Project Name - Rossman_Sales_Prediction

Project Type - Regression

Contribution - Individual

Project Summary -

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

GitHub Link -

Provide your GitHub Link here.

Problem Statement

We have historical sales data for 1,115 Rossmann stores. The task is to forecast the "Sales" column for the test set. Note that some stores in the dataset were temporarily closed for refurbishment. **Predict the sales by store for future**

Let's Begin !

1. Know Your Data

Import Libraries

```
# Importing Libraries
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Dataset Loading

```
# Load Dataset
sheet_url = "https://docs.google.com/spreadsheets/d/19bcp7_pd93Xjvawetmrd6j6
csv_url = sheet_url.replace("/edit?usp=sharing", "/export?format=csv")
```

```
ind_store = pd.read_csv(csv_url)
ind_store.head()
```

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMon
0	1	c	a	1270.0	
1	2	a	a	570.0	1
2	3	a	a	14130.0	1
3	4	c	c	620.0	
4	5	a	a	29910.0	

Next steps: Generate code with ind_store New interactive sheet

```
csv_url_2 = "https://docs.google.com/spreadsheets/d/1EGGhpUbxKC-kFYEptWMoDJ-
```

```
all_stores = pd.read_csv(csv_url_2)
```

Dataset First View

```
# Dataset First Look
all_stores.head(5)
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	Schoo
0	1	5	2015-07-31	5263	555	1	1	0	
1	2	5	2015-07-31	6064	625	1	1	0	
2	3	5	2015-07-31	8314	821	1	1	0	

```
ind_store.head(5)
```

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMon
0	1	c	a	1270.0	9
1	2	a	a	570.0	11
2	3	a	a	14130.0	12
3	4	c	c	620.0	9
4	5	a	a	29910.0	4

Next steps: Generate code with `ind_store` New interactive sheet

```
#merge of data
stores_df = pd.merge(all_stores, ind_store, on='Store')
stores_df
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday
0	1	5	2015-07-31	5263	555	1	1	0
1	2	5	2015-07-31	6064	625	1	1	0
2	3	5	2015-07-31	8314	821	1	1	0
3	4	5	2015-07-31	13995	1498	1	1	0
4	5	5	2015-07-31	4822	559	1	1	0
...	...	...	...	...	...	...	...	...
1017204	1111	2	2013-01-01	0	0	0	0	a
1017205	1112	2	2013-01-01	0	0	0	0	a
1017206	1113	2	2013-01-01	0	0	0	0	a
1017207	1114	2	2013-01-01	0	0	0	0	a
1017208	1115	2	2013-01-01	0	0	0	0	a

1017209 rows × 18 columns

Dataset Rows & Columns count

```
# Dataset Rows & Columns count
stores_df.shape
```

```
(1017209, 18)
```

Dataset Information

```
# Dataset Info
stores_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 18 columns):
 #   Column                     Non-Null Count    Dtype  
---  ------                     --------------    -----  
 0   Store                      1017209 non-null  int64  
 1   DayOfWeek                  1017209 non-null  int64  
 2   Date                       1017209 non-null  object 
 3   Sales                      1017209 non-null  int64  
 4   Customers                  1017209 non-null  int64  
 5   Open                       1017209 non-null  int64  
 6   Promo                      1017209 non-null  int64  
 7   StateHoliday               1017209 non-null  object 
 8   SchoolHoliday              1017209 non-null  int64  
 9   StoreType                  1017209 non-null  object 
 10  Assortment                 1017209 non-null  object 
 11  CompetitionDistance        1014567 non-null  float64
 12  CompetitionOpenSinceMonth  693861 non-null   float64
 13  CompetitionOpenSinceYear   693861 non-null   float64
 14  Promo2                     1017209 non-null  int64  
 15  Promo2SinceWeek            509178 non-null   float64
 16  Promo2SinceYear            509178 non-null   float64
 17  PromoInterval              509178 non-null   object 
dtypes: float64(5), int64(8), object(5)
memory usage: 139.7+ MB
```

Duplicate Values

```
# Dataset Duplicate Value Count
stores_df.duplicated().sum()
```

```
np.int64(0)
```

Missing Values/Null Values

```
# Missing Values/Null Values Count
stores_df.isnull().sum()
```

	0
Store	0
DayOfWeek	0
Date	0
Sales	0
Customers	0
Open	0
Promo	0
StateHoliday	0
SchoolHoliday	0
StoreType	0
Assortment	0
CompetitionDistance	2642
CompetitionOpenSinceMonth	323348
CompetitionOpenSinceYear	323348
Promo2	0
Promo2SinceWeek	508031
Promo2SinceYear	508031
PromoInterval	508031

dtype: int64

What did you know about your dataset?

Some of the competitionopensincemonth or year are null. It might be because there isn't much competition around the place.

For Promo2since week/year/interval it might be because promo2 has not been done

2. Understanding Your Variables

```
# Dataset Columns
stores_df.columns
```

```
Index(['Store', 'DayOfWeek', 'Date', 'Sales', 'Customers', 'Open', 'Promo',
       'StateHoliday', 'SchoolHoliday', 'StoreType', 'Assortment',
       'CompetitionDistance', 'CompetitionOpenSinceMonth',
       'CompetitionOpenSinceYear', 'Promo2', 'Promo2SinceWeek',
```

```
      'Promo2SinceYear', 'PromoInterval'],
     dtype='object')
```

```
# Dataset Describe
stores_df.describe()
```

	Store	DayOfWeek	Sales	Customers	Open	
count	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.017209e+06	1.
mean	5.584297e+02	3.998341e+00	5.773819e+03	6.331459e+02	8.301067e-01	3
std	3.219087e+02	1.997391e+00	3.849926e+03	4.644117e+02	3.755392e-01	4
min	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.
25%	2.800000e+02	2.000000e+00	3.727000e+03	4.050000e+02	1.000000e+00	0.
50%	5.580000e+02	4.000000e+00	5.744000e+03	6.090000e+02	1.000000e+00	0.
75%	8.380000e+02	6.000000e+00	7.856000e+03	8.370000e+02	1.000000e+00	1.
max	1.115000e+03	7.000000e+00	4.155100e+04	7.388000e+03	1.000000e+00	1.

Variables Description

Fields	Description
Id	Unique entry id
Store	store_id
Sales	Sales made for the day
Customers	Footfall for the day
Open	Open or closed
StateHoliday	State Holiday or not
SchoolHoliday	School Holiday or not
StoreType	Type of stores
Assortment	Type of assortment
CompetitionDistance	Distance from the nearest competition
Promo	Store running promotion or not
Promo2	Store running consecutive promotion or not

Check Unique Values for each variable.

```
# Check Unique Values for each variable.
stores_df.nunique()
```

	0
Store	1115
DayOfWeek	7
Date	942
Sales	21734
Customers	4086
Open	2
Promo	2
StateHoliday	5
SchoolHoliday	2
StoreType	4
Assortment	3
CompetitionDistance	654
CompetitionOpenSinceMonth	12
CompetitionOpenSinceYear	23
Promo2	2
Promo2SinceWeek	24
Promo2SinceYear	7
PromoInterval	3

dtype: int64

3. ***Data Wrangling***

Data Wrangling Code

```
# Write your code to make your dataset analysis ready.
stores_df = stores_df[~stores_df['CompetitionDistance'].isnull()]
```

```
#dropping unnecessary columns
stores_df.drop(['CompetitionOpenSinceMonth','CompetitionOpenSinceYear'], axi
stores_df.head(5)
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	Schoo
0	1	5	2015-07-31	5263	555	1	1	0	
1	2	5	2015-07-31	6064	625	1	1	0	
2	3	5	2015-07-31	8314	821	1	1	0	
3	4	5	2015-07-31	13995	1498	1	1	0	
4	5	5	2015-07-31	4822	559	1	1	0	

```
#fillig null values
stores_df['Promo2SinceWeek'].fillna(0, inplace=True)
stores_df['Promo2SinceYear'].fillna(0, inplace=True)
stores_df['PromoInterval'].fillna('Never',inplace = True)
```

```
stores_df.head(5)
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	Schoo
0	1	5	2015-07-31	5263	555	1	1	0	
1	2	5	2015-07-31	6064	625	1	1	0	
2	3	5	2015-07-31	8314	821	1	1	0	
3	4	5	2015-07-31	13995	1498	1	1	0	
4	5	5	2015-07-31	4822	559	1	1	0	

```
stores_df['PromoInterval'].unique()
```

```
array(['Never', 'Jan,Apr,Jul,Oct', 'Feb,May,Aug,Nov', 'Mar,Jun,Sept,Dec'],
      dtype=object)
```

```
#changing datatype to datetime for date
stores_df['Date'] = stores_df['Date'].astype('datetime64[ns]')
```

```
#making new column for further analysis
stores_df.head()
import datetime
```

```
stores_df['Year'] = pd.DatetimeIndex(stores_df['Date']).year
stores_df['Year'] = stores_df['Year'].astype(int)
stores_df['number_of_year_promotion_till_now'] = stores_df['Year'] - stores_
stores_df.head(5)
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	Schoo
0	1	5	2015-07-31	5263	555	1	1	0	
1	2	5	2015-07-31	6064	625	1	1	0	
2	3	5	2015-07-31	8314	821	1	1	0	
3	4	5	2015-07-31	13995	1498	1	1	0	
4	5	5	2015-07-31	4822	559	1	1	0	

```
stores_df['number_of_year_promotion_till_now'] = stores_df['number_of_year_p
```

```
stores_df.head()
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	Schoo
0	1	5	2015-07-31	5263	555	1	1	0	
1	2	5	2015-07-31	6064	625	1	1	0	
2	3	5	2015-07-31	8314	821	1	1	0	
3	4	5	2015-07-31	13995	1498	1	1	0	
4	5	5	2015-07-31	4822	559	1	1	0	

```
stores_df['month'] = stores_df['Date'].dt.month

stores_df.head()
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	Schoo
0	1	5	2015-07-31	5263	555	1	1	0	
1	2	5	2015-07-31	6064	625	1	1	0	
2	3	5	2015-07-31	8314	821	1	1	0	
3	4	5	2015-07-31	13995	1498	1	1	0	
4	5	5	2015-07-31	4822	559	1	1	0	

```
#unwanted column
stores_df.drop('Customers',axis = 1, inplace = True)
```

```
#removing rows when stores were closed
stores_df = stores_df[~(stores_df['Open'] == 0)]
```

```
stores_df.sort_values(by='Date',ascending = False).head(5)
```

	Store	DayOfWeek	Date	Sales	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	1	1	0	1
742	743	5	2015-07-31	5085	1	1	0	1
748	749	5	2015-07-31	6612	1	1	0	1
747	748	5	2015-07-31	7481	1	1	0	1
746	747	5	2015-07-31	10708	1	1	0	1

```
#stateholiday had different variable named a,b,c when there was a holiday sc
stores_df['StateHoliday'] = stores_df['StateHoliday'].apply(lambda x: x if x
```

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

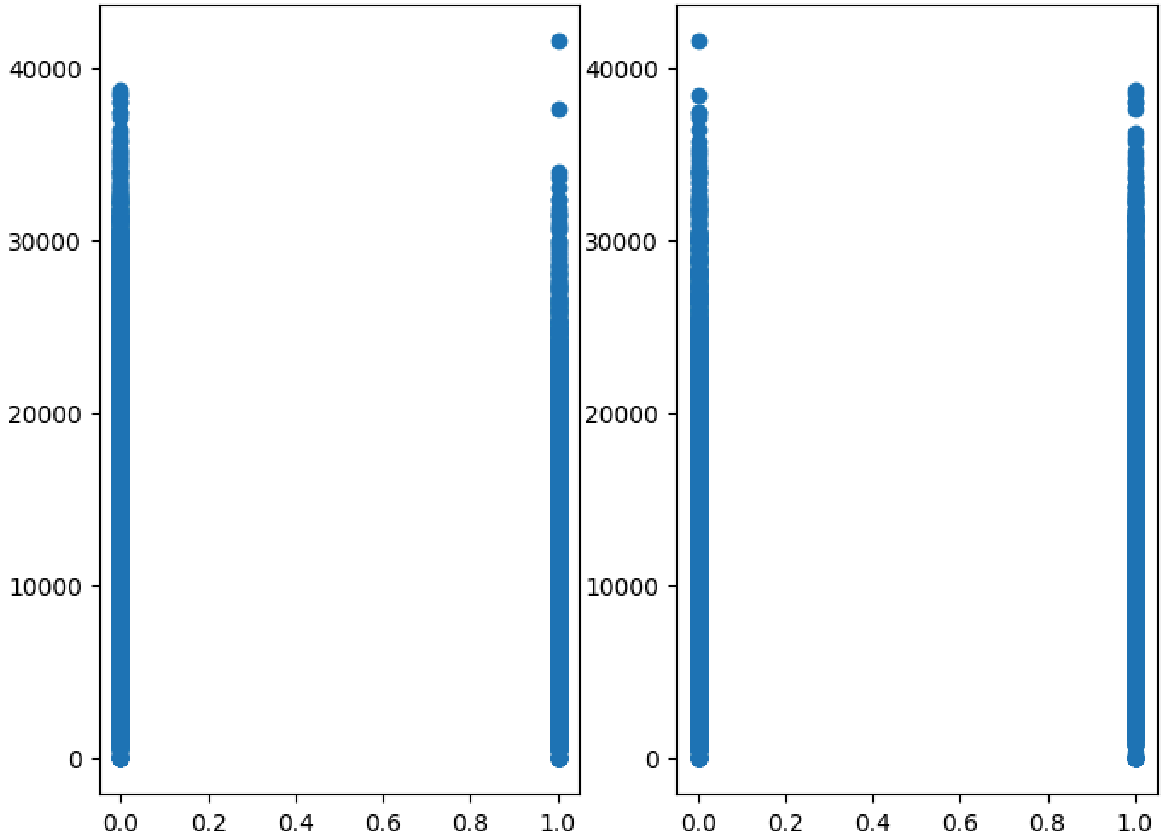
Chart - 1

```
# Chart - 1 visualization code
plt.figure(figsize=(8, 6))

plt.subplot(1,2,1)
plt.scatter(stores_df['Promo2'],stores_df['Sales'])

plt.subplot(1,2,2)
plt.scatter(stores_df['Promo'],stores_df['Sales'])
```

```
<matplotlib.collections.PathCollection at 0x7d7a12a645c0>
```



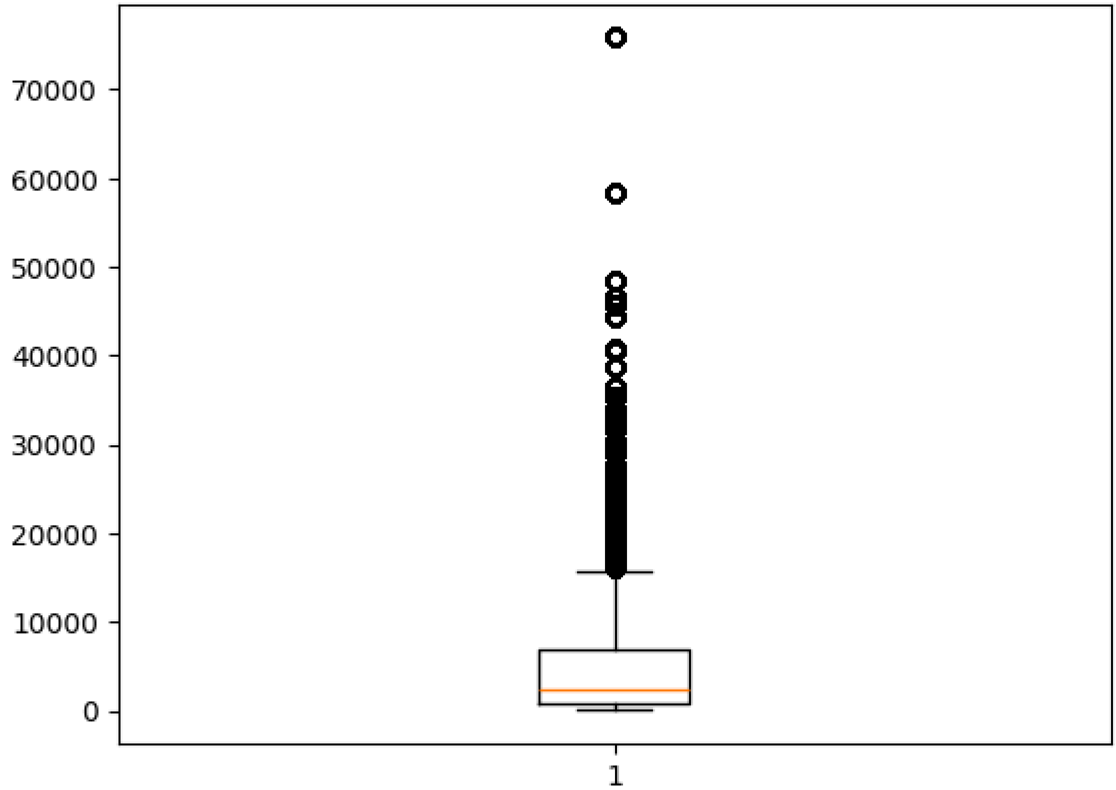
1. Why did you pick the specific chart?

Shows relationship between 2 variable in better format

Chart - 2

```
# Chart - 2 visualization code
plt.boxplot(stores_df['CompetitionDistance'])
```

{'whiskers': [<matplotlib.lines.Line2D at 0x7d7a12a3dfd0>,
  <matplotlib.lines.Line2D at 0x7d7a12a3e2d0>],
 'caps': [<matplotlib.lines.Line2D at 0x7d7a12a3e600>,
  <matplotlib.lines.Line2D at 0x7d7a12a3e930>],
 'boxes': [<matplotlib.lines.Line2D at 0x7d7a12a4e330>],
 'medians': [<matplotlib.lines.Line2D at 0x7d7a12a3ec00>],
 'fliers': [<matplotlib.lines.Line2D at 0x7d7a12a3ef90>],
 'means': []}



1. Why did you pick the specific chart?

To check the distribution of the value in the column

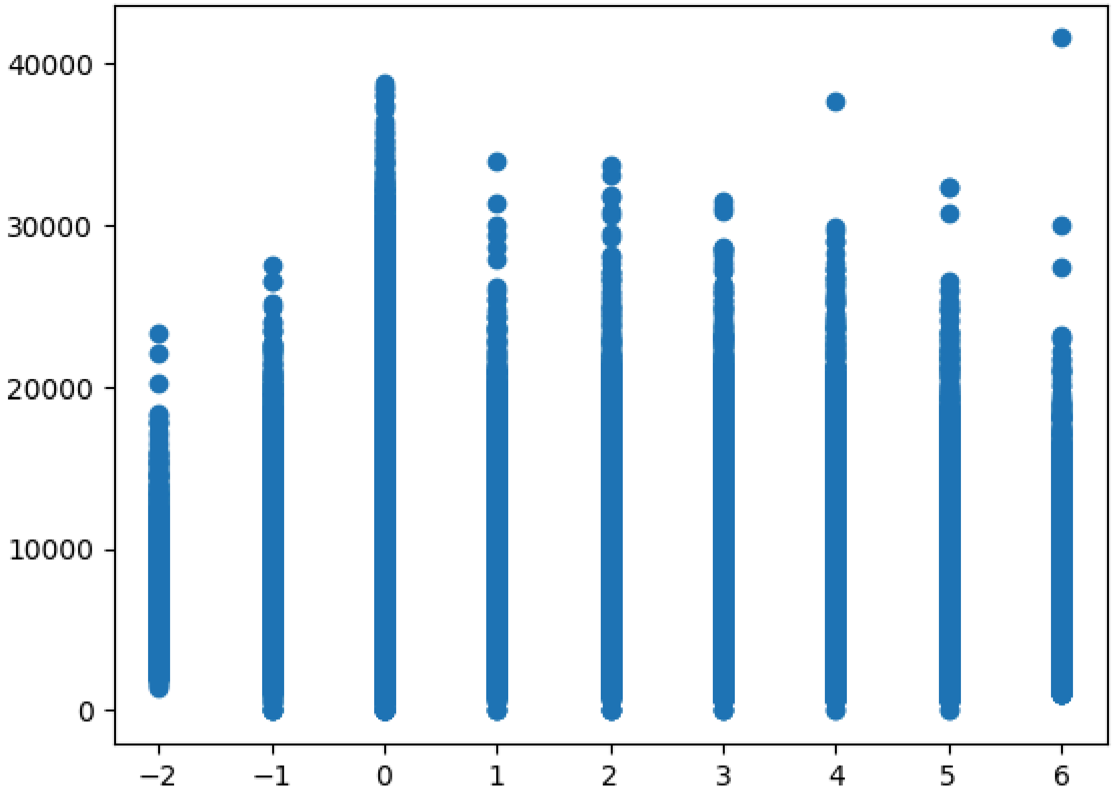
2. What is/are the insight(s) found from the chart?

Need to take care of outliers, because it has many values that are out of IQR that might affect the performance of the model

Chart - 3

```
# Chart - 3 visualization code
plt.scatter(stores_df['number_of_year_promotion_till_now'],stores_df['Sales'
```

```
<matplotlib.collections.PathCollection at 0x7d7a12a47590>
```



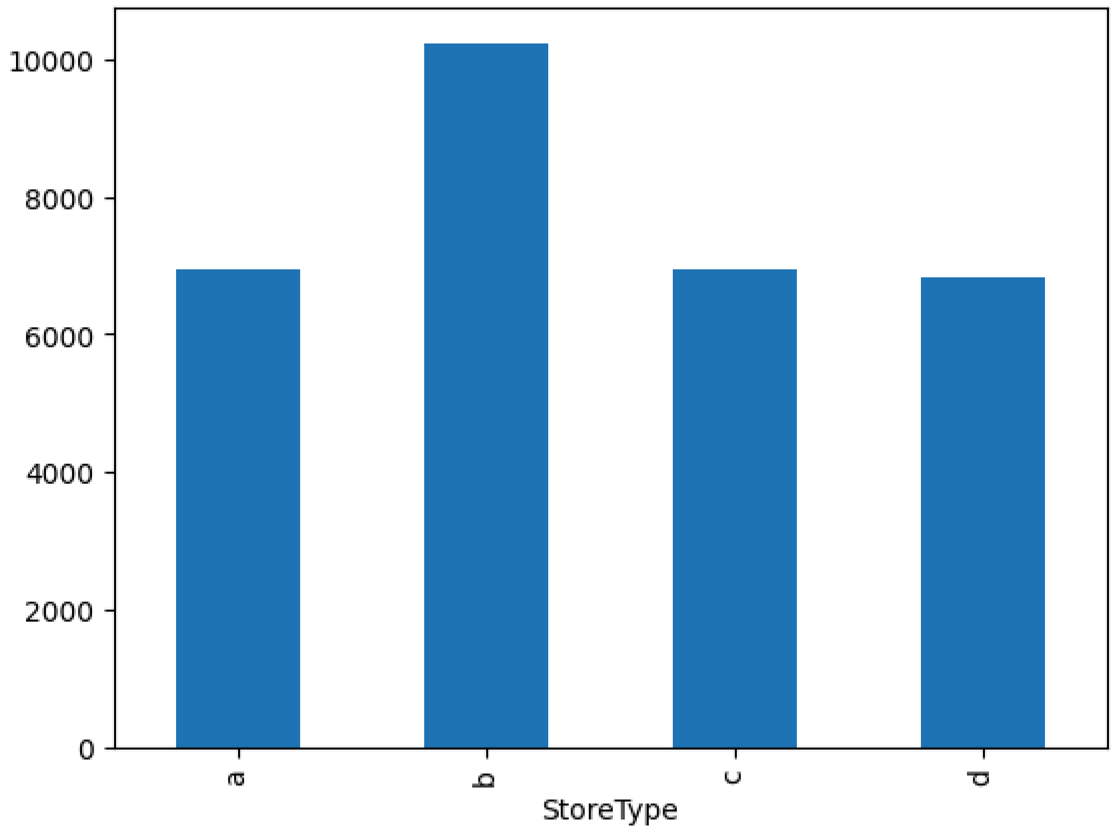
1. Why did you pick the specific chart?

good for showing relationship between variables

Chart - 4

```
# Chart - 4 visualization code
stores_df.groupby('StoreType')['Sales'].mean().plot(kind = 'bar')
```

```
<Axes: xlabel='StoreType'>
```



1. Why did you pick the specific chart?

best to check if categorical variable have any kind of relationship with the target variable

2. What is/are the insight(s) found from the chart?

Store b has way much more sales compared to other stores

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Answer Here

Chart - 5

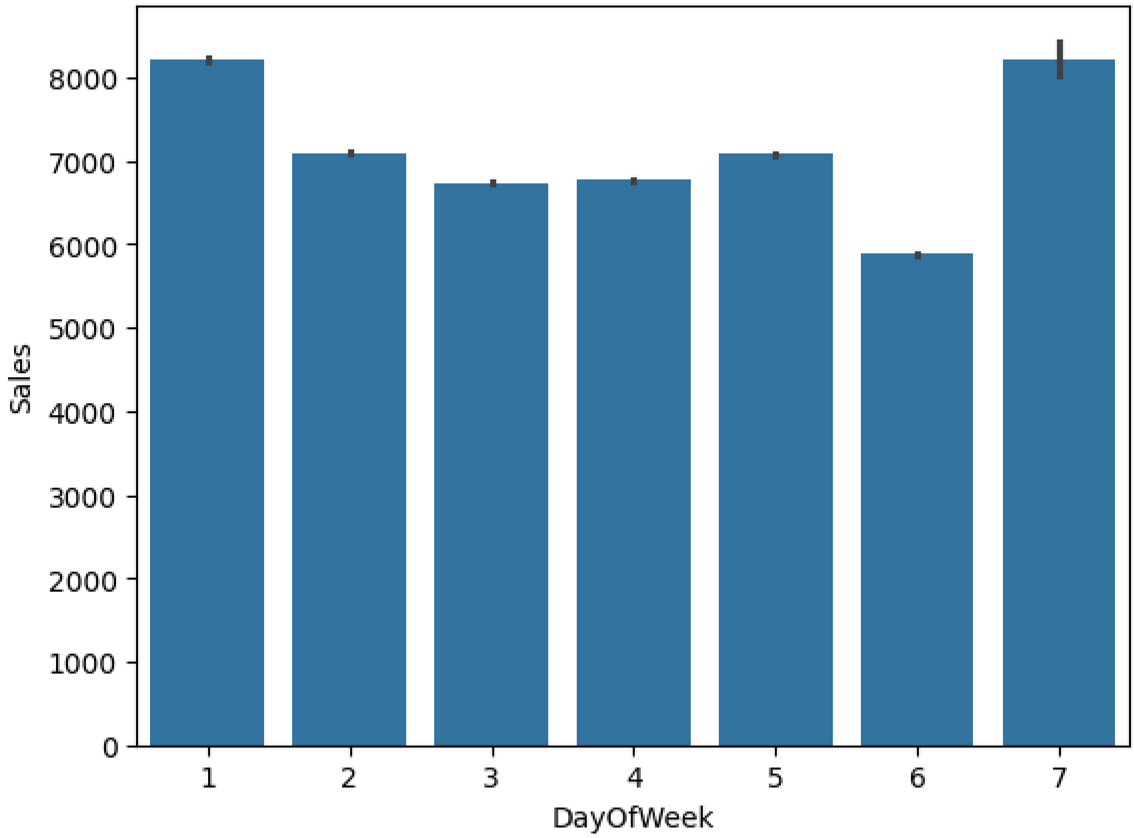
```
stores_df.groupby("DayOfWeek")["Sales"].mean()
```

DayOfWeek	Sales
1	8219.549208
2	7091.767012
3	6731.642902
4	6770.404154
5	7076.162931
6	5880.475133
7	8224.723908

dtype: float64

```
# Chart - 5 visualization code
sns.barplot(x = stores_df['DayOfWeek'],y = stores_df['Sales'])
```

```
<Axes: xlabel='DayOfWeek', ylabel='Sales'>
```



1. Why did you pick the specific chart?

Better for vizulaization of relationship between categorical and numeric value

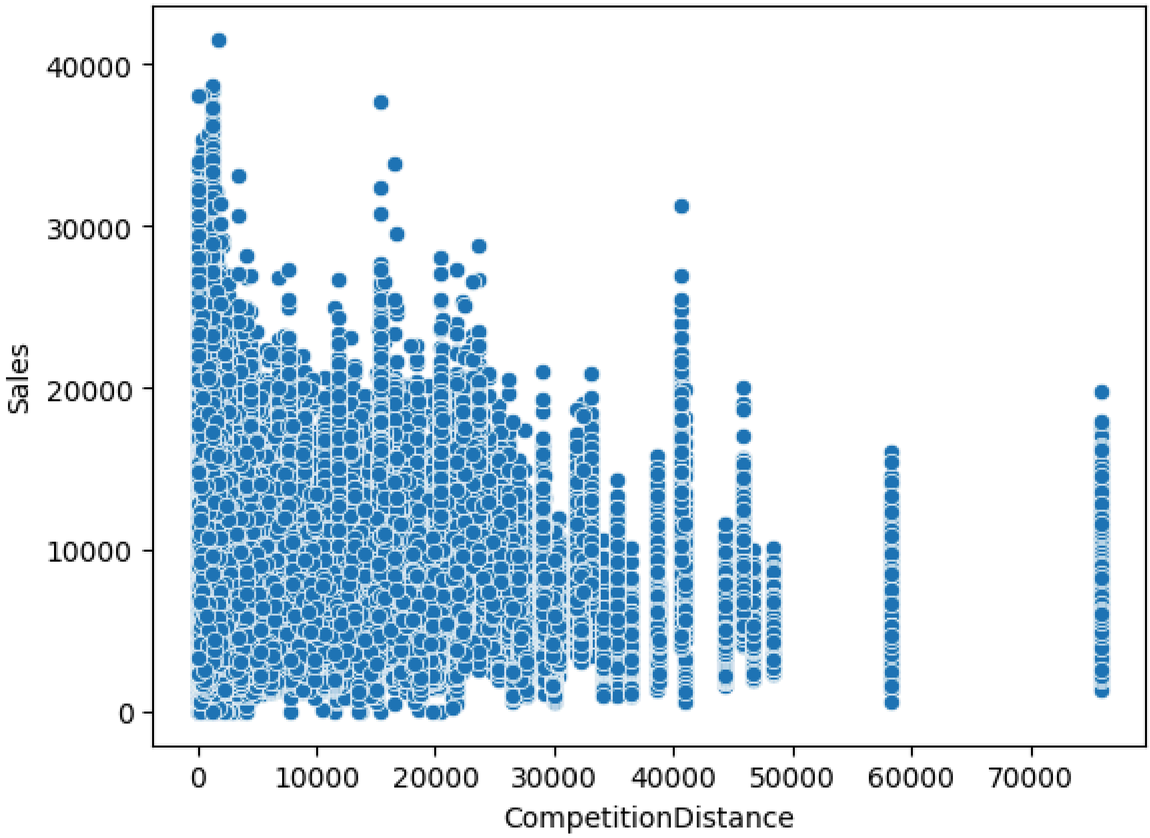
2. What is/are the insight(s) found from the chart?

Day of The week affect the sales, not linearly but they do

Chart - 6

```
# Chart - 6 visualization code
stores_df.head()
sns.scatterplot(x = stores_df['CompetitionDistance'], y = stores_df['Sales']
```

```
<Axes: xlabel='CompetitionDistance', ylabel='Sales'>
```



1. Why did you pick the specific chart?

Relationship between two variables

2. What is/are the insight(s) found from the chart?

sales has some kind of relationship with competition distance,not linear but some kind of

3. Will the gained insights help creating a positive business impact?

Are there any insights that lead to negative growth? Justify with specific reason.

Answer Here

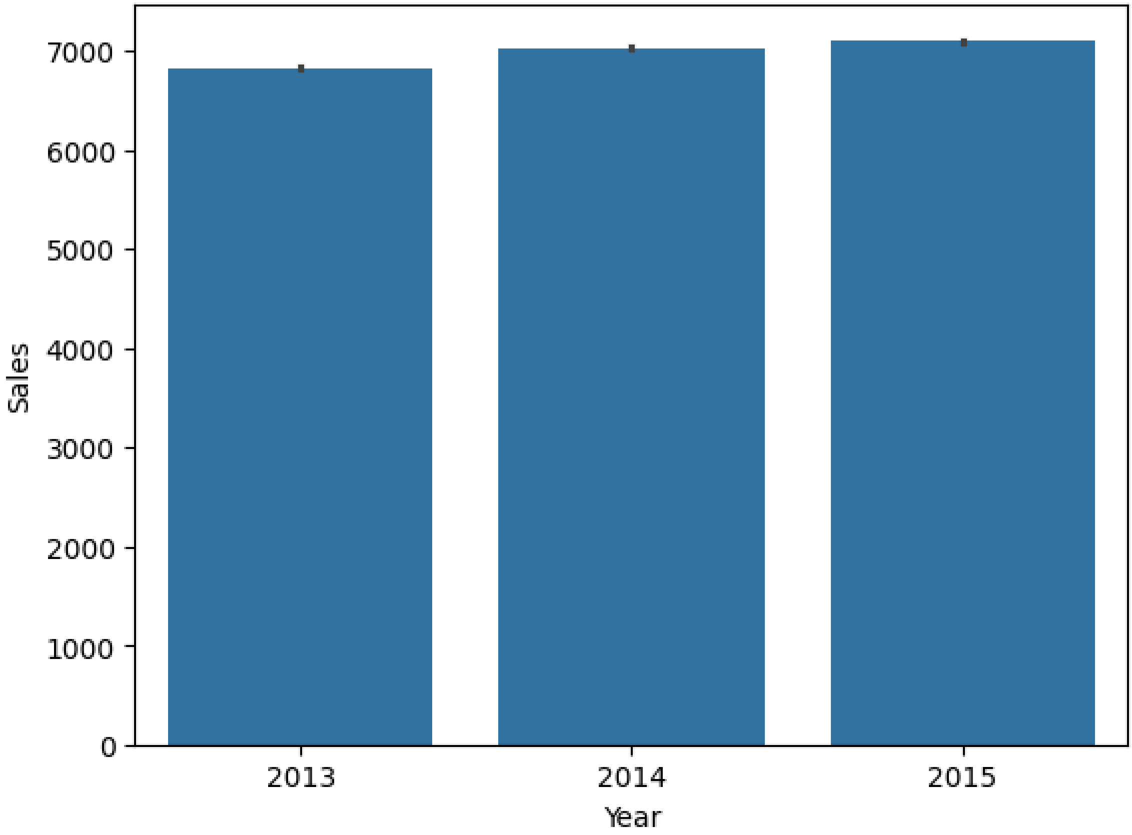
Chart - 7

```
stores_df['Year'].unique()
```

```
array([2015, 2014, 2013])
```

```
# Chart - 7 visualization code
sns.barplot(x = stores_df['Year'], y = stores_df['Sales'])
```

```
<Axes: xlabel='Year', ylabel='Sales'>
```




```
stores_df.columns
```

```
Index(['Store', 'DayOfWeek', 'Date', 'Sales', 'Open', 'Promo',
'StateHoliday',
       'SchoolHoliday', 'StoreType', 'Assortment', 'CompetitionDistance',
       'Promo2', 'Promo2SinceWeek', 'Promo2SinceYear', 'PromoInterval',
'Year',
       'number_of_year_promotion_till_now', 'month'],
      dtype='object')
```

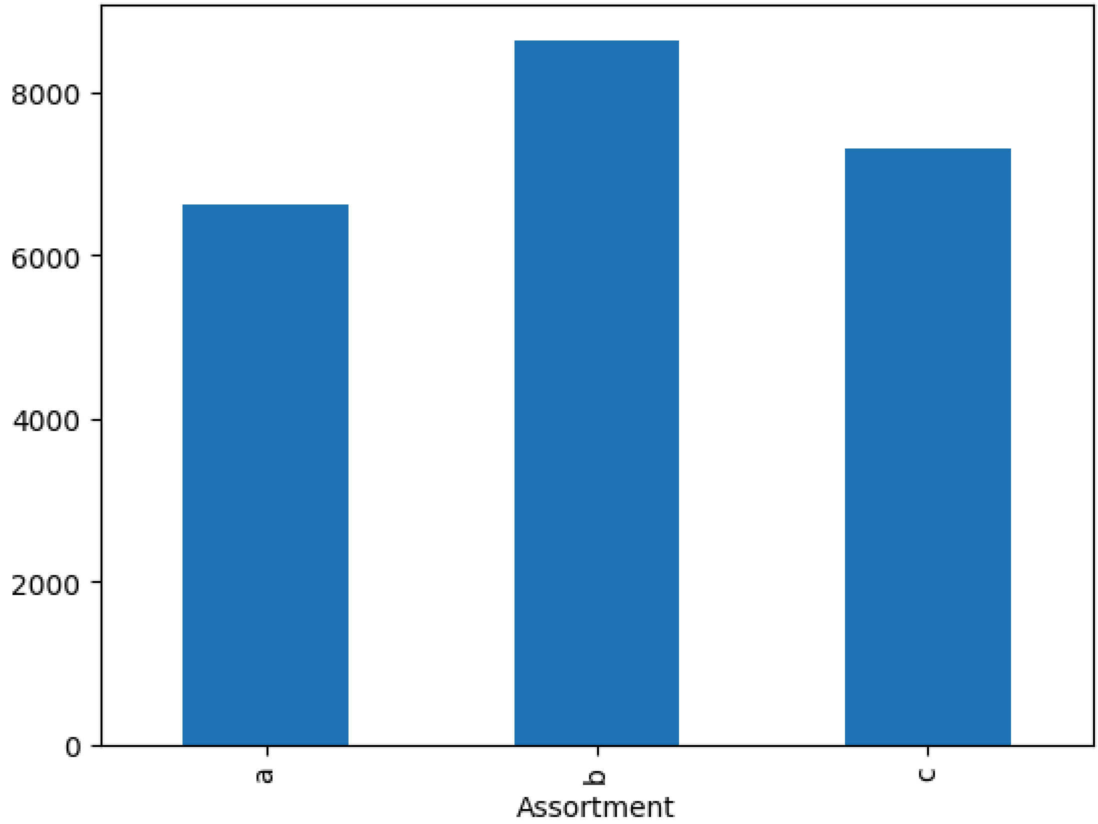
2. What is/are the insight(s) found from the chart?

with each year, sales is increasing

Chart - 8

```
# Chart - 8 visualization code
stores_df.groupby('Assortment')['Sales'].mean().plot(kind = 'bar')
```

```
<Axes: xlabel='Assortment'>
```



2. What is/are the insight(s) found from the chart?

assortment is some what related to sales

3. Will the gained insights help creating a positive business impact?

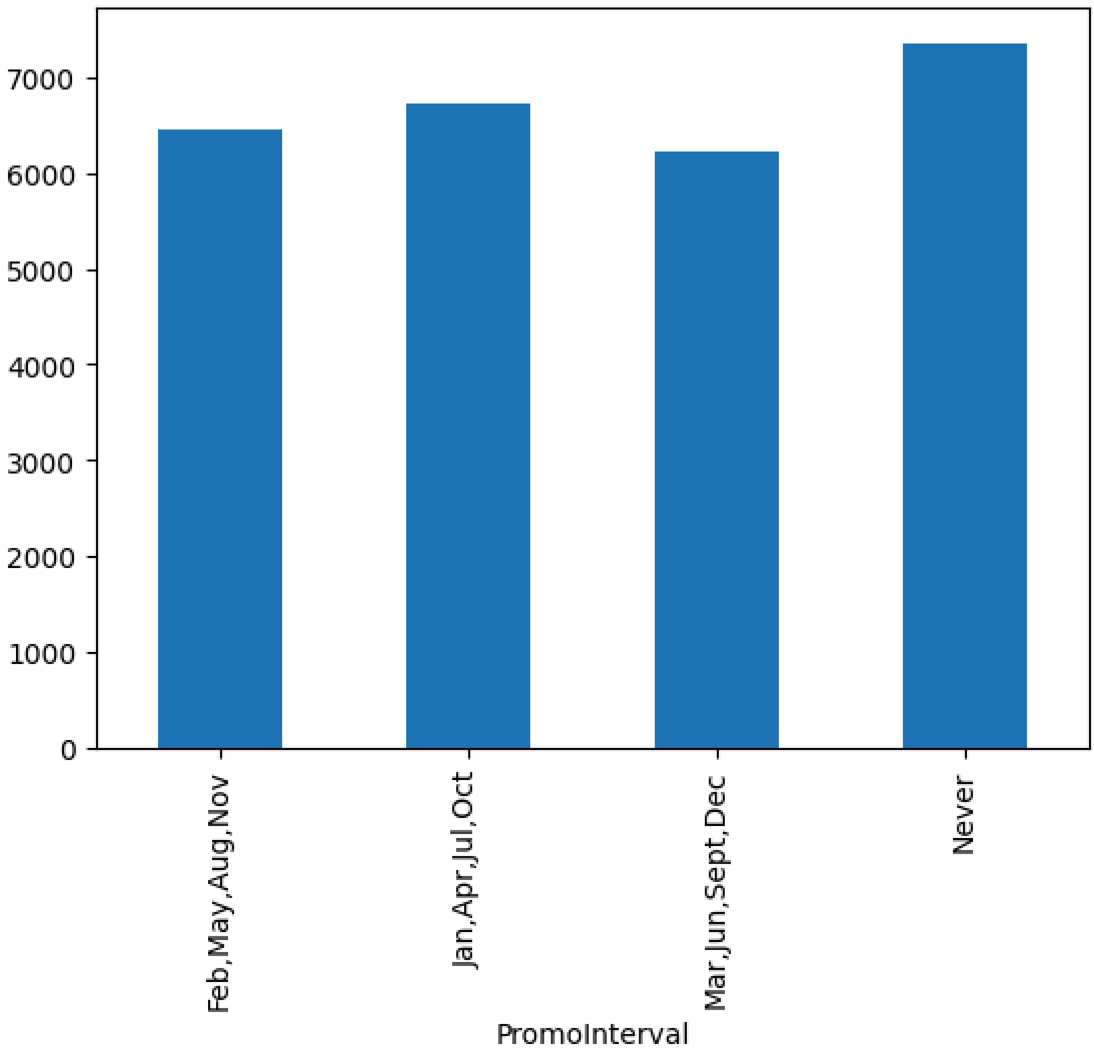
Are there any insights that lead to negative growth? Justify with specific reason.

Answer Here

Chart - 9

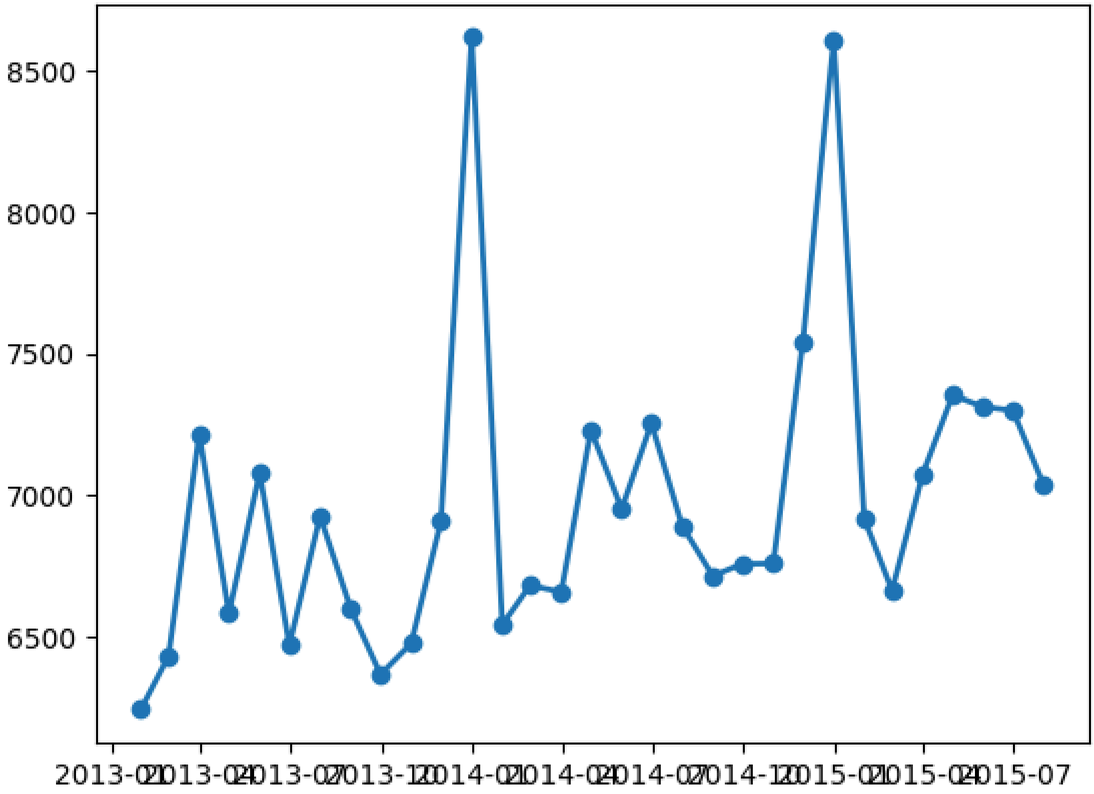
```
stores_df.groupby('PromoInterval')['Sales'].mean().plot(kind='bar')
```

```
<Axes: xlabel='PromoInterval'>
```




```
ts_df =stores_df.sort_values(by='Date')
ts_df = ts_df.set_index("Date")
monthly = ts_df["Sales"].resample("M").mean()
plt.plot(monthly.index, monthly.values, marker="o", linewidth=2)
plt.figure(figsize=(12,12))
```

```
<Figure size 1200x1200 with 0 Axes>
```



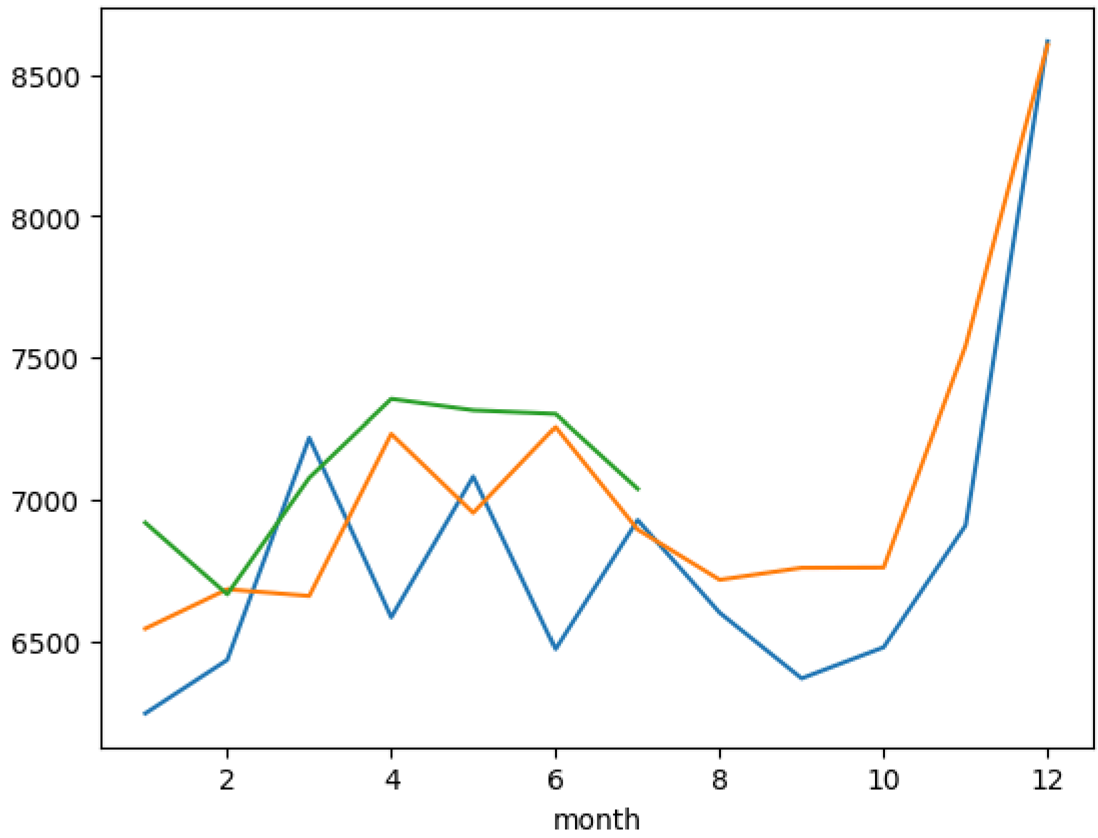

```
<Figure size 1200x1200 with 0 Axes>
```

There tends to be seasonal pattern as sales tends to rise at the end of the year

```
#feature creation for sample graph
t013_ = ts_df[ts_df['Year'] == 2013]
t014_ = ts_df[ts_df['Year'] == 2014]
t015_ = ts_df[ts_df['Year'] == 2015]
```

```
t013_.groupby('month')['Sales'].mean().plot(kind='line')
t014_.groupby('month')['Sales'].mean().plot(kind='line')
t015_.groupby('month')['Sales'].mean().plot(kind='line')
```

```
<Axes: xlabel='month'>
```



Each year sales tend to be better than year before

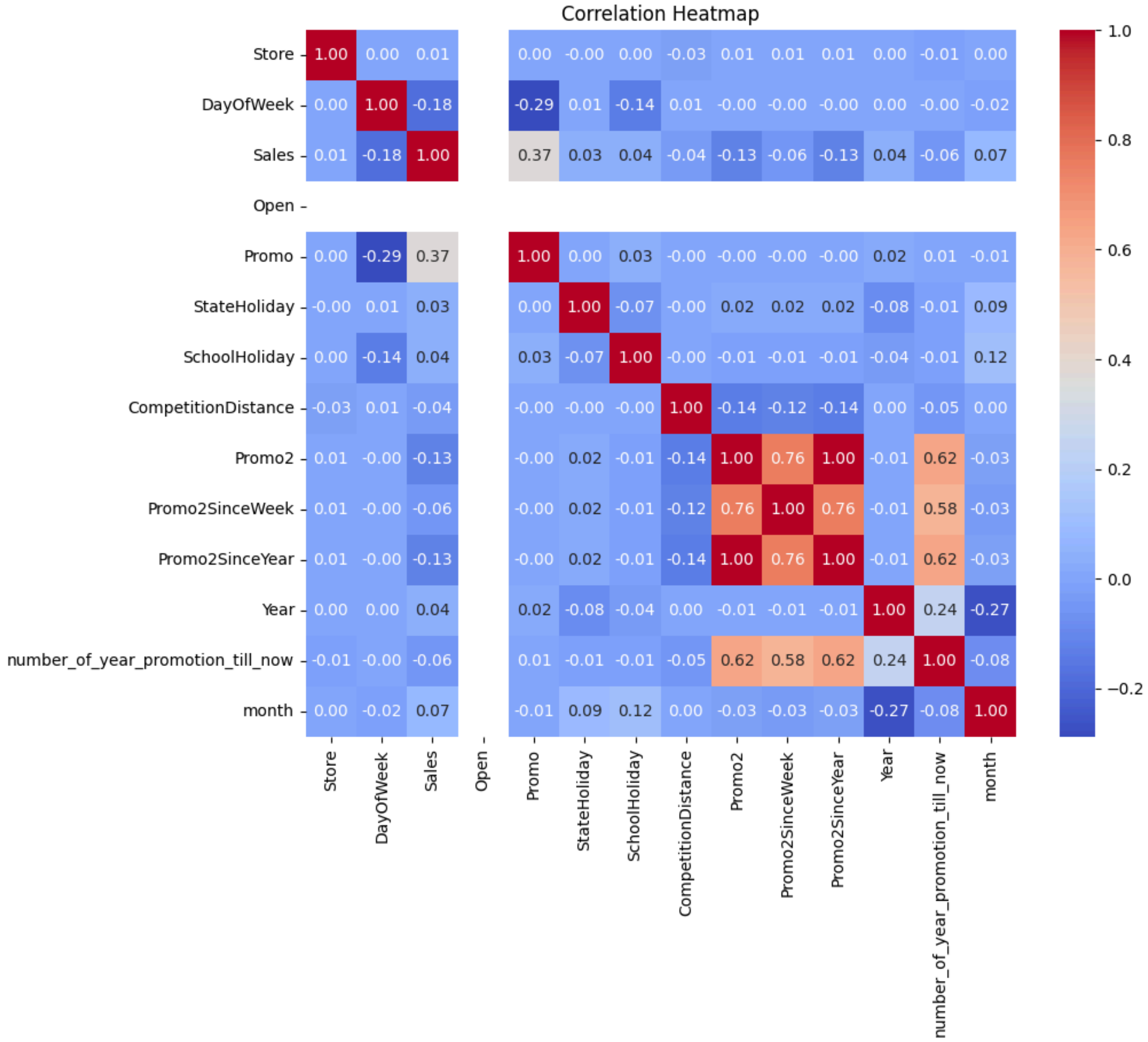
Chart - 10 - Correlation Heatmap

```
# Correlation Heatmap visualization code

numeric_df = stores_df.select_dtypes(include=['int64', 'float64','int32'])
```

```
corr_matrix = numeric_df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
```

```
Text(0.5, 1.0, 'Correlation Heatmap')
```



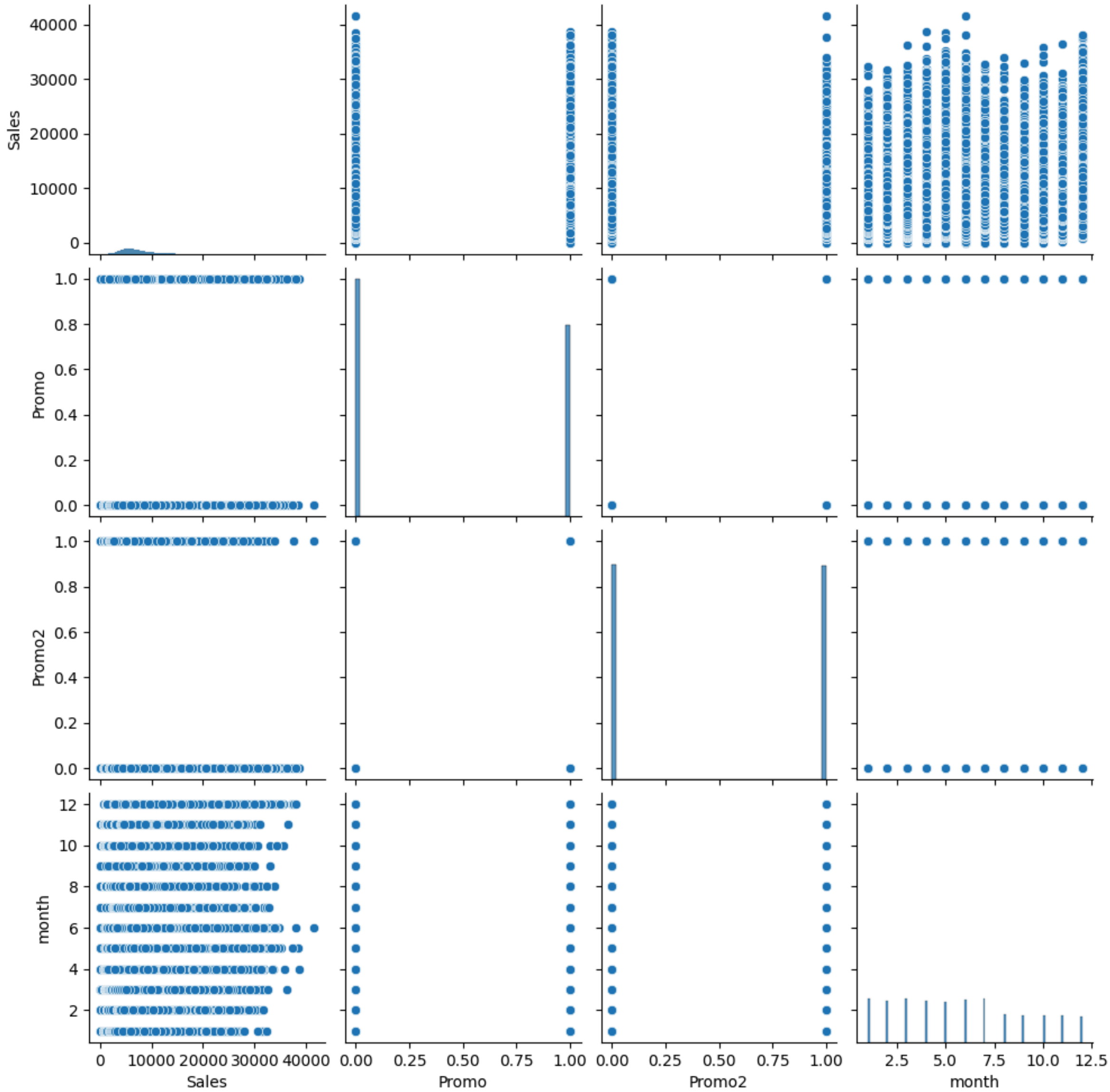
1. Why did you pick the specific chart?

Check the relationship between all the numeric feature

Chart - 11 - Pair Plot

```
# Pair Plot visualization code
cols = ['Sales','Promo','Promo2','month']
sns.pairplot(stores_df[cols])
```

<seaborn.axisgrid.PairGrid at 0x7d7a04e97b90>



1. Why did you pick the specific chart?

Check the relationship between all the numeric feature with Sales

5. Feature Engineering & Data Pre-processing

2. Handling Outliers

```
# Handling Outliers & Outlier treatments
stores_df.head()
stores_df['CompetitionDistance'].describe(percentiles=[0.25, 0.5, 0.75, 0.9€
```

	CompetitionDistance
count	842206.000000
mean	5457.979627
std	7809.437311
min	20.000000
25%	710.000000
50%	2320.000000
75%	6890.000000
90%	15720.000000
max	75860.000000

dtype: float64

```
#replaced all the ouliers of competiiton with 16000 as after 16000, sales di
stores_df['CompetitionDistance'] = stores_df['CompetitionDistance'].apply(la
```

```
stores_df.groupby(['Store','Assortment']).agg(mean_sales=('Sales', 'mean'),
    min_competition_distance=('CompetitionDistance', 'min')).sort_values(by=
```

		mean_sales	min_competition_distance
Store	**Assortment**		
315	**c**	4252.410714	16000.0
322	**a**	5021.172634	16000.0
318	**c**	7527.716667	16000.0
953	**a**	5323.455243	16000.0
963	**c**	10764.320924	16000.0
...	**...**	...	...
882	**a**	6607.352406	30.0
621	**a**	5916.224647	30.0
1008	**c**	5331.103316	30.0
988	**a**	4698.071429	30.0
516	**c**	5879.084724	20.0

1112 rows × 2 columns

```
stores_df.head()
```

	Store	DayOfWeek	Date	Sales	Open	Promo	StateHoliday	SchoolHoliday	S
0	1	5	2015-07-31	5263	1	1	1	1	
1	2	5	2015-07-31	6064	1	1	1	1	
2	3	5	2015-07-31	8314	1	1	1	1	
3	4	5	2015-07-31	13995	1	1	1	1	
4	5	5	2015-07-31	4822	1	1	1	1	

3. Categorical Encoding

One hot encoding for DayOfWeek, Month and Store Type

and manual encoding for assortment, store type and year

```
stores_df['Assortment'] = stores_df['Assortment'].astype(str).str.strip().st
```

```
new_df = stores_df.copy()
```

```
# Encode your categorical columns
new_df['Assortment'] = new_df['Assortment'].map({'a': 1,'b': 3,'c':2})
```

```
new_df['StoreType'] = new_df['StoreType'].apply(lambda x: 1 if x == 'b' else
```

```
 # drops first category
df_dummies = pd.get_dummies(new_df['DayOfWeek'], prefix='DayOfWeek',drop_fir
```

```
df = pd.concat([new_df, df_dummies], axis=1)
```

```
df.head(5)
```

	Store	DayOfWeek	Date	Sales	Open	Promo	StateHoliday	SchoolHoliday	S
0	1	5	2015-07-31	5263	1	1	1	1	
1	2	5	2015-07-31	6064	1	1	1	1	
2	3	5	2015-07-31	8314	1	1	1	1	
3	4	5	2015-07-31	13995	1	1	1	1	
4	5	5	2015-07-31	4822	1	1	1	1	

5 rows × 24 columns

```
df_dummies_2 = pd.get_dummies(new_df['month'], prefix='Month',drop_first=Tru
```

```
df = pd.concat([df,df_dummies_2],axis =1)
```

```
df['Year'] = df['Year'].map({2013:0,2014:1,2015:2})
```

```
df_dummies_3 = pd.get_dummies(df['Store'], prefix='Store',drop_first=True)
```

```
df = pd.concat([df,df_dummies_3],axis =1)
```

```
df.head(5)
```

	Store	DayOfWeek	Date	Sales	Open	Promo	StateHoliday	SchoolHoliday	S
0	1	5	2015-07-31	5263	1	1	1	1	
1	2	5	2015-07-31	6064	1	1	1	1	
2	3	5	2015-07-31	8314	1	1	1	1	
3	4	5	2015-07-31	13995	1	1	1	1	
4	5	5	2015-07-31	4822	1	1	1	1	

5 rows × 1146 columns

4. Feature Manipulation & Selection

1. Feature Manipulation

```
#cummulative sum feature for sample analysis, turns out there wasn't much im
df['cumm_sum'] = df.sort_values(['Store', 'Date']).groupby('Store')['Sales']
```

2. Feature Selection

```
df.columns
```

```
Index(['Store', 'DayOfWeek', 'Date', 'Sales', 'Open', 'Promo',
'StateHoliday',
       'SchoolHoliday', 'StoreType', 'Assortment',
       ...
       'Store_1107', 'Store_1108', 'Store_1109', 'Store_1110', 'Store_1111',
       'Store_1112', 'Store_1113', 'Store_1114', 'Store_1115', 'cumm_sum'],
      dtype='object', length=1147)
```

```
scale_df = df.copy()
```

```
#dropped unnecessary columns
scale_df.drop(['Store','Promo2SinceYear','Promo2SinceWeek','PromoInterval','
```

```
scale_df.columns
```

```
Index(['Date', 'Sales', 'Promo', 'StoreType', 'Assortment',
       'CompetitionDistance', 'Promo2', 'Year', 'month', 'DayOfWeek_2',
       ...
       'Store_1107', 'Store_1108', 'Store_1109', 'Store_1110', 'Store_1111',
```

```
       'Store_1112', 'Store_1113', 'Store_1114', 'Store_1115', 'cumm_sum'],
      dtype='object', length=1138)
```

Which all features you found important and why?

Storetype showed relationship with sales, storetype b has more sales than others, assortment showed relationship with sales, Competition distance showed relationship with sales, Year showed relationship with sales,

6. Data Scaling

```
# Scaling your data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
scale_df.head()
```

	Date	Sales	Promo	StoreType	Assortment	CompetitionDistance	Promo2	Ye
0	2015-07-31	5263	1	0	1	1270.0	0	
1	2015-07-31	6064	1	0	1	570.0	1	
2	2015-07-31	8314	1	0	1	14130.0	1	
3	2015-07-31	13995	1	0	2	620.0	0	
4	2015-07-31	4822	1	0	1	16000.0	0	

5 rows × 1138 columns

Scaling non-binary variables

```
numeric_cols = ['CompetitionDistance','cumm_sum','Assortment','Year']
scaled_cols = ['scaled_CompetitionDistance','scaled_cumm_sum','scaled_assort

scale_df[scaled_cols] = scaler.fit_transform(scale_df[numeric_cols])
```

```
final_df = scale_df.copy()
```

```
final_df.columns
```

```
Index(['Date', 'Sales', 'Promo', 'StoreType', 'Assortment',
       'CompetitionDistance', 'Promo2', 'Year', 'month', 'DayOfWeek_2',
       ...
       'Store_1111', 'Store_1112', 'Store_1113', 'Store_1114', 'Store_1115',
       'cumm_sum', 'scaled_CompetitionDistance', 'scaled_cumm_sum',
       'scaled_assortment', 'scaled_year'],
      dtype='object', length=1142)
```

```
final_df.drop(['CompetitionDistance','cumm_sum','Assortment','Year','month',
```

Which method have you used to scale you data and why?

8. Data Splitting

We need to predict the future value of the target variable('Sales'), so we can just go for random train and test split, so I have sorted the data by Date and then did the split on the database

```
# Split your data to train and test. Choose Splitting ratio wisely.
from sklearn.model_selection import train_test_split
final_df.sort_values(by='Date', inplace = True)
```

```
final_df.drop(['Date'], axis = 1 , inplace = True)
```

```
final_df['Sales'].describe()
```

	Sales
count	842206.000000
mean	6959.338682
std	3104.460543
min	0.000000
25%	4864.000000
50%	6373.000000
75%	8363.000000
max	41551.000000

dtype: float64

```
split = int(len(df) * 0.7)

train_df = final_df.iloc[:split]
test_df = final_df.iloc[split:]
```

```
x_test = test_df.drop(['Sales'], axis = 1)
y_test = test_df['Sales']
```

```
x_train = train_df.drop(['Sales'], axis = 1)
y_train = train_df['Sales']
```

What data splitting ratio have you used and why?

70:30 as it is better and standard split, also it will be better to split the 2015 data and predict the 2025 data with the help of previous data

6. ML Model Implementation

ML Model - 1

```
from sklearn.model_selection import GridSearchCV,TimeSeriesSplit
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline
```

```
# ML Model - 1 Implementation
#fit the model
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
#predict mdoel
y_pred = model.predict(x_test)
```

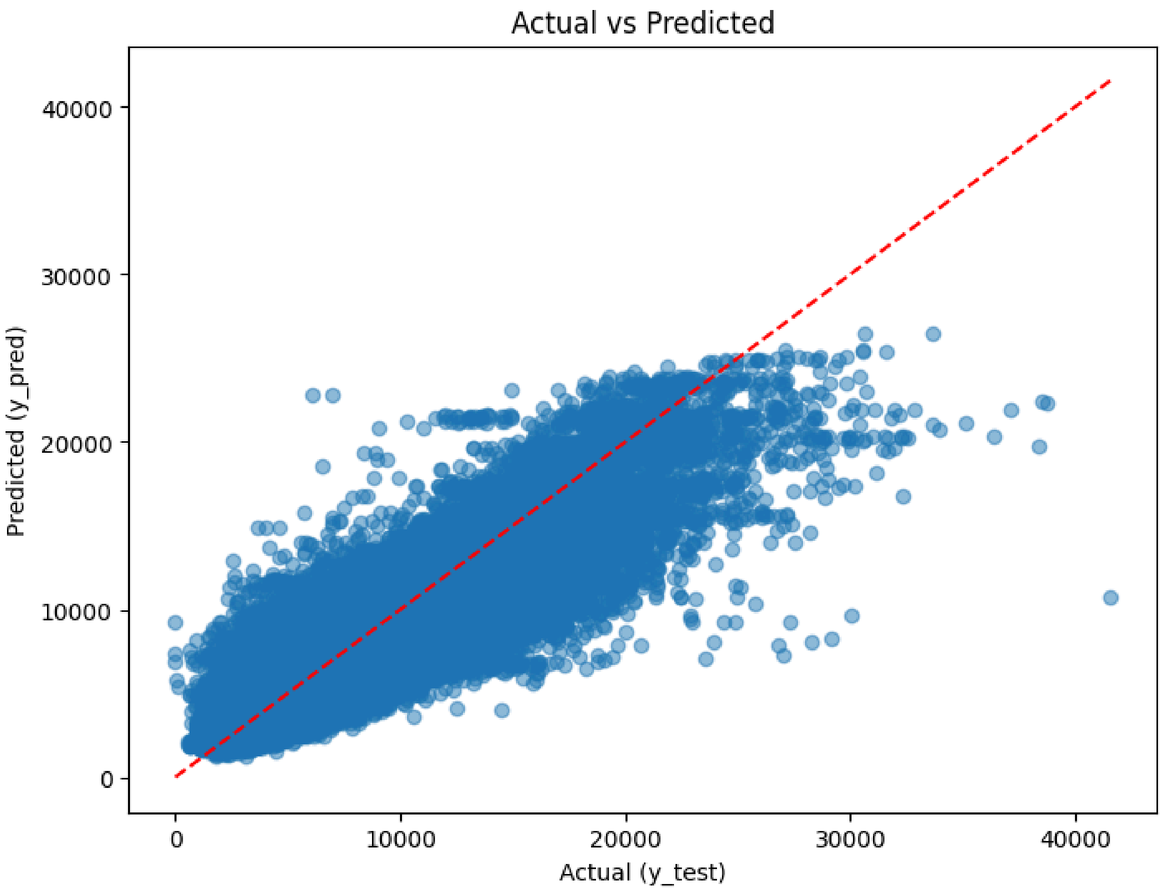
```
#check the error
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(mse)
print(r2)
```

```
2352947.1511876606
0.7644203308952344
```

```
#visualize the predict and actual value
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()], 'r--')  # perfect prediction line
plt.xlabel("Actual (y_test)")
plt.ylabel("Predicted (y_pred)")
plt.title("Actual vs Predicted")
plt.show()
```



1. Explain the ML Model used and it's performance using Evaluation metric Score Chart.

I have used linear regression model, as i was able to see some linear relationship in the model and r2 for test data showed me accuracy of 76.5 that is pretty good for a model

2. Cross- Validation & Hyperparameter Tuning

I tried to apply GridSearchCV, but it turns out that, dimension of my dataset is too huge for me, to run gridsearch on my pc or even TPU of google colab, i want to tried the polynomial features but turns out my computer doesn't have enough RAM. My guess is applying polynomial feature might increase the model accuracy

ML Model - 2

```
from sklearn.tree import DecisionTreeRegressor
```

```
#model implementation and fit
model_2 = DecisionTreeRegressor(max_depth=100,         # controls tree depth
    min_samples_split=5,  # minimum samples to split a node
    random_state=42)
model_2.fit(x_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=100, min_samples_split=5, random_state=42)
```

```
#predict the y value of test
y_pred = model_2.predict(x_test)
```

```
#predict the y value of train
y_train_pred = model_2.predict(x_train)
```

```
#training data prediction
mse = mean_squared_error(y_train, y_train_pred)
r2 = r2_score(y_train, y_train_pred)

print(f"MSE: {mse:.2f}")
print(f"R²: {r2:.3f}")
```

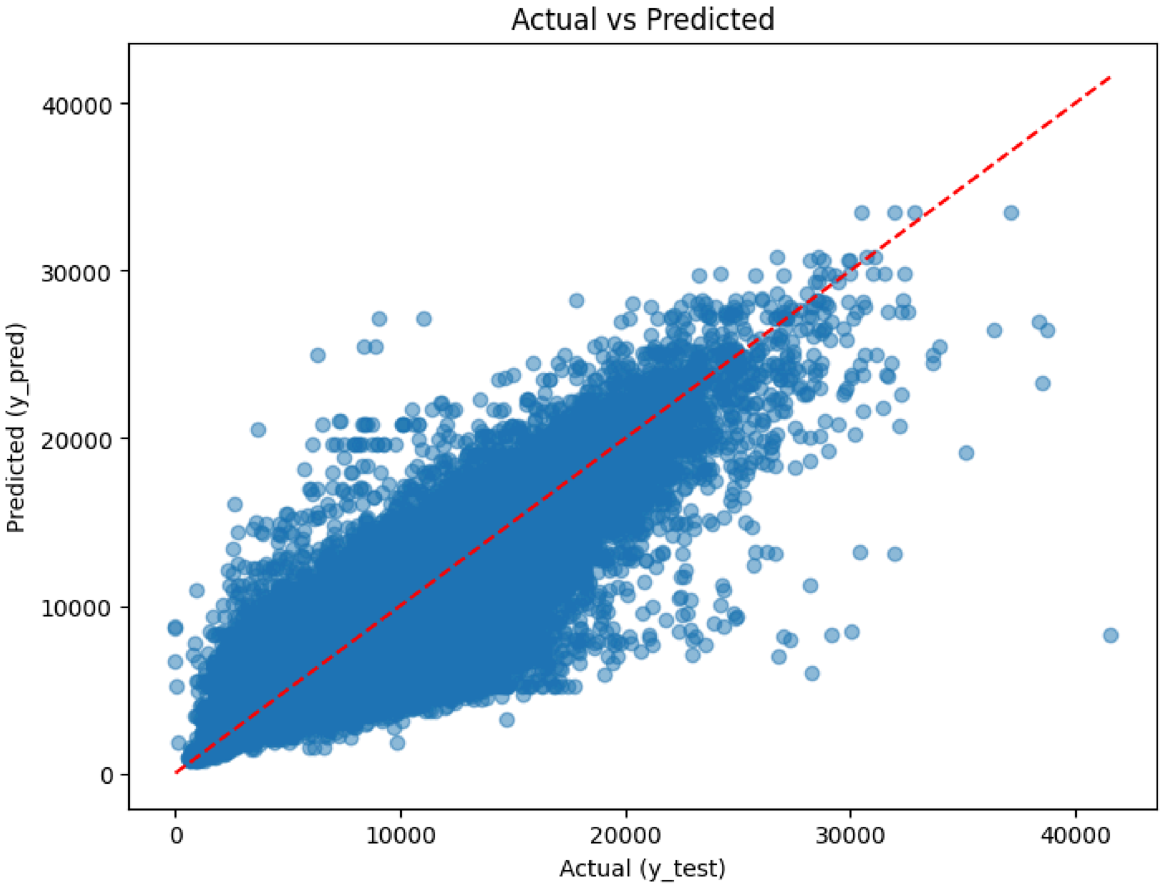
```
MSE: 971272.07
R²: 0.897
```

```
#prediction on test data
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MSE: {mse:.2f}")
print(f"R²: {r2:.3f}")
```

```
MSE: 1968046.43
R²: 0.803
```

```
#visulaize the predict y and actual y for test dataset
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()], 'r--')  # perfect prediction line
plt.xlabel("Actual (y_test)")
plt.ylabel("Predicted (y_pred)")
plt.title("Actual vs Predicted")
plt.show()
```




```
#converting predicting value in a way that machine understands
import pandas as pd
```