INTRODUCTION TO MACHINE LEARNING

LAB ASSIGNMENT – 3

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

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

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.

```
import pandas as pd
data={'Candy Variety':['Cadbuary','Britania','Parlie','Red Bell','Nestle'],
      'Date and Time':['02-03-2022 13:05','2-01-2022 18:50','23-12-2022 10:13','15-09-2022 09:00','10-10-2022 15:46'],
      'Day':['Sunday','Monday','Friday','Tuesday','Sunday'],
      'Length':[3, 3.5, 3.5, 3.5, 5], 'Breadth':[2,2,2.5,2,3], 'Price':[10.0, 5.0, 12.0, 3.0, 10.0]}
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	Cadbuary	2022-03-02 13:05:00	Sunday	3.0	2.0	10.0
1	Britania	2022-01-02 18:50:00	Monday	3.5	2.0	5.0
2	Parlie	2022-12-23 10:13:00	Friday	3.5	2.5	12.0
3	Red Bell	2022-09-15 09:00:00	Tuesday	3.5	2.0	3.0
4	Nestle	2022-10-10 15:46:00	Sunday	5.0	3.0	10.0

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

	Candy Variety	Date
0	Cadbuary	2022-03-02
1	Britania	2022-01-02
2	Parlie	2022-12-23
3	Red Bell	2022-09-15
4	Nestle	2022-10-10

```
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	Cadbuary	2022-03-02	1
1	Britania	2022-01-02	0
2	Parlie	2022-12-23	0
3	Red Bell	2022-09-15	0
4	Nestle	2022-10-10	1

```
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	Cadbuary	2022-03-02 13:05:00	Sunday	3.0	2.0	10.0
1	Britania	2022-01-02 18:50:00	Monday	3.5	2.0	5.0
2	Parlie	2022-12-23 10:13:00	Friday	3.5	2.5	12.0
3	Red Bell	2022-09-15 09:00:00	Tuesday	3.5	2.0	3.0
4	Nestle	2022-10-10 15:46:00	Sunday	5.0	3.0	10.0
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	Cadbuary	2022-03-02 13:05:00	Sunday	3.0	2.0	10.0
1	Britania	2022-01-02 18:50:00	Monday	3.5	2.0	5.0
2	Parlie	2022-12-23 10:13:00	Friday	3.5	2.5	12.0
3	Red Bell	2022-09-15 09:00:00	Tuesday	3.5	2.0	3.0
4	Nestle	2022-10-10 15:46:00	Sunday	5.0	3.0	10.0
5	Britania	22-10-2020 17:24:00	Thursday	3.5	2.0	8.0

In the above candy problem, you were given 5 records instead of one with the ‘Candy Variety’ missing. Using the above technique, you would predict the missing values as ‘Britania’ resulting in possibly predicting the high sales of Britania all through the year

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	Cadbuary	Sunday	Weekend
1	Britania	Monday	Weekday
2	Parlie	Friday	Weekday
3	Red Bell	Tuesday	Weekday
4	Nestle	Sunday	Weekend
5	Britania	Thursday	Weekday

The grouping of data can be done as:

1. Grouping of equal intervals
2. Grouping based on equal frequencies (of observations in the bin)
3. Grouping based on decision tree sorting (to establish a relationship with target)

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	Cadbuary	Sunday	Weekend	1	0
1	Britania	Monday	Weekday	0	1
2	Parlie	Friday	Weekday	0	1
3	Red Bell	Tuesday	Weekday	0	1
4	Nestle	Sunday	Weekend	1	0
5	Britania	Thursday	Weekday	0	1

1. Count and Frequency encoding- captures each label's representation,

2. Mean encoding -establishes the relationship with the target.

3.Ordinal encoding- number assigned to each unique label.

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']]
```

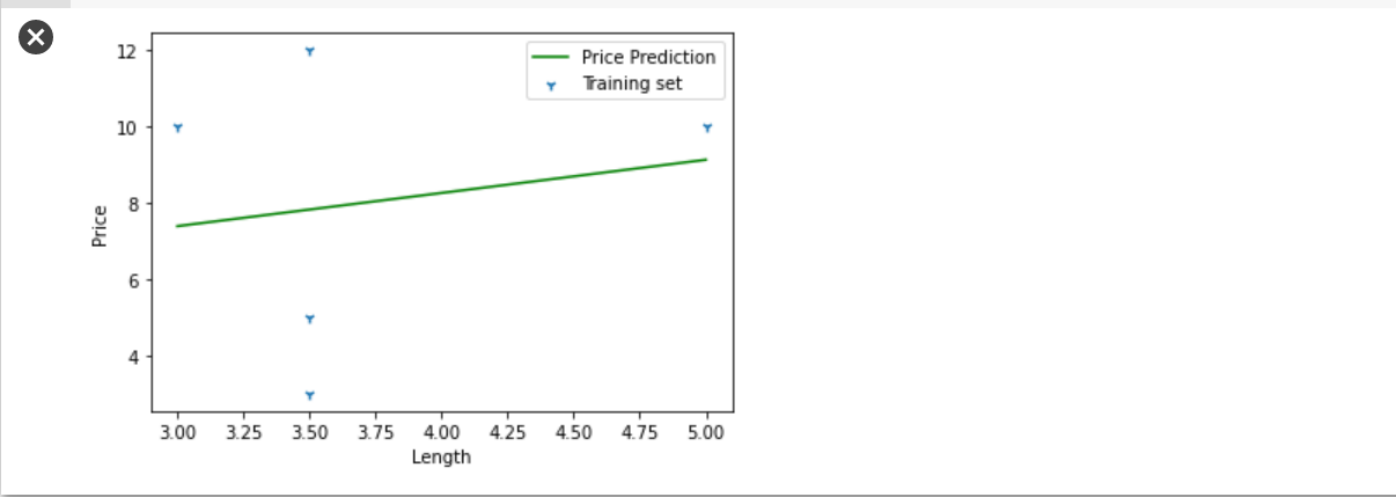
	Candy Variety	Date
0	Cadbuary	2022-03-02
1	Britania	2022-01-02
2	Parlie	2022-12-23
3	Red Bell	2022-09-15
4	Nestle	2022-10-10
5	Britania	2020-10-22

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.

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

```
[3] import pandas as pd
    data={'Candy Variety':['Cadbuary','Britania','Parlie','Red Bell','Nestle'],
          'Date and Time':['02-03-2022 13:05','2-01-2022 18:50','23-12-2022 10:13','15-09-2022 09:00','10-10-2022 15:46'],
          'Day':['Sunday','Monday','Friday','Tuesday','Sunday'],
          'Length':[3, 3.5, 3.5, 3.5, 5], 'Breadth':[2,2,2.5,2,3], 'Price':[10.0, 5.0, 12.0, 3.0, 10.0]}
    df = pd.DataFrame(data)

    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()
```

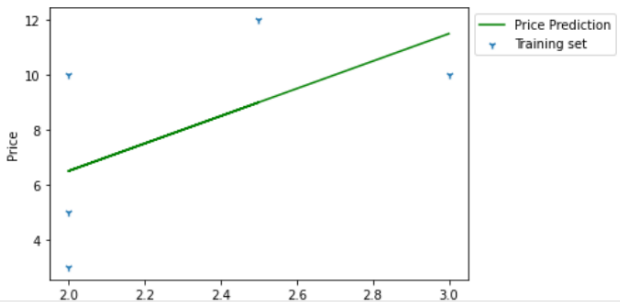


6)Variable Transformations : Variable transformation techniques could help with normalizing skewed data. One such popularly used transformation is the logarithmic transformation. Logarithmic transformations operate to compress the larger numbers and relatively expand the smaller numbers.

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

```
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()
```




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

	Candy Variety	Price	Size
0	Cadbuary	10.0	6.00
1	Britania	5.0	7.00
2	Parlie	12.0	8.75
3	Red Bell	3.0	7.00
4	Nestle	10.0	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()
```