Assignment#2. Comparing Models

Jae Dong Hwang

Verify that cross validations is selecting the correct data into each fold.

I implemented below function and verified a list of objects can be folded by index.

```
def fold_data(xTrainRaw, k):
    """
    Args:
        xTrainRaw: a list of xTrainRaw data
        k: number of folding
    Returns:
        (a list of training sets, a list of validation sets)
    """
    if not k > 1:
        raise ValueError("Expected {} > 1".format(k))

    # divid xTrainRaw data into k group
    grouped_xTrainRaw = divide_into_group(xTrainRaw, k)
    print("Groupped xTrainRaw into {} groups.".format(k))

    trains = []
    validations = []
    for i in range(k):
        group = list(grouped_xTrainRaw)
        validation = group.pop(i)
        train = list(itertools.chain.from_iterable(group))
        trains.append(train)
        validations.append(validation)

    return trains, validations


def divide_into_group(xTrainRaw, k):

    cnt = len(xTrainRaw) / k
    groups = []
    last = 0.0

    while last < len(xTrainRaw):
        groups.append(xTrainRaw[int(last):int(last + cnt)])
        last += cnt

    if not len(xTrainRaw) == sum([len(g) for g in groups]):
        raise AssertionError("Missing/Duplicated element {} vs {}"
                             .format(len(xTrainRaw),
                                     sum([len(g) for g in groups])))

    if not len(groups) == k:
        raise AssertionError("More or less groups found {} vs (expected){}"
                                 .format(len(groups), k))
    return groups
```

And I tested with below code and verified the outputs:

```
xTrainRaw = ['a', 'b', 'c']
trainings, validations = fold_data(xTrainRaw, 3)
for t, v in zip(trainings, validations):
    print("Training: {} Validation: {}".format(t, v))

# Code above produced output below:

# Groupped xTrainRaw into 3 groups.
# Training: ['b', 'c'] Validation: ['a']
# Training: ['a', 'c'] Validation: ['b']
# Training: ['a', 'b'] Validation: ['c']
```

The accuracy estimates from the train/test split run with error bounds

- Accuracy Estimates w/ Zn=1.96

Feature Selections	Accuracy	Upper	Lower
Top 10 Frequency	0.8550932568149211	0.8657645971081168	0.8444219165217254
Top 10 MI	0.9239598278335724	0.9319953854766465	0.9159242701904984

  (generated by comparing_models.py)

The accuracy estimates from the cross validations runs for the two model variants with error bound

Below table shows the results for cross validation with k=5.

- Accuracy Estimate from Cross Validation w/ Zn=1.96

Feature Selections	TotalCorrect	N	Accuracy	Upper	Lower
Top 10 Frequency	3294	10	0.7880382775119618	0.8004282490023151	0.7756483060216084
Top 10 MI	3881	10	0.9284688995215311	0.9362815620330935	0.9206562370099687

  (generated by k_fold_cross_validation.py)

  I also kept the evaluation results of model.predict(foldValidationX), foldValidationY) at each folding status.

  - **Logs for evaluations on each validation sets by *top 10 frequenct features***: Gradient descent for 0th folding

    - Statistics:

	1	0
1	(TP) 0	(FN) 107
0	(FP) 0	(TN) 729

Accuracy: 0.8720095693779905 Precision: 0.0 Recall: 0.0 FPR: 0.0 FNR: 1.0 Features selected: ['to', 'you', 'I', 'a', 'the', 'and', 'is', 'in', 'i', 'u']

Gradient descent for 1th folding

- Statistics:

	1	0
1	(TP) 0	(FN) 108
0	(FP) 0	(TN) 728

Accuracy: 0.8708133971291866 Precision: 0.0 Recall: 0.0 FPR: 0.0 FNR: 1.0 Features selected: ['to', 'you', 'I', 'a', 'the', 'and', 'in', 'is', 'i', 'u']

Gradient descent for 2th folding

- Statistics:

	1	0
1	(TP) 77	(FN) 25
0	(FP) 241	(TN) 493

Accuracy: 0.6818181818181818 Precision: 0.24213836477987422 Recall: 0.7549019607843137 FPR: 0.32833787465940056 FNR: 0.24509803921568626 Features selected: ['to', 'you', 'I', 'a', 'the', 'and', 'is', 'in', 'i', 'u']

Gradient descent for 3th folding

- Statistics:

	1	0
1	(TP) 78	(FN) 40
0	(FP) 255	(TN) 463

Accuracy: 0.6471291866028708 Precision: 0.23423423423423423 Recall: 0.6610169491525424 FPR: 0.3551532033426184 FNR: 0.3389830508474576 Features selected: ['to', 'you', 'I', 'a', 'the', 'and', 'is', 'in', 'i', 'u']

Gradient descent for 4th folding

- Statistics:

	1	0
1	(TP) 0	(FN) 110
0	(FP) 0	(TN) 726

Accuracy: 0.868421052631579 Precision: 0.0 Recall: 0.0 FPR: 0.0 FNR: 1.0 Features selected: ['to', 'you', 'I', 'a', 'the', 'and', 'is', 'in', 'i', 'u']

- **Logs for evaluations on each validation sets by *top 10 MI features***:

  Gradient descent for 0th folding

  - Statistics:

	1	**0**
1	(TP) 68	(FN) 39
0	(FP) 11	(TN) 718

    Accuracy: 0.9401913875598086 Precision: 0.8607594936708861 Recall: 0.6355140186915887 FPR: 0.015089163237311385 FNR: 0.3644859813084112 Features selected: ['I', 'Call', 'i', 'FREE', '&', 'mobile', 'my', 'claim', 'To', 'Txt']

  Gradient descent for 1th folding

  - Statistics:

	1	**0**
1	(TP) 43	(FN) 65
0	(FP) 3	(TN) 725

    Accuracy: 0.9186602870813397 Precision: 0.9347826086956522 Recall: 0.39814814814814814 FPR: 0.004120879120879121 FNR: 0.6018518518518519 Features selected: ['I', 'Call', 'i', 'claim', 'FREE', 'To', '&', 'mobile', 'my', 'call']

  Gradient descent for 2th folding

  - Statistics:

	1	**0**
1	(TP) 53	(FN) 49
0	(FP) 8	(TN) 726

    Accuracy: 0.9318181818181818 Precision: 0.8688524590163934 Recall: 0.5196078431372549 FPR: 0.010899182561307902 FNR: 0.4803921568627451 Features selected: ['I', 'Call', 'i', 'FREE', 'claim', 'mobile', '&', 'my', 'To', 'Txt']

  Gradient descent for 3th folding

  - Statistics:

	1	**0**
1	(TP) 56	(FN) 62
0	(FP) 7	(TN) 711

    Accuracy: 0.9174641148325359 Precision: 0.8888888888888888 Recall: 0.4745762711864407 FPR: 0.009749303621169917 FNR: 0.5254237288135594 Features selected: ['I', 'Call', 'i', 'FREE', 'claim', '&', 'Txt', 'To', 'my', 'mobile']

  Gradient descent for 4th folding

  - Statistics:

	1	**0**
1	(TP) 60	(FN) 50
0	(FP) 5	(TN) 721

Accuracy: 0.9342105263157895 Precision: 0.9230769230769231 Recall: 0.5454545454545454 FPR: 0.006887052341597796 FNR: 0.45454545454545453 Features selected: ['I', 'i', 'Call', 'FREE', 'claim', 'Txt', 'my', '&', 'To', 'mobile']

(generated by k_fold_cross_validation.py)
