Assignment Report

1. Data Preprocessing

1.1 Importing Libraries and Loading Datasets

Importing essential libraries, including pandas, numpy, matplotlib, seaborn, and scikit-learn. It then loads the training dataset (df_train) and test dataset (df_test) using the pd.read_csv function.

1.2. Handling Missing Values

Handling missing values in the datasets. Ensuring that missing values are appropriately handled, either through imputation or removal, is a critical preprocessing step that may need to be incorporated based on the dataset characteristics.

1.3. Dropping Unnecessary Columns

The 'ID' column is dropped from the training dataset using df_train.drop('ID', axis=1). Removing irrelevant or non-contributing columns helps streamline the dataset and avoids noise in model training.

1.4. Separating Numerical and Categorical Features

Numerical and categorical features are separated using df_train.select_dtypes. Numerical features are stored in df_num, while categorical features are stored in df_cat.

1.5. Scaling Numerical Features (Min-Max Scaling)

Min-Max scaling is applied to numerical features using MinMaxScaler. The fit_transform method scales numerical features to a specified range (usually [0, 1]).

1.6. Identifying and Dropping Features with Zero Variance

Features with zero variance are identified by checking the variance of numerical columns (variance_df_num). Columns with zero variance are then dropped from the training dataset (df_num_variance_with_zero_drop).

1.7. One-Hot Encoding for Categorical Features

One-hot encoding is performed on categorical features using pd.get_dummies(df_cat). This converts categorical variables into a binary matrix, making them suitable for machine learning algorithms that require numerical input.

1.8. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is applied to reduce the dimensionality of the dataset. The code uses the PCA class from scikit-learn, specifying the number of components as 24. The transformed data is stored in df_pca.

2. Model Development

2.1. Choice of Regression Models

I used mainly two powerful regression models: Random Forest Regressor and XGBoost. Both models are capable of capturing complex relationships in the data and are known for their robust performance.

2.2. Random Forest Regressor

Training the Model: The Random Forest Regressor is instantiated with n_estimators=100 and random_state=42 and then trained on the training dataset using rf_model.fit(X_train, y_train).

Evaluation with RMSE: The model's performance is evaluated on a test set, and the Root Mean Squared Error (RMSE) is calculated using mean_squared_error. RMSE provides a measure of how well the model's predictions align with the actual target values.

2.3. Cross-Validation

Assessing Generalization Performance: Cross-validation is performed using cross_val_score to assess how well the Random Forest model generalizes to unseen data. The negative mean squared error is calculated and converted to RMSE for interpretation.

2.4. XGBoost Model

- **Training the XGBoost Model:** The XGBoost model is implemented using the xgb library. The feature set is divided into training and validation sets using train_test_split. The model is trained using the xgb.train function with specified parameters, including early stopping criteria.
- **Custom Evaluation Metric (R-squared):** The evaluation of the XGBoost model utilizes a custom metric, R-squared (r2_score). This metric measures the proportion of the variance in the dependent variable that is predictable from the independent variables. Maximizing R-squared is indicative of better model fit.
- **Early Stopping:** Early stopping is implemented to halt the training process if the model's performance on the validation set ceases to improve. This helps prevent overfitting and ensures the model is trained to its optimal point.

3.Challenges Faced

- **Feature Engineering:** It involves extensive feature engineering, including handling categorical data, scaling, and dimensionality reduction. This process can be challenging and requires careful consideration of each step.
- **Model Selection:** The choice of Random Forest and XGBoost models is reasonable, but it's important to consider other algorithms and compare their performances to ensure the best model is chosen.

4.Learnings Outcomes

- **Data Preprocessing Techniques:** I have learned various data preprocessing techniques, including scaling, one-hot encoding, and PCA. These are crucial steps in preparing data for machine learning models.
- **Model Evaluation:** I have learned RMSE for Random Forest and a custom R-squared metric for XGBoost. Understanding different evaluation metrics and selecting appropriate ones is a valuable skill in machine learning.
- **Cross-Validation:** Cross-validation is used to assess model performance. This helps in obtaining a more robust estimate of the model's performance on unseen data.