ARTIFICIAL INTELLIGENCE: STUDY MATERIAL

CLASS XII

LEVEL 3: AI INNOVATE (UNIT 1 – UNIT 3)

TEACHER INSTRUCTION MANUAL

INDEX

Unit 1: Capstone Project

Title: Capstone Project	**Approach:** Hands on, Team Discussion, Web search**,** Case studies
Summary: The final project of an academic program, typically integrating all of the learning from the program is called the Capstone Project. Here students are made to look at real world examples and situations, exchange their points of view based on experiences and discuss potential solutions to the problem.	
Objectives: **1.** Students should apply their learning in solving real world problems **2.** Students should be comfortable in expressing their solution in non-technical words **3.** Students should be in a position to choose and apply right algorithm to solve the problem	
Key Concepts: AI Project Cycle , Model validation , RMSE , MSE , MAPE	

A capstone project is a project where students must research a topic independently to find a deep understanding of the subject matter. It gives an opportunity for the student to integrate all their knowledge and demonstrate it through a comprehensive project**.**

So, without further ado, let’s jump straight into some **Capstone project ideas** that will strengthen your base

1. Stock Prices Predictor
2. Develop A Sentiment Analyzer
3. Movie Ticket Price Predictor
4. Students Results Predictor
5. Human Activity Recognition using Smartphone Data set
6. Classifying humans and animals in a photo

The list can is huge but these are some simple projects, which you can consider to pick up to develop.

1. Understanding The Problem

Artificial Intelligence is perhaps the most transformative technology available today. At a high level, every AI project follows the following six steps:

1) Problem definition i.e. Understanding the problem

2) Data gathering

3) Feature definition

4) AI model construction

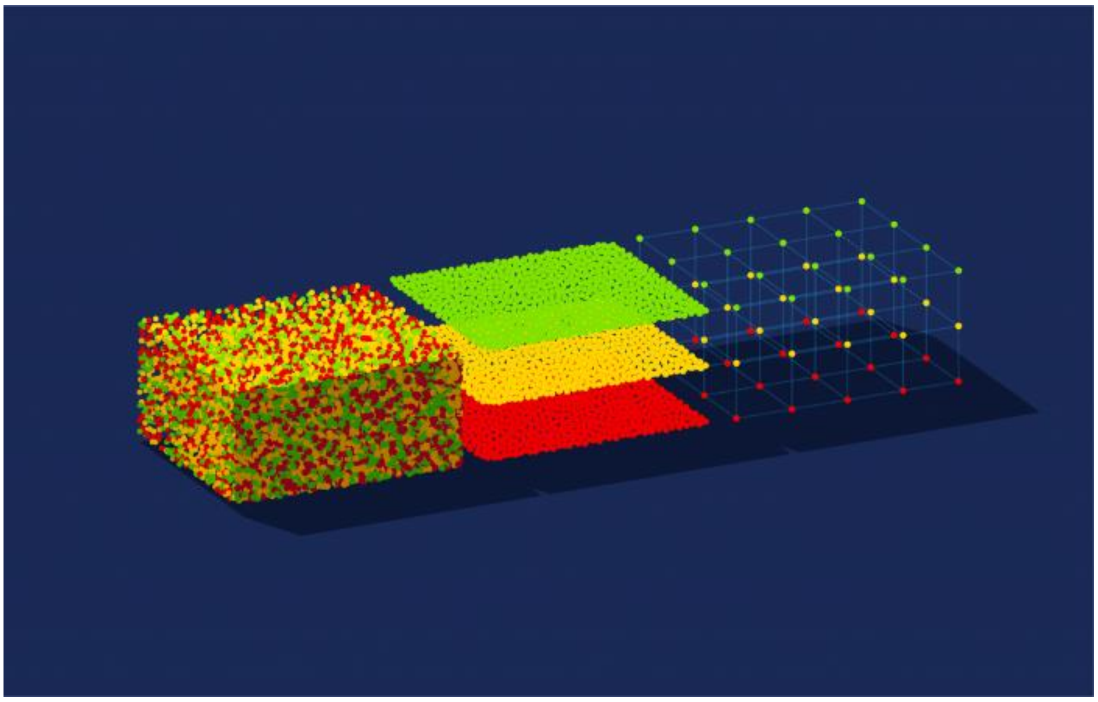
5) Evaluation & refinements

6) Deployment

In this section, I will share the best practices for the first step: "*understanding the problem*".

Begin formulating your problem by asking yourself this simple — *is there a pattern*? The premise that underlies all Machine Learning disciplines is that there needs to be a pattern. If there is no pattern, then the problem cannot be solved with AI technology. It is fundamental that this question is asked before deciding to embark on an AI development journey.

(If it is believed that there is a pattern in the data, then AI development techniques may be employed , else Don't apply AI techniques to solve the problem.



If it is believed that there is a pattern in the data, then AI development techniques may be employed. Applied uses of these techniques are typically geared towards answering five types of questions, all of which may be categorized as being within the umbrella of predictive analysis:

1) **Which category?** (Classification)

2) **How much or how many?** (Regression)

3) **Which group?** (Clustering)

4) **Is this unusual?** (Anomaly Detection)

5) **Which option should be taken?** (Recommendation)

It is important to determine *which* of these questions you're asking, and how answering it helps you solve your problem.

Project 1:

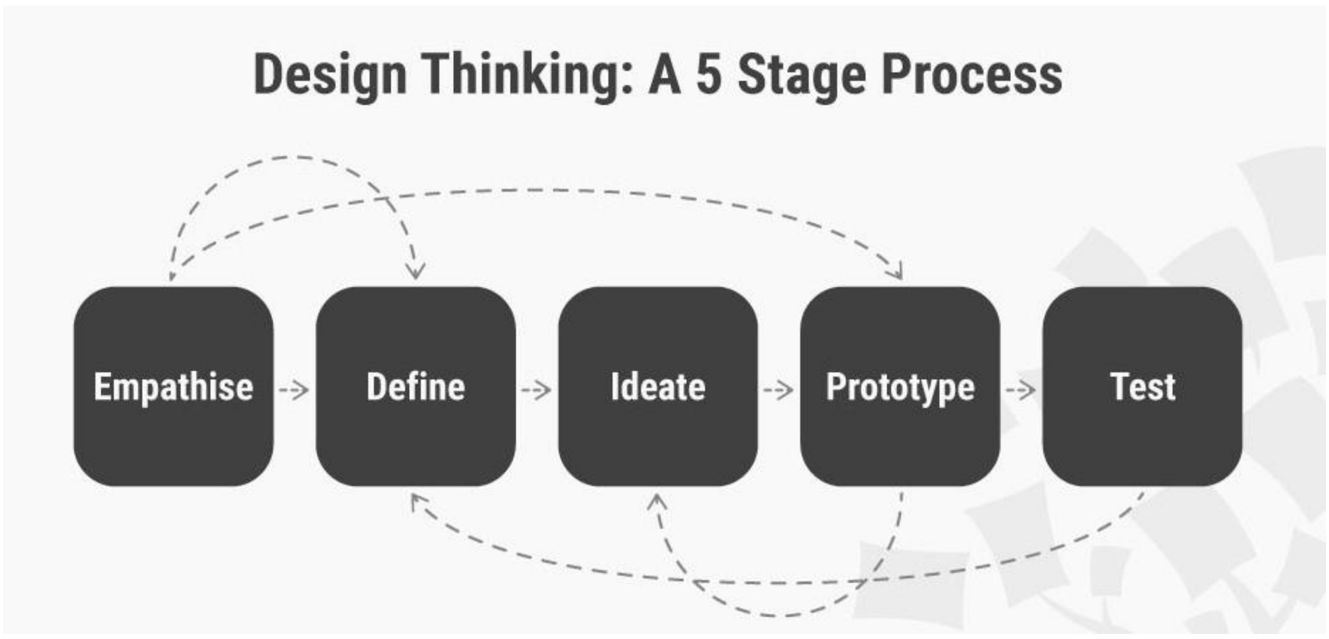
Form a team of 4-5 students. And submit a detailed report on the most critical problems and how AI can assist in addressing those problems. Report should include description of the problem and the proposed way in which AI can solve the problem.

1. Agriculture in India
2. School Education in India
3. Healthcare in India

2. Decomposing The Problem Through DT Framework

Design Thinking is a design methodology that provides a solution-based approach to solving problems. It's extremely useful in tackling complex problems that are ill-defined or unknown.

The five stages of Design Thinking are as follows: Empathize, Define, Ideate, Prototype, and Test.



Real computational tasks are complicated. To accomplish them you need to **break down the problem into smaller units before coding.**

Problem decomposition steps

1. Understand the problem and then restate the problem in your own words
    - Know what the desired inputs and outputs are
    - Ask questions for clarification (in class these questions might be to your instructor, but most of the time they will be asking either yourself or your collaborators)
2. Break the problem down into a few large pieces. Write these down, either on paper or as comments in a file.
3. Break complicated pieces down into smaller pieces. Keep doing this until all of the pieces are small.
4. Code one small piece at a time.
    1. Think about how to implement it
    2. Write the code/query
    3. Test it… on its own.
    4. Fix problems, if any

Example 1: Calculate the volume of a bunch of books

Data

```
length    width    height
   1        2        3
   2        4        3
```

1. Calculate the volume of a book [a. Function]
2. Run this calculation on all books [a. Loop]

Example 2: Imagine that you want to create your first app. This is a complex problem. How would you decompose the task of creating an app?

To decompose this task, you would need to know the answer to a series of smaller problems:

- what kind of app you want to create?
- what will your app will look like?
- who is the target audience for your app?
- what will the graphics will look like?
- what audio will you include?
- what software will you use to build your app?
- how will the user navigate your app?
- how will you test your app?

This list has broken down the complex problem of creating an app into much simpler problems that can now be worked out. You may also be able to get other people to help you with different individual parts of the app. For example, you may have a friend who can create the graphics, while another will be your test the app.

Example 3: (For Advance learners)

Decompose Time Series Data into Trend

Time series decomposition involves thinking of a series as a combination of level, trend, seasonality, and noise components. Decomposition provides a useful abstract model for thinking about time series generally and for better understanding problems during time series analysis and forecasting.

These components are defined as follows:

- **Level**: The average value in the series.
- **Trend**: The increasing or decreasing value in the series.
- **Seasonality**: The repeating short-term cycle in the series.
- **Noise**: The random variation in the series.

Example Airline Passengers Data set

The Airline Passengers dataset describes the total number of airline passengers over a period of time.

The units are a count of the number of airline passengers in thousands. There are 144 monthly observations from 1949 to 1960.

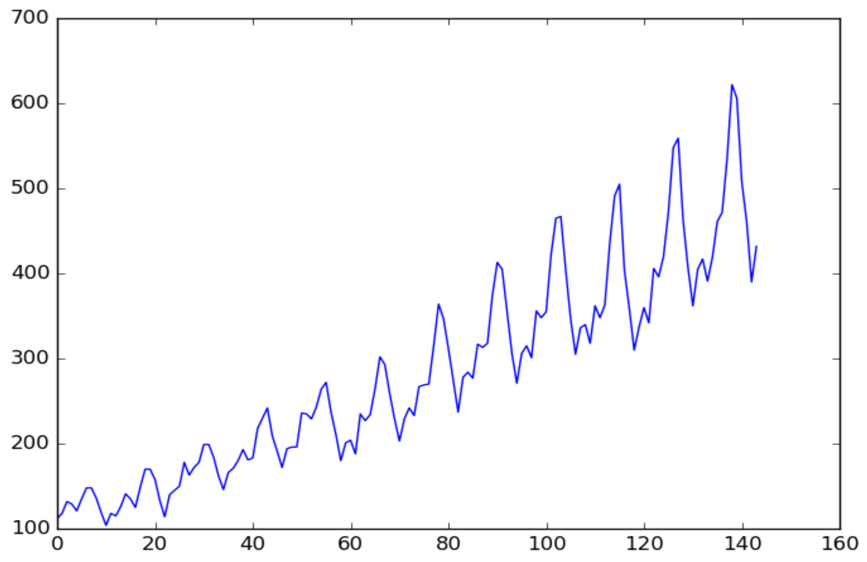
- Download the dataset.

Download the dataset to your current working directory with the filename “*airline-passengers.csv*“.

First, let’s graph the raw observations.

```
from pandas import read_csv

from matplotlib import pyplot

series = read_csv('airline-passengers.csv', header=0, index_col=0)

series.plot()

pyplot.show()
```

Reviewing the line plot, it suggests that there may be a linear trend, but it is hard to be sure from eye-balling. There is also seasonality, but the amplitude (height) of the cycles appears to be increasing, suggesting that it is multiplicative.



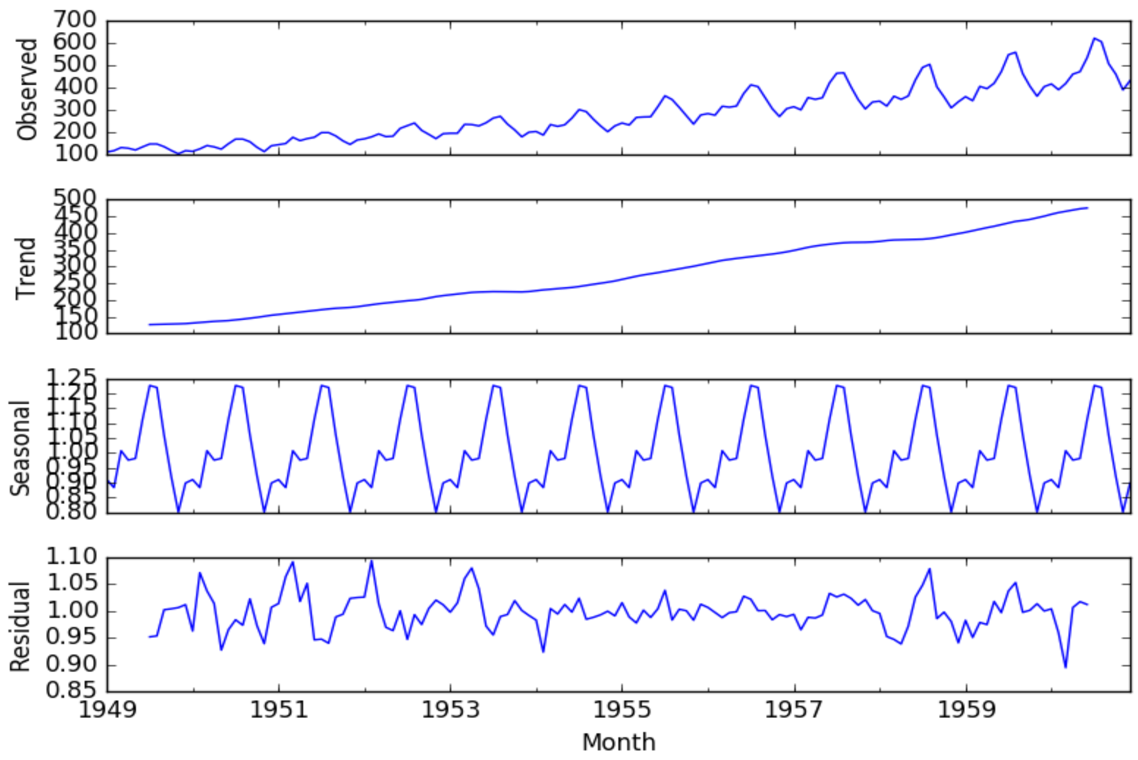
We will assume a multiplicative model.

The example below decomposes the airline passenger's dataset as a multiplicative model.

```
from pandas import read_csv

from matplotlib import pyplot

from statsmodels.tsa.seasonal import seasonal_decompose

series = read_csv('airline-passengers.csv', header=0, index_col=0)

result = seasonal_decompose(series, model='multiplicative')

result.plot()

pyplot.show()
```

Running the example plots the observed, trend, seasonal, and residual time series.

We can see that the trend and seasonality information extracted from the series does seem reasonable. The residuals are also interesting, showing periods of high variability in the early and later years of the series.



(Multiplicative Decomposition of Airline Passenger Dataset)

3.Analytic Approach

Those who work in the domain of AI and Machine Learning solve problems and answer questions through data every day. They build models to predict outcomes or discover underlying patterns, all to gain insights leading to actions that will improve future outcomes.

It is the ‘ Foundational Methodology of Data Science’ (https://www.ibmbigdatahub.com/blog/why-we-need-methodology-data-science) and it has 10 stages -

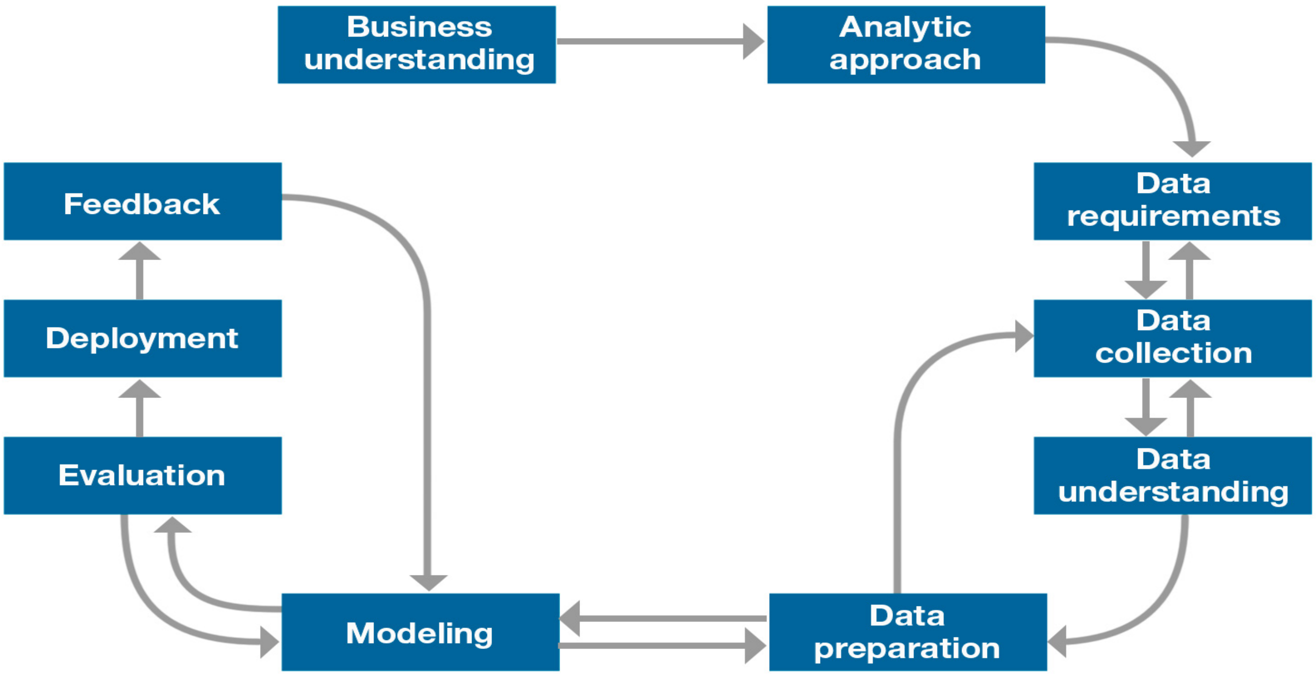
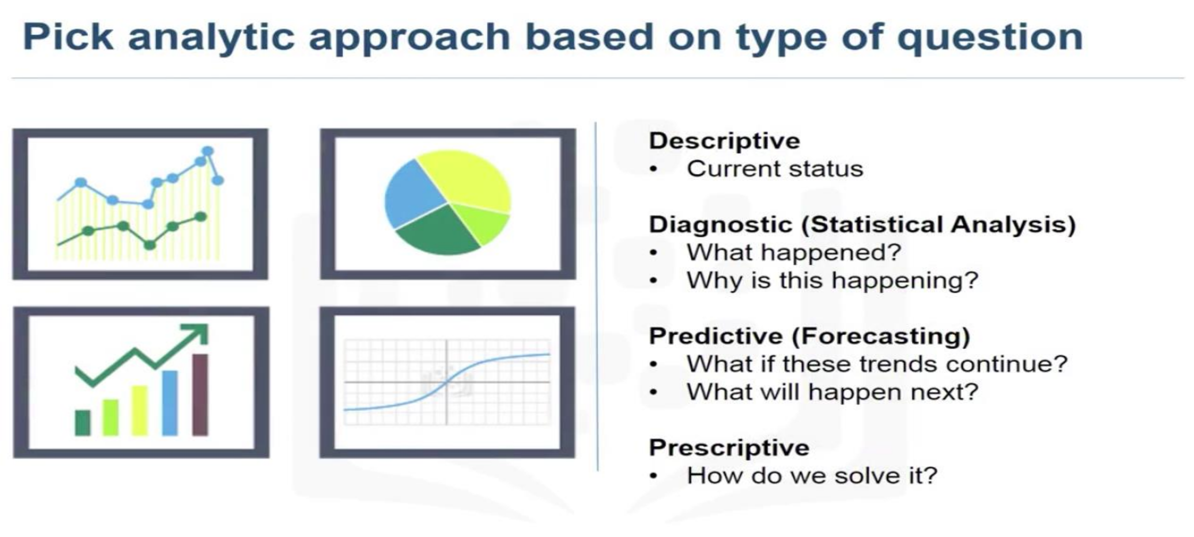


Figure 1. Foundational methodology for data science.

Every project, regardless of its size, starts with business understanding, which lays the foundation for successful resolution of the business problem. The business sponsors needing the analytic solution play the critical role in this stage by defining the problem, project objectives and solution requirements from a business perspective. And, believe it or not—even with nine stages still to go—this first stage is the hardest.

After clearly stating a business problem, the data scientist can define the analytic approach to solving it. Doing so involves expressing the problem in the context of statistical and machine learning techniques so that the data scientist can identify techniques suitable for achieving the desired outcome.

Selecting the right analytic approach depends on the question being asked. Once the problem to be addressed is defined, the appropriate analytic approach for the problem is selected in the context of the business requirements. This is the second stage of the data science methodology.



Pick analytic approach based on type of question

Descriptive
- Current status

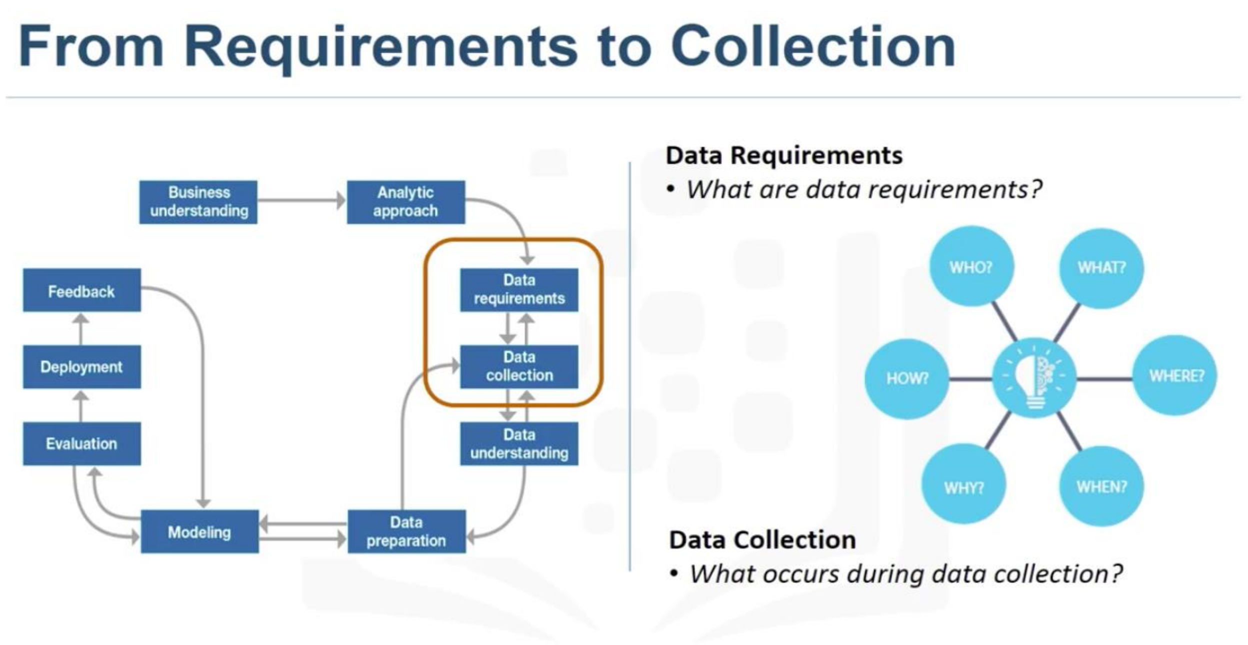
Diagnostic (Statistical Analysis)
- What happened?
- Why is this happening?

Predictive (Forecasting)
- What if these trends continue?
- What will happen next?

Prescriptive
- How do we solve it?

- If the question is to determine **probabilities of an action**, then a **predictive model** might be used.
- If the question is to **show relationships**, a **descriptive approach** maybe be required.
- **Statistical analysis** applies to problems that require counts: if the question requires a yes/ no answer, then a classification approach to predicting a response would be suitable.

4. Data Requirement



From Requirements to Collection

Data Requirements
- *What are data requirements?*

Data Collection
- *What occurs during data collection?*

If the problem that needs to be resolved is "a recipe", so to speak, and data is "an ingredient", then the data scientist needs to identify:

1. which ingredients are required?
2. how to source or the collect them?
3. how to understand or work with them?
4. and how to prepare the data to meet the desired outcome?

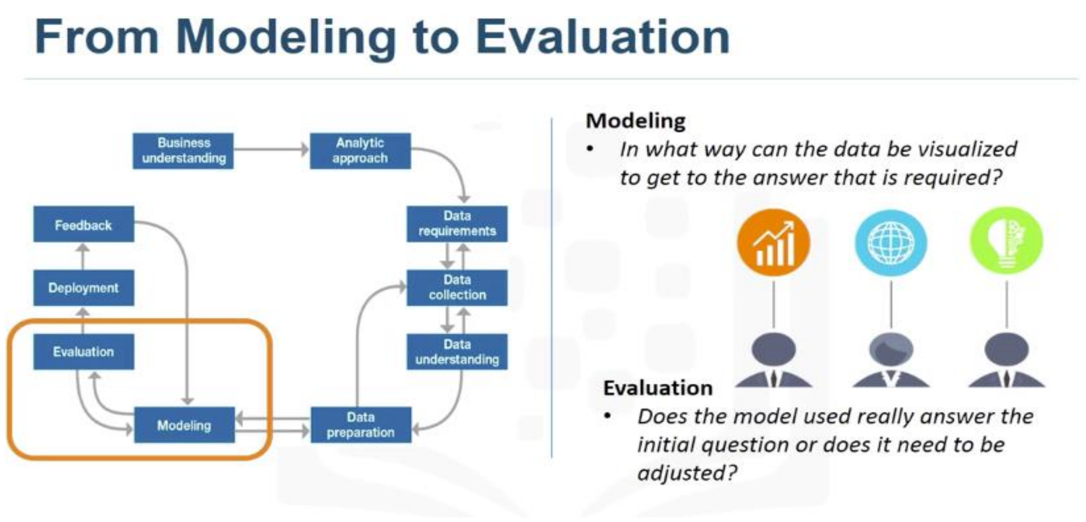
Prior to undertaking the data collection and data preparation stages of the methodology, it's vital to define the data requirements for decision-tree classification. This includes **identifying the necessary data content, formats and sources for initial data collection.**

In this phase the data requirements are revised and decisions are made as to whether or not the collection requires more or less data. Once the data ingredients are collected, the data scientist will have a good understanding of what they will be working with.

Techniques such as **descriptive statistics and visualization** can be applied to the data set, to assess *the content, quality, and initial insights* about the data. Gaps in data will be identified and plans to either fill or make substitutions will have to be made.

In essence, the ingredients are now sitting on the cutting board.

5. Modeling Approach


From Modeling to Evaluation
Business understanding
Analytic approach
Data requirements
Data collection
Data understanding
Data preparation
Modeling
Evaluation
Deployment
Feedback
Modeling
In what way can the data be visualized to get to the answer that is required?
Evaluation
Does the model used really answer the initial question or does it need to be adjusted?

Data Modeling focuses on developing models that are **either descriptive or predictive**.

- An example of a descriptive model might examine things like: if a person did this, then they're likely to prefer that.
- A predictive model tries to yield yes/no, or stop/go type outcomes. These models are based on the analytic approach that was taken, either statistically driven or machine learning driven.

The data scientist will use a **training set** for predictive modelling. A training set is a set of historical data in which the outcomes are already known. The training set acts like a gauge to determine if the model needs to be calibrated. In this stage, the data scientist will **play around with different algorithms** to ensure that the variables in play are actually required.

The success of data compilation, preparation and modelling, depends on the understanding of the problem at hand, and the appropriate analytical approach being taken. The data supports the answering of the question, and like the quality of the ingredients in cooking, sets the stage for the outcome.

Constant refinement, adjustments and tweaking are necessary within each step to ensure the outcome is one that is solid. The framework is geared to do **3 things**:

- First, understand the question at hand.
- Second, select an analytic approach or method to solve the problem.
- Third, obtain, understand, prepare, and model the data.

The end goal is to move the data scientist to a point where a data model can be built to answer the question.

6. How to validate model quality

6.1 Train-Test Split Evaluation

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values.

The train-test procedure is appropriate when there is a sufficiently large dataset available.

How to Configure the Train-Test Split

The procedure has one main configuration parameter, which is the size of the train and test sets. This is most commonly expressed as a percentage between 0 and 1 for either the train or test datasets. For example, a training set with the size of 0.67 (67 percent) means that the remainder percentage 0.33 (33 percent) is assigned to the test set.

There is no optimal split percentage.

You must choose a split percentage that meets your project’s objectives with considerations that include:

- Computational cost in training the model.
- Computational cost in evaluating the model.
- Training set representativeness.
- Test set representativeness.

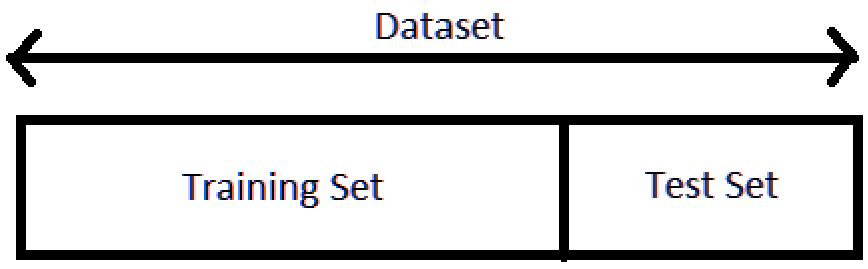
Nevertheless, common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

Now that we are familiar with the train-test split model evaluation procedure, let’s look at how we can use this procedure in Python.

Example 1: Training and Test Data in Python Machine Learning

As we work with datasets, a machine learning model works in two stages. We usually split the data around 20%-80% between testing and training stages. Under supervised learning, we split a dataset into a training data and test data in Python ML.



a. Prerequisites for Train and Test Data

We will need the following Python Libraries for this tutorial:

- Pandas
- Sklearn

We can install these with pip

1. pip install pandas
2. pip install sklearn

We use pandas to import the dataset and sklearn to perform the splitting. You can import these packages as:

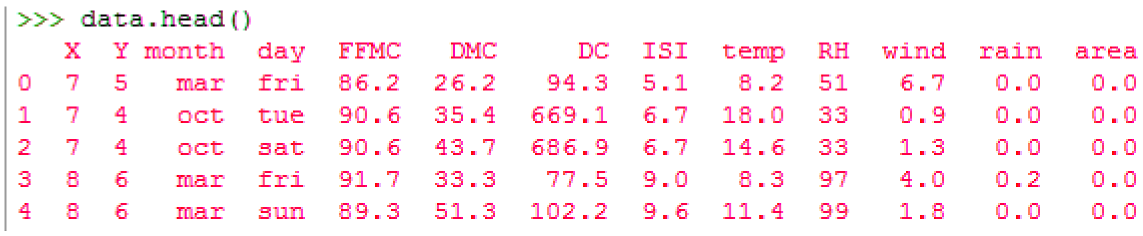
1. >>> import pandas as pd
2. >>> from sklearn.model_selection import train_test_split
3. >>> from sklearn.datasets import load_iris

Following are the process of Train and Test set in Python ML. So, let's take a dataset first.

Loading the Data set

Let's load the forestfires dataset using pandas.

1. >>> data=pd.**read_csv**('forestfires.csv')
2. >>> data.**head**()



```
>>> data.head()
   X  Y month  day  FFMC   DMC     DC  ISI  temp  RH  wind  rain  area
0  7  5   mar  fri  86.2  26.2   94.3  5.1   8.2  51   6.7   0.0   0.0
1  7  4   oct  tue  90.6  35.4  669.1  6.7  18.0  33   0.9   0.0   0.0
2  7  4   oct  sat  90.6  43.7  686.9  6.7  14.6  33   1.3   0.0   0.0
3  8  6   mar  fri  91.7  33.3   77.5  9.0   8.3  97   4.0   0.2   0.0
4  8  6   mar  sun  89.3  51.3  102.2  9.6  11.4  99   1.8   0.0   0.0
```

Train and Test Set in Python Machine Learning

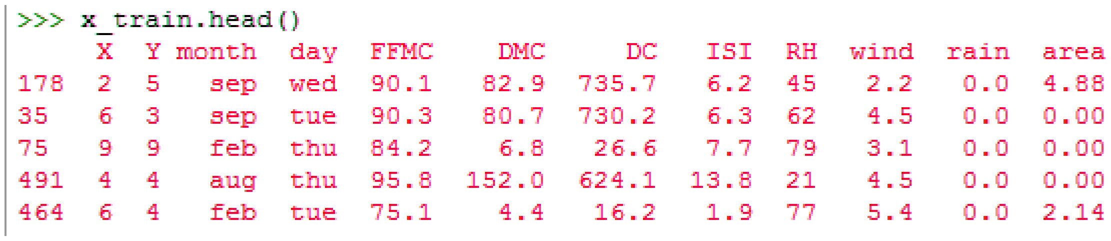
b. Splitting

Let's split this data into labels and features. Now, what's that? Using features, we predict labels. I mean using features (the data we use to predict labels), we predict labels (the data we want to predict).

1. >>> y=data.temp
2. >>> x=data.**drop**('temp',axis=1)

Temp is a label to predict temperatures in y; we use the drop() function to take all other data in x. Then, we split the data.

1. >>> x_train,x_test,y_train,y_test=**train_test_split**(x,y,test_size=0.2)
2. >>> x_train.**head**()



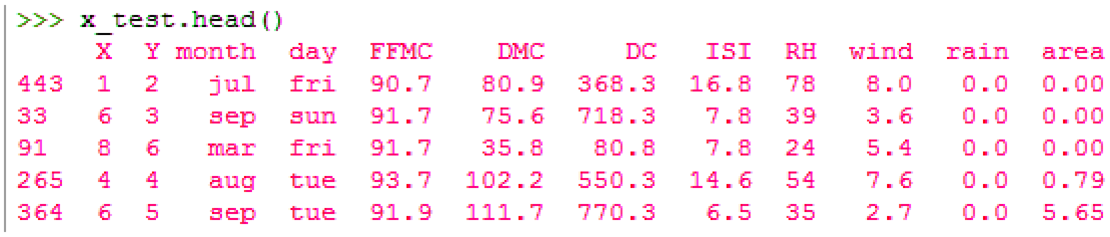
```
>>> x_train.head()
     X  Y month  day  FFMC    DMC     DC   ISI  RH  wind  rain  area
178  2  5   sep  wed  90.1   82.9  735.7   6.2  45   2.2   0.0  4.88
35   6  3   sep  tue  90.3   80.7  730.2   6.3  62   4.5   0.0  0.00
75   9  9   feb  thu  84.2    6.8   26.6   7.7  79   3.1   0.0  0.00
491  4  4   aug  thu  95.8  152.0  624.1  13.8  21   4.5   0.0  0.00
464  6  4   feb  tue  75.1    4.4   16.2   1.9  77   5.4   0.0  2.14
```

Train and Test Set in Python Machine Learning

1. >>> x_train.shape

(413, 12)

1. >>> x_test.**head**()



```
>>> x_test.head()
     X  Y month  day  FFMC    DMC     DC   ISI  RH  wind  rain  area
443  1  2   jul  fri  90.7   80.9  368.3  16.8  78   8.0   0.0  0.00
33   6  3   sep  sun  91.7   75.6  718.3   7.8  39   3.6   0.0  0.00
91   8  6   mar  fri  91.7   35.8   80.8   7.8  24   5.4   0.0  0.00
265  4  4   aug  tue  93.7  102.2  550.3  14.6  54   7.6   0.0  0.79
364  6  5   sep  tue  91.9  111.7  770.3   6.5  35   2.7   0.0  5.65
```

Train and Test Set in Python Machine Learning

1. >>> x_test.shape

(104, 12)

The line test_size=0.2 suggests that the test data should be 20% of the dataset and the rest should be train data. With the outputs of the shape () functions, you can see that we have 104 rows in the test data and 413 in the training data.

Example 2: Train-Test Split for Regression

We will demonstrate how to use the train-test split to evaluate a random forest algorithm on the housing dataset.

The housing dataset is a standard machine learning dataset composed of 506 rows of data with 13 numerical input variables and a numerical target variable.

The dataset involves predicting the house price given details of the house is in the suburbs of the American city of Boston.

- Housing Dataset (housing.csv)
- Housing Description (housing.names)

You will not need to download the dataset; we will download it automatically as part of our worked examples. The example below downloads and loads the dataset as a Pandas DataFrame and summarizes the shape of the dataset.

```
# load and summarize the housing dataset from pandas import read_csv

# load dataset

url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'

dataframe = read_csv(url, header=None)

# summarize shape

print(dataframe.shape)
```

Running the example confirms the 506 rows of data and 13 input variables and single numeric target variables (14 in total).

```
(506, 14)
```

1

e can now evaluate a model using a train-test split.

First, the loaded dataset must be split into input and output components.

```
...

# split into inputs and outputs

X, y = data[:, :-1], data[:, -1]

print(X.shape, y.shape)
```

Next, we can split the dataset so that 67 percent is used to train the model and 33 percent is used to evaluate it. This split was chosen arbitrarily.

```
...
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

We can then define and fit the model on the training dataset.

```
...
# fit the model
model = RandomForestRegressor(random_state=1)
model.fit(X_train, y_train)
```

Then use the fit model to make predictions and evaluate the predictions using the mean absolute error (MAE) performance metric.

```
...
# make predictions
yhat = model.predict(X_test)
# evaluate predictions
mae = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % mae)
```

Tying this together, the complete example is listed below.

```
# train-test split evaluation random forest on the housing dataset
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
# load dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
# split into inputs and outputs
```

```
X, y = data[:, :-1], data[:, -1]

print(X.shape, y.shape)

# split into train test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# fit the model

model = RandomForestRegressor(random_state=1)

model.fit(X_train, y_train)

# make predictions

yhat = model.predict(X_test)

# evaluate predictions

mae = mean_absolute_error(y_test, yhat)

print('MAE: %.3f' % mae)
```

Running the example first loads the dataset and confirms the number of rows in the input and output elements.

The dataset is split into train and test sets and we can see that there are 339 rows for training and 167 rows for the test set.

Finally, the model is evaluated on the test set and the performance of the model when making predictions on new data is a mean absolute error of about 2.211 (thousands of dollars).

```
(506, 13) (506,)

(339, 13) (167, 13) (339,) (167,)

MAE: 2.157
```

6.2 Introduce concept of cross validation

Machine learning is an iterative process.

You will face choices about predictive variables to use, what types of models to use, what arguments to supply those models, etc. We make these choices in a data-driven way by measuring model quality of various alternatives.

You've already learned to use train_test_split to split the data, so you can measure model quality on the test data. Cross-validation extends this approach to model scoring (or "model validation.") Compared to train_test_split, cross-validation gives you a more reliable measure of your model's quality, though it takes longer to run.

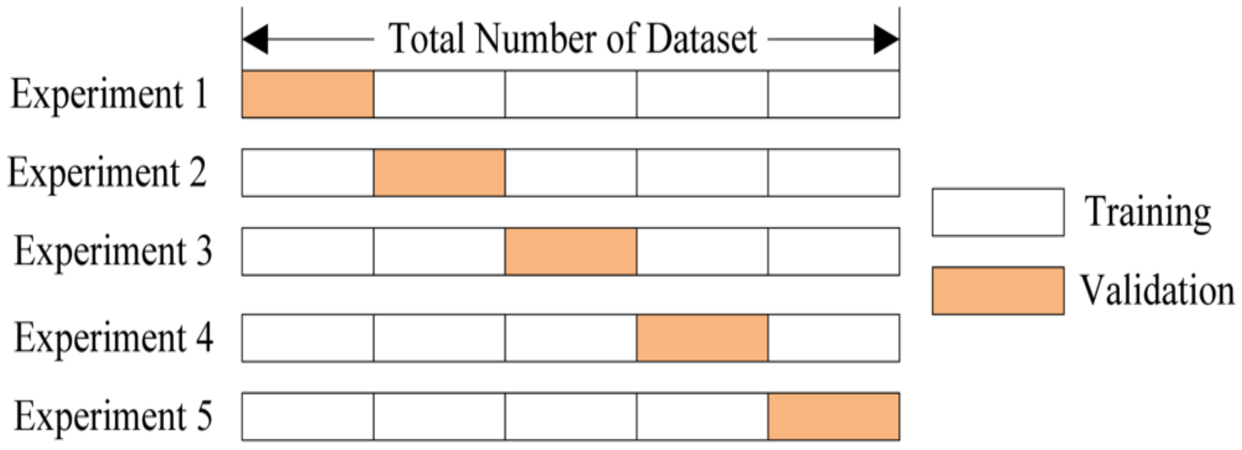
The Shortcoming of Train-Test Split

Imagine you have a dataset with 5000 rows. The train_test_split function has an argument for test_size that you can use to decide how many rows go to the training set and how many go to the test set. The larger the test set, the more reliable your measures of model quality will be. At an extreme, you could imagine having only 1 row of data in the test set. If you compare alternative models, which one makes the best predictions on a single data point will be mostly a matter of luck.

You will typically keep about 20% as a test dataset. But even with 1000 rows in the test set, there's some random chance in determining model scores. A model might do well on one set of 1000 rows, even if it would be inaccurate on a different 1000 rows. The larger the test set, the less randomness (aka "noise") there is in our measure of model quality.

The Cross-Validation Procedure

In cross-validation, we run our modeling process on different subsets of the data to get multiple measures of model quality. For example, we could have 5 **folds** or experiments. We divide the data into 5 pieces, each being 20% of the full dataset.



We run an experiment called experiment 1 which uses the first fold as a holdout set, and everything else as training data. This gives us a measure of model quality based on a 20% holdout set, much as we got from using the simple train-test split.

We then run a second experiment, where we hold out data from the second fold (using everything except the 2nd fold for training the model.) This gives us a second estimate of model quality. We repeat this process, using every fold once as the holdout. Putting this together, 100% of the data is used as a holdout at some point.

Returning to our example above from train-test split, if we have 5000 rows of data, we end up with a measure of model quality based on 5000 rows of holdout (even if we don't use all 5000 rows simultaneously.

Trade-offs Between Cross-Validation and Train-Test Split

Cross-validation gives a more accurate measure of model quality, which is especially important if you are making a lot of modeling decisions. However, it can take more time to run, because it estimates models once for each fold. So it is doing more total work.

Given these tradeoffs, when should you use each approach? On small datasets, the extra computational burden of running cross-validation isn't a big deal. These are also the problems where model quality scores would be least reliable with train-test split. So, if your dataset is smaller, you should run cross-validation.

For the same reasons, a simple train-test split is sufficient for larger datasets. It will run faster, and you may have enough data that there's little need to re-use some of it for holdout.

There's no simple threshold for what constitutes a large vs small dataset. If your model takes a couple minute or less to run, it's probably worth switching to cross-validation. If your model takes much longer to run, cross-validation may slow down your workflow more than it's worth.

Alternatively, you can run cross-validation and see if the scores for each experiment seem close. If each experiment gives the same results, train-test split is probably sufficient.

Example

First we read the data

```
import pandas as pd
data = pd.read_csv('../input/melb_data.csv')
cols_to_use = ['Rooms', 'Distance', 'Landsize', 'BuildingArea', 'YearBuilt']
X = data[cols_to_use]
y = data.Price
```

Then specify a pipeline of our modeling steps (It can be very difficult to do cross-validation properly if you aren't using pipelines)

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Imputer
my_pipeline = make_pipeline(Imputer(), RandomForestRegressor())
```

Finally get the cross-validation scores:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(my_pipeline, X, y, scoring='neg_mean_absolute_error')
print(scores)
```

```
[-322244.30158131 -305507.19909774 -284635.2229142 ]
```

You may notice that we specified an argument for *scoring*. This specifies what measure of model quality to report. The docs for scikit-learn show a list of options.

It is a little surprising that we specify *negative* mean absolute error in this case. Scikit-learn has a convention where all metrics are defined so a high number is better. Using negatives here allows them to be consistent with that convention, though negative MAE is almost unheard of elsewhere.

You typically want a single measure of model quality to compare between models. So we take the average across experiments.

```
print('Mean Absolute Error %2f' %(-1 * scores.mean()))
```

```
Mean Absolute Error 304128.907864
```

Conclusion

Using cross-validation gave us much better measures of model quality, with the added benefit of cleaning up our code (no longer needing to keep track of separate train and test sets. So, it's a good win.

Activity 1: Convert the code for your on-going project over from train-test split to cross-validation. Make sure to remove all code that divides your dataset into training and testing datasets. Leaving code you don't need any more would be sloppy.

Activity 2: Add or remove a predictor from your models. See the cross-validation score using both sets of predictors, and see how you can compare the scores.

7. Metrics of model quality by simple Math and examples

After you make predictions, you need to know if they are any good. There are standard measures that we can use to summarize how good a set of predictions actually are.

Knowing how good a set of predictions is, allows you to make estimates about how good a given machine learning model of your problem,

You must estimate the quality of a set of predictions when training a machine learning model.

Performance metrics like classification accuracy and root mean squared error can give you a clear objective idea of how good a set of predictions is, and in turn how good the model is that generated them.

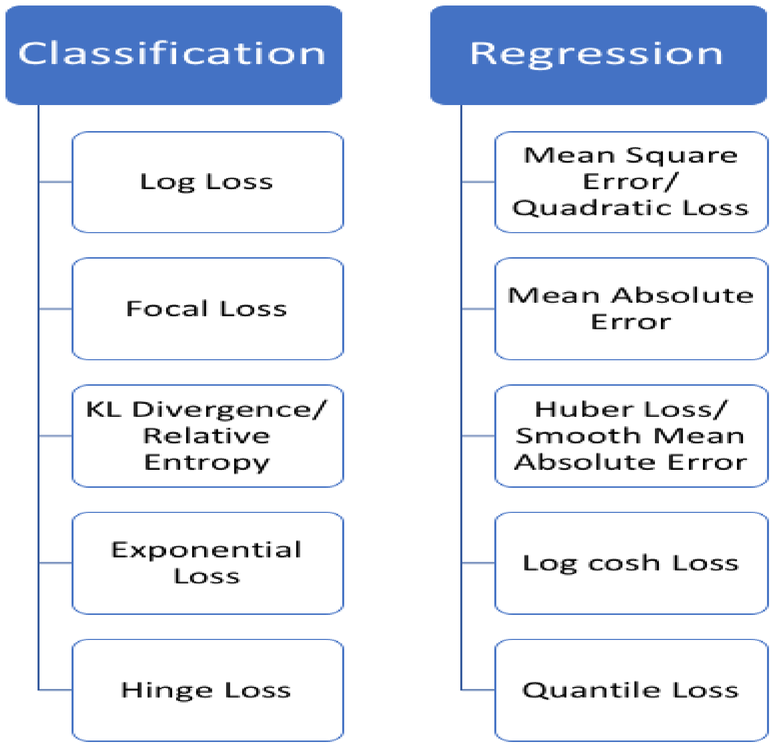
This is important as it allows you to tell the difference and select among:

- Different transforms of the data used to train the same machine learning model.
- Different machine learning models trained on the same data.
- Different configurations for a machine learning model trained on the same data.

As such, performance metrics are a required building block in implementing machine learning algorithms from scratch.

All the algorithms in machine learning rely on minimizing or maximizing a function, which we call “objective function”. The group of functions that are minimized are called “loss functions”. A loss function is a measure of how good a prediction model does in terms of being able to predict the expected outcome. A most commonly used method of finding the minimum point of function is “gradient descent”. Think of loss function like undulating mountain and gradient descent is like sliding down the mountain to reach the bottom most point.

Loss functions can be broadly categorized into 2 types: **Classification and Regression Loss**.



Regression functions predict a quantity, and classification functions predict a label.

7.1 RMSE (Root Mean Squared Error)

In this section, we will be looking at one of the methods to determine the accuracy of our model in predicting the target values. All of you reading this article must have heard about the term RMS i.e. Root Mean Square and you might have also used RMS values in statistics as well. In machine Learning when we want to look at the accuracy of our model we take the root mean square of the error that has occurred between the test values and the predicted values mathematically:

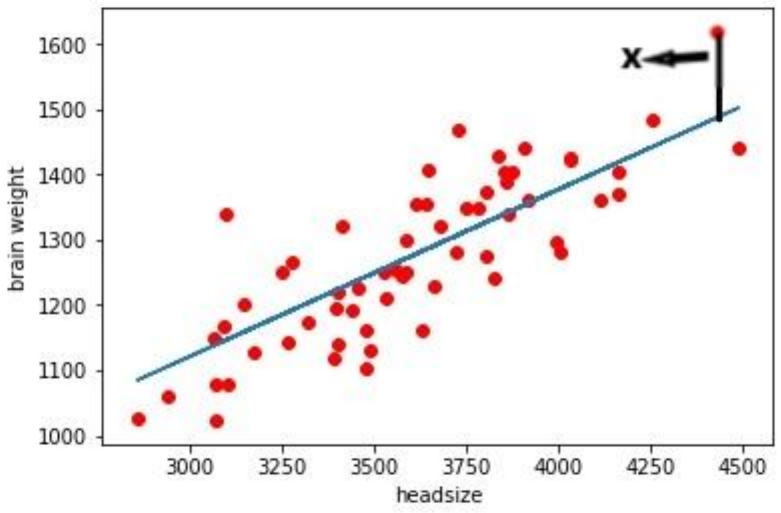
For a single value:

```
Let a= (predicted value- actual value) ^2
Let b= mean of a = a (for single value)
Then RMSE= square root of b
```

For a wide set of values RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Graphically:



As you can see in this scattered graph the red dots are the actual values and the blue line is the set of predicted values drawn by our model. Here X represents the distance between the actual value and the predicted line this line represents the error, similarly, we can draw straight lines from each red dot to the blue line. Taking mean of all those distances and squaring them and finally taking the root will give us RMSE of our model.

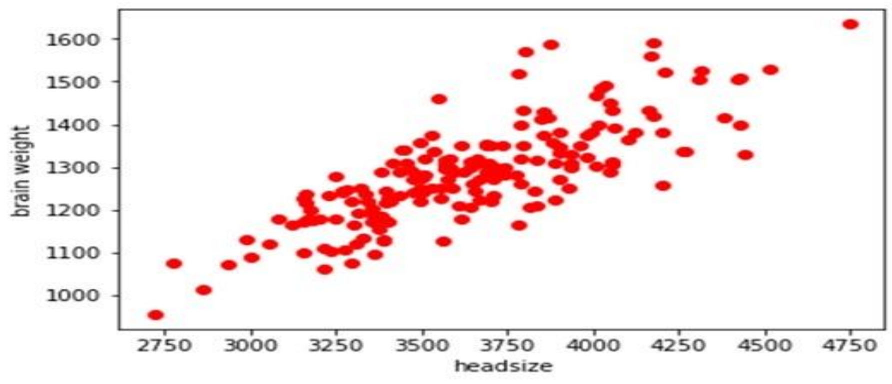
Example 1 (RMSE)

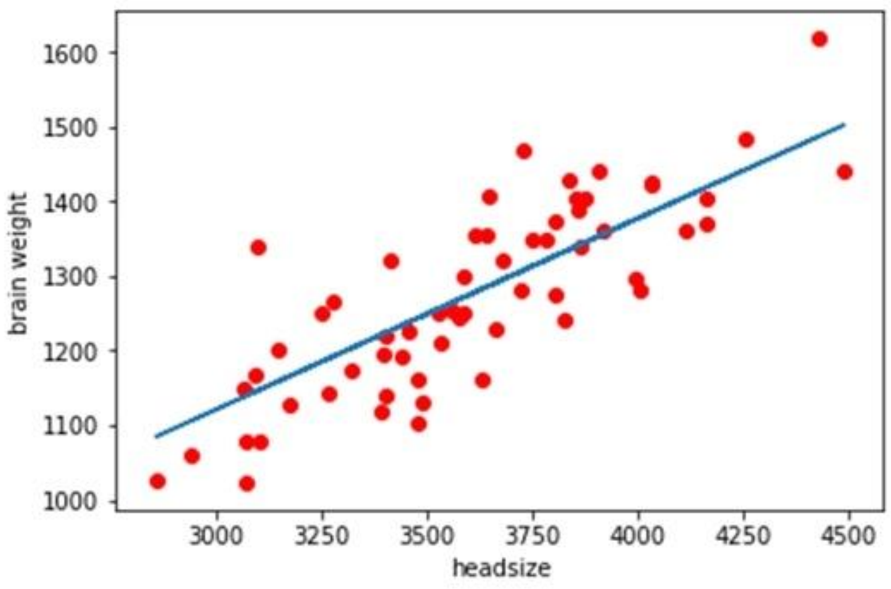
Let us write a python code to find out **RMSE** values of our model. We would be predicting the brain weight of the users. We would be using linear regression to train our model, the data set used in my code can be downloaded from here: headbrain6-

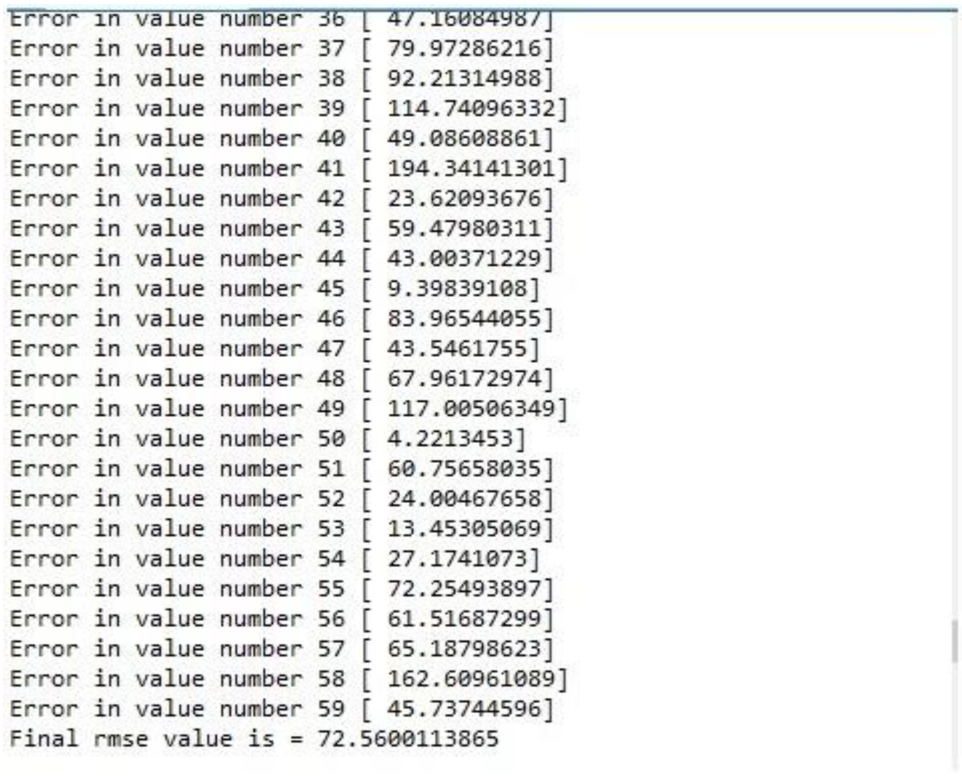
```
import time

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


#reading the data

"""



here the directory of my code and the headbrain6.csv file  is same make sure both the files are stored in same folder or directory

"""

data=pd.read_csv('headbrain6.csv')

data.head()

x=data.iloc[:,2:3].values

y=data.iloc[:,3:4].values


#splitting the data into training and test

from sklearn.cross_validation import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/4,random_state=0)


#fitting simple linear regression to the training set

from sklearn.linear_model import LinearRegression

regressor=LinearRegression()

regressor.fit(x_train,y_train)
```

```
#predict the test result
y_pred=regressor.predict(x_test)

#to see the relationship between the training data values
plt.scatter(x_train,y_train,c='red')
plt.show()

#to see the relationship between the predicted
#brain weight values using scattered graph
plt.plot(x_test,y_pred)
plt.scatter(x_test,y_test,c='red')
plt.xlabel('headsize')
plt.ylabel('brain weight')

#errorin each value
for i in range(0,60):
print("Error in value number",i,(y_test[i]-y_pred[i]))
time.sleep(1)

#combined rmse value
rss=((y_test-y_pred)**2).sum()
mse=np.mean((y_test-y_pred)**2)
print("Final rmse value is =",np.sqrt(np.mean((y_test-y_pred)**2)))
```

Output







```
Error in value number 36 [ 47.16084987]
Error in value number 37 [ 79.97286216]
Error in value number 38 [ 92.21314988]
Error in value number 39 [ 114.74096332]
Error in value number 40 [ 49.08608861]
Error in value number 41 [ 194.34141301]
Error in value number 42 [ 23.62093676]
Error in value number 43 [ 59.47980311]
Error in value number 44 [ 43.00371229]
Error in value number 45 [ 9.39839108]
Error in value number 46 [ 83.96544055]
Error in value number 47 [ 43.5461755]
Error in value number 48 [ 67.96172974]
Error in value number 49 [ 117.00506349]
Error in value number 50 [ 4.2213453]
Error in value number 51 [ 60.75658035]
Error in value number 52 [ 24.00467658]
Error in value number 53 [ 13.45305069]
Error in value number 54 [ 27.1741073]
Error in value number 55 [ 72.25493897]
Error in value number 56 [ 61.51687299]
Error in value number 57 [ 65.18798623]
Error in value number 58 [ 162.60961089]
Error in value number 59 [ 45.73744596]
Final rmse value is = 72.5600113865
```

The **RMSE** value of our is coming out to be approximately 73 which is not bad. A good model should have an RMSE value less than 180. In case you have a higher RMSE value, this would mean that you probably need to change your feature or probably you need to tweak your hyperparameters.

7.2 MSE (Mean Squared Error)

Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$

Below is a plot of an MSE function where the true target value is 100, and the predicted values range between -10,000 to 10,000. The MSE loss (Y-axis) reaches its minimum value at prediction (X-axis) = 100. The range is 0 to ∞.

Why use mean squared error

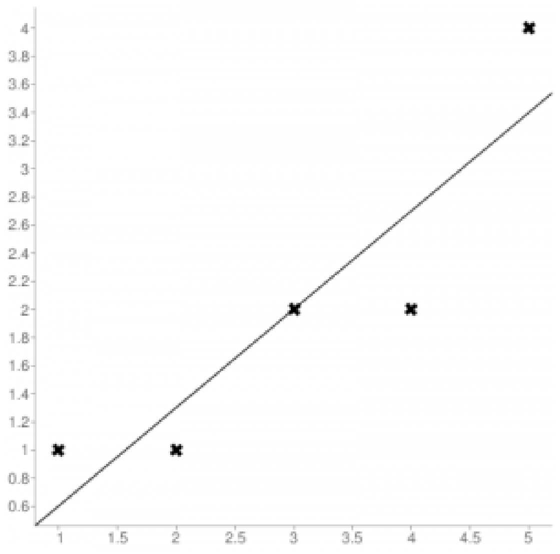
MSE is sensitive towards outliers and given several examples with the same input feature values, the optimal prediction will be their mean target value. This should be compared with Mean Absolute Error, where the optimal prediction is the median. MSE is thus good to use if you believe that your target data, conditioned on the input, is normally distributed around a mean value, and when it's important to penalize outliers extra much.

When to use mean squared error

Use MSE when doing regression, believing that your target, conditioned on the input, is normally distributed, and want large errors to be significantly (quadratically) more penalized than small ones.

Example-1: You want to predict future house prices. The price is a continuous value, and therefore we want to do regression. MSE can here be used as the loss function.

Example-2: Consider the given data points: (1,1), (2,1), (3,2), (4,2), (5,4)
You can use this online calculator to find the regression equation / line.



Regression line equation: Y = 0.7X – 0.1

x	y	Yi
1	1	0.6
2	1	1,39
3	2	1.99
4	2	2.69
5	4	3.4

```
from sklearn.metrics import mean_squared_error

# Given values
Y_true = [1,1,2,2,4]  # Y_true = Y (original values)

# calculated values
Y_pred = [0.6,1.29,1.99,2.69,3.4]  # Y_pred = Y'

# Calculation of Mean Squared Error (MSE)
mean_squared_error(Y_true,Y_pred)
```

Output: 0.21606

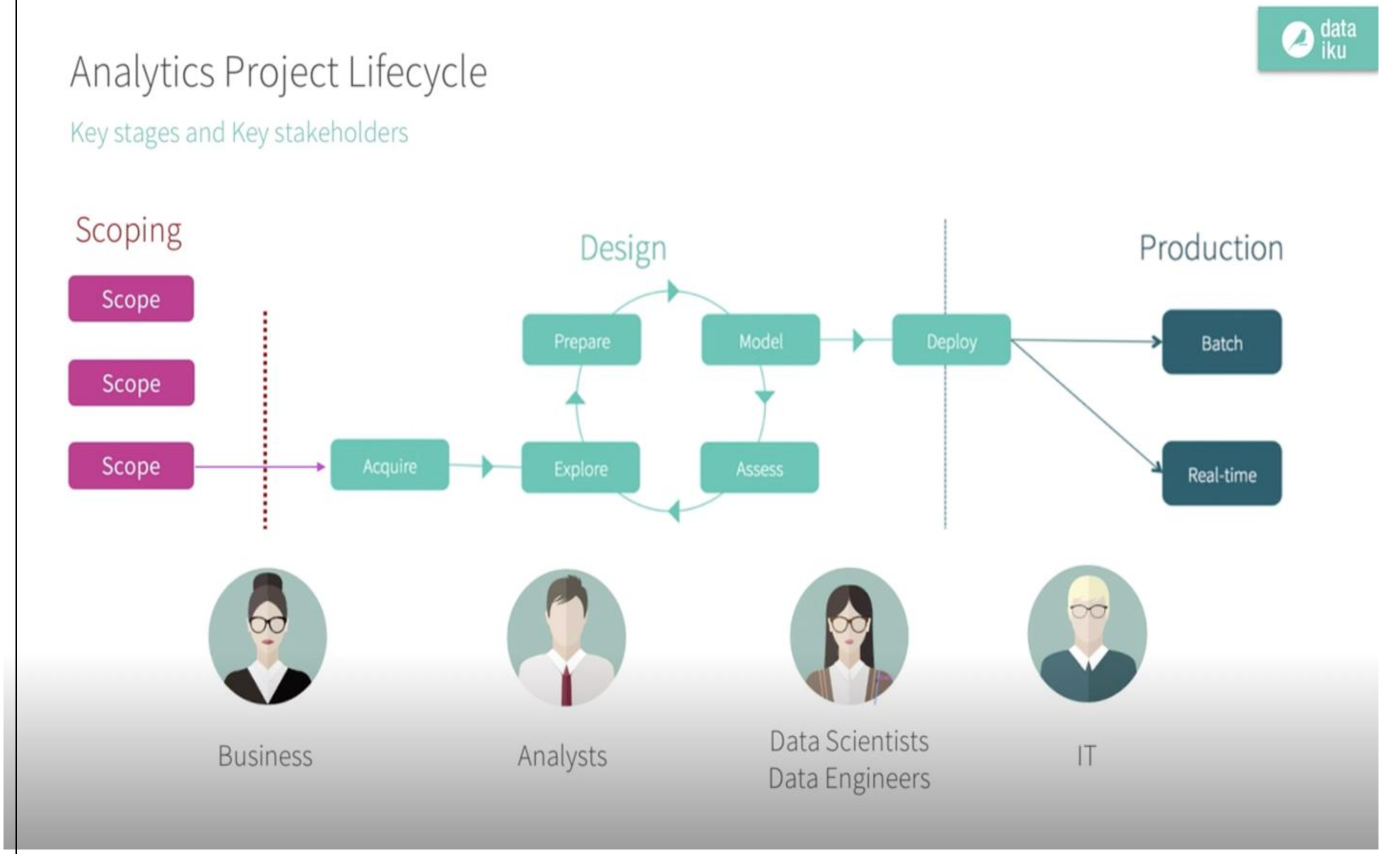
Unit 2: Model Life cycle

Title: Model Life cycle	**Approach:** Hands on, Team Discussion, Web search**,** Case studies
Summary: The machine learning life cycle is the cyclical process that AI or machine learning projects follow. It defines each step that an engineer or developer should follow. Generally, every AI project lifecycle encompasses three main stages: project scoping, design or build phase, and deployment in production. In this unit we will go over each of them and the key steps and factors to consider when implementing them. The expectation out of students would be to focus more on the hands-on aspect of AI projects.	
Objectives: **1.** Students should develop their capstone project using AI project cycle methodologies **2.** Students should be comfortable in breaking down their projects into different phases of AI project cycle **3.** Students should be in a position to choose and apply right AI model to solve the problem	
Learning Outcomes: 1. Students will demonstrate the skill of breaking down a problem in smaller sub units according to AI project life cycle methodologies 2. Students will demonstrate proficiency in choosing and applying the correct AI or ML model	
Key Concepts: AI Project Cycle, Model validation, AI deployment, IBM Watson	

1. AI Model Life Cycle

Understanding AI Project Cycle is crucial element for students to have a robust and sound AI foundation. In order to implement successful AI projects, students need to adopt a comprehensive approach to covering each step of the AI or machine learning life cycle - starting from project scoping and data prep, and going through all the stages of model building, deployment, management, analytics, to full-blown Enterprise AI.

Generally, every AI project lifecycle encompasses three main stages: **project scoping, design or build phase, and deployment in production.** Let's go over each of them and the key steps and factors to consider when implementing them.



(Source : https://blog.dataiku.com/ai-projects-lifecycle-key-steps-and-considerations)

Step 1: Scoping (Requirements analysis)

The first fundamental step when starting an AI initiative is scoping and selecting the relevant use case(s) that the AI model will be built to address. This is arguably the most important part of your AI project. Why? There's a couple of reasons for it. First, this stage involves the planning and motivational aspects of your project. It is important to start strong if you want your artificial intelligence project to be successful. There's a great phrase that characterizes this project stage: garbage in, garbage out. This means if the data you collect is no good, you won't be able to build an effective AI algorithm, and your whole project will collapse.

In this phase, it's crucial to precisely define the strategic business objectives and desired outcomes of the project, select align all the different stakeholders' expectations, anticipate the key resources and steps, and define the success metrics. Selecting the AI or machine learning use cases and being able to evaluate the return on investment (ROI) is critical to the success of any data project.

Step 2: Design/Building the Model

Once the relevant projects have been selected and properly scoped, the next step of the machine learning lifecycle is the Design or Build phase, which can take from a few days to multiple months, depending on the nature of the project. The Design phase is essentially an iterative process comprising all the steps relevant to building the AI or machine learning model: data acquisition, exploration, preparation, cleaning, feature engineering, testing and running a set of models to try to predict behaviours or discover insights in the data.

Enabling all the different people involved in the AI project to have the appropriate access to data, tools, and processes in order to collaborate across different stages of the model building is critical to its success. Another key success factor to consider is model validation: how will you determine, measure, and evaluate the performance of each iteration with regards to the defined ROI objective?

During this phase, you need to evaluate the various AI development platforms, e.g.:

- Open languages — Python is the most popular, with R and Scala also in the mix.
- Open frameworks — Scikit-learn, XGBoost, TensorFlow, etc.
- Approaches and techniques — Classic ML techniques from regression all the way to state-of-the-art GANs and RL
- Productivity-enhancing capabilities — Visual modelling, AutoAI to help with feature engineering, algorithm selection and hyperparameter optimization
- Development tools — DataRobot, H2O, Watson Studio, Azure ML Studio, Sagemaker, Anaconda, etc.

Different AI development platforms offer extensive documentation to help the development teams. Depending on your choice of the AI platform, you need to visit the appropriate webpages for this documentation, which are as follows:

- Microsoft Azure AI Platform;
- Google Cloud AI Platform;
- IBM Watson Developer platform;
- BigML;
- Infosys Nia resources.

Step 3: Testing

While the fundamental testing concepts are fully applicable in AI development projects, there are additional considerations too. These are as follows:

- The volume of test data can be large, which presents complexities.
- Human biases in selecting test data can adversely impact the testing phase, therefore, data validation is important.
- Your testing team should test the AI and ML algorithms keeping model validation, successful learnability, and algorithm effectiveness in mind.
- Regulatory compliance testing and security testing are important since the system might deal with sensitive data, moreover, the large volume of data makes performance testing crucial.
- You are implementing an AI solution that will need to use data from your other systems, therefore, systems integration testing assumes importance.
- Test data should include all relevant subsets of training data, i.e., the data you will use for training the AI system.
- Your team must create test suites that help you validate your ML models.

Unit 3: Storytelling

Refreshing what we learnt in Level 1, we will be re-visiting some concepts of storytelling.

Why storytelling is so powerful and cross-cultural, and what this means for data storytelling?

Stories create engaging experiences that transport the audience to another space and time. They establish a sense of community belongingness and identity. For these reasons, storytelling is considered a powerful element that enhances global networking by increasing the awareness about the cultural differences and enhancing cross-cultural understanding. Storytelling is an integral part of indigenous cultures.

Some of the factors that make storytelling powerful are its attribute to make information more compelling, the ability to present a window in order to take a peek at the past, and finally to draw lessons and to reimagine the future by affecting necessary changes. Storytelling also shapes, empowers and connects people by doing away with judgement or critic and facilitates openness for embracing differences.

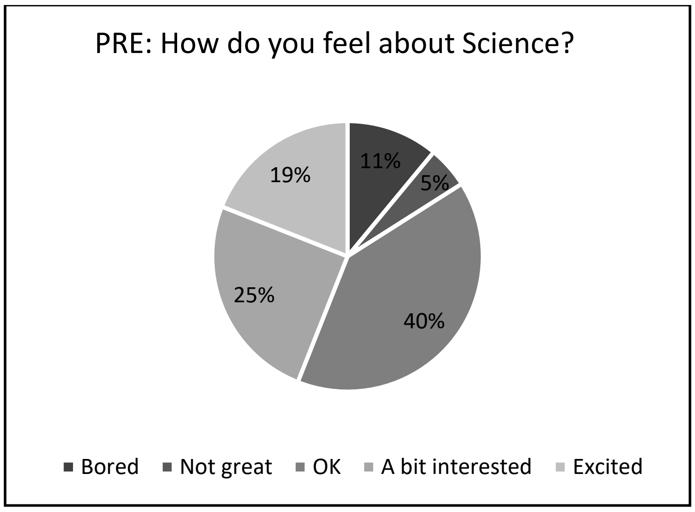
A well-told story is an inspirational narrative that is crafted to engage the audience across boundaries and cultures, as they have the impact that isn't possible with data alone. Data can be persuasive, but stories are much more. They change the way that we interact with data, transforming it from a dry collection of "facts" to something that can be entertaining, engaging, thought provoking, and inspiring change.

Each data point holds some information which maybe unclear and contextually deficient on its own. The visualizations of such data are therefore, subject to interpretation (and misinterpretation). However, stories are more likely to drive action than are statistics and numbers. Therefore, when told in the form of a narrative, it reduces ambiguity, connects data with context, and describes a specific interpretation – communicating the important messages in most effective ways. The steps involved in telling an effective data story are given below:

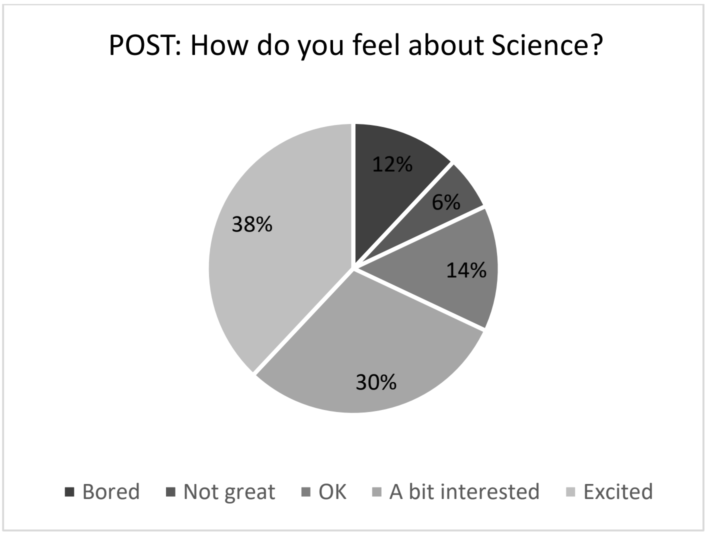
- Understanding the audience
- Choosing the right data and visualisations
- Drawing attention to key information
- Developing a narrative
- Engaging your audience

Activity

A new teacher joined the ABC Higher Secondary School, Ambapalli to teach Science to the students of Class XI. In his first class itself, he could make out that not everyone understood what was being taught in class. So, he decided to take a poll to assess the level of students. The following graph shows the level of interest of the students in the class.



Depending on the result obtained, he changed his method of teaching. After a month, he repeated the same poll once again to ascertain if there was any change. The results of poll are shown in the chart below.



With the help of the information provided create a good data story setting a strong narrative around the data, making it is easier to understand the pre and post data, existing problem, action taken by the teacher, and the resolution of the problem. Distribute A4 sheets and pens to the students for this activity.

Storytelling with Data

Purpose: To provide insight into data storytelling and how it can bring a story to life.

Say: "Now that you have understood what storytelling is and why it is needed, let us learn about a storytelling of a different kind - the art of data storytelling and in the form of a narrative or story."

Session Preparation

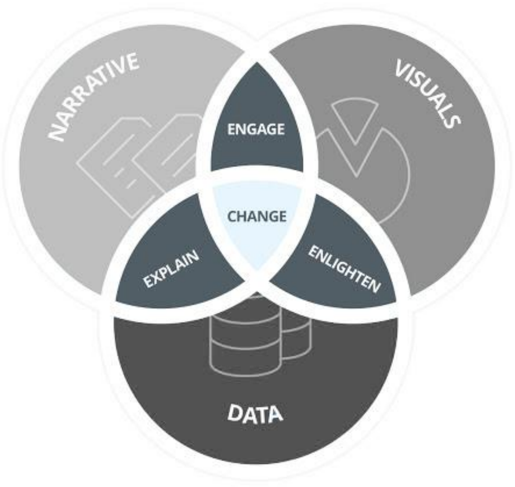
Logistics: For a class of ____ students. [Group Activity]

Materials Required

ITEM	QUANTITY
A4 sheets	Xx
Pens	Xx

Data storytelling is a structured approach for communicating insights drawn from data, and invariably involves a combination of three key elements: ***data*, *visuals*,** and ***narrative***. When the narrative is accompanied with data, it helps to ***explain*** the audience what's happening in the data and why a particular insight has been generated. When visuals are applied to data, they can ***enlighten*** the audience to the insights that they wouldn't perceive without the charts or graphs.

Finally, when narrative and visuals are merged together, they can ***engage*** or even entertain an audience. When you combine the right visuals and narrative with the right data, you have a data story that can influence and drive ***change***.



By the numbers: How to tell a great story with your data?

Presenting the data as a series of disjointed charts and graphs could result in the audience struggling to understand it – or worse, come to the wrong conclusions entirely. Thus, the importance of a narrative comes from the fact that it explains what is going on within the data set. It offers a context and meaning, relevance and clarity. A narrative shows the audience where to look and what not to miss and also keeps the audience engaged.

Good stories don't just emerge from data itself; they need to be unravelled from data relationships. Closer scrutiny helps uncover how each data point relates with other. Some easy steps that can assist in finding compelling stories in the data sets are as follows:

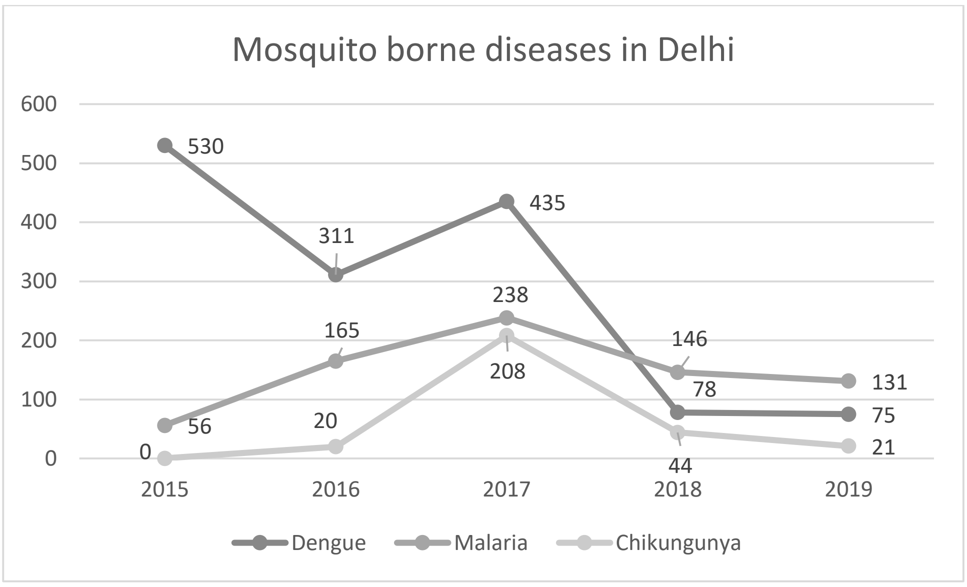
Step 1: Get the data and organise it.

Step 2: Visualize the data.

Step 3: Examine data relationships.

Step 4: Create a simple narrative embedded with conflict.

Activity: Try creating a data story with the information given below and use your imagination to reason as to why some cases have spiked while others have seen a fall.

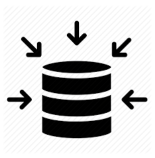


Data storytelling has acquired a place of importance because:

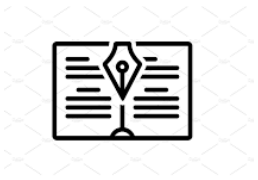
- It is an effective tool to transmit human experience. Narrative is the way we simplify and make sense of a complex world. It supplies context, insight, interpretation—all the things that make data meaningful, more relevant and interesting.
- No matter how impressive an analysis, or how high-quality the data, it is not going to compel change unless the people involved understand what is explained through a story.
- Stories that incorporate data and analytics are more convincing than those based entirely on anecdotes or personal experience.
- It helps to standardize communications and spread results.
- It makes information memorable and easier to retain in the long run.

Data Story elements challenge –

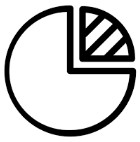
Identify the elements that make a compelling data story and name them



____



____



____

APPENDIX

Additional Resource for Advanced learners

The objective of this additional AI programming resource for Class 12 is to increase student knowledge and exposure to programming and help them create AI projects.

The Resources are divided into two categories:

- **Beginner** - For Students who have no experience in python programming
- **Advanced** – For Students who have some experience with python in earlier grades

The resources are shared in a cloud storage that has been made public. These resources are mainly python notebooks, links to open-source GitHub repositories and eBooks that will be updated on an ongoing basis.

Links:

Beginner - https://bit.ly/33spBZq

Advance - https://bit.ly/3b9US7V

Note: Please use google collab links for easy reference