MODERN EDUCATION SOCIETY'S

COLLEGE OF ENGINEERING, PUNE – 411001.

INDEX SHEET

Sr. No	Title	Page No.	Date		Remarks
			Performance	Assessment	
1.	To study supervised/unsupervised/Reinforcement learning approaches.				
2.	To acquire, visualize and analyze the data set (from time-domain/ frequency-domain/ etc.) .				
3.	To extract features from a given data set and establish training data.				
4.	To select relevant features using suitable techniques.				
5.	To classify features/To develop classification model and evaluate its performance (any one classifier).				
6.	To develop a regression model and evaluate its performance (any one algorithm).				
7.	Markov process for modelling manufacturing processes				
8.	GA for optimization of multi-dimensional function / path planning in robotics.				

CERTIFICATE

This is to certify that Mr. / Miss.____________________________of class______________Roll No._____Exam No. __________________has satisfactorily completed the Term - Work in the subject **Artificial Intelligence and Machine Learning** as detailed above within the four walls of the institute during period from___________to ___________.

Date: Staff Member Head of the Department

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO:** 01

TITLE: To study supervised/unsupervised/Reinforcement learning approach. **AIM:** To understand supervised/unsupervised/Reinforcement learning approach.

What is machine learning? Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.

Why is machine learning important? Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

Approaches to Machine Learning are as follows:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

1. Supervised Learning In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

How does supervised machine learning work? Supervised machine learning requiresthe data scientist to train the algorithm with both labeled inputs and desired outputs.

Supervised learning algorithms are good for the following tasks:

-Binary classification:Dividing data into two categories. -Multi-class classification: Choosing between more than two types of answers. -Regression modeling: Predicting continuous values. -Ensembling: Combining the predictions of multiple machine learning models to produce an accurate prediction.

- Examples of Supervised Learning Algorithms:- 1. Linear Regression 2. Nearest Neighbor 3. Gaussian Naive Bayes 4. Decision Trees 5. Support Vector Machine (SVM) 6. Random Forest

Advantages:- -Supervised learning allows collecting data and produces data output from previous experiences. -Helps to optimize performance criteria with the help of experience. -Supervised machine learning helps to solve various types of real-world computation problems.

Disadvantages:- -Classifying big data can be challenging. -Training for supervised learning needs a lot of computation time. So, it requires a lot of time.

2.Unsupervised learning This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.

How does unsupervised machine learning work? Unsupervised machine learning algorithms do not require data to be labeled. They sift through unlabeled data to look for patterns that can be used to group data points into subsets. Most types of deep learning, including neural networks, are unsupervised algorithms. Unsupervised learning algorithms are good for the following tasks:

-Clustering: Splitting the dataset into groups based on similarity. -Anomaly detection: Identifying unusual data points in a data set. -Association mining: Identifying sets of items in a data set that frequently occur together. -Dimensionality reduction: Reducing the number of variables in a data set

- Examples of unsupervised learning algorithms: 1. K-means clustering 2. KNN clustering 3. Hierarchical clustering

Advantages: -Labeling of data demands a lot of manual work and expenses. Unsupervised learning solves the problem by learning the data and classifying it without any labels. -The labels can be added after the data has been classified which is much easier. -It is very helpful in finding patterns in data, which are not possible to find using normal methods. -Dimensionality reduction can be easily accomplished using unsupervised learning. -This is the perfect tool for data scientists, as unsupervised learning can help to understand raw data. -We can also find up to what degree the data are similar. This can be accomplished with probabilistic methods. -This type of learning is similar to human intelligence in some way as the model learns slowly and then calculates the result.

Disadvantages: -The result might be less accurate as we do not have any input data to train from. -The model is learning from raw data without any prior knowledge. -It is also a time-consuming process. The learning phase of the algorithm might take a lot of time, as it analyses and calculates all possibilities. -For some projects involving live data, it might require continuous feeding of data to the model, which will result in both inaccurate and time-consuming results. -The more the features, the more the complexity increases.

3.Reinforcement learning Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

How does reinforcement learning work? Reinforcement learning works by programming an algorithm with a distinct goal and a prescribed set of rules for accomplishing that goal. Data scientists also program the algorithm to seek positive rewards -- which it receives when it performs an action that is beneficial toward the ultimate goal -- and avoid punishments -- which it receives when it performs an action that gets it farther away from its ultimate goal. Reinforcement learning is often used in areas such as:

-Robotics: Robots can learn to perform tasks the physical world using this technique. -Video gameplay: Reinforcement learning has been used to teach bots to play a number of video games. -Resource management: Given finite resources and a defined goal, reinforcement learning can help enterprises plan out how to allocate resources.

- Examples of reinforcement learning algorithms: 1.Playing games like Go: Google has reinforcement learning agents that learn to solve problems by playing simple games like Go, which is a game of strategy 2.Self-driving cars: Reinforcement learning is used in self-driving cars for various purposes such as the following.

Advantages: Reinforcement learning can be used to solve very complex problems that cannot be solved by conventional techniques. -This technique is preferred to achieve long-term results, which are very difficult to achieve. -This learning model is very similar to the learning of human beings. Hence, it is close to achieving perfection. -The model can correct the errors that occurred during the training process

Disadvantages: -Reinforcement learning as a framework is wrong in many different ways, but it is precisely this quality that makes it useful. -Too much reinforcement learning can lead to an overload of states, which can diminish the results. -Reinforcement learning is not preferable to use for solving simple problems.

Conclusion Hence,we studied about different approaches in machine learning and their advantages and disadvantages.

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO:** 2

TITLE: To acquire, visualize and analyze the data set (from time-domain/ frequency-domain/ etc.) .

STEPS TO ANALYZE A DATASET

1. Clean Up Your Data

Data wrangling—also called data cleaning—is the process of uncovering and correcting, or eliminating inaccurate or repeat records from your dataset. During the data wrangling process, you'll transform the raw data into a more useful format, preparing it for analysis.

It's imperative to clean your data before beginning analysis. This is particularly important if you'll be presenting your findings to business teams who may use the data for decision-making purposes. Teams need to have confidence that they're acting on a reliable source of information.

2. Identify the Right Questions

Once you've completed the cleaning process, you may have a lot of questions about your final dataset. There's so much potential that can be uncovered through analysis.

Identify the most important questions you hope to answer through your analysis. These questions should be easily measurable and closely related to a specific business problem. If the request for analysis is coming from a business team, ask them to provide explicit details about what they're hoping to learn, what they expect to learn, and how they'll use the information. You can use their input to determine which questions take priority in your analysis.

3. Break Down the Data Into Segments

It's often helpful to break down your dataset into smaller, defined groups. Segmenting your data will not only make your analysis more manageable, but also keep it on track.

For example, if you're attempting to answer questions about a specific department's performance, you'll want to segment your data by department. From there, you'll be able to glean insights about the group that you're concerned with and identify any relationships that might exist between each group.

4. Visualize the Data

One of the most important parts of data analysis is data visualization, which refers to the process of creating graphical representations of data. Visualizing the data will help you to easily identify any trends or patterns and obvious outliers.

By creating engaging visuals that represent the data, you're also able to effectively communicate your findings to key stakeholders who can quickly draw conclusions from the visualizations.

There's a variety of data visualization tools you can use to automatically generate visual representations of a dataset, such as Microsoft Excel, Tableau, and Google Charts.

5. Use the Data to Answer Your Questions

After cleaning, organizing, transforming, and visualizing your data, revisit the questions you outlined at the beginning of the data analysis process. Interpret your results and determine whether the data helps you answer your original questions.

If the results are inconclusive, try revisiting a previous step in the analysis process. Maybe your dataset was too large and should have been segmented further, or perhaps there's a different type of visualization better suited to your data.

6. Supplement with Qualitative Data

Finally, as you near the conclusion of your analysis, remember that this dataset is only one piece of the puzzle.

It's critical to pair your quantitative findings with qualitative information, which you may capture using questionnaires, interviews, or testimonials. While the dataset has the ability to tell you what's happening, qualitative information can often help you understand *why* it's happening.

CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

data=pd.read_csv(r"C:\Users\vishal gaikwad\Desktop\TE VISHAL\sem6\
AI&ML\input.csv")

data.tail(10)
```

```
   id  day  month  year  item_quantity        Name0
     1    12         3  2020              12    Oliver
1    2    13         3  2020              45     Henry
2    3    14         3  2020               8  Benjamin
3    4    15         3  2020              23      John
4    5    16         3  2020              31    Camili
5    6    17         3  2020              40    Rheana
6    7    18         3  2020              55    Joseph
7    8    19         3  2020              13       Raj
8    9    20         3  2020              29     Elias
9   10    21         3  2020              19     Emily
```

```
data.head(10)
```

```
   id  day  month  year  item_quantity        Name0
     1    12         3  2020              12    Oliver
1    2    13         3  2020              45     Henry
2    3    14         3  2020               8  Benjamin
3    4    15         3  2020              23      John
4    5    16         3  2020              31    Camili
5    6    17         3  2020              40    Rheana
6    7    18         3  2020              55    Joseph
7    8    19         3  2020              13       Raj
8    9    20         3  2020              29     Elias
9   10    21         3  2020              19     Emily
```

```
data.pop('year')
```

```
0    2020
1    2020
2    2020
3    2020
4    2020
5    2020
6    2020
7    2020
8    2020
9    2020
Name: year, dtype: int64
```

```
data.head(10)
```

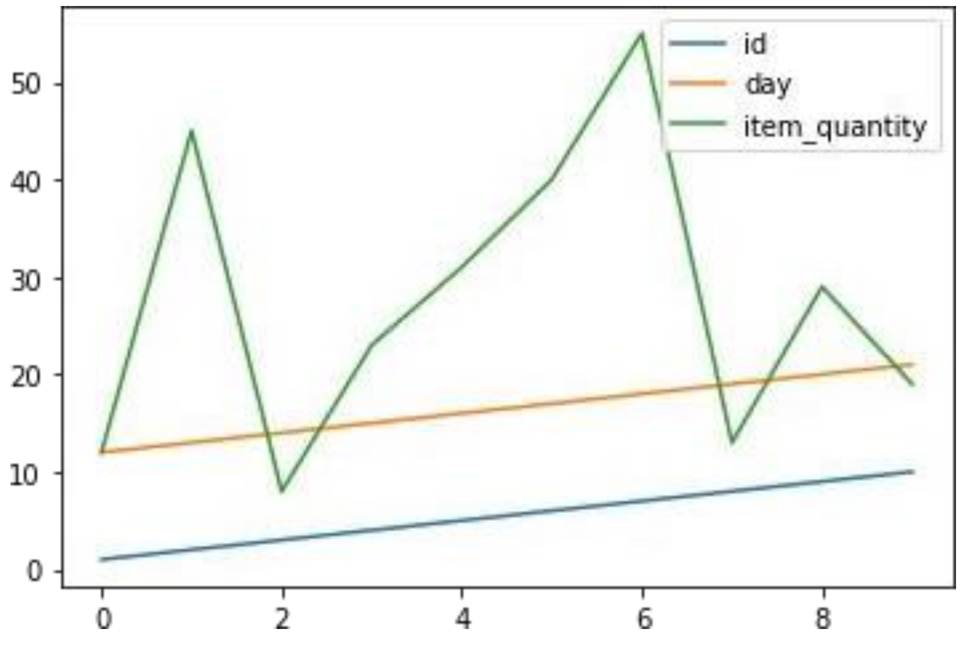
```
      id   day  month   item_quantity          Name0
          1      12           3                    12       Oliver
1        2      13           3                    45         Henry
2        3      14           3                      8   Benjamin
3        4      15           3                    23           John
4        5      16           3                    31       Camili
5        6      17           3                    40       Rheana
6        7      18           3                    55       Joseph
7        8      19           3                    13             Raj
8        9      20           3                    29         Elias
9   10      21           3                    19         Emily
```

```
data.pop('month')
```

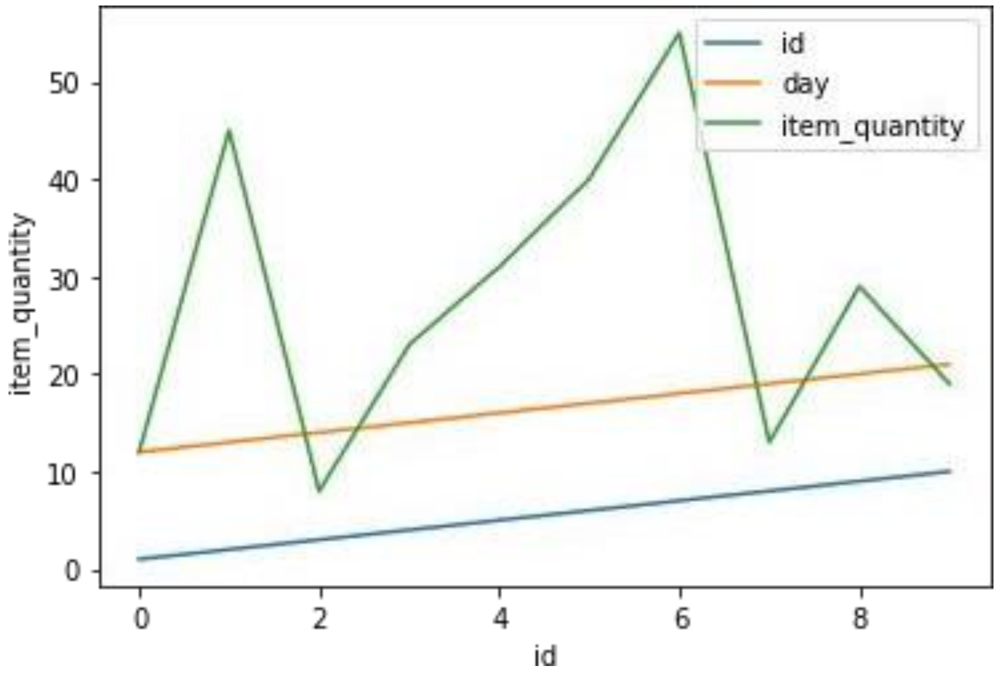
```
0    3
1    3
2    3
3    3
4    3
5    3
6    3
7    3
8    3
9    3
Name: month, dtype: int64
```

```
data.plot()
```

```
<AxesSubplot:>
```

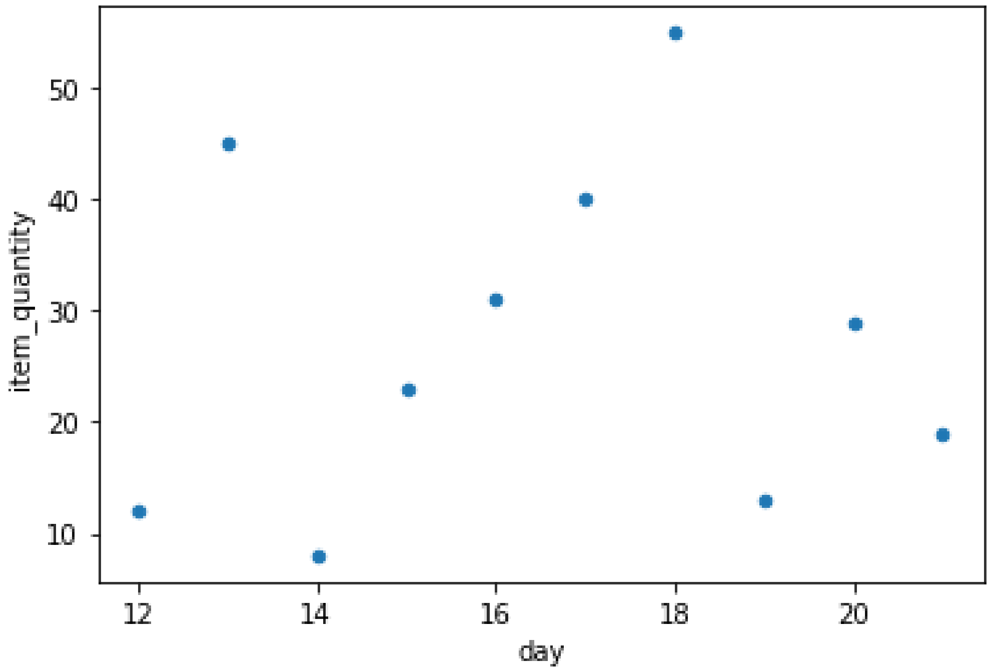


```
data.plot() plt.xlabel('id')
plt.ylabel('item_quantity')
plt.show()
```



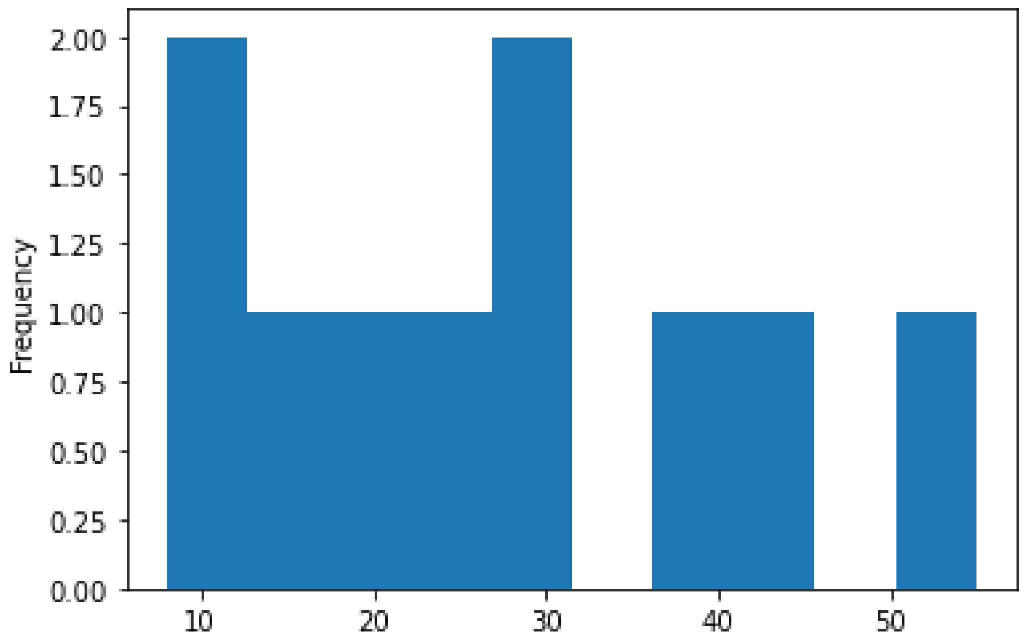
```
data.plot(kind='scatter',x='day',y='item_quantity')
```

```
<AxesSubplot:xlabel='day', ylabel='item_quantity'>
```

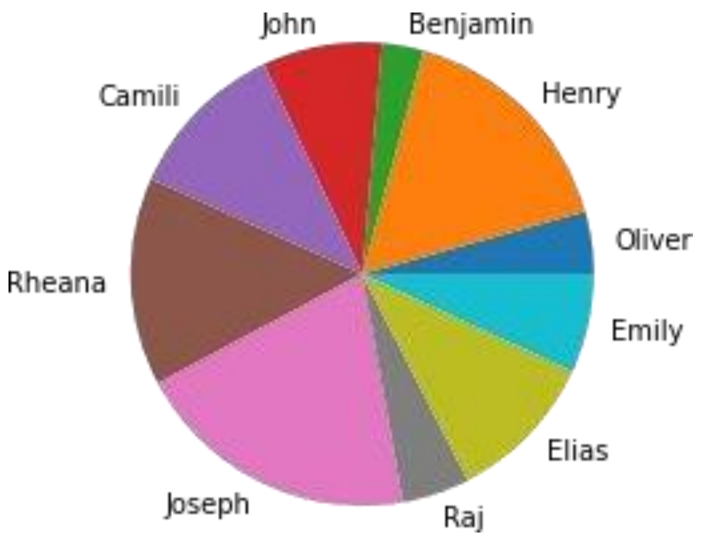


```
data["item_quantity"].plot(kind = 'hist')
```

```
<AxesSubplot:ylabel='Frequency'>
```



```
item_quantity = data["item_quantity"]
name=data["Name"] plt.pie(item_quantity,
labels=name)

 def reset(event):
        ax.patches = []
```



```
data.describe()
```

```
       id         day        item_quantity
count  10.00000   10.00000   10.000000
mean   5.50000    16.50000   27.500000
std    3.02765    3.02765    15.479377
min    1.00000    12.00000   8.000000
25%    3.25000    14.25000   14.500000
50%    5.50000    16.50000   26.000000
75%    7.75000    18.75000   37.750000
max    10.00000   21.00000   55.000000
```

<u>CONCLUSION:</u>

Hence we have studied how to acquire,analyze and visualize data set.

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO:** 3

AIM: To extract features from given data set and establish training data.

Objectives:

1. To learn how to prepare dataset.
2. To understand steps include to upload dataset.
3. To show top 10 Best Features using SelectKBest class

We all may have faced this problem of identifying the related features from a set of data and removing the irrelevant or less important features with do not contribute much to our target variable in order to achieve better accuracy for our model.

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve. Irrelevant or partially relevant features can negatively impact model performance.

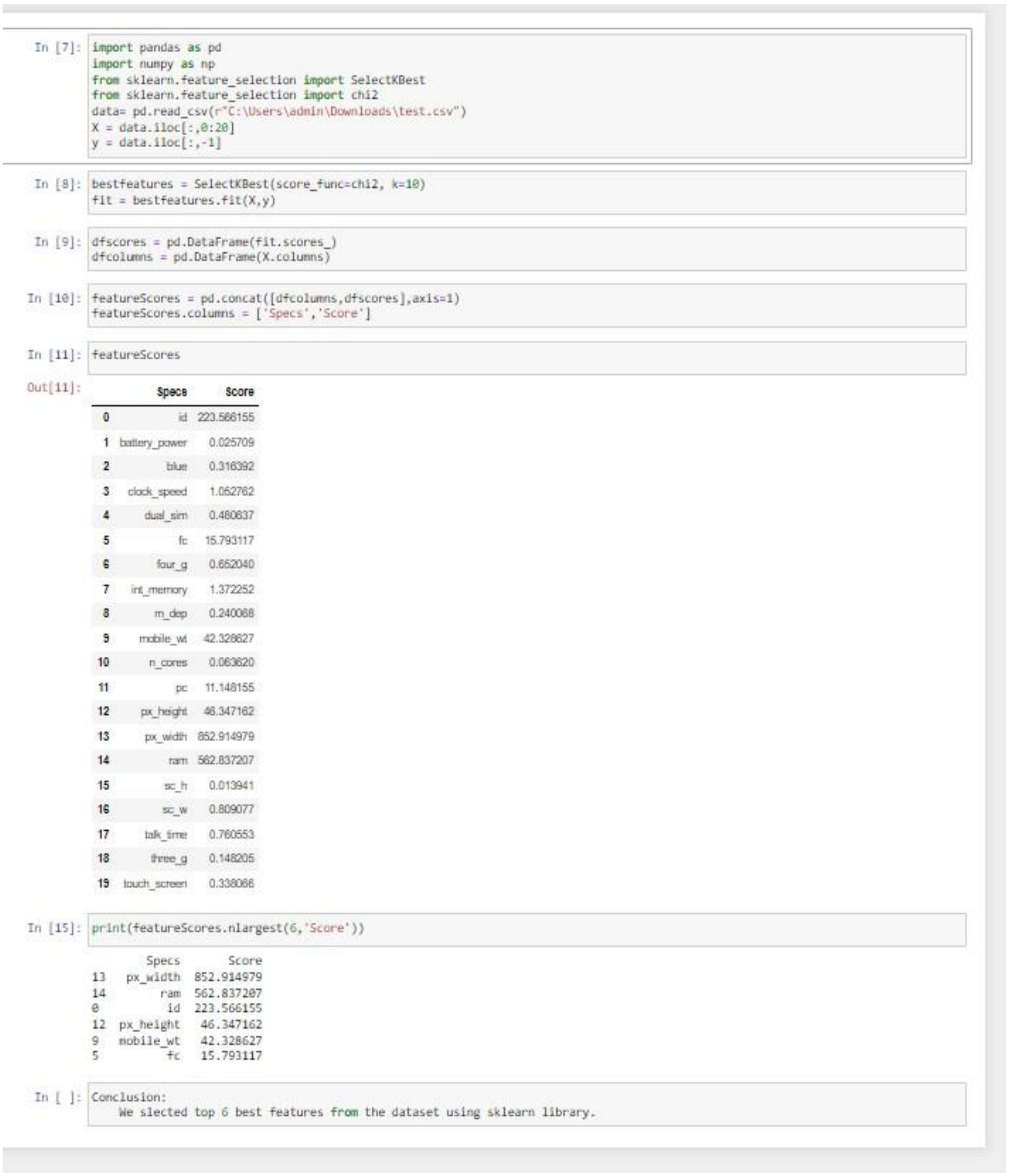
Feature selection and Data cleaning should be the first and most important step of your model designing..
Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features. How to select features and what are Benefits of performing feature selection before modeling your data?

- Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy: Less misleading data means modeling accuracy improves.
- Reduces Training Time: fewer data points reduce algorithm complexity and algorithms train faster.

You can download the dataset from here https://www.kaggle.com/iabhishekofficial/mobile-price-classification#train.csv

- Statistical tests can be used to select those features that have the strongest relationship with the output variable.
- The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.



```
In [7]: import pandas as pd
        import numpy as np
        from sklearn.feature_selection import SelectKBest
        from sklearn.feature_selection import chi2
        data= pd.read_csv(r"C:\Users\admin\Downloads\test.csv")
        X = data.iloc[:,0:20]
        y = data.iloc[:,-1]
```

```
In [8]: bestfeatures = SelectKBest(score_func=chi2, k=10)
        fit = bestfeatures.fit(X,y)
```

```
In [9]: dfscores = pd.DataFrame(fit.scores_)
        dfcolumns = pd.DataFrame(X.columns)
```

```
In [10]: featureScores = pd.concat([dfcolumns,dfscores],axis=1)
         featureScores.columns = ['Specs','Score']
```

```
In [11]: featureScores
```

Out[11]:

	Specs	Score
0	id	223.566155
1	battery_power	0.025709
2	blue	0.316392
3	clock_speed	1.052762
4	dual_sim	0.480637
5	fc	15.793117
6	four_g	0.652040
7	int_memory	1.372252
8	m_dep	0.240068
9	mobile_wt	42.328627
10	n_cores	0.063620
11	pc	11.148155
12	px_height	46.347162
13	px_width	852.914979
14	ram	562.837207
15	sc_h	0.013941
16	sc_w	0.809077
17	talk_time	0.760553
18	three_g	0.148205
19	touch_screen	0.338066

```
In [15]: print(featureScores.nlargest(6,'Score'))

        Specs       Score
13   px_width  852.914979
14        ram  562.837207
0          id  223.566155
12  px_height   46.347162
9   mobile_wt   42.328627
5          fc   15.793117
```

```
In [ ]: Conclusion:
            We slected top 6 best features from the dataset using sklearn library.
```

CONCLUSION:

Hence we have studied how to extract features from given data set and establish training data.

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: 4**

AIM: To select relevant features using suitable technique..

Objectives:

1. To learn how to prepare dataset.
2. To understand steps include to upload dataset.
3. To show top 10 Best Features using SelectKBest class

We all may have faced this problem of identifying the related features from a set of data and removing the irrelevant or less important features with do not contribute much to our target variable in order to achieve better accuracy for our model.

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve. Irrelevant or partially relevant features can negatively impact model performance.

Feature selection and Data cleaning should be the first and most important step of your model designing..,
Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features. How to select features and what are Benefits of performing feature selection before modeling your data?

- Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.

- Improves Accuracy: Less misleading data means modeling accuracy improves.
- Reduces Training Time: fewer data points reduce algorithm complexity and algorithms train faster.

You can download the dataset from here https://www.kaggle.com/iabhishekofficial/mobile-price-classification#train.csv (https://www.kaggle.com/iabhishekofficial/mobile-price-classification#train.csv)

- Statistical tests can be used to select those features that have the strongest relationship with the output variable.
- The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.
- The example below uses the chi-squared (chi²) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset.

In [6]:

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = pd.read_csv("archive/test.csv")
X = data.iloc[:,0:20]  #independent columns
y = data.iloc[:,-1]    #target column i.e price range
```

In [7]:

```
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
```

In [12]:

```
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
```

In [13]:

```
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
```

```
In [14]:
featureScores
```

	Specs	Score
0	id	223.566155
1	battery_power	0.025709
2	blue	0.316392
3	clock_speed	1.052762
4	dual_sim	0.480637
5	fc	15.793117
6	four g	0.652040
7	int_memory	1.372252
8	m_dep	0.240068
9	mobile wt	42.328627
10	n_cores	0.063620
11	pc	11.148155
12	px_height	46.347162
13	px_width	852.914979
14	ram	562.837207
15	sch	0.013941
16	SC_IN	0.809077
17	talk time	0.760553
18	three_g	0.148205
19	touch_screen	0.338066

```
In [15]:
print(featureScores.nlargest(5,'Score')) #print 5 best features
        Specs       Score
13  px_width  852.914979
14       ram  562.837207
0         id  223.566155
12 px_height   46.347162
9  mobile wt   42.328627
```

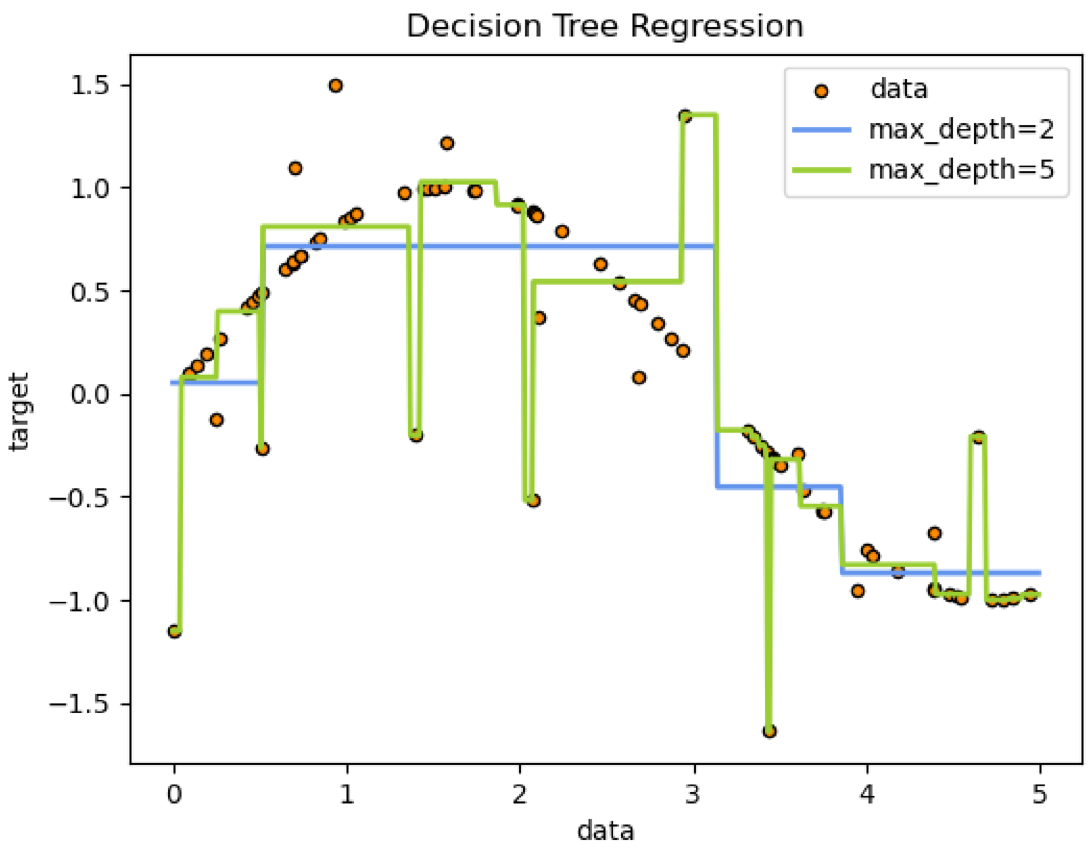
Conclusion: Hence we have selected relevant features using suitable technique

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: 5**

AIM: To classify features/To develop classification model and evaluate its performance (any one classifier).

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.



Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. However scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

```
import pandas
from sklearn import tree
import pydotplus
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import matplotlib.image as pltimg

df = pandas.read_csv("Book1.csv")

print(df)
```

```
    Age  Experience  Rank Nationality   Go
0    36          10     9          UK   NO
1    42          12     4         USA   NO
2    23           4     6           N   NO
3    52           4     4         USA   NO
4    43          21     8         USA  YES
5    44          14     5          UK   NO
6    66           3     7           N  YES
7    35          14     9          UK  YES
8    52          13     7           N  YES
9    35           5     9           N  YES
10   24           3     5         USA   NO
11   18           3     7          UK  YES
12   45           9     9          UK  YES
```

```
d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)

print(df)
```

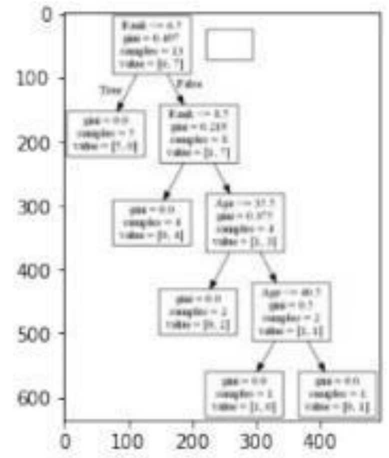
```
    Age  Experience  Rank  Nationality  Go
0    36          10     9            0   0
1    42          12     4            1   0
2    23           4     6            2   0
3    52           4     4            1   0
4    43          21     8            1   1
5    44          14     5            0   0
6    66           3     7            2   1
7    35          14     9            0   1
8    52          13     7            2   1
9    35           5     9            2   1
10   24           3     5            1   0
11   18           3     7            0   1
12   45           9     9            0   1
```

```
features = ['Age', 'Experience', 'Rank', 'Nationality']

X = df[features]
```

```
print(X)
print(y)
```

```
    Age  Experience  Rank  Nationality
0    36          10     9            0
1    42          12     4            1
2    23           4     6            2
3    52           4     4            1
4    43          21     8            1
5    44          14     5            0
6    66           3     7            2
7    35          14     9            0
8    52          13     7            2
9    35           5     9            2
10   24           3     5            1
11   18           3     7            0
12   45           9     9            0
0     0
1     0
2     0
3     0
4     1
5     0
6     1
7     1
8     1
9     1
10    0
11    1
12    1
Name: Go, dtype: int64
```

```
dtree = DecisionTreeClassifier()
dtree = dtree.fit(X, y)
data = tree.export_graphviz(dtree, out_file=None,
feature_names=features)
graph = pydotplus.graph_from_dot_data(data)
graph.write_png('mydecisiontree.png')

img=pltimg.imread('mydecisiontree.png')
imgplot = plt.imshow(img)
plt.show()
```



C0NCLUSION:

Hence we have successfully developed classification model and evaluate its performance.

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: 6**

AIM: To develop regression model and evaluate its performance .

Objectives:

1. To load dataset and separate training and testing data.
2. To train the model using training data and find its accuracy.
3. To learn how to execute python program.
4. To plot Regression Graph.

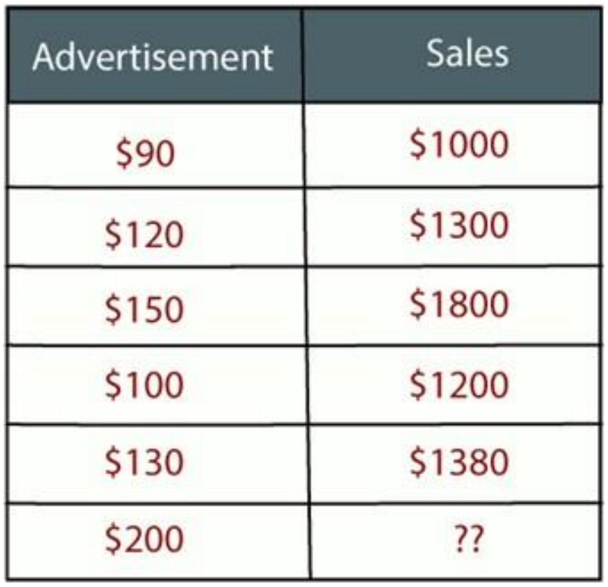
Problem Definition:- The problem definition includes the prediction of average monthly temperature of given year. The data consist of monthly average temperature from Jan to Dec. for year 1997 to 2017. The aim of this experiment is to find regression line for given data and check performance of regression model.

Target Variable: Monthly Avg. Temperature is the target variable.

Regression Analysis in Machine learning

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as temperature, age, salary, price, etc.

Example: Suppose there is a marketing company A, who does various advertisement every year and get sales on that. The below list shows the advertisement made by the company in the last 5 years and the corresponding sales:



Advertisement	Sales
$90	$1000
$120	$1300
$150	$1800
$100	$1200
$130	$1380
$200	??

Now, the company wants to do the advertisement of $200 in the year 2019 and wants to know the prediction about the sales for this year. So to solve such type of prediction problems in machine learning, we need regression analysis.

Terminologies Related to the Regression Analysis:

- Dependent Variable: The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called target variable.
- Independent Variable: The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a predictor.
- Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.
- Multicollinearity: If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
- Underfitting and Overfitting: If our algorithm works well with the training dataset but not well with test dataset, then such problem is called Overfitting. And if our algorithm does not perform well even with training dataset, then such problem is called underfitting.

Types of Regression

There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables. Here we are discussing some important types of regression which are given below:

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression
- Ridge Regression

```
In [17]: # Importing the required modules for
         # Linear Regression using Python
         import matplotlib.pyplot as plt
         import pandas as pd
```

```
In [19]: dataset = pd.read_csv(r"C:\Users\admin\Downloads\DataForLR.csv")


         Input = dataset.iloc[:, :-1].values


         output = dataset.iloc[:, 1].values
```

```
In [20]:
         from sklearn.model_selection import train_test_split

         # 30% data for testing, random state 1
         X_train, X_test, y_train, y_test = train_test_split(Input, output, train_size=.7, random_state=1)
```

```
In [10]: from sklearn.model_selection import train_test_split


         X_train, X_test, y_train, y_test = train_test_split(Input, output, train_size=.7, random_state=1)


         print("training output values: \n",y_train)
         print("Testing output values:\n",y_test)
```

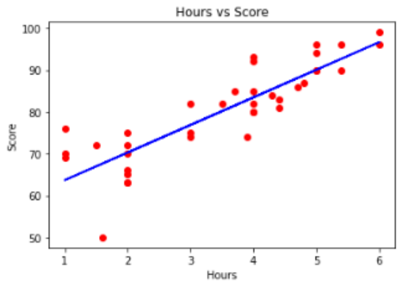
```
training output values:
 [63 50 86 82 82 90 90 85 74 70 74 96 85 72 81 82 96 99 83 87 66 65 93 94
 80 84 63 70 76 80 69 75 92 75 96 90 72]
Testing output values:
 [83 61 83 75 85 87 89 68 67 86 86 88 55 99 69 66]
```

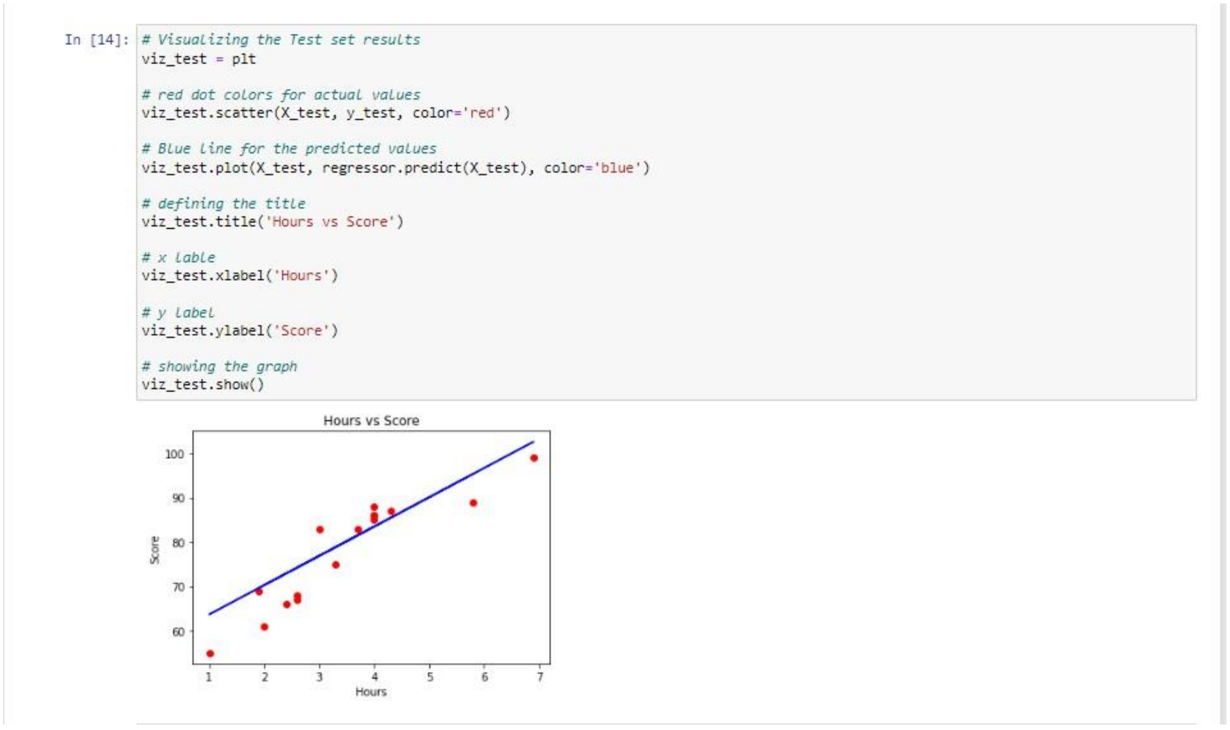
```
In [11]: # Importing linear regression form sklear
         from sklearn.linear_model import LinearRegression

         # initializing the algorithm
         regressor = LinearRegression()

         # Fitting Simple Linear Regression to the Training set
         regressor.fit(X_train, y_train)

         # Predicting the Test set results
         y_pred = regressor.predict(X_test)
```

```
In [12]: # Predicting single value
         Pred_Salary= regressor.predict([[3]])
         print(Pred_Salary)
```

```
[76.87543545]
```

```
In [13]: # Visualizing the Training set results
         viz_train = plt

         # ploting the training dataset in scattered graph
         viz_train.scatter(X_train, y_train, color='red')

         # ploting the testing dataset in line line
         viz_train.plot(X_train, regressor.predict(X_train), color='blue')
         viz_train.title('Hours vs Score')

         # labeling the input and outputs
         viz_train.xlabel('Hours')
         viz_train.ylabel('Score')

         # showing the graph
         viz_train.show()
```





```
In [14]: # Visualizing the Test set results
         viz_test = plt

         # red dot colors for actual values
         viz_test.scatter(X_test, y_test, color='red')

         # Blue line for the predicted values
         viz_test.plot(X_test, regressor.predict(X_test), color='blue')

         # defining the title
         viz_test.title('Hours vs Score')

         # x lable
         viz_test.xlabel('Hours')

         # y label
         viz_test.ylabel('Score')

         # showing the graph
         viz_test.show()
```

CONCLUSION:

Hence we have successfully developed regression model and evaluate its performance

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: 7**

AIM: Markov process for modelling manufacturing processes.

A Markov chain is a stochastic model created by Andrey Markov, which outlines the probability associated with a sequence of events occurring based on the state in the previous event. A very common and simple to understand model which is highly used in various industries which frequently deal with sequential data such as finance. The algorithm Google uses on its search engine to indicate which links to show first is called the Page Rank algorithm, it's a type of Markov chain. Through mathematics, this model will use our observations to predict an approximation of future events!

The main goal of the Markov process is to identify the probability of transitioning from one state to another. This is one of the main appeals to Markov, that the future state of a stochastic variable is only dependent on its present state. An informal definition of a stochastic variable is described as a variable whose values depends on the outcomes of random occurrences.

What is a Markov Chain

As stated above, a Markov process is a stochastic process which has memoryless characteristics. The term "memorylessness" in mathematics is a property of probability distributions. It generally refers to scenarios when the time associated with a certain event occurring does not depend on how much time has already elapsed. In other words, when a model has a memoryless property, it implies that the model 'forgets' which state the system is in. Hence, probabilities would not be influenced by the previous states of the process. This property of memorylessness is the main characteristic of a Markov process. The predictions associated with a Markov process are conditional on its current state and is independent of past and future states.

This memorylessness attribute is both a blessing and a curse to the Markov model in application. Imagine the scenario that you wish to predict words/sentences based on previously entered text (similar to how Google does for gmail). Well, the benefit of using the Markov process to do this is that the newly generated predictions would not be dependent on something you wrote paragraphs ago. However, the downfall to this is that you won't be able to predict text which has context in a previous state of the model. This is a common problem in NLP (natural language processing) and an issue many models face.

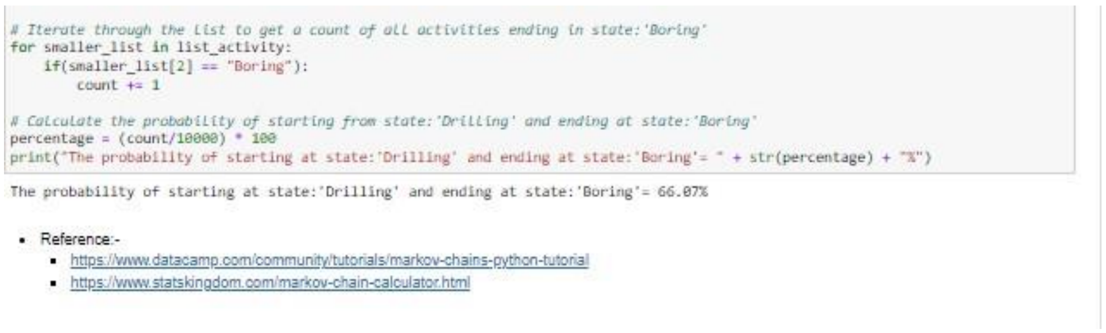
```
In [6]: import numpy as np
        import random as rm
```

```
In [11]: # The statespace
         states = ["Drilling","Milling","Boring"]

         # Possible sequences of events
         transitionName = [["DD","DB","DM"],["BD","BB","BM"],["MR","MB","MM"]]

         # Probabilities matrix (transition matrix)
         transitionMatrix = [[0.2,0.6,0.2],[0.1,0.6,0.3],[0.2,0.7,0.1]]
```

```
In [17]: if sum(transitionMatrix[0])+sum(transitionMatrix[1])+sum(transitionMatrix[1]) != 3:
             print("Somewhere, something went wrong. Transition matrix, perhaps?")
         else: print("Nothing wrong upto here, Move ahead!! ;)")

         Nothing wrong upto here, Move ahead!! ;)
```

```
In [16]: # A function that implements the Markov model to forecast the state/mood.
         def activity_forecast(days):
             # Choose the starting state
             activityToday = "Drilling"
             print("Start state: " + activityToday)
             # Shall store the sequence of states taken. So, this only has the starting state for now.
             activityList = [activityToday]
             i = 0
             # To calculate the probability of the activityList
             prob = 1
             while i != days:
                 if activityToday == "Drilling":
                     change = np.random.choice(transitionName[0],replace=True,p=transitionMatrix[0])
                     if change == "DD":
                         prob = prob * 0.2
                         activityList.append("Drilling")
                         pass
                     elif change == "DB":
                         prob = prob * 0.6
                         activityToday = "Boring"
                         activityList.append("Boring")
                     else:
                         prob = prob * 0.2
                         activityToday = "Milling"
                         activityList.append("Milling")
                 elif activityToday == "Boring":
                     change = np.random.choice(transitionName[1],replace=True,p=transitionMatrix[1])
                     if change == "BB":
                         prob = prob * 0.5
                         activityList.append("Boring")
                         pass
                     elif change == "BD":
                         prob = prob * 0.2
                         activityToday = "Drilling"
                         activityList.append("Drilling")
                     else:
                         prob = prob * 0.3
                         activityToday = "Milling"
                         activityList.append("Milling")
                 elif activityToday == "Milling":
                     change = np.random.choice(transitionName[2],replace=True,p=transitionMatrix[2])
                     if change == "MM":
                         prob = prob * 0.1
                         activityList.append("Milling")
                         pass
                     elif change == "MD":
                         prob = prob * 0.2
                         activityToday = "Drilling"
                         activityList.append("Drilling")
                     else:
                         prob = prob * 0.7
                         activityToday = "Boring"
                         activityList.append("Boring")
                 i += 1
             print("Possible states: " + str(activityList))
             print("End state after "+ str(days) + " days: " + activityToday)
             print("Probability of the possible sequence of states:{:.2f} ".format(prob))

         # Function that forecasts the possible state for the next 2 iterations
         activity_forecast(2)

         Start state: Drilling
         Possible states: ['Drilling', 'Boring', 'Boring']
         End state after 2 days: Boring
         Probability of the possible sequence of states:0.30
```

In [15]:
```python
def activity_forecast(days):
    # Choose the starting state
    activityToday = "Drilling"
    activityList = [activityToday]
    i = 0
    prob = 1
    while i != days:
        if activityToday == "Drilling":
            change = np.random.choice(transitionName[0],replace=True,p=transitionMatrix[0])
            if change == "DD":
                prob = prob * 0.2
                activityList.append("Drilling")
                pass
            elif change == "DB":
                prob = prob * 0.6
                activityToday = "Boring"
                activityList.append("Boring")
            else:
                prob = prob * 0.2
                activityToday = "Milling"
                activityList.append("Milling")
        elif activityToday == "Boring":
            change = np.random.choice(transitionName[1],replace=True,p=transitionMatrix[1])
            if change == "BB":
                prob = prob * 0.5
                activityList.append("Boring")
                pass
            elif change == "BD":
                prob = prob * 0.2
                activityToday = "Drilling"
                activityList.append("Drilling")
            else:
                prob = prob * 0.3
                activityToday = "Milling"
                activityList.append("Milling")
        elif activityToday == "Milling":
            change = np.random.choice(transitionName[2],replace=True,p=transitionMatrix[2])
            if change == "MM":
                prob = prob * 0.1
                activityList.append("Milling")
                pass
            elif change == "MD":
                prob = prob * 0.2
                activityToday = "Drilling"
                activityList.append("Drilling")
            else:
                prob = prob * 0.7
                activityToday = "Boring"
                activityList.append("Boring")
        i += 1
    return activityList

# To save every activityList
list_activity = []
count = 0

# `Range` starts from the first count up until but excluding the last count
for iterations in range(1,10000):
        list_activity.append(activity_forecast(2))

# Check out all the `activityList` we collected
#print(list_activity)
```



```
# Iterate through the list to get a count of all activities ending in state:'Boring'
for smaller_list in list_activity:
    if(smaller_list[2] == "Boring"):
        count += 1

# Calculate the probability of starting from state:'Drilling' and ending at state:'Boring'
percentage = (count/10000) * 100
print("The probability of starting at state:'Drilling' and ending at state:'Boring'= " + str(percentage) + "%")
```

```
The probability of starting at state:'Drilling' and ending at state:'Boring'= 66.07%
```

- Reference:-
  - https://www.datacamp.com/community/tutorials/markov-chains-python-tutorial
  - https://www.statskingdom.com/markov-chain-calculator.html

CONCLUSION:

Hence we have studied Markov process for modelling manufacturing processes

.

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR: TE	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: 8**

AIM: GA for optimization of multi-dimensional function / path planning in robotics.

Objectives:

1. To load dataset and separate training and testing data.
2. To plot the initial condition and draw path.
3. To use function fitness, ranking.
4. To plot best possible way through obstacle.

Package Used: - Python 3

Problem Definition:-

The problem definition includes the optimize the robot path through the obstacles. The obstacles in these case are red colour rectangle through which robot can't pass through. So it is our aim to find the way to robot to get to goal point through obstacle with optimized (taking minimum time) waypoint using python.

Input data

1. Dataset given in form of flat file.
2. Path points given as x and y coordinates of corresponding points.

Sr. No.	Point	X	Y
1.	P1	1	7
2.	P2	1	11
3.	P3	3	14
4.	P4	3	1
5.	P5	5	8
6.	P6	6	11
7.	P7	6	4
8.	P8	8	4
9.	P9	10	1
10.	P10	10	7
11.	P11	10	11
12.	P12	11	14
13.	P13	13	12
14.	P14	12	2
15.	P15	14	3
16.	P16	14	8

Program:-

```
from tools.population import population
from tools.fitness import fitness
from tools.ranking import ranking
from tools.dna import dna
from tools.draw_plot import show_plot

from config import Config

def main():
    chr_population = population()
    chr_pop_fitness, chr_best_fitness_index = fitness(chr_pop=chr_population)
    chr_ranked_population = ranking(chr_pop_fitness=chr_pop_fitness,
pop=chr_population)

    chr_crossover_mutated_population = dna(chr_pop_fitness=chr_pop_fitness,
        ranked_population=chr_ranked_population, chr_best_fitness_index=
        chr_best_fitness_index, last_pop=chr_population)
    show_plot(best_chromosome=chr_crossover_mutated_population[0])

    while not Config.stop_generation:

        prev_best_fit = chr_pop_fitness[chr_best_fitness_index[0], 0]

        chr_pop_fitness, chr_best_fitness_index = fitness(
            chr_pop=chr_crossover_mutated_population)

        chr_ranked_population = ranking(chr_pop_fitness=chr_pop_fitness,
            pop=chr_crossover_mutated_population)

        chr_crossover_mutated_population = dna(chr_pop_fitness=chr_pop_fitness,
            ranked_population=chr_ranked_population, chr_best_fitness_index=
            chr_best_fitness_index, last_pop=chr_crossover_mutated_population)

        if prev_best_fit == chr_pop_fitness[chr_best_fitness_index[0], 0]:
            Config.stop_criteria += 1
        else:
            Config.stop_criteria = 0
        if Config.stop_criteria >= 5:
            Config.stop_generation = True

        print("Best chromosome is:",
chr_crossover_mutated_population[chr_best_fitness_index[0]])
        show_plot(best_chromosome=chr_crossover_mutated_population[0])
        Config.generations += 1
    show_plot(best_chromosome=chr_crossover_mutated_population[0],
inf_time=True)

if __name__ == '__main__':

    main()
```

Output:-

1. Graph of Path optimization

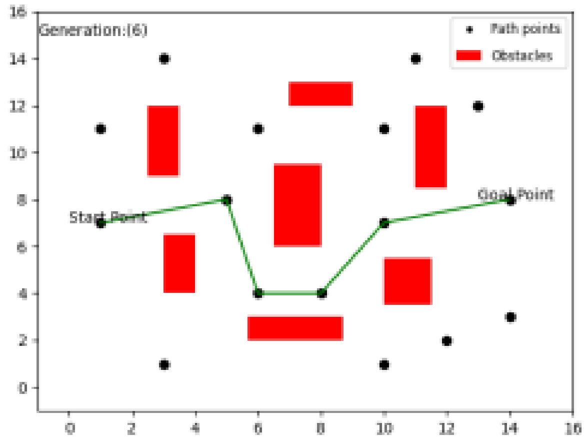


Figure 1 Robo optimized Path planning

2. The Best chromosome :

   Best chromosome is: [0. 4. 6. 7. 9. 15. 15. 15. 15.]

Conclusion:-

- The optimized path for the given points and initial and final point is found and plotted using Genetic algorithm.
- From the multiple path the optimized path and chromosome corresponding to that path is found out.