Sabanci University 2024 – 2025 Fall Term

CS412 – Machine Learning

Project Report

## 1. Introduction

In this project, we worked with Turkish Instagram data and developed Machine Learning pipelines to analyze Instagram data, focusing on influencer classification and like count prediction. The objectives are:

1. **Classification of Influencer Categories:** Predicting influencer profiles among ten categories, addressing challenges like class imbalance and Turkish-specific content.
2. **Regression for Like Counts:** Predicting logarithmic like counts, tackling heavy-tailed distributions and varying account popularity.

The project involves data annotation, feature extraction, and modeling. A real-world Instagram dataset underpins the analysis, utilizing advanced preprocessing, feature engineering, and tailored models. This report outlines methodologies, challenges, and results, showcasing machine learning's role in social media analytics.

## 2. Methodology

### 2.1 Classification

#### 2.1.1 Preprocessing

For the classification task, we utilized the provided Python Notebook but enhanced the model by introducing new preprocessing techniques and experimenting with additional machine learning models. Our preprocessing steps included the following improvements:

1. Text Preprocessing

- Emojis were replaced with an emoji_ placeholder and converted into their Turkish equivalents using the emoji library.
- URLs were replaced with urltokens.
- Mentions, such as @user, were replaced with mention_user.
- Hashtags, like #nature, were converted into hashtag_nature.
- Numbers were detected and replaced with numbertoken.

2. Lemmatization and Tokenization

- We applied lemmatization and tokenization to the text using the simplemma library's Turkish lemmatizer to better preprocess both user captions and biographies.

3. TF-IDF Vectorization
    - We experimented with various configurations of the TF-IDF vectorizer. After hyperparameter tuning, the optimal configuration was determined to be:
        - max_features=5000
        - min_df=5
        - Removal of Turkish stopwords using the nltk library.
    - We found that fitting the TF-IDF vectorizer solely on post captions, without including biography features, resulted in better accuracy.

These preprocessing steps significantly improved the model's performance and contributed to achieving higher accuracy in the classification task.

#### 2.1.2 Machine Learning Algorithms

After preprocessing, we implemented various machine learning algorithms to maximize the accuracy of profile classification. The models we trained and their configurations are as follows:

1. **XGBoost:**
    - Evaluation metric: mlogloss
    - Number of estimators: 100
2. **Logistic Regression:**
    - Maximum iterations: 1000
3. **Support Vector Machine (SVM):**
    - Kernel: Linear
    - Regularization parameter (C): 1
4. **Multinomial Naive Bayes:**
    - Default configuration as provided in the example file

To ensure better generalization, we applied **5-fold cross-validation** to all models. We calculated the **mean accuracy** and **standard deviation** for each model and selected the best-performing one based on these metrics. Our results indicated that the SVM classifier performed the best, achieving a mean accuracy of approximately 0.665 on the training data.

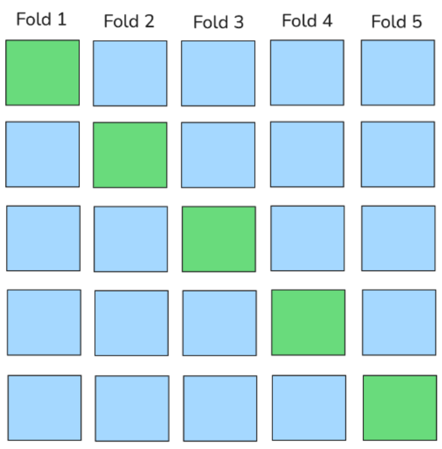


Figure 1: Cross-Validation split.

### 2.1.3 Evaluating Test Data Overlap

After selecting the SVM classifier as the best-performing model, we used it to predict the classes of test profiles for each round. During this process, we introduced a check that significantly improved the model's performance.

Before finalizing predictions, we checked if any test instances in a given round overlapped with the training data. For such overlapping instances, their true classes were already known, and we verified whether our SVM model predicted them correctly. **If the predictions were incorrect, we corrected them manually.**

Across all rounds, we found that **250–300** of the **1000** test instances typically overlapped with the training data. For these overlapping instances, we iterated through the test dataset and validated their classifications. For example, in Round 2:

- **Test Dataset Details:**
  - Total test instances: 1000
  - Overlapping instances: 277
- **Prediction Results:**
  - Correctly classified: 239
  - Incorrectly classified: 38

By applying this additional check, we ensured that our model leveraged existing knowledge from the training dataset, leading to improved performance on the test data.

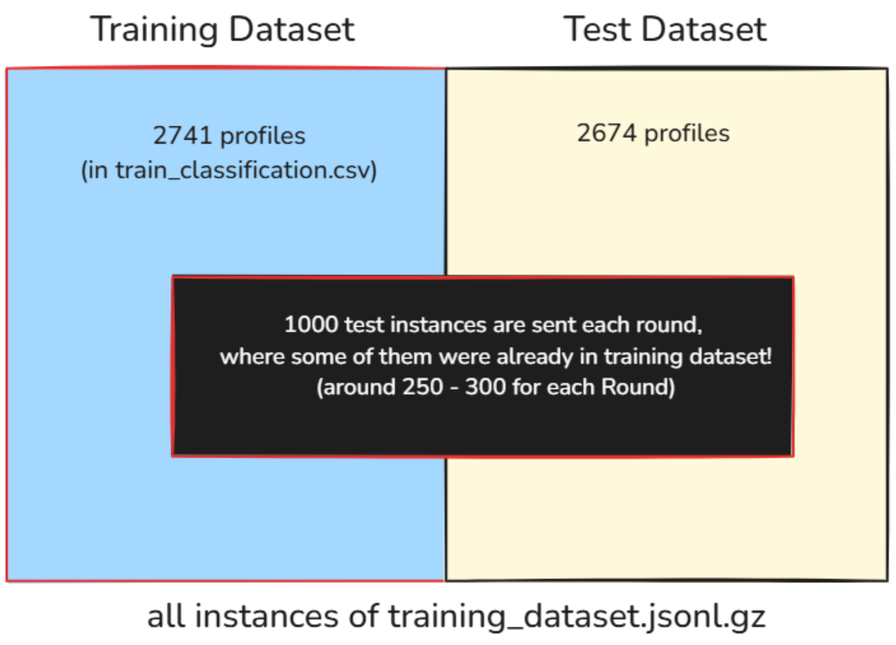

Training Dataset
Test Dataset
2741 profiles
(in train_classification.csv)
2674 profiles
1000 test instances are sent each round,
where some of them were already in training dataset!
(around 250 - 300 for each Round)
all instances of training_dataset.jsonl.gz

Figure 2: Test Data Overlap.

## 2.2 Regression

For the regression task, the model needed to predict logarithmic like counts of Instagram posts. Heavy-tailed distribution and popularity variance between accounts made this task challenging. Advanced feature engineering, modeling experiments, and iterative refinements were performed to overcome this challenge.

### 2.2.1 Feature Engineering

We began by extracting and processing a robust set of features derived from user data, post characteristics, and textual content to develop an effective model. The dataset consisted of 187,302 rows, each representing an individual post, with key columns such as comments_count, like_count, num_followers, and text captions.

**Key Features:**

1. **Engagement Metrics:**
   - comments_count, num_followers are the very basic features which are directly proportional to the engagement.
2. **Statistical Features:**
   - Aggregated metrics include average_like, maximum_like, and minimum_like.
   - Variability metrics, like the standard deviation of likes that a user received. However, this feature was dropped later since it did not add much value to performance.
3. **Temporal Features:**
   - Timestamps were converted into cyclic representation to model periodic trends in user engagement effectively.
4. **Exclusion TF-IDF:**
   - While explored, TF-IDF representations for captions did not notably improve performance and were omitted to simplify the pipeline.

### 2.2.2 Modeling Strategies

Our strategy for modeling was to try a number of algorithms and techniques in an attempt to balance the competing demands of predictive accuracy and computational efficiency. The following describes our approach:

**Gradient Boosting Regressor (GBR):**

**Baseline Model:**

- Selected because it can handle complex feature interactions and heavy-tailed distributions.
- Obtained a Logarithmic Mean Squared Error (Log MSE) of 0.33.

**Multi-Layer Perceptron (MLP):**

**Deep Learning Model**

- Designed to have 11 layers, allowing it to pick up non-linear relationships.
- The model's hyperparameters, including layer dimensions, learning rates, batch sizes, and dropout rates, were tuned.
- This model was slightly worse in its performance compared to GBR, with a Log MSE of 0.34.

**Ensemble Learning**

**Stacking Models**

- Base models in this regard include Random Forest Regressors, GBR, and MLPs.
- It was combined by using their outputs to train a meta-model that achieved a Log MSE of 0.32.

**Sequential Modeling**

**Iterative Refinement**

- Predictions from one model were used as features for the next.
- Four iterations of sequential modeling yielded the best results, beyond which overfitting was observed.
- Random Forest and GBR models were used consistently in sequential modeling due to their robustness.

### 2.2.3 Feature Scaling and Selection

Feature scaling was necessary to ensure that the model performs best, especially for gradient descent. Normalization techniques were applied to account for different magnitudes of features like comments_count and num_followers. Recursive Feature Elimination identified impactful features for the interpretability of the model, reducing noise in the data.

### 2.2.4 Final Framework

This final framework combined sequential and ensemble modeling to leverage strengths from multiple approaches. Each model along the sequence was trained and saved to make efficient predictions on the test dataset without retraining. This resulted in a best Log MSE of **0.15**, which evidences that iteration refinement and ensemble learning work very well.

### 2.2.5 Insights and Future Work

By excluding TF-IDF features, the focus was shifted to numerical and categorical data transformations that enhanced the efficiency of the models. Future work may incorporate other temporal patterns or external data sources, such as hashtags or post categories, to further improve the predictions. These results emphasize the importance of balancing feature engineering with model complexity and generalization in predictive analytics.

## 3. Discussion of BERT Models & Results

Building on the previously discussed models, we incorporated BERT embeddings to further enhance our system's performance. Given their dominance in the NLP domain, we integrated models like **BERTurk** and **TurkishBERTweet** for classification and regression tasks.

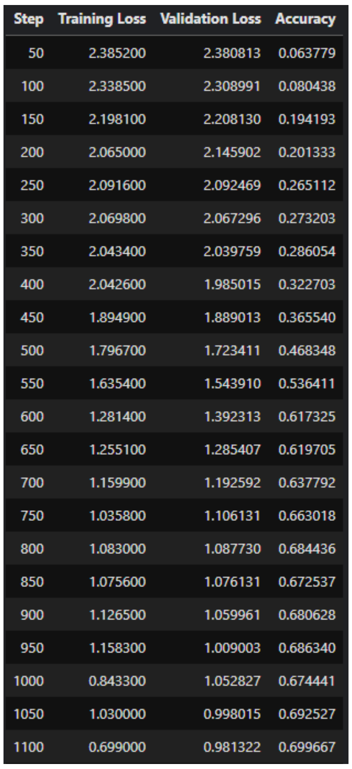
### 3.1 Classification Using BERT

For classification, we introduced BERTurk and TurkishBERTweet embeddings. The TurkishBERTweet **preprocessor** was used for preprocessing, followed by chunking to handle larger inputs effectively. Key hyperparameters included:

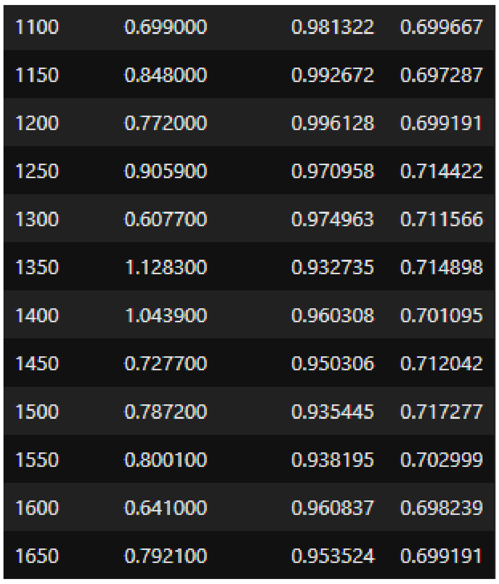
- Learning rate 1e-5
- AdamW optimizer
- 10 Epochs
- Early stopping: Enabled with a patience of 3 epochs

Early stopping proved critical as the model tended to overfit beyond a certain accuracy threshold. For instance, without early stopping, validation loss increased significantly after extended training. The BERTurk model achieved the best results, converging to an accuracy of approximately 72%, as depicted in the training performance charts.

Please note that for the classification task, even though the fine-tuned BERTurk embeddings achieved 72% accuracy, we applied a further evaluation of predictions on the given training data and Round III test. For the third round, among the 266 overlapping instances in the training dataset and 1000 classification test instances, the SVM model misclassified 40 instances, while the fine-tuned BERTurk embeddings misclassified 80 instances. This discrepancy may be attributed to the early stopping applied during training, which may have limited the BERTurk model's potential. Ultimately, we proceeded with the SVM model as it consistently provided 65% test accuracy across 5-fold cross-validation.



| Step | Training Loss | Validation Loss | Accuracy |
|---|---|---|---|
| 50 | 2.385200 | 2.380813 | 0.063779 |
| 100 | 2.338500 | 2.308991 | 0.080438 |
| 150 | 2.198100 | 2.208130 | 0.194193 |
| 200 | 2.065000 | 2.145902 | 0.201333 |
| 250 | 2.091600 | 2.092469 | 0.265112 |
| 300 | 2.069800 | 2.067296 | 0.273203 |
| 350 | 2.043400 | 2.039759 | 0.286054 |
| 400 | 2.042600 | 1.985015 | 0.322703 |
| 450 | 1.894900 | 1.889013 | 0.365540 |
| 500 | 1.796700 | 1.723411 | 0.468348 |
| 550 | 1.635400 | 1.543910 | 0.536411 |
| 600 | 1.281400 | 1.392313 | 0.617325 |
| 650 | 1.255100 | 1.285407 | 0.619705 |
| 700 | 1.159900 | 1.192592 | 0.637792 |
| 750 | 1.035800 | 1.106131 | 0.663018 |
| 800 | 1.083000 | 1.087730 | 0.684436 |
| 850 | 1.075600 | 1.076131 | 0.672537 |
| 900 | 1.126500 | 1.059961 | 0.680628 |
| 950 | 1.158300 | 1.009003 | 0.686340 |
| 1000 | 0.843300 | 1.052827 | 0.674441 |
| 1050 | 1.030000 | 0.998015 | 0.692527 |
| 1100 | 0.699000 | 0.981322 | 0.699667 |



| | | | |
|---|---|---|---|
| 1100 | 0.699000 | 0.981322 | 0.699667 |
| 1150 | 0.848000 | 0.992672 | 0.697287 |
| 1200 | 0.772000 | 0.996128 | 0.699191 |
| 1250 | 0.905900 | 0.970958 | 0.714422 |
| 1300 | 0.607700 | 0.974963 | 0.711566 |
| 1350 | 1.128300 | 0.932735 | 0.714898 |
| 1400 | 1.043900 | 0.960308 | 0.701095 |
| 1450 | 0.727700 | 0.950306 | 0.712042 |
| 1500 | 0.787200 | 0.935445 | 0.717277 |
| 1550 | 0.800100 | 0.938195 | 0.702999 |
| 1600 | 0.641000 | 0.960837 | 0.698239 |
| 1650 | 0.792100 | 0.953524 | 0.699191 |

Figure 3: Result of BERTurk with early stopping on a 80-20 split.

### 3.2 Regression Using BERT

For the regression task, we extended our approach by combining BERT embeddings with clustering and sentiment analysis. Key steps included:

1. **Clustering with GMMs:**
    - GMMs were employed to group similar data points, aiding in feature aggregation.
2. **Sentiment Analysis:**
    - We used bert-base-multilingual-uncased-sentiment to derive sentiment scores, which were integrated into the regression pipeline.

This combination of clustering and sentiment analysis with Gradient Boost pipelines resulted in a test Log MSE of approximately **0.15 with 5-fold cross validation**. While promising, this approach highlighted the need for further fine-tuning and additional features to achieve better accuracy.

## 4. Conclusion

In this project, we successfully developed and evaluated machine learning pipelines for the classification and regression of Turkish Instagram data, achieving meaningful insights and performance improvements. Through the use of advanced preprocessing techniques, feature engineering, and model evaluation, we tackled challenges such as class imbalance, heavy-tailed distributions, and diverse account popularity.

The classification pipeline demonstrated the effectiveness of SVM models, consistently achieving **65% accuracy on test data with 5-fold cross-validation**. While BERT-based models, such as BERTurk, achieved higher training accuracy, their performance on overlapping test instances highlighted limitations in generalization, underscoring the robustness of SVM for this task.

For regression, the combination of feature engineering and ensemble modeling provided a robust framework, achieving a best **Log MSE of 0.15**. Clustering and sentiment analysis using BERT embeddings further enhanced our regression approach, demonstrating the potential of leveraging contextual information from textual data.

This work highlights the importance of iterative refinement, careful model selection, and domain-specific preprocessing in achieving effective machine learning solutions for real-world datasets. Future research could explore additional data sources, more sophisticated model architectures, and domain-specific features to further enhance the performance of both classification and regression pipelines.

## 5. References

dbmdz. (n.d.). *bert-base-turkish-cased*. Hugging Face. Retrieved January 12, 2025, from https://huggingface.co/dbmdz/bert-base-turkish-cased

nlptown. (n.d.). *bert-base-multilingual-uncased-sentiment*. Hugging Face. Retrieved January 12, 2025, from https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment

ViralLab. (2023). *TurkishBERTweet: Fast and Reliable Large Language Model for Social Media Analysis*. GitHub. Retrieved January 12, 2025, from https://github.com/ViralLab/TurkishBERTweet