DS 200 Introduction to Data Sciences

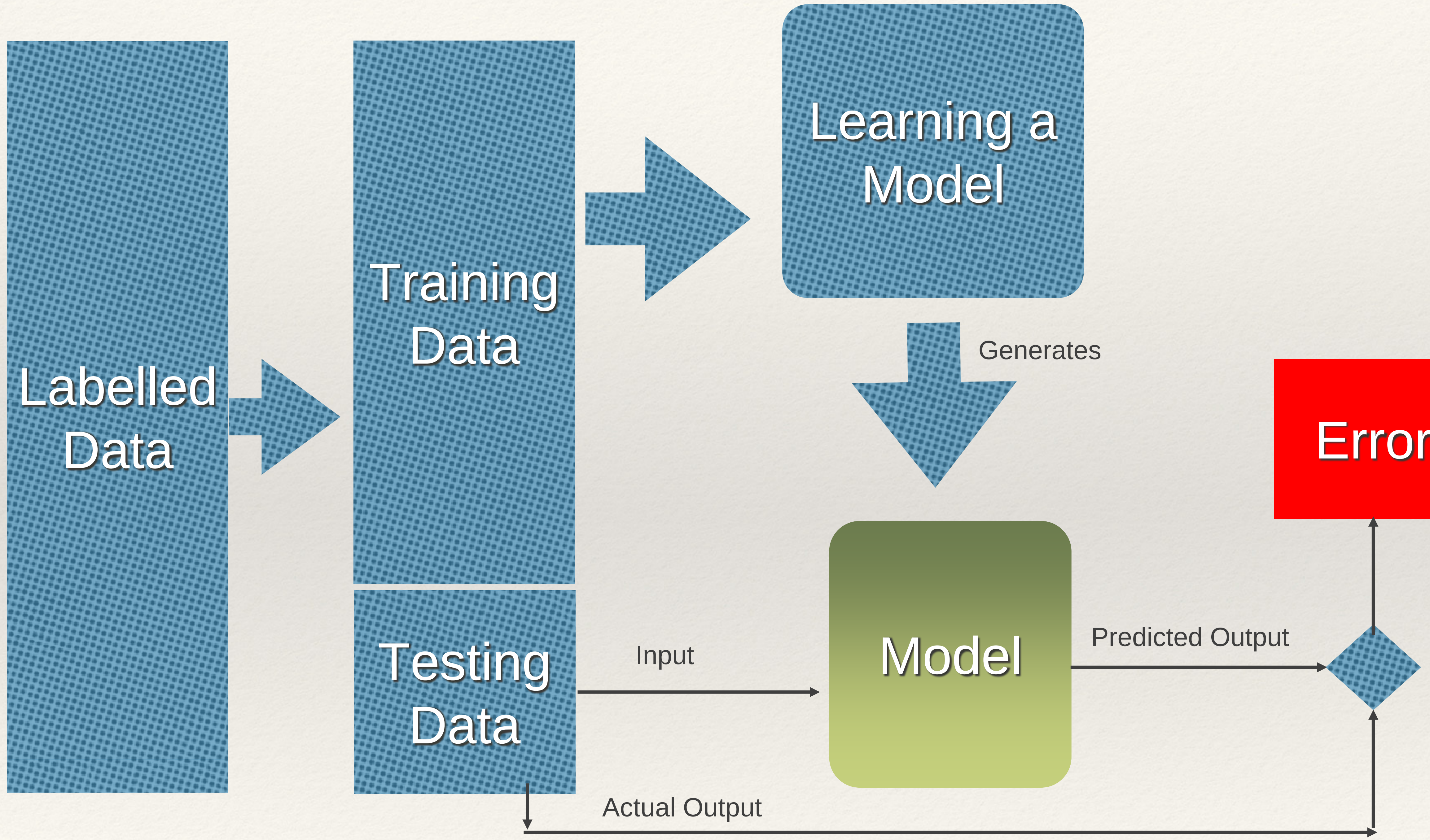
Topic 6 Lab 6: Stratified k-fold Cross Validation, f1 Measure

John Yen
Fall 2018

Learning Objectives

- Be able to understand the merit of cross-validation for a rigorous assessment of a predictive model
- Be able to use f1 measure, positive/negative predictive errors, specificity, and recalls to assess a model.
- Be able to describe the challenge of evaluating a predictive model constructed from a training data set that has a high percentage of data with one class label.
- Be able to understand the importance of “balancing the training data” and methods for achieving it.
- Be able to understand Python codes used for Stratified k-fold Cross Validation

Evaluating a Model Using Testing Data



Limitation

- The outcome of the test may depend on the particular characteristics of the testing data
  - If the testing data happens to be too easy to predict, the testing result may be better than the actual generalizability of the model.
  - If the testing data happens to be too difficult to predict, the testing result may be worse than the actual generalizability of the model.
- How to address this limitation?
- Repeat the training-testing process for different splits of the labelled (tagged) data → this is called **Cross Validation**

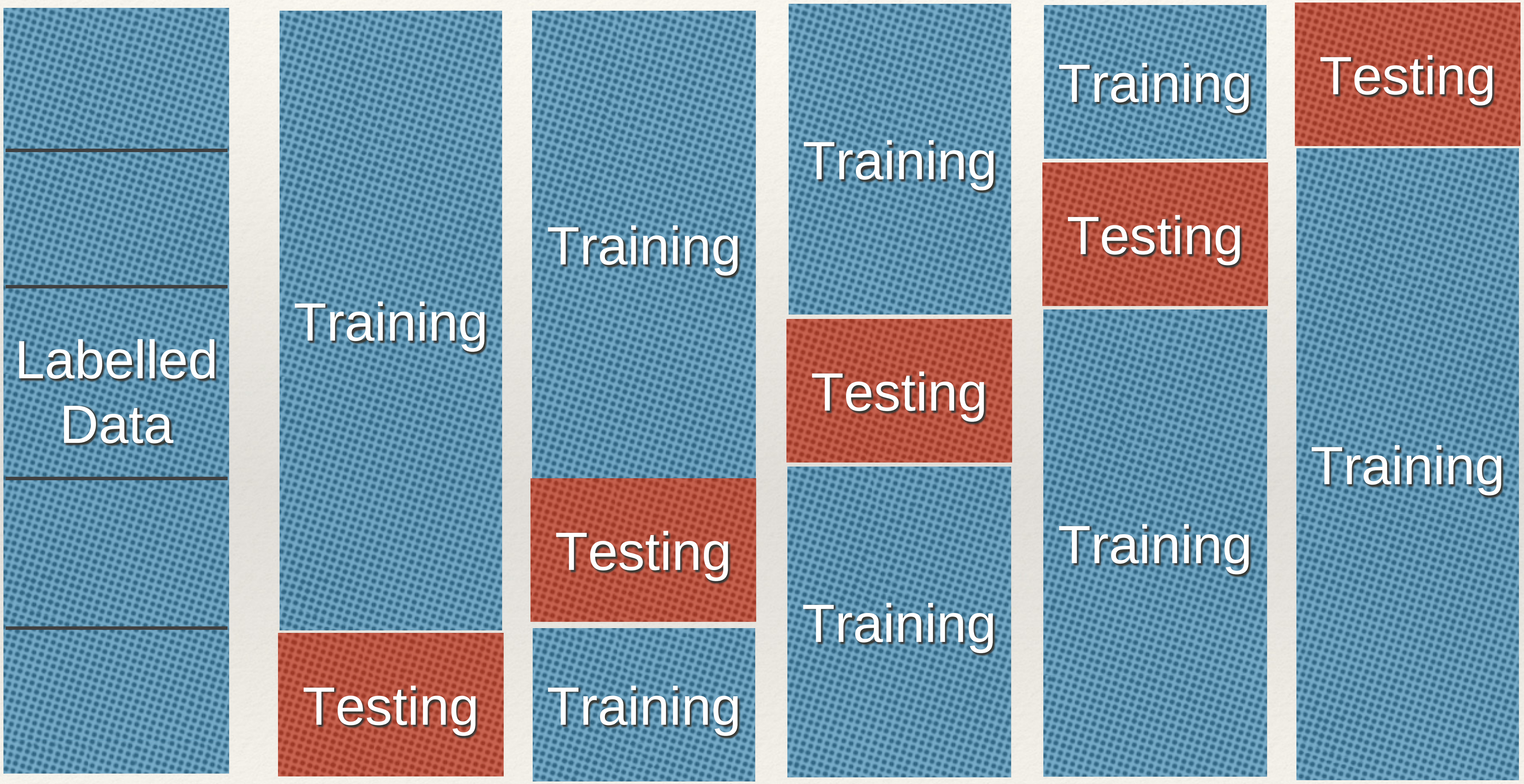
N-fold Cross Validation

1. Partition the labelled data into
2. Use N-1 portions of the labelled data to train the model
3. Use 1 portion of the labelled data to test the model.
4. Repeat steps 2 and 3 for N times, with each of the N

Cross-validation

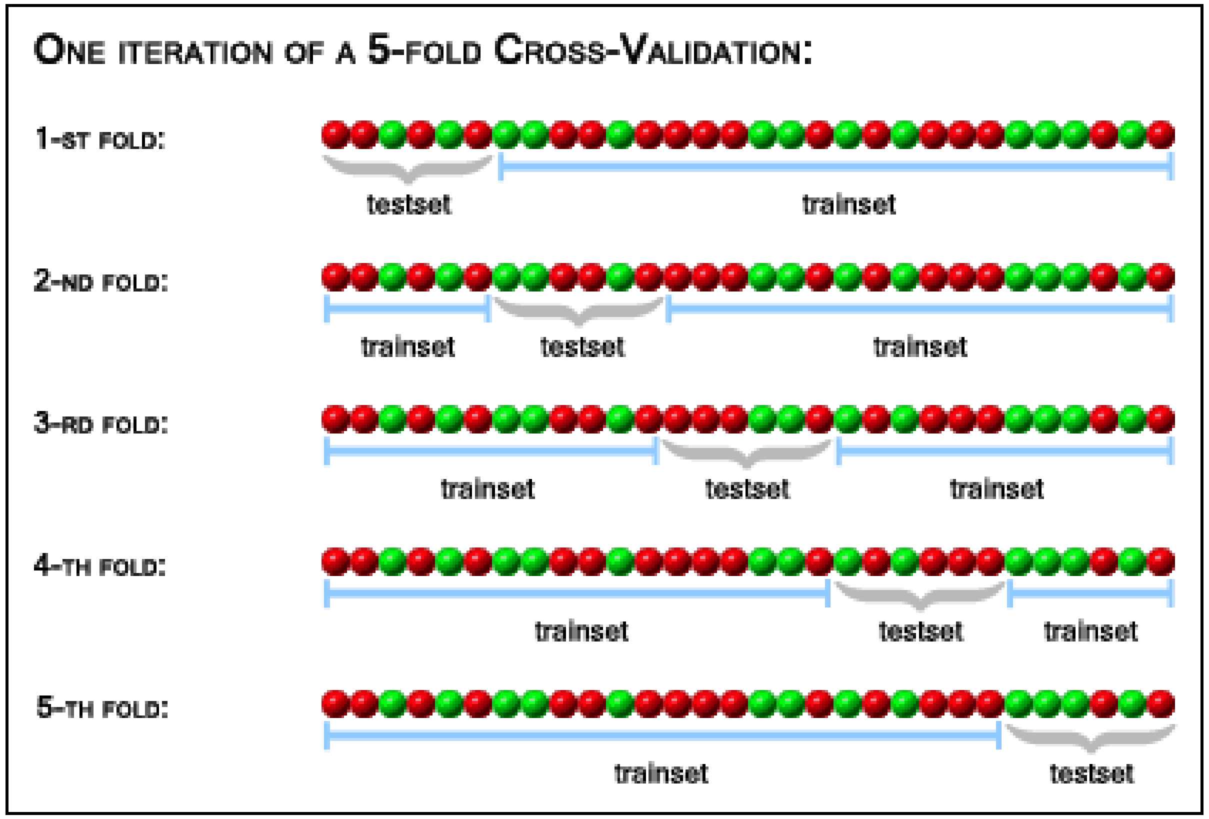
- *k*-fold cross-validation
- The original dataset is randomly partitioned into *k* equal size sub-datasets
- Of the *k* sub-datasets, one single sub-dataset is retained as the testing data, and the remaining *k-1* sub-datasets are used as training data.
- The process is repeated *k* times, with each of the *k* sub-datasets used exactly once as testing data.
- The *k* results are averaged (or combined) to produce a single estimation.

5-Fold Cross-validation



An Example of 5-fold Cross Validation

- Red: negative training instance
- Green: positive training instance



Typical Choices of Folds

- Between 5 fold and 10 fold
- 5 fold: 80% training, 20% testing; repeat 5 times.
- 10 fold: 90% training, 10% testing; repeat 10 times

iClicker Q

- Which one of the following is the “toughest” evaluation of a predictive model?
- A) 5-fold cross validation
- B) 6-fold cross validation
- C) 7-fold cross validation
- D) 8-fold cross validation
- E) 10-fold cross validation

Benefit of cross-validation

- Give an indication of how well the model (classifier) will do when it is asked to make new predictions for data it has not already seen

- Advantage over simple one time train-test split
  - The evaluation of train-test division may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made.
  - The evaluation result of cross-validation is <u>more reliable</u>.

Three Methods To Do Cross Validation in scikit-learn

1. Use cross_val_score
   - Simplest, but only provide accuracy measure

2. Use cross_validate
   - Can provide precision, recall, and f1 measures
   - Can not control the partition of the labelled datasets such that each fold is balanced

3. Use StratifiedKFold (recommended)
   - Control the partition of the labelled datasets such that each fold is balanced
   - Can calculate precision, recall, and f1 measures
   - Can show the decision trees generated in each fold

Method 1: Use cross_val_score in scikit-learn

In [10]:
```
from datascience import *

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
from sklearn import tree
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split

import pandas as pd
import numpy as np
```

Method 1 of Cross-validation in scikit-learn

- Use cross_val_score method
- Calculate accuracy of each fold

```
8]: # Method1: Using cross_val_score
clf = Pipeline([('vect', CountVectorizer()),
                ('clf', tree.DecisionTreeClassifier())])
scores = cross_val_score(clf, data_X, data_y, cv=5)

print('scores:', scores)
print('Mean of accuracy:', scores.mean())
print('SD of accuracy:', scores.std())
```

```
scores: [ 0.77777778  0.83870968  0.73770492  0.86885246  0.70491803]
Mean of accuracy: 0.785592573007
SD of accuracy: 0.0610185469551
```

Number of folds *(k) of cross validation (cv)*

Method 2 of Cross Validation: Use cross_validate

- Can choose what “scoring” metrics to use by importing relevant metrics

from sklearn.metrics import f1_score

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.model_selection import cross_validate

```
clf = Pipeline( [('vect', CountVectorizer()),
                ('clf', tree.DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_leaf=2))])

# The scoring variable stores the array of measures we want to generate during cross validation
# These measures are already defined in sklearn.metrics
# The import command "from sklearn.metrics import f1_score" in the first cell of this notebook
# enables us to use f1 as one of the scoring metric.
# Similarly, the import of precision_score and recall_score enable us to use precision and recall
# as the scoring metrics.

scoring = ['precision', 'recall', 'f1']
scores= cross_validate( clf, data_tagged_X, data_tagged_Y, scoring=scoring, cv=5, return_train_score=True)
print(type(scores))
print(scores.keys())
```

```
<class 'dict'>
dict_keys(['fit_time', 'score_time', 'test_precision', 'train_precision', 'test_recall', 'train_recall', 'test_f1', '
train_f1'])
```

```
print('Average Precision', np.mean(scores['test_precision']) )
print('Average Recall', np.mean(scores['test_recall']))
print('Average f1 score', np.mean(scores['test_f1']))
```

Method 3 of Cross Validation: Use StratifiedKFold

```
from sklearn.model_selection import StratifiedKFold
…
skf = StratifiedKFold(n_splits=5, random_state=1, shuffle= True)
for train_index, test_index in skf.split(data_tagged_X, data_tagged_Y):
    x_train= list(data.take(train_index)['text'])
    y_train= list(data.take(train_index)[‘Stance'])
    x_test= list(data.take(test_index)['text'])
    y_test= list(data.take(test_index)[‘Stance'])

    clf.fit(x_train, y_train)

    predicted_y_test = clf.predict(x_test)

    precision.append(precision_score(y_test, predicted_y_test))
    recall.append(recall_score(y_test, predicted_y_test))
     f1.append(f1_score(y_test, predicted_y_test))
    print("Precision Score:", precision_score(y_test, predicted_y_test ))
    print("Recall Score:", recall_score(y_test, predicted_y_test))
    print("f1 Score:", f1_score(y_test, predicted_y_test))
```

Assessing a Model beyond Accuracy

	Predicted Negative	Predicted Positive
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

- Positive predictive value (also called “Precision”) = TP / (TP + FP)
- Negative predictive value = TN / (TN + FN)
- Sensitivity (also called “Recall”) = TP / (TP + FN)

Can we use 1 measure to evaluate a model?

- F1 measure = 2 / (1/ Precision + 1/Recall)
- = 2 TP / (2 TP + FP + FN)
- F1 measure is higher if
  - TP is high
  - FP and FN is low

iClicker : Which of the following assessment results is correct for the confusion matrix?

	Predicted Against Trump	Predicted Support Trump
Actual Against Trump	30	20
Actual Support Trump	50	300

A) Positive Predictive Value is low
B) Negative Predictive Value is low
C) Sensitivity/Recall is low
D) F1 measure is low
E) None of the above

Two Principles for Evaluating Classification Models

- Classification Performance
  - High Accuracy
  - Low False Positive
  - Low False Negative
  - Be aware of their tradeoff: Reducing False Positive can Increase False Negative
- Generalization Capability
  - How well the model can be “generalized” to data NOT used for training?

Unbalanced Testing Data

- An Example of Tagged Stance of Tweets (among 300 Relevant tweets)
- Negative (Disapprove): 270 tweets
- Positive (Supportive): 30 tweets

A DT Generates the Following Confusion Matrix

	Predicted Disapprove (Negative)	Predicted Supportive (Positive)
Actual Disapprove (Negative)	250	20
Actual Supportive (Positive)	15	15

- What is Accuracy?
- What is the probability that a predicted supportive tweet is actually supportive (i.e., positive predicted value)?
- What is the probability that a predicted disapproving tweet is actually disapproving (i.e., negative predicted value)?

Consider the following tweets

- Support Gun Control: 350 tweets
- Against Gun Control: 50 tweets

A DT Generates the Following Confusion Matrix

	Predicted Against GC	Predicted Support GC
Actual Against GC	30	20
Actual Support GC	50	300

- What is positive predicted value (precision) ?
- What is negative predicted value?
- What is sensitivity (recall) ?

The Problem of Unbalanced Training Data

- When the training data is highly unbalanced (i.e., one class is much larger than the other in the training set), it creates two problems.

1. The model lacks examples to learn for the class that has less training data.
2. The accuracy of the model can be “misleading”, if we are not careful.

Techniques for Balancing the Training Data

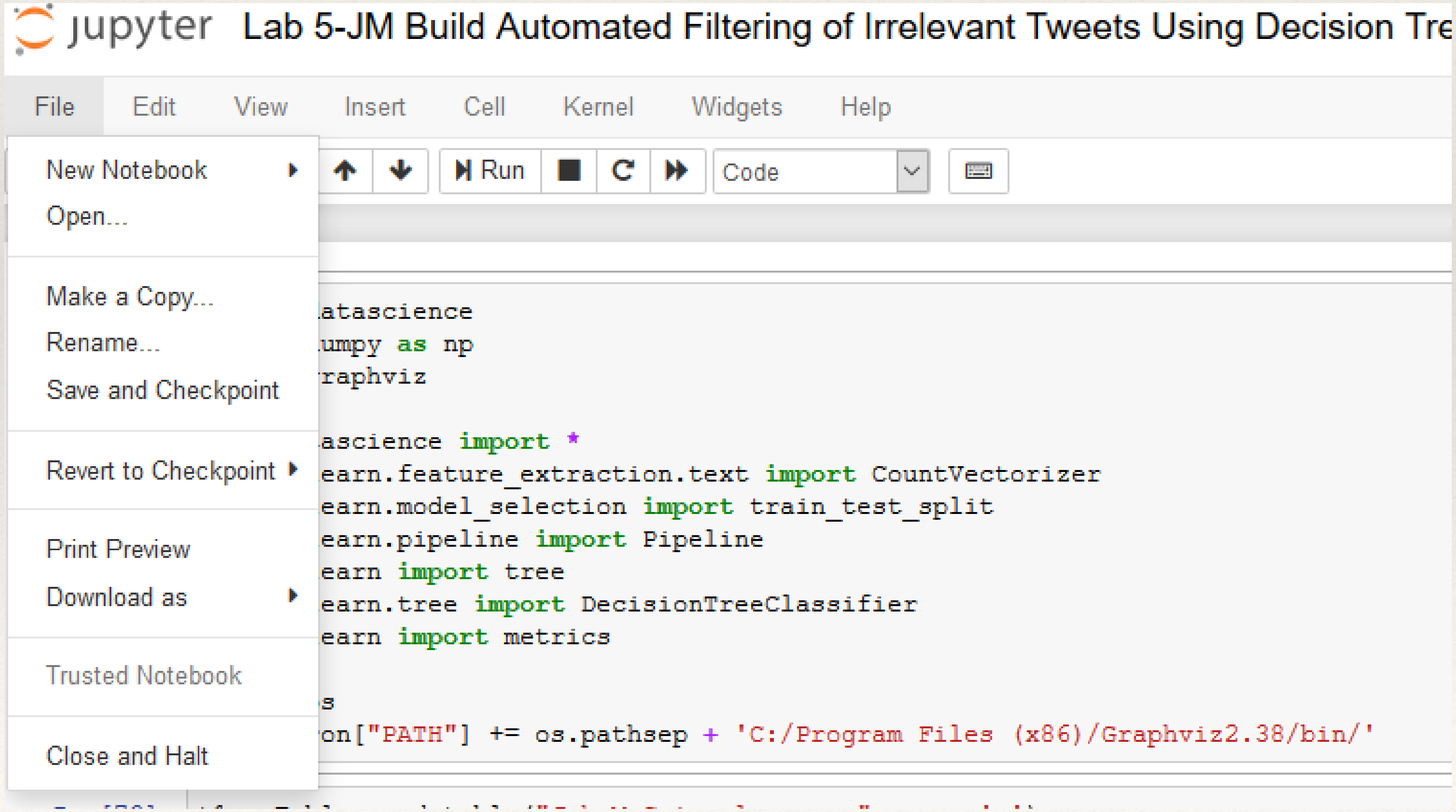
1. Gather and label more data to add additional training data of the “minority group” until the size of the two classes are about the same.
2. Use training data of the minority group multiple times in building the model (called “resampling”).
3. The first approach is preferred.

Lab 6

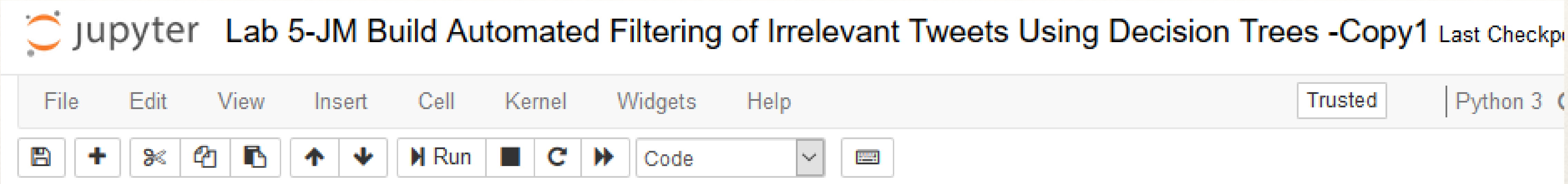
- Construct a Stance Classification decision tree using your tagged twitter data.
- Use Stratified K-fold Cross Validation to evaluate your model (k=5)
- Generate a visualization of the Decision Tree for each fold
- Assess the result of 5-fold cross validation

Steps

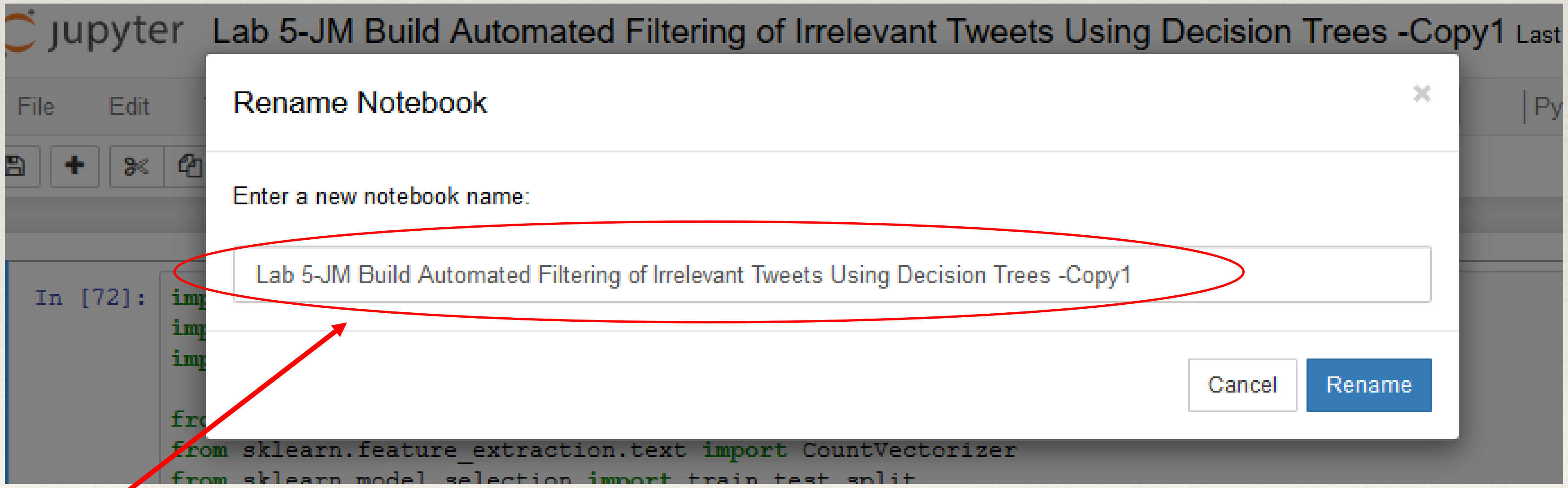
1. Open the Jupyter Notebook from Lab 5, Make a Copy of the Jupyter Notebook (File -> Make a Copy)



Rename Jupyter Notebook



- Change the Name of the copied Jupyter Notebook by clicking on the name.



Type a new name (e.g., “Lab 6 Stance Classification and Stratified k-fold CV”)

Import StratifiedKFold and related metrics

```
]: import datascience
   import numpy as np
   import graphviz

   from datascience import *
   from sklearn.feature_extraction.text import CountVectorizer
   from sklearn.model_selection import train_test_split
   from sklearn.pipeline import Pipeline
   from sklearn import tree
   from sklearn.tree import DecisionTreeClassifier
   from sklearn import metrics


   from sklearn.metrics import f1_score
   from sklearn.metrics import precision_score
   from sklearn.metrics import recall_score
   from sklearn.model_selection import cross_val_score
   from sklearn.model_selection import cross_validate
   from sklearn.base import ClassifierMixin
   from sklearn.model_selection import StratifiedKFold
   from sklearn.metrics import precision_recall_fscore_support

   import pandas as pd
   import numpy as np

   from sklearn.externals import joblib

   import os
   os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
```

Select Tweets with Stance Tag

```
t1 = Table.read_table("JohnMcCain_clean.csv", sep =',')
```

```
t1_positive = t1.where('Stance', are.equal_to(1))
t1_negative = t1.where('Stance', are.equal_to(0))
t1_positive
```

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
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
1.02e+18	Rayne	Mon Aug 27 21:21:06 +0000 2018	United States	@brooklyn3r @RichPrice65 @FoxNews @POTUS @SenJohnMcCain ...	1	1
1.03e+18	Grace	Mon Aug 27 21:19:09 +0000 2018	Arizona USA	@CBSEveningNews Too little too late. President Trump dis ...	1	1
7.01912e+08	omgitsnaname	Mon Aug 27 21:20:06 +0000 2018	God's Country	@cheetofacts @Acosta DJT Horrible Deal for Our Country! ...	1	1

Create a new table for all tweets with a Stance tag

- Appends t1_positive and t1_negative into a new table (data) that contains all the tweets with a stance tag as 1 (positive) or a stance tag as 0 (negative)

```
data = t1_positive.append(t1_negative)
```

The new table does not contain any irrelevant tweets, which do not have stance tags.

Obtain Text and Tags of Labelled Tweets

```
# Use the data for constructing a Decision Tree Stance Classifier
data_tagged_X= list(data['text'])
data_tagged_Y= list(data['Stance'])

print('tagged data input size', len(data_tagged_X))
print('tagged data target prediction size', len(data_tagged_Y))
```

```
tagged data input size 26
tagged data target prediction size 26
```

Setup Stratified K-Fold Cross Validation

```
# Use Stratified Kfold Cross Validation so that
#   each fold contains the same ratio of positive/negative instances
skf = StratifiedKFold(n_splits=2, random_state=1, shuffle= True)
```

This determines the number of folds (i.e., the value k) for a Stratified k-fold cross validation.

We use 2-fold because the small number of labelled data.

Recommend to choose k between 5 and 10.

Use for loop to iterate k time

```
fold = 1
# The following three lists record the precision, recall, and f1_score of each fold
precision = []
recall= []
f1=[]
for train_index, test_index in skf.split(data_tagged_X, data_tagged_Y):
    print("Fold Number:", fold)
 #   print("Training Data Index:", train_index)
    print("Testing Data Index:", test_index)

    x_train= list(data.take(train_index)['text'])
#    print("Training Data:", x_train)
    y_train= list(data.take(train_index)['Stance'])
#    print("Training Data Target Output:", y_train)
    x_test= list(data.take(test_index)['text'])
#    print("Testing Data:", x_test)
    y_test= list(data.take(test_index)['Stance'])
    print("Testing Data Target Output:", y_test)

    count_vect = CountVectorizer()
    X_word_vect = count_vect.fit_transform(x_train)
    clf = tree.DecisionTreeClassifier(criterion='entropy', random_state = 100, max_depth=15, \
                                min_samples_leaf =2)
    clf.fit(X_word_vect, y_train)
```

Each Iteration Calculates precision, recall, and f1 score

```
dot_data= tree.export_graphviz(clf, out_file=None, feature_names=count_vect.get_feature_names())
graph = graphviz.Source(dot_data)
file_name = 'StanceClassifier-Fold' + str(fold)
graph.render(file_name)
print("Saving Decision Tree Visualization in ", file_name)

x_test_word_vect = count_vect.transform(x_test)
predicted_y_test = clf.predict(x_test_word_vect)
print("Testing Data Prediction Output:", predicted_y_test)
# Add the precision, recall, and f1_score of the fold to the list
precision.append(precision_score(y_test, predicted_y_test))
recall.append(recall_score(y_test, predicted_y_test))
f1.append(f1_score(y_test, predicted_y_test))
print("Precision Score:", precision_score(y_test, predicted_y_test ))
print("Recall Score:", recall_score(y_test, predicted_y_test))
print("f1 Score:", f1_score(y_test, predicted_y_test))
fold=fold+1
```

A visualization of the tree is saved in a file in each fold called 'StanceClassifier-Fold1', 'StanceClassifier-Fold2', …

```
Fold Number: 1
Testing Data Index: [ 1  2  3  4  6  7 10 13 16 17 18 19 21 24]
Testing Data Target Output: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Saving Decision Tree Visualization in  StanceClassifier-Fold1
Testing Data Prediction Output: [1. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 1. 1.]
Precision Score: 0.375
Recall Score: 0.375
f1 Score: 0.375
```

The list of precision, recall, and f1

```
print("Precision List:", precision)
print("Recall List:", recall)
print("f1 List:", f1)
```

```
Precision List: [0.375, 0.4]
Recall List: [0.375, 0.2857142857142857]
f1 List: [0.375, 0.3333333333333333]
```

First Fold

Second Fold

Lab 6 (45 points)

- Submit the following in ONE word or PDF file
- The Screenshots of the Jupyter Notebook for Stratified K-fold Cross Validation of Decision Trees for Stance Classification
- Discuss the result of the k-fold cross validation (using the list of precision, recall, and f1 score).
- PDF files of all the visualization of stance classifiers.
- Identify one or two common features you found in multiple decision trees and their roles in the tree.
- Due: 10 pm, October 1st