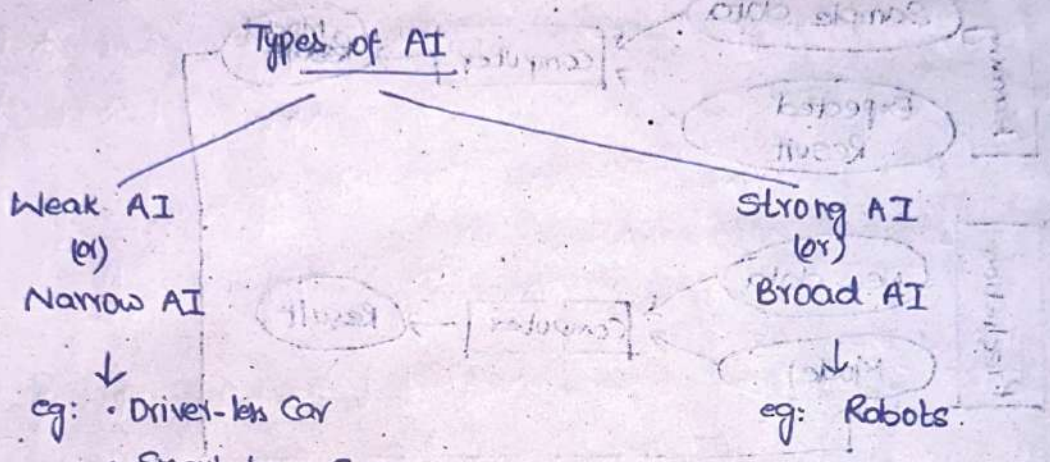
6/06/24 Edunet Foundations Topic: Technical Session
IBM SkillsBuild.

Artificial Intelligence

Engineering of making intelligent machines & programs.

Machine learning:- Part of AI an Ability to learn without being explicitly programmed.

Deep learning :- learned based on deep neural network.



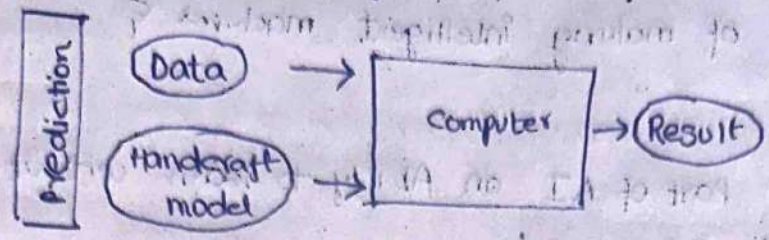
E-commerce Real time Example of AI : Zomato, Flipkart

* **Machine learning:**

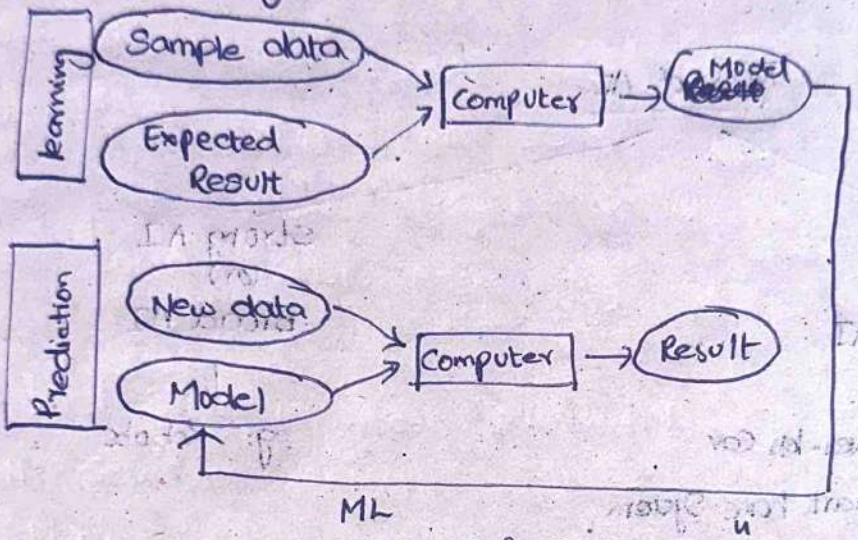
- ML is a way for computers to learn from data & make decisions, without being explicitly programmed for each task.
- Its like teaching a computer to recognize patterns & make sense of information on its own
- ML uses data to detect various patterns in a given dataset
- It learn from past data & improve automatically.

Traditional Programming vs Machine learning:

Traditional modeling:



Machine learning



Application to Understand Concept in Real World: Techable Machine

ML is one of the powerful way to acheive Artificial intelligence

Types of ML

Supervised learning
- Feed everything to Computer
  eg: Input output everything
- Label Data
  eg: Opening a mobile screen with face/voice recognization

Unsupervised learning
- Un-label Data
  eg: Supermarket
- feeding the machine only data, the data is divided into multiple clusters
  eg: fruit clustering

Reinforcement learning
- Action & Rewards
- eg: chess game, UBER Customer Satisfaction.

Working Mechanism of ML

1) Get Data
2) Clean, Prepare & Manipulate Data
3) Train Model
4) Test Data
5) Improve

} Data was split into training & testing (80:20)

80/- training
20/- testing

difference b/w these two values is error value.

Unstructured data - Consists of different types of rating
eg: chat section of zoom, one may give image reactions one may send answerd through text & i.e combination of all types of data.

* Binary Classification:

Representation: <=50k, >50k etc.. (from adult.csv shared in session)

→ To handle large data set we use 'padas'
→ They are simple to use for data structures & manipulating

- head() - display first 5 entires (data(default case))
- tail() - display last 5 entries (default case)
- preprocessing - 'data cleaning process' which is an important prerequisite (helpful for the machine to learn data in better manner to produce eff data models)
- inconsistent data set :- some features are not going to contribute much for students employee table.
- data shape :- for fetching the details of rows & columns in a table.

Box plot concept:

import matplotlib.pyplot as plt #

code:

```
import pandas as pd
data = pd.read_csv("path\adult.csv")
data.shape (rows & columns)
(48842, 15)
data.head(3) (first 3 entries)
```

SNo	Age	Work class	fnlwgt	Education	Ed-NO	Marital stat	Occ	Rel
0	25	Pvt	22648	11th	7	Not	Fishing	child
1	38	Pvt	89861	HS-grad	9	Mar-civ -spouse	Farming	Husb
2	28	Local -gov	336951	Assoc-acdm	12	Soft Dcy		Husb

only first 3 entries were printed (data.head(3))
but in case of default case we get first 5 entries.

Now function 'tail':

data.tail(4)

Similarly last 4 entries were printed.

+ Null values n Missing values

data.isna() - function used to check whether there is missing values are not in particular column

output is binary format (returns either 0 or 1)

data.isna().sum() - count of those missing values

Employe burnout prediction Process — Project

Output: data.isna().sum()

age	0
workclass	0
fnlwgt	0
education	0
education-no	0
marital-stat	0
occupation	0
relationship	0

No missing values in given dataset

Mean, Median, Mode, arbitary
- different modes to address missing values in the given data

* print(data. workclass. value()) [Function]

- operations to be performed on
- only workclass column
- no. of values of count of each catageroy

Output:

workclass

Private	33906
self-emp-not-inc	3862
local-gov	3136
?	2799
State-gov	1981
without-pay	21

others category

In place of '?' we can replace with other category & use same data.

function:

```
data.workclass.replace({'?': 'NOTListed'},
                       inplace = True)
print(data['workclass'].value_counts())
```

output:

workclass	
Private	33906
Self-emp-not-inc	3862
local-gov	3136
→ NOTListed	2799
State-gov	1981
Self-emp-inc	1695

Similarly occupation column.

irrelevant features

```
data = data[data['workclass'] != "Without-pay"]
data = data[data['workclass'] != 'Never-worked']
```

↳ double Array

```
data.shape
```

output: (48811, 15)

```
print(data['workclass'].value_counts())
```

output:

workclass

table.

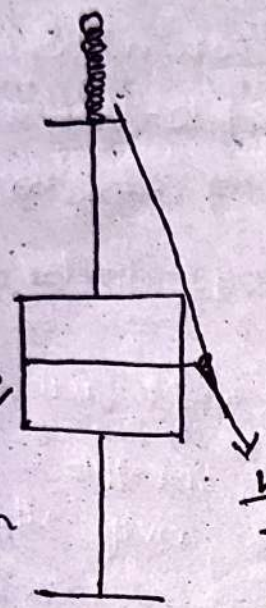
Visualization of outlier detection.

```
import matplotlib.pyplot as plt
plt.boxplot(data['age'])   # numeric values.
plt.show()
```

Box plot concept

only numerical columnred values are taken.

output:



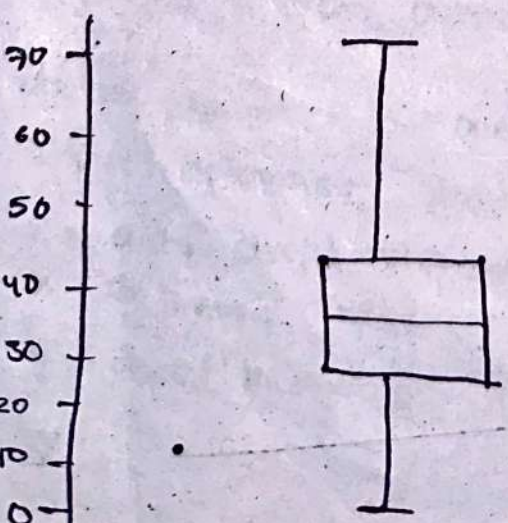
Box represents Most useful information in our dataset

Line represent least information in our data set

* Box plot is only applicable to numeric values
* Bar charts, scatter plots for chategerical values.

These are our outliers in our dataset

```
# to remove outliers in data set
data = data[(data['age']<=75) & (data['age']>=17)]
plt.boxplot(data['age'])   # numerical values.
plt.show()
```

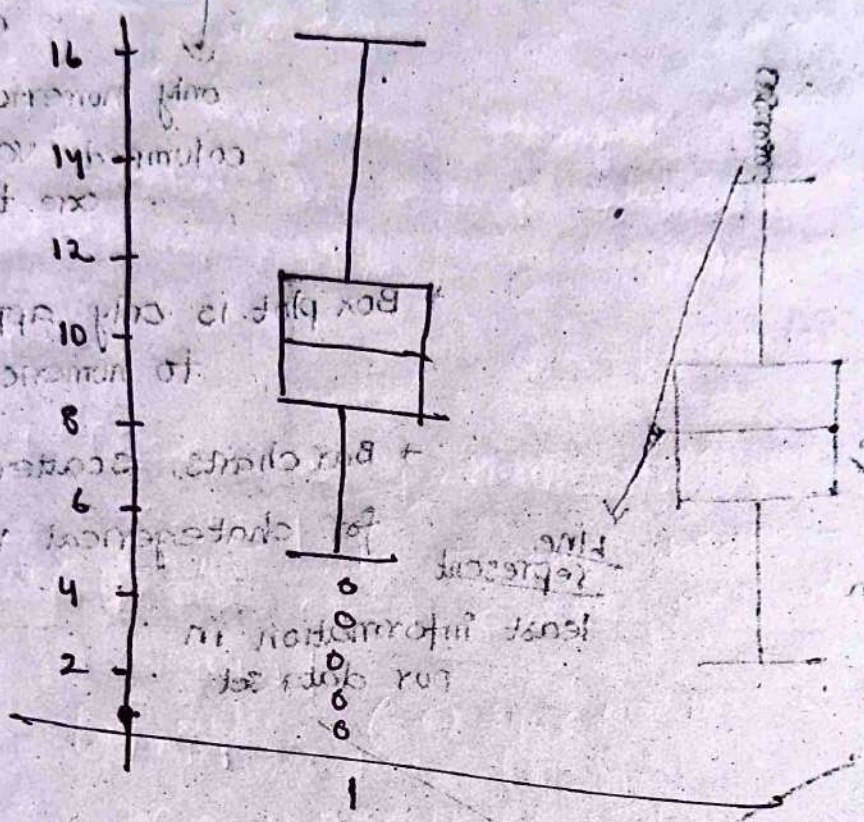


You can observe the outliers are removed from graph

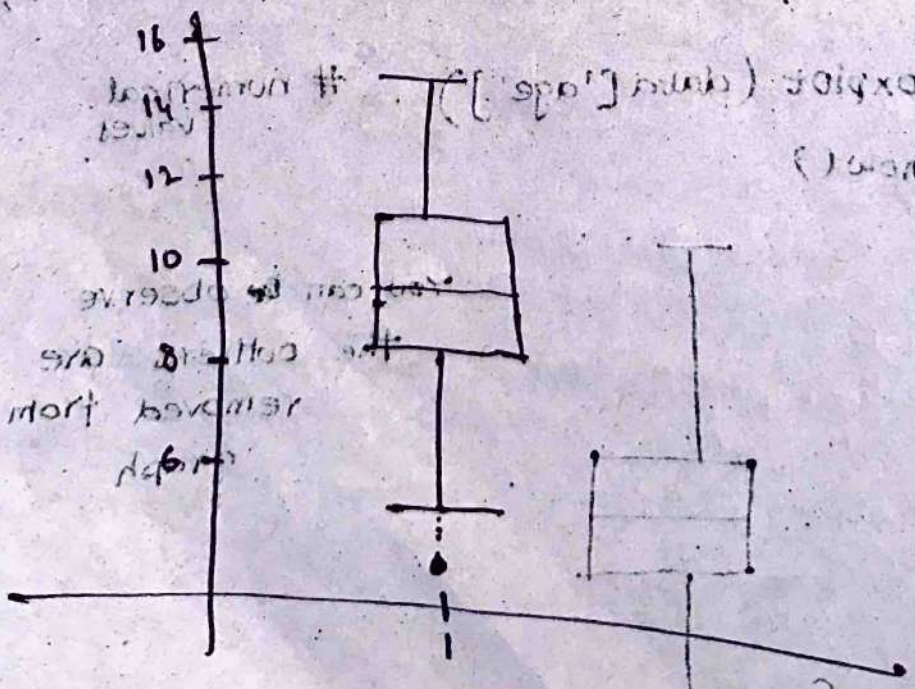
take another column & repeat it

```
plt.boxplot(data['educational-num']) # numerical values
plt.show()
```

output:




```
data = data[(data['educational-num']<=16) &
            (data['educational-num']>=5)]
plt.boxplot(data['educational-num']) # numerical values
plt.show()
```



* Redundant information

```
data = data.drop(columns = ['education'])
```

drop - delete the column from dataset
sparse data - The group with less data points.

* Conversion of categorical values into numerical values.

```
from sklearn.preprocessing import LabelEncode
encoder = LabelEncoder()          #import lib
                                  - invoking an object
data['workclass'] = encoder.fit_transform
                    (data['workclass'])
data['marital-status'] = encoder.fit_transform
                    (data['marital-st'])
data['race'] = encoder.fit_transform(data['race'])
                    .
                    .
                    .
                Similarly other categories
```

All categorical values transform to numeric values

Now

split input from output for data processing

output column: last one (here it is income)
Input columns: remaining 14 columns considered inputs.

```
X = data.drop(columns = ['income'])  # input
X  # ~~income~~ input
Y = data['income']  # output
Y
```

* Normalization

```
from sklearn.preprocessing import MinMaxScaler
Scaler = MinMaxScaler() # object
X = Scaler.fit_transform(x) #input
X #range of 0 & 1
```

All these array values arranged themselves like 0 to 1 to make it easy for machine

Output:

```
array([[0.137, 0.5 , 0.1415, ..., 0. , 0.397,
        0.9512],
       [0.362, 0.5 , 0.052, ..., 0. , 0.5 ,
        0.95121],
       [0.189 , 0.1666 , 0.219, ..., 0. , 0.397,
        0.95121],
       ...,
       ])
```

eg: 1 2 3 4 5 range - 1 to 5

Conversion to Smaller values. 0.2 0.4 0.6 0.8 1 range - 0 to 1

* training & testing data set:

```
from sklearn.model_selection import train_test_split
                                    # training & testing
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                    test_size = 0.2, random_st = 23,
                                    stratify = y)
```

xtrain → input training variable
xtest → input testing variable
2 variables split in (4) for efficient solving

xtrain

* logistic regression.

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(xtrain, ytrain)
```

***

LogisticRegression → for prediction process and also accuracy
- LogisticRegression() → Binary classification

```
predict3 = lr.predict(xtest)
predict3
```

output either positive / negative.

```
array(['<=50K', '<=50K', '<=50K', ..... '<=50K'],
      dtype = object)
```

```
accuracy_score(ytest, predict3)
```

difference of actual output and the predicted output.

output: 0.8203125

↳ 82%.

*

Decision tree - heirarchial tree structure based on decision at each heirarchy

- An Machine learning algorithm
- At each heirarchy you take decision
- Binary classification

Employee Burnout Prediction

Linear Regression

- power statistical technique to model the relationship between a dependent variable and one or more independent variable.

What is Employee Burnout Prediction?

- Predicting the stress level of a employee

→ For an organization it is required to go through the Nature & stress levels of an Employee So it is going to contribute the whole organization's growth.

→ So, it is important to address, the employees mental health, stress levels.

What is Supervised learning:

→ Basically we will be feeding the machine / computer both the input & output so it can generate data & apply it to future datasets.

→ This model uses Labelled data

→ In this project (Employee Burnout Prediction) we will be using Linear Regression Technique.

Dependent and independent Variables

Dependent variable – like an outcome that we want to predict or explain

Independent variable – factors that influence the Dependent variable

eg: • Final examination score is DV

• All our struggle, efforts, internal marks, class Assignment, activies that influence our final score is IV

Defining Employee Burnout:

- Employee Burnout is a state of emotional, physical & mental exhaustion caused by prolonged or excessive stress

1) Emotional exhaustion
2) De-personalization
3) physical symptoms
4) Reduced personal Accomplishment.

Data Collection & Pre-Processing

- online resources : Kaggle.

1) Data Acquisition
2) Data cleaning
3) Feature Engineering.

- All these machine learning models are based on iterations / data
- ML is impossible without data (quality data leads to optimized result)

Work-related Factors (contributing to Employee Burnout)

- Workload, Job control, Role clarity, Work-life Balance.

Personal Factors

- Age, Gender, Personality Traits, Emotional intelligence

Organizational Factors

- Organizational culture, leadership style, support systems.

Linear Regression:

General Formula:-

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

where,

y - dependent variable (outcome of employee Burnout)

$x_1, x_2 \ldots x_n$ - independent variables (factors which influence dependent variables)

$\beta_1, \beta_2 \ldots \beta_n$ - coefficients that give you co-relation b/w Dependent & independent variable.

β_0 - intercept

ϵ - error term (difference b/w observed & predicted values of y)

Google Co-Lab: online platform where you can implement Machine Learning & Data Science related

The dataset consist of 9 columns.

- Employee ID
- Date of Joining
- Gender
- Company Type
- WFH set-up available
- Designation
- Resource allocation
- Mental Fatigue score

} input column.

- Burn Rate. [output column]

Libraries:

numpy: numerical python (Mathematical & Array to support related computations)

why numpy? → 'n' dimensions (like Matrices, Arrays)

Pandas: data Manipulation (dataset from different Platform)

Matplotlib.pyplot, seaborn } used for visualizations, graphs, animated graphs, charts

relation blw data with visualization (pictorial rep)

sklearn.model_selection : In the ML model we divide data for testing & training.

sklearn.preprocessing : Converting statistical data into unit variance with the help of standard scalar.

sklearn.linear_model : Linear Regression Model. (created using training data-set)

sklearn.metrices : to evaluate the performance of our model, metrices i.e we have; mean_squared_error, mean_absolute_error, r2_score

Pickle
os } Storing transmitted data from one format to another
↓
opening a file, reading a file etc.

Data overview:

1) data.head() - in default case it gives the first '5' entries.
Top '5' entries of the data-set

2) data.tail(3) - displays the last '3' entries of a data set.

3) data.describe() - it gives the statistical information of your data set (Min, Max, count values... , mean, SD)

4) data.columns.tolist() - List of column values in the data set

5) data.nunique() - No. of unique values under each category / columns it will show you count

6) data.info() - Gives the information of your dataset.

7) data.isnull().sum() - Count of Null values in each column.

↓ returns boolean values (whether given column has Null value or not)

8) data.isnull().sum().values.sum() - Count of total No. of Null values in whole data set

9) data.corr(numeric_only=True)['Burn Rate'][:-1]

- Finds the correlation b/w Burn Rate & all other columns having only numerical values.

→ '-1' → it excludes to find correlation among Burn Rate itself

10.) sns.pairplot(data)
plt.show()
- Visualization of correlation of data in terms of the graphical representation.

(correlation between burnout rate & remaining 3 columns) mental fatigue, resource allocation, designation

Observation: to check/ infer the co-rrelation b|w variables.

11) data = data.dropna() - Drop off all Null values of our dataframe.

why? because, due to the missing values we have some less correlation b|w the models/ variables thats why we drop all Null values.

12.) data.shape - to verify whether values are dropped off or Not.

13) data.dtypes - Analyze data type of Each column.

14.) data = data.drop('Employee ID', axis = 1)

- dropping the entire employee ID column as it consists of inconsistent data.

To check Employee Seniority can contribute much for the data set or Not.

Checking the correlation of Date of Joining with Target Variable.

→ The answer is; Date of Joining to No: of years of Work we can find the seniority.

One-hot Encoding for categorical values

- Based on Number of categories it will be converted into 0 & 1's.

(Converting categorical features into numerical features)

Preprocessing

- dividing the data into input & output.
  - y - output
  - x - input
- split the data into training & testing
  - training size: 70%
  - testing size: 30%

We create a Linear Regression Model where, we feed both the input & output.

Then, we evaluate the model based on error metrics.

- For calculating the errors we will be feeding the machine both predicted & original outcome so it can generate the error.

Final Observation:

- Based on the evaluation metrics, the Linear Regression model appears to be the best model for predicting burnout analysis.
- It has the lowest mean squared error, root mean squared error, and mean absolute error, indicating better accuracy and precision in its predictions.
- Additionally, it has highest R-squared score indicating a good fit to the data and explaining a higher & explaining a higher ~~propagation~~ proportion of the variance in the target variable.
- So, we are choosing this model for deployment.

Output:

Linear Regression Model performance Metrics:

Mean Squared Error: 0.0031569779113610717

Root Mean Squared Error: 0.056186990588 2831

Mean Absolute Error: 0.045950326447 73

R-squared score: 0.9188226742 47248.