

---

# Do (wo)men talk too much in films?

---

Anonymous Author(s)

Affiliation

Address

email

## 1 Introduction

In a movie, there is almost always a main character that the rest of the film is in some way centered around. In some films the main character is not very nice, but most of the time, the audience is expected to sympathize with this character or even look up to, and be inspired by them. This is especially true in children's movies. Since it is often easier to be inspired by someone you can relate to, it is important to make sure that the distribution of main characters in movies at least approximates the distribution of the audience. If the main character is always a man, that makes it more difficult than necessary for girls and women to find someone to be inspired by and vice versa.

This paper investigates how the gender of the lead actor (who plays the main character), can be predicted using metrics such as the year when the movie was made and the age of the lead actor. Being able to make such predictions could give an insight into how the movie industry works with respect to the gender of the lead actor. This could provide clues about what areas to investigate further to understand why those connections exist and ultimately, make sure that everyone has a chance to be inspired by someone they can relate to.

## 2 Methods

We have chosen to focus on approaches using logistic regression, k-NN, LDA and QDA to classify the lead actor's gender.

In order to make the methods as comparable as possible, we have used a common set of transformations of the input variables for all tested methods.

To compare between families of models and between which tuning is better we chose to focus on two measures: accuracy (on average, of how often model makes a correct prediction) and AUC (area under ROC curve).

Several methods have a hyper-parameter  $\lambda$  that needs to be tuned. In order to find a value of  $\lambda$  that performs well on the data, cross-validation is used to find the optimal value in a finite set  $\Lambda = \{\lambda_1, \dots, \lambda_k\}$ . Cross-validation works by splitting the data into  $n$  equally sized partitions and training the data separately on the  $n$  choices of  $n - 1$  partitions and testing on the partition that was left out. The test error  $E_{new}$  is estimated by the mean misclassification rate across the partitions. This procedure is repeated for each  $\lambda_j \in \Lambda$  and the value resulting in the lowest estimated test error is chosen.

Since cross-validation is used to estimate the hyper-parameter  $\lambda$ , this method cannot be used to estimate the test error of the whole procedure. Instead, the dataset has to be split into a training set and a testing set. Cross-validation done on the training set and the test error is

34 estimated by evaluating the performance of the model on the testing set. However, this can  
 35 yield significantly different estimates of the test error since only one split into training and  
 36 testing data is considered. To get a better estimate of the actual testing error, a bootstrap  
 37 procedure is performed.

38 Since the full dataset is an iid sample from some unknown distribution, the estimated  
 39 test error  $\hat{E}_{new}$  is a random variable. By repeating the whole procedure  $B$  times (i.e.  $B$   
 40 independent splits into training and testing data with subsequent fitting and cross-validation),  
 41 a bootstrap sample of  $\hat{E}_{new}$  is obtained which can be used to obtain a better estimate of  
 42  $E_{new}$ . This is very computationally intensive if  $B$  is large.

## 43 2.1 Input transformations

44 In the given dataset, there are columns for the total number of words spoken as well as the  
 45 number of words spoken by the lead, the co-lead etc. This could present a problem since  
 46 if we compare a movie where the lead says 10 out of 100 total words and another movie  
 47 where the lead says 100 out of 1000 words, most models would think that the lead speaks  
 48 more in the second movie and miss the fact that the *proportion* of words spoken by the  
 49 lead is the same. For that reason we have transformed several input variables to express a  
 50 proportion instead of absolute numbers. We also believe it might be important to have a  
 51 dummy variable indicating if the lead or the co-lead is oldest. All transformations are given  
 52 in Table 1.

Original column	New column	Transformation
Number of words lead	Proportion of words lead	$\frac{\text{Number of words lead}}{\text{Total words}}$
N/A	Proportion of words co-lead	$\frac{\text{Number of words lead} - \text{Difference in words lead and co-lead}}{\text{Total words}}$
Difference in words lead and co-lead	Ratio words co-lead lead	$\frac{\text{Proportion of words co-lead}}{\text{Proportion of words lead}}$
Number words female	Proportion of words female	$\frac{\text{Number words female}}{\text{Total words} - \text{Number of words lead}}$
Number of female actors	Proportion of female actors	$\frac{\text{Number of female actors}}{\text{Number of female actors} + \text{Number of male actors}}$
Number of male actors	Number of actors	$\frac{\text{Number of male actors} + \text{Number of female actors}}{\text{Number of actors}}$
N/A	Older lead	$\begin{cases} 1, \text{Age lead} > \text{Age Co-Lead} \\ 0, \text{else} \end{cases}$

Table 1: Transformations of input variables.

53 Note that when determining 'Proportion of words female', this should only measure the  
 54 words spoken by non-lead female actors so we have to subtract the lead's contribution to the  
 55 total number of words.

56 The column 'Number of male actors' was dropped since all necessary information in this  
 57 column is contained in 'Proportion of female actors' together with 'Number of actors'.

58 In order to improve regularization and k-NN, all remaining numerical input variables were  
 59 centered and scaled by their standard deviation. This means that columns with proportions  
 60 have values in the unit interval  $[0, 1]$  and the other numerical variables have values that are  
 61 of roughly the same magnitude. This scaling was not done for QDA as it is not necessary for  
 62 that method.

## 63 2.2 Logistic Regression

64 Logistic regression is a *general linear model* (GLM), i.e. the relationship between the data  
65  $X \in \mathcal{X} \subseteq \mathbb{R}^p$  and the outcome  $Y$  is on the form

$$E(Y|X = x) = g^{-1}(x \cdot \beta) \quad (1)$$

66 where  $\beta \in \mathbb{R}^p$  and  $g$  is the link function. In the case of logistic regression,  $Y|(X =$   
67  $x) \sim \text{Ber}(p(x))$  and the canonical link function is the logit link  $g(x) = \log\left(\frac{x}{1-x}\right)$  with  
68  $g^{-1}(x) = \frac{\exp(x)}{1+\exp(x)}$ . Since  $Y|(X = x) \sim \text{Ber}(p(x))$ , we get  $E(Y|X = x) = p(x) = g^{-1}(x \cdot \beta)$ .  
69 In other words,  $P(Y = 1|X = x) = g^{-1}(x \cdot \beta)$ , which we can use to predict  $Y$  given data  $x$ .

70 To do the regression, we find  $\hat{\beta} \in \arg \min_{\beta} \sum_{i=1}^n (y_i - \hat{y}(x_i; \beta))^2$  where  $\hat{y}(x; \beta) = g^{-1}(x \cdot \beta)$ .  
71 This is the MLE estimator of  $\beta$ . This minimizes the mean squared error (MSE) loss function.  
72 A potential problem with this approach is that there are no restrictions on the components  
73 of  $\beta$  and that can lead to overfitting, especially if  $n$  is not much larger than  $p$ . To address  
74 that issue, one can introduce regularization.

75 In general, regularization is done by adding a penalizing term to the loss function that restricts  
76  $\beta$  in some way. If  $L(\beta; x_i, y_i)$  is the loss function before regularization, we instead consider  
77 the new loss function  $L(\beta; x_i, y_i) + \lambda R(\beta)$  and find  $\hat{\beta}_{reg} \in \arg \min_{\beta} (L(\beta; x_i, y_i) + \lambda R(\beta))$ .  
78  $R$  is some penalizing function and  $\lambda$  is a hyper-parameter that can be tuned. The two most  
79 common forms of regularization is LASSO and Ridge regression.

80 LASSO regression uses  $L_1$ -regularization, meaning that  $R_{LASSO}(\beta) = \|\beta\|_1 = \sum_{i=1}^p |\beta_i|$   
81 while Ridge regression uses  $L_2$ -regularization,  $R_{Ridge}(\beta) = \|\beta\|_2^2 = \sum_{i=1}^p \beta_i^2$ .

## 82 2.3 k-Nearest Neighbors

83 The  $k$ -nearest neighbors ( $k$ -NN) method is based on the simple principle of finding the  $k$   
84 closest neighboring points with respect to the input data  $X \in \mathcal{X} \subseteq \mathbb{R}^p$ . In the case of  
85 classification the outcome  $Y$  is then determined by a majority vote among the  $k$  nearest data  
86 points. The method is based on the idea that if a test data point is close to some training  
87 data point then the prediction should be that they have the same outcome  $Y$ .

88 The algorithm for  $k$ -NN can be implemented in a simple manner with a brute force algorithm  
89 for measuring the distance from the test data point  $\mathbf{x}_{\star}$  to each training data point  $\mathbf{x}_i$ , where  
90  $i = 1, \dots, n$  using some distance function  $d(x, y)$ . It is standard to use the Minkowski distance  
91 for a certain order  $p$ , depending on the problem, which is given by

$$d(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}, \text{ where } \mathbf{x} = (x_1, \dots, x_p), \mathbf{y} = (y_1, \dots, y_p) \in \mathbb{R}^p. \quad (2)$$

92 Note that  $p = 1$  gives the Manhattan distance, and  $p = 2$  gives the Euclidean distance. The  
93 brute force algorithm for  $k$ -NN is given by [1]

94

- 
- 95 1. Calculate the distance  $d(\mathbf{x}_i, \mathbf{x}_{\star})$  for each  $i = 1, \dots, n$
  - 96 2. Set  $\mathcal{N}_{\star} = \{\mathbf{x}_i : \text{Where } \mathbf{x}_i \text{ is one of the } k \text{ nearest points}\}$
  - 97 3. Return  $\hat{y}(\mathbf{x}_{\star}) = \text{MajorityVote}\{y_j : j \in \mathcal{N}_{\star}\}$
- 

98

99 A problem with the brute force algorithm is that for each point we need to calculate the  
100 distance to every other point, which is computationally demanding for larger datasets.

101 There are however more computationally efficient algorithms to find the  $k$ -NN compared to  
102 the brute force search such as ball-tree and  $k$ -d tree which are not explained in detail here.

103 All three of these algorithms were tested and no significant difference in the results were  
 104 noted thus the choice of algorithm was set to "auto" which chooses the best suited algorithm  
 105 for a given problem, further described in the Scikit-learn documentation. [2]

106 For our problem we consider the Minkowski distance and let  $p$  and  $k$  be hyper-parameters  
 107 which are to be tuned. This is done in an analogous manner to the case of finding the hyper-  
 108 parameter  $\lambda$  in the regularization problem with logistic regression previously considered.

109 An alternative approach is to use weighted  $k$ -NN. The idea is to let the distance of the  
 110 training data point to the test data point influence the strength of the vote. We compared  
 111 *uniform weights* (standard  $k$ -NN, where all weights equal 1) and *distance weights* where the  
 112 weights are given by

$$\frac{1}{d(\mathbf{x}_*, \mathbf{x}_i)}, \quad (3)$$

113 for each of the  $k$ -nearest neighbors. This reinforces the idea that proximity of test data  
 114 points to training data points ought to be a good predictor. [3]

## 115 2.4 LDA and QDA

116 For classification we construct a discriminative classifier from a generative model based on  
 117 Bayes' theorem for the classes  $m = 1, 2, \dots, M$

$$p(y = m | \mathbf{x}) = \frac{p(\mathbf{x} | y = m)p(y = m)}{\sum_{i=1}^M p(\mathbf{x} | y = i)p(y = i)}. \quad (4)$$

118 We estimate the *uninformative prior probability* as  $\hat{p}(y = m) = \frac{n_m}{n}$  where  $n_m = \sum_{i=1}^n \mathbb{1}\{y_i =$   
 119  $m\}$  and assume that  $p(\mathbf{x} | y = m)$  is a normal density with expected value  $\mu_m$  and covariance  
 120 matrix  $\Sigma_m$ . The assumption that distinguishes LDA and QDA is that for LDA we assume  
 121 that  $\Sigma_1 = \Sigma_2 = \dots = \Sigma_M$  but for QDA we make no such assumption, that is, we allow for  
 122 the covariance matrices to differ. A consequence is that LDA is a special case of QDA, hence  
 123 QDA is a model of higher complexity.

124 The estimates for the normal distribution parameters for each class is given by

$$\hat{\mu}_m = \frac{1}{n_m} \sum_{i:y_i=m} \mathbf{x}_i, \quad (5)$$

$$\hat{\Sigma}_m = \frac{1}{n_m - 1} \sum_{i:y_i=m} (\mathbf{x}_i - \hat{\mu}_m)(\mathbf{x}_i - \hat{\mu}_m)^T \quad (6)$$

125 derived from maximum likelihood estimation and adjusting  $\hat{\Sigma}_m$  to make it unbiased. The  
 126 *pooled covariance estimate* (weighted average of the covariance matrix estimates within each  
 127 class) is given by

$$\hat{\Sigma} = \frac{\sum_{m=1}^M (n_m - 1) \hat{\Sigma}_m}{\sum_{m=1}^M (n_m - 1)} = \frac{1}{n - M} \sum_{m=1}^M \sum_{i:y_i=m} (\mathbf{x}_i - \hat{\mu}_m)(\mathbf{x}_i - \hat{\mu}_m)^T. \quad (7)$$

128 With these estimators we may express the discriminant analysis classifier as

$$\hat{p}(y = m | \mathbf{x}) = \frac{n_m \mathcal{N}(\mathbf{x} | \hat{\mu}_m, \hat{\Sigma})}{\sum_{i=1}^M n_i \mathcal{N}(\mathbf{x} | \hat{\mu}_m, \hat{\Sigma})} \quad (8)$$

129 where  $\mathcal{N}(\mathbf{x} | \mu, \Sigma) = \frac{1}{(2\pi)^{M/2} |\Sigma|^{1/2}} \exp[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)]$  is the density for the normal  
 130 distribution with mean  $\mu$  and covariance matrix  $\Sigma$ .

## 131 3 Results

132 When comparing different models, it is important to have a baseline, or a null model to  
 133 compare against. In this case, an obvious null model is the constant model that always

134 predicts the same outcome regardless of input. The best null model is the one with highest  
 135 accuracy, i.e. the constant model that predicts the most frequently occurring outcome. The  
 136 model that always predicts a male lead has an accuracy of 0.756 and is thus chosen as the  
 137 baseline.

### 138 3.1 Logistic Regression

139 For all logistic regression models fitted, the set of regularization parameters,  $\Lambda$ , consisted of  
 140 10 logarithmically spaced values between  $10^{-4}$  and  $10^4$ . This was the default value in the  
 141 methods from scikit learn and having more densely packed values did not affect the model  
 142 performance in any appreciable way. The number of folds used in cross-validation was also  
 143 10, no improvement was observed by increasing this value.

144 In Tables 2 and 3, the accuracy and AUC are estimated using the mean of 100 bootstrap  
 145 samples in the case of LASSO regression and 400 in the case of Ridge regression. The reason  
 146 for having different sample sizes is that computing the LASSO regression is much more  
 147 computationally demanding.

Input	Regularization	Accuracy	AUC
Before transformations	None	0.870	0.878
	LASSO	0.871	0.880
	Ridge	0.871	0.880
After transformations	None	0.893	0.920
	LASSO	0.895	0.921
	Ridge	0.894	0.921

Table 2: Accuracy and AUC for logistic regression models. 70% training data.

Input	Regularization	Accuracy	AUC
Before transformations	None	0.876	0.878
	LASSO	0.875	0.883
	Ridge	0.871	0.880
After transformations	None	0.895	0.924
	LASSO	0.897	0.924
	Ridge	0.898	0.923

Table 3: Accuracy and AUC for logistic regression models. 90% training data.

148 We see that the regularization does not affect the model performance much. LASSO and  
 149 Ridge regularization perform almost identically and yield at best around 0.003 extra accuracy  
 150 but considering that the different splits of the data yielded estimated test errors in a range  
 151 from 0.8 to 0.98, we cannot reject that regularization does not matter in this case.

### 152 3.2 k-Nearest Neighbors

153 When hyper-tuning  $k$ -NN the set of  $p$ -values and  $k$ -values were given by  $\{1, 1.25, 1.5, \dots, 4\}$   
 154 and  $\{1, 2, 3, \dots, 25\}$  respectively. The number of folds used in cross-validation was again set to  
 155 10. We found that  $p = 2$  (Euclidean distance) and  $k = 4$  performed best for our transformed  
 156 data set. Using these parameters the  $k$ -NN algorithm was tested and performance was  
 157 measured with the mean of 100 bootstraps samples, the size of the sample was chosen with  
 158 regards to  $k$ -NN being computationally demanding. The results are summarized in tables 4  
 159 and 5.

160 It is obvious that  $k$ -NN is drastically improved by transforming the data, before transfor-  
 161 mations the model performance was even outperformed by, or just slightly better than the  
 162 best null model. Weighted  $k$ -NN with the distance weight seemed to perform better than

Input	Weighted k-NN	Accuracy	AUC
Before transformations	Uniform	0.745	0.675
	Distance	0.780	0.688
After transformations	Uniform	0.864	0.883
	Distance	0.872	0.888

Table 4: Accuracy and AUC using k-NN with 70% training data.

Input	Weighted k-NN	Accuracy	AUC
Before transformations	Uniform	0.750	0.678
	Distance	0.783	0.693
After transformations	Uniform	0.875	0.891
	Distance	0.882	0.901

Table 5: Accuracy and AUC using k-NN with 90% training data.

the uniform weight both before and after the transformation, the impact seems to be an increase of 0.007-0.008 in accuracy after transformation and an increase of almost 0.03 in accuracy before transformations.

### 3.3 LDA and QDA

The given dataset was bootstrapped 400 times and both DA models were tested before and after input transformations. From the Tables 6 and 7 we can draw the conclusion that QDA

Input	DA Model	Accuracy	AUC
Before transformations	LDA	0.856	0.870
	QDA	0.818	0.849
After transformations	LDA	0.900	0.917
	QDA	0.945	0.984

Table 6: Accuracy and AUC for discriminant analysis models using bootstrap. 70% training data.

Input	DA Model	Accuracy	AUC
Before transformations	LDA	0.866	0.877
	QDA	0.840	0.869
After transformations	LDA	0.900	0.918
	QDA	0.947	0.984

Table 7: Accuracy and AUC for discriminant analysis models using bootstrap. 90% training data.

seems to be more apt for this problem after transformations. It is unclear which method performs better before the transformation. For QDA before transformations, the variance in accuracy is high which can be explained by the fact that the original inputs are close to being colinear. This in turn makes  $\Sigma$  close to being singular resulting in inaccurate matrix inversion. For example, the standard deviation of accuracy and AUC of the QDA classifier, before transformations, on 400 bootstrapped datasets with 90% training data were 0.076 and 0.081 respectively, compared to 0.021 and 0.019 after transformations, ceteris paribus. Fortunately, the transformed inputs are not colinear.

Cross validation was carried out to estimate the accuracy using 150 folds resulting in an estimated accuracy of 0.948 using 70% training data. As can be seen in figure 1, there is a

noticeable variance in accuracy for the different training sets suggesting that there might be outliers in the data that the model has problems accounting for. Increasing the number of folds also increases the minimum accuracy that can be found in the corresponding box-plot, as to be expected. Also using cross validation to compare the effects of input transformations we see a 0.090 increase in accuracy using the transformed inputs compared to the original inputs. Adding the variables 'Years' and 'Gross' back into the inputs we see a decrease in accuracy of 0.007 hence they are left out.

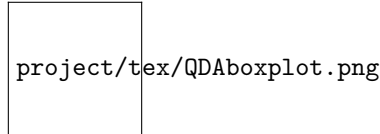


Figure 1: Accuracy estimation using cross validation with 150 folds.

185

## 186 4 Conclusions

It seems like  $k$ -NN was the worst performing of the models. Worth noting is the drastic effect that data transformations has on the  $k$ -NN model, performing at the same level as the baseline model before any transformations. The best results for the  $k$ -NN were a mean accuracy score of 0.882 and a mean AUC Score of 0.901 which is found in table 5.

Logistic regression slightly outperformed  $k$ -NN. Looking at the best performing setup we see that logistic regression had a mean accuracy score of 0.898 and a mean AUC score of 0.923 which is seen in table 3.

LDA had a mean accuracy score of 0.903 which is only a 0.5% increase from the logistic regression case, considering that both the results from logistic regression and LDA had some variance we cannot reject that LDA and logistic regression performance is similar.

QDA however had the best performance among all the models performing at best with a mean accuracy score of 0.942 and a mean AUC score of 0.977(!). Almost a 4% increase in accuracy compared to the second best method. One problem with the QDA model was however the outliers seen in the boxplot of Figure 1. Similar outliers were found in the case of  $k$ -NN and logistic regression these could probably be explained by the occurrence of a "bad split" where for example many movies deviating from the average movie end up in the test portion resulting in a bad performance. However the model performed well enough overall such that QDA was chosen for production.

1. Do men or women dominate speaking roles in Hollywood?

Based on the data it seems like males are in the lead. In 75.6% of the cases the lead is male which was found by looking at the baseline model. Also the mean for "Proportion of words female" was calculated to 34,6% indicating that Hollywood movies consist of almost 65% male speaking.

2. Has gender balance in speaking roles changed over time (i.e. years)?

Since none of the models we used seem to be effected by removing the year variable we cannot say that this is the case. However looking at the correlation in the data we see that there is a positive correlation between "Years" and each of the variables that show female activity in movies (i.e. "Number of words female", "Number of female actors" and "Mean Age Female"), indicating that there may be some changes in the gender balance over time.

3. Do films in which men do more speaking make a lot more money than films in which women speak more?

219 Since none of the models seem to be effected by removing the gross variable we  
220 cannot say that gross is a good indicator for determining whether there is more  
221 male speaking than female in a given movie. That is films with more male speaking  
222 cannot be said to make more money than a film with more female speaking.

## 223 5 Feature Importance

224 In order to determine which of the features Year, Gross or Number of female actors is most  
225 important, we fitted models excluding one or more of the features. Using a logistic regression  
226 model with LASSO (L1) regularization, we found that the model performance in terms of  
227 prediction accuracy was completely unaffected by removing either or both of the features  
228 Year and Gross. On the other hand, any model that excluded the variable Proportion of  
229 words female (which contains all information about how much male and female actors speak),  
230 had its performance reduced drastically. As seen in Table 3, the model including all features  
231 had an accuracy of 90%. This dropped to 80% when removing the Proportion of words  
232 female. Comparing this to the null accuracy of 75.6%, we conclude that the Proportion of  
233 words female is a very important feature in the model.

234 The same results hold for both QDA and k-NN as well, the models are completely unaffected  
235 by removing either Year or Gross (or both), while suffering 8% accuracy loss for k-NN and  
236 12% loss for QDA. Further, not only accuracy is affected but AUC is affected in the same  
237 way, showing that Year and Gross are more or less redundant features in the model, while  
238 Proportion of words female is incredibly important for predicting whether the lead is male  
239 or female.



## 240 Appendix A: Code

241 Transformations of input variables and various common functions.

```
242 import numpy as np
243 import pandas as pd
244 import sklearn.preprocessing as skl_pre
245
246 rawData = pd.read_csv('train.csv')
247
248 cols_to_norm = [
249     'Total words',
250     'Year',
251     'Gross',
252     'Mean Age Male',
253     'Mean Age Female',
254     'Age Lead',
255     'Age Co-Lead',
256     'Number of actors'
257 ]
258
259 def pre_process(raw_data, cols_to_norm):
260     data = raw_data.copy()
261
262     data['Lead'] = pd.get_dummies(data['Lead'])
263     data['Number of words co-lead'] = data['Number of words lead'] -
264         data['Difference in words lead and co-lead']
265     data['Proportion of words lead'] = data['Number of words lead']/data
266         ['Total words']
267     data['Proportion of words co-lead'] = data['Number of words co-lead']
268         /data['Total words']
269     data['Ratio words co-lead lead'] = data['Number of words co-lead']/
270         data['Number of words lead']
271     data['Proportion of words female'] = data['Number words female']/((
272         data['Total words'] - data['Number of words lead']))
273     data['Number of actors'] = data['Number of male actors'] + data['
274         Number of female actors']
275     data['Proportion of female actors'] = data['Number of female actors']
276         /data['Number of actors']
277     data['Older lead'] = data['Age Lead'] < data['Age Co-Lead']
278     data['Older lead'] = pd.get_dummies(data['Older lead'])
279
280     scaler = skl_pre.StandardScaler()
281     data[cols_to_norm] = scaler.fit_transform(data[cols_to_norm])
282
283     return data
284
285 data = pre_process(rawData, cols_to_norm)
286
287 def fit_and_test(classifier, train, test, features, target, suppress_output
288     = False):
289     classifier.fit(train[features], train[target])
290     if not suppress_output:
291         skl_met.plot_roc_curve(classifier, test[features], test[
292             target])
293         print('accuracy: ' + str(classifier.score(test[features],
294             test[target])))
295         print(' auc: ' + str(skl_met.roc_auc_score(test[target],
296             classifier.predict_proba(test[features])[:,1])) + '\n')
```

```

297         print(skl_met.classification_report(test[target], classifier.
298             predict(test[features])))
299     return classifier
300
301 print('Null accuracy: ' + str(max([np.mean(data[target]), 1 - np.mean(data[
302     target]))]))

```

303 Logistic Regression

```

304 trainRatio = config['Train Ratio'][0]
305 seed = config['Random Seed'][0]
306 train, test = skl_ms.train_test_split(data, train_size=trainRatio)
307
308 featureSet1 = [
309     'Year',
310     'Gross',
311     'Number of actors',
312     'Proportion of female actors',
313     'Mean Age Male',
314     'Mean Age Female',
315     'Age Lead',
316     'Age Co-Lead',
317     'Total words',
318     'Proportion of words lead',
319     'Proportion of words co-lead',
320     'Ratio words co-lead lead',
321     'Proportion of words female',
322     'Older lead'
323 ]
324
325 features = featureSet1.copy()
326 #features.remove('Proportion of words female')
327 #features.remove('Year')
328 #features.remove('Gross')
329 #features = ['Proportion of words lead']
330 target = 'Lead'
331
332 # No regularization
333
334 B = 100
335 accuracies = []
336 aucs = []
337 for i in range(B):
338     train, test = skl_ms.train_test_split(data, train_size=trainRatio)
339     logReg = fit_and_test(skl_lm.LogisticRegression(penalty='none',
340         solver='newton-cg'), train, test, features, target,
341         suppress_output=True)
342     accuracies.append(logReg.score(test[features], test[target]))
343     aucs.append(skl_met.roc_auc_score(test[target], logReg.predict_proba
344         (test[features])[:,1]))
345
346 # LASSO
347
348 B = 100
349 accuracies = []
350 aucs = []
351 for i in range(B):
352     train, test = skl_ms.train_test_split(data, train_size=trainRatio)

```

```

353     logRegLasso = fit_and_test(sk_lml.LogisticRegressionCV(Cs=10, cv=10,
354                             penalty='l1', solver='liblinear', n_jobs=10), train, test,
355                             features, target, suppress_output=True)
356     accuracies.append(logRegLasso.score(test[features], test[target]))
357     aucs.append(sk_lmet.roc_auc_score(test[target], logRegLasso.
358                                     predict_proba(test[features])[:,1]))
359
360 # Ridge
361
362 B = 400
363 accuracies = []
364 aucs = []
365 for i in range(B):
366     train, test = skl_ms.train_test_split(data, train_size=trainRatio)
367     logRegLasso = fit_and_test(sk_lml.LogisticRegressionCV(Cs=10, cv=10,
368                             penalty='l2', solver='liblinear', n_jobs=10), train, test,
369                             features, target, suppress_output=True)
370     accuracies.append(logRegLasso.score(test[features], test[target]))
371     aucs.append(sk_lmet.roc_auc_score(test[target], logRegLasso.
372                                     predict_proba(test[features])[:,1]))

```

## 373 References

- 374 [1] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, *Supervised Machine Learning*.  
375 Pre-pub:Cambridge university Press, draft version: january 12, 2021 ed., 2021.
- 376 [2] “Scikit-learn, documentation - nearest neighbors.” [https://scikit-learn.org/stable/](https://scikit-learn.org/stable/modules/neighbors.html)  
377 [modules/neighbors.html](https://scikit-learn.org/stable/modules/neighbors.html). Accessed: 2021-02-23.
- 378 [3] “Scikit-learn, documentation - sklearn.neighbors.kneighborsclassifier.” [https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn-neighbors-kneighborsclassifier)  
379 [KNeighborsClassifier.html#sklearn-neighbors-kneighborsclassifier](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn-neighbors-kneighborsclassifier). Ac-  
380 cessed: 2021-02-23.