\sum_{\text{docs} \in \text{testing set}} \frac{\text{true score}}{\log_2(\text{predicted rank} + 1)}$$

~~Normalised~~ Discounted Cumulative Gain

predicted rank	1	2	3	4	5	6
true rank	2	1	3	5	6	4
true score	9.6	9.8	6	1.4	1.2	4.0

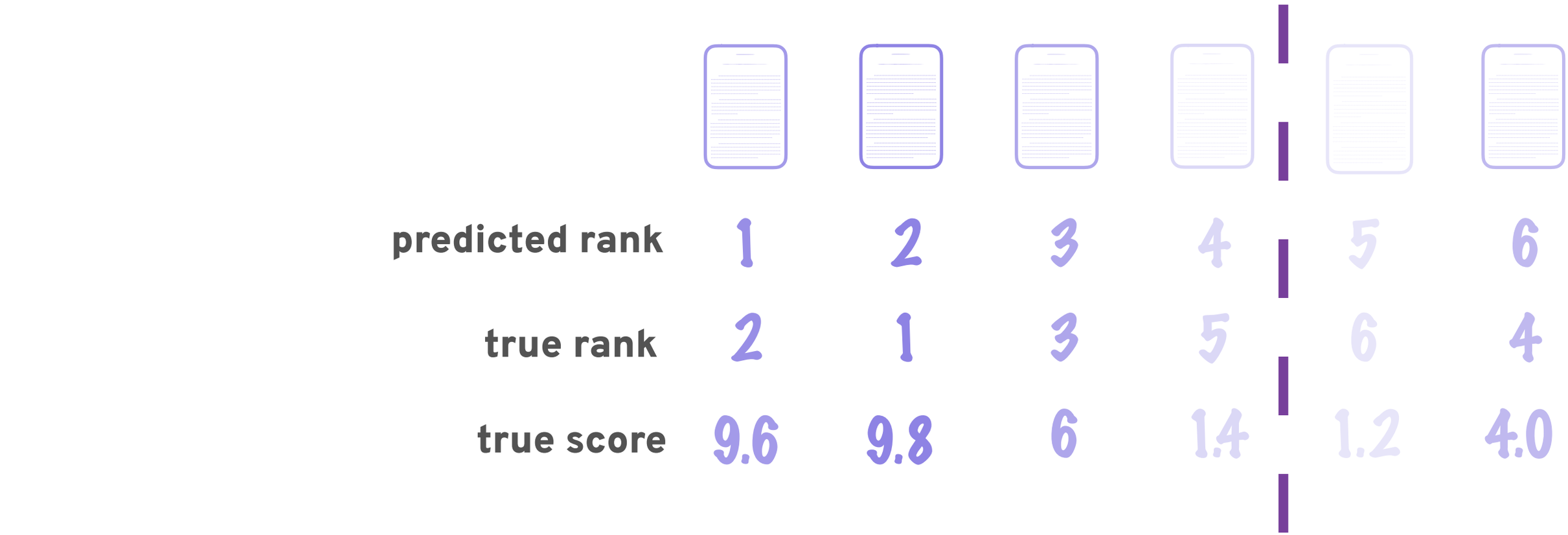
$$\sum_{\text{docs} \in \text{testing set}} \frac{\text{true score}}{\log_2(\text{predicted rank} + 1)} = \frac{9.6}{\log_2(1+1)} + \frac{9.8}{\log_2(2+1)} + \dots + \frac{4}{\log_2(6+1)}$$

Normalised Discounted Cumulative Gain

predicted rank	1	2	3	4	5	6
true rank	2	1	3	5	6	4
true score	9.6	9.8	6	1.4	1.2	4.0

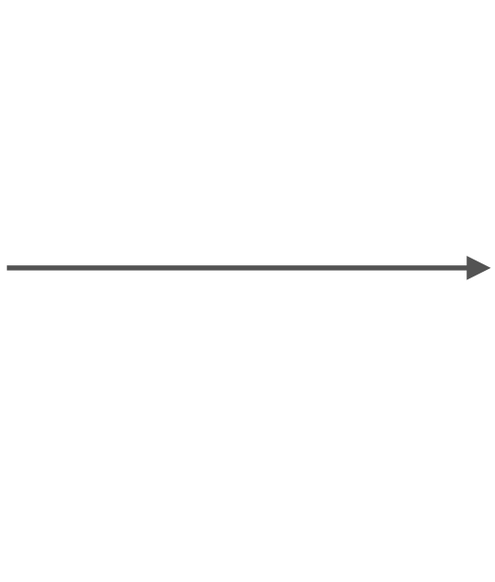
$$nDCG = \frac{\text{discounted cumulative gain}}{\text{optimal discounted cumulative gain}}$$

Normalised Discounted Cumulative Gain



$$nDCG@4 = \frac{\text{discounted cumulative gain for top 4}}{\text{optimal discounted cumulative gain for top 4}}$$

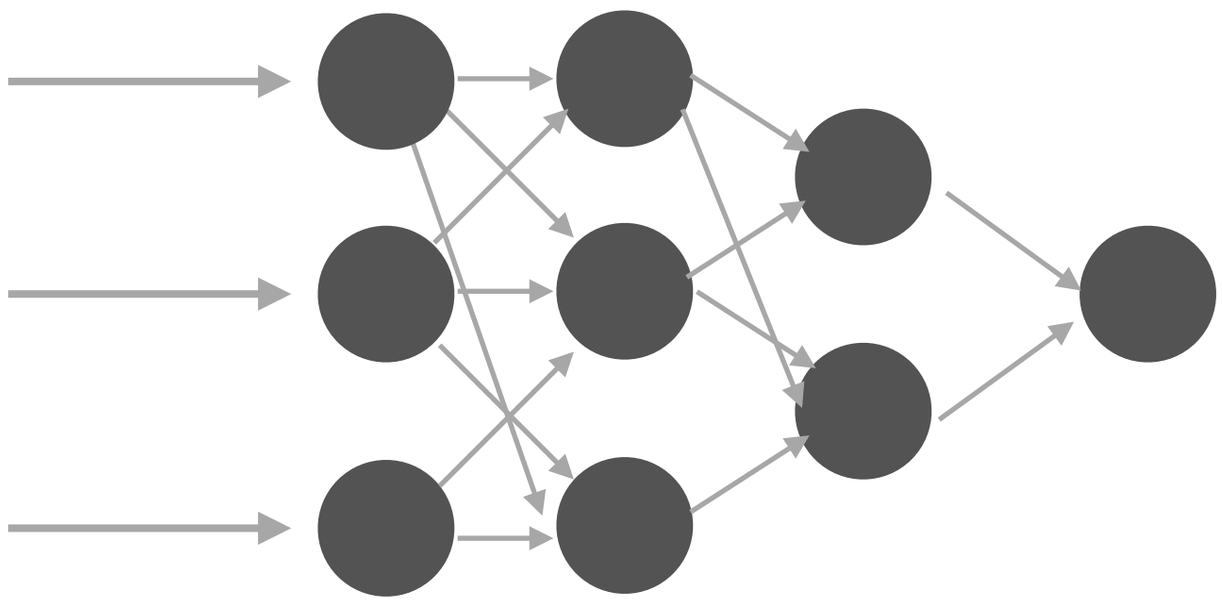
Pairwise Ranking

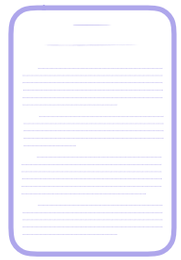


more relevant

less relevant

RankNet

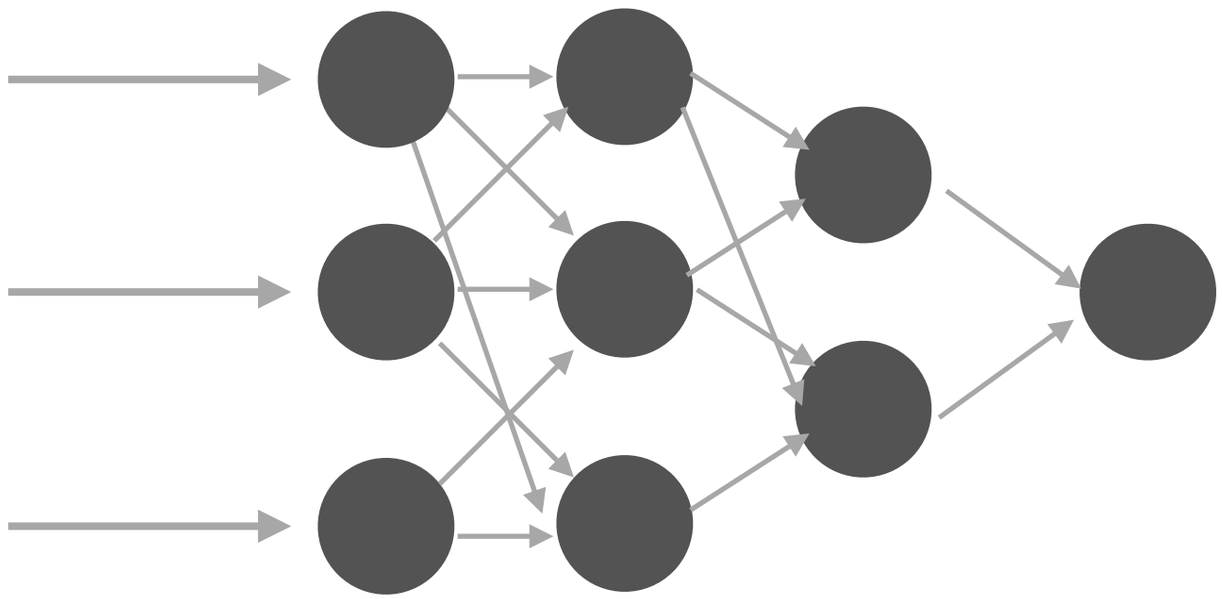




more relevant

less relevant

RankNet

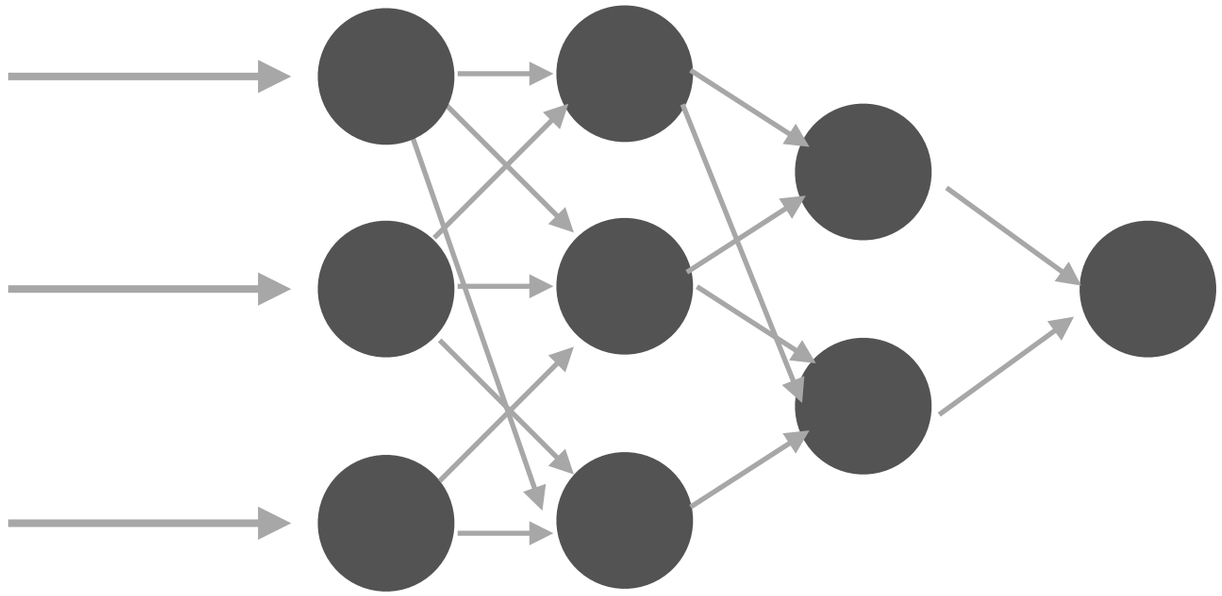


Even
more relevant

Even
less relevant

RankNet

LambdaRank



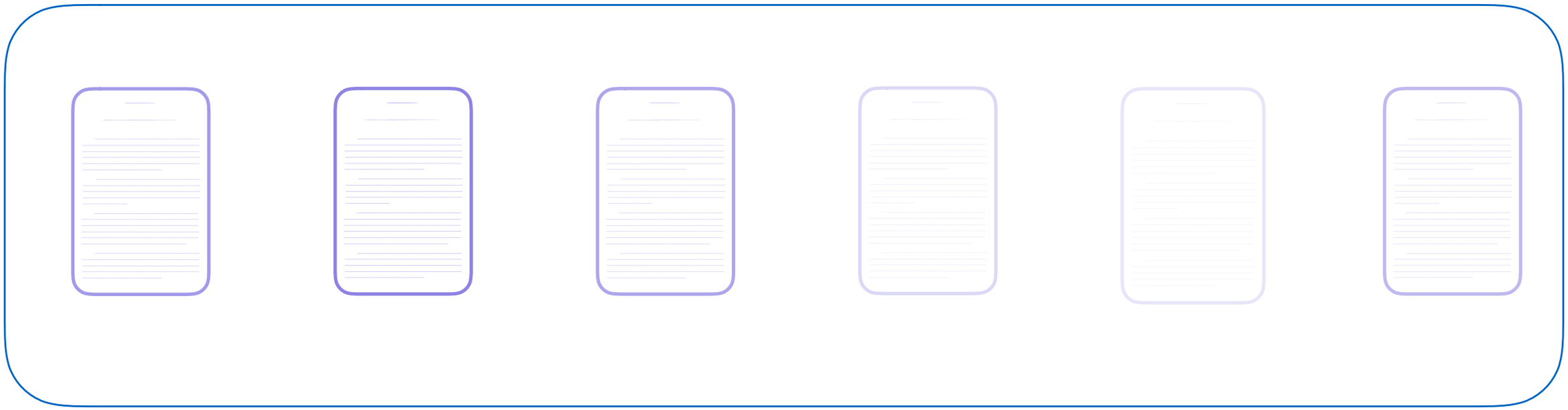
more relevant,
in relation to ndcg

less relevant,
in relation to ndcg

LambdaRank

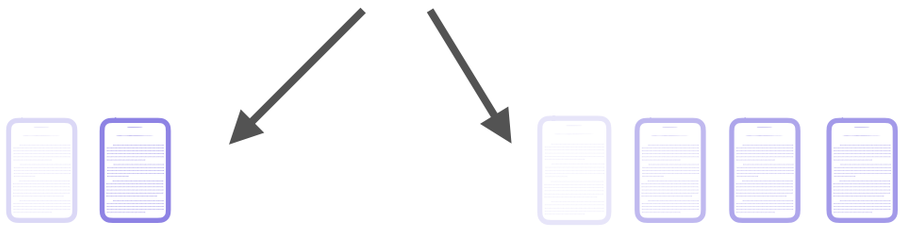
LambdaRank

Listwise Ranking

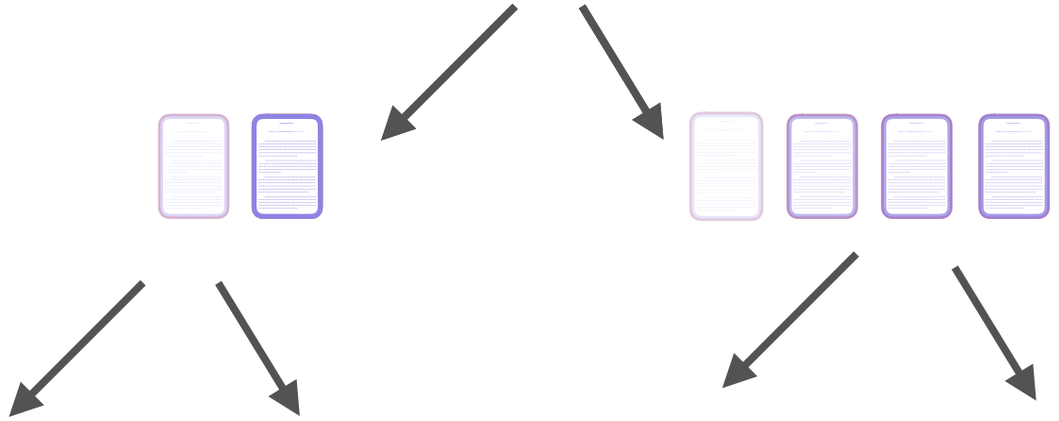


LambdaMART

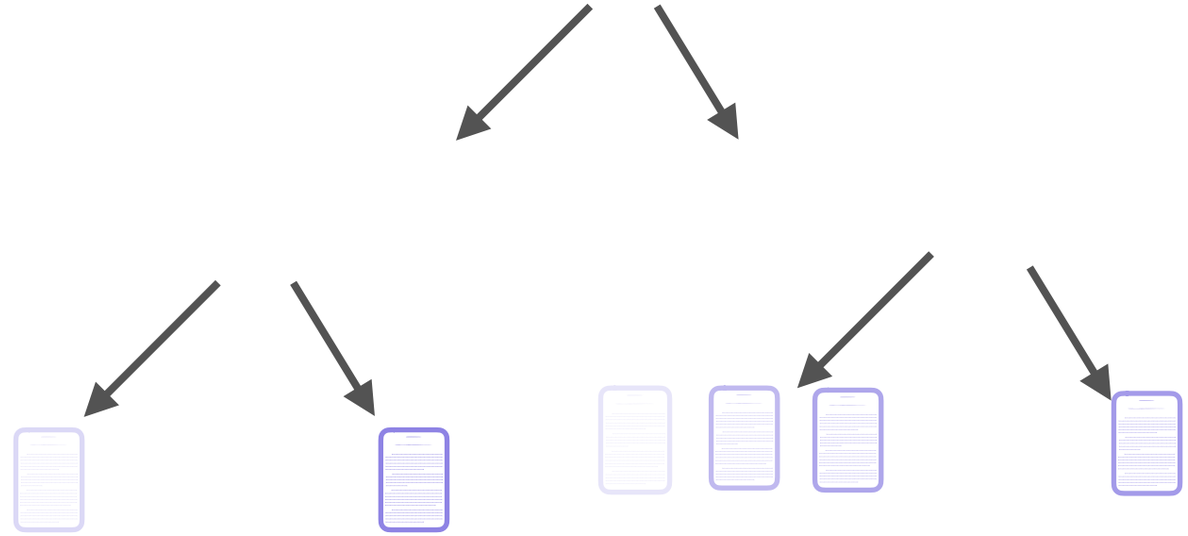
LambdaMART



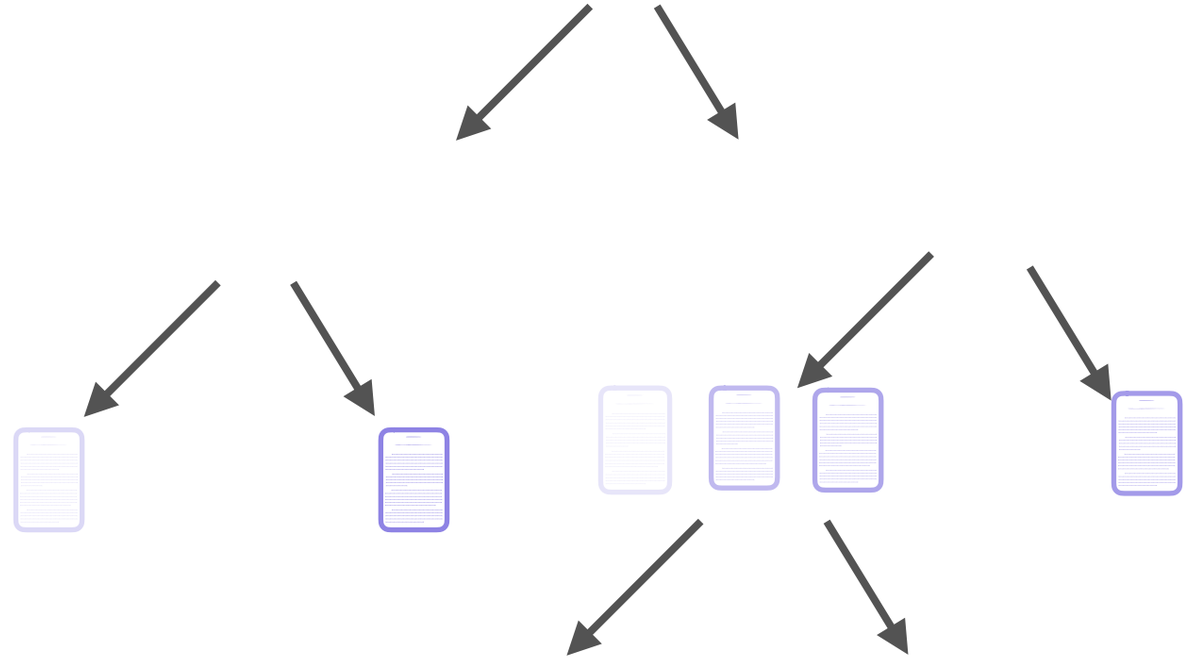
LambdaMART



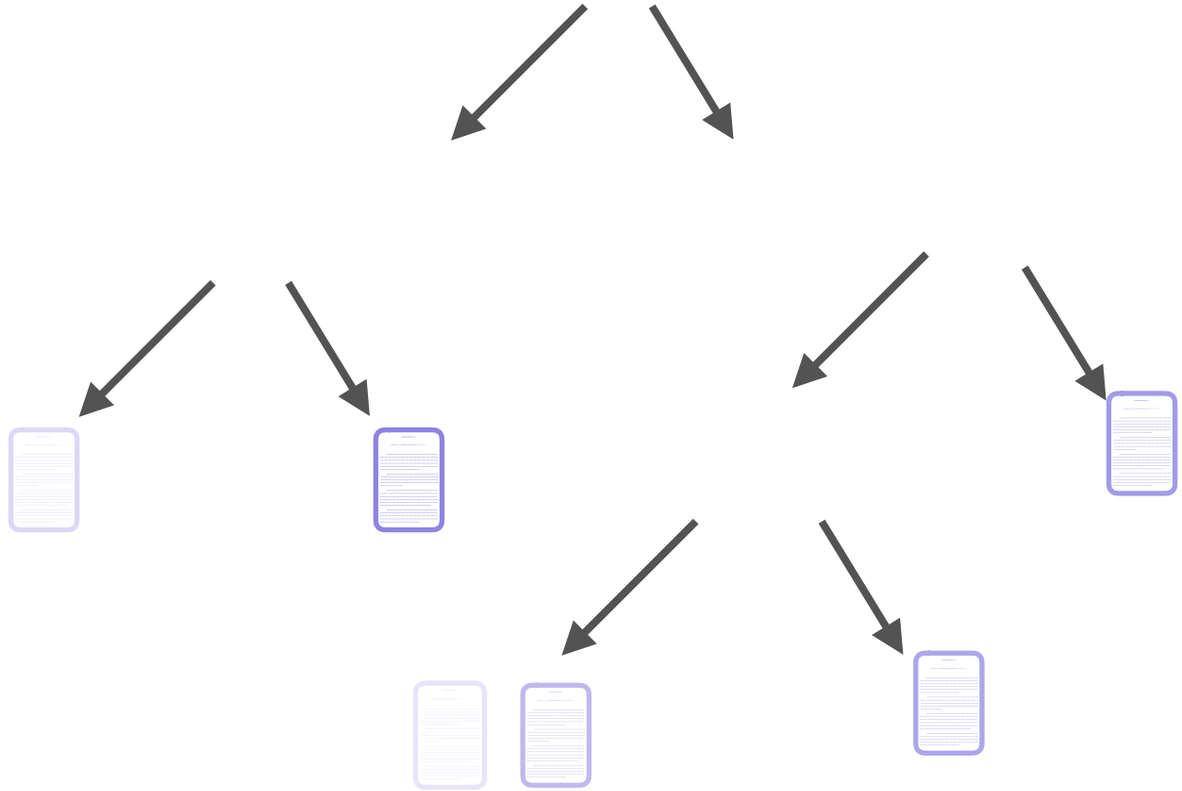
LambdaMART



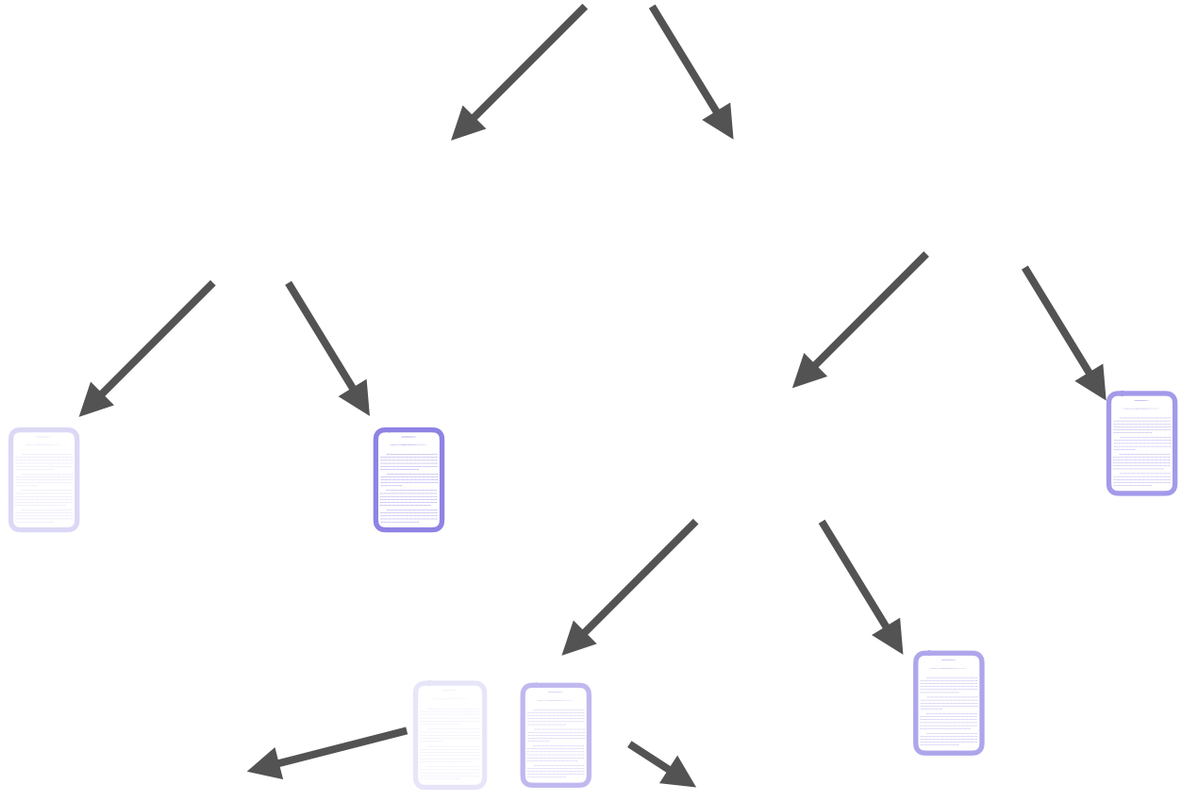
LambdaMART



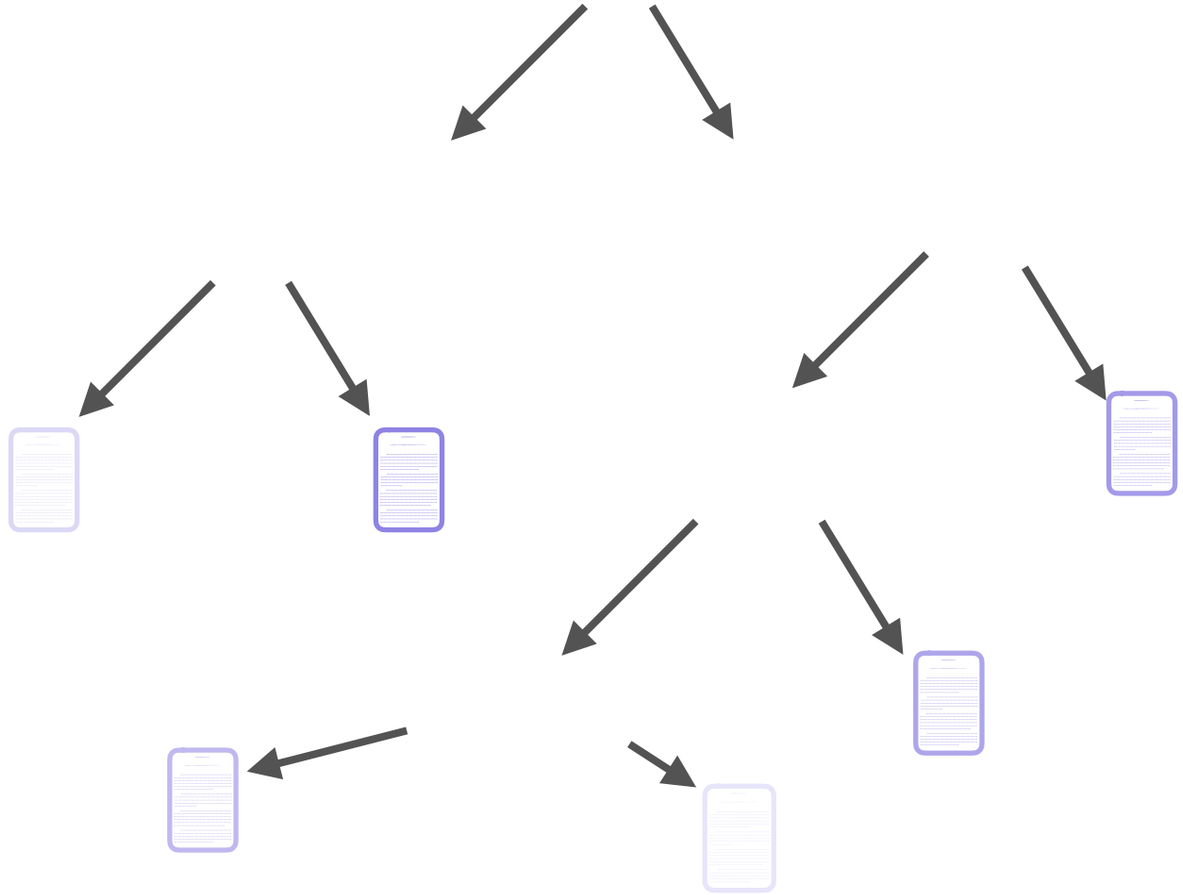
LambdaMART

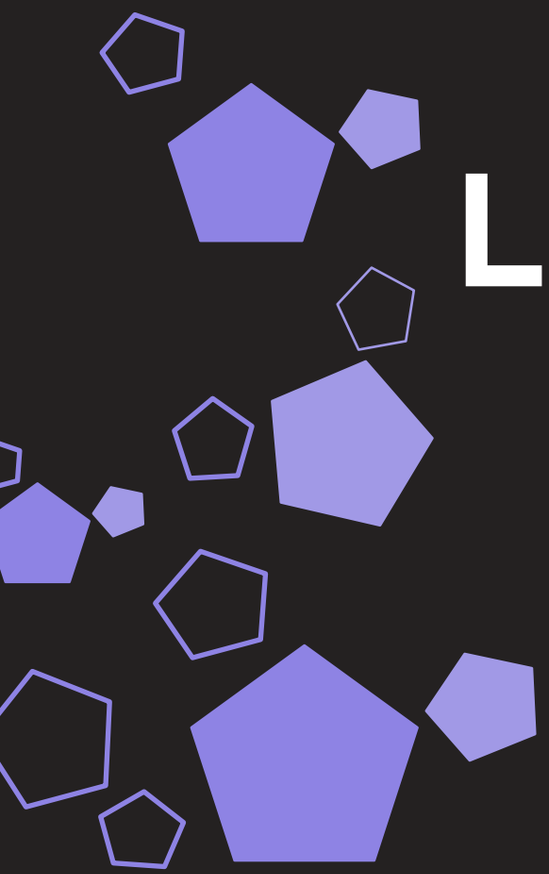


LambdaMART



LambdaMART





Learning to Rank in Action

MSLR-WEB10K

- Open Source
- 10,000 queries
- relevance values from 0 to 4
- schema: relevance, query id, feature vector
- training, testing and validation data sets

2 qid:13 1:2 2:0 3:2 4:1 5:2 6:1 7:0 8:1 9:0.5 …

133:7 134:0 135:0 136:0

relevance

2 qid:13 1:2 2:0 3:2 4:1 5:2 6:1 7:0 8:1 9:0.5 …
133:7 134:0 135:0 136:0

relevance

2 qid:13 1:2 2:0 3:2 4:1 5:2 6:1 7:0 8:1 9:0.5 …
133:7 134:0 135:0 136:0

query id

relevance

features

2 qid:13 1:2 2:0 3:2 4:1 5:2 6:1 7:0 8:1 9:0.5 …

133:7 134:0 135:0 136:0

query id

```
from pyspark.ml.regression import LinearRegression
```

```
lr = LinearRegression(featuresCol='featurevec', labelCol='scoreval')
```

```
predicted = lr.transform(testset)
```

dmlc

XGBoost

```
param = {'max_depth':6, 'eta':0.3, 'silent':0, 'objective':'rank:pairwise', 'num_round':10}
```

RankNet = 'rank:map'

```
param = {'max_depth':6, 'eta':0.3, 'silent':0, 'objective':'rank:pairwise', 'num_round':10}
```

RankNet = 'rank:map'

LamdaRank = 'rank:pairwise'

```
param = {'max_depth':6, 'eta':0.3, 'silent':0, 'objective':'rank:pairwise', 'num_round':10}
```

RankNet = 'rank:map'

LamdaRank = 'rank:pairwise'

LambdaMART = 'rank:ndcg'

```
param = {'max_depth':6, 'eta':0.3, 'silent':0, 'objective':'rank:pairwise', 'num_round':10}
```

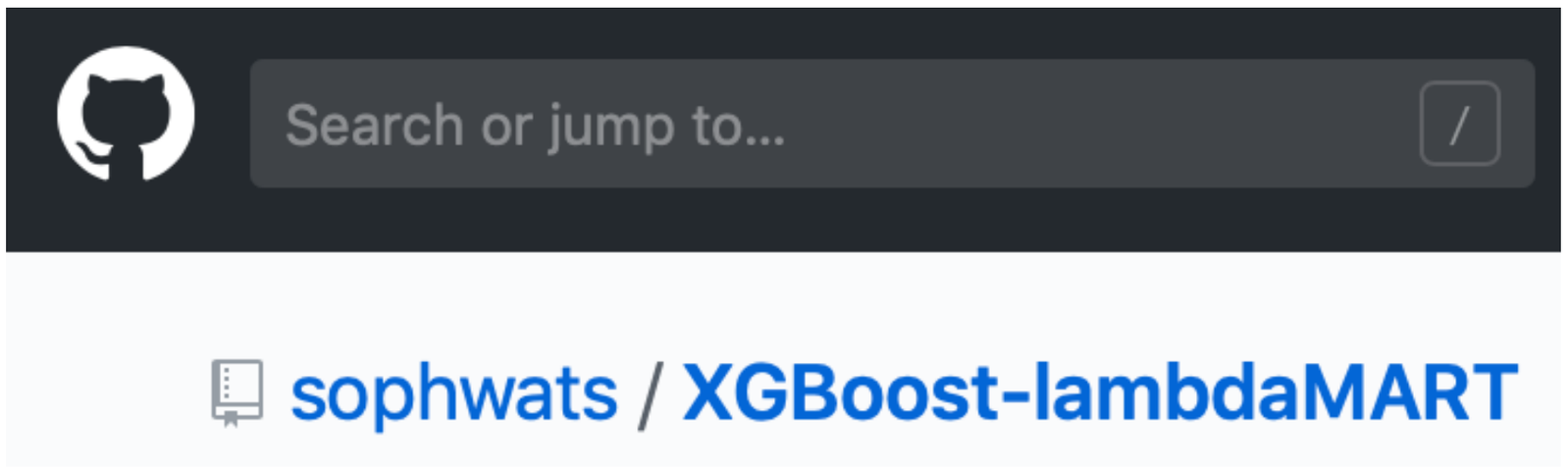
```
model = xgb.train(param, training_data)
```

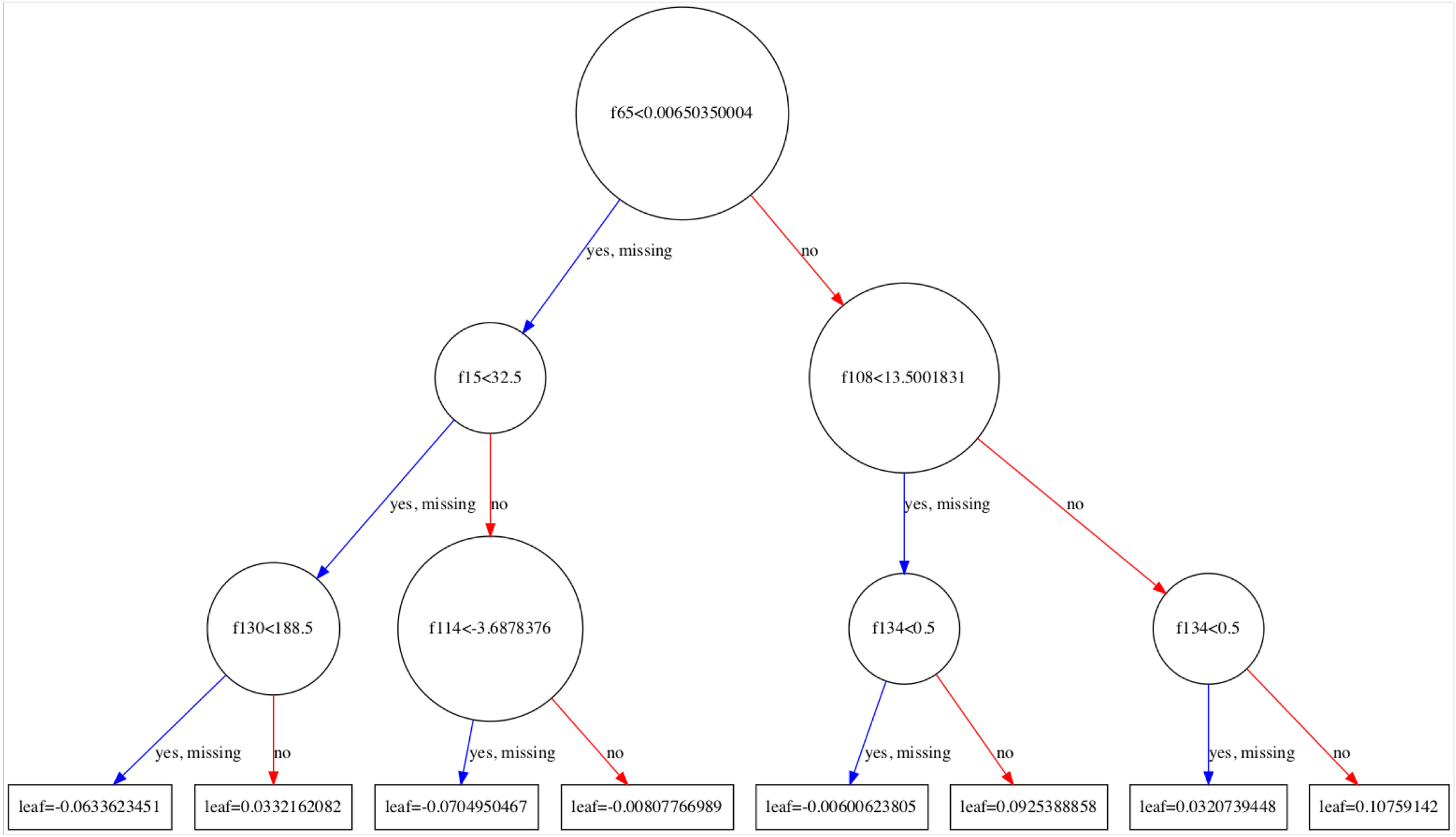
```
preds = model.predict(testing_data)
```

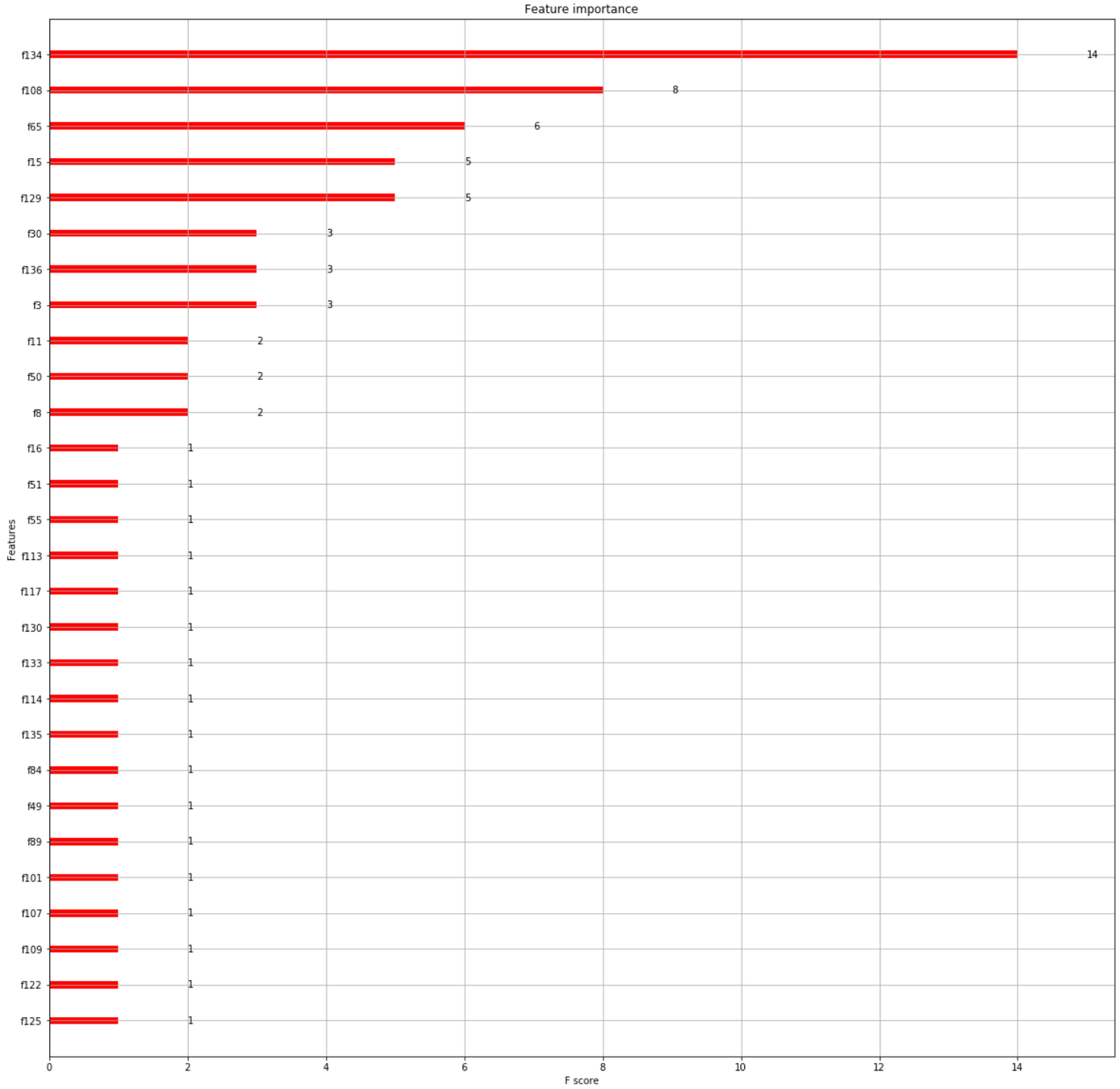
```
param = {'max_depth':6, 'eta':0.3, 'silent':0, 'objective':'rank:pairwise', 'num_round':10}
```

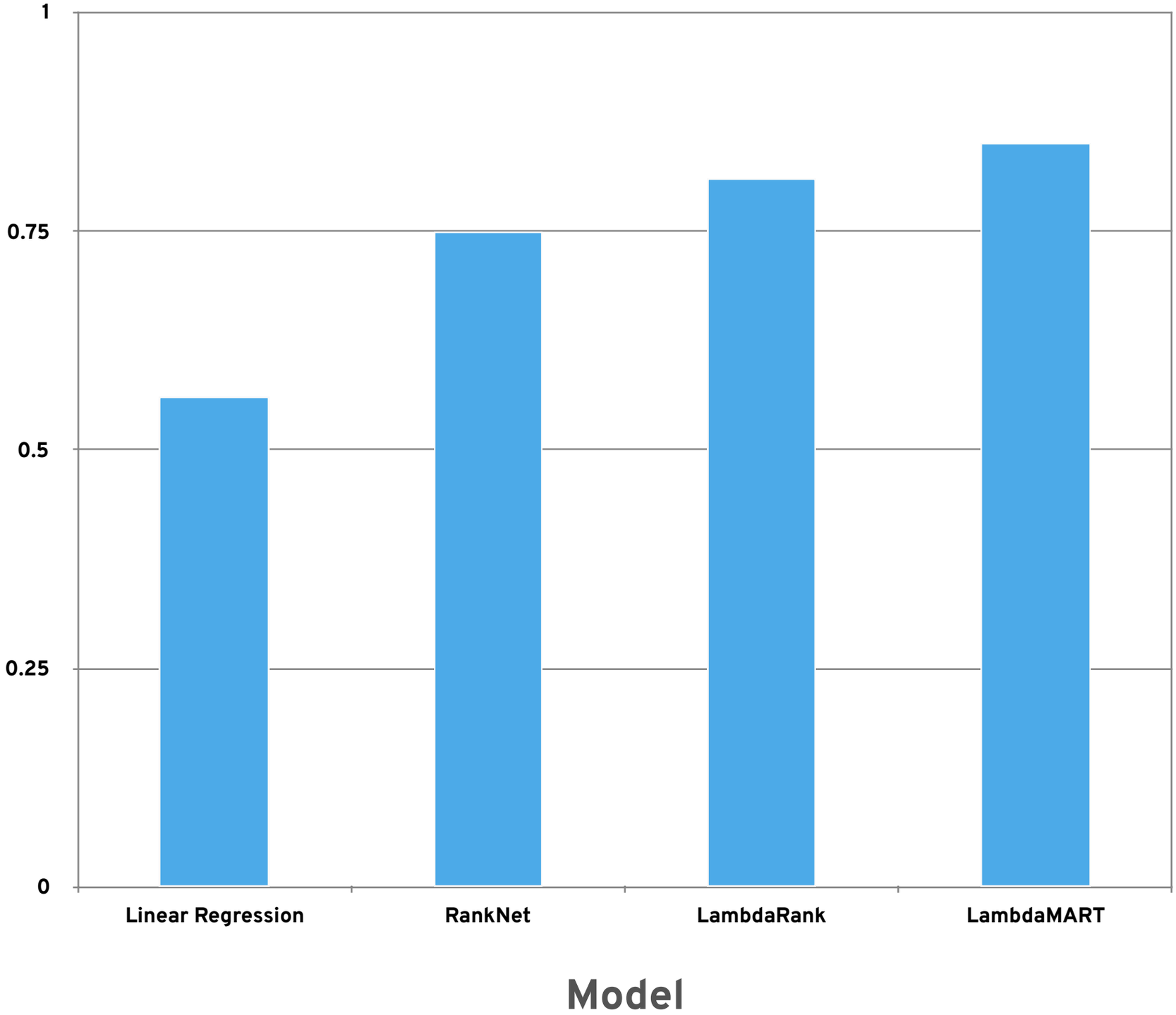
```
model = xgb.train(param, training_data)
```

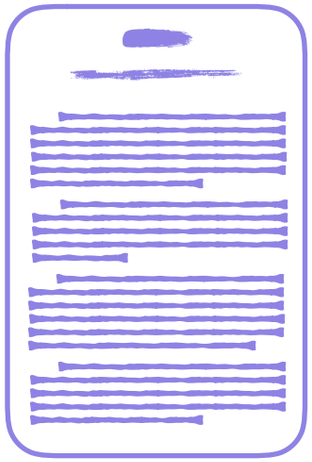
```
preds = model.predict(testing_data)
```

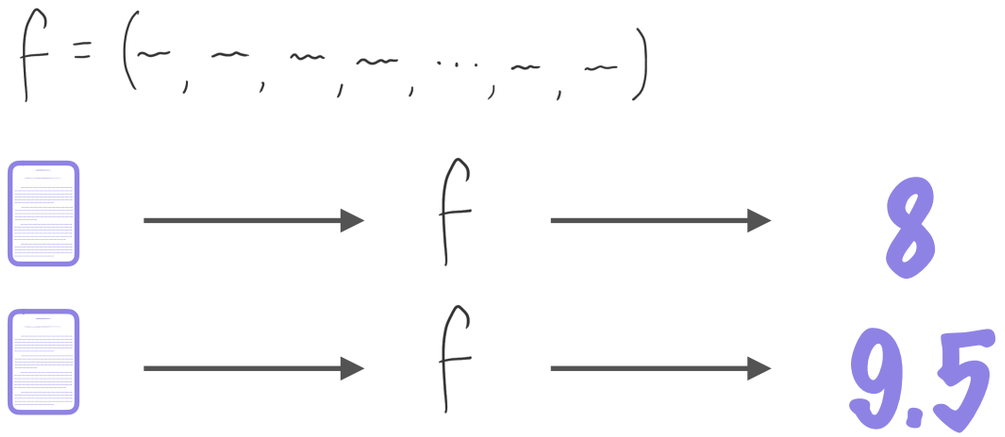


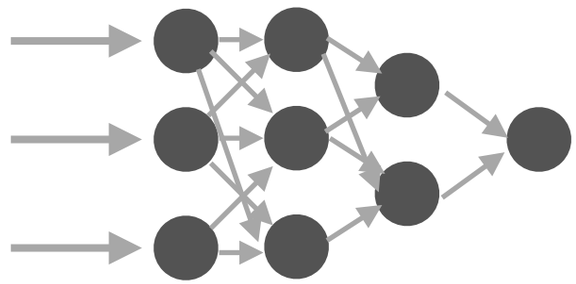

f65<0.00650350004
yes, missing
no
f15<32.5
f108<13.5001831
yes, missing
no
yes, missing
no
f130<188.5
f114<-3.6878376
f134<0.5
f134<0.5
yes, missing
no
yes, missing
no
yes, missing
no
yes, missing
no
leaf=-0.0633623451
leaf=0.0332162082
leaf=-0.0704950467
leaf=-0.00807766989
leaf=-0.00600623805
leaf=0.0925388858
leaf=0.0320739448
leaf=0.10759142

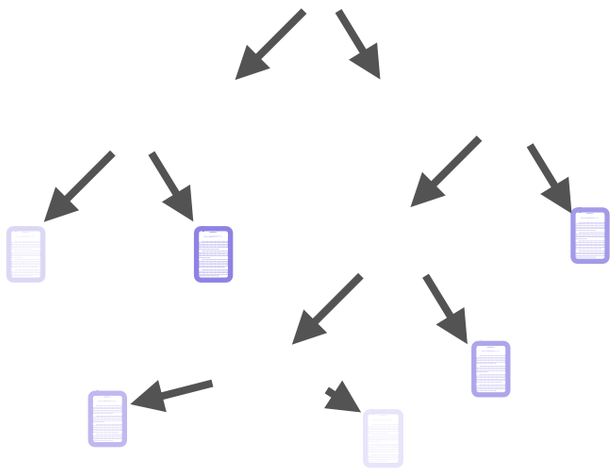

Feature importance
f134
f108
f65
f15
f129
f30
f136
f3
f11
f50
f8
f16
f51
f55
f113
f117
f130
f133
f114
f135
f84
f49
f89
f101
f107
f109
f122
f125
14
8
6
5
5
3
3
3
2
2
2
1
1
1
1
1
1
1
1
1
1
1
1
1
1
Features
0
2
4
6
8
10
12
14
F score


nDCG
1
0.75
0.5
0.25
0
Linear Regression
RankNet
LambdaRank
LambdaMART
Model





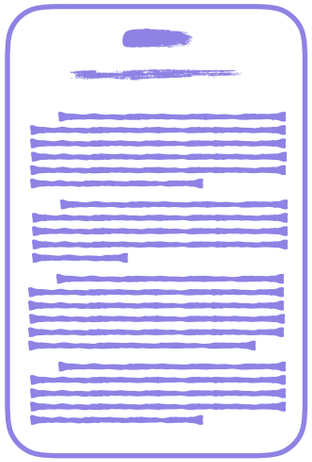


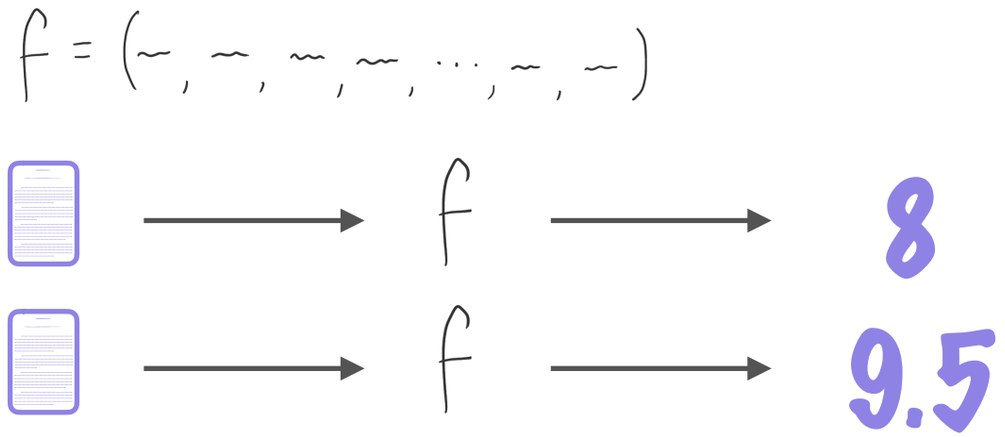


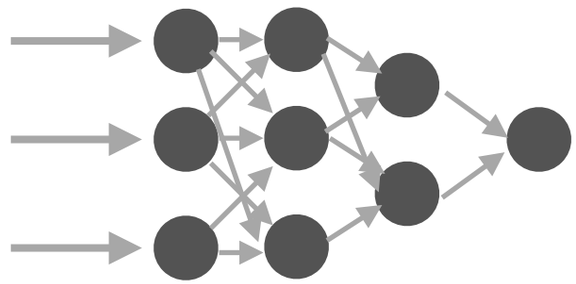
dmlc XGBoost

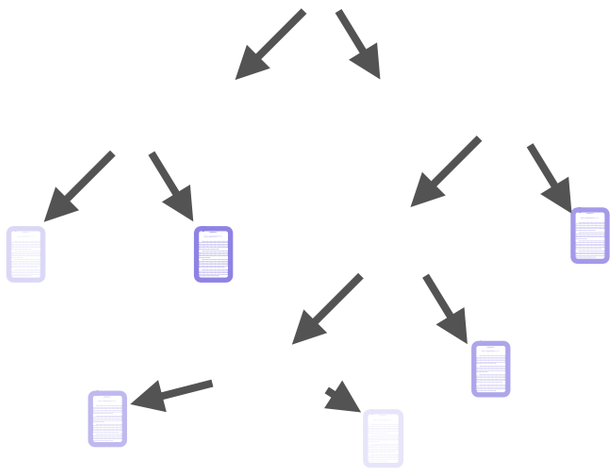
sophie@redhat.com

@sophwats









dmlc XGBoost

sophie@redhat.com

@sophwats