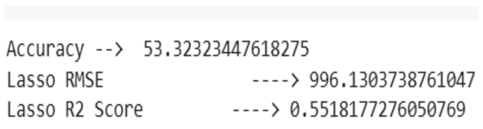
BIGMART SALES DATA

To forecast product sales at BigMart stores, this Python application uses a thorough machine learning approach. Data collection, exploratory analysis, feature engineering, thorough preprocessing, model training, and evaluation are all vital steps in the process, which ends with model export for real-world application.

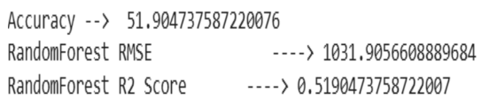
1. **Data Acquisition:** The script uses scikit-learn and XGBoost for machine learning tasks and pandas, NumPy, matplotlib, and seaborn for data management and visualization.Data loading is accomplished by using pandas' read_csv method to load training and test datasets straight from GitHub.
2. **Exploratory Data Analysis:** Data Inspection: To comprehend variable distributions, types, and missing value patterns, the script looks at each dataset's first few rows, shape, data kinds, and summary statistics. Visual Exploration: Provides information about feature distributions and possible anomalies by using a variety of plots, including count/pie plots for categorical features and histograms, KDE, boxplots, violin plots,for numerical features.
3. **Data Preprocessing:** Missing Value Handling, Outlier Identification and Removal, Feature Cleaning and Transformation were performed in this.
4. **Feature Engineering:** The ability to extract categories from item identifiers and engineer them as extra categorical predictors is a new feature. When identifier columns' informational value is entirely captured elsewhere, they are dropped.
5. **Model Preparation & Splitting:** divides the data after separating the predictors (X) and the goal (Y): Train/Test Split: 20% for assessment and 80% for training.
6. **Modeling: Regression Algorithms Implemented:**Linear Regression, Regularized Regression, Random Forest Regressor, XGBoost Regressor, Gradient Boosting were the ML models developed.
   Evaluation Metrics: For every model, calculates: The root mean squared error, or RMSE, measures the typical model prediction error. The goodness of fit is measured by the R2 Score (Coefficient of Determination). Transparency regarding overfitting is ensured by reporting both training and test performance.
7. **Model Export:**The process is completed by storing the top-performing Lasso model pipeline in a pickle file (BigMart_Sales_Prediction_Model.pkl) for use in subsequent scoring or deployment.

Linear Regression



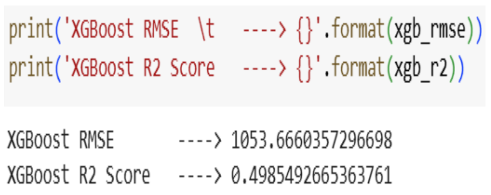
```
Accuracy -->  53.32323447618275
Lasso RMSE              ----> 996.1303738761047
Lasso R2 Score          ----> 0.5518177276050769
```

Random Forest



```
Accuracy -->  51.904737587220076
RandomForest RMSE           ----> 1031.9056608889684
RandomForest R2 Score       ----> 0.5190473758722007
```

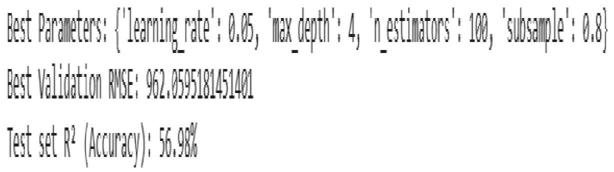
XG Boost



```
print('XGBoost RMSE  \t   ----> {}'.format(xgb_rmse))
print('XGBoost R2 Score   ----> {}'.format(xgb_r2))
```

```
XGBoost RMSE      ----> 1053.6660357296698
XGBoost R2 Score  ----> 0.4985492665363761
```

Gradient Boosting Regressor



```
Best Parameters: {'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 100, 'subsample': 0.8}
Best Validation RMSE: 962.059518145140l
Test set R² (Accuracy): 56.98%
```

8. **Conclusion:** Gradient Boosting Regressor (with grid search) is probably the best-performing model and has the highest predictive accuracy according to the code's selection procedure, test set R 2 R 2, and reporting of "Best Validation RMSE" following hyperparameter adjustment. However, for deployment purposes, the script saves the Lasso Regression pipeline, perhaps for pragmatic reasons.

   To sum up:

   - Performance/rank highest: Gradient Boosting Regressor (with grid search)
   - Exported model for real-world application: Lasso Regression pipeline
   - It is the Gradient Boosting Regressor that has the best prediction ability (lowest RMSE and highest test R 2 R 2).