INTRODUCTION TO MACHINE LEARNING

LAB ASSIGNMENT – 3

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

Feature Engineering Techniques for Machine Learning -Deconstructing the 'art'

```
import pandas as pd
data={'Candy Variety':['Chocolate Hearts','Sour Jelly','Candy Canes','Sour Jelly','Fruit Drops'],
      'Date and Time':['09-02-2020 14:05','24-10-2020 18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
      'Day':['Sunday','Saturday','Friday','Sunday','Sunday'], 'Length':[3, 3.5, 3.5, 3.5, 5], 'Breadth':[2,2,2.5,2,3],
      'Price':[7.5, 7.6, 8, 7.6, 9]}
df = pd.DataFrame(data)
df['Date and Time'] = pd.to_datetime(df['Date and Time'], format="%d-%m-%Y %H:%M")
df
```

	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	2020-02-09 14:05:00	Sunday	3.0	2.0	7.5
1	Sour Jelly	2020-10-24 18:00:00	Saturday	3.5	2.0	7.6
2	Candy Canes	2020-12-18 20:13:00	Friday	3.5	2.5	8.0
3	Sour Jelly	2020-10-25 10:00:00	Sunday	3.5	2.0	7.6
4	Fruit Drops	2020-10-18 15:46:00	Sunday	5.0	3.0	9.0

```
df['Date']=df['Date and Time'].dt.date
df[['Candy Variety','Date']]
```

	Candy Variety	Date
0	Chocolate Hearts	2020-02-09
1	Sour Jelly	2020-10-24
2	Candy Canes	2020-12-18
3	Sour Jelly	2020-10-25
4	Fruit Drops	2020-10-18

```
import numpy as np
df['Weekend'] = np.where(df['Day'].isin(['Saturday', 'Sunday']), 1, 0)
df[['Candy Variety','Date','Weekend']]
```

	Candy Variety	Date	Weekend
0	Chocolate Hearts	2020-02-09	1
1	Sour Jelly	2020-10-24	1
2	Candy Canes	2020-12-18	0
3	Sour Jelly	2020-10-25	1
4	Fruit Drops	2020-10-18	1

1) Imputation : Imputation deals with how to handle data that has missing values. While one solution to this problem is to delete entries that lack specific values, doing so may result in the loss of some important data.

```
data={'Candy Variety':['Chocolate Hearts','Sour Jelly','Candy Canes','Sour Jelly','Fruit Drops'],
      'Date and Time':['09-02-2020 14:05','24-10-2020 18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
      'Day':['Sunday','Saturday','Friday','Sunday','Sunday'],
      'Length':[3, 3.5, 3.5, 3.5, 5], 'Breadth':[2,2,2.5,2,3], 'Price':[7.5, 7.6, 8, 7.6, 9]}
df = pd.DataFrame(data)
df['Date and Time'] = pd.to_datetime(df['Date and Time'], format="%d-%m-%Y %H:%M")

#Appending a row with missing values
df.loc[len(df.index)] =[np.NaN,'22-10-2020 17:24:00','Thursday', 3.5, 2, np.NaN]
df
```

	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	2020-02-09 14:05:00	Sunday	3.0	2.0	7.5
1	Sour Jelly	2020-10-24 18:00:00	Saturday	3.5	2.0	7.6
2	Candy Canes	2020-12-18 20:13:00	Friday	3.5	2.5	8.0
3	Sour Jelly	2020-10-25 10:00:00	Sunday	3.5	2.0	7.6
4	Fruit Drops	2020-10-18 15:46:00	Sunday	5.0	3.0	9.0
5	NaN	22-10-2020 17:24:00	Thursday	3.5	2.0	NaN

```
[ ] df['Candy Variety']=df['Candy Variety'].fillna(df['Candy Variety'].mode()[0])
    df['Price']=df['Price'].fillna(df['Price'].mean())
    df
```

	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	2020-02-09 14:05:00	Sunday	3.0	2.0	7.50
1	Sour Jelly	2020-10-24 18:00:00	Saturday	3.5	2.0	7.60
2	Candy Canes	2020-12-18 20:13:00	Friday	3.5	2.5	8.00
3	Sour Jelly	2020-10-25 10:00:00	Sunday	3.5	2.0	7.60
4	Fruit Drops	2020-10-18 15:46:00	Sunday	5.0	3.0	9.00
5	Sour Jelly	22-10-2020 17:24:00	Thursday	3.5	2.0	7.94

2) Discretization : Discretization is essentially the process of taking a set of data values and logically classifying them into bins (or buckets). Binning is applicable to both qualitative and numerical variables. Although the granularity of the data is lost, this may assist prevent data from overfitting.

```
[ ] df['Type of Day']=np.where(df['Day'].isin(['Saturday', 'Sunday']), 'Weekend', 'Weekday')
    df[['Candy Variety','Day','Type of Day']]
```

	Candy Variety	Day	Type of Day
0	Chocolate Hearts	Sunday	Weekend
1	Sour Jelly	Saturday	Weekend
2	Candy Canes	Friday	Weekday
3	Sour Jelly	Sunday	Weekend
4	Fruit Drops	Sunday	Weekend
5	Sour Jelly	Thursday	Weekday

3) Categorical Encoding : Categorical encoding is the technique used to encode categorical features into numerical values which are usually simpler for an algorithm to understand.

```
for x in df['Type of Day'].unique():
  df[x]=np.where(df['Type of Day']==x,1,0)
df[['Candy Variety','Day','Type of Day','Weekend','Weekday']]
```

	Candy Variety	Day	Type of Day	Weekend	Weekday
0	Chocolate Hearts	Sunday	Weekend	1	0
1	Sour Jelly	Saturday	Weekend	1	0
2	Candy Canes	Friday	Weekday	0	1
3	Sour Jelly	Sunday	Weekend	1	0
4	Fruit Drops	Sunday	Weekend	1	0
5	Sour Jelly	Thursday	Weekday	0	1

4) Feature Splitting : Splitting features into parts can sometimes improve the value of the features toward the target to be learned. For instance, in this case, Date better contributes to the target function than Date and Time.

```
df['Date and Time'] = pd.to_datetime(df['Date and Time'])
df['Date']=df['Date and Time'].dt.date
df[['Candy Variety','Date']]
```

	Candy Variety	Date
0	Chocolate Hearts	2020-02-09
1	Sour Jelly	2020-10-24
2	Candy Canes	2020-12-18
3	Sour Jelly	2020-10-25
4	Fruit Drops	2020-10-18
5	Sour Jelly	2020-10-22

5) Handling Outliers : Outliers are unusually high or low values in the dataset which are unlikely to occur in normal scenarios. Since these outliers could adversely affect your prediction they must be handled appropriately.

6) Variable Transformations : Variable transformation techniques could help with normalizing skewed data. One such popularly used transformation is the logarithmic transformation. Logarithmic transformations operate to compress the larger numbers and relatively expand the smaller numbers.

7) Scaling : Feature scaling is done owing to the sensitivity of some machine learning algorithms to the scale of the input values. This technique of feature scaling is sometimes referred to as feature normalization.

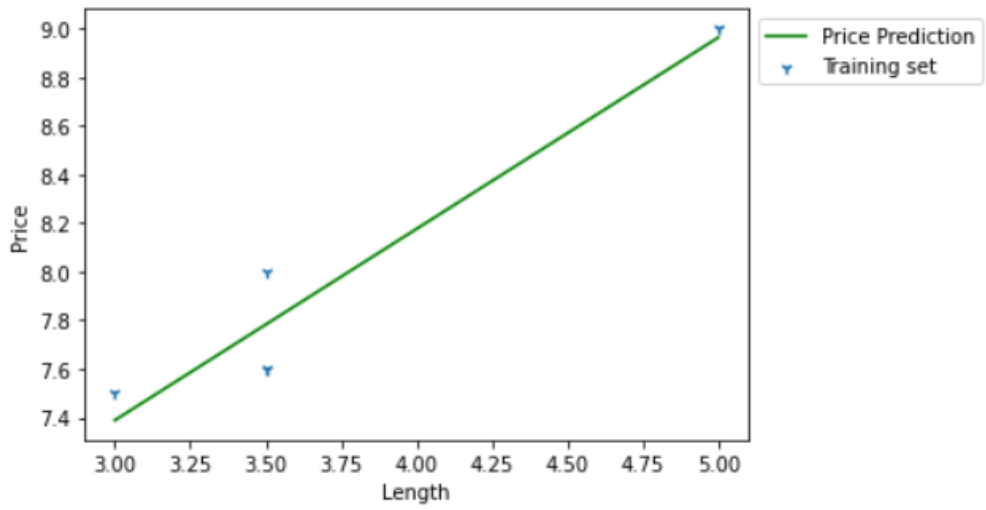
8) Creating Features : This can be done by simple mathematical operations such as aggregations to obtain the mean, median, mode, sum, or difference and even product of two values. These features, although derived directly from the given data, when carefully chosen to relate to the target can have an impact on the performance

```
data={'Candy Variety':['Chocolate Hearts','Sour Jelly','Candy Canes','Sour Jelly','Fruit Drops'],
      'Date and Time':['09-02-2020 14:05','24-10-2020 18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
      'Day':['Sunday','Saturday','Friday','Sunday','Sunday'], 'Length':[3, 3.5, 3.5, 3.5, 5],
      'Breadth':[2,2,2.5,2,3], 'Price':[7.5, 7.6, 8, 7.6, 9]}
df = pd.DataFrame(data)
df
```

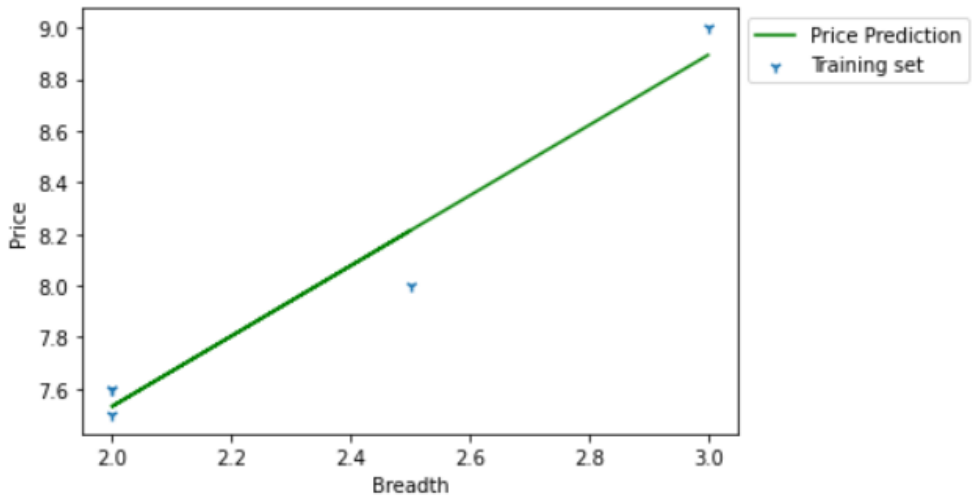
	Candy Variety	Date and Time	Day	Length	Breadth	Price
0	Chocolate Hearts	09-02-2020 14:05	Sunday	3.0	2.0	7.5
1	Sour Jelly	24-10-2020 18:00	Saturday	3.5	2.0	7.6
2	Candy Canes	18-12-2020 20:13	Friday	3.5	2.5	8.0
3	Sour Jelly	25-10-2020 10:00	Sunday	3.5	2.0	7.6
4	Fruit Drops	18-10-2020 15:46	Sunday	5.0	3.0	9.0

```python
import matplotlib.pyplot as plt
import numpy as np

def simple_linear_regression(x, y):
    # number of observations
    n = np.size(x)

    mean_x = np.mean(x)
    mean_y = np.mean(y)


    xy = np.sum(y*x) - n*mean_y*mean_x
    xx = np.sum(x*x) - n*mean_x*mean_x

    # calculating slope
    m = xy / xx

    #calculating intercept
    c = mean_y - m*mean_x

    return m,c
```

```
x=df['Length'].to_numpy()
y=df['Price'].to_numpy()

m,c = simple_linear_regression(x,y)
y_pred = c + m*x

plt.plot(x, y_pred , color = "g", label='Price Prediction')
plt.scatter(df['Length'].to_numpy() , y, marker='1', label='Training set')
plt.xlabel('Length')
plt.ylabel('Price')
plt.legend(bbox_to_anchor=(1, 1))
plt.show()
```



```
x=df['Breadth'].to_numpy()
y=df['Price'].to_numpy()

m,c = simple_linear_regression(x,y)
y_pred = c + m*x

plt.plot(x, y_pred , color = "g", label='Price Prediction')
plt.scatter(df['Breadth'].to_numpy() , y, marker='1', label='Training set')
plt.xlabel('Breadth')
plt.ylabel('Price')
plt.legend(bbox_to_anchor=(1, 1))
plt.show()
```



+ Code

```
df['Size']=df['Breadth']*df['Length']
df[['Candy Variety','Price', 'Size']]
```

	Candy Variety	Price	Size
0	Chocolate Hearts	7.5	6.00
1	Sour Jelly	7.6	7.00
2	Candy Canes	8.0	8.75
3	Sour Jelly	7.6	7.00
4	Fruit Drops	9.0	15.00

```
x=df['Size'].to_numpy()
y=df['Price'].to_numpy()

m,c = simple_linear_regression(x,y)
y_pred = c + m*x

plt.plot(x, y_pred , color = "g", label='Price Prediction')
plt.scatter(df['Size'].to_numpy() , y, marker='1', label='Training set')
plt.xlabel('Size')
plt.ylabel('Price')
plt.legend(bbox_to_anchor=(1, 1))
plt.show()
```