**Kennesaw State University**

**College of Computing**

**and Software Engineering**

**DEPARTMENT OF COMPUTER SCIENCE**

**CS 4850 – Senior Project**

****

**Predicting Movie Success Using Machine Learning**

**Derek Steele, Leon Haralambus, Yekta Yalcin, Ubeyd Karasahin**

**29 November 2018**

**Table of Contents**

[**Abstract**](#_bk65yynikt1b) **3**

[Objective](#_ddcy2ejqfdxf) 3

[Introduction](#_2ez9mq36y7t0) 3

[**Design Approach**](#_mwzsoroejye4) **3**

[Data collection](#_u9mos241qfwt) 3

[Program](#_3m3italxbu07) 4

[Output](#_f9dnjojnc0lh) 4

[**Results**](#_m5cn2z6j6xau) **4**

[**Conclusion**](#_evyxvpg35s) **5**

[Issues Encountered](#_gjdgxs) 5

[Was Our Objective Met?](#_mvjb61t8fdkj) 5

[**References**](#_4lz8xeyis629) **6**

## Abstract

#### Objective

Our objective is to better our understanding of machine learning while building a tool that can benefit the general public. More specifically, this project aims to hone our algorithm selection skills (Naive Bayes vs KNN vs Neural Networks vs SVMs vs etc.), better acquaint us with machine learning techniques (supervised vs unsupervised) and lastly, gain experience solving real problem. In addition, this project will reinforce many general programming and algorithm analysis techniques to produce the most efficient program, improve our development operations, improve our many skills necessary for a competent developer such as group communication, remote working, and working within a constrained schedule.

#### Introduction

Movie studios and producers struggle with predicting how well their future movies might perform in the box office on release date; however, there are a few aspects that many successful movies share such as lead actors, genre, and release date. By collecting a list of released movies, we believe we can predict how well the movie will perform when it is released if we can analyze the aspects mentioned before with a ML algorithm. We are choosing to analyze this data with a ML algorithm since, as the name implies, it is focused on “learning” based on a set of testing data and it’s outputs (predictions, in this case) can grow more accurate over the sample set.

## 

## Design Approach

#### Data collection

To gather the data for this project, we created a program in Java to parse data from IMDB using their publicly available API. We collected the rank, title, genre, description, actors, year, runtime, rating, votes, revenue, and metascore for about 1000 previously released movies and stored that data into a .csv file that our program could process. In order to determine the accuracy of the Support Vector Machine (SVM) algorithm, we need to know in which cases the program is predicting correctly and which cases it isn’t. Because of that, we added a new column at the end of the data set called “Success” in which we indicate (using 1s and 0s) whether the movie was successful in our opinion or not.

#### Program

On the program side of our project, we built a program in Python that uses several libraries to split the data into train/test sets, format the data, fit the training set to the SVM model, then analyzes the output. The primary libraries used in our project is “pandas” for reading data out of the .csv file and removing data that is null or not available, and “sklearn” which provides many models used for analyzing data such as SVM, K-Nearest Neighbors (KNN), Naïve Bayes, and more.

#### Output

The output from our program comes in the form of 4 data points: a confusion matrix, accuracy, precision, and recall. The confusion matrix is divided into 4 quadrants:

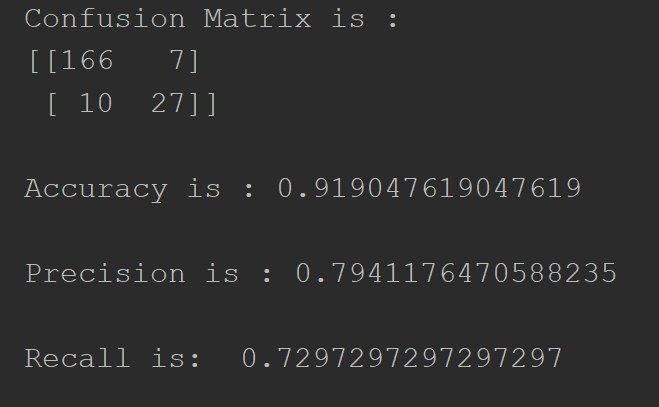
|  |  |
| --- | --- |
| True Negative (TN) | False Positive (FP) |
| False Negative (FN) | True Positive (TP) |

The two “true” quadrants tells us how often the program’s prediction matched up with the actual results while the two “false” quadrants tells us how often the program’s prediction was incorrect. Ideally, we are looking for high numbers in the “true” quadrants and low numbers in the “false” quadrants. Accuracy tells how often our classification was correct overall, so we would like to see a number as close to 1 as possible. Precision tells us how often our classification was correct when it predicts positive, so we would like to see a number as close to 1 as possible. Finally, recall indicates the classifier’s ability to find all positive samples.

# 

## Results

After running the program against our dataset, below is the output from the program:



As shown in the image above, our confusion matrix is showing higher numbers in the first and last (“true”) quadrants which indicates that the program’s estimations are in line with the actual data. The accuracy is above 90% which is a bit higher than we initially expected, but overall a good sign. The precision is around 80% which although a good result is a bit lower than we had anticipated. Finally, the recall value is around 73% which, again, is a good result, but is a bit lower than expected. Overall, the Support Vector Machines algorithm worked well with this data set which is consistent with our research on different ML algorithms. While we didn’t build a ML agent based on the Naive Bayes (the next ML algorithm on our list), we expect the results from the SVM agent to be higher overall based on the fact that Support Vector Machines work well with small, multi-featured data sets while Naive Bayes tends to work best with larger, less featured data sets.

## Conclusion

#### Issues Encountered

During this project we ran into multiple issues:

1. Coding a ML agent in Java - As we soon found out, trying to implement a ML agent using SVM in Java was not a good choice since the availability of ML libraries for java was lacking and we could only find “adapted” libraries online that tried to replicate functionality found in common and well documented Python libraries. Instead of trying to implement these libraries on our own and risk running into a problem that none of us could resolve, we decided to shift our implementation language to Python. At the start of this project, none of use had any Python programming experience, but with the use of library documentation, we were able to work through all of the compilation issues.
2. Scraping data from Twitter, YouTube, and Rotten Tomatoes - Our initial goal was to scrape data from a number of sources and compile all of our data into a master list. This would give us a holistic view of which movies were successful and in what ways (e.g. views in theater, box office revenue, rotten tomato score). Unfortunately, due to Rotten Tomatoes not granting us access to their API, and Twitter and Youtube having APIs that are difficult to parse data from we ended up sticking with one data source for our data collection.
3. Learning Python - As mentioned earlier, because we made the switch from Java to Python, we were giving ourselves another hurdle to jump. Through several online resources, YouTube videos, trial-and-error, and python documentation we were able to implement our project.

#### Was Our Objective Met?

During this project, we researched several ML algorithms and made an informed decision as to which one would work best for our data, we worked together to develop a program which implements the algorithm we chose, and we delivered the project before the project deadline, so overall we were successful in meeting our objectives.

## References

Markham, Kevin. “Simple Guide to Confusion Matrix Terminology.” *Data School*, Data School, 31 Oct. 2018, [www.dataschool.io/simple-guide-to-confusion-matrix-terminology/](http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/).

“1. Supervised Learning.” *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, [www.scikit-learn.org/stable/supervised\_learning.html](http://www.scikit-learn.org/stable/supervised_learning.html).

Shanelynn. “Using Iloc, Loc, & Ix to Select Rows and Columns in Pandas DataFrames.” *Shane Lynn*, 9 July 2017, [www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/](http://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/).

“API Reference.” *Pandas: Powerful Python Data Analysis Toolkit - Pandas 0.23.4 Documentation*, [www.pandas.pydata.org/pandas-docs/stable/api.html](http://www.pandas.pydata.org/pandas-docs/stable/api.html).