Examining the Impact of Movie Ratings and College Majors

Introduction

Resources:

- https://www.datacamp.com/tutorial/random-forests-classifier-python
- https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
- INFO2950 FA23 HW4: Normalizer() function definition
- F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872
- https://stackoverflow.com/questions/5552555/unicodedecodeerror-invalid-continuation-byte
- https://stackoverflow.com/questions/57958432/how-to-add-table-title-in-python-preferably-with-pandas
- https://www.geeksforgeeks.org/concatenate-strings-from-several-rows-using-pandas-groupby/
- https://stackoverflow.com/questions/61688906/column-in-pandas-series-is-appearing-in-a-row-above-the-rest
- https://www.geeksforgeeks.org/convert-list-like-column-elements-to-separate-rows-in-pandas/
- https://www.geeksforgeeks.org/plot-a-horizontal-line-in-matplotlib/
- https://www.statology.org/cannot-mask-with-non-boolean-array-containing-na-nan-values/
- https://www.tutorialspoint.com/how-to-avoid-overlapping-of-labels-and-autopct-in-a-matplotlib-pie---chart#:~:text=Use%20pie()%20method%20to,figure%2C%20use%20show()%20method.
- https://datascienceparichay.com/article/matplotlib-label-points-on-scatter-plot/
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

About the Two Main Data Sources:

- GroupLens.org:
  - Where: https://grouplens.org/datasets/movielens/ (under ml-latest)
  - Who: The creators of the MovieLens dataframes are a group called GroupLens which is composed of researchers at the University of Minnesota from various fields (computer science, psychology, sociology, etc.) whose mission statement is "We advance the theory and practice of social computing by building and understanding systems used by real people." The data is

collected from a website called MovieLens which offers personalized movie recommendations based off of user provided recommendations.

- What: As described by the creators of the dataset: "approximately 33,000,000 ratings and 2,000,000 tag applications applied to 86,000 movies by 330,975 users. Includes tag genome data with 14 million relevance scores across 1,100 tags. Last updated 9/2018."
    - Rows: The dataset is split into multiple different dataframes, but when compiled each row represents a movie rating created by an individual on the MovieLens platform.
    - Columns: When the individual dataframes are joined on common columns there are 12 columns: movieId, relevance, tag, imdbId, tmdbId, title, genres, userId, tag, timestamp, rating, timestamp.
- Why: There was no stated purpose released by the vreators of the dataframe. We used this dataset because out of all of the movie datasets we could find it was applicable to our research questions due to is wide variety of movies and ratings from individuals.
- Potential Impacts: the data points come from users creating it, so movies that are less popular among users of the Movie Lens recommendation system might not be reflected in this dataset. The dataset also only contains information and ratings from users who have reviewed more than 20 movies, so movies that are more preferred by less-frequent movie-watchers might also not be represented. Also it is ambiguous as to whether or not users were alerted to the knowledge that their data was going to used for datasets.

- FiveThirtyEight/The Economic Way to Pick a Major:
    - Where: https://github.com/fivethirtyeight/data/tree/master/college-majors
    - Who: The data collection was done by the FiveThirtyEight organization which is part of ABC News. FiveThirtyEight collects data through opinion polls. The data was collected for an article by Ben Casselman titled "The Economic Guide to Picking a College Major". The data was collected through the American Community Survey 2010-2012 Public Use Microdata Series.
    - What: The article states that the purpose of the dataset is to help students pick a major that would place them above the lowest 25% of college grads who earn less than high school grads that did not go to college.
        - Rows: Each row represents a college major.
        - Columns: There are a total of 22 columns that represent the following: Major, Rank, Major_code, Major, Major_category, Total, Sample_size, Men, Women, ShareWomen, Employed, Full_time, Part_time, Full_time_year_round, Unemployed, Unemployment_rate, Median, P25th, P75th, College_jobs, Non_college_jobs, Low_wage_jobs.
    - Why: The dataframe was originally created to help high school students decide on what major to pick using statistics such as employment rate post graduation from college.
    - Potential Impacts: Increase in students picking majors that have higher payoff based off of graduate employment rates and earnings.

Research Questions

1. How does choice of graduate school major and professional field reflect trends in Netflix movie streaming, based on ratings of different genres?
2. Is there a strong enough association to suggest a potential influence of academia portrayed on media while students choose their field of study/career?
3. Does the number of newly released movies pertaining to specific fields affect people's choices of majors in graduate school? Similarly, does the average rating of those movies have any effect on major choice?

Importance of Our Research Questions:

Our research questions look at the social impacts of the media people consume and how it might reflect individuals' long term decision making. Movies as a communication medium are extremely impactful due to their visual storytelling elements and, when watched at keypoints in an individual's life, can become key parts of their subconcious moral compass or heavily influence their interests (think Star Wars, Harry Potter, and Hunger Games).

Pregistration Statement

- Preregistration Statement #1:
  - Hypothesis: Some Netflix movie genres, such as Crime and Sci-Fi, are more predictive than other genres towards their relevant major/career field.
  - Analysis: We will use a random forest to classify which majors/career fields belong to which genre of movie. This can be done in multiple different ways with different categorical variables (movie genre, tags related to movie genre), but the end result would allow us to understand which movie genres and majors/career fields are actually related to one another and to what extent.
- Preregistration Statement #2:
  - Hypothesis: The number of movie titles released according to genre is predictive of the popularity of related academic fields and professions in the following years.
  - Analysis: We aim to run multiple regression models such as linear regression, random forest classification, and K-means clustering to determine the extent to which media portrayal of certain fields influences:
    1. The application rate of certain majors and
    2. Entrants into a career path. Based on this, we will also analyze various variables such as user ratings, demographic data, and genre to determine the feature importance of each predictor. We would also like to determine the most predictive of

models, and optimize a decision threshold that would output the likelihood of a movie’s popularity influencing the audience based on its ratings, tags, and streaming numbers.

How Proposed Analyses Would Address Our Research Questions:

- Random Forest Classification: Under ideal circumstances, using a Random Forest Classification model would allow us to see what majors and what movie genres have relationships to one another.
- K-Mean clustering: We would be able to vizaulize the number of different clustered categories that could be used as reference to divide our data.
- Logistic Regression: Logistic models will predict changes in majors' popularities by using changes of related movies' ratings and numbers. We could then see which of these inputs, if any, are good predictors.
- Linear Regression: Examine if there is any predictive relationship between movie genres and major popularity.

Summary of Findings:

- Initial Random Forest Calssifier: We trained models to predict a major based off of a movie's average rating, primary genre, and release year. We found that our model was only able to predict accurate results a little less than 50% of the time, which means there is not signifigant enough evidence to accept the hypothesis from pre-registration statement #1.
- Logistic Regression: We trained models to predict whether a major would have grown more popular based on how many related movies were released at least six years prior, as well as on how these movies were rated. Using a logistic model, we found that if a specific movie category becomes larger in proportion, it is more likely that the related major grows more popular. We did not find using the change in average rating of the movie category, however, to be a useful predictor of the growth of the number of graduates in related fields.
- K-means clustering: The output of k-means clustering allows us to group movies into several categories depending on similarities in how they predict choice of graduate fields. While it is difficult to detemine exactly what each of these categories signify, this data can be used as reference for future extensions wherein the features of such titles (theme, genre, time of release) audience can be analyzed more precisely. In this way, specific features can be selectively defined and examined from each cluster to obtain a better-fitting logistic regresison model to predict whether they are likely to encourage application into related fields.
- Linear Regression: There was no strong association between the increase of a field of study in demand based on the change in proportion of movie titles released under related categories.

Data Analysis:

Data Description

Data Composition

- 3 datasets were used for this project.
  - 538 college majors - Consists of data collected between 2010 and 2012 on student graduation based on major field. This was drawn from the American Community Survey 2010-2012 Public Use Microdata Series and was originally used as a primary for purposes of helping students decide a college major
    - Source: https://www.kaggle.com/datasets/fivethirtyeight/fivethirtyeight-college-majors-dataset
  - Netflix Data Analysis - Titles of all movies currently streaming on Netflix
    - https://www.kaggle.com/code/ameyagrawal/netflix-data-analysis/input?select=titles.csv
  - Movie Lens - Ratings of movies by users and their demographics (age, educational background, career, etc)
    - https://grouplens.org/datasets/movielens/latest/ ### Rationale The college majors and movie rating datasets were selected based on previous research on potential associations between the 2 variables Hence, a netflix title dataset with genres that could be easily associated to a set of given graduation fields had to be chosen.Both were collected around the same time. Of course, it is difficult to pinpoint an exact timeline of how and when movie popularity would influence one's academic deicions, but they both span around the same 10 year range and allow for enough time Hence the data could be easily compared based on categorization of appropriate movie genres into their relevant academic basis. Netflix Data Analysis was chosen to provide an updated list of titles that could be cross-verified with the Movie Lens ratings to ensure their current relevance. ### Preprocessing of Majors, Movies, and Netflix datasets
- MovieLens: We had to do a LOT of preprocessing to makes this df useable for our project. The data sets are extremely large so it was important to filter out data we did not need in the initial phases of df consolidation. We filtered out movies before 2000 and not on Netflix. We removed a lot of data including irrelevant columns including, but not limited to imdbid and timestamp. We also found the average movie rating for each movie instead of having individual ratings from different users. In an ideal dataset there would be job information about each user so we could more carefully examine the patterns between jobs and movie ratings, however this data was not available.
- Majors: Because of the large amount of majors and lack of time to assign tags to each major, we narrowed down the Major df by selecting the following majors: 'PRE-LAW AND LEGAL STUDIES', 'CRIMINAL JUSTICE AND FIRE PROTECTION',mCOMPUTER SCIENCE' , 'MATHEMATICS', 'COMPUTER ADMINISTRATION MANAGEMENT AND SECURITY', 'MILITARY TECHNOLOGIES', 'MASS MEDIA', 'CRIMINOLOGY', 'INTERNATIONAL RELATIONS', 'POLITICAL SCIENCE AND GOVERNMENT', 'ECOLOGY', 'GENETICS', 'MICROBIOLOGY', 'GENERAL EDUCATION','COMMERCIAL ART AND GRAPHIC DESIGN', 'MUSIC', 'STUDIO ARTS', 'ENGLISH

LANGUAGE AND LITERATURE', 'HISTORY', 'PHILOSOPHY AND RELIGIOUS STUDIES', 'MINING AND MINERAL ENGINEERING', 'MECHANICAL ENGINEERING', 'ENVIRONMENTAL ENGINEERING','GENERAL BUSINESS','ASTRONOMY AND ASTROPHYSICS', 'NUCLEAR, INDUSTRIAL RADIOLOGY, AND BIOLOGICAL TECHNOLOGIES','MULTI-DISCIPLINARY OR GENERAL SCIENCE','TREATMENT THERAPY PROFESSIONS', 'HEALTH AND MEDICAL PREPARATORY PROGRAMS', 'GENERAL MEDICAL AND HEALTH SERVICES'. We also removed columns including, but not limited to gendered columns and Major Code, and Rank.

- Netflix: There was not much to do with the Netflix data, which is why we didn't include much information about it. It became a digital platform after 2006 so it is already relevant to the Movie dataframe. We used the movie titles column to remove any movies in the Movie df that was not on Netflix.

Final Variable Descriptions

- The final datasets that were used consisted of and examined the following primary variables of interest:
  - Major - The specific major with which students graduated in.
  - The overarching department/category under which these majors fell under
  - Grad_total - The students who were enrolled as graduate students between 2010 and 2012
  - recent_grad_total - The total number of students who recently graduated and entered the work force between 2010 and 2012.
  - avg_rating - Ratings by the general public which gauged interest/favorability of a movie
  - title - Original name of the movie as listed on Netflix
  - genres - The genre each movie was listed under: can be multiple for each movie. This is the field that was used to compare data with major categories
  - release_year - The year each movie was released during. This allows us to look at how the timeline of each movie's promotion/release may have affected the major choices of graduate school students and career choices of those who chose to start working.
  - relevant_tags - Additional keywords that the movie titles on Netflix were listed with. These provide further information on how we can sort for specific tags related to job industries (\"crime\", \"police\", \"hospital\", etc.)."

Once we decided on the route of analysis for our data, we had to do some additional cleaning to create a dataframe that caters to our needs. While the part of our data cleaning that was submitted in Phase II has been relegated to an appendix, this is here for documentation of what has been done after Phase II. We can later move this to our appendix.

In [2]:
```
import pandas as pd
import duckdb as db
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math as math
import random

from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn import preprocessing, metrics
from sklearn.preprocessing import LabelEncoder as label_encoder
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, \
recall_score, ConfusionMatrixDisplay, roc_auc_score, silhouette_score
from scipy.stats import randint
```

Dataframe Importation

In [4]:
```
# All of these dataframes need to be created by running all cells in the "PhaseVDataCleaning" jupyter notebook
# in the git repository
maj_mov_df = pd.read_csv('majors_movies.csv')
movies_new = pd.read_csv('new_movies.csv')
movies_old = pd.read_csv('old_movies.csv')
movies_titles = pd.read_csv('new_movies.csv', header = 0)
majmovietags_final = pd.read_csv('majmovietags_final.csv')
```

Random Forest Categorization

We began our analyzation process by performing an initial random forest categorization to see if there is any association between movies and majors.

In [42]:
```
# Making Random Forest classifications
rando = RandomForestClassifier(random_state = 2950, n_estimators = 200)
majmovietags_final = majmovietags_final.fillna(0)
x_movie = majmovietags_final[['avg_rating', 'release_year', 'primary_genre']]
y_major = majmovietags_final[['PRE-LAW AND LEGAL STUDIES', 'CRIMINAL JUSTICE AND FIRE PROTECTION',
                 'COMPUTER SCIENCE' , 'MATHEMATICS', 'COMPUTER ADMINISTRATION MANAGEMENT AND SECURITY',
                  'MILITARY TECHNOLOGIES',
                  'CRIMINOLOGY', 'INTERNATIONAL RELATIONS', 'POLITICAL SCIENCE AND GOVERNMENT',
                  'ECOLOGY', 'GENETICS', 'MICROBIOLOGY',
                  'COMMERCIAL ART AND GRAPHIC DESIGN', 'MUSIC', 'STUDIO ARTS',
                  'ENGLISH LANGUAGE AND LITERATURE', 'PHILOSOPHY AND RELIGIOUS STUDIES',
                  'MINING AND MINERAL ENGINEERING', 'MECHANICAL ENGINEERING', 'ENVIRONMENTAL ENGINEERING',
                  'GENERAL BUSINESS',
                  'ASTRONOMY AND ASTROPHYSICS',
           'NUCLEAR, INDUSTRIAL RADIOLOGY, AND BIOLOGICAL TECHNOLOGIES',
           'MULTI-DISCIPLINARY OR GENERAL SCIENCE']]

# 'MASS MEDIA', 'HISTORY', 'TREATMENT THERAPY PROFESSIONS' are not included because there are no
# no movies associated with these majors

x_train_movie, x_test_movie, y_train_major, y_test_major = train_test_split(
    x_movie, y_major, test_size=0.3, random_state=2950)

rando.fit(x_train_movie, y_train_major)
major_prediction = rando.predict(x_test_movie)
major_precision = precision_score(y_test_major, major_prediction, average = 'micro')
major_recall = recall_score(y_test_major, major_prediction, average = 'micro')
major_acc = accuracy_score(y_test_major, major_prediction)
print(f"Model precision: {np.round(major_precision,3)}\nModel recall: {np.round(major_recall,3)}\
\nModel Accuracy: {np.round(major_acc,3)}")
```

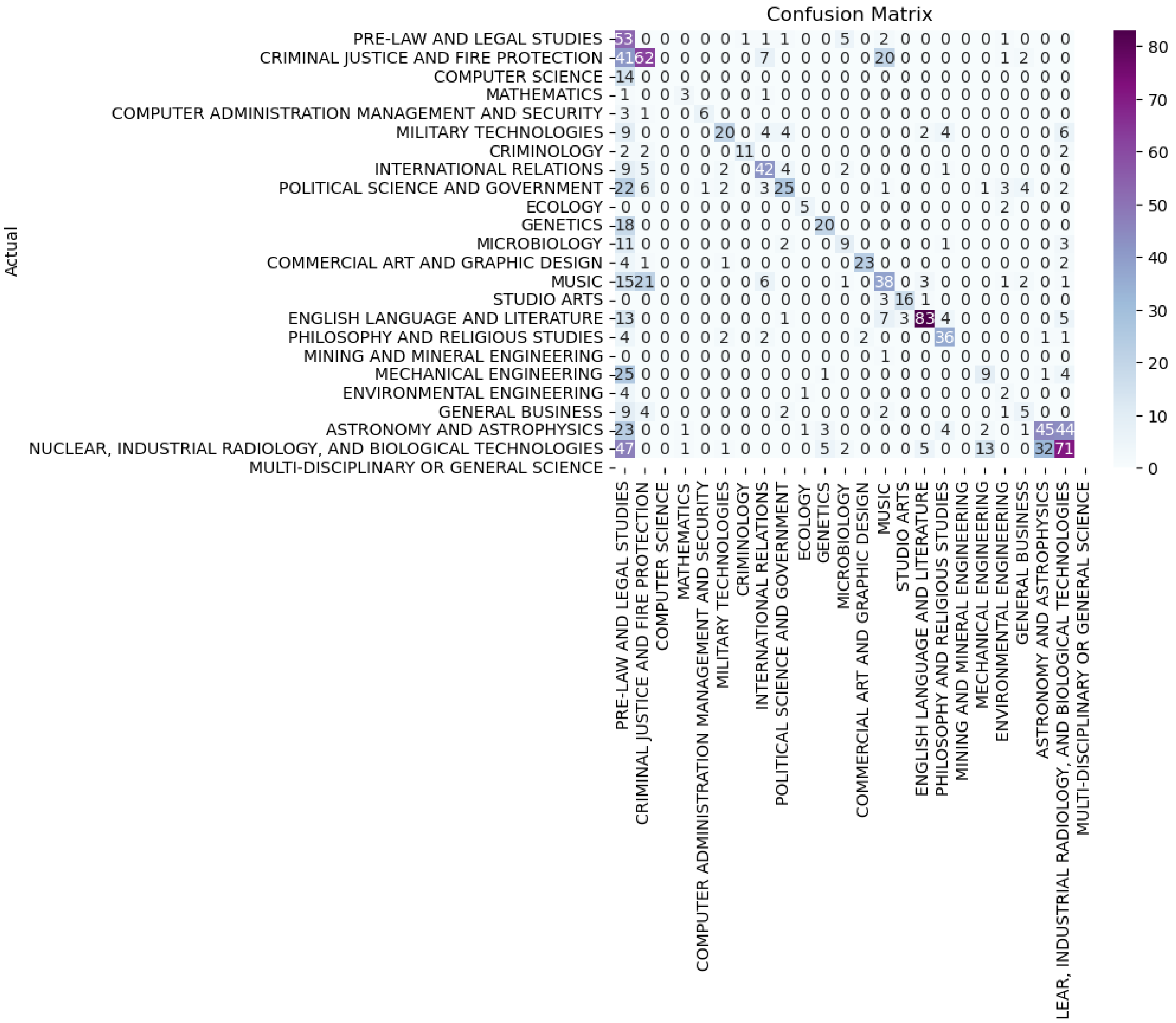
```
Model precision: 0.628
Model recall: 0.48
 Model Accuracy: 0.495
```

Model evaluation

- Precision analysis: Our random forest classifier has a higher precision which means that it is good at predicting which movies with certain tags, genres, and ratings correspond to what majors. Our model is good at predicting positives.
- Recall analysis: Our model has a less than .5 recall, which means that our model predicts true positives out of all positives a little less than half the time.
- Accuracy analysis: Our model has a slightly less than .5 accuracy which means that our model is only predicting the majors a movie might have a relationship with less than 50% of the time.

In order to visualize these results, we decided to create a confusion matrix

```
In [48]: # creating the confusion matrix
mmt_confusionmatrix = confusion_matrix(y_test_major.values.argmax(axis=1), major_prediction.argmax(axis=1))
sns.heatmap(mmt_confusionmatrix, annot=True, fmt='d', cmap='BuPu', \
            xticklabels=y_major.columns, yticklabels=y_major.columns)
#labeling the axes and title of the confusion matrix
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

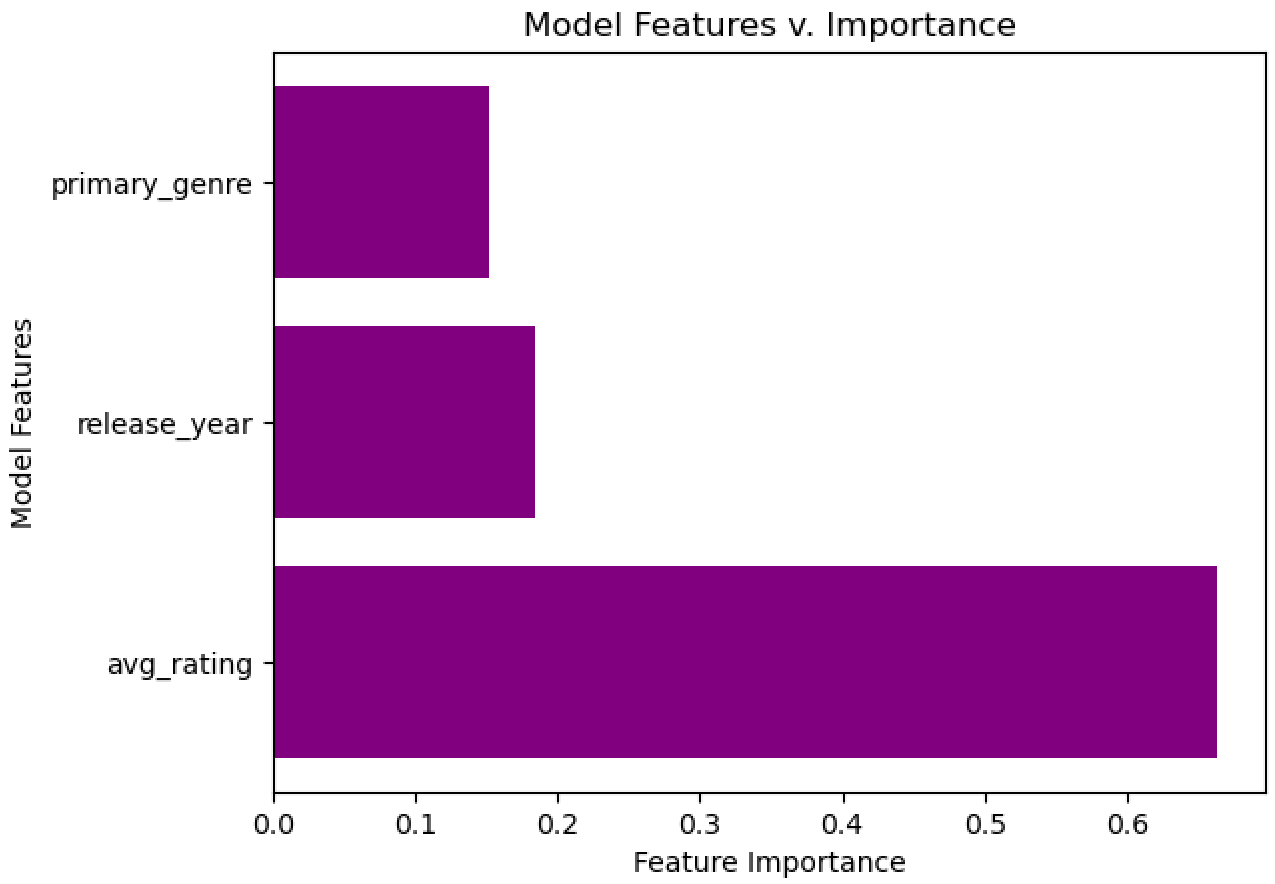


	PRE-LAW AND LEGAL STUDIES	CRIMINAL JUSTICE AND FIRE PROTECTION	COMPUTER SCIENCE	MATHEMATICS	COMPUTER ADMINISTRATION MANAGEMENT AND SECURITY	MILITARY TECHNOLOGIES	CRIMINOLOGY	INTERNATIONAL RELATIONS	POLITICAL SCIENCE AND GOVERNMENT	ECOLOGY	GENETICS	MICROBIOLOGY	COMMERCIAL ART AND GRAPHIC DESIGN	MUSIC	STUDIO ARTS	ENGLISH LANGUAGE AND LITERATURE	PHILOSOPHY AND RELIGIOUS STUDIES	MINING AND MINERAL ENGINEERING	MECHANICAL ENGINEERING	ENVIRONMENTAL ENGINEERING	GENERAL BUSINESS	ASTRONOMY AND ASTROPHYSICS	NUCLEAR, INDUSTRIAL RADIOLOGY, AND BIOLOGICAL TECHNOLOGIES	MULTI-DISCIPLINARY OR GENERAL SCIENCE
PRE-LAW AND LEGAL STUDIES	53	0	0	0	0	0	1	1	1	0	0	5	0	2	0	0	0	0	0	1	0	0	0	
CRIMINAL JUSTICE AND FIRE PROTECTION	41	62	0	0	0	0	0	7	0	0	0	0	0	20	0	0	0	0	0	1	2	0	0	
COMPUTER SCIENCE	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
MATHEMATICS	1	0	0	3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
COMPUTER ADMINISTRATION MANAGEMENT AND SECURITY	3	1	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
MILITARY TECHNOLOGIES	9	0	0	0	0	20	0	4	4	0	0	0	0	0	0	0	2	4	0	0	0	0	6	
CRIMINOLOGY	2	2	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	
INTERNATIONAL RELATIONS	9	5	0	0	0	2	0	42	4	0	0	2	0	0	0	0	1	0	0	0	0	0	0	
POLITICAL SCIENCE AND GOVERNMENT	22	6	0	0	1	2	0	3	25	0	0	0	0	1	0	0	0	0	1	3	4	0	2	
ECOLOGY	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	2	0	0	0	
GENETICS	18	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	
MICROBIOLOGY	11	0	0	0	0	0	0	0	2	0	0	9	0	0	0	0	1	0	0	0	0	0	3	
COMMERCIAL ART AND GRAPHIC DESIGN	4	1	0	0	0	1	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	2	
MUSIC	15	21	0	0	0	0	0	6	0	0	0	1	0	38	0	3	0	0	0	1	2	0	1	
STUDIO ARTS	0	0	0	0	0	0	0	0	0	0	0	0	0	3	16	1	0	0	0	0	0	0	0	
ENGLISH LANGUAGE AND LITERATURE	13	0	0	0	0	0	0	0	1	0	0	0	0	7	3	83	4	0	0	0	0	0	5	
PHILOSOPHY AND RELIGIOUS STUDIES	4	0	0	0	0	2	0	2	0	0	0	0	2	0	0	0	36	0	0	0	0	1	1	
MINING AND MINERAL ENGINEERING	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
MECHANICAL ENGINEERING	25	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	9	0	0	1	4	
ENVIRONMENTAL ENGINEERING	4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	
GENERAL BUSINESS	9	4	0	0	0	0	0	0	2	0	0	0	0	2	0	0	0	0	0	1	5	0	0	
ASTRONOMY AND ASTROPHYSICS	23	0	0	1	0	0	0	0	0	1	3	0	0	0	0	0	4	0	2	0	1	45	44	
NUCLEAR, INDUSTRIAL RADIOLOGY, AND BIOLOGICAL TECHNOLOGIES	47	0	0	1	0	1	0	0	0	0	5	2	0	0	0	5	0	0	13	0	0	32	71	
MULTI-DISCIPLINARY OR GENERAL SCIENCE																								

The model seems to falsely predict that a lot of movies are associated with the 'PRE -LAW AND LEGAL STUDIES' when in actuality they are not. The best two predicted majors based off of movie tags are 'ENGLISH LANGUAGE AND LITERATURE' with 81 true positives, 'MULTI-DISCIPLINARY OR GENERAL SCIENCE' with 71 true positives and 'CRIMINAL JUSTICE AND FIRE PROTECTION' with 62 true positives.

From here, we wondered "Which features impact the classification of each major?". We are able to evalute this by using a feautre importance model.

In [49]:
```
# creating a bar chart with x = features and y = feature importance
plt.barh(x_movie.columns, rando.feature_importances_, color = 'purple')

# labeling the axes and title of the chart
plt.xlabel('Feature Importance')
plt.ylabel('Model Features')
plt.title('Model Features v. Importance')
plt.show()
```



This visualization illustrates that the average rating of a movie is the most important feature when analyzing what major it might have a relationship with and the primary is actually the least important, which would actually give us cause to reject the hypothesis from preregistration #1.

Logistic Regression

In [8]:
```
def Normalizer(df_cols):
    scaler = preprocessing.StandardScaler().fit(df_cols)
    return(scaler.transform(df_cols))
```

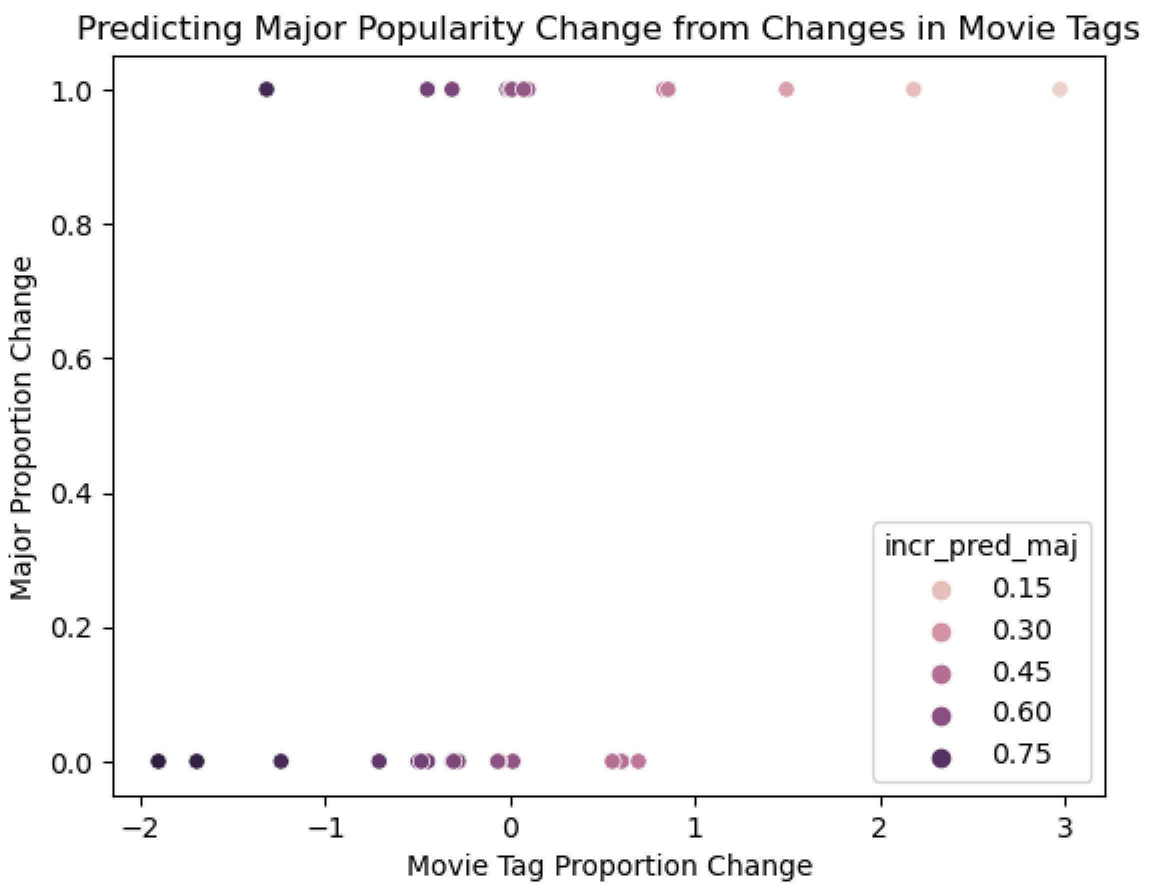
After we did an initial random forest calssification to evaluate if there is possible relationship between movie characteristics and majors, we wanted to conduct a logistic regression to see how well the changes in movie tag proportions and average ratings predict the change in proportions of graduates in the related major.

In [9]:
```
mov_old_norm = Normalizer(maj_mov_df[['movies_old_prop']])[:,0]
mov_new_norm = Normalizer(maj_mov_df[['movies_new_prop']])[:,0]
maj_mov_df['mov_prop_diff'] = mov_new_norm-mov_old_norm
maj_mov_df['mov_prop_diff_unnorm'] = maj_mov_df['movies_new_prop']-maj_mov_df['movies_old_prop']
X = Normalizer(maj_mov_df[['mov_prop_diff_unnorm']])[:,0]
X=np.array(X).reshape(-1,1)
maj_mov_df['mov_prop_norm']=X
#X = np.array(mov_new_norm-mov_old_norm).reshape(-1,1)

maj_prop_diff = maj_mov_df['current_grads_prop']-maj_mov_df['old_grads_prop']
maj_has_increased = np.array(maj_prop_diff>0)
maj_mov_df['maj_has_increased'] = maj_has_increased


log_model = LogisticRegression().fit(X, maj_has_increased)
print(f'The model has coefficient {round(log_model.coef_[0][0],3)} and intercept {round(log_model.intercept_[0],3)}')
```

```
The model has coefficient 0.911 and intercept -0.278
```

The positive coefficient shows that the relation found by the model positively correlated the change in proportion of movies with specific tags with the likelihood of the related major having increased. This supports our hypothesis that higher popularity in relevant movies boost popularity of related fields. However, to figure out how well the model works, we can see the graph of our predictions.

In [10]:
```
incr_pred=log_model.predict_proba(X)[:,0]
maj_mov_df['incr_pred_maj'] = incr_pred
scatter = sns.scatterplot(data=maj_mov_df, x='mov_prop_norm', y='maj_has_increased', hue='incr_pred_maj');
scatter.set_title('Predicting Major Popularity Change from Changes in Movie Tags');
scatter.set_xlabel('Movie Tag Proportion Change');
scatter.set_ylabel('Major Proportion Change');
```



Here, we see a considerable amount of overlap (with respect to their x coordinates) between the dots at y=1 and those at y=0. On this overlapped interval, the model predicted a certain probability of the major having become more popular but the real data shows that only some of them actually did increase in popularity. However, we do see that there are regions of no overlap; movie categories with a large magnitude of change do seem to be correctly matched with whether or not the related major shows an increase in popularity. To quantify how well the model works, we can look at the area under the ROC curve.

In [11]:
```
preds = log_model.predict_proba(X)

metrics.roc_auc_score(maj_mov_df['maj_has_increased'], preds[:,1])
```

Out[11]:
```
0.755656108597285
```

The AUC is nearly 0.8, meaning the change in the prominence of a specific movie category is a fairly good predictor of whether or not related majors will increase in popularity.

Some tag categories had no associated movies in one category (old or recent) but not the other. In this case, there are deceptively high differences, so we removed these by setting the corresponding category to 0 movies for the other (old/young) case as well.

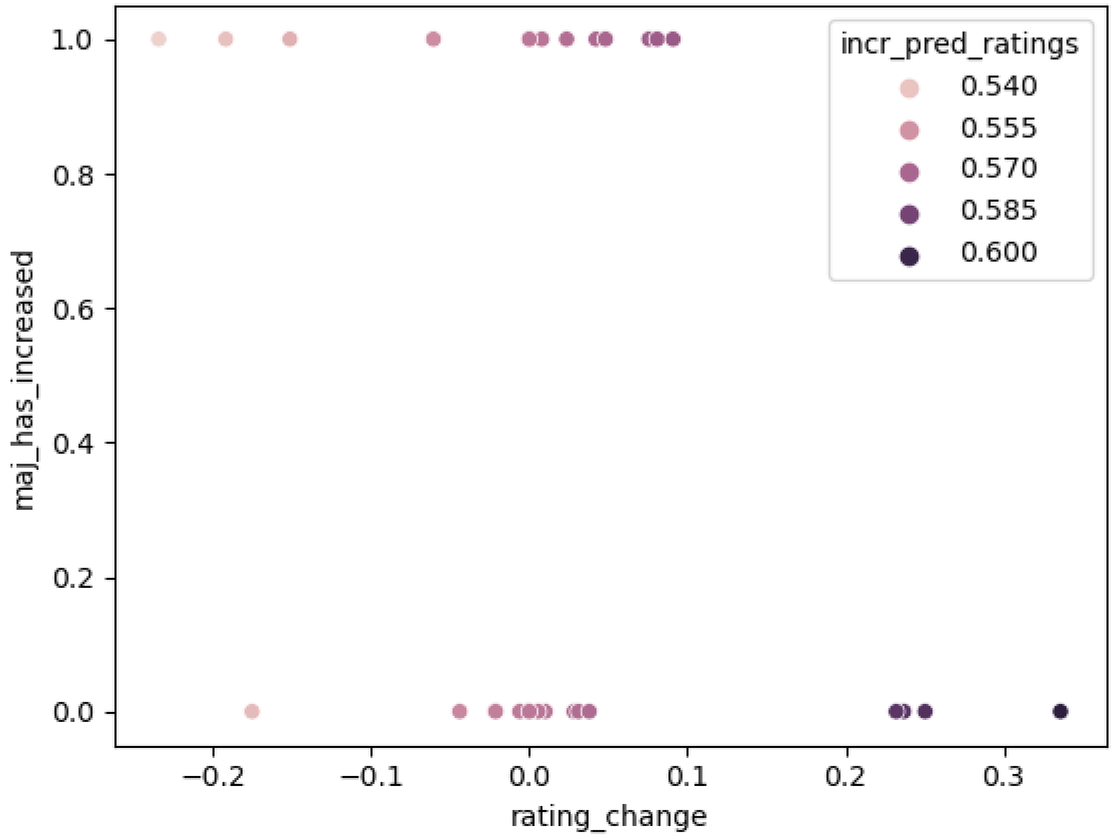
Now we can also see if the movies' ratings had any effect on whether the related major's proportion changed.

In [12]:
```
rating_change=maj_mov_df['new_ratings']-maj_mov_df['old_ratings']
X_ratings = (np.array(rating_change)).reshape(-1,1)
maj_mov_df['rating_change']= X_ratings
maj_mov_df['maj_prop_change']=X

#log_model = LogisticRegression().fit({X, X_ratings}, maj_has_increased)
```

In [13]:
```
log_model = LogisticRegression().fit(X_ratings, maj_has_increased)
print(f'The model has coefficient {round(log_model.coef_[0][0],3)} and intercept {round(log_model.intercept_[0],3)}')
```

```
The model has coefficient -0.488 and intercept -0.258
```

In [14]:
```
incr_pred=log_model.predict_proba(X_ratings)[:,0]
maj_mov_df['incr_pred_ratings'] = incr_pred
scatter = sns.scatterplot(data=maj_mov_df, x='rating_change', y='maj_has_increased', hue='incr_pred_ratings');
```



```
In [15]: preds = log_model.predict_proba(X_ratings)

         metrics.roc_auc_score(maj_mov_df['maj_has_increased'], preds[:,1])
```

```
Out[15]: 0.5475113122171946
```

The negative coefficient shows us that there is a negative correlation between changes in ratings and likelihood of increased major popularity. However, the graph shows a considerable overlap between the top and bottom points, suggesting that rating is not a very strong predictor of whether a major's popularity has increased. This is further corroborated by the fact that the AUC score is around 0.5, suggesting that the model's predictions are no better than if random.

```
In [16]: X_new = Normalizer(maj_mov_df[['rating_change']+['mov_prop_diff']])
         log_model = LogisticRegression().fit(X_new, maj_mov_df['maj_has_increased'])
         print(f'The coefficient for rating is {round(log_model.coef_[0][0],3)} and the one \
         for movie category proportion is {round(log_model.coef_[0][1],3)}')
         print(f"The model's intercept is {round(log_model.intercept_[0],3)}")
```

```
The coefficient for rating is -0.67 and the one for movie category proportion is 0.946
The model's intercept is -0.313
```

In [44]:

```
preds_comb = log_model.predict_proba(X_new)

roc_auc_score_ = metrics.roc_auc_score(maj_mov_df['maj_has_increased'], preds_comb[:,1])
print(f'ROC-AUC score: {np.round(roc_auc_score_,3)}')
```

```
ROC-AUC score: 0.783
```

Predicting changes in graduate field popularity

1. Linear Regression Model

First, we constructed a Linear Regression model to see if the difference in proprtions of movie titles released could be used to predict the difference in proportion of graduates into the corresponding fields.

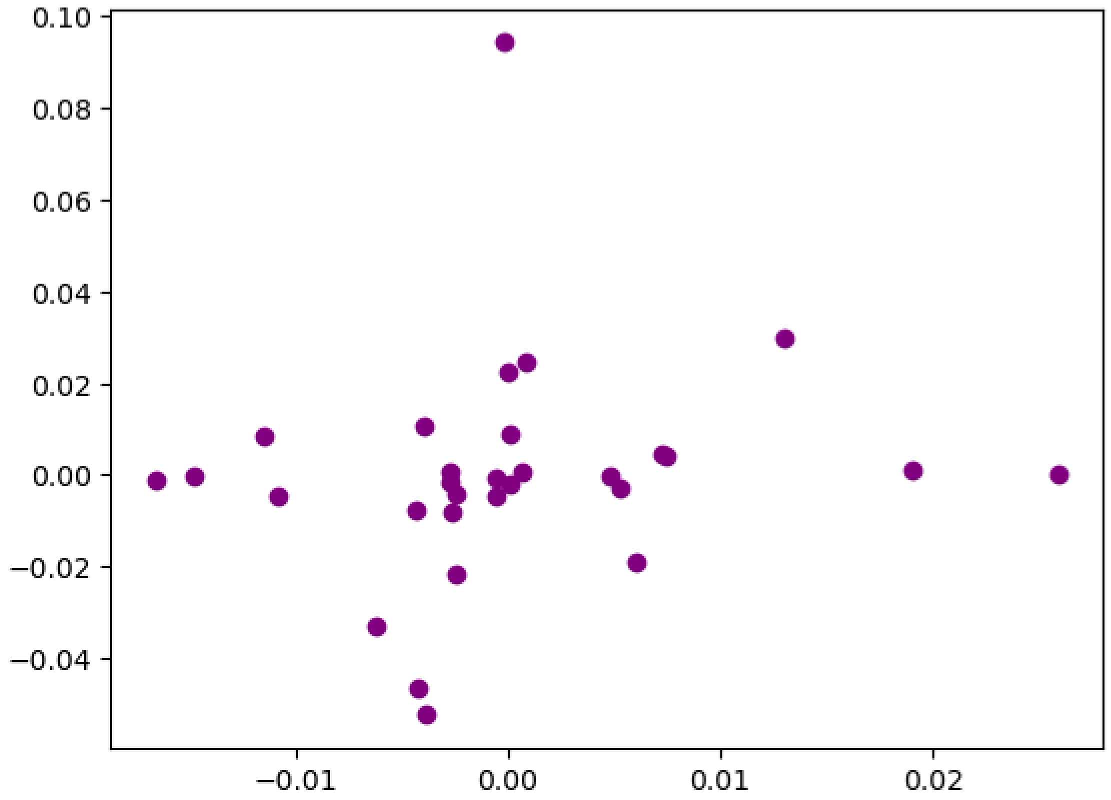
In [50]:

```
mov_to_grad = LinearRegression()
mov_to_grad.fit(maj_mov_df['movies_prop_diff'].values.reshape(-1,1),
                maj_mov_df['grad_prop_diff'])
r2 = mov_to_grad.score(maj_mov_df['movies_prop_diff'].values.reshape(-1,1),
                       maj_mov_df['grad_prop_diff'])
print(f'R Squared score: {np.round(r2, 5)}')
plt.scatter(maj_mov_df['movies_prop_diff'].values.reshape(-1,1),
            maj_mov_df['grad_prop_diff'], color = 'purple')
```

```
R Squared score: 0.02434
```

Out[50]:

```
<matplotlib.collections.PathCollection at 0x195d46c8af0>
```



From the scatterplot above and the r^2 coefficient of determination, no strong association can be modelled based on the given data. Therefore, we decided to use another random forest classification model to predict whether the binary output of a graduate field of study increases in demand depending on the change in proportion of movie titles released under that category.

2. Random Forest Classifier

Training/Test Data Splitting

Next we decided to do another random forest classification but this time with proportion of movie titles and increase of major popularity.

In [19]:
```
x_train, x_test, y_train, y_test = train_test_split(maj_mov_df['movies_prop_diff'].values.reshape(-1, 1),
                                                    maj_mov_df['maj_has_increased'], test_size=0.3)
maj_mov_rf = RandomForestClassifier()
```

```
maj_mov_rf.fit(x_train, y_train)
y_pred = maj_mov_rf.predict(x_test)
```

Here, we first split the movie/major data into training and test set according to a 70-30% division. We then predicted the y variable of whether nunber of graduate applicants into a field increased or not, based on the training data provided.

Model Accuracy, Hyperparameter Retuning

In [20]:

```
accuracy_y_pred = accuracy_score(y_test, y_pred)
print(accuracy_y_pred)
random.seed(2950)
hparameters = {'n_estimators': randint(1,1000),
              'max_depth': randint(1,50)}

best_rf = RandomForestClassifier()

search = RandomizedSearchCV(best_rf,
                                 param_distributions = hparameters,
                                 n_iter=5,
                                 cv=5)

search.fit(x_train, y_train)

final_rf = search.best_estimator_
print('Best hyperparameters:',  search.best_params_)
```

```
0.5555555555555556
Best hyperparameters: {'max_depth': 49, 'n_estimators': 12}
```

The accuracy of this model was then calculated, outputting a value of 0.78. The hyperparameters were then retuned using random sampling from integer ranges to optimize 1. the number of decision branches that would be used to predict the y variable and 2. the depth of each tree in the forest (to ensure a balance between under and overfitting). From this, we see an optimized max depth of 34 and number of estimators as 412.

K-Means Clustering and Visualization of Categories

The next step of our analysis was to finally perform a k-means clustering method to visualize how many different categories could be used as reference to divide the genres of movies and their related academic fields. This would allow us to pinpoint more detailed patterns depending on the clusters. From this, we can arrive at more accurate prediction models, rather than examining all the data together as the same type.

In [21]:
```
maj_mov = pd.DataFrame(maj_mov_df[['movies_prop_diff','grad_prop_diff']])
print(maj_mov)
sil_scores = []
for i in range(2,30):

    KMean_maj_mov = KMeans(n_clusters = i).fit(maj_mov)
    label = KMean_maj_mov.predict(maj_mov)
    sil_scores.append((i,silhouette_score(maj_mov,label)))


sil_scores = pd.DataFrame(sil_scores)

plt.scatter(sil_scores[0],sil_scores[1])

max_sil = max(sil_scores[1])
optimal_clusters = (sil_scores[sil_scores[1] == max_sil])[0]
optimal_clusters
```

```
    movies_prop_diff  grad_prop_diff
0           0.000122       -0.002149
1          -0.003883       -0.052316
2           0.006032       -0.018869
3           0.000839        0.024758
4           0.005244       -0.002708
5          -0.002736        0.000456
6          -0.002445       -0.021419
7          -0.004334       -0.007707
8          -0.010788       -0.004432
9          -0.011462        0.008401
10         -0.002727       -0.001588
11         -0.014750       -0.000095
12          0.007225        0.004585
13         -0.000160        0.094161
14         -0.004182       -0.046509
15         -0.016555       -0.001124
16         -0.000514       -0.004611
17          0.012996        0.030013
18          0.000000        0.022291
19          0.000093        0.008788
20          0.000637        0.000517
21          0.007428        0.004166
22          0.004810       -0.000371
23         -0.006172       -0.033004
24          0.025879        0.000008
25         -0.000586       -0.000528
26          0.018990        0.000907
27         -0.002445       -0.004164
28         -0.003899        0.010594
29         -0.002656       -0.008052
```

```
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
```

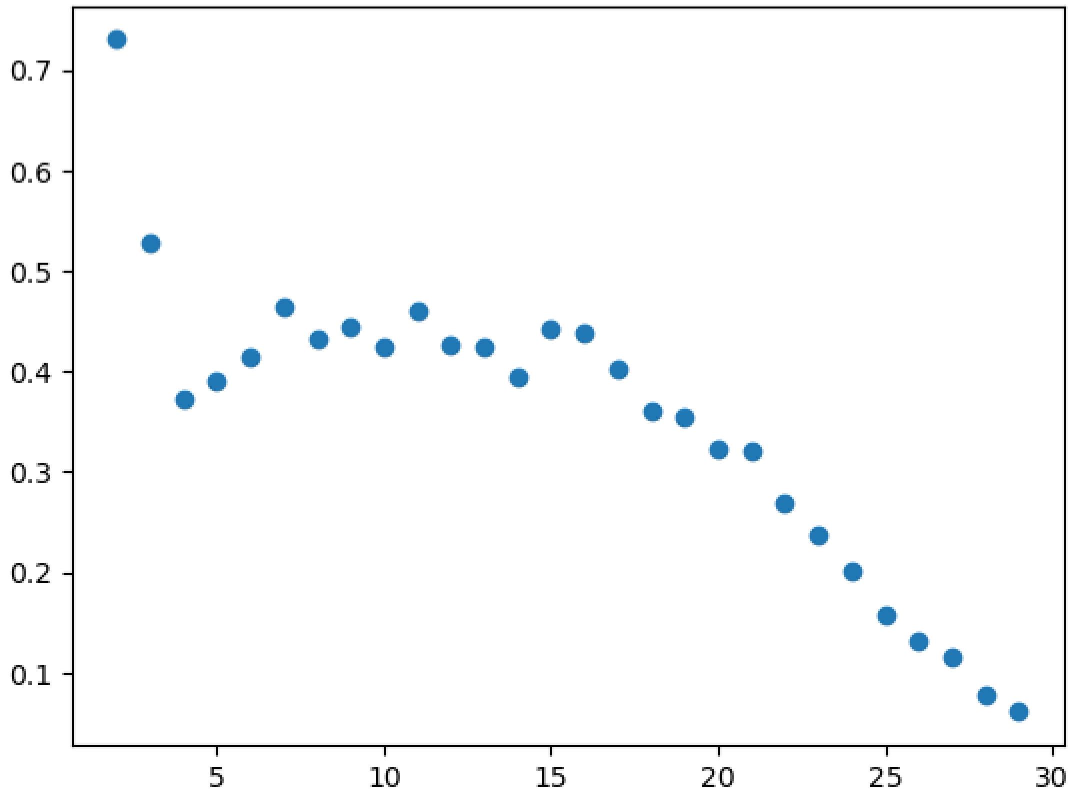
```
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
```

```
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[21]:

```
0    2
Name: 0, dtype: int64
```

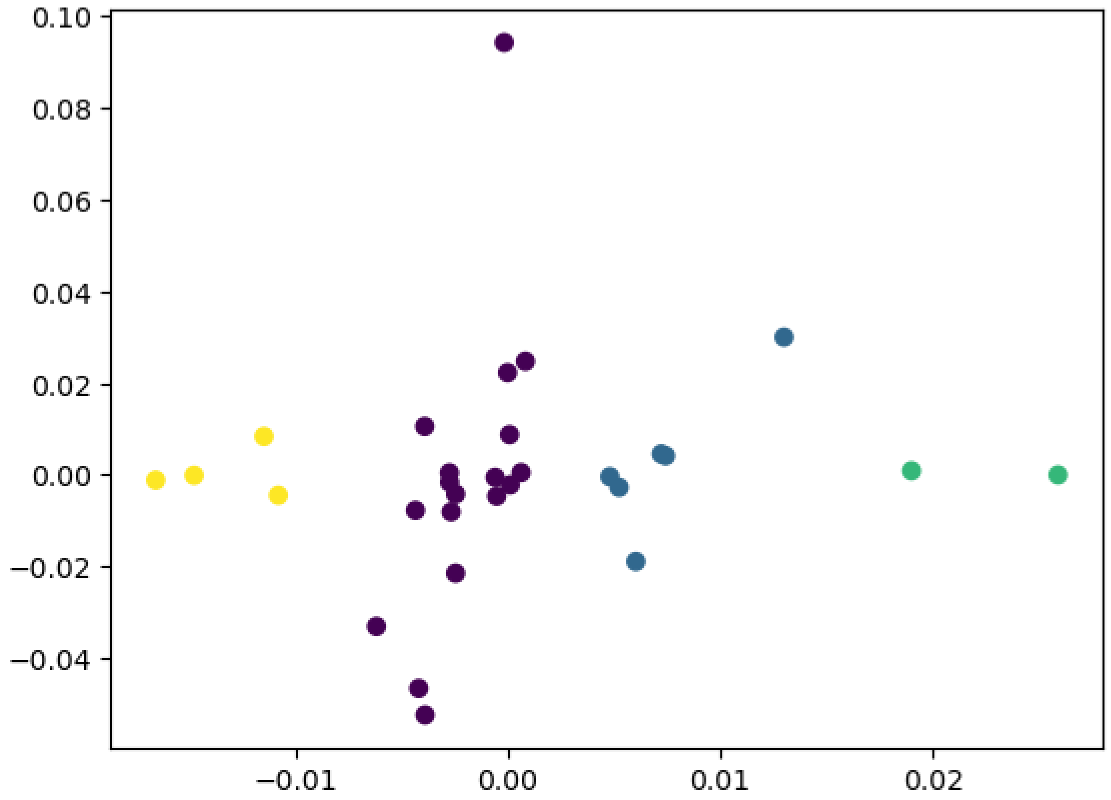


First, a new dataframe was created with only columns of movie proportion differences and graduate proportion differences. To optimize the number of clusters that were to be used, we used a loop and calculated the silhouette scores for possible cluster counts of 2-29.(a measure of how similar a datapoint is to its own cluster, in comparison to an outside one). These were then plotted on a scatterplot to visualize how the silhouette scores change with increasing number of clusters used. The most optimal number was derived to be 4.

In [22]:
```
KMean_optimized = KMeans(n_clusters=4)

labels_final = KMean_optimized.fit_predict((maj_mov['movies_prop_diff'].values.reshape(-1,1)),
                                           maj_mov['grad_prop_diff'])
plt.scatter(maj_mov_df['movies_prop_diff'].values.reshape(-1,1),maj_mov_df['grad_prop_diff'],
            c=labels_final)
```

```
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default va
lue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\glurm\anaconda3\envs\info2950\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known
to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x195cfeb54c0>
```



With this cluster count, we then applied a final k means clustering model on all the data together. We did not split any data into train/test sets. These points were then visualized with the difference in proportions of movies released on the x axis, and the number of graduate applications on the y axis. The colors used indicate the cluster that each movie/major category belongs to.

In [23]:
```
maj_mov_df['labels_final'] = labels_final
maj_mov_df.sort_values(by = 'labels_final')
```

Out[23]:

	Unnamed: 0	major	movies_old	movies_old_prop	movies_new	movies_new_prop	old_ratings	new_ratings	old_grads	old_grads
0	0	PRE-LAW AND LEGAL STUDIES	15.0	0.018337	29.0	0.018460	3.310907	3.547766	13528	0.0
27	27	TREATMENT THERAPY PROFESSIONS	2.0	0.002445	0.0	0.000000	0.000000	0.000000	48491	0.0
25	25	NUCLEAR, INDUSTRIAL RADIOLOGY, AND BIOLOGICAL ...	1.0	0.001222	1.0	0.000637	3.669028	3.625000	2116	0.0
23	23	GENERAL BUSINESS	29.0	0.035452	46.0	0.029281	3.574812	3.553524	234590	0.1
20	20	MINING AND MINERAL ENGINEERING	0.0	0.000000	1.0	0.000637	0.000000	0.000000	756	0.0
19	19	PHILOSOPHY AND RELIGIOUS STUDIES	27.0	0.033007	52.0	0.033100	3.421802	3.429698	54814	0.0
18	18	HISTORY	0.0	0.000000	0.0	0.000000	0.000000	0.000000	141951	0.0
16	16	STUDIO ARTS	15.0	0.018337	28.0	0.017823	3.650815	3.644748	16977	0.0
28	28	HEALTH AND MEDICAL PREPARATORY PROGRAMS	11.0	0.013447	15.0	0.009548	3.653979	3.461977	12740	0.0
13	13	GENERAL EDUCATION	22.0	0.026895	42.0	0.026735	3.242384	3.318111	143718	0.0
14	14	COMMERCIAL ART AND GRAPHIC DESIGN	18.0	0.022005	28.0	0.017823	3.342725	3.348226	103480	0.0
29	29	GENERAL MEDICAL AND HEALTH SERVICES	23.0	0.028117	40.0	0.025461	3.534976	3.359606	33599	0.0
1	1	CRIMINAL JUSTICE AND FIRE	112.0	0.136919	209.0	0.133036	3.392238	3.420520	152824	0.0

	Unnamed: 0	major	movies_old	movies_old_prop	movies_new	movies_new_prop	old_ratings	new_ratings	old_grads	old_grads
		PROTECTION								
3	3	MATHEMATICS	4.0	0.004890	9.0	0.005729	3.668549	3.692159	72397	0.0
7	7	CRIMINOLOGY	15.0	0.018337	22.0	0.014004	3.527668	3.505979	19879	0.0
6	6	MASS MEDIA	2.0	0.002445	0.0	0.000000	0.000000	0.000000	52824	0.0
5	5	MILITARY TECHNOLOGIES	34.0	0.041565	61.0	0.038829	3.252002	3.294188	124	0.0
10	10	ECOLOGY	9.0	0.011002	13.0	0.008275	3.217837	3.449898	9154	0.0
2	2	COMPUTER SCIENCE	6.0	0.007335	21.0	0.013367	3.407228	3.438356	128319	0.0
22	22	ENVIRONMENTAL ENGINEERING	7.0	0.008557	21.0	0.013367	3.537581	3.575498	4047	0.0
21	21	MECHANICAL ENGINEERING	21.0	0.025672	52.0	0.033100	3.137661	3.185875	91227	0.0
4	4	COMPUTER ADMINISTRATION MANAGEMENT AND SECURITY	3.0	0.003667	14.0	0.008912	3.026077	3.362060	8066	0.0
12	12	MICROBIOLOGY	17.0	0.020782	44.0	0.028008	3.580728	3.429302	15232	0.0
17	17	ENGLISH LANGUAGE AND LITERATURE	43.0	0.052567	103.0	0.065563	3.540220	3.305760	194673	0.1
24	24	ASTRONOMY AND ASTROPHYSICS	46.0	0.056235	129.0	0.082113	3.032757	3.113644	1792	0.0
26	26	MULTI-DISCIPLINARY OR GENERAL SCIENCE	86.0	0.105134	195.0	0.124125	3.111654	3.202563	62052	0.0
15	15	MUSIC	88.0	0.107579	143.0	0.091025	3.428737	3.427395	60633	0.0
8	8	INTERNATIONAL RELATIONS	51.0	0.062347	81.0	0.051560	3.493779	3.503813	28187	0.0

	Unnamed: 0	major	movies_old	movies_old_prop	movies_new	movies_new_prop	old_ratings	new_ratings	old_grads	old_grads
9	9	POLITICAL SCIENCE AND GOVERNMENT	88.0	0.107579	151.0	0.096117	3.619341	3.558680	182621	0.0
11	11	GENETICS	23.0	0.028117	21.0	0.013367	3.045143	3.295484	3635	0.0

30 rows × 23 columns

The labels were then added to the original data frame, consisting of the cluster numbers (from 0-3, inclusive). This therefore allows us to examine the association between movie and graduates data based on specific groupings and analyze which features specifically are responsible for these differences in association.

Conclusion

Limitations:

- Lack of available data:
  - MovieLens: While there are a variety of dataframes available for movie ratings, it was difficult to find a df that included demographic information on each reviewer. There a lot of MovieLens dataframes, but the ones that had this information only had movies going up until 2000.
  - Majors: There are not a lot of dataframes available for college majors that include a major by major breakdown of graduation and employment statistics. The ideal dataframe would include information on each graduating class, transfers in and out of majors, and employment rates post graduation. However, there are no dataframes dataframes like this that we could find.
- Lack of desired variables:
  - No Time Series for Majors df: There is no time series for the Majors df, which means that we did know what graduating class the data could be attributed to between 2010-2012. We had to treat all of the years as one, which could have caused us to miss important year by year differences in the Major dfs.
  - No individual user information for Movies df: Originally, we examined a dataset from GroupLens that included demographic information for each individual user. This would have given us a common col between the Majors and Movies dataframes, because the Majors dataframe also had data relating to jobs post graduation. However, the movie data was not relevant to the Major df because it only had movies from before 2000, which does not correspond to the 2010-2012 collection of the Major data.

- No common column between main datasets:
  - The main challenge we faced in the analyzation of our data is that there were no common columns over the Major and Movie dfs. To address this problem we assigned specific movie tags to related majors. However, this process was time consuming and leaves a lot of room for error, which is discussed in the next section. If the Major df had time series data relating to the statistics of majors for each year from 2010-2012 we would have had a way to analyze the data because of the related column
- Tags and Major:
  - Hand assignment: Because there was no common column between Major and Movie dfs, we had to hand assign movie tags that we thought were relevant to each selected major. This has a lot of room for human error because it is tedious and is up to the researcher's discretion to decide what tags are releveant to what majors.
  - Lack of data: Despite the fact that there were over three hundred tags, there is still a lack of tag data because there are so many words that can relate to a major or career field that were not listed in the tag data. There are also possibly some tags that we didn't select that could be relevant to the majors.

Potential Harms, Data Gaps, and Impacts:

- Variety of variables that go into choosing a major: Movies that come out in a certain time period are one of the possibly millions of reasons that an individual might choose a career path. It is an interesting environmental/societal variable, but has lesser impact than variables like finances and familial pressures.
- Financial wellbeing taking precedence over interests: Although personal interests have an impact on the choices that people make, but some individuals might work in an industry they are not interested because it has good job prospects.
- Social pressures: Another way that an individual might may put interest as a low priority is because of familial or other social pressures. A student might feel pressured to pick a major if all of their family is in a certain industry.