**Movie Recommendation Engine**

The rapid growth of data collection has led to a new era of information. Data is being used to create more efficient systems and this is where Recommendation Systems come into play. Recommendation Systems are a type of information filtering systems as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

They are used to predict the rating or preference that a user would give to an item. Almost every major tech company has applied them in some form or the other: Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on autoplay, and Facebook uses it to recommend pages to like and people to follow. Moreover, companies like Netflix and Spotify depend highly on the effectiveness of their recommendation engines for their business and success.

In this Project, we will try and gain some insights using dataset of about 45000 movies with metadata collected from TMDB. Using this data, we will try and answer various questions on how the dataset can be used to predict important feature and also classify if the movie is a hit or not. Another motivation is to build a movie recommendation system.

1. **Problem Statement:**

This project is divided into two parts:

**1. Analysis and prediction model:**

* Exploratory Data Analysis, which is performed on Movie Metadata about Movie Revenues, Genre, Budgets, etc. through the years.
* Model to classify if a movie will be a success or not. Through this model, we also aim at discovering what features have the most significant impact in determining the success of a movie.
  + A **Classifier** that identifies if a movie will be a hit or will make the producers lose money.

**2. Movie Recommender Systems:**

This part is focused around building Content based recommendation engines using the scikit-learn library. This type of recommendation systems takes in a movie that a user currently likes as input. Then it analyzes the contents (storyline, genre, cast, director etc.) of the movie to find out other movies which have similar content. Then it ranks similar movies according to their similarity scores and recommends the most relevant movies to the user.

1. **The Data:**

Dataset link : <https://www.kaggle.com/rounakbanik/the-movies-dataset/data>

movies\_metadata.csv: The main Movies Metadata file. Contains information on around 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies. The movies available in this dataset are in correspondence with the movies that are listed in the MovieLens Latest Full Dataset comprising of 26 million ratings on 45,466 movies from 27,000 users, comprises of 24 features as displayed below –

A screenshot of a cell phone

Description automatically generated

**B.1 Description of Features:**

* **adult:** Indicates if the movie is X-Rated or Adult.
* **belongs\_to\_collection:** A stringified dictionary that gives information on the movie series the particular film belongs to.
* **budget:** The budget of the movie in dollars.
* **genres:** A stringified list of dictionaries that list out all the genres associated with the movie.
* **homepage:** The Official Homepage of the movie.
* **id:** The ID of the movie.
* **imdb\_id:** The IMDB ID of the movie.
* **original\_language:** The language in which the movie was originally shot in.
* **original\_title:** The original title of the movie.
* **overview:** A brief of the movie.
* **popularity:** The Popularity Score assigned by TMDB.
* **poster\_path:** The URL of the poster image.
* **production\_companies:** A stringified list of production companies involved with the making of the movie.
* **production\_countries:** A stringified list of countries where the movie was shot/produced in.
* **release\_date:** Theatrical Release Date of the movie.
* **revenue:** The total revenue of the movie in dollars.
* **runtime:** The runtime of the movie in minutes.
* **spoken\_languages:** A stringified list of spoken languages in the film.
* **status:** The status of the movie (Released, To Be Released, Announced, etc.)
* **tagline:** The tagline of the movie.
* **title:** The Official Title of the movie.
* **video:** Indicates if there is a video present of the movie with TMDB.
* **vote\_average:** The average rating of the movie.
* **vote\_count:** The number of votes by users, as counted by TMDB.

1. **DATA WRANGLING**

This section describes the various data cleaning and data wrangling methods applied on the Movie datasets to make it more suitable for further analysis.

The following sections are divided based on the procedures followed.

C.1 **Removing Unnecessary Features** Some features such as the Backdrop Path, Adult and IMDB ID were unnecessary attributes and were dropped to reduce the dimensions of the dataset.

C.2 **Cleaning The dataset** had a lot of features which had 0s for values it did not possess.

These values were converted to NaN. Some features were still in the form of a Stringified JSON Object. They were converted into Python Dictionaries using Python’s ast library.

These were further reduced into lists since we did not have a need for ID, timestamp and other attributes. The dataframe was exploded wherever the analysis demanded it (for instance, genres and production countries). Finally, most of the features were converted into a Python basic type (integer, string, float) by removing all the unclean values. The date string was converted into a Pandas Datetime and from it, we extracted the month, year and day of release of every movie.

1. **EXPLORATORY DATA VISUALIZATION AND ANALYSIS**

In this section, the various insights produced through descriptive statistics and data visualization is presented.

**Title and Overview Wordclouds:**

Let's consider certain words that figure more often in Movie Titles and Movie overviews and see the words which are considered more potent and considered more worthy of a title.

**Title Wordcloud:**

A close up of a logo

Description automatically generated

The words day, love & girl are the most commonly used word in movie titles. Day, man, boy and night are also among the most commonly occurring words.

**Overview Wordcloud:**

**A close up of a newspaper

Description automatically generated**

Life is the most commonly used word in Movie titles. Family and Find are also popular in Movie overviews. Together with world, one and Friend, these wordclouds give us a pretty good idea of the most popular themes present in movies.

**Original Languages**

There are over 93 languages represented in our dataset. As we had expected, English language films form the overwhelmingly majority. French and Italian movies come at a very distant second and third respectively

A picture containing drawing

Description automatically generated

As mentioned earlier, French and Italian are the most commonly occurring languages after English. Japanese and Hindi form the majority as far as Asian Languages are concerned.

**Popularity, Vote Average and Vote Count**

A screenshot of a social media post

Description automatically generated

**A screenshot of a cell phone

Description automatically generated**

* + Minions is the most popular movie by the TMDB Popularity Score. Wonder Woman and Beauty and the Beast, two extremely successful woman centric movies come in second and third respectively.
  + Inception and The Dark Knight, two critically acclaimed and commercially successful Christopher Nolan movies figure at the top of The Most Voted on Movies Chart.

A screenshot of a social media post

Description automatically generated

* + The Shawshank Redemption and The Godfather are the two most critically acclaimed movies in the TMDB Database.

**Budget**

A screenshot of a social media post

Description automatically generated

* + This is right skewed suggesting the mean value is influenced by outliers.

A screenshot of a cell phone

Description automatically generated

* + The distribution of movie budgets shows an exponential decay. More than 75% of the movies have a budget smaller than 25 million dollars.

**Revenue**

The revenue is probably the most important numeric quantity associated with a movie. We will try to predict the revenue for movies given a set of features in a later section. The treatment of revenue will be very similar to that of budget and we will once again begin by studying the summary statistics.

A screenshot of a social media post

Description automatically generated

We see that the majority of the movies have a recorded revenue of 0. This indicates that we do not have information about the total revenue for these movies. Although this forms the majority of the movies available to us, we will still use revenue as an extremely important feature going forward from the remaining 7000 movies.

A screenshot of a cell phone

Description automatically generated

The mean gross of a movie is 6.87 million dollars whereas the median gross is much lower at 16.8 million dollars, suggesting the skewed nature of revenue. The lowest revenue generated by a movie is just 1 dollar whereas the highest grossing movie of all time has raked in an astonishing \*2.78 billion dollars.

A screenshot of a social media post

Description automatically generated

The distribution of revenue undergoes exponential decay just like budget.

**Correlation Matrix**

**A screenshot of a cell phone

Description automatically generated**

The revenue and budget are found to be strongly correlated. Also, Revenue and vote\_count are highly correlated with Pearson correlation as 0.81. One interesting observation is vote\_count comes out be an important factor for both budget and revenue.

**Genres**

**A screenshot of a cell phone

Description automatically generated**

Drama is the most commonly occurring genre with almost half the movies identifying itself as a drama film. Comedy comes in at a distant second with 25% of the movies having adequate doses of humor. Other major genres represented in the top 10 are Action, Horror, Crime, Mystery, Science Fiction, Animation and Fantasy.

**Pre-processing of data:**

Created new features like is\_English, is\_drama, is\_animation, is\_TVmovie, is\_foreign, is\_documentary, is\_Friday, is\_holiday, is\_musical and so on. For example: in is\_drama, if movie belongs to drama genre it is encoded as 1 else 0.

1. As we understood that the genres are important feature, we turned each genre into a feature like is\_drama, is\_triller etc. Same encoding is followed.
2. Similarly, by making use of original\_language we created is\_English and is\_foreign features. Same encoding is followed.
3. If movie belongs to a collection, it is coded as 1 else 0 (belongs\_to\_collection).
4. Similarly, we have production\_companies, production\_countries, spoken\_languages for which length the value is considered, for example: if a movie belongs to 2 production\_companies, the length will be 2.
5. Imputing missing values with mean – For runtime and vote\_average, we have imputed missing values with mean strategy.

We have displayed the total columns in our final dataset, represented by final\_df .

**Introduction of the models/algorithms used:**

Classification is a subcategory of supervised learning where the goal is to predict the categorical class labels (discrete, unordered values, group membership) of new instances based on past observations. Classification basically involves assigning new input variables to the class to which they most likely belong in based on a classification model that was built from the training data that was already labeled. Labeled data is used to train a classifier so that the algorithm performs well on data that does not have a label (not yet labeled).

In our case, we have binary classes to classify if a movie will be successful or not.

Our target variable (return) has class 1 hit movie is and class 0 flop movie.

So, for classification, we have preferred ensemble models, ensemble models comprise several supervised learning models that are individually trained and the results merged in various ways to achieve the final prediction. This result has higher predictive power than the results of any of its constituting learning algorithms independently. These methods are designed to improve the stability and the accuracy of Machine Learning algorithms. So, we have selected three models they are Random Forest classifier, bagging classifier and voting classifier.

Before we train the model, we split the preprocessed data into train set (80%) and test set (20%). Our class variable “return” is imbalanced, so we have used stratified shuffle split strategy to split the data, from the library

(sklearn. model\_selection- StratifiedShuffleSplit).

* 1. **Voting Classifier:**

Voting is one of the simplest ways of combining the predictions from multiple machine learning algorithms. Voting classifier isn’t an actual classifier but a wrapper for set of different ones that are trained and valuated in parallel in order to exploit the different peculiarities of each algorithm. It is a way to create a better classifier by aggregating the predictions of each classifier and predict the class that gets the most votes.

Consider three Classifiers: logistic regression, a RandomForestClassifier, and SVC classifier. Models are pitted against each other and selected upon best performance by voting using the ‘VotingClassifier’ Class from sklearn.ensemble.

**For Hard Classification:**

A screenshot of a cell phone

Description automatically generated

We got accuracy of 73% for logistic regression, Random forest classifier accuracy was 75% and for SVC it was 70% and amongst the lowest. Overall accuracy of voting classifier is around 75% on hard classification.

**For Soft Classification:**

A screenshot of a cell phone

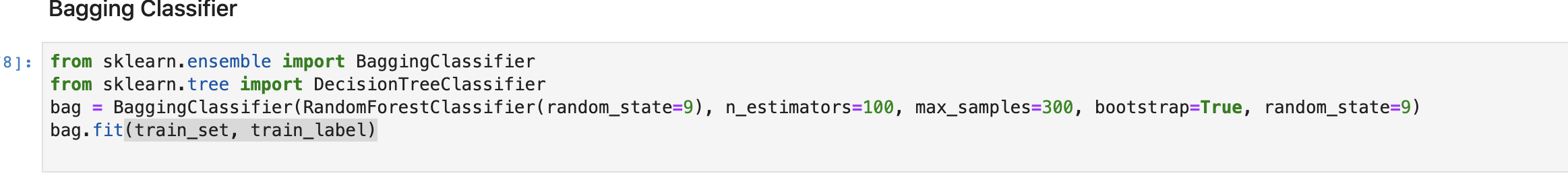
Description automatically generated

Soft classifiers explicitly estimate the class conditional probabilities and then perform classification based on estimated probabilities. In contrast, hard classifiers directly target the classification decision boundary without producing the probability estimation. In our case soft classification gave better accuracy than hard classification.

* 1. **Bagging Classifier:**

Approach to use the same training algorithm for every predictor and train them on different random subsets of the training set. When sampling is performed with replacement, this method is called [bagging](https://homl.info/20).  Bagging allows training instances to be sampled several times for the same predictor. Once all predictors are trained, the ensemble can make a prediction for a new instance by simply aggregating the predictions of all predictors. Each individual predictor has a higher bias than if it were trained on the original training set, but aggregation reduces both bias and variance. Generally, the net result is that the ensemble has a similar bias but a lower variance than a single predictor trained on the original training set.

The Bagging Classifier automatically performs soft voting instead of hard voting if the base classifier can estimate class probabilities (i.e., if it has a predict\_proba() method), which is the case with Decision Tree classifiers.



Here we are taking n\_estimator (number of trees in random forest) as 100,

max\_samples (maximum number of samples random forest can have) as 300 and keeping bootstrap =True means we want to sample data points with replacement.

With this we are getting accuracy of about 76%.

**A screenshot of a cell phone

Description automatically generated**

**Hyperparameter Tuning:**

* **n\_estimators** = number of trees in the forest
* **max\_features** = max number of features considered for splitting a node
* **max\_depth** = max number of levels in each decision tree
* **min\_samples\_split** = min number of data points placed in a node before the node is split
* **min\_samples\_leaf** = min number of data points allowed in a leaf node
* **bootstrap** = method for sampling data points (with or without replacement)

**A screenshot of a cell phone

Description automatically generated**

After GridsearchCV, best parameter is coming to be max\_samples: 500 and n\_estimators :300.

Also, the accuracy after tuning bagging classifier is around 77%.

* 1. **Random forest Classifier:**

 Random forest is an ensemble of Decision Trees, generally trained via the bagging method.  Instead of building a BaggingClassifier and passing it a DecisionTreeClassifier, we can instead use the RandomForestClassifier class, which is more convenient and optimized for Decision Trees. When we are growing a tree in a Random Forest, at each node only a random subset of the features is considered for splitting. It is possible to make trees even more random by also using random thresholds for each feature rather than searching for the best possible threshold. We can control overfitting by tuning the hyperparameters.

We have used “**GridSearchCV**” to get the best parameters and the best model for our problem. Below are the best parameters we get.

A screenshot of a cell phone

Description automatically generated

After training the model with best parameters we are getting accuracy of around 77%

**RECOMMENDATION SYSTEMS**

[A **content-based recommender system**](https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/) works on the data generated from a user. The data can be generated either explicitly (like clicking likes) or implicitly (like clicking on links). This data will be used to create a user profile for the user which contain the metadata of the item’s user interacted. More the data it receives more accurate the system or engine becomes.

This type of recommendation systems takes in a movie that a user currently likes as input.  
Then it analyzes the contents (tagline and overview of movie as description) of the movie to find out other movies which have similar content.

- **TfidfVectorizer:**

The fundamental idea is to convert the texts or words into a vectors.

- **cosine similarity:**

Now, we need to find cosine similarity between these vectors to find out how similar they are from each other. We can calculate this using cosine similarity. Then it ranks similar movies according to their similarity scores and recommends the most relevant movies to the user.

We have created a flask application where in if you type the Title of the movie in the search box it will

give similar results using cosine similarity.

* 1. Front page of the Application:

A screenshot of a cell phone

Description automatically generated

* 1. Results page:

A screenshot of a cell phone

Description automatically generated

The results will show top 10 movies according to cosine similarity. The top result has the highest cosine score.

Github link for code : <https://github.com/Astha7838/movie-recommendation>

Applications -

* Movie classification can be used to classify the movie success, can also extend its application in classifying the TV series, documentaries etc.
* Content-based recommendation can be used to recommend books and articles to the Users

Future scope

* Collecting user reviews from many possible social forums so we can consider the reviews of more people as possible to increase the accuracy of classification.
* Adding tags to the movie like based on genre, production company, cast and crew can increase performance of content-based recommender.
* Implementation of collaboration and hybrid recommender.