“Sentiment And Social Media Analysis”

As The

Assessment submitted

By

Yogesh Sharma

(Machine Learning Intern)

Yogsharma1207@gmail.com

+91-96699-36028

INTRODUCTION

1.1 OVERVIEW

This project focuses on the task of predicting emotion intensity in tweets, leveraging a dataset provided by the competition hosted on the Codalab platform. The objective is to develop models that can accurately estimate the degree of various emotions, including joy, sadness, fear, and anger, expressed in tweets. Our approach involves exploring both purely statistical methods and deep learning techniques to tackle the problem. We preprocess the tweet data using TF-IDF vectorization and employ a variety of regression models, including Lasso regression, SVM regression, and Convolutional Neural Network (CNN). Through extensive experimentation and evaluation, we aim to identify the most effective approach for emotion intensity prediction. The findings of this study will contribute to understanding the nuances of emotion expression in text data and advance the development of computational models for sentiment analysis and emotion detection in social media content.

The project dives into the world of sentiment analysis, focusing on understanding how people express their feelings on Twitter. We're interested in figuring out how intense emotions like joy, sadness, fear, and anger are when people tweet. To do this, we got tweets from different emotions and see how strongly those emotions come across.

Here's a breakdown of what we're doing:

1. **Getting the Data Ready**: Once we have the tweets, we clean them up and prepare them for analysis.
2. **Finding Important Clues**: We look at the words and phrases used in the tweets to understand what kind of characters and special symbols included in the tweets.
3. **Building Models**: We use different methods like Support Vector Regression (SVR), Lasso Regression, and Convolutional Neural Networks (CNN) to create models that predict the intensity of emotions in tweets.

4. **Testing and Hyper-parameter Tuning**: We test our models to make sure they're accurate by selecting the best parameters for prediction.
5. **Sharing Our Discoveries**: Finally, we're documenting everything we've learned in this technical paper so others can understand our process and findings.

DATA

Data Description:

For our analysis, we utilize the training and test datasets, disregarding the development data due to its limited size compared to the comprehensive training and test sets.

Training Dataset: The training dataset comprises four distinct files, each categorized by emotion (anger, fear, joy, sadness). These files are tab-separated text files.

1. The Anger set has a shape of (857, 4).
2. The Joy set has a shape of (823, 4).
3. The Fear set has a shape of (1147, 4).
4. The Sadness set has a shape of (786, 4). Each file contains four columns: Id, tweet, emotion, and score (intensity).

Test Dataset: Similar to the training dataset, the test dataset is also divided into four classes.

1. The Anger set has a shape of (760, 4).
2. The Joy set has a shape of (714, 4).
3. The Fear set has a shape of (995, 4).
4. The Sadness set has a shape of (673, 4). The test dataset columns include Id, tweet, emotion, and score (intensity).

Data Types:

1. Id (dtype: int64): The id serves as a sequence number for tweets and is not considered a feature for our model.
2. Tweets (dtype: object): This column contains the actual tweet text, including random emojis and characters.
3. Emotion (dtype: object): Emotion classes are predetermined categories to which tweets belong (Anger, Joy, Fear, Sadness).
4. Score (dtype: float64): The score column serves as the label (dependent variable) for training the model, aiming for improved accuracy and reduced error rates.

Preprocessing:

1. **Tokenization**:
    - **Importance**: Tokenization involves breaking down text into smaller units such as words or subwords. It helps in standardizing the input text and splitting it into meaningful units that the model can understand.
    - **Effect on Data**: Tokenization transforms raw text into a structured format, making it easier for the model to process and extract features.
2. **Removal of Special Characters and Punctuation**:
    - **Importance**: Special characters and punctuation do not usually contribute to the sentiment or emotion in text and can introduce noise. Removing them helps in focusing on the essential words and context.
    - **Effect on Data**: This step results in cleaner text data, reducing irrelevant information that could confuse the model during training.
3. **Lowercasing**:
    - **Importance**: Lowercasing ensures consistency in the text by converting all characters to lowercase. It helps in treating words with different cases as the same and reduces the vocabulary size.
    - **Effect on Data**: Lowercasing prevents the model from treating the same word with different cases as distinct entities, leading to more accurate representations of words in the vocabulary.

4. **Stopword Removal**:
   - **Importance**: Stopwords are common words such as "and," "the," "is," etc., that occur frequently in the language but often do not carry significant meaning. Removing stopwords reduces noise and focuses on content-bearing words.
   - **Effect on Data**: Eliminating stopwords helps in reducing the dimensionality of the data and improving the efficiency of the model by focusing on relevant words.
5. **Stemming or Lemmatization**:
   - **Importance**: Stemming and lemmatization aim to reduce words to their root form, simplifying the vocabulary and capturing the essence of words regardless of their inflections or variations.
   - **Effect on Data**: By reducing words to their base forms, stemming or lemmatization ensures that variations of the same word are treated as identical, enhancing the model's ability to generalize and make accurate predictions.
6. **Vectorization (e.g., TF-IDF)**:
   - **Importance**: Vectorization converts text data into numerical representations that machine learning models can process. TF-IDF (Term Frequency-Inverse Document Frequency) assigns weights to words based on their frequency in a document and across the entire corpus.
   - **Effect on Data**: Vectorization transforms textual features into numerical features, enabling machine learning models to understand and learn from the data. TF-IDF assigns higher weights to words that are more important in a specific document but less common across the entire corpus, capturing the unique characteristics

Models Used:

Statistical Machine Learning Models:

1. Lasso Regression
2. Ridge Regression
3. SVM Regression
4. Decision Tree
5. Random Forest

Deep Learning Models:

1. Classical Neural Networks (Multilayer Perceptron)
2. Convolutional Neural Network (CNN)

Approach:

A. **Objective of Multiple Regressions:**

- The inclusion of multiple regression models aims to enhance prediction accuracy by leveraging diverse algorithms' strengths and characteristics. Each model offers unique perspectives on the data, contributing to a comprehensive understanding and more precise predictions.

B. **Hyper-parameter Tuning:**

- Hyper-parameter tuning has been diligently performed for relevant models to optimize their performance. This iterative process involves systematically adjusting model parameters to identify the configuration that maximizes predictive accuracy and generalization to unseen data.

Observation & Result:

Based on the provided results for different regression models, here are the observations and findings:

1. **Ridge Regression Model:**

    - Both the train and test mean squared errors are relatively low compared to other models, indicating good performance in capturing the variance in the data.

- The mean absolute error for both train and test sets is moderate, suggesting that the model's predictions are reasonably close to the actual values.

2. **Lasso Regression Model:**

   - The mean squared error for both train and test sets is higher compared to the Ridge Regression model, indicating slightly poorer performance.
   - The mean absolute error is also higher, suggesting that the model's predictions deviate more from the actual values.

3. **Decision Tree Regression Model:**

   - The mean squared error is higher than that of the Ridge Regression model but lower than the Lasso Regression model.
   - The mean absolute error is moderate, indicating reasonable performance.

4. **Random Forest Regression Model:**

   - The mean squared error is the lowest among all models, suggesting better performance in capturing the variance in the data.
   - The mean absolute error is also low, indicating relatively accurate predictions.

5. **SVM Regression Model:**

   - The mean squared error is low, similar to the Ridge Regression model, indicating good performance.
   - The mean absolute error is moderate, suggesting reasonably accurate predictions.

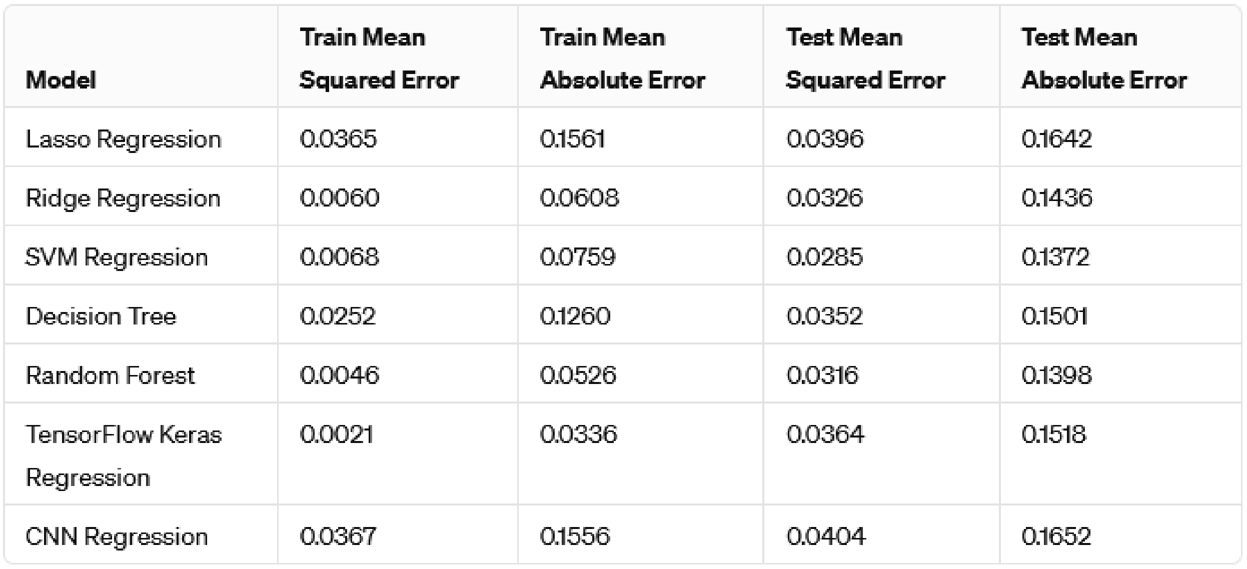
6. **TensorFlow Keras Regression Model:**

   - The mean squared error is very low, indicating excellent performance in capturing the variance in the data.

- The mean absolute error is also very low, suggesting highly accurate predictions.

7. **CNN Regression Model:**

- The mean squared error is relatively high compared to other models, indicating poorer performance.
- The mean absolute error is also high, suggesting larger deviations from the actual values.



Model	Train Mean Squared Error	Train Mean Absolute Error	Test Mean Squared Error	Test Mean Absolute Error
Lasso Regression	0.0365	0.1561	0.0396	0.1642
Ridge Regression	0.0060	0.0608	0.0326	0.1436
SVM Regression	0.0068	0.0759	0.0285	0.1372
Decision Tree	0.0252	0.1260	0.0352	0.1501
Random Forest	0.0046	0.0526	0.0316	0.1398
TensorFlow Keras Regression	0.0021	0.0336	0.0364	0.1518
CNN Regression	0.0367	0.1556	0.0404	0.1652

The Random Forest Regression model and the TensorFlow Keras Regression model perform the best among all models, with low mean squared errors and mean absolute errors on both train and test sets. The CNN Regression model shows relatively poorer performance compared to other models, with higher mean squared errors and mean absolute errors