DS 200
Topic 5 and Lab5

Word Vector Representation of Text & Learning Decision Tree for Filtering Irrelevant Tweets

Instructor: John Yen

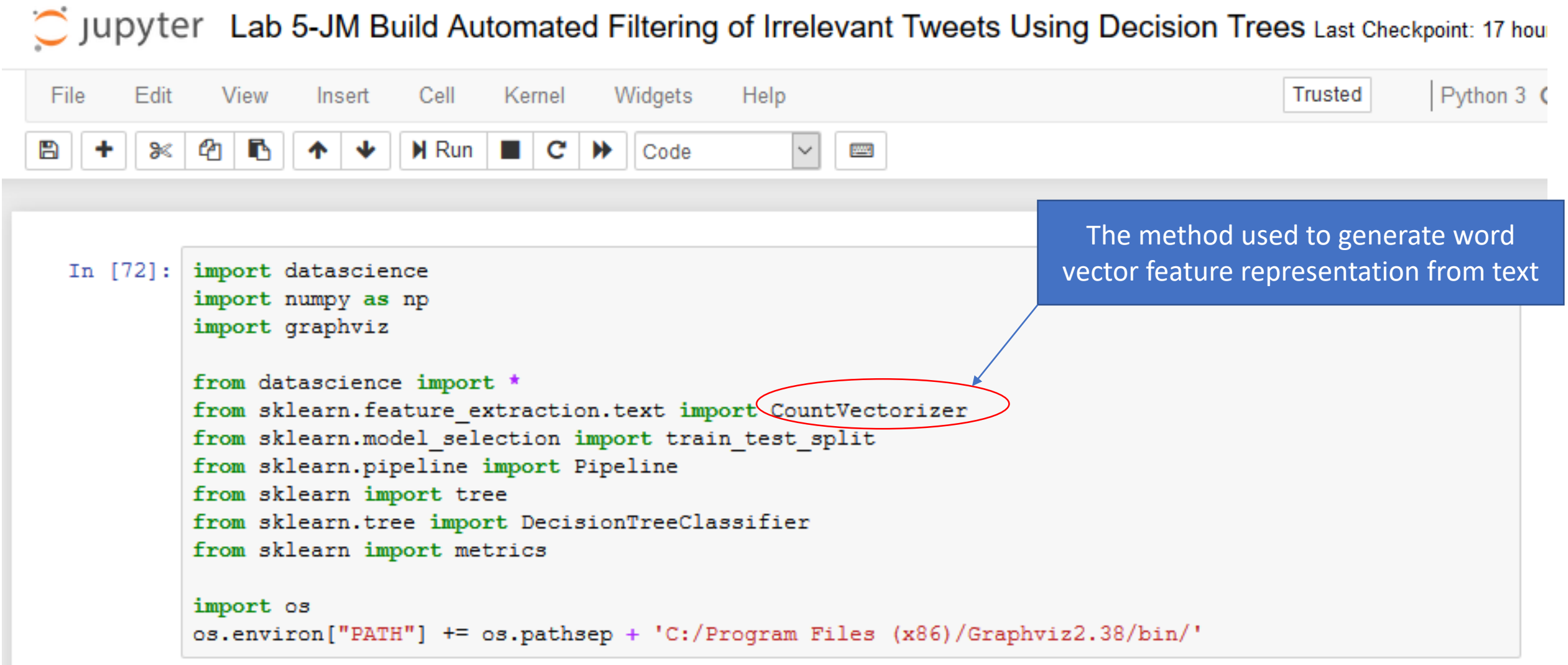
LA: Luwei Lei

September 18, 2018

In this lab, you will learn to

1. Login into Vlab, launch Jupyter Notebook and Python 3. Import relevant Python modules and methods
2. Lab 5-1: Split tagged twitter data into two sets: (1) a ***training*** set and (2) a ***testing*** set. (slide 5-13)
3. Lab 5-2: Build a decision-tree Relevant Classifier from training data (using scikit-learn package). (slide 14-19)
4. Lab 5-2 Evaluate the predictive model on the testing data. (15-19)
5. Lab 5-2 Visualize the decision-tree predictive model (20-21)

Import Relevant Python Modules and Methods



Step 2: Split Tagged Data into Training and Testing sets

Type the following in Jupyter to load tagged tweets (the CSV file that contains your tagging results)

t1 = Table.read_table('Z:\Downloads\<your-tagged-twitter-file>.csv', sep = ',')

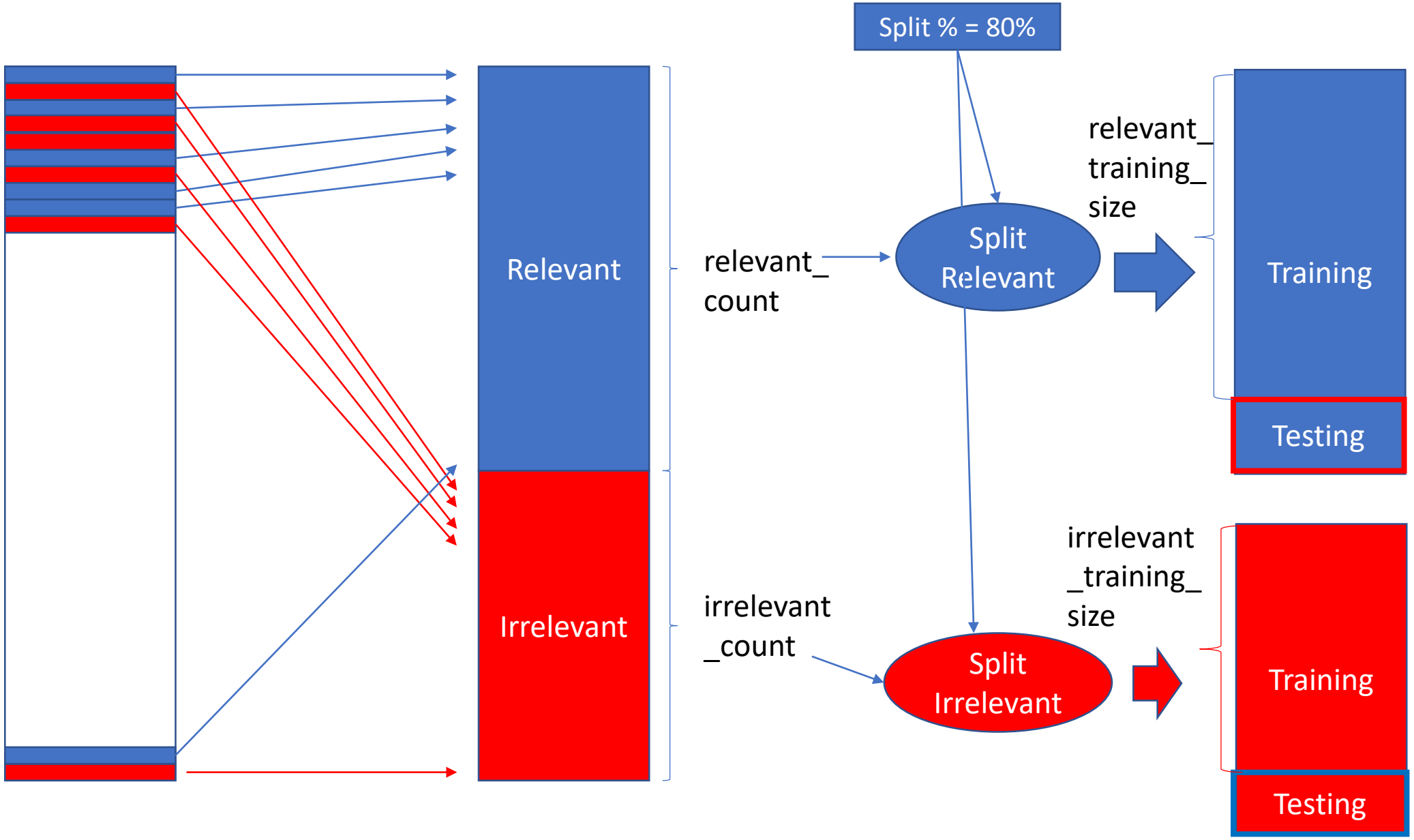
If you use Tab as delimiter in a text file, use '\t' as the value of sep here.

```
In [73]: t1 = Table.read_table("JohnMcCain_clean.csv", sep =',')

In [74]: t1
```

Out[74]:

user_id	user_name	tweet_time	location	text	Relevant	Stance
9.27e+17	Gilly Mobile	Mon Aug 27 21:20:09 +0000 2018	None	#JohnMcCain true American hero fought for his country #D ...	1	1
7.59e+17	The Complainster	Mon Aug 27 21:19:48 +0000 2018	USA	@1butterflywild @markknoller @SenJohnMcCain @NavalAcadem ...	1	0
9.55e+17	Sl2nasty	Mon Aug 27 21:20:19 +0000 2018	Phillys Main Line	@AaronBlake @AmericanLegion thank you for standing up fo ...	1	1
7.68e+17	Hasan	Mon Aug 27 21:18:41 +0000 2018	stanbul Trkiye	@abdulrahman_q0 @SenJohnMcCain fcking Arabs ! https://t. ...	0	nan

Splitting Labelled Data into a Training Set and a Testing Set



Create train and test sets

- 3. Separate data into relevant and irrelevant tweets

```
In [75]: tweets_relevant = t1.where('Relevant', are.equal_to(1))
```

```
In [76]: tweets_irrelevant = t1.where('Relevant', are.equal_to(0))
```

```
In [77]: tweets_relevant
```

Out[77]:

user_id	user_name	tweet_time	location	text	Relevant	Sta
9.27e+17	Gilly Mobile	Mon Aug 27 21:20:09 +0000 2018	None	#JohnMcCain true American hero fought for his country #D ...	1	
7.59e+17	The Complainster	Mon Aug 27 21:19:48 +0000 2018	USA	@1butterflywild @markknoller @SenJohnMcCain @NavalAcadem ...	1	
9.55e+17	SI2nasty	Mon Aug 27 21:20:19 +0000 2018	Phillys Main Line	@AaronBlake @AmericanLegion thank you for standing up fo ...	1	

Count the total number of relevant tweets and irrelevant tweets

```
In [79]: relevant_count = tweets_relevant.num_rows

In [80]: relevant_count
Out[80]: 26

In [81]: irrelevant_count = tweets_irrelevant.num_rows

In [82]: irrelevant_count
Out[82]: 9
```

Compute the Size of Relevant Training Tweets and Irrelevant Training Tweets

```
In [83]: relevant_training_size = round(relevant_count * 0.8 )

In [84]: irrelevant_training_size = round(irrelevant_count * 0.8)

In [85]: relevant_training_size

Out[85]: 21

In [86]: irrelevant_training_size

Out[86]: 7
```

Obtain the Index of Relevant Training Tweets and Irrelevant Training Tweets

```
In [87]: relevant_rows_train = list(range(relevant_training_size))

In [88]: relevant_rows_train

Out[88]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

In [89]: irrelevant_training_size= round(irrelevant_count * 0.8)

In [90]: irrelevant_rows_train= list(range(irrelevant_training_size))

In [91]: irrelevant_rows_train

Out[91]: [0, 1, 2, 3, 4, 5, 6]
```

Create a List of Relevant Tweets + a List of Irrelevant Tweets for Training

a Table of relevant tweets

The **take** method of Table object takes a specified list of rows (by row index), and return a new Table formed by these rows and the column(s) specified.

Select the 'text' column of each row in the Table

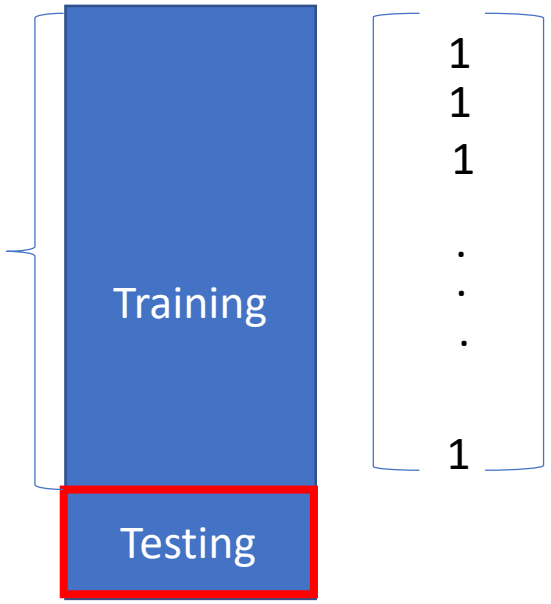
```
In [92]: X_train = list(tweets_relevant.take(relevant_rows_train)['text']) + \
         list(tweets_irrelevant.take(irrelevant_rows_train)['text'])
```

Convert into a list

```
print(X_train)
```

```
['#JohnMcCain true American hero fought for his country #DonaldTrump ignorant and no respect', '@1
butterflywild @markknoller @SenJohnMcCain @NavalAcademy Toddler should be grateful. #JohnMcCain #A
mericanHero too https://t.co/tTeCRhxfk1', '@AaronBlake @AmericanLegion thank you for standing up f
or @SenJohnMcCain and all veterans. I am sorry that https://t.co/Ogj5QWJN4K', '@AliVelshi @maddow
Additionally the Dept. of Veteran Affairs did NOT have their flags at half staff and were or https
://t.co/d0PXAh3HyD', '@anna_deardorff @Brian4Progresss @realDonaldTrump @SenJohnMcCain QAnon told
me John McCain faked his own death so h https://t.co/gL0zv3BPSr', '@brooklyn3r @RichPrice65 @FoxNe
ws @POTUS @SenJohnMcCain What do you want from him? McCain had already spoke how he https://t.co/J
xLnM9OD04', '@CBSEveningNews Too little too late. President Trump disrespected @SenJohnMcCain ever
y chance he got.....even went https://t.co/Bs58Uqen52', '@Channel4News @realDonaldTrump @SenJohnMc
Cain The Dotard really is a morally bankrupt sad excuse for a human being.', "@Channel4News @Sonia
```

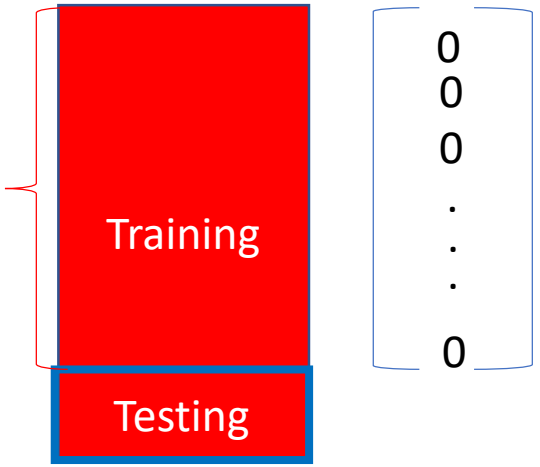
Creating Expected Outputs for the Training Data



1: Relevant

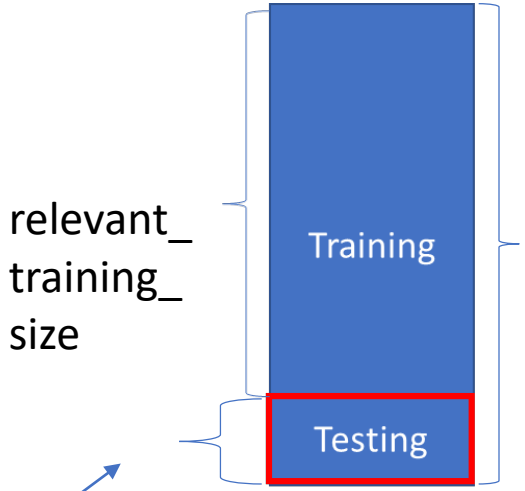
0: Irrelevant

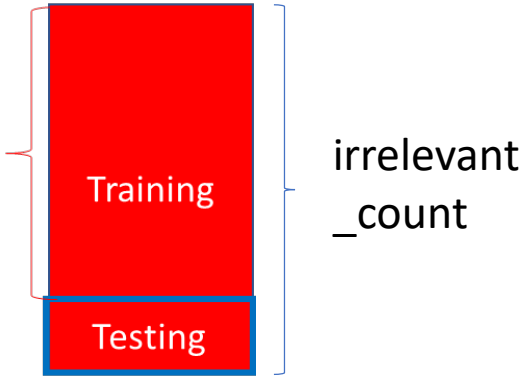
Expected Outputs should be consistent with the tags.




```
y_train = [1] * relevant_training_size + [0] * irrelevant_training_size
```

Create a List of Relevant Tweets and Irrelevant Tweets for Testing





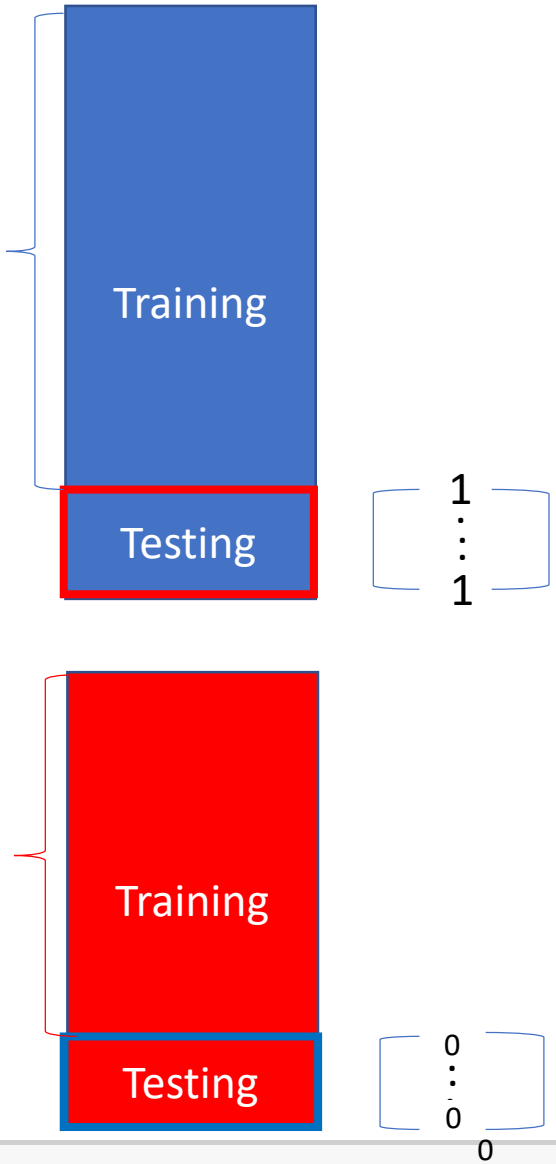
The sublist of relevant tweet index starting from the row index after the last row used for training

```
relevant_rows_test = list(range(relevant_count))[relevant_training_size :]
```

```
irrelevant_rows_test = list (range (irrelevant_count))[irrelevant_training_size :]
```

```
X_test = list (tweets_relevant.take(relevant_rows_test) ['text']) + \
list(tweets_irrelevant.take(irrelevant_rows_test) ['text'])
```

Create Expected Outputs of Testing Data




```
y_test = [1]*(relevant_count - relevant_training_size) + \
[0] * (irrelevant_count - irrelevant_training_size)
```

Convert Training Text Data into Word Feature Vectors

Transform training tweets to word feature vectors

A(i,j) is 1 if word W_j appears in tweet T_i

```
count_vect = CountVectorizer()
```

```
X_word_vect = count_vect.fit_transform(X_train)
```

```
print(X_word_vect.shape)
```

```
(28, 273)
```

Number of training tweets

Number of Word Features

Create a Decision Tree for Filtering Irrelevant Tweets

- You can choose your maximum tree depth (max_depth) and minimum number of training data in the leaf node (min_samples_leaf), suggest > 1

```
In [122]: clf = tree.DecisionTreeClassifier(criterion='entropy', random_state = 100, max_depth=15, \
                                            min_samples_leaf =2)

In [123]: clf.fit(X_word_vect, y_train)

Out[123]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=15,
                                 max_features=None, max_leaf_nodes=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=2, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, presort=False, random_state=100,
                                 splitter='best')
```

Create a decision tree classifier using the word-frequency features of training tweets and their target outputs (y_train)

Generate a Visualization of the Decision Tree

Obtain the names of the word features

```
dot_data= tree.export_graphviz(clf, out_file=None, feature_names=count_vect.get_feature_names())
```

```
from graphviz import *
```

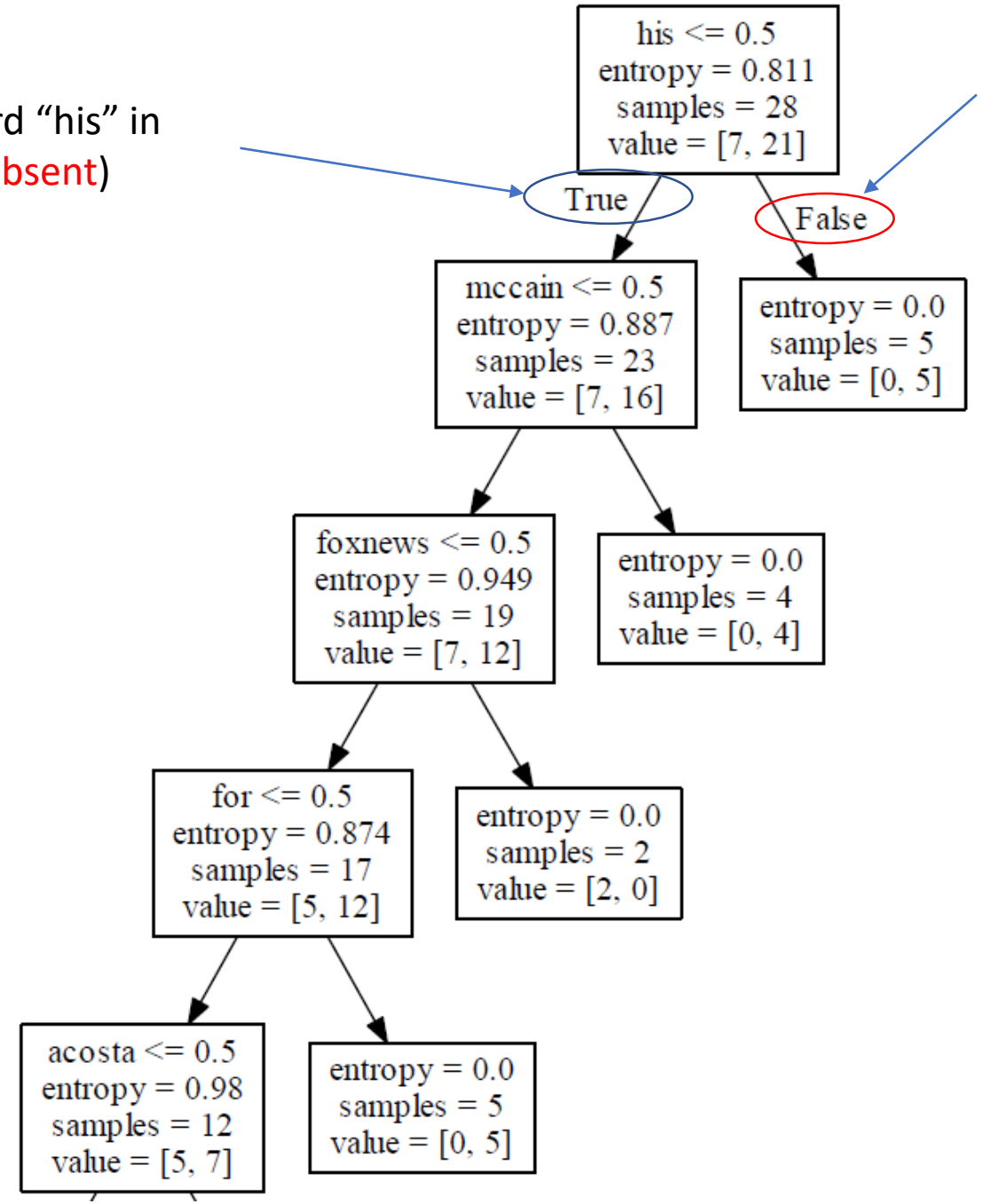
```
graph = graphviz.Source(dot_data)
```

```
graph.render('RelevantClassifier')
```

```
'RelevantClassifier.pdf'
```

The frequency of the word “his” in the tweet is <= 0.5 (i.e., absent)

The frequency of the word “his” in the tweet is > 0.5 (i.e., present)



Evaluate the Automated Filter Using Testing Data

```
X_test_word_vect = count_vect.transform(X_test)
```

Transforms the testing tweets into their word features

```
print(X_test_word_vect.shape)
```

```
(7, 273)
```

```
predicted_y = clf.predict(X_test_word_vect)
```

Predict whether each testing tweet is relevant or irrelevant.

```
print(predicted_y)
```

```
[1 1 0 0 0 0 0]
```

```
print(y_test)
```

```
[1, 1, 1, 1, 1, 0, 0]
```

```
np.mean(predicted_y == y_test)
```

```
0.5714285714285714
```

Accuracy (Not Good; due to very small number of training data)

Classification Report

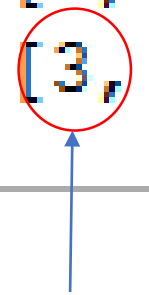
```
print(metrics.classification_report(y_test, predicted_y))
```

```
             precision    recall  f1-score   support

          0       0.40      1.00      0.57         2
          1       1.00      0.40      0.57         5

avg / total       0.83      0.57      0.57         7
```

Confusion Matrix

```
metrics.confusion_matrix(y_test, predicted_y)
```

```
array([[2, 0],
       [3, 2]], dtype=int64)
```



False Negative (relevant tweets classified to be irrelevant) too high.

Why Evaluate the Model using a data set DIFFERENT from that used for training?

Use training data to evaluate a predictive model introduces risk for overfitting.

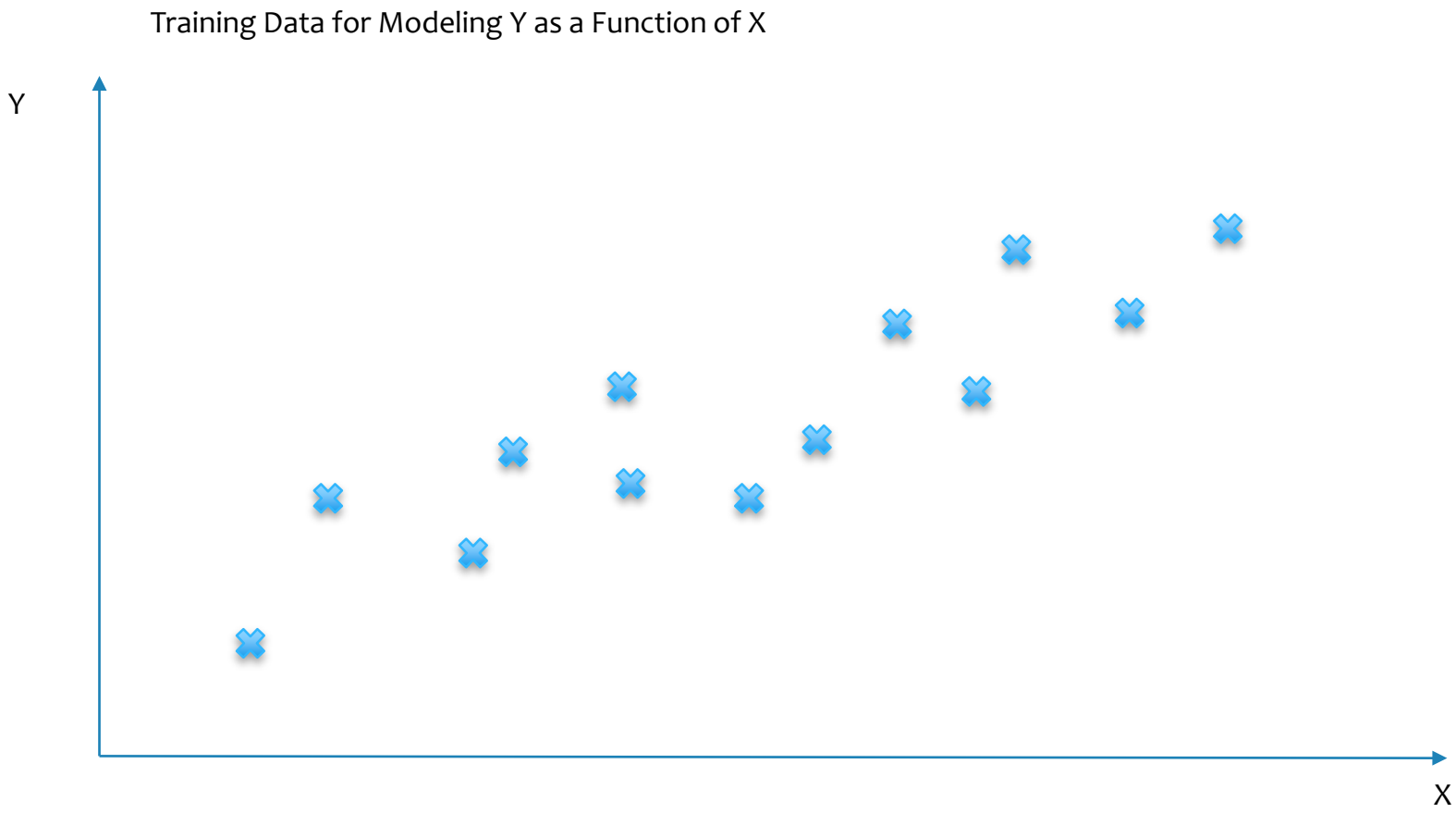
Why data scientists do not go to tailors?

- Because they are afraid of "over-fitting"

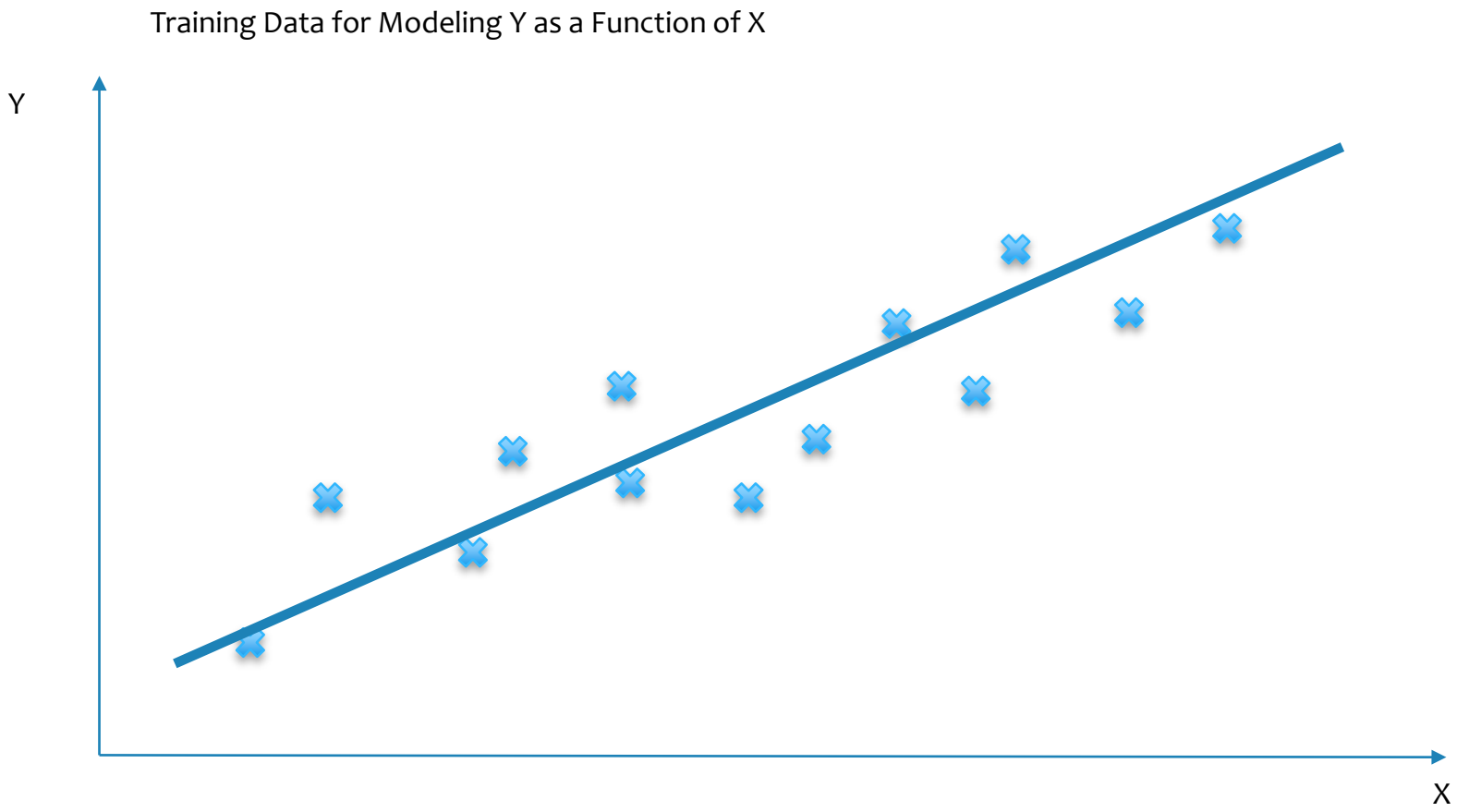
Generalizability of Model

- Avoid the potential problem of “over-fitting”
- Over-fitting: Fit a model TOO much to the training data that it does NOT generalize to data not used for training the model.

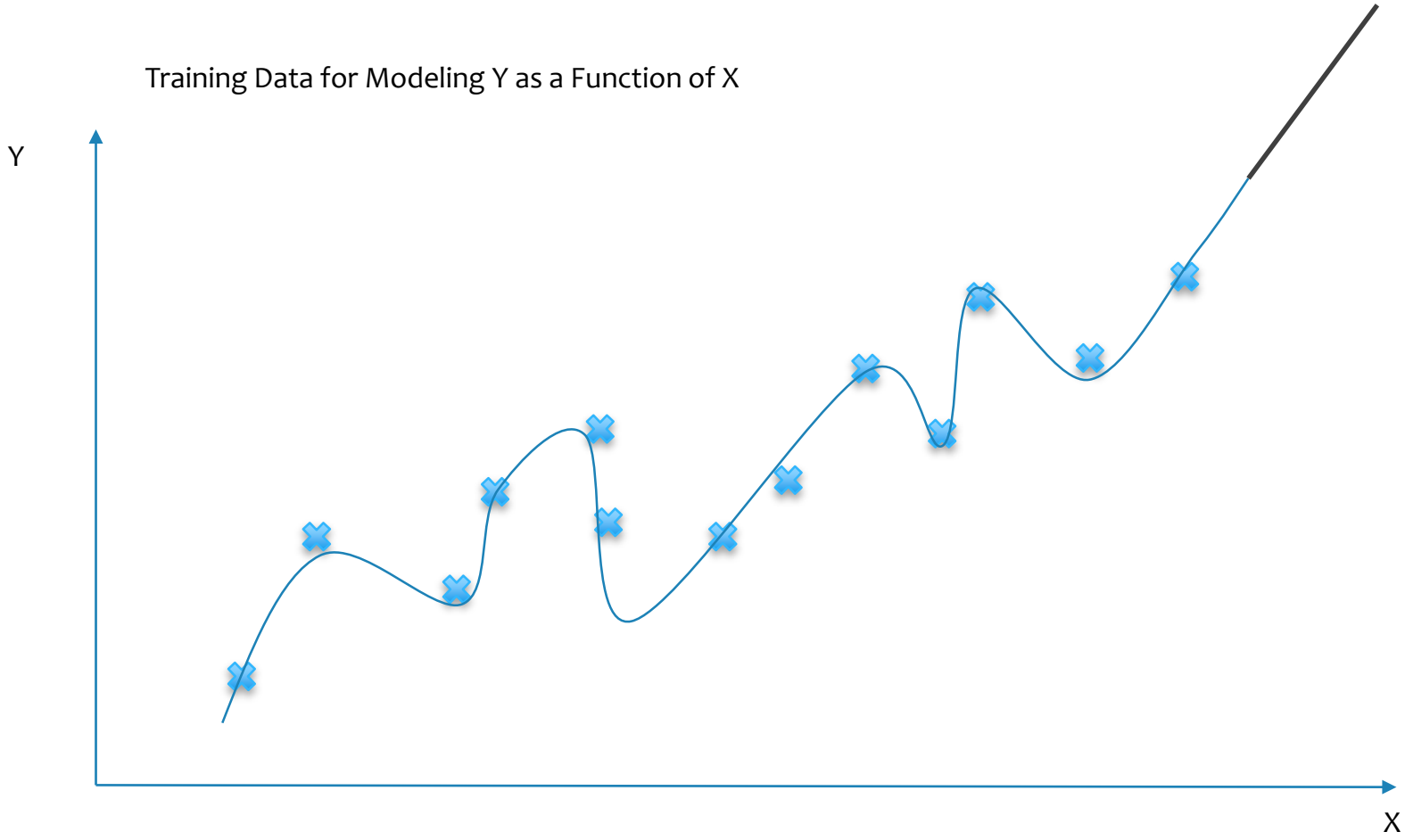
An Example



An Example



An Example of Overfitting

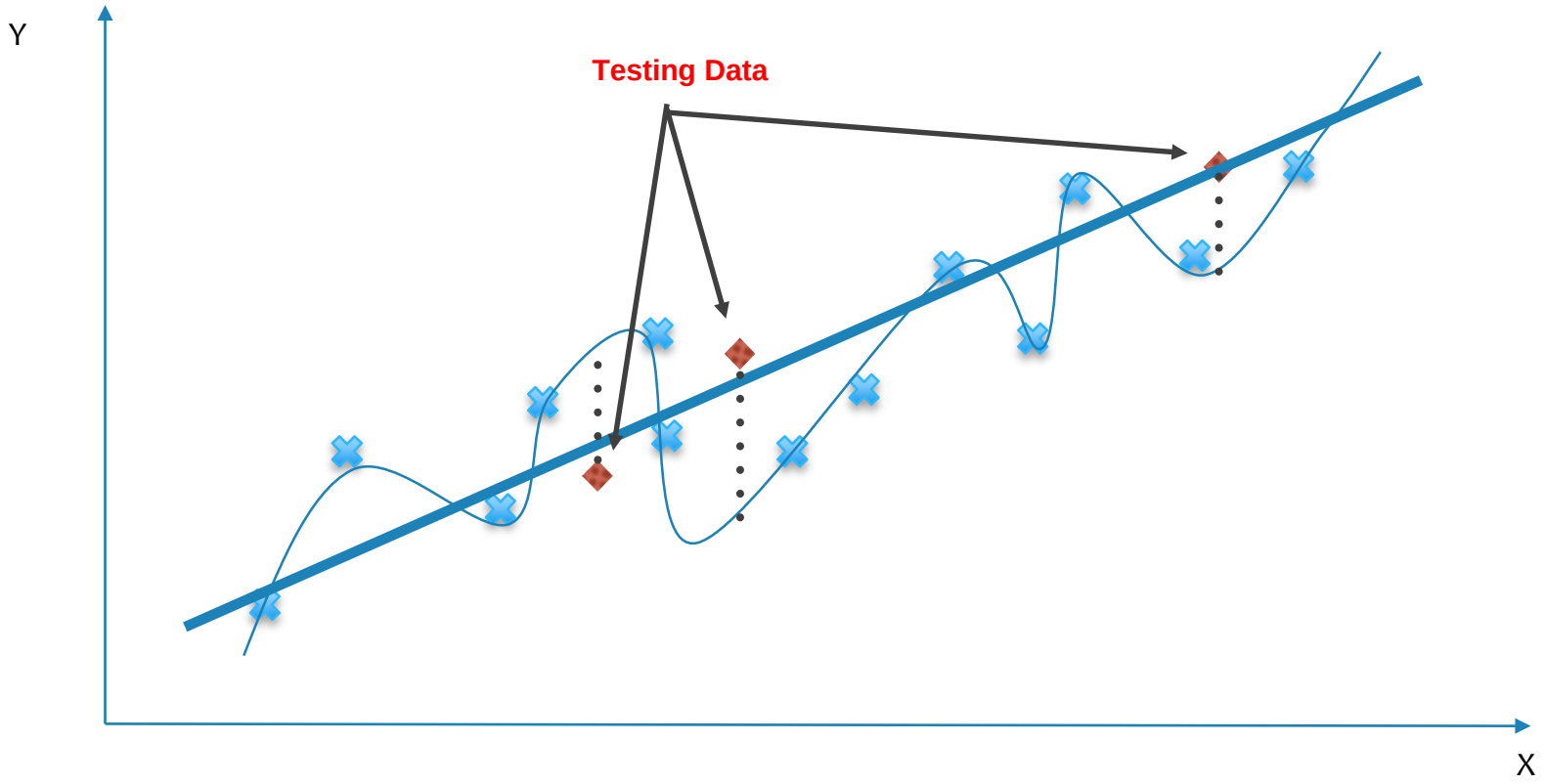


How to reduce the risk of overfitting?

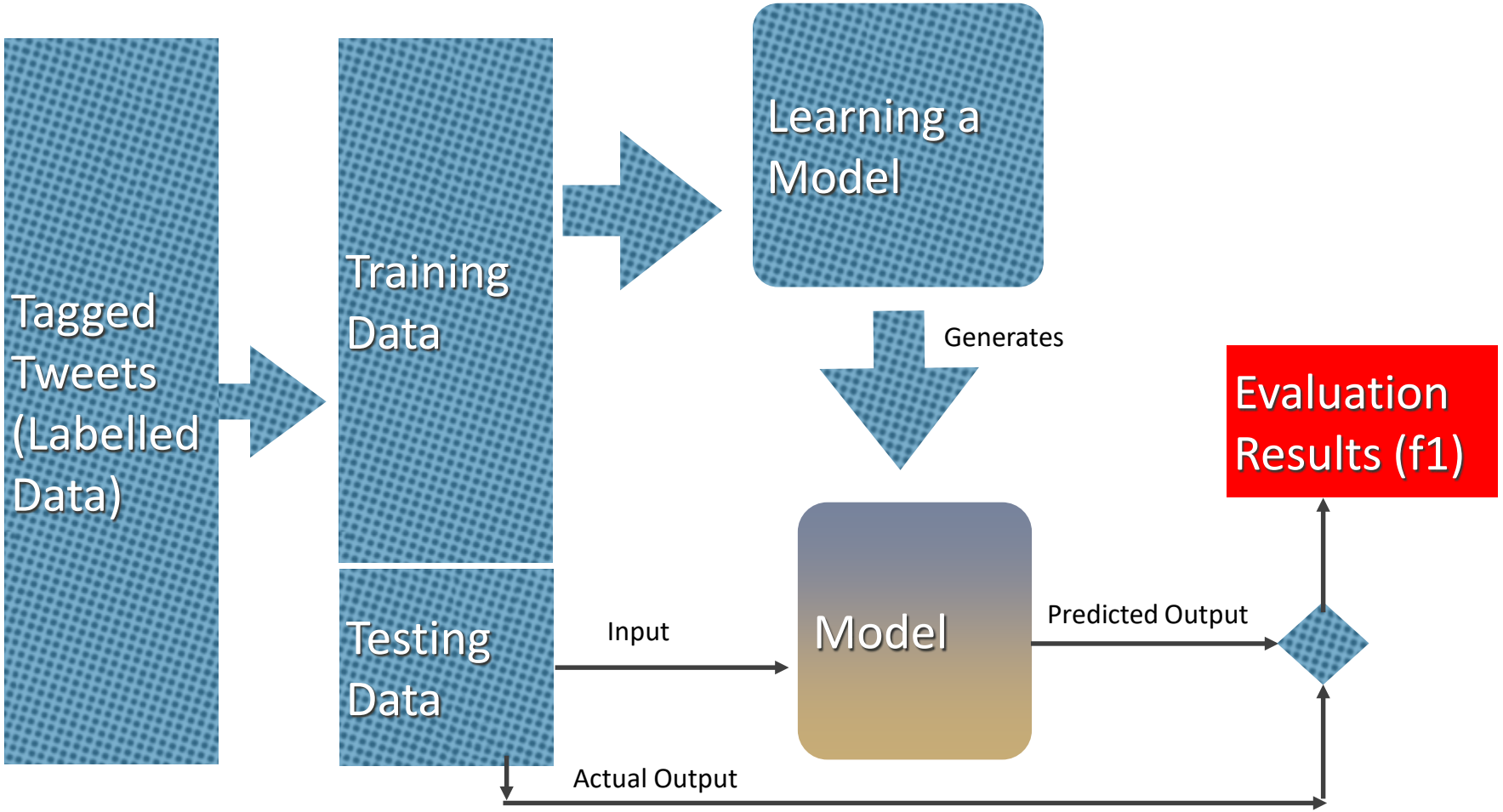
- Evaluate the model using data not used in training

Evaluating a Model by Data not used in Training

Applying untrained data (i.e., testing data) to a predictive model can reveal a large prediction error for an overfitted model.



Evaluating a Model Using Data NOT involved in Constructing the Model



How much of the labelled data do we use for training, how much for testing?

- Best Practice: Use 70-90% of the labelled data (i.e., tagged tweets)
- Why?
  - Using too few labelled data for training the model may not provide machine enough information to create a model with acceptable predictability.
  - Using too few labelled data for testing the model can reduce our confidence about the evaluation result.
- In this lab, we will use 80% for training, 20% for testing

- How to divide the labelled data into 80-20 split?

Two Approaches to Split the Labelled Data

1. Randomly sample (without replacement) 80% of labelled data for training; the remaining for testing.
2. Divide the labelled data into two groups, based on their tags (e.g., Divide the tagged tweets into Relevant tweets and Irrelevant tweets).
   - Sample 80% of labelled Relevant Tweets for training; the remaining for testing.
   - Sample 80% of labelled Irrelevant Tweets for training; the remaining for testing.

Best Practice: The Second Approach

- Benefits of the second approach:
- The percentage of relevant vs irrelevant tweets in the training data is the same as the testing data.
- This reduces the chance that training data or testing data are highly unbalanced (Will elaborate on this later).

Fit_transform generates a word frequency representation for each tweet.

A tweet can be represented by different type of “feature”

- “Bag of word” features
- “Word frequency” (WF) features
- emoji features
- Frequency of positive emotion words and negative emotion words based on a predefined “dictionary” (e.g., LIWC Dictionary).
- A combination of the above (e.g., WF + emoji, WF + LIWC)

A Word Frequency Representation of Tweets

- A mathematical representation of the words in a text.
- A row of a two-dimension array
- Each column of the array corresponds to a word or a hashtag.
- The value of each entry in the array indicates whether the tweet contains the word/hashtag corresponding to the column (1: present, 0: not present).

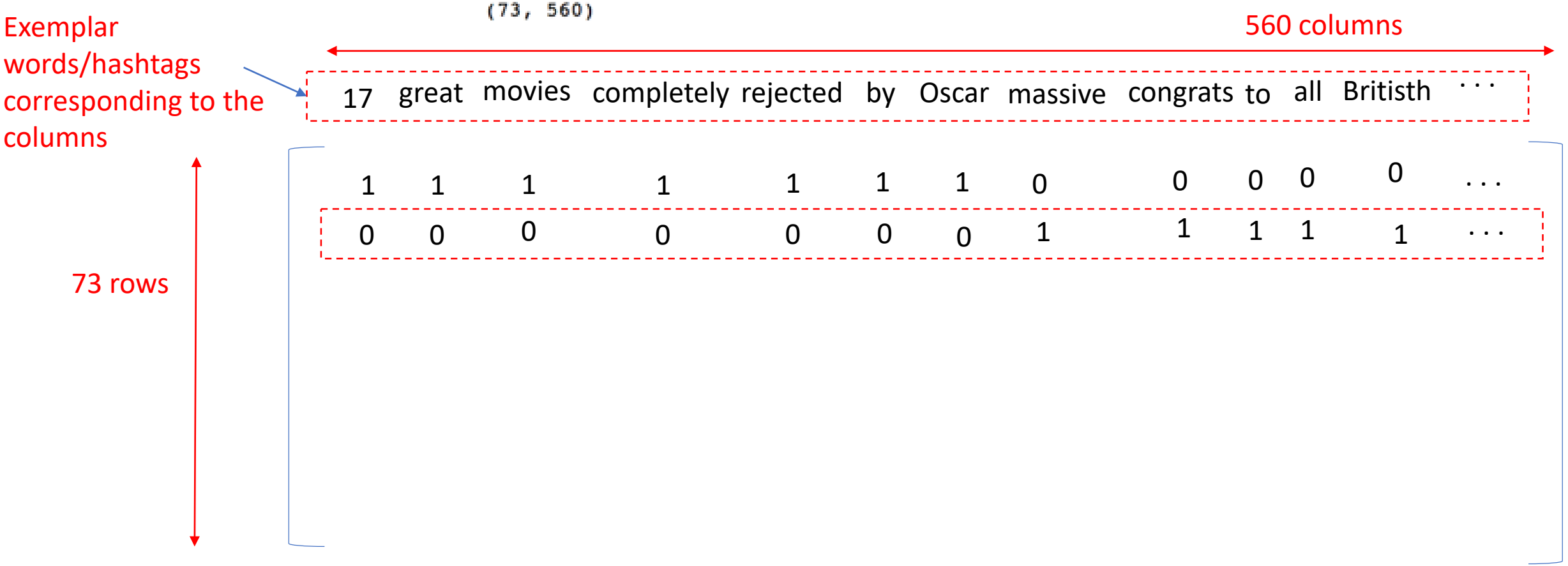
- Tweet1: “Congratulations #Philadelphia Eagles”
- Tweet2: “I am so happy that Eagles won the #SuperBowl.”
- Tweet3: “#foley fantastic”
- Tweet4: “A super win for super Eagles.”

An Example of Word Frequency Features of Tweets

	I	am	so	happy	that	Eagles	won	Congratulations	#Philadelphia	#foley	the	fantastic	#SuperBowl	super	a	win	for	....
Tweet1 →	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	
Tweet2 →	1	1	1	1	1	1	1	0	0	0	1	0	1	0	0	0	0	
Tweet3 →	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	
Tweet4 →	0	0	0	0	0	1	0	0	0	0	0	0	0	2	1	1	1	

fit_transform generates a word frequency feature matrix for the set of tweets in x_train

```
In [94]: x_train_counts = count_vect.fit_transform(x_train)

In [95]: print(x_train_counts.shape)
         (73, 560)
```



Lab5 Assignment (30 points)

- Build a Decision Tree predictive model that classifies tweets as “relevant” based on your tagging results of HW2.
- Submit a doc, docx, or PDF file containing the following information:
    1. Screenshot or Jupyter notebook showing tweets data loaded into Jupyter Notebook as Table object.
    2. Screenshots or Jupyter notebook showing the sizes of your training and testing sets (x_train, y_train, x_test, y_test).
    3. Screenshots or Jupyter notebook showing the confusion matrix of evaluating your Decision Tree-based relevant classifier using TESTING data, and identify false positive and false negative in your evaluation result.
    4. A visualization of your decision tree.
    5. A description of a rule based on the tree.
    6. A short summary of your evaluation result using the confusion matrix.