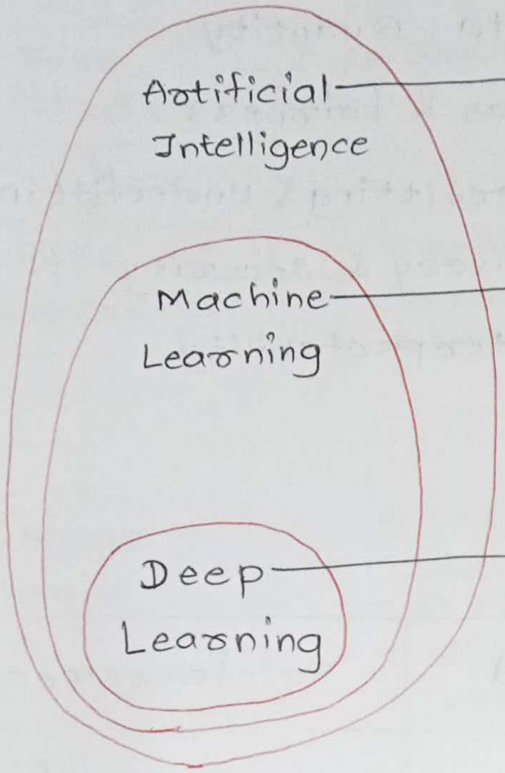
* ML Fundamentals *



Artificial Intelligence → Ability of machine to imitate human intelligence

Machine Learning → Algos to incorporate intelligence into machine by automatically learning from data.

Deep Learning → Algos that mimic human brain to incorporate intelligence into machines

⊙ **Artificial Intelligence**: study of how to train computers, so that computers can do things which humans can do.

Artificial (Human/non-natural made thing.) + Intelligence (Ability to understand/think)

⊙ **Machine Learning (ML)**: Enable computers to learn automatically from past data, it can improve its performance by gaining more data.

↳ ML uses algos for building math models & make predictions using historical data/info.

• Applications of ML:

1. Image Recognition
2. Speech Recognition
3. Recommendation Systems
4. Fraud Detection
5. Self Driving cars
6. Medical Diagnosis
7. Stock Market Trading

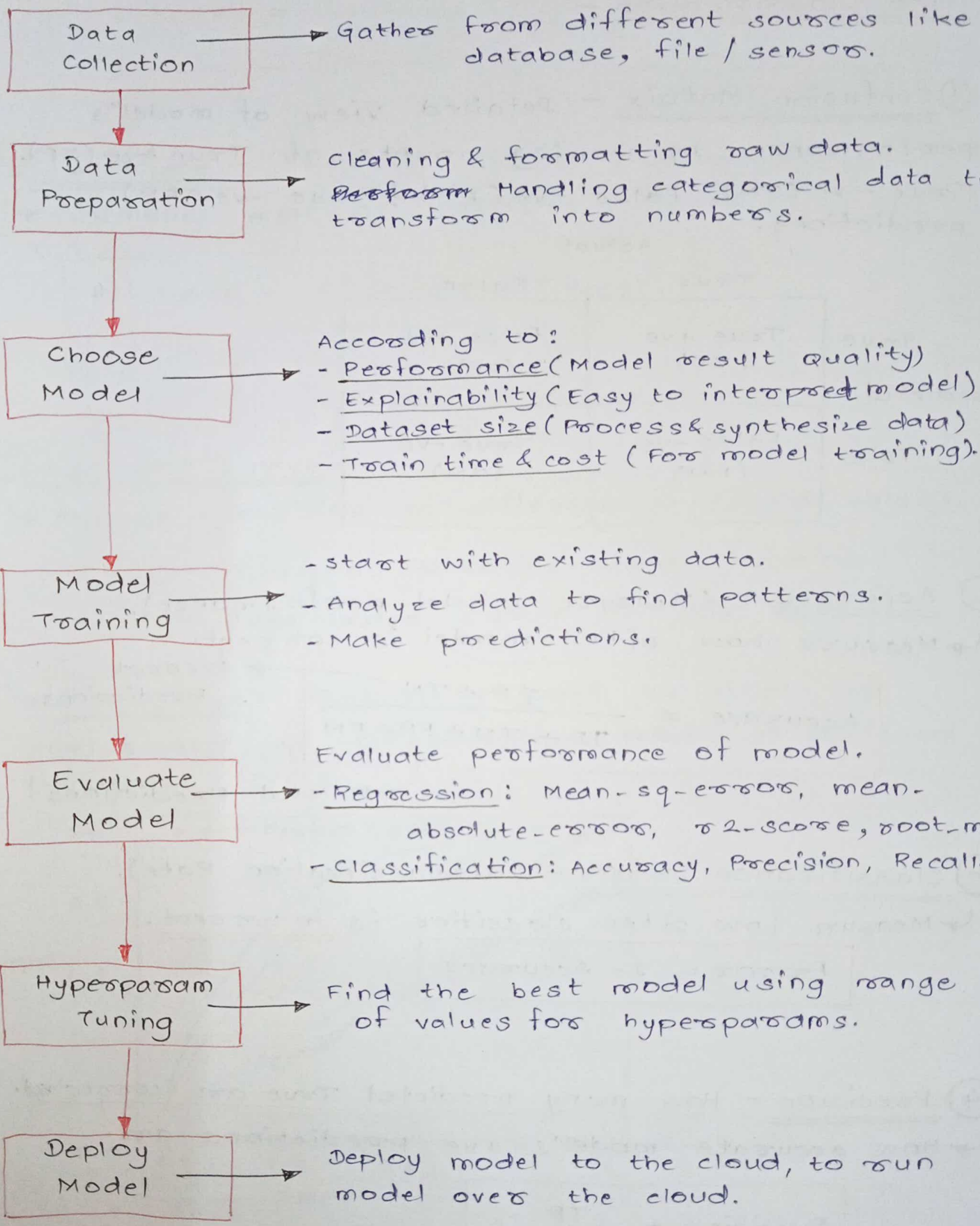
• Limitations of ML:

1. Data Quality
2. Data Quantity
3. Bias & fairness
4. Overfitting & Underfitting
5. Privacy & security
6. Interpretability

• Types of ML:-

Supervised	Unsupervised	Reinforcement
Calculate Outcome.	Discover patterns.	Learn series of action
Learn pattern betⁿ i/p & their labels.	Find pattern in i/p & divide in classes.	Find best reward for actions.
Build & train, then test model.	Build & train, then test model.	Train & test model simultaneously.
Classificaⁿ & Regression.	Clustering & Association	Exploration & Exploitation.
• Algos – ① Linear Regⁿ ② Decision Tree ③ KNN ⑤ SVM ④ Random Forest.	• Algos – ① K-mean clustering ② Agglomerative	• Algos – ① Q-Learning ② SARSA ③ Deep Q network
• Examples – ① Image Detection ② Population Growth Prediction.	• Examples – ① Cust Segmentation. ② Feature Elicitation. ③ Target Market.	• Examples – ① Driver less car. ② Self-navigating Cleaners.

Machine Learning workflow –



- **Model Evaluation Techniques in ML:**

- **For Classification –**

① **Confusion Matrix** – Detailed view of model's performance by showing counts of True +ve (TP), True -ve (TN), False +ve (FP) & False -ve (FN) predictions.

Predicted \ Actual	True	False
True	True +ve (TP)	False +ve (FP)
False	False -ve (FN)	True -ve (TN)

② **Accuracy** – (Measure model performance).

↳ Measure how often model is correct.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

TP + TN → Correct Predictions

TP + TN + FP + FN → Total Predictions

③ **Classification Error** – (Misclassification Rate).

↳ Measure how often classifier is incorrect.

$$\text{Error} = 1 - \text{Accuracy}$$

④ **Precision** – How many predicted True are corrected.

↳ How accurate model's +ve predictions are.

$$\text{Precision} = \frac{TP}{TP + FP}$$

⑤ **Recall** — How many actual True correctly predicted.

$$\text{Recall} = \frac{TP}{TP+FN}$$

⑥ **F1-Score** — Evaluate overall performance of model.

↳ Harmonic mean of Precision & Recall.

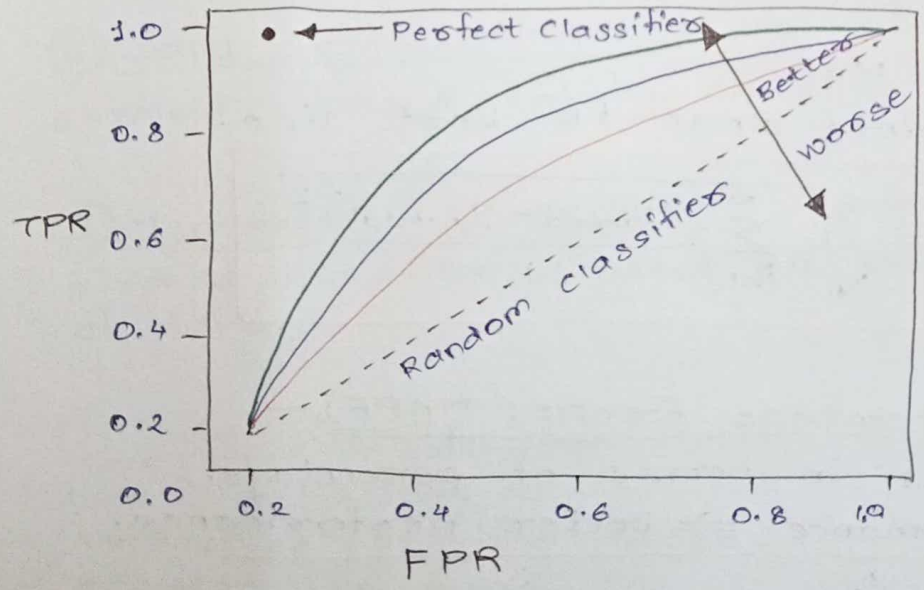
$$\text{F1-Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

⑦ **AUC-ROC curve** —

↳ Analyze classifier at different threshold values.

- **ROC** — Receiver Operating characteristics
- **AUC** — Area under curve of ROC Plot.

↳ **ROC curve** — Probability graph to show the performance of classification model at different threshold levels.



- **FPR (False +ve Rate)** $= \frac{FP}{FP+TN}$
- **TPR (True +ve Rate)** $= \frac{TP}{TP+FN}$

- For Regression –

① Mean Absolute Error –

↳ Analyze loss over whole dataset.

↳ Error: Difference betn predicted & actual values.

$$MAE = \sum_{i=1}^{N} \left| \frac{y_{pred} - y_{actual}}{N} \right|$$

② Mean Squared Error – Most commonly used.

↳ Used to calculate loss.

↳ Find error & square it, then find average over the whole dataset.

↳ Always +ve, bcz we square values.

↳ Small MSE, better performance.

$$MSE = \sum_{i=1}^{N} \frac{(y_{pred} - y_{actual})^2}{N}$$

③ Root Mean Squared Error – Evaluate performance.

↳ Indicates how much data points spread around the best line.

↳ Std deviation of MSE.

↳ Lower MSE – Points close to best line.

$$RMSE = \sqrt{MSE} = \sqrt{\sum_{i=1}^{N} \frac{(y_{pred} - y_{actual})^2}{N}}$$

④ Mean Absolute Percentage Error (MAPE) –

Express errors in terms of percentage.

Smaller performance => better performance.

$$MAPE = \frac{\sum_{i=1}^{N} \left(\frac{(y_{pred} - y_{actual})}{y_{actual}} \right)}{N} * 100\%$$

⊙ Exploratory Data Analysis (EDA) :-

↳ Methods to study & explore record sets to apprehend their predominant traits, discover patterns, locate outliers, & identify relationships between variables.

• Goals of EDA –

(1) Data cleaning: Handle missing values, duplicates, outliers & handle categorical data.

(2) Data Visualization: Visual techniques represent statistics graphically. Histograms, box plots, scatter plots, line plots, heat maps & bar charts to identify styles, trends & relationships within facts.

(3) Feature Engineering: contain scaling, normalization, binning, encoding variables.

(4) correlation & Relationships: Allow discover relationships & dependencies bet^n variables. correlation analysis, scatter-plots & pass-tabulations.

(5) Data segmentation – Divide info into significant segments based totally on sure standards / traits.

(6) Hypothesis Generation – Generating hypotheses/ studies questions based on preliminary exploration of data.

(7) Data Quality Assessment – Permits assessing nice & reliability of the info.
↳ Involve checking record integrity, consistency & accuracy to make info suitable for analysis.

⊙ EDA Implementation using Python:

- **Get Info about Dataset –**

```
>>> df.shape
```

O/p → (rows, cols)

```
>>> df.describe()
```

O/p → Give results of basic statistical computations on the numeric columns.

```
>>> df.info()
```

O/p → Information (feature name, no-null counts, dtype) of dataframe.

- **Changing Dtype (Object → Datetime) –**

```
>>> df['Date'] = pd.to_datetime(df['Date'])
```

- **Unique elements in dataset –**

```
>>> df.nunique()
```

O/p → No. of unique elements in each column.

⊙ Handling Missing Values –

- **isnull()** – Check any missing values in dataset.

```
>>> df.isnull().sum()
```

O/p → Return sum of np.nan (NULL) values in each column.

- fillna() – Fill value at NULL places.

```
>>> df['column'].fillna(value, inplace=True)
```

- replace() – Used to replace values in dataset.

```
>>> df['col'] = df['col'].replace(np.nan, value)
```

- dropna() – Drop records with NULL values.

```
>>> df['col'].dropna(axis=0, how='any')
```

- duplicated() – Check if duplicates present in dataset.

```
>>> df.duplicated()
```

O/p → Return Boolean (False → Original, True → Duplicate) for each row in Dataset.

```
>>> df.duplicated().sum()
```

O/p → Return total number of duplicate rows.

- drop_duplicates – Drop the duplicate rows.

```
>>> df.drop_duplicates(keep='first', inplace=True)
```

O/p → Keep first copy & remove all other duplicates.

Check count of values in each column –

- value_counts() – Return count of each unique value in the column.

```
>>> df['col'].value_counts()
```

Data Encoding –

↳ Encode categorical data into numerical values.

↳ One-hot Encoding / Label Encoding.

```
>>> from sklearn.preprocessing import LabelEncoder
>>> encoder = LabelEncoder()
>>> df['col'] = encoder.fit_transform(df['col'])
```

O/p → Assign num to each category, starting from 0.

Data Visualization :-

↳ Analyze data in the form of graphs/maps, easy understand trends/patterns.

① Histogram – count of numeric values present in regular intervals.

```
>>> sns.histplot(x = 'col', data = df)
```

② Boxplot –

```
>>> sns.boxplot (x = 'col1', y='col2', data = df)
```

③ pairplot() – Pairwise distributions in a dataset.

```
>>> sns.pairplot(df, hue = 'col', height = 2)
```

Handling Outliers:-

↳ Outlier - Out or range of normal objects.

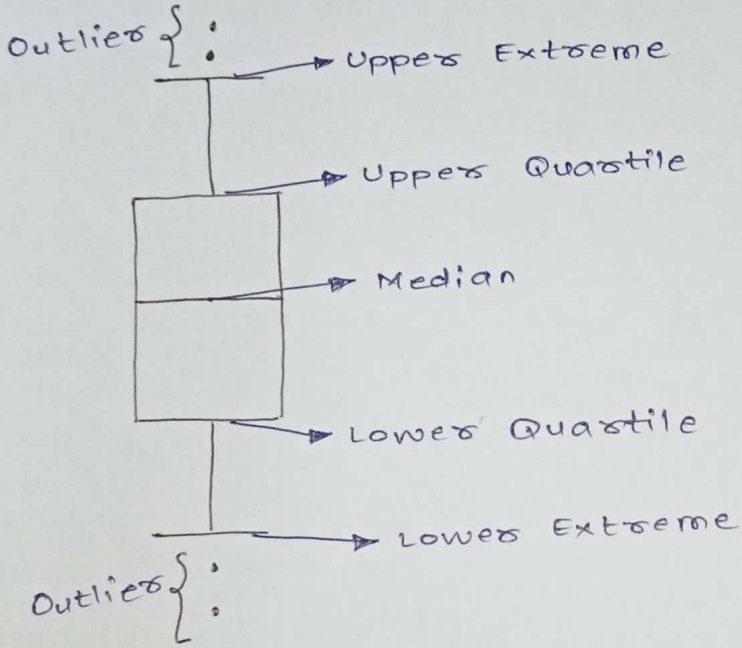
↳ Outlier detection analysis cla outlier mining.

↳ Many ways to detect, but same way for removal.

• Outlier Detection:

```
>>>sns.boxplot(x = 'col 1', data = df)
```

• Removing Outliers:



• IQR (Inter Quartile Range):

↳ Most commonly used technique.
This is Outlier base value.

IQR = Quartile 3 - Quartile 1

```
Q1 = np.percentile(df['col'], 25, method = 'midpoint')
Q3 = np.percentile(df['col'], 75, method = 'midpoint')
IQR = Q3 - Q1
```

• Upper & Lower bound:

upper = Q3 + 1.5 * IQR
lower = Q1 - 1.5 * IQR

Then, use upper & lower value to remove outliers.

```
upper_val = np.where(df['col'] >= upper)
lower_val = np.where(df['col'] <= lower)

#Remove Outliers
df.drop(upper_val[0], inplace = True)
df.drop(lower_val[0], inplace = True)
```