DSC630_Week8_Assignment_Guruprasad- Part1-Using R

Guruprasad Velikadu Krishnamoorthy

2024-01-31

```r
# Calling the Libraries used
library(readxl)
library(dplyr)
library(lubridate)
library(readr)
library(ggplot2)
library(ggthemes)
library(tidyr)
library(DT)
library(scales)
library(stringr)
library(knitr)
library(FactoMineR)
library(ggpubr)
library(kableExtra)
library(magrittr)
library(ggfortify)
library(visdat)
library(janitor)
library(Metrics)
```

```r
# Reading the source file and creating a Dataframe
retail_df_orig <- read.csv("us_retail_sales.csv", stringsAsFactors = FALSE)
# Printing dimensions of the Dataframe
dim(retail_df_orig)
```

```
## [1] 30 13
```

1. Plot the data with proper labeling and make some observations on the graph.

```r
# Formatting the column names to camel case for display purpose
retail_df_orig <- retail_df_orig %>%
    clean_names(case = "big_camel")
# Printing the top few rows from the Dataframe
kbl(head(retail_df_orig[1:6, ]), caption = "Retail Sales Data", booktabs = T) %>%
    kable_styling(latex_options = c("striped", "hold_position"))
```

Table 1: Retail Sales Data

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1992	146925	147223	146805	148032	149010	149800	150761	151067	152588	153521	153583	155614
1993	157555	156266	154752	158979	160605	160127	162816	162506	163258	164685	166594	168161
1994	167518	169649	172766	173106	172329	174241	174781	177295	178787	180561	180703	181524
1995	182413	179488	181013	181686	183536	186081	185431	186806	187366	186565	189055	190774
1996	189135	192266	194029	194744	196205	196136	196187	196218	198859	200509	200174	201284
1997	202371	204286	204990	203399	201699	204675	207014	207635	208326	208078	208936	209363

```r
# For formatting reasons changing the format of the dataframe from Columns to
# Rows using the gather function
retail_df <- retail_df_orig %>%
    gather(key = Month, value = Sales, -Year) %>%
    arrange(Year)
# Printing data after reshaping
kbl(head(retail_df[1:6, ]), caption = "Retail Sales Data (after Reshaping)", booktabs = T) %>%
    kable_styling(latex_options = c("striped", "hold_position"))
```

Table 2: Retail Sales Data (after Reshaping)

Year	Month	Sales
1992	Jan	146925
1992	Feb	147223
1992	Mar	146805
1992	Apr	148032
1992	May	149010
1992	Jun	149800

```r
# Examining the structure of the dataframe
str(retail_df)
```

```
## 'data.frame':    360 obs. of  3 variables:
##  $ Year : int  1992 1992 1992 1992 1992 1992 1992 1992 1992 1992 ...
##  $ Month: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ Sales: int  146925 147223 146805 148032 149010 149800 150761 151067 152588 153521 ...
```

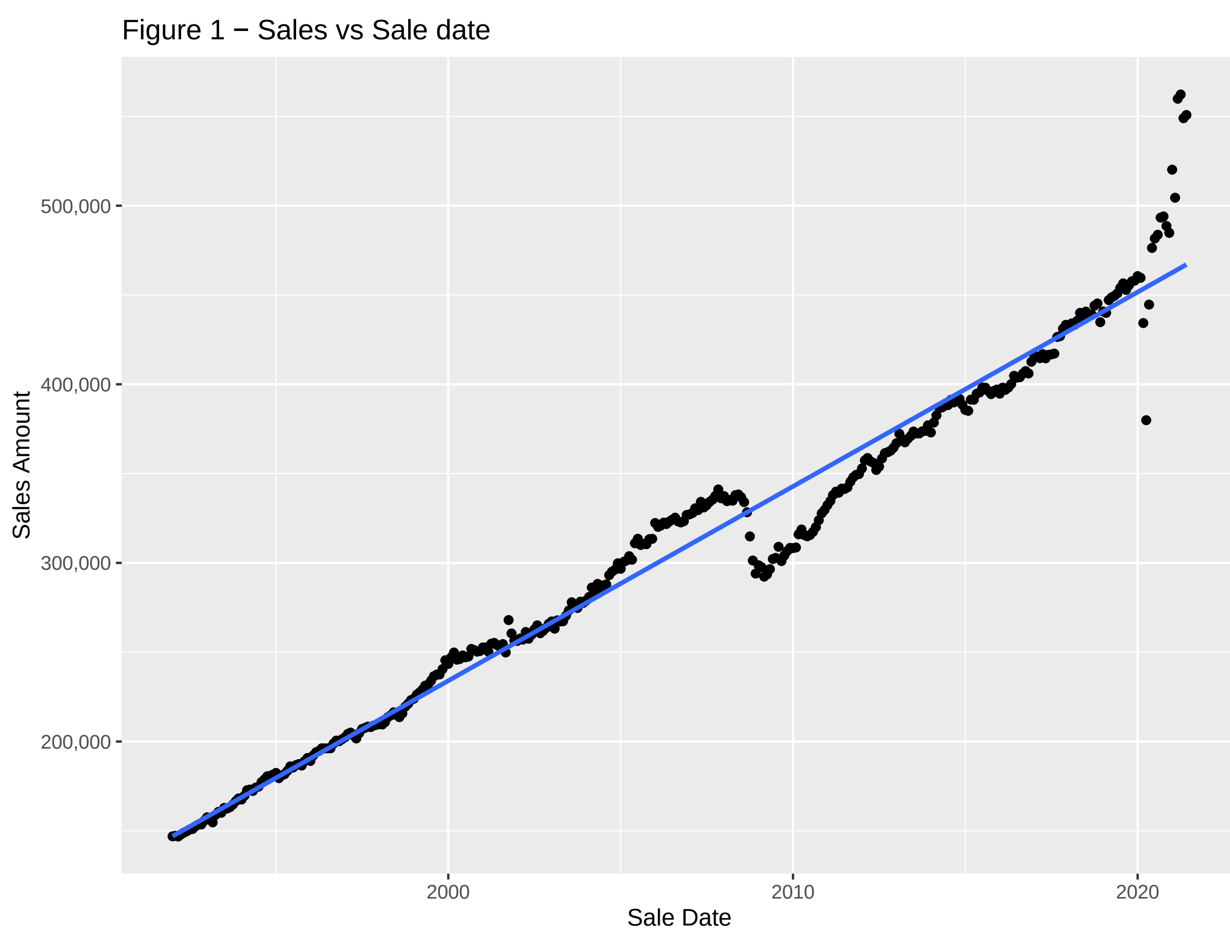
```r
# Adding a new column Sale date with the Year and Month fields
retail_df$Sale_Date <- paste(as.character(retail_df$Year), retail_df$Month, "01")
# Converting the Sale date to Date format
retail_df$Sale_Date <- as.Date(retail_df$Sale_Date, format = "%Y %b %d")
# Converting the Year and Month as Dates for display purposes
retail_df$Year <- year(retail_df$Sale_Date)
retail_df$Month <- month(retail_df$Sale_Date, label = TRUE)
# Dropping nulls from the dataframe
retail_df <- drop_na(retail_df)
# Printing data after reshaping
```

```
kbl(head(retail_df[1:10, ]), caption = "Retail Sales Data (after Formatting)", booktabs = T) %>%
    kable_styling(latex_options = c("striped", "hold_position"))
```

Table 3: Retail Sales Data (after Formatting)

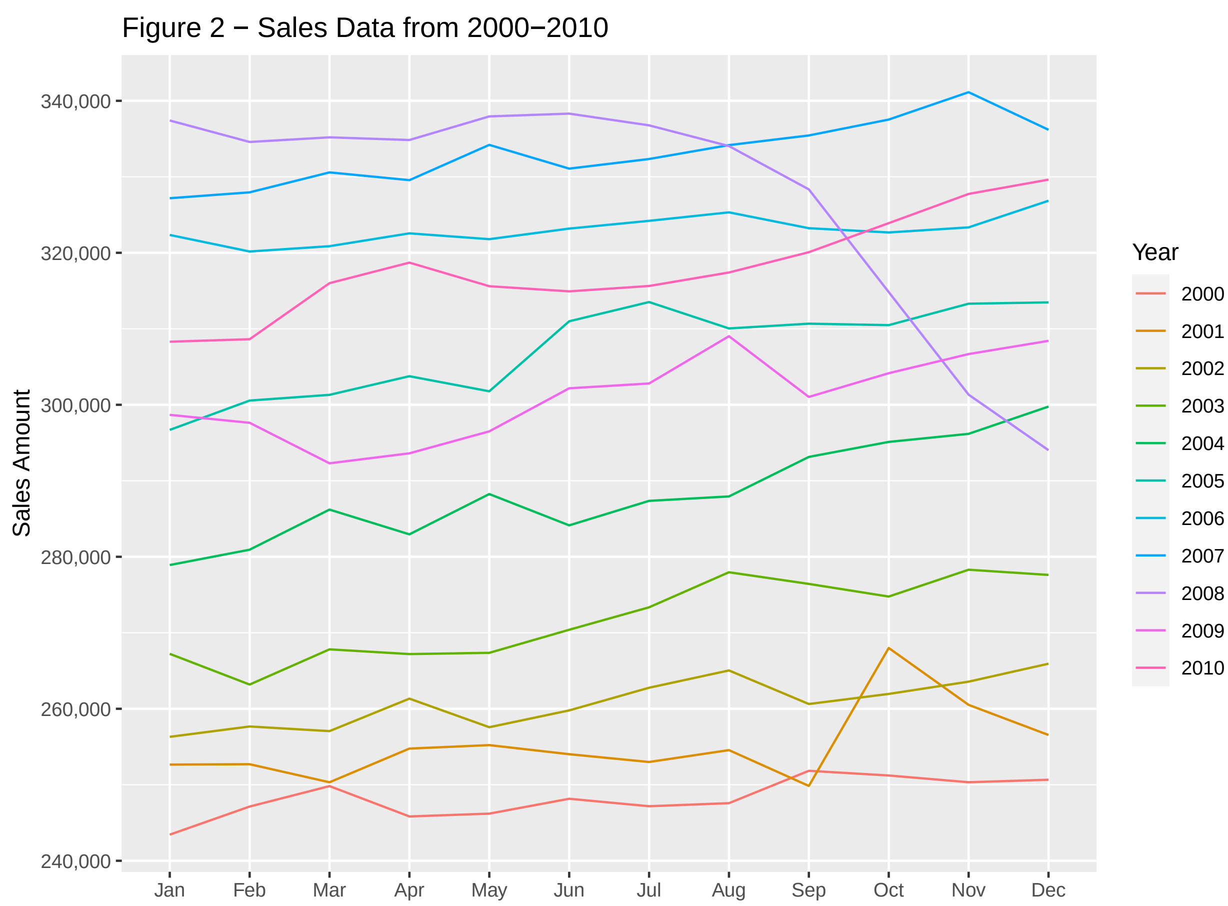
Year	Month	Sales	Sale_Date
1992	Jan	146925	1992-01-01
1992	Feb	147223	1992-02-01
1992	Mar	146805	1992-03-01
1992	Apr	148032	1992-04-01
1992	May	149010	1992-05-01
1992	Jun	149800	1992-06-01

```
# Plotting a scatter plot of Sales vs Sale date with a linear regression line
ggplot(retail_df, aes(Sale_Date, Sales)) + geom_point() + geom_smooth(method = "lm",
    se = FALSE) + scale_y_continuous(labels = comma) + labs(title = "Figure 1 - Sales vs Sale date",
    x = "Sale Date", y = "Sales Amount")
```



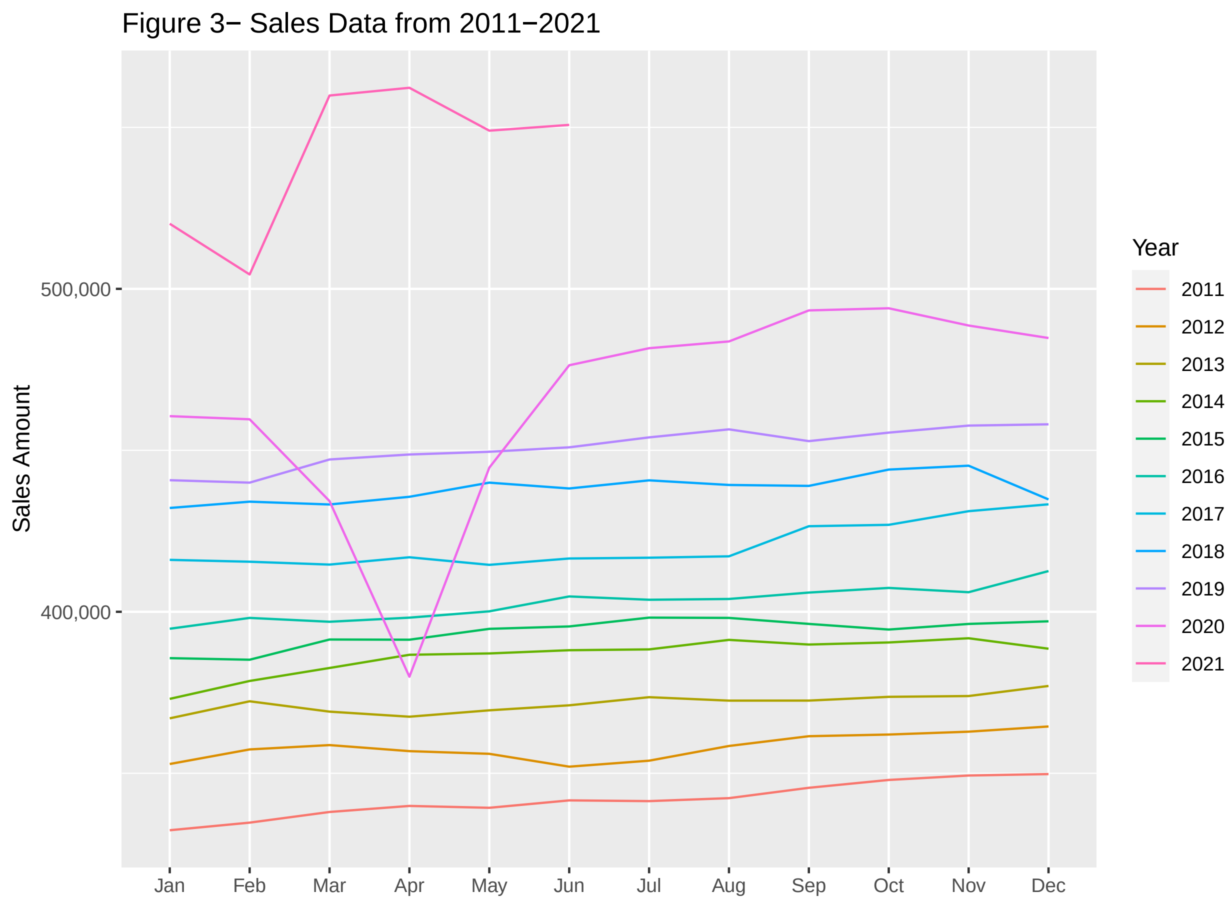
Observations: From the plot it appears the sales increased over time. However there is a drop in 2009 which may due to the global recession. Also there is a drop in 2020 due to Covid Pandemic.

```r
# Creating a new dataframe that contains Sales data from 2000 to 2010
sales_from_2000_2010 <- retail_df %>%
    filter(Year >= 2000 & Year <= 2010)
# Creating a Line plot of Sales amount vs Months with data from 2000 to 2010
# Data is grouped by Year
ggplot(sales_from_2000_2010, aes(x = Month, y = Sales)) + geom_line(aes(color = factor(Year),
    group = Year)) + scale_y_continuous(labels = comma) + scale_color_discrete(name = "Year") +
    labs(title = "Figure 2 - Sales Data from 2000-2010", x = "", y = "Sales Amount")
```



Observations: From the 2000 to 2010 plot, it appears the sales in first half of 2008 was higher and the sales dropped significantly during the second half due to Global recession. Also the sale numbers in 2007 was higer than 2009 and 2010 due to the recession.

```r
# Creating a new dataframe that contains Sales data from 2011-2021
sales_after_2010 <- retail_df %>%
    filter(Year > 2010)
# Creating a Line plot of Sales amount vs Months with data from 2011-2021. Data
# is grouped by Year
ggplot(sales_after_2010, aes(x = Month, y = Sales)) + geom_line(aes(color = factor(Year),
    group = Year)) + scale_y_continuous(labels = comma) + scale_color_discrete(name = "Year") +
    labs(title = "Figure 3- Sales Data from 2011-2021", x = "", y = "Sales Amount")
```



Observations: From the 2011 to 2021 plot,the sales were pretty flat for most of the years except for 2020 and 2021. There is huge dip in the sales between 2020 March to June which coincides with the period when Covid hit the US. Also the sale recovered rapidly and the numbers were higher in the first half of 2021 compared to rest of the years. As the dataset only contained data until June 2021, the 2021 line only extends through half of the year.

2. Split this data into a training and test set. Use the last year of data (July 2020 – June 2021) of data as your test set and the rest as your training set.

```
# Creating Training data with Sale date before 2020 June and only selecting the
# sale date and Sales columns
train_data <- retail_df %>%
    select(Sale_Date, Sales) %>%
    filter(Sale_Date <= "2020-06-01")
# Creating Training data with Sale date after 2020 June and only selecting the
# sale date and Sales columns
test_data <- retail_df %>%
    select(Sale_Date, Sales) %>%
    filter(Sale_Date >= "2020-07-01")
```

```
# Printing rows from Training dataset
kbl(head(train_data[1:6, ]), caption = "Training Dataset", booktabs = T) %>%
    kable_styling(latex_options = c("striped", "hold_position"))
```

Table 4: Training Dataset

Sale_Date	Sales
1992-01-01	146925
1992-02-01	147223
1992-03-01	146805
1992-04-01	148032
1992-05-01	149010
1992-06-01	149800

```r
# Printing rows from Test dataset
kbl(head(test_data[1:6, ]), caption = "Test Dataset", booktabs = T) %>%
    kable_styling(latex_options = c("striped", "hold_position"))
```

Table 5: Test Dataset

Sale_Date	Sales
2020-07-01	481627
2020-08-01	483716
2020-09-01	493327
2020-10-01	493991
2020-11-01	488652
2020-12-01	484782

3. Use the training set to build a predictive model for the monthly retail sales.

Building a Linear Regression model using R

```r
# Creating a Linear Regression model from Sale date in Training data to predict
# the Sales
reg_model <- lm(Sales ~ Sale_Date, data = train_data)
```

```r
# Printing the summary of Regression model
summary(reg_model)
```

```
##
## Call:
## lm(formula = Sales ~ Sale_Date, data = train_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -66948  -4802   -642   7813  27753
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -79588.259   3105.098  -25.63   <2e-16 ***
## Sale_Date       28.683      0.229  125.27   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12720 on 340 degrees of freedom
## Multiple R-squared:  0.9788, Adjusted R-squared:  0.9787
## F-statistic: 1.569e+04 on 1 and 340 DF,  p-value: < 2.2e-16
```

Observations: The R-squared value is very high and is close to 0.98, so there is a chance that the model may be overfitted. Also the P values are in acceptable range of lesser than 0.05.

4. Use the model to predict the monthly retail sales on the last year of data.

```r
# Creating a tibble of Sale date from test data to be used to predict using the
# model
explanatory_data <- tibble(Sale_Date = test_data$Sale_Date)
# Adding a new column called Predicted sales in test data with the prediction
# results
test_data <- test_data %>%
    mutate(predicted_sales = predict(reg_model, explanatory_data))
test_data$predicted_sales <- as.integer(test_data$predicted_sales)
```

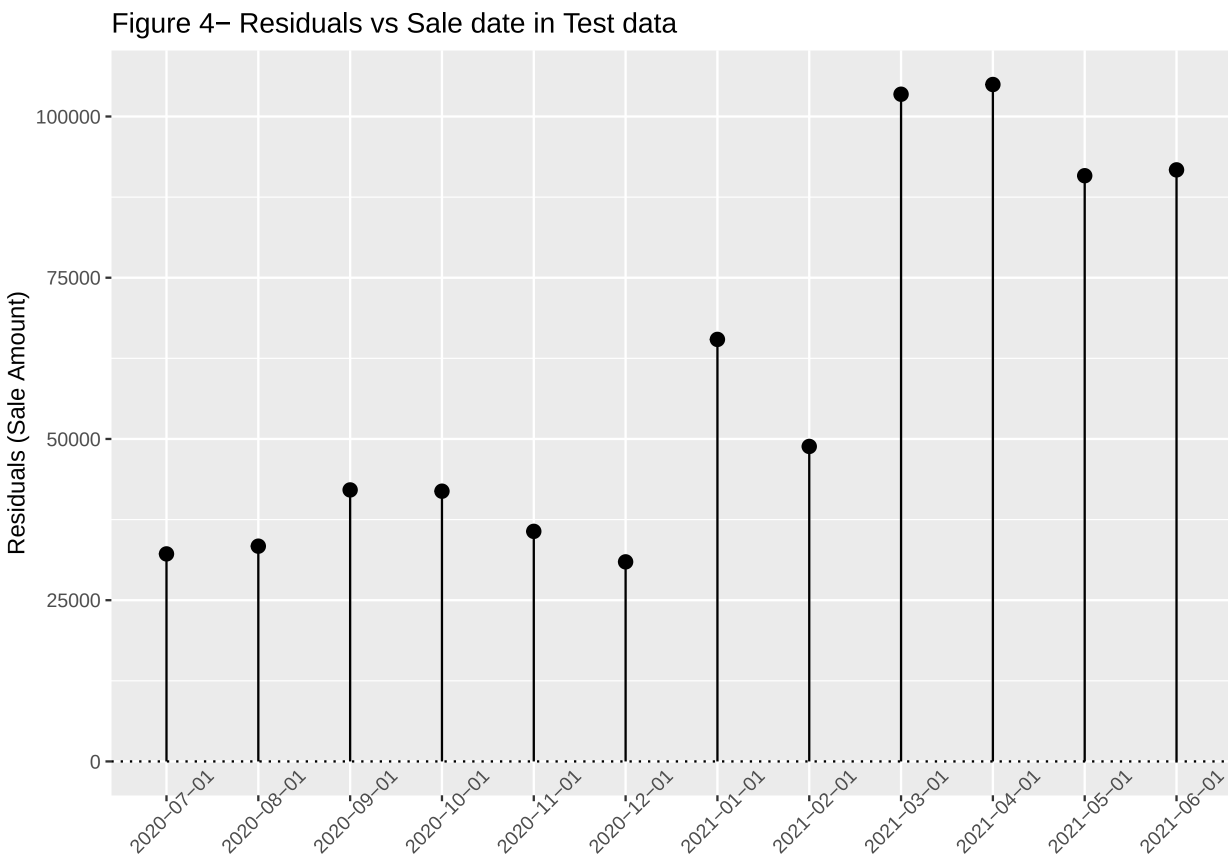
```r
# Printing rows from Test dataset with Prediction results
kbl(head(test_data[1:6, ]), caption = "Test Dataset (with Predicted sales)", booktabs = T) %>%
    kable_styling(latex_options = c("striped", "hold_position"))
```

Table 6: Test Dataset (with Predicted sales)

Sale_Date	Sales	predicted_sales
2020-07-01	481627	449450
2020-08-01	483716	450339
2020-09-01	493327	451228
2020-10-01	493991	452089
2020-11-01	488652	452978
2020-12-01	484782	453838

```r
# Calculating residuals of the Linear regression model
test_data$residuals <- test_data$Sales - test_data$predicted_sales

# Building a Residual plot
ggplot(test_data, aes(x = as.factor(Sale_Date), y = residuals)) + geom_pointrange(aes(ymin = 0,
    ymax = residuals)) + geom_hline(yintercept = 0, linetype = 3) + labs(title = "Figure 4- Residuals vs
    x = "", y = "Residuals (Sale Amount)") + theme(axis.text.x = element_text(angle = 45))
```



5. Report the RMSE of the model predictions on the test set.

```
# Calculating the Root Mean square error
rmse(test_data$Sales, test_data$predicted_sales)
```

```
## [1] 66429.51
```

Observations The results of RMSE indicates that there is an error of upto $64500 in the Sales predictions by the model. As the Linear regression may not be the best choice for preidting Time series data, a different method will be used using Python.

Part2 - Model building Using Python

As the Simple Linear Regression may not be a great choice for predicting Time series data, Holt Winters method is used using Python to build the model.

In [1]:
```
# Importing the required libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing as HWES
```

In [2]:
```
# Creating a Dataframe After reading the retails_df file imported from the manipulation done in R
retail_df = pd.read_csv('retail_df.csv', header=0, infer_datetime_format=True, parse_dates=[1]
                        , index_col=[1]
                       )
# Setting the index frequency as Months
retail_df.index.freq = 'MS'
```

In [3]:
```
# Printing the top few rows in the dataframe
retail_df.head()
```

Out[3]:

	Sales
Sale_Date	
1992-01-01	146925
1992-02-01	147223
1992-03-01	146805
1992-04-01	148032
1992-05-01	149010

In [4]:
```
# Setting the index frequency as Months
retail_df.index.freq = 'MS'
```

In [5]:
```
# Splitting the dataset into Train and test sets
retail_df_train=retail_df.loc[:"2020-06-01"]
# Data from July 2020 until June 2021 are used as Test datasets
retail_df_test = retail_df.loc["2020-07-01":]
```

In [6]:
```
# Printing the data in the Test dataset
retail_df_test
```

Out[6]:

	Sales
Sale_Date	
2020-07-01	481627
2020-08-01	483716
2020-09-01	493327
2020-10-01	493991
2020-11-01	488652
2020-12-01	484782
2021-01-01	520162
2021-02-01	504458
2021-03-01	559871
2021-04-01	562269
2021-05-01	548987
2021-06-01	550782

In [7]:
```
# Building a model object using Holt winters with period set to 12
holtwinter_model = HWES(retail_df_train, seasonal_periods=12, trend='add', seasonal='mul')
# Fitting the model
```

```
fitted_object = holtwinter_model.fit()
```

```
C:\Users\Gurup\anaconda3_2021.11\lib\site-packages\statsmodels\tsa\holtwinters\model.py:915: ConvergenceWarning: Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```

In [8]:
```
# Printing the summary of the Model
print(fitted_object.summary())
```

```
                       ExponentialSmoothing Model Results
================================================================================
Dep. Variable:                    Sales   No. Observations:                  342
Model:             ExponentialSmoothing   SSE                    11844033536.443
Optimized:                         True   AIC                           5969.216
Trend:                         Additive   BIC                           6030.573
Seasonal:                Multiplicative   AICC                          5971.333
Seasonal Periods:                    12   Date:                 Sat, 03 Feb 2024
Box-Cox:                          False   Time:                         19:18:26
Box-Cox Coeff.:                    None
=================================================================================
                          coeff                 code              optimized
---------------------------------------------------------------------------------
smoothing_level               0.9242857                alpha                True
smoothing_trend                  0.0001                 beta                True
smoothing_seasonal            0.0432653                gamma                True
initial_level                 1.4979e+05                 l.0                True
initial_trend                 887.75783                  b.0                True
initial_seasons.0             0.9999993                  s.0                True
initial_seasons.1             0.9956907                  s.1                True
initial_seasons.2             0.9967751                  s.2                True
initial_seasons.3             1.0002459                  s.3                True
initial_seasons.4             1.0005226                  s.4                True
initial_seasons.5             1.0004347                  s.5                True
initial_seasons.6             0.9999762                  s.6                True
initial_seasons.7             1.0000104                  s.7                True
initial_seasons.8             1.0008482                  s.8                True
initial_seasons.9             1.0002168                  s.9                True
initial_seasons.10            1.0009653                 s.10                True
initial_seasons.11            1.0043148                 s.11                True
---------------------------------------------------------------------------------
```

In [9]:
```
# Predicting the sales using forecast function and stepps are set to 12 to predict 12 month sales
```

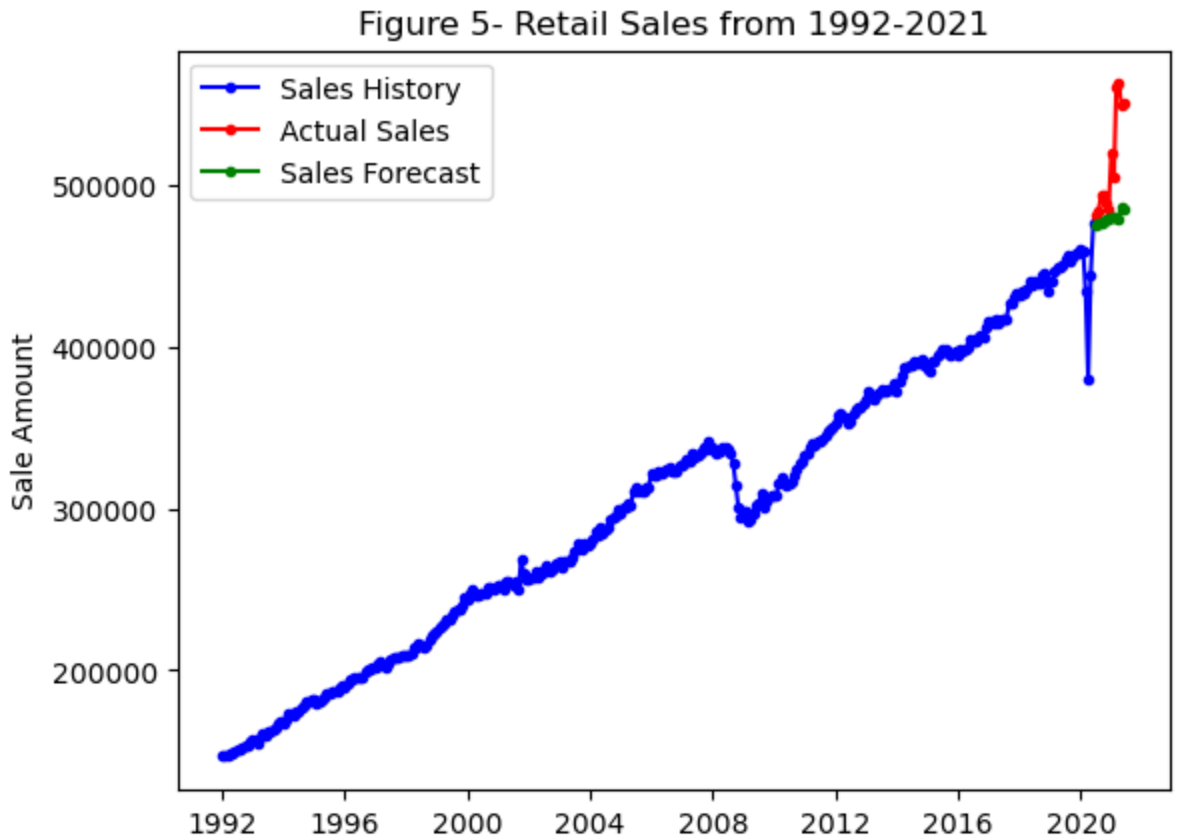
```
sales_forecast = fitted_object.forecast(steps=12)
sales_forecast.head()
```

Out[9]:

```
2020-07-01    474871.718730
2020-08-01    475695.133314
2020-09-01    475856.047180
2020-10-01    477395.170172
2020-11-01    478259.621682
Freq: MS, dtype: float64
```

In [10]:

```
#plot the training data, the test data and the forecast on the same plot
# Assigning the Title to the plot
plt.title('Figure 5- Retail Sales from 1992-2021')
plt.ylabel('Sale Amount')
# Creating an object for the Past sales using the Training Dataset
past, = plt.plot(retail_df_train.index, retail_df_train, 'b.-', label='Sales History')
# Creating an object for the Future sales using the Test Dataset
future, = plt.plot(retail_df_test.index, retail_df_test, 'r.-', label='Actual Sales')
# Creating an object for the Predicted sales using the Prediction numbers
predicted_future, = plt.plot(retail_df_test.index, sales_forecast, 'g.-', label='Sales Forecast')
plt.legend(handles=[past, future, predicted_future])
plt.show()
```



The Plot indicates that the Actual sales were higher than the predicted sales . The Root mean square error is calculated in the next step that shows the expected error in the predictions

In [11]:
```python
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
def create_regression_metrics(y_obj,predicted_tgt):
    """
    This function takes the Predicted values as input and returns the results RMSE
    """
    # using mean_squared_error function to calculate the MSE
    mse=mean_squared_error(y_obj,predicted_tgt)
    print(f"MSE: {mse}")
    # using mean_squared_error function to calculate the RMSE by taking square root of MSE
    rmse= np.sqrt(mean_squared_error(y_obj,predicted_tgt))
    print(f"RMSE: {rmse}")
    return (rmse)
```

In [12]:
```python
# Calculating the RMSE using the Holt winters model
create_regression_metrics(retail_df_test,sales_forecast)
```

```
MSE: 2056170184.524826
RMSE: 45345.01278558455
```

Out[12]:
```
45345.01278558455
```

The RMSE results indicates that we can expect an error upto approximately 45000 in the Sales predictions which is lower than the RMSE using simple linear regression in R.