**Predicting IMDb Scores Using Machine Learn**

***Phase 4 Submission Document***

***Project : Predicting IMDb Scores***



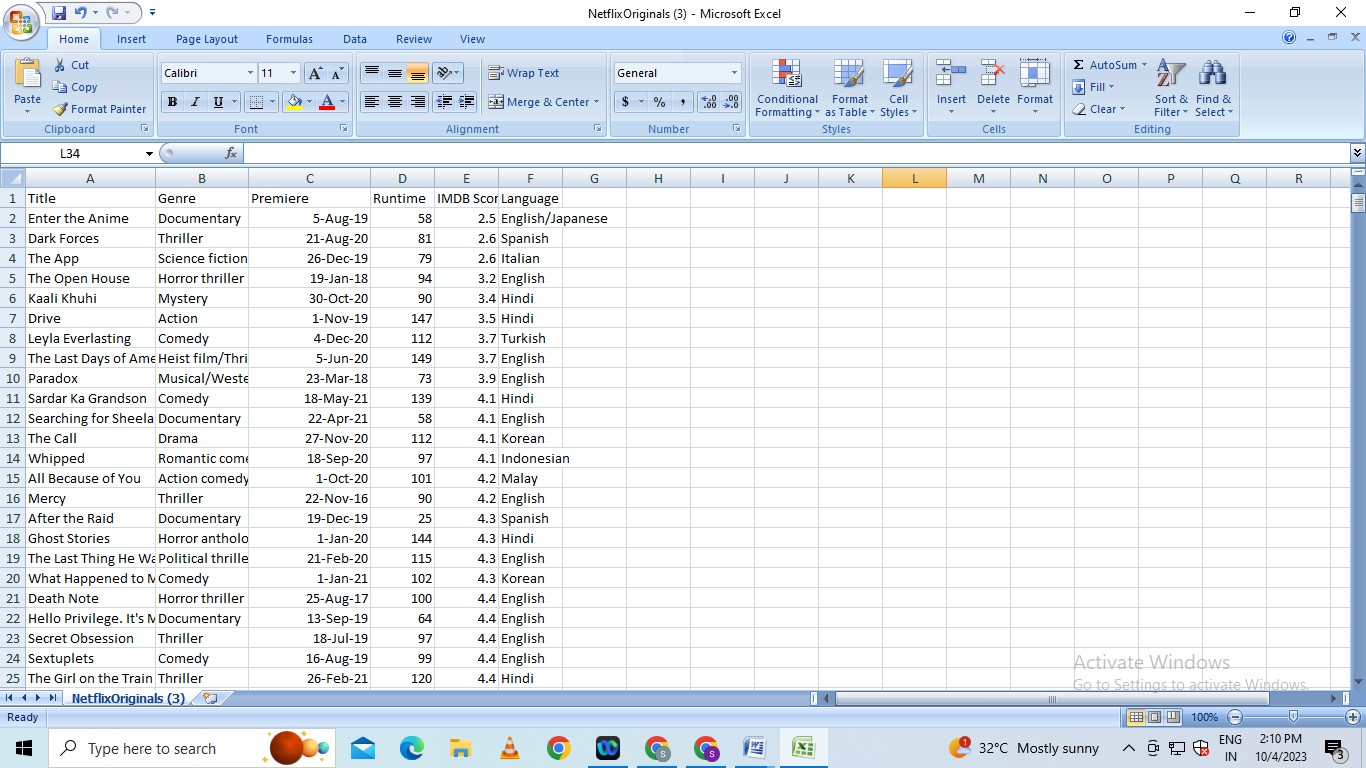
**Introduction**

* 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 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.

**Data Source**

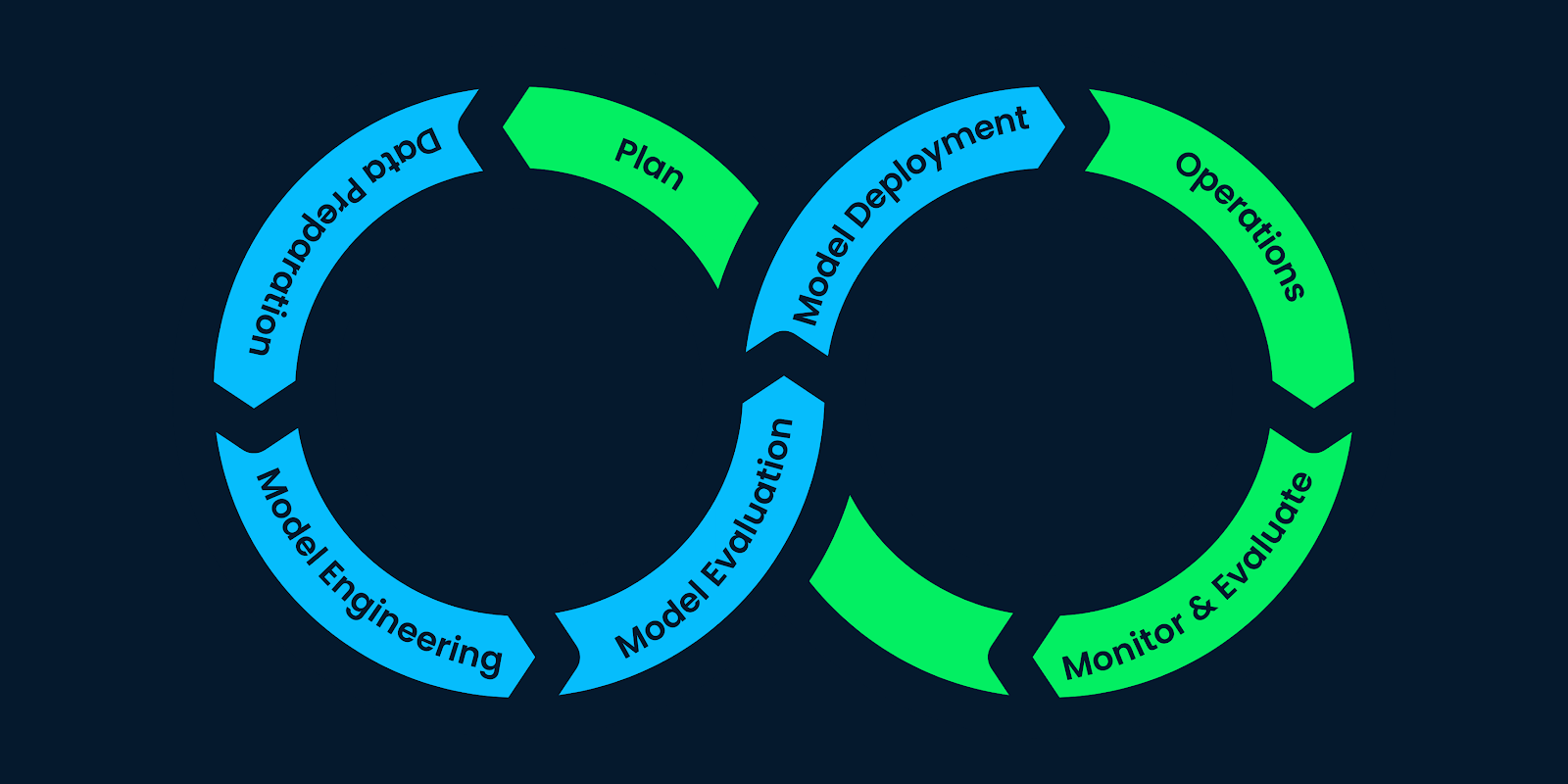
A Good Data for Predicting IMDb Scores using machine learning model should be Accurate , complete , accessible

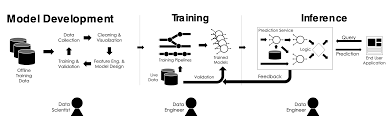
**Dataset Link : (**[**https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores**](https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores)

**What is Feature Engineering?**Feature engineering refers to manipulation — addition, deletion, combination, mutation — of your data set to improve machine learning model training, leading to better performance and greater accuracy. Effective feature engineering is based on sound knowledge of the business problem and the available data sources.

Creating new features gives you a deeper understanding of your data and results in more valuable insights. When done correctly, feature engineering is one of the most valuable techniques of [data science](https://domino.ai/data-science-dictionary/data-science/), but it is also one of the most challenging. A common example of feature engineering is when your doctor uses your body mass index (BMI). BMI is calculated from both body weight and height, and serves as a surrogate for a characteristic that is very hard to accurately measure: the proportion of lean body mass.

**Feature Engineering in ML Lifecycle**





Some common types of feature engineering include:

* **Scaling and normalization** means adjusting the range and center of data to ease learning and improve the interpretation of the results.
* **Filling missing values** implies filling in null values based on expert knowledge, heuristics, or by some [machine learning](https://domino.ai/data-science-dictionary/machine-learning/) techniques. Real-world datasets can be missing values due to the difficulty of collecting complete datasets and because of errors in the data collection process.
* **Feature selection** means removing features because they are unimportant, redundant, or outright counterproductive to learning. Sometimes you simply have too many features and need fewer.
* **Feature coding** involves choosing a set of symbolic values to represent different categories. Concepts can be captured with a single column that comprises multiple values, or they can be captured with multiple columns, each of which represents a single value and has a true or false in each field. For example, feature coding can indicate whether a particular row of data was collected on a holiday. This is a form of feature construction.
* **Feature construction** creates a new feature(s) from one or more other features. For example, using the date you can add a feature that indicates the day of the week. With this added insight, the algorithm could discover that certain outcomes are more likely on a Monday or a weekend.
* **Feature extraction** means moving from low-level features that are unsuitable for learning — practically speaking, you get poor testing results — to higher-level features that are useful for learning. Often feature extraction is valuable when you have specific data formats — like images or text — that have to be converted to a tabular row-column, example-feature format.

**What is Model Training?**

Model training is at the heart of the data science development lifecycle where the data science team works to fit the best weights and biases to an algorithm to minimize the loss function over prediction range. Loss functions define how to optimize the ML algorithms. A data science team may use different types of loss functions depending on the project objectives, the type of data used and the type of algorithm.

When a supervised learning technique is used, model training creates a mathematical representation of the relationship between the data features and a target label. In unsupervised learning, it creates a mathematical representation among the data features themselves.

**Importance of Model Training**

Model training is the primary step in machine learning, resulting in a working model that can then be validated, tested and deployed. The model’s performance during training will eventually determine how well it will work when it is eventually put into an application for the end-users.

Both the quality of the training data and the choice of the algorithm are central to the model training phase. In most cases, training data is split into two sets for training and then validation and testing.

The selection of the algorithm is primarily determined by the end-use case. However, there are always additional factors that need to be considered, such as algorithm-model complexity, performance, interpretability, computer resource requirements, and speed. Balancing out these various requirements can make selecting algorithms an involved and complicated process.

**How To Train a Machine Learning Model**

Training a model requires a systematic, repeatable process that maximizes your utilization of your available training data and the time of your data science team. Before you begin the training phase, you need to first determine your problem statement, access your data set and clean the data to be presented to the model.

In addition to this, you need to determine which algorithms you will use and what parameters (hyperparameters) they will run with. With all of this done, you can split your dataset into a training set and a testing set, then prepare your model algorithms for training.

**Split the Dataset**

Your initial training data is a limited resource that needs to be allocated carefully. Some of it can be used to train your model, and some of it can be used to test your model – but you can’t use the same data for each step. You can’t properly test a model unless you have given it a new data set that it hasn’t encountered before. Splitting the training data into two or more sets allows you to train and then validate the model using a single source of data. This allows you to see if the model is overfit, meaning that it performs well with the training data but poorly with the test data.

A common way of splitting the training data is to use cross-validation. In [10-fold cross-validation](https://domino.ai/blog/guide-to-building-models-with-cross-validation), for example, the data is split into ten sets, allowing you to train and test the data ten times.

**Select Algorithms to Test**

In machine learning, there are thousands of algorithms to choose from, and there is no sure way to determine which will be the best for any specific model. In most cases, you will likely try dozens, if not hundreds, of algorithms in order to find the one that results in an accurate working model.

**Tune the Hyperparameters**

Hyperparameters are the high-level attributes set by the data science team before the model is assembled and trained. While many attributes can be learned from the training data, they cannot learn their own hyperparameters.

As an example, if you are using a [regression algorithm](http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/153-penalized-regression-essentials-ridge-lasso-elastic-net/), the model can determine the regression coefficients itself by analyzing the data. However, it cannot dictate the strength of the penalty it should use to regularize an overabundance of variables. As another example, a model using the random forest technique can determine where decision trees will be split, but the number of trees to be used needs to be tuned beforehand.

**Fit and Tune Models**

Now that the data is prepared and the model’s hyperparameters have been determined, it’s time to start training the models. The process is essentially to loop through the different algorithms using each set of hyperparameter values you’ve decided to explore.

Next, select another set of hyperparameter values you want to try for the same algorithm, cross-validate it again and calculate the new score. Once you have tried each hyperparameter value, you can repeat these same steps for additional algorithms.

Think of these trials as track and field heats. Each algorithm has demonstrated what it can do with the different hyperparameter values. Now you can select the best version from each algorithm and send them on to the final competition.

**Choose the Best Model**

Now it’s time to test the best versions of each algorithm to determine which gives you the best. Once the testing is done, you can compare their performance to determine which are the better models. The overall winner should have performed well (if not the best) in training as well as in testing. It should also perform well on your other performance metrics (like speed and [empirical loss](https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss)), and – ultimately – it should adequately solve or answer the question posed in your problem statement.

What is Model Evaluation?

Model evaluation is the process of using different evaluation metrics to understand a machine learning model’s performance, as well as its strengths and weaknesses. Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

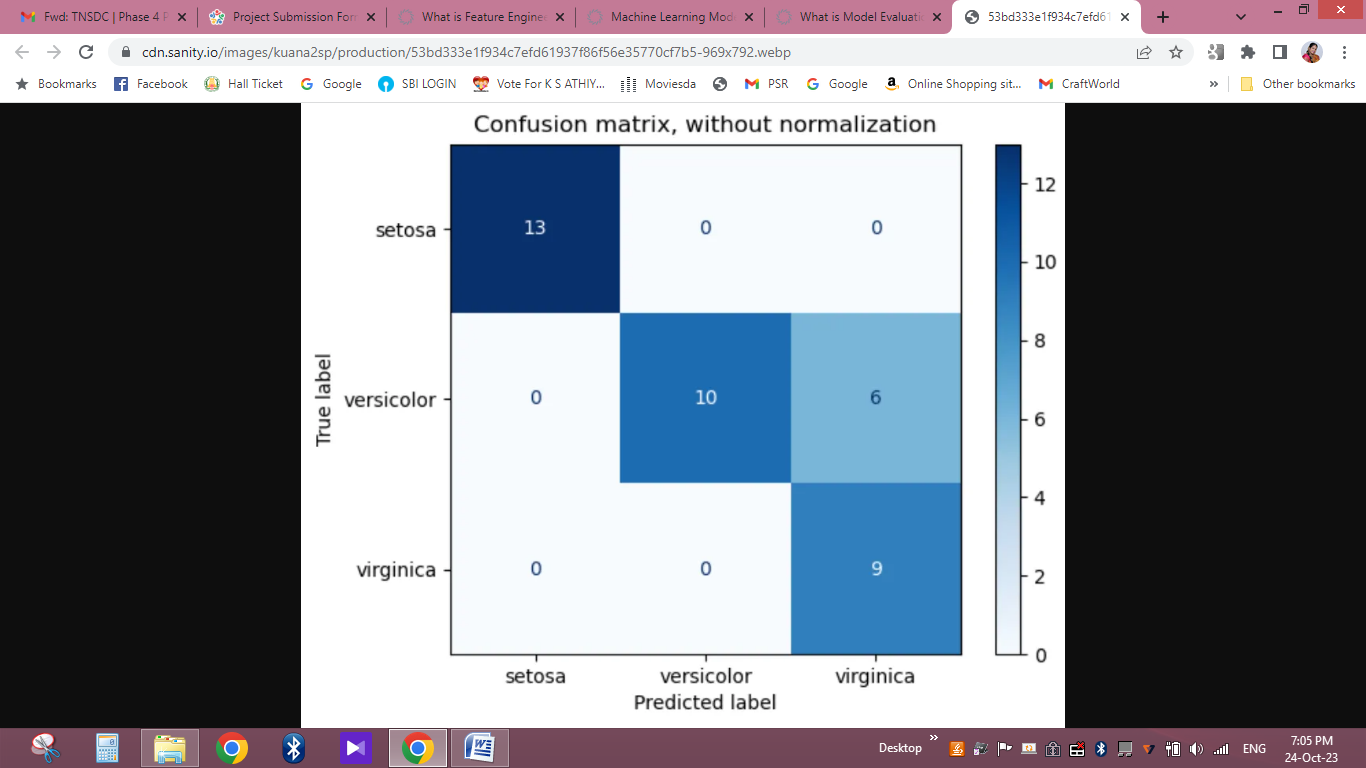
To understand if your model(s) is working well with new data, you can leverage a number of evaluation metrics.

Classification

The most popular metrics for measuring classification performance include accuracy, precision, confusion matrix, log-loss, and AUC (area under the ROC curve).

* **Accuracy** measures how often the classifier makes the correct predictions, as it is the ratio between the number of correct predictions and the total number of predictions.
* **Precision** measures the proportion of predicted Positives that are truly Positive. Precision is a good choice of evaluation metrics when you want to be very sure of your prediction. For example, if you are building a system to predict whether to decrease the credit limit on a particular account, you want to be very sure about the prediction or it may result in customer dissatisfaction.
* The **confusion matrix** (or confusion table) shows a more detailed breakdown of correct and incorrect classifications for each class. Using a confusion matrix is useful when you want to understand the distinction between classes, particularly when the cost of misclassification might differ for the two classes, or you have a lot more test data on one class than the other. For example, the consequences of making a false positive or false negative in a cancer diagnosis are very different.

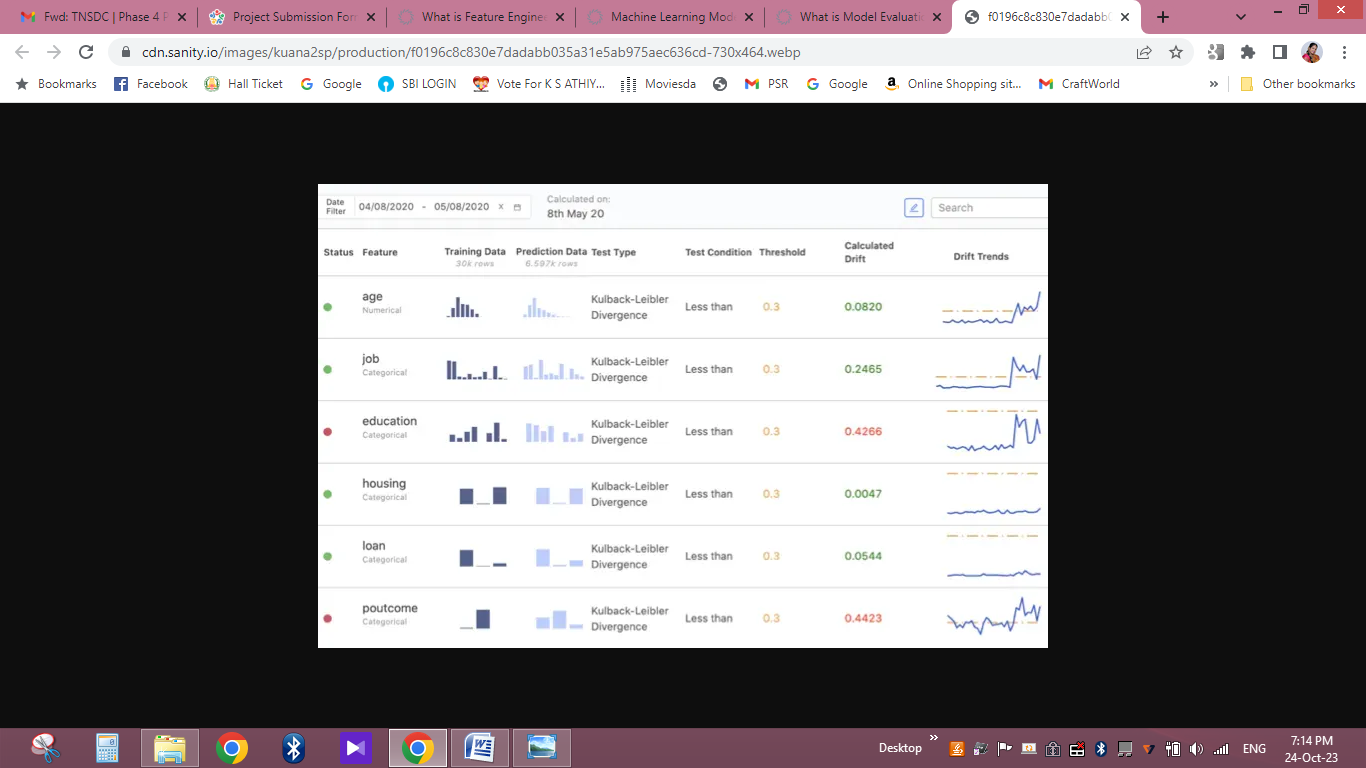
Example of Confusion Matrix on Iris Flower Dataset



* **Log-loss** (logarithmic loss) can be used if the raw output of the classifier is a numeric probability instead of a class label. The probability can be understood as a gauge of confidence, as it is a measurement of accuracy.
* **AUC** (Area Under the ROC Curve) is a performance measurement for classification problems at various thresholds settings. It tells how much a model is capable of distinguishing between classes. The higher the AUC, better the model is at predicting when a 0 is actually a 0 and a 1 is actually a 1. Similarly, the higher the AUC, the better the model is at distinguishing between patients with a disease and with no disease.

Other popular metrics exist for regression models, like R Square, Adjusted R Square, MSE (Mean Squared Error), RMSE (Root Mean Squared Error), and MAE (Mean Absolute Error).

**Domino Model Monitor**

Machine Learning Operations teams often monitor multiple models at once by checking model predictions, checking (input) data drift, and checking concept drift. [Model monitoring](https://domino.ai/data-science-dictionary/model-monitoring/) tools, like Domino Model Monitor, areavailable to facilitate model evaluation.

Conclusion

Feature engineering is an important stage when building machine learning models, and getting this stage right can ensure ML models are more accurate, use less computational resources, and process at higher speeds.

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.

Evaluation metrics are quantitative measures used to assess the performance and effectiveness of a statistical or machine learning model. These metrics provide insights into how well the model is performing and help in comparing different models or algorithms.