INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

DEPARTMENT
OF
INDUSTRIAL AND MANAGEMENT ENGINEERING

MBA652A

STATISTICAL MODELLING FOR BUSINESSS ANALYSIS

Telecom Customer Churn Prediction

Submitted By-

GROUP 7

Abhinav Pateria **(19114001)**

Abhishek Rai **(19114002)**

Shivam Sharma **(19114012)**

Souradip Patra **(19114014)**

Submitted To-

Dr. Devlina Chatterjee

Assistant Professor ,IIT Kanpur

S.No.	Content
1.	Acknowledgement
2.	Declaration
3.	Introduction • Objective , Methodology , Theory
4.	Data Source
5.	Variables • Dependent Variables • Independent Variables
6.	Exploratory Data Analysis
7.	Data Visualization
8.	Splitting The data set Correlation Matrix
9.	Model Building • Logit Model • Probit Model
10.	Multi-collinearity check Wald test VIF
11.	Recursive Feature Elimination
12.	MaFadden’s Pseudo R-squared Confusion Matrix
13.	ROC Curve
14.	Precision and Recall Tradeoff
15.	Random Forest
16.	Omitted Variable Bias
17.	Conclusion
16.	Python Codes

Acknowledgement

We are highly indebted to **Prof. Devlina Chatterjee,** for her guidance and continuous support in completing this project. It is because of the knowledge and skills acquired during the course work, along with her comprehensive style of teaching, that we are able to understand the subject in a better way and are able to complete this modelling project successfully.

Declaration:

This is to certify that the project report entitled **'Telecom customer churn prediction'** is based on our original research work. Our indebtedness to other works, studies and publication's have been duly acknowledge at the relevant places.

Souradip Patra	AbhishekRai
(IMEMTech)	(IMEMTech)
Shivam Sharma	Abhinav Pateria
(IMEMTech)	(IMEMTech)

Introduction:

Customer churn, also known as customer attrition, customer turnover, or customer defection, is the loss of clients or customers. Telephone service companies, Internet service providers, pay TV companies, insurance firms, and alarm monitoring services, often use customer attrition analysis and customer attrition rates as one of their key business metrics because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients.

Companies usually make a distinction between voluntary churn and involuntary churn. Voluntary churn occurs due to a decision by the customer to switch to another company or service provider, involuntary churn occurs due to circumstances such as a customer's relocation to a long-term care facility, death, or the relocation to a distant location. In most applications, involuntary reasons for churn are excluded from the analytical models.

Analysts tend to concentrate on voluntary churn, because it typically occurs due to factors of the company-customer relationship which companies control, such as how billing interactions are handled or how after-sales help is provided. Predictive analytics use churn prediction models that predict customer churn by assessing their propensity of risk to churn. Since these models generate a small prioritized list of potential defectors, they are effective at focusing customer retention marketing programs on the subset of the customer base who are most vulnerable to churn.

In this dataset we have to predict whether a particular customer will churn or not. So the variable of interest, i.e. the target variable here is ‘Churn’ which will tell us whether or not a particular customer has churned. It is a binary variable 1 means that the customer has churned and 0 means the customer has not churned. With 21 predictor variables we need to predict whether a particular customer will switch to another telecom provider or not.

The dataset contains 21 features including the churn, along with 7042 observations.

Objective:

The objective of the project is to predict whether a customer will churn or not using Logistic Regression Technique and to formulate models depicting the effects of various factors. We will try to evaluate all the possible combination of variables that explains the reason of churning and try to conclude the best possible combination.

Methodology:

- Summary of data and datavisualization.
- Start building model using logit and probit .
- Then on the basis of threshold p-values and VIF(Variance Inflation Factor) we have donebackward elimination and have eliminated insignificantvariables.

Theory:

One heuristic dataset commonly used for regression analysis of telecom customer's churning, Telecom Customer Churn Prediction dataset. Former analysis have found that the churning of a customer in the dataset is most strongly dependent on contract, tenure, total charges.

Data Source:

https://www.kaggle.com/blastchar/telco-customer-churn

Variables:

1) Dependentvariable(Y):

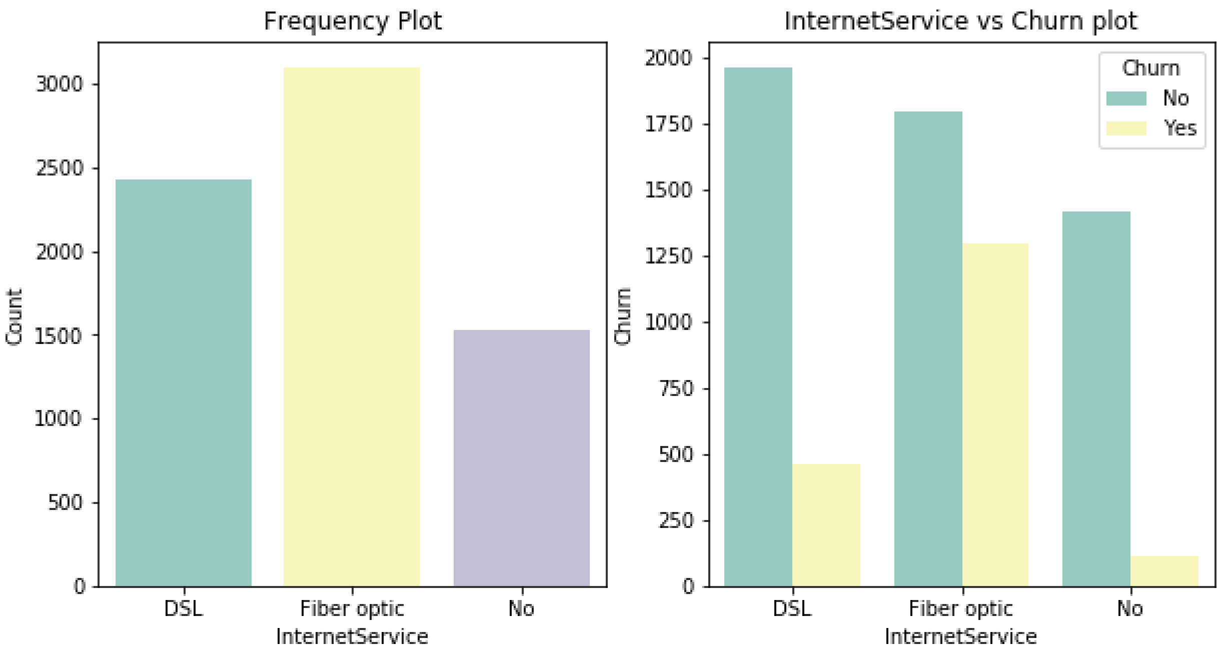
Churn : Whether the customer churned or not (Yes or No).

2) Independentvariables(X):

Customer ID : Customer ID
genderCustomer : gender (female, male)
SeniorCitizen : Whether the customer is a senior citizen or not (1, 0)
Partner : Whether the customer has a partner or not (Yes, No)
Dependents : Whether the customer has dependents or not (Yes, No)
tenure : Number of months the customer has stayed with the company
PhoneService : Whether the customer has a phone service or not (Yes, No)
MultipleLines : Whether the customer has multiple lines or not (Yes, No, No phone service)
InternetService : Customer's internet service provider (DSL, Fiber optic, No)
OnlineSecurity : Whether the customer has online security or not (Yes, No, No internet service)
OnlineBackup : Whether the customer has online backup or not (Yes, No, No internet service)
DeviceProtection : Whether the customer has device protection or not (Yes, No, No internet service)
TechSupport : Whether the customer has tech support or not (Yes, No, No internet service)
StreamingTV : Whether the customer has streaming TV or not (Yes, No, No internet service)
StreamingMovies : Whether the customer has streaming movies or not (Yes, No, No internet service)
Contract : The contract term of the customer (Month-to-month, One year, Two year)
PaperlessBilling : Whether the customer has paperless billing or not (Yes, No)
PaymentMethod : The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
MonthlyCharges : The amount charged to the customer monthly
TotalCharges : The total amount charged to the customer

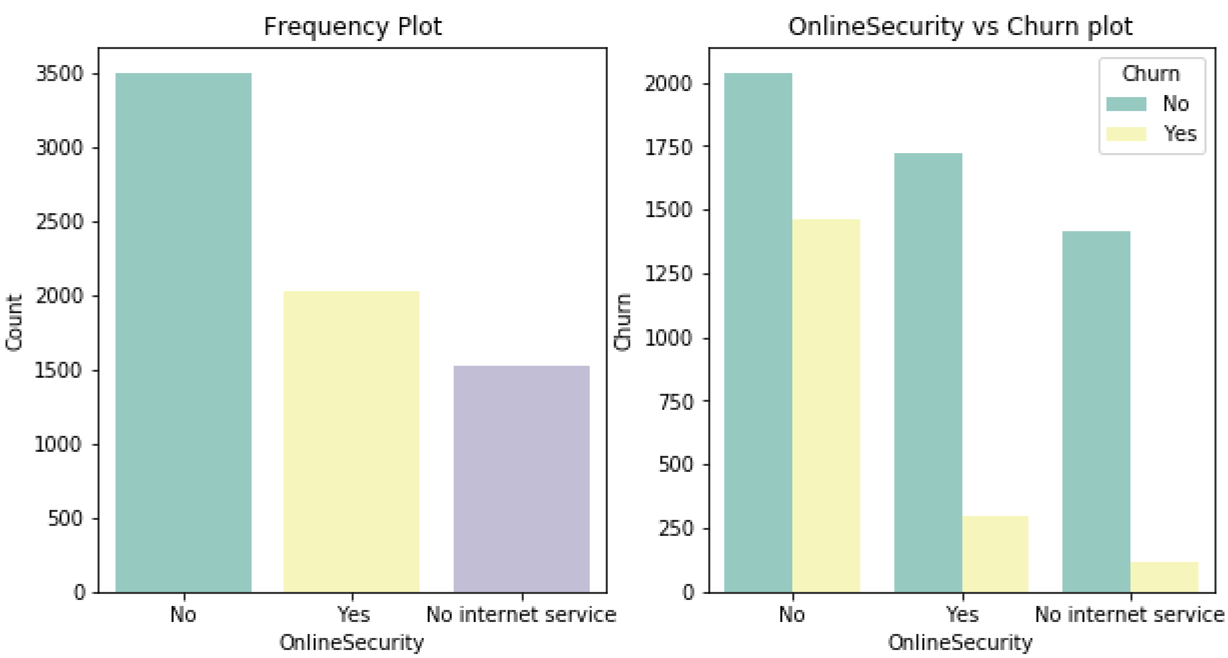
Exploratory Data Analysis:

1) Performing univariate analysis to understand the churn.



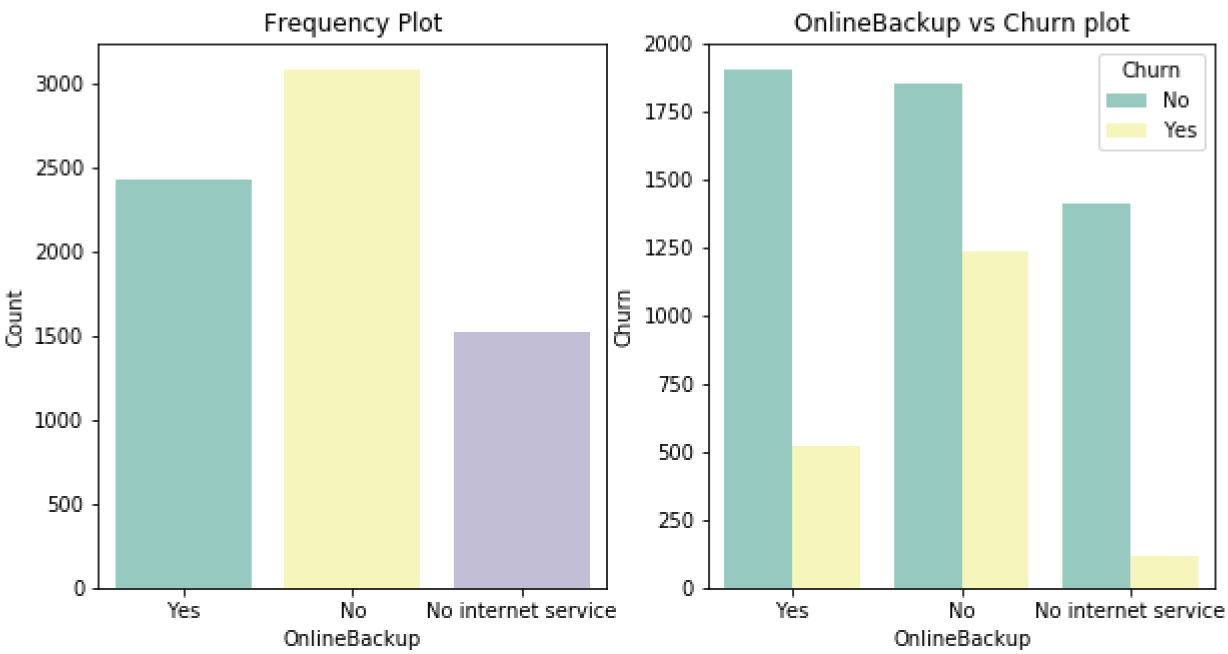
Observations:-

- Churn rate for Fiber optic customers are more when compared to DSL and No service provider.



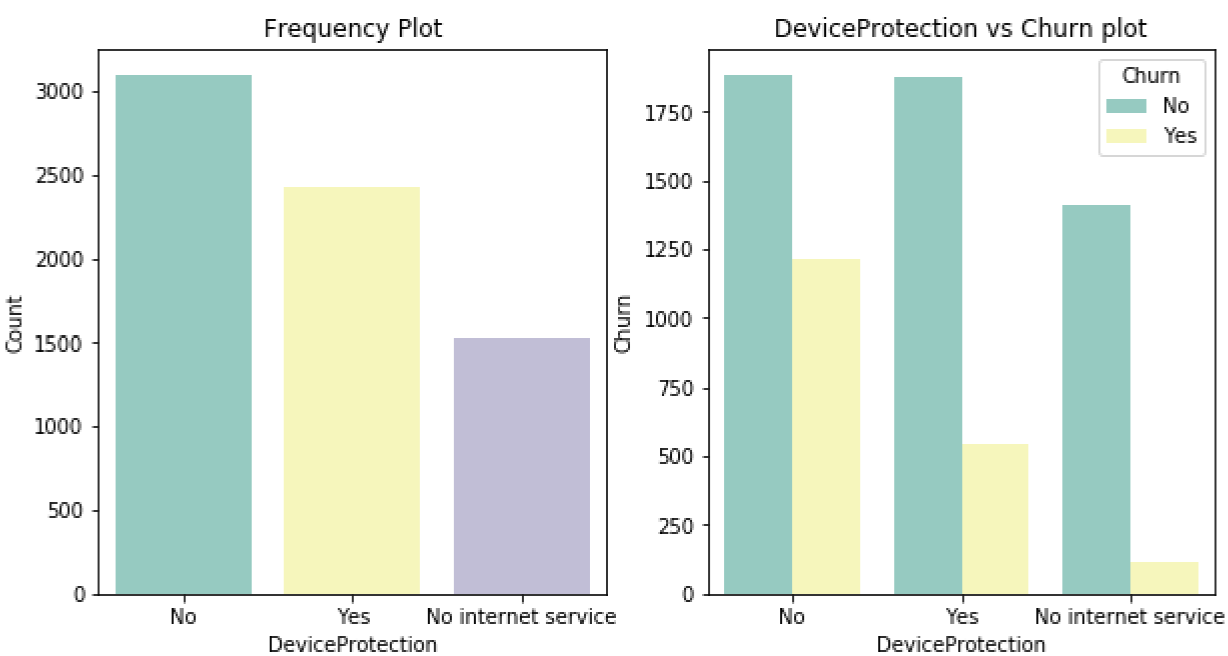
Observations:-

- Churn rate is higher for the subscribers who doesn't have online security.



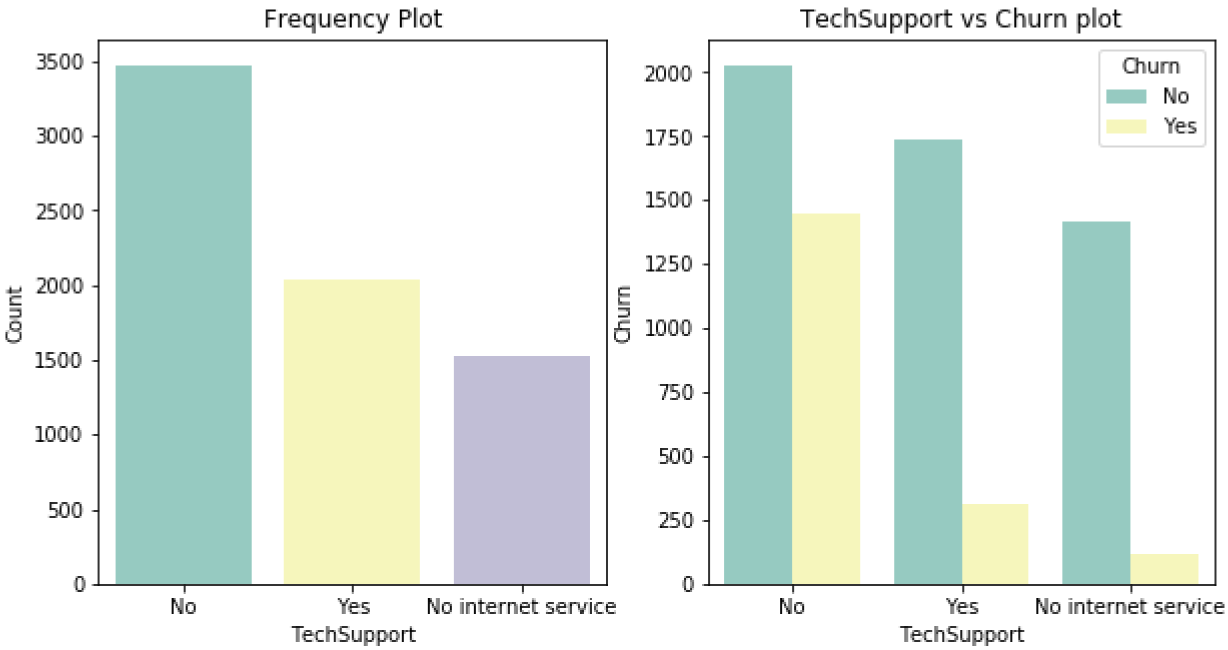
Observations:-

- Churn rate is higher for the subscribers who doesn't have online Backup.



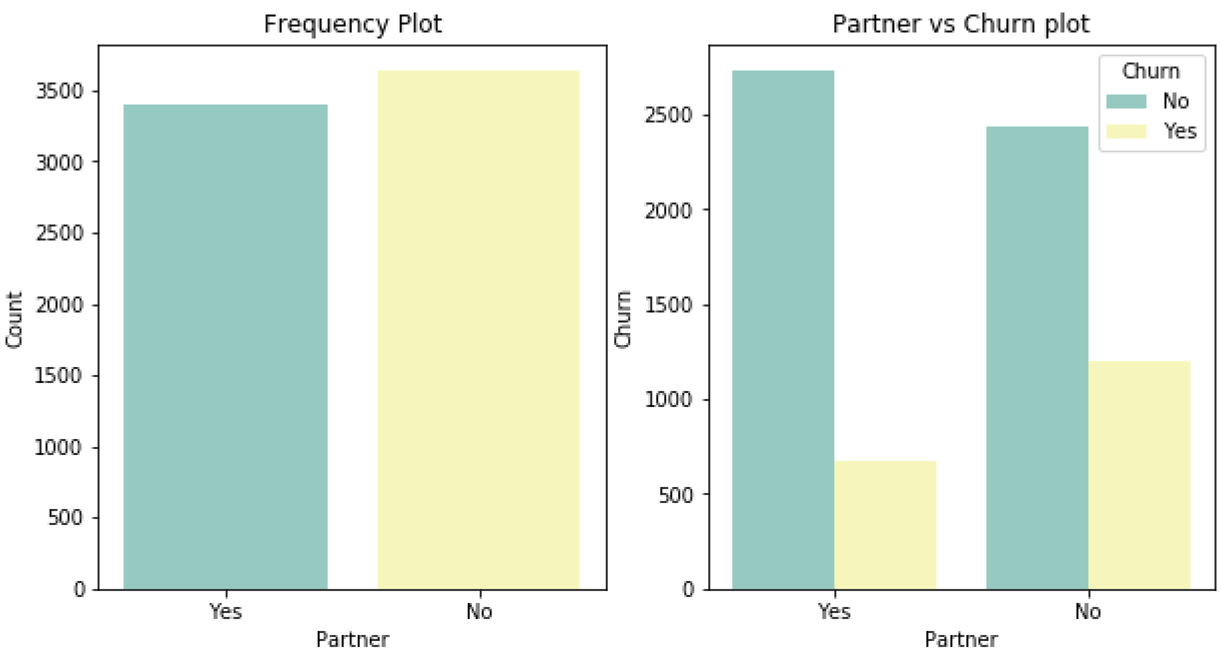
Observations:-

- Churn rate is high for subscribers having no device protection.



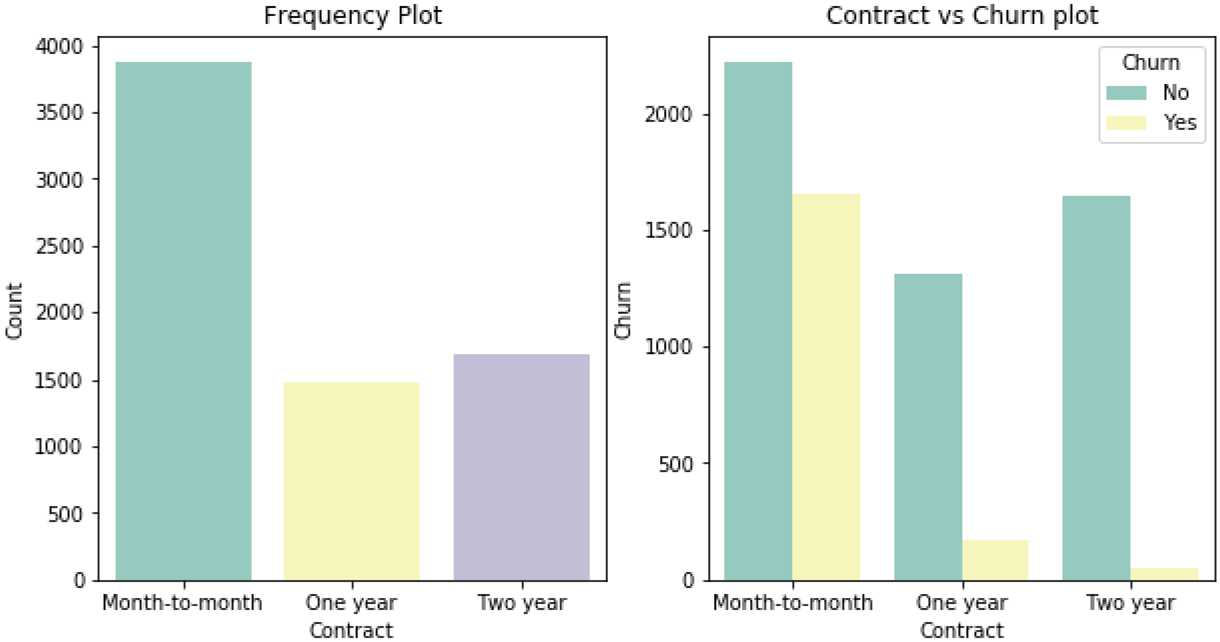
Observations:-

- Churn rate is higher for the subscribers who are not subscribed to Tech support.



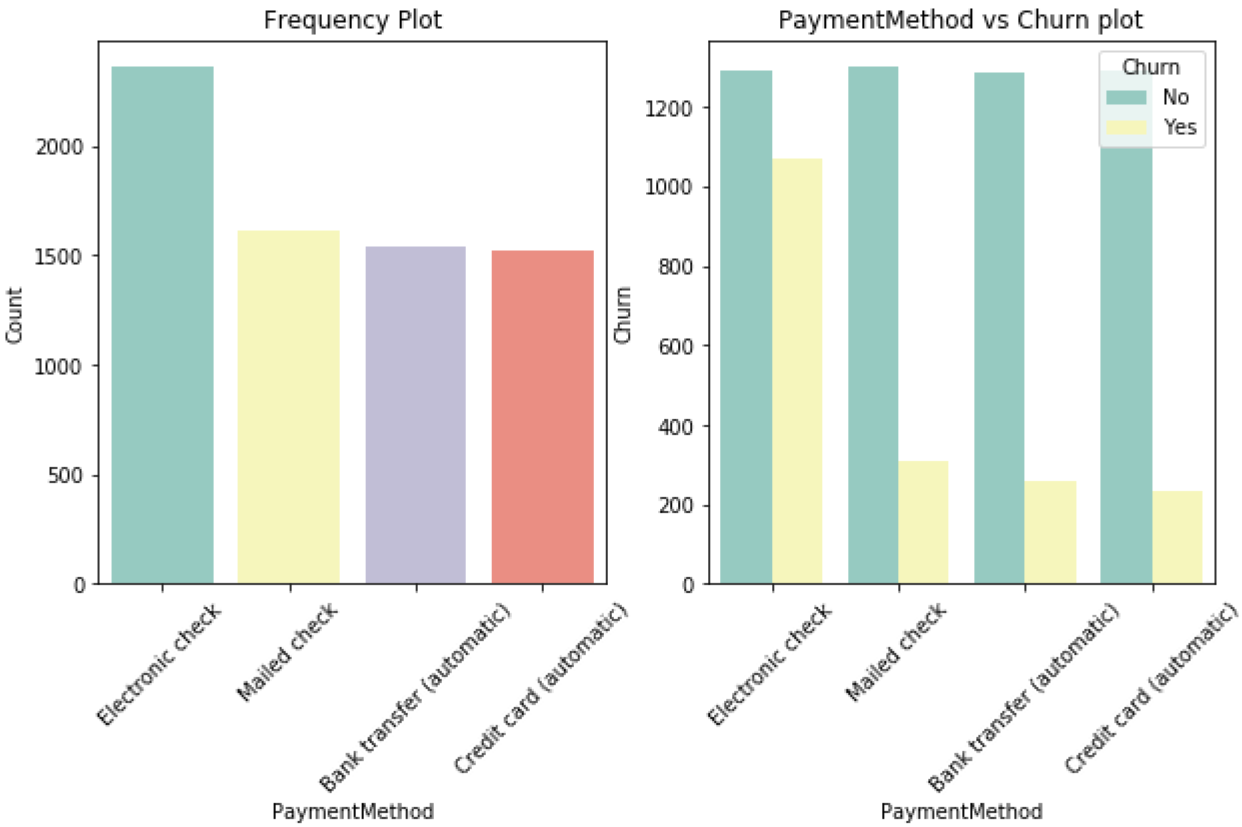
Observations:-

- Churn rate is higher for the subscriber who doesn't have a partner.



Observations:-

- Churn rate is higher for the subscribers having month to month contract.



Observations:-

- Most of the Churn cases are the subscribers who do the payment method by Electronic check.

FINAL OBSERVATIONS:

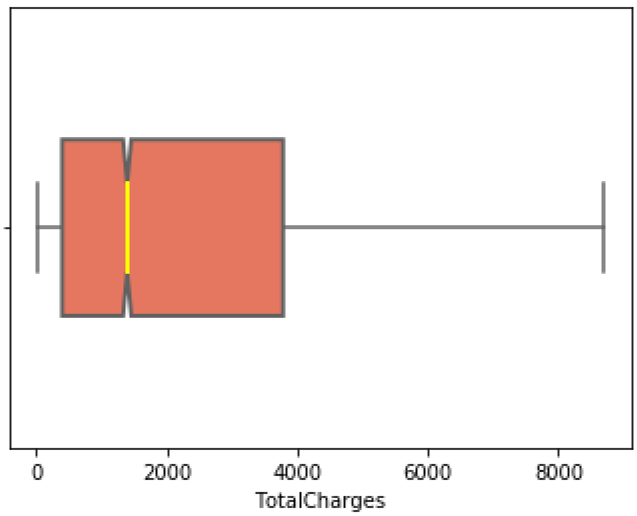
- We cannot see a real Impact of gender
- Seniors have less loyalty
- Partners are more loyal
- Dependents are more loyal
- Customers does not have multiple lines are more loyal
- Customer are not happy with Optical Fiber and Leaving with rate of other internet services
- Customers with month-to-month contract are more willing to leave
- Paperless customers are more willing to leave than paper billing
- Customer pay using electronic check is more willing to leave

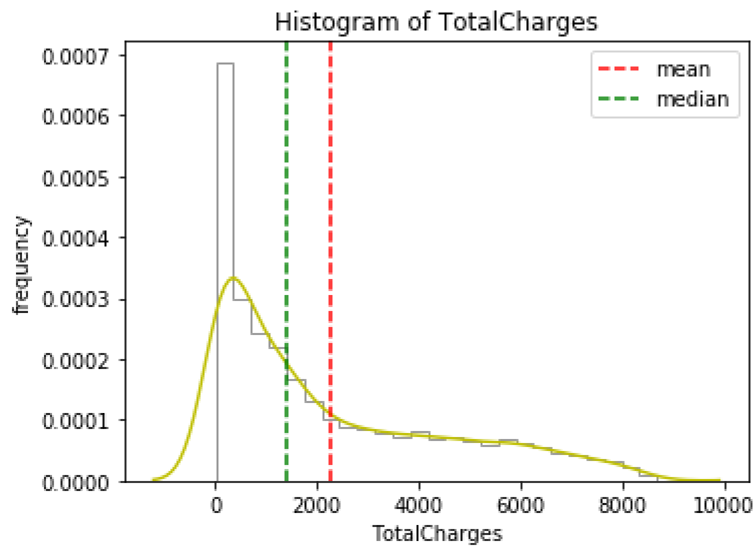
We can conclude that mostly customers are suffering from the services , and specially advanced customers who are using paperless billing and electronic payment. Some variables have no real impact of Churn but as a first trial for the model we will include all variables, should remove variables in the tuning phase.

Data Visualization:

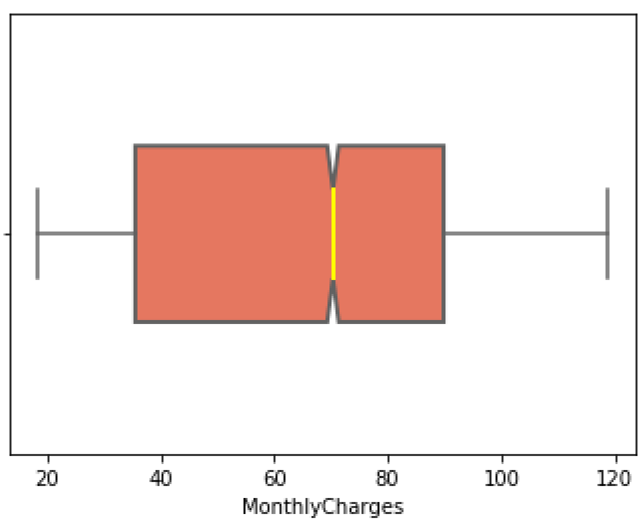
Box plots and Histogram:

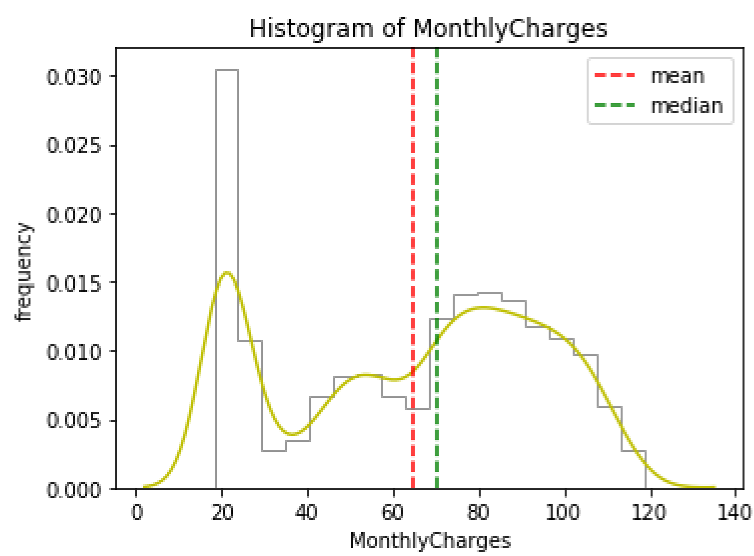
1) This is box plot and histogram of Variable **Total Charges**



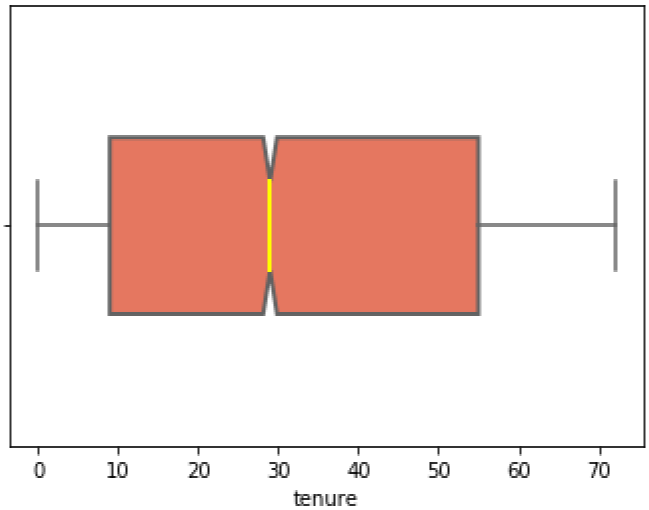


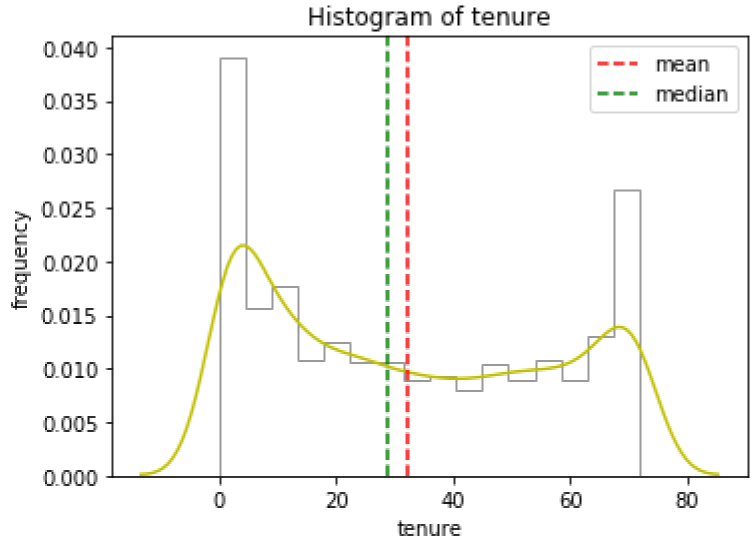
2) This is box plot and histogram of Variable **Monthly Charges**



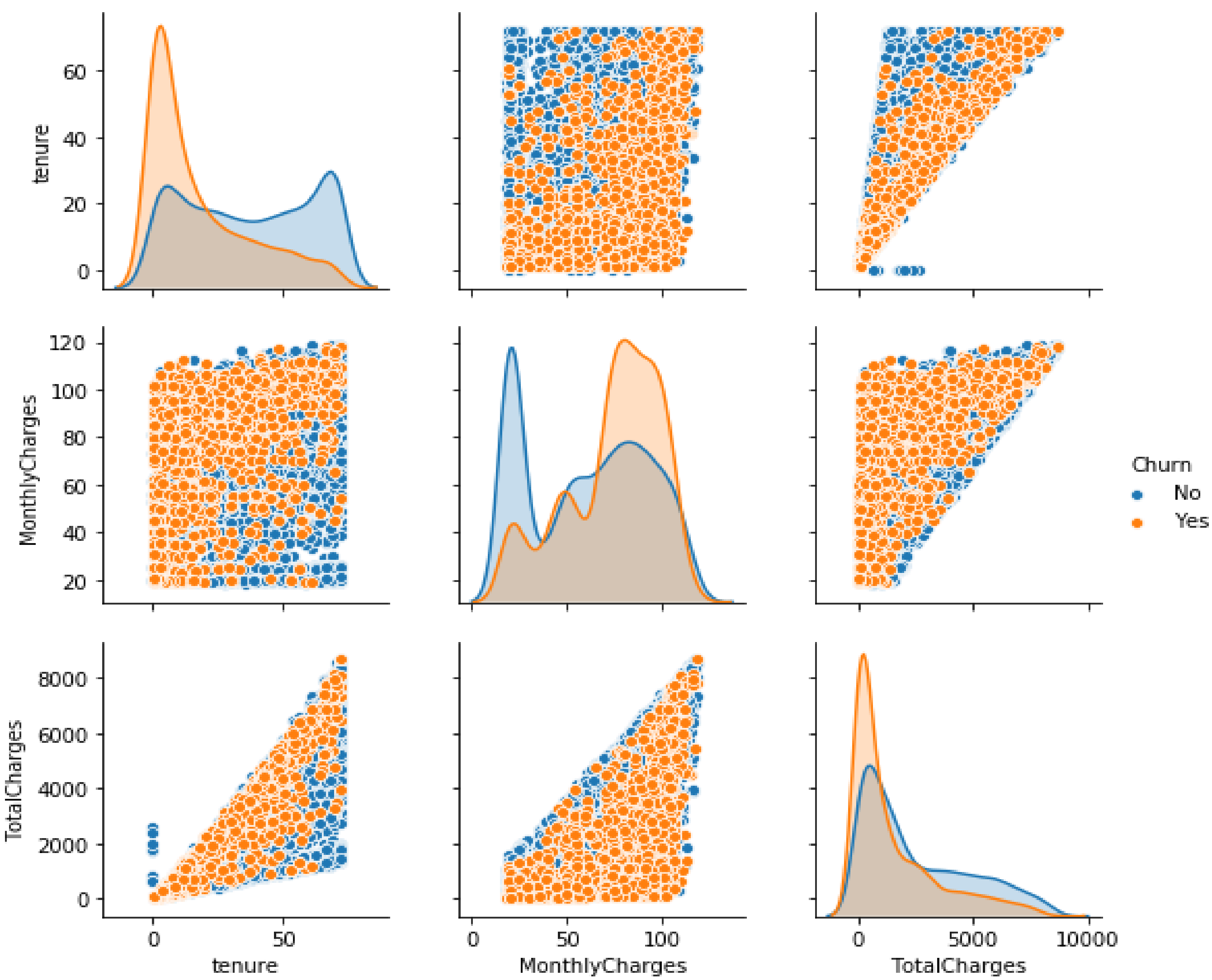


3) This is boxplot and histogram of Variable **Tenure**





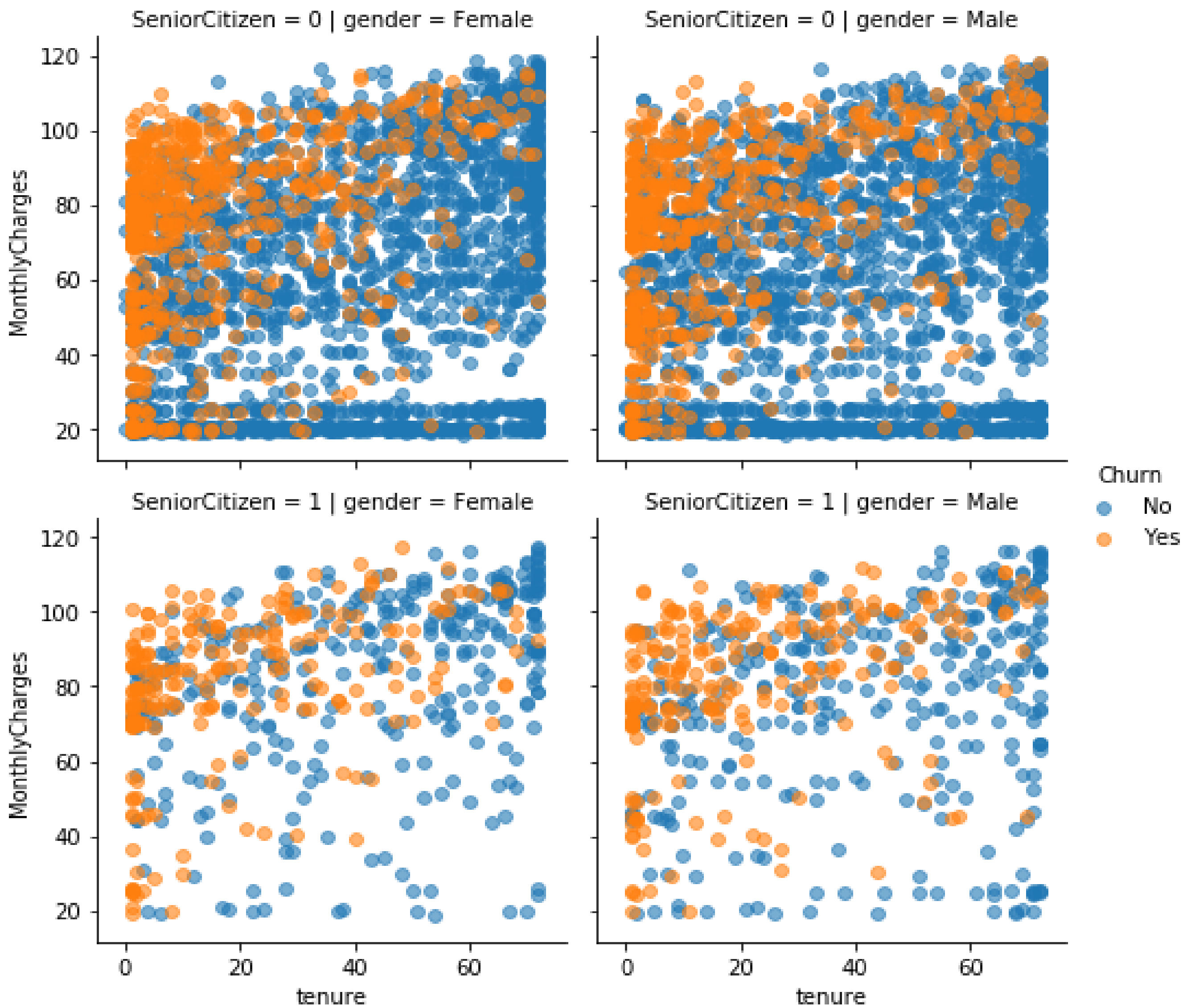
4) Pair plot of Tenure, Monthly charges , Total charges



Observation:

People having lower tenure and higher monthly charges are tend to churn more. Also as you can see above; having month-to-month contract and fiber optic internet have a really huge effect on churn probability.

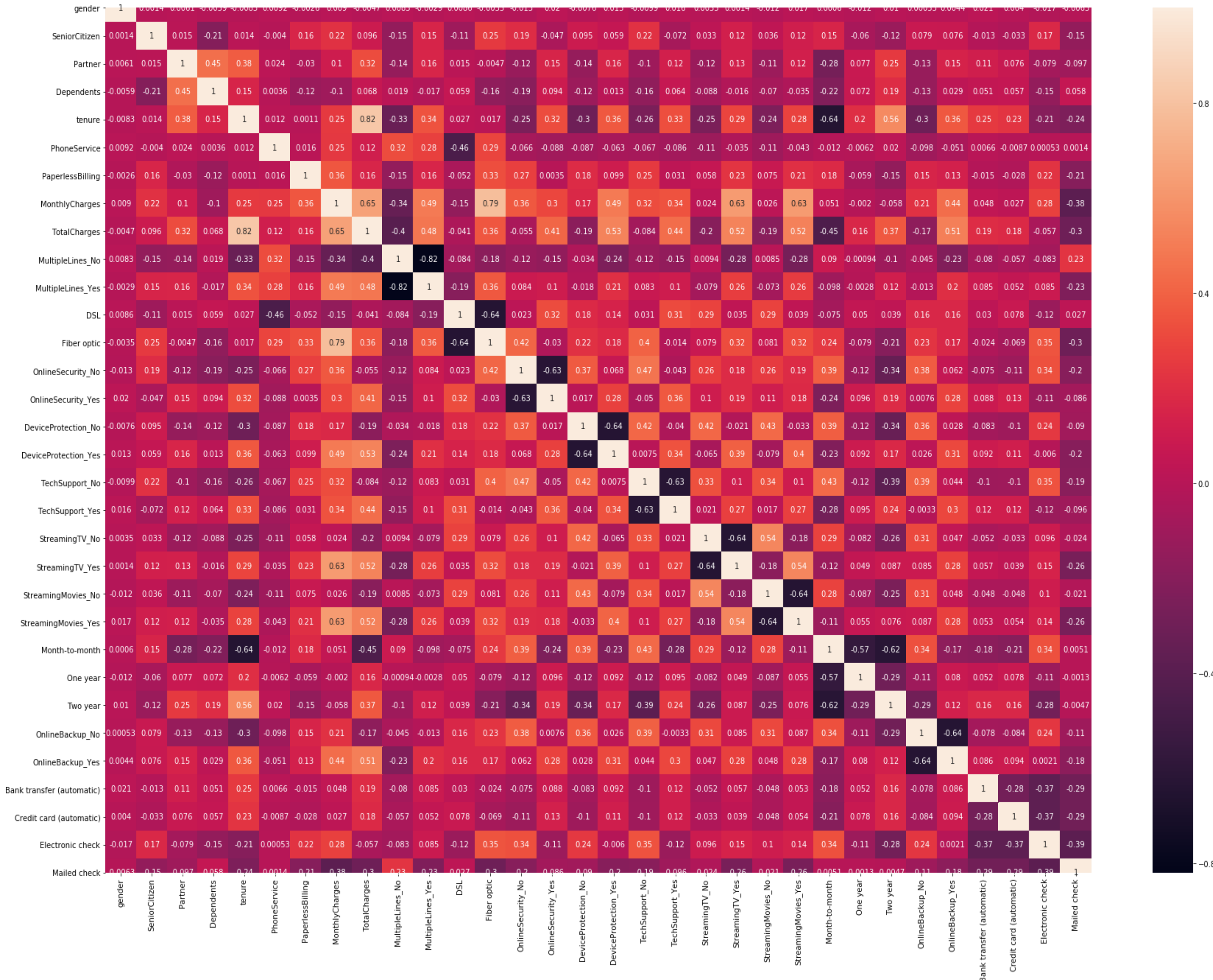
Scatter Plot:



Obsevation:

Gender is not an indicative of churn. Senior Citizens are only 16% of customers, but they have a much higher churn rate: 42% against 23% for non-senior customers. There are no special relations between this categorical values and the main numerical features.

Co-realtion Matrix:



Observation:

From the co-relation matrix, we found out that Mutliplelines_No,Onlinesecurity_No, Deviceprotection_No, Streamingtv_No, Techsupport_No, Streamingmovies_No variables are highly correalated .so we removed them.

Splitting the Dataset into train and test:

- The training dataset and test dataset must be similar, usually have the same predictors orvariables.
- They differ on the observations and specific values in the variables. If we fit the model on thetraining dataset, then we implicitly minimize error or find correct responses. The fitted model provides a good prediction on the training dataset. Then we test the model on the test dataset.
- So, by splitting dataset into training and testing subset, we can efficiently measure our trainedmodel.
- Since it never sees testing data before. Thus it's possible to preventoverfitting.
- We are just splitting dataset into 30% of test data and remaining 70% will used for training themodel.

Model Building:

In statistics, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a *logit*, from ***log**istic un**it***, hence the alternative names. Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a *constant* rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the odds ratio.

According to Correlation Matrix, we find out the ranking of features which have the most important relationship on churning:

- Tenure
- Total charges
- Monthly charges
- Fiber optics
- Month-to-month
- Electronic check
- Two year
- Online backup

Models:

Logit Model

Deviance=4035.8 ; chi2= 5.68e+03 ; Pseudo-R squared=0.2914

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.2510	1.560	-2.085	0.037	-6.308	-0.194
gender	0.0835	0.078	1.069	0.285	-0.070	0.237
SeniorCitizen	0.3137	0.102	3.076	0.002	0.114	0.514
Partner	0.0371	0.093	0.397	0.691	-0.146	0.220
Dependents	-0.2403	0.107	-2.243	0.025	-0.450	-0.030
tenure	-1.3222	0.175	-7.536	0.000	-1.666	-0.978
PhoneService	0.2097	0.777	0.270	0.787	-1.313	1.733
PaperlessBilling	0.3399	0.090	3.768	0.000	0.163	0.517
MonthlyCharges	-1.1702	1.150	-1.018	0.309	-3.424	1.084
TotalCharges	0.5663	0.187	3.023	0.003	0.199	0.934
MultipleLines_Yes	0.4781	0.213	2.246	0.025	0.061	0.895
DSL	1.9151	0.966	1.982	0.047	0.021	3.809
Fiber optic	3.6205	1.909	1.897	0.058	-0.121	7.362
OnlineSecurity_Yes	-0.1229	0.214	-0.573	0.566	-0.543	0.297
DeviceProtection_Yes	0.1546	0.212	0.728	0.467	-0.262	0.571
TechSupport_Yes	-0.3215	0.217	-1.480	0.139	-0.747	0.104
StreamingTV_Yes	0.5843	0.392	1.491	0.136	-0.184	1.352
StreamingMovies_Yes	0.5418	0.392	1.382	0.167	-0.226	1.310
Month-to-month	-0.4446	0.525	-0.846	0.397	-1.474	0.585
One year	-1.1111	0.527	-2.108	0.035	-2.144	-0.078
Two year	-1.6953	0.539	-3.147	0.002	-2.751	-0.639
OnlineBackup_Yes	-0.0662	0.210	-0.316	0.752	-0.477	0.345
Bank transfer (automatic)	-0.8358	0.398	-2.099	0.036	-1.616	-0.055
Credit card (automatic)	-0.9328	0.399	-2.338	0.019	-1.715	-0.151
Electronic check	-0.5636	0.395	-1.428	0.153	-1.337	0.210
Mailed check	-0.9187	0.399	-2.304	0.021	-1.700	-0.137

- This is the result of logit model applied and the second column shows the coefficients value.

Probit Model:

In statistics, a **probit model** is a type of regression where the dependent variable can take only two values, for example married or not married. The word is a portmanteau, coming from ***prob**ability + un**it***. The purpose of the model is to estimate the probability that an observation with particular characteristics will fall into a specific one of the categories; moreover, classifying observations based on their predicted probabilities is a type of binary classification model.

A probit model is a popular specification for a binary response model. As such it treats the same set of problems as does logistic regression using similar techniques. When viewed in the generalized linear model framework, the probit model employs a probit link function. It is most often estimated using the maximum likelihood procedure, such an estimation being called a **probit regression**.

Pseudo-R-squared value=0.2881

Const	-2.0051	1.560	-2.085	0.037	-6.308	-0.194
Gender	0.0425	0.045	0.943	0.346	-0.046	0.131
SeniorCitizen	0.1868	0.060	3.121	0.002	0.069	0.304
Partner	0.0119	0.054	0.220	0.826	-0.094	0.117
Dependents	-0.1404	0.061	-2.313	0.021	-0.259	-0.021
Tenure	-0.5899	0.087	-6.797	0.000	-0.760	-0.420
PhoneService	0.2330	0.447	0.522	0.602	-0.642	1.108
PaperlessBilling	0.1830	0.052	3.547	0.000	0.082	0.284
MonthlyCharges	-0.7650	0.662	-1.156	0.248	-2.062	0.532
TotalCharges	0.1422	0.095	1.500	0.134	-0.044	0.328
MultipleLines_Yes	0.3035	0.123	2.476	0.013	0.063	0.544
DSL	1.1900	0.555	2.145	0.032	0.102	2.277
Fiber optic	2.2811	1.098	2.078	0.038	0.130	4.433
OnlineSecurity_Yes	-0.0452	0.124	-0.366	0.714	-0.287	0.197
DeviceProtection_Yes	0.1104	0.123	0.901	0.368	-0.130	0.351
TechSupport_Yes	-0.1605	0.125	-1.289	0.197	-0.405	0.083
StreamingTV_Yes	0.3728	0.226	1.650	0.099	-0.070	0.816
StreamingMovies_Yes	0.3621	0.226	1.605	0.108	-0.080	0.804
Month-to-month	-0.3441	1.28e+06	-2.69e-07	1.000	-2.5e+06	2.5e+06
One year	-0.7186	1.28e+06	-5.63e-07	1.000	-2.5e+06	2.5e+06
Two year	-0.9423	1.28e+06	-7.38e-07	1.000	-2.5e+06	2.5e+06
OnlineBackup_Yes	-0.0111	0.121	-0.092	0.927	-0.248	0.226
Bank transfer (automatic)	-0.5204	1.51e+06	-3.44e-07	1.000	-2.96e+06	2.96e+06
Credit card (automatic)	-0.5627	1.51e+06	-3.72e-07	1.000	-2.96e+06	2.96e+06
Electronic check	-0.3497	1.51e+06	-2.31e-07	1.000	-2.96e+06	2.96e+06
Mailed check	-0.5723	1.51e+06	-3.78e-07	1.000	-2.96e+06	2.96e+06

- This is the result of probit model applied and the second column shows the coefficients value.

RECURSIVE FEATURE ELIMINATION

In machine learning and statistics, **feature selection**, also known as **variable selection**, **attribute selection** or **variable subset selection**, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for several reasons:

- Simplification of models to make them easier to interpret by researchers/users
- Shorter training times,
- To avoid the curse of dimensionality.
- Enhanced generalization by reducing overfitting (formally, reduction of variance)

The central premise when using a feature selection technique is that the data contains some features that are either *redundant* or *irrelevant,* and can thus be removed without incurring much loss of information. Redundant and irrelevant are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated.

Feature selection techniques should be distinguished from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points). Archetypal cases for the application of feature selection include the analysis of written texts and DNA microarray data, where there are many thousands of features, and a few tens to hundreds of samples.

Now we use RFE to eliminate the features that are not useful in analyzing the churning. RFE gives the value ‘TRUE’ to variables that we should keep and ‘FALSE’ to variables that we should drop.

columns	**Rank**	**support**	
0	Const	1	True
21	OnlineBackup_Yes	1	True
20	Two year	1	True
18	Month-to-month	1	True
15	TechSupport_Yes	1	True
13	OnlineSecurity_Yes	1	True
11	DSL	1	True
9	TotalCharges	1	True
12	Fiber optic	1	True
7	PaperlessBilling	1	True
2	SeniorCitizen	1	True
8	MonthlyCharges	1	True
5	Tenure	1	True
25	Mailed check	1	True
6	PhoneService	1	True
23	Credit card (automatic)	2	False
22	Bank transfer (automatic)	3	False
4	Dependents	4	False
19	One year	5	False
10	MultipleLines_Yes	6	False
14	DeviceProtection_Yes	7	False
24	Electronic check	8	False
1	Gender	9	False
3	Partner	10	False
17	StreamingMovies_Yes	11	False
16	StreamingTV_Yes	12	False

Model 1 :(Logistic Regression)

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.5498	0.361	-4.293	0.000	-2.257	-0.842
SeniorCitizen	0.3773	0.100	3.792	0.000	0.182	0.572
tenure	-1.3554	0.174	-7.768	0.000	-1.697	-1.013
PhoneService	-0.8904	0.177	-5.043	0.000	-1.236	-0.544
PaperlessBilling	0.3734	0.090	4.169	0.000	0.198	0.549
MonthlyCharges	0.5356	0.159	3.375	0.001	0.225	0.847
TotalCharges	0.5916	0.187	3.160	0.002	0.225	0.959
DSL	0.5257	0.226	2.329	0.020	0.083	0.968
Fiber optic	0.8956	0.331	2.706	0.007	0.247	1.544
OnlineSecurity_Yes	-0.4300	0.102	-4.214	0.000	-0.630	-0.230
TechSupport_Yes	-0.6408	0.107	-5.973	0.000	-0.851	-0.430
Month-to-month	0.7471	0.128	5.847	0.000	0.497	0.998
Two year	-0.6051	0.205	-2.958	0.003	-1.006	-0.204
OnlineBackup_Yes	-0.3527	0.095	-3.721	0.000	-0.538	-0.167
Mailed check	-0.2256	0.109	-2.065	0.069	-0.440	-0.012

Pseudo R-squared : 0.2866

The Pseudo R- Squared value of this model is 02866.

Here the p-value of variable “Mailed check” is higher than the significance level so we drop the feature in the next model.

Model 2 :(Logistic Regression)

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.6013	0.360	-4.446	0.000	-2.307	-0.895
SeniorCitizen	0.3841	0.099	3.864	0.000	0.189	0.579
tenure	-1.3012	0.172	-7.555	0.000	-1.639	-0.964
PhoneService	-0.9174	0.176	-5.219	0.000	-1.262	-0.573
PaperlessBilling	0.3856	0.089	4.316	0.000	0.210	0.561
MonthlyCharges	0.5776	0.157	3.670	0.000	0.269	0.886
TotalCharges	0.5474	0.186	2.945	0.003	0.183	0.912
DSL	0.5476	0.225	2.430	0.015	0.106	0.989
Fiber optic	0.9231	0.331	2.791	0.005	0.275	1.571
OnlineSecurity_Yes	-0.4409	0.102	-4.329	0.000	-0.641	-0.241
TechSupport_Yes	-0.6516	0.107	-6.082	0.000	-0.862	-0.442
Month-to-month	0.7560	0.128	5.922	0.000	0.506	1.006
Two year	-0.6062	0.204	-2.966	0.003	-1.007	-0.206
OnlineBackup_Yes	-0.3543	0.095	-3.742	0.000	-0.540	-0.169

Pseudo R-squared:0.2858

Observations:-

- All variables are significant as all the features have very small p-value

  Therefore, we checked VIF of the variables to drop the features one by one.

Multi-collinearity check:-

	Features	vif
0	const	89.90
8	Month-to-month	19.70
5	MonthlyCharges	18.91
6	TotalCharges	10.43
7	DSL	7.63
2	tenure	6.71
11	Fiber optic	2.35
3	PhoneService	1.90
12	Two year	1.85
10	TechSupport_Yes	1.64
13	OnlineBackup_Yes	1.52
9	OnlineSecurity_Yes	1.47
4	PaperlessBilling	1.21
1	SeniorCitizen	1.12

- Dropping attribute “Month-to-month” in the next model as it is highly co-related due to high VIF value.

<u>Model 3 :(Logistic Regression)</u>

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.3329	0.354	-3.764	0.000	-2.027	-0.639
SeniorCitizen	0.4454	0.099	4.507	0.000	0.252	0.639
tenure	-1.4507	0.170	-8.550	0.000	-1.783	-1.118
PhoneService	-0.8796	0.175	-5.036	0.000	-1.222	-0.537
PaperlessBilling	0.4246	0.089	4.790	0.000	0.251	0.598
MonthlyCharges	0.4953	0.156	3.175	0.001	0.190	0.801
TotalCharges	0.5472	0.184	2.970	0.003	0.186	0.908
DSL	0.7241	0.223	3.248	0.001	0.287	1.161
Fiber optic	1.2414	0.326	3.811	0.000	0.603	1.880
OnlineSecurity_Yes	-0.4651	0.101	-4.604	0.000	-0.663	-0.267
TechSupport_Yes	-0.6942	0.106	-6.561	0.000	-0.902	-0.487
Two year	-0.9771	0.191	-5.123	0.000	-1.351	-0.603
OnlineBackup_Yes	-0.3513	0.094	-3.736	0.000	-0.536	-0.167

Pseudo R-squared:0.279.

	Features	vif
0	const	89.24
8	TotalCharges	19.16
5	MonthlyCharges	18.84
6	Fiber optic	10.42
7	DSL	7.45
2	tenure	6.33
3	PhoneService	1.90
11	Two year	1.69
10	TechSupport_Yes	1.62
12	OnlineBackup_Yes	1.52
9	OnlineSecurity_Yes	1.47
4	PaperlessBilling	1.20
1	SeniorCitizen	1.11

- Dropping variable “TotalCharges” in the next model as it is highly correlated.

Model 4 :(Logistic Regression)

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.1167	0.345	-3.235	0.001	-1.793	-0.440
SeniorCitizen	0.4507	0.099	4.543	0.000	0.256	0.645
tenure	-0.9889	0.061	-16.257	0.000	-1.108	-0.870
PhoneService	-0.9000	0.172	-5.225	0.000	-1.238	-0.562
PaperlessBilling	0.4186	0.088	4.731	0.000	0.245	0.592
MonthlyCharges	0.6944	0.141	4.916	0.000	0.418	0.971
DSL	0.5708	0.217	2.633	0.058	0.146	0.996
Fiber optic	1.0448	0.319	3.278	0.001	0.420	1.669
OnlineSecurity_Yes	-0.4597	0.101	-4.545	0.000	-0.658	-0.261
TechSupport_Yes	-0.6901	0.106	-6.514	0.000	-0.898	-0.482
Two year	-0.9428	0.189	-4.985	0.000	-1.314	-0.572
OnlineBackup_Yes	-0.3354	0.094	-3.570	0.000	-0.519	-0.151

Pseudo R-squared:0.277

- Dropping variable “DSL” as it is insignificant in the next model as its p-value is greater than 0.05.

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.3457	0.184	-1.874	0.061	-0.707	0.016
SeniorCitizen	0.4529	0.099	4.557	0.000	0.258	0.648
tenure	-1.0386	0.058	-17.809	0.000	-1.153	-0.924
PhoneService	-1.1646	0.141	-8.246	0.000	-1.441	-0.888
PaperlessBilling	0.4335	0.088	4.912	0.000	0.261	0.607
MonthlyCharges	0.9529	0.103	9.287	0.000	0.752	1.154
Fiber optic	0.3205	0.163	1.969	0.079	0.002	0.640
OnlineSecurity_Yes	-0.4595	0.102	-4.512	0.000	-0.659	-0.260
TechSupport_Yes	-0.7186	0.106	-6.758	0.000	-0.927	-0.510
Two year	-0.9925	0.188	-5.280	0.000	-1.361	-0.624
OnlineBackup_Yes	-0.3574	0.094	-3.791	0.000	-0.542	-0.173

Pseudo R-squared: 0.2765.

- Dropping variable “Fiber optic” as it is insignificant in terms of p-value as well as VIF.

Model 6 :(Logistic Regression)

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.1725	0.162	-1.063	0.288	-0.491	0.146
SeniorCitizen	0.4621	0.099	4.659	0.000	0.268	0.657
tenure	-1.0519	0.058	-18.144	0.000	-1.166	-0.938
PhoneService	-1.1513	0.141	-8.139	0.000	-1.429	-0.874
PaperlessBilling	0.4336	0.088	4.920	0.000	0.261	0.606
MonthlyCharges	1.1187	0.060	18.756	0.000	1.002	1.236
OnlineSecurity_Yes	-0.5114	0.098	-5.209	0.000	-0.704	-0.319
TechSupport_Yes	-0.7845	0.101	-7.780	0.000	-0.982	-0.587
Two year	-1.0009	0.188	-5.322	0.000	-1.370	-0.632
OnlineBackup_Yes	-0.3930	0.092	-4.258	0.000	-0.574	-0.212

Final Model :

Prob(Y=churn)=F[-0.1725+ (0.4621)*SeniorCitizen+ (-1.0519)*tenure+ (-1.1513)*PhoneService + (0.4336)*PaperlessBilling+ (1.1187)*MonthlyCharges+ (-0.5114)*OnlineSecurity_Yes+ (-0.7845)*TechSupport_Yes+ (-1.0009)*Two Year+ (-0.3930)*OnlineBackup_Yes]

[F(.)=sigmoid function]

Pseudo R-squared of final Model: 0.2758

-Here all the features are significant so we checked vif value.

	Features	vif
0	const	19.15
5	MonthlyCharges	1.97
2	tenure	1.80
8	Two year	1.63
9	OnlineBackup_Yes	1.42
7	TechSupport_Yes	1.39
6	OnlineSecurity_Yes	1.30
4	PaperlessBilling	1.20
3	PhoneService	1.19
1	SeniorCitizen	1.10

Here all the vif values are less than 5 so this is the final model.

McFadden's Pseudo R-squared:

McFadden Pseudo R2 helps in determining the overall fitness of the model. According to set standards, a score in the range of 0.2 to 0.4 denotes a perfect model fit.

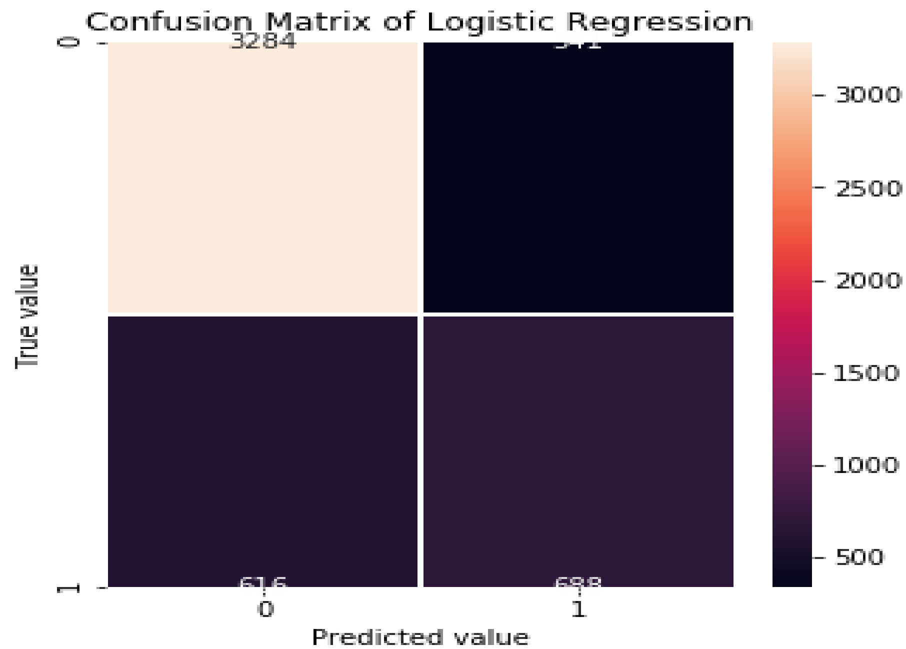
Our model had a score of **0.2758**.

Confusion Matrix:

As our data is imbalanced. So it is important to look at the confusion matrix according to these two algorithms. With imbalanced datasets, the highest accuracy does not give the best model. Assume we have 1000 total rows, 10 rows are churn and 990 rows are non-churn. If we find all these 10 churn rows as non-churn, then the accuracy will be still 99%.

Confusion matrix gives us FN(false negative), FP(false positive), TN(true negative) and TP(true positive) values.

	precision	recall	f1-score	support
0	0.84	0.91	0.87	3625
1	0.67	0.53	0.59	1304
accuracy			0.81	4929
macro avg	0.76	0.72	0.73	4929
weighted avg	0.80	0.81	0.80	4929



Observations:

1) Accuracy of the model : 0.8058429701765064
2) Recall: 0.5276073619631901
3) Precision: 0.6686103012633625
4) Sensitivity (True Positive Rate) =TP / TP + FN : 0.5276073619631901
5) Specificity= TN / (TN + FP) : 0.9059310344827586
6) False positive rate= FP / (TN + FP) : 0.09406896551724138

ROC Curve:

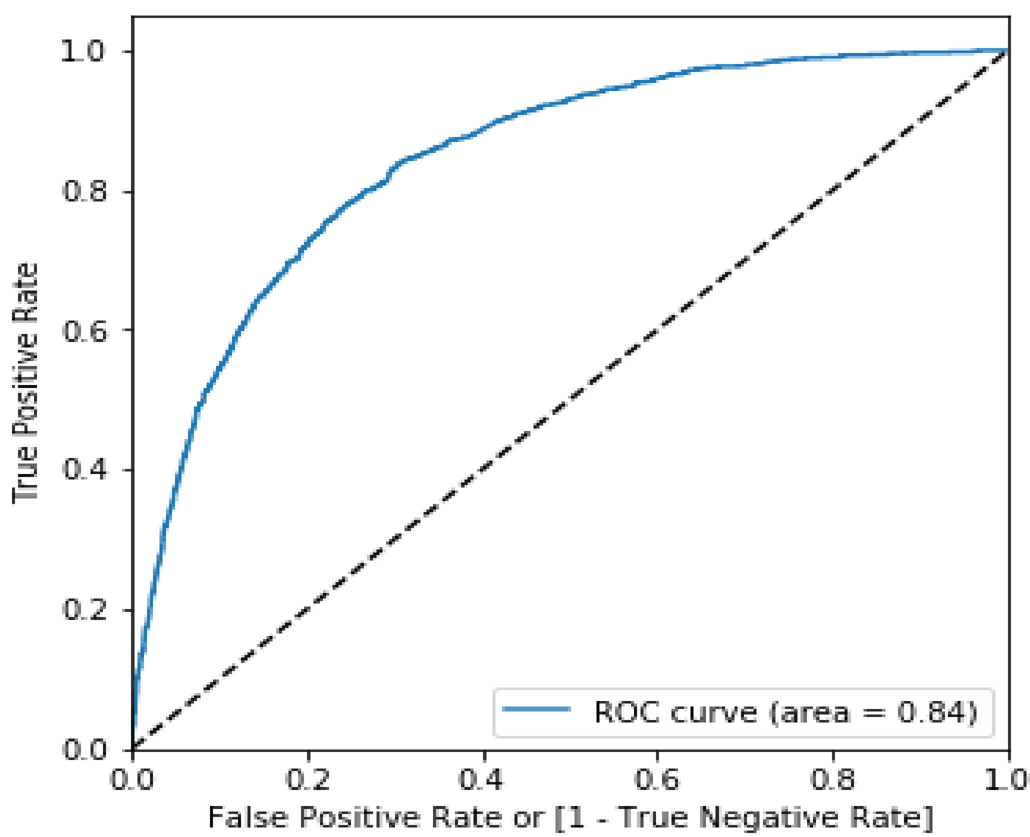
A **receiver operating characteristic curve**, or **ROC curve**, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or *probability of detection* in machine learning. The false-positive rate is also known as *probability of false alarm* and can be calculated as (1 − specificity). It can also be thought of as a plot of the power as a function of the Type I Error of the decision rule (when the performance is calculated from just a sample of the population, it can be thought of as estimators of these quantities). The ROC curve is thus the sensitivity or recall as a function of fall-out. In general, if the probability distributions for both detection and false alarm are known, the ROC curve can be generated by plotting the cumulative distribution function (area under the probability distribution from - to the discrimination threshold) of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability on the x-axis.

ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making.

The ROC curve was first developed by electrical engineers and radar engineers during World War II for detecting enemy objects in battlefields and was soon introduced to psychology to account for perceptual detection of stimuli. ROC analysis since then has been used
in medicine, radiology, biometrics, forecasting of natural hazards, meteorology, model performance assessment, and other areas for many decades and is increasingly used in machine learning and data mining research.

The ROC is also known as a relative operating characteristic curve, because it is a comparison of two operating characteristics (TPR and FPR) as the criterion changes.



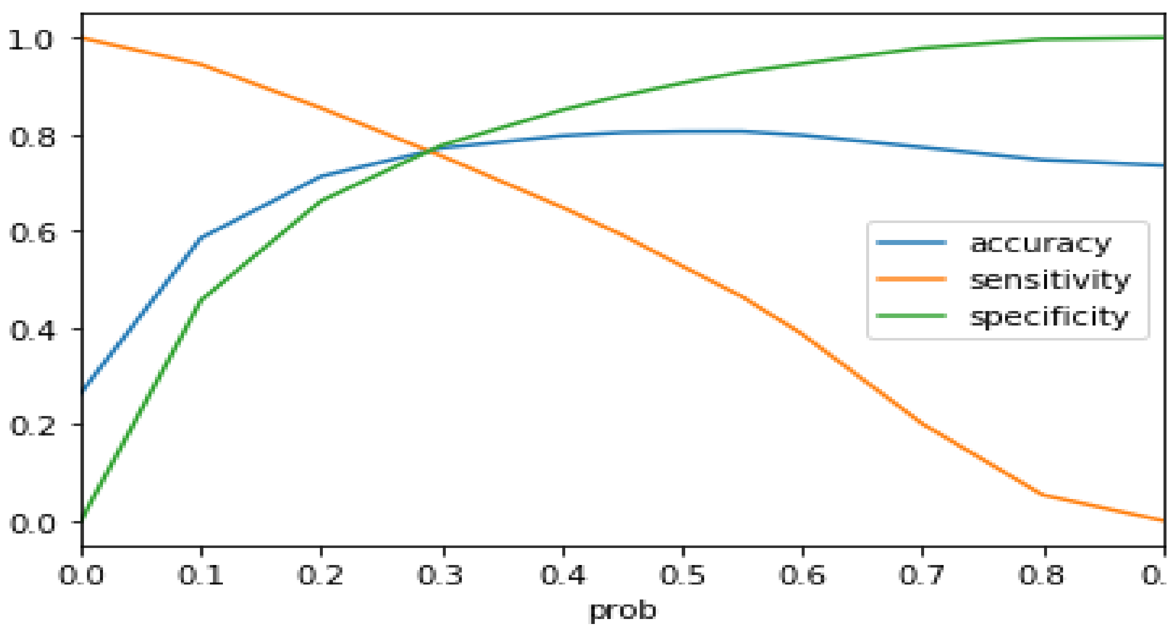
<u>Optimal cut-off point:</u>

-finding optimal cutoff point
-Optimal cutoff probability is that probability where we get balanced sensitivity and specificity

- It is obtained by plotting accuracy ,sensitivity, specificity with probability.

prob	accuracy	sensitivity	specificity

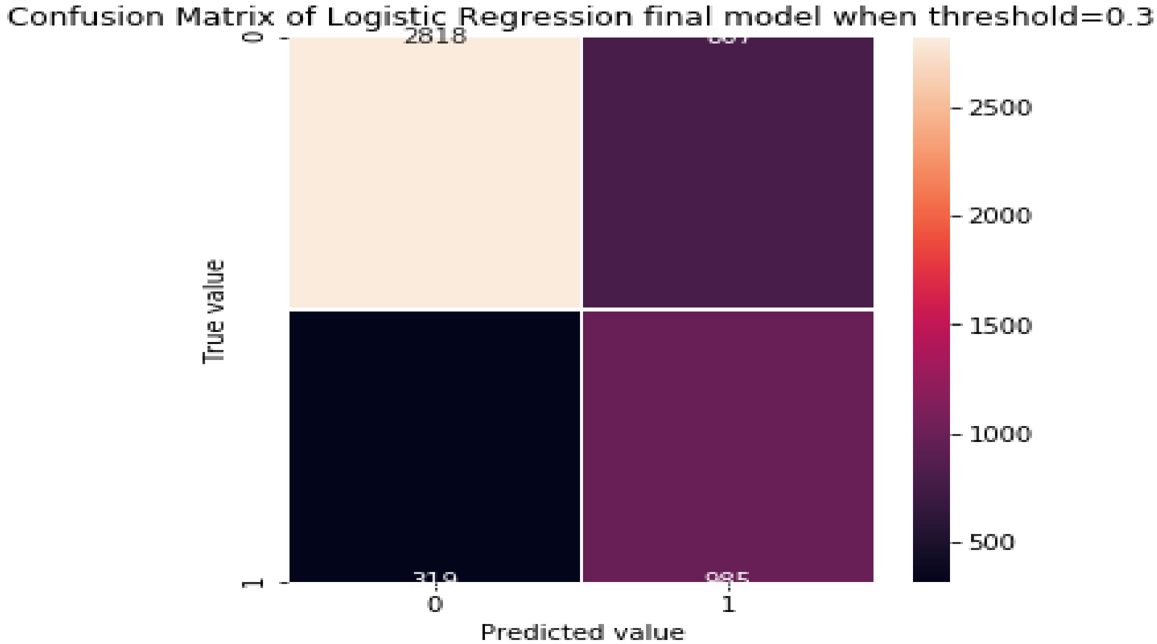
prob	accuracy	sensitivity	specificity	
0.00	0.00	0.264557	1.000000	0.000000
0.10	0.10	0.585514	0.944785	0.456276
0.20	0.20	0.713329	0.854294	0.662621
0.30	0.30	0.771556	0.755368	0.777379
0.40	0.40	0.797119	0.649540	0.850207
0.45	0.45	0.803814	0.592025	0.880000
0.50	0.50	0.805843	0.527607	0.905931
0.55	0.55	0.806046	0.463957	0.929103
0.60	0.60	0.798336	0.386503	0.946483
0.70	0.70	0.772976	0.200920	0.978759
0.80	0.80	0.747210	0.053681	0.996690
0.90	0.90	0.735646	0.000767	1.000000



From the curve above, 0.3 is the optimum point to take it as a cutoff probability.

<u>Now calculating accuracy and other parameters by taking cut_off point as 0.3:</u>

Final Accuracy : 0.7715560965713126
Confusion matrix
[2818 807]
[319 985]]



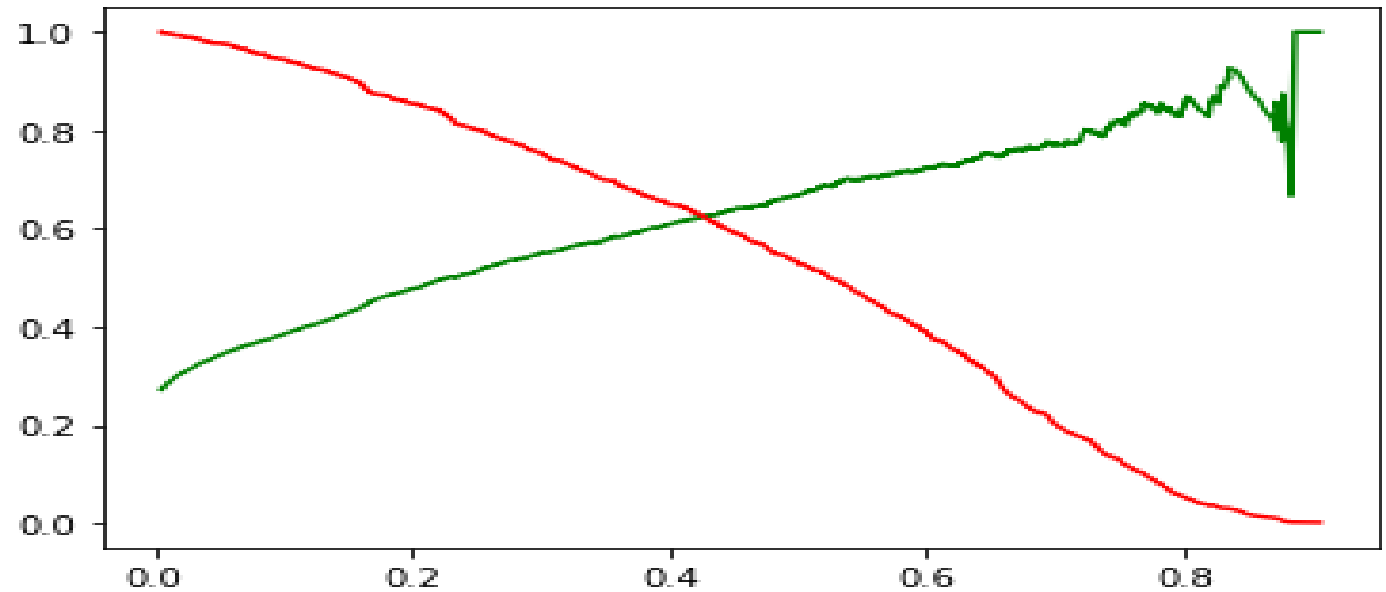
	precision	recall	f1-score	support
0	0.90	0.78	0.83	3625
1	0.55	0.76	0.64	1304
accuracy			0.77	4929
macro avg	0.72	0.77	0.73	4929
weighted avg	0.81	0.77	0.78	4929

-optimal threshold : 0.3
-sensitivity : 0.7553680981595092
-specificity : 0.7773793103448275
-false_positive_rate : 0.22262068965517245
-positive_predictive_rate : 0.5496651785714286
-negative_predictive_rate : 0.8983104877271278

Calculating Precision and Recall trade off:

Precision-recall tradeoff occur due to increasing one of the parameter(precision or recall) while keeping the model same.

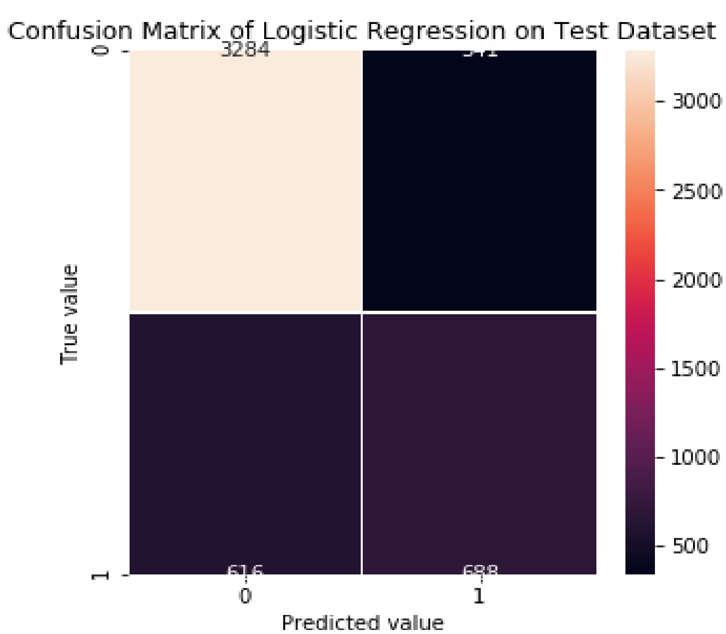
In an ideal scenario where there is a perfectly separable data, both precision and recall can get maximum value of 1.0. But in most of the practical situations, there is noise in the dataset and the dataset is not perfectly separable. There might be some points of positive class closer to the negative class and vice versa. In such cases, shifting the decision boundary can either increase the precision or recall but not both. Increasing one parameter leads to decreasing of the other. In other words, binary classifier will miss classify some points always. Miss classification means classifying data point from negative class as positive and from positive class as negative. This miss rate is either compromising precision or recall score.

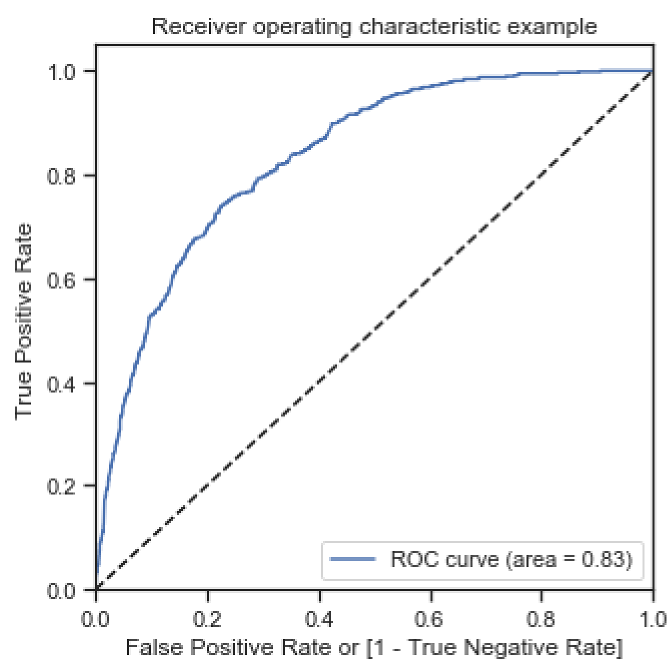


Prediction on test data set:

-Test Sensitivity : 0.49911504424778763
-Test Specificity : 0.8921188630490956

	precision	recall	f1-score	support
0	0.83	0.89	0.86	1548
1	0.63	0.50	0.56	565
accuracy			0.79	2113
macro avg	0.73	0.70	0.71	2113
weighted avg	0.78	0.79	0.78	2113



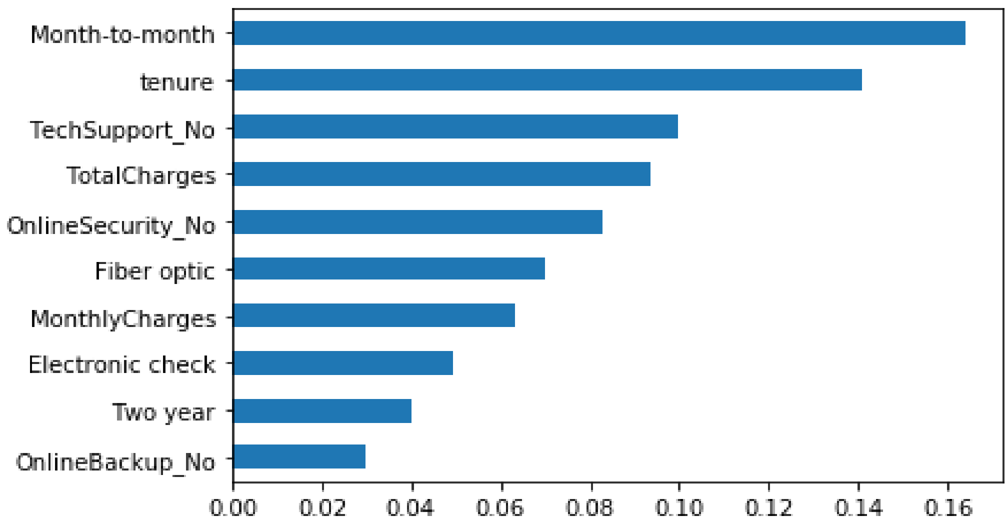


Random Forest:

We also applied Random Forest to check accuracy and F1-score of our model .

	precision	recall	f1-score	support
0	0.83	0.90	0.86	1548
1	0.64	0.50	0.56	565
accuracy			0.79	2113
macro avg	0.74	0.70	0.71	2113
weighted avg	0.78	0.79	0.78	2113

Feature Importance in descending order according to Random Forest:



Observations:

From random forest algorithm, monthly contract, tenure, TechSupport_No and total charges are the most important predictor variables to predict churn. The results from random forest are very similar to that of the logistic regression and in line to what we had expected from our EDA.

Omitted Variable Bias:

In statistics, **omitted-variable bias** (**OVB**) occurs when a statistical model leaves out one or more relevant variables. The bias results in the model attributing the effect of the missing variables to those that were included.

More specifically, OVB is the bias that appears in the estimates of parameters in a regression analysis, when the assumed specification is incorrect in that it omits an independent variable that is a determinant of the dependent variable and correlated with one or more of the included independent variables.

The Gauss–Markov theorem states that regression models which fulfill the classical linear regression model assumptions provide the most efficient, linear and unbiased estimators. In ordinary least squares, the relevant assumption of the classical linear regression model is that the error term is uncorrelated with the regressors.

The presence of omitted-variable bias violates this particular assumption. The violation causes the OLS estimator to be biased and inconsistent. The direction of the bias depends on the estimators as well as the covariance between the regressors and the omitted variables. A positive covariance of the omitted variable with both a regressor and the dependent variable will lead the OLS estimate of the included regressor's coefficient to be greater than the true value of that coefficient.

We have observed in our dataset that there can be some omitted variables -

- Income of customer
- Age
- Postpaid/Prepaid connection

CONCLUSIONS:

Final Model from Logistic regression:

Prob(Y=churn)=F[-0.1725+ (0.4621)*SeniorCitizen+ (-1.0519)*tenure+ (-1.1513)*PhoneService + (0.4336)*PaperlessBilling+ (1.1187)*MonthlyCharges+ (-0.5114)*OnlineSecurity_Yes+ (-0.7845)*TechSupport_Yes+ (-1.0009)*Two Year+ (-0.3930)*OnlineBackup_Yes]

[F(.)=sigmoid function]

- Overall accuracy on test dataset is 79% and AUC of ROC is 0.83 using Logistic Regression.
- Since data set is imbalanced, we preferred to use F1 score rather than accuracy.
- Logistic Regression and Random Forest provide the same F1 Score, so both are the best model.
- Gender has no impact on churn.
- People having month-to-month contract tend to churn more than people having long term contracts.
- As the tenure increases, the probability of churn decreases.
- As the monthly charges increases, the probability of churn increases.

Python Codes:

```
importnumpyasnp# linear algebra
importpandasaspd# data processing, CSV file I/O (e.g. pd.read_csv)

importos for dirname, _, filenames in os.walk('/kaggle/input'):
for filename in filenames:
print(os.path.join(dirname, filename))
```

```
pd.set_option('display.max_rows', 500) pd.set_option('display.max_columns', 500)
importwarnings warnings.filterwarnings('ignore')
```

Step 1: **Read data from Sources:**

```
# import customer data
customer_df = pd.read_csv('C:/Users/user/Desktop/SMBA PROJECT2/customer_data.csv')
```

```
# import churn_data
churn_df = pd.read_csv('C:/Users/user/Desktop/SMBA PROJECT2/churn_data.csv')
```

```
# import internet data
internet_df = pd.read_csv('C:/Users/user/Desktop/SMBA PROJECT2/internet_data.csv')
```

Lets explore the data

```
customer_df.head()
customer_df.shape
```

```
churn_df.head()
internet_df.head()
```

```
# join all the columns by customer ID
print(len(np.setdiff1d(customer_df.customerID, internet_df.customerID)))telecom_df = pd.merge(internet_df, df1, ho
w='inner', on='customerID')
#merge customer and churn dataframes into df1
df1 = pd.merge(customer_df, churn_df, how='inner', on='customerID')
```

```
# merge df1 and internet dataframes to telecom df
telecom_df = pd.merge(internet_df, df1, how='inner', on='customerID')
```

```
# explore final telecom dftelecom_df.head()
```

```
telecom_df.columns
```

```
Index(['customerID', 'MultipleLines', 'InternetService', 'OnlineSecurity',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
       dtype='object')
```

```
# check the data types
telecom_df.info()
TotalCharges is an object and not float!!!
```

```
# We don´t have null values but from error we can see that column 'TotalCharges' contains whitespace = ' '
# telecom_df['TotalCharges'] = pd.to_numeric(telecom_df['TotalCharges'])
# How many whitespace = ' ' we have in column 'TotalCharges'
telecom_df['TotalCharges'].str.isspace().value_counts
```

```
telecom_df['TotalCharges'].isnull().sum()
# Replacing whitespace to NAN values and converting to numeric data (float)
telecom_df['TotalCharges'] = telecom_df['TotalCharges'].replace(' ', np.nan)
telecom_df['TotalCharges'] = pd.to_numeric(telecom_df['TotalCharges'])
```

```
# How many NAN values is in column
telecom_df['TotalCharges'].isnull().sum()
```

```
# Replacing NAN values with mean value from all data in column 'TotalCharges'
#new_value = telecom_df['TotalCharges'].astype('float').mean(axis=0)
new_value = (telecom_df['TotalCharges']/telecom_df['MonthlyCharges']).mean()*telecom_df['MonthlyCharges']
telecom_df['TotalCharges'].replace(np.nan, new_value, inplace=True)
```

```
# How many NAN values is in column 'TotalCharges' after replacing NAN with mean
telecom_df['TotalCharges'].isnull().sum()
# Checking for null valuestelecom_df.isnull().sum()
```

```
telecom_df.TotalCharges.dtype
```

```
# analyze customerID
telecom_df.customerID.nunique()
# analyze MultipleLinestelecom_df.MultipleLines.value_counts()
```

```
# analyze OnlineSecurity
telecom_df.OnlineSecurity.value_counts()
# analyze InternetService
telecom_df.InternetService.value_counts()
```

```
#analyze DeviceProtection
telecom_df.DeviceProtection.value_counts()
```

```
# analye TechSupport
```

```
telecom_df.TechSupport.value_counts()
# analye StreamingTV
telecom_df.StreamingTV.value_counts()
```

```
# analyze StreamingMovies
telecom_df.StreamingMovies.value_counts()
 analyze gender
telecom_df.gender.value_counts()
# analyze SeniorCitizen
telecom_df.SeniorCitizen.value_counts()
# analyze partner
telecom_df.Partner.value_counts()
# analyze dependents
telecom_df.Dependents.value_counts()
# analyze tenure
np.sort(telecom_df.tenure.unique())
 analyze phone services
telecom_df.PhoneService.value_counts()
telecom_df.Contract.value_counts()
```

```
# analyze paperless billing
telecom_df.PaperlessBilling.value_counts()
# analyze paymentmethod
telecom_df.PaymentMethod.value_counts()
telecom_df.Churn.value_counts()
sns.countplot(x="Churn",data=telecom_df)
```

```
# import required visual libraries
importmatplotlib.pyplotaspltimportseabornassns%matplotlib inline
```

```
defcategory_plot(df_src, df_by, h_v='h'):
    frequency_table(df_src)
    fig, ax = plt.subplots(1,2,figsize=(10,5))
    ax[1] = sns.countplot(x=df_src, hue=df_by, ax=ax[1], palette="Set3")
    ax[1].set(xlabel=df_src.name, ylabel=df_by.name, title = df_src.name +' vs '+ df_by.name +' plot')
    values = df_src.value_counts(normalize=True)*100
    ax[0] = sns.countplot(x=df_src, palette='Set3', ax=ax[0])
    ax[0].set(xlabel=df_src.name, ylabel ='Count', title='Frequency Plot')
if(h_v =='v'):
        ax[0].set_xticklabels(ax[0].get_xticklabels(), rotation=45)
        ax[1].set_xticklabels(ax[1].get_xticklabels(), rotation=45)

    plt.show()
```

```
defget_percent(value, total, round_number =2):
```

```
return round(100* value / total, round_number)
```

```
def frequency_table(df,with_percent=True, with_margins=False):
    freq_df = pd.crosstab(index=df, columns="count", margins=with_margins).reset_index()
if with_percent:
        freq_df['percent(%)'] = get_percent(freq_df['count'] , df.shape[0])
print(freq_df)
```

perform univariant analysis to understand the churn

```
categorial_columns = telecom_df.select_dtypes(['object']).columnscategorial_columns
```

```
# univariant analyis on MultipleLines
category_plot(telecom_df.MultipleLines, telecom_df.Churn)
# univariant analyis on InternetServices
category_plot(telecom_df.InternetService, telecom_df.Churn)
```

```
# univariant analyis on OnlineSecurity
category_plot(telecom_df.OnlineSecurity, telecom_df.Churn)
```

```
# univariant analyis on OnlineBackup
category_plot(telecom_df.OnlineBackup, telecom_df.Churn)
```

```
# univariant analyis on DeviceProtection
category_plot(telecom_df.DeviceProtection, telecom_df.Churn)
# univariant analyis on TechSupport
category_plot(telecom_df.TechSupport, telecom_df.Churn)
```

```
 univariant analyis on StreamingTVc
ategory_plot(telecom_df.StreamingTV, telecom_df.Churn)
```

```
# univariant analyis on Streaming Movies
category_plot(telecom_df.StreamingMovies, telecom_df.Churn)
```

```
# univariant analyis on Partner
category_plot(telecom_df.Partner, telecom_df.Churn)
# univariant analyis on Dependentscategory_plot(telecom_df.Dependents, telecom_df.Churn)
```

```
# univariant analyis on PhoneService
category_plot(telecom_df.PhoneService, telecom_df.Churn)
```

```
# univariant analyis on Contract
category_plot(telecom_df.Contract, telecom_df.Churn)
```

```
# univariant analyis on Contract
category_plot(telecom_df.PaperlessBilling, telecom_df.Churn)
```

```
# univariant analyis on Contract
category_plot(telecom_df.PaymentMethod, telecom_df.Churn, h_v='v')
```

```
# check for missing values in the data set
telecom_df.isnull().sum()
```

```
get_boxplot(df,ax):
    ax = sns.boxplot(df, ax=ax, palette="Reds",
                      medianprops =dict(linestyle='-', linewidth=2, color='Yellow'),
                     width =0.4, notch=True,
                     boxprops =dict(linestyle='-', linewidth=2))
return ax
```

```
defdist_plot(df, plots=1):
    fig, ax = plt.subplots(1,2, figsize=(10,4))
    ax[0] = get_boxplot(df, ax[0])
    ax[1] = sns.distplot(df, ax=ax[1], kde_kws={"color": "y"},  hist_kws={"histtype": "step", "color": "k"})
    ax[1].axvline(x = df.mean(), color ='r', linewidth=1.5, linestyle='--', label='mean')
    ax[1].axvline(x = df.median(), color ='g', linewidth=1.5, linestyle='--', label='median')
    ax[1].set(xlabel = df.name, ylabel='frequency', title='Histogram of '+ df.name)
    plt.legend()
    plt.tight_layout()
```

```
dist_plot(telecom_df.TotalCharges)
```

```
dist_plot(telecom_df.MonthlyCharges)
```

```
dist_plot(telecom_df.tenure)
```

```
sns.pairplot(telecom_df,vars= ['tenure','MonthlyCharges','TotalCharges'], hue="Churn")
```

```
g = sns.FacetGrid(telecom_df, row='SeniorCitizen', col="gender", hue="Churn", height=3.5)g.map(plt.scatter, "tenure
", "MonthlyCharges", alpha=0.6)g.add_legend();
```

Step 4: **Data preparation**

```
set(telecom_df.dtypes)

# List of variables to map
varlist =  ['PhoneService', 'PaperlessBilling', 'Churn', 'Partner', 'Dependents']
# Defining the map function
defbinary_map(x):
return x.map({'Yes': 1, "No": 0})
# Applying the function to the housing list
telecom_df[varlist] = telecom_df[varlist].apply(binary_map)
```

```
telecom_df.head()
```

```
telecom_df.columns
```

```
# Creating dummy variables for the variable 'MultipleLines'
ml = pd.get_dummies(telecom_df['MultipleLines'], prefix='MultipleLines')
# Dropping MultipleLines_No phone service column
ml1 = ml.drop(['MultipleLines_No phone service'], 1)
#Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,ml1], axis=1)
```

```
# Creating dummy variables for the variable 'Internetservice'
iss = pd.get_dummies(telecom_df.InternetService)
# Dropping InternetService_No column
iss = iss.drop(['No'], axis=1)telecom_df = pd.concat([telecom_df, iss], axis=1)
```

```
# Creating dummy variables for the variable 'OnlineSecurity'
.os = pd.get_dummies(telecom_df['OnlineSecurity'], prefix='OnlineSecurity')os1 = os.drop(['OnlineSecurity_No intern
et service'], 1)
# Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,os1], axis=1)
```

```
# Creating dummy variables for the variable 'DeviceProtection'.
```

```
dp = pd.get_dummies(telecom_df['DeviceProtection'], prefix='DeviceProtection')dp1 = dp.drop(['DeviceProtection_No internet service'], 1)
# Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,dp1], axis=1)
```

```
# Creating dummy variables for the variable 'TechSupport'. ts = pd.get_dummies(telecom_df['TechSupport'], prefix='TechSupport')ts1 = ts.drop(['TechSupport_No internet service'], 1)
# Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,ts1], axis=1)
```

```
# Creating dummy variables for the variable 'StreamingTV'.
st =pd.get_dummies(telecom_df['StreamingTV'], prefix='StreamingTV')st1 = st.drop(['StreamingTV_No internet service'], 1)
# Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,st1], axis=1)
```

```
# Creating dummy variables for the variable 'StreamingMovies'.
sm = pd.get_dummies(telecom_df['StreamingMovies'], prefix='StreamingMovies')sm1 = sm.drop(['StreamingMovies_No internet service'], 1)
# Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,sm1], axis=1)
```

```
# Defining the map function
defgender_map(x):
return x.map({'Female': 1, "Male": 0})
# Applying the function to the housing list
telecom_df['gender'] = telecom_df[['gender']].apply(gender_map)
```

```
cc = pd.get_dummies(telecom_df.Contract)
# Adding the results to the master dataframet
telecom_df = pd.concat([telecom_df,cc], axis=1)
```

```
# Creating dummy variables for the variable 'OnlineBackup'
ob = pd.get_dummies(telecom_df['OnlineBackup'], prefix='OnlineBackup')
ob1 = ob.drop(['OnlineBackup_No internet service'], 1)
# Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,ob1], axis=1)
```

```
pm = pd.get_dummies(telecom_df.PaymentMethod)
# Adding the results to the master dataframe
telecom_df = pd.concat([telecom_df,pm], axis=1)
```

Final Dataset after Data Preparation

```
telecom_df.head()
```

```
telecom_df.columns
```

```
# drop the original columns
telecom_df = telecom_df.drop(['MultipleLines', 'InternetService', 'OnlineSecurity','OnlineBackup', 'DeviceProtection', 'TechSupport','StreamingTV', 'StreamingMovies'], axis=1)
```

```
telecom_df = telecom_df.drop(['PaymentMethod', 'Contract',], axis=1)
```

Final Dataset after dropping Original Columns for which dummies are created

```
telecom_df.head()
```

```
telecom_df.shape
```

```
# import required libraries
fromsklearn.model_selectionimport train_test_split
```

```
# Create dependent and independent data frames
X = telecom_df.drop(['customerID', 'Churn'], axis=1)# drop CustomerID and churn columns from X
Y = telecom_df['Churn']
```

```
X.shape
Y.shape
# perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size=0.7, test_size=0.3, random_state =100)
```

Step 6: **Feature Scaling**

```
# import Standard Scaler from preprocesing module
fromsklearn.preprocessingimport StandardScaler
```

```
scaler = StandardScaler()
```

Also, apply normalization to x in order to scale all values

```
scale_columns = ['MonthlyCharges', 'tenure', 'TotalCharges']X_train[scale_columns] = scaler.fit_transform(X_train[scale_columns])
```

```
X_train.head()
```

```
scale_columns = ['MonthlyCharges', 'tenure', 'TotalCharges']X_test[scale_columns] = scaler.fit_transform(X_test[scal
e_columns])
X_test.columns
X_train.shape
X_test.shape
X_train.index
```

```
defrate(df):
print(df.name,  ' Rate : ', round(100*sum(df) /len(df), 2), '%')
```

```
# check churn rate in the data set.
rate(y_train)
```

```
Churn  Rate :  26.46 %
```

Co-relation Matrix

```
# check co-realtion between the variables
plt.figure(figsize=(30,20))sns.heatmap(X_train.corr(), annot=True)plt.show()
```

```
#drop co-realted columns
corelated_cols = ['MultipleLines_No', 'OnlineSecurity_No', 'OnlineBackup_No', 'DeviceProtection_No', 'TechSupport
_No',
'StreamingTV_No', 'StreamingMovies_No']X_train = X_train.drop(corelated_cols, axis=1)X_test = X_test.drop(corela
ted_cols, axis=1)
```

```
X_train.head()
X_train.shape
X_test.shape
```

```
importstatsmodels.apiassm
```

```
defget_lrm(y_train, x_train):
    lrm = sm.GLM(y_train, (sm.add_constant(x_train)), family = sm.families.Binomial())
    lrm = lrm.fit()
print(lrm.summary())
return lrm
```

LOGIT MODEL¶

```
# running the logistic regression model once
lrm_1 =get_lrm(y_train, X_train)
X_train = sm.add_constant(X_train)
```

```
logit = sm.Logit(endog=y_train, exog = X_train)
result = logit.fit()
result.summary()
```

PROBIT MODEL

```
X_train = sm.add_constant(X_train)
probit = sm.Probit(endog=y_train, exog = X_train)
result = probit.fit()
result.summary()
```

RECURSIVE FEATURE ELIMINATION¶

```
# using RFE remove some features.# import required libraries
fromsklearn.feature_selectionimport RFE
fromsklearn.linear_modelimport LogisticRegression
```

```
lg_reg = LogisticRegression()rfe = RFE(lg_reg, 15)rfe = rfe.fit(X_train, y_train)
```

```
rfe_df = pd.DataFrame({'columns': list(X_train.columns), 'rank' : rfe.ranking_, 'support' : rfe.support_ }).sort_values(b
y='rank', ascending=True)rfe_df
```

```
# get supported columns
rfe_columns = X_train.columns[rfe.support_]rfe_columns
```

```
# import vif from statsmodel
fromstatsmodels.stats.outliers_influenceimport variance_inflation_factor
```

```
defcalculate_vif(df):
    vif = pd.DataFrame()
    vif['Features'] = df.columns
    vif['vif'] = [variance_inflation_factor(df.values, i) for i inrange(df.shape[1])]
    vif['vif'] =round(vif['vif'],2)
    vif = vif.sort_values(by='vif', ascending=False)
print(vif)
```

```
X_train_lg_1 = X_train[rfe_columns]log_reg_1 = get_lrm(y_train, X_train_lg_1)
```

```
#Pseudo R-squared value of MODEL 1
X_train = sm.add_constant(X_train)
```

```
logit = sm.Logit(endog=y_train, exog = X_train_lg_1)
result = logit.fit()
result.summary()
```

```
#drop Mailed check column due to high p-value
X_train_lg_2 = X_train_lg_1.drop(['Mailed check'], axis=1)
```

MODEL 2

```
log_reg_2 = get_lrm(y_train, X_train_lg_2)
```

```
calculate_vif(X_train_lg_2)
```

```
# drop Month-to-Month as it it highly co-realted
X_train_lg_3 = X_train_lg_2.drop(['Month-to-month'], axis=1)
```

```
#Pseudo R-squared value of MODEL 2
X_train = sm.add_constant(X_train)
logit = sm.Logit(endog=y_train, exog = X_train_lg_2)
result = logit.fit()
result.summary()
```

MODEL 3

```
log_reg_3 = get_lrm(y_train, X_train_lg_3)
```

```
calculate_vif(X_train_lg_3)
```

```
# drop 'TotalCharges' as it is hightly co-realted with other features
X_train_lg_4 = X_train_lg_3.drop(['TotalCharges'], axis=1)
```

```
#Pseudo R-squared value of MODEL 3
X_train = sm.add_constant(X_train)
logit = sm.Logit(endog=y_train, exog = X_train_lg_3)
result = logit.fit()
result.summary()
```

MODEL 4

```
log_reg_4 = get_lrm(y_train, X_train_lg_4)
```

```
# drop DSL, as it is insignificant
X_train_lg_5 = X_train_lg_4.drop(['DSL'], axis=1)
#Pseudo R-squared value of MODEL 4
X_train = sm.add_constant(X_train)
logit = sm.Logit(endog=y_train, exog = X_train_lg_4)
result = logit.fit()
result.summary()
```

MODEL 5

```
log_reg_5 = get_lrm(y_train, X_train_lg_5)
```

```
# drop fiber optic as it is insignificant
X_train_lg_6 = X_train_lg_5.drop(['Fiber optic'], axis=1)
```

```
#Pseudo R-squared value of MODEL 5
X_train = sm.add_constant(X_train)
logit = sm.Logit(endog=y_train, exog = X_train_lg_5)
result = logit.fit()
result.summary()
```

MODEL 6

```
log_reg_6 = get_lrm(y_train, X_train_lg_6)
```

```
# looks all features are significant, lets check VIF
calculate_vif(X_train_lg_6)
```

```
#Pseudo R-squared value of MODEL 6
X_train = sm.add_constant(X_train)
logit = sm.Logit(endog=y_train, exog = X_train_lg_6)
result = logit.fit()
result.summary()
```

```
# predict the values from the model
y_train_pred = log_reg_6.predict(sm.add_constant(X_train_lg_6))y_train_pred[:10]
```

```
y_train_pred_values = y_train_pred.values.reshape(-1)y_train_pred_values[:10]
```

```
X_train_lg_6.columns
```

```
# create a data frame having actual, customerID and predicted
churn_df = pd.DataFrame({'Churn_actual': y_train.values, 'Churn_prob' : y_train_pred_values})
churn_df['Cust_ID'] = y_train.indexchurn_df.head()
```

```
fromsklearnimport metricsfromsklearn.ensemble
import RandomForestClassifierfromsklearn.metrics
import confusion_matrix, classification_report, accuracy_score, roc_curve, auc
# All the features are significant and there is no co-realtion between the variables.
# calucate the confusion matrix.
cnf_matrix = metrics.confusion_matrix(churn_df.Churn_actual, churn_df.Churn_Pred)cnf_matrix
```

```
print(classification_report(churn_df.Churn_actual, churn_df.Churn_Pred))
```

```
# confusion matrix visualization
f, ax = plt.subplots(figsize = (5,5))sns.heatmap(cnf_matrix, annot =True, linewidths =0.5, color ="red", fmt =".0f", ax
=None)plt.xlabel("Predicted value")plt.ylabel("True value")plt.title("Confusion Matrix of Logistic Regression")plt.sho
w()
```

```
# calculate the accuracy
print('Accuracy of the model : ', metrics.accuracy_score(churn_df.Churn_actual, churn_df.Churn_Pred))
```

```
Accuracy of the model :  0.8058429701765064
```

```
print('Recall : ', metrics.recall_score(churn_df.Churn_actual, churn_df.Churn_Pred))
```

```
Recall :  0.5276073619631901
```

```
print('Precision : ', metrics.precision_score(churn_df.Churn_actual, churn_df.Churn_Pred))
```

```
Precision :  0.6686103012633625
```

```
tn = cnf_matrix[0,0]fn = cnf_matrix[1,0]fp = cnf_matrix[0,1]tp = cnf_matrix[1,1]
```

```
# Sensistivity , True Positive rateprint('Sensitivity (True Positive Rate) TP / TP + FN : ', tp / (tp + fn))
```

```
Sensitivity (True Positive Rate) TP / TP + FN :  0.5276073619631901
```

```
# specificity, print('Specificity TN / (TN + FP) : ', tn / (tn + fp))
```

```
Specificity TN / (TN + FP) :  0.9059310344827586
```

```
# False positive rateprint('False positive rate FP / (TN + FP) : ', fp / (tn+fp))
```

```
False positive rate FP / (TN + FP) :  0.09406896551724138
```

Step 8 : **ROC Curve**

```
defdraw_roc_curve(actual, probs):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs, drop_intermediate =False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)'% auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.legend(loc="lower right")
    plt.show()
```

```
draw_roc_curve(churn_df.Churn_actual, churn_df.Churn_prob)
```

```
# to the predict for different thresholds
tresholds = [float(x)/10for x inrange(10)]tresholds.append(0.45)tresholds.append(0.55)tresholds =sorted(tresholds)
for i insorted(tresholds):
    churn_df[i] = churn_df.Churn_prob.map(lambda row: 1if row > i else0)
churn_df.head()
```

```
optimal_df = pd.DataFrame(columns=['prob', 'accuracy', 'sensitivity', 'specificity'])for i in tresholds:
    cm = metrics.confusion_matrix(churn_df.Churn_actual, churn_df[i])
    tn = cm[0,0]
    fn = cm[1,0]
    fp = cm[0,1]
    tp = cm[1,1]
    accuracy = (tn + tp) / (tn + tp + fp + fn)
    specificity = tn / (tn + fp)
    sensitivity = tp / (tp + fn)
    optimal_df.loc[i] = [i, accuracy, sensitivity, specificity]
```

```
optimal_df
```

```
# plot the curve
optimal_df.plot(x ='prob', y=['accuracy', 'sensitivity', 'specificity'])plt.show()
```

```
# from the above curve, optimal value
optimal_value =0.3
```

```
churn_df['final_pred'] = churn_df.Churn_prob.map(lambda x: 1if x >0.3else0)
churn_df.head()
```

```
# calcualte the accuracyfinal_accuracy = metrics.accuracy_score(churn_df.Churn_actual, churn_df.final_pred)print('F
inal Accuracy : ', final_accuracy)
```

```
Final Accuracy :  0.7715560965713126
```

```
# calcualte the other parameters
final_cm = metrics.confusion_matrix(churn_df.Churn_actual, churn_df.final_pred)
print('Confusion matrix \n', final_cm)
# confusion matrix visualization
f, ax = plt.subplots(figsize = (5,5))sns.heatmap(final_cm, annot =True, linewidths =0.5, color ="red", fmt =".0f", ax=
None)
plt.xlabel("Predicted value")
plt.ylabel("True value")
plt.title("Confusion Matrix of Logistic Regression final model when threshold=0.3")
plt.show()
```

```
print(classification_report(churn_df.Churn_actual, churn_df.final_pred))
```

```
tn = final_cm[0,0]fn = final_cm[1,0]fp = final_cm[0,1]tp = final_cm[1,1]
sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)
false_positive_rate =1– specificity
positive_predictive_rate = tp / (tp + fp)
negative_predictive_rate = tn / (tn + fn)
```

```
print('optimal threshold : ', optimal_value)
print('sensitivity : ', sensitivity)
print('specificity : ', specificity)
print('false_positive_rate : ', false_positive_rate)
print('positive_predictive_rate : ', positive_predictive_rate)
print('negative_predictive_rate : ', negative_predictive_rate)
```

```
# precision and recall trade off
```

```
fromsklearn.metricsimport precision_recall_curve
```

```
p, r, tresholds = precision_recall_curve(churn_df.Churn_actual, churn_df.Churn_prob)
```

```
plt.plot(tresholds, p[:-1], 'g-')plt.plot(tresholds, r[:-1], 'r-')plt.show()
```

```
X_test = sm.add_constant(X_test)
```

```
fromsklearn.linear_modelimport LogisticRegression
lr_model = LogisticRegression()lr_model.fit(X_train,y_train)
accuracy_lr = lr_model.score(X_test,y_test)
print("Logistic Regression accuracy on test dataset is :",accuracy_lr)
```

Logistic Regression accuracy on test dataset is : 0.7884524372929484

```
X_test = X_test[X_train_lg_6.columns]X_test.head()
```

```
# predict the X_test
y_test_pred = log_reg_6.predict(sm.add_constant(X_test))
```

```
test_pred_df = pd.DataFrame(y_test)
test_pred_df.head(5)
```

```
y_test_df = pd.DataFrame(y_test_pred)y_test_df['CustID'] = y_test_df.index
y_test_df.head()
```

```
y_test_df.reset_index(drop=True, inplace=True)
test_pred_df.reset_index(drop=True, inplace=True)
```

```
test_pred_final_df = pd.concat([ test_pred_df, y_test_df], axis=1)
test_pred_final_df.head()
```

```
test_pred_final_df= test_pred_final_df.rename(columns={0 : 'Churn_Prob', 'Churn': 'Churn_Actual'})
test_pred_final_df.head()
```

```
test_pred_final_df['Churn_final_pred'] = test_pred_final_df.Churn_Prob.map(lambda x : 1if x >0.42else0)
```

```
test_pred_final_df.head()
```

```
test_cm = metrics.confusion_matrix(test_pred_final_df.Churn_Actual, test_pred_final_df.Churn_final_pred)
test_cm
print('Test Sensitivity : ', test_cm[1,1] / (test_cm[1,1] + test_cm[1,0]))
print('Test Specificity : ', test_cm[0,0] / (test_cm[0,0] + test_cm[0,1]))
```

```
Test Sensitivity :  0.49911504424778763
Test Specificity :  0.8921188630490956
```

```
print(classification_report(test_pred_final_df.Churn_Actual, test_pred_final_df.Churn_final_pred))
# confusion matrix visualization
f, ax = plt.subplots(figsize = (5,5))sns.heatmap(cnf_matrix, annot =True, linewidths =0.5, color ="red", fmt =".0f", ax
=None)
plt.xlabel("Predicted value")
plt.ylabel("True value")
plt.title("Confusion Matrix of Logistic Regression on Test Dataset")
plt.show()
#plot ROC
draw_roc_curve(test_pred_final_df.Churn_Actual, test_pred_final_df.Churn_final_pred)
```

```
fromsklearn.ensembleimport RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size=0.7, test_size=0.3, random_state =100)model_rf = R
andomForestClassifier(n_estimators=1000 , oob_score =True, n_jobs =-1,
                                   random_state =50, max_features ="auto",
                                   max_leaf_nodes =30)model_rf.fit(X_train, y_train)
# Make predictionsy_
predicted = model_rf.predict(X_test)
print (metrics.accuracy_score(y_test, y_predicted))
importances = model_rf.feature_importances_weights = pd.Series(importances, index=X.columns.values)weights.sort
_values()[-10:].plot(kind ='barh')
```