**TED Talk Views Prediction**

**Nargis Nasreen**

TEDx is a grassroots initiative, created in the spirit of TED’s overall mission to research and discover “ideas worth spreading.” TEDx brings the spirit of TED to local communities around the globe through TEDx events. These events are organized by passionate individuals who seek to uncover new ideas and to share the latest research in their local areas that spark conversations in their communities. TEDx events include live speakers and recorded TED Talks, and are organized independently under a free license granted by TED. These events are not controlled by TED, but event organizers agree to abide by format, and are offered guidelines for curation, speaker coaching, event organizing and more. They learn from each other. More than 3000 events are now held annually.

Here the main objective is to build a predictive model, which could help in predicting the views of the videos uploaded on the TEDx website.

We have:-

* Number of instances: 4,005 and
* Number of attributes: 19

The dataset contains features like:

* **talk\_id**: Talk identification number provided by TED
* **title**: Title of the talk
* **speaker\_1**: First speaker in TED's speaker list
* **all\_speakers**: Speakers in the talk
* **occupations**: Occupations of the speakers
* **about\_speakers**: Blurb about each speaker
* **recorded\_date**: Date the talk was recorded
* **published\_date**: Date the talk was published to TED.com
* **event**: Event or medium in which the talk was given
* **native\_lang**: Language the talk was given in
* **available\_lang**: All available languages (lang\_code) for a talk
* **comments**: Count of comments
* **duration**: Duration in seconds
* **topics**: Related tags or topics for the talk
* **related\_talks**: Related talks (key='talk\_id',value='title')
* **url**: URL of the talk
* **description**: Description of the talk
* **transcript**: Full transcript of the talk

**Target Variable : Views**

* **Views**: The number of views for each talk

**Ted Talks that are constantly using Machine Learning algorithms to optimize the number of views the videos receive. They do this by understanding the dependency of views with other relevant features to increase the viewers satisfaction by recommending videos according to the average views that have been affected by features like topics, comments, etc.**

* **Exploratory Data Analysis**

After loading the dataset I looked for duplicate values in the ‘talk\_id’ column. There were none. So We performed EDA by comparing our target variable that is Views with other independent variables. This process helped me figuring out various aspects and relationships among the target and the independent variables. It gave me a better idea of which feature behaves in which manner compared to the target variable.

* **Numerical Variables:**
* Talk\_id
* Views
* Comments
* duration
* **Textual Variables:**
* Title
* Speaker\_1
* Recorded\_date
* Published\_date
* Event
* Native\_lang
* Url
* Description
* **Dictionaries:**
* Speakers
* Occupations
* About\_speakers
* Related\_talks
* **List:**
* topics

Out of all the continuous variables, ‘views’ is the target variable.

The target variable ‘views’ was a skewed variable.

The other continuous variables have distributions as:

* Comments
* Duration

All of the data had very skewed continuous variable distributions.

* **Null values Treatment**

The dataset contains around 400 null values which might tend to disturb our mean absolute score hence I have performed KNN imputer for numerical features and replaced categorical features nan values with the value ‘Other’. I chose to impute nan values and not drop them due to the size of the data set

* **Encoding of categorical columns**

I used Target Encoding for replacing the values of categorical variables with the mean of the views. This was done to not increase the dimensions to the data set while also keeping the relationship of variables with views into consideration.

**Feature engineering**

Feature engineering is the process of tansforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved accuracy on unseen data.

And I did target encoding here to replace a categorical value with the mean of the target variable. Any non-categorical columns are automatically dropped by the target encoder model. ... This can help improve machine learning accuracy since algorithms tend to have a hard time dealing with high cardinality columns.

* **Feature Selection**

For Feature Selection I have introduced new numerical features from the categorical features,combined features and also I have used f\_regression in which we have taken the features with the maximum f-scores.

* **Outlier Treatment**

I have done outlier treatment on variables like duration and occupation. This was done by replacing outliers with the extreme values at the first and third quartiles. I have done outlier treatment to prevent high errors that were influenced by outliers.

For modelling I tried various regression algorithms like:

* **XGBoost Regressor**
* **Random Forest Regressor**

Tuning the hyperparameters of respective algorithms is necessary for less error values,regularization and to avoid overfitting in case of tree based models.

I have used only non-parametric models for prediction because two of the hypotheses such as linearity between output and input variables and errors normally distributed were not met.

* **XGBoost Regression:**

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems.Extreme Gradient Boosting, or XGBoost for short, is an efficient open-source implementation of the gradient boosting algorithm. It is computationally effectively faster with better model performance.

XGboost can be optimized by fixing the number of trees, fixing learning rate,tuning gamma, tuning regularization and various hyper parameter tuning.

* **Random Forest Regressor:**

Every decision tree has high variance, but when I combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn’t depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs.

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

**Model performance:**

Model can be evaluated by various metrics such as:

* **Root Mean Square Error**-

Root Mean Square Error(RMSE) is the standard deviation of the residuals. Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells me how concentrated the data is around the best fit line.

It gets influenced by outliers.

* **Mean Absolute Error**- *Mean Absolute Error* is a model evaluation metric used with regression models. The mean absolute error of a model with respect to a test set is the mean of the absolute values of the individual prediction errors on all instances in the test set. Each prediction error is the difference between the true value and the predicted value for the instance.

I choose MAE and not RMSE as the deciding factor because of

* RMSE is heavily influenced by outliers as the higher the values get the more the RMSE increases. MAE doesn’t increase with outliers.
* MAE is linear and RMSE is quadratically increasing.

**Hyper parameter tuning:**

Hyperparameters are sets of information that are used to control the way of learning an algorithm. Their definitions impact parameters of the models, seen as a way of learning, change from the new hyperparameters. This set of values affects performance, stability and interpretation of a model. Each algorithm requires a specific hyperparameters grid that can be adjusted according to the business problem. Hyperparameters alter the way a model learns to trigger this training algorithm after parameters to generate outputs.

I used Grid Search CV, Randomized Search CV and Bayesian Optimization for hyperparameter tuning. This also results in cross validation and in this case I divided the dataset into different folds. The best performance improvement among the three was by Bayesian Optimization.

**Conclusion:**

I Started with loading the data so far I have done EDA , null values treatment, encoding of categorical columns, feature selection and then model building.

In all of these models the errors have been in the range of 2,00,000 which is around 5% of the average views. I have been able to correctly predict views 95% of the time.

After hyper parameter tuning, I have prevented overfitting and decreased errors by regularizing and reducing learning rate.

Given that only 5% is errors, the models have performed very well on unseen data due to various factors like feature selection,correct model selection,etc.