Store Sales Prediction

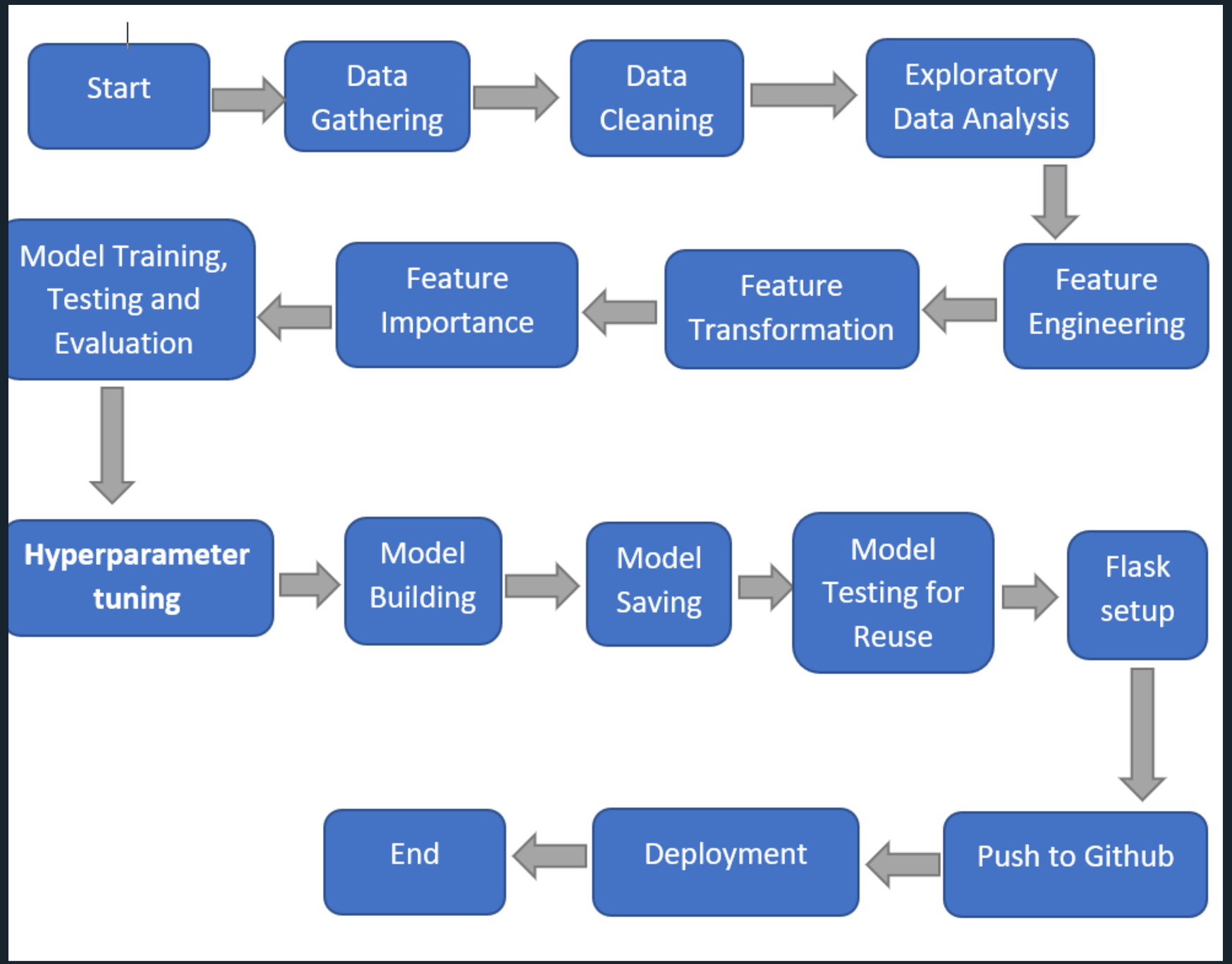
Problem Statement:

Nowadays, shopping malls and Big Marts keep track of individual item sales data in order to forecast future client demand and adjust inventory management. In a datawarehouse, these data stores hold a significant amount of consumer information and particular item details. By mining the data store from the data warehouse, more anomalies and common patterns can be discovered.

Approach: The classical machine learning tasks like Data Exploration, Data Cleaning, Feature Engineering, Model Building and Model Testing. Try out different machine learning algorithms that’s best fit for the above case.

Results: You have to build a solution that should able to predict the sales of the different stores of Big Mart according to the provided dataset.

Project Architecture:



Data Description:

- Item_Identifier: Unique product ID
- Item_Weight: Weight of product
- Item_Fat_Content: Whether the product is low fat or not
- Item_Visibility: The % of total display area of all products in a store allocated to the particular product.
- Item_Type: The category to which the product belongs
- Item_MRP: Maximum Retail Price (list price) of the product
- Outlet_Identifier: Unique store ID
- Outlet_Establishment_Year: The year in which store was established
- Outlet_Size: The size of the store in terms of ground area covered
- Outlet_Location_Type: The type of city in which the store is located
- Outlet_Type: Whether the outlet is just a grocery store or some sort of supermarket
- Item_Outlet_Sales: Sales of the product in the particulat store. This is the outcome variable to be predicted.

We have train (8523) and test (5681) data set, train data set has both input and output variable(s). We need to predict the sales for test data set.

Architecture Description

Data Gathering :

Data source - INeuron Internship Portal, Train and Test data are stored in .csv format.

Data Cleaning:

- Missing values imputated for Item Weight, Outlet Size and Item Visibility feature.
- Outliers - Log transformation applied on Item Visibility and Outlet Sales, to convert skewed data and handle outliers.

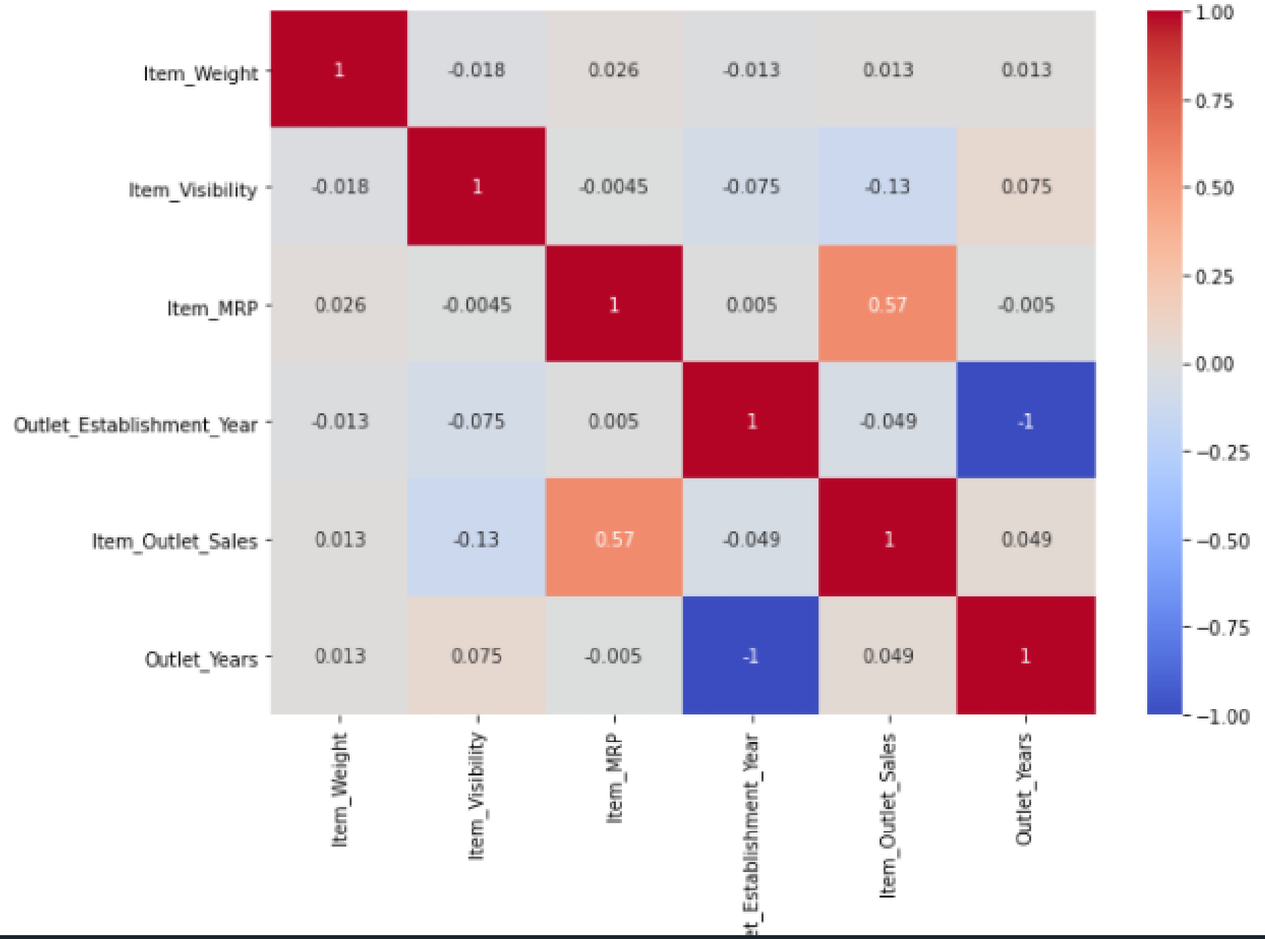
Correlation:

Feature Correlation

- Item MRP and Item Outlet Sales are positively correlated.
- Outlet Establishment Year is negatively correlated with Outlet Years - which is the new feature created subtracting establishment year with 2013.
- Outlet Establishment Year will be dropped from the model, replaced by Outlet Years(Outlet Years of Operations)

In [124]:
```
corr_data = train.corr()
plt.figure(figsize = (10,7))
sns.heatmap(corr_data,annot = True, cmap='coolwarm')
```

Out[124]: <AxesSubplot:>



Architecture Description

Feature Engineering:

- Creating new features from “Outlet Establishment Year” as “Outlet Year” subtract it with 2013 to get the age of the outlet when data was collected.
- New Item Type attribute to be created after extracting first 2 characters from the “Item Identifier” feature. FC: Food, DR: Drinks, and NC: Non-Consumables.
- Mapping and combining: Item Fat Content
- 'LF', 'low fat’ to “Low Fat”
- 'reg' to “Regular”

Architecture Description

Feature Transformation:

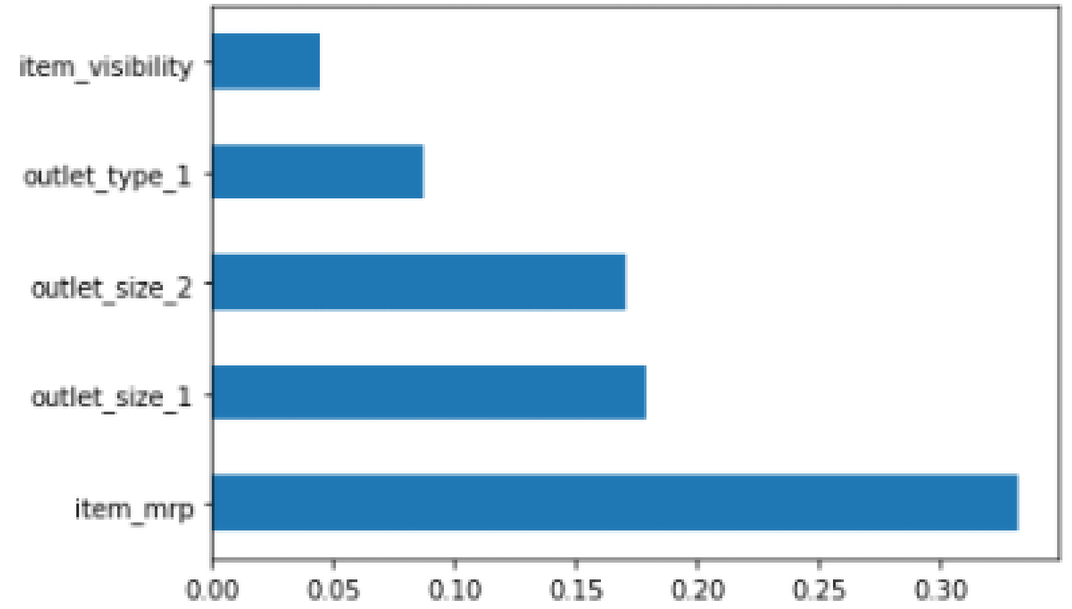
- Outlier handling - Log Transformation.
- Dropping “Outlet Establishment Year”, “Item Identifier” and “Outlet Identifier”.
- Label Encoding.
- One Hot Encoding.
- Feature Scaling.
  - scaling down the data of all the numerical variables to bring them into similar scale.

Architecture Description

Feature Importance :

Extra Trees Regressor to plot features having more impact on the outlet sales.

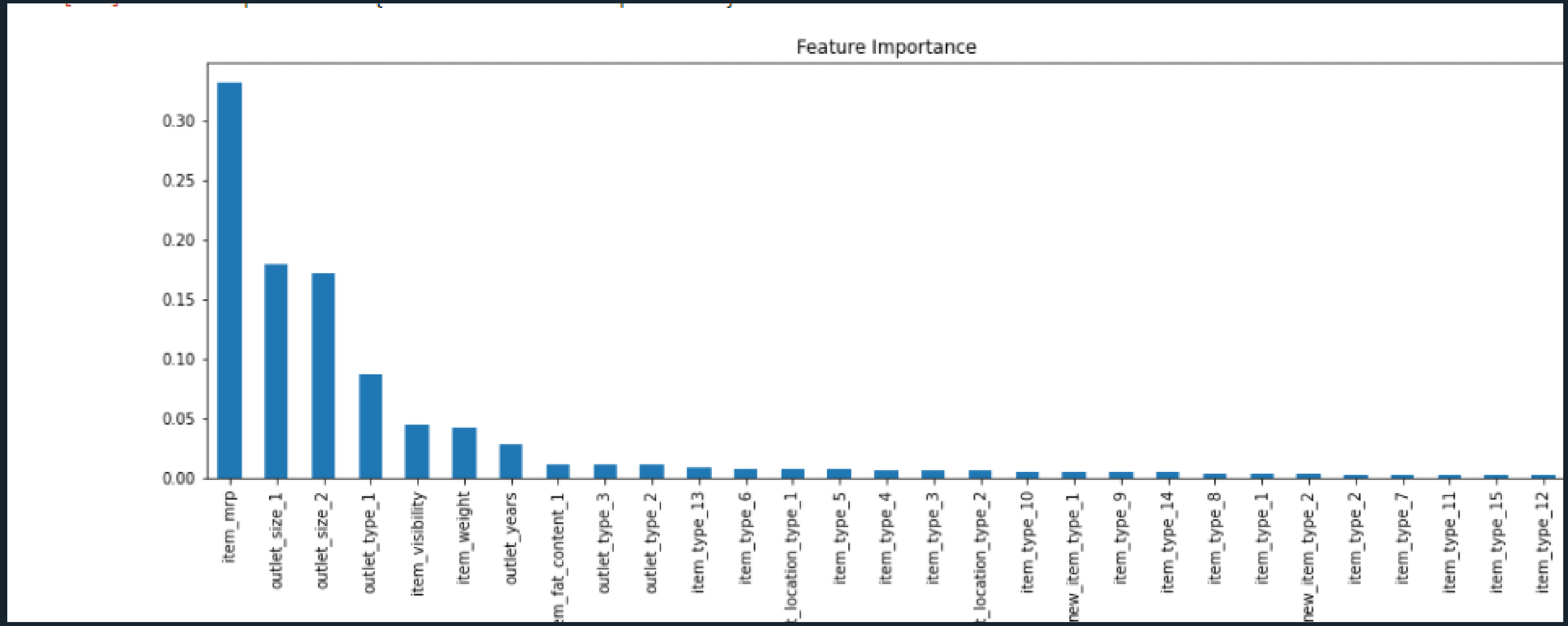
In [158]:
```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



Architecture Description

Feature Importance :

Extra Trees Regressor to plot features having more impact on the outlet sales.



Architecture Description

Model Training and Testing :

- Trained the model on Linear Regression, Ridge and Lasso Regression, and Random Forest Regressor.
- Best model with highest Prediction score (R Squared) and Lowest Error (Root Mean Squared Error) is selected.

Model Evaluation:

- Predict and Evaluate the model on validation dataset.

Hyperparameter Tuning:

- Tuning parameters to get the best score and best parameters combinations using Randomised Search Cross Validation.

Architecture Description

Model Building :

In [205]:
```
# Fitting Random Forest Regressor with best parameters as suggested after Hyperparameter tuning Random Forest Regressor.
```

In [195]:
```
rf_regressor = RandomForestRegressor(n_estimators = 700,
 min_samples_split = 10,
 min_samples_leaf = 5,
 max_features = 'auto',
 max_depth = 30)
```

In [196]:
```
rf_regressor.fit(X_train, Y_train)
```

Out[196]:
```
RandomForestRegressor(max_depth=30, min_samples_leaf=5, min_samples_split=10,
                      n_estimators=700)
```

In [197]:
```
y_pred_rf_htuned = rf_regressor.predict(X_val)
```

In [198]:
```
print("R Squared: " , r2_score(Y_val,y_pred_rf_htuned))
print("Mean Absolute Error: ", mean_absolute_error(Y_val,y_pred_rf_htuned))
print("Mean Squared Error: ", mean_squared_error(Y_val,y_pred_rf_htuned))
print("Root Mean Squared Error of RandomForestRegressor without CV: %.4g" % np.sqrt(mean_squared_error(Y_val,y_pred_rf_htuned)))
```

```
R Squared:  0.7158047804081784
Mean Absolute Error:  0.4333712437112187
Mean Squared Error:  0.30215227628523345
Root Mean Squared Error of RandomForestRegressor without CV: 0.5497
```

- Building the model with suggested parameters from Hyperparameter tuned model, testing and evaluating the model.
- Tuned Random Forest Regressor got highest accuracy and lowest error score. With 71.56% R Squared, and RMSE of 0.5499.

Architecture Description

Saving the model :

- Model is saved in pickle format as pkl.

Model Testing for Reuse:

- Predicting from the saved pickle file to validate if it’s working.

Flask Setup:

- Web application is created using Flask, which takes user inputs and passes it to the model to predict sales.

Push to GitHub:

- Project Directory pushed to Github.

Deployment : App link- Store Sales Prediction - INeuron

- The cloud environment was set up and the project was deployed from GitHub into the Heroku cloud platform.

Q & A

Q1) What's the source of data?

- The data for training and testing is provided at INeuron portal.

Q2) What was the type of data?

- Combination of Numerical and Categorical values.

Q3) What's the complete flow you followed in this Project?

- Explained in the Architecture Description.

Q4) After the File validation what do you do with incompatible files which didn't pass the validation?

- Files like these are moved to the Archieve Folder and a list of these files will be shared with the client and removed.

Q & A

Q5) How logs are managed?

- Using different logs as per the steps that we follow in validation and modeling like validation log, database log, preprocessing log, model training log, etc..

Q6) What techniques were you using for data pre-processing?

- Removing unwanted features.
- Visualizing relation of independent variables with each other and output variables.
- Log transformed features with outliers.
- Cleaning data and imputing null values.
- Converting categorical data into numeric values.
- Scaling down the data using Standard Scaler.

Q & A

Q7) How training was done or what models were used?

- Scaled and Encoded data was passed to various algorithms
- Data was trained on Linear Regression, Lasso, Ridge, Random forest Regressor.
- Models were fit on Tranining data, and predicted from Validation dataset.

Q 8) How was Prediction done?

- Passing the data to the best model which is saved in pickle format and get the prediction.

Q 9) Where is the model deployed?

- After saving the model, deployed it on Heroku platform. This model is a web application where user can enter inputs and this gets passed to the backend and user gets the prediction result.