LogReg

February 23, 2021

0.1 Imports and data ingestion

`linear_model` for Logistic Regression,

`model_selection` for train-test split and cross-validation,

`metrics` for ROC and AIC

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.linear_model as skl_lm
import sklearn.ensemble as skl_en
import sklearn.model_selection as skl_ms
import sklearn.metrics as skl_met
import sklearn.preprocessing as skl_pre
```

[2]:
```
rawData = pd.read_csv('train.csv')
```

0.2 Get Dummies

`Lead` is `0` if the lead is `female`.

[3]:
```
data = rawData.copy()
data['Lead'] = pd.get_dummies(data['Lead'])
```

0.3 Feature Engineering

We first extract the number of words spoken by the co-lead.

[4]:
```
data['Number of words co-lead'] = data['Number of words lead'] -␣
↪data['Difference in words lead and co-lead']
```

It is probably more important to know proportion of words spoken by lead than the absolute number of words.

[5]:
```
data['Proportion of words lead'] = data['Number of words lead']/data['Total␣
↪words']
```

We look at the proportion of dialogue from co-lead as well as the ratio of dialogue between co-lead and lead.

[6]:
```
data['Proportion of words co-lead'] = data['Number of words co-lead']/
    ↪data['Total words']
data['Ratio words co-lead lead'] = data['Number of words co-lead']/data['Number␣
    ↪of words lead']
```

Similarly for number of words spoken by male or female actors. However, here we normalize by `Total words - Number of words lead` since this only measures non-lead dialogue. We only need `Proportion of words female` since a hypothetical `Proportion of words male` is just `1 - Proportion of words female`.

[7]:
```
data['Proportion of words female'] = data['Number words female']/(data['Total␣
    ↪words'] - data['Number of words lead'])
```

It might be important to look at the ratio of female actors.

[8]:
```
data['Number of actors'] = data['Number of male actors'] + data['Number of␣
    ↪female actors']
```

[9]:
```
data[['Total words','Number of words lead','Difference in words lead and␣
    ↪co-lead','Number of words co-lead','Number words male','Number words female']]
```

[9]:
```
      Total words  Number of words lead  Difference in words lead and co-lead  \
0            6394                2251.0                                   343
1            8780                2020.0                                  1219
2            4176                 942.0                                   787
3            9855                3440.0                                  2623
4            7688                3835.0                                  3149
...           ...                   ...                                   ...
1034         2398                1334.0                                  1166
1035         8404                1952.0                                   187
1036         2750                 877.0                                   356
1037         3994                 775.0                                    52
1038        11946                3410.0                                  1536

      Number of words co-lead  Number words male  Number words female
0                      1908.0               2631                 1512
1                       801.0               5236                 1524
2                       155.0               3079                  155
3                       817.0               5342                 1073
4                       686.0               2536                 1317
...                       ...                ...                  ...
1034                    168.0                761                  303
1035                   1765.0               5820                  632
1036                    521.0                547                 1326
1037                    723.0               2757                  462
1038                   1874.0               5801                 2735

[1039 rows x 6 columns]
```

```
[10]: data['Proportion of female actors'] = data['Number of female actors']/
       ↪data['Number of actors']
```

Perhaps it is important if the lead or co-lead is oldest.

```
[11]: data['Older lead'] = data['Age Lead'] < data['Age Co-Lead']
      data['Older lead'] = pd.get_dummies(data['Older lead'])
```

```
[12]: for col in data.columns:
          print("'" + col + "',")
```

```
'Number words female',
'Total words',
'Number of words lead',
'Difference in words lead and co-lead',
'Number of male actors',
'Year',
'Number of female actors',
'Number words male',
'Gross',
'Mean Age Male',
'Mean Age Female',
'Age Lead',
'Age Co-Lead',
'Lead',
'Number of words co-lead',
'Proportion of words lead',
'Proportion of words co-lead',
'Ratio words co-lead lead',
'Proportion of words female',
'Number of actors',
'Proportion of female actors',
'Older lead',
```

We center and scale all computed columns as well as some of the original columns

```
[13]: cols_to_norm = [
          'Total words',
          'Year',
          'Gross',
          'Mean Age Male',
          'Mean Age Female',
          'Age Lead',
          'Age Co-Lead',
          'Number of actors'#,
          #'Proportion of words lead',
          #'Proportion of words co-lead',
          #'Ratio words co-lead lead',
          #'Proportion of words female',
```

```
    #'Proportion of female actors',
]
```

[14]:
```
scaler = skl_pre.StandardScaler()
data[cols_to_norm] = scaler.fit_transform(data[cols_to_norm])
```

[15]:
```
data
```

[15]:
```
      Number words female  Total words  Number of words lead  \
0                    1512    -0.676591                2251.0   
1                    1524    -0.326435                2020.0   
2                     155    -1.002092                 942.0   
3                    1073    -0.168675                3440.0   
4                    1317    -0.486691                3835.0   
...                   ...          ...                   ...   
1034                  303    -1.263021                1334.0   
1035                  632    -0.381615                1952.0   
1036                 1326    -1.211363                 877.0   
1037                  462    -1.028801                 775.0   
1038                 2735     0.138188                3410.0   

      Difference in words lead and co-lead  Number of male actors      Year  \
0                                      343                      2 -0.467462   
1                                     1219                      9  0.109371   
2                                      787                      7 -3.063211   
3                                     2623                     12  0.205509   
4                                     3149                      8 -1.140434   
...                                    ...                    ...       ...   
1034                                  1166                      5 -2.582517   
1035                                   187                      6 -0.755879   
1036                                   356                      2  0.013232   
1037                                    52                      8 -0.371324   
1038                                  1536                     13  0.686204   

      Number of female actors  Number words male     Gross  Mean Age Male  \
0                           5               2631  0.203383       1.170591   
1                           4               5236 -0.488825      -0.413237   
2                           1               3079  1.746018       0.018716   
3                           2               5342 -0.607490      -0.912739   
4                           4               2536 -0.469048       0.370678   
...                       ...                ...       ...            ...   
1034                        2                761  0.414341       0.108306   
1035                        2               5820  0.401156      -0.663877   
1036                        3                547 -0.383346      -1.901076   
1037                        3               2757 -0.521788       0.064425   
1038                        4               5801 -0.521788       0.222330   
```

```
      ...  Age Co-Lead  Lead  Number of words co-lead  \
0     ...     2.451143     1                   1908.0
1     ...    -0.123416     0                    801.0
2     ...     0.125734     0                    155.0
3     ...    -1.036970     0                    817.0
4     ...     0.291835     0                    686.0
...   ...          ...   ...                      ...
1034  ...    -0.953919     0                    168.0
1035  ...    -0.123416     1                   1765.0
1036  ...    -0.870869     0                    521.0
1037  ...    -0.289517     1                    723.0
1038  ...     1.039288     0                   1874.0

      Proportion of words lead  Proportion of words co-lead  \
0                     0.352049                     0.298405
1                     0.230068                     0.091230
2                     0.225575                     0.037117
3                     0.349061                     0.082902
4                     0.498829                     0.089230
...                        ...                          ...
1034                  0.556297                     0.070058
1035                  0.232270                     0.210019
1036                  0.318909                     0.189455
1037                  0.194041                     0.181022
1038                  0.285451                     0.156873

      Ratio words co-lead lead  Proportion of words female  Number of actors  \
0                     0.847623                    0.364953         -0.940919
1                     0.396535                    0.225444          0.379885
2                     0.164544                    0.047928         -0.720785
3                     0.237500                    0.167264          0.600019
4                     0.178879                    0.341812          0.159751
...                        ...                         ...               ...
1034                  0.125937                    0.284774         -0.940919
1035                  0.904201                    0.097954         -0.720785
1036                  0.594071                    0.707955         -1.381187
1037                  0.932903                    0.143523         -0.060383
1038                  0.549560                    0.320408          1.260421

      Proportion of female actors  Older lead
0                        0.714286           0
1                        0.307692           1
2                        0.125000           1
3                        0.142857           1
4                        0.333333           0
...                           ...         ...
1034                     0.285714           1
```

```
1035                    0.250000            0
1036                    0.600000            1
1037                    0.272727            0
1038                    0.235294            0

[1039 rows x 22 columns]
```

Combining all pre-processing into a function

```
[16]: def pre_process(raw_data, cols_to_norm):
    data = raw_data.copy()

    data['Lead'] = pd.get_dummies(data['Lead'])
    data['Number of words co-lead'] = data['Number of words lead'] -␣
 ↪data['Difference in words lead and co-lead']
    data['Proportion of words lead'] = data['Number of words lead']/data['Total␣
 ↪words']
    data['Proportion of words co-lead'] = data['Number of words co-lead']/
 ↪data['Total words']
    data['Ratio words co-lead lead'] = data['Number of words co-lead']/
 ↪data['Number of words lead']
    data['Proportion of words female'] = data['Number words female']/
 ↪(data['Total words'] - data['Number of words lead'])
    data['Number of actors'] = data['Number of male actors'] + data['Number of␣
 ↪female actors']
    data['Proportion of female actors'] = data['Number of female actors']/
 ↪data['Number of actors']
    data['Older lead'] = data['Age Lead'] < data['Age Co-Lead']
    data['Older lead'] = pd.get_dummies(data['Older lead'])

    scaler = skl_pre.StandardScaler()
    data[cols_to_norm] = scaler.fit_transform(data[cols_to_norm])

    return data
```

1 Logistic Regression

```
[17]: data = pre_process(rawData, cols_to_norm)
```

```
[205]: config = pd.read_csv('config.csv')
trainRatio = config['Train Ratio'][0]
seed = config['Random Seed'][0]
train, test = skl_ms.train_test_split(data, train_size=trainRatio) #,␣
 ↪random_state=seed)
```

[206]:
```python
rawFeatures = [
    'Number words female',
    'Total words',
    'Number of words lead',
    'Difference in words lead and co-lead',
    'Number of male actors',
    'Year',
    'Number of female actors',
    'Number words male',
    'Gross',
    'Mean Age Male',
    'Mean Age Female',
    'Age Lead',
    'Age Co-Lead'
]

featureSet1 = [
    'Year',
    'Gross',
    'Number of actors',
    'Proportion of female actors',
    'Mean Age Male',
    'Mean Age Female',
    'Age Lead',
    'Age Co-Lead',
    'Total words',
    'Proportion of words lead',
    'Proportion of words co-lead',
    'Ratio words co-lead lead',
    'Proportion of words female',
    'Older lead'
]
```

[207]:
```python
set(featureSet1) - set(cols_to_norm)
```

[207]:
```
{'Older lead',
 'Proportion of female actors',
 'Proportion of words co-lead',
 'Proportion of words female',
 'Proportion of words lead',
 'Ratio words co-lead lead'}
```

[208]:
```python
features = featureSet1.copy()
#features.remove('Proportion of words female')
#features.remove('Year')
#features.remove('Gross')
#features = ['Proportion of words lead']
```

```
target = 'Lead'
```

[209]:
```
def fit_and_test(classifier, train, test, features, target, suppress_output =␣
↪False):
    classifier.fit(train[features], train[target])
    if not suppress_output:
        skl_met.plot_roc_curve(classifier, test[features], test[target])
        print('accuracy: ' + str(classifier.score(test[features], test[target])))
        print('     auc: ' + str(skl_met.roc_auc_score(test[target], classifier.
↪predict_proba(test[features])[:,1])) + '\n')
        print(skl_met.classification_report(test[target], classifier.
↪predict(test[features])))
    return classifier
```

[210]:
```
print('Null accuracy: ' + str(max([np.mean(data[target]), 1 - np.
↪mean(data[target])])))
```

```
Null accuracy: 0.7555341674687199
```

1.1 No regularization

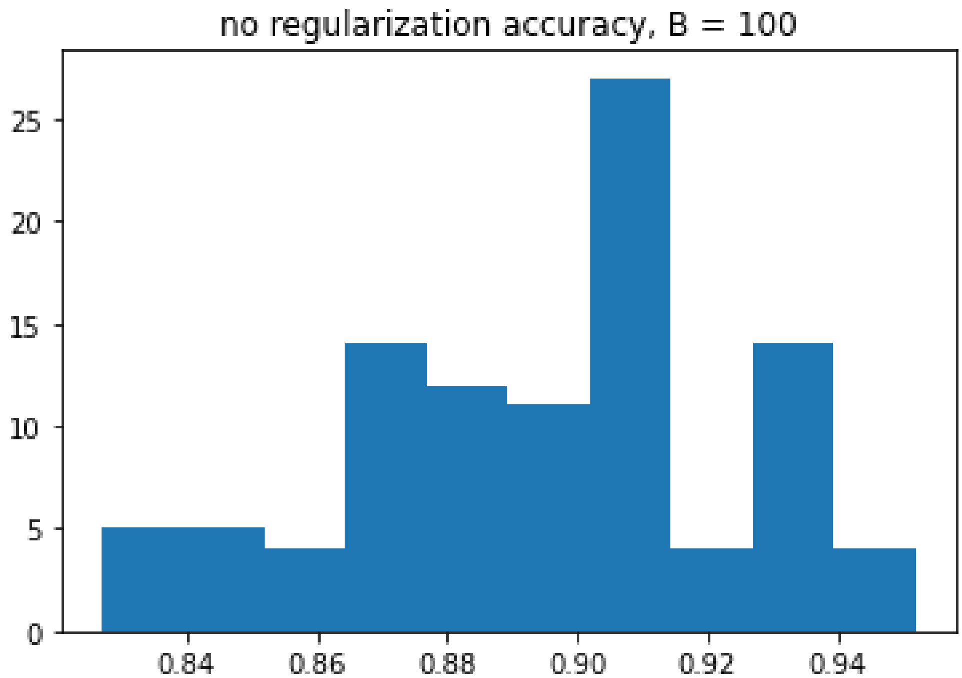
[211]:
```
%%time
B = 100
accuracies = []
aucs = []
for i in range(B):
    train, test = skl_ms.train_test_split(data, train_size=trainRatio)
    logReg = fit_and_test(skl_lm.LogisticRegression(penalty='none',␣
↪solver='newton-cg'), train, test, features, target, suppress_output=True)
    accuracies.append(logReg.score(test[features], test[target]))
    aucs.append(skl_met.roc_auc_score(test[target], logReg.
↪predict_proba(test[features])[:,1]))
```

```
CPU times: user 8.2 s, sys: 21.2 s, total: 29.4 s
Wall time: 2.6 s
```

[212]:
```
print('mean accuracy: ' + str(np.mean(accuracies)))
print('     mean auc: ' + str(np.mean(aucs)))
```

```
mean accuracy: 0.8953846153846154
     mean auc: 0.9244967487778538
```

[213]:
```
plt.hist(accuracies)
plt.title(f'no regularization accuracy, B = {B}')
plt.show()
```



1.2 Lasso

Lasso is L1 regularization

We do 20 folds, use `liblinear` solver that is better for smaller datasets and 10 threads in the cross-validation step.

The method will try 10 values of lambda between `1e-4` and `1e4` to determine the best regularization parameter.

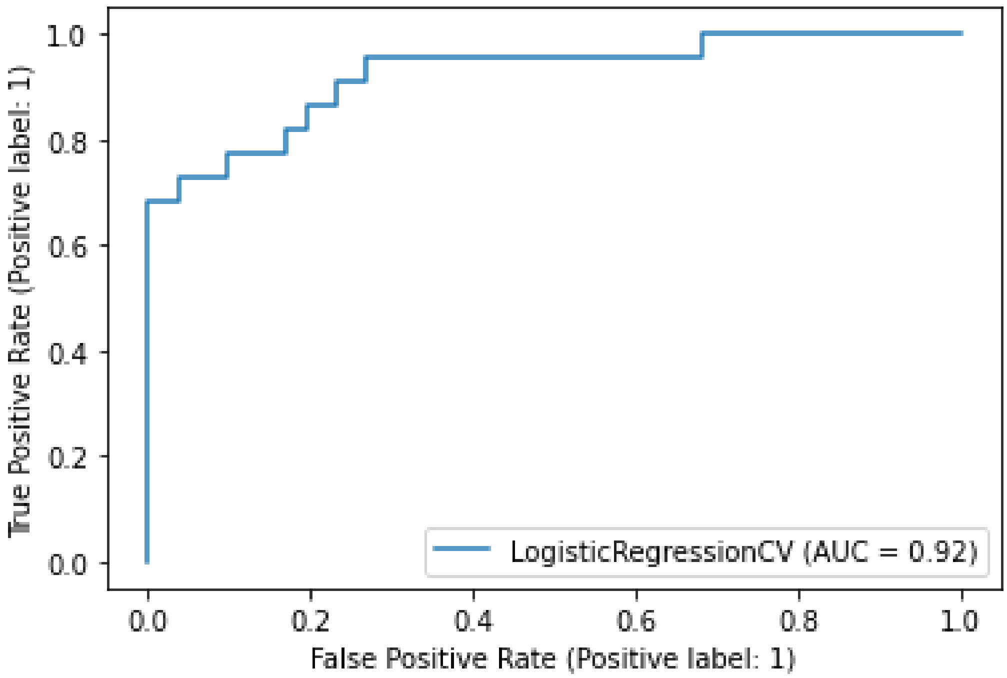
[214]:
```
logRegLasso = fit_and_test(skl_lm.LogisticRegressionCV(Cs=10, cv=50,␣
↪penalty='l1', solver='liblinear', n_jobs=10), train, test, features, target)
```

```
accuracy: 0.9038461538461539
     auc: 0.9235033259423504

              precision    recall  f1-score   support

           0       0.92      0.96      0.94        82
           1       0.83      0.68      0.75        22

    accuracy                           0.90       104
   macro avg       0.88      0.82      0.85       104
weighted avg       0.90      0.90      0.90       104
```



We want to know how the algorithm performs over different splits of training and testing data

[215]:
```
%%time
B = 100
accuracies = []
aucs = []
for i in range(B):
    train, test = skl_ms.train_test_split(data, train_size=trainRatio)
    logRegLasso = fit_and_test(skl_lm.LogisticRegressionCV(Cs=10, cv=10,␣
 ↪penalty='l1', solver='liblinear', n_jobs=10), train, test, features, target,␣
 ↪suppress_output=True)
    accuracies.append(logRegLasso.score(test[features], test[target]))
    aucs.append(skl_met.roc_auc_score(test[target], logRegLasso.
 ↪predict_proba(test[features])[:,1]))
```

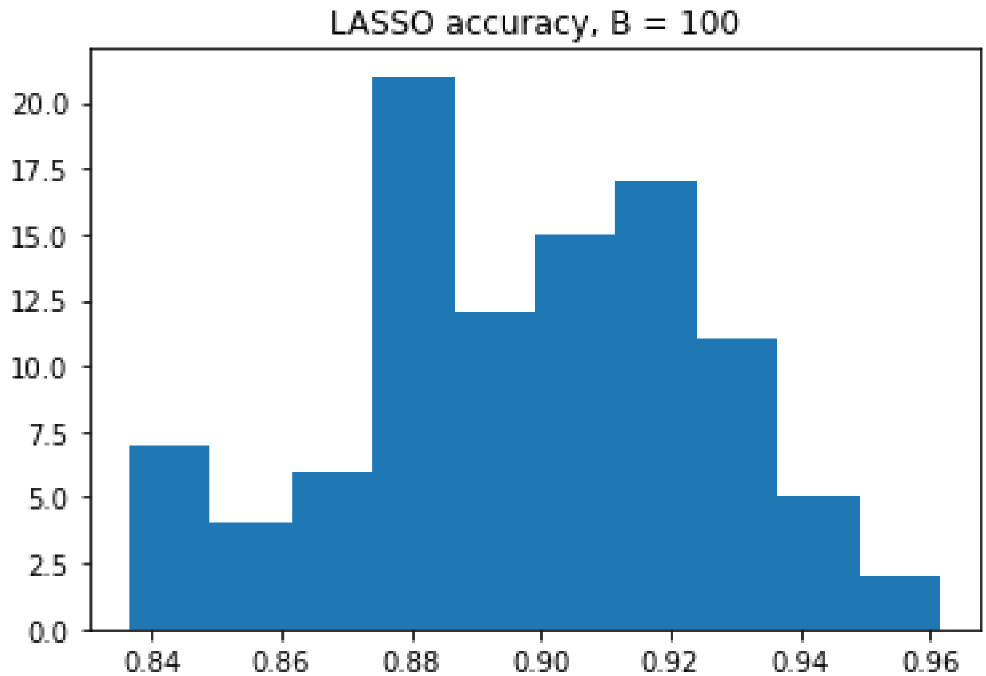
```
CPU times: user 4.31 s, sys: 46.4 ms, total: 4.36 s
Wall time: 23.2 s
```

[216]:
```
print('mean accuracy: ' + str(np.mean(accuracies)))
print('     mean auc: ' + str(np.mean(aucs)))
```

```
mean accuracy: 0.8972115384615386
     mean auc: 0.9244666002902575
```

[217]:
```
plt.hist(accuracies)
plt.title(f'LASSO accuracy, B = {B}')
```

```
plt.show()
```



1.3 Ridge

Ridge is L2-regularization

We do 10 folds, use `liblinear` solver that is better for smaller datasets and 10 threads in the cross-validation step.

The method will try 100 values of lambda between `1e-4` and `1e4` to determine the best regularization parameter.

[218]:
```
logRegRidge = fit_and_test(skl_lm.LogisticRegressionCV(Cs=10, cv=20,␣
↪penalty='l2', solver='liblinear', n_jobs=10), train, test, features, target)
```

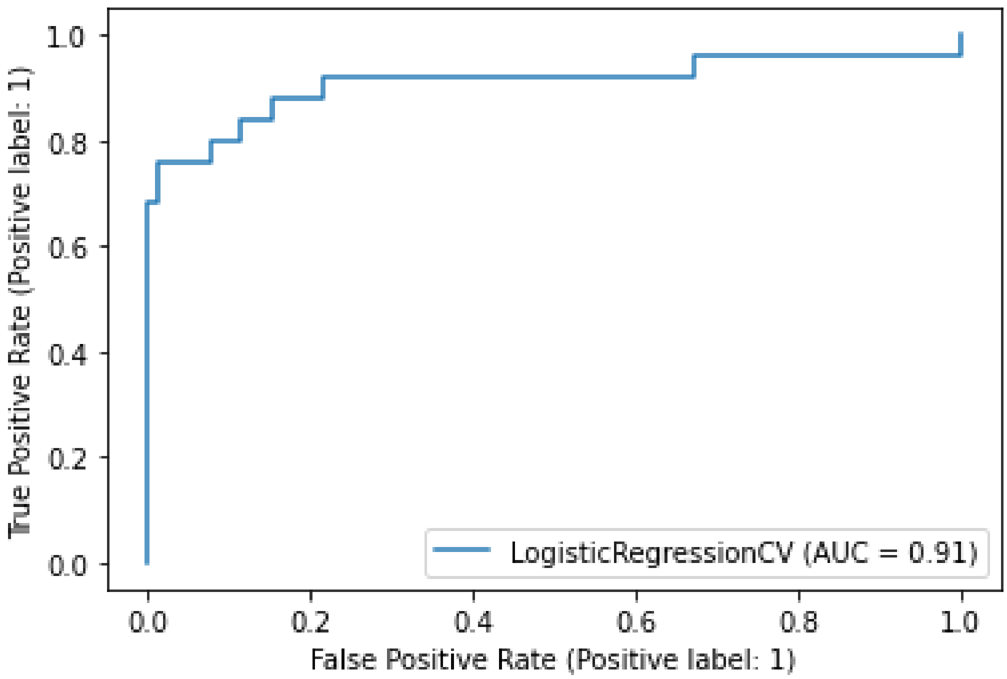
```
accuracy: 0.9038461538461539
     auc: 0.9098734177215191

              precision    recall  f1-score   support

           0       0.93      0.95      0.94        79
           1       0.83      0.76      0.79        25

    accuracy                           0.90       104
   macro avg       0.88      0.85      0.86       104
```

```
weighted avg       0.90      0.90      0.90       104
```



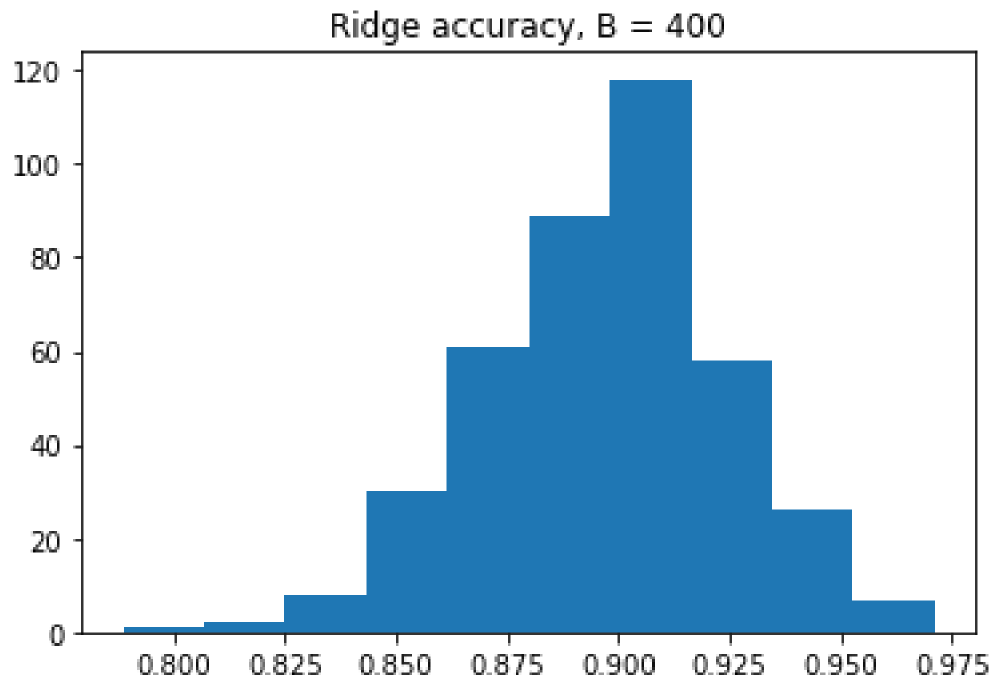
[219]:
```
%%time
B = 400
accuracies = []
aucs = []
for i in range(B):
    train, test = skl_ms.train_test_split(data, train_size=trainRatio)
    logRegLasso = fit_and_test(skl_lm.LogisticRegressionCV(Cs=10, cv=10,␣
 ↪penalty='l2', solver='liblinear', n_jobs=10), train, test, features, target,␣
 ↪suppress_output=True)
    accuracies.append(logRegLasso.score(test[features], test[target]))
    aucs.append(skl_met.roc_auc_score(test[target], logRegLasso.
 ↪predict_proba(test[features])[:,1]))
```

```
CPU times: user 5.96 s, sys: 233 ms, total: 6.19 s
Wall time: 20 s
```

[220]:
```
print("mean accuracy: " + str(np.mean(accuracies)))
print('     mean auc: ' + str(np.mean(aucs)))
```

```
mean accuracy: 0.8977884615384616
     mean auc: 0.9231520433528324
```

[221]:
```
plt.hist(accuracies)
plt.title(f'Ridge accuracy, B = {B}')
plt.show()
```



1.4 Comparison

[222]:
```
logRegLasso.C_
```

[222]:
```
array([10000.])
```

[223]:
```
logRegRidge.C_
```

[223]:
```
array([21.5443469])
```

[224]:
```
predLasso = logRegLasso.predict(test[features])
predRidge = logRegRidge.predict(test[features])
```

[225]:
```
logRegLassoCoefs = dict(zip(features, logRegLasso.coef_[0]))
```

There is a reason that `Proportion of words female` is an indicator that the lead is male. If the Lead is male, `Number words female` includes the number of words spoken by the co-lead and hence `Number words female` is the total number of words spoken by **all** female actors while `Number words male` is the number of words spoken by the **non-lead** male actors. Hence,

[226]:
```
for var, coef in logRegLassoCoefs.items():
    print(f'{var:40} {coef:.5f}')
```

```
Year                                     -0.07222
Gross                                    -0.02181
Number of actors                         0.56105
Proportion of female actors              18.62474
Mean Age Male                            0.19473
Mean Age Female                          0.05373
Age Lead                                 -0.30258
Age Co-Lead                              0.21755
Total words                              -0.58851
Proportion of words lead                 2.64110
Proportion of words co-lead              3.81626
Ratio words co-lead lead                 3.30966
Proportion of words female               -11.55700
Older lead                               -1.21839
```

[227]:
```
logRegRidgeCoefs = dict(zip(features, logRegRidge.coef_[0]))
```

[228]:
```
for var, coef in logRegRidgeCoefs.items():
    print(f'{var:40} {coef:.5f}')
```

```
Year                                     -0.05445
Gross                                    -0.04454
Number of actors                         0.41936
Proportion of female actors              16.99463
Mean Age Male                            0.15562
Mean Age Female                          0.09000
Age Lead                                 -0.32796
Age Co-Lead                              0.19087
Total words                              -0.48615
Proportion of words lead                 2.14504
Proportion of words co-lead              4.16098
Ratio words co-lead lead                 2.88327
Proportion of words female               -10.79023
Older lead                               -1.23370
```

[229]:
```
sum(predLasso == predRidge)/len(test)
```

[229]: 0.9615384615384616

[230]:
```
pd.crosstab(test[target], predLasso)
```

[230]:

col_0	0	1
Lead		
0	69	4
1	5	26

[231]:

```python
pd.crosstab(test[target], predRidge)
```

[231]:

col_0	0	1
Lead		
0	71	2
1	5	26

[ ]:

```python

```