Do (wo)men talk too much in films? Project in Machine Learning

Aleix N. Juscafresa
aleixnieto@gmail.com

Dinindu Seneviratne
dinindu.seneviratne@gmail.com

Vasileios Toumpanakis
billtoubanakis@yahoo.gr

Abstract

The classification project for the 1RT700 course in statistical machine learning is contained in this document. The problem is identifying the gender of Hollywood films' two lead actors (one male, one female). For this, we are provided a training set of 1039 movies and we want to see whether certain features, such as the amount of dialogue the actors have, the year the film was made, and others, can predict the gender of the leading actor in a movie. In this project, we choose from a variety of models, one model to use in production against a test set. After comparing several models we have come to the conclusion that the model that works best for this set of data is a quadratic discriminant analysis with feature selection. The project concludes with a summary of the findings and a discussion of its possible future works.

1 Introduction

Identifying which of the two actors is male and which is female based on numerous film features is the challenge we want to tackle. This is a binary classification problem, although we are predicting the gender of the two leading actors. The lead actor is a woman if the co-lead actor is a man and vice versa. In this project, we will see to which extent determining whether the male or female lead role is predictable by comparing the quantity of conversation the actors had, the year the movie was created, the amount of money the movie made, and other factors. To be able to do this, the project has been divided into two major blocks.

The first block, before we start to build a model, consists in analyzing some statistics of the data set. In this step, we will gain some insights into the data which will help us to interpret the results of our methods later in the project. The data analysis has been divided into a general analysis where we present some dataset statistics and examine how to approach the handling of outliers and the unbalanced problem in the predicting variable. Also, using a graphical and statistical approach, we respond to three relevant questions.

In the second block, we will explore different classification methods and decide which is the best one in terms of performance in this specific scenario. To do this, the dataset will be divided into train and test samples. The algorithm will be trained using the training sample, and k-fold cross-validation (cv) will be performed to tune the hyperparameters (if the model requires it). The best model will then be chosen for each family of models (such as logistic regression, k-NN, random forest, etc.), and the performance of the selected model for each one of these families of models will be compared on the test set. By doing so, we may examine the models' generalization to new unseen data and determine whether they show overfitting issues. Among all these models we will select the one that performs better in the test set as the best model to put into production.

The project concludes with a summary of the findings and a discussion of different ways or additions that could be applied to make the project more complete.

2 Data analysis

In this section, we will examine our dataset and attempt to draw some conclusions by examining how different variables interact with gender class. In order to perform this analysis, we will both examine some dataset statistics and interpret graphs that will provide us with some insights into the data.

2.1 General analysis

Our dataset is made up of 1039 instances (rows), in this case, movies, and 14 variables (columns). Of these 14 variables, 13 are input variables (predictors) and one is an output variable, that is, the dependent variable that we will try to model from the inputs (target variable).

The target variable is defined as:

$$Y = \begin{cases} Female & \text{if} \quad \text{the person who says the most words in the movie is a woman} \\ Male & \text{if} \quad \text{the person who says the most words in the movie is a man} \end{cases}$$

A detailed description of the input variables can be found in Appendix A.1.

Missing data and outliers

Our dataset has no missing data in any of the variables. On the other hand, despite not having missing data, it can have outliers that come from measurement errors, sampling problems, etc. To study the outliers, statistical procedures have been left aside and a non-exhaustive and more heuristic analysis has been carried out by visualizing the data via boxplots. Later, in these boxplots, the interquartile range has been used to determine whether or not certain values are outliers. See Appendix A.1.

Data imbalance

Before digging into the data imbalance we must see how our target variable Y is distributed. Looking at the histogram in Appendix A.1 we can clearly see that the number of movies where the man leads is significantly bigger than the movies where the lead actor is a woman, concretely a 75.5% of movies are lead by men and the 24.5% are lead by women.

In our case, we are in a situation where the women who lead a film are the minority class because percentage-wise we have fewer examples of this class. This feature, in terms of data analysis, presents a problem known as to *unbalanced data*. This problem generally affects the algorithms in their process of generalization of the information harming the minority class.

Some of the strategies to counteract this problem include *oversampling*, *undersampling*, and *artificial sampling*. These strategies are recommended if we have populations of 99% and 1%. Since this is not our case, and after seeing in the models that the model does not improve oversampling and undersampling techniques, we have decided not to apply any of these techniques.

The imbalance of our dataset is not very significant and we do not believe that we can obtain an increase in the performance of our models. Other techniques such as changing the performance metric and/or the weight of individual data points could be performed to treat unbalanced datasets.

To determine whether one class is biased against another during model training, we will divide the data set. A percentage greater than 50% the observations is taken to train the model and the remaining percentage is used to test the model's error. In other words, the model is validated with observations that have not been used to train it, thus seeing how the model behaves with respect to the two classes.

Feature correlation

Later in the project, we will study the importance of the features when it comes to predicting our target variable. It is essential to know if particular features are correlated since high-correlation features have almost the same influence on the dependent variable. When two features have a high correlation, we can drop one of them. Among others, we can see that some features like *"number words male"* and *"total words"* or *"numbber words male"* and *"number of male actors"* are highly correlated. In the supplementary material, we can find the code where we have included automatic profiling obtained with *pandas-profiling*[1] where we can see all the correlations between features.

[1]Pandas-profiling primary goal is to provide a one-line exploratory data analysis (EDA) experience in a consistent and fast solution. Like *pandas df.describe()* function, pandas-profiling delivers an extended analysis of a dataframe while allowing the data analysis to be exported in different formats such as *html* and *json*.

2.2 Do men or women dominate speaking roles in Hollywood movies?

Yes, men clearly outnumber women in speaking roles in Hollywood movies. As we discussed in the data imbalance section in the general analysis we can see that 785 lead actors are men and 254 are women in our data. We also have discussed how data distributes in the number of words spoken by gender (taking into account the lead actor) which can be found in Appendix A.2.

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

When the average female words contribution per film is plotted against the time span given in the dataset, we can note a trend of growth in female words contribution across the timeline. See Appendix A.3.

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

Men talk more in films than women do. We estimated the average gross of films in which males speak more than women. The average gross is displayed in Table 1. We can also see a slight positive correlation between male speaking (including male leads when there are) and the gross income of the movie. Moreover, we have performed a t-test to compare the significance of the obtained average between male and female samples. We have obtained a p-value of 0.1337 which concludes that even though we see a positive trend the two means are significantly different. See Appendix A.4.

Table 1: Average gross income

Gender	Male	Female
Average gross	115.16	98.74

3 The implementation of the methods

In this section, we will give a brief overview of each method we considered as well as an explanation of how we applied it to solve our problem. We will also go through the tuning of the parameters as well as the reasons for the decisions made. Also, we will investigate which features are most accurate at predicting who speaks the most in movies.

Remember that the goal of the project is to try different classification methods, evaluate their performance on the problem, and then choose the best one to use and 'put in production' against a test set. But how do we choose the best model?

The strategy we have followed for selecting the best possible model is the following:

1. Split the data in a train and a test samples (As described in Section 2.1).
2. Using only the training data, select the right model among each method family using k-fold cross-validation (in case there are hyperparameters). Also in this step, we will perform the hyperparameter tuning that will lead to the best performance in each method type.
3. Train the selected methods among each method type in the whole training data and evaluate performances with some metric.
4. Having the best model for each model type at hand, we must not yet assume that the performance obtained in the training set is the "real" performance of each model. At this point, we introduce the last step, accompanied by the test dataset. For each of the best-selected models, we use the model to classify the test data and calculate the test error metrics. Here, the test set introduces another layer that minimizes the chances that a model benefits from pure randomness. This step is crucial in detecting if the model is overfitting by comparing training and test results and proving how well is each model generalizing in new unseen data.

A more detailed explanation of why this approach has been selected can be found in Appendix B.1.

In our project, we divided the data into the same train/test for evaluating all the models using the *sklearn* function *train_test_split* with a fixed random seed. For doing so we iterated over 5000 random seeds and selected the one that lead to the highest statistical similarity between the train and the test sets following the same procedure as [1]. Here we have minimized the Kolmogorov-Smirnov distances across all the cumulative distributions of our features in the training dataset and in the test dataset, the distance is defined as the maximum distance between these curves. The training set contains 75% of the observations and the validation set contains the remaining 25%. In Table 2, we see how the distributions of the samples have turned out.

Table 2: Splitting data

Class	**Sample**	**Train sample**	**Test sample**
Male	785	589	196
Female	254	190	64
Total	1039	779	260

Recall that for the majority of these methods the function *GridSearchCV*[2] has been used as an exhaustive search over specified parameter values for selecting the best hyperparameters via k-fold cross-validation (k will be specified for each method).

Before we can make and assess predictions we need to create a baseline, a reasonable metric that we aim to beat with our model. If our model cannot outperform the baseline, it is a failure, and we should try an alternative model or concede that machine learning is not the correct solution for our situation. In our problem, the baseline will be a naive classifier that always predicts male as the lead actor.

Let's now begin with the procedure for each one of the implemented methods.

3.1 Logistic regression

All possible combinations of features have been checked in order to choose the one with the best performance. Scaling of input variables was used (test inputs were scaled with the same scaler fitted on the train set), and the model was created with *sklearn.linear_model.LogisticRegression*. The optimization algorithm for finding the maximum likelihood estimator of θ, was not influential on the final test accuracies, while the use of scaling was. In the above section, we noted practical issues in creating the model, see Appendix B.3.1 for more theoretical details.

3.2 Linear discriminant analysis (LDA)

All subsets of columns were checked. Scaling did not appear to improve test accuracy. The function *sklearn.discriminant_analysis.LinearDiscriminantAnalysis* was used. See Appendix B.3.2 for more details.

3.3 Quadratic discriminant analysis (QDA)

Experimentation with columns showed drastic improvements (feature engineering). It is important to note that in the initial set of features (for every subset of which we computed the train and test accuracy), we included both the given features together with three new ones that we have created from the other features. Concretely the derived features are percentages with respect to the total number of words: *"lead words percentage"*, *"co-lead words percentage"*, and *"female word percentage"*.

To come up with the best QDA model we used a brute force algorithm of checking all possible combinations of features (given features together with new ones created in the data analysis process). The top models (test accuracy-wise) produced by this technique had the phenomenon of having higher test accuracy compared to train accuracy. For this reason, after some experimentation, we selected a more mediocre model where the two accuracies were closer, with the test accuracy being slightly lower. (It remains an open question the reason why this happened, and whether the choice of selecting a mediocre model is efficient). The selected features were: *"total words"*, *"number of male actors"*, *"number of female actors"*, *"mean age male"*, *"mean age*

[2] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

female", *"age lead"*, *"age co-lead"*, *"lead words percentage"*, *"colead words percentage"*, *"female word percentage"*. Except for the last feature, all the rest are symmetric gender-wise. See Appendix B.3.3 for the explanation of the last three features. The function from *sklearn.linear sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis* was used. See Appendix B.3.3 for more details.

3.4 k-nearest neighbor (k-NN)

Within the context of this project, we have tried to use the k-NN classifier to predict the classes of the problem we are trying to solve. The steps we have taken to construct and test the classifier are discussed at a high level in this section, and a more in-depth description is provided in the Appendix B.3.4. k-NN is a distance-based, non-parametric classifier. This means that the classifier explicitly uses the training data when making predictions as opposed to a parametric model where the training data will be needed only to learn the parameters.

Hence we have given focus to the scaling of data and the use of cross-validation to find the best parameters. Furthermore, we have implemented a feature combination iterator that looks for the best k-NN model we could get from all possible feature combinations possible. See Appendix B.3.4b.

3.5 Classification tree

An explanation of how the decision trees work can be found in Appendix B.3.5. Regarding the application of this family of models to our problem, a fairly general application has been chosen because we know that this method is usually outperformed by other models that we will see below such as bagging or random forest. For this general approach, we have trained the default decision tree given by the *sklearn.tree* function *DecisionTreeClassifier* and we have performed a hyperparameter tuning with the previously mentioned function *GridSearchCV* to improve the performance with a tuned tree. The parameters taken into account for the hyperparameter tuning have been *min_samples_split*, *min_samples_leaf*, *max_depth*, and *max_leaf_nodes*. By doing this we have been able to improve the test accuracy from a 78% to an 80%. However, the precision and recall values we have obtained are indicative that the model is not treating the genders with the same importance paying more attention to male predictions than female predictions.

The trees before and after tuning and the hyperparameters that have been taken into account for the tuning can be found at Appendix B.3.5. In this appendix, we can also see the comparison between Gini and permutation feature importances from where we have obtained that *"number of female actors"* is the feature with a higher predictive power in the model.

3.6 Bagging

Bagging, or Bootstrap Aggregating, gets its name because it combines bootstrapping and aggregation to form one ensemble model. It is a method of aggregating homogeneous models based on the majority vote or the average in the regression case. A general explanation of how bagging works can be found in Appendix B.3.6.

The approach we have taken is training the default bagging model given by the *sklearn.ensemble* function *BaggingClassifier* and we have performed a hyperparameter tuning with the function *GridSearchCV*. The hyperparameters taken into account for the hyperparameter tuning have been *max_features*, *max_samples*, and *n_estimators*. The hyperparameters that have been taken into account for the tuning can be found at Appendix B.3.6. We also have obtained that *"number words female"* is the feature that has a stronger predictive power in our model.

3.7 Random forest

Random forest algorithm combines the methods of decision trees and bagging with a substantial modification. An explanation of how the random forest work can be found in Appendix B.3.7.

The approach we have taken for this model is training the default random forest given by the *sklearn* function *RandomForestClassifier* and we have performed a hyperparameter tuning with the function *GridSearchCV*. In this case, differently from the decision tree, we have not tuned the depth of the trees as trees are ideal candidates for bagging, since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias. Since trees are notoriously noisy,

they benefit greatly from the majority vote. The parameters taken into account for the hyperparameter tuning have been *"n_estimators"*, *"max_features"*, *"min_samples_split"*, and *"min_samples_leaf"*. A detailed description of the hyperparameters that have been taken into account for the tuning can be found at Appendix B.3.7. We also have obtained that *"number of female actors"* is the feature that has a stronger predictive power in our model.

3.8 Boosting

In this work, AdaBoost with base models of logistic regression, decision tree, and random forest have been fit to the training data. Through a thorough search over all feature combinations of length $>= 8$, and using stratified k-fold cross-validation, we could see that AdaBoost with logistic regression is the best-performing boosted model. The interpretability of the model is decreased or lost by applying boosting. A deeper explanation of boosting and its implementation is found in Appendix B.3.8.

3.9 Neural network

As the last model we have implemented, we can find the neural networks. We have decided to apply a neural network with a single hidden layer because we have seen that with more hidden layers the model overfits and its performance is not optimal for our data. The *Keras* and *Tensorflow* libraries have been used to apply the neural network. Through a 5-fold cross-validation, several hyperparameters of the neural network have been tuned with the help of [2]. With the optimal hyperparameters, the neural network has been trained with all the training data and finally evaluated with the test sample. For the hyperparameter tuning, we have followed a sequential optimization of parameters focusing on the areas where we thought we would get the biggest improvement first. In our case, we have started the tuning with the structure of the network (layers and neurons). We also have imputed the outliers of 3 features, *"Number words male"*, *"Year"*, and *"Gross"*, as they lead to a better performance of the neural network. A detailed explanation of the hyperparameters that have been taken into account for the tuning can be found at Appendix B.3.9.

Finally, we have investigated how important the variables are at the time of prediction by looking at how they modify the performance of the model when they are not taken into account for its training. This has been done with *PermutationImportance* from the *eli5.sklearn* library. The three variables with the greatest impact on the prediction have been *"difference in words lead and co-lead"*, *"number of words female"*, and *"number of female actors"*.

4 The performance of the models

Some ideas in this section have been extracted from [3] and [4, Chapter 4].

In this part, we evaluate how well each method solves the problem. The best model (in terms of highest accuracy[3]) in the training data may be good, but it is also likely that one model benefited by being better suited to the random pattern in the training data.

This is not how we want to evaluate our models; we do not want to select a model based on the assumption that it is better, only to discover that the model outperformed due to chance. As a result, we introduce another layer that reduces the probability that a model benefits from pure randomness, the test set. But what if the training and test accuracies differ? We cannot use different data splits until our final model does better. This is because re-applying this procedure to find a better model under the same setting simply means, that we are probably finding a model that overfits the data, but may again work less well on new data in the future. This is why the test set should only be used once (after selecting models and hyperparameters) to estimate the new error for each final model.

By comparing the accuracies of the train and test sets, we can evaluate if the model is overfitting and compare how the models perform in unseen data being the parameter to decide which model to choose among all the different models. Finally, the test performance will provide us with an unbiased estimate of the new error, which can be reported to the customer as the expected performance.

[3] Accuracy is used as a synonym of performance, as accuracy is the selected metric for evaluating our classification models. A further discussion of why accuracy was selected could be discussed in future works.

Table 3: Accuracy comparison across all the implemented methods

Model	Training accuracy	Test accuracy
Naive Bayes classifier	0.757	0.750
Logistic regression (LR)	0.874	0.849
Linear discriminant analysis (LDA)	0.867	0.845
Quadratic discriminant analysis (QDA)	0.940	0.930
k-nearest neighbor (k-NN)	0.845	0.823
Decision tree (DT)	0807	0.804
Bagging	0.871	0.804
Random forest (RF)	0.933	0.830
AdaBoost with LR	0.924	0.881
Neural network	0.916	0.900

5 The best model

As already described in Section 4 we will use the performance of the algorithms in the test set to select the best possible model to put into production. Is this the only possible approach to the problem? Not really, we could have also tackled the problem from a more statistics-driven point of view as developed in Appendix B.2.

For this database and following the theory of splitting the data into train and test sets we believe that the model with the best performance is QDA. It shows robust values for precision and recall metrics which means it is not discriminating any class in the prediction. It also seems not to overfit when we compare train and test accuracy metrics. See Table 4.

Table 4: Comparison of 4 performance metrics between train and test datasets.

Metric	Training set	Testing set
Accuracy	0.940	0.930
Precision	0.954	0.944
Recall	0.967	0.964
F1-score	0.961	0.954

It should also be noted that the variable that has allowed us to obtain the best performance of the model is one of the variables that we have created, specifically the percentage of women's words with respect to the total number of words.

6 Conclusions

We performed a general analysis of the data and we even went further to see how some features are related, which later helped us to implement the methods. Regarding the implementation of the methods, we have separated our data into a training set and a test set, the same for all the methods, which have been implemented independently by the group members. Once all the models have been optimized through k-fold cross-validation, the best model of each family has been trained in the training set and its performance in this set has been compared with the performance onto the test set. In this way, the best model has been selected among all the methods with respect to our dataset to put into production, which has turned out to be QDA with feature engineering. To finish, we also have observed which features were more important in the prediction for some of the methods. As a general conclusion, we can say that the variables linked to the female class are the ones that have had the greatest impact. The reason why the variables related to the female class had more impact is something we would like to discuss in future works.

6.1 Possible future works

There are a number of avenues of future work that could be further explored. See Appendix B4.

References

[1] G. Malato, "Are your training and test sets comparable?," May 2022.

[2] J. Brownlee, "How to grid search hyperparameters for deep learning models in python with keras," Aug 2022.

[3] G. Röhrich, "Training, validating and testing–successfully comparing model performances," Sep 2020.

[4] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[5] J. Brownlee, "Statistical significance tests for comparing machine learning algorithms," Aug 2019.

[6] N. S. Chauhan, "Model evaluation metrics in machine learning," May 2020.

[7] J. Brownlee, "How to use discretization transforms for machine learning," May 2020.

Appendix

Code

The code for the project can be found at https://github.com/aleixnieto/SML_project as supplementary material.

A Data analysis

A.1 Descriptive univariant analysis

Table 5: Some descriptive statistics of the input features with their description.

Feature name	Feature description	Mean	Standard deviation	Min. value	25% perc.	50% perc.	75% perc.	Max. value
Year	Year that the film was released	2334.256	2157.217	0	904	1711	3030.5	17658
Number of female actors	Number of female actors with major speaking roles	11004.369	6817.397	1351	6353.5	9147	13966.5	67548
Number of male actors	Number of male actors with major speaking roles	4108.257	2981.251	318	2077	3297	5227	28102
Gross	Profits made by film	2525.024	2498.747	1	814.5	1834	3364	25822
Total words	Total number of words spoken in the film	7.767	3.901	1	5	7	10	29
Number of words male	Number of words spoken by all other male actors in the film (excluding lead if lead is male)	1999.862	10.407	1939	1994	2000	2009	2015
Number of words female	Number of words spoken by all other female actors in the film (excluding lead if lead is female)	3.507	2.089	1	2	3	5	16
Number of words lead	Number of words spoken by lead	4561.856	3417.856	0	2139.5	3824	5887.5	31146
Difference in words lead and co-lead	Difference in number of words by lead and the actor of opposite gender who speaks most	111.149	151.762	0	22	60	143.5	1798
Lead age	Age of lead actor	42.354	7.817	19	37.481	42.6	47.333	71
Co-lead age	Age of co-lead actor	35.93	8.957	11	29.5	35	41.5	81.333
Mean age male	Mean age of all male characters	38.716	12.286	11	30	38	46	81
Mean age female	Mean age of all female characters	35.486	12.047	7	28	34	41	85

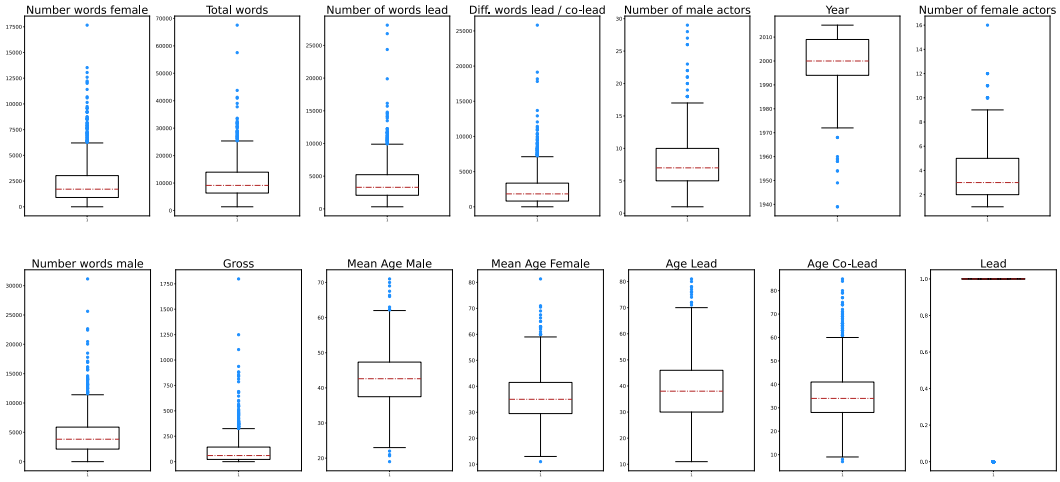

Figure 1: Boxplots for each one of the features. Some features like *"number words female"*, *"number words male"*, *"gross"* have at least one movie where these features take an extreme value.

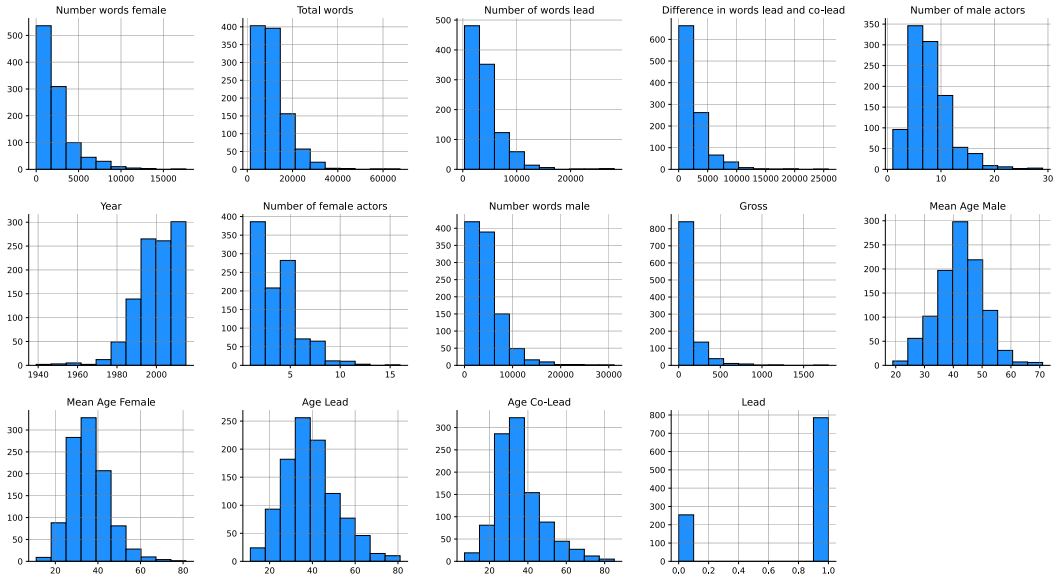

Figure 2: Histograms for each one of the features. The last one corresponds to the target variable where we can see a major number of men leading the movies. By analyzing the histograms in more detail we can extract some than other observations. For example, as we approach today, the number of films increases, and the average age of men follows a normal distribution while women's distribution is left-skewed, meaning that a woman is more likely to be young than old in a movie. Another important observation is that in many films the number of women is very small (there are even 21 films where no woman appears) while looking at the men's distribution we can see that in most of the movies are always at least some men.

A.2 Do men or women dominate speaking roles in Hollywood movies?

Since the contribution of words is highly influenced by the imbalance in the gender proportion, we looked at answering the question: How would the words contribution for each gender be distributed given that there was a balanced gender presence in the films? Hence we normalized the data based on the gender presence to see that the distribution is still skewed slightly towards a higher contribution of words by male actors, with a mean of 55% to be precise.

The word contribution for a given gender here is the percentage of words spoken by the actors of that gender with respect to the total number of words spoken by all actors. The gender presence for a given gender is the percentage of actors of that gender with respect to the total number of actors contributed. Hence we divide the gender-wise words contribution by the gender presence to arrive at a normalized distribution of words contribution which helps us to answer the question mentioned above.

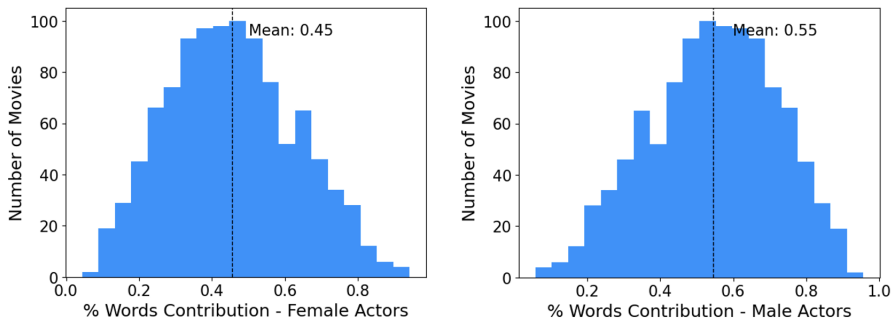

Figure 3: Normalized distribution of the percentage of words contribution for each gender.

From the empirical cumulative distribution function (Figure 4 - right), we see male and female leads having similar distributions of word percentage. It is easier to see it from the empirical cumulative distribution function since the histograms (Figure 4 - left) have different sample sizes.

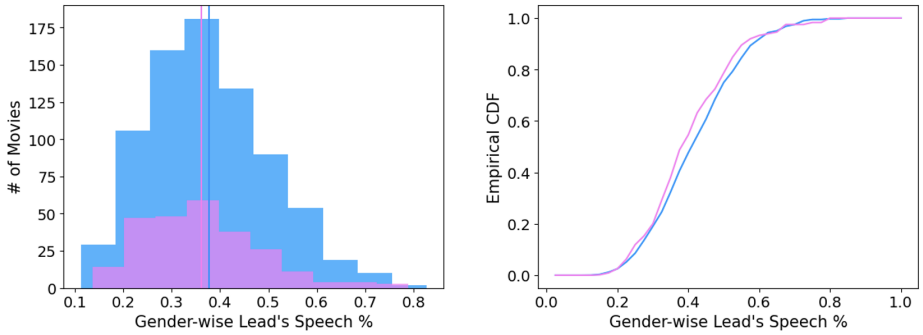

Figure 4: Gender-wise lead's percentage speech distribution.

Similarly, in Figure 5 we see that male co-leads speak more, compared to female co-leads.

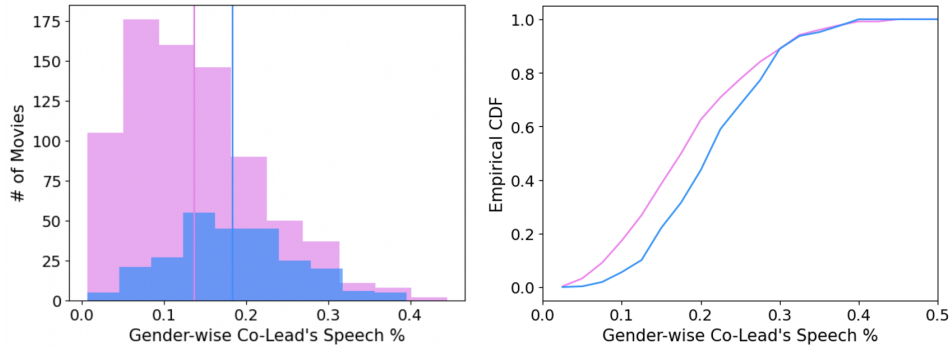

Figure 5: Gender-wise co-lead's percentage speech distribution.

An extra approach we have taken to compare the words of men and women have been to check the difference between the words spoken by males and females. Note that in these words we are not taking into account the words spoken by the lead. Denote D_i the difference in words spoken by male and female actors in the *ith* movie, $i = 1...N$. Summing up all these differences we obtain $D =$ *words male - words female*.

Then to answer the question above, we are interested in statements about $\mathbb{E}[D] \approx \dfrac{\sum_1^n D_i}{n} = \bar{D}$.

Using Hoeffdings inequality, a confidence interval for $\mathbb{E}[D]$ is given by:

$$\mathbb{P}(\bar{D} - \sqrt{(b-a)^2 \ln(2/\alpha)/(2n)} < \mathbb{E}[D] < \bar{D} + \sqrt{(b-a)^2 \ln(2/\alpha)/(2n)}) > 1 - \alpha$$

For a 95% confidence interval, we obtain:

$$\mathbb{P}(\mathbb{E}[D] \in [1697, 7490]) > 0.95$$

Thus, there seems to be a difference in words not spoken by the lead in movies between the two genders. An assumption in the construction of this interval is that the data is representative of all Hollywood movies.

A.3 Has gender balance in speaking roles changed over time (i.e. years)?

On the left plot of Figure 6, we can see a positive trend in the words spoken by females across the years. On the right plot of Figure 6, we can see a positive trend in the female lead percentage across the years. Note that some years have not been taken into account as we had no female leading moves in those years.

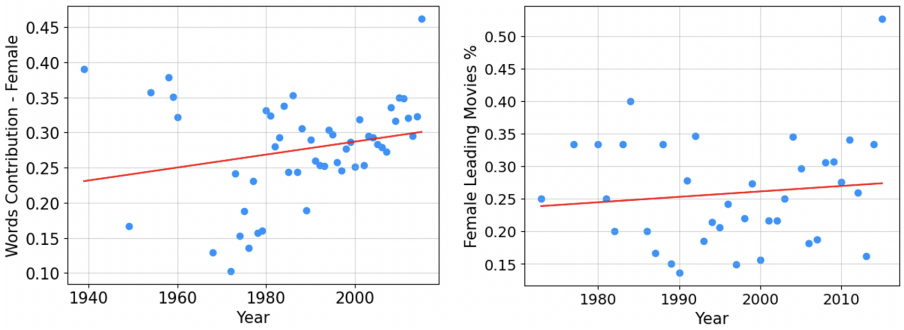

Figure 6: Female actor words contribution and percentage of males leading movies vs year.

A.4 Do films in which men do more speaking make a lot more money than films in which women speak more?

In Figure 7 we can see that there is a tendency to earn higher income when the contribution from male lead actors is higher.

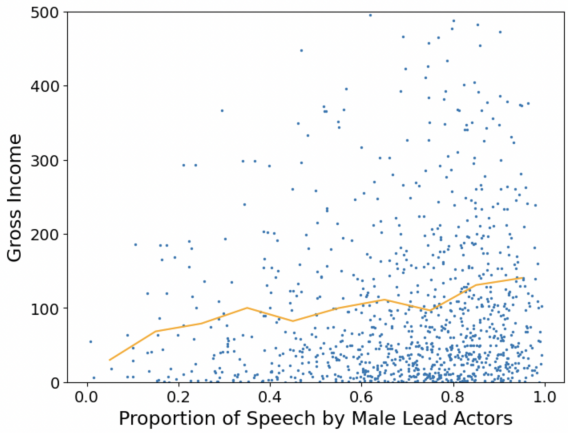

Figure 7: Relationship between male lead actors vs gross income

B Methods

B.1 Reason behind the chosen approach for evaluating model performance and model comparison

A question that may arise is whether it is really necessary to train/test split the data instead of performing cross-validation on all available data for model comparison, as we only have around 1000 data points and with the first technique, we lose data points with the test split. In our case where we have just a few data available, the answer to this dilemma is clear: always have an independent test split.

The less data we have, the more likely we will overfit a given set. This is also the reason why we use cross-validation in the training set for smaller datasets. Our performance estimates will be much more robust if we do cross-validation in our training set. In small datasets, such as our case, it is strongly advised because the train/test split can be noisy. However, the idea is that distributions match so we could use the p-value criteria for this. It may sound like a contradiction: we do not have much data available and we are splitting it. But think of it like this, it is less data, so having less to train on means our training process will pick a more conservative model (less max-depth of trees for example).

On the other hand, if we have a lot of data and a large validation set, one can be a bit overconfident, as we do not overfit too much even if you compared many points in a hyper-parameter grid. Pick the hyperparameters and choose the best model amongst model classes on the validation set.

Coming back to the discussion of why do not just apply cross-validation to all the available data. A very important use of the cv-error practice is to choose between methods and select different types of hyperparameters such that the cv error becomes as small as possible. By using cross-validation we get an estimate of the new error at the same time as a model trained on the entire dataset. However, much as we cannot use the training data error to estimate the new data error selecting models and hyperparameters based on cross-validation error will invalidate its use as an estimator of the new error. When we cross-validate the test scores represent the average performance across folds. This gives you a reasonable expectation of performance against many trials from your training but because the model is ultimately trained using all the data it leaves you with no out-of-sample performance metrics. If it is important to have a good estimate of the final new error, it is wise to first set aside another hold-out dataset, the one we have referred to as the test set. This test is an unbiased, not over-confident estimate of the actual error rate and should be used only once (after selecting models and hyperparameters), to estimate the new error for the final model.

If we wanted to apply cross-validation If we were going to choose the cross-validate option then it would be recommended to select a large number of k-folds in cross-validation or even leave-one-out cv. This is because the key in k-fold cross-validation is that if k is large enough, the intermediate models are quite similar to the final model since they are trained on almost the same dataset.

B.2 Statistical approach for model selection

The problem is that we want to ensure the robustness of your model as best we can. Just because one model shows 82% accuracy on your set of data and another shows 81%, it does not necessarily mean the 82% is better across new data. One potential option is to use some sort of iterative bootstrapping approach for training your models, and then build a distribution of model accuracies, from which you can determine if one model is significantly better than another for your metric using a simple test statistic. For more information see [5].

B.3 Deeper analysis behind the methods

B.3.1 Logistic regression

In binary classification, the model of logistic regression for every input x, is the following:

$$x \longmapsto \frac{e^{\theta^T x}}{1 + e^{\theta^T x}}$$

This output models the probability that the input belongs to class $+1$ (instead of -1). Then we try to find the maximum likelihood estimation of θ, that is, the parameters θ that maximize the *"probability"* (the density) that the model generated the outputs, given the inputs.

The maximization of the estimator, as said in the theory, does not have a closed-form solution, so an optimization algorithm is used.

For making *"hard"* decisions, i.e., choosing a class, we choose the most probable one.

B.3.2 Linear discriminant analysis (LDA)

Linear discriminant analysis assumes a multi-variate Gaussian distribution on the inputs of each specific class, with the conditions that all distributions have the same covariance matrix $\hat{\Sigma}$. That is, they are the same distribution translated to different means.

The MLE exists as a closed-form solution:

$$\hat{\mu_m} = \frac{1}{n_m} \sum_{i:y_i=m} x_i$$

$$\hat{\Sigma} = \frac{1}{n} \sum_{m=1}^{M} \sum_{i:y_i=m} (x_i - \hat{\mu_m})(x_i - \hat{\mu_m})^T,$$

where $m = 1, ..., M$ represents the M classes. However *sklearn* offers different solvers.

B.3.3 Quadratic discriminant analysis (QDA)

Quadratic discriminant analysis is more flexible than LDA. It fits a Gaussian mixture model, that is, we assume that the inputs of each specific class follow a multi-variate Gaussian distribution $P(x|Y = m) = N(x|\mu_m, \Sigma_m)$, for all inputs x and $m = 1, 2, ..., M$.

MLE exists in closed-form:

$$\hat{\mu_m} = \frac{1}{n_m} \sum_{i:y_i=m} x_i$$

$$\hat{\Sigma_m} = \frac{1}{n_m} \sum_{m=1}^{M} \sum_{i:y_i=m} (x_i - \hat{\mu_m})(x_i - \hat{\mu_m})^T,$$

for $m = 1, 2, ..., M$ the M classes. These estimates determine the Gaussian mixture model. For classifying a new input x_*, we use the Bayes rule to compute the class probabilities as follows:

$$p(y = m|x_*) = \frac{\hat{\pi_m} N(x_*|\hat{\mu_m}, \hat{\Sigma_m})}{\sum_{i=1}^{M} \hat{\pi_i} N(x_*|\hat{\mu_i}, \hat{\Sigma_i})}$$

For obtaining a *"hard"* prediction we select the most probable class.

Both QDA and LDA compute a specific Gaussian mixture model. That model captures the joint distribution of inputs and outputs. It can generate such pairs (generative model).

The three new features compute percentages of words respect the total number of words. The features are created as follows:

$$\textit{Lead words percentage} = \frac{\textit{Number of words lead}}{\textit{Total words}}$$

$$\textit{Co-Lead words percentage} = \frac{\textit{Number of words lead} - \textit{Difference of words lead and colead}}{\textit{Total words}}$$

$$\textit{Female word percentage} = \frac{\textit{Number of words female}}{\textit{Total words}}$$

Finally, we look at the feature importance of the bagging model by looking at the permutation feature importance:

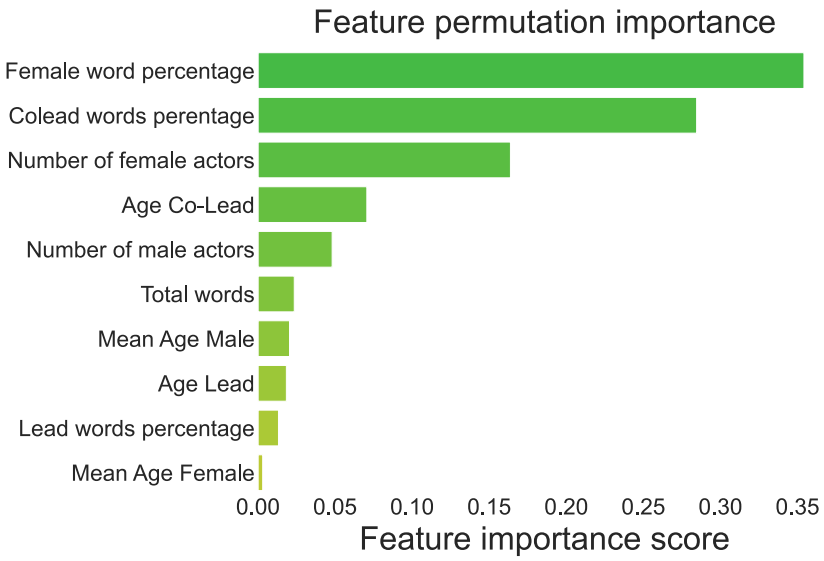

Figure 8: Feature permutation importance. The features are ordered from highest to lowest according to their contribution. As the feature permutation importance gives us the change in the model's performance, its values have been standardized to be represented on a scale relative to 1.

We can see that *"female word percentage"* and *"colead words percentage"* are the two features with a higher predictive power. Note that these two features were created from the original features.

B.3.4 K-nearest neighbor (k-NN)

B.3.4a Importance of Scaling - Specifically in KNN

The fact that KNN uses "euclidean distances" to predict an output raises some concerns regarding the input data. One important factor to consider is the scales of data of the input features. If two features are having different scales (say $X1 \rightarrow [1, 10]$ and $X2 \rightarrow [1, 1000]$), the classifier naturally will give precedence to the values of the feature with larger magnitude when distance is calculated.

To overcome this issue, we use *sklearn.preprocessing.StandardScaler* which standardizes the input values we give into the function via the standard normal transformation $\frac{(x-\mu)}{\sigma}$.

A major precaution we have to take is to not do the scaling before we split the dataset into train and test. Doing this might lead to test data leakage which could give us inaccurate results. Therefore, we have taken steps to split the data initially, scale both datasets based on training data and use them for training and testing.

B.3.4b Iterating to find the best k value for each feature combination

The strategy we used here was to go through each possible feature combination ($2^{13} - 1 = 8191$ to be precise) and find the best k value by considering the accuracy they give for each of them. We used *sklearn.model_selection.GridSearchCV* and *sklearn.pipeline.Pipeline* to implement this strategy.

A visual representation of how the model iterator was implemented is shown below in Figure 8.

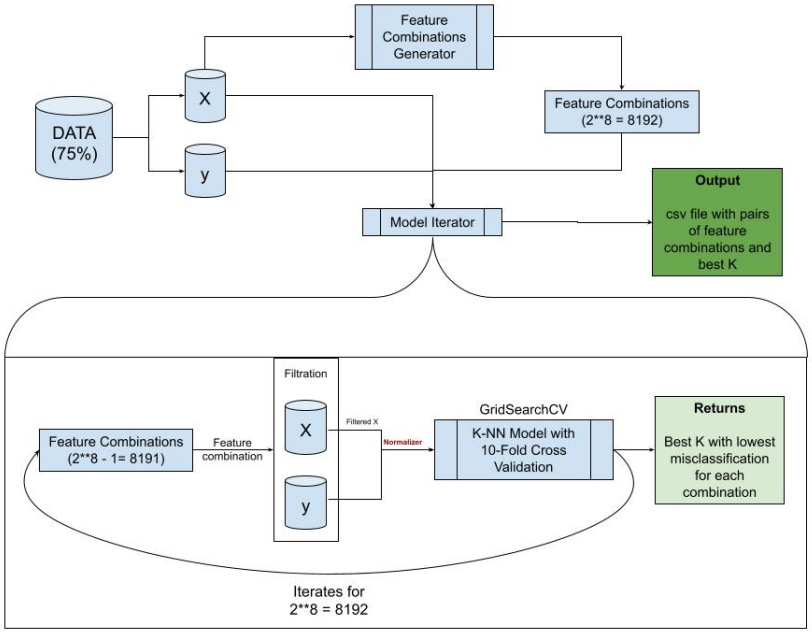

Figure 9: KNN model implementation

The models generated by the iterator were then trained and tested in the 75:25 split dataset. The models which showed the top 25 highest cross-validation accuracy are shown below.

From the iteration results, we picked the model at the top with the following parameters; K value and 9 features.

- *K value (Number of neighbors) - 10*
- *Number of words female*
- *Number of words lead*
- *Difference in words lead and co-lead*

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	best_k	CV accuracy	iteration_no	# of features	train accuracy	test accuracy
1	0	1	1	1	0	1	1	0	1	0	1	1	10	82.932%	7472	9	85.494%	79.615%
1	0	0	1	1	0	1	1	1	0	1	1	1	12	82.930%	7573	9	85.623%	80.000%
1	0	1	1	1	0	1	1	0	0	1	1	1	12	82.929%	7473	9	84.852%	80.769%
1	1	0	1	1	0	1	1	0	1	0	1	1	10	82.806%	7352	9	85.366%	80.000%
1	0	1	1	1	0	1	1	1	0	1	1	1	12	82.801%	8003	10	85.366%	79.231%
1	1	0	1	1	0	1	1	0	0	1	1	1	7	82.799%	7353	9	85.494%	80.000%
1	0	1	1	1	0	1	1	0	1	1	1	1	20	82.669%	8004	10	82.927%	77.308%
1	1	1	1	1	0	1	1	0	0	1	1	1	12	82.547%	7858	10	85.237%	80.769%
1	1	1	1	1	0	1	1	0	1	0	1	1	12	82.546%	7857	10	85.237%	78.846%
1	1	0	1	1	0	1	1	1	0	1	1	1	10	82.544%	7958	10	86.264%	79.231%
1	1	1	1	1	0	1	1	1	0	1	1	1	8	82.419%	8124	11	86.778%	80.385%
1	1	1	1	1	0	1	0	1	0	1	1	0	4	82.409%	7147	9	88.832%	78.077%
1	1	1	1	0	0	1	1	0	0	1	1	1	10	82.288%	7213	9	85.109%	80.000%
1	1	1	1	1	0	1	0	1	0	1	1	1	12	82.288%	7862	10	84.596%	79.615%
1	0	1	1	1	1	1	1	1	0	1	1	1	10	82.288%	8158	11	85.237%	78.462%
1	1	0	1	1	0	1	0	1	0	1	1	1	6	82.284%	7357	9	86.008%	79.615%
1	1	0	1	0	0	1	1	0	1	1	1	1	8	82.283%	7390	9	86.008%	80.385%
1	0	0	1	1	0	1	1	1	1	1	1	0	8	82.161%	7570	9	86.650%	78.846%
1	0	0	1	1	0	1	1	1	1	0	1	1	12	82.158%	7572	9	85.237%	80.769%
1	1	1	1	1	0	1	0	0	1	1	1	1	18	82.158%	7863	10	83.184%	77.692%
1	1	0	1	1	0	1	0	1	1	0	1	1	10	82.155%	7356	9	84.852%	79.615%
1	0	1	1	1	0	1	0	0	1	1	1	1	20	82.153%	7478	9	82.798%	79.231%
1	1	1	1	1	0	1	0	0	0	1	1	1	12	82.030%	7153	9	84.339%	79.615%
1	0	1	1	1	0	1	0	1	0	1	1	1	14	82.030%	7477	9	83.697%	79.615%
0	1	0	1	1	0	1	1	0	1	1	1	1	14	82.030%	7739	9	83.055%	78.462%

Figure 10: Results of iterations

- *Number of male actors*
- *Number of female actors*
- *Number of words male*
- *Mean age male*
- *Age lead*
- *Age co-lead*

After retraining selected model using the training set, the following metrics were observed.

Table 6: Comparison of 4 performance metrics between train and test datasets.

Metric	Training set	Testing set
Accuracy	0.845	0.823
Precision	0.849	0.834
Recall	0.966	0.954
F1-score	0.904	0.889

Finally we have looked at the feature importance of the KNN model by considering permutation feature importance.

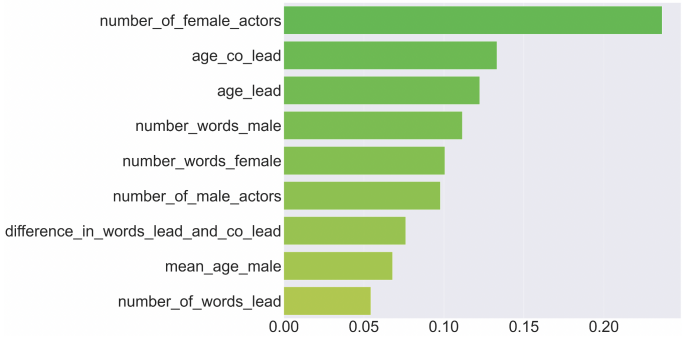

Figure 11: Feature permutation importance - KNN model

We can see that the most important feature in the model is *"number of female actors"* followed by *"age co-lead"* and *"age lead"*.

B.3.5 Classification tree

Many machine learning models fail in situations where the relationship between features and outcome is nonlinear or where features interact with each other. It is time for the decision tree (DT) to shine. Tree based models progressively split the data multiple times based on certain feature cutoff values until they reach sets that are small enough to be described by some label, with each instance belonging to one of them. In each split, the decision tree tries to split the data into two or more groups (we will use binary splits), so that the groups are as heterogeneous as possible from each other, and the data points that fall into the same group are homogeneous. For a multitude of reasons, decision trees are immensely popular, the most notable of which being their interpretability. They can be trained quickly and easily, which expands their potential well beyond the scientific context.

Before the parameter tuning, we trained a tree setting a maximum depth of 3 to avoid the overfitting we were getting not limiting this parameter (we were getting a 100% accuracy on the training set). The tree trained in all the training data with the default settings of the function *sklearn.tree.DecisionTreeClassifier* has achieved a 80.7% accuracy and takes the following form:

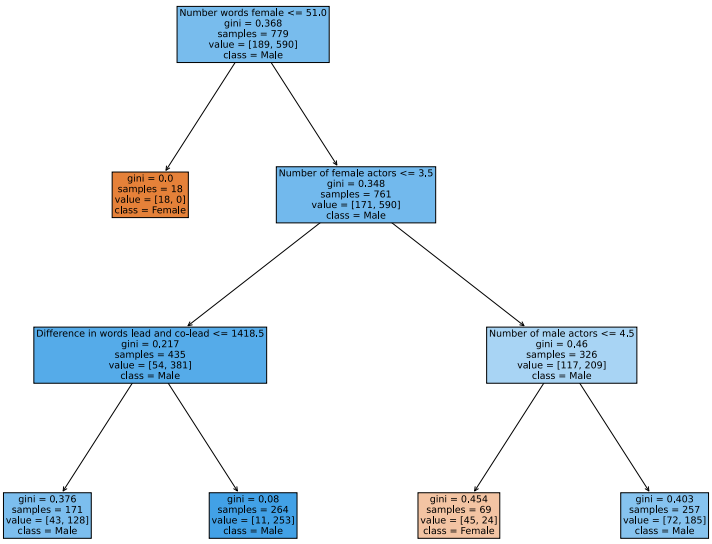

Figure 12: Default classification tree with max_depth = 3.

For the hyperparameter tuning we have selected the following hyperparameters:

Table 7: Hyperparameter description and its proposed values.

Hyperparameter	**Description**	**Proposed values**
min_samples_split	Minimum number of samples required to split an internal node.	[2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 40, 50]
min_samples_leaf	Minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least "min_samples_leaf" training samples in each of the left and right branches.	[2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 40, 50]
max_depth	The maximum depth of the tree.	[2, 3, 4]
max_leaf_nodes	The number of leaves in the tree must be less or equal than this value.	[2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 40, 50]

The values for *max_depth* that have been chosen are $2, 3$, and 4 as it is a way of preventing the tree from overfitting. We also have selected a 3-fold cross-validation as with higher values of folds the tree was paying too much attention to the training data and not generalizing well. Once performed cross-validation, we have obtained the following hyperparameters:

Table 8: Selected hyperparameters in the tree classifier. The selection has been performed via a 3-fold cross-validation using *accuracy* as the evaluation metric.

min_samples_split	**min_samples_leaf**	**max_depth**	**max_leaf_nodes**
2	15	3	5

Once obtained the hyperparameters, we can evaluate the decision tree performance by looking at different validation metrics in our training and test datasets:

Table 9: Comparison of 4 performance metrics between train and test datasets. The better performance in the training set is due to these data being observed during training, in contrast, to test instances that remain unseen. Such little difference ensures low overfitting and, hence, the good generalization of the model. See [6] for a detailed insight into these metrics.

Metric	**Training set**	**Testing set**
Accuracy	0.807	0.804
Precision	0.818	0.810
Recall	0.959	0.964
F1-score	0.883	0.880

The decision tree we have obtained is the following tree:

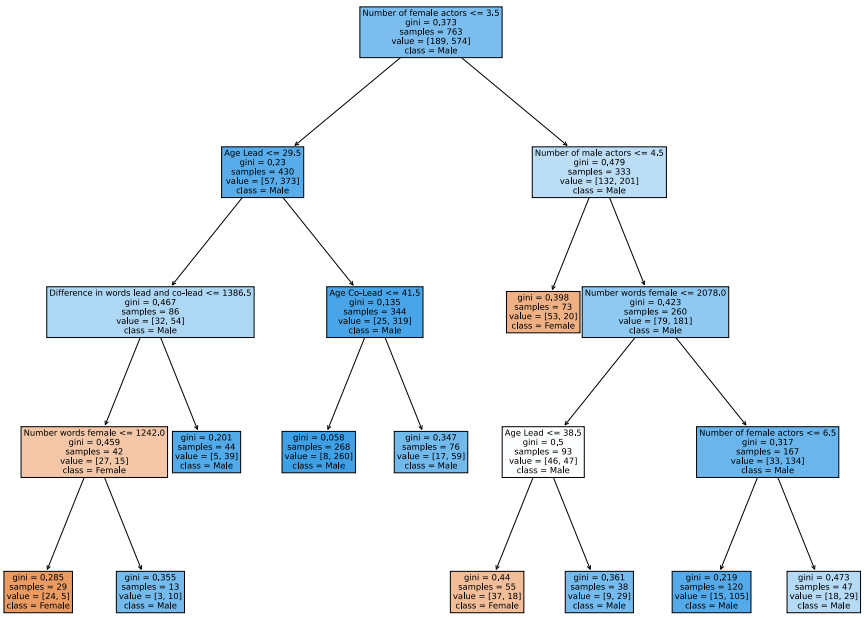

Figure 13: Tuned decision tree.

Finally, we look at the feature importance of the decision tree by looking at the Gini and permutation feature importances:

Figure 14: Global importances based on Gini and feature permutation importances. The features are ordered from highest to lowest according to their contribution. As the feature permutation importance gives us the change in the model's performance, its values have been standardized to be represented on a scale relative to 1.

We can see that in both cases the same variables are the most important in the two models where *"number of female actors"* stands out in both cases.

B.3.6 Bagging

Due to the theoretical variance of the training dataset (we remind that our dataset is an observed sample coming from a true unknown underlying distribution), the fitted model is also subject to variability: if another dataset had been observed, we would have obtained a different model. That is the variance of the model.

The idea of bagging is then simple: we want to fit several independent models and "average" their predictions in order to obtain a model with a lower variance. However, we cannot, in practice, fit fully independent models as it would require too much data. So, we rely on the good "approximate properties" of bootstrap samples to fit models that are almost independent.

Roughly speaking, as the bootstrap samples are approximately independent and identically distributed (i.i.d.), so are the learned base models. Intuitively we can understand it in this way. Suppose we have n random variables $X_1, X_2, \ldots, X_n$ which are independent and identically distributed drawn from a distribution of expected value given by μ and finite variance given by σ^2, then the *Central Limit Theorem* states that:

$$\bar{X}_n \xrightarrow{\mathcal{L}} \mathcal{N}(\mu, \frac{\sigma^2}{n}).$$

In other words, what this means is that for n large enough, $\bar{X}_n$ converges in law to a normal distribution $\mathcal{N}(\mu, \frac{\sigma^2}{n})$.

Here we can see that there is a reduction in variance when there is an aggregation happening. Then, "averaging" weak learners' outputs do not change the expected answer but reduce its variance, leading to an improvement of accuracy.

Focusing on the implementation of our model, before the parameter tuning, we were getting a 99% accuracy on the training set, i.e., the model was overfitting. In order to correct this overfitting we have performed a hyperparameter tuning via 3-fold cross-validation where we have selected the following hyperparameters:

Table 10: Hyperparameter description and its proposed values.

Hyperparameter	**Description**	**Proposed values**
n_estimators	The number of base estimators in the ensemble.	[100, 200, 300, 500]
max_samples	The number of samples to draw to train each base estimator with replacement.	[50, 75, 100]
max_features	The number of features to draw to train each base estimator without replacement.	[1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13]

We have limited the number of estimators and the maximum number of samples to 500 and 100 respectively as with higher values the model was overfitting on the training data. Once performed cross-validation, we have obtained the following hyperparameters:

Table 11: Selected hyperparameters in the tree classifier. The selection has been performed via a 3-fold cross-validation using *accuracy* as the evaluation metric.

n_estimators	**max_samples**	**max_features**
300	100	13

Once obtained the hyperparameters, we can evaluate the model performance by looking at different validation metrics in our training and test datasets:

Table 12: Comparison of 4 performance metrics between train and test datasets.

Metric	Training set	Testing set
Accuracy	0.871	0.804
Precision	0.856	0.803
Recall	0.998	0.979
F1-score	0.922	0.882

We see that the model clearly performs better on the training set inducing clear overfitting although we limited some hyperparameters. Our model does not produce false negatives as we see on the recall values, which tells us that pays a lot of attention to predicting males as the leading class.

Finally, we look at the feature importance of the bagging model by looking at the permutation feature importance:

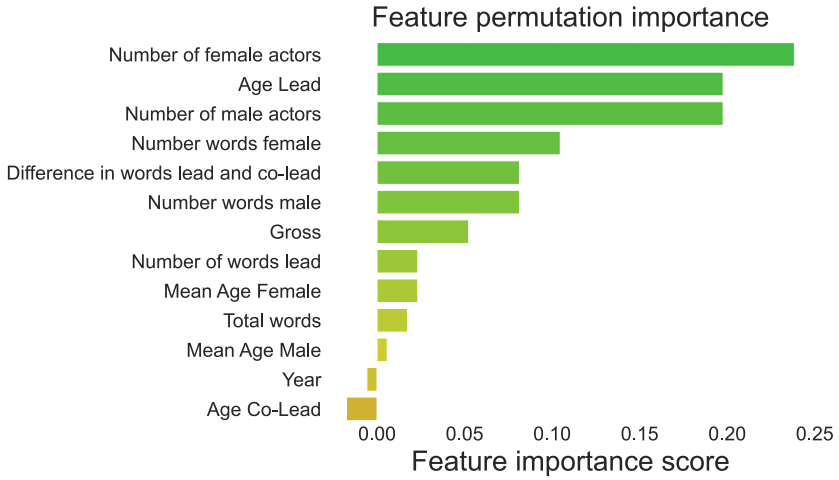

Figure 15: Feature permutation importance. The features are ordered from highest to lowest according to their contribution. As the feature permutation importance gives us the change in the model's performance, its values have been standardized to be represented on a scale relative to 1.

We can see that the most important feature in the model is *"number of female actors"* followed by *"age lead"* and *"number of male actors"*.

B.3.7 Random Forest

Trees are ideal candidates for bagging since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias. Since trees are notoriously noisy, they benefit greatly from the majority vote. Moreover, since each tree generated in bagging is identically distributed (i.d.), the expectation of an average of such M trees is the same as the expectation of any one of them. This means the bias of bagged trees is the same as that of the individual trees, and the only hope of improvement is through variance reduction. By the *Central Limit Theorem*, an average of M i.i.d. random variables, each with variance σ^2, has variance σ^2/M. In the case of bagged trees, as they are simply i.d., trees may not be independent when there is a variable with strong predictive power. Suppose each tree has variance σ^2 and the correlation between two trees is $\rho > 0$. Then, in the regression scenario, the variance of the average of M trees is:

$$
\begin{aligned}
Var\left(\frac{1}{M}\sum_{m=1}^{M}\hat{f}_m^*(x)\right) &= \frac{1}{M^2}\sum_{m=1}^{M}\sum_{l=1}^{M}Cov(\hat{f}_m^*(x),\hat{f}_l^*(x)) \\
&= \frac{1}{M^2}\sum_{m=1}^{M}\sum_{l\neq m}^{M}\Big(Cov(\hat{f}_m^*(x),\hat{f}_l^*(x)) + Var(\hat{f}_m^*(x))\Big) \\
&= \frac{1}{M^2}\sum_{m=1}^{M}((M-1)\rho\sigma^2+\sigma^2) = \frac{M((M-1)\rho\sigma^2+\sigma^2)}{M^2} \\
&= \frac{M^2\rho\sigma^2 + M\sigma^2(1-\rho)}{M^2} = \rho\sigma^2 + \frac{\sigma^2(1-\rho)}{M}.
\end{aligned}
$$

As $M \to \infty$, the second term disappears, and the variance of the model depends uniquely on $\rho\sigma^2$. The idea in random forest is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much, which will lead to a reduction in the overall model variance.

Here is where the modification of bagging comes into play, as we create a huge collection of *de-correlated* trees. When using bootstrap samples, at the splitting node step, we have the full disposal of features to choose from, which will cause that the data will mostly split off at the same features throughout each model. In order to correct this and reduce the correlation between trees, in random forests, instead of only sampling over the observations in the dataset to generate a bootstrap sample, we also sample over features and keep only a random subset of them to build the tree.

Emphasize that the trees obtained are maximal, that is, they are not pruned, having low bias but high variance and so are appropriate candidates for the bagging method that is mainly focused on reducing variance.

Sampling over features has indeed the effect that all trees do not look at the exact same information to make their decisions, leading to more diversification and reducing the correlation between the different returned outputs. Another advantage of sampling over the features is that it makes the decision-making process more robust to missing data: observations (from the training dataset or not) with missing data can still be regressed or classified based on the trees that take into account only features where data are not missing. Thus, the random forest algorithm combines the concepts of bagging and random feature subspace selection to create more robust models.

The low correlation between trees is the key, as a large number of relatively uncorrelated trees operating as a committee will outperform any of the individual constituent models. The reason for this wonderful effect is that the trees protect each other from their individual errors. While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.

As for our model, before the parameter tuning and just like the bagging model, we were getting a 100% accuracy on the training set, i.e., the model was overfitting. In order to correct this overfitting we have performed a hyperparameter tuning via 3-fold cross-validation where we have selected the following hyperparameters:

Table 13: Hyperparameter description and its proposed values.

Hyperparameter	Description	Proposed values
n_estimators	Number of trees in the random forest.	[25, 50, 75, 100, 150, 200]
max_features	Denote M the total number of model available features. The hyperparameter *"max_features"* $\leq M$ is the number of features considered at each split.	$\{\sqrt{M}, M\}$
min_samples_split	Minimum number of samples required to split an internal node.	[5, 6, 8, 9, 10, 15, 20, 25]
min_samples_leaf	Minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least *"min_samples_leaf"* training samples in each of the left and right branches.	[5, 6, 7, 8, 9, 10, 15, 20, 25]

By applying a 3-fold cross-validation across all different hyperparameters combinations we have obtained the following hyperparameters:

Table 14: Selected hyperparameters in the tree classifier. The selection has been performed via a 3-fold cross-validation using *accuracy* as the evaluation metric.

n_estimators	max_features	min_samples_split	min_samples_leaf
100	13	5	7

Once obtained the hyperparameters, we can evaluate the model performance by looking at different validation metrics in our training and test datasets:

Table 15: Comparison of 4 performance metrics between train and test datasets.

Metric	Training set	Testing set
Accuracy	0.933	0.830
Precision	0.922	0.835
Recall	0.994	0.964
F1-score	0.957	0.895

We see that the model clearly performs better on the training set inducing clear overfitting although we limited some hyperparameters. Our model does not produce false negatives as we see on the recall values, which tells us that pays a lot of attention to predicting males as the leading class.

Finally, we look at the feature importance of the bagging model by looking at the permutation feature importance in Figure 16. We can see that *"number of female actors"* and *"number words female"* are the two features that have a stronger impact on the model performance when are not used in the model. Also, highlight that some features like *"year"* or *"number word lead"* increase the model performance when are not taken into account.

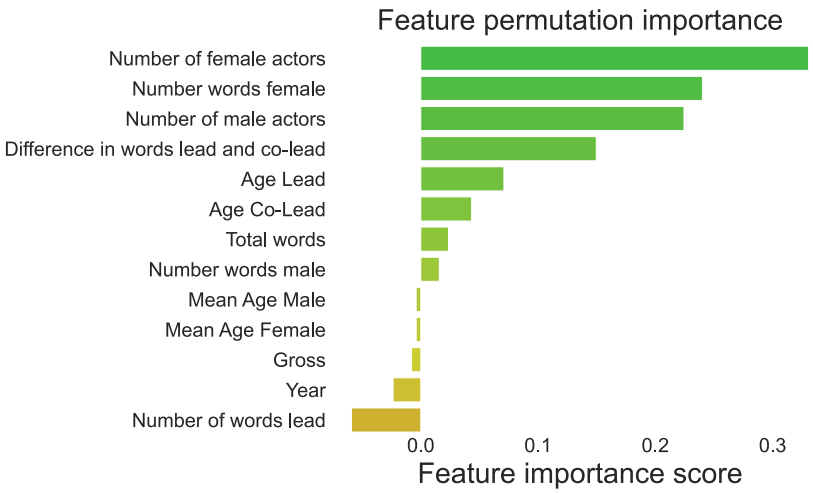

Figure 16: Feature permutation importance. The features are ordered from highest to lowest according to their contribution. As the feature permutation importance gives us the change in the model's performance, its values have been standardized to be represented on a scale relative to 1.

B.3.8 Boosting

Boosting is an ensemble method that aims to combine the output of "weak" models into one ensemble model. More precisely, the model is applied to different, re-weighted data sets, resulting in a sequence of models. In classification, the prediction of each model is combined to produce the final classification. The re-weighted data is done by applying weights $w_1, w_2, .., w_n$ to each data point. If the previous model misclassified a data point, then that data point is given a higher weight for the subsequent model.

A base classifier is a model that is sequentially trained and tested, and a wide variety of models can be used for this purpose. Naturally, as boosting is a bias-reducing algorithm, a natural candidate is a decision tree with low max depth. This work made use of the AdaBoost classifier in *scikit-learn* ensemble library to investigate the performance of boosting, by applying all combinations of features of length $>= 8$.

Surprisingly, in this study, the base classifier of logistic regression had the best accuracy metric relative to a decision tree and a random forest. The features used for this model are: *"number words female"*, *"number of words lead"*, *"number of male actors"*, *"number of female actors"*, *"number words male"*, *"mean age female"*, *"age lead'*, and *"age co-lead"*. As previous literature mentions that boosting is a bias reduction method, one would expect that a decision tree could perform the best, nonetheless, a logistic regression model had the best accuracy. These accuracy scores were obtained using stratified k-fold cross-validation. Stratification is important when the number of observations is small, or one class is dominating the observations. Since the last condition was true in this study, stratification was deemed advantageous. Further studies may investigate the performance of other boosting methods such as gradient boosting, and may further try different base classifiers not implemented here, such as QDA.

B.3.9 Deep neural network

For the hyperparameter tuning we have optimized the following hyperparameters:

Table 16: Tuned parameters.

Hyperparameter	Description	Proposed values
Number of neurons	Input neurons in the hidden layer.	[13, 12, 11, 10, 9, 8, 7, 6, 5, 4]
Epochs	An epoch means training the neural network with all the training data for one cycle.	[10, 25 ,50, 100, 150, 175, 200, 220, 250]
Optimizer	Function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate.	['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
Initializer	Initializer for the kernel weights matrix.	["RandomNormal", "RandomUniform", "TruncatedNormal", "Zeros", "Ones", "GlorotNormal", "GlorotUniform", "HeUniform", "Identity", "Orthogonal", "Constant", "VarianceScaling"]
Learning rate	Learning rate is a hyperparameter that controls the weights of our neural network with respect to the loss gradient. It defines how quickly the neural network updates the concepts it has learned.	[0.001, 0.01, 0.1, 0.2, 0.3]
Activation function in the hidden layer	Activation function to use.	['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']
Dropout rate	Fraction of neurons to be zeroed out.	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
Kernel constraint	Constraint function applied to the kernel weights matrix.	[0.0, 1.0, 2.0, 3.0, 4.0, 5.0]
Kernel L1 regularizer in the hidden and the output layer	Regularizer function applied to the kernel weights matrix.	hidden layer = [1e-2, 1e-3, 1e-4, 1e-5] output layer = [1e-2, 1e-3, 1e-4, 1e-5]

By applying a 5-fold cross-validation across all different hyperparameters combinations we have obtained the following hyperparameters:

Table 17: Selected hyperparameters.

Hyperparameter	Selected value
Number of neurons	7
Epochs	150
Optimizer	Adam
Initializer	GlorotUniform
Learning rate	0.01
Activation function in the hidden layer	ReLu
Dropout rate	0.1
Kernel constraint	4.0
Kernel L1 regularizer in the hidden and the output layer	1e-05, 1e-05

Finally, we look at the feature importance of the neural network by looking at the permutation feature importance in Figure 16. We can see that *"number of female actors"* and *"number words female"* are the two features that have a stronger impact on the model performance when are not used in the model. Also, highlight that some features like *"year"* or *"mean age male"* increase the model performance when are not taken into account.

Weight	Feature
0.1006 ± 0.0221	Number of female actors
0.0739 ± 0.0194	Number words female
0.0601 ± 0.0102	Difference in words lead and co-lead
0.0501 ± 0.0224	Number of male actors
0.0367 ± 0.0138	Number words male
0.0344 ± 0.0199	Age Co-Lead
0.0323 ± 0.0242	Age Lead
0.0118 ± 0.0070	Number of words lead
0.0003 ± 0.0010	Mean Age Female
0.0003 ± 0.0025	Total words
-0.0015 ± 0.0010	Gross
-0.0021 ± 0.0026	Year
-0.0044 ± 0.0075	Mean Age Male

Figure 17: Feature permutation importance. The features are ordered from highest to lowest according to their contribution. The feature permutation importance gives us the change in the model's performance.

B.4 Future work

1. Could applying principal component analysis (PCA) to the data before training our methods could improve their performance?
2. There is data leakage in the dataset?
3. Pay more attention to other metrics such as precision, recall, etc, where we can see how the model treats the male and female classes classification.
4. Could the models have been improved by discretizing some features? See [7] for a deeper insight into this topic.
5. Is accuracy the best metric for tackling the proposed problem?
6. Is there a data leakage in our original dataset which makes the female features more important when it comes to predictive importance.?
7. SHAP is an explainable artificial intelligence technique that is model agnostic, i.e., it can be applied to any machine learning model. SHAP develops a global explanation theory for the importance of the features in a model, something that could be applied in these models and see how it is compared with the feature permutation importance.
8. Try the feature engineering we have performed for QDA in other methods and see if they can also be improved.