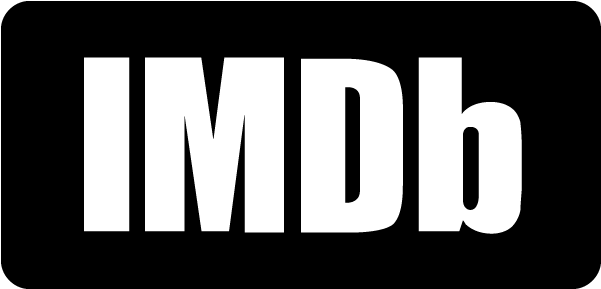
**Predicting IMDb Scores Using Machine Learning**

TEAM MEMBER

732521104305 : POORANA G

**Phase 3 Submission Document**

**Project :** Predicting IMDb Scores



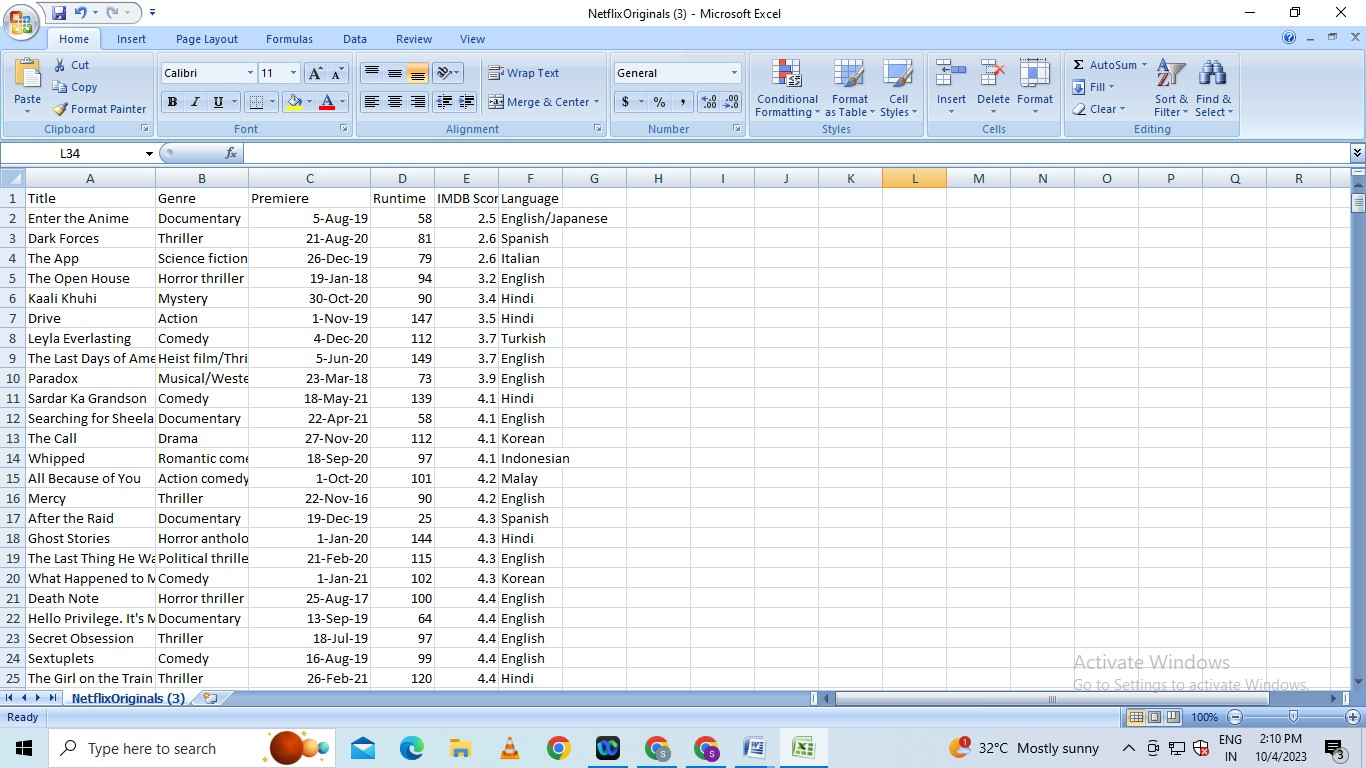
Introduction:

* 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 quality of your predictions depends on the quality and quantity of data, as well as the choice 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.

**Content For Project Phase 3 :**

* this part you will begin building your project by loading and preprocessing the dataset.
* Begin building the IMDb score prediction model by loading and preprocessing the dataset.
* Load the movie dataset and preprocess the data for analysis.

**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))

**Data loading :**

Data Loading is defined as copying data from one electronic file or database into another. Data loading implies converting from one format into another; for example, from one type of production database into a decision support database from a different vendor.

This article will give a Comprehensive Guide on Data Loading

## Table of Contents :

* [Introduction to Data Loading](https://hevodata.com/learn/data-loading/#Introduction-to-Data-Loading)
* [Challenges with Data Loading](https://hevodata.com/learn/data-loading/#Challenges-with-Data-Loading)
* [Methods for Data Loading](https://hevodata.com/learn/data-loading/#Methods-for-Data-Loading)
* [Types of Data Loading](https://hevodata.com/learn/data-loading/#Types-of-Data-Loading-)
* [Data Loading: Refresh versus Update](https://hevodata.com/learn/data-loading/#Refresh-versus-Update)
* [Cloud-Based ETL tools](https://hevodata.com/learn/data-loading/#Cloud-Based-ETL-Tools)
* [Conclusion](https://hevodata.com/learn/data-loading/#Conclusion)

## Introduction to Data Loading :

* Data loading defines the LOAD component of the ETL process. ETL stands for Extraction, Transformation, and Load. Extraction deals with the retrieval and combining of data from multiple sources.
* Transformation deals with cleaning and formatting of the Extracted Data. Data Loading deals with data getting loaded into a storage system, such as a cloud data warehouse.
* ETL aids in the data integration process that standardizes diverse data types to make them available for querying, manipulation, or reporting for many different individuals and teams.
* Because today’s organizations are increasingly dependent upon their own data to make smarter, faster business decisions, ETL needs to be scalable and streamlined to provide the most benefit.
* Data loading is quite simply the process of packing up your data and moving it to a designated data warehouse.
* It is at the beginning of this transitory phase where you can begin planning a roadmap, outlining where you would like to move forward with your data and how you would like to use it.
* Data Loading is the ultimate step in the ETL process. In this step, the extracted data and the transformed data are loaded into the target database.
* To make the data loading efficient, it is necessary to index the database and disable the constraints before loading the data.
* All three steps in the ETL process can be run parallel. Data extraction takes time and therefore the second phase of the transformation process is executed simultaneously.
* This prepares the data for the third stage that is data loading. As soon as some data is ready, data loading is done without waiting for the previous steps to be completed.

**Simplify Data Analysis with Hevo’s No-code Data Pipeline :**

* Hevo Data, a No-code Data Pipeline helps to load data from any data source such as Google Search Console, Databases, SaaS applications, Cloud Storage, SDKs, and Streaming Services and simplifies the ETL process.
* It supports [100+ data sources](https://hevodata.com/integrations/) (**including 30+ free data sources**) and is a 3-step process by just selecting the data source, providing valid credentials, and choosing the destination. Hevo not only loads the data onto the desired Data Warehouse/destination but also enriches the data and transforms it into an analysis-ready form without having to write a single line of code.
* Its completely automated pipeline offers data to be delivered in real-time without any loss from source to destination. Its fault-tolerant and scalable architecture ensure that the data is handled in a secure, consistent manner with zero data loss and supports different forms of data. The solutions provided are consistent and work with different BI tools as well.

**Check out why Hevo is the Best :**

* **Secure**: Hevo has a fault-tolerant architecture that ensures that the data is handled in a secure, consistent manner with zero data loss.
* **Schema Management**: Hevo takes away the tedious task of schema management & automatically detects the schema of incoming data and maps it to the destination schema.
* **Minimal Learning**: Hevo, with its simple and interactive UI, is extremely simple for new customers to work on and perform operations.
* **Hevo Is Built To Scale**: As the number of sources and the volume of your data grows, Hevo scales horizontally, handling millions of records per minute with very little latency.
* **Incremental Data Load**: Hevo allows the transfer of data that has been modified in real-time. This ensures efficient utilization of bandwidth on both ends.
* **Live Support**: The Hevo team is available round the clock to extend exceptional support to its customers through chat, email, and support calls.
* **Live Monitoring**: Hevo allows you to monitor the data flow and check where your data is at a particular point in time.

## Challenges with Data Loading :

Many ETL solutions are cloud-based, which accounts for their speed and scalability. But large enterprises with traditional, on-premise infrastructure and data management processes often use custom-built scripts to collect and perform data loading on their own data into storage systems through customized configurations. This can:

* **Slow down analysis.** Each time a data source is added or changed, the system has to be reconfigured, which takes time and hampers the ability to make quick decisions.
* **Increase the likelihood of errors.** Changes and reconfigurations open up the door for human error, duplicate or missing data, and other problems.
* **Require specialized knowledge.** In-house IT teams often lack the skill (and bandwidth) needed to code and monitor ETL functions themselves.
* **Require costly equipment.** In addition to investment in the right human resources, organizations have to purchase, house, and maintain hardware and other equipment to run the process on-site.
* **Unorganized Data:** Loading your data can become unorganized very fast. For ETL voyagers, common roadblocks that many encounters early on can be resolved with proper planning and delivery.
* **Universal formatting:** Before you begin loading your data, make sure that you identify where it is coming from and where you want to go.
* **Loss of data:** Tracking the status of all data is critical for a smooth loading process.
* **Speed:** Although it’s exciting to be closer to your final destination, do not rush through this phase. Errors are most likely to occur during this time.

## Methods for Data Loading :

* Since data loading is part of the larger ETL process, organizations need a proper understanding of the types of ETL tools and methods available, and which one(s) work best for their needs, budget, and structure.
* In the process of Data Loading the data is physically moved to the data warehouse. The Data Loading takes place within a “load window. The tendency is close to real-time updates for data warehouses as warehouses are growing used for operational applications.

**Cloud-based :**

* ETL tools in the cloud are built for speed and scalability, and often enable real-time data processing. They also include the ready-made infrastructure and expertise of the vendor, who can advise on best practices for each organization’s unique setup and needs.

**Batch processing :**

* ETL tools that work off batch processing move data at the same scheduled time every day or week. It works best for large volumes of data and for organizations that don’t necessarily need real-time access to their data.

**Open-source:**

* Many open-source ETL tools are quite cost-effective as their codebase is publicly accessible, modifiable, and shareable. While a good alternative to commercial solutions, these tools can still require some customization or hand-coding.

## Types of Data Loading :

* Soon after your departure from the extraction phase, you will be faced with the decision of which loading process that you would like to deploy.
* The **data** **loading** process is the physical movement of the data from the computer systems storing the source database(s) to that which will store the data warehouse database.
* The entire process of transferring data to a data warehouse repository is referred to in the following ways:
* **Full Load :**  This is where all of your data is selected, moved in bulk, and then replaced by new data. Although it is not as complex to navigate through, loading time is much slower. With the overwhelming amount of data being moved at once, it is much easier for data to get lost within the big move.
* **Incremental Load:**  This is where you are moving new data in intervals. Due to its intricate nature, delivery time is much faster than its counterpart. However, this speed comes at a cost. Incremental loads are more likely to encounter problems due to the nature of having to manage them as individual batches rather than one big group.
* **Incremental Load** :  Periodically applies ongoing changes as per the requirement. After the data is loaded into the data warehouse database, verify the referential integrity between the dimensions and the fact tables to ensure that all records belong to the appropriate records in the other tables. The DBA must verify that each record in the fact table is related to one record in each dimension table that will be used in combination with that fact table.
* **Initial Load**: For the very first time loading all the data warehouse tables.
* **Full Refresh**: Deleting the contents of a table and reloading it with fresh data.

## Data Loading: Refresh versus Update

After the initial load, the data warehouse needs to be maintained and updated and this can be done by the following two methods:

* **Update**-application of incremental changes in the data sources.
* **Refresh**-complete reloads at specified intervals.

**Future Work :**

We will discuss potential avenues for future work, such as incorporating additional data sources (e.g., real-time economic indicators), exploring deep learning models for prediction, or expanding the project into a web application with more features and interactivity.

## Data Preprocessing in Machine Learning :

* The real-world data needs processing before feeding it to a machine-learning model. We know that 80% of a data scientist’s time goes into data preprocessing and 20% of the time into model building.
* The statement is true because if we feed unclean, noisy data to the model, it will either fail to process it or generate erroneous output. Hence, data preprocessing is a crucial step in machine learning. This article will use examples to teach different data preprocessing steps, like data cleaning, transformation, quality assessment, and transformation.

**Why do We Need Data Preprocessing** :

* **Improving Data Quality**: Data preprocessing is essential for enhancing the quality of data by handling inconsistencies, inaccuracies, and errors, which is critical for ensuring reliable and robust analytics.
* **Dealing with Missing Values**: Data preprocessing includes techniques like imputation that are critical for dealing with missing data effectively, as datasets often have missing values which can significantly hinder the performance of machine learning models.
* **Normalizing and Scaling**: Data preprocessing helps in normalizing or scaling features, which is especially important for algorithms that are sensitive to the scale of the input. This ensures that all the features are on a comparable scale, which is crucial for the accurate performance of many machine learning algorithms.
* **Handling Outliers**: Through data preprocessing, outliers can be identified and managed appropriately. This is important as outliers can have a disproportionate effect on the modeling process and can lead to misleading results.
* **Dimensionality Reduction**: Data preprocessing includes techniques such as Principal Component Analysis (PCA) for reducing the number of input features, which not only helps in improving the performance of models but also makes the dataset more manageable and computationally efficient.

**Steps in Data Preprocessing :**

Data preprocessing consists of several steps. In this article, we will cover the following items:

### 1. ****Imputing Missing Values:****

* The missing value is introduced in data for several reasons, including lost data in the channel or sometimes the customer denies providing information(people are reluctant to share their earnings over a survey).

Mean or median is used to impute the value for a numerical feature.

Again, though, the median is preferred as the mean is influenced by the outliers and skewness in the data. The most occurring value, i.e., by mode, is favored for categorical features. When more than a certain percentage(say 40%) of data is missing for a particular column, it's preferred to discard that column instead of imputing it.

### 2. ****Removal of Outliers:****

BoxPlot helps identify outliers present in data. Next, all the values above and below three standard deviations are considered an outlier and can be removed. It can be achieved using one-line code in python.

df = df[df.apply(lambda x :(x-x.mean()).abs()<(3\*x.std()) ).all(1)]

### 3. ****Data Normalisation:****

Normalization is transforming the entire data range so that the data has a mean of 0 and a standard deviation of 1. It does not distort the difference between sample points. It helps optimization functions converge faster. Some standard normalization techniques are discussed below:

* **Min-Max normalization:** In this data normalization technique, a linear transformation is performed on the original data. The minimum and maximum values from data are fetched, and each value is replaced according to the following formula. �′=(�−����)/(����−����)*v*′=(*v*−*minA*​)/(*maxA*​−*minA*​)
  + Where, A is the attribute data.
  + ����*minA*​ and ����*maxA*​ are A's minima and maximum absolute values, respectively.
  + v' is the new value of each entry in the data.
  + v is the old value of each entry in the data.
* **Z-score normalization or Zero mean normalization:** In this technique, values are normalized based on a mean and standard deviation of the data A. The formula used is �′=(�−�ˉ)/��*v*′=(*v*−*A*ˉ)/*σA*​.
  + Here, v', and v is the new and old of each entry in data, respectively.
  + ��*σA*​, �ˉ*A*ˉ is the standard deviation and mean of A, respectively.

### 4. ****Encoding of Categorical Variable:****

Categorical feature encoding is another essential step in data preprocessing as the machine learning model cannot consume categorical features in their raw form. Categorical varaibles are of two types:

* **Ordinal categorical variables:** These variables can be ordered (grades in an exam. Here, one can say that C<B<A). For ordinal categorical features, label encoding is preferred. Label Encoding converts the labels into a numeric form to convert them into a machine-readable format. Here, grades can be mapped to numeric values as follows-> {C:1, B:2, A:3}
* **Nominal categorical variables:** These variables can’t be ordered (colors of a car). One Hot Encoding(OHE) is preferred in such a situation. In OHE, Each categorical value is converted into a new categorical column and assigned a binary value of 1 or 0 to those columns. The figure below will help to understand the function of OHE.

### 5. ****Train Validation and Test Split:****

After the data preparation, the next step is dividing the data into three parts: training, validation, and test data. First, training data is used to build the model. The model identifies the hidden patterns in this dataset and generates model parameters.

Next, the model is validated on validation data. It helps to determine how the model is performing. Together, Validation and Training accuracy helps to identify any overfitting or underfitting in the data. Validation data also helps to tune the model hyper-parameters. Finally, test data is the unseen data the model uses to predict the output.

## Data Preprocessing Examples

Data preprocessing is a crucial step in the data mining process. Below, we are going to show you three simple examples of data preprocessing using Python:

1. Handling Missing Values
2. Scaling Features
3. Encoding Categorical Variables

Let's assume we have a small dataset of people's information that includes their age, height, weight, and country.

**Example Data:**

| **Age** | **Height (cm)** | **Weight (kg)** | **Country** |
| --- | --- | --- | --- |
| 25 | 175 | 72 | USA |
| 32 |  | 80 | Canada |
|  | 168 | 56 | India |

As you can see, there are missing values in this dataset.

### Example 1: Handling Missing Values

In this example, we will replace the missing values in the 'Age' column with the average age, and in the 'Height' column with the average height.

import pandas as pd

import numpy as np

*# Sample Data*

data = {

'Age': [25, 32, np.nan],

'Height': [175, np.nan, 168],

'Weight': [72, 80, 56],

'Country': ['USA', 'Canada', 'India']

}

*# Create a DataFrame*

df = pd.DataFrame(data)

*# Fill missing values*

df['Age'].fillna(df['Age'].mean(), inplace=True)

df['Height'].fillna(df['Height'].mean(), inplace=True)

print(df)

**Output:**

Age Height Weight Country

0 25.0 175.0 72 USA

1 32.0 171.5 80 Canada

2 28.5 168.0 56 India

### Example 2: Scaling Features

Here, we will scale the features 'Height' and 'Weight' between 0 and 1.

from sklearn.preprocessing import MinMaxScaler

*# Initialize the Scaler*

scaler = MinMaxScaler()

*# Scaling Height and Weight*

df[['Height', 'Weight']] = scaler.fit\_transform(df[['Height', 'Weight']])

print(df)

**Output:**

Age Height Weight Country

0 25.0 1.000000 0.545455 USA

1 32.0 0.500000 1.000000 Canada

2 28.5 0.000000 0.000000 India

### Example 3: Encoding Categorical Variables

Finally, we will convert the 'Country' column from categorical to numerical form.

*# Encoding Country column*

df = pd.get\_dummies(df, columns=['Country'], drop\_first=True)

print(df)

**Output:**

Age Height Weight Country\_India Country\_USA

0 25.0 1.000000 0.545455 0 1

1 32.0 0.500000 1.000000 0 0

2 28.5 0.000000 0.000000 1 0

In this final output, the 'Country' column has been encoded into numerical features. Each country is represented as a separate column with binary values (0 or 1).

## Data Preprocessing: Best Practices

A machine learner practitioner should follow these things about data preprocessing:

* Understanding data, its domain, and each feature's meaning are crucial.
* Visualize data with the help of statistical and visualization tools. It will make data understanding easier.
* Try performing a data quality assessment regarding the number of duplicates, percentage of missing values, and outliers in the data.
* Drop the fields which are intuitively less meaningful. Use feature selection and dimension reduction methods to reduce the dimension.
* Perform feature engineering on remaining attributes and figure out the important done.

## Conclusion :

* we have discussed what data preprocessing is in machine learning, why we need to perform data preprocessing in machine learning, different steps of data preprocessing, etc..
* Some examples of data preprocessing and best practices for data processing are also delivered here for a better understanding of readers.
* Data visualization is not discussed here. Instead, it will be delivered in the following article.