Product demand prediction with machine learning

Name:Swathi.T

Reg.no:912421104048

Phase 2 submission document

INTRODUCTION:

A product company plans to offer discounts on its product during the upcoming holiday season. The company wants to find the price at which its product can be a better deal compared to its competitors. For this task, the company provided a dataset of past changes in sales based on price changes. You need to train a model that can predict the demand for the product in the market with different price segments.

Data source:

DatasetLink:
https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning

ID	Store ID	Total Price	Base Price	Units Sold
1	8091	99.0375	111.8625	20
2	8091	99.0375	99.0375	28
3	8091	133.95	133.95	19
4	8091	133.95	133.95	44

5	8091	141.075	141.075	52
9	8091	227.2875	227.2875	18
10	8091	327.0375	327.0375	47
13	8091	210.9	210.9	50
14	8091	190.2375	234.4125	82
17	8095	99.0375	99.0375	99
18	8095	97.6125	97.6125	120
19	8095	98.325	98.325	40
22	8095	133.2375	133.2375	68
23	8095	133.95	133.95	87
24	8095	139.65	139.65	186

27	8095	236.55	280.0125	54
28	8095	214.4625	214.4625	74
29	8095	266.475	296.4	102
30	8095	173.85	192.375	214
31	8095	205.9125	205.9125	28
32	8095	205.9125	205.9125	7
33	8095	248.6625	248.6625	48
34	8095	200.925	200.925	78
35	8095	190.2375	240.825	57
37	8095	427.5	448.1625	50
38	8095	429.6375	458.1375	62

39	8095	177.4125	177.4125	22
42	8094	87.6375	87.6375	109
43	8094	88.35	88.35	133
44	8094	85.5	85.5	11
45	8094	128.25	180.975	9
47	8094	127.5375	127.5375	19
48	8094	123.975	123.975	33
49	8094	139.65	164.5875	49
50	8094	235.8375	235.8375	32
51	8094	234.4125	234.4125	47
52	8094	235.125	235.125	27

53	8094	227.2875	227.2875	69
54	8094	312.7875	312.7875	49

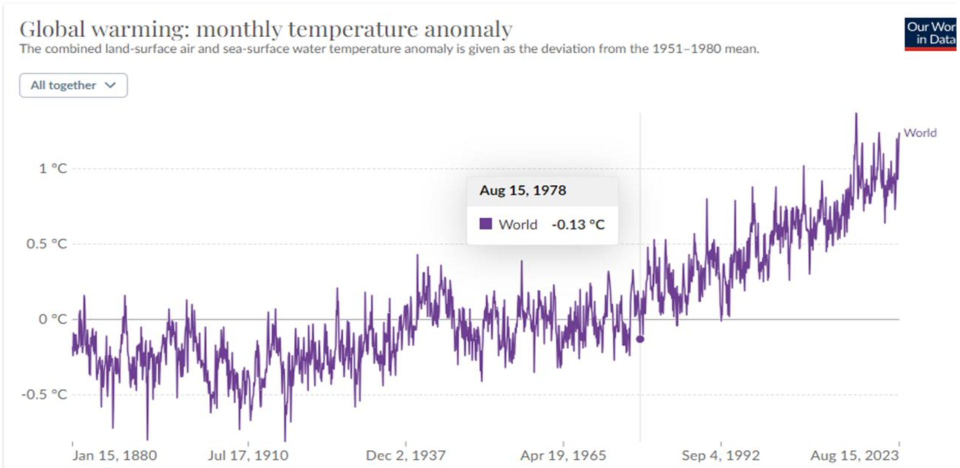
Demand forecasting:

Demand forecasting is the process of predicting what customers' appetite will be for existing products or services, determining what adjustment you should make and what new offerings will spark interest. But predicting what people will want, in what quantities and when is no small feat.

Design thinking:

- Data collection
- Data preparation
- Feature engineering
- Model selection
- Model training
- Evaluation

Data about climate change



Data preprocessing:

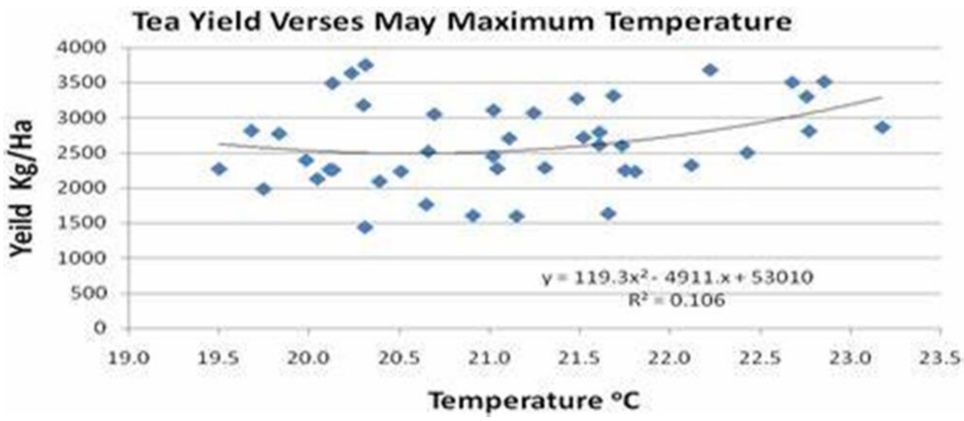
Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance,[1] and is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data collection methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, and missing values, amongst other issues.

Methods:

- Clean and preprocess the data
- Handle missing values

- Convert categorical features into numerical representation

Linear regression:



ARIMA

ARIMA is a method for forecasting or predicting future outcomes based on a historical time series.

Data wrangling technique:

- Merging several data sources into one data-set for analysis.
- Identifying gaps or empty cells in data and either filling or removing them.
- Deleting irrelevant or unnecessary data.
- Identifying severe outliers in data and either explaining the inconsistencies or deleting them to facilitate analysis.

EXAMPLE PROGRAM:

```
import pandas as pd
import numpy as np
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor

data = 
pd.read_csv("https://raw.githubusercontent.com/amankha
rwal/Website-data/master/demand.csv")
```

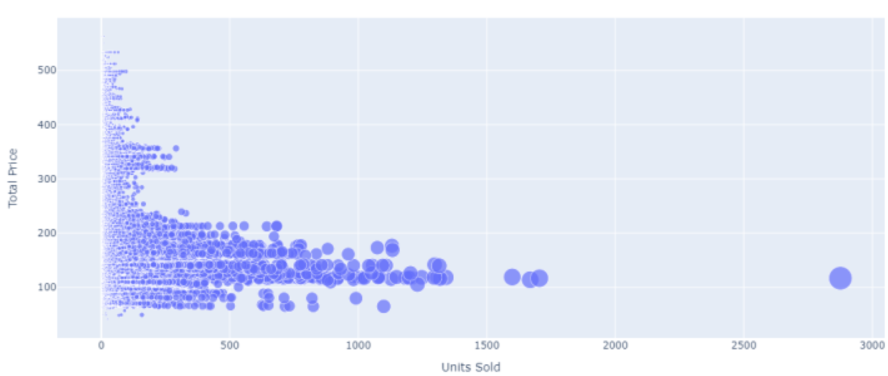
data.head()

OUTPUT:

	ID	Store ID	Total Price	Base Price	Units Sold
0	1	8091	99.0375	111.8625	20
1	2	8091	99.0375	99.0375	28
2	3	8091	133.9500	133.9500	19
3	4	8091	133.9500	133.9500	44
4	5	8091	141.0750	141.0750	52

```
fig = px.scatter(data, x="Units Sold", y="Total Price",
                 size='Units Sold')
fig.show()
```

OUTPUT:



features = np.array([[133.00, 140.00]])

model.predict(features)

OUTPUT:

array([27.])

Prophet:

make an in-sample forecast

from pandas import read_csv

from pandas import to_datetime

from pandas import DataFrame

from fbprophet import Prophet

from matplotlib import pyplot

load data

path = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/monthly-car-sales.csv'

df = read_csv(path, header=0)

prepare expected column names

df.columns = ['ds', 'y']

```
df['ds']= to_datetime(df['ds'])
# define the model

model = Prophet()
# fit the model

model.fit(df)
# define the period for which we want a prediction

future = list()
for i in range(1, 13):
date = '1968-%02d' % i
future.append([date])
future = DataFrame(future)
future.columns = ['ds']
future['ds']= to_datetime(future['ds'])
# use the model to make a forecast

forecast = model.predict(future)
# summarize the forecast

print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].head())
```

plot forecast

model.plot(forecast)

pyplot.show()

Output:

	ds	yhat	yhat_lower	yhat_upper
0	1968-01-01	14364.866157	12816.266184	15956.555409
1	1968-02-01	14940.687225	13299.473640	16463.811658
2	1968-03-01	20858.282598	19439.403787	22345.747821
3	1968-04-01	22893.610396	21417.399440	24454.642588
4	1968-05-01	24212.079727	22667.146433	25816.191457

What is Mean Squared Error or MSE

- The Mean Absolute Error is the squared mean of the difference between the actual values and predictable values.

<u>PROGRAM:</u>

Product Demand Prediction:

Import pandas as pd

Import numpy as np

Import plotly.express as px

Import seaborn as sns

Import matplotlib.pyplot as plt

From sklearn.model_selection import train_test_split

From sklearn.tree import DecisionTreeRegressor

Data=pd.read_csv(“C:\Users\mabir\AppData\Local\Microsoft\Windows\INetCache\IE\AHLGJQP8\archive[1].zip “)

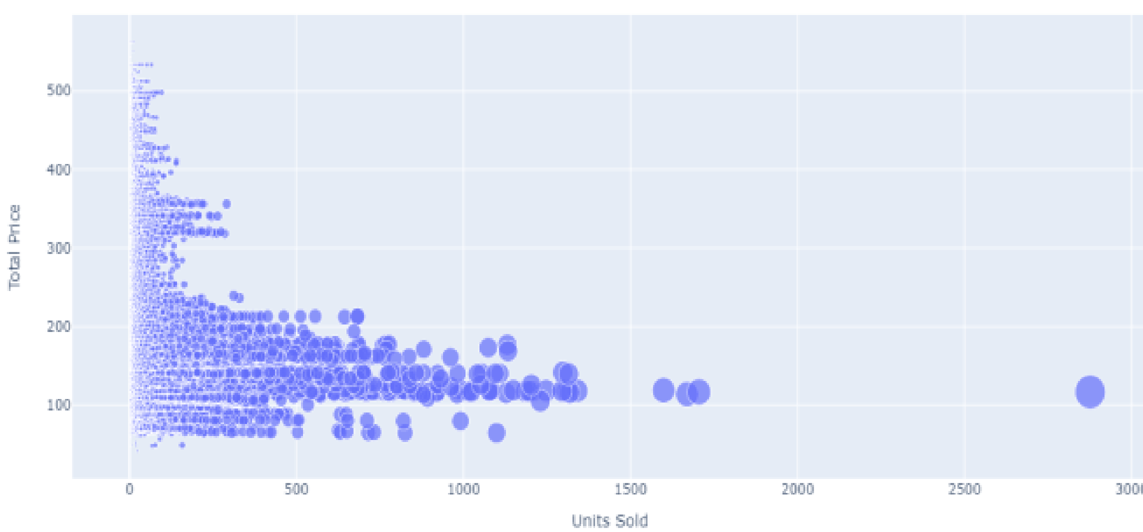
Data.head()

<u>Relationship between price and demand for the product:</u>

Fig = px.scatter(data, x=”Units Sold”, y=”Total Price”,
                Size=’Units Sold’)

Fig.show()

<u>Output:</u>



Correlation between the features of the dataset:

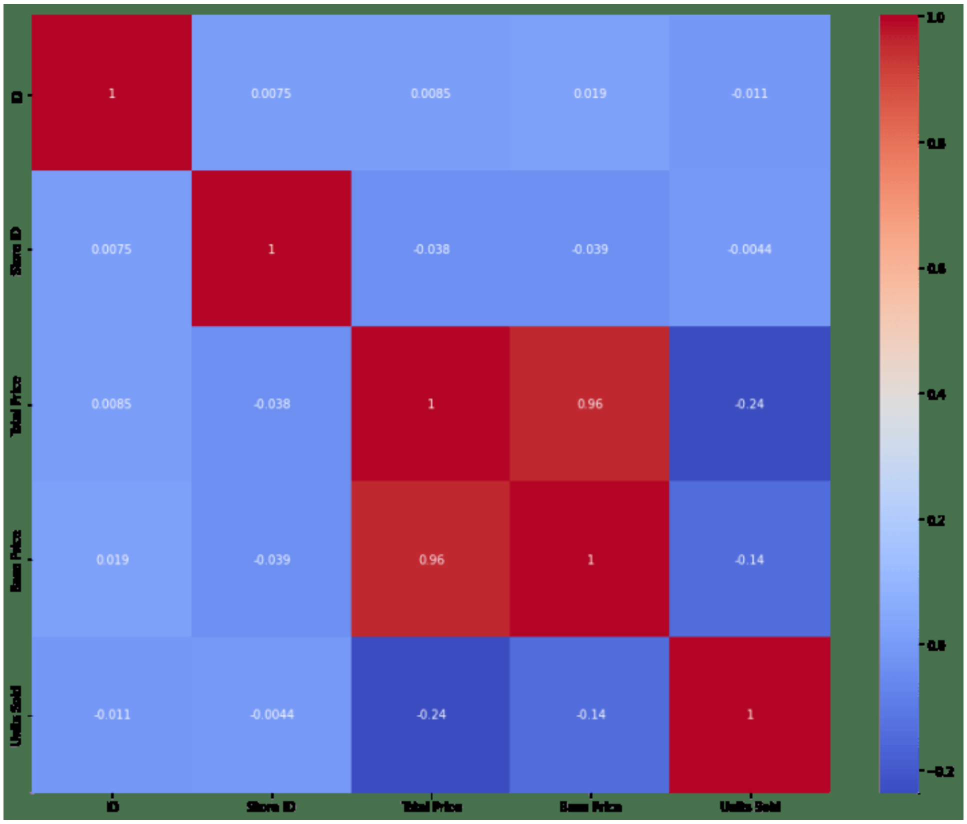
Print(data.corr())

Output:

	ID	Store ID	Total Price	Base Price	Units Sold
ID	1.000000	0.007464	0.008473	0.018932	-0.010616
Store ID	0.007464	1.000000	-0.038315	-0.038848	-0.004372
Total Price	0.008473	-0.038315	1.000000	0.958885	-0.235625
Base Price	0.018932	-0.038848	0.958885	1.000000	-0.140032
Units Sold	-0.010616	-0.004372	-0.235625	-0.140032	1.000000

```
Correlations = data.corr(method=’pearson’)
Plt.figure(figsize=(15, 12))
Sns.heatmap(correlations, cmap=”coolwarm”, annot=True)
Plt.show()
```

<u>Output:</u>




```
# fit an ARIMA model and plot residual errors
From pandas import datetime
From pandas import read_csv
From pandas import DataFrame
From statsmodels.tsa.arima.model import ARIMA
From matplotlib import pyplot
# load dataset
Def parser(x):
Return datetime.strptime('190'+x, '%Y-%m')
Series = read_csv('shampoo-sales.csv', header=0, index_col=0, parse_dates=True, squeeze=True, date_parser=parser)
Series.index = series.index.to_period('M')
```

```
# fit model
Model = ARIMA(series, order=(5,1,0))
Model_fit = model.fit()
# summary of fit model
Print(model_fit.summary())
# line plot of residuals
Residuals = DataFrame(model_fit.resid)
Residuals.plot()
Pyplot.show()
# density plot of residuals
Residuals.plot(kind='kde')
Pyplot.show()
# summary stats of residuals
Print(residuals.describe())
```

Output:

SARIMAX Results

Dep. Variable: Sales No. Observations:
36

Model: ARIMA(5, 1, 0) Log Likelihood -
198.485

Date: Thu, 10 Dec 2020 AIC
408.969

Time: 09:15:01 BIC 418.301

Sample: 01-31-1901 HQIC
412.191

- 12-31-1903

Covariance Type: opg

Coef std err z P>|z| [0.025 0.975]

Ar.L1 -0.9014 0.247 -3.647 0.000 -1.386 -
0.417

Ar.L2 -0.2284 0.268 -0.851 0.395 -0.754
0.298

Ar.L3	0.0747	0.291	0.256	0.798	-0.497	0.646
Ar.L4	0.2519	0.340	0.742	0.458	-0.414	0.918
Ar.L5	0.3344	0.210	1.593	0.111	-0.077	0.746
Sigma2	4728.9608	1316.021	3.593	0.000	2149.607	7308.314

Ljung-Box (L1) (Q):	0.61	Jarque-Bera (JB):	0.96
Prob(Q):	0.44	Prob(JB):	0.62
Heteroskedasticity (H):	1.07	Skew:	0.28
Prob(H) (two-sided):	0.90	Kurtosis:	2.41

Prophet:

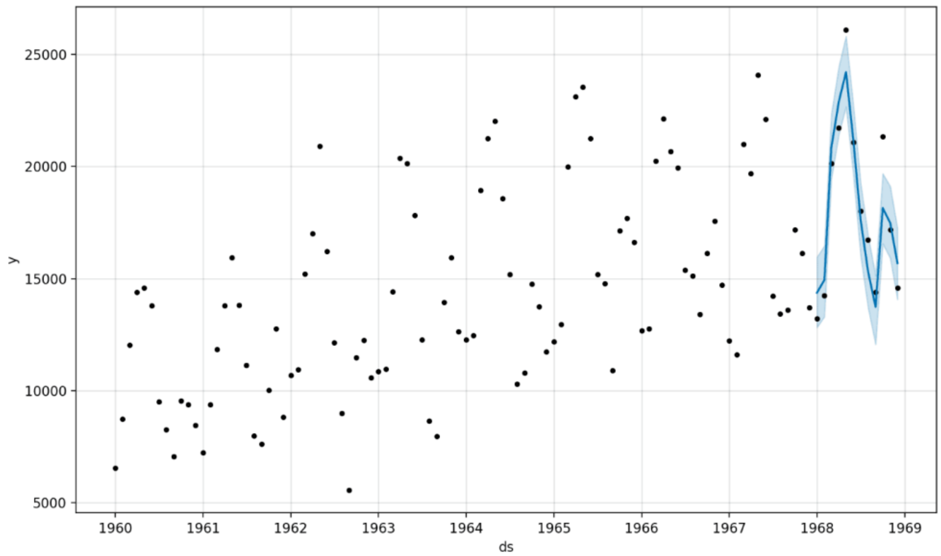
```
# make an in-sample forecast
from pandas import read_csv
from pandas import to_datetime
from pandas import DataFrame
from fbprophet import Prophet
from matplotlib import pyplot
# load data
path = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/monthly-car-sales.csv'
df = read_csv(path, header=0)
# prepare expected column names
df.columns = ['ds', 'y']
df['ds']= to_datetime(df['ds'])
# define the model
model = Prophet()
# fit the model
model.fit(df)
# define the period for which we want a prediction
future = list()
```

```
for i in range(1, 13):
date = '1968-%02d' % i
future.append([date])
future = DataFrame(future)
future.columns = ['ds']
future['ds']= to_datetime(future['ds'])
# use the model to make a forecast
forecast = model.predict(future)
# summarize the forecast
print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].head())
# plot forecast
model.plot(forecast)
pyplot.show()
```

Running the example forecasts the last 12 months of the dataset.

<u>OUTPUT:</u>

```
          ds          yhat    yhat_lower    yhat_upper
0 1968-01-01  14364.866157  12816.266184  15956.555409
1 1968-02-01  14940.687225  13299.473640  16463.811658
2 1968-03-01  20858.282598  19439.403787  22345.747821
3 1968-04-01  22893.610396  21417.399440  24454.642588
4 1968-05-01  24212.079727  22667.146433  25816.191457
```



Conclusion:

*** In this phase-2 project conclusion, we will summarize the key findings and insights from the incorporating time series techniques .we will reiterate the impact of these time series techniques .These techniques provide valuable insights into patterns, seasonality, and trends within the data, enabling more accurate predictions and forecasts.**