SALES PREDICTION WITH MACHINE LEARNING USING PYTHON

What is sales prediction?

A sales prediction or forecast is a projected measure of how a market will respond to a company's go-to-market efforts.

A go-to-market strategy is a plan that details how an organisation can engage with customers to convince them to buy their product or service and to gain a competitive advantage.

Why is sales forecasting important?

An accurate sales forecast helps in

1. Improving decision-making about the future
2. Developing budget for hiring
3. Attaining credibility in the market

Steps involved:

1. Start by importing the necessary libraries
2. Import the data set
3. Train the model

The following libraries have been used in the code:

1. Pandas

- It is an open-source python package used to work with data.
- It provides flexible data structures to make working with data easier.

2. NumPy

- It is used perform myriad mathematical operations on arrays.

3. Matplotlib

- It is an open-source library that helps in graphical plotting and data visualisation
- It works efficiently with data frames and arrays

4. Seaborn

- It makes statistical graphics in python and is comfortable in handling pandas data frames.

5. **Sklearn**
    - ♦ It is a library that provides all the tools that are required for machine learning and statistical modelling.

- **The dataset:**

  link

- **Algorithms:**

Linear Regression:

Definition:
It is a machine learning algorithm based on supervise learning, that uses a dependent variable to predict future outcomes based on independent variable(s).

Mathematical form:

$$F(x) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$$

Where θ_0, θ_1, …., θ_n are parameters.

How to find out the parameters?

We define an error function.

Error function (E):

For one training sample,

$$E = (y^i - F(x^i))^2$$

Total error:

$$E_t = {}^m\sum_{i=1} (y^i - F(x^i))^2$$

Where m is the total number of training samples.

Now, E_t needs to be minimised with respect to the parameters using gradient descent.

Gradient descent

Definition:
It is an optimization algorithm, which trains machine learning models by means of minimising errors between actual and expected results.

Objective function:

$$E_t = {}^{m}\sum_{i=1} (y^i - F(x^i))^2$$

Objectives:

- Find the minimum value of the function
- Find the parameters for which the minimum occurs

Gradient descent on linear regression:

- **Linear regression function:**

$$F(x) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$$

- **Error function:**

$$E_t = {}^{m}\sum_{i=1} (y^i - F(x^i))^2$$

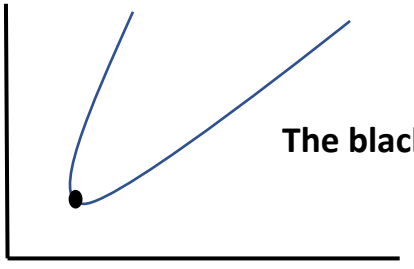
Gradient descent update rule:

$$(\theta_i)^{t+1} = (\theta_i)^t - r * \delta E / \delta \theta_i$$

Algorithm:

1. Randomly set values to the parameters ($\theta_0, \theta_1, ..., \theta_n$)
2. Repeat until convergence

F(x) vs θ



The black dot is where we want to reach

Code:

Step 1: Import the necessary libraries

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
```

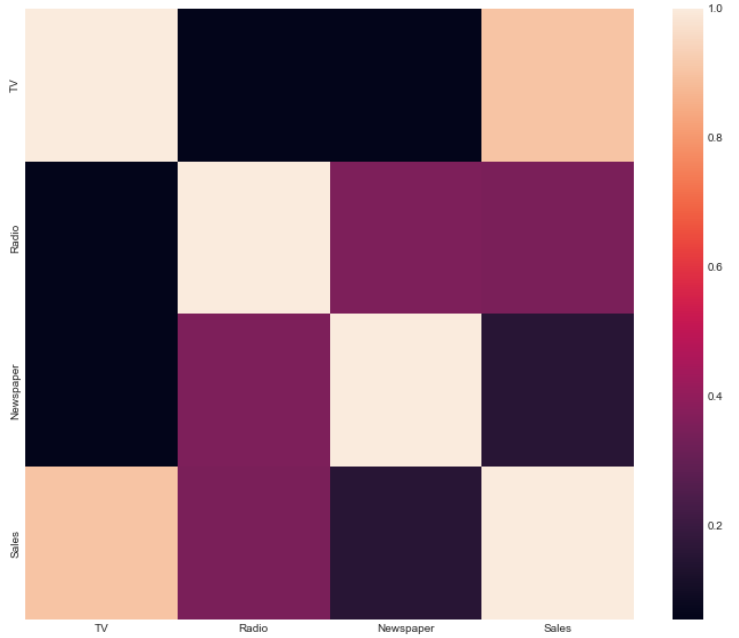
Step 2: Import the dataset and print the values

```
In [2]: data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/advertising.csv")
        print(data.head())
```

```
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
```

Step 3: Data Visualisation

```
In [3]: plt.style.use('seaborn-whitegrid')
        plt.figure(figsize=(12, 10))
        sns.heatmap(data.corr())
        plt.show()
```



Step 4: Splitting the data and training it

```
In [4]:
x = np.array(data.drop(["Sales"], 1))
y = np.array(data["Sales"])
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(xtrain, ytrain)
ypred = model.predict(xtest)
```

Step 5: predicting the data

```
In [5]: data = pd.DataFrame(data={"Predicted Sales": ypred.flatten()})
print(data)
```

Step 6: Printing the data

	Predicted Sales
0	17.034772
1	20.409740
2	23.723989
3	9.272785
4	21.682719
5	12.569402
6	21.081195
7	8.690350
8	17.237013
9	16.666575
10	8.923965
11	8.481734
12	18.207512
13	8.067507
14	12.645510
15	14.931628
16	8.128146
17	17.898766
18	11.008806
19	20.478328
20	20.806318
21	12.598833
22	10.905183
23	22.388548
24	9.417961
25	7.925067
26	20.839085
27	13.815209
28	10.770809
29	7.926825
30	15.959474
31	10.634909
32	20.802920
33	10.434342
34	21.578475
35	21.183645
36	12.128218
37	22.809533
38	12.609928
39	6.464413

References:

https://www.anaplan.com/blog/sales-forecasting-guide/

https://www.gartner.com/en/sales/glossary/go-to-market-gtm-strategy

https://www.geeksforgeeks.org/ml-linear-regression/

https://thecleverprogrammer.com/2020/11/20/linear-regression-with-python/

https://www.ibm.com/cloud/learn/gradient-descent

https://www.javatpoint.com/gradient-descent-in-machine-learning