Unified Mentor Climate Change Modeling - Adimalla Nithin Siddhartha

Introduction

This document summarizes the work performed on the Climate Change Modeling project, which aims to develop a machine learning model to predict climate change indicators using historical data. The project involved analyzing historical climate data, preprocessing it for modeling, engineering relevant features, training various regression models, and evaluating their performance.

Data Description

The project utilized the climate_nasa.csv dataset. This dataset contains the following columns:

- **date**: The date and time of the observation.
- **likesCount**: The number of likes (this was the target variable for prediction).
- **profileName**: An anonymized identifier for the profile.
- **commentsCount**: The number of comments.
- **text**: The text associated with the observation.

Initial exploration of the data revealed that 'commentsCount' had missing values, which were imputed with the mean, and 'text' had missing values, which were filled with empty strings. The 'date' column was converted to a datetime object for time-based feature extraction.

Data Preprocessing and Feature Engineering

Before training the models, the data underwent several preprocessing and feature engineering steps:

- **Handling Missing Values**: Missing values in 'commentsCount' were filled with the mean of the existing values, and missing values in 'text' were filled with empty strings.
- **Splitting Data**: The dataset was split into training and testing sets using a 80/20 ratio.
- **Feature Engineering**: Time-based features ('year', 'month', 'day_of_week') were extracted from the 'date' column.
- **Preprocessing Pipeline**: A ColumnTransformer was used to apply transformations to different columns:
    - Numerical columns ('commentsCount') were scaled using StandardScaler.

- Categorical columns ('text') were encoded using OneHotEncoder. The 'profileName' and 'date' columns were dropped from the processed data for the initial modeling phase.

Model Training and Evaluation

Several regression models were trained to predict 'likesCount' using the preprocessed and engineered features. The models trained were:

- Linear Regression
- Ridge Regression
- Lasso Regression
- Elastic Net
- Decision Tree Regressor
- Random Forest Regressor
- Gradient Boosting Regressor

The models were evaluated using the following metrics on the test set:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R-squared (R2)

The evaluation results are as follows:

Model	**MSE**	**RMSE**	**MAE**	**R2**
Linear Regression	190.1530	13.7896	6.6525	-0.0583
Ridge	161.7344	12.7175	5.0258	0.0998

Model	MSE	RMSE	MAE	R2
Lasso	165.1268	12.8502	5.2834	0.0809
ElasticNet	167.6816	12.9492	5.4643	0.0667
Decision Tree	169.5280	13.0203	4.1443	0.0565
Random Forest	169.9385	13.0360	3.6772	0.0542
Gradient Boosting	164.2079	12.8144	4.4973	0.0861

Based on the R2 scores, which represent the proportion of the variance in the dependent variable that is predictable from the independent variables, all models performed poorly with very low or even negative R2 values. Ridge, Lasso, and Gradient Boosting showed slightly better performance compared to others, but the overall predictive power is low. The MAE values indicate the average absolute difference between the predicted and actual likes counts.

Conclusion and Future Work

The initial modeling attempt to predict 'likesCount' yielded models with limited predictive power, as indicated by the low R2 scores. This suggests that the current set of features may not be sufficient to accurately predict the number of likes.

Future work could involve:

- **Exploring more advanced feature engineering**: Incorporating natural language processing (NLP) techniques to extract features from the 'text' column, as the content of the text likely influences engagement (likes and comments).
- **Considering other models**: Experimenting with different types of regression models or techniques suitable for potentially skewed target variables.
- **Hyperparameter tuning**: Optimizing the hyperparameters of the better-performing models (Ridge, Lasso, Gradient Boosting) to potentially improve their performance.

- **Acquiring additional data**: Including more relevant features or a larger dataset could significantly improve model performance.