SML-model for gender prediction of lead character in a movie

Marius Vangeli, Julia Sulyaeva and Viktor Umenius
Statistical Machine Learning
Uppsala University

March 6, 2023

Abstract

Gender inequality continues to be one of the main problems of developed societies, visible in every field of work. This report outlines the analysis of gender bias data in 1039 Hollywood movies and development of several machine learning models for prediction of a film's lead actor's gender, trained on this data. Finally, Quadratic Discriminant Analysis (QDA), is chosen as the best prediction model, and it's performance is analysed. This model coupled with some common optimization strategies results in an algorithm capable of predicting the gender of the main character in movies with 90% accuracy.

1 Introduction

The task description includes developing and testing several machine learning models as a solution to a binary classification problem of predicting gender of a movie's lead character. The goal of the project is choosing the best performing model. The given data for model training is a dataset consisting of 1039 movies, including 13 numerical variables describing movie's various features: such as number of male and female actors, movie's gross profits, number of words in a movie divided by actor gender, and others. The classifier variable is a binary categorical description of the lead actor's gender, expressed as a string 'male' or 'female'. A basic analysis of the data is performed in graphical form to highlight existing gender bias in movies through answering three questions connecting actors' gender to amount of roles, a film's profits and change over time (see section 2).

Model development and training is performed in Python using a module `scikit-learn`, containing implementations for all most common machine learning methods. The process includes defining the best suitable model evaluation methods and metrics, preparing the dataset and implementing 5 different

algorithms: Logistic regression, K nearest neighbors, Linear and Quadratic discriminant analysis and Classification tree (see section 3.2).

2 Data analysis

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

It is clear that men dominate speaking roles in Hollywood. The average number of major speaking roles held by men in movies is 7.8, which is more than double that of women which is 3.5. To further illustrate the difference, we made a graph showing the total number of speaking roles in our data set of 1039 movies, separated by sex (see Figure 1).

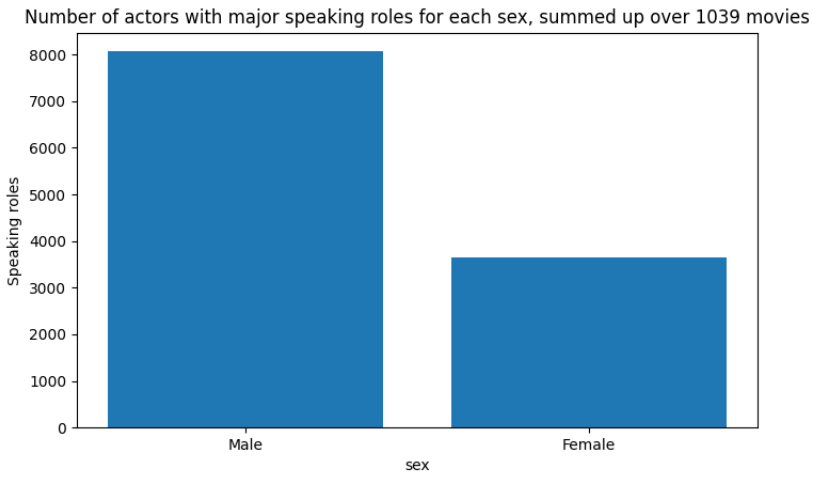

Figure 1: Amount of speaking actors divided by gender

2.2 Relationship between actors' gender and gross profits

The following plot (see Figure 2) presents the amount of total words said by actors of specific gender in Hollywood movies, sorted by the profits of each film. Most of the films lie in the range of 0-500 in terms are profits, but there are a few outliers. Most of the movies, resulting in a blue peak top on the graph, have more words spoken by men than women, with the few clearly visible exceptions to this rule highlighted in the graph with red arrows. Almost all films with exceptionally high profits have many more words spoken by male than female actors, and in the more average profit range (0-500) there's a gradual decline of female words. This suggests a negative correlation between how much women

speak in a film and the film's profits.

Most of the high-grossing Hollywood movies are action-movies or thrillers, which due to the nature and setting of a film (for example a western cowboy feature or a war thriller) and the main target audience (dominantly male) might be one possible explanation of such correlation.

Figure 2: Amount of men's (blue) and women's (orange) spoken words in films, sorted by the film's profit.

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

The number of speaking roles in Hollywood films throughout the years seem to consistently be dominated by men (as seen in 3). As represented by the blue peaks, speaking roles held by men dominate from the late 1930s until the middle of the 2010s. The number of speaking roles held by women increase from the first half of the 1900s to the 2000s. Although there is a trend showing an increase in speaking roles among women over time, a majority of movies made still have a greater number of men in speaking roles. This shows that the gender balance in speaking roles seem to improve over the years, but the number of speaking roles between the genders is still unbalanced.

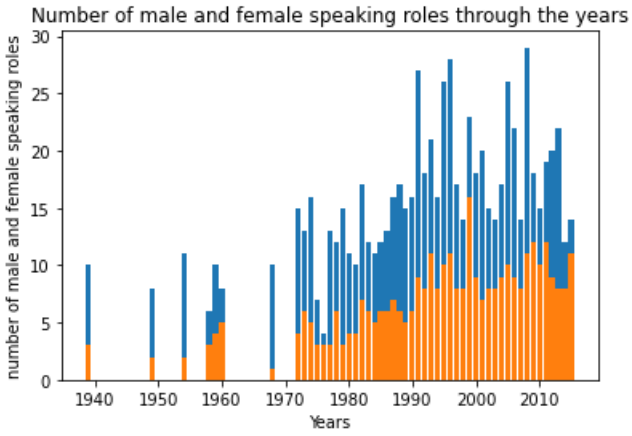

Figure 3: Amount of men's (blue) and women's (orange) speaking roles in films throughout the years

3 Model development

3.1 Methods

3.1.1 Logistic regression

Logistic regression is an altered version of linear regression that is used for classification problems. It takes the set of input features and assigns to each a coefficient. It then generates a score with a linear computation of the inputs and weights just like the linear regression model (Lindholm et al., 2022).

The linear regression model:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p = \theta^T \mathbf{x} \tag{1}$$

This is passed through the logistic (sigmoid) function:

$$h(z) = \frac{e^z}{e^z + 1} \tag{2}$$

Which results in the logistic regression model:

$$g(\mathbf{x}) = \frac{e^{\theta^T \mathbf{x}}}{e^{\theta^T \mathbf{x}} + 1} \tag{3}$$

The transformation of the output gives us an s-shaped curve bounded by [0,1], which is used to predict the probability of a data point belonging to the class 0 or 1, in our case male or female. Training the model is done with maximum likelihood. The goal is to minimize the error which is the difference in our

prediction and the true class label of the data point by adjusting the input weights. The learned model is achieved by solving the following equation:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \ln p(y_i|\mathbf{x}_i;\theta) \tag{4}$$

3.1.2 K Nearest Neighbors

K Nearest Neighbors classification algorithm (as opposed to regression algorithm) is a supervised learning algorithm consisting of 4 steps (Guo et al., 2003):

1. Calculating distance between new datapoint requiring classification and all other individual datapoints
2. Choosing K nearest neighbors to the new datapoint
3. Voting: calculating amount of neighbors belonging to each class
4. Assigning a class label based on majority vote

The algorithm can be summarized with the following formula:

$$\hat{y}(x^*) = MajorityVote(y_i = j \in N_x) \tag{5}$$

The distance between two datapoints is defined with either standard Euclidean distance (Zhang, 2016):

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{6}$$

Manhattan distance ("grid distance"):

$$d(x,y) = \sum_{i=1}^{n} |(x_i - y_i)| \tag{7}$$

or Minkowski distance, a generalization of both above definitions:

$$d(x,y) = (\sum_{i=1}^{n} |(x_i - y_i)|^p)^{1/p} \tag{8}$$

Other hyperparameters for the KNN algorithm include the possibility of adding weights to datapoints (for example depending on distance), and choosing the specific algorithm used to compute the nearest neighbors. The options are assembling the datapoints into a BallTree, a KDTree, using brute force (calculating distance to each point) or letting the algorithm choose.

3.1.3 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a generative model that uses each class information to determine what class new data points belong.

LDA assumes that the data in each of the classes have a Gaussian distribution and a covariance matrix. For each class it is assumed that the covariance matrix is the same for all classes, and is used to determine the highest likelihood of the new input(Joyce, James, 2019).

LDA focuses on projecting features of a higher dimension to a lower dimension by using a calculated decision boundary. The linear combination of input features that maximizes the between class variance and minimizes is called the discriminant function and is used to classify new inputs based on their features. This can be done in three steps.

1. Calculate the separability between classes, which is the mean of different classes. (Between class variance)
2. Calculate the distance between the mean and sample of each class. (Within-class variance)
3. Construct the lower-dimensional space which maximizes the between class variance and minimizes the within-class variance.

The discriminant function for LDA is given by:

$$g_i(\mathbf{x}) = \mathbf{x}^\top \sum_i \mu_i - \frac{1}{2}\mu_i^\top \sum_i \mu_i + \ln \pi(y = C_l) \tag{9}$$

LDA uses Bayes' theorem (Pedregosa et al, 2011)

$$P(Y|X) = \frac{P(X|Y)P}{P}(X) \tag{10}$$

which estimates the probability of each class and the probability of the data belonging to the class.

3.1.4 Quadratic Discriminant Analysis

Like LDA, the Quadratic discriminant Analysis (QDA) is a generative model that uses the statistical characteristics of the classes to make a decision to what class a new input belongs.

LDA and QDA are very similar methods that both assumes that all data in each class has a Gaussian distribution, however does QDA not assume that the covariance matrix is the same for all classes, instead it uses different covariance matrix for class.

The math behind QDA is very similar to the math behind LDA, but as a result of using different covariance matrices for each class, the decision boundary is quadratic surface instead of being linear. This results in QDA being able to have a more complex classification of new inputs(Joyce, James, 2019).

To classify new inputs, the probability of the point belonging to a class is calculated, based on the training data and its features. The new input is the assigned to the class with the highest probability.

The discriminant functions for QDA

$$g_k(\mathbf{x}) = -\frac{1}{2}\ln|\Sigma_k| - \frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln P(\omega_k) \quad (11)$$

3.1.5 Classification tree

The classification tree method works by taking the input data and dividing it in two based on the feature that best predicts the class of the training data (Lindholm et al., 2022). This creates a root node with two branches that separates the data based on the value of the chosen feature. This step is then repeated for the new nodes recursively until no more splits can be done or a specific limit is met.
The decision tree method is prone to over fitting so limits are set like for example a depth limit to the tree.
In the end you end up with a binary tree where each leaf node represents a predicted class for a subset of data points. The splitting criterion decides where the split is made. There are three different criterion.

Misclassification rate:

$$Q_l = 1 - \max(r, 1-r) \quad (12)$$

Entropy:

$$Q_l = -r\ln r - (1-r)\ln(1-r) \quad (13)$$

Gini index:

$$Q_l = 2r(1-r) \quad (14)$$

Where r is the proportion of instances in node l that belong to one of the classes.

3.2 Application

The data preparation steps consisted of shuffling the dataset, splitting it into a train and test subset (constituting roughly 80/20 split) and separating the classifier variable `Lead`. The train dataset was then used for training and tuning with the use of K-fold cross validation, and performance of the final models was evaluated on the test dataset. The data did not have any missing values or faulty outliers and thus did not require any cleaning. Furthermore, all the chosen models could handle a binary categorical classifier variable (as opposed to requiring a numerical one), which is why no classifier transformation was performed and no dummy variables used. Data reduction and normalization steps were performed separately for each method and chosen based on resulting accuracy scores. See below for explanation of the evaluation metric choice.

Each method was implemented using in-built functions from the python module `scikit-learn`. The dataset size and model complexity did not require

any specific k-value, allowing for an arbitrary choice of k=10 for the cross validation (Cunningham, Delany, 2007). A function performed Recursive Feature Elimination (RFE) on each model as a method for increasing accuracy by removing unnecessary features. A grid-search algorithm was then used which tested different combinations hyperparameters for each model for improved accuracy. Different normalization methods were tested on the models and compared with the original dataset to see if accuracy could be improved. The techniques that were used are: Min-Max scaling, standard scaling, robust scaling and the basic normalization method.

The models' performance was evaluated and compared to the performance of a naive classifier which only predicted the lead role to be male (with an accuracy score of 0.756). Accuracy was chosen as the metric for model evaluation, due to the nature of the problem not requiring more detailed information (as there was no specific weight on false positive or false negative results justifying use of f1-value) (Korstanje, 2021).

3.3 Evaluation

The following section describes the final parameters used and performance for the models, including the choice of data normalization, feature selection, the algorithm's hyperparameters and resulting accuracy on the train-validate (as a part of K-fold cross validation) and test datasets.

The selection is the result of following the steps described in the Application section above.

3.3.1 Logistic regression

1. Features dropped: 'Gross', 'Mean Age Female', 'Number words male'
2. Normalisation: None.
3. Regularization: L2.
4. C = 0.1.
5. Solver: liblinear.
6. Max iteration: 100.
7. Train accuracy = 0.879
8. Test accuracy = 0.875

3.3.2 KNN

The following list contains all features of the best KNN model created in this project. The model uses standard normalization as opposed to Robust or Min-Max normalization techniques, which is reasonable because the distribution of

the data is likely Gaussian (if Central Limit theorem is to be followed in this case). It also, rather susprisingly, uses Manhattan distance instead of Euclidean distance. With a test accuracy of 0.78 the model has close to 1:3 error ratio, which is not a desirable outcome.

K Nearest Neighbors, being a powerful but quite simple algorithm, is not the best fit for the film data in question.

1. Features dropped: 'Mean Age Male', 'Age Lead', 'Number of words lead', 'Age Co-Lead'
2. Normalisation: Standard
3. Number of neighbors: 8
4. Algorithm: auto
5. Metric: Manhattan
6. Neighbor weights: uniform
7. Train accuracy = 0.866
8. Test accuracy = 0.78

3.3.3 Decision tree

1. Features dropped: 'Gross', 'Year', 'Mean Age Male', 'Age Co-Lead', 'Mean Age Female', 'Age Lead', 'Total words'
2. Normalisation: Standard
3. Max depth: 10
4. Criterion: Entropy
5. Train accuracy = 0.968
6. Test accuracy = 0.810

3.3.4 QDA

1. Features dropped: 'Number words male', 'Gross', 'Year'
2. Normalisation: Robust
3. Train accuracy = 0.909
4. Test accuracy = 0.899
5. Tuning hyper parameters with grid search did not improve the accuracy of QDA so the default parameters in the `scikit-learn` module was used in the final model.

3.3.5 LDA

1. Features dropped 'Total words' and 'Gross'
2. Normalisation: Standard
3. Train Accuracy: 0.874
4. Test Accuracy: 0.855
5. Best solver: svd

3.4 Model selection

The method that produced the best model based on accuracy scores on test and train datasets was Quadratic Linear Regression (QDA), stably performing better than the best models of the other methods on a variety of randomly shuffled subsets. The accuracy on the train dataset is around 90.9%, and accuracy on test dataset is around 89.9%. This indicates that there is no over-fitting, which also makes it a good choice.

4 Conclusions

The problem presented in this task was to learn a machine learning model to predict the gender of lead actors in movies. 5 different models were trained with the best one having an accuracy of 90%. It is surprising how poorly some models performed. The decision tree classifier and KNN ended up with 0.81 and 0.78 accuracy respectively, a performance only slightly better than the naive classifier with 0.756 accuracy. QDA consistently out-performed the other methods with around 0.9 in accuracy for both training and test data sets. It is a high number but still only 20% better than the naive classifier. This could mean that the problem is hard to solve with machine learning or that the strategy and methods used here were not the best to solve this task.

References

[1] Joyce, James, "Bayes' Theorem", The Stanford Encyclopedia of Philosophy (Spring 2019 Edition), Edward N. Zalta (ed.), plato.stanford.edu/archives/spr2019/entries/bayes-theorem/, Accessed 20 February 2023

[2] Pedregosa et al., Scikit-learn: Machine Learning in Python, (2011) scikit-learn.org/stable/modules/lda_qda.html, Accessed 20 February 2023

[3] Cunningham, Padraig Delany, Sarah. (2007). k-Nearest neighbour classifiers. Mult Classif Syst. 54. 10.1145/3459665.

[4] Zhang Z. Introduction to machine learning: k-nearest neighbors. Ann Transl Med. (2016) Jun;4(11):218. doi: 10.21037/atm.2016.03.37. PMID: 27386492; PMCID:

PMC4916348.

[5] Guo et al. (2003). KNN Model-Based Approach in Classification. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-39964-3_62

[6] Korstanje, Joos. (2021) The F1 score: All you need to know about the F1 score in machine learning. With an example applying the F1 score in Python. towardsdatascience.com/the-f1-score-bec2bbc38aa6. Accessed 20 February 2023

[7] Lindholm, Wahlström, Lindsten and Schön. (2022) Machine Learning - A First Course for Engineers and Scientists. Cambridge University Press. https://smlbook.org. Accessed 20 February 2023