statinfer

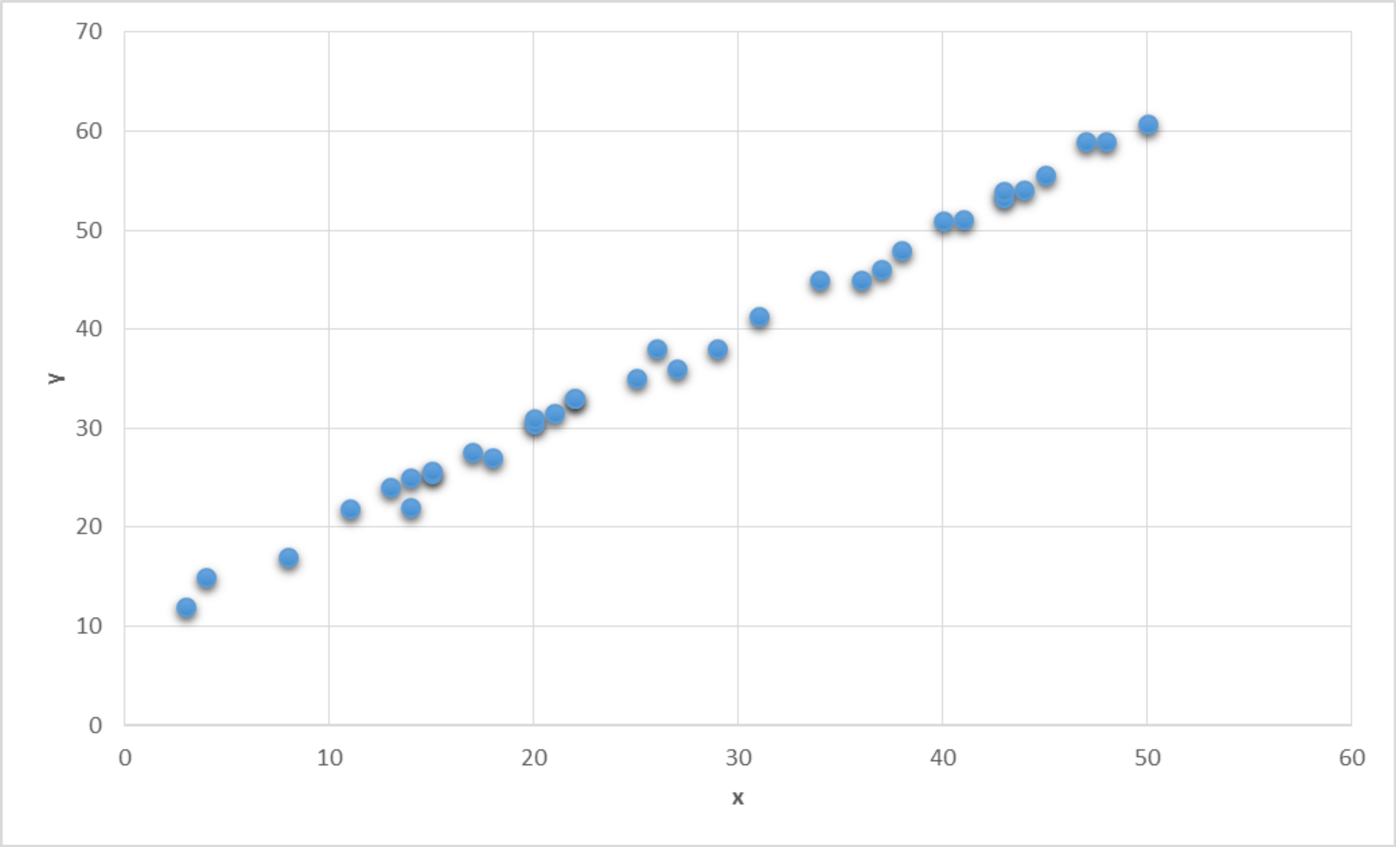
Linear Regression and Logistic Regression

Venkata Reddy Konasani

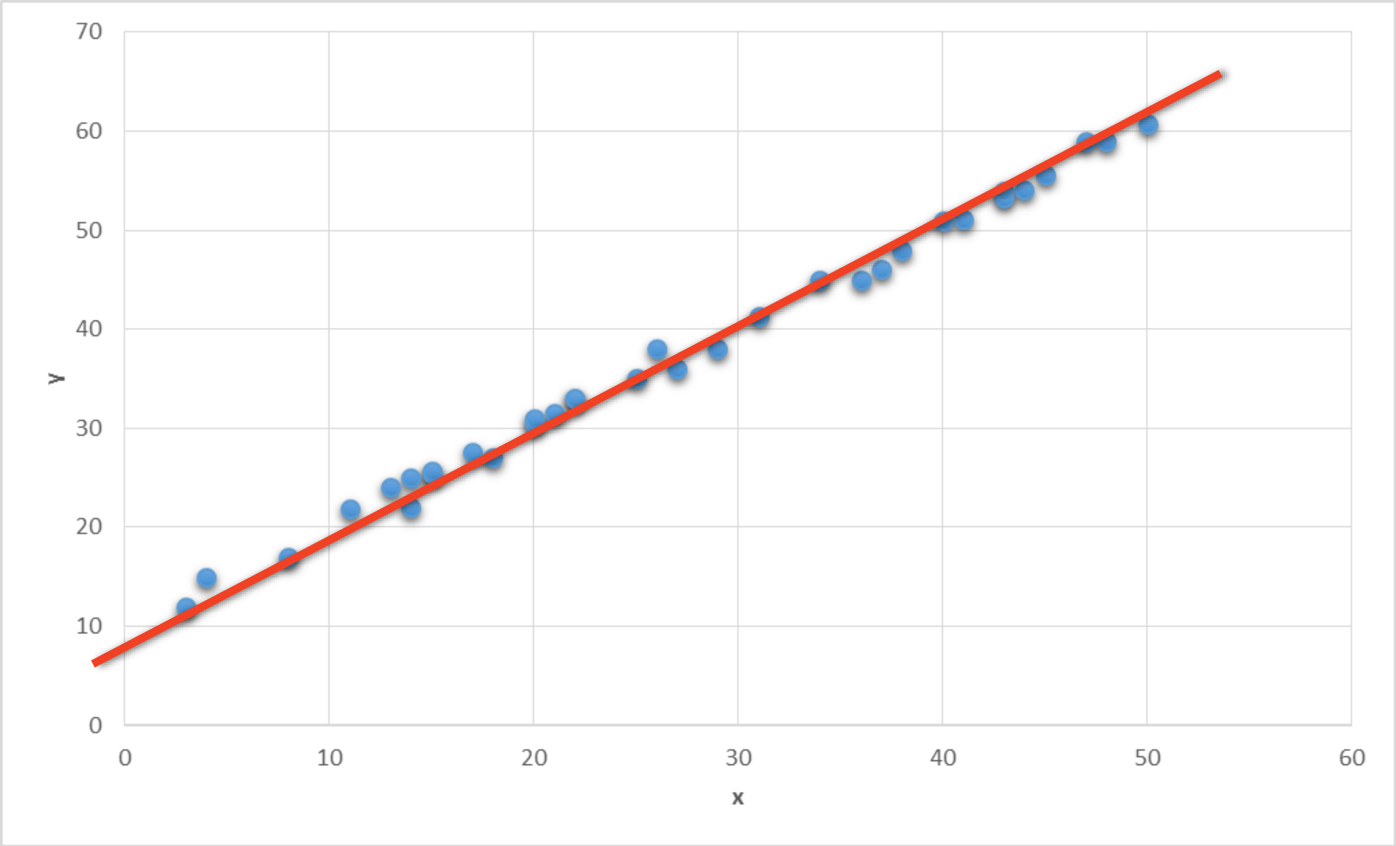
Contents

- Simple Regression
- R-Squared
- Multiple Regression
- Logistic Regression
- Accuracy
- Multiple Logistic Regression

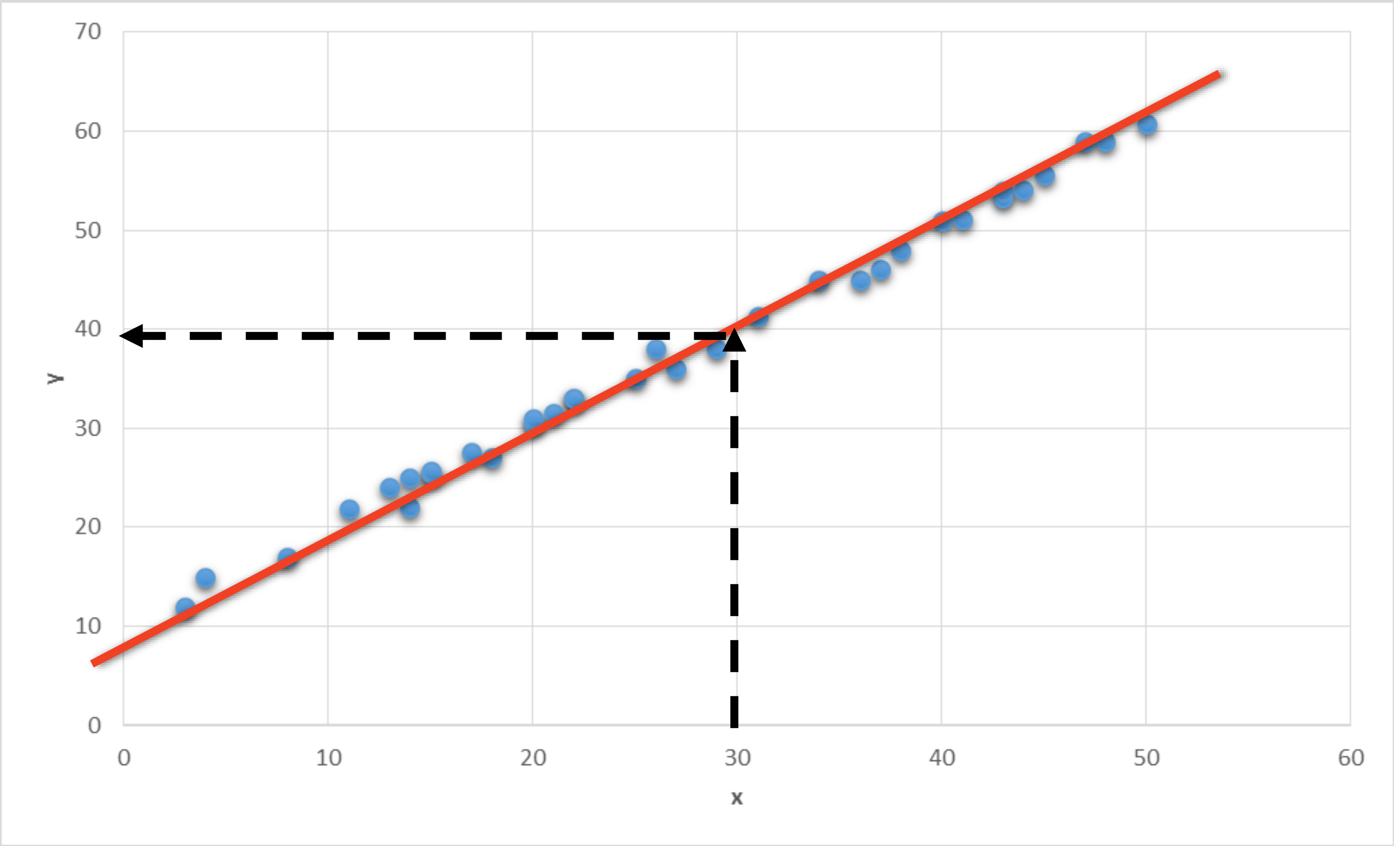
X Vs Y



Prediction



Prediction



Line Equation

Straight Line equation

$$y = mx + c$$

Regression terminology

$$y = \beta_0 + \beta_1 x$$

What is Regression

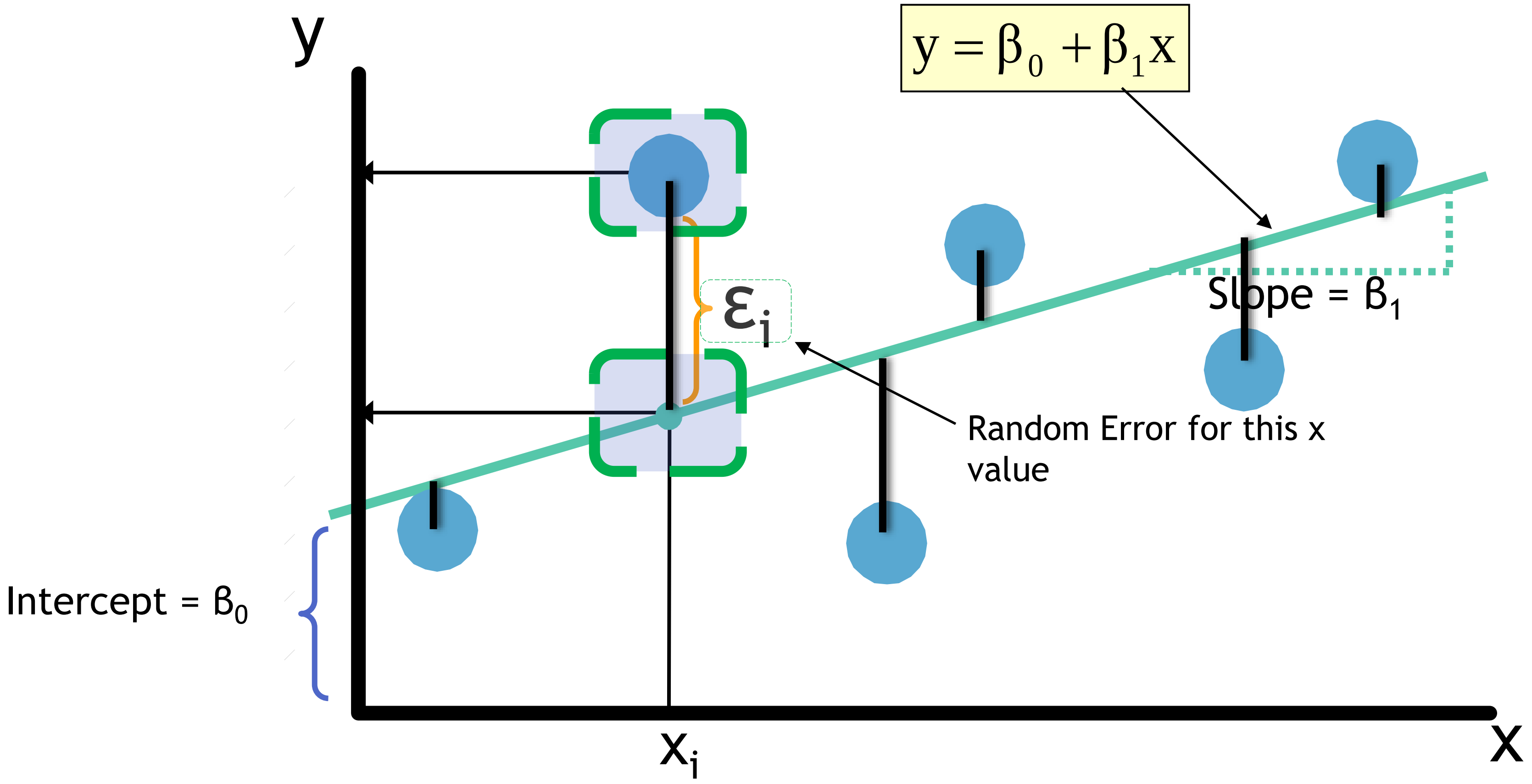
- A regression line is a mathematical formula that quantifies the general relation between a predictor/independent (or known variable x) and the target/dependent (or the unknown variable y)
- Below is the regression line. If we have the data of x and y then we can build a model to generalize their relation

- What is the best fit for our data?
- The one which goes through the core of the data
- The one which minimizes the error

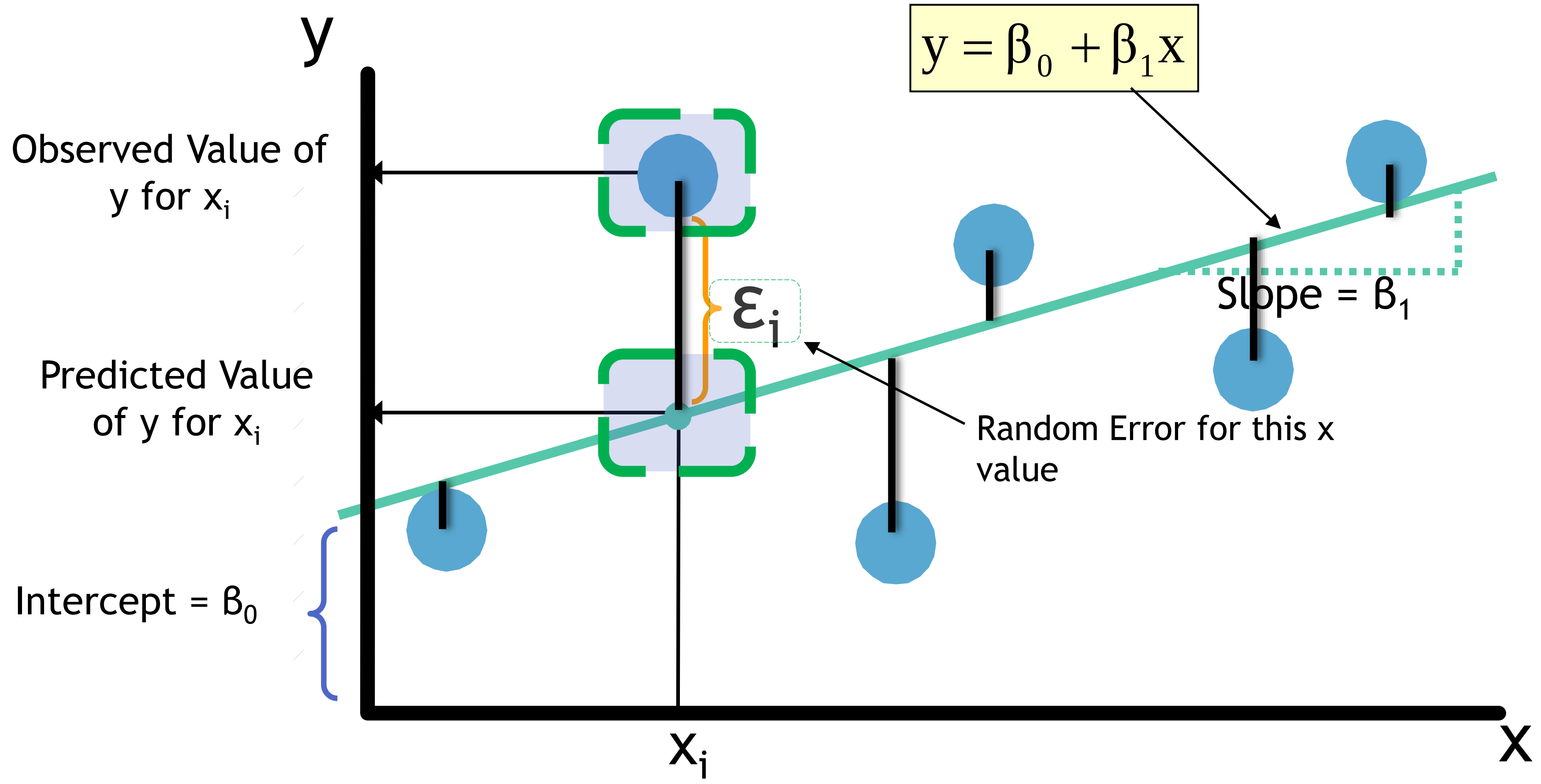
$$\boxed{y = \beta_0 + \beta_1 x}$$

Regression Line fitting-Least Squares Estimation

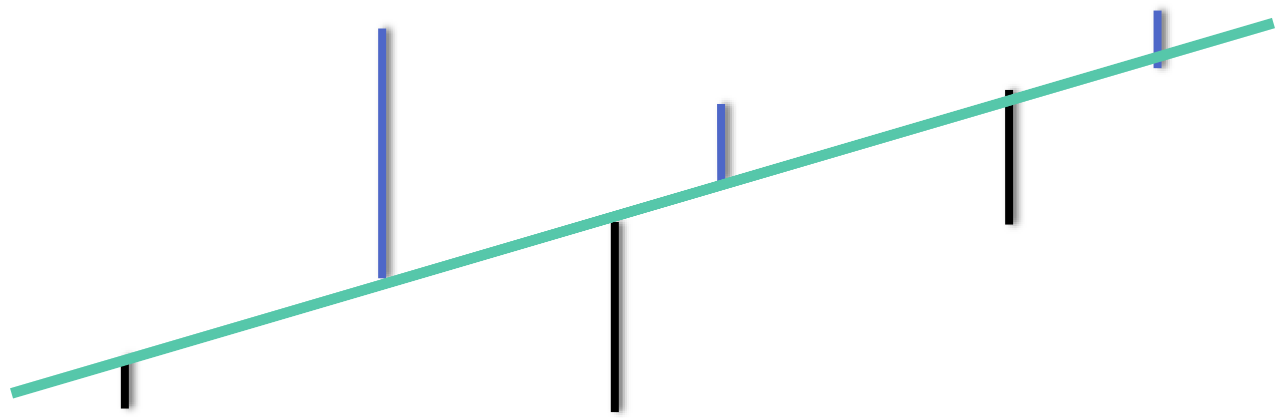
Regression Line fitting



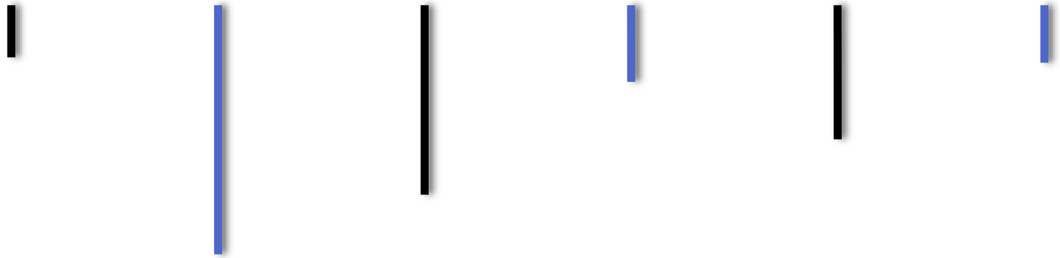
Regression Line fitting



Regression Line fitting



Minimizing the error



- The best line will have the minimum error
- Some errors are positive and some errors are negative. Taking their sum is not a good idea
- We can either minimize the squared sum of errors Or we can minimize the absolute sum of errors
- Squared sum of errors is mathematically convenient to minimize
- The method of minimizing squared sum of errors is called least squared method of regression

Least Squares Estimation

- X: x1, x2, x3, x4, x5, x6, x7,……..
- Y:y1, y2, y3, y4, y5, y6, y7…….
- Imagine a line through all the points
- Deviation from each point (residual or error)
- Square of the deviation
- Minimizing sum of squares of deviation

$$\sum e^2 = \sum (y - \hat{y})^2$$
$$= \sum (y - (\beta_0 + \beta_1 x))^2$$

β_0 and β_1 are obtained by minimize the sum of the squared residuals

LAB: Regression Line Fitting

- Dataset: Air Travel Data\Air_travel.csv
- Fit a regression line between Promotion_Budget and Passengers

Code: sklearn vs statsmodels

- Several package options for building regression lines in python
- sklearn and statsmodels are two most widely used options
- sklean is first choice. But gives limited summary statistics
- But statmodels gives well formatted (R-like) summary and model statistics.
- You can use any one of them. Use sklearn of you are not interested in model statistics. Use stastmodels when you are at learning phase.

- We will use both

Code: Regression Line Fitting

```
import statsmodels.formula.api as sm
model = sm.ols(formula='Passengers ~ Promotion_Budget', data=air)
fitted1 = model.fit()
fitted1.summary()
```

How good is my regression line?

Two models

- Model-1 : Passengers vs. Promo budget
- Model-2: Passengers vs. inter metro flight ratio

- Model-1 vs Model-2 to predict the same target. Which model to pick?

How good is my regression line?

Model-1

X1	Y Actual	Y Pred
	30K	31K
	40K	39K
	35K	35K
	27K	26K
	32K	32K
	33K	35K
	28K	26K

Model-2

X2	Y Actual	Y Pred
	30K	42K
	40K	49K
	35K	15K
	27K	20K
	32K	32K
	33K	38K
	28K	20K

SSE

X1	Y Actual	Y Pred	Error
	30K	31K	
	40K	39K	
	35K	35K	
	27K	26K	1K
	32K	32K	
	33K	35K	
	28K	26K	

SSE

X1	Y Actual	Y Pred	Error
	30K	31K	-1K
	40K	39K	1K
	35K	35K	0K
	27K	26K	1K
	32K	32K	0K
	33K	35K	-2K
	28K	26K	2K

SSE

X1	Y Actual	Y Pred	Error	Squared Error
	30K	31K	-1K	
	40K	39K	1K	
	35K	35K	0K	
	27K	26K	1K	
	32K	32K	0K	
	33K	35K	-2K	
	28K	26K	2K	
				SSE

SSE, SSR and SST

X1	Y Actual	Y Pred	Error	Squared Error
	30K	31K	-1K	
	40K	39K	1K	
	35K	35K	0K	
	27K	26K	1K	
	32K	32K	0K	
	33K	35K	-2K	
	28K	26K	2K	
	SST	SSR		SSE

How good is my regression line?

- Take an (x,y) point from data.
- Imagine that we submitted x in the regression line, we got a prediction as y_{pred}
- If the regression line is a good fit then the we expect $y_{pred}=y$ or $(y-y_{pred}) = 0$
- At every point of x, if we repeat the same, then we will get multiple error values $(y-y_{pred})$ values
- Some of them might be positive, some of them may be negative, so we can take the square of all such errors

$$\boxed{SSE = \sum (y - \hat{y})^2}$$

SSE

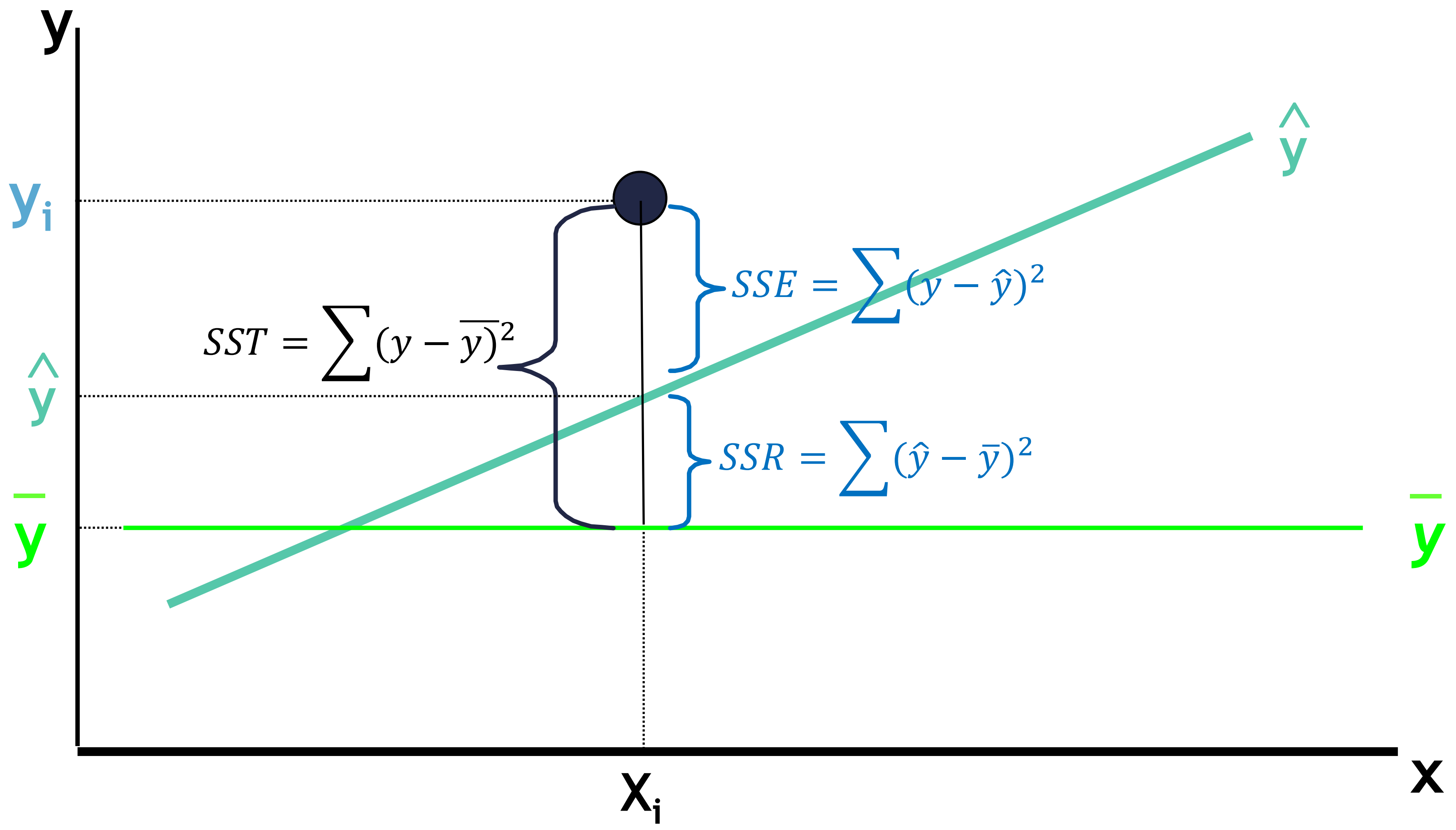
- For a good model we need SSE to be zero or near to zero
- Standalone SSE will not make any sense, For example SSE= 100, is very less when y is varying in terms of 1000's. Same value is is very high when y is varying in terms of decimals.
- We have to consider variance of y while calculating the regression line accuracy

$$\boxed{SSE = \sum (y - \hat{y})^2}$$

How good is my regression line?

- Error Sum of squares (SSE- Sum of Squares of error)
  - $SSE = \sum(y - \hat{y})^2$
- Total Variance in Y (SST- Sum of Squares of Total)
  - $SST = \sum(y - \overline{y})^2$
  - $SST = \sum(y - \hat{y} + \hat{y} - \overline{y})^2$
  - $SST = \sum(y - \hat{y} + \hat{y} - \overline{y})^2$
  - $SST = \sum(y - \hat{y})^2 + \sum(\hat{y} - \bar{y})^2$
  - $\mathrm{SST} = \mathrm{SSE} + \sum(\hat{y} - \bar{y})^2$
  - $\mathrm{SST} = \mathrm{SSE} + SSR$
- So, total variance in Y is divided into two parts,
  - Variance that can't be explained by x (error)
  - Variance that can be explained by x, using regression

Explained and Unexplained Variation



statinfer

How good is my regression line?

- So, total variance in Y is divided into two parts,
  - Variance that can be explained by x, using regression
  - Variance that can't be explained by x

$$SST = SSE + SSR$$

- Total sum of Squares

Sum of Squares Error

Sum of Squares Regression

$$SST = \sum (y - \bar{y})^2$$

$$SSE = \sum (y - \hat{y})^2$$

$$SSR = \sum (\hat{y} - \bar{y})^2$$

R-Squared

R-Squared

- A good fit will have
  - SSE (Minimum or Maximum?)
  - SSR (Minimum or Maximum?)
  - And we know SST= SSE + SSR
  - SSE/SST(Minimum or Maximum?)
  - SSR/SST(Minimum or Maximum?)
- The coefficient of determination is the portion of the total variation in the dependent variable that is explained by variation in the independent variable
- The coefficient of determination is also called R-squared and is denoted as R^2

$$\boxed{R^2 = \frac{SSR}{SST}}$$ where $$\boxed{0 \le R^2 \le 1}$$

Lab: R- Square

- What is the R-square value of Passengers vs Promotion_Budget model?
- What is the R-square value of Passengers vs Inter_metro_flight_ratio

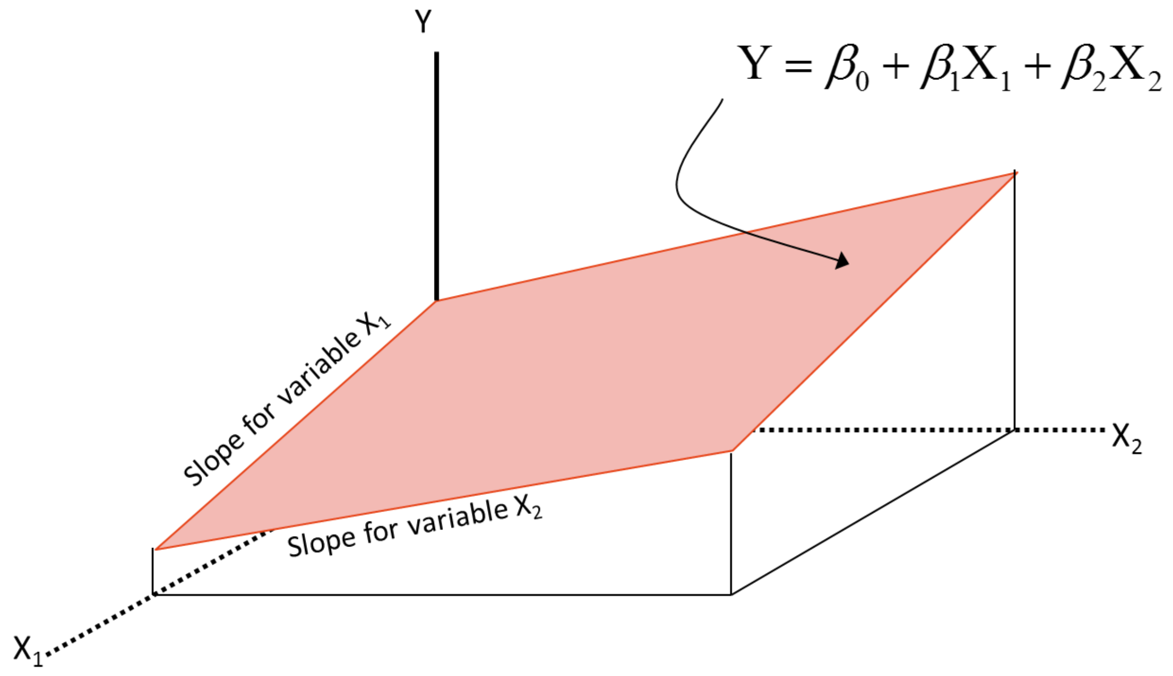
Code: R- Square

```
#What is the R-square value of Passengers vs Promotion_Budget model?
fitted1.summary()

#What is the R-square value of Passengers vs Inter_metro_flight_ratio
fitted2.summary()
```

Multiple Regression

Multiple Regression

- Using multiple predictor variables instead of single variable
- We need to find a perfect plane here



Code-Multiple Regression

```
import statsmodels.formula.api as sm

model = sm.ols(formula='Passengers ~ Promotion_Budget +
Inter_metro_flight_ratio + Service_Quality_Score ', data=air)

fitted = model.fit()
fitted.summary()
```

Logistic Regression

Venkat Reddy

What is the need of non-linear regression?

LAB: Need of logistic regression?

- Dataset: Product Sales Data/Product_sales.csv
- What are the variables in the dataset?
- Build a predictive model for Bought vs Age
- What is R-Square?
- If Age is 4 then will that customer buy the product?
- If Age is 105 then will that customer buy the product?

Code: Need of logistic regression?

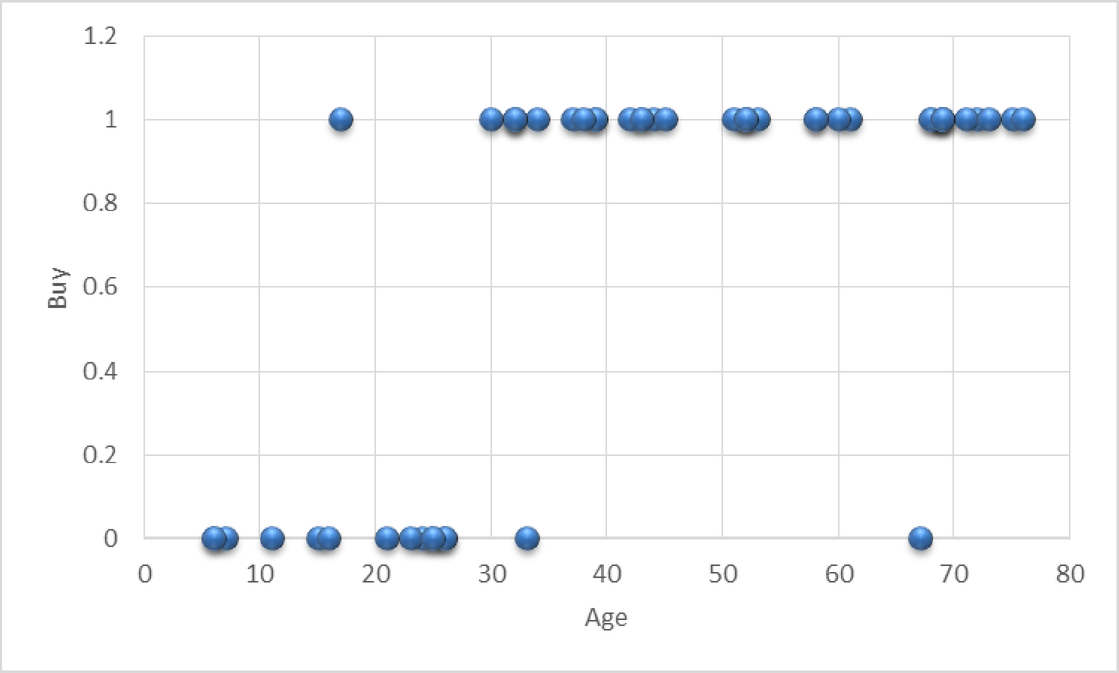
```
import sklearn as sk
from sklearn import linear_model

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(sales[["Age"]], sales[["Bought"]])

d1=pd.DataFrame({“age1”:[4]})
predict1=lr.predict(d1)
predict1

d2=pd.DataFrame({“age1”:[105]})
predict1=lr.predict(d2)
predict1
```

Something wrong

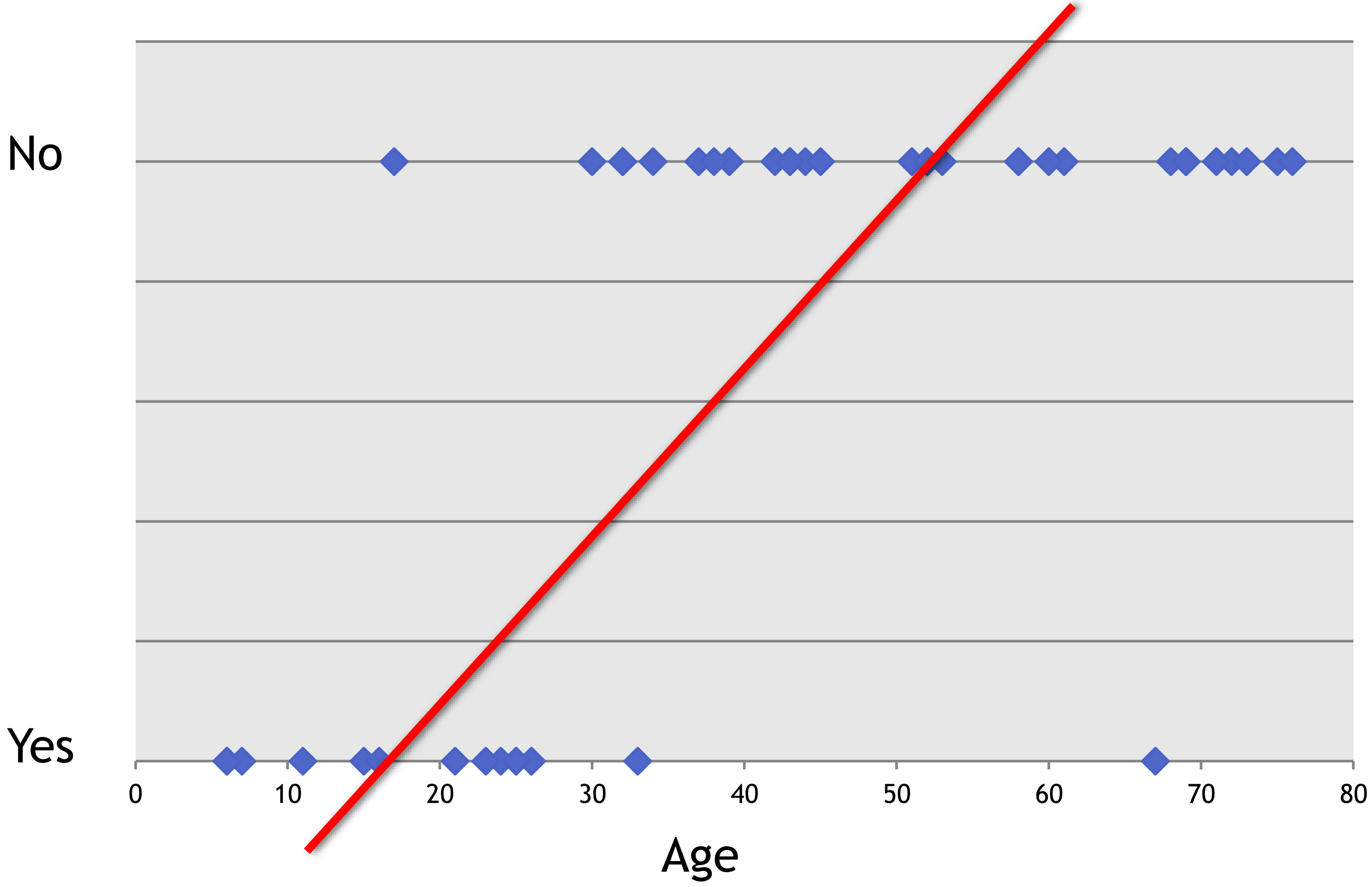
- The model that we built above is not right.
- There is certain issues with the type of dependent variable
- The dependent variable is not continuous it is binary
- We can’t fit a linear regression line to this data

Why not linear ?

- Consider Product sales data. The dataset has two columns.
  - Age – continuous variable between 6-80
  - Buy(0- Yes ; 1-No)



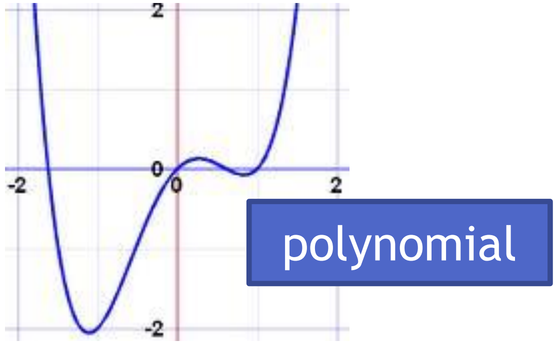
Why not linear ?

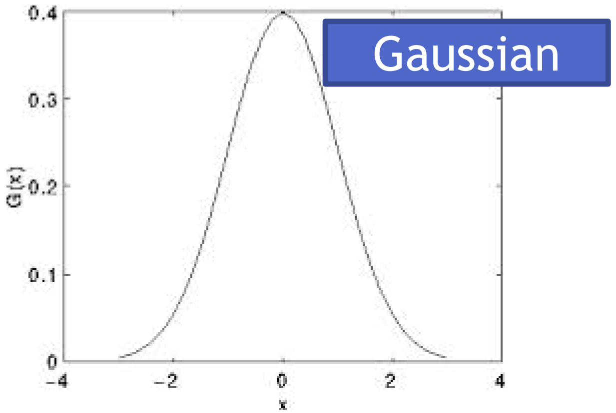


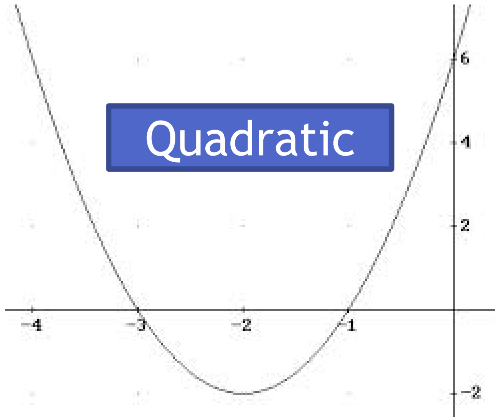
Real-life examples

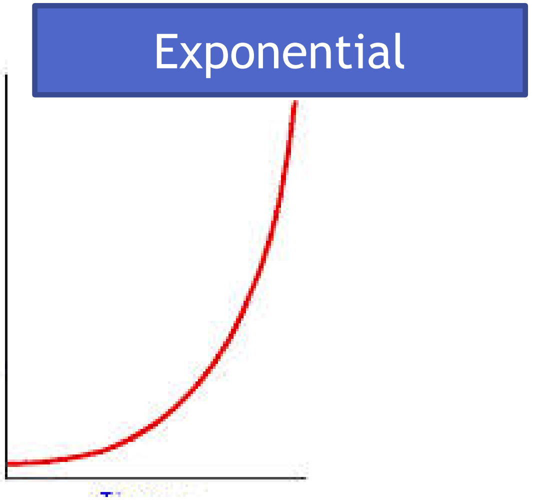
- Sales - Buying vs. Not buying
- Marketing – Response vs. No Response
- Credit card & Loans – Default vs. Non Default
- Operations – Attrition vs. Retention
- Websites – Click vs. No click
- Fraud identification –Fraud vs. Non Fraud
- Healthcare –Cure vs. No Cure

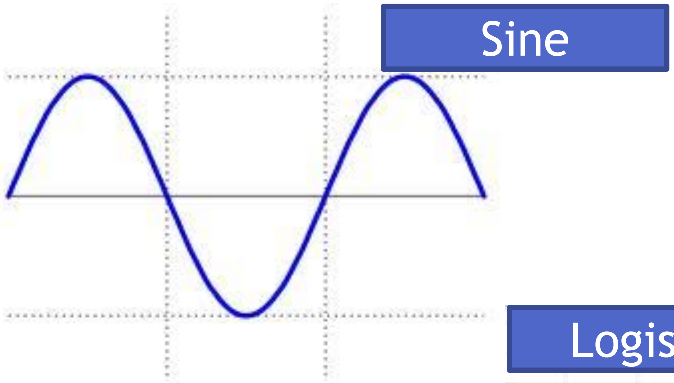
Some Nonlinear functions

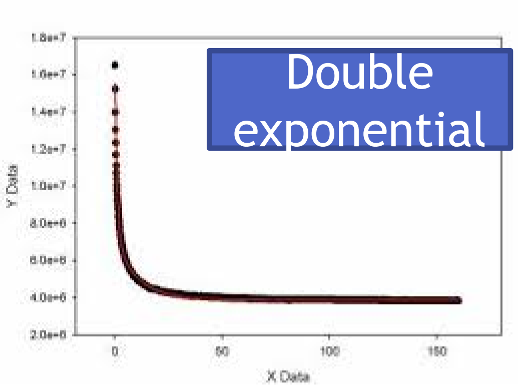


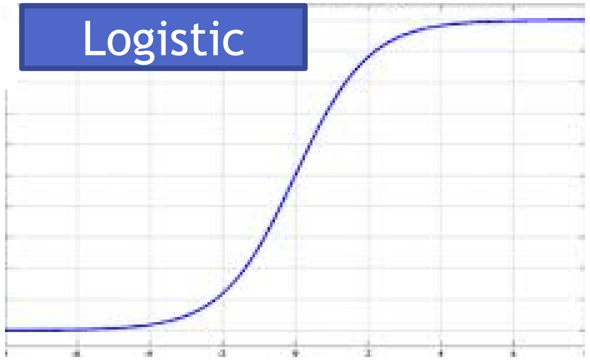




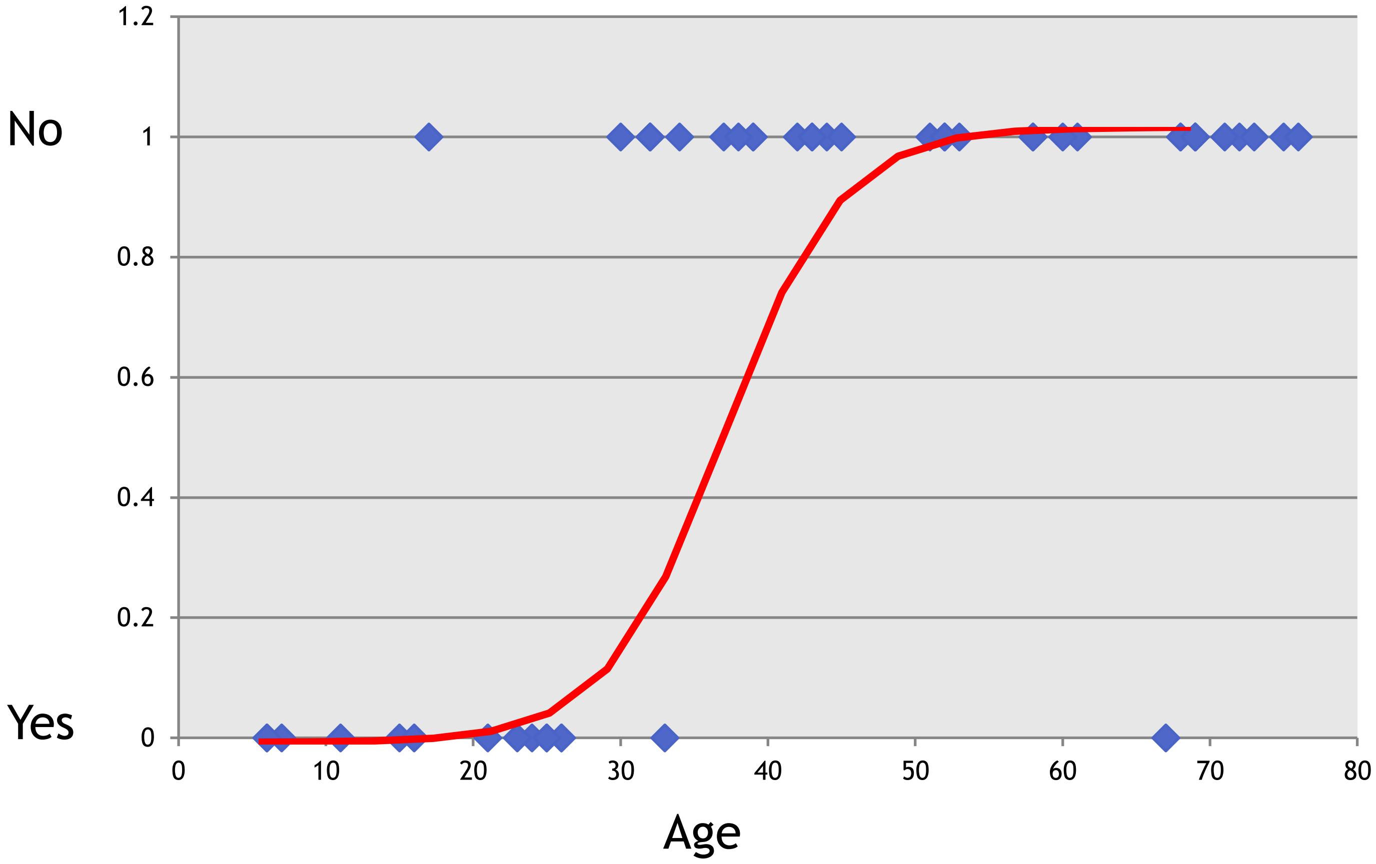








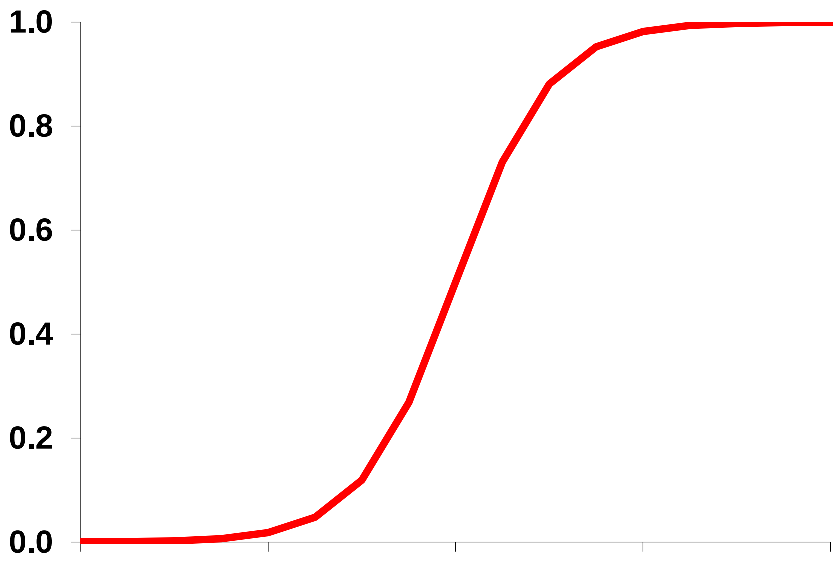
A Logistic Function



The Logistic function

- We want a model that predicts probabilities between 0 and 1, that is, S-shaped.
- There are lots of s-shaped curves. We use the logistic model:

$$y = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$



Logistic Regression Output

- In logistic regression, we try to predict the probability instead of direct values
- Y is binary, it takes only two values 1 and 0 instead of predicting 1 or 0 we predict the probability of 1 and probability of zero
- This suits aptly for the binary categorical outputs like YES vs NO; WIN vs LOSS; Fraud vs Non Fraud

Logistic Regression Line

Lab: Logistic Regression

- Dataset: Product Sales Data/Product_sales.csv
- Build a logistic Regression line between Age and buying
- A 4 years old customer, will he buy the product?
- If Age is 105 then will that customer buy the product?

Code: Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression()
logistic.fit(sales[["Age"]],sales["Bought"])

logistic.coef_
logistic.intercept_
#A 4 years old customer, will he buy the product?
#age1=4
predict_age1=logistic.predict(d1)
print(predict_age1)

#If Age is 105 then will that customer buy the product?
#age2=105
predict_age2=logistic.predict(d2)
print(predict_age2)
```

Linear Regression vs Logistic Regression

1. Predicting loss percentage
2. Predicting Buying vs. Not buying
3. Predicting number of customers
4. Predicting Response vs. No Response
5. Predicting revenue
6. Predicting the product price
7. Predicting Attrition vs. Retention
8. Predicting Click vs. No click
9. Predicting-Fraud vs. Non Fraud
10. Predicting the amount of fraud

Multiple Logistic Regression

Multiple Logistic Regression

$$y = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\cdots+\beta_k x_k}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\cdots+\beta_k x_k}}$$

Multiple Logistic Regression

- The dependent variable is binary
- Instead of single independent/predictor variable, we have multiple predictors
- Like buying / non-buying depends on customer attributes like age, gender, place, income etc.,

LAB: Multiple Logistic Regression

- Dataset: Fiberbits/Fiberbits.csv
  - Active_cust variable indicates whether the customer is active or already left the network.
- Build a model to predict the chance of attrition for a given customer using all the features.

Code: Multiple Logistic Regression

```
Fiber=pd.read_csv("D:\\Google Drive\\Training\\Datasets\\Fiberbits\\Fiberbits.csv")
list(Fiber.columns.values)  ###to get variables list

#Build a model to predict the chance of attrition for a given customer using all the features.
from sklearn.linear_model import LogisticRegression
logistic1= LogisticRegression()

###fitting logistic regression for active customer on rest of the variables#######
logistic1.fit(Fiber[["income"]+['months_on_network']+['Num_complaints']+['number_plan_changes']+['relocated']+['monthly_bill']+['technical_issues_per_month']+['Speed_test_result']],Fiber[['active_cust']])
```

Goodness of fit for a logistic regression

Actual vs Predicted

x1	x2	x3	..	Y actual	Y pred
.	.	.	.	0	0
				0	1
				0	0
				1	0
				1	1
				1	1
				0	1
				1	1
				0	0
				0	0

4	2
1	3

Actual vs Predicted

Y actual	Y pred
0	0
0	1
0	0
1	0
1	1
1	1
0	1
1	1
0	0
0	0

	Y Predicted		
		0	1
Y Actual	0		
	1		

Classification Table & Accuracy

		Predicted	
		0	1
Actual	0	True positive (TP) Zero Predicted as Zero	False Negatives(FN) Zero Predicted as One
	1	False positive (FP) One Predicted as Zero	True Negatives(TN) One Predicted as One

- Also known as confusion matrix
- Accuracy=(TP+TN)/(TP+FN+FP+TN)

LAB: Confusion Matrix & Accuracy

- Create confusion matrix for Fiber bits model
- Find the accuracy value for fiber bits model
- Change try three different threshold values and note down the changes in accuracy value

Code: Confusion Matrix & Accuracy

```
from sklearn.~~cross_validation~~ import train_test_split
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix###for using confusion matrix###

predict1=logistic1.predict(Fiber[["income"]+['months_on_network']+['Num_complaints']+['number_plan_changes']+['relocated']+['monthly_bill']+['technical_issues_per_month']+['Speed_test_result']])
predict1

cm1 = confusion_matrix(Fiber[['active_cust']],predict1)
print(cm1)

#####from confusion matrix calculate accuracy
total1=sum(sum(cm1))
print(total1)

accuracy1=(cm1[0,0]+cm1[1,1])/total1
accuracy1
```

Cross Validation

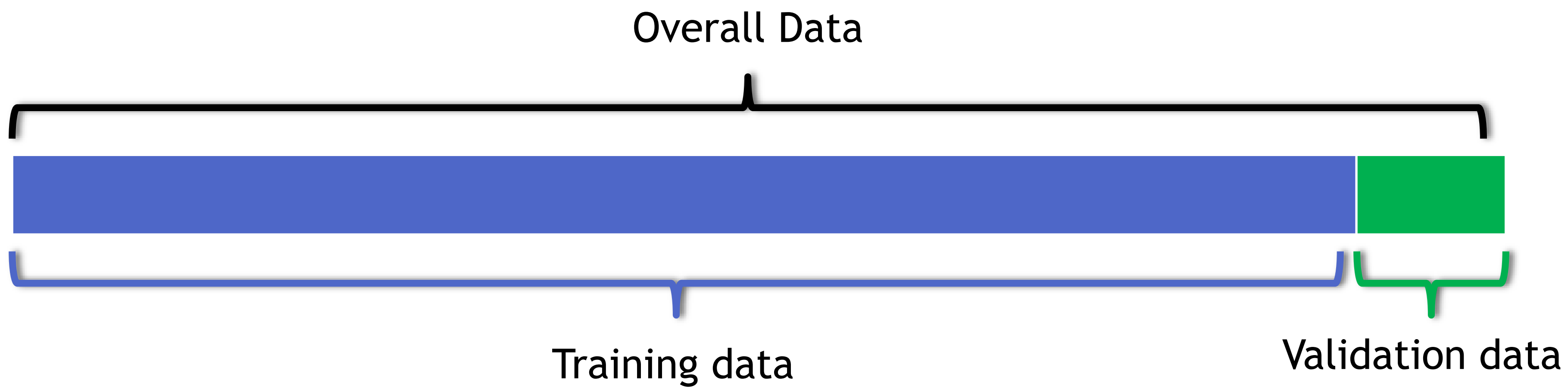
The Training Error

- The accuracy of our best model is 95%. Is the 5% error model really good?
- The error on the training data is known as training error.
- A low error rate on training data may not always mean the model is good.
- What really matters is how the model is going to perform on unknown data or test data.
- We need to find out a way to get an idea on error rate of test data.
- We may have to keep aside a part of the data and use it for validation.
- There are two types of datasets and two types of errors

statinfer

Two types of datasets

- There are two types of datasets
  - **Training set:** This is used in model building. The input data
  - **Test set:** The unknown dataset. This dataset is gives the accuracy of the final model
- We may not have access to these two datasets for all machine learning problems. In some cases, we can take 90% of the available data and use it as training data and rest 10% can be treated as validation data



Types of errors

- The training error
  - The error on training dataset
  - In-time error
  - Error on the known data
  - Can be reduced while building the model
- The test error
  - The error that matters
  - Out-of-time error
  - The error on unknown/new dataset.

"A good model will have both training and test error very near to each other and close to zero"

The problem of over-fitting

- The model is made really complicated, that it is very sensitive to minimal changes
- By complicating the model the variance of the parameters estimates inflates
- Model tries to fit the irrelevant characteristics in the data
- Over fitting
  - The model is super good on training data but not so good on test data
  - Less training error, high testing error
  - The model is over complicated with too many predictors
  - Model need to be simplified
  - A model with lot of **variance**

LAB : Cross Validation

```
from sklearn import model_selection
train_data,test_data = model_selection.train_test_split(Fiber, test_size=0.2)

print("train Data Shape ",train_data.shape)
print("test Data Shape  ",test_data.shape)
```

```
train Data Shape  (80000, 9)
test Data Shape   (20000, 9)
```

LAB : Cross Validation

```
logistic2= LogisticRegression(max_iter=200)
###fitting logistic regression for active customer on rest of the variables#######
logistic2.fit(train_data[["income"]+['months_on_network']+['Num_complaints']+['number_plan_changes']+['relocated']+['monthly_bill']+['technical_issues_per_month']+['Speed_test_result']],train_data[['active_cust']])

predict=logistic2.predict(train_data[["income"]+['months_on_network']+['Num_complaints']+['number_plan_changes']+['relocated']+['monthly_bill']+['technical_issues_per_month']+['Speed_test_result']])
cm_train = confusion_matrix(train_data[['active_cust']],predict)
accuracy_train=(cm_train[0,0]+cm_train[1,1])/sum(sum(cm_train))
print("accuracy on train data" , accuracy_train)

predict=logistic2.predict(test_data[["income"]+['months_on_network']+['Num_complaints']+['number_plan_changes']+['relocated']+['monthly_bill']+['technical_issues_per_month']+['Speed_test_result']])
cm_test = confusion_matrix(test_data[['active_cust']],predict)
accuracy_test=(cm_test[0,0]+cm_test[1,1])/sum(sum(cm_test))
print("accuracy on test data" , accuracy_test)
```

```
accuracy on train data 0.8663125
accuracy on test data 0.8601
```

statinfer

Thank you