Data Science Techniques for Social Media Prediction Challenge

Balaji Balasubramanian
(20182628)
balaji.balasubramanian@umontreal.ca

Patcharin Cheng
(20182620)
patcharin.cheng@umontreal.ca

Arjun Vaithilingam Sudhakar
(20182449)
arjun.vaithilingam.sudhakar@umontreal.ca

1. INTRODUCTION

The main goal of the Social Media Prediction Challenge is to predict the number of likes a person would get based on 24 features(including image) that contains the person's characteristics like his location, language, number of followers, status updates, picture, etc. The data is multimodal and it contains numerical, text and image data. The features are both continuous and categorical and the dataset is dirty because it has missing values, skewed features filled with outliers, sparse categorical features and features which have little correlation with the label. The error metric used to evaluate the algorithm is Root Mean Squared Logarithmic Error (RMSLE). Our first step was to improve the quality of data by performing data cleaning and transformation to convert the raw data into high-quality data that would make it easier for the algorithms to find patterns and provide good performance. We tried several algorithms on the dataset like linear regression[5], non-linear support vector regression[3] and ensemble methods like Xgboost[1]. We also stacked the non-linear models to improve the overall performance on the leaderboard.

2. DATA CLEANING

2.1 Dataset

The dataset has 7500 train samples and 2500 test samples. There are 23 tabular features and 1 image feature to train the model. The model has to predict the number of likes the person gets which is an integer value. This makes it a regression task.

2.2 Dropping Columns

Our first objective was to drop the columns that don't provide any useful information. Initially, we found that three columns were irrelevant. The Id column is useful while submitting the prediction to the Kaggle Competition page but not useful for machine learning algorithms. User Name and Profile Image columns contain random values like Mf9vfld4Vfe and AL85S14OMDPF01I9.png respectively. Hence, these three columns were dropped and we were left with 20 tabular columns and 1 image feature.

2.3 Handling Missing Values

We find that 10 columns have missing values and it is necessary to deal with them because most machine learning libraries don't accept datasets with missing values. We found that out of the 10 columns that have missing values, 6 of them have categorical values. Two columns, 'Personal URL' and 'Location' have text values with a huge number of missing values. There are two ways of handling missing values, which are to drop the training samples with missing values or to impute them with some other value. We decided to go with the imputation method because we did not want to drop the columns as it would reduce our training data size.

In categorical columns, the missing value can be added to one of the categories. In continuous numerical columns, the missing values can be imputed with mean or median values. The two text columns 'Personal URL' and 'Location' were a little tricky to handle. 'Personal URL' column was converted into binary values, 0 if the column has NaN value and 1 if the column has some value written.

3. EXPLORATORY DATA ANALYSIS

3.1 Visualization Techniques

The three plots we used to visualize the data are Count Plot, Histogram and Heatmap.
Count Plot is used for categorical columns and it tells the number of samples for each category. We used this to detect sparse features.
Histograms are used for continuous(numerical) columns as it helps understand the distribution of data using which we can decide if data transformation is required.
Heatmaps display the values of the samples in terms of colours which makes it easier to understand a feature's correlation with other features and labels.

3.2 Log Transform

We find that continuous features like 'Num of Followers' are skewed and have outliers. Log transform is applied to these features to convert them to Gaussian Distribution which makes it easier for the model to learn from this distribution. We found that the model trained on log based 10 transformed data performs better than log based 2 and natural log transformed data. Similar Log Transformation operation was performed to other continuous-valued columns like 'Num of Status Updates', 'Num of Direct Messages', 'Profile Category', 'Avg Daily Profile Visit Duration in seconds' and 'Avg Daily Profile Clicks'. The missing values were imputed with the mean of the column.

3.3 Categorical Data

After performing EDA, two problems were found with the data. The first problem is that the data has inconsistencies which can affect a machine learning model's performance. For example, 'Location Public Visibility' column has 5 unique values which are 'Enabled', 'Disabled', 'enabled', disables' and '??'. The four text classes are transformed into two classes, 'enabled' and 'disabled' to fix its inconsistency. The '??' value is converted to 'enabled' in order to convert this feature into a binary value.

The second problem is that Categorical Features are sparse. For example, 'User Language' feature has many categories and most of them have a very low frequency. Hence, the top four most frequent categories are chosen and the remaining are clubbed into a new category 'others'. Having fewer categories of data with each category having a decent frequency makes it easier for the model to learn from the data. Similarly, the 'User Time Zone' column has 145 unique categories and most of them are sparse. The categories represent the names of cities, countries and different time zones. We created seven new categories which are 'USA', 'Europe', 'Latin America', 'Asia', 'Middle East', 'Africa' and 'Others' and filled them with all the samples. The 'Others' column also had NaN values assigned to it.

3.4 Feature Selection

The three components of our feature selection process are:-

Removing redundancy- The three features 'User Time Zone', 'Location' and 'UTC Offset' are similar to each other. Since, we have already performed data transformation on 'User Time Zone' and extracted useful location-related information, the other two features have become redundant and they were removed.

Images- We trained a Convolutional Neural Network model on the images and we found that its performance was bad because there is no significant correlation between image feature and the label. Hence, the image feature was not selected.

Color- The three color related categorical features 'Profile Text Color', 'Profile Page Color' and 'Profile Theme Color' had very sparse categories and selecting them reduced the performance of the model. Hence, they were removed.

4. METHODOLOGY

4.1 Preparation

Various other preprocessing steps were performed on the data because sending them as an input to the model. They are:-

4.1.1 Categorical Features

The Categorical Features had categories in text form that cannot be directly fed to the model, so they were converted to numerical form. The column 'Profile Cover Image Status' has two values, 'Set' and 'Not Set' which are converted to 0 and 1. 'Is Profile View Size Customized?' has three unique values, 'Not verified', 'Pending' and 'Verified' which are converted to 0, 1 and 2 respectively. Similar steps were done for other categorical features like 'Profile Verification Status', 'User Language', 'Profile Category' and 'User Time Zone'.

4.1.2 Log Transform on label

We have already performed log transform on the continuous numerical input features. This has changed the scale of the input features compared to the original label 'Number of Profile Likes'. Hence, we have to perform log transform on the labels to bring them to the same scale as the input features. This provided a boost in RMSLE score of 0.2.

4.1.3. Standard Scaler

Standard Scaler rescales the distribution of all input features so that they all have that same scale of mean 0 and standard deviation 1. This makes it easier for the model to learn from the data and converge faster.

4.1.4. Feature Engineering

We used the pre-existing features to create new features. The 'Profile Creation Timestamp' column is stored in date-time format. We created a new feature 'MonthsInSocialMedia' that tells us how many months the user has spent on the social media platform. Besides, using 'MonthsInSocialMedia' to derive other extra features significantly further reduces the error. Four new features were also created by dividing the columns 'Num of Followers', 'Num of People Following', 'Num of Status Updates' and 'Num of Direct Messages' columns by 'MonthsInSocialMedia'. The derived columns improved the performance of the model by helping it learn the hidden information in the data.

4.2 Models

Various models have been tried on this problem which are:-

4.2.1 Linear Regression

Linear regression[5] is the simplest machine learning algorithm that tries to fit a linear equation to the input features to predict the label. It performs badly in this competition because the data has non-linear correlation between the input features and output labels and the model suffers from high bias. Its validation score on performing K-Fold Cross Validation is 1.83.

4.2.2 Support Vector Regression

Support Vector Regression(Machines)[3] uses the kernel trick to deal with non-linear data in which it transforms the input features into a higher dimensional space where it is linearly separable. We tried both RBF Kernel and Polynomial Kernel and we found that RBF Kernel performs the best. It uses the concept of maximum margin hyperplane in which it tries to fit the data points within the margin of epsilon from the function. This improves the generalization ability of the model. Its validation score on performing K-Fold Cross Validation is 1.73 which is 0.1 better than linear regression.

4.2.3 *Ensemble Methods*

Decision Tree Regressor is a machine learning model that partitions the input space of the training data into several regions based on several test cases. These test cases are performed on the test data point which is assigned to one of the regions. The predicted value is equal to the average value of all the training points in that region. The model can suffer from underfitting if the number of test cases are few and overfitting if the number of test cases are too many.

Hence, we create an ensemble in which multiple decision trees are combined to predict the label. There are two ways of ensembling decision trees, bagging and boosting. Bagging aims to reduce variance by training multiple decision trees in parallel and averaging their predictions. Boosting aims to reduce bias by training multiple decision trees sequentially where every subsequent tree focuses on training points misclassified by the earlier tree. We found that boosting works better than bagging for this problem because the dataset is complex and reducing bias is more important than reducing variance. We used two gradient boosting algorithms (libraries) Xgboost and LightGBM which are very similar to each other and we found that Xgboost performs better than LightGBM.

4.2.4 *Xgboost*

Xgboost[1] is a boosted tree library that has a few interesting properties that make it better than a typical gradient boost library like Sklearn. It uses parallel processing to train the model simultaneously on multiple cores of a CPU. It uses L1 and L2 regularizers to deal with overfitting. It performs cross validation in every iteration of the boosting process which enables it to achieve the optimal performance in a fewer number of iterations. It uses post pruning technique to build the trees, in which it first builds the tree to the max depth (passed as a hyperparameter) and then prunes the tree in the backward direction to remove unnecessary test cases. Xgboost scored 1.72 on the local cross validation set which is 0.01 better than SVR. The ensembling effect in Xgboost reduced its variance which enabled it to perform better than SVR.

4.2.5 *Stacking*

We combined our best and diverse non-linear models, Xgboost and SVR using stackingCV Regressor library[2]. Figure 1 describes the algorithm. There are two first-level regressors, SVR and Xgboost which are trained using K_Fold Cross Validation to improve their generalizability. The models are trained on k-1 folds of data separately and their predictions on the validation set are stored. The second level regressor is Xgboost which is trained on both the predictions of the first-level regressor and the raw training data. Once the first-level regressors are properly trained and their hyperparameters are selected, they are fitted on the entire training data and are now ready to be used on the test data. This model performed the best and has a score of 1.70 on the local cross validation set which is 0.02 better than Xgboost alone.

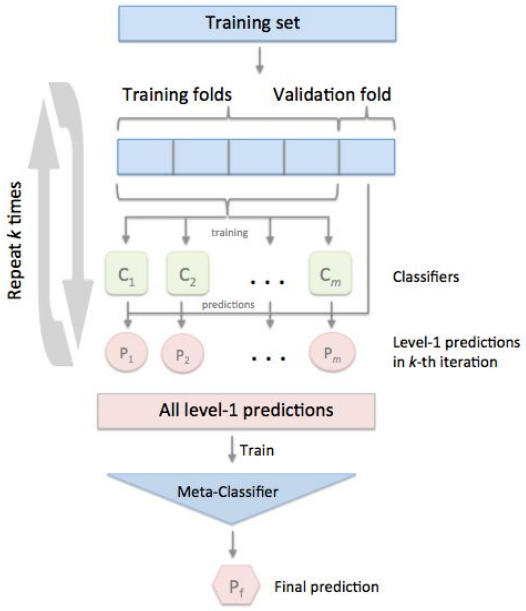

Figure 1: Working of stacking models[2]

4.2.6 *Convolutional Neural Network*

A Convolutional Neural Network model[4] was built on the image feature. The model has two Convolution layers(with max pooling) and two dense layers out of which one is the output layer. It performed badly on the data and is even worse than linear regression. Hence, we decided not to use the image feature.

5. RESULT AND DISCUSSIONS

5.1 Validation Strategies

5.1.1 *Train-Validation Split*

The training data is split into train and validation sets. The data is shuffled before splitting it into train and validation sets and a random seed argument is used for reproducibility. The model is then trained on the training set and the trained model's performance on the validation set informs us about its generalizability. The hyperparameters of the model are tuned to improve the performance on the validation set. Once the model is built using the right hyperparameters, it is used for prediction on the test set. The Train-Validation Split technique is very fast but it suffers from selection bias of the validation set.

5.1.2 *K-Fold Cross Validation*

In K fold Cross Validation, the data is split into k fold. After that, the model is trained on k-1 folds of data and the trained model is used for prediction on the one fold that is left out from the train set. This process is repeated k times, and each time a different fold is used for validation. The 'k' validation scores are averaged to get an unbiased estimate of the generalization capacity of the model. We found that K-Fold Cross Validation is a better strategy for hyperparameter optimization but it takes more time to perform

than train-validation split. Hence, it was only performed on a few hyperparameter values that had performed well on Train-Validation Split to get the best hyperparameters before the model can be used on Kaggle's test set. We chose the value of k as 10. This allowed us to train the model on a larger portion of the training dataset and further reduce the bias of the estimator. We found that k=10 provided an error estimator that was very close to the public leaderboard score.

5.2 Hyperparameter Tuning

We used two methods to select hyperparameter values from a search space of values for K-Fold Cross Validation which are Random Search and Grid Search. In Random Search, hyperparameter values are randomly selected from the search space. In Grid Search, we define a grid of values and the performance of each position in the grid is evaluated. We used a combination of grid search and random search to select the hyperparameters. We first used random search to find the promising regions in the search space. Then we performed grid search on these promising regions.

We found the best hyperparameters using the validation strategies. For Support Vector Regression, we found that the 'RBF' kernel performed better than the Polynomial kernel. We selected the value of hyperparameter epsilon as 2, which represents the distance from the hyperplane within which no penalty is assigned in case of error (this helps with generalization). We used L2 regularizer which is controlled by Sklearn's C hyperparameter. C is inversely proportional to regularization strength and we selected its value as 0.75.

In the case of Xgboost, we found that the model has many hyperparameters and we decided to tune a few of them. We set L1 regularizer represented by Xgboost's alpha to 2. We chose learning_rate as 0.03 which represents the amount of learning in each iteration of the boosting process. We chose the n_estimators as 250 which represents the number of trees used in the ensemble and the maximum depth of each tree as 3.

5.3 Kaggle Results

Table 1 shows the RMSLE score that our four models Linear Regression, Support Vector Regression, Xgboost and StackingCV Regressor achieved on the leaderboard. As you can see from the table, Linear Regression performed the worst and the StackingCV Regressor performed the best.

Model	Public RMSLE	Private RMSLE
Linear Regression	1.84500	1.70345
Support Vector Regression	1.75782	1.61980
Xgboost	1.72112	1.60737
StackingCV Regressor	1.70724	1.59018

Table 1: Comparison of the model used in the kaggle challenge

Linear Regression performed the worst due to the non-linear characteristics of the dataset. Support Vector Regression (SVR) performed much better than Linear Regression due to its properties like kernel trick which helps it build a non-linear representation of the data and maximum margin hyperplane which improves its generalizability. The ensemble model 'Xgboost' performed better than SVR because the presence of multiple weak learners helps it build a complex representation that has low bias and is also highly generalizable reducing its variance. Our best model was built using the StackingCV Regressor in which we stacked SVR and Xgboost to combine their strengths and managed to achieve fourth place on the private leaderboard of the Kaggle Competition.

6. Conclusion and Future work

The secret sauce for our performance in this competition consists of feature selection and engineering, log transform of input and output features, and hyperparameter tuning and efficient stacking of our models. With all the steps proposed in this report, we are able to build a good data science pipeline and achieved an RMSLE Score of 1.59018 on the private leaderboard.

Future work on this problem includes working on better feature selection and feature engineering of tabular data to improve the quality of the data. Feature encoding can also be tried on categorical data to decrease its sparsity. Different feature extraction techniques on the image features can be used to detect properties like gender and emotions. There is always scope for trying new machine learning models on the tabular data and also improve the hyperparameters of existing models.

7. References

1. Xgboost- https://arxiv.org/abs/1603.02754

2. StackingCV Regressor- http://rasbt.github.io/mlxtend/user_guide/regressor/StackingCVRegressor/

3. Support Vector Regression- https://scikit-learn.org/stable/modules/svm.html#regression

4. Convolutional Neural Network- https://www.tensorflow.org/tutorials/images/cnn

5. Linear Regression- https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares