

---

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

---

**Anonymous Author(s)**

Affiliation

Address

email

**Abstract**

1

# 1 Introduction

## 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.

### 2.1 Input transformations

In the given dataset, there are columns for the total number of words spoken as well as the number of words spoken by the lead, the co-lead etc. This could present a problem since if we compare a movie where the lead says 10 out of 100 total words and another movie where the lead says 100 out of 1000 words, most models would think that the lead speaks more in the second movie and miss the fact that the *proportion* of words spoken by the lead is the same. For that reason we have transformed several input variables to express a proportion instead of absolute numbers. We also believe it might be important to have a dummy variable indicating if the lead or the co-lead is oldest. All transformations are given in Table 2.1.

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	$\frac{\text{Number of male actors} + \text{Number of female actors}}{\text{Number of male actors} + \text{Number of female actors}}$
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.

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

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

In order to improve regularization and k-NN, all remaining numerical input variables were centered and scaled by their standard deviation. This means that columns with proportions have values in the unit interval  $[0, 1]$  and the other numerical variables have values that are of roughly the same magnitude.

### 2.2 Logistic Regression

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

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

where  $\beta \in \mathbb{R}^p$  and  $g$  is the link function. In the case of logistic regression,  $Y|X \sim \text{Ber}(p)$  and the canonical link function is the logit link  $g(x) = \log\left(\frac{x}{1-x}\right)$  with  $g^{-1}(x) = \frac{\exp(x)}{1+\exp(x)}$ . Since

30  $Y|X \sim \text{Ber}(p)$ , we get  $E(Y|X) = p = g^{-1}(X \cdot \beta)$ . In other words,  $P(Y = 1|X = x) = g^{-1}(x \cdot \beta)$ ,  
 31 which we can use to predict  $Y$  given data  $x$ .

32 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  $\hat{y}(x; \beta) = g^{-1}(x \cdot \beta)$ . This  
 33 minimizes the mean squared error (MSE) loss function. A potential problem with this approach is  
 34 that there are no restrictions on the components of  $\beta$  and that can lead to overfitting, especially if  $n$  is  
 35 not much larger than  $p$ . To address that issue, one can introduce regularization.

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

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

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

49 Since cross-validation is used to estimate the hyper-parameter  $\lambda$ , this method cannot be used to  
 50 estimate the test error of the whole procedure. Instead, the dataset has to be split into a training set  
 51 and a testing set with a specified fraction of the total data in each set. The whole procedure above is  
 52 done on the training set and the test error is estimated by evaluating the performance of the model on  
 53 the testing set. However, this can yield significantly different estimates of the test error since only  
 54 one split into training and testing data is considered. To get a better estimate of the actual testing  
 55 error, a bootstrap procedure is performed.

56 Since the full dataset is an iid sample from some unknown distribution, the estimated test error  $\hat{E}_{new}$   
 57 is a random variable. By repeating the whole procedure  $B$  times (i.e.  $B$  independent splits into  
 58 training and testing data and subsequent fitting and cross-validation), a bootstrap sample of  $\hat{E}_{new}$  is  
 59 obtained which can be used to estimate the distribution (or at least properties thereof) of  $\hat{E}_{new}$ . This  
 60 is very computationally intensive but gives a much clearer view of the variability of the test error  
 61 compared to just computing it for one split.

## 62 2.3 k-Nearest Neighbors

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

68 The algorithm for  $k$ -NN can be implemented in a simple manner with a brute force algorithm  
 69 measuring the distance from the test data point  $x_*$  to each training data point  $x_i$ , where  $i = 1, \dots, n$   
 70 using some distance function  $d(x, y)$ . It is normal to use the Minkowski distance of order  $p$  which is  
 71 given by

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

72 Where  $p = 1$  is the Manhattan distance, and when  $p = 2$  we have the Euclidean distance of course  
 73 any distance function could be used. The brute force algorithm for  $k$ -NN is given by

74

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

78

79 ??A problem with the brute force algorithm is that all the training data has to be stored and each  
 80 distance has to be calculated which can be rather computer intensive. There are however algorithms  
 81 as the ball-tree and k-d tree which speeds up these calculations. The general principle is still the  
 82 same, exactly how these algorithms are performed are thus left out of the report.

83 In this case we let the Minkowski distance be our distance function and we let  $p$  and  $k$  be hyper-  
 84 parameters which are to be tuned. This is done in an analogous manner as in the case of finding the  
 85 hyper-parameter  $\lambda$  in the Logistic regression case above.

86 Weighted  $k$ -NN is an alternative approach to the normal  $k$ -NN where the  $k$  nearest neighbors also are  
 87 weighted based on how far or close from the test data point they actually are effecting the majority  
 88 vote such that for example closer points have "stronger" vote. In our case we tested between *uniform*  
 89 weights (Standard  $k$ -NN) and *distance* weights where the weight points equals

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

90 for each of the  $k$ -nearest neighbors, this results in giving closer neighbors a stronger influence. ??

## 91 2.4 LDA and QDA

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

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

94 We estimate the *uninformative prior probability* as  $\hat{p}(y = m) = \frac{n_m}{n}$  where  $n_m = \sum_{i=1}^n \mathbb{1}\{y_i = m\}$   
 95 and assume that  $p(\mathbf{x} | y = m)$  is a normal density with expected value  $\mu_m$  and covariance matrix  
 96  $\Sigma_m$ . The assumption that distinguishes LDA and QDA is that for LDA we assumes that  $\Sigma_1 = \Sigma_2 =$   
 97  $\dots = \Sigma_M$  but for QDA we make no such assumption, that is, we allow for the covariance matrices  
 98 to differ. A consequence is that LDA is a special case of QDA, hence QDA is a model of higher  
 99 complexity. The estimates for the normal distribution parameters for each class is given by

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

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

100 The *pooled covariance estimate* (weighted average of the covariance matrix estimates within each  
 101 class) is given by

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

102 Finally we may express the discriminant analysis classifier as

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

103 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 function for  
 104 the normal distribution with mean  $\mu$  and covariance matrix  $\Sigma$ .

### 3 Results

#### 3.1 Logistic Regression

When comparing different models, it is important to have a baseline, or a null model to compare against. In this case, an obvious null model is the constant model that always predicts the same outcome regardless of input. The best null model is the one with highest accuracy, i.e. the constant model that predicts the most frequently occurring outcome. The model that always predicts a male lead has an accuracy of 0.756 and is thus chosen as the baseline.

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

The model performance was measured by accuracy (1 - misclassification rate) and AUC (area under ROC curve). In Tables 2 and 3, the accuracy and AUC are estimated using the mean of 100 bootstrap samples in the case of LASSO regression and 400 in the case of Ridge regression. The reason for having different sample sizes is that computing the LASSO regression is much more 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.

We see that the regularization does not affect the model performance much. LASSO and Ridge regularization perform almost identically and yield at best around 0.3% extra accuracy but considering that the different splits of the data yielded estimated test errors in a range from 0.8 to 0.98, we cannot reject that regularization does not matter in this case.

#### 3.2 k-Nearest Neighbors

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

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 for k-NN models. 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 for k-NN models. 90% training data.

133 It is obvious that k-NN is drastically improved by transforming the data, before transformations the  
134 model performance was even outperformed or just slightly better than the best null model. Weighted  
135 k-NN with the *distance* weight seemed to perform better than the *uniform* weight both before and  
136 after the transformation, the impact seems to be 0.7 – 0.8% extra accuracy after transformation and  
137 almost 3% extra accuracy before transformations.

### 138 3.3 LDA and QDA

139 The given dataset was bootstrapped 400 times and both DA models were tested before and after  
input transformations. From the results it is clear that QDA seems to have better performance after

Input	DA Model	Mean Accuracy	Mean AUC
Before transformations	LDA	0.856	0.870
	QDA	0.818	0.849
After transformations	LDA	0.893	0.913
	QDA	0.933	0.966

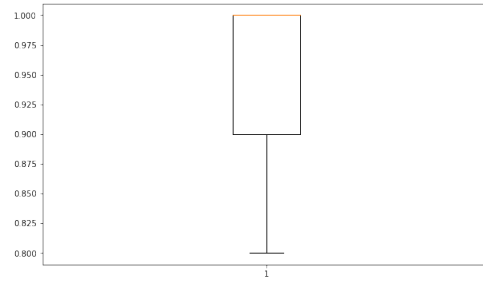
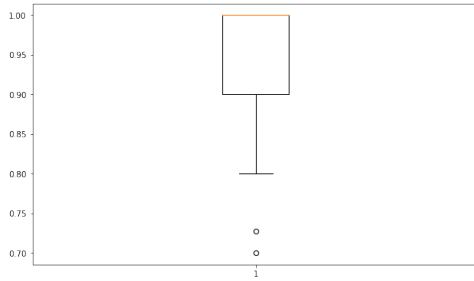
Table 6: Mean Accuracy and Mean AUC for discriminant analysis models. 70% training data.

Input	DA Model	Mean Accuracy	Mean AUC
Before transformations	LDA	0.866	0.877
	QDA	0.840	0.869
After transformations	LDA	0.903	0.915
	QDA	0.942	0.977

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

140 transformations. It is unclear which method performs better before the transformation because for  
141 QDA the variance is high. The high variance can for QDA can be explained by the fact that the  
142 original inputs are close to being colinear making  $\Sigma$  close to being singular which in turn results in an  
143 inaccurate matrix inversion. For example, the standard deviation of accuracy and AUC of the QDA  
144 classifier, before transformations, on 400 bootstrapped datasets with 90% training data were 0.076  
145 and 0.081 respectively compared to 0.021 and 0.019 after transformations ceteris paribus.

147 –Based on the results it seems that LDA is not complex enough of a model for  
148 this task. SKRIV OM VARFÖR QDA ÄR BÄTTRE ÄN LDA FÖR DET HÄR



– några

149 fall av väldigt dåliga resultat (70-75%) går detta att undvika? har johannes modell lika? är det pga  
 150 otur i uppdelningen av datan vid cross validation?? – 0.9428 accuracy cross validation 100 folds 70%  
 151 training data – ENDAST EFTER TRANSFORMATION – 0.9420 accuracy cross validation 100 folds  
 152 90% training data  
 153

## 154 4 Conclusions

155 It seems like  $k$ -NN was the worst performing of the models worth noting is the drastic effect that  
 156 data transformations has on the  $k$ -NN model, performing at the same level as the baseline model  
 157 before any transformations. Since the model is somewhat "blind" to feature importance and is greatly  
 158 affected by variances in the data. Normalization of the data is thus crucial for good results.

## 159 5 Feature Importance