**Predicting IMDb Scores Using Machine Learning**

TEAM MEMBER

732521104048 : SUBHASHINI P

**Phase 4 Submission Document**

**Project :** Predicting IMDb Scores



**Introduction:**

* Predicting IMDb scores for movies or TV shows typically involves using machine learning models and features such as cast, crew, genre, user reviews, and more. You can use regression algorithms to build a predictive model.
* The of features and model.
* In this project , we will explore advanced regression techniques to enhance the accuracy and robustness of IMDb scores prediction models.
* Highlight the limitations of traditional linear regression models in capturing complex relationships.
* Emphasize the need for advanced regression techniques like Gradient Boosting and Neural Networks to enchance prediction accuracy.
* quality of your predictions depends on the quality and quantity of data, as well as the choice

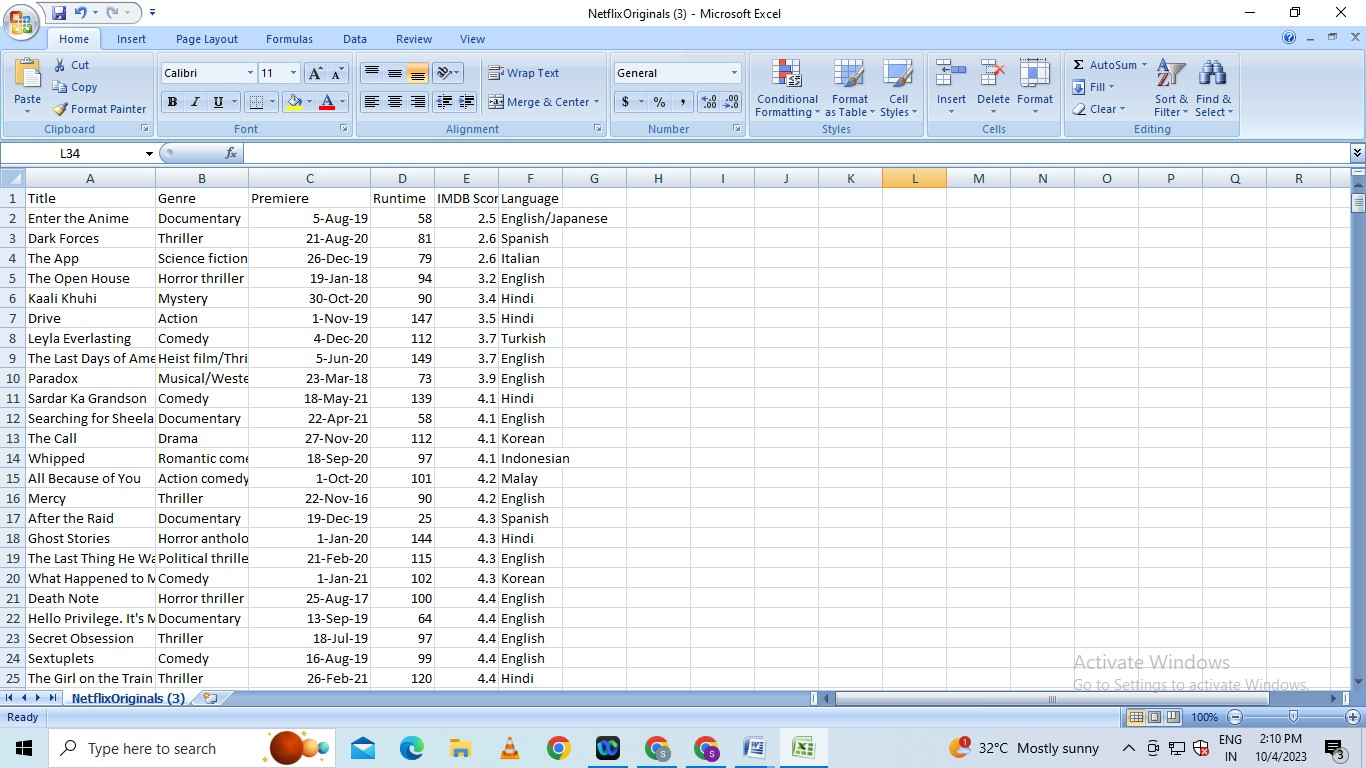
**Content For Project Phase 4:**

Phase 4: Development Part 2

**Data Source :**

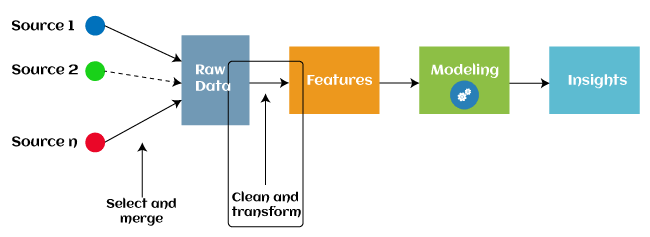
A Good Data for Predicting IMDb Scores using machine learning model should be Accurate , complete , accessible

**Dataset Link : (**[**https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores**](https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores))



**What is Feature Engineering?**

**Feature engineering is the pre-processing step of machine learning, which extracts features from raw data**. It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data. The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.



Since 2016, automated feature engineering is also used in different machine learning software that helps in automatically extracting features from raw data. Feature engineering in ML contains mainly four processes: **Feature Creation, Transformations, Feature Extraction, and Feature Selection.**

**These processes are described as below:**

**feature Creation**: Feature creation is finding the most useful variables to be used in a predictive model. The process is subjective, and it requires human creativity and intervention. The new features are created by mixing existing features using addition, subtraction, and ration, and these new features have great flexibility.

**Transformations**: The transformation step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model. For example, it ensures that the model is flexible to take input of the variety of data; it ensures that all the variables are on the same scale, making the model easier to understand. It improves the model's accuracy and ensures that all the features are within the acceptable range to avoid any computational error.

## Feature Extraction: Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data. The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling. Feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA).

**Feature Selection:** While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features, which is done with the help of feature selection in machine learning. ***"Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features."***

**Steps in Feature Engineering**

The steps for feature engineering vary per different Ml engineers and data scientists. Some of the common steps that are involved in most machine-learning algorithms are:

### 1. Data Cleansing

Data cleansing (also known as data cleaning or data scrubbing) involves identifying and removing or correcting any errors or inconsistencies in the dataset. This step is important to ensure that the data is accurate and reliable.

### 2. Data Transformation

### Data transformation involves converting and scaling variables in the dataset to make them more useful for machine learning. This can include techniques like normalization, standardization, and log transformation.

### 3. Feature Extraction

Feature extraction involves creating new features from the existing variables in the dataset. This can include techniques like principal component analysis (PCA), text parsing, and image processing.

### 4. Feature Selection

Feature selection involves selecting the most relevant features from the dataset for use in machine learning. This can include techniques like correlation analysis, mutual information, and stepwise regression.

### 5. Feature Iteration

Feature iteration involves refining and improving the features based on the performance of the machine learning model. This can include techniques like adding new features, removing redundant features and transforming features in different ways.

Overall, the goal of feature engineering is to create a set of informative and relevant features that can be used to train a machine learning model and improve its accuracy and performance. The specific steps involved in the process may vary depending on the type of data and the specific machine-learning problem at hand.

 useful for reducing the impact of small variations in the data and making it easier to analyze. Binning is the process of grouping continuous features into discrete bins. This can help simplify the feature and reduce noise in the data. Binning can be performed using equal width or equal frequency intervals.

### 6. Feature Split

Feature split is the process of splitting a single variable into multiple variables. This is often done when a variable contains multiple pieces of information that can be more easily analyzed separately. Feature split involves splitting a feature into multiple features. For example, a feature representing a date can be split into year, month, and day features. This can help capture more information about the data and improve the performance of machine learning models.

**What is model training?**

A training model is a dataset that is used to train an ML algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model.

This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model.

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.

Supervised learning is possible when the training data contains both the input and output values. Each set of data that has the inputs and the expected output is called a supervisory signal. The training is done based on the deviation of the processed result from the documented result when the inputs are fed into the model.

Unsupervised learning involves determining patterns in the data. Additional data is then used to fit patterns or clusters. This is also an iterative process that improves the accuracy based on the correlation to the expected patterns or clusters. There is no reference output dataset in this method.

## Why Is Model Training Important?

Model training aims to build the best mathematical representation of the relationship between data and a target (supervised) or among the data itself (unsupervised).

Metrics such as accuracy define how well the model has learned this representation, i.e. they report the model’s performance. The better the model performance, the more benefits using the model in real life will bring. These benefits could include increased revenue, reduced costs, or improved user experience.

Investing time and resources for optimal model training means having access to the right expertise and an appropriate engineering backbone setup within a production-first approach to ML. Such an investment can prove a real differentiator for business success. In fact, leading ML-driven businesses achieve 44% higher productivity and 40% better customer experience among other gains than counterparts.

## How to Train a Machine Learning Model?

The process of training ML models can be divided into four steps.

**Data Set Split for Training and Evaluation**

The training data set is used for model training, and the evaluation set for performance evaluation of the trained model. It is essential that these sets do not intersect and that data in the evaluation sets has not been seen during training in order to ensure an unbiased performance estimate.

**Algorithm Selection**

First, we should select a simpler algorithm than our model’s, or a heuristic, to use as a baseline to compare the final trained model’s performance again.

Then, it is common to select multiple algorithms for training, speed, costs, data size and type, available infrastructure, and desired unless one specific algorithm is clearly the best fit for the use case and data. The most appropriate algorithm(s) to deploy is dependent upon training and inference offline performance.

Some of the most common machine modeling techniques are:

* Linear regression, SVM, random forest, boosted trees, and neural networks\*, for supervised learning
* K-means for unsupervised learning

For deep learning, there is a follow-up phase of “model architecture development” to define the exact layers—optionally on top of pretrained networks—to be used for the final neural network model.

### Hyperparameter Tuning

Each algorithm has a set of default hyperparameters, which is unlikely to be the most performant for any use case and data. We perform hyperparameter tuning on a data subset before training the final model on the complete data set to maximize the performance from each algorithm.

We should also provide a validation set when performing [model tuning](https://www.iguazio.com/glossary/model-tuning/) for evaluation with different hyperparameter selections so as to keep the evaluation set unseen for the final model evaluation.

**Fit and Tune Models**

Now that we’ve split our dataset into training and test sets, and we’ve learned about hyperparameters and cross-validation, we’re ready fit and tune our models. Basically, all we need to do is perform the entire cross-validation loop detailed above on each **set of hyperparameter values** we’d like to try.

The high-level pseudo-code looks like this:

Pseudocode for Tuning Hyperparameters

|  |  |
| --- | --- |
|  | For each algorithm (i.e. regularized regression, random forest, etc.):  For each set of hyperparameter values to try:  Perform cross-validation using the training set.  Calculate cross-validated score. |

|  |  |
| --- | --- |
|  |  |

At the end of this process, you will have a cross-validated score for each set of hyperparameter values… for each algorithm.

For example:



Then, we’ll pick the best set of hyperparameters *within each algorithm*.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Pseudocode for Choosing Hyperparameters   |  |  | | --- | --- | |  | For each algorithm:   Keep the set of hyperparameter values with best cross-validated score.  Re-train the algorithm on the entire training set (without cross-validation). | |

It’s kinda like the Hunger Games… each algorithm sends its own “representatives” (i.e. model trained on the best set of hyperparameter values) to the final selection.

## What Is Model Evaluation?

Model evaluation in machine learning is the process of determining a model’s performance via a metrics-driven analysis. It can be performed in two ways:

* Offline: The model is evaluated after training during experimentation or continuous retraining.
* Online: The model is evaluated in production as part of model monitoring.

**Classification**

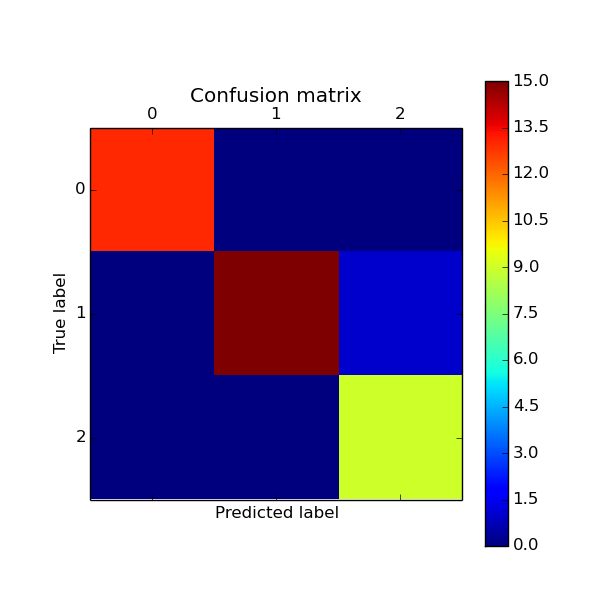
When analyzing classification models choosing the right metric for evaluating machine learning models is of paramount importance. Here is the list of model metrics:

* **Accuracy**is defined as the proportion of correct outcomes to total cases. Strive for a high level of precision.
* **Log loss** is a clinical outcome that shows the classifier’s advantage over a random guess. The log loss quantifies your model’s uncertainty by comparing the probability of its outputs against known values. You want to reduce log loss for the whole model.
* **The Confusion Matrix** is the relationship between the label and the categorization of the model. A confusion matrix has one axis for the expected label and another for the actual label.
* **The area under the curve (AUC)** is calculated by plotting false positives on the x-axis and true positives on the y-axis. This statistic is significant since it gives a single value that allows you to compare different types of models.
* **Precision** is defined as the ratio of correct outcomes to all positive results.
* **The recall** is the percentage of correct answers provided by the model.
* **The F1-score**is another machine learning model evaluation metric. It’s the weighted average of accuracy and recall between 0 and 1, with 1 being the optimal F-score.

There are two critical steps in developing an ML model for a specific problem statement: training and testing. During the training phase, the models adapt from the data and forecast the final results. However, the generated model’s predictions must be correct. Because it can ensure how accurate the outcomes were to execute for the specified problem, testing is the most important step.

**Example of confusion matrix**

Example of confusion matrix usage to evaluate the quality of the output of a classifier on the iris data set. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.



**Script output**:

[[13 0 0]

[ 0 15 1]

[ 0 0 9]]

**Python source code:** [plot\_confusion\_matrix.py](https://scikit-learn.org/0.15/_downloads/plot_confusion_matrix1.py)

print(\_\_doc\_\_)

from sklearn import svm, datasets

from sklearn.cross\_validation import [train\_test\_split](https://scikit-learn.org/0.15/modules/generated/sklearn.cross_validation.train_test_split.html#sklearn.cross_validation.train_test_split)

from sklearn.metrics import [confusion\_matrix](https://scikit-learn.org/0.15/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix)

import matplotlib.pyplot as plt

*# import some data to play with*

iris = datasets.load\_iris()

X = iris.data

y = iris.target

*# Split the data into a training set and a test set*

X\_train, X\_test, y\_train, y\_test = [train\_test\_split](https://scikit-learn.org/0.15/modules/generated/sklearn.cross_validation.train_test_split.html#sklearn.cross_validation.train_test_split)(X, y, random\_state=0)

*# Run classifier*

classifier = [svm.SVC](https://scikit-learn.org/0.15/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)(kernel='linear')

y\_pred = classifier.fit(X\_train, y\_train).predict(X\_test)

*# Compute confusion matrix*

cm = [confusion\_matrix](https://scikit-learn.org/0.15/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix)(y\_test, y\_pred)

print(cm)

*# Show confusion matrix in a separate window*

[plt.matshow](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.matshow)(cm)

[plt.title](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.title)('Confusion matrix')

[plt.colorbar](http://matplotlib.org/api/colorbar_api.html#matplotlib.colorbar)()

[plt.ylabel](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.ylabel)('True label')

[plt.xlabel](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.xlabel)('Predicted label')

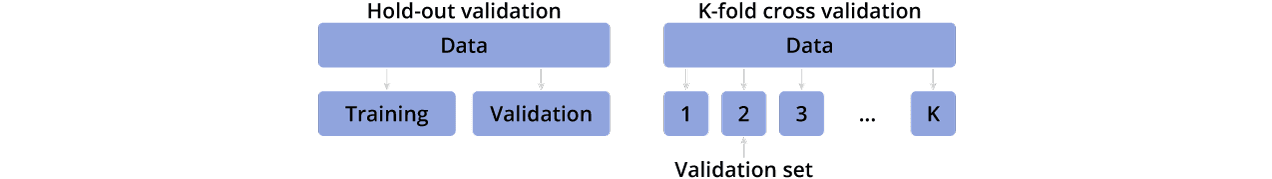
[plt.show](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show)()

**Model Evaluation Techniques**

We have known that the model evaluation is an Integral part in Machine Learning. Initially, the dataset is divided into two types, they are “Training dataset” and “Test dataset”. We build the machine learning model using the training dataset to see the functionality of the model. But we evaluate the designed Model using a test dataset, which consists of unseen or unknown samples of the data that are not used for training purposes. Evaluation of a model tells us how accurate the results were. If we use the training dataset for evaluation of the model, for any instance of the training data it will always show the correct predictions for the given problem with high accuracy measures, in that case our model is not adequately effective to use.

There are two methods that are used to evaluate a model performance. They are

1. Holdout
2. Cross Validation



The Holdout method is used to evaluate the model performance and uses two types of data for testing and training. The test data is used to calculate the performance of the model whereas it is trained using the training data set.  This method is used to check how well the machine learning model developed using different algorithm techniques performs on unseen samples of data. This approach is simple, flexible and fast.

Cross-validation is a procedure of dividing the whole dataset into data samples, and then evaluating the machine learning model using the other samples of data to know accuracy of the model. i.e., we train the model using a subset of data and we evaluate it with a complementary data subset. We can calculate cross validation based on the following 3 methods, namely

1. Validation
2. Leave one out cross validation (LOOCV)
3. K-Fold Cross Validation

In the method of validation, we split the given dataset into 50% of training and 50% for testing purpose. The main drawback in this method is that the remaining 50% of data that is subjected to testing may contain some crucial information that may be lost while training the model. So, this method doesn’t work properly due to high bias.

In the method of LOOCV, we train all the datasets in our model and leave a single data point for testing purpose. This method aims at exhibiting lower bias, but there are some chances that this method might fail because, the data-point that has been left out may be an outlier in the given data; and in that case we cannot produce better results with good accuracy.

K-fold cross validation is a popular method used for evaluation of a Machine Learning model. It works by splitting the data into k-parts. Each split of the data is called a fold. Here we train all the k subsets of data to the model, and then we leave out one (k-1) subset to perform evaluation on the trained model. This method results in high accuracy and produces data with less bias.

**Conclusion**

### Feature engineering, like so many things in data science, is an iterative process. Investigating, experimenting, and doubling back to make adjustments are crucial. The insights you stand to gain into the structure of your data and the potential improvements to model performance are usually well worth the effort.

We have seen that the machines can beat human champions in games such as Chess, AlphaGO, which are considered very complex. You have seen that machines can be trained to perform human activities in several areas and can aid humans in living better lives. Machine Learning can be a Supervised or Unsupervised.

Model evaluation and selection play a pivotal role in the success of any machine learning project. Evaluating models using appropriate metrics and techniques enables us to identify the best-performing models and build robust, accurate systems. By leveraging techniques like cross-validation, grid search, and randomized search, data scientists can efficiently explore model options and arrive at optimal hyperparameters.