assignment8-2JRahman

July 31, 2022

[1]:
```
# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Loading csv into dataframe
sales_df = pd.read_csv('us_retail_sales.csv')
# Checking dataframe
sales_df
```

[2]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	\
0	1992	146925	147223	146805	148032	149010	149800	150761.0	151067.0	
1	1993	157555	156266	154752	158979	160605	160127	162816.0	162506.0	
2	1994	167518	169649	172766	173106	172329	174241	174781.0	177295.0	
3	1995	182413	179488	181013	181686	183536	186081	185431.0	186806.0	
4	1996	189135	192266	194029	194744	196205	196136	196187.0	196218.0	
5	1997	202371	204286	204990	203399	201699	204675	207014.0	207635.0	
6	1998	209666	209552	210832	213633	214639	216337	214841.0	213636.0	
7	1999	223997	226250	227417	229037	231235	231903	233948.0	236566.0	
8	2000	243436	247133	249825	245831	246201	248160	247176.0	247576.0	
9	2001	252654	252704	250328	254763	255218	254022	252997.0	254560.0	
10	2002	256307	257670	257059	261333	257573	259786	262769.0	265043.0	
11	2003	267230	263188	267820	267197	267362	270396	273352.0	277965.0	
12	2004	278913	280932	286209	282952	288252	284133	287358.0	287941.0	
13	2005	296696	300557	301308	303760	301776	310989	313520.0	310046.0	
14	2006	322348	320171	320869	322561	321794	323184	324204.0	325324.0	
15	2007	327181	327953	330579	329560	334202	331076	332342.0	334169.0	
16	2008	337412	334584	335193	334843	337947	338311	336771.0	334045.0	
17	2009	298673	297631	292300	293614	296501	302169	302802.0	309023.0	
18	2010	308299	308628	316003	318707	315604	314925	315632.0	317408.0	
19	2011	332357	334710	338007	339884	339303	341600	341373.0	342288.0	
20	2012	352862	357379	358719	356849	356018	352043	353891.0	358450.0	
21	2013	367009	372291	369081	367514	369493	371041	373554.0	372489.0	
22	2014	373033	378581	382601	386689	387100	388106	388359.0	391305.0	
23	2015	385648	385157	391420	391356	394718	395464	398193.0	398105.0	
24	2016	394749	398105	396911	398190	400143	404756	403730.0	403968.0	
25	2017	416081	415503	414620	416889	414540	416505	416744.0	417179.0	
26	2018	432148	434106	433232	435610	439996	438191	440703.0	439278.0	

```
27  2019  440751  439996  447167  448709  449552  450927  454012.0  456500.0
28  2020  460586  459610  434281  379892  444631  476343  481627.0  483716.0
29  2021  520162  504458  559871  562269  548987  550782       NaN       NaN

         SEP       OCT       NOV       DEC
0   152588.0  153521.0  153583.0  155614.0
1   163258.0  164685.0  166594.0  168161.0
2   178787.0  180561.0  180703.0  181524.0
3   187366.0  186565.0  189055.0  190774.0
4   198859.0  200509.0  200174.0  201284.0
5   208326.0  208078.0  208936.0  209363.0
6   215720.0  219483.0  221134.0  223179.0
7   237481.0  237553.0  240544.0  245485.0
8   251837.0  251221.0  250331.0  250658.0
9   249845.0  267999.0  260514.0  256549.0
10  260626.0  261953.0  263568.0  265930.0
11  276430.0  274764.0  278298.0  277612.0
12  293139.0  295115.0  296177.0  299763.0
13  310673.0  310479.0  313303.0  313473.0
14  323236.0  322678.0  323343.0  326849.0
15  335442.0  337530.0  341133.0  336189.0
16  328343.0  314830.0  301332.0  294025.0
17  301033.0  304154.0  306675.0  308413.0
18  320080.0  323900.0  327745.0  329627.0
19  345496.0  347924.0  349304.0  349744.0
20  361470.0  361991.0  362876.0  364488.0
21  372505.0  373663.0  373914.0  377032.0
22  389860.0  390506.0  391805.0  388569.0
23  396248.0  394503.0  396240.0  397052.0
24  405958.0  407395.0  406061.0  412610.0
25  426501.0  426933.0  431158.0  433282.0
26  438985.0  444038.0  445242.0  434803.0
27  452849.0  455486.0  457658.0  458055.0
28  493327.0  493991.0  488652.0  484782.0
29       NaN       NaN       NaN       NaN
```

[3]:
```
# Convert to long format
sales_df_long = pd.melt(sales_df, id_vars = 'YEAR', value_vars = ['JAN', 'FEB',␣
 ↪'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG',
                                                                   'SEP', 'OCT',␣
 ↪'NOV', 'DEC'])
sales_df_long
```

[3]:
```
    YEAR variable     value
0   1992      JAN  146925.0
1   1993      JAN  157555.0
2   1994      JAN  167518.0
```

```
3    1995      JAN  182413.0
4    1996      JAN  189135.0
..    …        …       …
355  2017      DEC  433282.0
356  2018      DEC  434803.0
357  2019      DEC  458055.0
358  2020      DEC  484782.0
359  2021      DEC       NaN

[360 rows x 3 columns]
```

[4]:
```
# Renaming columns
sales_df_long = sales_df_long.rename(columns = {'variable':'MONTH', 'value':
 ↪'SALES'})
sales_df_long
```

[4]:
```
     YEAR MONTH     SALES
0    1992   JAN  146925.0
1    1993   JAN  157555.0
2    1994   JAN  167518.0
3    1995   JAN  182413.0
4    1996   JAN  189135.0
..    …    …       …
355  2017   DEC  433282.0
356  2018   DEC  434803.0
357  2019   DEC  458055.0
358  2020   DEC  484782.0
359  2021   DEC       NaN

[360 rows x 3 columns]
```

[5]:
```
# Converting months to numerical
sales_df_long['MONTH'] = sales_df_long['MONTH'].replace({'JAN':1, 'FEB':2,␣
 ↪'MAR':3, 'APR':4, 'MAY':5, 'JUN':6, 'JUL':7,
                                                          'AUG':8, 'SEP':9,␣
 ↪'OCT':10, 'NOV':11, 'DEC':12})
sales_df_long
```

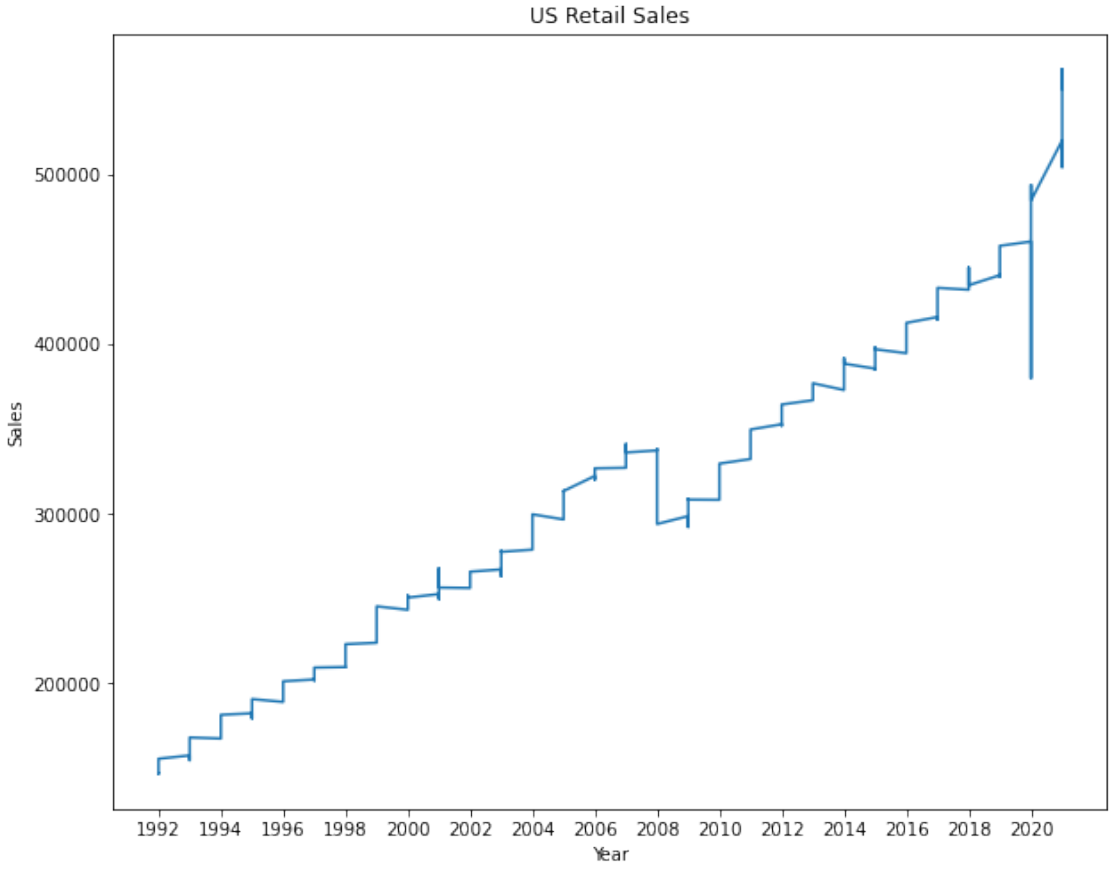
[5]:
```
     YEAR  MONTH     SALES
0    1992      1  146925.0
1    1993      1  157555.0
2    1994      1  167518.0
3    1995      1  182413.0
4    1996      1  189135.0
..    …    …       …
355  2017     12  433282.0
356  2018     12  434803.0
```

```
357  2019     12  458055.0
358  2020     12  484782.0
359  2021     12       NaN

[360 rows x 3 columns]
```

[6]:
```
# Sorting by year then month and dropping NaN values
sales_df_long = sales_df_long.sort_values(by = ['YEAR', 'MONTH'])
sales_df_long.dropna(inplace=True)
sales_df_long
```

[6]:
```
      YEAR  MONTH     SALES
0     1992      1  146925.0
30    1992      2  147223.0
60    1992      3  146805.0
90    1992      4  148032.0
120   1992      5  149010.0
..     …    …        …
59    2021      2  504458.0
89    2021      3  559871.0
119   2021      4  562269.0
149   2021      5  548987.0
179   2021      6  550782.0

[354 rows x 3 columns]
```

[7]:
```
# Plot the data with proper labeling and make some observations on the graph.
plt.figure(figsize=(10,8))
plt.plot('YEAR', 'SALES', data = sales_df_long)
plt.title('US Retail Sales')
plt.xlabel('Year')
plt.ylabel('Sales')
plt.xticks(np.arange(1992, 2021, 2))
plt.show()
```



Over the years sales have risen linearly. However, there are three times where it didn't rise like usually. In 2008, 2020, and 2021. In 2008, there was a recession because of a housing crises and it caused a dip in sales. In 2020, Covid-19 caused a dip in sales. In 2021, the vaccine rollout and inflation rates caused a spike in sales.

[8]:
```
# 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.
sales_train = sales_df_long[:-12]
sales_test = sales_df_long[-12:]
X_train = sales_train[['YEAR', 'MONTH']]
X_test = sales_test[['YEAR', 'MONTH']]
y_train = sales_train['SALES']
y_test = sales_test['SALES']
```

[9]:
```
# Use the training set to build a predictive model for the monthly retail sales.
from sklearn.linear_model import LinearRegression
from sklearn import metrics

model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

[9]:
```
LinearRegression()
```

[10]:
```
# Use the model to predict the monthly retail sales on the last year of data.
pred = model.predict(X_test)
```

[11]:
```
# Report the RMSE of the model predictions on the test set.
print('RMSE:', metrics.mean_squared_error(y_test, pred, squared=False))
```

```
RMSE: 66511.08639264444
```

[ ]:
```

```