CS412 - Machine Learning Term Project

TEAM MEMBERS:

Erkan Kasil - 32660
Arda Taşkoparan - 32512
Zeynep Neval Yaprak - 32516

INTRODUCTION

In this project, we have two main tasks: classification and regression. In the classification part, our aim is to correctly classify influencer accounts from instagram into one of 10 categories that are as following:

Fashion: Highlights latest fashion trends, style guides, seasonal outfits, and fashion industry news.

Gaming: Streams and reviews of popular video games, game tips, tutorials, and industry updates for casual and hardcore gamers alike.

Food: Offers recipes, cooking tips, food photography, and dining experiences for food enthusiasts.

Tech: In-depth reviews of the latest gadgets, comparisons of electronic devices, and insights into tech trends.

Mom and Children: Practical advice on parenting, creative activities for families, and tips for childcare.

Health and Lifestyle: Offers advice on fitness, mental well-being, and maintaining a balanced lifestyle.

Travel: Travel guides, outdoor adventure stories, and tips for adventurous travelers looking to explore new destinations.

Sports: Covers a range of sports, fitness tips, and discussions on maintaining physical health.

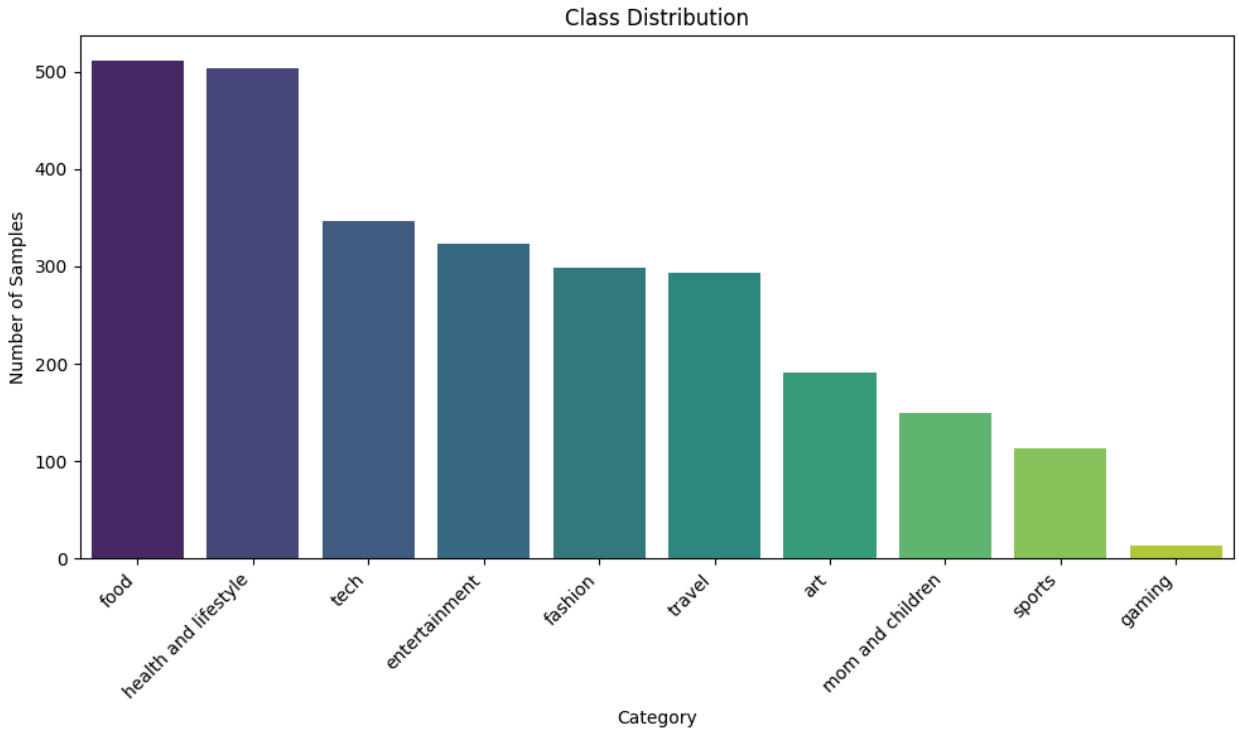
Art: Showcases artwork, explores different art styles, and provides insights into the art world.

Entertainment: Reviews and insights into concerts, shows, and events, along with suggestions for leisurely enjoyment.

The purpose of our regression task is to determine how many likes an instagram post will get, based on the regression models output that has been trained on a dataset containing information about instagram influencer accounts and their posts.

Our general approach was to try different models and see how we can improve, whether by adjusting model hyperparameters or feature selection.

The graph below shows the distribution of different classes of accounts on the provided dataset to use for training.



ROUND 1

- **Classification:**

In the first round of classification, we developed and evaluated a stacking classifier that included Gradient Boosting, Logistic Regression (L1 and L2), SVM, and Multi-Layer Perceptron (MLP) as base estimators, with a Random Forest classifier acting as the meta-model. Although this was an initial attempt, it helped establish a baseline for subsequent rounds.

Preprocessing Steps and Feature Engineering

For Round 1, we implemented a comprehensive preprocessing pipeline to prepare the dataset for classification. These steps were foundational and later became part of our improved approach in Round 2:

1. **Text Preprocessing**:
    - The text data, including user biographies and post captions, was cleaned using a pipeline that removed URLs, special characters (except hashtags and emojis), and numbers. The text was also normalized to lowercase, accounting for Turkish-specific letters.
    - This preprocessing pipeline aimed to reduce noise and extract meaningful textual information for the classification task.
2. **TF-IDF Vectorization**:
    - The cleaned text corpus was transformed using a TF-IDF vectorizer, which generated a high-dimensional feature matrix. In Round 1, all 7500 features were retained without dimensionality reduction, which introduced computational overhead and potential noise in the data.
3. **Numerical Features**:
    - In addition to text features, numerical data such as follower_count, following_count, and average_like were incorporated. While these features showed limited correlations in the analysis, they were included to capture user-level metadata that could aid classification.
4. **Train-Test Split**:
    - To ensure class balance, we performed a stratified train-test split. This split was vital due to the imbalanced nature of the dataset.

Results and Model Performance

The stacking classifier in Round 1 delivered an initial validation accuracy of approximately 0.65 on our dataset. While the validation score highlighted room for improvement, it provided critical insights into the model's strengths and limitations.

- Competition Results:
    - Our submission for Round 1 achieved a classification accuracy of 0.783186 and an F1-weighted score of 0.773254 in the competition. These results positioned us at Rank 37 for accuracy and Rank 40 for the F1-weighted score among 141 submissions.
- Analysis and Observations:

- Preprocessing Impact: The preprocessing pipeline, though robust, may not have fully leveraged the dataset's potential due to the lack of dimensionality reduction. The high-dimensional TF-IDF matrix likely introduced noise, affecting the model's performance.
- Model Limitations: The stacking classifier used in Round 1 employed default or suboptimal hyperparameters for the base estimators and meta-model. This contributed to lower performance compared to later rounds.
- Feature Set: While auxiliary features like follower_count and average_like added value, their contribution could not fully offset the high noise level in the TF-IDF representation.

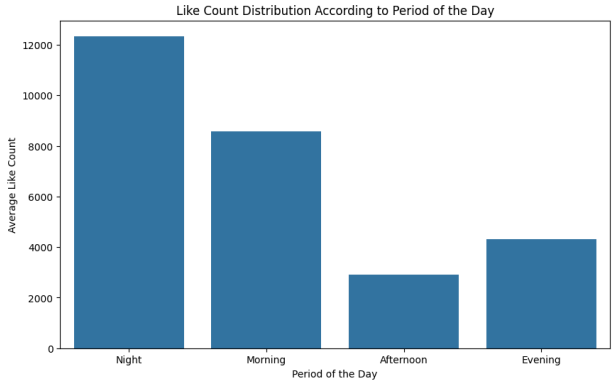
Insights for Improvement

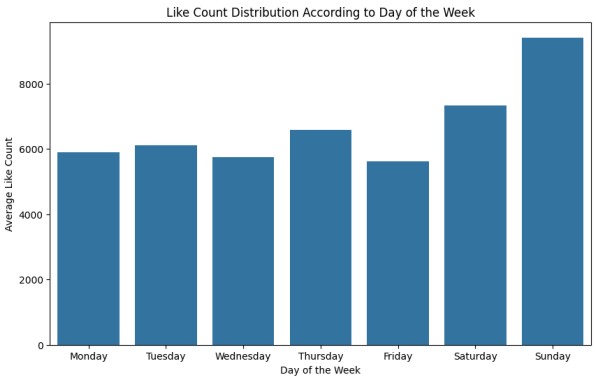
Round 1 was pivotal in identifying key areas for enhancement, including:

- The need for dimensionality reduction to manage high-dimensional text features effectively.
- Optimization of hyperparameters for all classifiers in the stacking ensemble.
- Exploration of more sophisticated approaches to feature selection and engineering.

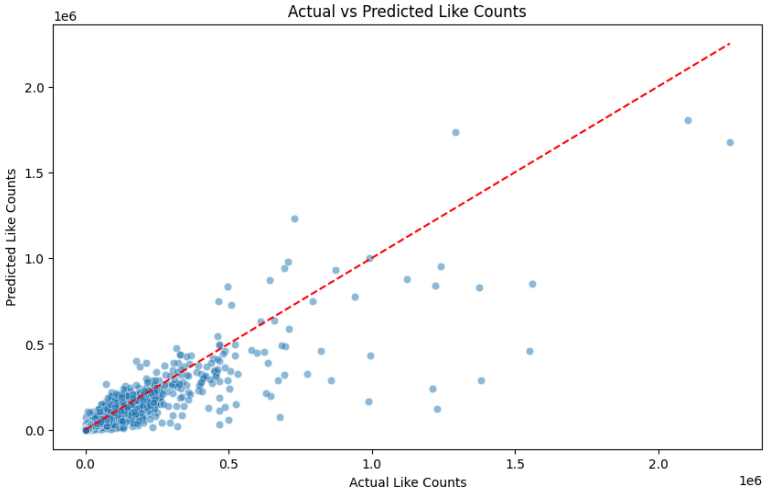
- **Regression:**

We mainly focused on feature selection since it’s the key of performance. We tried different combinations of features and this stage took lots of time because some features which seam to contribute actually end up decreasing model performance. As features, we used some classical ones like follower_count, following_count, average_like_count etc. At some point, we think of adding a new feature called until_now_averag_count which is logical but didn’t work because of restrictions of the dataset. Because there were not enough posts per user. Secondly, we dividend timestamp into more meaningful groups: day_of_week, day_of_year, time period of day(night, morning, afternoon, evening). I believe that these features capture really good amount of information and these are good metrics which effect like count. We filled all null feature instances with zeros. You can see importance of some time features:





After selecting our features, we tried some models. Random forest regressor was a good start and it gave approximately 0.48 log mse on our train-validation split. We tried SVM too but the result was not good and it took too much time.



Lastly, we opened test set and selected same features as training set. average_like_count is calculated based on training set. Then we recorded our result into a json file.

ROUND 2

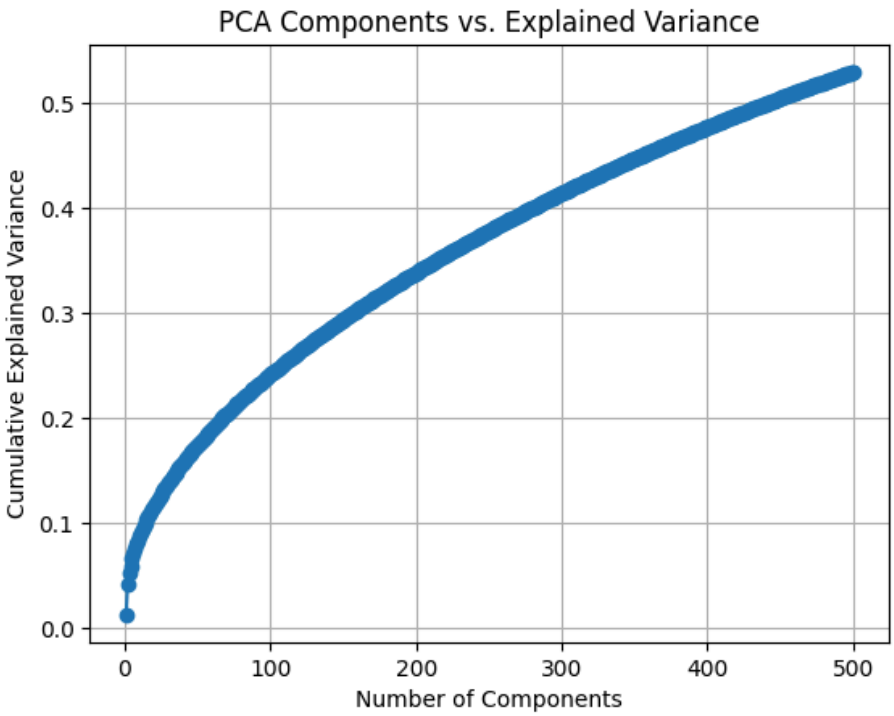
- **Classification**:

For the second round of the competition, we employed a more refined preprocessing and model training approach. The preprocessing pipeline built upon the techniques used in

Round 1, while adding additional steps such as creating a correlation matrix and conducting dimensionality reduction via PCA and Truncated SVD.

Preprocessing Steps:

1. **Text Preprocessing**:
    - Similar to Round 1, text preprocessing included URL removal, special character filtering (retaining emojis and hashtags), and standard tokenization techniques.
    - We applied a TF-IDF vectorizer with 7500 features to transform text data into feature matrices.
2. **Feature Engineering**:
    - In this round, we explored feature relationships via a correlation matrix, visualizing the relationships between attributes like follower_count, is_verified, and average_like. This analysis helped identify redundant features and informed decisions about feature selection.
    - **Correlation Matrix Insights**:
        - Attributes such as is_verified and average_like showed a relatively higher positive correlation (0.269), whereas is_business_account and average_like exhibited a negative correlation (-0.189). These insights provided a deeper understanding of the data's structure.



3. **Dimensionality Reduction**:
    - To address feature redundancy and improve computational efficiency, we applied Principal Component Analysis (PCA) and Truncated SVD:

- PCA was used to analyze cumulative explained variance, revealing that 500 components explained approximately 50% of the variance in the dataset. This insight shaped our strategy for dimensionality reduction.
- Truncated SVD reduced the feature set to 1000 components, which retained the most critical information while significantly reducing dimensionality.

Model and Training:

- The same **StackingClassifier** framework from Round 1 was employed, but with hyperparameter tuning to improve model performance.
  - **Base Estimators**:
    - Gradient Boosting (learning rate: 0.1, max_depth: 7, 200 estimators)
    - Logistic Regression (L1 and L2 penalties)
    - Support Vector Machines (RBF kernel, C=50, gamma=0.01)
    - MLP Classifier (hidden_layer_sizes=(100, 100), learning_rate='adaptive')
  - **Final Estimator**: Random Forest (max_depth=50, min_samples_leaf=2)
- A 5-fold cross-validation technique was applied to evaluate the model's performance on the training dataset. This approach provided a more robust estimate of the model's accuracy and ensured that the results were not biased by a single train-test split.

Results:

- **Validation Accuracy**: On our internal dataset, the model achieved a cross-validated accuracy of approximately 65%. This reflected consistent improvement in feature engineering but still indicated room for model optimization.
- **Competition Results**:
  - The competition platform reported a classification accuracy of 82.6% and an F1-weighted score of 0.819. These results placed our submission at Rank 51/138 for accuracy and Rank 52/138 for F1-weighted performance.

- **Regression:** Same thing with round 1 since we end up in first place. Only difference is that we used all training set to train the model, and performed cross validation to see the model's MSE.

ROUND 3

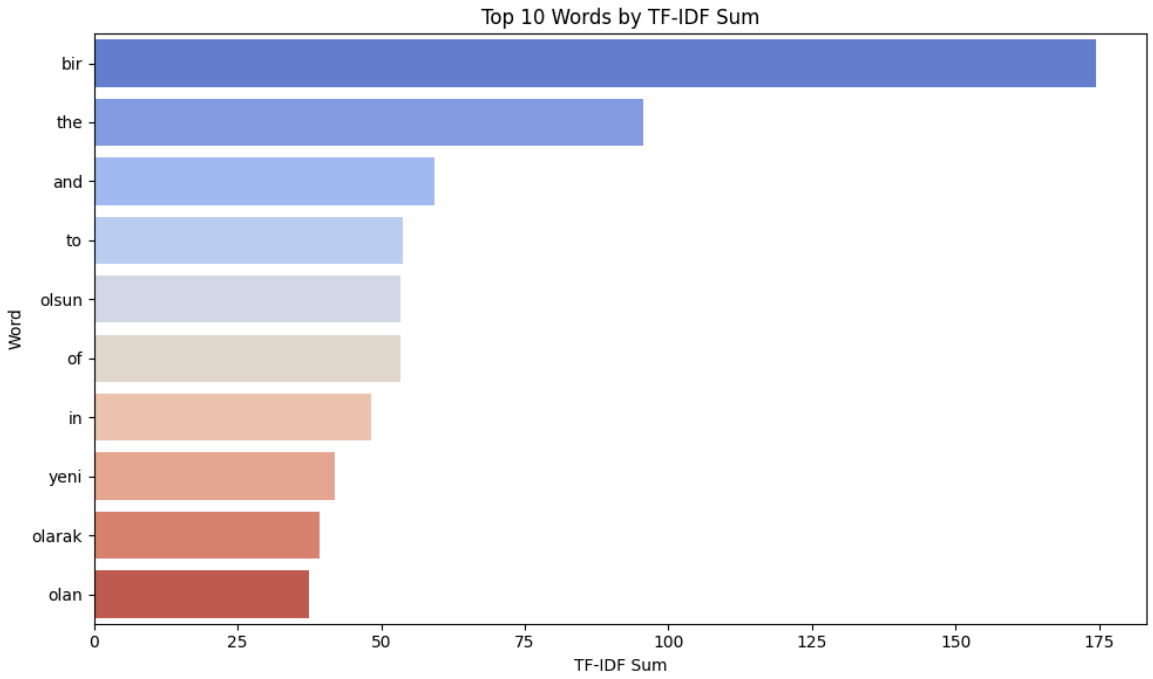
- **Classification:**

  Our approach to the classification in round 3 begins by **collecting user posts** from Instagram and applying **mild text preprocessing** that preserves emojis, URLs (as tokens), and punctuation, while removing numerical noise and extra whitespace. We then convert the cleaned captions into **TF-IDF features**, which are scaled to help certain models perform more effectively. In this round, we opted to prevent emojis in captions from being removed.

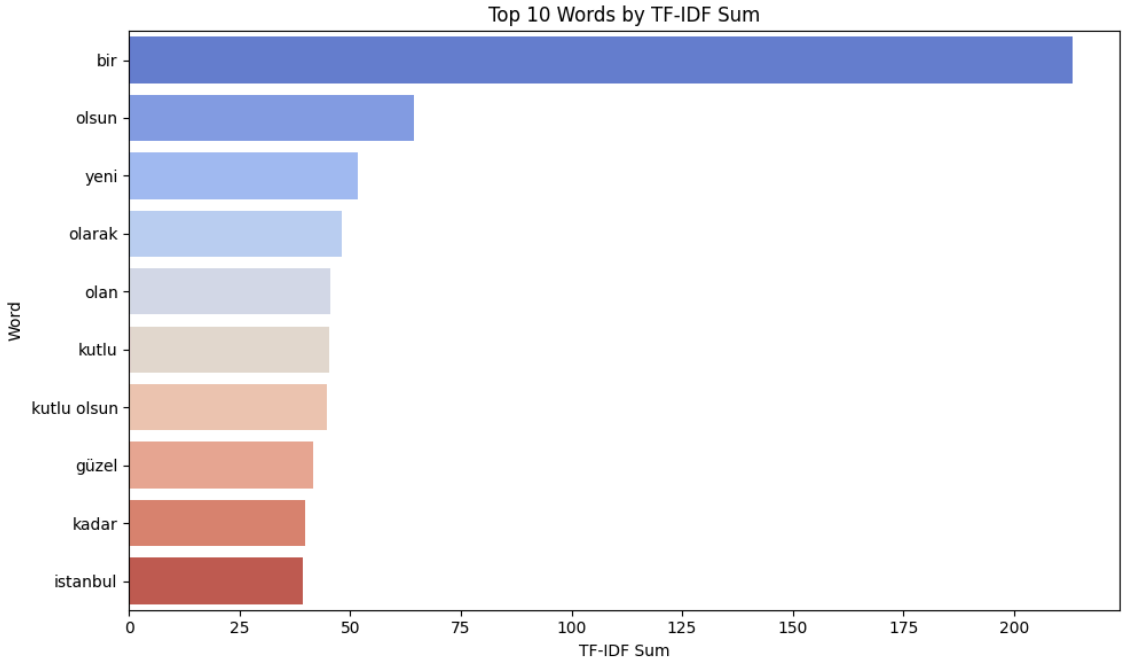
  To address **class imbalance**, we apply **SMOTE** for synthetic oversampling, ensuring that minority classes receive adequate representation. We experiment with and tune several supervised algorithms—**Logistic Regression, RandomForest, LinearSVC, and an MLP neural network**—assessing their performance via **validation accuracy** and confusion matrices. We initially considered LightGBM as well, however we chose to not use it as it produced very low accuracy and took a lot of computational resources and time to run.

  Additionally, we combined these models in a **Stacking Classifier** for a more robust final prediction. Throughout the process, we log and visualize key statistics such as the distribution of caption lengths, the most frequent words, and per-class confusion matrices.

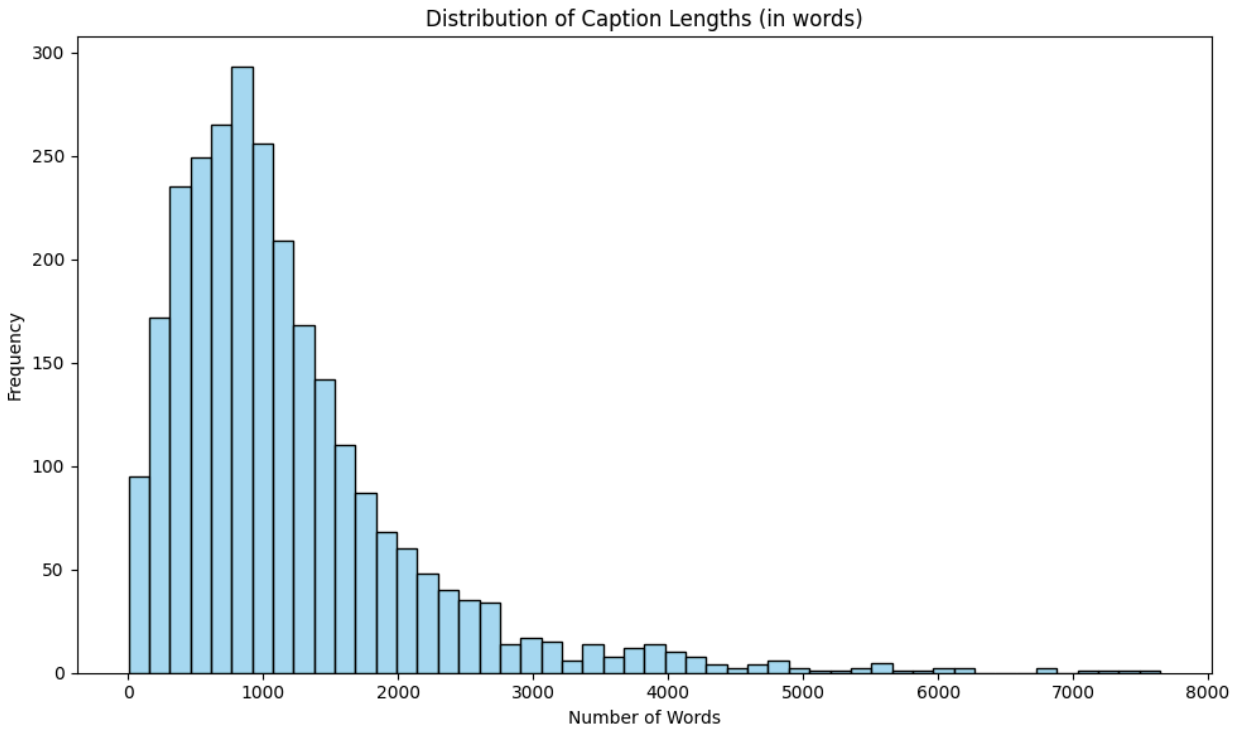
  However, as the code took a lot of time to run with the given setting, we decided to only use logistic regression as it proved to be the one providing higher accuracy. Additionally, after visualizing the top 10 TF-IDF vectors, we realized there were also English stop words included. So we removed those in the last iteration.



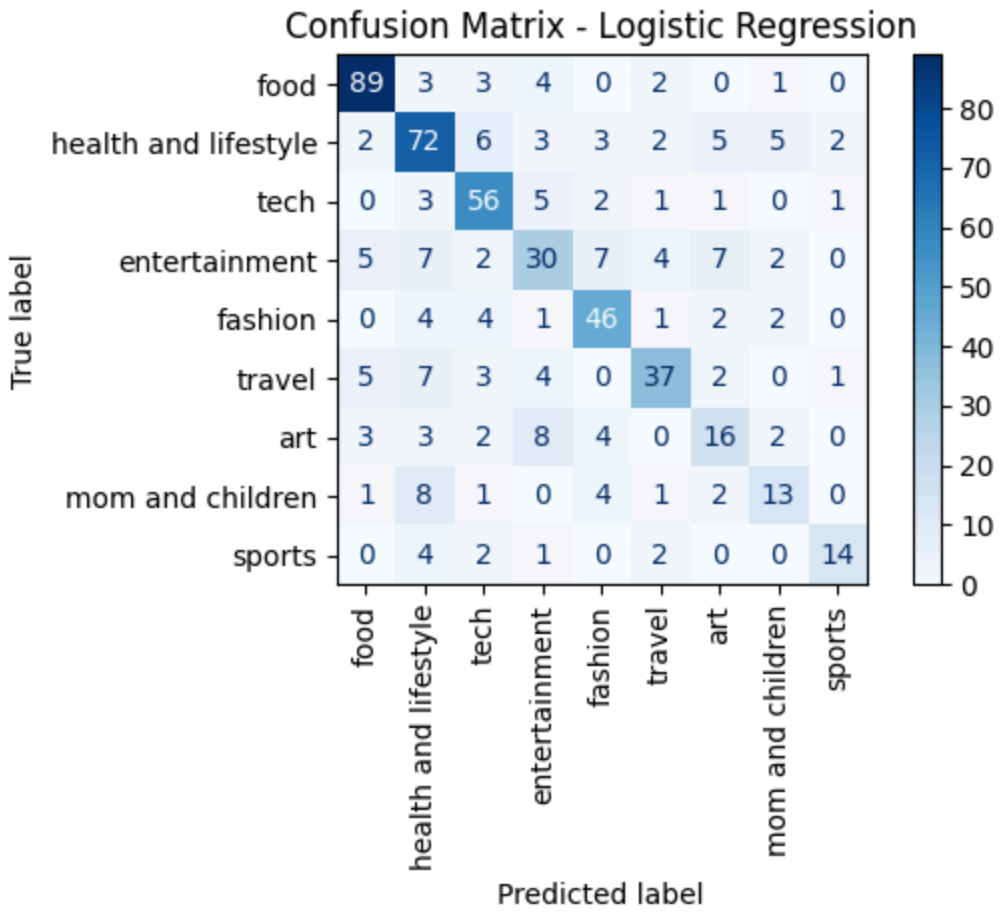
The Plot above shows the 10 words with highest TF-IDF Sum before removing the English stopwords together with Turkish stopwords. The plot below is the final version that shows the remaining top 10 after English stopwords have been removed.



The graph below shows the distribution of caption lengths of the posts after removing the stopwords and processing the captions.



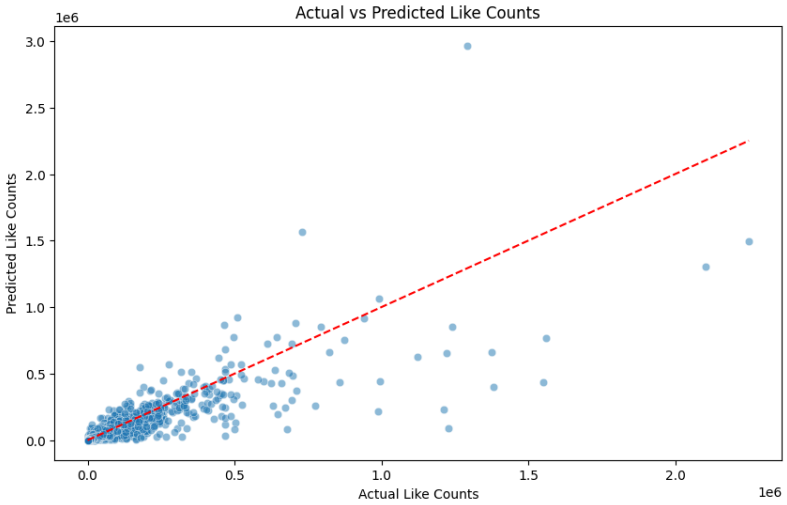
Below is the confusion matrix for the predictions on the part of the provided training dataset that has been set as test test.



- **Regression**:

We used same features as before but we also added two more which are caption_length and hashtags_count. First one count characters in captions and second one count #s in captions. These didn't contribute that much but slightly decreased MSE.

We also tried different models like gradient boosting regressor and XGB regressor. Gradient boosting outperformed random forest with train-validation approach but with cross validation, the MSE was bigger. But XGB gave us the best results with log MSE of approximately 0.445 with train-validation approach and MSE of around 0.44 with cross validation approach. So we continued with XGB and we fit the model on all of the training set, because we wanted to get advantage of all instances to train the model. You can see result for train-validation approach:



TEAM CONTRIBUTIONS:

We initially had a meeting and brainstormed together as to what approach could be followed for the classification and regression tasks. We decided to each try training different models with different parameters and improve the best one.

32660 worked on the regression task. Since his code proved to be very effective (got 1st rank on 1st round) we kept on using the same code.

32516 and 32512 worked on the classification task. We each trained some models and compared accuracy and had collective group discussions as to which route to go. In the first 2 rounds we submitted the output of 32512's code and in the last round 32516's.

In the process, we all exchanged opinions and code we had written. We collectively worked on the documentation on the github repository and the project report.