In [1]:

```
import pandas as ps
import matplotlib.pyplot as plt
import seaborn as sns
```

- Get the data
- Preprocess the data
- Seperate the input data and output data
- Seperate the data into traininig and testing data
- Train the model
- Test the model
- Evalauate the model

In [4]:

```
# Read the data
data = ps.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Development-Corpo
data.head()
```

Out[4]:

	Height	Weight	Size
0	158	58	M
1	158	59	M
2	158	63	M
3	160	59	M
4	160	60	M

In [5]:

```
data.shape
```

Out[5]:

```
(18, 3)
```

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 3 columns):
Height    18 non-null int64
Weight    18 non-null int64
Size      18 non-null object
dtypes: int64(2), object(1)
memory usage: 512.0+ bytes
```

In [7]:

```
1 data.isna().sum()
```

Out[7]:

```
Height    0
Weight    0
Size      0
dtype: int64
```

Preprocessing is not required for this datasets because our datasets not having any null values

In [8]:

```
1 # Seperate the data into input and output data
2 input_labels = data[['Height','Weight']]
3 input_labels.head()
```

Out[8]:

	Height	Weight
0	158	58
1	158	59
2	158	63
3	160	59
4	160	60

In [9]:

```
1 output_data = data['Size']
2 output_data.head()
```

. . .

In [10]:

```
1 # get_dummies
2 # LabelEncoder
3 dum = ps.get_dummies(data['Size'])
4 dum
```

. . .

In [12]:

```
1 from sklearn.preprocessing import LabelEncoder
2 lab = LabelEncoder()
3 dum1 = lab.fit_transform(data['Size'])
4 dum1
5 # M=0 and L=1
```

Out[12]:

```
array([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [13]:

```
data['Size']=dum1
```

In [14]:

```
output_data = data['Size']
output_data.head()
```

Out[14]:

```
0    1
1    1
2    1
3    1
4    1
Name: Size, dtype: int32
```

In this dataset we have only 18 samples (Less number of samples) so that's why we are not split the data into train and test data

In [15]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
```

In [16]:

```
knn.fit(input_labels,output_data)
```

Out[16]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

In [17]:

```
# test the model using predict
pred_output = knn.predict(input_labels)
```

In [18]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_score(output_data,pred_output)
```

Out[18]:

```
0.8333333333333334
```

In [19]:

```
confusion_matrix(output_data,pred_output)
```

Out[19]:

```
array([[10,  1],
       [ 2,  5]], dtype=int64)
```

In [20]:

```
1 data.shape
```

Out[20]:

```
(18, 3)
```

In [22]:

```
1 15/18
```

Out[22]:

```
0.8333333333333334
```

In [23]:

```
1 3/18
```

Out[23]:

```
0.16666666666666666
```

In [25]:

```
1 help(knn.fit(input_labels,output_data))
```

```
Help on KNeighborsClassifier in module sklearn.neighbors.classification ob
ject:

class KNeighborsClassifier(sklearn.neighbors.base.NeighborsBase, sklearn.n
eighbors.base.KNeighborsMixin, sklearn.neighbors.base.SupervisedIntegerMix
in, sklearn.base.ClassifierMixin)
 |  KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='aut
o', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=Non
e, **kwargs)
 |
 |  Classifier implementing the k-nearest neighbors vote.
 |
 |  Read more in the :ref:`User Guide <classification>`.
 |
 |  Parameters
 |  ----------
 |  n_neighbors : int, optional (default = 5)
 |      Number of neighbors to use by default for :meth:`kneighbors` queri
es.
 '
```

KNN Regressor

In [26]:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
```

In [27]:

```
df = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Development-Corporat
df.head()
```

Out[27]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

In [28]:

```
df.columns
```

Out[28]:

```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profit'],
dtype='object')
```

In [29]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
R&D Spend          1000 non-null float64
Administration     1000 non-null float64
Marketing Spend    1000 non-null float64
State              1000 non-null object
Profit             1000 non-null float64
dtypes: float64(4), object(1)
memory usage: 39.1+ KB
```

In [30]:

```
df.State.unique()
```

Out[30]:

```
array(['New York', 'California', 'Florida'], dtype=object)
```

In [31]:

```
du = pd.get_dummies(df['State'])
du
```

. . .

In [33]:

```
from sklearn.preprocessing import LabelEncoder
end = LabelEncoder()
du1 = end.fit_transform(df['State'])
du1

# 'New York'=2, 'California'=0, 'Florida'=1
```

. . .

In [34]:

```
df['State'] = du1
```

In [35]:

```
df.head()
```

Out[35]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	2	192261.83
1	162597.70	151377.59	443898.53	0	191792.06
2	153441.51	101145.55	407934.54	1	191050.39
3	144372.41	118671.85	383199.62	2	182901.99
4	142107.34	91391.77	366168.42	1	166187.94

In [36]:

```
input_data = df.drop('Profit',axis=1)
```

In [37]:

```
out_data = df['Profit']
```

In [38]:

```
df.shape
```

Out[38]:

```
(1000, 5)
```

In [39]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(input_data,
                                                 out_data,
                                                 test_size=0.3,
                                                 random_state=3)
```

In [40]:

```
from sklearn.neighbors import KNeighborsRegressor
knn1 = KNeighborsRegressor()
```

In [41]:

```
knn1.fit(x_train,y_train)
```

Out[41]:

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
          weights='uniform')
```

In [42]:

```
knn1.score(x_train,y_train)
```

Out[42]:

```
0.9704378406458822
```

In [43]:

```
knn1.score(x_test,y_test)
```

Out[43]:

```
0.7764952995964944
```

```
test_size 80 20
          70  30
     Train data > test data
```

```
Training accuracy  testing accuracy
     high              high         ---> perfect
     high              low          ---> Over fitting
     Low               High         ---> Under fitting
```

- Take Multi-class dataset and apply to knn Algorithm
- Take that any dataset and apply to knn Regressor comare accuracy

In [ ]:

```

```