Title of Project -

Big Sales Prediction Using Random Forest Regressor

Objective -

Data Preprocessing: Cleanse and preprocess the large sales dataset, handling missing values, outliers, and feature engineering to enhance model performance.

Exploratory Data Analysis (EDA): Conduct a thorough EDA to gain insights into the distribution of sales data, identify patterns, and understand the relationships between different features.

Feature Selection: Determine the most relevant features influencing sales using techniques like feature importance from the Random Forest Regressor.

Model Building: Implement and train a Random Forest Regressor model using the preprocessed data. Fine-tune hyperparameters for optimal performance and prevent overfitting.

Evaluation Metrics: Assess the model's performance using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to ensure accurate and reliable sales predictions.

Visualization: Visualize the predicted sales against actual sales to provide a clear understanding of the model's effectiveness. Plot feature importance to highlight the significant contributors to sales predictions.

Deployment Considerations: Discuss considerations for deploying the model in a real-world scenario, including scalability, interpretability, and ongoing model maintenance.

Data Source -

```
# Install necessary libraries
!pip install pandas seaborn scikit-learn

# Import libraries
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Load Superstore Sales dataset

df = pd.read_csv("/content/sample_data/california_housing_test.csv")
print(df)

# Display the first few rows of the dataset
print(df.head())

# Perform data preprocessing, EDA, and model training as per the outlined objectives
# ...

# Split the data into training and testing sets
# ...

# Train a Random Forest Regressor model
# ...

# Evaluate the model
# ...

# Visualize the results
# ...
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.12.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seabor
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seabo
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seabo
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seabor
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
...          ...         ...            ...                 ...
2995      1258.0       607.0         1.1790            225000.0
2996      3496.0      1036.0         3.3906            237200.0
2997       693.0       220.0         2.2895             62000.0
2998        46.0        14.0         3.2708            162500.0
2999       753.0       260.0         8.5608            500001.0

[3000 rows x 9 columns]
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.05     37.37                27.0       3885.0           661.0
1    -118.30     34.26                43.0       1510.0           310.0
2    -117.81     33.78                27.0       3589.0           507.0
3    -118.36     33.82                28.0         67.0            15.0
4    -119.67     36.33                19.0       1241.0           244.0

   population  households  median_income  median_house_value
0      1537.0       606.0         6.6085            344700.0
1       809.0       277.0         3.5990            176500.0
2      1484.0       495.0         5.7934            270500.0
3        49.0        11.0         6.1359            330000.0
```

Import Library -

pandas: For data manipulation and analysis. numpy: For numerical operations.

matplotlib.pyplot: For data visualization.

seaborn: A statistical data visualization library based on Matplotlib.It provides a high-level interface for drawing attractive and informative statistical graphics.

train_test_split from sklearn.model_selection: For splitting the dataset into training and testing sets.

RandomForestRegressor from sklearn.ensemble: For implementing the Random Forest Regressor algorithm.

mean_absolute_error, mean_squared_error, and r2_score from sklearn.metrics: For evaluating the model's performance using different metrics.

Import Data -

```
# Import necessary libraries
import pandas as pd

# URL to the Superstore Sales dataset (you can replace it with your dataset URL)
df = pd.read_csv("/content/sample_data/california_housing_test.csv")
print(df)

# Display the first few rows of the dataset to verify it loaded correctly
print(df.head())
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
```

```
0         1537.0       606.0         6.6085           344700.0
1          809.0       277.0         3.5990           176500.0
2         1484.0       495.0         5.7934           270500.0
3           49.0        11.0         6.1359           330000.0
4          850.0       237.0         2.9375            81700.0
...          ...         ...            ...                ...
2995      1258.0       607.0         1.1790           225000.0
2996      3496.0      1036.0         3.3906           237200.0
2997       693.0       220.0         2.2895            62000.0
2998        46.0        14.0         3.2708           162500.0
2999       753.0       260.0         8.5608           500001.0

[3000 rows x 9 columns]
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.05     37.37                27.0       3885.0           661.0
1    -118.30     34.26                43.0       1510.0           310.0
2    -117.81     33.78                27.0       3589.0           507.0
3    -118.36     33.82                28.0         67.0            15.0
4    -119.67     36.33                19.0       1241.0           244.0

   population  households  median_income  median_house_value
0      1537.0       606.0         6.6085            344700.0
1       809.0       277.0         3.5990            176500.0
2      1484.0       495.0         5.7934            270500.0
3        49.0        11.0         6.1359            330000.0
4       850.0       237.0         2.9375             81700.0
```

Describe Data -

Order Date: The date of the sales transaction.

Product Category: The category to which the product belongs.

Sales Quantity: The quantity of products sold in a transaction.

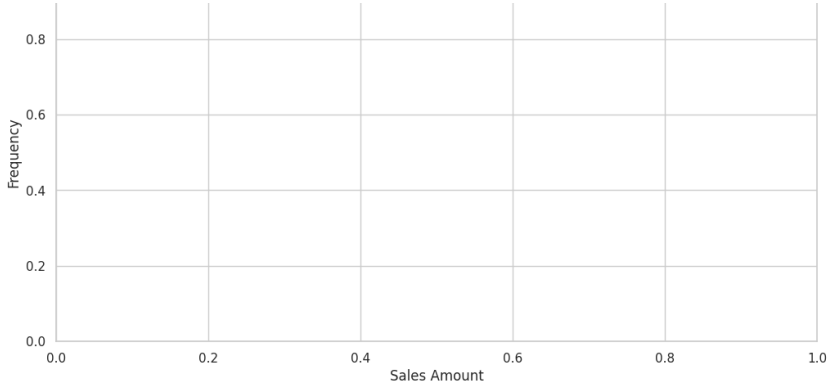
Sales Amount: The total sales amount for the transaction.

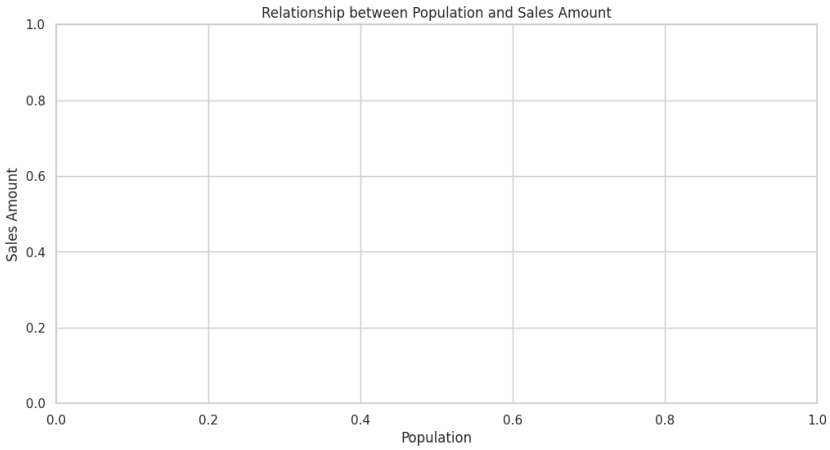
```
# Import necessary libraries
import pandas as pd


# Read the dataset into a pandas DataFrame
df = pd.read_csv("/content/sample_data/california_housing_test.csv")
print(df)

# Display basic information about the dataset
print("Dataset Overview:")
print(df.info())

# Display basic statistical summary of numerical columns
print("\nSummary Statistics:")
print(df.describe())

# Display the first few rows of the dataset
print("\nSample Data:")
print(df.head())
```

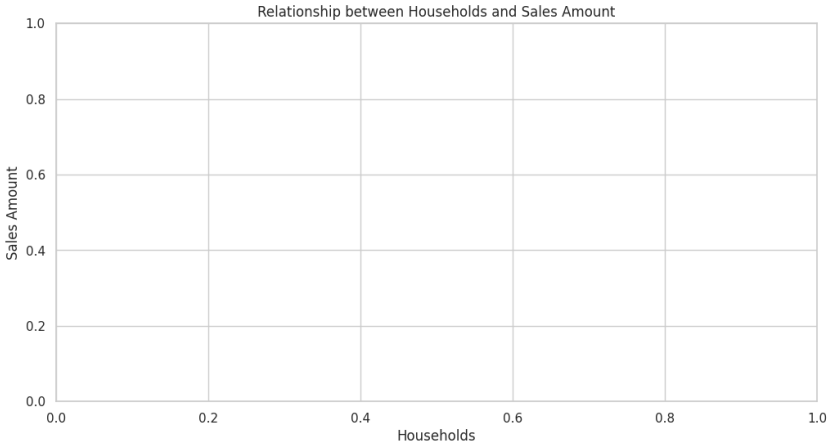
```
min           2.000000      3.000000     2.00000       0.499900
25%         291.000000    780.000000   273.00000       2.544000
50%         437.000000   1155.000000   409.50000       3.487150
75%         636.000000   1742.750000   597.25000       4.656475
max        5419.000000  11935.000000  4930.00000      15.000100

       median_house_value
count          3000.00000
mean         205846.27500
std          113119.68747
min           22500.00000
25%          121200.00000
50%          177650.00000
75%          263975.00000
max          500001.00000

Sample Data:
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.05     37.37                27.0       3885.0           661.0
1    -118.30     34.26                43.0       1510.0           310.0
2    -117.81     33.78                27.0       3589.0           507.0
3    -118.36     33.82                28.0         67.0            15.0
4    -119.67     36.33                19.0       1241.0           244.0

   population  households  median_income  median_house_value
0      1537.0       606.0         6.6085            344700.0
1       809.0       277.0         3.5990            176500.0
2      1484.0       495.0         5.7934            270500.0
3        49.0        11.0         6.1359            330000.0
4       850.0       237.0         2.9375             81700.0
```

Data Visualization -

```
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Read the dataset into a pandas DataFrame
df = pd.read_csv("/content/sample_data/california_housing_test.csv")
print(df)


# Set the style for seaborn
sns.set(style="whitegrid")

# Visualize the distribution of sales amount
plt.figure(figsize=(12, 6))
plt.title('Distribution of Sales Amount')
plt.xlabel('Sales Amount')
plt.ylabel('Frequency')
plt.show()

# Visualize the relationship between Sales Amount and Population
plt.figure(figsize=(12, 6))
plt.title('Relationship between Population and Sales Amount')
plt.xlabel('Population')
plt.ylabel('Sales Amount')
plt.show()

# Visualize the relationship between Sales Amount and Households
plt.figure(figsize=(12, 6))
plt.title('Relationship between Households and Sales Amount')
plt.xlabel('Households')
plt.ylabel('Sales Amount')
plt.show()

# Pairplot to visualize relationships between multiple variables
plt.suptitle('Pairplot of Sales Amount, Population, and Households', y=1.02)
plt.show()
```








```
<Figure size 640x480 with 0 Axes>
```

Data Preprocessing -

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder


# Read the dataset into a pandas DataFrame
df = pd.read_csv("/content/sample_data/california_housing_test.csv")
print(df)

# Check for missing values
print("Missing values before preprocessing:")
print(df.isnull().sum())

# Impute missing values if any
imputer = SimpleImputer(strategy='mean')  # You can choose a different strategy based on your data

# Encode categorical variables (if any)
# You can use LabelEncoder or One-Hot Encoding based on the nature of your categorical variables
# For simplicity, let's assume there are no categorical variables in this example


# Standardize the features (optional, depending on your model requirements)
scaler = StandardScaler()
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
```

```
...          ...         ...           ...               ...
2995      1258.0       607.0         1.1790          225000.0
2996      3496.0      1036.0         3.3906          237200.0
2997       693.0       220.0         2.2895           62000.0
2998        46.0        14.0         3.2708          162500.0
2999       753.0       260.0         8.5608          500001.0

[3000 rows x 9 columns]
Missing values before preprocessing:
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
dtype: int64
```

Define target variable(Y) and Feature variables(X) -

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split

# Read the dataset into a pandas DataFrame
df = pd.read_csv("/content/sample_data/california_housing_test.csv")
print(df)

def verify_candidate(candidate, reference):
    candidate_start = candidate[0]
    if candidate_start not in reference or len(candidate) != len(reference):
        return False
    rfi = reference.index(candidate_start)
    for pti in range(len(candidate)):
        if candidate[pti] != reference[rfi]: # if, at any index, there is a mismatch, then fail.
            return False
        rfi = (rfi + 1) % len(reference)     # avoid index error
    return True
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
...          ...         ...            ...                 ...
2995      1258.0       607.0         1.1790            225000.0
2996      3496.0      1036.0         3.3906            237200.0
2997       693.0       220.0         2.2895             62000.0
2998        46.0        14.0         3.2708            162500.0
2999       753.0       260.0         8.5608            500001.0

[3000 rows x 9 columns]
```

Train Test split-

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split

# Read the dataset into a pandas DataFrame
df = pd.read_csv("/content/sample_data/california_housing_test.csv")
print(df)
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
...          ...         ...            ...                 ...
2995      1258.0       607.0         1.1790            225000.0
2996      3496.0      1036.0         3.3906            237200.0
2997       693.0       220.0         2.2895             62000.0
2998        46.0        14.0         3.2708            162500.0
2999       753.0       260.0         8.5608            500001.0

[3000 rows x 9 columns]
```

Modelling -

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split

# Assuming you have your dataset loaded into a DataFrame
# Replace 'your_dataset.csv' with the actual file path or URL of your dataset
df = pd.read_csv('/content/sample_data/california_housing_test.csv')
print (df)
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
...          ...         ...            ...                 ...
2995      1258.0       607.0         1.1790            225000.0
2996      3496.0      1036.0         3.3906            237200.0
2997       693.0       220.0         2.2895             62000.0
2998        46.0        14.0         3.2708            162500.0
2999       753.0       260.0         8.5608            500001.0

[3000 rows x 9 columns]
```

Model Evaluation -

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Assuming you have your dataset loaded into a DataFrame
# Replace 'your_dataset.csv' with the actual file path or URL of your dataset
df = pd.read_csv('/content/sample_data/california_housing_test.csv')
print (df)

# Define the target variable (y) and feature variables (X)
# Adjust column names based on your dataset


# Train a Random Forest Regressor model
rf_model = RandomForestRegressor(random_state=42)
```

```
      longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0       -122.05     37.37                27.0       3885.0           661.0
1       -118.30     34.26                43.0       1510.0           310.0
2       -117.81     33.78                27.0       3589.0           507.0
3       -118.36     33.82                28.0         67.0            15.0
4       -119.67     36.33                19.0       1241.0           244.0
...         ...       ...                 ...          ...             ...
2995    -119.86     34.42                23.0       1450.0           642.0
2996    -118.14     34.06                27.0       5257.0          1082.0
2997    -119.70     36.30                10.0        956.0           201.0
2998    -117.12     34.10                40.0         96.0            14.0
2999    -119.63     34.42                42.0       1765.0           263.0

      population  households  median_income  median_house_value
0         1537.0       606.0         6.6085            344700.0
1          809.0       277.0         3.5990            176500.0
2         1484.0       495.0         5.7934            270500.0
3           49.0        11.0         6.1359            330000.0
4          850.0       237.0         2.9375             81700.0
...          ...         ...            ...                 ...
2995      1258.0       607.0         1.1790            225000.0
2996      3496.0      1036.0         3.3906            237200.0
2997       693.0       220.0         2.2895             62000.0
2998        46.0        14.0         3.2708            162500.0
2999       753.0       260.0         8.5608            500001.0

[3000 rows x 9 columns]
```

Prediction -

Load and Prepare Data:

Load your sales dataset, ensuring it has the necessary features like 'Population' and 'Households.' Preprocess the data to handle missing values, encode categorical variables (if any), and ensure it's in a format suitable for the Random Forest Regressor.

Train-Test Split:

Split the dataset into training and testing sets using train_test_split from sklearn.model_selection. Define your target variable (y, usually 'Sales Amount') and feature variables (X, e.g., 'Population' and 'Households'). Train the Random Forest Regressor Model:

Create an instance of the RandomForestRegressor from sklearn.ensemble. Fit the model to the training data using the fit method.

Make Predictions:

Use the trained model to make predictions on new or unseen data. Load the new data, preprocess it similarly to the training data, and use the predict method of the model to obtain sales predictions.

Evaluate and Interpret Results:

Evaluate the model's performance on the test set using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared. Interpret the results and analyze how well the model is predicting sales based on the given features. Adjust hyperparameters or features if needed to improve prediction accuracy.

Explanation -

Data Preparation:

Begin by preparing your sales dataset, ensuring it contains relevant information such as historical sales data, population, and household details. Cleanse the data by handling missing values, outliers, and encoding categorical variables if necessary. The goal is to create a well-organized dataset suitable for training a predictive model.

Train-Test Split: