In [1]:
```
# Importing Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet
from sklearn.metrics import root_mean_squared_error
```

In [2]:
```
df=pd.read_excel(r"C:\Users\Admin\Downloads\archive (1)\Sample - Superstore.xlsx
```

In [3]:
```
df
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Count
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	Unit Stat
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	Unit Stat
2	3	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	Unit Stat
3	4	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	Unit Stat
4	5	US-2015-108966	2015-10-11	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	Unit Stat
...	...	...	...	...	...	...	...	...	
9989	9990	CA-2014-110422	2014-01-21	2014-01-23	Second Class	TB-21400	Tom Boeckenhauer	Consumer	Unit Stat
9990	9991	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	Unit Stat
9991	9992	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	Unit Stat
9992	9993	CA-2017-121258	2017-02-26	2017-03-03	Standard Class	DB-13060	Dave Brooks	Consumer	Unit Stat
9993	9994	CA-2017-119914	2017-05-04	2017-05-09	Second Class	CC-12220	Chris Cortes	Consumer	Unit Stat

9994 rows × 21 columns

In [4]:
```
df.info() # Checking shape,datatype of columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Row ID         9994 non-null   int64         
 1   Order ID       9994 non-null   object        
 2   Order Date     9994 non-null   datetime64[ns]
 3   Ship Date      9994 non-null   datetime64[ns]
 4   Ship Mode      9994 non-null   object        
 5   Customer ID    9994 non-null   object        
 6   Customer Name  9994 non-null   object        
 7   Segment        9994 non-null   object        
 8   Country        9994 non-null   object        
 9   City           9994 non-null   object        
 10  State          9994 non-null   object        
 11  Postal Code    9994 non-null   int64         
 12  Region         9994 non-null   object        
 13  Product ID     9994 non-null   object        
 14  Category       9994 non-null   object        
 15  Sub-Category   9994 non-null   object        
 16  Product Name   9994 non-null   object        
 17  Sales          9994 non-null   float64       
 18  Quantity       9994 non-null   int64         
 19  Discount       9994 non-null   float64       
 20  Profit         9994 non-null   float64       
dtypes: datetime64[ns](2), float64(3), int64(3), object(13)
memory usage: 1.6+ MB
```

In [5]:
```
df.isnull().sum() # Checking for missing values
```

Out[5]:
```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

In [6]:
```
df.duplicated().sum() # Checking for duplicates
```

Out[6]:
```
0
```

In [7]:
```
df=df.drop_duplicates() # Dropping duplicates
```

In [8]:
```
df['Order Date']=pd.to_datetime(df['Order Date']) #Converting Order date to Date
df['Ship Date']=pd.to_datetime(df['Ship Date']) # Converting Ship Date to Dateti
```

In [9]:
```
# Extracting Year,Month,Week,Month_Year from Order Date
df['Year']=df['Order Date'].dt.year
df['Month']=df['Order Date'].dt.month
df['Week']=df['Order Date'].dt.isocalendar().week
df['Month_Year'] = df['Order Date'].dt.to_period('M')
```

In [10]:
```
# Aggregating sales by month yearwise
monthly_sales = df.groupby('Month_Year').agg({'Sales': 'sum'}).reset_index()
monthly_sales['Month_Year'] = monthly_sales['Month_Year'].astype(str)
monthly_sales.rename(columns={'Month_Year': 'ds', 'Sales': 'y'}, inplace=True) #
```

In [11]:
```
monthly_sales['ds'] = pd.to_datetime(monthly_sales['ds']) # Ensuring ds is in Da
```

In [12]:
```
model = Prophet() # Initialising model
model.fit(monthly_sales) # Fitting the model
```

```
10:32:35 - cmdstanpy - INFO - Chain [1] start processing
10:32:37 - cmdstanpy - INFO - Chain [1] done processing
```

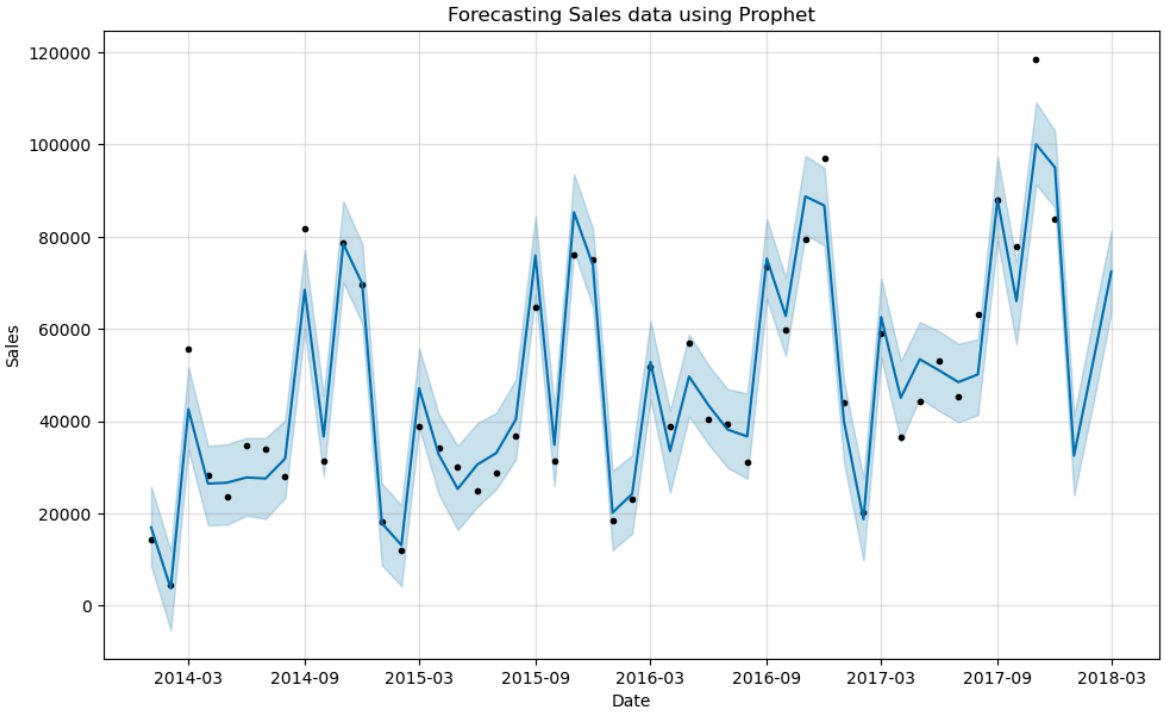
Out[12]:
```
<prophet.forecaster.Prophet at 0x1cc4d2a56d0>
```

In [13]:
```
future = model.make_future_dataframe(periods=3, freq='ME') # Adding future timep
forecast = model.predict(future) # Forecasting the future sales
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

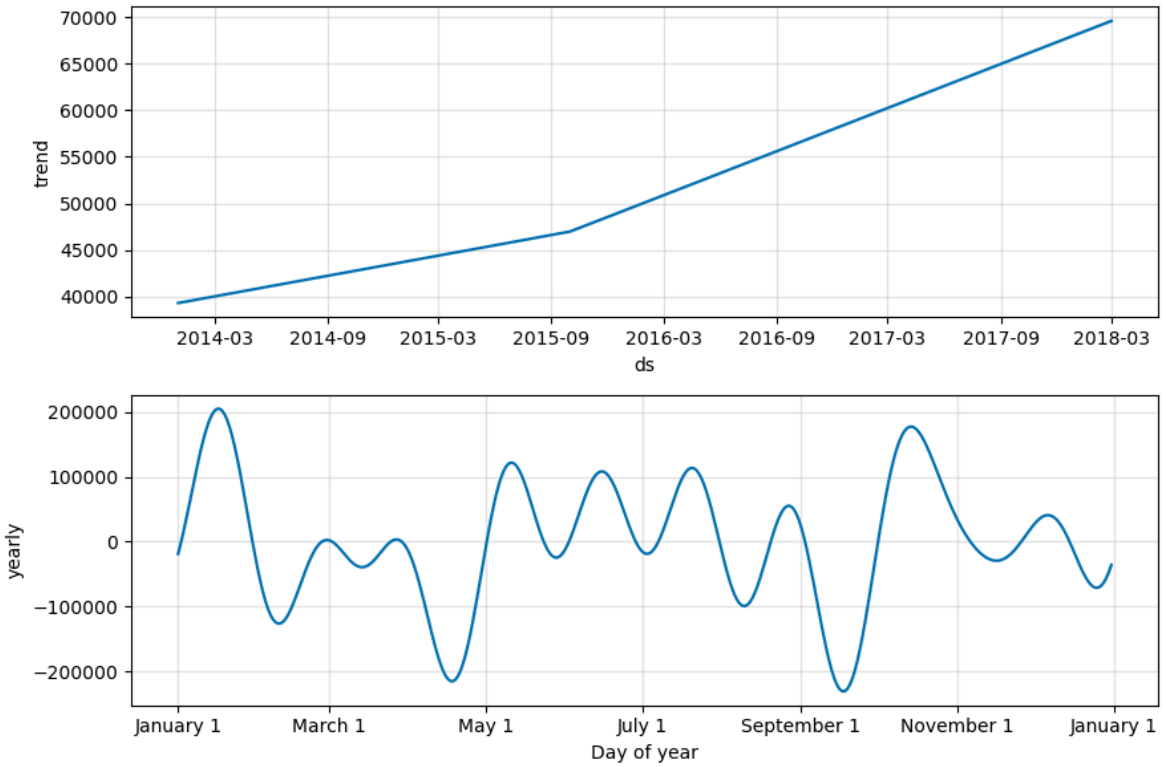
Out[13]:

	ds	yhat	yhat_lower	yhat_upper
46	2017-11-01	100051.534648	91305.333845	109134.025954
47	2017-12-01	94971.216483	86303.433181	102863.922375
48	2017-12-31	32512.761868	23897.964845	41129.934507
49	2018-01-31	53334.163113	44170.953766	62521.889263
50	2018-02-28	72434.094502	63640.631801	81195.041901

In [14]:
```
model.plot(forecast);
plt.title("Forecasting Sales data using Prophet")
plt.xlabel("Date")
plt.ylabel("Sales")
plt.show()
```



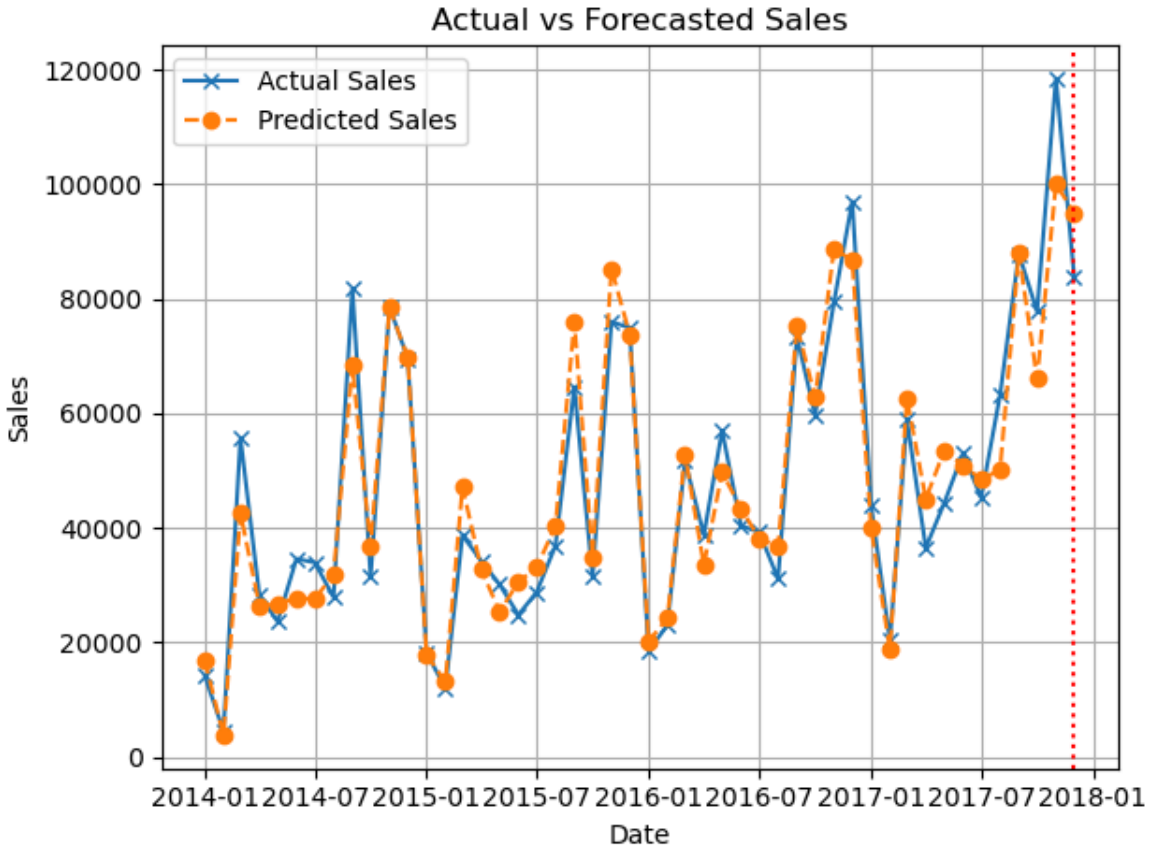
In [15]:
```
model.plot_components(forecast);
plt.show()
```



In [16]:
```
merged=pd.merge(monthly_sales,forecast[['ds','yhat']],on='ds',how='left') # Merg
```

In [17]:
```
# Plotting Forecasted values against Actual values
plt.plot(merged['ds'],merged['y'],label="Actual Sales",marker='x')
plt.plot(merged['ds'],merged['yhat'],label="Predicted Sales",linestyle='--',mark
plt.title('Actual vs Forecasted Sales')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.grid(True)
```

```
plt.tight_layout()
cutoff_date = monthly_sales['ds'].max()
plt.axvline(cutoff_date, color='red', linestyle=':', label='Forecast Start')
plt.show()
```



In [18]:
```
evaluation_df = merged.dropna(subset=['y'])
```

In [19]:
```
rmse=root_mean_squared_error(evaluation_df['y'],evaluation_df['yhat'])
rmse
```

Out[19]: 6762.715680295835

In [20]:
```
average_sales=evaluation_df['y'].mean()
relative_error=rmse/average_sales*100
print("Realtive error:",relative_error)
```

```
Realtive error: 14.130690888380046
```

In [23]:
```
# Exporting final data for dashboard building
forecast_trimmed = forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
final_df = pd.merge(forecast_trimmed, monthly_sales, on='ds', how='left')
final_df.to_excel("sales_forecast_dashboard_data.xlsx", index=False)
```