PROJECT REPORT
Prediction of YouTube Views using Sentiment Analysis

Team:
Sanjana Gowda Hetthur Chandrashekar (2114504)
Vishnu Vamshi Vidyapathi Sharma (2116673)

Dataset: YouTube Data

Contents:

Abstract – We predicted the YouTube view counts using Sentiment Analysis. Leveraging the YouTube Data API v3, we extracted sentiment data from 100 comments on a diverse set of videos, that included the most liked, most disliked, and a random selection. The sentiment analysis was conducted using the Valence Aware Dictionary and sEntiment Reasoner (VADER), which provided polarity scores indicative of the commenters' sentiments. These polarity scores were then utilized as a feature in our dataset, serving as a proxy for audience reaction. By incorporating sentiment data into our machine learning models, we aimed to establish a predictive relationship between the sentiment of video comments and the subsequent view counts.

1. Introduction

In the last decade, YouTube's influence as a video-sharing platform has been second to none, with billions of users turning to it for entertainment, information, and education. The number of views a video garners serves as a measure to its popularity and audience engagement. View counts have been considered a direct measure of content success, but the underlying factors contributing to these

numbers remain complex. Our project delves into the predictive analysis of YouTube view counts in a unique perspective: the sentiment of the audience as expressed in the comments section. We utilized the YouTube Data API v3 to extract comments from a wide array of videos, including the most liked, most disliked, and a random sampling. The sentiment analysis was performed using VADER (Valence Aware Dictionary and sEntiment Reasoner), a tool adept at deciphering the nuances of social media language. The polarity scores derived from VADER, indicative of the positive or negative sentiment of comments, were then incorporated as a feature into our dataset. This integration allowed us to explore the potential of sentiment as a predictor for view counts. This report outlines the methodology employed in gathering and analyzing the data, the machine learning models used for prediction, and the implications of our findings. Through this study, we aim to contribute to the understanding of content performance on YouTube and offer a predictive tool that could aid content creators in strategizing their releases for maximum impact.

2. Literature Review

In our literature review, we focus on studies that have utilized machine learning techniques to forecast YouTube video performance metrics.One of the works in this domain is a study that conducted an exploratory data analysis (EDA) on YouTube's trending videos across five different countries. The researchers aimed to understand the impact of various factors such as likes, dislikes, comments, and shares on video view counts. They employed a range of machine learning models, including Multiple Linear Regression (MLR), Random Forest Regressor (RFR), Decision Tree Regressor (DTR), XGBoost Regressor (XGB), and Gradient Boosting Regressor (GBR), to predict the view counts. Their findings highlighted the superior performance of the Random Forest technique over other models in predicting view counts. This study aligns with our project's objective to predict view counts using sentiment analysis of comments. It underscores the potential of machine learning models in understanding and leveraging user-generated data to forecast content performance on digital platforms. Our project extends this line of inquiry by incorporating sentiment analysis as a predictive feature, which has been less explored in the literature.

3. Methodology

3.1 Data Collection

Our data collection process was designed to harness the predictive power of sentiment analysis for YouTube view counts. Utilizing the API , we initiated the extraction of comments from videos that was found from the video IDs identified within our Kaggle dataset we initially started with. The videos were categorized into three distinct groups: the 8000 most liked, 8000 most disliked, and 8000 randomly selected videos which we sorted from the Kaggle dataset. Due to various constraints such as disabled comments, duplicate video IDs, and deleted comments, we successfully retrieved 1239 most liked videos' comments, 1456 most disliked videos' comments and 1697 random videos' comments. For each video, we collected 100 comments, ensuring a substantial sample size for our sentiment analysis. To enhance the quality of our dataset, we implemented a filtering process that removed hyperlinks and prioritized text-rich comments over those containing primarily emojis. This step was crucial in maintaining the relevance and integrity of the data, as it directly influenced the accuracy of the sentiment analysis performed by VADER. Only the rows with these videos and rest of the features from the original Kaggle dataset is considered to work with and the view count is our target variable.

3.2 Sentiment Analysis and Dataframe Construction

Having collected and preprocessed the comments, we proceeded to the sentiment analysis phase. Utilizing VaderSentiment, we calculated the average sentiment polarity for the comments associated with each video. This computation provided us with a quantitative measure of the overall sentiment for each video. The average polarity scores were then matched with their respective video IDs, resulting in the creation of a new dataframe which served as the foundation for our predictive analysis, correlating sentiment with potential view counts. The newly constructed dataframe consisted a total of 16 features across 4152 rows.

3.3 Data Exploring and Preprocessing

Now that we have our dataset ready, we proceeded with data exploring and preprocessing to understand and make use of any insights it may provide us.

Explored data to assess dataset integrity, summarizing key statistics, checking for null values, and verifying null values. We also checked for correlations of our target feature with other features alongside other plots which provided us great information as to what data we are dealing with. For preprocessing we imported the video category names in its category id for a better understanding for analysis and split the dataset into 80% training and 20% testing.

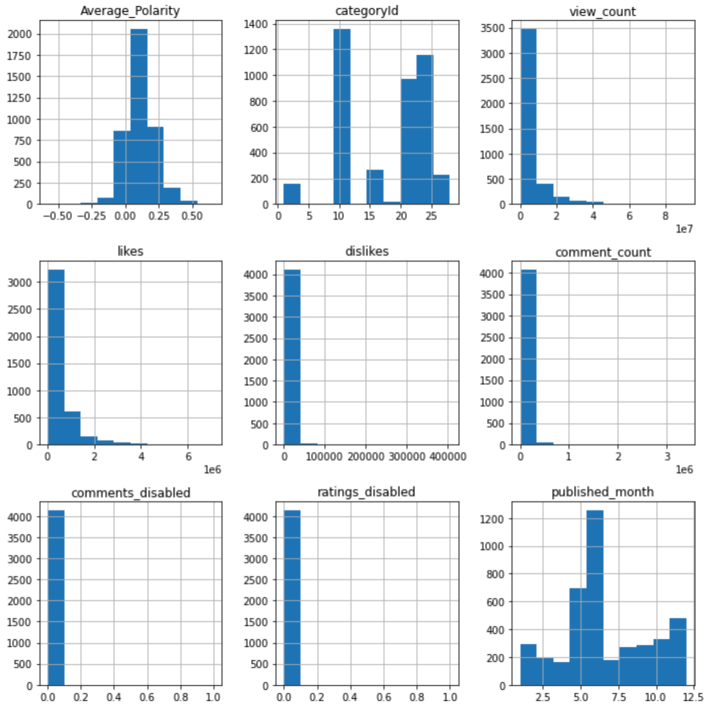

Fig 1. Histograms for numerical columns

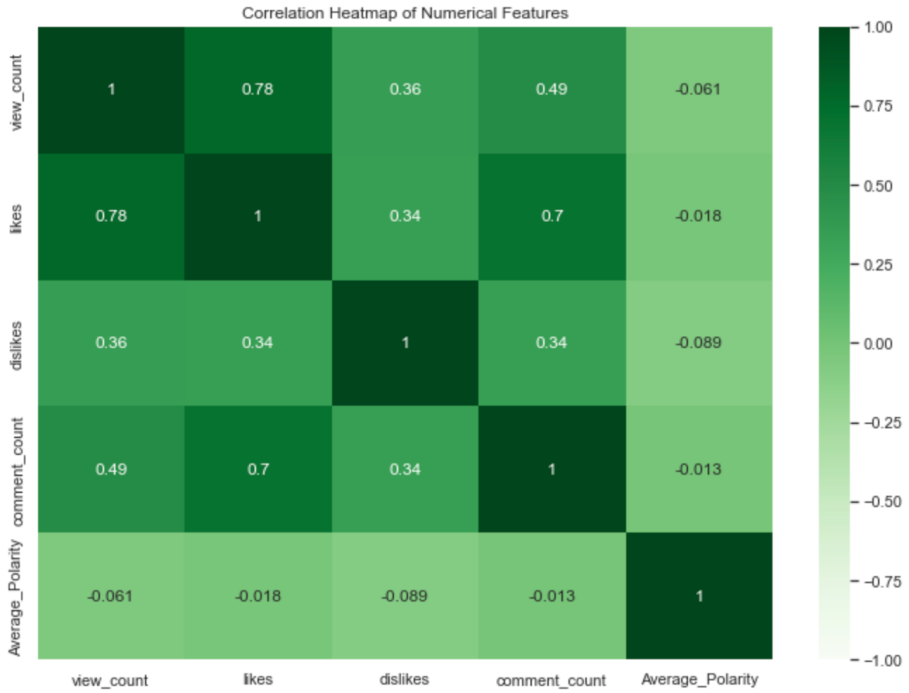

Fig 2. Correlation plot

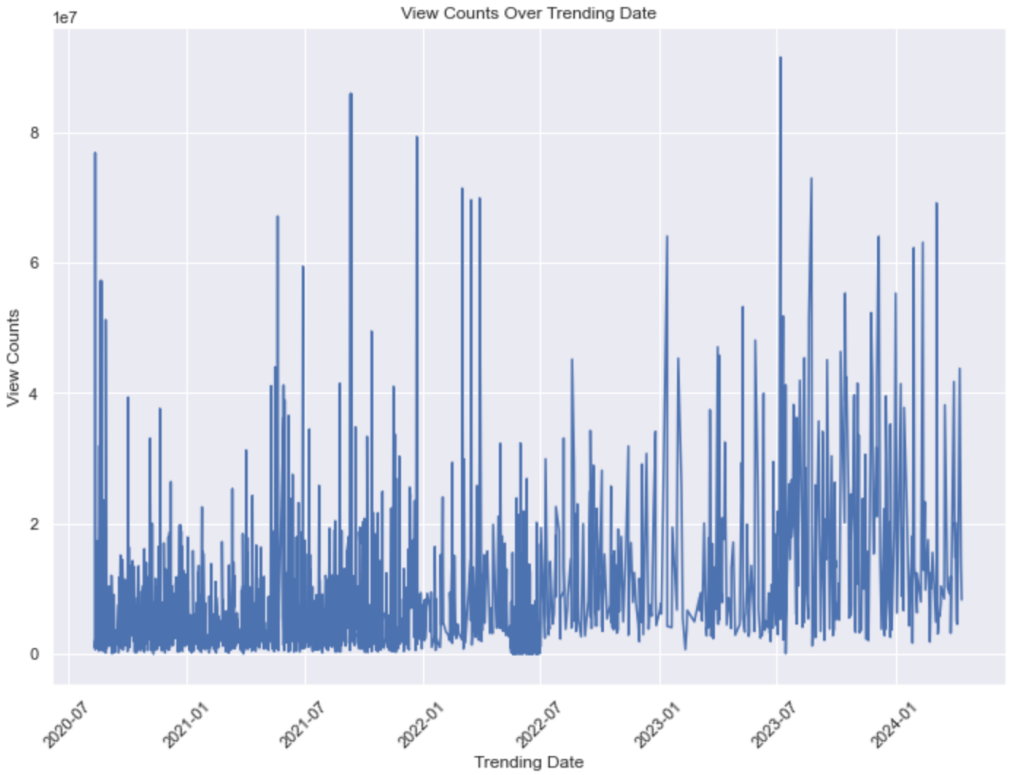

Fig 3. View Counts over trending date

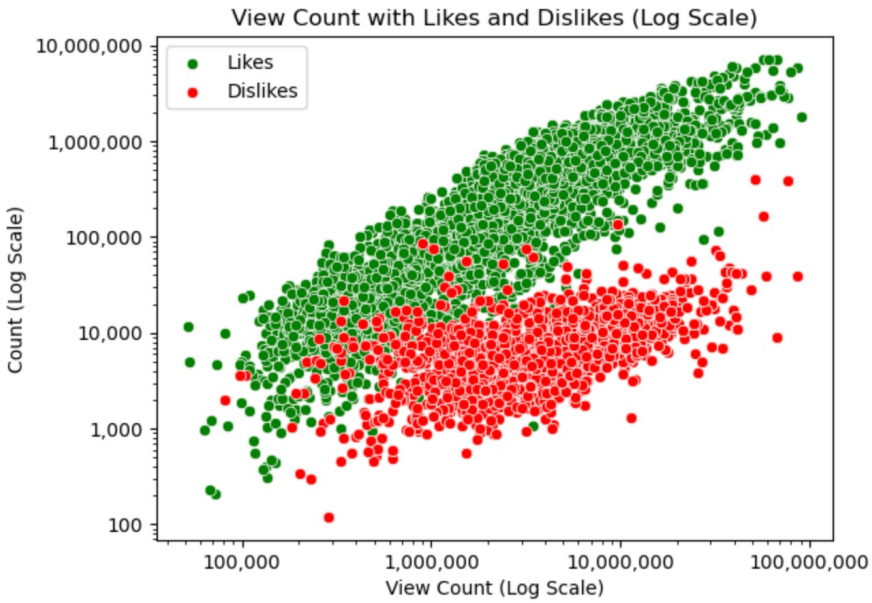

Fig 4. View Count with Likes and Dislikes

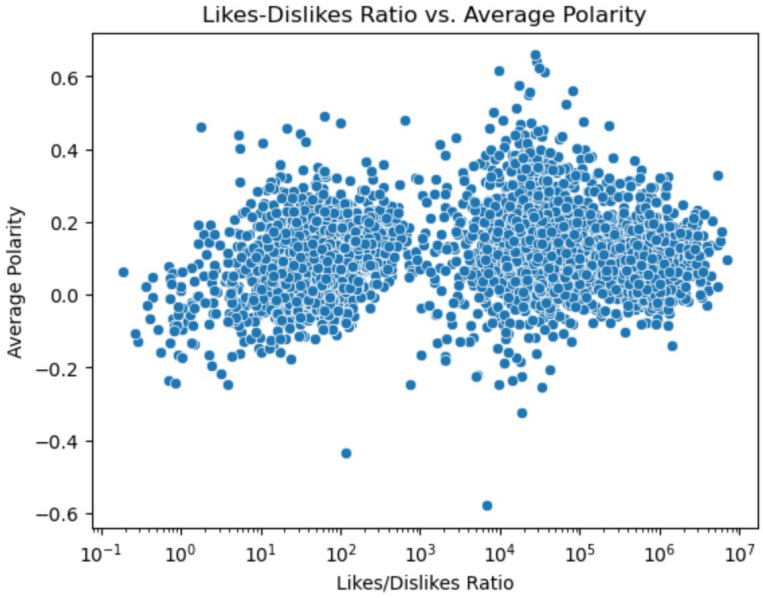

Fig 5. Likes-Dislikes Ratio vs. Average Polarity

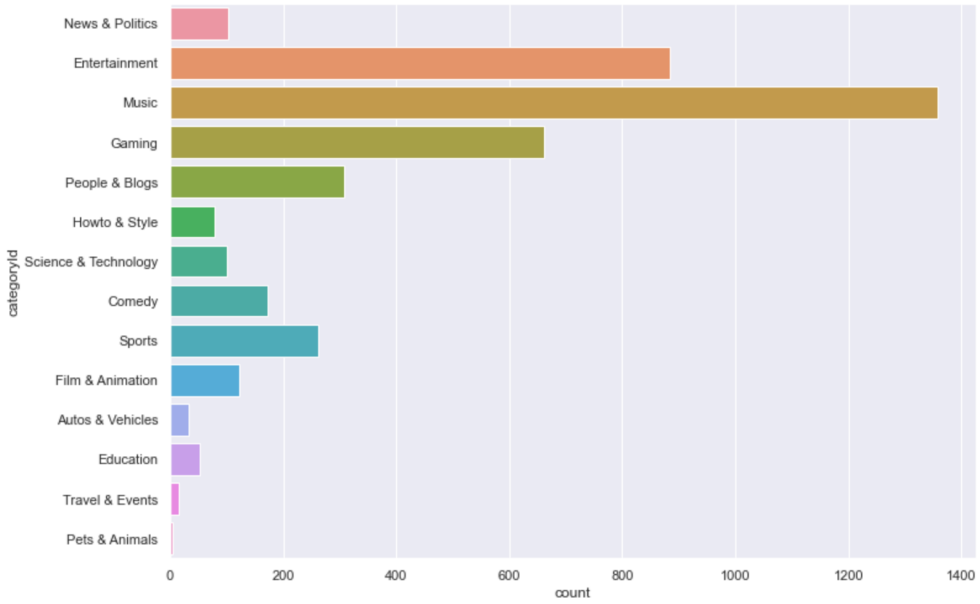

Fig 6. Category Count

3.4 Linear Regression

Now, we move to our prediction techniques. We start with Linear Regression to predict the view counts with relevant features and the average polarity scores. Linear regression is a statistical method that models the relationship between a dependent variable and one or more independent variables, providing a way to predict outcomes and understand the strength and form of the relationship. Performed Linear Regression to predict view count using features Average Polarity, category ID, likes, dislikes, comment count, comments disabled, ratings disabled, published month as our predictors.
Model Evaluation: The trained model was then evaluated using the testing subset of our dataset. We used metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared to assess the accuracy and performance of the model. We see the results in Fig 7.

3.5 Decision Tree Regression

We employed Decision Tree Regression as one of our predictive algorithms next. Decision Tree Regression is a non-parametric supervised learning method that is widely used for its simplicity and interpretability. It works by partitioning the

dataset into subsets based on feature value conditions, which are then represented as branches of the tree.
Model Training: We trained our Decision Tree Regression model on the training subset of our dataset. The model learned to make predictions based on the features, including the average sentiment polarity scores and other relevant features.
Model Evaluation: The trained model was then evaluated using the testing subset of our dataset. We used metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared to assess the accuracy and performance of the model. We see the results in Fig 8.

3.6 Random Forest Regression

Building on the insights gained from Decision Tree Regression, we further refined our predictive modeling by employing Random Forest Regression. This ensemble learning method combines multiple decision trees to improve the predictive accuracy and control over-fitting, which is a common challenge with single decision trees.
Model Training: The Random Forest model was trained using the same dataset that was utilized for the Decision Tree Regression. The ensemble approach aggregates the results of numerous decision trees trained on different subsets of the dataset, each providing individual predictions.
Model Evaluation: The effectiveness of the Random Forest model was evaluated against our testing data. We continued to use robust metrics like MSE, RMSE, and R-squared to quantify the model's performance and compare it with the Decision Tree model. We see the results in Fig 9.

3.7 Ridge Regression

We implemented Ridge Regression, a technique that addresses multicollinearity in data through regularization. By introducing a penalty term, Ridge Regression shrinks the coefficients of less important features and helps in preventing overfitting.
Model Training: The Ridge Regression model was trained on our dataset, which included sentiment polarity scores and other video-related features. This model is particularly adept at handling datasets with a large number of features, as is the case with our YouTube comments analysis.

Model Evaluation: We evaluated the performance of the Ridge Regression model using the same metrics applied to previous models: MSE, RMSE, and R-squared. We see the results in Fig 10.

3.8 K-Nearest Neighbors Regression

We next incorporated K-Nearest Neighbors (KNN) Regression. KNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation.
Model Training: We trained the KNN Regression model using our dataset, which included the sentiment polarity scores among other features. KNN operates by finding the 'k' most similar instances (neighbors) in the training data to a new instance and deriving the output by averaging the values of these neighbors.
Model Evaluation: The KNN model was evaluated using the same metrics as our previous models: MSE, RMSE, and R-squared. We see the results in Fig 11.

3.9 Gradient Boosting Regression

This powerful machine learning technique builds an ensemble of decision trees in a sequential manner, where each subsequent tree aims to correct the errors made by the previous ones.
Model Training: The Gradient Boosting model was trained on our dataset, which included sentiment polarity scores and other video-related features. The model iteratively added trees that addressed the residuals of the predictions, refining the accuracy with each step.
Model Evaluation: We assessed the performance of the Gradient Boosting model using our standard set of metrics: MSE, RMSE, and R-squared. These metrics provided a comprehensive view of the model's predictive power and its ability to generalize to unseen data. We see the results in Fig 12.

3.10 Hyperparameter Tuning

3.10.1 Random Forest Regression

To fine-tune our Random Forest Regression model, we employed RandomizedSearchCV, exploring a range of hyperparameters to enhance model performance. We varied the number of trees between 50 and 200,

the tree depth from 5 to 50, and the minimum number of samples for splitting and for a leaf node between 2 to 20 and 1 to 20, respectively. Over 100 iterations, with 5-fold cross-validation and parallel processing, we assessed the models using negative mean squared error, identifying the optimal combination of hyperparameters that minimized prediction error and improved accuracy.

3.10.2 Gradient Boosting Regression

For the Gradient Boosting Regression model, we applied hyperparameter tuning using RandomizedSearchCV to navigate through a broad spectrum of hyperparameters. We adjusted the number of estimators from 50 to 200, the learning rate between 0.01 to 0.5, and the tree depth with options of 3, 5, 10, or unlimited. We also varied the minimum number of samples required for splitting from 2 to 20 and for a leaf node from 1 to 20. Over 100 iterations and 5-fold cross-validation, we evaluated the models based on negative mean squared error, seeking the optimal hyperparameters to minimize error and enhance the model's predictive accuracy.

3.11 Stacking

Here, we utilized a stacking ensemble method to enhance the predictive accuracy of YouTube view counts, combining the strengths of Gradient Boosting and Random Forest Regression models. We trained both base models with 100 estimators, ensuring robust predictions. A Linear Regression model served as the meta-model, learning to optimally blend the base models' predictions. The Stacking Regressor was then fitted on our training data, allowing the meta-model to adjust the final predictions based on the base models' outputs. This stacked model approach aimed to capitalize on the individual models' predictive capabilities, resulting in a more accurate and generalizable final model.

4. Results

```
Linear Regression:
Training R^2 Score: 0.6311938528971844
Testing R^2 Score: 0.6291407022511248
Mean Squared Error: 37286600558277.31
Root Mean Squared Error: 6106275.506253981
```

Fig 7. Linear Regression Results

The linear regression model demonstrates a moderate ability to explain the variance in the target variable with an R^2 score above 0.6 for both training and testing datasets. However, the high values of MSE and RMSE highlight significant prediction errors, suggesting that while the model captures some patterns in the data, there might be room for improvement. Potential steps to enhance the model could include feature engineering, outlier removal, or exploring more complex algorithms

```
Decision Tree Regression:
Training R^2 Score: 1.0
Testing R^2 Score: 0.6996961578311628
Mean Squared Error: 30192877668251.34
Root Mean Squared Error: 5494804.60692201
```

Fig 8. Decision Tree Regression Results

The decision tree regression model achieved an R^2 score of 1.0 on the training data, indicating perfect fit and capturing all the variance. However, on the testing data, the R^2 score dropped to 0.6997, suggesting reduced generalization ability. The model's Mean Squared Error (MSE) on the test set is 30,192,877,668,251.34, and the Root Mean Squared Error (RMSE) is 5,494,804.61. While the model performs excellently on training data, the disparity in R^2 scores and the substantial RMSE indicate potential overfitting, implying it captures noise rather than underlying patterns, which impacts its prediction accuracy on unseen data.

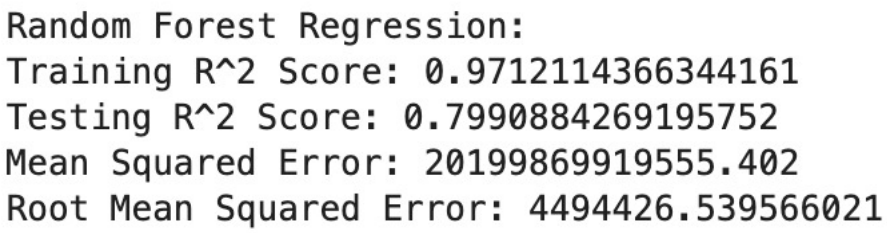

Fig 9. Random Forest Regression Results

The random forest regression model shows strong performance, with an R^2 score of 0.9712 on the training data, indicating that it explains 97.12% of the variance. On the testing data, it maintains a high R^2 score of 0.7991, reflecting good generalization with 79.91% of the variance explained. The model's Mean Squared Error (MSE) on the test set is 20,199,869,919,555.40, and the Root Mean Squared Error (RMSE) is 4,494,426.54. These metrics suggest that the random forest regression model provides accurate predictions with relatively lower error compared to other models, balancing both bias and variance effectively.

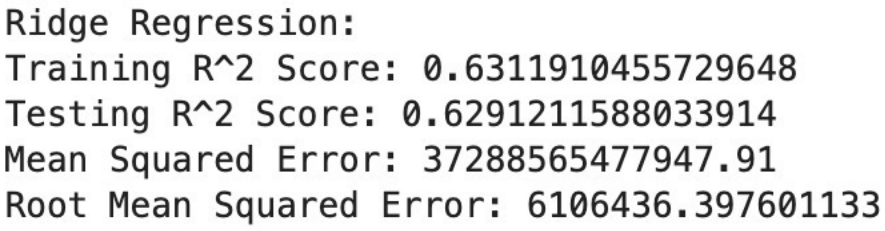

Fig 10. Ridge Regression Results

The ridge regression model achieves an R^2 score of 0.6312 on the training data, indicating that it explains 63.12% of the variance. On the testing data, the R^2 score is slightly lower at 0.6291, demonstrating consistent performance and good generalization. The model's Mean Squared Error (MSE) on the test set is 37,288,565,477,947.91, and the Root Mean Squared Error (RMSE) is 6,106,436.40. These metrics are similar to those of the linear regression model, suggesting that ridge regression, while regularizing the model to prevent overfitting, still results in significant prediction errors.

```
K-Nearest Neighbors Regression:
Training R^2 Score: 0.752641140200159
Testing R^2 Score: 0.6182095461770813
Mean Squared Error: 38385630979375.29
Root Mean Squared Error: 6195613.850085824
```

Fig 11. KNN Regression Results

The k-nearest neighbors (KNN) regression model achieves an R^2 score of 0.7526 on the training data, indicating that it explains 75.26% of the variance. However, the R^2 score drops to 0.6182 on the testing data, suggesting a decrease in model performance and generalization ability. The model's Mean Squared Error (MSE) on the test set is 38,385,630,979,375.29, and the Root Mean Squared Error (RMSE) is 6,195,613.85. These metrics indicate that while the KNN regression model fits the training data reasonably well, it experiences a notable drop in prediction accuracy on unseen data, likely due to overfitting or sensitivity to noise in the data.

```
Gradient Boosting Regression:
Training R^2 Score: 0.8601844714372964
Testing R^2 Score: 0.7794198674787509
Mean Squared Error: 22177368458430.77
Root Mean Squared Error: 4709285.344766313
```

Fig 12. Gradient Boosting Regression Results

The gradient boosting regression model demonstrates strong performance across multiple evaluation metrics. On the training data, it achieves an R^2 score of 0.8602, indicating it explains 86.02% of the variance in the target variable. When applied to the testing data, the model maintains a high R^2 score of 0.7794, demonstrating robust generalization with 77.94% of the variance explained. The Mean Squared Error (MSE) on the test set is 22,177,368,458,430.77, and the Root Mean Squared Error (RMSE) is 4,709,285.34. These metrics collectively highlight that the gradient boosting regression model not only fits the data well but also effectively minimizes prediction errors, making it a strong candidate for accurate predictions in practical applications.

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best Random Forest Model:
Training R^2 Score: 0.9709635473476095
Testing R^2 Score: 0.7988511480578977
Mean Squared Error: 20223726196558.54
Best Parameters: {'max_depth': 27, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 116}
Root Mean Squared Error: 4497079.7409606315
```

Fig 13. Random Forest Hyperparameter tuning

This random forest model with optimized hyperparameters demonstrates excellent performance. It achieves a high R^2 score of 0.97096 on the training data, indicating a strong fit and explanation of 97.10% of the variance. On the testing data, the model maintains a solid R^2 score of 0.79885, indicating good generalization ability with 79.89% of the variance explained. The MSE and RMSE values are 20,223,726,196,558.54 and 4,497,079.74 respectively, suggesting that the model's predictions are accurate and relatively low in error.

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best Gradient Boosting Model:
Training R^2 Score: 0.9580774196129591
Testing R^2 Score: 0.7733486648481622
Mean Squared Error: 22787773829872.77
Best Parameters: {'learning_rate': 0.05043648330859884, 'max_depth': 10, 'min_samples_leaf': 9, 'min_samples_split': 13, 'n_estimators': 178}
Root Mean Squared Error: 4773654.138065804
```

Fig 14. Gradient Boosting Hyperparameter tuning

This gradient boosting model demonstrates strong performance after hyperparameter optimization. It achieves a high R^2 score of 0.95808 on the training data, indicating a robust fit and explaining 95.81% of the variance. On the testing data, the model maintains a respectable R^2 score of 0.77335, suggesting good generalization ability with 77.34% of the variance explained. The MSE and RMSE values are 22,787,773,829,872.77 and 4,773,654.14 respectively, indicating the model's predictions are accurate with relatively low error.

5. Conclusion and Future Work

Stacking, an advanced ensemble technique, excels in generalization as demonstrated by its robust performance on the testing set. By leveraging the complementary strengths of multiple base regression models, stacking aggregates their predictions to achieve superior accuracy compared to any single model. This not only mitigates the limitations of individual models but also optimizes predictive performance, making it highly suitable for applications prioritizing precision.

In future, there are promising avenues for further improvement and exploration. One key focus could be on reducing the Mean Squared Error (MSE) by refining feature selection techniques or introducing novel features that capture additional relevant information. Furthermore, enhancing the interpretability of the stacked model would provide valuable insights into how different base models contribute to predictions, thereby fostering trust and facilitating informed decision-making in real-world scenarios. Overall, continuous refinement and innovation in stacking methodology hold significant potential to deliver more precise and actionable predictions in diverse applications.