Report on Regression Challenge 2

1. Overview of Findings

In this regression challenge, the objective was to minimize the loss, and models were evaluated based on their performance in reducing the score. Among all models:

- **Neural Networks** performed the worst with a score of **13,045,505.99** due to their complexity and sensitivity to hyperparameter tuning.
- **Linear Regression and Logistic Regression** trained quickly but were unable to achieve a score below **~13586287.71959**.
- Other models, such as **XGBoost Regressor** and **Gradient Boosting Regressor**, performed significantly better due to their ability to capture non-linear relationships and feature importance.

2. Model-Specific Performance

Linear Regression

- **Initial Approach:**
    - Trained as a baseline model to evaluate the dataset's linear separability.
    - Quick to train due to its simplicity.
- **Challenges:**
    - Could not capture non-linear relationships in the data.
    - Assumed feature independence, which was not true for this dataset.
- **Result:**
    - Achieved a score of **12,855,050**, which was better than Neural Networks but suboptimal compared to ensemble methods.

Logistic Regression

- **Initial Approach:**
    - Tested as another baseline model, even though it is typically used for classification tasks.
    - Similar results to Linear Regression due to shared assumptions about the data.
- **Result:**
    - Score was close to Linear Regression (**~12,855,050**), confirming that simple models were not suitable for this task.

KNN Regression

- **Initial Approach:**
    - Focused on local interpolation by predicting values based on the closest neighbors.

    - Tuned `n_neighbors` to balance overfitting and underfitting.
- **Challenges:**
    - Computationally expensive with larger datasets.
    - Sensitive to scaling, requiring normalization of features.
- **Result:**
    - Performed slightly better than baseline models, achieving a score of **12,800,000**, but lacked the sophistication of tree-based methods.

Regression Trees

- **Initial Approach:**
    - Used a single decision tree to model the data.
    - Captured non-linear relationships better than Linear or Logistic Regression.
- **Challenges:**
    - Prone to overfitting on the training data due to its nature.
- **Result:**
    - Improved the score to **12735459.87143** but was outperformed by ensemble methods.

Random Forest Regressor

- **Initial Approach:**
    - Combined multiple decision trees to reduce variance and improve generalization.
    - Tuned `n_estimators` and `max_depth` to optimize performance.
- **Result:**
    - Achieved a score of **12571692.07745**, significantly better than simpler models due to its ability to reduce overfitting.

AdaBoost Regressor

- **Initial Approach:**
    - Focused on sequentially correcting errors from previous weak models.
    - Tuned learning rate and the number of weak estimators.
- **Challenges:**
    - Sensitive to noisy data points, which could lead to overfitting.
- **Result:**
    - Improved performance slightly, with a score of **~12,600,000**.

Gradient Boosting Regressor

- **Initial Approach:**
    - Combined multiple weak models sequentially to correct errors.
    - Tuned `learning_rate`, `n_estimators`, and `max_depth` for optimal performance.
- **Result:**

    - Achieved a score of **12,750,000**, highlighting its capability to capture non-linear relationships effectively.

XGBoost Regressor

- **Initial Approach:**
    - Extended Gradient Boosting with regularization techniques to prevent overfitting.
    - Tuned hyperparameters extensively, including `learning_rate`, `n_estimators`, `gamma`, and `max_depth`.
- **Why It Succeeded:**

Data was normalised using MinMax Scalar

    - Handled feature importance and missing values effectively.
    - Included regularization to balance bias and variance.
    - The parameters which brought Success:

```
xgb = XGBClassifier(
    n_estimators=11000,
    learning_rate=0.004,
    max_depth=3,
    subsample=0.73,
    colsample_bytree=0.73,
    eval_metric='logloss',
    gamma=0.1,
    reg_alpha=0.01,
    reg_lambda=1
)
```

- **Result:**
    - Performed among the best, with a score of **12459426.04079**.

Neural Networks

- **Initial Approach:**
    - Designed a deep learning model with multiple layers and tested various architectures.
- **Challenges:**
    - Sensitive to hyperparameters like the number of layers, neurons, and activation functions.

    - Required extensive tuning and computational resources but failed to converge effectively.
    - Overfitted due to insufficient regularization and struggled with generalization.
- **Result:**
    - Worst performance, with a score of **13045505.99**, due to the lack of optimization and overcomplexity.

Stacking

- **Initial Approach:**
    - Combined predictions from multiple models (e.g., Random Forest, XGBoost) to improve performance.
    - Used Linear Regression as the meta-model to blend predictions.
- **Challenges:**
    - Computationally expensive and required careful model selection.
- **Result:**
    - Marginally improved the overall score, achieving **12690039.80453** but required significant resources. Please note that submission file on kaggle is named as (XGB_regressor_submission(1).csv) happened by mistake that file named was set misleading.

3. Challenges and Solutions

Challenge 1: Non-linear Relationships

- **Problem:** Linear models failed to capture complex relationships.
- **Solution:** Focused on tree-based and boosting methods that excel in capturing non-linear patterns.

Challenge 2: Overfitting in Neural Networks

- **Problem:** Neural Networks overfitted due to lack of proper regularization.
- **Solution:** Experimented with dropout, batch normalization, and tuning, but results remained poor due to time constraints.

Challenge 3: Feature Scaling

- **Problem:** Models like KNN and Neural Networks required proper feature scaling.
- **Solution:** Applied Min-Max scaling and normalization to standardize the data.

Challenge 4: Computational Complexity

- **Problem:** Boosting and stacking methods required significant time for training.
- **Solution:** Reduced `n_estimators` and implemented early stopping to optimize training time.

4. Conclusion

Best Performing Models: XGBoost , with score of **~12459426.04079**, .

- **Key Learnings:**
  - Simpler models (Linear Regression, Logistic Regression) are quick but struggle with non-linear data.
  - Advanced ensemble methods (Random Forest, XGBoost) provide robust performance when tuned effectively.
  - Neural Networks can fail if not optimized properly, especially in regression tasks with limited resources.