Agenda

01 Overview of business problem

02 Exploratory Data Analysis(EDA)

03 Missing Data Analysis

04 Outlier Analysis

05 Chi-Square Test

5.1 Information Value Analysis

5.2 Multicolinearity Analysis

06 Train and Test Split

Agenda

Overview of business problem

1. Overview of Business Problem

Consider a bank who wants to launch its insurance product to target the customers with high probability. They are unaware of which customer would opt for this product.

In this case the bank will retrieve the data of the customer from their previous interactions and leverage machine learning algorithms like Logistic regression to predict which customer will be interested in the insurance and which will not.

2. What is Exploratory Data Analysis ?

It is the process of performing a prior investigation on the data where it is collected, analysed and presented in an understandable way.

EDA - Code Walk

Step 1: Import dataset

```
data=pd.read_csv('bank-additional-full.csv', sep=';')
```

```
data.head(20)
```

Step 2: Displaying first 20 rows of dataset with head() function

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56.0	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57.0	services	married	high.school	unknown	no	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37.0	services	married	high.school	no	yes	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40.0	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56.0	services	married	high.school	no	no	yes	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
5	45.0	services	married	basic.9y	unknown	no	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
6	59.0	admin.	married	professional.course	no	no	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
7	41.0	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
8	24.0	technician	single	professional.course	no	yes	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
9	25.0	services	single	high.school	no	yes	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
10	41.0	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1.0	999.0	0.0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

EDA - Code Walk

Step 3: Checking all the statistical data.

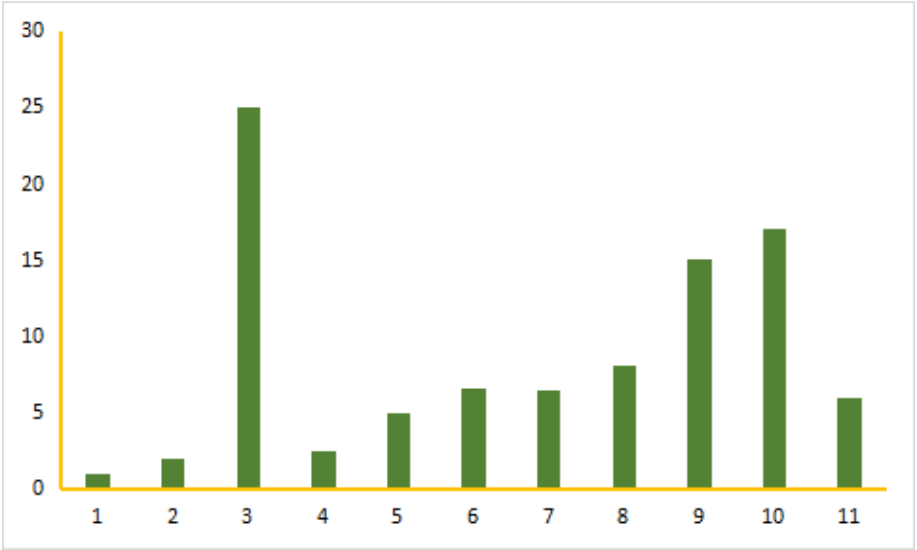
describe() helps us to check for mean, std, quartiles, minimum and maximum values in our dataset.

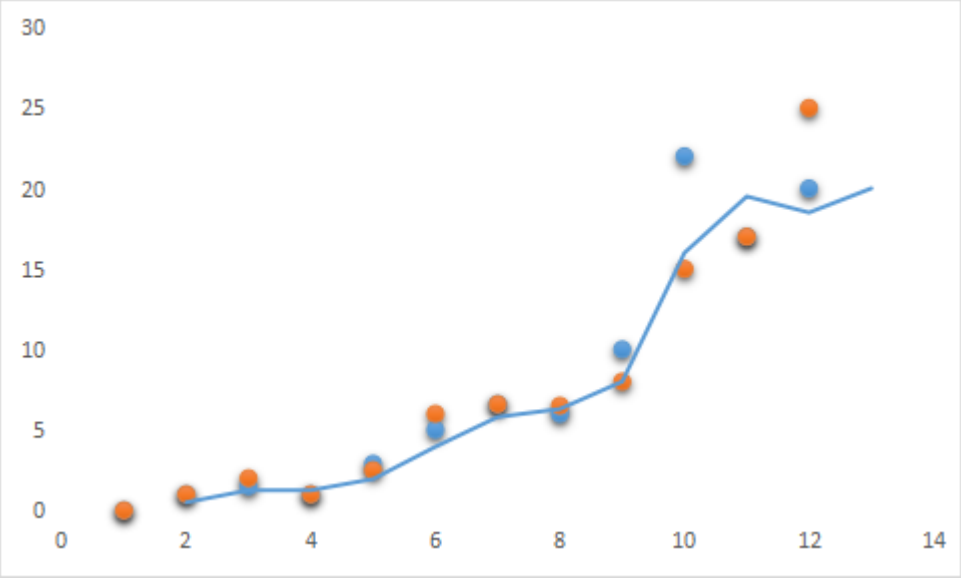
```
data.describe()
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41195.000000	41196.000000	41196.000000	41196.000000	41196.000000	41196.000000	41195.000000	41196.000000	41196.000000	41196.000000
mean	40.030319	258.281265	2.567676	962.482547	0.173124	0.081656	93.575867	-40.504600	3.620787	5166.996405
std	10.431818	259.254211	2.769751	186.893451	0.494987	1.570894	0.578999	4.629972	1.734655	72.300104
min	17.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.000000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.000000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.000000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.000000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

EDA In-depth

Visualization is the best way to present the data in a most understandable way.





Visualization can be done with different types of plots like bar, line, box, scatter plot and so on.

EDA - Code Walk

Step 4: Check for datatypes of each column

dtypes() helps us to find the datatype.

```
data.dtypes
```

```
age               float64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration          float64
campaign          float64
pdays             float64
previous          float64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
```

3. What is Missing Data Analysis?

It is the process where the missing value pattern is described and replaced with values using certain regression.

Question:

Now, what do you think we need to do with these missing values?

Missing Data Analysis - Code Traverse

```
data.isna().sum()
```

```
age               4
job               5
marital           5
education         5
default           4
housing           3
loan              4
contact           4
month             3
day_of_week       3
duration          3
campaign          3
pdays             3
previous          3
poutcome          5
emp.var.rate      3
cons.price.idx    4
cons.conf.idx     3
euribor3m         3
nr.employed       3
y                 3
dtype: int64
```

Step 5: check for missing values in dataset.

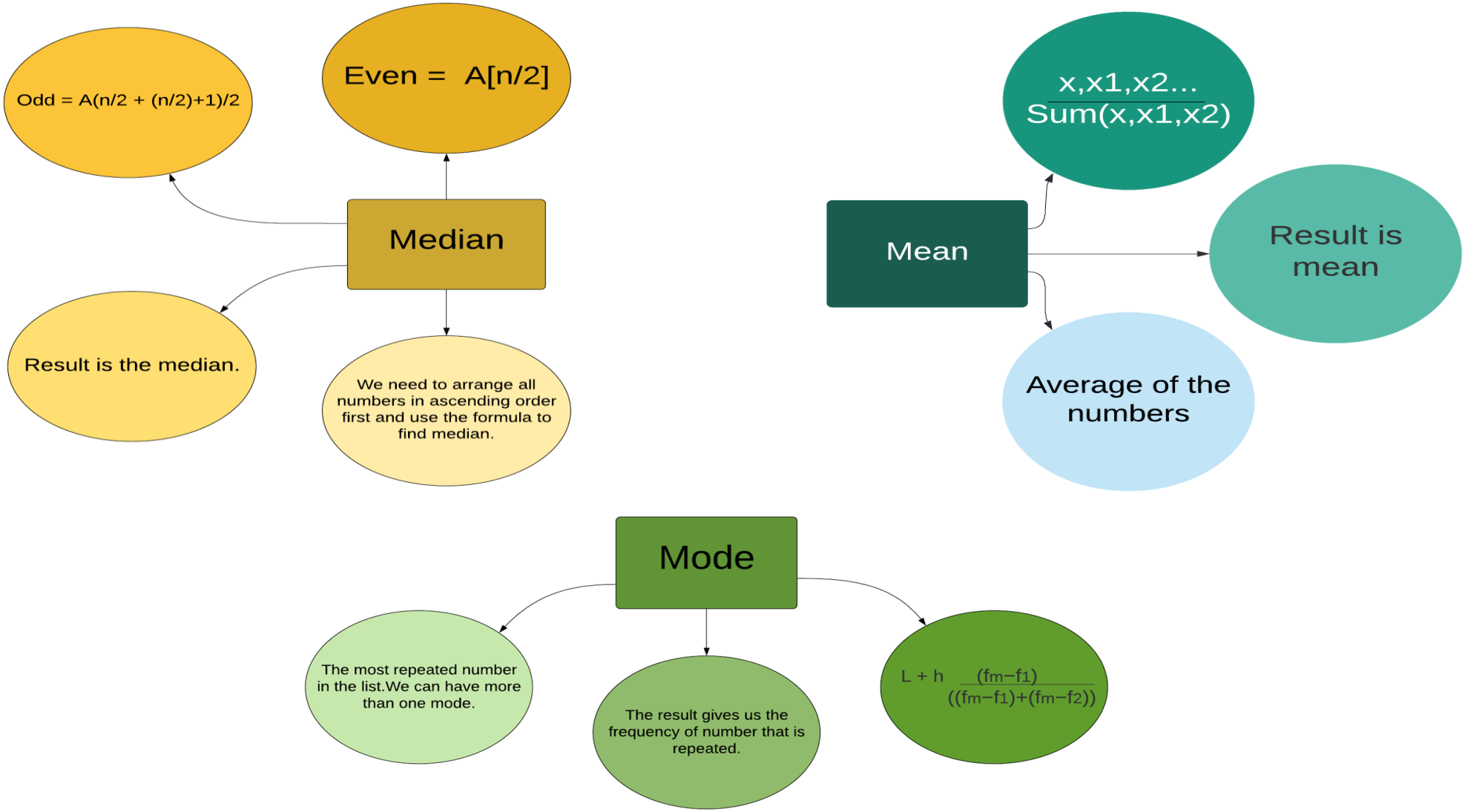
isna() helps us to find out the missing values in dataset.

Missing Data Imputation

Missing values for quantitative or numeric data is replaced by mean or median of the column.

Missing values for qualitative data is replaced by mode of the data.

Cheatsheet for Missing Data Imputation



Missing Data Analysis - Code Traverse

Step 6: Replacing missing data.

If it is a categorical or object type data replace it with mode.

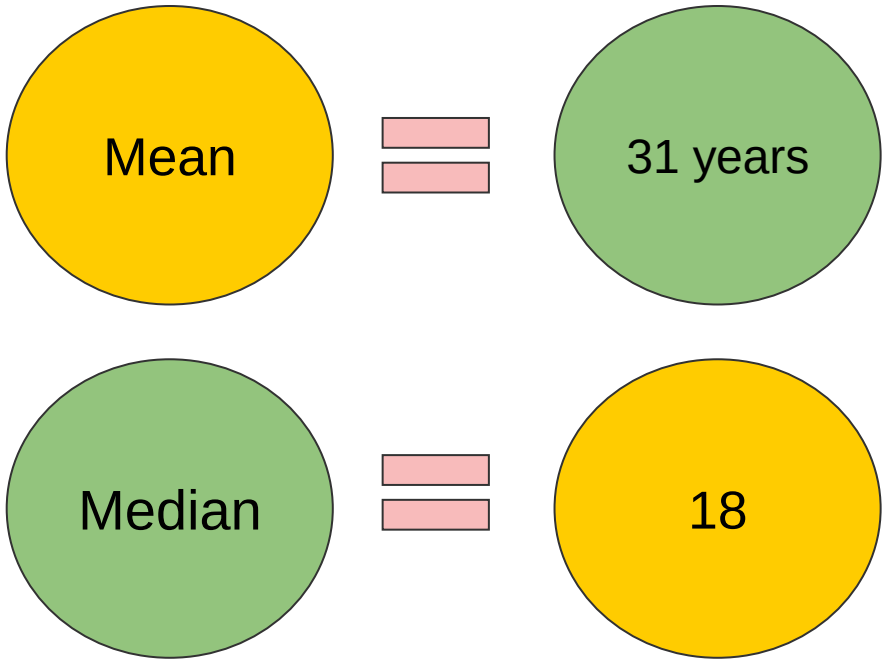
```
for col in col_list:
    if data[col].dtypes=='object':
        #print('ob')
        data[col] = data[col].fillna(data[col].mode()[0])
    else:
        #print('num')
        data[col] = data[col].fillna(data[col].mean())
```

Else if it is a numerical data replace it with mean.

Mean Vs Median - Quick Look

Let us consider an example, consider below given ages where we have to predict the missing age with mean and median.

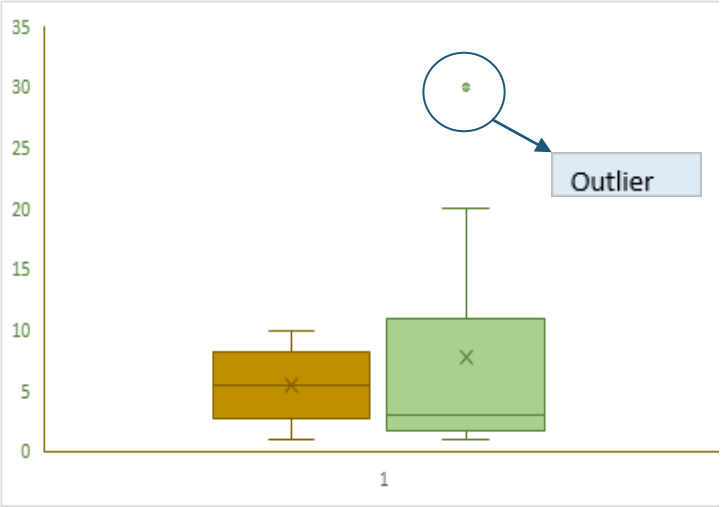
Sl Num	Age (years)
1	18
2	20
3	15
4	?
5	70

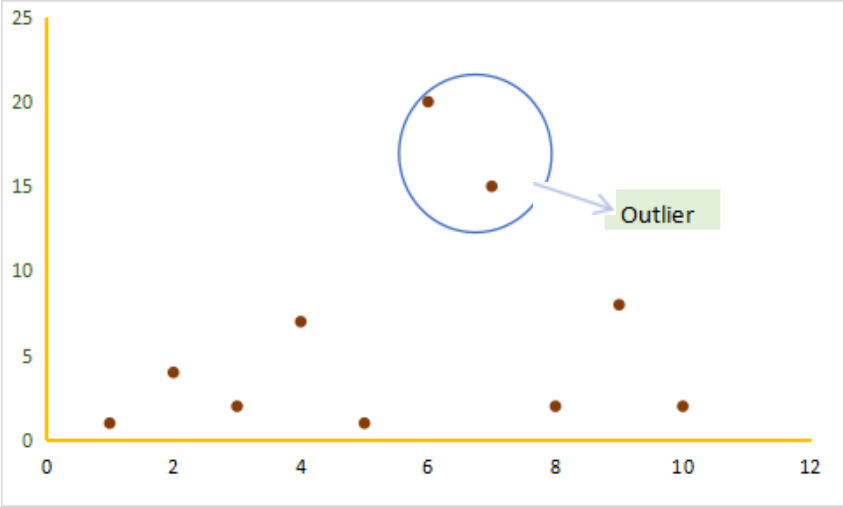


If we see the above example closely we can see that median gives us a more realistic value. Due to this nature of median we prefer median over mean.

4. What is Outlier Analysis?

The process of discovering the abnormal observation in a dataset is referred to as outlier analysis.





Question:

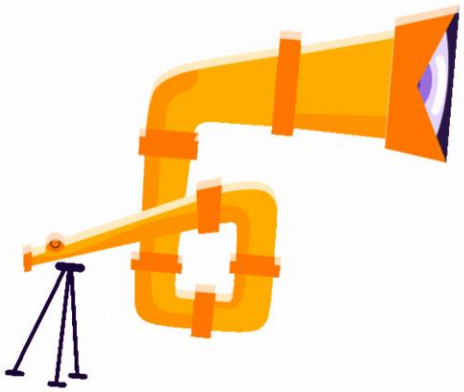
Now how do you think we can handle the outliers?

Outlier Analysis - Description

Well, we use a technique called capping.

Values lower than the first percentile value are substituted with the first percentile value. This is called as lower capping.

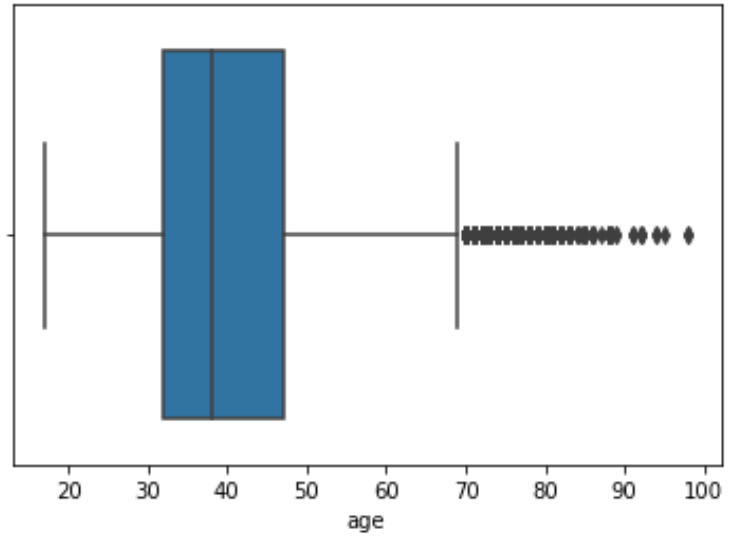
Values larger than the ninety ninth percentile value are substituted with the ninety ninth percentile value. This is called upper capping.



Outlier Analysis - Code View

```
import seaborn as sns
sns.boxplot(x=data['age'])
```

```
<AxesSubplot:xlabel='age'>
```



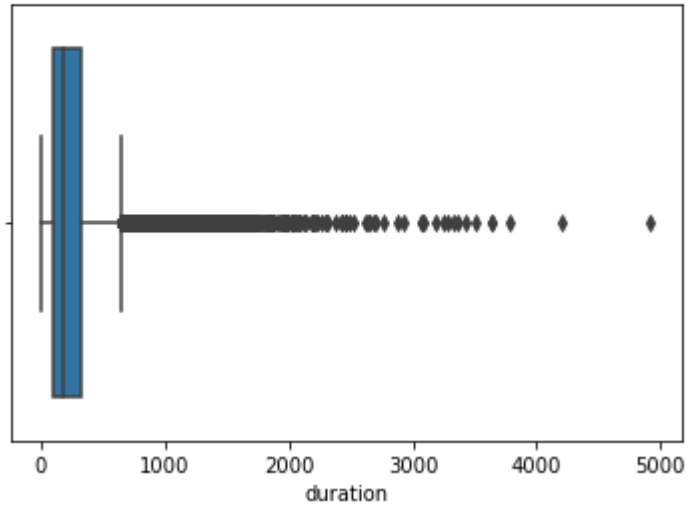
Here we can see the outlier values for 'age' column.

Step 7: Checking for outlier values.

Here we can see the outlier values for 'duration' column.

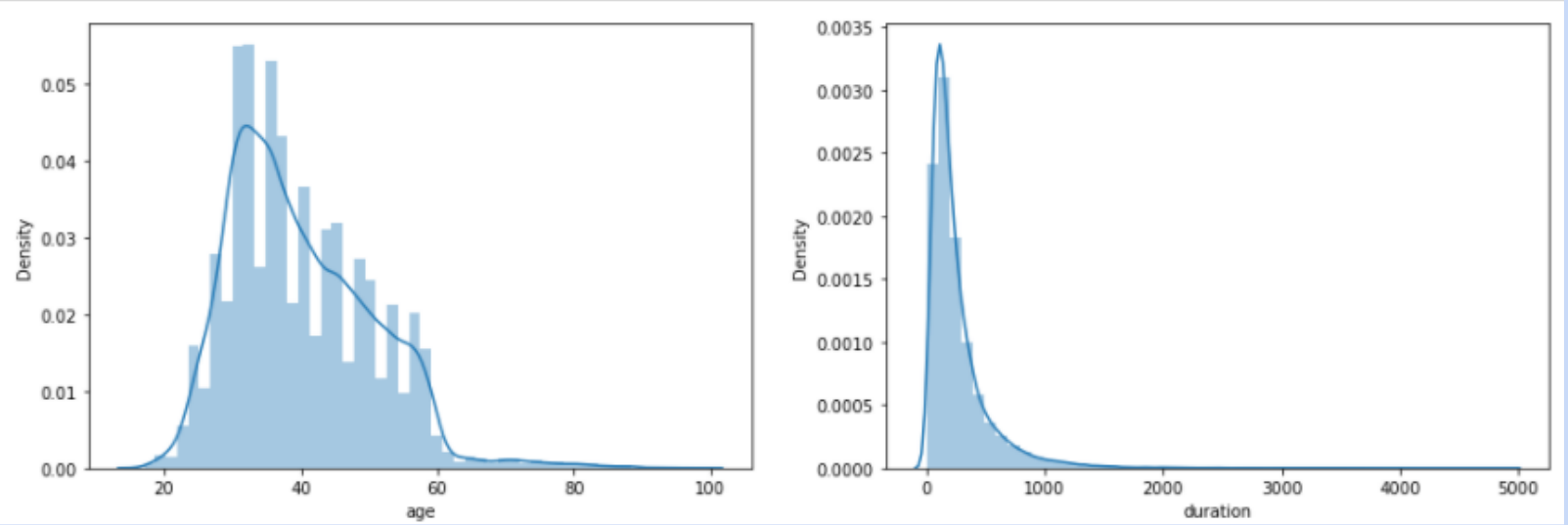
```
sns.boxplot(x=data['duration'])
```

```
<AxesSubplot:xlabel='duration'>
```



Outlier Analysis - Code View

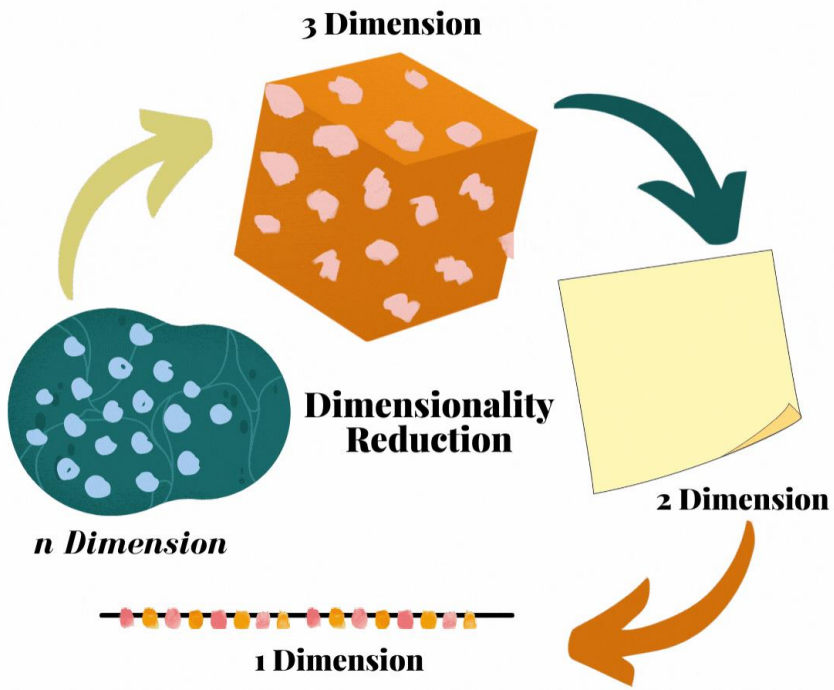
```
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(data['age'])
plt.subplot(1,2,2)
sns.distplot(data['duration'])
plt.show()
```



The graph shows the outliers for age and duration.

5.1. What is Chi Square Test?

A chi-square test is used to see if observed findings match predicted outcomes and to eliminate the possibility that observations are random.



It is used to demonstrate if two category variables have a link with each other.

Chi Square Test - Study

A p-value less than or equal to your significance threshold 0.05 in a Chi-square test shows that there is enough evidence to determine that the observed distribution is not the same as the expected distribution.

This means that the independent categorical variable is having a strong correlation with dependent categorical variable.

You can deduce that the categorical variables have a relationship.

Col Values	P-Value	Col Values	P-Value
Age	0.0	Contact	0.00
Job	0.0	Month	0.0
Marital	0.0	Poutcome	0.0
Education	0.0	Housing	0.017991
Default	0.0	Loan	0.299648
		day_of_week	3.12227e-09

Chi Square Test - Computing

Step 8: Calculating the p-value in order to decide whether null values needs to be eliminated or not.

1.Chi-Square test

```
Chi-Square test is normally done on Categorical data(object) ,where we will select each independent attribute and calculating
chi-Square statistics value with the help of  observed value and expected value.

if observed and expected values are close then we'll have high chi-square statistics.(high chi-square stats implies attributes
are highly independent)
if observed and expected values are far then we'll have less chi-square statistics.(less chi-square stats implies attributes are
not independent)

With the above concept we'll set null and alternative hypothesis and check for its acceptance or rejection through p-value and
alpha value (significance value)

if p -value > alpha value ,we reject null hypothesis
if p-value  < alpha value ,we accept null hypothesis
```

Chi Square Test - Computing

In [93]:
```
###Chisq Test for Independence for all object fields
col_list = list(data.columns)
col_list.remove('y')
df=pd.DataFrame(columns=['Feature','P-value'])

for col in col_list:
    if data[[col]][col].dtype == 'object':
        dataset_table=pd.crosstab(data[col],data['y'])
        Observed_Values = dataset_table.values
        val=chi2_contingency(dataset_table)
        Expected_Values=val[3]
        chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
        chi_square_statistic=chi_square[0]+chi_square[1]
        no_of_rows=Observed_Values.shape[0]
        no_of_columns=Observed_Values.shape[1]
        ddof=(no_of_rows-1)*(no_of_columns-1)
        alpha = 0.05
        critical_value=scipy.stats.chi2.ppf(q=1-alpha,df=ddof)
        p_value=1-chi2.cdf(x=chi_square_statistic,df=ddof)
        df=df.append({'Feature':col, 'P-value': p_value}, ignore_index=True)

df
```

	Feature	P-value
0	age	0.00
1	job	0.00
2	marital	0.00
3	education	0.00
4	default	0.00
5	housing	0.06
6	loan	0.58
7	contact	0.00
8	month	0.00
9	day_of_week	0.00
10	poutcome	0.00

```
#so from above output we can see ,loan and housing attribute have p-value greater than alpha value so we'll be removing the attributes
```

5.2. What is Information Value?

When it comes to picking predictor variables for binary classification problem, the Information Value is a prominent strategy.

Variable transformation can also be performed with this technique!

Information Value Cutoffs

Col Values	IV-Value	Col Values	IV-Value
Poutcome	0.55	Age	0.07
Month	0.49	Education	0.05
Contact	0.25	Marital	0.03
Job	0.19	Housing	0.0
Default	0.13		

Information Value	Predictive Power
<0.02	Useless
0.02 – 0.1	Weak predictors
0.1 - 0.3	Medium Predictors
0.3 - 0.5	Strong predictors
>0.5	Suspicious

If IV <= 0.1 we can drop the variable.

Information Value - Code Snippet

```
# IV analysis

IV analysis is done on features to check the predictive power of features(independent attributes) by binning the values and
calculating the weight of eveidence and finaly computing IV-Score which gives you the predictive power based on predefined
range of IV-Score. below are the range of data

#Information Value          Predictive power

#<0.02                      Useless

#0.02 to 0.1                Weak predictors

#0.1 to 0.3                 Medium Predictors

#0.3 to 0.5                 Strong predictors

#>0.5                       Suspicious

Note:before calculating the weight of evidence make sure to bin the features if they are not categorical.and make sure each bin
should have atleast 5% of data distributed in each bin
```

Information Value - Code Snippet

```
def calculate_woe_iv(dataset, feature, target):
    lst = []
    for i in range(dataset[feature].nunique()):
        val = list(dataset[feature].unique())[i]
        lst.append({
            'Value': val,
            'All': dataset[dataset[feature] == val].count()[feature],
            'Good': dataset[(dataset[feature] == val) & (dataset[target] == 1)].count()[feature],
            'Bad': dataset[(dataset[feature] == val) & (dataset[target] == 0)].count()[feature]
        })

    dset = pd.DataFrame(lst)
    dset['Distr_Good'] = dset['Good'] / dset['Good'].sum()
    dset['Distr_Bad'] = dset['Bad'] / dset['Bad'].sum()
    dset['WoE'] = np.log(dset['Distr_Good'] / dset['Distr_Bad'])
    dset = dset.replace({'WoE': {np.inf: 0, -np.inf: 0}})
    dset['IV'] = (dset['Distr_Good'] - dset['Distr_Bad']) * dset['WoE']
    iv = dset['IV'].sum()

    dset = dset.sort_values(by='WoE')

    return dset, iv
```

Information Value - Code Snippet

```
col_list = list(data.columns)
for col in col_list:
    if col == 'y':
        continue
    elif data[col].dtype == 'object':
        df, iv = calculate_woe_iv(data, col, 'y')
        df_new=df_new.append({'Feature':col, 'IV-Score': iv}, ignore_index=True)
df_new
```

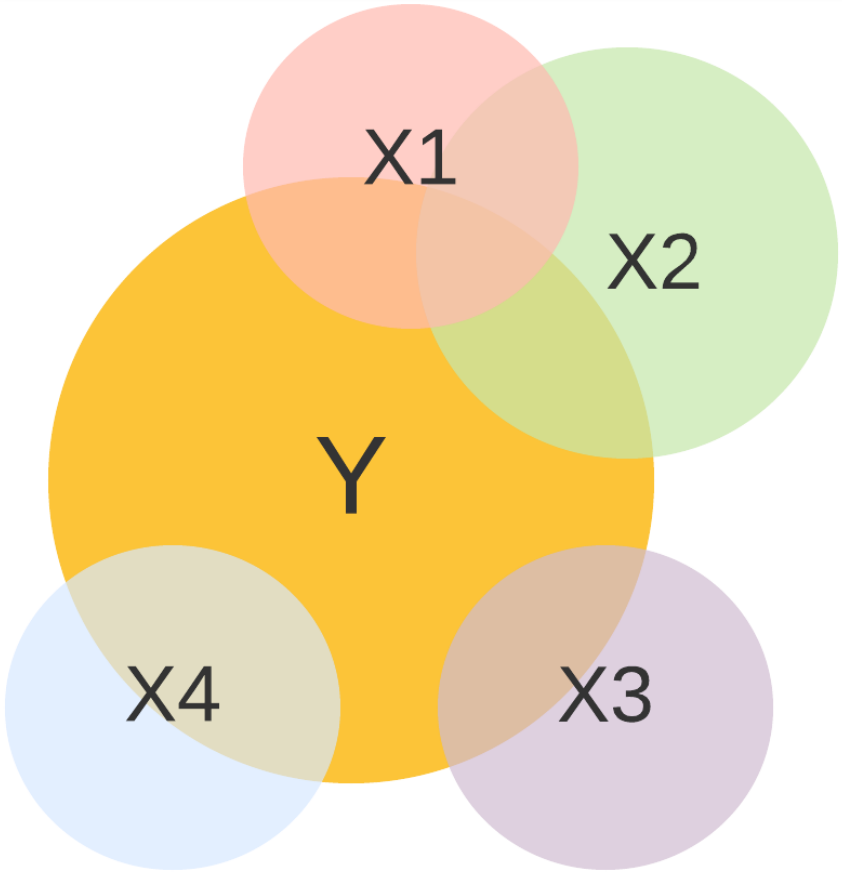
```
C:\Users\Admin\anaconda3\new ann\New folder (2)\lib\site-packages\pandas\core\arraylike.p
encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

	Feature	IV-Score
0	age	0.03
1	job	0.19
2	marital	0.03
3	education	0.05
4	default	0.13
5	contact	0.25
6	month	0.49
7	day_of_week	0.01
8	poutcome	0.55

```
You can observe that these features (age, marital,education,day_of_week) have less predictive power ,henceforth we will be
dropping those columns
```

6. What is Multicollinearity Analysis ?

This is the process of occurrence of multiple intercorrelation among one or more independent variables

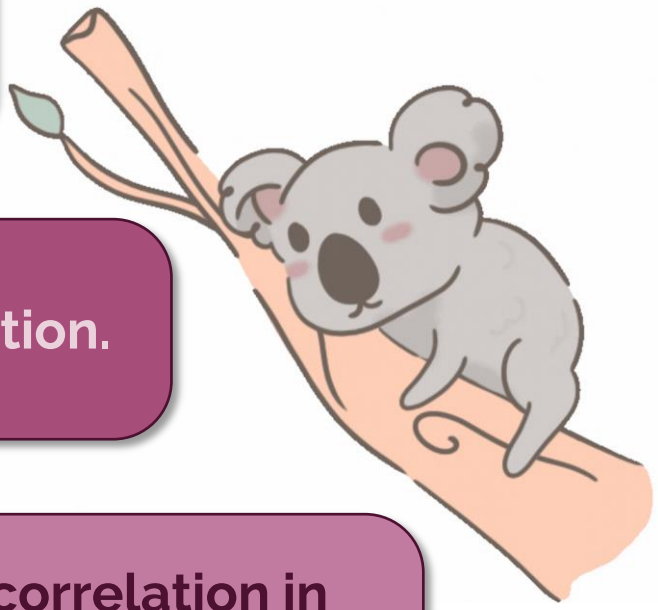


Question

What do you think the diagram is trying to tell us?

Multicollinearity Analysis - Sampling

So, in the diagram we can see that Y has a correlation with X1, X2, X3 and X4.



We can also see that X1 and X2 also have a high correlation.

Think of overlap between these bubbles as correlation in variables.

This process helps to avoid overlap between these independent variables.

Multicollinearity Analysis - Instance

Let us consider an example where we show the correlation of 'Y' with 'X1', 'X2' and 'X3'.

Y	-	X1
Y	-	X2
Y	-	X3

Now, if we can see here Y has correlation with X1, X2 and X3.

Higher correlation between Y and X1 , X2 = Good Model.

Question

But, what if X1 ~ X2?

Why Collinearity Analysis

In that case we will have duplicate data.

Now, how do we check for that duplicate data?

In order to see that let us consider an example.

	X1	X2	X3	X4
X1	1	0.2	0.8	0.7
X2	0.3	1	0.3	0.1
X3	0.2	0.8	1	0.2
X4	0.5	0.1	0.9	1

If we look into the table X1 has the highest correlation with X3 and X4 which means values of X3 and X4 are similar to X1.

Now the question is how to know which to eliminate between X3 and X4.

Why Collinearity Analysis - Continuation

Simply putting we first eliminate the highest correlated variable and then eliminate the next highest correlated variable.

The variables with high correlation can be removed by using chi square test.

Multicollinearity Analysis - Pop Quiz

The most important question is why do we need multicollinearity if we have collinearity?

Why Multicollinearity Analysis - Instance

Consider the below example,

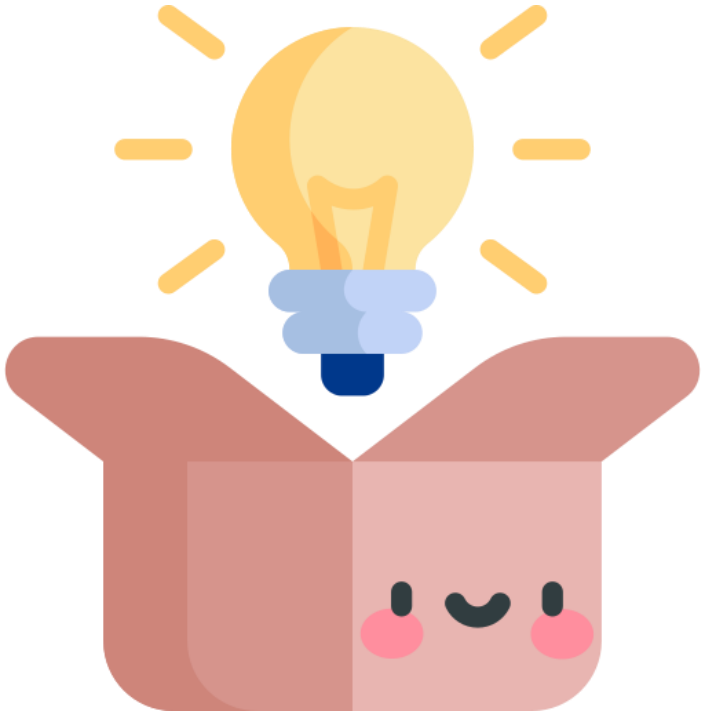
0.1 — If it rains, a person will go shopping

0.2 — If there is an earthquake(mild) a person may still go shopping.

0.9 — If there is rain and an earthquake at the same time then a person wouldn't go shopping due to fear as this not expected and new to the locals. Hence, the probability will be low.

Multicollinearity Analysis - Continuation

From the above example we can say that hidden correlations can occur.



As collinearity analysis cannot find the hidden correlations, we use multicollinearity analysis.

Evaluating Multicollinearity Analysis

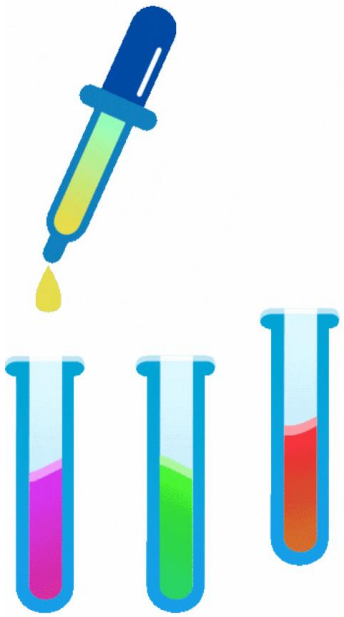
X1	~ (R2)	X2	X3	X4	(0.9)
X2	~ (R2)	X1	X2	X4	(0.1)
X3	~ (R2)	X1	X2	X4	(0.8)
X4	~ (R2)	X1	X2	X3	(0.2)

Higher the R value higher is the correlation.

Firstly, eliminate X1 value and run the analysis again

Evaluating Multicollinearity Analysis

We prefer VIF over R2.



This is because VIF helps us to differentiate between R2 values better.

Formula:

$$\frac{1}{1-r2}$$

Multicollinearity Analysis Evaluation

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
col_list = []
for col in df_combined.columns:
    if ((df_combined[col].dtype != 'object') & (col != 'y') ):
        col_list.append(col)

X = df_combined[col_list]
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]
print(vif_data)
```

	feature	VIF
0	duration	2.023846
1	campaign	1.945188
2	pdays	341.999288
3	previous	6.661406
4	emp.var.rate	93.373468
5	cons.price.idx	63666.920811
6	cons.conf.idx	388.198365
7	euribor3m	784.210485
8	nr.employed	80758.405503
9	job_blue-collar	2.021589
10	job_entrepreneur	1.152564
11	job_housemaid	1.108832
12	job_management	1.291418
13	job_retired	1.204522
14	job_self-employed	1.139838
15	job_services	1.409385
16	job_student	1.118825
17	job_technician	1.659036
18	job_unemployed	1.103726
19	job_unknown	1.039577
20	default_unknown	1.392964
21	default_yes	1.000871
22	contact_telephone	5.173560
23	month_aug	8.114738
24	month_dec	1.140407
25	month_jul	5.393212
26	month_jun	3.763630
27	month_mar	1.251443
28	month_may	7.499817
29	month_nov	3.936757
30	month_oct	1.598761
31	month_sep	1.517159
32	poutcome_nonexistent	37.461051
33	poutcome_success	11.510954

Step 9: Checking for VIF values

Here we can see that the highlighted value has the highest VIF Value so we need to eliminate the value.

We only drop the highest VIF value not all VIF values.

It is good practice to remove values one by one.

Evaluating Multicollinearity Analysis

X2	~ R2	X3	X4	(0.1)
X3	~ R2	X2	X4	(0.1)
X4	~ R2	X2	X3	(0.1)

We either use R2 value or VIF value to detect multicollinearity.

VIF is similar to R2 but with this we can differentiate between correlations in a better way.

If VIF>0.5 we eliminate the variable.

Multicollinearity Analysis - Code Walk

Step 9.1: Eliminating columns with high VIF value.

```
df_combined=df_combined.drop(['nr.employed'], axis = 1)
```

The previously highlighted value is eliminated but likewise we have many other values with high VIF values. Hence we need to eliminate them all.

```
col_list = []
for col in df_combined.columns:
    if ((df_combined[col].dtype != 'object') & (col != 'y') ):
        col_list.append(col)

X = df_combined[col_list]
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]
print(vif_data)
```

	feature	VIF
0	duration	2.022773
1	campaign	1.939034
2	pdays	341.277640
3	previous	6.534721
4	emp.var.rate	36.155853
5	cons.price.idx	1084.610064
6	cons.conf.idx	228.103548
7	euribor3m	216.546435
8	job_blue-collar	2.018708
9	job_entrepreneur	1.152380
10	job_housemaid	1.108832
11	job_management	1.291390
12	job_retired	1.203720
13	job_self-employed	1.139833
14	job_services	1.409011
15	job_student	1.116980
16	job_technician	1.658778
17	job_unemployed	1.103721
18	job_unknown	1.039255
19	default_unknown	1.392245
20	default_yes	1.000871
21	contact_telephone	4.874594
22	month_aug	6.273920
23	month_dec	1.140370
24	month_jul	5.373671
25	month_jun	3.747964
26	month_mar	1.247694
27	month_may	7.373027
28	month_nov	3.686317
29	month_oct	1.508358
30	month_sep	1.510405
31	poutcome_nonexistent	36.920687
32	poutcome_success	11.509035

Multicollinearity Analysis - Conclusion

As we have seen above by evaluating the hidden correlation of X1 with other variables helped eliminate correlation of X3 with other variables.

This simply means that sometimes eliminating hidden correlations can result in a stable correlations among other variables.

Multicollinearity Analysis - Result

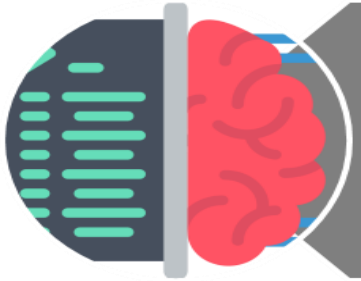
```
col_list = []
for col in df_combined.columns:
    if ((df_combined[col].dtype != 'object') & (col != 'y') ):
        col_list.append(col)

X = df_combined[col_list]
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]
print(vif_data)
```

	feature	VIF
0	duration	1.896700
1	campaign	1.859587
2	previous	1.769046
3	emp.var.rate	1.678231
4	job_blue-collar	1.852786
5	job_entrepreneur	1.122194
6	job_housemaid	1.091261
7	job_management	1.237912
8	job_retired	1.168198
9	job_self-employed	1.114562
10	job_services	1.331760
11	job_student	1.098208
12	job_technician	1.541098
13	job_unemployed	1.088323
14	job_unknown	1.034300
15	default_unknown	1.381653
16	default_yes	1.000807
17	month_aug	1.653780
18	month_dec	1.040511
19	month_jul	2.000485
20	month_jun	1.643107
21	month_mar	1.067991
22	month_may	2.422329
23	month_nov	1.374161
24	month_oct	1.120931
25	month_sep	1.105058
26	poutcome_success	1.465064

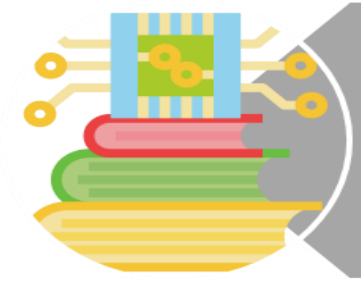
All high VIF values were eliminated with multiple iterations.

The final result has all VIF lower than cutoff of 0.5.

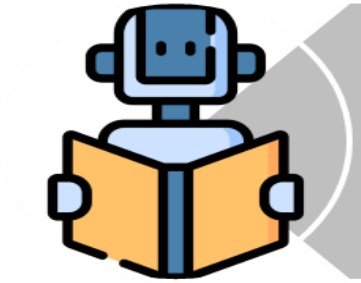
7. What is Train and Test of Data ?



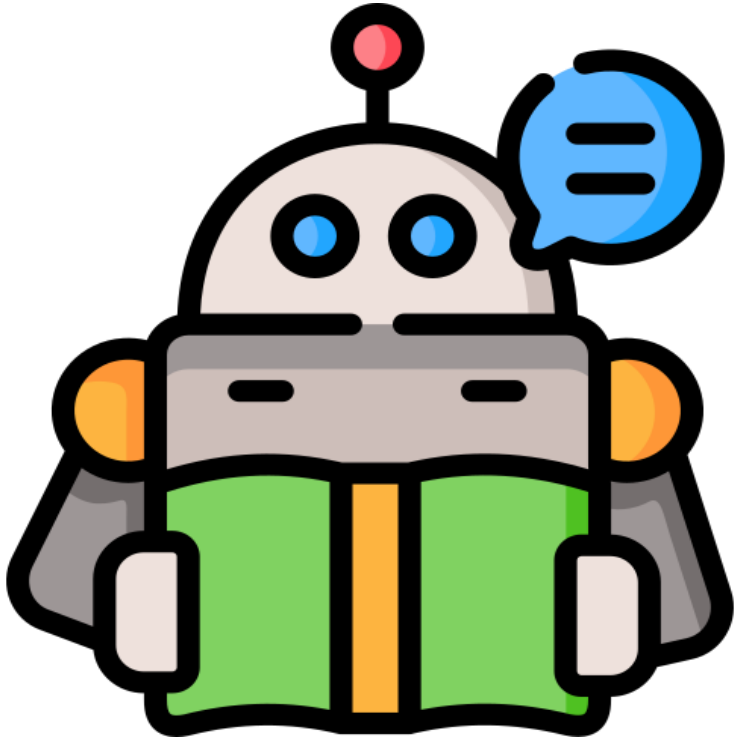
It is the process of splitting data into train and test sections so we can train the model with the train data and test if our model is working fine with test data.



Training and testing of model helps us to gauge the performance of our model.



We always divide in a way where train will have more data and test will have comparatively less data.



Train and Test of Data - Computing

Step 10: Splitting the data into train and test set.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df_ind, df_dep, test_size=0.25, random_state=0)
```

```
x_train.shape
```

```
(30899, 27)
```

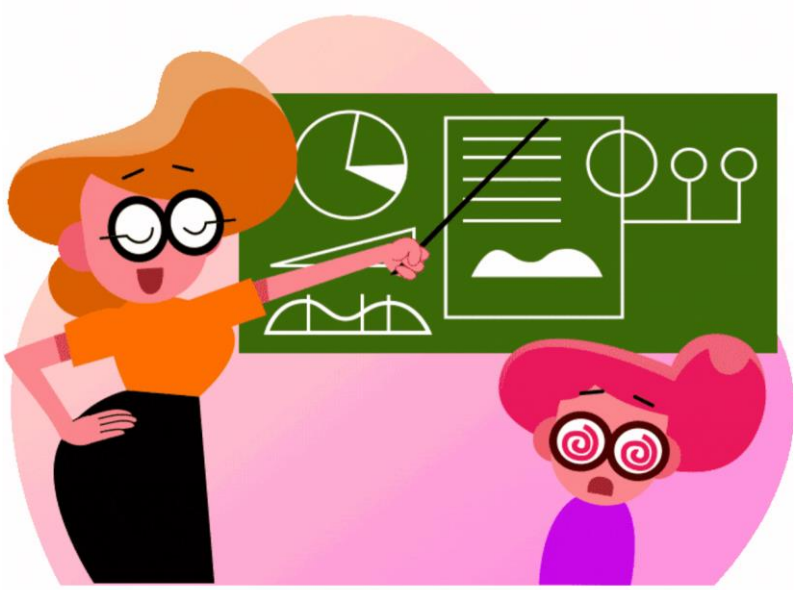
```
x_test.shape
```

```
(10300, 27)
```

8. What is Model Training ?

The process of training the model on the required data is called model training.

We also refer to it as model fitting.



Model Training - Code Snippet

```
from sklearn.linear_model import LogisticRegression

logisticRegr = LogisticRegression()

#####Model Fitting/Training###
logisticRegr.fit(x_train, y_train)

LogisticRegression()
```

Step 11: Training our model.

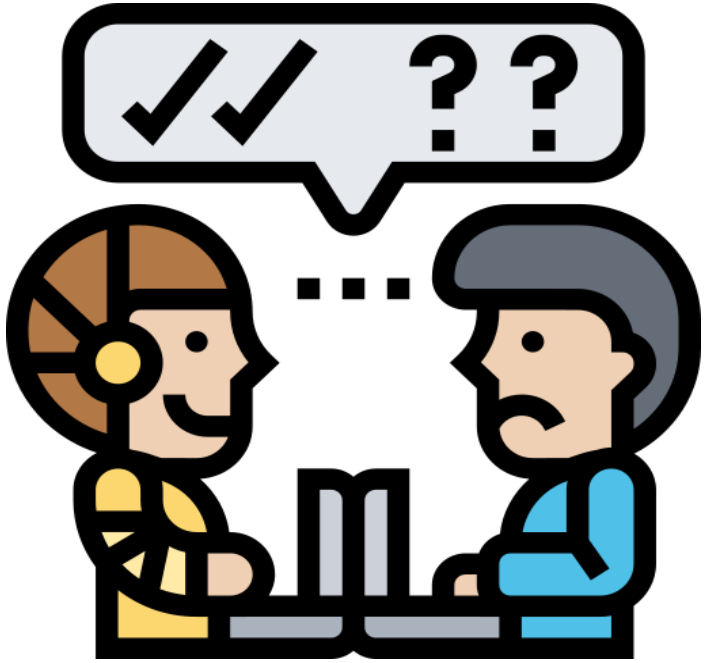
9. What is Model Validation ?

The process of testing the model with the test data after training the model is called model validation.

This gives us the insights on the model performance.

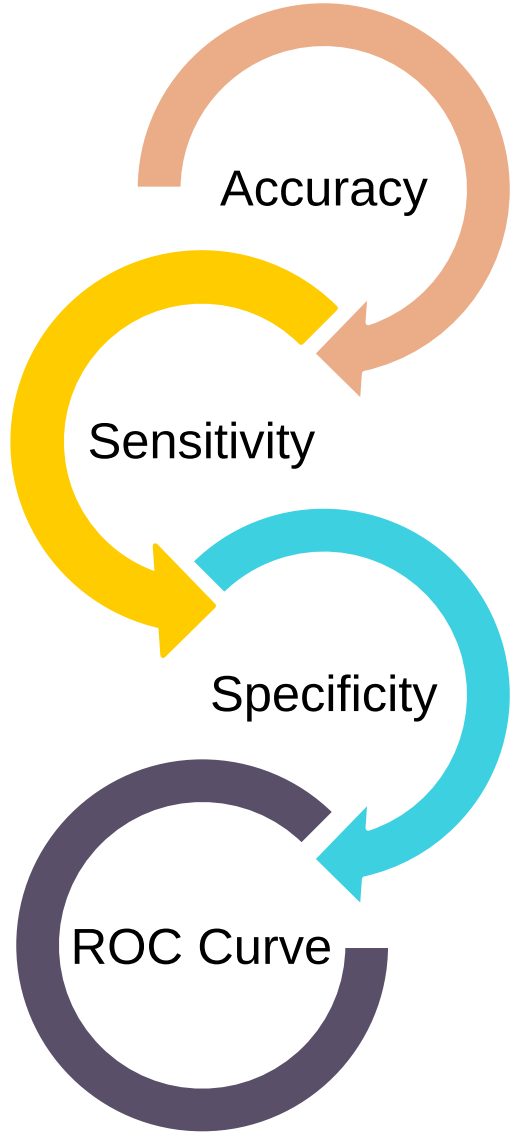
This gives us the insites on the model performance.

This is also called model testing.



What is Model Validation?

Shown are the steps for model validation.



Model Validation

Step 12: Scoring our model.

```
test_pred = logisticRegr.predict(x_test)
```

```
np.unique(test_pred)
```

```
array([0, 1])
```

```
x_test.shape
```

```
(10300, 27)
```

```
test_pred.shape
```

```
(10300,)
```

```
test_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

9.1 What is Accuracy?

The process of checking how accurate the model is during model validation is called as accuracy.

Formula:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

High precision and is required to get high accuracy of the model.

Calculating Accuracy

```
Accuracy =  (8863 +417)/(8863  +249+771  +417)
Accuracy*100
```

```
90.09708737864077
```

Step 13: Finding the accuracy of the model.

9.2 What is Sensitivity?

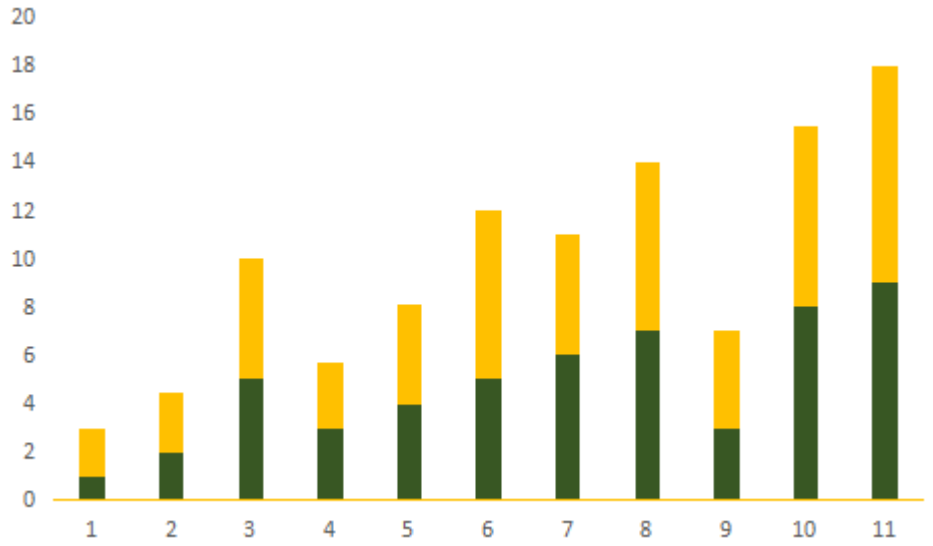
The process of determining how the output varies in response to changes in the input is known as sensitivity.

It could provide insight into the predictability of a model.

It is used to check the model's ability to predict true positive data.

Formula:

$$\frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$



Calculating Sensitivity

```
#Sensitivity=TP/TP+FN

Sensitivity = 8874/(8874 + 238 )
Sensitivity*100

97.38805970149254
```

Step 14: Calculating the sensitivity.

9.3 What is Specificity?

The process of predicting the negative value correctly is called specificity.

It is used to predict the ability of the model that whether or not an observation belongs to a specific category.

It's used to see how well the model can predict true negative data.

Specificity - Formula

Formula:

$$\frac{\text{True Negative}}{\text{True Negative + False Positive}}$$

Calculating Specificity

Step 15: Calculating specificity..

```
#Specificity=TN/TN+FP

Specifiicty = 481/(481 + 707 )
Specifiicty*100

40.48821548821549
```

9.4 What is ROC Curve?

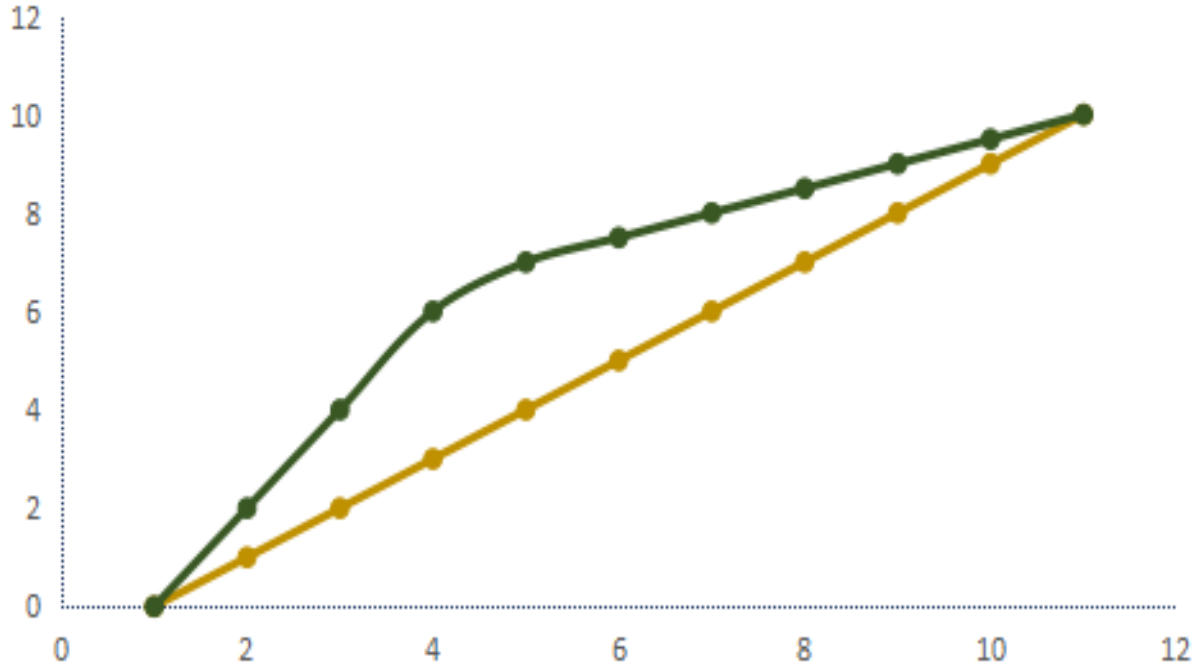
The ROC curve depicts the sensitivity-specificity relationship.

We construct true positive rate against the false positive rate to get a ROC Curve.

The better performance in ROC curve is seen when the curve is closer to top-left corner and bad performance is seen when the curve is more toward 45 degrees.

What is ROC Curve?

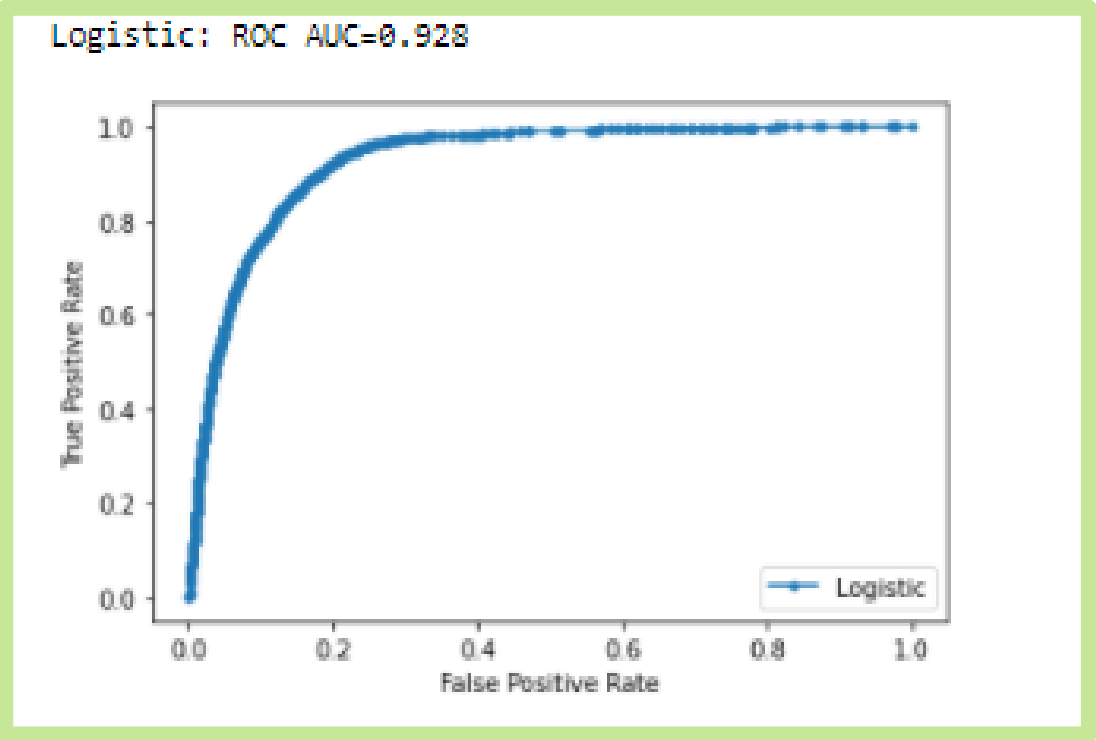
ROC Curve shows us the relationship between true positive and true negative.



ROC Curve

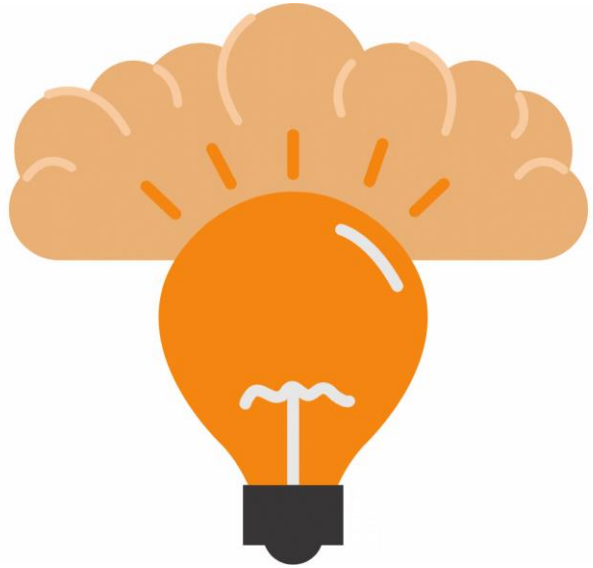
```
# roc curve and auc
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

# calculate scores
auc = roc_auc_score(np.array(y_test), test_pred_prob[:, 1])

# summarize scores

print('Logistic: ROC AUC=%.3f' % (auc))

# calculate roc curves

fpr, tpr, _ = roc_curve(np.array(y_test), test_pred_prob[:, 1])
# plot the roc curve for the model

pyplot.plot(fpr, tpr, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

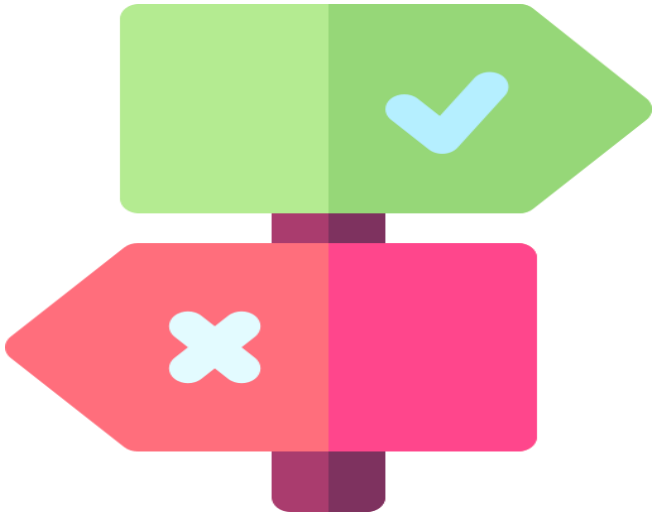
Step 16: Plotting an ROC Curve.



10. What is Cut-off?

Cut-off refers to the point where the decision is made.





Eg., Consider a cut-off, if probability of a is more than cut-off than b then the machine chooses a as a final decision.

Cut-off

Finding Optimum Cutoff of the Model:

For finding optimum cut-off value we will use predict_proba().

This helps us to get an array of list containing class of probabilities for input points.

Cut-off - Code

```
proba_valid = pd.DataFrame(logisticRegr.predict_proba(x_test)[:, 1])
```

```
proba_valid
```

	0
0	0.016558
1	0.137892
2	0.188031
3	0.014164
4	0.019575
...	...
10295	0.053365
10296	0.006462
10297	0.009940
10298	0.026222
10299	0.013465

10300 rows × 1 columns

Finding optimum cutoff.

Cut-off - Code

Combining the predictions by the model and the probability of predicting.

```
df_new=pd.DataFrame({'Predictions':test_pred})
```

```
df_new
```

	Predictions
0	0
1	0
2	0
3	0
4	0
...	...
10295	0
10296	0
10297	0
10298	0
10299	0

10300 rows × 1 columns

Cut-off - Code

```
df_new.insert(1, "Y_predict_proba", proba_valid)
```

```
df_new
```

	Predictions	Y_predict_proba
0	0	0.016558
1	0	0.137892
2	0	0.188031
3	0	0.014164
4	0	0.019575
...	...	...
10295	0	0.053365
10296	0	0.006462
10297	0	0.009940
10298	0	0.026222
10299	0	0.013465

10300 rows × 2 columns

y_predict_proba contains the probability value of getting either 0 or 1.

Checking for null values with isna().

```
df_new.isna().sum()
```

```
Predictions       0
Y_predict_proba   0
dtype: int64
```

Cut-off

Firstly, we will categories the data.

If y_predict_proba > 0.1 then probability = 1.
Else 0

Same goes for other cutoffs.

Cut-off - Code

```
df_new['Y_pred_0.1']=np.where((df_new['Y_predict_proba']>0.1), 1,0)
```

```
df_new
```

	Predictions	Y_predict_proba	Y_pred_0.1
0	0	0.016558	0
1	0	0.137892	1
2	0	0.188031	1
3	0	0.014164	0
4	0	0.019575	0
...	...	...	...
10295	0	0.053365	0
10296	0	0.006462	0
10297	0	0.009940	0
10298	0	0.026222	0
10299	0	0.013465	0

10300 rows × 3 columns

y_predict_proba > 0.1=1

y_predict_proba <= 0.1=0

Repeating the process for all cutoffs.

```
df_new['Y_pred_0.2']=np.where(df_new['Y_predict_proba']>0.2,1,0)
df_new['Y_pred_0.3']=np.where(df_new['Y_predict_proba']>0.3, 1,0)
df_new['Y_pred_0.4']=np.where(df_new['Y_predict_proba']>0.4, 1,0)
df_new['Y_pred_0.5']=np.where(df_new['Y_predict_proba']>0.5, 1,0)
df_new['Y_pred_0.6']=np.where(df_new['Y_predict_proba']>0.6, 1,0)
df_new['Y_pred_0.7']=np.where(df_new['Y_predict_proba']>0.7, 1,0)
df_new['Y_pred_0.8']=np.where(df_new['Y_predict_proba']>0.8, 1,0)
df_new['Y_pred_0.9']=np.where(df_new['Y_predict_proba']>0.9, 1,0)
```

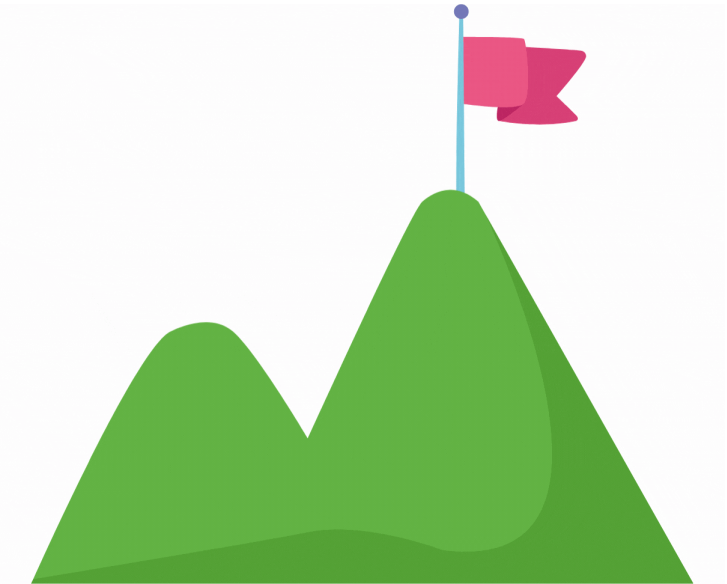
```
df_new
```

	Predictions	Y_predict_proba	Y_pred_0.1	Y_pred_0.2	Y_pred_0.3	Y_pred_0.4	Y_pred_0.5	Y_pred_0.6	Y_pred_0.7	Y_pred_0.8	Y_pred_0.9
0	0	0.016558	0	0	0	0	0	0	0	0	0
1	0	0.137892	1	0	0	0	0	0	0	0	0
2	0	0.188031	1	0	0	0	0	0	0	0	0
3	0	0.014164	0	0	0	0	0	0	0	0	0
4	0	0.019575	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
10295	0	0.053365	0	0	0	0	0	0	0	0	0
10296	0	0.006462	0	0	0	0	0	0	0	0	0
10297	0	0.009940	0	0	0	0	0	0	0	0	0
10298	0	0.026222	0	0	0	0	0	0	0	0	0
10299	0	0.013465	0	0	0	0	0	0	0	0	0

10300 rows × 11 columns

Cut-off

From the above cutoffs we will be calculating sensitivity and specificity of every cutoff.



Which ever has the highest sensitivity + specificity value we will be considering that as an optimal cutoff value.

Cut-off - Code

```
sen1=c1[0,0]/(c1[0,0]+c1[0,1])
sen2=c2[0,0]/(c2[0,0]+c2[0,1])
sen3=c3[0,0]/(c3[0,0]+c3[0,1])
sen4=c4[0,0]/(c4[0,0]+c4[0,1])
sen5=c5[0,0]/(c5[0,0]+c5[0,1])
sen6=c6[0,0]/(c6[0,0]+c6[0,1])
sen7=c7[0,0]/(c7[0,0]+c7[0,1])
sen8=c8[0,0]/(c8[0,0]+c8[0,1])
sen9=c9[0,0]/(c9[0,0]+c9[0,1])
```

```
sep1=c1[1,1]/(c1[1,1]+c1[1,0])
sep2=c2[1,1]/(c2[1,1]+c2[1,0])
sep3=c3[1,1]/(c3[1,1]+c3[1,0])
sep4=c4[1,1]/(c4[1,1]+c4[1,0])
sep5=c5[1,1]/(c5[1,1]+c5[1,0])
sep6=c6[1,1]/(c6[1,1]+c6[1,0])
sep7=c7[1,1]/(c7[1,1]+c7[1,0])
sep8=c8[1,1]/(c8[1,1]+c8[1,0])
sep9=c9[1,1]/(c9[1,1]+c9[1,0])
```

Calculating sensitivity and specificity.

Appending values into dataframe.

```
d_cutoff_value=d_cutoff_value.append({'cutoff':0.1, 'Sensitivity': sen1,'Specificity':sep1}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.2, 'Sensitivity': sen2,'Specificity':sep2}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.3, 'Sensitivity': sen3,'Specificity':sep3}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.4, 'Sensitivity': sen4,'Specificity':sep4}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.5, 'Sensitivity': sen5,'Specificity':sep5}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.6, 'Sensitivity': sen6,'Specificity':sep6}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.7, 'Sensitivity': sen7,'Specificity':sep7}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.8, 'Sensitivity': sen8,'Specificity':sep8}, ignore_index=True)
d_cutoff_value=d_cutoff_value.append({'cutoff':0.9, 'Sensitivity': sen9,'Specificity':sep9}, ignore_index=True)
```

```
d_cutoff_value
```

	cutoff	Sensitivity	Specificity
0	0.1	0.781295	1.000000
1	0.2	0.882603	1.000000
2	0.3	0.938551	1.000000
3	0.4	0.975088	1.000000
4	0.5	1.000000	1.000000
5	0.6	1.000000	0.734234
6	0.7	1.000000	0.513514
7	0.8	1.000000	0.345345
8	0.9	1.000000	0.214715

Cut-off

From above table we have analyzed that highest total value is 2 and for that cutoff value is 0.5.

Therefore, the optimal cutoff value = 0.5.

Cut-off - Code

Choosing the cutoff with highest Total_val.

	cutoff	Sensitivity	Specificity	Total_val
0	0.1	0.781295	1.000000	1.781295
1	0.2	0.882603	1.000000	1.882603
2	0.3	0.938551	1.000000	1.938551
3	0.4	0.975088	1.000000	1.975088
4	0.5	1.000000	1.000000	2.000000
5	0.6	1.000000	0.734234	1.734234
6	0.7	1.000000	0.513514	1.513514
7	0.8	1.000000	0.345345	1.345345
8	0.9	1.000000	0.214715	1.214715

Here, 2 is the highest Total_val. Hence, 0.5 is chosen as the cutoff.

India: +91-7022374614

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

support@intellipaat.com

24/7 Chat with Our Course Advisor