# MY ML Journey

## Introduction to ML

* Training vs Inference
* Sophisticated vs unsophisticated
* Classification vs regression

## NumPy

### Basic Operations

#### Array Creation

* Myarr=np.array([1,2])

#### Creating Custum Array

* Np.zeros((2,3)) (array with zeros and 2\*3 dimention)
* Np.arrange(20,30,2) (array elements from 20-30 even only)
* Np.linspace(10,20,5) (array with 5 elements with proper spacing)

#### Dimension dealing

* Myarr.ndim (dimentions)

#### Datatype dealing

* Myarr.dtype (datatype)
* Myarr=np.array([],dtype=np.floa64) (explicit datatype )
* Np.astype(“float”)

#### Array size

* Myarr.size (number of elements)
* Myarr.itemsize (size of each element)

#### Array shape

* Myarr.shape (row and columns)

#### Array reshape

* Myarr.reshape(3,2) (change dimention but must be compatible)
* MYarr.reshape(3,2,-1) (-1 for unknown dimention)
* Np.reshape(-1)
* Np.ravel (flattened copy)
* Np.flatten (flattended view)

#### Array copy and view

* X=Myarr.copy()
* Y=Myarr.view()
* X.base (to check does it own new data or not)

#### Array iterations

* Np.nditer(arr) (for x in np.nditer(arr):\_)
* np.ndenumerate(arr)

#### Array Stacking or combiing

* Np.hstack((arr1,arr2))
* Np.vstack((arr1,arr2))

#### Array spliting or cutting

* Np.hsplit (arr1,2)
* Np.vsplit (arr1,2)

#### Array Sorting

* Np.sort(arr)

#### Array Filtering

* newarr = arr[arr % 2 == 0]

#### Applying math Functions

* Myarr.min() and Myarr.max() myarr.argmax()
* Myarr.sum(axis=1) for row addition Myarr.sum(axis=0) for column addition
* Np.sqrt(Myarr) , Np.square(Myarr) , Np.std(MyarrNp.sort(Myarr) (sorting)

#### Matrix Operations

* np[:,2:]
* np.hstack(()),np.vstack(()),np.hsplit(),np.vsplit()
* np[np[“name”]== “hamza”]
* argmax

# Data Visualizing and processing

### Read, Write, Excel Csv

* Skiprow=1
* Header=1
* Names=[] give name to each column
* Nrows=5
* Dp.to\_csv(“f.csv”,index=False,header=false )
* df = pd.read\_excel('multi\_sheet.xlsx', sheet\_name='Sheet1')
* pd.merge(file1, file2, on='ID')
* pd.concat([file1, file2], axis=0) # axis=0 for rows, axis=1 for columns

#### 

### Series *a = [1, 7, 2]*

Creation  
 s=pd.Series(a) #by default values are label by index

#### Indexing

s=pd.Series(a,index=[“a”, “b”])

#### Object as Series

calories = {"day1": 420, "day2": 380, "day3": 390}

s=pd.Series(calories)

s=pd.Series(calories,index=[“day1”, “day2”]

### DataFrame data = {   "calories": [420, 380, 390],   "duration": [50, 40, 45] }

#### Creation

Df=pd.dataFrame(data)

Df=pd.read\_csv(“data.txt”)

#### Indexing

Df=pd.dataFrame(data,index= ["day1","day2","day3"])

Df.set\_index(“title”,inplace=True)

Df.reset\_index(inplace=True)

#### Iteration

* + Df.loc[]
  + Df.iloc[] (essential if row and columns are non-numericals)

#### Printing

Print(df.to\_string())

#### Information

* Df.shape[]
* Df.head(7)
* Df.tail(4)
* Df.sample(4)
* Df.info()
* Df.describe
* Df.columns
* Df[“industry”].unique()
* Df[“industry”].value\_counts()

#### New column from existing columns

* Df[“newColumn”]= df[“release\_year”].apply(lamda X: 2023-x)]
* Df[“profit”]=df.apply(lamda X:x[‘revenue’] – x[“budget”],axis=1)

#### Column Deletion

* Dp.drop(index=1)
* Dp.drop(columns=[“movies”])

## Cleaning the Data

## Cleaning Empty cells

1. Dp.dropna()
2. Dp.dropna(how=”all”) if drop only those columns in the row are na
3. Dp.dropna(thresh=2) (should have 2 non na value other wise will drop it)
4. Df.fillna(0,inplace=True)
5. df["Calories"].fillna(130, inplace = True)
6. Df.fillna({
7. “column”:df.temperature.mean()
8. })
9. Df.fillna(method=’ffill”) (previous value become na value)
10. Df.fillna(method=’ffill”, axis=’columns’) side column value become na
11. Df.fillna(method=’ffill”, limit=1) fill na with previous vaue just one
12. Df.interpolate(); place average in na between two rows

## Wrong Format Data Handling

* df['Date'] = pd.to\_datetime(df['Date'])
* you can also delete wrong format

## Wrong Data

Set a policy to remove the wrong data

* for x in df.index:  
    if df.loc[x, "Duration"] > 120:  
      df.drop(x, inplace = True)
* Df.replace(-99999,np.nan),
* Df.replace([-99999,-88888],np.nan)
* Df.replace({

‘column’: -999,

‘column’: [-999,03989]

},np.nan)

## Remove Duplicated

print(df.duplicated())

df.drop\_duplicates(inplace = True)

## Grouping Data

* Df.groupby(“columnName”)
* Df.groupby(lamda index: grouper(df, index, “temperature”) (function will return a key and for that key function data will groped

## Data concatenation and merging

df = pd.concat([india\_weather, us\_weather], keys=["india", "us"], ignore\_index=False)

df=pd.merge(india\_weather, us\_weather,on=”temperature”,how=”inner”)

Matplotlib and Seaborn

#### Line Chart plt.plot(df\_sales["Quarter"],df\_sales["Fridge"],color="orange",label="Fridgle")

plt.plot(df\_sales["Quarter"],df\_sales["Dishwasher"],label="Dishwasher")

plt.plot(df\_sales["Quarter"],df\_sales["Washing Machine"],label="Washing Machine")

plt.ylabel("Revenue in (min $) ")

plt.xlabel("Quarter")

plt.legend()

plt.show()

Py Chart

s=d[["Fridge","Dishwasher","Washing Machine"]].sum()

plt.pie(s.values,labels=s.index,autopct="%1.1f%%",explode=(0.1,0.1,0),shadow=True,startangle=140)

# Bar Chart

d.plot(kind="bar",x="Quarter")

plt.xticks(rotation=45)

# Histogram

sns.histplot(d["Washing Machine"],kde=True)

Scatter chart  
sns.scatterplot(x=d["Quarter"],y=d["Fridge"])

# Supervised machine learning Regression

## Simple + Multi Linear Regression (dependent and independent var relation)

**Simple Linear Regression** is a basic type of regression analysis used to model the relationship between **two variables**:

* **One independent variable (X)** — also called the *predictor* or *input*
* **One dependent variable (Y)** — also called the *response* or *output*

### 📈 The Goal:

To find the **best-fitting straight line** (called the regression line) through the data points, which predicts Y based on X.

### Approach

The approach is to minimize the distance of the random line based on the distance between the data point

∑ (Change i)2

This approach of minimizing the distance of the straight line is called

Gradient descent

Simple linear: mX + b

Muliple: b+ mx1 +…..mxN

## Cost Function

The error that occur because the predicted value is like a average of the previous all value pattern. But if we try to check the previous value it would not be same there would be some difference

Absolute Error = | ∑ y – y^ |

#### MSE

Mean square Error = 1/n ∑ (y – y)2 (square to penalize if error)

## Derivative, chain rule and partial derivative

Derivative: give us the rate of change of function at any point  
Partial derivative: give us the rate of change of function with respect to only one variable while keeping other constant

# Train, Test , Split



# Model Evaluation: Matrics

* Mean\_square\_error
* r2\_score
* mean\_absolute\_error
* ∑ (y- y^) ^2 (SSR squared sum of residuels)
* ∑ ( y – mean ) ^2 (sst squared sum of total)
* R2\_error= 1-(ssr/sst)

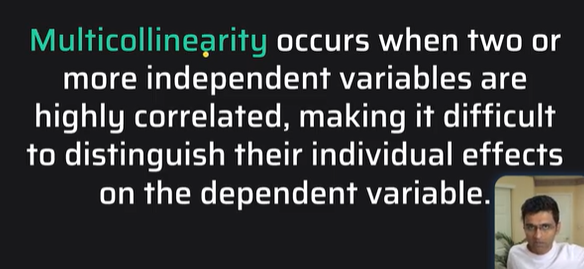
## One Hot Encoding

For ordinal Data -> label Encoding

i.e. bachelor<master<phd

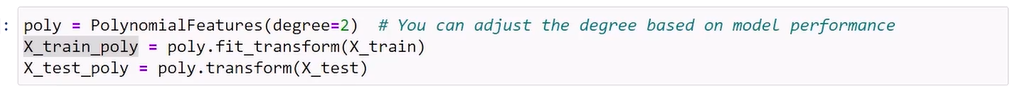
For Nominal

i.e. ali – hamza – zaid



f  


# Polynomial regression



# Reasons and remedies for overfitting/undefitting

1. feature selection
2. Inadequate validation
3. Epoch value
4. Fregulization

# L1 (Lasso) and L2 (ridge) regulization

L1 sum of the absolute weights

L2 sum of the square weights

