
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 to 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.

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.

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'.

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.

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

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

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

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 minimizes the mean squared error (MSE) loss function. A potential problem with this approach is that there are no restrictions on the components of β and that can lead to overfitting, especially if n is not much larger than p . To address that issue, one can introduce regularization.

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

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

In order to find a value of λ 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 λ , 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

63 and a testing set with a specified fraction of the total data in each set. The whole procedure above is
 64 done on the training set and the test error is estimated by evaluating the performance of the model on
 65 the testing set. However, this can yield significantly different estimates of the test error since only
 66 one split into training and testing data is considered. To get a better estimate of the actual testing
 67 error, a bootstrap procedure is performed.

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

74 2.3 k-Nearest Neighbors

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

80 The algorithm for k -NN can be implemented in a simple manner with a brute force algorithm
 81 measuring the distance from the test data point \mathbf{x}_\star to each training data point \mathbf{x}_i , where $i = 1, \dots, n$
 82 using some distance function $d(\mathbf{x}, \mathbf{y})$. It is normal to use the Minkowski distance of order p which is
 83 given by

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

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

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

90
 91 ??A problem with the brute force algorithm is that all the training data has to be stored and each
 92 distance has to be calculated which can be rather computer intensive. There are however algorithms
 93 as the ball-tree and k -d tree which speeds up these calculations. The general principle is still the
 94 same, exactly how these algorithms are performed are thus left out of the report.

95 In this case we let the Minkowski distance be our distance function and we let p and k be hyper-
 96 parameters which are to be tuned. This is done in an analogous manner as in the case of finding the
 97 hyper-parameter λ in the Logistic regression case above.

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

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

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

103 2.4 LDA and QDA

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

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

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

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

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

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

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

114 Finally we may express the discriminant analysis classifier as

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

115 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
 116 the normal distribution with mean μ and covariance matrix Σ .

117 3 Results

118 To compare between families of models and between which tuning is better we chose to focus on
 119 two measures: accuracy (on average, of how often does the model make a correct prediction) and
 120 ROC/AUC.

121 3.1 Logistic Regression

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

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

132 The model performance was measured by accuracy (1 - misclassification rate) and AUC (area under
 133 ROC curve). In Tables 2 and 3, the accuracy and AUC are estimated using the mean of 100 bootstrap
 134 samples in the case of LASSO regression and 400 in the case of Ridge regression. The reason for

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.

135 having different sample sizes is that computing the LASSO regression is much more computationally
136 demanding.

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

141 3.2 k-Nearest Neighbors

142 When hyper-tuning the algorithm the set of p -values and k -values were given by $\{1, 1.25, 1.5, \dots, 4\}$
143 and $\{1, 2, 3, \dots, 25\}$ respectively. The number of folds used in cross-validation was again set to 10.
144 It was found that $p = 2$ (Euclidean distance) and $k = 4$ performed best for our data set. Using
145 these parameters the k-NN algorithm was tested and performance was measured with the mean of
146 100 bootstraps samples as before, the size of the sample was chosen with regards to k-NN being
147 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.

148 It is obvious that k-NN is drastically improved by transforming the data, before transformations the
149 model performance was even outperformed or just slightly better than the best null model. Weighted

150 k-NN with the *distance* weight seemed to perform better than the *uniform* weight both before and
 151 after the transformation, the impact seems to be 0.7 – 0.8% extra accuracy after transformation and
 152 almost 3% extra accuracy before transformations.

153 3.3 LDA and QDA

154 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 seems to be more

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.

155
 156 apt for this problem after transformations. It is unclear which method performs better before the
 157 transformation. For QDA the variance in accuracy is high which can be explained by the fact that
 158 the original inputs are close to being colinear making Σ close to being singular which in turn results
 159 in an inaccurate matrix inversion. For example, the standard deviation of accuracy and AUC of the
 160 QDA classifier, before transformations, on 400 bootstrapped datasets with 90% training data were
 161 0.076 and 0.081 respectively compared to 0.021 and 0.019 after transformations ceteris paribus.

162 Cross validation was carried out to estimate the accuracy using 150 folds resulting in an estimated
 163 accuracy of 0.948 using 70% training data. As can be seen in table 8 there is a noticeable variance
 164 in accuracy for the different training sets suggesting that there might be outliers in the data that the
 165 model has problems accounting for. Increasing the number of folds also increases the minimum
 166 accuracy that can be found in the corresponding box-plot, as to be expected. Also using cross
 167 validation to compare the effects of input transformations we see a 0.090 increase in accuracy using
 168 the transformed inputs compared to the original inputs. Adding the variables 'Years' and 'Gross'
 169 back into the inputs we see a decrease in accuracy of 0.007 hence they are left out.

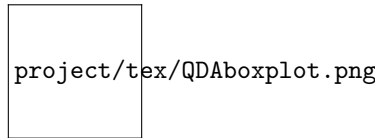


Table 8: Accuracy estimation using cross validation with 150 folds.

170 4 Conclusions

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

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

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

181 QDA however had the best performance among all the models performing at best with a mean
182 accuracy score of 0.942 and a mean AUC score of 0.977(!). Almost a 4% increase in accuracy
183 compared to the second best method. One problem with the QDA model was however the outliers
184 seen in the boxplot of 8. Similar outliers were found in the case of k -NN and logistic regression
185 these could probably be explained by the occurrence of a "bad split" where for example many movies
186 deviating from the average movie end up in the test portion resulting in a bad performance. However
187 the model performed well enough overall such that QDA was chosen for production.

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

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

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

194 Since none of the models we used seem to be effected by removing the year variable we
195 cannot say that this is the case.

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

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

202 5 Feature Importance

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

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