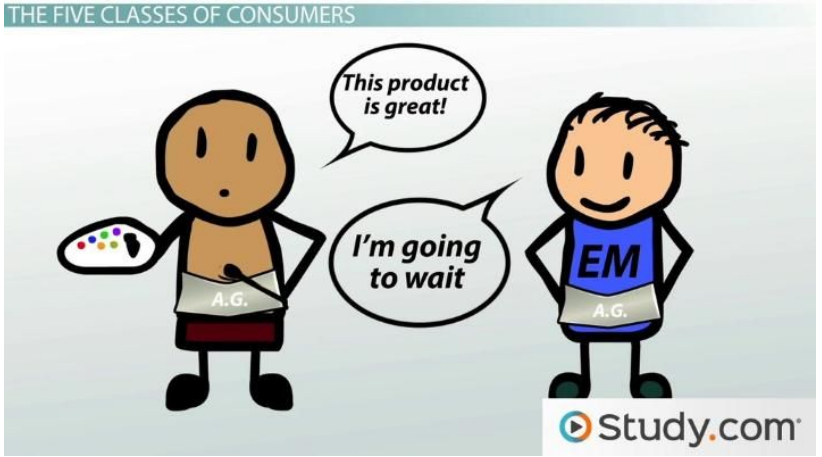
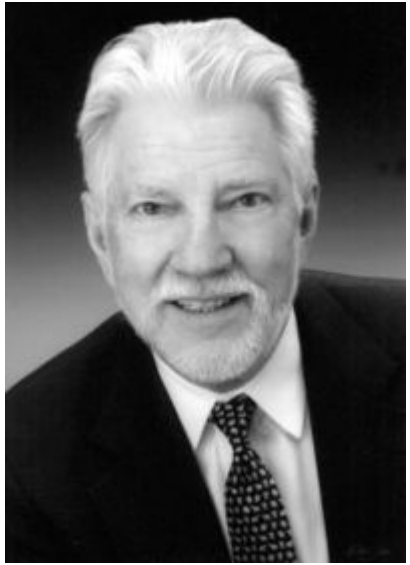
Pre-launch New Product Demand Forecasting Using The Bass Model:- A Machine Learning And Statistical Approach

Diffusion

- **Diffusion** is the process by which a new idea or new product is accepted by the market.
- The **rate of diffusion** is the speed with which the new idea spreads from one consumer to the next.



Innovator



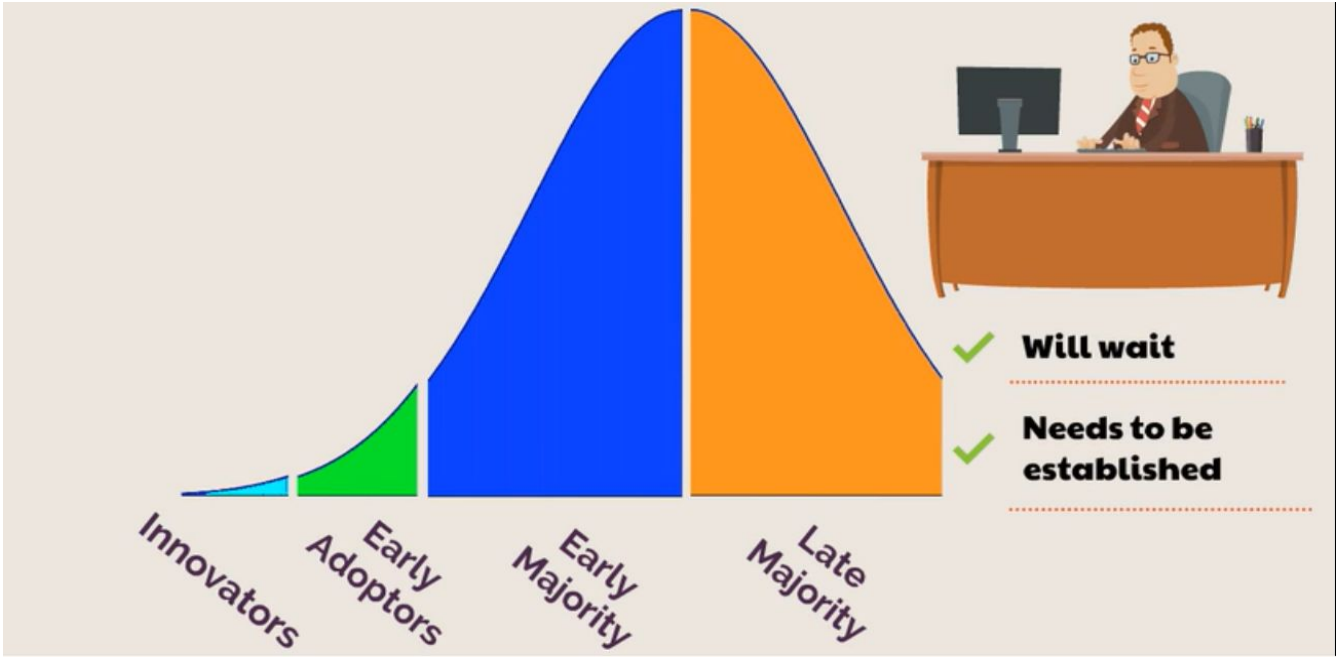
Everett M. Rogers (March 6, 1931 – October 21, 2004) was an eminent American communication theorist and sociologist, who originated the ***diffusion of innovation*** theory and introduced the term ***early adopter.***

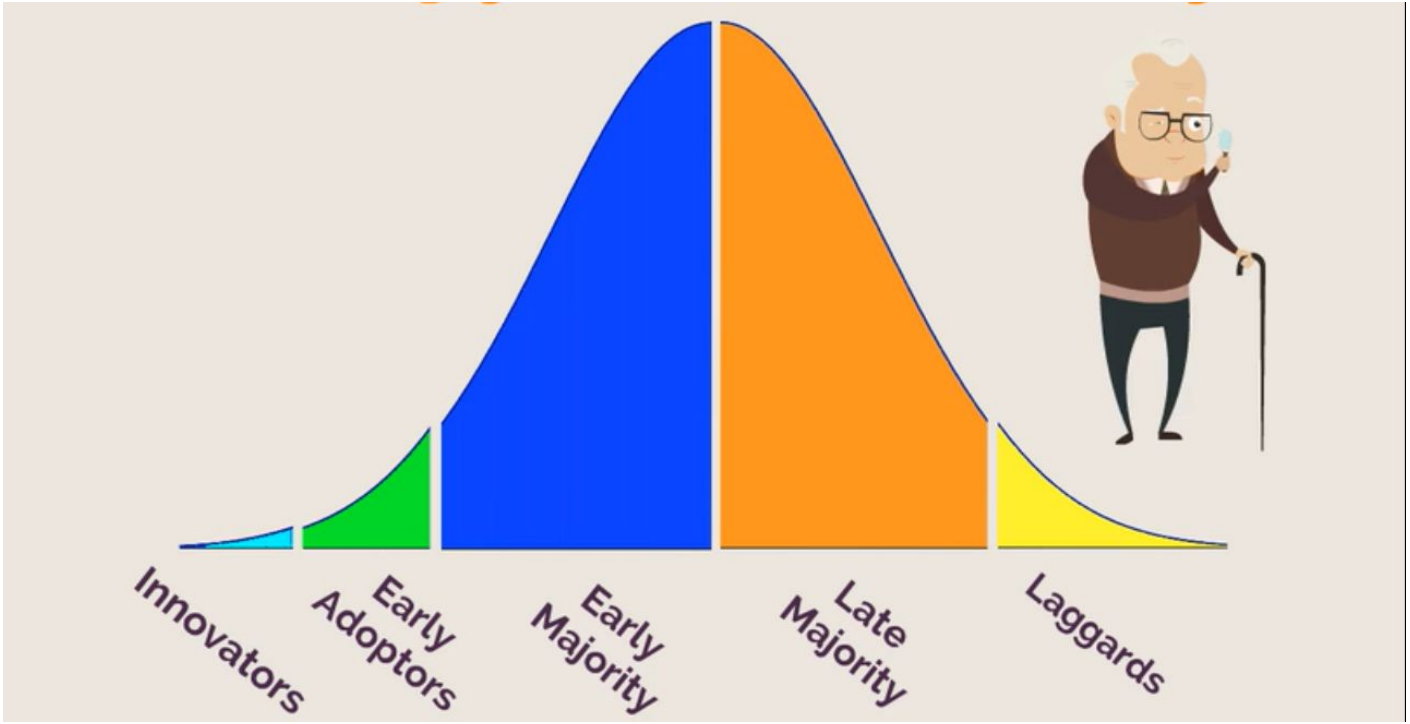
Book : Rogers, E. M. *Diffusion of innovations* , New York, NY: Free Press. (2nd most cited book in late 90's)

Classification


Innovators
Early Adoptors
Visionaries
Strategic Opportunities
Risk Takers

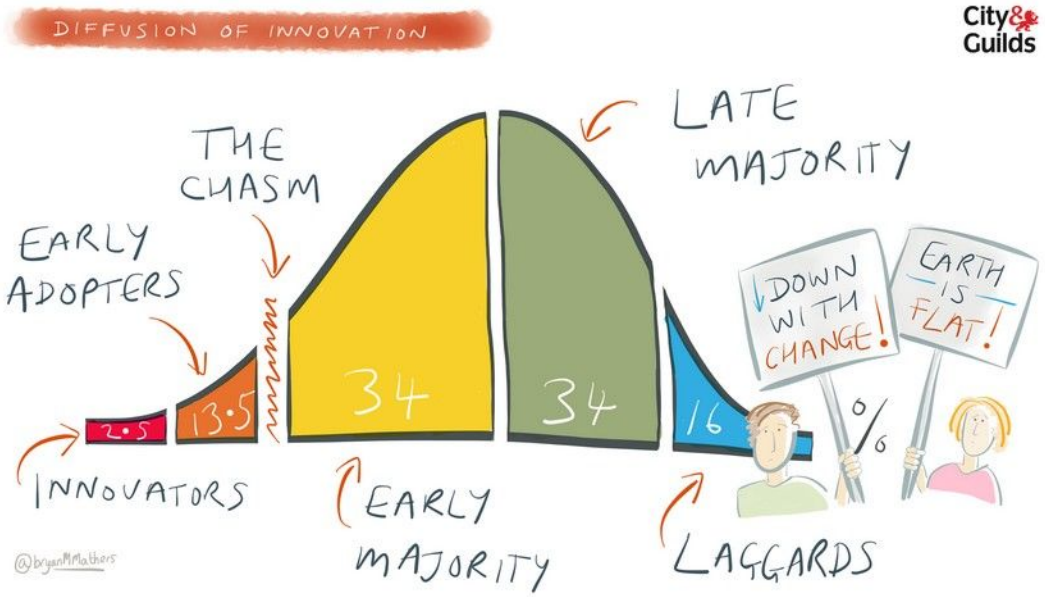

Innovators
Early Adoptors
Early Majority
Pragmatists
Risk Averse
Require References


Innovators
Early Adoptors
Early Majority
Late Majority
Will wait
Needs to be established


Innovators
Early Adoptors
Early Majority
Late Majority
Laggards

Market Segmentation :

- Early Market
- Mainstream Market



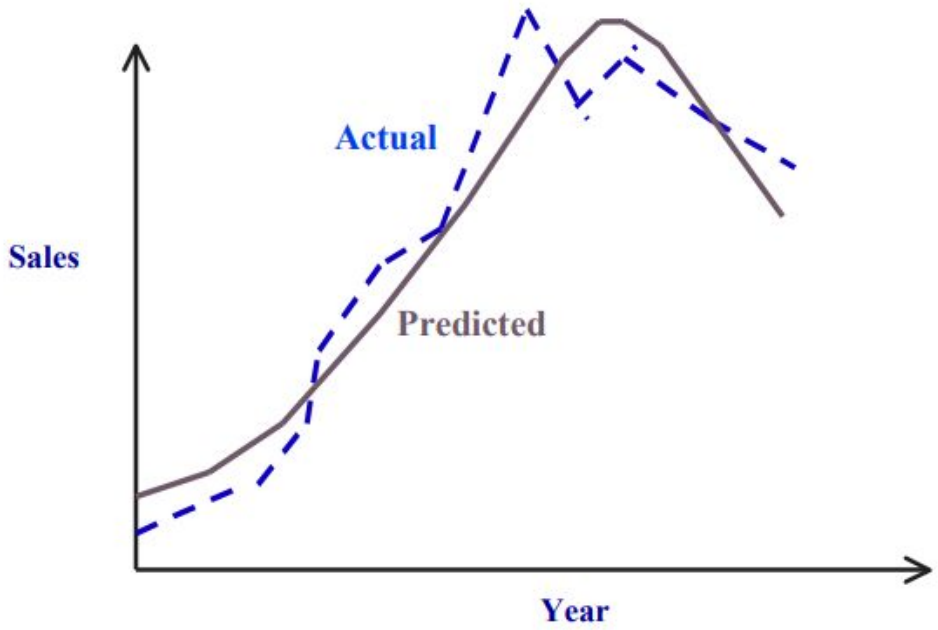
Bass Diffusion Model

This model designed to answer the question :

How many *customers will eventually adopt the new product and* ***when****?*

Research Paper : https://www.jstor.org/stable/2628128

Actual sales vs Predicted sales of A.C



A simple and elegant model with just three interpretable parameters can represent the sales trajectory quite well.

Assumptions

- There is no repetitive purchase.
- There is no Supply shock.
- The probability that an initial purchase will be made at T given that no purchase has yet been made is a linear function of the number of previous buyers.

$$P(T) = p + (q/m)*Y(T)$$

where,

- p ,q,m (total number purchasing at period) are constants.
- Y(T) is the number of previous buyers.
- The constant p is the probability of an initial purchase at T = 0 and its magnitude reflects the importance of innovators in the social system.
- The product q/m times Y(T) reflects the pressures operating on imitators as the number of previous buyers increases.

Suppose that the (cumulative) probability that someone in the target segment will adopt the innovation by time t is given by a nondecreasing continuous function F(T).

$$F(T) = \int_{\infty}^{T} f(t)\,.dt \quad , \quad F(0)=0$$

Here, f(t) indicates the rate at which the probability of adoption is changing at time t.

A customer will adopt the innovation at exactly time t since introduction, given that the customer has not adopted before that time.

$$H(T) = f(T) / [1-F(T)] = P(T) = p + (q/m)*Y(T)$$

$$dF(t) / dt = (p+q*F(t))[1- F(t)]$$

Now, after solving this non-linear differential equation we get :

$$F(t) = \frac{(1 - e^{-(p+q)t})}{(1 + \frac{q}{p} * e^{-(p+q)t})}$$

... initial case F(0)=0

Bass Model Parameters Estimation Methods

1.Ordinary least Squares estimation(OLS)

2. Non Linear least square estimation(NLS)

3. Maximum Likelihood Estimation(MLE)

OLS Estimation

The equation

$$dN(t) / dt = (p+q/m*N(t))*(m- N(t)) \quad \text{.....(i)}$$

The above equation can be discretized and written as:-

$$N(t_i) - N(t_{i-1}) = pm + (q-p)N(t_{i-1}) - \frac{q}{m}N^2(t_{i-1}),$$

$$X(i) = \alpha_1 + \alpha_2 N(t_{i-1}) + \alpha_3 N^2(t_{i-1}),$$

cont..

The estimates of the parameters are easily obtained by:-

$$\hat{p} = \frac{-\hat{\alpha}_2 + \sqrt{\hat{\alpha}_2^2 - 4\hat{\alpha}_1\hat{\alpha}_3}}{2},$$

$$\hat{q} = \frac{\hat{\alpha}_2 + \sqrt{\hat{\alpha}_2^2 - 4\hat{\alpha}_1\hat{\alpha}_3}}{2}, \quad \text{and}$$

$$\hat{m} = \frac{-\hat{\alpha}_2 - \sqrt{\hat{\alpha}_2^2 - 4\hat{\alpha}_1\hat{\alpha}_3}}{2\hat{\alpha}_3}.$$

Why NLS

- Overcomes the problem of multicollinearity.

- More valid estimates of standard errors for the parameter estimates.

Bass Model Parameter Calculations Using NLS method

The cumulative distribution function given by:-

$$F(t) = \frac{(1 - e^{-(p+q)t})}{(1 + \frac{q}{p} * e^{-(p+q)t})}$$

Using the cumulative distribution function Satoh Srinivasan and Mason suggest that parameter estimates p. q, and m can be obtained by using the following expression for the **number of adopters X(i)** in the **ith** time interval **(t(i-1) , t(i))**

$$X(i)=m(F(t_i)-F(t_{i-1}))+u_i$$

u_i= additive error

Calculation and Visualization of Parameters

The parameters are calculated using the R language by defining the nls function.

```
> summary(Bass.nls)

Formula: x ~ M * (((P + Q)^2/P) * exp(-(P + Q) * x))/(1 + (Q/P) * exp(-(P +
    Q) * x))^2

Parameters:
   Estimate Std. Error t value Pr(>|t|)
M 5.907e+01  6.564e+00   8.999 0.000844 ***
P 1.425e-02  9.765e-04  14.592 0.000128 ***
Q 4.481e-01  3.968e-02  11.293 0.000350 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2004 on 4 degrees of freedom

Number of iterations to convergence: 8
Achieved convergence tolerance: 1.023e-06
```

link:-https://www.wolframcloud.com/objects/64098d95-a588-4c20-bbfc-9e81dd1af6ac

Analogical Approach for Pre-Launch Forecasting

Assumption:-New product will behave as analogous products do.

Using a historical relationship between the parameters and attributes of analogous products can be a promising solution.

Methodology

Two types of Databases are required to developed:-

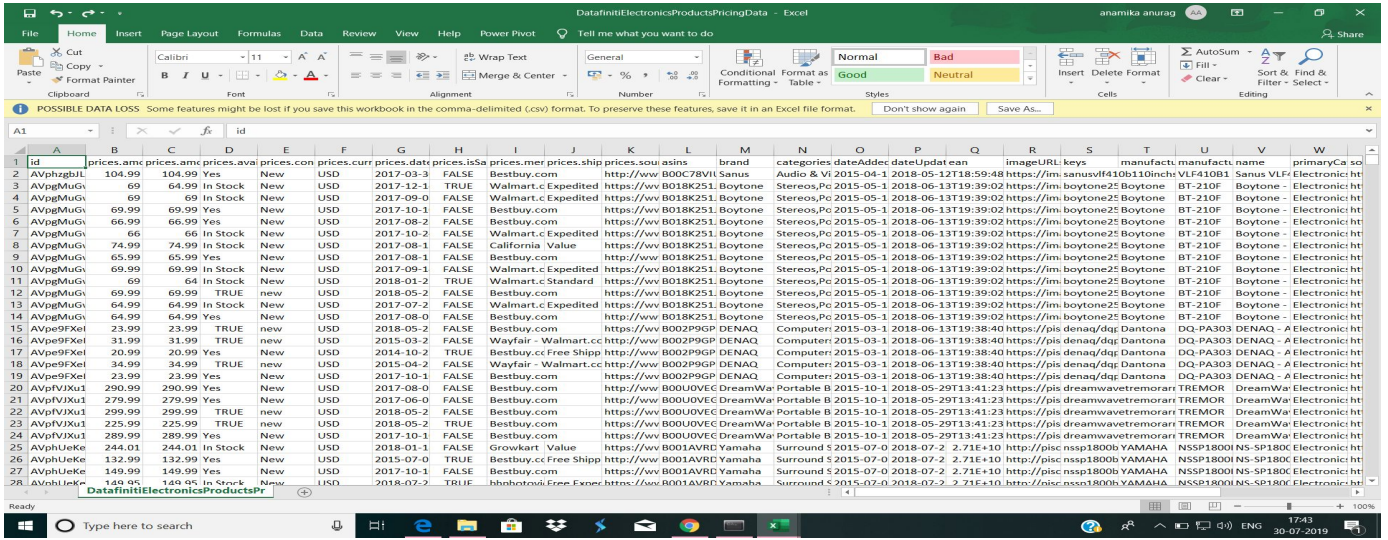
1. **Product Diffusion** :- contains the bass diffusion parameters of existing products.
2. **Product Attributes** :- various attributes of the products.

By taking the product attribute DB as inputs and product diffusion DB as targets,single prediction models are developed.

Datasets

The dataset was extracted from :-https://www.kaggle.com/data

Original Dataset:-



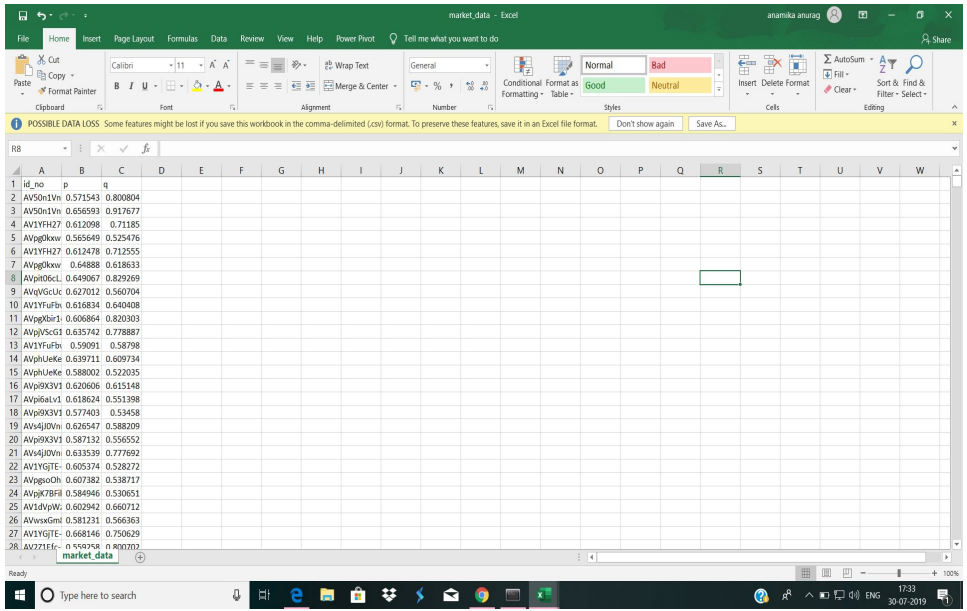
contd..

Product Diffusion

Out[4]:

	ASG	DN	DU	EI	NCG	NS	RTC	TIE	PL
0	1	1	2	5	5	3	2	0	1
1	1	1	2	5	5	3	2	0	5
2	1	0	2	4	1	1	2	0	4
3	0	0	3	2	1	3	3	0	1
4	1	1	2	4	1	1	2	0	4
5	1	0	3	2	1	4	3	0	4
6	1	1	5	4	3	2	4	0	5
7	1	0	3	2	4	4	2	1	3
8	0	0	2	3	1	2	2	0	4
9	1	1	4	5	5	2	4	0	3
10	1	1	4	3	3	1	3	0	5
11	1	1	2	3	1	1	2	0	3
12	1	0	4	1	1	3	4	1	4
13	1	1	4	1	1	4	4	1	1
14	0	0	4	2	1	2	4	0	4
15	1	1	4	1	1	3	4	0	3
16	0	0	4	2	1	2	4	0	2
17	1	0	3	3	4	4	2	0	3
18	0	0	4	2	1	1	4	0	3
19	1	0	3	3	4	1	2	0	5
20	1	1	2	1	3	4	2	0	2

Product Attributes



Final Dataset

t[4]:

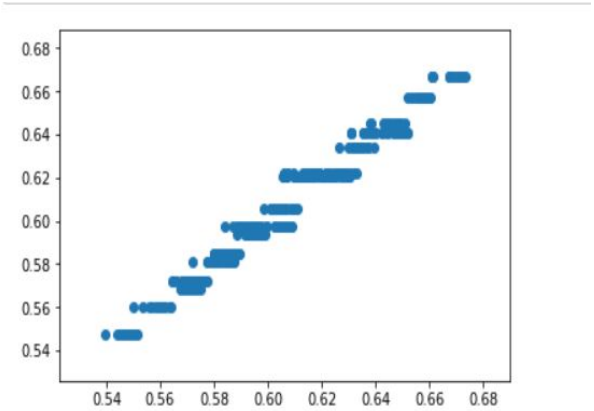
	ASG	DN	DU	EI	NCG	NS	RTC	TIE	PL	p	q
0	1	1	2	5	5	3	2	0	1	0.571543	0.800804
1	1	1	2	5	5	3	2	0	5	0.656593	0.917677
2	1	0	2	4	1	1	2	0	4	0.612098	0.711850
3	0	0	3	2	1	3	3	0	1	0.565649	0.525476
4	1	1	2	4	1	1	2	0	4	0.612478	0.712555
5	1	0	3	2	1	4	3	0	4	0.648880	0.618633
6	1	1	5	4	3	2	4	0	5	0.649067	0.829269
7	1	0	3	2	4	4	2	1	3	0.627012	0.560704
8	0	0	2	3	1	2	2	0	4	0.616834	0.640408
9	1	1	4	5	5	2	4	0	3	0.606864	0.820303
10	1	1	4	3	3	1	3	0	5	0.635742	0.778887
11	1	1	2	3	1	1	2	0	3	0.590910	0.587980
12	1	0	4	1	1	3	4	1	4	0.639711	0.609734
13	1	1	4	1	1	4	4	1	1	0.588002	0.522035
14	0	0	4	2	1	2	4	0	4	0.620606	0.615148
15	1	1	4	1	1	3	4	0	3	0.618624	0.551398
16	0	0	4	2	1	2	4	0	2	0.577403	0.534580
17	1	0	3	3	4	4	2	0	3	0.626547	0.588209
18	0	0	4	2	1	1	4	0	3	0.587132	0.556552
19	1	0	3	3	4	1	2	0	5	0.633539	0.777692
20	1	1	2	1	3	4	2	0	2	0.605374	0.528272

ALGORITHMS USED:

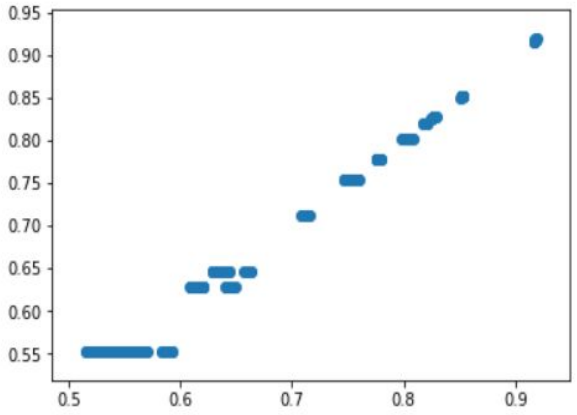
- KNN Regression
- Multivariate Linear Regression (MLR)
- Artificial Neural Networks (ANN)
- Classification And Regression Tree (CART)
- Gaussian Process Regression (GPR)

CART

CART is an alternative decision tree building algorithm which can handle both classification and regression tasks.



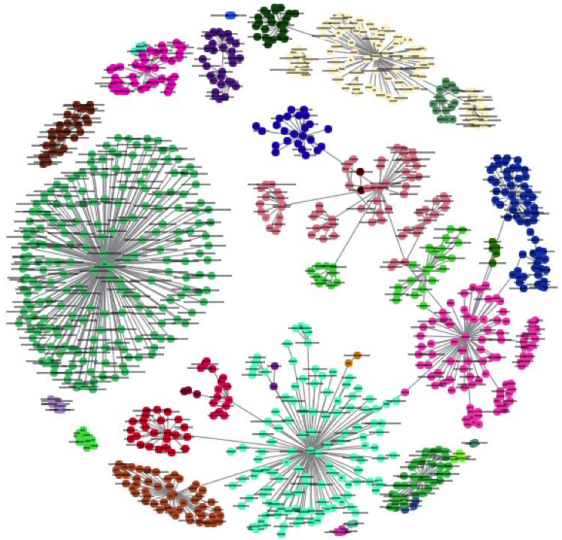
For p values



For q values

KNN Regression:

- It is one of the most simplest of all algorithms used for classification and regression both.
- It stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions).
- It works by taking the average of the numerical target of the K nearest neighbors.
- For determining optimal value of k , we calculate error for each k and then select that k which gives minimum error.



KNN Regression (contd):

- KNN is implemented by scikit-learn library. Here is the implementation

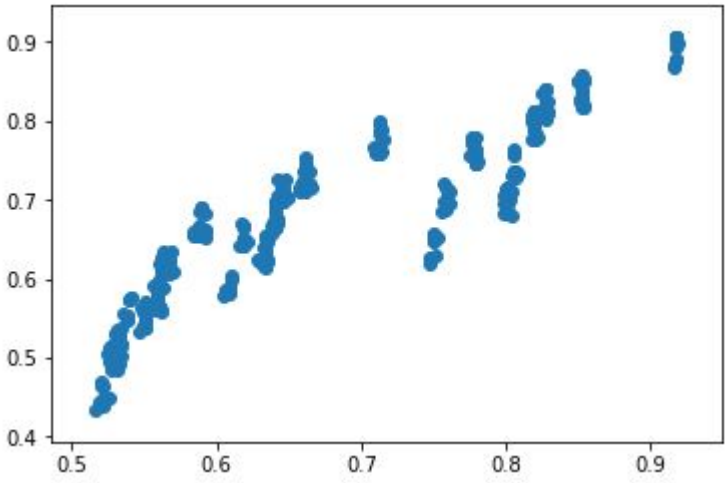
```
def model(x_train,y_train,x_test):
    model=neighbors.KNeighborsRegressor(n_neighbors=3)
    model.fit(x_train,y_train)
    pred_test=model.predict(x_test)
    return(pred_test)
```

```
p=model(x_train,y_train["p"],x_test)
```

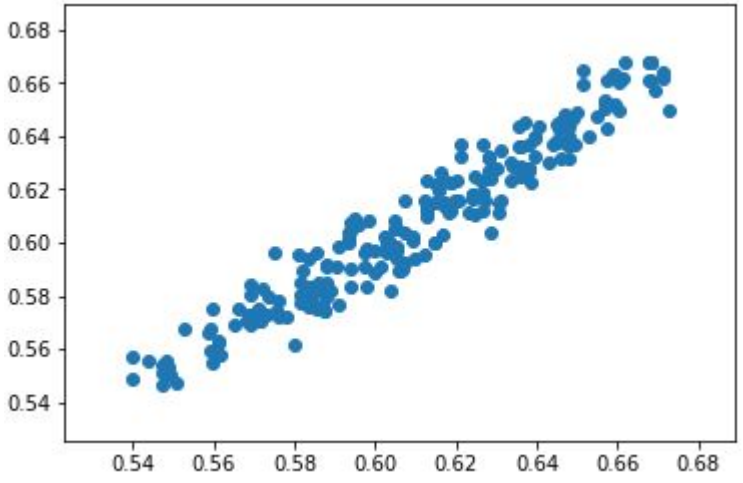
```
q=model(x_train,y_train["q"],x_test)
```

KNN Regression (contd):

- Here is the scatter plot between test and predicted value for p and q respectively



For p



For q

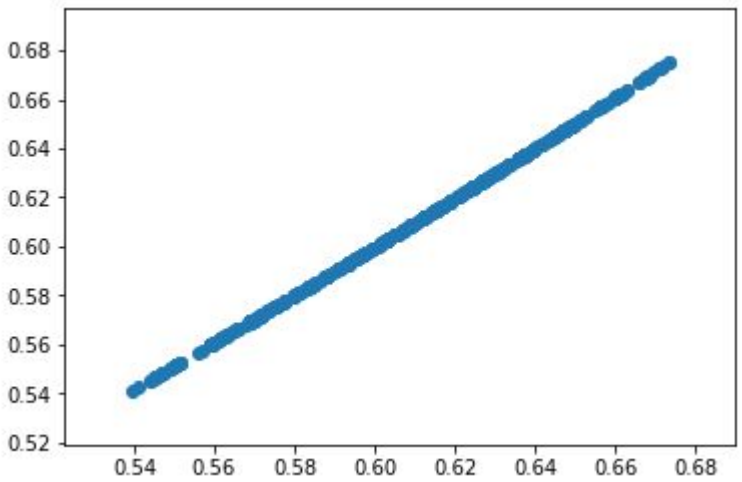
Multivariate Linear Regression:

- We have implemented MLR by using scikit-learn library.
- Here is the implementation

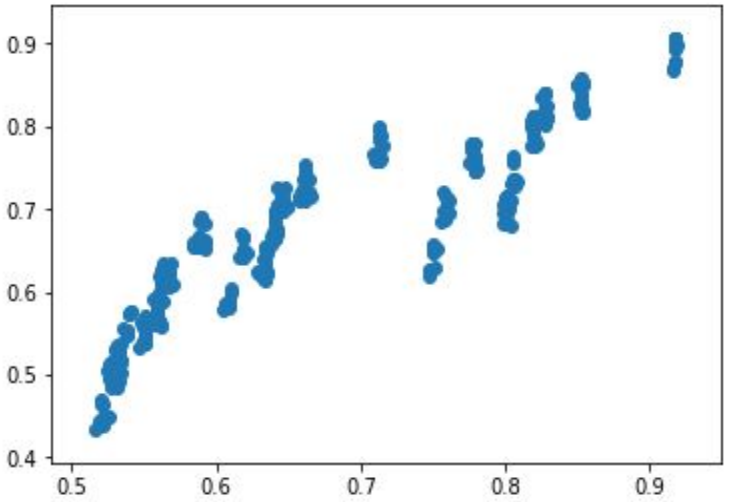
```
#Fitting MLR to X1_train and Y1_train
reg1 = LinearRegression()
reg1.fit(X1_train,Y1_train)
Y1_pred = reg1.predict(X1_test)
Y1_pred
```

Multivariate Linear Regression (contd):

- Here is the scatter plot between test and predicted value for p and q respectively



For p which is Y1



For q which is Y2

Artificial Neural Networks:

- It is also implemented using scikit-learn library. Implementation is shown as follows

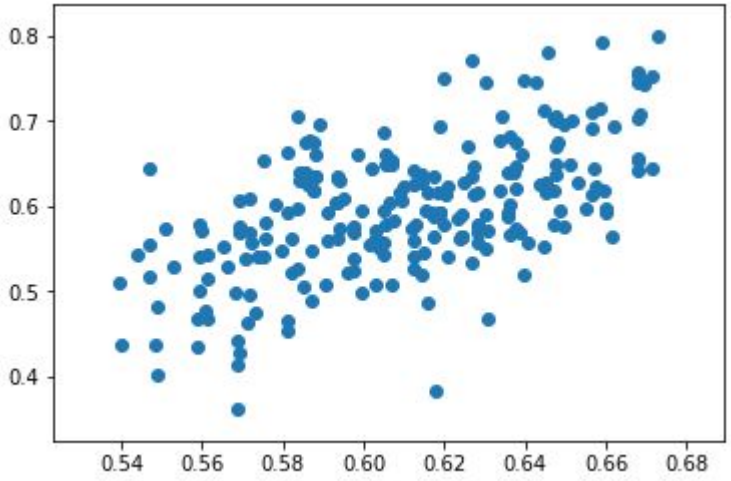
```
def model_nn(x_train,y_train,x_test):
    reg=MLPRegressor(alpha=0.0001,hidden_layer_sizes=(100,10))
    reg.fit(x_train,y_train)
    y_pred=reg.predict(x_test)
    return(y_pred)
```

```
p_nn=model_nn(x_train,y_train["p"],x_test)
```

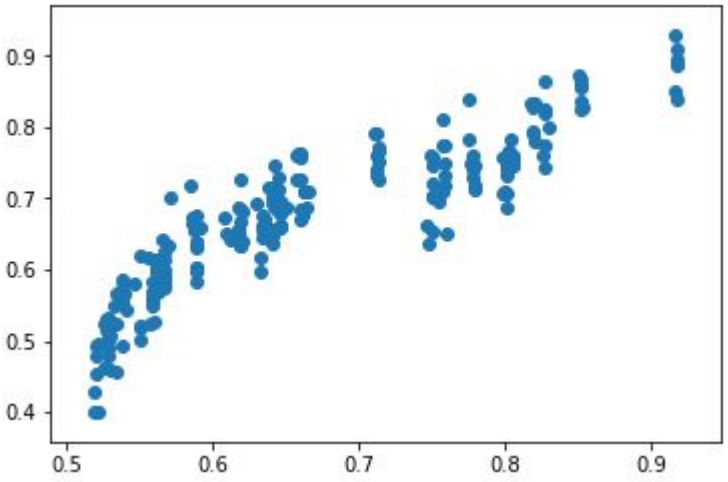
```
q_nn=model_nn(x_train,y_train["q"],x_test)
```

Artificial Neural Networks (Contd):

- Here is the scatter plot between test and predicted value for p and q respectively



For p



For q

Gaussian Process Regression:

- Gaussian process regression (GPR) is a nonparametric, Bayesian approach to regression.
- Unlike many popular supervised machine learning algorithms that learn exact values for every parameter in a function, the Bayesian approach infers a probability distribution over all possible values.
- In GPR, the target y is expressed as a linear combination of the inputs with a Gaussian noise as follows:

  $y = f(x) + \varepsilon$, $f(x) = x^T w$, where x is n dimensional input vector and w is n dimensional weight vector, assuming that the noise follows an independent, identically distributed (i.i.d.) Gaussian distribution with zero mean and variance σ^2 , $\varepsilon \sim N(0; \sigma)$

Gaussian Process Regression (contd):

- The posterior distribution is given as

$$p(w|y,X) = \frac{p(y|X,w)}{p(y|X)} \propto p(y|X,w)p(w)$$

Where p(w) is the prior distribution on w vector ,$w \sim N(0, \sum_p)$ where $\sum_p$ is the covariance matrix

- The likelihood, which is the probability density of the given data and parameters, can be directly obtained as

$$p(y|X,w) = \prod_{i=1}^{n} p(y_i|x_i,w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\,\sigma} e^{\left(-\frac{(y_i - x_i^T w)^2}{2\sigma^2}\right)}$$

Gaussian Process Regression (contd):

$$= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}|\mathbf{y}-\mathbf{X}^T\mathbf{w}|^2\right) = \mathcal{N}\left(\mathbf{X}^T\mathbf{w}, \sigma^2 I\right)$$

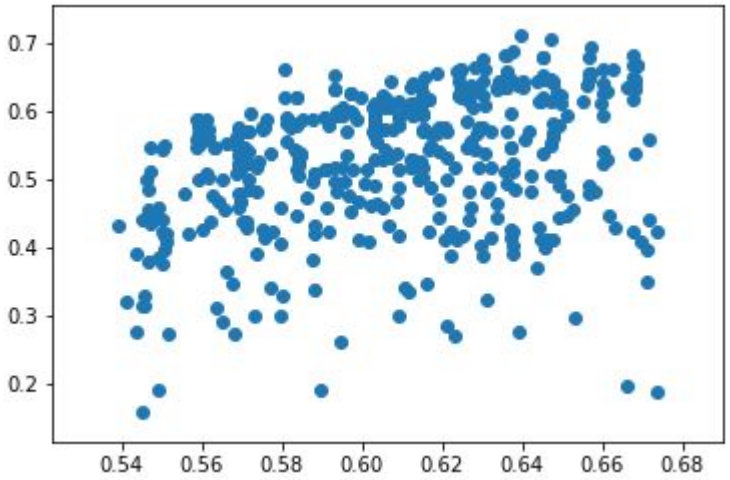
Gaussian Process Regression (contd):

- It is also implemented using scikit-learn library. Here is the implementation

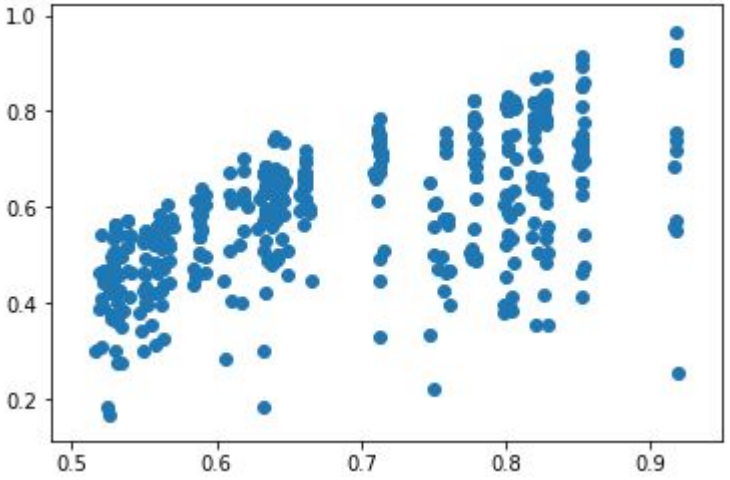
```
#Fitting GPR to X1_train and Y1_train
gpr1 = GaussianProcessRegressor(kernel=None, random_state=0)
gpr1.fit(X1_train,Y1_train)
gprY1_pred = gpr1.predict(X1_test)
gprY1_pred
```

Gaussian Process Regression (contd):

- Here is the scatter plot between test and predicted value for p and q respectively



For p



For q

Evaluation of Models

The prediction models were evaluated in terms of -

1. Mean Absolute Error(MAE)

MAE= $\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|,$

2. Root Mean Squared Error(RMSE)

RMSE= $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2},$

Results

- The prediction performances of each regression algorithm for p and q is given in the form of table

Algorithm	p		q	
	MAE	RMSE	MAE	RMSE
MLR	0.00450	0.0055	0.04057	0.05026
KNN	0.00685	0.00869	0.02128	0.03443
ANN	0.00685	0.00869	0.04198	0.05086
CART	0.00655	0.0080	0.048	0.07
GPR	0.08427	0.12307	0.09815	0.14767

References

- https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html
- https://towardsdatascience.com/quick-start-to-gaussian-process-regression-36d838810319
- https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/