श्रवण,मनन,निदिध्यासन

Machine Learning Workflow

Vaibhav Zodge

7020616260

info.dnyanyog@gmail.com

https://www.dnyanyog.org

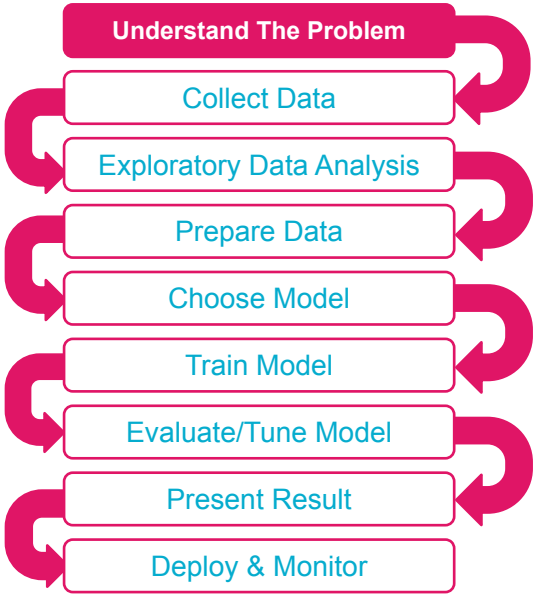
https://github.com/zodgevaibhav

Understand Steps of Machine Learning Steps

Machine Learning Process : Steps

1. What do I want to cook? (Understand The Problem)
2. Get ingredients (Collect Data)
3. Taste/check ingredients (Exploratory Data Analysis)
4. Wash, cut, prepare (Prepare Data)
5. Pick recipe (Choose Model)
6. Cook (Train Model)
7. Taste & adjust (Evaluate/Tune Model by using MSE, RMSE, etc…)
8. Serve (Present Result)
9. Improve recipe next time (Deploy & Monitor)

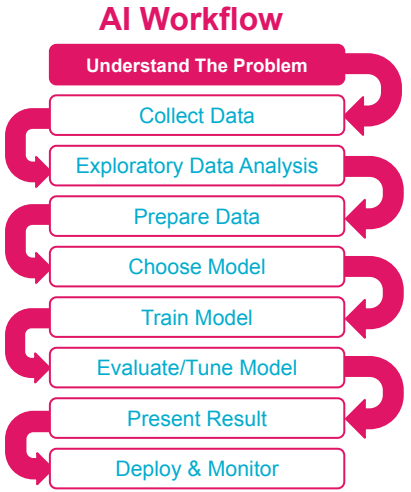
AI Workflow



Understand The Problem

Frame the Problem

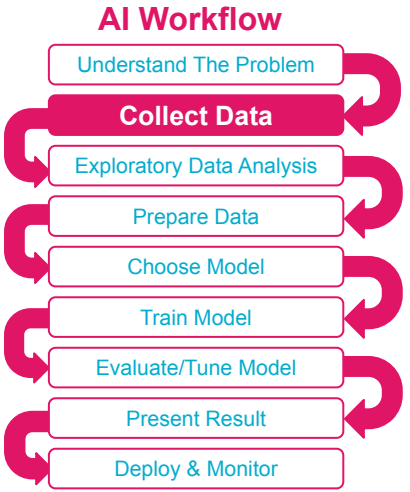
- The first question to ask "What problem we are going to solve"
- Building a model is probably not the end goal
- How does the company expect to use and benefit from this model?
  - Will this model will be final product or will be used under some product
  - Who will be the end users ? (Chatbot? Plugin? Service?)
- Problems statement should be clear and particular (should not be broad & vague)
  - Predicting next year sales vs next month sales
- Meaningful and useful for decision making
  - Which feature of application is being used
- Aligned with business goal
  - Cost Reduction, Customer Retention, Sales Increase (Product Capabilities increase)
- Knowing the objective is important because it will determine
  - How you frame the problem (Whether it is Regression, Classification problem or clustering)
  - Which algorithms you will select
  - Which performance measure you will use to evaluate your model. How do I know model is behaving right or wrong.
  - How much effort you will spend tweaking it



Collect The Data

- Gather data from multiple sources: databases, APIs, files, sensors, web scraping.
- Ensure you have **enough examples** for reliable training and testing.
- Record **metadata** (where data came from, collection process).
- Check data **quality and reliability** of sources.
- Note potential issues: missing values, bias, or inconsistencies.

  https://www.bbc.com/news/technology-45809919



Exploratory Data Analysis

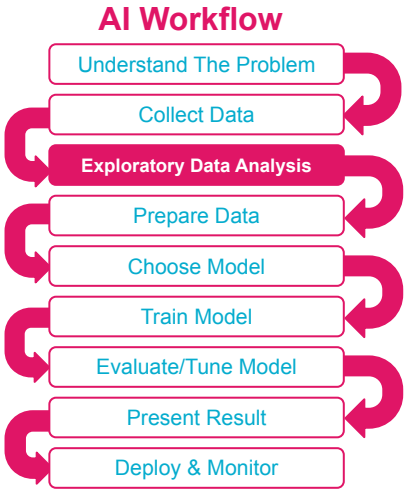
- Visualize the data
  - Convert data in to diagram, charts, graphs or tables
- Use libraries like matplotlib or seaborn for data visualization
- Understand the pattern and relationship
  - Data analysis to be done here from visualization
- Look for correlation
  - Is house price dependent on location ? Yes (Correlation found)
  - Is house price depend on #rooms ? Yes (Correlation found)
  - Is house price is depend on id ? No (No correlation)
- Experiment with attribute combinations

id	location	#rooms	prics
1	Kalyani nagar	2	1.2 cr
2	Sinhgad	3	1.2 cr
3	Magarpatta	2	4 cr
4	Shivaji Nagar	5	1.5 cr

Independent Column

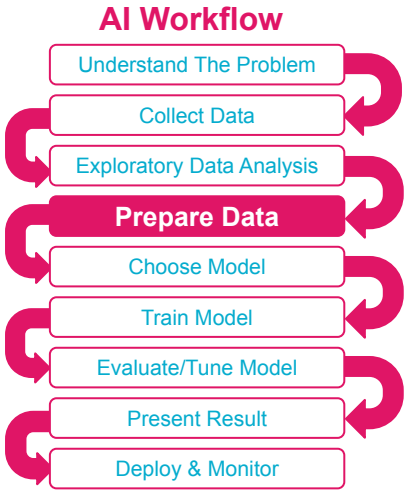
Dependent Column

Correlation



Prepare The Data

- Handle **missing data** (drop, fill, default).
- Encode **categorical variables** (Label Encoding, One-Hot Encoding/number-encoding).
- Normalize/standardize numerical features (scaling).
- Feature selection or creation (engineer new meaningful variables).
- Split dataset into **Training, Validation, and Test sets** for reliable evaluation.



Choose The Model

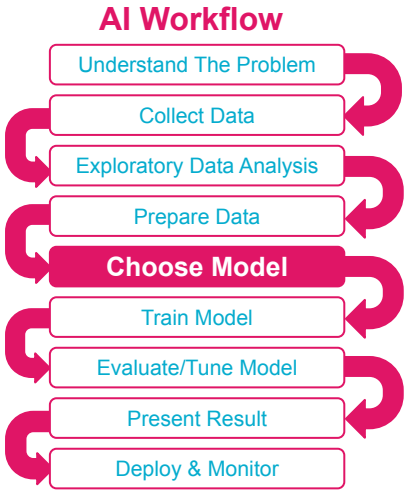
Select algorithm based on **problem type**:

- Regression → Linear, Polynomial, Ridge/Lasso.
- Classification → Logistic Regression, Decision Tree, Random Forest, SVM, Neural Net.
- Clustering → K-Means, DBSCAN, Hierarchical.

Consider dataset size and complexity.

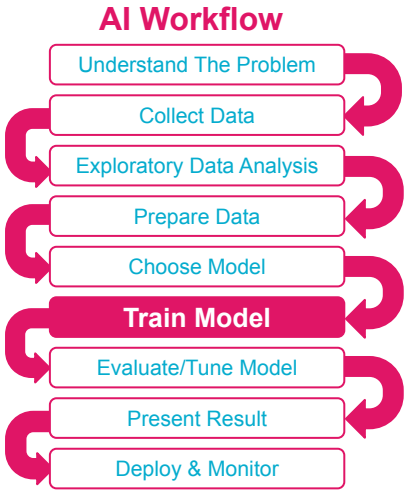
Balance between **simplicity and accuracy**.

Start with baseline/simple models before moving to complex ones.



Train The Model

- Feed training data into the selected algorithm.
- Use libraries/frameworks like **scikit-learn, TensorFlow, PyTorch**.
- Monitor training process (loss, accuracy).
- Avoid overfitting → compare training vs validation results.
- Save trained model for reuse/deployment.



Evaluate The Model

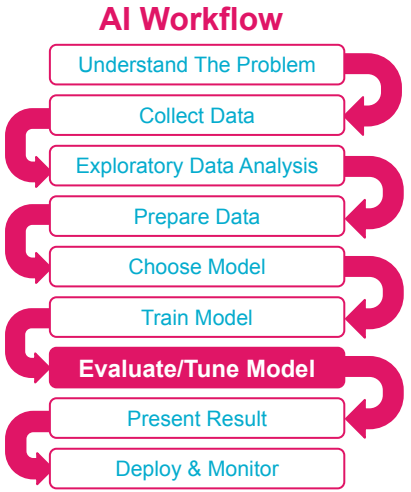
Evaluate on test/validation data using chosen metrics:

- Regression → MSE, RMSE, MAE, R^2.
- Classification → Accuracy, Precision, Recall, F1-score, ROC-AUC.

Compare results with baseline or simpler models.
Perform **hyperparameter tuning** (GridSearch, RandomSearch).

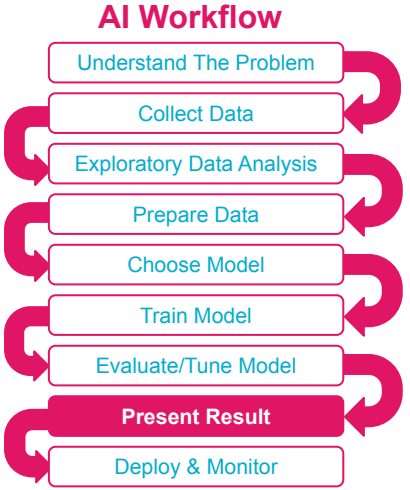
Apply **cross-validation** for more reliable performance estimates.

Refine features (add/remove/transform) for improvements.



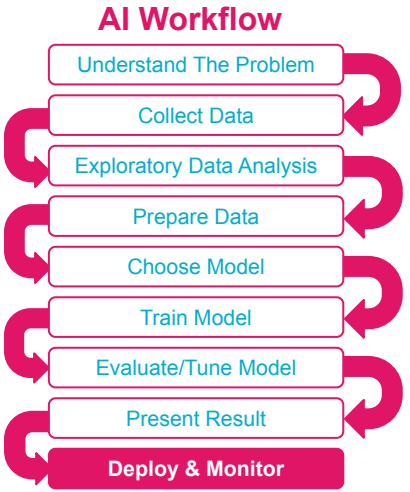
Present The Result

- Summarize model performance in **clear metrics**.
- Use **visualizations** (confusion matrix, error plots, ROC curve).
- Highlight **key insights** found during analysis.
- Document assumptions, challenges, and limitations.
- Communicate results in simple language to stakeholders.



Deploy & Monitor

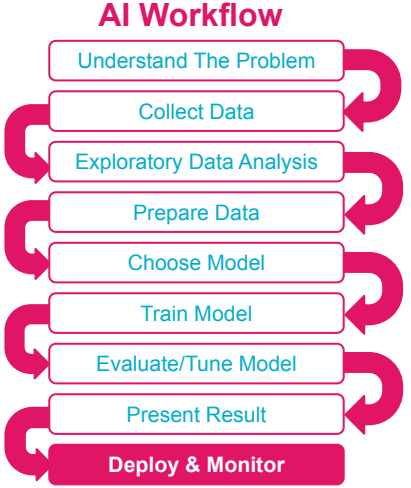
- Deploy model in production (as API, web service, embedded system).
- Monitor **real-world performance** (accuracy, drift, latency).
- Set up alerts for significant performance drops.
- Continuously **retrain model** with new data.
- Maintain proper **version control and documentation**.



Save the Model

Save the trained model in to binary file so that we don’t need to train again and again

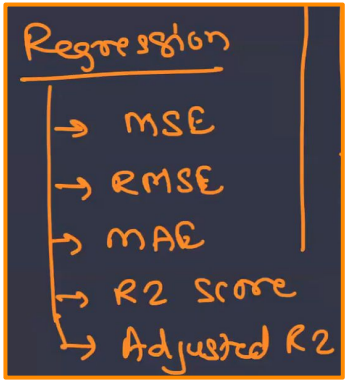
Format	Full Name / Origin	Why this name?	Best For	Pros	Cons
`.pkl` **(Pickle)**	**Pickle =** “to preserve” (like food pickling)	Named because it “preserves” Python objects by serializing them	Small ML models, quick experiments, prototyping	Built into Python, easy to use	Slow with large NumPy arrays, Python-only, unsafe with untrusted files
`.joblib`	From **Joblib** library (Job Library)	Designed for saving objects used in computational jobs (esp. NumPy, scikit-learn)	Scikit-learn models, NumPy-heavy ML	Faster & more efficient than pickle for large arrays	Still Python-only
`.onnx`	**Open Neural Network Exchange**	Created to exchange models between frameworks (Microsoft + Facebook project)	Production, cross-platform, deploying models in other languages	Framework-agnostic, works in C#, Java, C++, mobile	Conversion needed, not all models fully supported
`.h5` **(HDF5)**	**Hierarchical Data Format v5**	A scientific file format to store structured data in hierarchies	Deep learning models in Keras/TensorFlow	Stores architecture + weights + optimizer, widely used in DL	Mainly for Keras/TensorFlow, not for scikit-learn

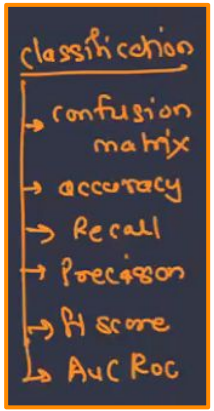


End to End Process : Looking at Big Picture

Select a Performance Measure

- Your next step is to select a performance measure
- A typical performance measure for regression problems is the Root Mean Square Error (RMSE)
- It gives an idea of how much error the system typically makes in its predictions, with a higher weight for large errors





End to End Process : Discover and Visualize the Data to Gain Insights

End to End Process : Prepare the Data for Machine Learning Algorithms

Data Cleaning

Process of cleaning the data set to prepare it for ML algorithm

Steps

Check for the missing data

Check for wrong data types

Add features if needed

Remove unwanted features

Feature Scaling

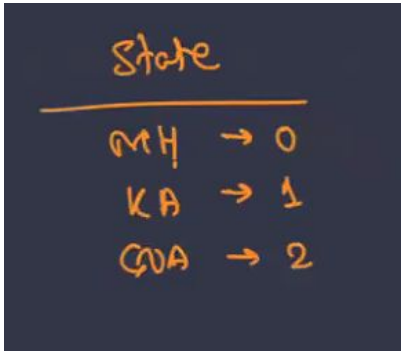
ML algorithms don't perform well when the input numerical attributes have very different scale

Scale the features to bring all of them to a single scale

Handle categorical / text data

Use transformers to convert "categorical or textual" to numerical. We must use numerical data

We can use encoders to convert textual data in to number



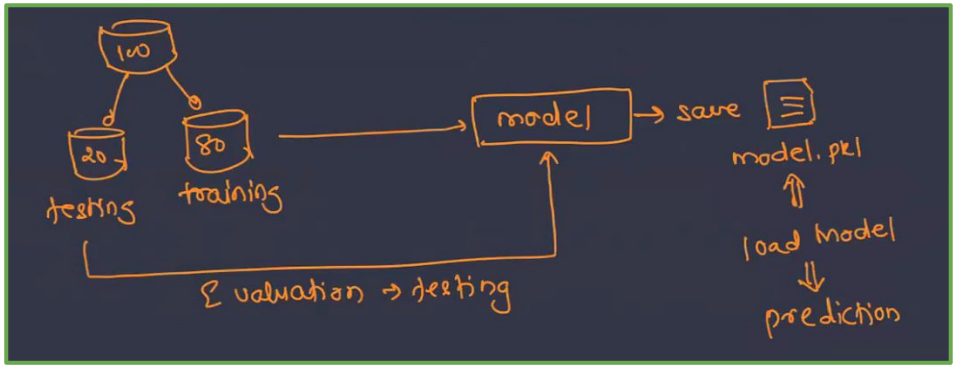
End to End Process : Select and Train a Model

Training the model using train data set

- Create a model using selected algorithm
- Save the model for future use

Evaluation the model

- Evaluate the model to see if there is any chance to improve the accuracy
- Techniques
  - Cross Validation



Understand various equations

End to End Process : Fine-Tune Your Model

Grid Search

- One option would be to fiddle with the hyperparameters manually, until you find a great combination of hyperparameter values
- This would be very tedious work, and you may not have time to explore many combinations
- You can also automate this process using libraries like sci-kit

Randomized Search

- The grid search approach is fine when you are exploring relatively few combinations
- But when the hyperparameter search space is large, it is often preferable to use randomized search

Ensemble Methods

- Another way to fine-tune your system is to try to combine the models that perform best
- The group (or “ensemble”) will often perform better than the best individual model, especially if the individual models make very different types of errors.

Analyze the Best Models and Their Errors

Evaluate Your System on the Test Set

End to End Process : Launch, Monitor, and Maintain Your System

Deploy the application for the end users

Monitor the application's performance

If the data keeps evolving, update your datasets and retrain your model regularly

- More the data better the accuracy

You should probably automate the whole process as much as possible (using MLOps)

- Collect fresh data regularly and label it
- Write a script to train the model and fine-tune the hyperparameters automatically.
- This script could run automatically, for example every day or every week, depending on your needs.
- Write another script that will evaluate both the new model and the previous model on the updated test set, and deploy the model to production if the performance has not decreased (if it did, make sure you investigate why)

Predict Price (2D data for training) : Linear Regression

Problem Statement :

Predict the price on the basis of mileage and age of car

To find formula we need some data to analyse and find formula (train the mode)

Understand Data & Code:

Here Age and Mileage is the independent data

Unlike previous example here we are giving two variables for prediction

Price is the only dependent data

As seen in code we dropped price column and given rest column as feature

While prediction we must need to give two values as we trained model using two values

Age	Mileage	Price
7	16394	21547.3
4	74032	21447.4
13	8890	16582.5
11	46606	17364.7
8	92313	16879.35

```
import pandas as pd
from sklearn.linear_model import LinearRegression

df = pd.read_csv("car_price_data.csv")

# Separate the features (independent variables) and
the target (dependent variable)
x = df.drop('Price', axis=1)  # Features: all columns
except 'Price'
y = df['Price']  # Target: 'Price' column

model = LinearRegression()
model.fit(x, y)

predictions = model.predict([[2,20000]])
print(predictions)
```

Ref: https://github.com/zodgevaibhav/gen-ai-learning/tree/main/3.ModelToFindPriceBasedOnMileageAndAgeOfCar

How Prediction works ? (Predict Salary (1D data for training) : Linear Regression)

We used Linear Regression Algorithm :

Linear Regression is a Supervised Learning Algorithm used to predict the continuous variables

Linear Regression :

When output change with constant/linear rate of input

Salary is directly proportional to years of experience

Considered change in year of experience make change in salary at constant rate

Whenever we want to predict the value on such linear rate of change then we should use Linear Regression Algorithm

Examples :

Salary Prediction on the basis of experience

Car Price prediction on the basis of Age and Mileage

How price prediction on the basis of Squarefoot

Supervised Learning :

Supervised learning meaning learning use independent data.

Independent Data guide/supervise the model for prediction hence called SL

Continuous Variables:

Continuous variable is type of variable which can take any numeric value

It just mean any number for any given range

1. Data Analysis & Preprocessing

Understanding the Data:

- The model first **analyzes the dataset** to understand the relationship between dependent (target) and independent (predictor) variables.
- Example: **Salary Prediction** → Years of Experience (X) vs. Salary (Y).

Handling Missing & Outlier Values:

- Missing values are **handled (imputation, removal, etc.)**.
- Outliers are **detected and analyzed** as they might impact the regression model.

Checking for Linearity:

- A **scatter plot** is used to see if there is a linear relationship.

2. Finding the Best-Fit Line (Model Training Process)

Mathematical Representation:

- The model assumes the relation follows the equation of a line: $Y=mX+cY = mX + cY=mX+c$ where:
  - YYY = Dependent variable (Salary)
  - XXX = Independent variable (Years of Experience)
  - mmm = Slope (Rate of increase in salary per year)
  - ccc = Intercept (Base salary with 0 experience)

Cost Function (Error Measurement):

- To find the best-fit line, the model calculates **errors (differences between actual and predicted values).**
- The most common cost function used is **Mean Squared Error (MSE):**

Finding the Optimal Parameters (m & c):

- The model **adjusts the values of mmm (slope) and ccc (intercept)** to minimize the MSE.
- This process is called **optimization**, typically done using **Gradient Descent** or **Ordinary Least Squares (OLS)**.

How Prediction works ? (Predict Salary (1D data for training) : Linear Regression)

1. Data Analysis & Preprocessing

Understanding the Data:

- The model first **analyzes the dataset** to understand the relationship between dependent (target) and independent (predictor) variables.
- Example: **Salary Prediction** → Years of Experience (X) vs. Salary (Y).

Handling Missing & Outlier Values:

- Missing values are **handled (imputation, removal, etc.)**.
- Outliers/issues are **detected and analyzed** as they might impact the regression model.

2. Finding the Best-Fit Line (Model Training Process)

Mathematical Representation:

- The model assumes the relation follows the equation of a line: $Y = mX+c$
  where:
  - Y = Dependent variable (Salary)
  - X = Independent variable (Years of Experience)
  - m = Slope (Rate of increase in salary per year)
  - c = Intercept (Base salary with 0 experience)
- Model file the slope value and constant (intercept) from the data trend.
- Then using mX + C formula it calculates the salary.
  Assume calculated values from trends are : m (slope) = 10,000 and c (Intercept) = 30,000
  Calculate the salary for Years of Experience **5**

  $Y = (10,707 \times 5) + 20,000 = 73,538$

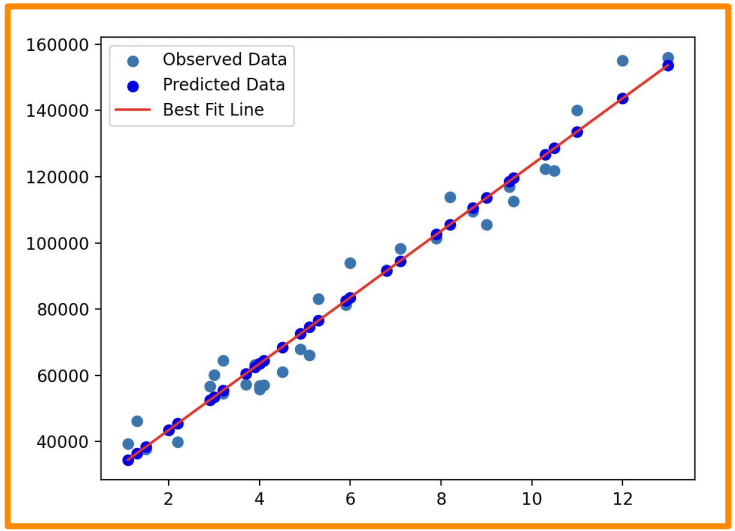
Visualize Predicted Values and how the best fit line shows

As we seens model analyse the given data, find the relation and draw the best fit line

Best fit line is representation of the prediction

Hence if we want to visualize the best fit line then draw scatter on input and predicted o/p

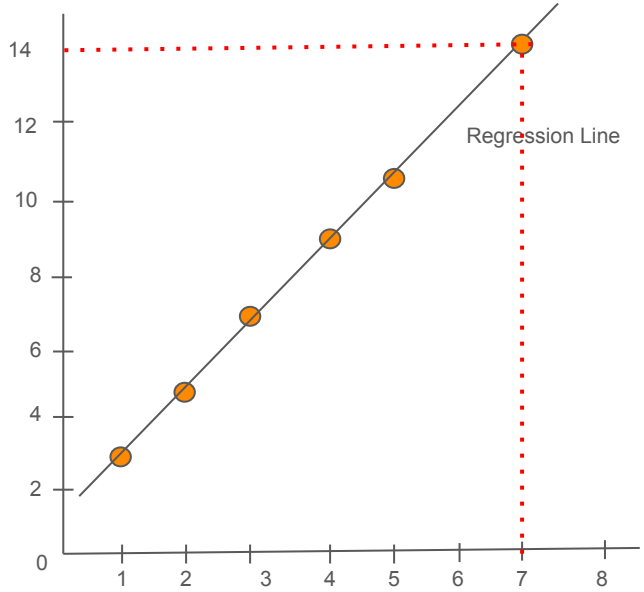
Also if we draw plot (line) on predicted values then will get best fit line




```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression

# Load the dataset from a CSV file
df = pd.read_csv("salary_data.csv")

# Separate the features (independent variables) and the target (dependent
variable)
x = df.drop('Salary', axis=1)
y=df['Salary']

# Initialize the Linear Regression model
model = LinearRegression()

# Fit the model to the data
model.fit(x,y)

# Since we have model which is trained, we can plot the BEST FIT REGRESSION LINE
plt.scatter(df['Experience'], df['Salary'],label='Observed Data')
plt.scatter(df['Experience'], model.predict(x), color='blue',label='Predicted
Data')
plt.plot(df['Experience'], model.predict(x), color='red',label='Best Fit Line')
plt.legend()
plt.show()
```

Understand Linear Regression

x	y
1	3
2	5
3	7
4	9
5	11
7	?



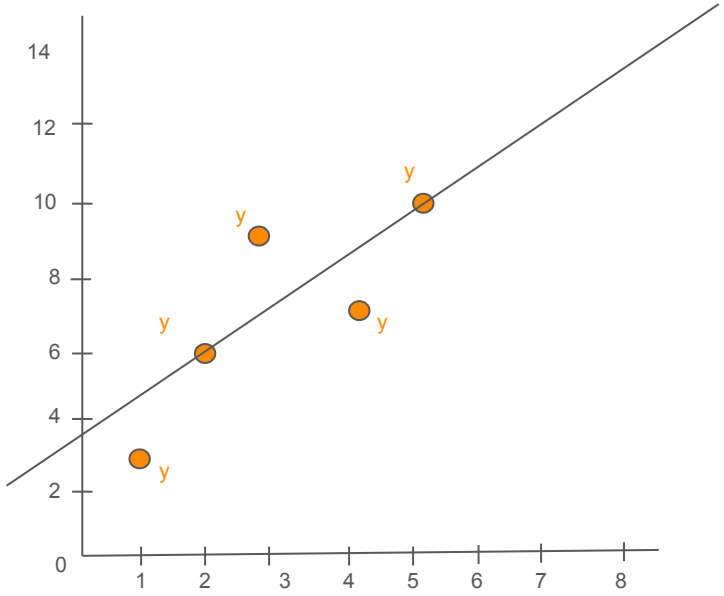
Understand Linear Regression

Many time will not get Linear Straight Line

Hence we need to validate if my drawn line is near to best fit line ?

y

x	y
1	3
2	6
3	9
4	7
5	10
7	?



Confidence on Formula

Let's go back to our example…

When a = 2 then b = 4
When a = 3 then b = 9
When a = 4 then b = 4
When a = 5 then b = 5
When a = 6 then b = 36
When a = 7 then b = ?
When a = 8 then b = ?

Formula we derived out of above dependent and independent variable is

b = a * a

Independent data

Depender data

a	b
2	4
3	9
4	16
5	25
6	36
7	49
8	64