**SML Report**

Introduction

There have been allegations against Hollywood for unrestrained sexism and racism in movie roles. Over the years, white men dominate movie roles -and there is now data to prove it. A motivated team [ref], gathered data from eight thousand (8,000) screenplays segregated into gender and age, then analyzed it to confirm the allegations. In this project, we set out to produce machine learning models that can predict the gender of the lead role using data such as year the movie was released, number of female actors, profits made by the film, number of words for each gender, etc. The algorithms used were Logistic Regression, Discriminant Analysis, K-nearest Neighbors, Random Forests (Tree based methods) and Boosting. The highest performing algorithm will be used to classify an unknown test set.

1. K-Nearest Neighbor (KNN)  
   The KNN algorithm tries to find only one value, *k*, which is number of nearest neighbors that has the least error on a given training set. Once a *k* value is found, given a test input, a majority vote is carried out for classification problems, and average for regression problems. KNN can be used for both regression and classification problems but performs better on classification problems. It is recommended to use an odd value for *k* in classification problems since a majority vote will always predict one class, in the case *k* is even, we can recalculate the distances and predict the class closest to k or simply pick one of the classes.

1.1 Feature selection  
An essential approach to tackling any supervised learning problem is feature engineering and selection. One way to remove noise from the data is to identify linear dependencies in the data and drop features that are highly dependent. To achieve this, we use correlation to figure out the linear dependencies and then proceed to prune the features.

A screenshot of a computer

Description automatically generated with low confidence

From the diagram above, the features “*Total words*”, “*Difference in words lead and co-lead*” and “*Number of words lead*” have high correlation so we decided to drop “Total words”. Leading to the features below

A screenshot of a computer

Description automatically generated with low confidence

Algorithm

1. Read features from file
2. Scale features with ***StandardScaler*** to make it easier for a model to learn, understand the problem and quickly learn from the data thereby achieving higher accuracy.
3. Create a train-test-split from the data with our selected features. The ***test\_size*** is set to 0.1 because we will run a 12-fold cross validation on the training set, so we picked a small size for the final test set.
4. To try out multiple parameters of the K-Nearest Neighbors algorithm we created a pipeline and applied grid search cross validation to run the algorithm for each parameter and return the best estimator model. We experimented with the various parameters:
   1. *k\_values* = range(1, 30): Values which k can take, for each value find the error and select the value of k with the least error
   2. *weights* = ['uniform', 'distance']: For each value of ***k***, apply “*uniform”* weights to all data points such that all points have the same influence on a test point; and “*distance*” weight which is closer points have higher influence to a test point.
   3. metrics = ['minkowski', 'manhattan', 'euclidean']: For each value of ***k*** and ***weights***, run the metrics used for the distance calculation between points, i.e. *minkowski,* 'manhattan', euclidean'. In essence try the different distance calculation methods, and return the best method that minimizes the error in the classification.

Each cross-validation fold use the same approach so the overall train time is increased.

Best estimator parameter for KNN

The parameters for the best estimator of the KNN we found is metric = “manhattan”, “n\_neighbors” = “14” and weights = “uniform”. That is, using the Manhattan distance for distance calculation with 14 neighbors and uniform weight

Logistic Regression

Logistic regression is quite like linear regression but makes use of an activation function which makes it a classifier. In this type of regression, we find the value of z =