Open in Colab

In [ ]:

Project 3 - Name Gender Classifier

by Don Padmaperuma & Grace Han

Assignment

Using any of the three classifiers described in chapter 6 of Natural Language Processing with Python, and any features you can think of, build the best name gender classifier you can.

Begin by splitting the Names Corpus into three subsets: 500 words for the test set, 500 words for the dev-test set, and the remaining 6900 words for the training set.

Then, starting with the example name gender classifier, make incremental improvements.

Use the dev-test set to check your progress. Once you are satisfied with your classifier, check its final performance on the test set.

How does the performance on the test set compare to the performance on the dev-test set? Is this what you'd expect?

In [1]:
```
import collections
import nltk
nltk.download('names')
from nltk.corpus import names
from nltk.classify import apply_features
from nltk.metrics import ConfusionMatrix, accuracy, precision, recall, f_measure
import random
```

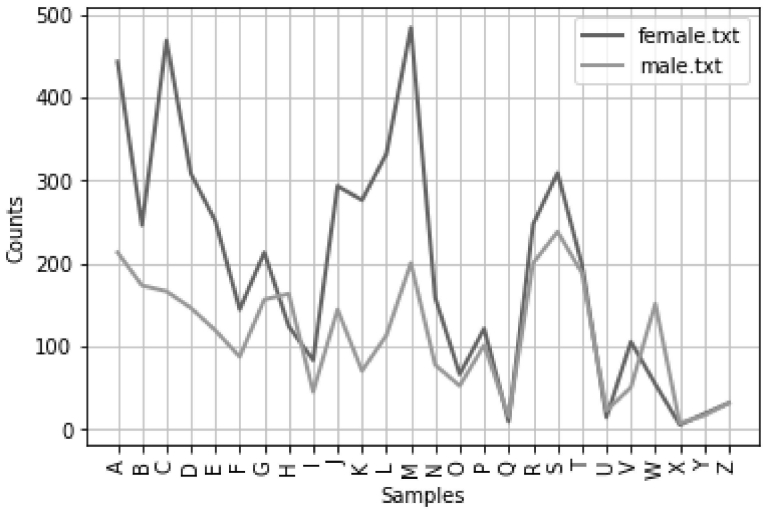
```
[nltk_data] Downloading package names to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package names is already up-to-date!
```

In [2]:
```
len(names.words('female.txt')), len(names.words('male.txt'))
```

Out[2]: (5001, 2943)

We can see from above numbers that there are more female names (almost twise as male names) compared to the male names in the dataset.

In [3]:
```
# Check distribution of the first letter of the name.
gender_freq = nltk.ConditionalFreqDist((fileid, name[0])
  for fileid in names.fileids()
  for name in names.words(fileid))
gender_freq.plot()
```



Out[3]: <AxesSubplot:xlabel='Samples', ylabel='Counts'>

Above graph shows the comparison of distribution of beginning letter of the names by gender. We can see that more female names starts with letter A, C, M and S.

In [4]:
```
# combine male and female names and shuffle them.
names = ([(name, 'male') for name in names.words('male.txt')] + \
        [(name, 'female') for name in names.words('female.txt')])
random.shuffle(names)
```

In [5]:
```
# Take a look at the combined data
names[1:5]
```

Out[5]:
```
[('Lebbie', 'female'),
 ('Gabbey', 'female'),
 ('Boyd', 'male'),
 ('Cathi', 'female')]
```

In [6]:
```
# unique names
len(set(item[0] for item in names))
```

Out[6]: 7579

We removed the names that was listed as both males and females and was listed twice.

In [7]:
```
# names that are not unique
names_only = [item[0] for item in names]
names_dist = nltk.FreqDist(names_only)
names_duplicates = [(k,v) for k,v in names_dist.items() if v>1 ]
names_duplicates
```

Out[7]:
```
[('Julie', 2),
 ('Sydney', 2),
 ('Terry', 2),
 ('Adrian', 2),
 ('Christy', 2),
 ('Maddie', 2),
 ('Kelley', 2),
 ('Daniel', 2),
 ('Scotty', 2),
 ('Nikki', 2),
 ('Germaine', 2),
 ('Sibyl', 2),
```

```
('Demetris', 2),
('Timmie', 2),
('Carmine', 2),
('Sayre', 2),
('Kellen', 2),
('Penny', 2),
('Clair', 2),
('Sunny', 2),
('Gene', 2),
('Kim', 2),
('Mattie', 2),
('Beau', 2),
('Jamie', 2),
('Eddy', 2),
('Addie', 2),
('Theo', 2),
('Nickie', 2),
('Glen', 2),
('Niki', 2),
('Dell', 2),
('Alfie', 2),
('Lonnie', 2),
('Jan', 2),
('Tammie', 2),
('Tracie', 2),
('Alex', 2),
('Jodie', 2),
('Tally', 2),
('Lindsey', 2),
('Connie', 2),
('Casey', 2),
('Merrill', 2),
('Angie', 2),
('Simone', 2),
('Pen', 2),
('Gayle', 2),
('Billie', 2),
('Mead', 2),
('Leslie', 2),
('Gretchen', 2),
('Sonnie', 2),
('Michal', 2),
('Pat', 2),
('Dannie', 2),
('Haleigh', 2),
('Kerry', 2),
('Donnie', 2),
('Cody', 2),
('Clem', 2),
('Sean', 2),
('Hillary', 2),
('Andy', 2),
('Morgan', 2),
('Lesley', 2),
('Brett', 2),
('Martie', 2),
('Shay', 2),
('Ashley', 2),
('Timmy', 2),
('Whitney', 2),
('Chris', 2),
('Devon', 2),
('Van', 2),
('Carroll', 2),
('Bryn', 2),
```

```
('Kris', 2),
('Denny', 2),
('Rikki', 2),
('Kirby', 2),
('Ruby', 2),
('Joey', 2),
('Hazel', 2),
('Adrien', 2),
('Teddy', 2),
('Ikey', 2),
('Judy', 2),
('Nichole', 2),
('Randi', 2),
('Cecil', 2),
('Christie', 2),
('Jess', 2),
('Abbey', 2),
('Erin', 2),
('Wallie', 2),
('Drew', 2),
('Tracy', 2),
('Claude', 2),
('Willie', 2),
('Cass', 2),
('Cam', 2),
('Eddie', 2),
('Lyn', 2),
('Val', 2),
('Cory', 2),
('Gail', 2),
('Tobe', 2),
('Quentin', 2),
('Trace', 2),
('Sammy', 2),
('Claire', 2),
('Constantine', 2),
('Ricki', 2),
('Clare', 2),
('Tammy', 2),
('Caryl', 2),
('Quinn', 2),
('Gale', 2),
('Brook', 2),
('Jessie', 2),
('Shea', 2),
('Danny', 2),
('Wally', 2),
('Stacy', 2),
('Vinny', 2),
('Leland', 2),
('Bo', 2),
('Randie', 2),
('Kit', 2),
('Justin', 2),
('Jerrie', 2),
('Morlee', 2),
('Sonny', 2),
('Barrie', 2),
('Tate', 2),
('Bobby', 2),
('Teddie', 2),
('Shayne', 2),
('Gay', 2),
('Tallie', 2),
('Gerri', 2),
```

```
('Sasha', 2),
('Rene', 2),
('Geri', 2),
('Dory', 2),
('Winnie', 2),
('Glenn', 2),
('Ike', 2),
('Bill', 2),
('Brooke', 2),
('Gill', 2),
('George', 2),
('Frank', 2),
('Chad', 2),
('Haley', 2),
('Pooh', 2),
('Lynn', 2),
('Ginger', 2),
('Averil', 2),
('Judith', 2),
('Rey', 2),
('Shaine', 2),
('Brandy', 2),
('Esme', 2),
('Willi', 2),
('Nat', 2),
('Lou', 2),
('Mickie', 2),
('Chrissy', 2),
('Bennie', 2),
('Mel', 2),
('Robbie', 2),
('Juanita', 2),
('Dale', 2),
('Tim', 2),
('Willy', 2),
('Bert', 2),
('Holly', 2),
('Reggie', 2),
('Tabbie', 2),
('Frankie', 2),
('Lorne', 2),
('Lorrie', 2),
('Daffy', 2),
('Jermaine', 2),
('Jude', 2),
('Ronnie', 2),
('Daryl', 2),
('Lin', 2),
('Millicent', 2),
('Britt', 2),
('Henrie', 2),
('Felice', 2),
('Jody', 2),
('Nicky', 2),
('Augustine', 2),
('Regan', 2),
('Noel', 2),
('Andie', 2),
('Tony', 2),
('Wynn', 2),
('Abby', 2),
('Emmy', 2),
('Donny', 2),
('Vin', 2),
('Bernie', 2),
```

```
('Fred', 2),
('Sascha', 2),
('Virgie', 2),
('Meredith', 2),
('Jere', 2),
('Shane', 2),
('Heath', 2),
('Dennie', 2),
('Jo', 2),
('Kelly', 2),
('Wallis', 2),
('Allie', 2),
('Tommie', 2),
('Jodi', 2),
('Darryl', 2),
('Shelley', 2),
('Kip', 2),
('Francis', 2),
('Michel', 2),
('Darcy', 2),
('Darby', 2),
('Courtney', 2),
('Isador', 2),
('Micky', 2),
('Ariel', 2),
('Corey', 2),
('Fran', 2),
('Carlin', 2),
('Terri', 2),
('Pennie', 2),
('Kyle', 2),
('Saundra', 2),
('Gerry', 2),
('Alexis', 2),
('Jean', 2),
('Dana', 2),
('Corrie', 2),
('Ted', 2),
('Blake', 2),
('Christian', 2),
('Shelby', 2),
('Hannibal', 2),
('Karel', 2),
('Sandy', 2),
('Lindsay', 2),
('Tabby', 2),
('Patsy', 2),
('Meryl', 2),
('Dorian', 2),
('Lane', 2),
('Bertie', 2),
('Meade', 2),
('Randy', 2),
('Isadore', 2),
('Lindy', 2),
('Cammy', 2),
('Ricky', 2),
('Alix', 2),
('Jerry', 2),
('Aubrey', 2),
('Valentine', 2),
('Dionis', 2),
('Michele', 2),
('Angel', 2),
('Paige', 2),
```

```
('Dani', 2),
('Toby', 2),
('Winny', 2),
('Ali', 2),
('Cal', 2),
('Dominique', 2),
('Lauren', 2),
('Tracey', 2),
('Edie', 2),
('Jackie', 2),
('Patty', 2),
('Carey', 2),
('Laurie', 2),
('Shell', 2),
('Austin', 2),
('Jordan', 2),
('Devin', 2),
('Patrice', 2),
('Shawn', 2),
('Georgia', 2),
('Jesse', 2),
('Lee', 2),
('Marietta', 2),
('Vale', 2),
('Pattie', 2),
('Barry', 2),
('Kelsey', 2),
('Dion', 2),
('Hilary', 2),
('Robin', 2),
('Ollie', 2),
('Phil', 2),
('Abbie', 2),
('Isa', 2),
('Benny', 2),
('Ajay', 2),
('Elisha', 2),
('Franky', 2),
('Freddie', 2),
('Brooks', 2),
('Bobbie', 2),
('Torey', 2),
('Merle', 2),
('Billy', 2),
('Freddy', 2),
('Cris', 2),
('Clemmie', 2),
('Harley', 2),
('Ronny', 2),
('Matty', 2),
('Merry', 2),
('Tobie', 2),
('Shaun', 2),
('Sam', 2),
('Vinnie', 2),
('Deane', 2),
('Sal', 2),
('Rickie', 2),
('Gabriell', 2),
('Luce', 2),
('Marion', 2),
('Perry', 2),
('Muffin', 2),
('Georgie', 2),
('Lanny', 2),
```

```
('Carlie', 2),
('Maxie', 2),
('Rory', 2),
('Max', 2),
('Maddy', 2),
('Ray', 2),
('Cary', 2),
('Evelyn', 2),
('Leigh', 2),
('Allyn', 2),
('Blair', 2),
('Marty', 2),
('Andrea', 2),
('Cat', 2),
('Grace', 2),
('Dallas', 2),
('Jaime', 2),
('Marlo', 2),
('Lind', 2),
('Shannon', 2),
('Del', 2),
('Gus', 2),
('Loren', 2),
('Gabriel', 2),
('Page', 2),
('Tommy', 2),
('Ira', 2),
('Maurise', 2)]
```

In [8]:
```
# FROM Shovan
# Remove the duplicates and show total number of unique names
names2remove = [item[0] for item in names_duplicates]
final_names = [item for item in names if not item[0] in names2remove]
len(final_names)
```

Out[8]: 7214

In [9]:
```
names = final_names
len(names)
```

Out[9]: 7214

Split into Train, DevTest and Test Data

In [10]:
```
## from SilverN
train_set = names[1000:]        # Training set
test_set = names[:500]   # Test
devtest_set = names[500:1000]   #development test
```

In [11]:
```
## from Showvan

# Confirm the size of the three subsets
print("Training Set = {}".format(len(train_set)))
print("Dev-Test Set = {}".format(len(devtest_set)))
print("Test Set = {}".format(len(test_set)))
```

```
Training Set = 6214
Dev-Test Set = 500
Test Set = 500
```

```
## GH: this section should be placed before the 1st feature
```

In [12]:
```
## THis is the right location
## from Shovan
# Write a function for manually calculate the recall error and precision formance, for
#(to compare the automatically generated performance indicator)

def performance_metrics(model, train, digits=4):
    """Prints the precision and recall of an NLTK Naive Bayes model."""

    reference = collections.defaultdict(set)
    test = collections.defaultdict(set)

    for i, (features, label) in enumerate(train):
        reference[label].add(i)
        pred = model.classify(features)
        test[pred].add(i)

    m_precision = round(precision(reference['male'], test['male']), digits)
    f_precision = round(precision(reference['female'], test['female']), digits)

    m_recall = round(recall(reference['male'], test['male']), digits)
    f_recall = round( recall(reference['female'], test['female']), digits)

    print('Male precision: ', m_precision)
    print('Female precision: ', f_precision)
    print('Male recall: ', m_recall)
    print('Female recall: ', f_recall)
```

In [ ]:
```
```

In [ ]:
```
```

Features - First Feature

The first feature contains the first letter, last letter, prefix and suffix, as well as the vowels (aeiou, we did not include the y in here which is optional)

In [13]:
```
## from SilverN

def gender_features1(name):
    features = {}
    features["firstletter"] = name[0].lower()
    features["lastletter"] = name[-1].lower()
    features["suffix2"]= name[-2:].lower()
    features["preffix2"]= name[:2].lower()

    for letter in 'aeiou':   ## vowels
        features["count(%s)" % letter] = name.lower().count(letter)
        features["has(%s)" % letter] = (letter in name.lower())
    return features
```

In [14]:
```
featuresets = [(gender_features1(n), g) for (n,g) in names]
featuresets[0]
```

Out[14]:
```
({'firstletter': 'l',
  'lastletter': 'e',
  'suffix2': 'ie',
  'preffix2': 'le',
```

```
  'count(a)': 0,
  'has(a)': False,
  'count(e)': 2,
  'has(e)': True,
  'count(i)': 1,
  'has(i)': True,
  'count(o)': 0,
  'has(o)': False,
  'count(u)': 0,
  'has(u)': False},
 'female')
```

Split Data Based on 1st Feature

In [17]:
```
### Split data
train_set_fe = featuresets[1000:]
test_set_fe =featuresets[:500]
devtest_set_fe =featuresets[500:1000]
```

Classifier - NaiveBayes -on 1st Feature

In [18]:
```
classifier = nltk.NaiveBayesClassifier.train(train_set_fe)
print(classifier.classify(gender_features1('Stefani')))
print(classifier.classify(gender_features1('Steffan')))
```

```
female
male
```

In [19]:
```
# Show Accuracy
print("train_set: ", nltk.classify.accuracy(classifier, train_set_fe))
print("test_set: ", nltk.classify.accuracy(classifier, test_set_fe))
print("devtest_set: ", nltk.classify.accuracy(classifier, devtest_set_fe))
```

```
train_set:  0.8434180881879627
test_set:  0.824
devtest_set:  0.83
```

In [20]:
```
# Show important features
classifier.show_most_informative_features(5)
```

```
Most Informative Features
                suffix2 = 'na'           female : male   =    145.2 : 1.0
             lastletter = 'k'              male : female =     69.6 : 1.0
                suffix2 = 'la'           female : male   =     65.2 : 1.0
                suffix2 = 'ra'           female : male   =     54.7 : 1.0
                suffix2 = 'rt'             male : female =     52.6 : 1.0
```

In [21]:
```
# Check errors
## GH: May not be necessary
errors = []
for (name, tag) in devtest_set:
    guess = classifier.classify(gender_features1(name))
    if guess != tag:
        errors.append( (tag, guess, name) )
```

In [22]:
```
## GH: Not to print these out
## works wo error
for (tag, guess, name) in sorted(errors): # doctest: +ELLIPSIS +NORMALIZE_WHITESPACE
   print('correct=%-8s guess=%-8s name=%-30s' % (tag, guess, name))
```

```
correct=female   guess=male     name=Allison
```

```
correct=female   guess=male     name=Arden
correct=female   guess=male     name=Audry
correct=female   guess=male     name=Beth
correct=female   guess=male     name=Blondy
correct=female   guess=male     name=Brier
correct=female   guess=male     name=Brigid
correct=female   guess=male     name=Cher
correct=female   guess=male     name=Coriss
correct=female   guess=male     name=Corly
correct=female   guess=male     name=Darell
correct=female   guess=male     name=Debor
correct=female   guess=male     name=Doralin
correct=female   guess=male     name=Elonore
correct=female   guess=male     name=Ethelind
correct=female   guess=male     name=Frances
correct=female   guess=male     name=Gray
correct=female   guess=male     name=Gredel
correct=female   guess=male     name=Grethel
correct=female   guess=male     name=Hetty
correct=female   guess=male     name=Hildegaard
correct=female   guess=male     name=Jacquelin
correct=female   guess=male     name=Jourdan
correct=female   guess=male     name=Loreen
correct=female   guess=male     name=Mag
correct=female   guess=male     name=Maud
correct=female   guess=male     name=Michell
correct=female   guess=male     name=Molly
correct=female   guess=male     name=Murial
correct=female   guess=male     name=Nert
correct=female   guess=male     name=Nerty
correct=female   guess=male     name=Rahel
correct=female   guess=male     name=Robbin
correct=female   guess=male     name=Rochell
correct=female   guess=male     name=Sally
correct=female   guess=male     name=Sibby
correct=female   guess=male     name=Siouxie
correct=female   guess=male     name=Tatum
correct=female   guess=male     name=Tess
correct=female   guess=male     name=Tiff
correct=female   guess=male     name=Wendy
correct=female   guess=male     name=Wileen
correct=female   guess=male     name=Zarah
correct=male     guess=female   name=Adolphe
correct=male     guess=female   name=Andri
correct=male     guess=female   name=Anthony
correct=male     guess=female   name=Ashby
correct=male     guess=female   name=Ashish
correct=male     guess=female   name=Benjie
correct=male     guess=female   name=Brady
correct=male     guess=female   name=Cain
correct=male     guess=female   name=Caldwell
correct=male     guess=female   name=Carleigh
correct=male     guess=female   name=Chance
correct=male     guess=female   name=Chancey
correct=male     guess=female   name=Chane
correct=male     guess=female   name=Chaunce
correct=male     guess=female   name=Che
correct=male     guess=female   name=Clarance
correct=male     guess=female   name=Demetri
correct=male     guess=female   name=Durante
correct=male     guess=female   name=Ellis
correct=male     guess=female   name=Elmore
correct=male     guess=female   name=Elvis
correct=male     guess=female   name=Eustace
correct=male     guess=female   name=Fairfax
```

```
correct=male     guess=female   name=Irvine
correct=male     guess=female   name=Jeffie
correct=male     guess=female   name=Keil
correct=male     guess=female   name=Laurance
correct=male     guess=female   name=Lemmie
correct=male     guess=female   name=Lewis
correct=male     guess=female   name=Marlin
correct=male     guess=female   name=Marshal
correct=male     guess=female   name=Marwin
correct=male     guess=female   name=Natale
correct=male     guess=female   name=Nathan
correct=male     guess=female   name=Odie
correct=male     guess=female   name=Prentice
correct=male     guess=female   name=Saxe
correct=male     guess=female   name=Si
correct=male     guess=female   name=Thane
correct=male     guess=female   name=Tye
correct=male     guess=female   name=Uli
correct=male     guess=female   name=Zebadiah
```

In [23]:
```
print("Error count: ", len(errors))
```

```
Error count:  85
```

In [24]:
```
## GH modified --- from shovan

train_set = [(gender_features1(n), g) for (n,g) in train_set]
devtest_set = [(gender_features1(n), g) for (n,g) in devtest_set]
test_set = [(gender_features1(n), g) for (n,g) in test_set]

nb1 = nltk.NaiveBayesClassifier.train(train_set)

print('Devset accuracy is')
print(nltk.classify.accuracy(nb1, devtest_set))
print('Test accuracy is')
print(nltk.classify.accuracy(nb1, test_set))
```

```
Devset accuracy is
0.83
Test accuracy is
0.824
```

In [25]:
```
# from shovan
performance_metrics(nb1, train_set)
performance_metrics(nb1, devtest_set)
```

```
Male precision:  0.7639
Female precision:  0.8911
Male recall:  0.808
Female recall:  0.8629
Male precision:  0.7828
Female precision:  0.8609
Male recall:  0.7868
Female recall:  0.8581
```

Classifier - DecisionTree-on 1st Feature

In [26]:
```
classifier_tree = nltk.DecisionTreeClassifier.train(train_set_fe)

print("train_set: ", nltk.classify.accuracy(classifier_tree, train_set_fe))
print("test_set: ", nltk.classify.accuracy(classifier_tree, test_set_fe))
print("devtest_set: ", nltk.classify.accuracy(classifier_tree, devtest_set_fe))
```

```
train_set:  0.9779530093337625
test_set:  0.798
devtest_set:  0.78
```

Features - 2nd Feature

In [46]:
```
## GH to fix the error no attributes .lower()
# extract name's characteristics
def gender_features2nd(name):
    name = name.lower()
    return{
        'first_letter': name[0].lower(),
        'first2_letter': name[0:2].lower(),
        'first3_letter': name[0:3].lower(),
        'last_letter': name[-1].lower(),
        'last2_letter': name[-2:].lower(),
        'last3_letter': name[-3:].lower(),
        'last_vowel': (name[-1] in 'aeiou')
    }
```

In [47]:
```
gender_features2nd("Mary")
```

Out[47]:
```
{'first_letter': 'm',
 'first2_letter': 'ma',
 'first3_letter': 'mar',
 'last_letter': 'y',
 'last2_letter': 'ry',
 'last3_letter': 'ary',
 'last_vowel': False}
```

In [ ]:

In [ ]:

Vectorize the Features

In [48]:
```
import numpy as np

# Vectorize the features function
features = np.vectorize(gender_features2nd)
#print(features(['Rose', 'Mike']))
print(features(['Mary', 'Joe']))
```

```
[{'first_letter': 'm', 'first2_letter': 'ma', 'first3_letter': 'mar', 'last_letter':
'y', 'last2_letter': 'ry', 'last3_letter': 'ary', 'last_vowel': False}
 {'first_letter': 'j', 'first2_letter': 'jo', 'first3_letter': 'joe', 'last_letter':
'e', 'last2_letter': 'oe', 'last3_letter': 'joe', 'last_vowel': True}]
```

In [49]:
```
# Extract the features for entire dataset
X = np.array(features(names))[:, 0] # X contains the features

# Get the gender column
y = np.array(names)[:, 1]           # y contains the targets

print("Name: %s, features=%s, gender=%s" % (names[0][0], X[0], y[0]))
```

```
Name: Lebbie, features={'first_letter': 'l', 'first2_letter': 'le', 'first3_letter': 'le
b', 'last_letter': 'e', 'last2_letter': 'ie', 'last3_letter': 'bie', 'last_vowel': Tru
e}, gender=female
```

In [50]:
```
# Shuffle and split:  train, dev-test, test
from sklearn.utils import shuffle
X,y = shuffle(X,y)

X_test, X_dev_test, X_train = X[:500], X[500:1000], X[1000:]
y_test, y_dev_test, y_train = y[:500], y[500:1000], y[1000:]

print("test: " , len(X_test))
print("devtest: ", len(X_dev_test))
print("train: ", len(X_train))
```

```
test:  500
devtest:  500
train:  6214
```

In [51]:
```
# Use vectorizer to transform the features into feature-vectors.
from sklearn.feature_extraction import DictVectorizer

#print(features(['Rose', 'Mike']))
print(features(['Mary', 'Joe']))

# train the vectorizer to know the possible features and values.
vectorizer = DictVectorizer()
vectorizer.fit(X_train)

#transform = vectorizer.transform(features(['Rose', 'Mike']))

transform = vectorizer.transform(features(['Mary', 'Joe']))
print(transform)
print(type(transform))
print(transform.toarray()[0][12])
print(vectorizer.feature_names_[12])
```

```
[{'first_letter': 'm', 'first2_letter': 'ma', 'first3_letter': 'mar', 'last_letter':
'y', 'last2_letter': 'ry', 'last3_letter': 'ary', 'last_vowel': False}
 {'first_letter': 'j', 'first2_letter': 'jo', 'first3_letter': 'joe', 'last_letter':
'e', 'last2_letter': 'oe', 'last3_letter': 'joe', 'last_vowel': True}]
  (0, 127)      1.0
  (0, 964)      1.0
  (0, 1481)     1.0
  (0, 1696)     1.0
  (0, 1836)     1.0
  (0, 3067)     1.0
  (0, 3069)     0.0
  (1, 108)      1.0
  (1, 818)      1.0
  (1, 1478)     1.0
  (1, 1653)     1.0
  (1, 2303)     1.0
  (1, 3048)     1.0
  (1, 3069)     1.0
<class 'scipy.sparse.csr.csr_matrix'>
0.0
first2_letter=ap
```

Classifier - DecisionTree¶

In [52]:
```
from sklearn.tree import DecisionTreeClassifier
# DT classifier to extract discriminating rules from the features.
DT_classifier = DecisionTreeClassifier()

DT_classifier.fit(vectorizer.transform(X_train), y_train)
```

Out[52]:
```
DecisionTreeClassifier()
```

In [53]:
```
print(DT_classifier.predict(vectorizer.transform(features(["Sebastian", "Amy"]))))
```

```
['male' 'female']
```

In [54]:
```
# Accuracy
print("Accuracy on training set: ", DT_classifier.score(vectorizer.transform(X_train),
print("Accuracy on dev-test set: ",DT_classifier.score(vectorizer.transform(X_dev_test)
print("Accuracy on test set: ",DT_classifier.score(vectorizer.transform(X_test), y_test
```

```
Accuracy on training set:  0.998712584486643
Accuracy on dev-test set:  0.862
Accuracy on test set:  0.87
```

Cross Validation

In [55]:
```
# cross validation
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score

pred_train = cross_val_predict(DT_classifier, vectorizer.transform(X_train), y_train, c
pred_dev_test = cross_val_predict(DT_classifier, vectorizer.transform(X_dev_test), y_de
pred_test = cross_val_predict(DT_classifier, vectorizer.transform(X_test), y_test, cv =

score_train = accuracy_score(y_train, pred_train)
score_dev_test = accuracy_score(y_dev_test, pred_dev_test)
score_test = accuracy_score(y_test, pred_test)

print("Cross Validation")
print("Train Score = {0:5f}".format(score_train))
print("Dev Test Score = {0:5f}".format(score_dev_test))
print("Test Score = {0:5f}".format(score_test))
```

```
Cross Validation
Train Score = 0.850982
Dev Test Score = 0.812000
Test Score = 0.778000
```

In [ ]:

In [45]:
```
## Summary
```

In [ ]:

In [ ]: