



Logistic Regression

Venkat Reddy



Contents

Contents

- What is the need of logistic regression?
- Building logistic Regression line
- Goodness of fit measures
- Multicollinearity
- Individual Impact of variables
- Model selection



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

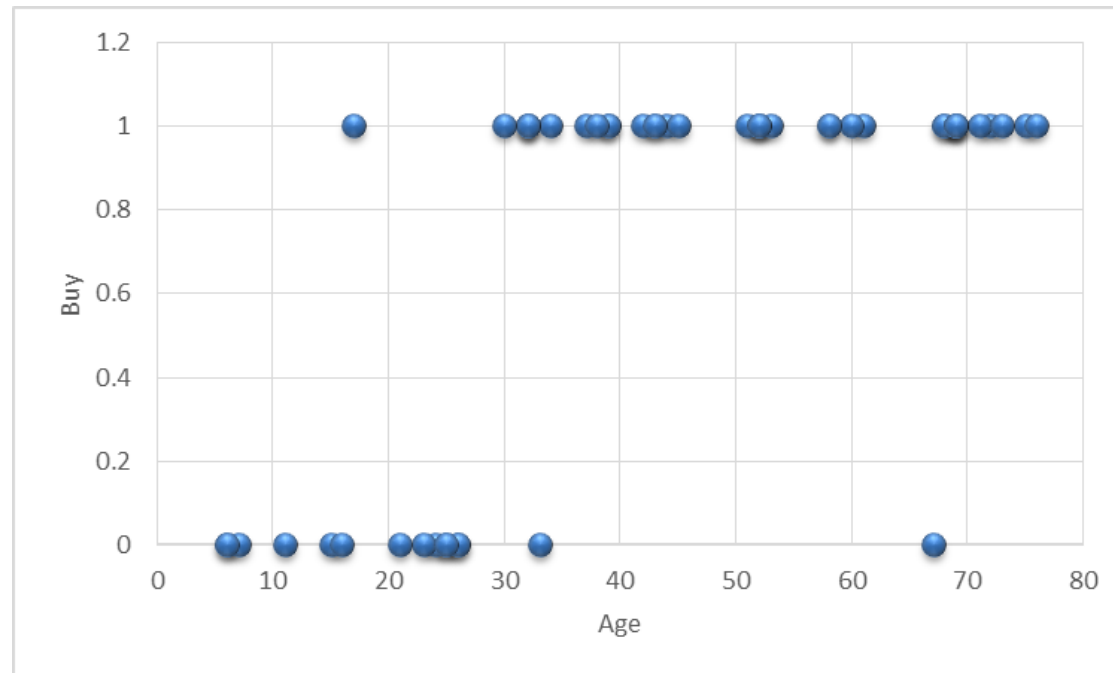
- The model that we built above is not right.
- There is certain issues with the type of dependent variable
- The dependent variable is not continuous it is binary
- We can't fit a linear regression line to this data



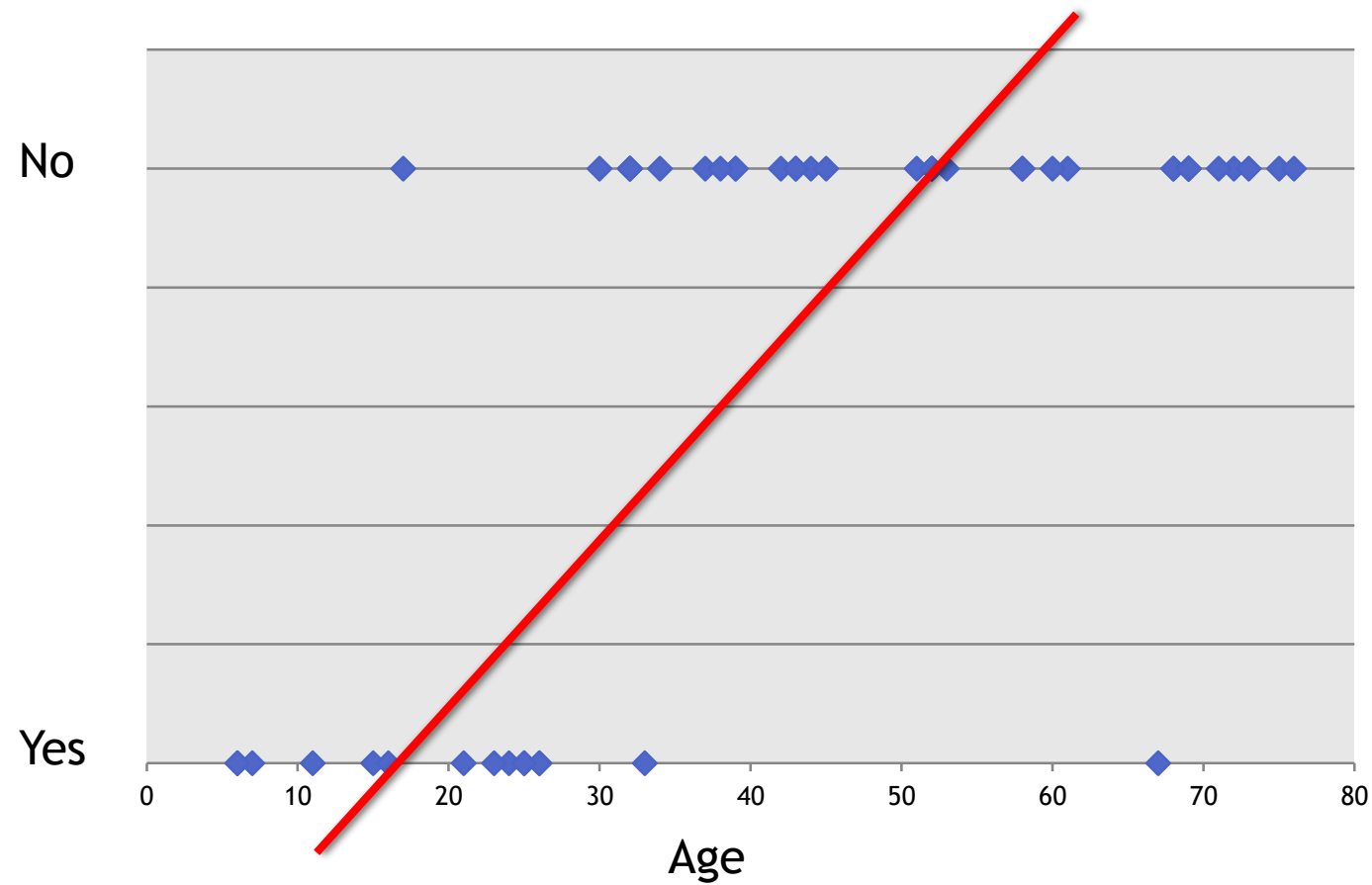
Why not linear ?

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 ?



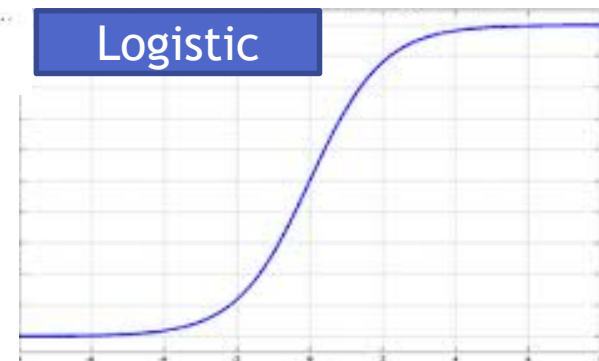
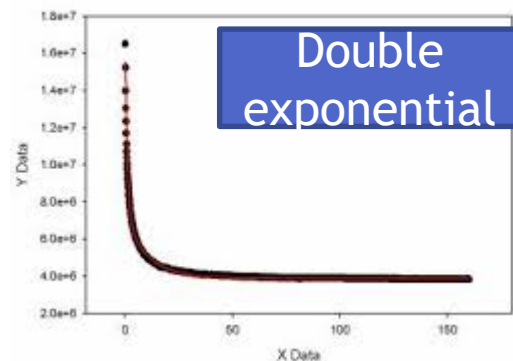
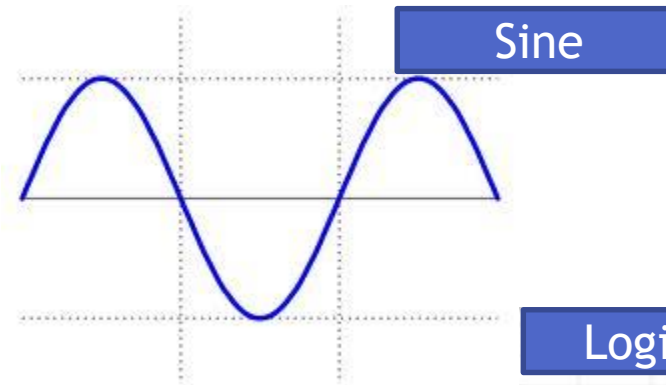
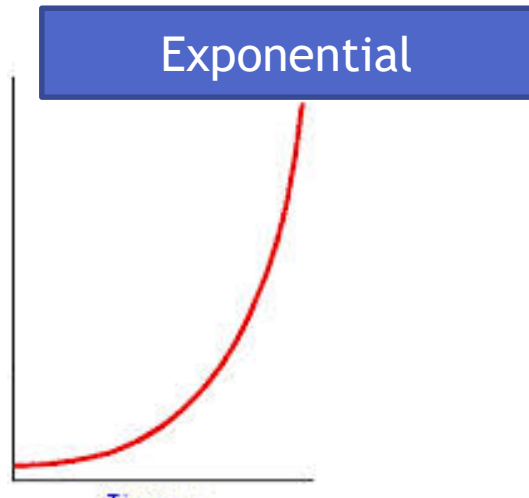
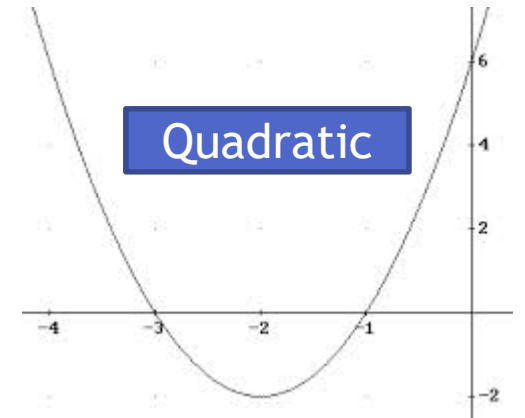
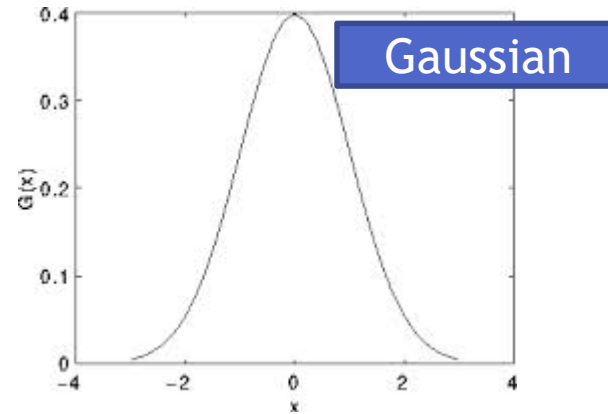
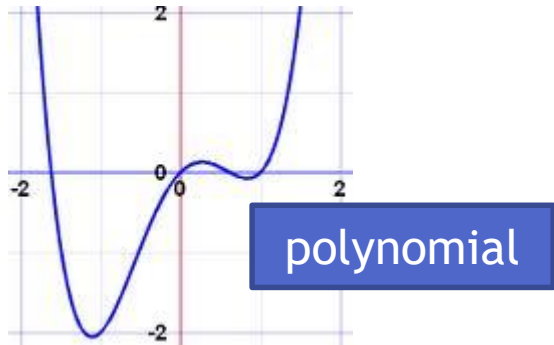
Real-life examples

- Gaming - Win vs. Loss
- 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

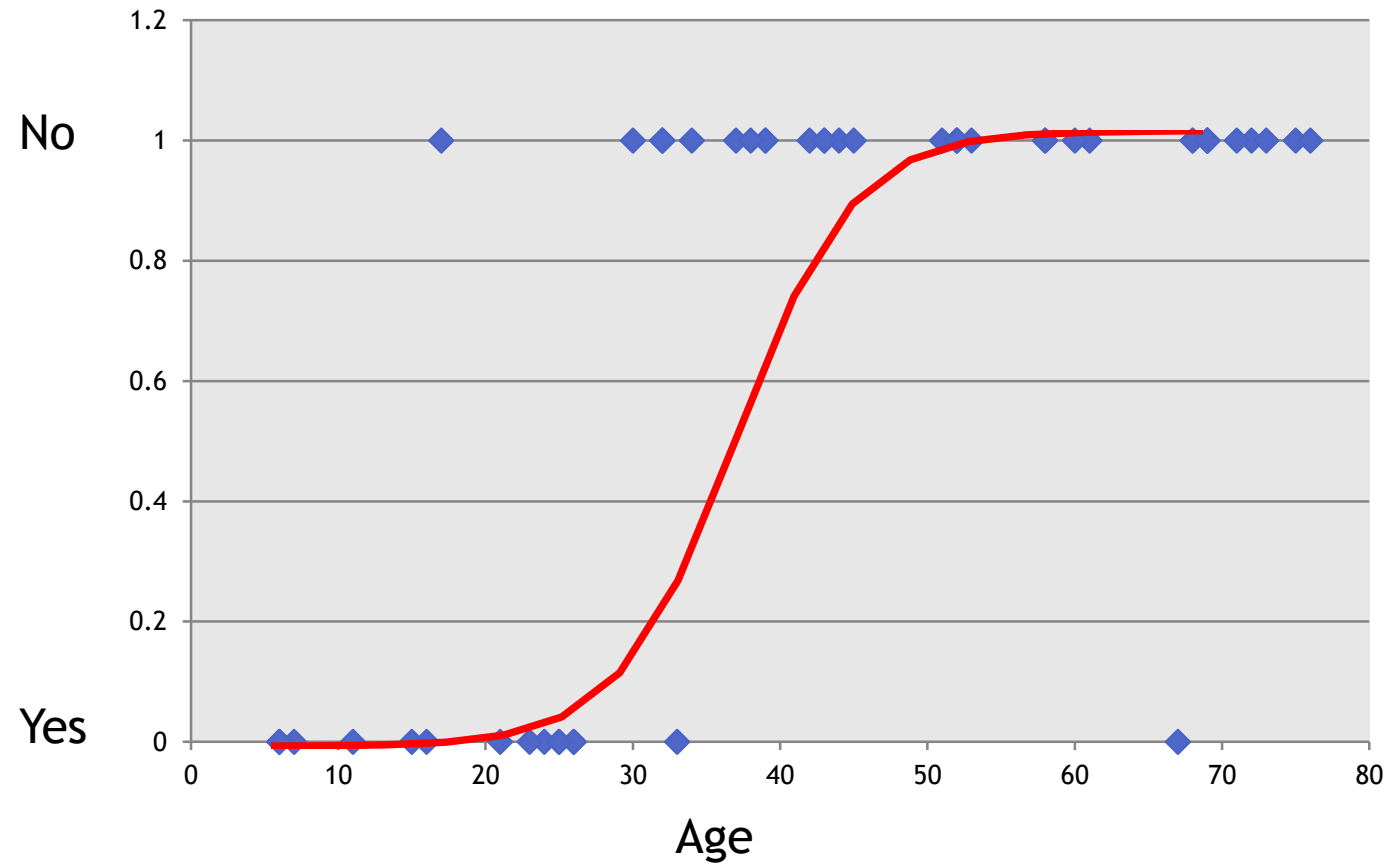


A Logistic Function

Some Nonlinear functions

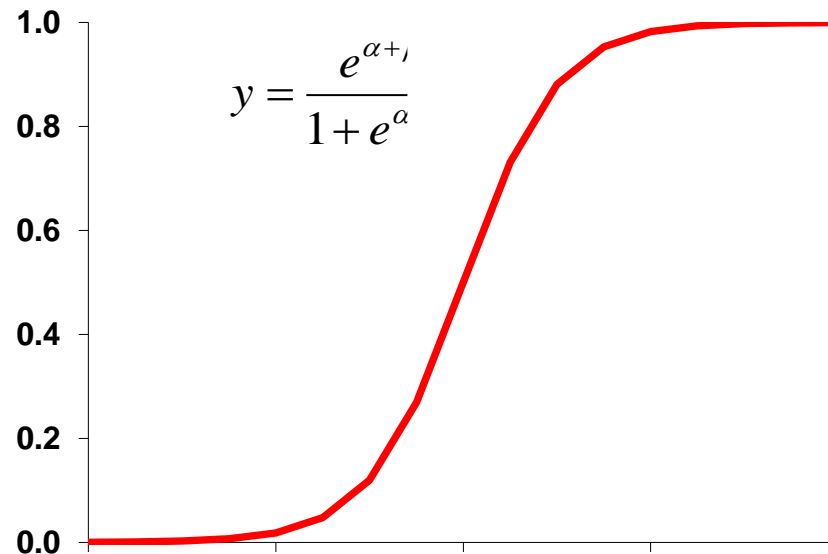


A Logistic Function



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:
- $y = \exp(\beta_0 + \beta_1 X) / [1 + \exp(\beta_0 + \beta_1 X)]$



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

- Predicting loss percentage
- Predicting Buying vs. Not buying
- Predicting number of customers
- Predicting Response vs. No Response
- Predicting revenue
- Predicting the product price
- Predicting Attrition vs. Retention
- Predicting Click vs. No click
- Predicting-Fraud vs. Non Fraud
- Predicting the amount of fraud



Multiple Logistic Regression

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']+['numbe
r_plan_changes']+['relocated']+['monthly_bill']+['technical_issues_per_month']+
['Speed_test_result']],Fiber[['active_cust']])
```




Goodness of fit for a logistic regression

Goodness of fit for a logistic regression

- Classification Matrix
- AIC and BIC
- ROC & AUC

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
- $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FN} + \text{FP} + \text{TN})$

Confusion Matrix

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

3	2
1	3

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
```



Multicollinearity

Multicollinearity

- The relation between X and Y is non linear, we used logistic regression
- The multicollinearity is an issue related to predictor variables.
- Multicollinearity need to be fixed in logistic regression as well.
- Otherwise the individual coefficients of the predictors will be effected by the interdependency
- The process of identification is same as linear regression

LAB-Multicollinearity

- Is there any multicollinearity in fiber bits model?
- Identify and remove multicollinearity from the model

LAB-Multicollinearity

```
def vif_cal(input_data, dependent_col):  
    x_vars=input_data.drop([dependent_col], axis=1)  
    xvar_names=x_vars.columns  
    for i in range(0,xvar_names.shape[0]):  
        y=x_vars[xvar_names[i]]  
        x=x_vars[xvar_names.drop(xvar_names[i])]  
        rsq=sm.ols(formula="y~x", data=x_vars).fit().rsquared  
        vif=round(1/(1-rsq),2)  
        print (xvar_names[i], " VIF = " , vif)  
  
#Calculating VIF values using that function  
vif_cal(input_data=Fiber, dependent_col="active_cust")
```



Individual Impact of Variables

Individual Impact of Variables

- Out of these predictor variables, what are the important variables?
- If we have to choose the top 5 variables what are they?
- While selecting the model, we may want to drop few less impacting variables.
- How to rank the predictor variables in the order of their importance?
- We can simply look at the z values of the each variable. Look at their absolute values
- Or calculate the Wald chi-square, which is nearly equal to square of the z-score
- Wald Chi-Square value helps in ranking the variables

LAB: Individual Impact of Variables

- Identify top impacting and least impacting variables in fiber bits models
- Find the variable importance and order them based on their impact

Code: Individual Impact of Variables

```
import statsmodels.discrete.discrete_model as sm
```

```
m1=sm.Logit(Fiber['active_cust'],Fiber[["income"]+['months_on_network']+['Num_compl  
aints']+[ 'number_plan_changes']+['relocated']+['monthly_bill']+['technical_issues_p  
er_month']+['Speed_test_result']])
```

```
m1
```

```
m1.fit()
```

```
m1.fit().summary()
```

Steps in Building Logistic Regression Line

1. When the output is class and it is not continuous
2. Benchmark model with all the variables
3. Confusion matrix ; Accuracy \rightarrow 80%
4. VIF
 1. If $\text{vif} < 5$ then keep variables
 2. If $\text{vif} \geq 5$ then drop variables
5. P-Value
 1. If $\text{p-value} < 0.05$ then keep variable
 2. If $\text{p-value} \geq 0.05$ then drop variable
6. Wald Chi-square (z^2)
 1. Higher the value better the variable



Conclusion: Logistic Regression

Conclusion: Logistic Regression

- Logistic Regression is the base of all classification algorithms
- A good understanding on logistic regression and goodness of fit measures will really help in understanding complex machine learning algorithms like neural networks and SVMs
- One has to be careful while selecting the model, all the goodness of fit measures are calculated on training data. We may have to do cross validation to get an idea on the test error



Thank you
