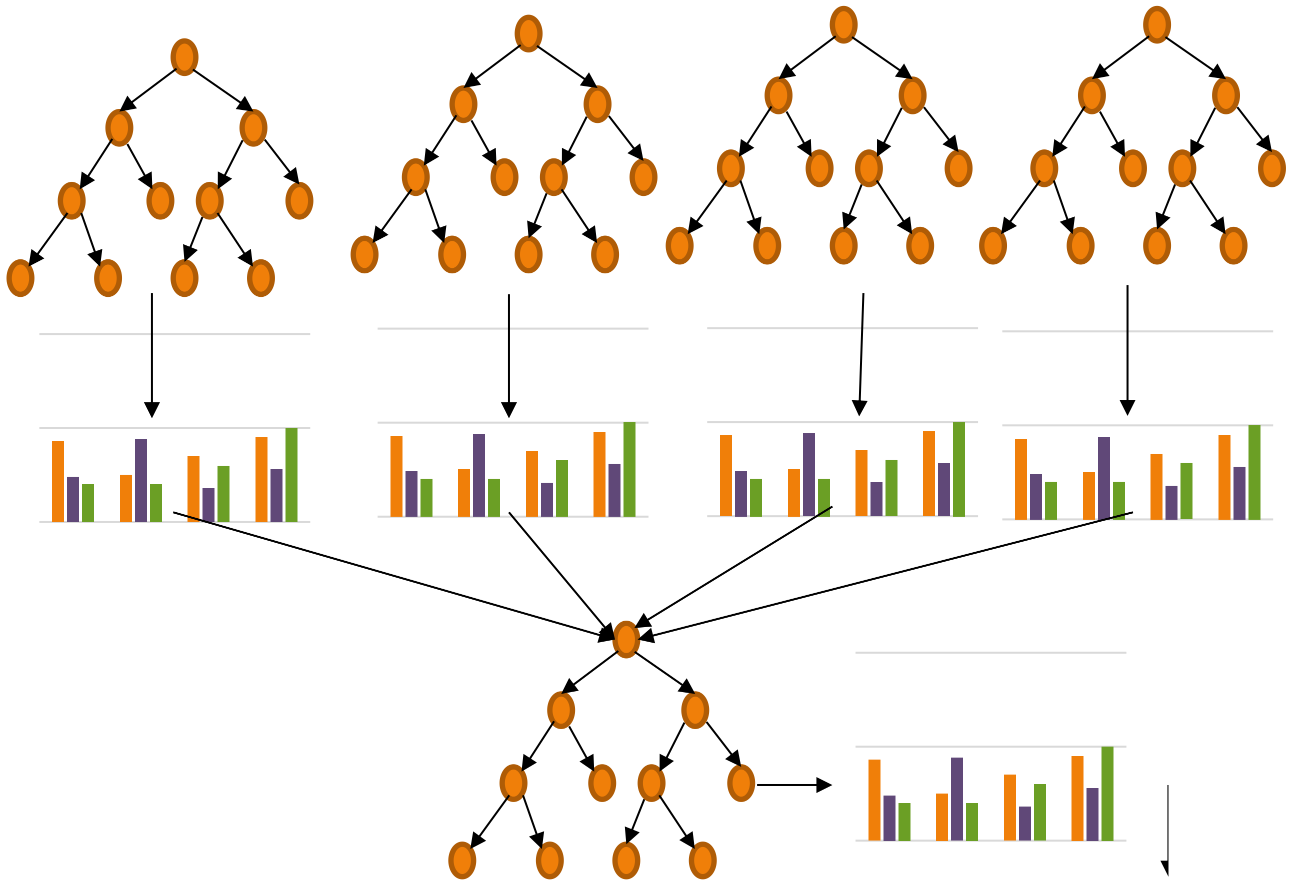
Random forest

Customer Churn Analysis in Telecom



Agenda

01 Business Problem

02 Solution Approach

03 EDA

04 Missing Value analysis

05 Dimensionality Reduction

- 5.1 Chi-Square test
- 5.1 Variable importance in Random Forest

6 Train and Test Split

Agenda

7 Model Fitting

8 Model Validation

8.3 Accuracy

8.2 Sensitivity

8.3 Specificity

8.4 ROC Curve

9 Finding Optimal Cut-off

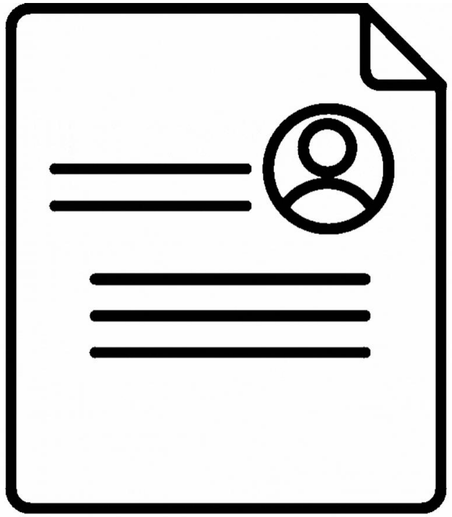
10 Classification Report

Business Problem - Background

In the telecom sector, a massive quantity of information is being generated on a day by day foundation because of a sizable customer base. Decision makers and commercial enterprise analysts emphasized that achieving new clients is dearer than maintaining the present ones. Business analysts and purchaser dating management analyzers want to recognize the reason behind churning clients.

Solution Approach

Churn prediction model that uses classification, as well as, techniques to identify the churn customers and provides the factors behind the churning of customers in the telecom sector.



We will train our machine learning model using random forest to learn the patterns in the churn data.

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

```
churn_data=pd.read_csv(r'C:\Users\lenovo\Downloads\customer_churn.csv')
```

```
churn_data.head()
```

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

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

EDA - Code Walk

Step 3: Checking all the statistical data.

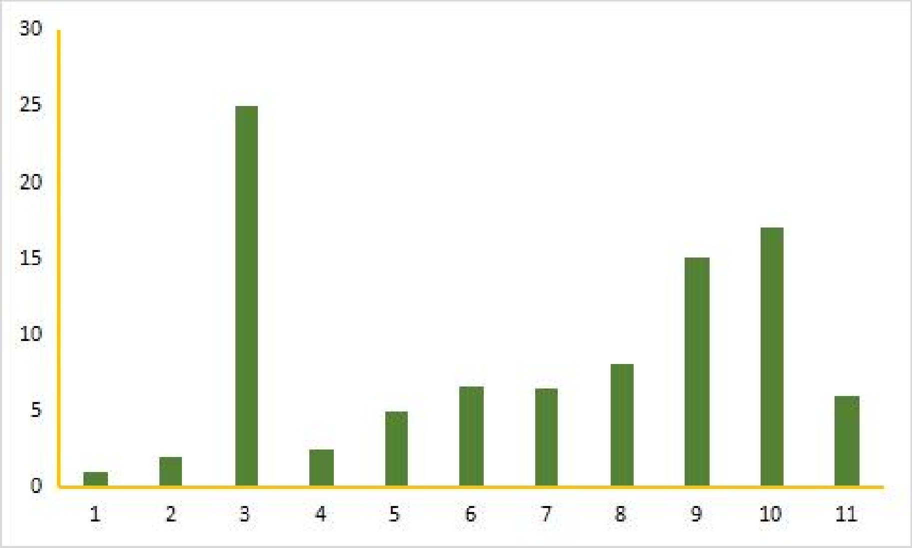
describe() helps us to check for mean, std, quartiles, minimum and maximum values in our dataset.

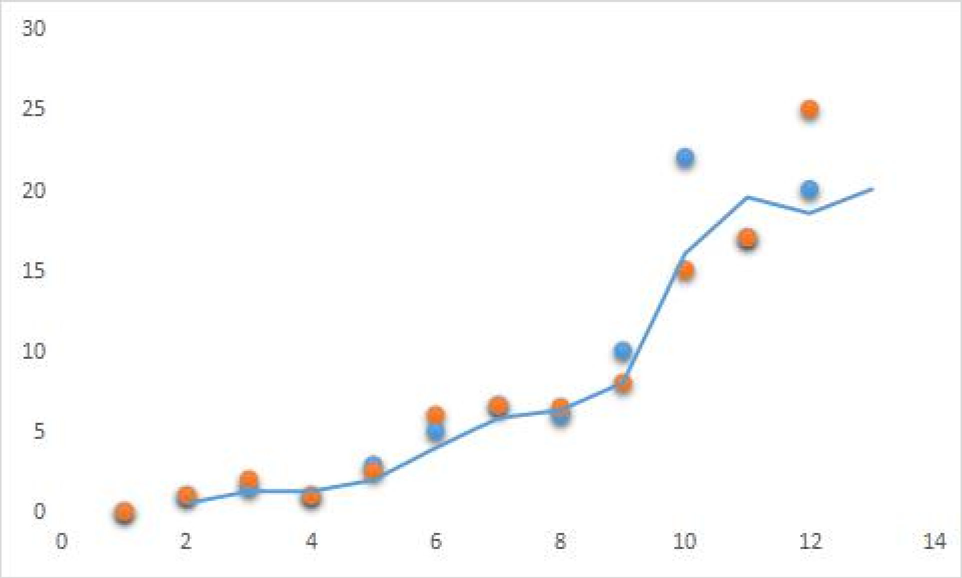
```
churn_data.describe()
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7032.000000
mean	0.162147	32.371149	64.761692	2283.300441
std	0.368612	24.559481	30.090047	2266.771362
min	0.000000	0.000000	18.250000	18.800000
25%	0.000000	9.000000	35.500000	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.850000	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

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.

```
churn_data.dtypes
```

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                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

```
churn_data.isna().sum()

customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

Check for the missing value by the use of isna() function.

Missing Data Analysis - Code Traverse

```
churn_data.dropna(inplace=True)
```

```
churn_data.isna().sum()
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

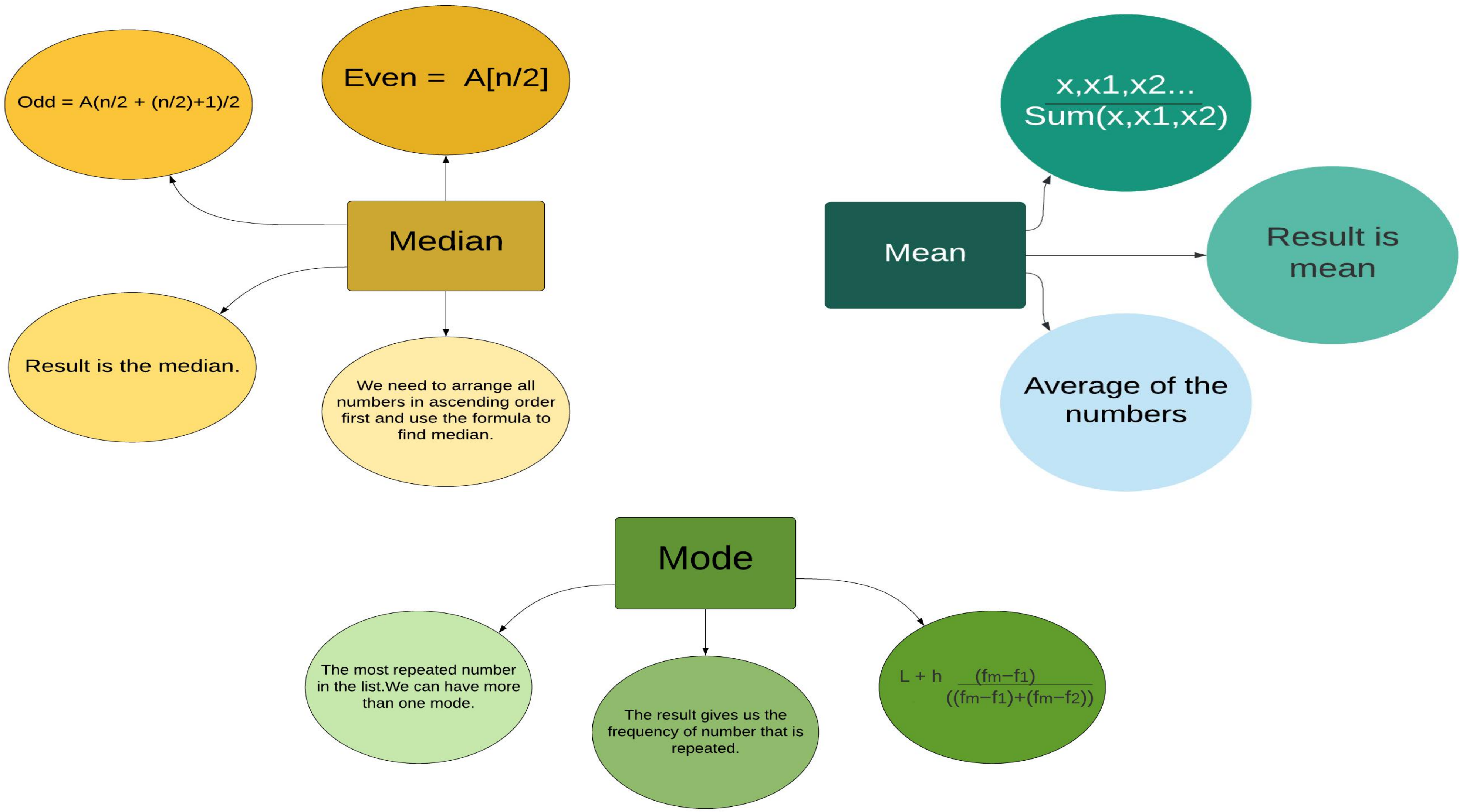
Treat the missing value by dropping the columns which have null values

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



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

Mean = 31 years

Median = 18

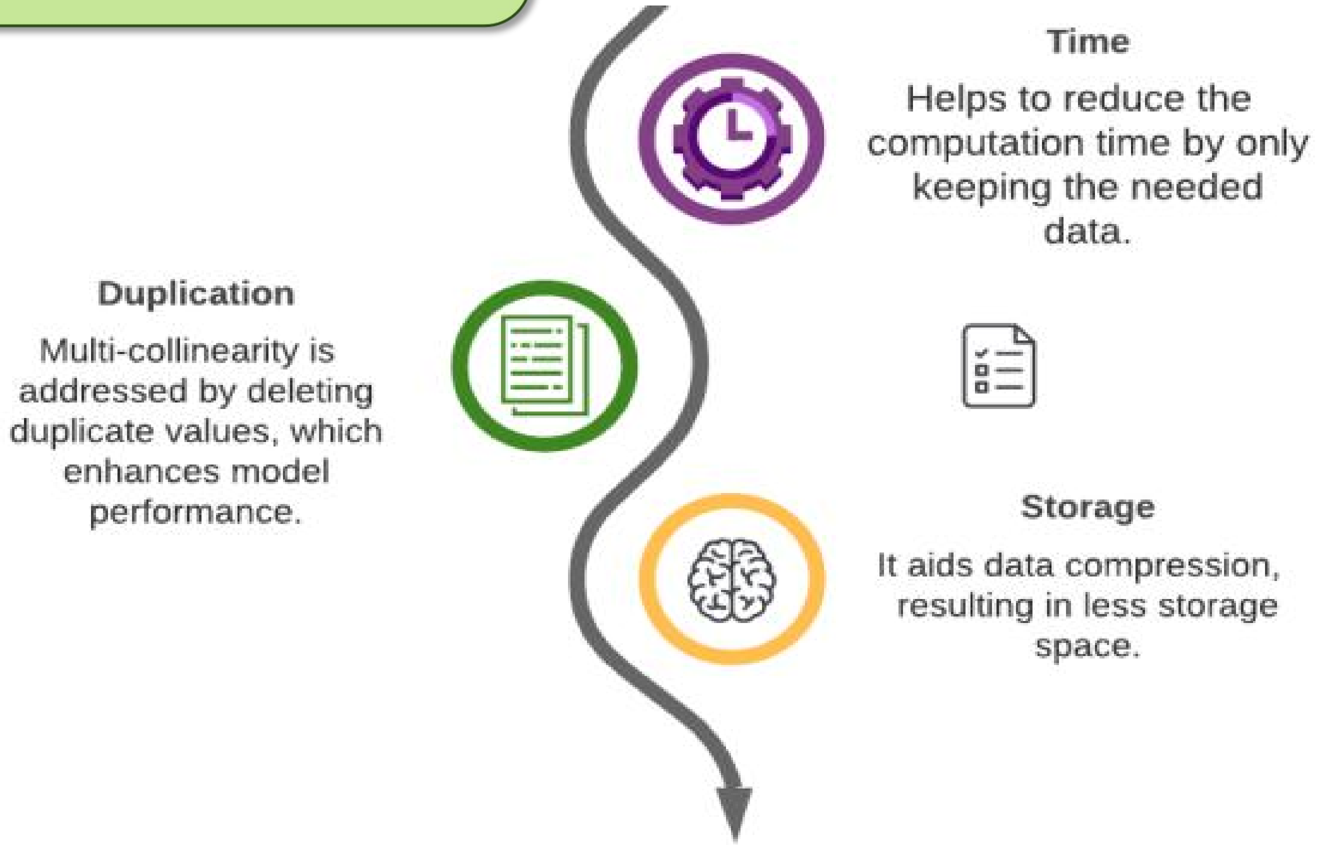
If we see the above example closely we can see that median gives us a more realistic value. Due to this nature of median we prefer median over mean.

5. What is Dimensionality Reduction?

By getting a collection of primary variables, dimensionality reduction reduces the number of repeated variables under consideration and it also helps in identifying strong and weak predators.

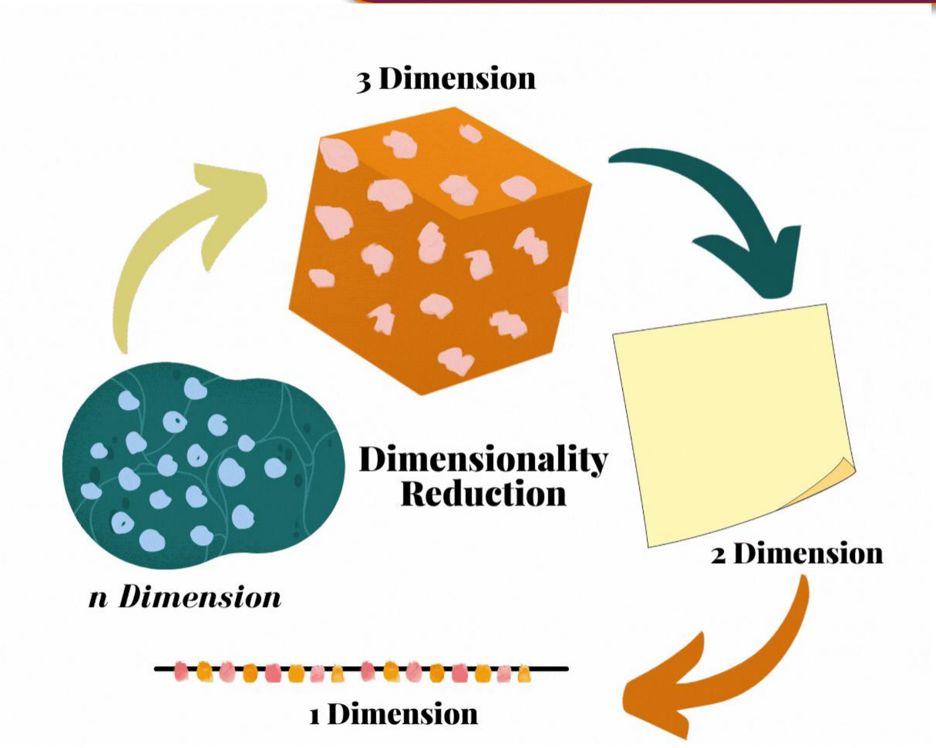
Chi-Square Test

Variable Importance



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 corelation between target variable and independent variable is strong or not. if p<0.05 ,variable importance is high.

```python
###Chisq Test for Independence for all object fields
col_list = list(churn_data.columns)
col_list.remove('Churn')
df=pd.DataFrame(columns=['Feature','P-value'])

for col in col_list:
    if churn_data[[col]][col].dtype == 'object':

        ###Chisq Test for Independence
        dataset_table=pd.crosstab(churn_data[col],churn_data['Churn'])
        #print(dataset_table)


        #Observed Values
        Observed_Values = dataset_table.values
        #print("Observed Values :-\n",Observed_Values)

        val=chi2_contingency(dataset_table)
        #val

        Expected_Values=val[3]
        #Expected_Values

        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=len(dataset_table.iloc[0:2,0])
        no_of_columns=len(dataset_table.iloc[0,0:2])
        ddof=(no_of_rows-1)*(no_of_columns-1)
        #print("Degree of Freedom:-",ddof)
        alpha = 0.05
        #print("chi-square statistic:-",chi_square_statistic)
        #scipy.stats.chi2.ppf() function

        critical_value=scipy.stats.chi2.ppf(q=1-alpha,df=ddof)
        #print('critical_value:',critical_value)

        #p-value
        p_value=1-chi2.cdf(x=chi_square_statistic,df=ddof)
        #print(col)
        #print('p-value:',p_value)
        df=df.append({'Feature':col, 'P-value': p_value}, ignore_index=True)
        #print('Significance Level: ',alpha)
        #print('Degree of Freedom: ',ddof)
        #print('p-value:',p_value)
        #df.append()
df
```

	Feature	P-value
0	gender	0.473665
1	Partner	0.000000
2	Dependents	0.000000
3	PhoneService	0.326886
4	MultipleLines	0.000787
5	InternetService	0.000000
6	OnlineSecurity	0.000000
7	OnlineBackup	0.000000
8	DeviceProtection	0.000000
9	TechSupport	0.000000
10	StreamingTV	0.000000
11	StreamingMovies	0.000000
12	Contract	0.000000
13	PaperlessBilling	0.000000
14	PaymentMethod	0.000000

5.2 Variable Importance

Variable importance is checking which variable is having more predictive power.

High-importance variables are outcome drivers, and their values have a big impact on the final numbers.

5.2 Variable Importance

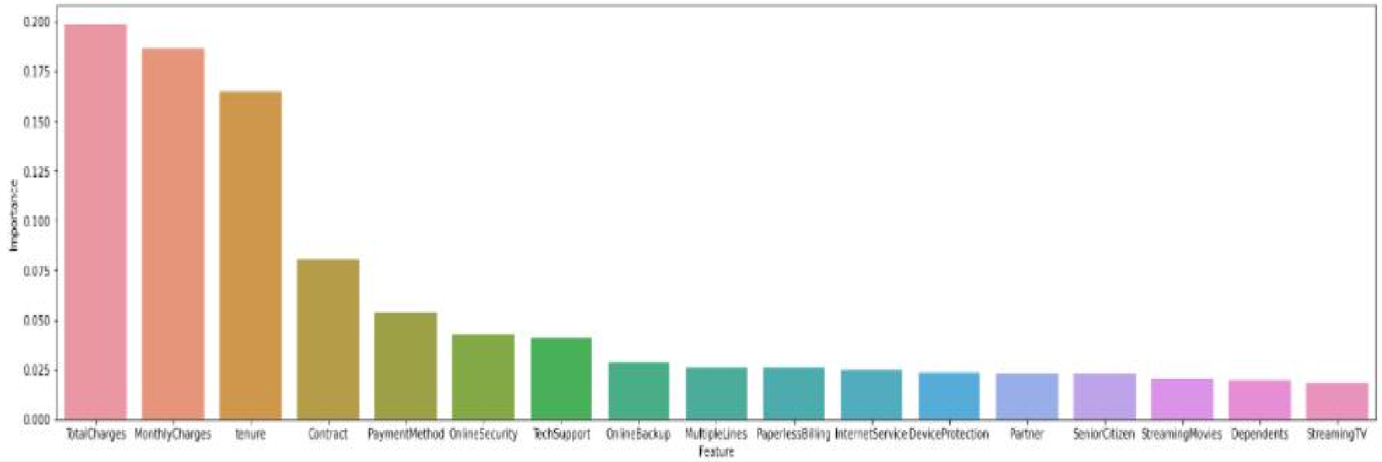
Checking for variable importance and visualizing it.

```python
from sklearn.datasets import make_classification
importance = classifier1.feature_importances_
```

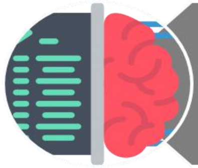
```python
importance= pd.Series(importance)
importance
```

```python
import seaborn as sns
plt.figure(figsize=(25,6))
# make barplot and sort bars
sns.barplot(x='Feature',y="Importance",data=df, order=df.sort_values('Importance', ascending=False).Feature)
```

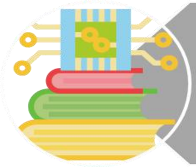
```
<AxesSubplot:xlabel='Feature', ylabel='Importance'>
```



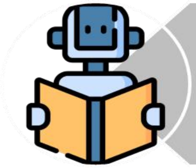
6. What is Train and Test of Data ?



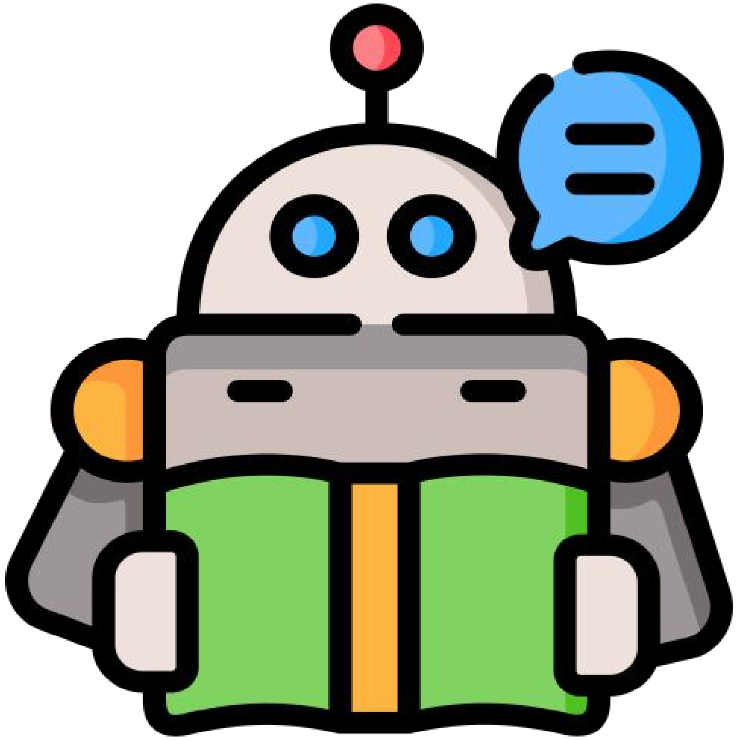
It is the process of splitting data into train and test sections so we can train the model with the train data and test if our model is working fine with test data.



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.



```
X = churn_data.iloc[:, :-1].values
y = churn_data.iloc[:, -1].values
```

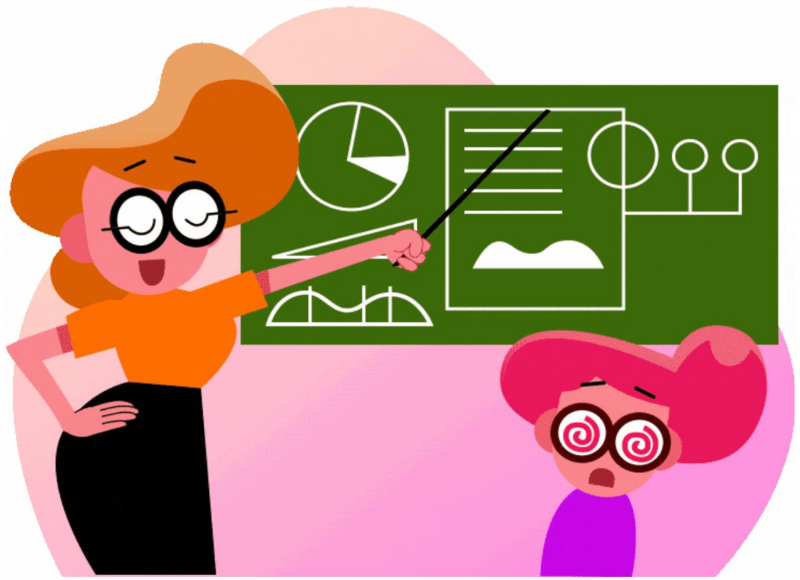
```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.2, random_state=0)
```

7. What is Model Fitting ?

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

We also refer to it as model fitting.

```
classifier1 = RandomForestClassifier(n_estimators=500, random_state=0)
classifier1.fit(X_train1, y_train1)
y_pred1 = classifier1.predict(X_test1)
```



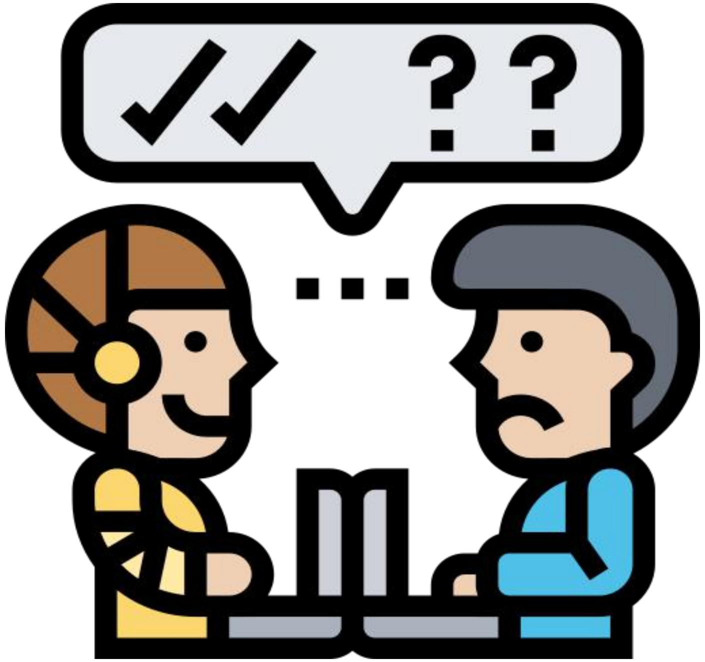
8. What is Model Validation ?

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

This gives us the description on the model performance.

This gives us the insites on the model performance.

This is also called model testing.



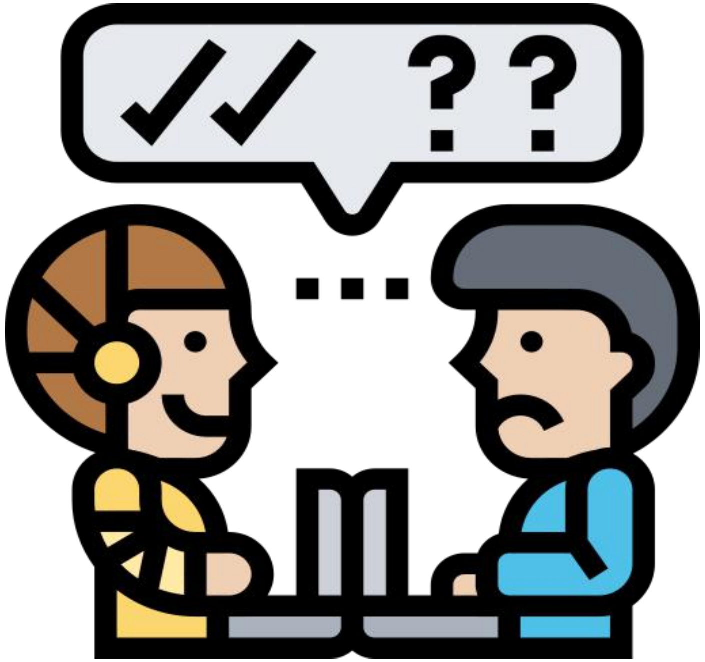
What is Model Validation ?

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

This gives us the description on the model performance.

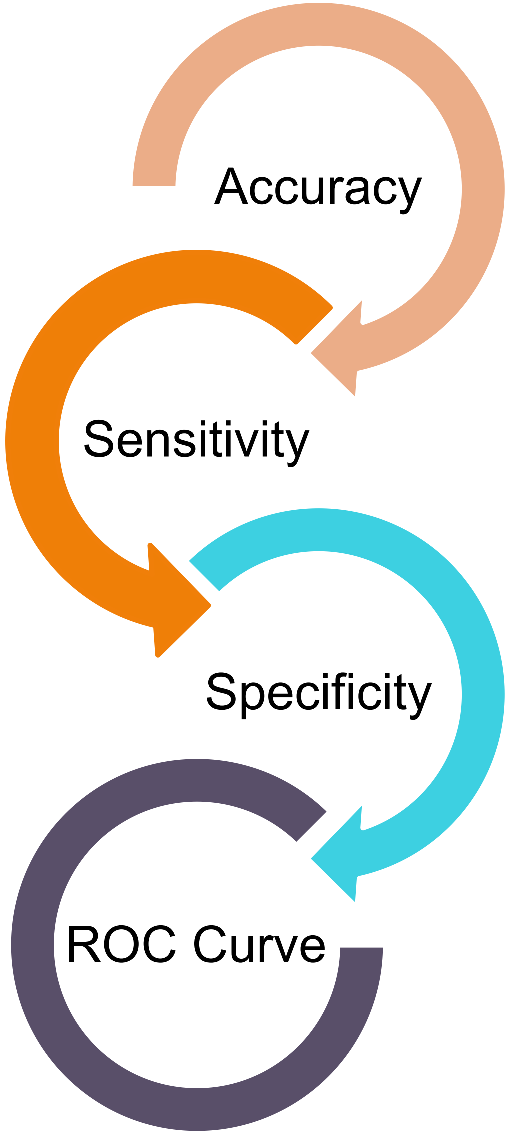
This gives us the insites on the model performance.

This is also called model testing.



Model Validation

Shown are the steps for model validation.



Model Validation

Step 12: Scoring our model.

```
y_pred1
```

```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

```
y_pred1.shape
```

```
(1407,)
```

```
c1=confusion_matrix(y_test1, y_pred1)
print(c1)
```

```
[[926 112]
 [190 179]]
```

```
a1=accuracy_score(y_test1, y_pred1)
print(a1)
```

```
0.7853589196872779
```

8.1 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

```
a1=accuracy_score(y_test1, y_pred1)
print(a1)
```

```
0.7853589196872779
```

Step 13: Finding the accuracy of the model.

8.2 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

```
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])
```

Step 14: Calculating the sensitivity.

8.3 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..

```
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])
```

8.4 What is ROC Curve?

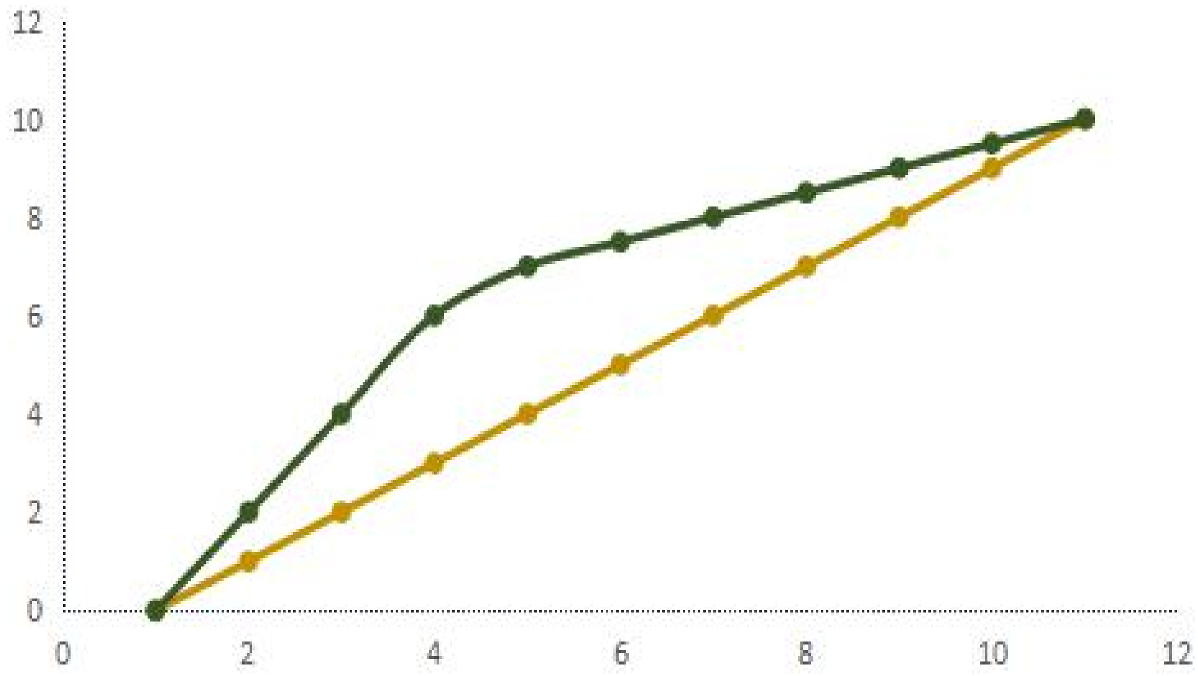
The ROC curve depicts the sensitivity-specificity relationship.

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?

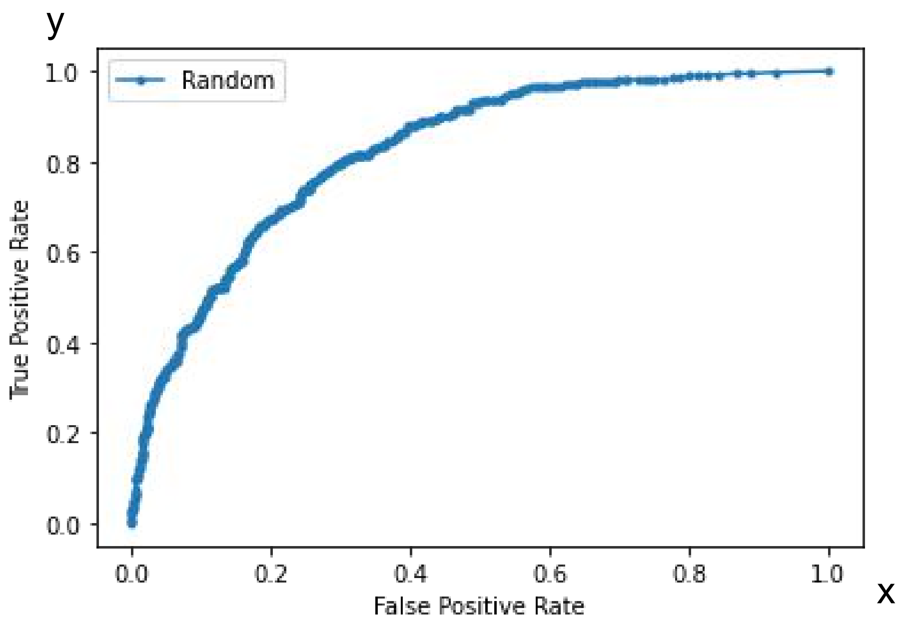
ROC Curve shows us the relationship between true positive and true negative.



ROC Curve

```
#roc curve

# roc curve and auc
from sklearn.datasets import make_classification
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_test1), test_pred_prob[:, 1])

# summarize scores

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

# calculate roc curves

fpr, tpr, _ = roc_curve(np.array(y_test1), test_pred_prob[:, 1])
# plot the roc curve for the model

pyplot.plot(fpr, tpr, marker='.', label='Random')
# 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.



x=False Positive Rate
y=True Positive Rate

9.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 = classifier1.predict_proba(X_test1)[:, 1]
```

```
proba_valid
```

```
array([0.2923, 0.122 , 0.1085, ..., 0.8   , 0.114 , 0.378 ])
```

Finding optimum cutoff.

Cut-off - Code

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

```
df_new=pd.DataFrame({'Predictions':y_pred1})
```

```
df_new
```

	Predictions
0	0
1	0
2	0
3	0
4	1
...	...
1402	0
1403	0
1404	1
1405	0
1406	0

Cut-off - Code

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

```
df_new
```

	Predictions	Y_predict_proba
0	0	0.2923
1	0	0.1220
2	0	0.1085
3	0	0.2980
4	1	0.9960
...	...	...
1402	0	0.0800
1403	0	0.1680
1404	1	0.8000
1405	0	0.1140
1406	0	0.3780

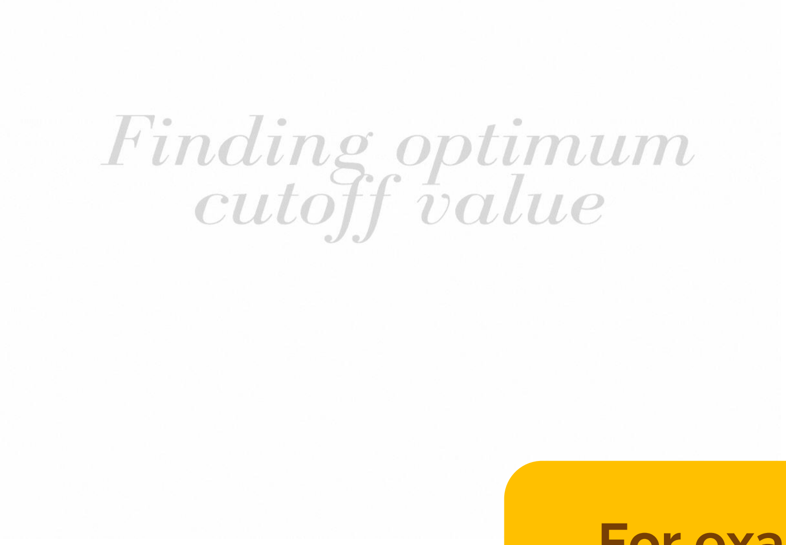
Checking for null values with isna().

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

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

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

Cut-off



For example if cut-off is 0.1 then y_predict_proba=1 else 0 .

Firstly, we will categories the data.

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['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	0.2923	1	1	0	0	0	0	0	0	0
1	0	0.1220	1	0	0	0	0	0	0	0	0
2	0	0.1085	1	0	0	0	0	0	0	0	0
3	0	0.2980	1	1	0	0	0	0	0	0	0
4	1	0.9960	1	1	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...
1402	0	0.0800	0	0	0	0	0	0	0	0	0
1403	0	0.1680	1	0	0	0	0	0	0	0	0
1404	1	0.8000	1	1	1	1	1	1	1	0	0
1405	0	0.1140	1	0	0	0	0	0	0	0	0
1406	0	0.3780	1	1	1	0	0	0	0	0	0

1407 rows × 11 columns

Repeating the process for all cutoffs.

y_predict_proba > 0.1=1

y_predict_proba <= 0.1=0

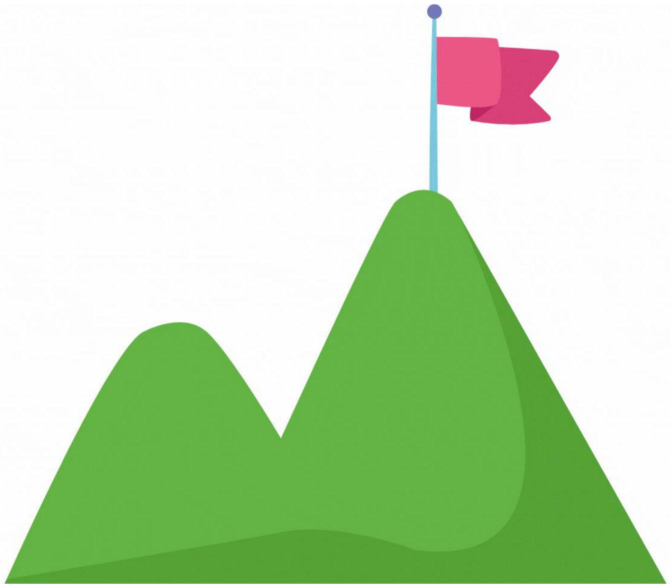
```
df_new['Predictions'] = df_new['Predictions'].map({'Yes': 1, 'No': 0})
```

```
df_new#0.5 as cutoff
#label predictions
```

	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	NaN	0.2923	1	1	0	0	0	0	0	0	0
1	NaN	0.1220	1	0	0	0	0	0	0	0	0
2	NaN	0.1085	1	0	0	0	0	0	0	0	0
3	NaN	0.2980	1	1	0	0	0	0	0	0	0
4	NaN	0.9960	1	1	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...
1402	NaN	0.0800	0	0	0	0	0	0	0	0	0
1403	NaN	0.1680	1	0	0	0	0	0	0	0	0
1404	NaN	0.8000	1	1	1	1	1	1	1	0	0
1405	NaN	0.1140	1	0	0	0	0	0	0	0	0
1406	NaN	0.3780	1	1	1	0	0	0	0	0	0

Cut-off

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



Whichever 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=pd.DataFrame(columns=['cutoff','Sensitivity','Specificity'])
```

```
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)
```

Cut-off

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

```
d_cutoff_value
```

	cutoff	Sensitivity	Specificity	Total_val
0	0.1	0.471326	1.000000	1.471326
1	0.2	0.659498	1.000000	1.659498
2	0.3	0.792115	1.000000	1.792115
3	0.4	0.906810	1.000000	1.906810
4	0.5	1.000000	1.000000	2.000000
5	0.6	1.000000	0.749141	1.749141
6	0.7	1.000000	0.529210	1.529210
7	0.8	1.000000	0.298969	1.298969
8	0.9	1.000000	0.144330	1.144330

Therefore, the optimal cutoff value = 0.5.

Cut-off - Code

Choosing the cutoff with highest Total_val.

```
for i in range(9):
    d_cutoff_value.loc[i,'Total_val']=d_cutoff_value.loc[i,'Sensitivity']+d_cutoff_value.loc[i,'Specificity'
```

```
d_cutoff_value
```

	cutoff	Sensitivity	Specificity	Total_val
0	0.1	0.471326	1.000000	1.471326
1	0.2	0.659498	1.000000	1.659498
2	0.3	0.792115	1.000000	1.792115
3	0.4	0.906810	1.000000	1.906810
4	0.5	1.000000	1.000000	2.000000
5	0.6	1.000000	0.749141	1.749141
6	0.7	1.000000	0.529210	1.529210
7	0.8	1.000000	0.298969	1.298969
8	0.9	1.000000	0.144330	1.144330

```
test_pred_prob = classifier1.predict_proba(X_test1)
test_pred_prob
```

```
array([[0.7077, 0.2923],
       [0.878 , 0.122 ],
       [0.8915, 0.1085],
       ...,
       [0.2   , 0.8   ],
       [0.886 , 0.114 ],
       [0.622 , 0.378 ]])
```

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

Final Prediction

Final prediction has been calculated.

```
for i in range(1407):
    if(df_pred_final.loc[i,'Actual_churn']==1 and df_pred_final.loc[i,'Pred_for_0.5_cutoff']==1):
        df_pred_final.loc[i,'Label']='TP'
    elif(df_pred_final.loc[i,'Actual_churn']==0 and df_pred_final.loc[i,'Pred_for_0.5_cutoff']==1):
        df_pred_final.loc[i,'Label']='FP'
    elif(df_pred_final.loc[i,'Actual_churn']==1 and df_pred_final.loc[i,'Pred_for_0.5_cutoff']==0):
        df_pred_final.loc[i,'Label']='FN'
    elif(df_pred_final.loc[i,'Actual_churn']==0 and df_pred_final.loc[i,'Pred_for_0.5_cutoff']==0):
        df_pred_final.loc[i,'Label']='TN'
```

```
df_pred_final
```

	Actual_churn	Pred_for_0.5_cutoff	Label
0	0	0	TN
1	0	0	TN
2	0	0	TN
3	1	0	FN
4	1	1	TP
...	...	...	...
1402	0	0	TN
1403	1	0	FN
1404	1	1	TP
1405	0	0	TN
1406	0	0	TN

1407 rows × 3 columns

10.Classification Report

```
#getting count of TP labels
df_pred_final[df_pred_final['Label']=='TP'].shape
```

```
(179, 3)
```

```
from sklearn.metrics import classification_report
```

```
print(classification_report(df_pred_final['Actual_churn'], df_pred_final['Pred_for_0.5_cutoff']))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1038
           1       0.62      0.49      0.54       369

    accuracy                           0.79      1407
   macro avg       0.72      0.69      0.70      1407
weighted avg       0.77      0.79      0.78      1407
```

	Predictions	Y_predict_proba
0	0	0.2923
1	0	0.1220
2	0	0.1085
3	0	0.2980
4	1	0.9960