Title: To explore gender biases in film scenes using machine learning

Cliona O'Doherty (15315679) & Hannah Craddock (20324412)

Repository link: https://github.com/hanmacrad2/Movie_gender_classifier_freq_pattern_mining

Introduction

Film has a definite and quantifiable gender bias. The Bechdel-test is a well-known measure of the representation of women in fiction, and Google have even developed a machine learning metric of gender representation within feature length films: the Geena Davis Inclusion Quotient (GD-IQ). Here, we explore what objects co-occur in a movie scene with males and females, and whether these serve to perpetuate gender stereotypes. Using a dataset of automatically labelled movies, we test if the labels that are most associated with each gender can be used as predictive features for that gender in a linear model. It should be noted that the dataset used here was itself generated by a deep learning automatic labelling service, Amazon Rekognition. Thus, our analyses and conclusions are not solely insightful for the question of stereotypes in movies, but may also tell us about the in-built biases present in this large-scale commercial deep learning service. For example, it's possible that the labels learned by Rekognition and returned during object detection are themselves biased and we are picking up on this trend instead of strong biases in the movies themselves. Either way, by using labels of objects from movies as inputs to frequent pattern mining and linear models, we ask interesting questions about disparity in commercial gender representations.

In this project, frequent pattern mining was firstly used to determine the features that co-occur most with a given gender. Specifically the algorithm FPgrowth was used. Classifier models, logistic regression and SVM, were then trained on the data to predict the presence male or female. The outputs presented here include the most associated labels with each gender and the binary classification performance of linear models for each gender.

Dataset and Features

The raw data were obtained from a dataset collected by a research assistant in the Cusack Lab at Trinity. 158.4 hr of live-action movies were input to Amazon Web Services' cloud-based computer

(a) (b)

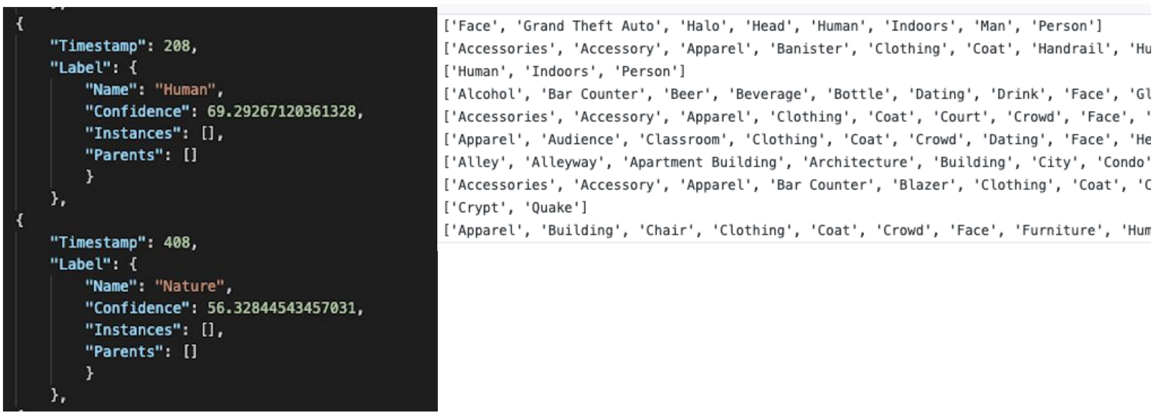

Figure 1: (a) An example of the raw data returned by Amazon Rekognition. (b) The preprocessed data, with each array corresponding to a timepoint in the videos.

vision software, Rekognition. This software returned labels of the objects that were detected in the movie frames, at a sampling rate of every 200 ms of video processed. The labels were returned in json format as shown in Fig. 1a.

Next, the raw labels were preprocessed. Using the Python script create_itemsets.py, all labels were grouped into a list of lists, each corresponding to a particular 200 ms window of the videos. The output of this script, shown in Fig. 1b, was the main dataset used for subsequent analyses. It consisted of 2,871,272 timepoints with a total of 41,330,953 labels of which 2,466 were unique. The timepoints that contained male and female labels were found. For females, the label returned by Amazon Rekognition was 'Female', whereas there was no 'Male' counterpart for males. Instead, 'Man' was used to denote the presence of a male in the scene. The label dataset was split into those timepoints containing 'Man' (n = 610,639) and 'Female' (n = 593,261). From this, features were generated using an association analysis.

Feature Engineering

Frequent pattern mining was run to determine which items co-occur most frequently with a given gender. The algorithm FPGrowth was used, the output of which is a list of lists. These frequent items were then used as the input features to the classifier models. The features were one-hot encoded using a TransactionEncoder(), to be of the correct form for the subsequent classifier models. The function *get_features* was developed to execute the described steps.

Methods

Using the features from the frequent pattern mining, classifier models were trained on the data to predict the presence of a given gender. These included logistic regression and SVM tested against baseline models.

Frequent Pattern mining - FPgrowth

Frequent pattern mining is a rule-based, unsupervised machine learning method used to find frequent patterns between variables in large datasets. The algorithm returns frequent itemsets which are items or groups of items that occur with some frequency ('support') in the dataset. The algorithms apriori and FPGrowth were tested (See Experimental Notebook in Github). However, FPGrowth is optimised to work on larger datasets and has a significantly faster computational time. Apriori can be computationally expensive, mainly because candidate sets need to be repeatedly calculated, e.g. for 10^4 frequent items, 10^7 candidate sets would be created and the original dataset needs to be repeatedly scanned. However, in FPGrowth the algorithm first compresses the data in a tree form where each node is an item and each branch is the association of the items. Then from the tree, a conditional pattern base is created from which the frequent patterns are found.

Logistic Regression

Logistic regression is a supervised machine learning algorithm that is used for classification problems. In contrast to the continuous predictions of a linear regression, the dependent variable of the logistic regression is categorical, making it suitable for classification. The model is sign($\theta^T x$) such that +1 is predicted when $\theta^T x > 0$ and -1 when $\theta^T x < 0$ (if the decision boundary has been defined as x = 0). In this way, input feature vectors are assigned a predicted label for a class depending on which side of the decision boundary they lie on. The cost function used for logistic regression is shown in Eqn. 1, where y is equal to either 1 or -1, and J(θ) is solved using the classic gradient descent. A penalty term can be included to regularise the model. Although common practice, this is optional.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} log(1 + e^{-y^{(i)}\theta^T x^{(i)}}) + \frac{\theta^T\theta}{C} \qquad \textit{Equation 1}$$

Support vector machine

Support-vector machines (SVMs) are supervised learning models. The objective of the algorithm is to find a hyperplane in an N-dimensional space (N = number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. The objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. In an SVM the 'hinge' loss function is used, which is non-differentiable, i.e non-smooth and is given by max(0, 1 - yθ_T x). Large values of θ are penalised by the inclusion of a penalty term in the cost function. The weighting parameter C allows the influence of the penalty term to be controlled. Larger values of C makes the penalty less important. The final SVM cost function is as follows:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} max(0, 1 - y^{(i)}\theta^T x^{(i)}) + \frac{\theta^T\theta}{C} \qquad \textit{Equation 2}$$

Results

Frequent pattern mining results

The top twenty frequent items that co-occurred with 'Female' and 'Man' are shown in Table 1. It was found that while the top few labels with the most support for each gender were similar, stereotypes began to emerge later down the list. For example, 'Girl', 'Kid' and 'Child' all appear very frequently with 'Female', but the 'Boy' counterpart is not strongly associated with 'Man'. Interestingly, the support values remain higher for the female-associated items than for male. This suggests that there are more typical and strong connections made when displaying women in movies than men. Reiterating this, the inputs to the frequent pattern mining were preprocessed such that labels that always co-occurred with 'Female' or 'Man' were not included. The redundant labels found using the function *check_redundant* already revealed differences is male and female portrayal. While both included labels such as 'Head', 'Person', 'Face', 'Human' and 'Indoors', the female label also appeared every time 'Smile' and 'Selfie' were present, whereas 'Man' was always present alongside 'Halo' and 'Grand Theft Auto'. It is likely that this reveals more about the nature of the Rekognition object detection than the movies themselves. For example, the Grand Theft Auto label is likely something spurious that has been learned during Rekognition's training. Indeed, when manually cross-checking the timepoints for these labels to the movies it was often found linked to men of colour in any sort of motor vehicle - a clear sign of the worrying bias and discrimination that can be built into any machine learning system trained on unfair datasets.

Table 1. Frequent Pattern Mining Results showing the top 20 most associated labels to 'Female' and to 'Man'. Note that many more labels were used as input features to the model than those shown here.

Female associations		**Male associations**	
Object label	**Support**	**Object label**	**Support**
Clothing	0.84435013	Apparel	0.74859451
Apparel	0.84435013	Clothing	0.74859451
Girl	0.35933088	Coat	0.40390149
Hair	0.35674349	Photo	0.32864917
Photo	0.2740497	Photography	0.32864917
Photography	0.2740497	Overcoat	0.31030118
Coat	0.25177114	Room	0.2873711

Room	0.24646151	Accessories	0.25464145
Furniture	0.23730196	Accessory	0.25464145
People	0.23729185	Suit	0.23948356
Kid	0.20874623	Portrait	0.2366259
Child	0.20874623	Hair	0.19369546
Fashion	0.20531773	Crowd	0.18197986
Portrait	0.20494184	Jacket	0.18054202
Teen	0.20325455	People	0.17764014
Crowd	0.17901733	Furniture	0.15948703
Robe	0.17883528	Tie	0.15814581
Accessory	0.1720103	Pub	0.14900293
Accessories	0.17200861	Performer	0.14755363
Gown	0.17063653	Bar Counter	0.14051346

Hyperparameters – Choice of C in Logistic Regression & SVM via Cross Validation

The optimal value of the penalty term C in both the logistic regression model and the SVM classifier was determined in the range [0.001, 0.01, 1, 10, 30, 50, 100, 500, 1000]. Decreasing C gives the penalty more importance. The optimal value was determined based on the value of C that corresponded to the greatest F1 score as shown in Figure 2. The F1 score is the harmonic mean of the precision and the recall and so is a good overall metric for comparison. The plots were obtained using the function *choose_c_cv* and *choose_c_svm_cv* and 5-fold cross validation was used to train and test the model. For the logistic regression model, no gain in performance was achieved for varying the penalty term C, the F1 score was 0.66 across all values of C. Thus for the final model, the penalty term was left at the default value of 1. In the case of the SVM classifier, the F1 score decreased significantly as the penalty term C was increased. Again, the default value of 1 resulted in the optimal F1 score.

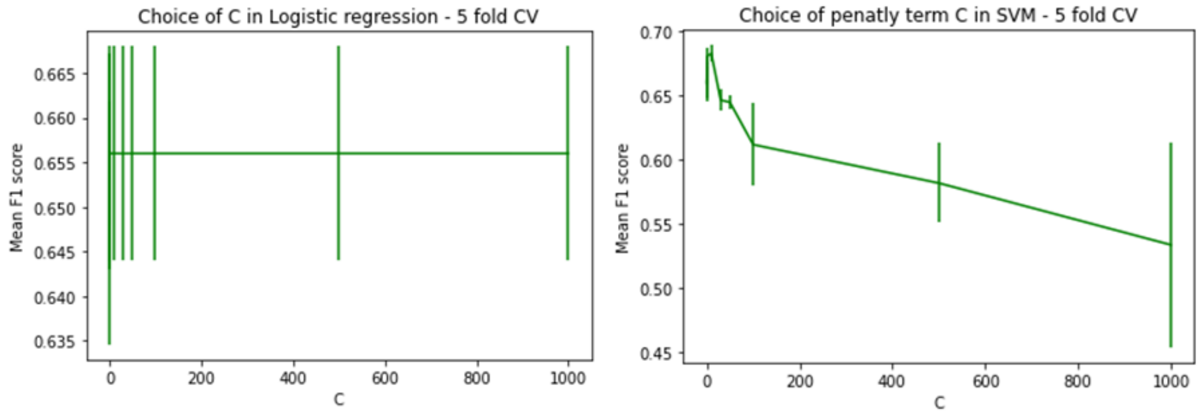

Figure 2: Penalty term C in Logistic Regression (left) and SVM (right) vs corresponding F1 score. Model results obtained using 5-fold cross validation. 1.0 was chosen for both models.

Model Training

Once the hyperparameters were determined (the default value of C in both cases) for both the logistic regression model and the SVM model, the final models were trained. A dummy classifier which predicts the most common class for all observations was used as a baseline model. The most common class in the dataset is the negative class (No presence of a given gender). To train and test the models, a train size of 0.67 and a test size of 0.33 was chosen.

Performance metrics

The primary metrics used in evaluating the performance of a classifier model were precision, recall, accuracy, the f1 score and the AUC (Area under the Curve). They can be described as follows:

- Precision - The precision, also known as the positive predictive value, is the fraction of correctly classified observations amongst all classified observations. The formula for which is TP/(TP + FP).
- Recall - The recall is the fraction of the total amount of actual positive cases that were actually retrieved or identified correctly. In binary classification, recall of the positive class is also known as 'sensitivity', while recall of the negative class is 'specificity'.
- F1 score - The F1 score is the harmonic mean of the precision and recall.
- Accuracy - The accuracy is the fraction of correct predictions made by the model. It is the number of correct predictions made by model as a fraction of the total number of predictions
- The AUC - The AUC is the area under the Receiver Operator Curve. The AUC ranges in value from 0 to 1. A perfect model is able to discriminate between two classes with 100% sensitivity and 100 % specificity and would have an AUC of 1.0.

  The confusion matrices for the female and male models are shown in Tables 2 and 3 respectively. The performance metrics are shown below in Table 4 and the ROC plots in Figures 3-5.

Confusion Matrices

Table 2: Female model confusion matrices

Logistic Regression			**SVM**			**Baseline model**		
	Predicted			**Predicted**			**Predicted**	
True	1 (Female)	0	**True**	1 (Female)	0	**True**	1 (Female)	0
1 (Female)	725466	19579	1 (Female)	730501	14544	1 (Female)	745045	0
0	82200	113675	0	87831	108044	0	195875	0

Table 3: Male model Confusion matrices

Logistic Regression			**SVM**			**Baseline model**		
	Predicted			**Predicted**			**Predicted**	
True	1 (Man)	0	**True**	1 (Man)	0	**True**	1 (Man)	0
1 (Man)	701652	38058	1 (Man)	706648	33062	1 (Man)	739710	0
0	113795	87415	0	119627	81583	0	201210	0

Classification Report

Table 4: Performance metrics for both Male and Female models.

Gender	Model	Precision	Recall	F1 score	Accuracy	AUC
Female	Logistic Regression	0.90 False, 0.85 True	0.97 False, 0.58 True	0.93 False, 0.69 True	0.89	0.91
	SVM	0.89 False, 0.88 True	0.98 False, 0.55 True	0.93 False, 0.68 True	0.89	0.91
	Dummy (most freq)	0.79 False, 0.00 True	1.00 False, 0.00 True	0.88 False, 0.00 True	0.79	0.5
Male	Logistic Regression	0.86 False, 0.70 True	0.95 False, 0.43 True	0.90 False, 0.54 True	0.84	0.85
	SVM	0.86 False, 0.71 True	0.96 False, 0.41 True	0.90 False, 0.52 True	0.84	0.85
	Dummy (most freq)	0.79 False, 0.00 True	1.00 False, 0.00 True	0.88 False, 0.00 True	0.79	0.5

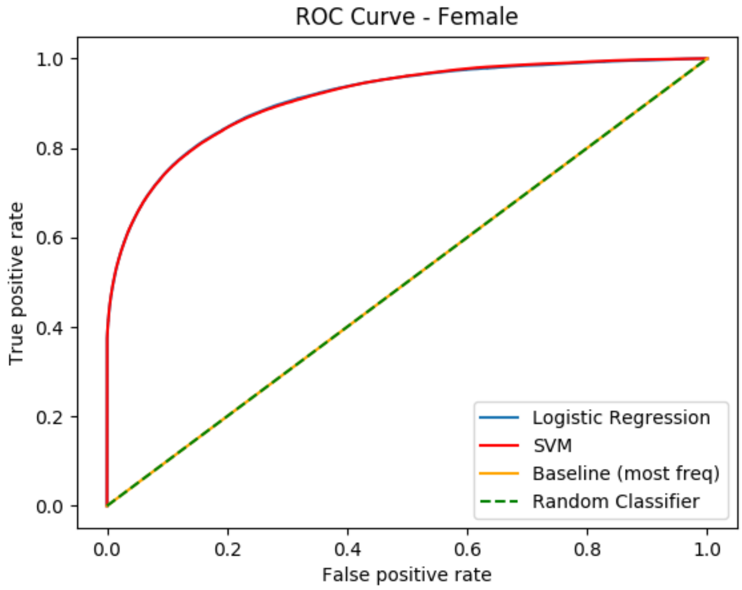

Figure 3: ROC for the female models. Both the SVM and logistic regression performed better than baseline, with equal performance for both linear models.

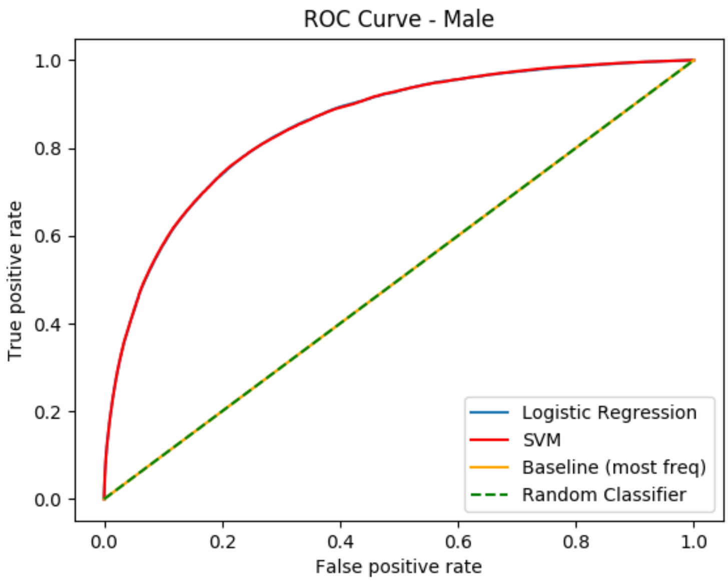

Figure 4: ROC for Male models. Both SVM and logistic regression performed better than baseline, and each linear model had equal performance.

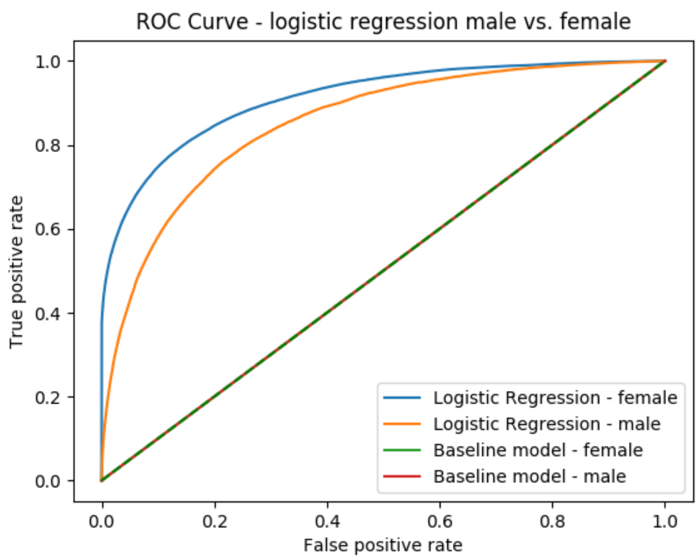

Figure 5: Logistic regression results for both male and female models. It was found that the female model reached higher performance.

Discussion

For both genders, very similar results were obtained from the Logistic Regression and SVM models. Both models performed better than a dummy classifier baseline which naively picked the most common class. Overall, the logistic regression model was considered to be the best performing model. Its training time was significantly faster than the SVM, albeit with an equally high performance, and so it was deemed the classifier of choice.

Both the logistic regression classifier and the SVM classifier performed well when their ROC curves are considered (Fig. 3 and 4). A perfect classifier (100% true positives, 0% false positives) would produce a point in the top-left corner of the ROC plot and the ROC curves of both models are reasonably close to this point. Corresponding to this, the AUC of both models is high, 0.91 for the female features and 0.85 for the male features for both models, which is reasonably close to a perfect classifier AUC of 1.

Both models perform significantly better than the random and baseline classifiers, i.e at the optimal threshold, their True Positive Rate is higher and their False Positive rate is lower than the baseline models. The latter models could be considered 'no-skill' classifiers in that they cannot discriminate between classes and predict a random class in the case of the former and a constant class (the majority class) for all observations in the case of the latter. Their ROC Curves thus lie on the 45 degree line, which can be interpreted as the 'flip a coin' line, meaning the model is no better than flipping a coin to categorise a binary response. As a result, this model produces an AUC of 0.5.

These results also correspond with the confusion matrices in Tables 2 and 3. It can be quickly observed that the logistic regression model and the SVM model are performing well, given that the diagonal entries are both high, showing the models' abilities to detect the true positives and true negatives. Conversely, the off-diagonals are lower which shows that there is a low number of false positives and false negatives.

A classification report was generated using sklearn which outputs the precision, recall, f1 score and accuracy as shown in Table 4. Across all models the precision is significantly higher than the recall for the positive class; the recall of the positive class is also known as the sensitivity. The majority of predictions the models make regarding the presence of a female or male are correct; the model is precise however many instances of female or male go undetected and so the recall is low. Across the two classifiers the precision is in the range [0.7- 0.9], while the sensitivity is in the range [0.41 - 0.58]. This could be due to the fact that for some instances of a given gender in an itemset, very few explanatory features are present and so the instances are misclassified as not having a gender present. The F1 score is the harmonic mean of the precision and recall and so balances out the results of the two metrics. It's results are in the range [0.52 - 0.69] for the positive class and [0.9-0.92] for the negative class. The dummy classifier predicts the majority class for all cases, i.e no presence of a given gender. Thus it performs well on the negative class, with an F1 score of 0.88 however it is unable to detect the presence of the positive class (gender) and so the F1 score is 0.0 for the positive class.

Summary

Here we used machine learning to explore the patterns of co-occurrences of objects in a large movie dataset and tested whether certain objects can be a useful predictive cue for the presence of males and females. It was found that the most strongly associated objects with the labels 'Man' and 'Female' did reflect some societal gender stereotypes, for example, that women are always present with children (Table 1). This association analysis also revealed biases in the commercial object detection service used to generate these labels, and highlights a key ethical issue with wide scale deployment of these methods trained on inherently biased datasets.

It was found that the female logistic regression classifier performed the best. Perhaps this higher accuracy was due to the stronger links of females in movie scenes with their 'typical' objects and attributes, enabling higher classification performance. An additional experiment tested whether using the male features would affect the results of predicting a female label and vice versa, however no obvious trends were detected from the results.

To conclude, this exploratory machine learning experiment for gender biases in movies has revealed some interesting trends, and future work should improve and develop upon the methods described in this brief report to make further progress towards equality and accurate representations in media.

Code Overview

Repository:
https://github.com/hanmacrad2/Movie_gender_classifier_freq_pattern_mining/blob/main/Experiment_with_Frequent_Pattern_Mining.ipynb

Main script:
https://github.com/hanmacrad2/Movie_gender_classifier_freq_pattern_mining/blob/main/gender_modelling.py (Hannah and Cliona)

Obtaining data:
https://github.com/hanmacrad2/Movie_gender_classifier_freq_pattern_mining/blob/main/create_itemsets.py (Cliona)

Experimenting with Frequent Pattern Mining (apriori & fp growth) -

https://github.com/hanmacrad2/Movie_gender_classifier_freq_pattern_mining/blob/main/Experiment_with_Frequent_Pattern_Mining.ipynb (Hannah)

Contributions

Cliona O'Doherty

- Report
    - Introduction
    - Dataset and features (minus feature engineering)
    - Methods - logistic regression
    - Results - frequent pattern mining results
    - Summary
- Code
    - Data preprocessing, create itemsets.py
    - In gender_modelling.py
        i. check_redundant()
        ii. Extending the analyses to a second gender
        iii. ROC plots

Hannah Craddock

- Report
    - Feature engineering in dataset and features
    - Methods - frequent pattern mining, support vector machine
    - Results - cross validation, model training, performance metrics
    - Discussion
- Code
    - In gender_modelling.py
        i. Initial setup and writing of the script
        ii. get_df_items(), get_fp_gender(), get_features()

        iii. Cross validation for all linear models

    - Experiment_with_Frequent_Pattern_Mining.ipynb

We are happy that each member of the team made an equal contribution to the production of this project report.

Electronic initials

COD

HC

Appendix

Main script: gender_modelling.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 14 20:51:30 2020
@author: Hannah Craddock, Cliona O'Doherty
"""

#Imports
import numpy as np
import pandas as pd
import itertools
import matplotlib.pyplot as plt

from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import (apriori,
                                       association_rules)
from collections import Counter

from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, confusion_matrix,
classification_report, f1_score, roc_curve, auc
from sklearn.dummy import DummyClassifier
from sklearn.svm import LinearSVC

#***********************************************
#i. Data
itemsets = pd.read_pickle("./itemsets.pickle")
```

```
#Inspect items

list_itemsets = [inner for outer in itemsets for inner in outer]

#count_items = Counter(list_itemsets)

#count_items.most_common()


#*****************************************************************

#Frequent Pattern Mining


#i. One hot encode data (list of lists)

def get_df_items(itemsets):

    '''Creates a one-hot encoded dataframe of the given itemsets'''

    transaction_encoder = TransactionEncoder()

    transaction_encoded_ary =
transaction_encoder.fit(itemsets).transform(itemsets)

    #Dataframe

    df = pd.DataFrame(transaction_encoded_ary, columns=
transaction_encoder.columns_)

    return df


#ii. Frequent Pattern Mining

def get_fp_gender(itemsets, gender, redundant_labels):

    '''Get model input X and y for a given gender.  '''


    #i. Extract the itemsets for which the given gender is present

    itemsets_genderI = [inner_items for inner_items in itemsets if gender
in inner_items]


    #ii. Remove redundant labels

    itemsets_gender = [[item for item in inner_items if item not in
redundant_labels] for inner_items in itemsets_genderI]
```

```
    #iii. Create ohe dataframe of items that contain the gender

    df_gender = get_df_items(itemsets_gender)


    #iii. Frequent Patterns - FP Growth

    df_fp = fpgrowth(df_gender, min_support= 0.01, max_len = 1,
use_colnames=True)


    #iv.Extract strings from frozen sets

    df_fp["itemsets_strings"] = df_fp["itemsets"].apply(lambda x: ',
'.join(list(x))).astype("unicode")


    return df_fp


def get_features(df_fp, df_ohe, gender):

    '''Extract features X and y from dataframe of frequent itemsets. Only
extract features that have occured with female or male'''

    #i.Get list of frequent items

    list_gender_freq_items = list(df_fp["itemsets_strings"])


    #ii. Extract from main one hot encoded dataframe

    df_ohe_gender_frs = df_ohe.loc[:, list_gender_freq_items]


    #Data

    X = df_ohe_gender_frs.drop([gender], axis=1)


    #Extract X, y

    y = df_ohe_gender_frs[gender]


    return X, y


def check_redundant(l, ref):

    """Find the elements of l which always occur with word ref"""
```

```
    #i. Condense l to only those elements containing ref

    l = [i for i in l if ref in i]


    #ii. Define valid which checks whether two words in an element of l
always occur together

    def valid(p):

        for s in l:

            if any(e in s for e in p) and not all(e in s for e in p):

                return False

            return True


    #iii. Find unique words

    elements = list(set(b for a in l for b in a))


    #iv. Check all pairs of combinations and store in a list to return

    pairs = []

    for c in itertools.combinations(elements, 2):

        if ref in c:

            if valid(c):

                pairs.append(c)


    pairs = list(set(b for a in pairs for b in a))

    pairs.remove(ref)


    return pairs


#Apply - get ohe dataframe of itemsets

df_ohe = get_df_items(itemsets)


#Get female features
```

```
#use check_redundant to get all labels that when they appear, always appear with 'Female'

#redundant_labels_f = check_redundant(itemsets, 'Female')


#Manually elect the synonymous or uninformative labels that occur redundantly with 'Female' (from check_redundant)

redundant_labels_f = ['Person','Face','Woman','Human','Indoors','Head']

df_fp_f = get_fp_gender(itemsets, 'Female', redundant_labels_f)


X_f, y_f = get_features(df_fp_f, df_ohe, 'Female')


#Get male features

#redundant_labels_m = check_redundant(itemsets, 'Man')

redundant_labels_m = ['Person','Face','Human','Indoors','Head']

df_fp_m = get_fp_gender(itemsets, 'Man', redundant_labels_m)


X_m, y_m = get_features(df_fp_m, df_ohe, 'Man')


# Get train test splits for each gender

testSizeX = 0.33 #67:33 split

Xtrain_f, Xtest_f, ytrain_f, ytest_f = train_test_split(X_f, y_f, test_size= testSizeX, random_state=42)

Xtrain_m, Xtest_m, ytrain_m, ytest_m = train_test_split(X_m, y_m, test_size= testSizeX, random_state=42)


#**************************************************************

#Modelling


#*****************************************************************************************************

#Model 1 - Logistic Regression
```

```
#i. Choose c
def choose_C_cv(X, y, c_range, plot_color):
    '''Implement 5 fold cross validation for testing
    regression model (lasso or ridge) and plot results'''

    #Param setup
    kf = KFold(n_splits = 5)
    mean_f1 =[]; std_f1 =[]

    #Loop through each k fold
    for c_param in c_range:
        print('C = {}'.format(c_param))
        count = 0; f1_temp = []
        model = LogisticRegression(penalty= 'l2', C = c_param)

        for train_index, test_index in kf.split(X):
            count = count + 1
            print('count kf = {}'.format(count))
            model.fit(X.iloc[list(train_index)], y[train_index])
            ypred = model.predict(X.iloc[list(test_index)])
            f1X = f1_score(y[test_index],ypred)
            #mse = mean_squared_error(y[test_index],ypred)
            f1_temp.append(f1X)

        #Get mean & variance
        mean_f1.append(np.array(f1_temp).mean())
        std_f1.append(np.array(f1_temp).std())

    #Plot
```

```
    plt.errorbar(c_range, mean_f1, yerr=std_f1, color = plot_color)
    plt.xlabel('C')
    plt.ylabel('Mean F1 score')
    plt.title('Choice of C in Logistic regression - 5 fold CV')
    plt.show()

#Implement
c_range = [0.001, 0.01, 1, 10, 30, 50, 100, 500, 1000]
plot_color = 'g'

#i. get female cv results
#choose_C_cv(X_f, y_f, c_range, plot_color)
#ii. get male cv results
#choose_C_cv(X_m, y_m, c_range, plot_color)

#Final model (use default penalty term - no performance improvement for
varying penalty)

def run_logistic(Xtrain, Xtest, ytrain, ytest):
    log_reg_model = LogisticRegression(penalty= 'l2')
    log_reg_model.fit(Xtrain, ytrain)

    #log_reg_model.intercept_
    #log_reg_model.coef_
    #Predictions
    predictions = log_reg_model.predict(Xtest)

    #Performance
    print(confusion_matrix(ytest, predictions))
    print(classification_report(ytest, predictions))
```

```
    #Auc
    scores = log_reg_model.predict_proba(Xtest)
    fpr, tpr, _= roc_curve(ytest, scores[:, 1])
    print('AUC = {}'.format(auc(fpr, tpr)))

    return log_reg_model

# Run the logistic regression model
# i. Use the matching gender's features
log_reg_model_f = run_logistic(Xtrain_f, Xtest_f, ytrain_f, ytest_f)
log_reg_model_m = run_logistic(Xtrain_m, Xtest_m, ytrain_m, ytest_m)

#***************************************************************************
******************************
#SVM
def choose_C_SVM_cv(X, y, c_range, plot_color):
    '''Implement 5 fold cross validation for testing
    regression model (lasso or ridge) and plot results'''

    #Param setup
    kf = KFold(n_splits = 5)
    mean_f1 =[]; std_f1 =[]

    #Loop through each k fold
    for c_param in c_range:
        print('C = {}'.format(c_param))
        count = 0; f1_temp = []
        model = LinearSVC(C = c_param)

        for train_index, test_index in kf.split(X):
            count = count + 1
```

```
            print('count kf = {}'.format(count))
            model.fit(X.iloc[list(train_index)], y[train_index])
            ypred = model.predict(X.iloc[list(test_index)])
            f1X = f1_score(y[test_index],ypred)
            f1_temp.append(f1X)

        #Get mean & variance
        mean_f1.append(np.array(f1_temp).mean())
        std_f1.append(np.array(f1_temp).std())

    #Plot
    plt.errorbar(c_range, mean_f1, yerr=std_f1, color = plot_color)
    plt.xlabel('C')
    plt.ylabel('Mean F1 score')
    plt.title('Choice of penatly term C in SVM - 5 fold CV')
    plt.show()

#Implement
c_range = [0.001, 0.01, 1, 10, 100]
plot_color = 'g'

#i. female cv results
#choose_C_SVM_cv(X_f, y_f, c_range, plot_color)
#i. male cv results
#choose_C_SVM_cv(X_m, y_m, c_range, plot_color)

def run_svm(Xtrain, Xtest, ytrain, ytest, c_param=1.0):
    svm_model = LinearSVC(C = c_param)
    svm_model.fit(Xtrain, ytrain)
```

```
    #log_reg_model.intercept_
    #log_reg_model.coef_
    #Predictions
    predictions = svm_model.predict(Xtest)

    #Performance
    print(confusion_matrix(ytest, predictions))
    print(classification_report(ytest, predictions))

    #Auc
    scores = svm_model.decision_function(Xtest)
    fpr, tpr, _= roc_curve(ytest, scores)
    print('AUC = {}'.format(auc(fpr, tpr)))

    return svm_model

# Run svm model

#Use C=1.0 (default) as per cross val results
# i. Use the matching gender's features
#svm_model_f = run_svm(Xtrain_f, Xtest_f, ytrain_f, ytest_f)
#svm_model_m = run_svm(Xtrain_m, Xtest_m, ytrain_m, ytest_m)

#*******************************************
#3. Baseline model
def run_dummy(Xtrain, Xtest, ytrain, ytest):
    dummy_clf = DummyClassifier(strategy="most_frequent")
    dummy_clf.fit(Xtrain, ytrain)
    predictions_dummy = dummy_clf.predict(Xtest)
```

```
    #Evaluation
    print(confusion_matrix(ytest, predictions_dummy))
    print(classification_report(ytest, predictions_dummy))

    #Auc
    scores_bl = dummy_clf.predict_proba(Xtest)
    fpr, tpr, _= roc_curve(ytest, scores_bl[:, 1])
    print('AUC = {}'.format(auc(fpr, tpr)))

    return dummy_clf

# i. Use the matching gender's features
dummy_clf_f = run_dummy(Xtrain_f, Xtest_f, ytrain_f, ytest_f)
dummy_clf_m = run_dummy(Xtrain_m, Xtest_m, ytrain_m, ytest_m)

#**************************************************
#Compare performance - ROC curve

def plot_roc_models(Xtest, ytest, log_reg_model, svm_model, dummy_clf,
gender=''):
    'Plot ROC Curve of implemented models'

    #Logistic Regression model
    scores = log_reg_model.decision_function(Xtest)
    fpr, tpr, _= roc_curve(ytest, scores)
    plt.plot(fpr,tpr, label = 'Logistic Regression')
    print('AUC = {}'.format(auc(fpr, tpr)))

    #svm model
    scores = svm_model.decision_function(Xtest)
    fpr, tpr, _= roc_curve(ytest, scores)
```

```
    plt.plot(fpr,tpr, color = 'r', label = 'svm')

    print('AUC = {}'.format(auc(fpr, tpr)))


    #Baseline Model

    scores_bl = dummy_clf.predict_proba(Xtest)

    fpr, tpr, _= roc_curve(ytest, scores_bl[:, 1])

    plt.plot(fpr,tpr, color = 'orange', label = 'baseline model')

    print('AUC = {}'.format(auc(fpr, tpr)))


    #Random Choice

    plt.plot([0, 1], [0, 1],'g--')


    #Labels

    plt.xlabel('False positive rate')

    plt.ylabel('True positive rate')

    plt.title('ROC Curve - {}'.format(gender))


    plt.legend(['Logistic Regression', 'SVM', 'Baseline (most
freq)','Random Classifier'])

    plt.savefig('./roc_{}'.format(gender))

    plt.show()

    plt.close()


#Implement

plot_roc_models(Xtest_f, ytest_f, svm_model_f, log_reg_model_f,
dummy_clf_f, gender='Female')

plot_roc_models(Xtest_m, ytest_m, svm_model_m, log_reg_model_m,
dummy_clf_m, gender='Male')


#****************

#Compare ROC curves

#****************
```

```
#Logistic Regression model - matched features
scores = log_reg_model_f.decision_function(Xtest_f)
fpr, tpr, _= roc_curve(ytest_f, scores)
plt.plot(fpr,tpr, label = 'Logistic Regression - female')
print('AUC = {}'.format(auc(fpr, tpr)))

#Logistic Regression model - crossed features
scores = log_reg_model_m.decision_function(Xtest_m)
fpr, tpr, _= roc_curve(ytest_m, scores)
plt.plot(fpr,tpr, label = 'Logistic Regression - male')
print('AUC = {}'.format(auc(fpr, tpr)))

#Baseline Model
scores_bl = dummy_clf_f.predict_proba(Xtest_f)
fpr, tpr, _= roc_curve(ytest_f, scores_bl[:, 1])
plt.plot(fpr,tpr, label = 'Baseline model - female')
print('AUC = {}'.format(auc(fpr, tpr)))

#Baseline Model
scores_bl = dummy_clf_m.predict_proba(Xtest_m)
fpr, tpr, _= roc_curve(ytest_m, scores_bl[:, 1])
plt.plot(fpr,tpr, label = 'Baseline model - male')
print('AUC = {}'.format(auc(fpr, tpr)))

#Random Choice
plt.plot([0, 1], [0, 1],'g--')

#Labels
plt.xlabel('False positive rate')
```

```
plt.ylabel('True positive rate')
plt.title('ROC Curve - logistic regression male vs. female')


plt.legend()
plt.savefig('./roc_comp')
plt.show()
plt.close()
```