

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
```

```
sales=pd.read_csv('superstore.csv',encoding='latin-1',parse_dates=['Order Date'])
```

```
print(sales.info())
print(sales.describe())
display(sales)
sales = sales.dropna()
```

```
<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  object
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](1), float64(3), int64(3), object(14)
memory usage: 1.6+ MB
None
```

	Row ID	Order Date	Postal Code	Sales \
count	9994.000000	9994	9994.000000	9994.000000
mean	4997.500000	2016-04-30 00:07:12.259355648	55190.379428	229.858001
min	1.000000	2014-01-03 00:00:00	1040.000000	0.444000
25%	2499.250000	2015-05-23 00:00:00	23223.000000	17.280000
50%	4997.500000	2016-06-26 00:00:00	56430.500000	54.490000
75%	7495.750000	2017-05-14 00:00:00	90008.000000	209.940000
max	9994.000000	2017-12-30 00:00:00	99301.000000	22638.480000
std	2885.163629	NaN	32063.693350	623.245101

	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000
mean	3.789574	0.156203	28.656896
min	1.000000	0.000000	-6599.978000
25%	2.000000	0.000000	1.728750
50%	3.000000	0.200000	8.666500
75%	5.000000	0.200000	29.364000
max	14.000000	0.800000	8399.976000
std	2.225110	0.206452	234.260108

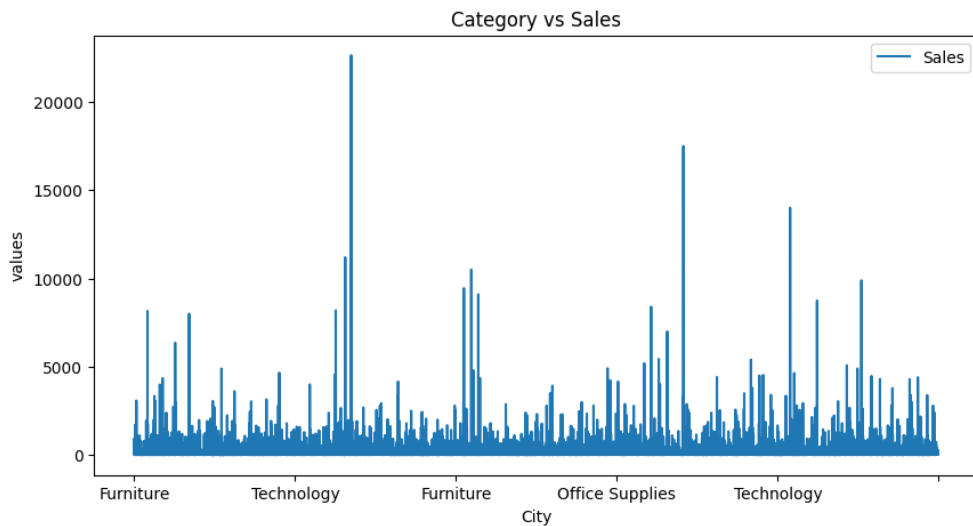
	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales
0	1	CA-2016-152156	2016-11-08	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.96
1	2	CA-2016-152156	2016-11-08	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.94
2	3	CA-2016-138688	2016-06-12	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.62
3	4	US-2015-108966	2015-10-11	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.57
4	5	US-2015-108966	2015-10-11	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.36
...
9989	9990	CA-2014-110422	2014-01-21	1/23/2014	Second Class	TB-21400	Tom Boeckenhauer	Consumer	United States	Miami	33180	South	FUR-FU-10001889	Furniture	Furnishings	Ultra Door Pull Handle	25.24
9990	9991	CA-2017-121258	2017-02-26	3/3/2017	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	92627	West	FUR-FU-10000747	Furniture	Furnishings	Tenex B1-RE Series Chair Mats for Low Pile Car...	91.96
9991	9992	CA-2017-121258	2017-02-26	3/3/2017	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	92627	West	TEC-PH-10003645	Technology	Phones	Aastra 57i VoIP phone	258.57
9992	9993	CA-2017-121258	2017-02-26	3/3/2017	Standard Class	DB-13060	Dave Brooks	Consumer	United States	Costa Mesa	92627	West	OFF-PA-10004041	Office Supplies	Paper	It's Hot Message Books with Stickers, 2 3/4" x 5"	29.60
9993	9994	CA-2017-119914	2017-05-04	5/9/2017	Second Class	CC-12220	Chris Cortes	Consumer	United States	Westminster	92683	West	OFF-AP-10002684	Office Supplies	Appliances	Acco 7-Outlet Masterpiece Power Center, Whintou...	243.16

9994 rows x 21 columns

```
print(sales.columns)
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
      'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
      'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
      'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

```
sales.plot(x='Category',y=['Sales', 'City'],kind='line',figsize=(10,5))
plt.title('Category vs Sales')
plt.xlabel('City')
plt.ylabel('values')
plt.show()
```



```
X=sales[['Sales']]
y=sales[['Profit']]
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

```
monthly_sales = sales.set_index('Order Date')['Sales'].resample('ME').sum().reset_index()
print(monthly_sales)
```

	Order Date	Sales
0	2014-01-31	14236.8950
1	2014-02-28	4519.8920
2	2014-03-31	55691.0090
3	2014-04-30	28295.3450
4	2014-05-31	23648.2870
5	2014-06-30	34595.1276
6	2014-07-31	33946.3930
7	2014-08-31	27909.4685
8	2014-09-30	81777.3508
9	2014-10-31	31453.3930
10	2014-11-30	78628.7167
11	2014-12-31	69545.6205
12	2015-01-31	18174.0756
13	2015-02-28	11951.4110
14	2015-03-31	38726.2520
15	2015-04-30	34195.2085
16	2015-05-31	30131.6865
17	2015-06-30	24797.2920
18	2015-07-31	28765.3250
19	2015-08-31	36898.3322
20	2015-09-30	64595.9180
21	2015-10-31	31404.9235
22	2015-11-30	75972.5635
23	2015-12-31	74919.5212
24	2016-01-31	18542.4910
25	2016-02-29	22978.8150
26	2016-03-31	51715.8750
27	2016-04-30	38750.0390
28	2016-05-31	56987.7280
29	2016-06-30	40344.5340
30	2016-07-31	39261.9630
31	2016-08-31	31115.3743
32	2016-09-30	73410.0249
33	2016-10-31	59687.7450
34	2016-11-30	79411.9658
35	2016-12-31	96999.0430
36	2017-01-31	43971.3740
37	2017-02-28	20301.1334
38	2017-03-31	58872.3528
39	2017-04-30	36521.5361
40	2017-05-31	44261.1102
41	2017-06-30	52981.7257
42	2017-07-31	45264.4160
43	2017-08-31	63120.8880
44	2017-09-30	87866.6520
45	2017-10-31	77776.9232
46	2017-11-30	118447.8250
47	2017-12-31	83829.3188

```
df_prophet = monthly_sales.rename(columns={'Order Date': 'ds', 'Sales': 'y'})
print(df_prophet)
```

	ds	y
0	2014-01-31	14236.8950
1	2014-02-28	4519.8920
2	2014-03-31	55691.0090
3	2014-04-30	28295.3450
4	2014-05-31	23648.2870
5	2014-06-30	34595.1276
6	2014-07-31	33946.3930
7	2014-08-31	27909.4685
8	2014-09-30	81777.3508
9	2014-10-31	31453.3930
10	2014-11-30	78628.7167
11	2014-12-31	69545.6205
12	2015-01-31	18174.0756
13	2015-02-28	11951.4110
14	2015-03-31	38726.2520
15	2015-04-30	34195.2085
16	2015-05-31	30131.6865
17	2015-06-30	24797.2920
18	2015-07-31	28765.3250

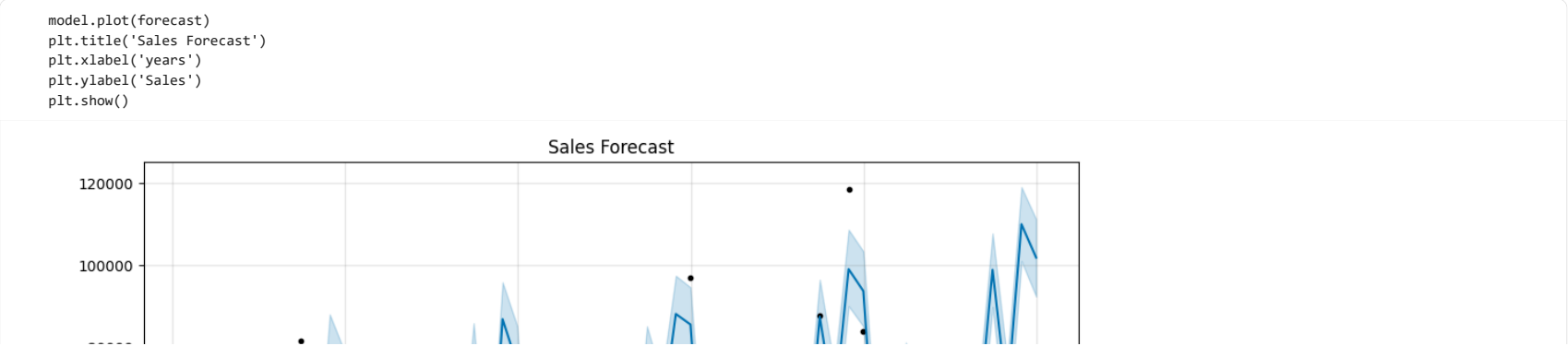
19	2015-08-31	36898.3322
20	2015-09-30	64595.9180
21	2015-10-31	31404.9235
22	2015-11-30	75972.5635
23	2015-12-31	74919.5212
24	2016-01-31	18542.4910
25	2016-02-29	22978.8150
26	2016-03-31	51715.8750
27	2016-04-30	38750.0390
28	2016-05-31	56987.7280
29	2016-06-30	40344.5340
30	2016-07-31	39261.9630
31	2016-08-31	31115.3743
32	2016-09-30	73410.0249
33	2016-10-31	59687.7450
34	2016-11-30	79411.9658
35	2016-12-31	96999.0430
36	2017-01-31	43971.3740
37	2017-02-28	20301.1334
38	2017-03-31	58872.3528
39	2017-04-30	36521.5361
40	2017-05-31	44261.1102
41	2017-06-30	52981.7257
42	2017-07-31	45264.4160
43	2017-08-31	63120.8880
44	2017-09-30	87866.6520
45	2017-10-31	77776.9232
46	2017-11-30	118447.8250
47	2017-12-31	83829.3188

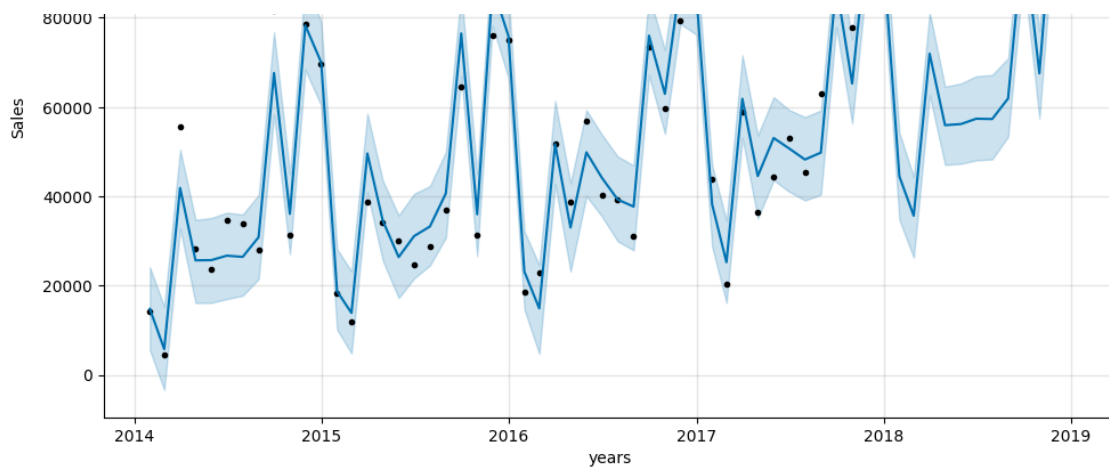
```
from prophet import Prophet
model = Prophet(yearly_seasonality=True, weekly_seasonality=False, daily_seasonality=False)
model.fit(df_prophet)

DEBUG:cmdstanpy:input tempfile: /tmp/tmpf2vxucdg/rkppomuw.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpf2vxucdg/7nuv3xfe.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=15747', 'data', 'file=/tmp/tmpf2vxucdg/rkppomuw.json', 'init=
11:40:18 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
11:40:18 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<prophet.forecaster.Prophet at 0x7b01707c1940>
```

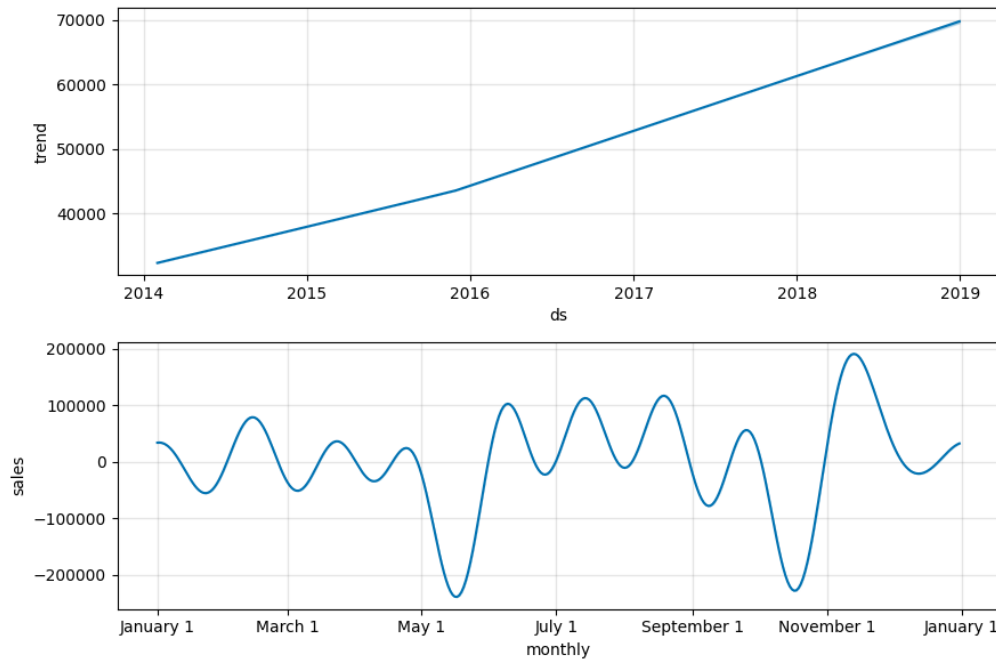
```
future=model.make_future_dataframe(periods=12,freq='ME')
forecast=model.predict(future)
print(forecast)
```

	ds	trend	yhat_lower	yhat_upper	trend_lower \
0	2014-01-31	32303.446177	5526.572268	24142.594057	32303.446177
1	2014-02-28	32773.351064	-3364.238144	15277.297606	32773.351064
2	2014-03-31	33293.602900	32800.538350	50445.107060	33293.602900
3	2014-04-30	33797.072419	16102.815640	34730.598127	33797.072419
4	2014-05-31	34317.324253	16152.868624	35165.520956	34317.324253
5	2014-06-30	34820.793767	17002.149002	36353.624786	34820.793767
6	2014-07-31	35341.045599	17753.011380	35956.062709	35341.045599
7	2014-08-31	35861.297428	21458.892637	40310.981024	35861.297428
8	2014-09-30	36364.766943	58273.879070	76733.133289	36364.766943
9	2014-10-31	36885.018775	27057.597366	45028.543752	36885.018775
10	2014-11-30	37388.488291	68326.395910	88070.620536	37388.488291
11	2014-12-31	37908.740128	60329.954998	79088.484792	37908.740128
12	2015-01-31	38428.991966	10121.847687	28168.631381	38428.991966
13	2015-02-28	38898.896856	4811.006078	23212.207565	38898.896856
14	2015-03-31	39419.164460	40851.209612	58541.265243	39419.164460
15	2015-04-30	39922.649238	25603.996784	43541.224466	39922.649238
16	2015-05-31	40443.866696	17245.455405	35741.889838	40443.866696
17	2015-06-30	40948.342908	21613.055020	40561.102167	40948.342908
18	2015-07-31	41469.634993	24527.851801	42322.838672	41469.634993
19	2015-08-31	41993.240372	30762.650389	49969.670315	41993.240372
20	2015-09-30	42506.004469	67001.088800	85928.713588	42506.004469
21	2015-10-31	43035.860701	26614.486039	45081.505776	43035.860701
22	2015-11-30	43550.949728	78241.391226	95861.797340	43550.949728
23	2015-12-31	44269.855059	66522.892365	85237.200196	44269.855059
24	2016-01-31	44988.760389	14658.490252	32175.252372	44988.760389
25	2016-02-29	45661.750357	4737.259800	24595.722274	45661.750357
26	2016-03-31	46382.514667	42695.881959	61363.200855	46382.514667
27	2016-04-30	47080.028515	23181.343443	42951.521497	47080.028515
28	2016-05-31	47800.949981	40060.879448	59309.251091	47800.949981
29	2016-06-30	48498.661678	35519.281139	53876.943513	48498.661678
30	2016-07-31	49219.630432	29939.752104	48933.590745	49219.630432
31	2016-08-31	49941.292871	27911.506903	46936.267112	49941.292871
32	2016-09-30	50639.904985	67114.316391	85187.065155	50639.904985
33	2016-10-31	51361.804170	53979.321761	72788.419562	51361.804170
34	2016-11-30	52060.416284	78639.678900	97454.572494	52060.416284
35	2016-12-31	52782.315546	76288.248312	94642.083323	52782.315546
36	2017-01-31	53504.214808	28728.986101	46988.498693	53504.214808
37	2017-02-28	54156.252852	16129.283298	34570.242963	54156.252852
38	2017-03-31	54878.152119	53121.248070	71597.155140	54878.152119
39	2017-04-30	55576.764314	35034.844310	53674.898479	55576.764314
40	2017-05-31	56298.663581	43624.862426	62223.344909	56298.663581
41	2017-06-30	56997.275775	41070.515993	59374.383781	56997.275775
42	2017-07-31	57719.175043	39091.062244	57768.245230	57719.175043
43	2017-08-31	58441.074310	40382.076259	59232.111346	58441.074310
44	2017-09-30	59139.686504	77932.924533	96497.970590	59139.686504
45	2017-10-31	59861.585772	56339.063913	75229.176457	59861.585772
46	2017-11-30	60560.197966	90080.327638	108616.933743	60560.197966
47	2017-12-31	61282.097233	85149.895068	103380.296711	61282.097233
48	2018-01-31	62003.996500	34731.477171	54251.351658	62000.114925
49	2018-02-28	62656.034548	26267.589286	44349.322728	62642.989381
50	2018-03-31	63377.933816	62655.815980	81227.934784	63350.859372
51	2018-04-30	64076.546010	47071.008373	64562.227593	64032.362271
52	2018-05-31	64798.445277	47308.605953	65294.663846	64735.241935
53	2018-06-30	65497.057472	48081.192194	66929.713738	65410.365652
54	2018-07-31	66218.956739	48295.983106	67190.710122	66106.402841
55	2018-08-31	66940.856006	53346.788149	70962.239902	66802.894483
56	2018-09-30	67639.468201	89873.266257	107716.997674	67477.906827





```
model.plot_components(forecast)
plt.xlabel('monthly')
plt.ylabel('sales')
plt.show()
```



Start coding or [generate](#) with AI.