Error Analysis

Prof. Dr. M. Elif Karslıgil

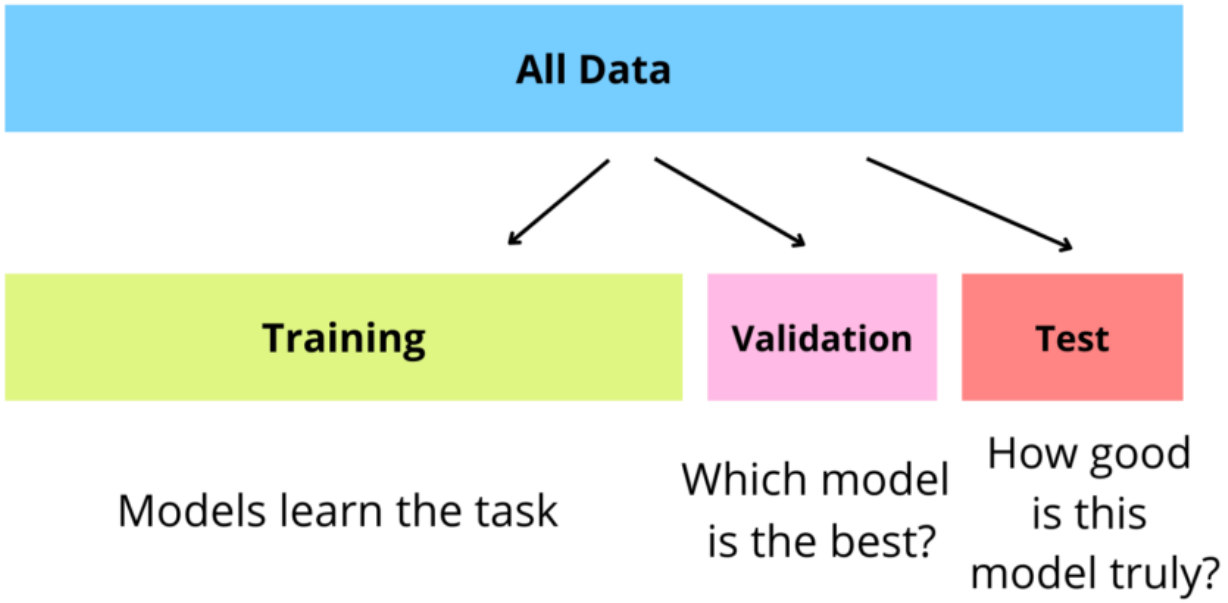
Dataset in Machine Learning

- The goal of a good machine learning model is to build a model that performs well on new, unseen data.
- *A dataset* is a collection of examples used to teach a machine learning model how to make decisions.

- Features (inputs) → the information or attributes used to make predictions
  - –Example: height, weight, color, pixel, voice signal

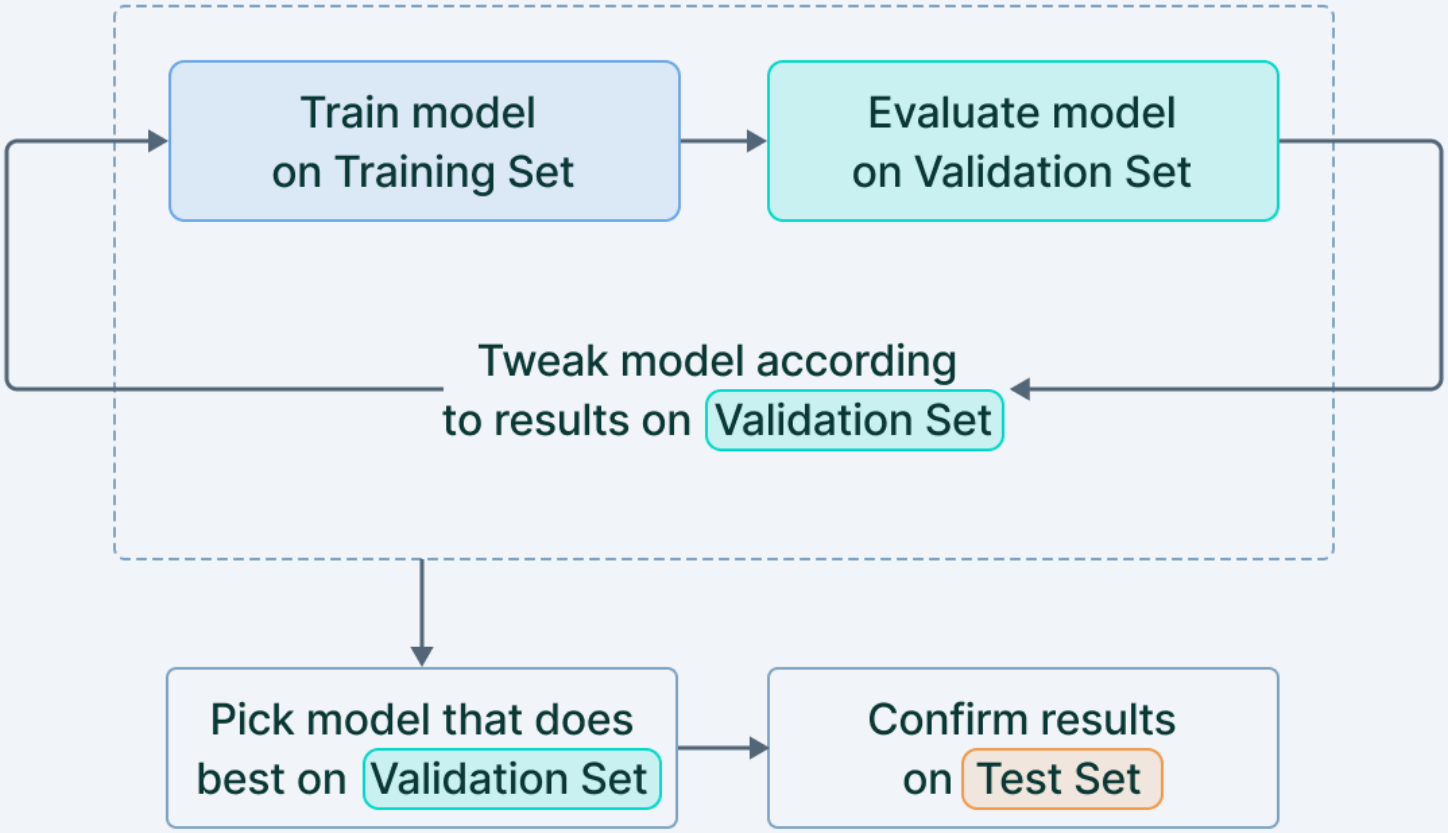
- Labels / Targets (outputs) → the correct answers (only for supervised learning)
  - –Example: age, category, spam/non-spam, yes/no

Measuring the Performance of an ML Model



Don't use *the same dataset* for model training and model evaluation.

Training data/validation/test



V7 Labs

Splitting Dataset

We split data into train/validation/test to evaluate performance.

- Training set(LEARN): used to train model.
- Validation set (TUNE): to tune parameters, select features, and make other decisions regarding the learning algorithm.
- Test set (EVALUATE): Used only at the end to evaluate final performance.

Splitting Dataset

Dataset Size	Suggested Split
Large dataset	**80% train / 10% validation / 10% test**
Medium dataset	**70% / 15% / 15%**
Small dataset	Use **cross-validation** instead of a fixed split

- Shuffle the data before splitting (important!)
- If working with time-series, do not shuffle — split chronologically
- Ensure same data distribution in each split
- Stratified split if classification

Holdout Method

- the dataset is split into train, validation, and test sets.
- For large datasets: 80% train, 10% validation, 10% test
- Performance depends on how the data was split
- May not generalize as well with small datasets
- in general, the more data we have, the better are the estimated parameters

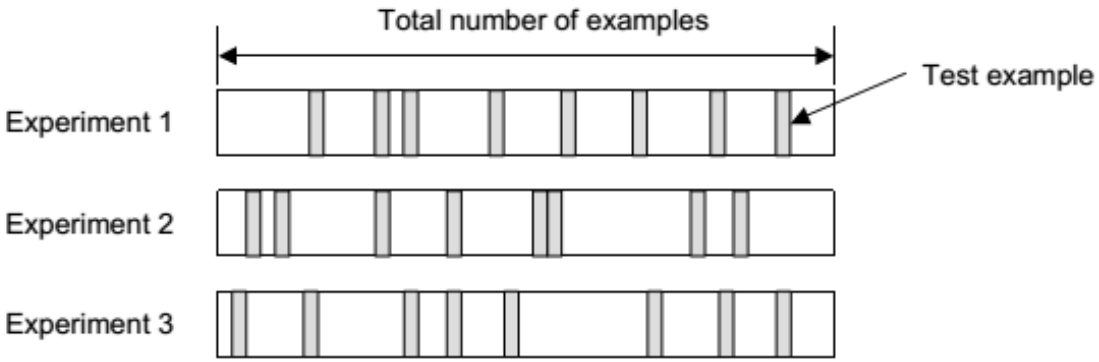
Holdout Method

- For “unbalanced” datasets, samples might not be representative
  - Few or none instances of some classes

- *Stratified sample: advanced version of balancing the data*
  - each class is represented with approximately equal proportions in both subsets

Random subsampling (Repeated HoldOut Method)

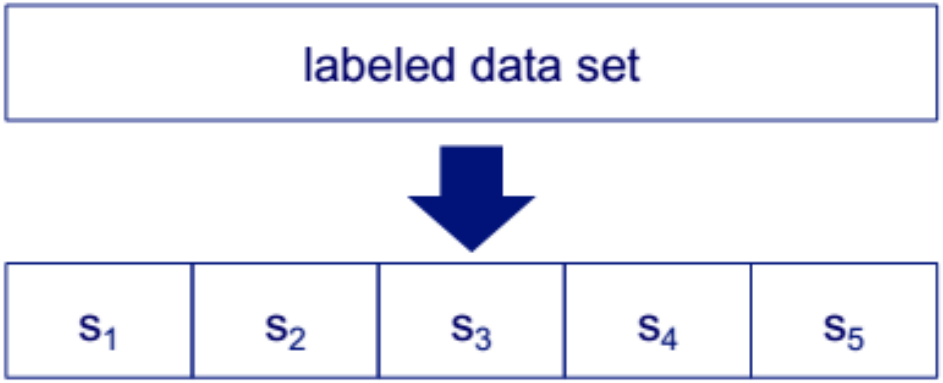
- Randomly split the dataset into train and test sets
- Repeat the process many times with different random splits
- Average the performance results



$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Cross Validation

partition data into *n* subsamples



iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Leave-One-Out Cross Validation

- a particular form of cross-validation:
  - Set number of folds to number of training instances
  - I.e., for *n* training instances, build classifier *n* times
- Makes best use of the data

Cross Validation Example

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	S_2 S_3 S_4 S_5	S_1	11 / 20
2	S_1 S_3 S_4 S_5	S_2	17 / 20
3	S_1 S_2 S_4 S_5	S_3	16 / 20
4	S_1 S_2 S_3 S_5	S_4	13 / 20
5	S_1 S_2 S_3 S_4	S_5	16 / 20

accuracy = 73/100 = 73%

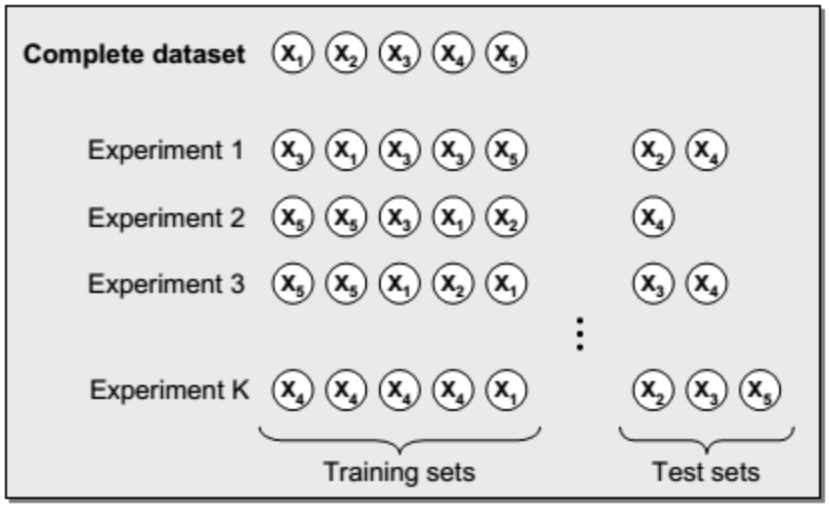
$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Hold-Out vs. Cross Validation

Feature	Hold-Out	Cross-Validation
Dataset size	Large datasets	Small–medium datasets
Speed	Fast	Slower
Stability/Reliability	Medium	High
Common Use	Practical ML pipelines	Academic / small data or when tuning models

Bootstrap

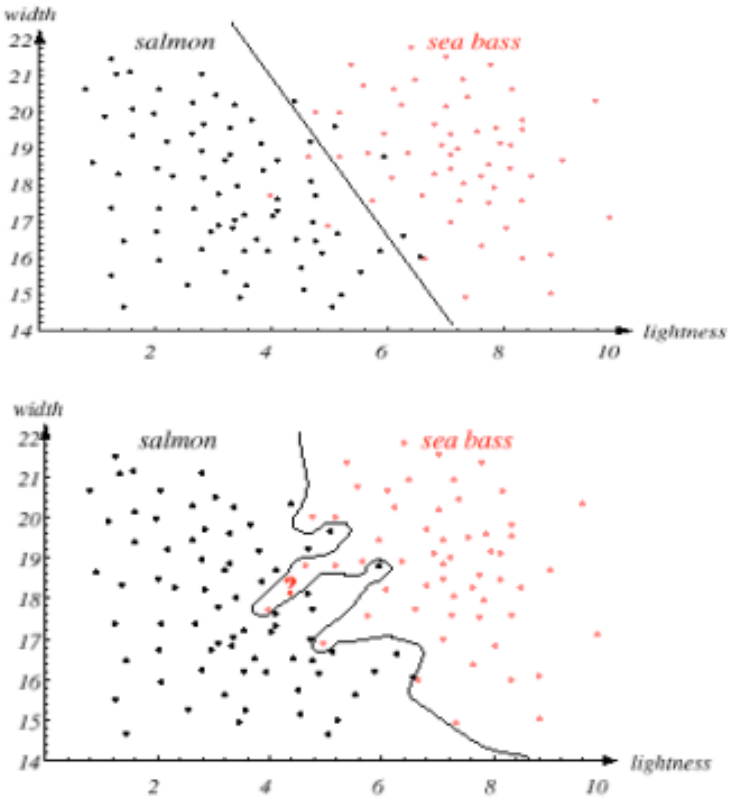
- From a dataset with N examples
  - Randomly select (with replacement) N examples and use this set for training
  - The remaining examples that were not selected for training are used for testing
    - This value is likely to change from fold to fold
- Repeat this process for a specified number of folds (K)
- As before, the true error is estimated as the average error rate on test examples

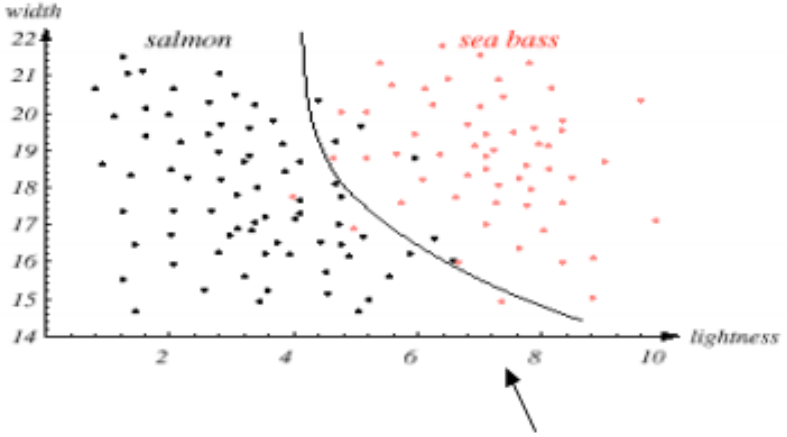


Summary of Evaluation Methods

- Use train, test, validation sets for large data
- Use Cross-validation for small data
- Don’t use test data for parameter tuning - use separate validation data
- Balance un-balanced data
- Enhance the accuracy of your results by using stratified sampling to ensure fair representation of all subgroups.

Which model is best?



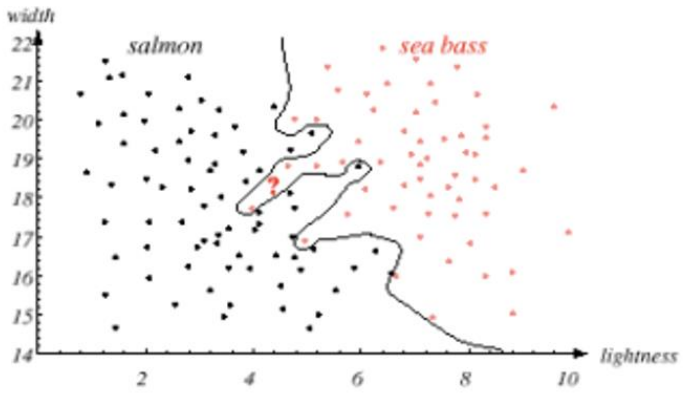


People like this one. Why?

Overfitting

- If a model does much better on the training set than on the test set, then it is likely overfitting.

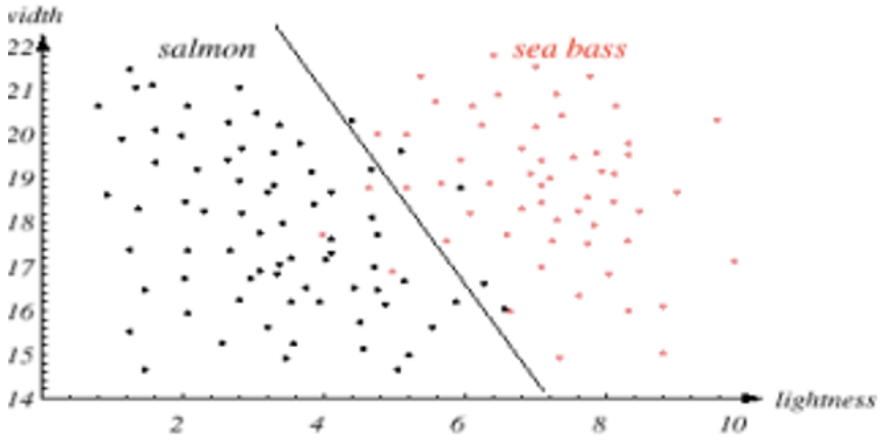
- Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data



Underfitting

- Underfitting refers to a model that can neither generalize the training data nor generalize to new data.

- Underfitting occurs when a model is too simple.



Basic Error Analysis

- Don’t start off trying to design and build the perfect system.

- **Build and train a basic system as quickly as possible**.

- Then use error analysis to identify the most promising directions and iteratively improve the algorithm from there

Basic Error Analysis

- Your dev/test set performance is usually worse than your training set performance.

- If you are getting 85% accuracy on the examples your algorithm has seen, there's no way you're getting 95% accuracy on examples your algorithm hasn't even seen.

Basic Error Analysis

- Carry out error analysis by manually examining ~100 dev set examples the algorithm misclassifies and counting the major categories of errors.

- Use this information to prioritize what types of errors to work on fixing
  - Categorize errors
  - Clean up mislabeled examples if necessary
    - Check the errors due to mislabeled examples

Basic Error Analysis

- Split the dev set into an *Eyeball dev set* (we are looking at it with our eyes) which you will manually examine, and a *Blackbox dev set*, which you will not manually examine.

- If dev set is not big enough to split this way, use the entire dev set as an Eyeball dev set for manual error analysis, model selection, and hyperparameter tuning

Setting up development and test sets

- When starting out on a brand-new application, try to establish dev/test sets and a metric quickly

- Choose a **single-number** evaluation metric. Having dev/test sets and a single-number evaluation metric helps quickly evaluate algorithms.
  - Example: use accuracy, if you really care about precision and recall use F1 score.

Setting up development and test sets

- Choose dev and test sets from a distribution that reflects what data you expect to get in the future and want to do well on

- Choose dev and test sets from the same distribution if possible

Setting up development and test sets

- The old heuristic of a 70%/30% train/test split does not apply for problems where you have a lot of data; the dev and test sets can be much less than 30% of the data
  - dev set: should be large enough to detect meaningful changes in the accuracy of your algorithm, but not necessarily much larger.
  - test set: should be big enough to give you a confident estimate of the final performance of your system.

Setting up development and test sets

- If the dev set overfits : get more dev set data.
- If the real data different from the dev/test set distribution: get new dev/test set data.
- If the metric is no longer measuring what is most important: change the metric.

Setting up development and test sets

- If your dev set and metric are no longer pointing your team in the right direction, quickly change them:
  - If you had overfit the dev set, get more dev set data.
  - If the actual distribution you care about is different from the dev/test set distribution, get new dev/test set data.
  - If your metric is no longer measuring what is most important to you, change the metric.

Bias and Variance

- The **bias** is the error rate on the training set
- **The variance** is how much worse the algorithm does on the dev (or test) set than the training set.
- **Bias** and **variance** describe how a model learns patterns from data.
- They help explain **why a model performs poorly** and **what to fix**

Bias (Underfitting)

- Bias means the model is **too simple** to learn the true pattern.
- It **underfits** — misses important relationships in data.
- **High Bias:** Bad performance on training and test data
- **Low Bias:** Model fits training data well

Fixing High Bias

- Add more features
- Use more complex models (e.g. NN instead of linear regression)
- Reduce Regularization
- Train longer for deep models

Variance (Overfitting)

- Variance occurs when the model is **too complex**.

- Model is memorizing instead of generalizing. For example fits the noise instead of the true pattern.

- **High Variance:** Very low training error, high testing error
- **Low Variance:** Model fits training and testing data well

Reducing Variance

- **Add more training data:** helps with variance problems, but it usually has no significant effect on bias.
- **Add regularization:** reduces variance but increases bias
- Add early stopping: (i.e., stop gradient descent early, based on dev set error): This technique reduces variance but increases bias. Early stopping behaves a lot like regularization methods,

Reducing Variance

- **Feature selection to decrease number/type of input features:** might help with variance problems, but it might also increase bias.
- **Decrease the model size (such as number of neurons/layers):** This technique could decrease variance, while possibly increasing bias
- **Modify input features:** new features could help with both bias and variance.
- **Modify model architecture**

Fixing High Variance

- Get more data
- Use simpler model
- Reduce number of features
- Add Regularization (L1/L2)
- Apply dropout(Neural Networks)

Bias and Variance

When model performance is bad, try:

1. **More training data**
2. **Better features**
3. **Regularization**
4. **Try different algorithms**
5. **Tune hyperparameters**

Always guided by **bias-variance diagnosis**.

Problem	Training Error	Test Error	Solution
Underfitting (High Bias)	High	High	More complex model, add features, reduce regularization
Overfitting (High Variance)	Low	High	More data, regularization, simpler model

Bias and Variance Examples

- Target error rate: 5% (95% accuracy),
- the training set error rate : 15%
- first problem to solve is **to improve your algorithm's performance on your training set.**

Bias and Variance Examples

- the training set error rate : 15%
- the dev set error rate : 16%
- We break the 16% error into two components:
  - training error rate : 15% : algorithm's bias.
  - how much worse the algorithm does on the dev (or test) set than the training set. It is algorithm's variance. (1% : 16%-15%)

Bias and Variance Examples

- Training error = 1% Dev error = 11% :
  - Bias : % 1
  - Variance : % (11-1) : %10 : **high variance**
- The classifier has very low training error, but it is failing to generalize to the dev set.
- This is called **overfitting**.

Bias and Variance Examples

- Training error = 15% Dev error = 16% :
  - Bias : % 15 : **high bias**
  - Variance : % (16-15) : %1 : **low variance**

- The classifier has high training error, called **underfitting**

Bias and Variance Examples

- Training error = 15% Dev error = 30% :
  - Bias : % 15 : **high bias**
  - Variance : % (30-15) : %15 : **high variance**

- The classifier has high training error, it is also failing to generalize to the dev set.
- It is simultaneously **overfitting** and **underfitting**.

Optimal error rate (unavoidable bias)

- In a learning model:
  - Training error rate : 15%
  - Dev error rate : 30%

  - Expected error rate : 14% : **unavoidable bias**
  - Avoidable bias: 1% (15-14)
  - Bias = 15% (unavoidable bias + avoidable bias)
  - Variance = 15% (30-15)

Optimal error rate (unavoidable bias)

- In a learning model:
  - Expected error rate : 14% : **unavoidable bias**
  - Training error rate : 15%
  - Dev error rate : 16%

  - Bias = 15%
  - Avoidable bias: 1% (15-14)
  - Variance = 1% (16-15)
  - **Good Algorithm (dev error rate – unavoidable bias = 2%)**

Reducing Avoidable Bias

- Increase the model size
  - If it increases variance, then use regularization

- Modify input features based on insights from error analysis :
  - adding more features could increase the variance; then use regularization

Reducing Avoidable Bias

- Increase the model size
  - If it increases variance, then use regularization

- Modify input features based on insights from error analysis :
  - adding more features could increase the variance; then use regularization

Addressing Bias and Variance

- If you have high avoidable bias:
  - increase the size of your model (for example, increase the size of your neural network by adding layers/neurons)

- If you have high variance:
  - add data to your training set

High Bias & High Variance

- First fix high bias (make model learn enough), then fix high variance (make it generalize).

- Since the model is too simple and unstable, you need to:
  - Increase Model Complexity
  - Get More Data
  - Feature Engineering
  - Reduce Noise in Data
  - Try Different Model Type
  - Regularization Tuning
    - For high bias → **decrease regularization**
    - For high variance → **increase regularization**

Low Bias & Low Variance

- Training error = 0.5% Dev error = 1% :
  - Bias : % 0.5 : **low bias**
  - Variance : % 0.5 **low variance**

- Great performance

Bias and Variance Road Map

Action	Helps With
Increase model size	Bias ↓
Add more data	Variance ↓
Add more features	Bias ↓
Regularize more	Variance ↓
Regularize less	Bias ↓
Better features	Bias ↓ & Variance ↓ ✅

Bias-Variance Tradeoff

Bias	Variance	Behavior
High	Low	Underfitting
Low	High	Overfitting
Low	Low	✅ Best (ideal)
High	High	❌ Worst (model is both wrong & unstable)

Minimize **total error**:

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

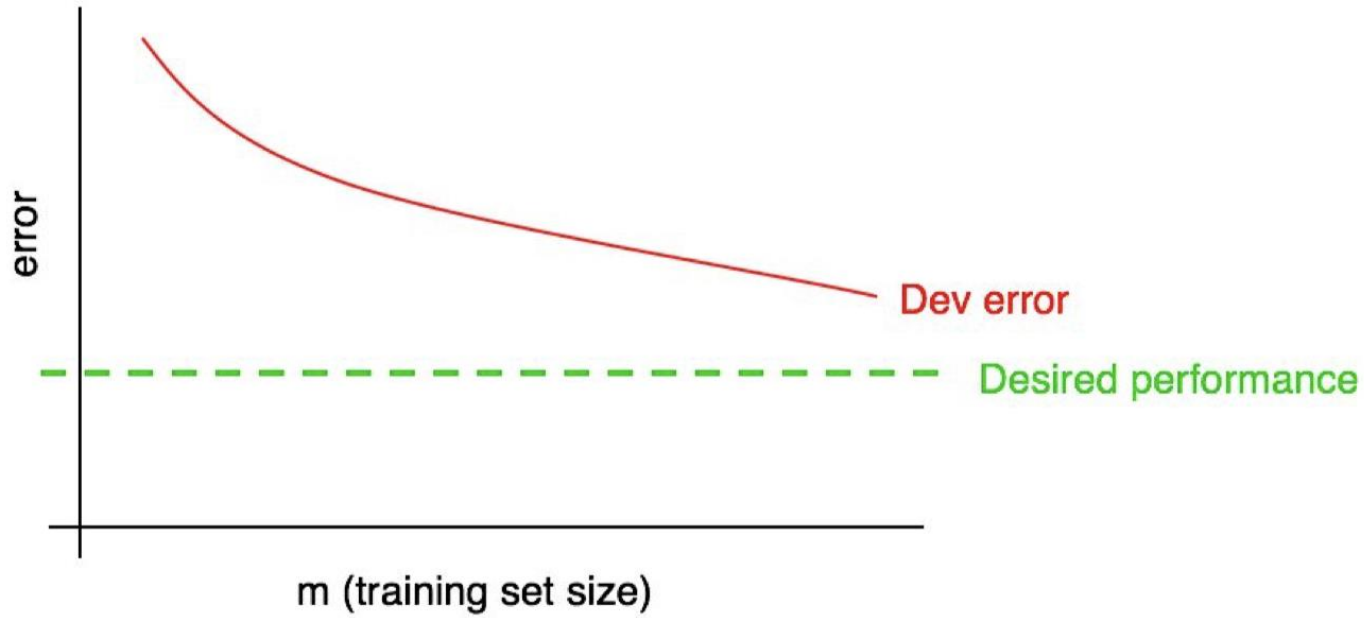
Where **noise** is irreducible.

Bias vs.Variance Tradeoff

- increasing the size of model—adding neurons/layers in a neural network, or adding input features—generally reduces bias but could increase variance.

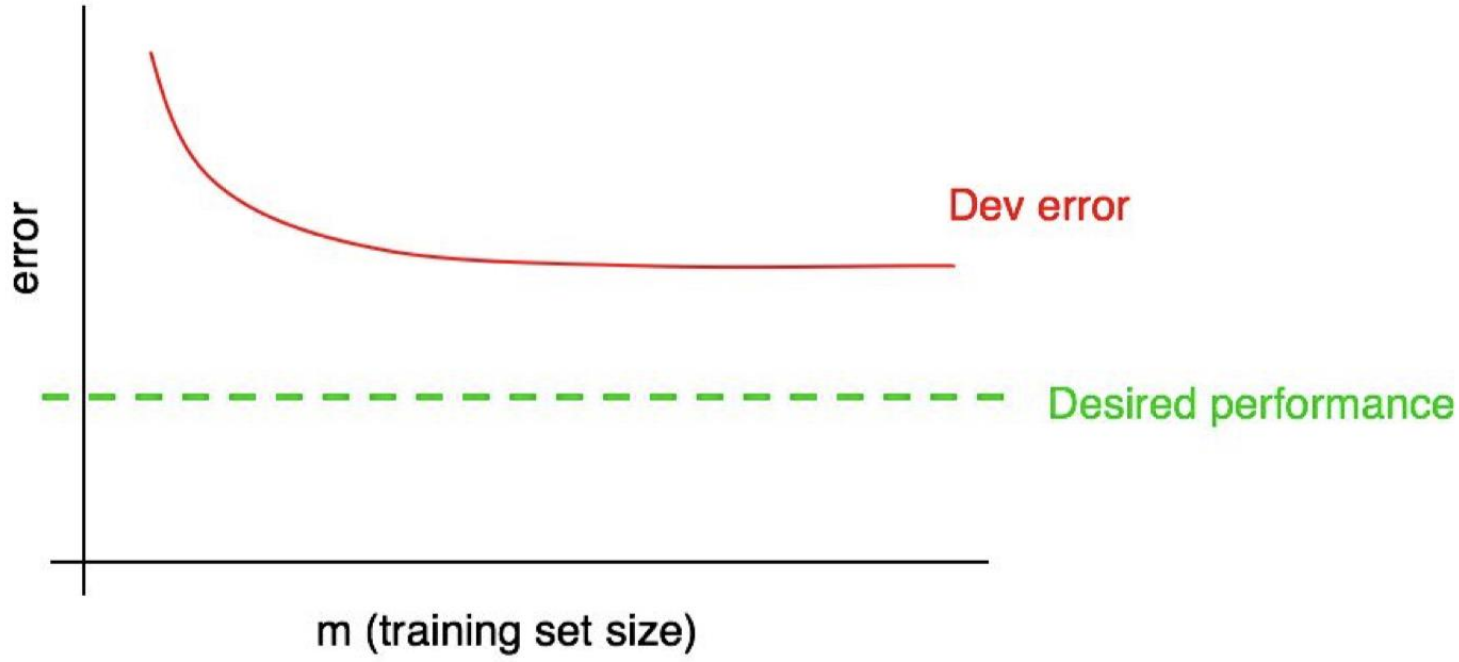
- Alternatively, adding regularization generally increases bias but reduces variance.

Learning Curve



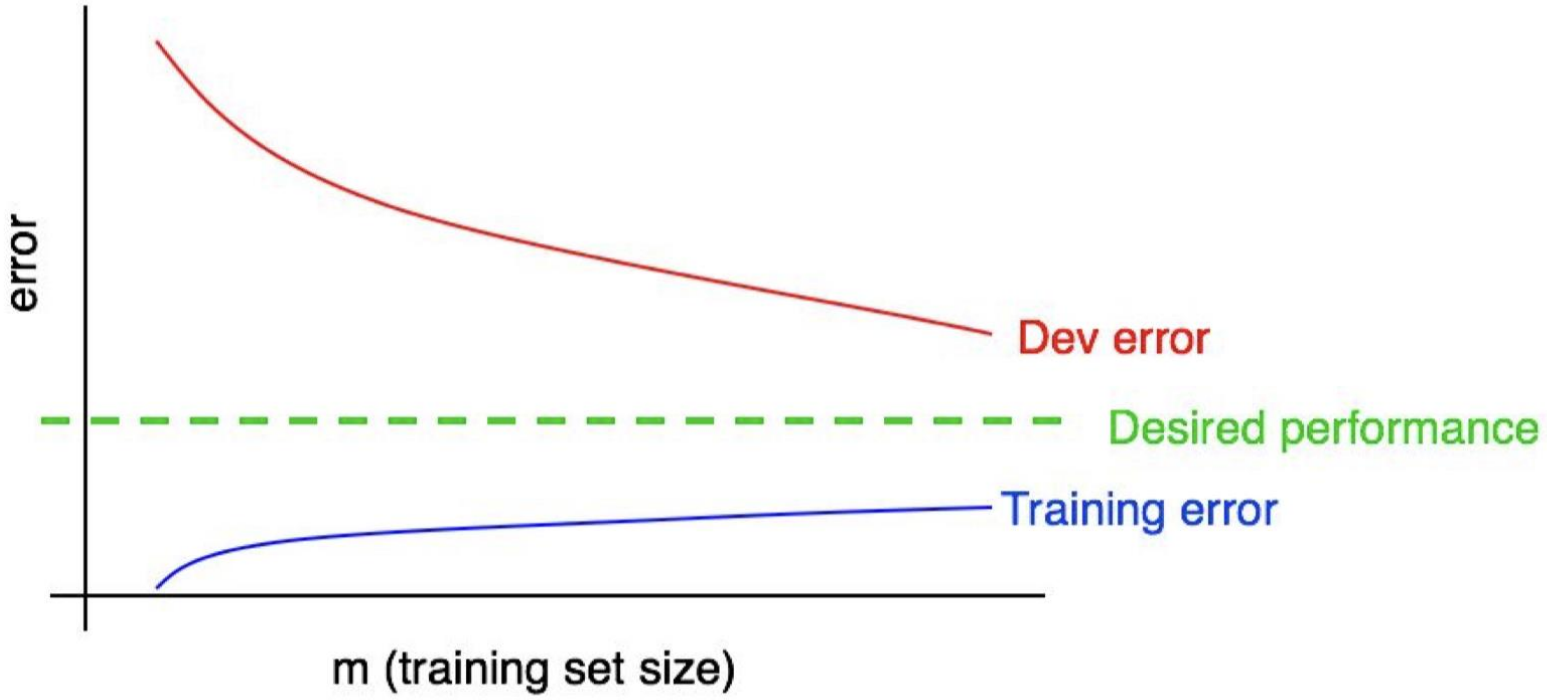
doubling the training set size might allow reaching the desired performance

Learning Curve



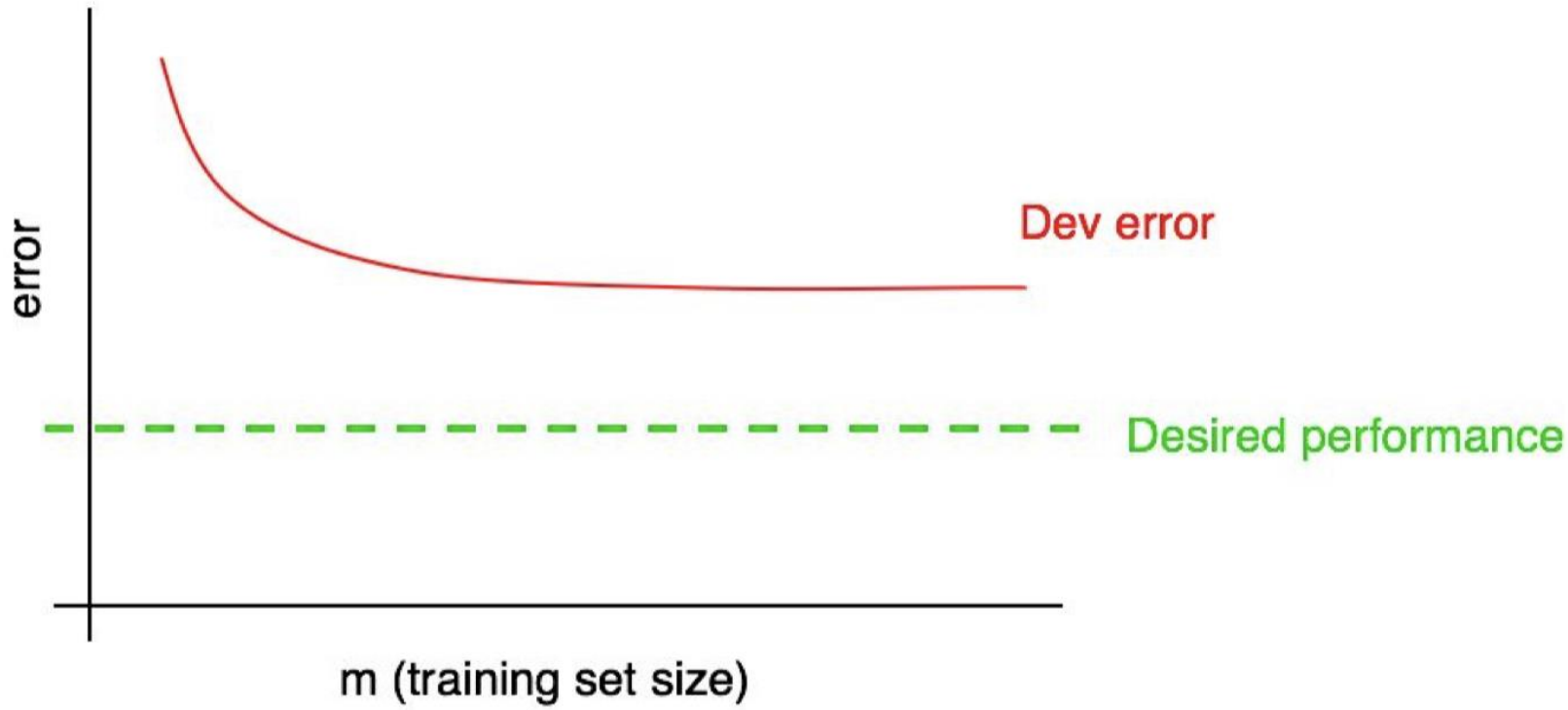
adding more data won't help increasing performance

Plotting Training error

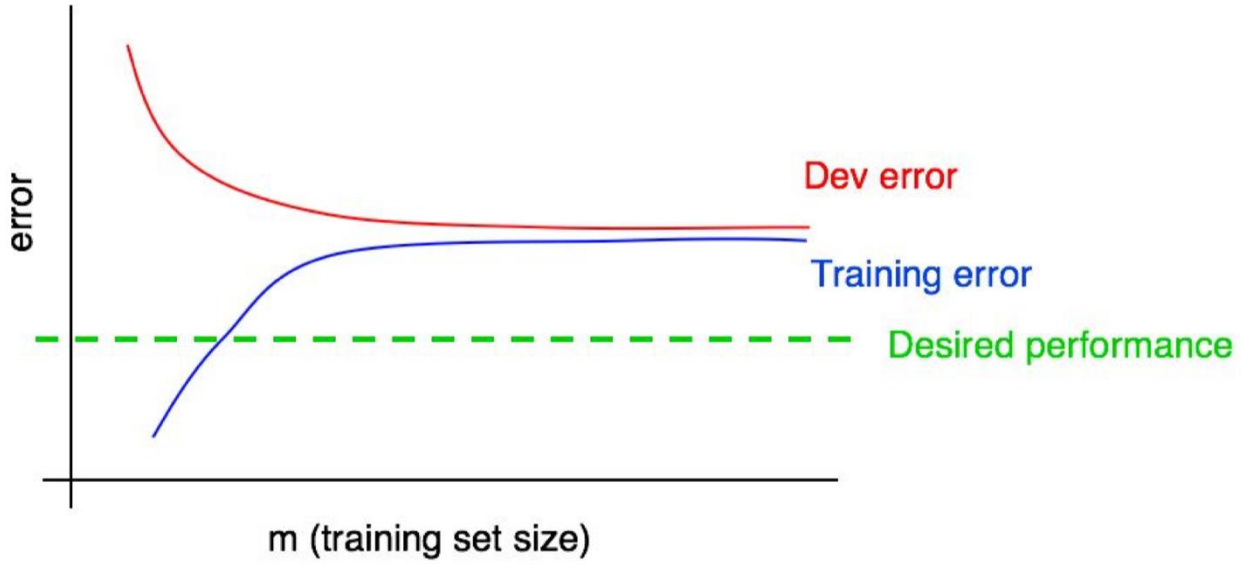


- dev set (and test set) error should decrease as the training set size grows.
- training set error usually increases as the training set size grows.

High Bias



High Bias



adding more data will not, by itself, be sufficient.

Low Bias – High Variance

Adding more training data will probably help close the gap between dev error and training error

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Binary Classification

- Two major types of binary classification problems.
  - "X versus Y" : positive versus negative sentiment.
  - "X versus not-X" : spam versus non-spam

- For X versus not-X use : precision/recall metric

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- **Confusion Matrix:**

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

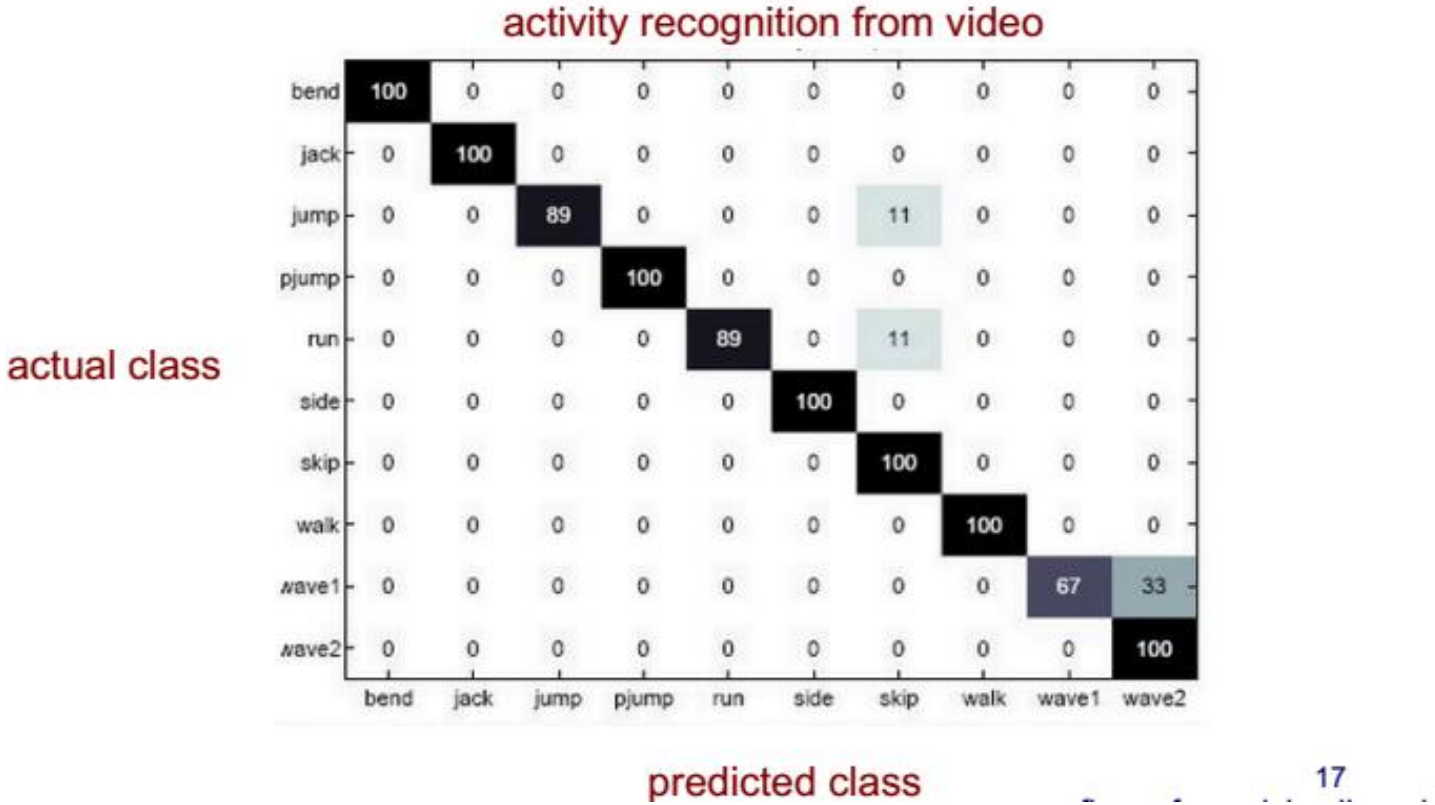
b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Confusion Matrix

How can we understand what types of mistakes a learned model makes?



Metrics for Performance Evaluation

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Single number evaluation metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Metrics for Performance Evaluation

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

(harmonic mean of precision and recall)

a: TP (true positive) **b: FN (false negative)**

c: FP (false positive) **d: TN (true negative)**

Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

When to use F1 score ?

- In the extreme case where P = R, then F = P = R.
- In the imbalanced case, for instance P = 0.1 and R = 0.9, the overall f-measure is 0.18.

	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.00	0.20	0.26	0.30	0.32	0.33
0.4	0.00	0.26	0.40	0.48	0.53	0.57
0.6	0.00	0.30	0.48	0.60	0.68	0.74
0.8	0.00	0.32	0.53	0.68	0.80	0.88
1.0	0.00	0.33	0.57	0.74	0.88	1.00

Which classifier do you choose?

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms