Customer Churn Reduction

Rohit Haritash

30/05/18

rohit.haritash@gmail.com

Contents

Chapter 1

Introduction

Customer churn is defined as the rate when a customer leaves or stop paying for a product of service. Reduction customer churn is important because cost of acquiring a new customer is higher than retaining an existing one. The full cost of customer churn includes both lost revenue and the marketing costs involved with replacing those customers with new ones. Reducing customer churn is a key business goal of every business. Predicting and preventing customer churn represents a huge additional potential revenue source for every business.

1.1 Problem Statement

In this problem statement, we were provided with a train and test dataset of a telecom company. The data set consists of 20 variables describing various services and charges associated with them, duration and any service calls made by customer. The dataset also contains geographic location of customers in form of state. 'Churn' is the target variable, which tells us weather the customer has churned out or not.

1.2 Dataset

Two different datasets were provided as train data and test data. Data contains 20 predictor variables and 1 target variables. Train data had 3333 observations while test data consist of 1667. Total 21 variables were present in data. All the variables are described in table 1 below.

Variables	Description
State *	State to which customer belongs
Account length	Service usage period
Area code	Telephone area code
Phone number *	Customer's phone number
International plan *	'yes' if customer opted for international plan else 'no'
Voice mail plan *	'yes' if customer opted for voice mail plan else 'no'
Number vmail messages	Number of voice messages stored or received by customer
Total day minutes	Total minutes in day time usage
Total day calls	Total calls made in day time
Total day charges	Charges for services used during day time
Total eve minutes	Total minutes in evening time usage
Total eve calls	Total calls made in evening time
Total eve charges	Charges for services used during evening time
Total night minutes	Total minutes in night time usage
Total night calls	Total calls made in night time
Total night charges	Charges for services used during night time
Total intl minutes	Total international minutes used
Total inil calls	Total international calls made
Total intl charges	Charges for international calls
Number customer services call	Services call made by customer
Churn	Target - 'True.' If customer churned else 'False'

Table1. Variables description

Out of 20 variables, 04 were categorical and 16 were continuous. Categorical variables are marked as * in table. Sample data is shown below.

state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes
KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4
OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5
NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2
OH	84	408	375-9999	yes	no	0	299.4	71	50.9	61.9
OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3
AL	118	510	391-8027	yes	no	0	223.4	98	37.98	220.6
MA	121	510	355-9993	no	yes	24	218.2	88	37.09	348.5
MO	147	415	329-9001	yes	no	0	157	79	26.69	103.1
LA	117	408	335-4719	no	no	0	184.5	97	31.37	351.6
WV	141	415	330-8173	yes	yes	37	258.6	84	43.96	222
IN	65	415	329-6603	no	no	0	129.1	137	21.95	228.5
RI	74	415	344-9403	no	no	0	187.7	127	31.91	163.4

total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn
99	16.78	244.7	91	11.01	10	3	2.7	1	False.
103	16.62	254.4	103	11.45	13.7	3	3.7	1	False.
110	10.3	162.6	104	7.32	12.2	5	3.29	0	False.
88	5.26	196.9	89	8.86	6.6	7	1.78	2	False.
122	12.61	186.9	121	8.41	10.1	3	2.73	3	False.
101	18.75	203.9	118	9.18	6.3	6	1.7	0	False.
108	29.62	212.6	118	9.57	7.5	7	2.03	3	False.
94	8.76	211.8	96	9.53	7.1	6	1.92	0	False.
80	29.89	215.8	90	9.71	8.7	4	2.35	1	False.
111	18.87	326.4	97	14.69	11.2	5	3.02	0	False.
83	19.42	208.8	111	9.4	12.7	6	3.43	4	True.
148	13.89	196	94	8.82	9.1	5	2.46	0	False.

Figure 1. Sample train data

Chapter 2

Methodology

Customer churn reduction is a business scenario in which a company is trying to retain a customer which is more likely to leave the services. For reducing churn rate, we need to identify which customers are most likely to churn and which are not. So Churn reduction is a classification problem.

The solution is divided into 3 parts.

1. Exploratory data analysis(EDA) was performed to explore the structure of data. Some of the basic assumptions were made about the data ie. Which variables are most likely causing churn. During exploration dataset was checked for missing values, multi collinearity and other model/algorithm specific assumptions.
2. After EDA, for learning two models were used, logistic regression and random forest. Some data pre-processing was done to prepare training data for learning model.
3. In the last part, performance tuning was done to increase the accuracy of models.

Both the algorithms and EDA were implemented in R and python. Both implementations were similar with little difference due to difference in learning algorithm implementation.

2.1 Exploratory Data Analysis (EDA)

Exploratory data analysis a.k.a. EDA was performed on training data using R and python. We looked at the structure of training data and found 20 predictors, 1 target variable and 3333 observations.

2.1.1 The Target Variable - Churn

The target variable was 'churn'. Initially if the customers churned out it flagged as 'True.' , otherwise 'False.'. Later during it was changes as True = 1 and False = 0. Out of 3333 customers, 483 customers churned out and 2850 didn't churned out.

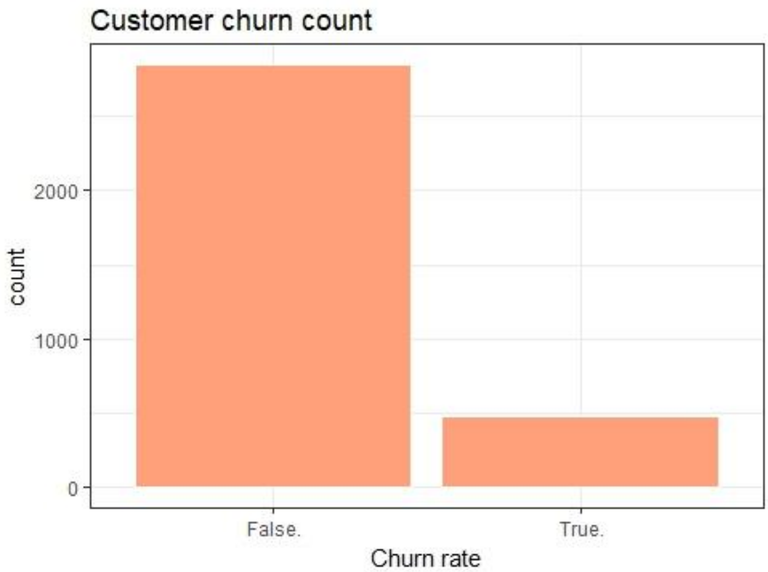

Fig2. Customer churn frequency

Since the churn event rate is approx. 14 % so it was concluded that it was not a highly imbalanced class. It won't affect our analysis significantly.

Also since our target variable is binary class, it is following the first assumption of logistic regression which is required dependent variable to be binary.

2.1.2 Missing Value Analysis

No missing values were present in the training and test dataset.

account length	0	international plan	0
voice_mail_plan	0	number_vmail_messages	0
total_day_minutes	0	total_day_calls	0
total_day_charge	0	total_eve_minutes	0
total_eve_calls	0	total_eve_charge	0
total_night_minutes	0	total_night_calls	0
total_night_charge	0	total_intl_minutes	0
total_intl_calls	0	total_intl_charge	0
number_customer_service_calls	0	churn	0

Table2. Missing value count

2.1.3 Multicollinearity

Multicollinearity exists whenever two or more of the predictors in a regression model are moderately or highly correlated. Multicollinearity is the condition when one predictor can be used to predict other. The basic problem is multicollinearity results in unstable estimation of coefficients which makes it difficult to access the effect of independent variable on dependent variable. Correlation plot was used in R and python to detect highly collinear variables.

From the correlation plot we can see that –

1. 'Total day minutes' and 'total day charges' are highly collinear
2. 'Total eve minutes' and 'total eve charges' are highly collinear
3. 'Total night minutes' and 'total night charges' are highly collinear
4. 'Total intl minutes' and 'total intl charges' are highly collinear

Random forest can handle multi collinear variables because of bagging approach. So we will not remove these variables in random forest implementation.

One of the assumptions of logistic regression is that logistic regression requires there to be little or no multicollinearity among the independent variables. This means that the independent variables should not be too highly correlated with each other. Due to this assumption, one the predictors from each set was removed when logistic learner was trained.

Multicollinearity matrix is visualized in figure 3.

Variables are highly collinear are highlighted with red colour with their corresponding score. Apart from correlation matrix, multicollinearity was again checked during logistic regression diagnostics using VIF (Variation inflation factor).

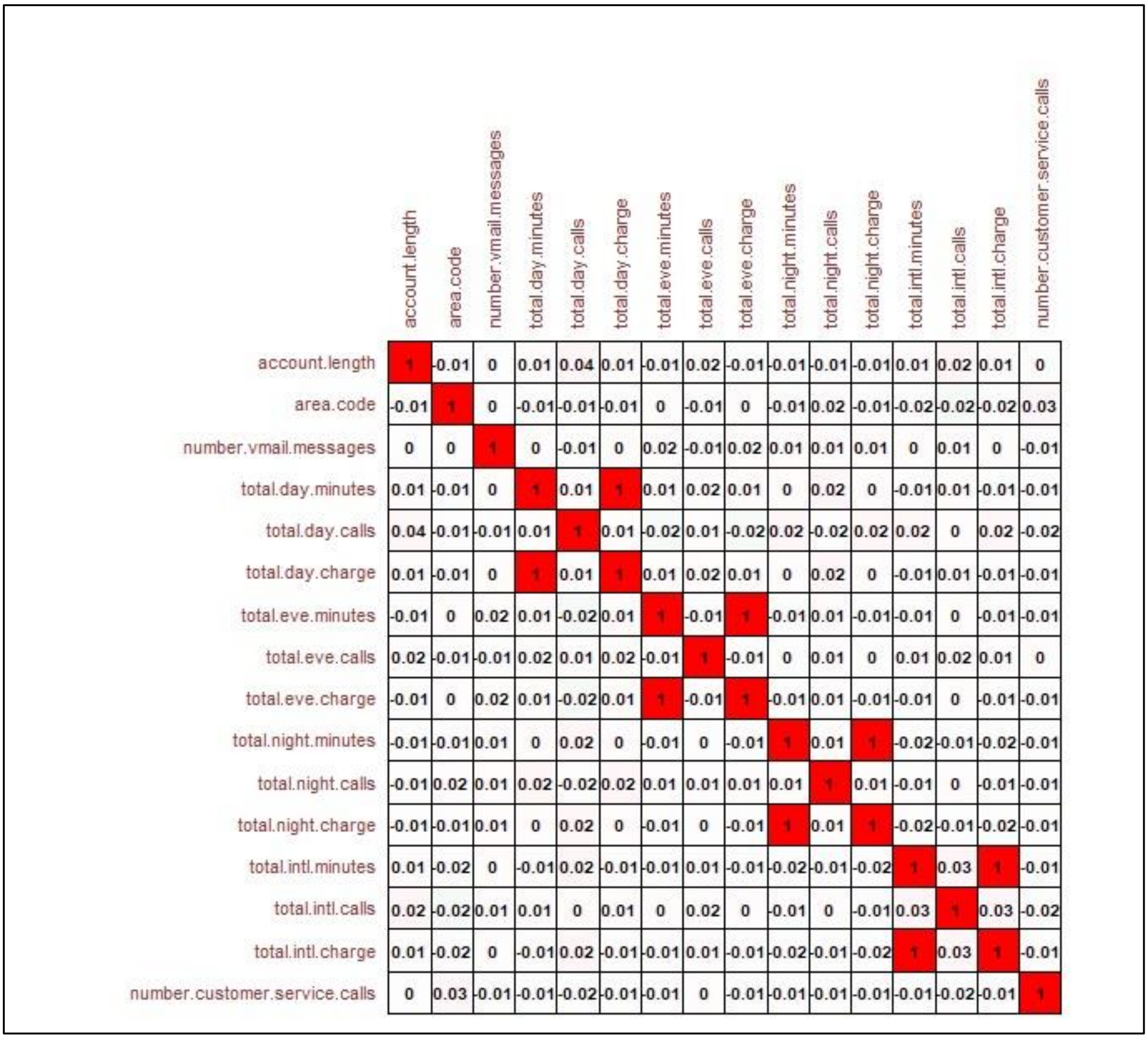

Fig 3. Multicollinearity matrix

2.1.4 Analysis of churn ratio with different predictors

The churn ratio with each predictors was analysed using boxplot and bar chart.

1. **State**

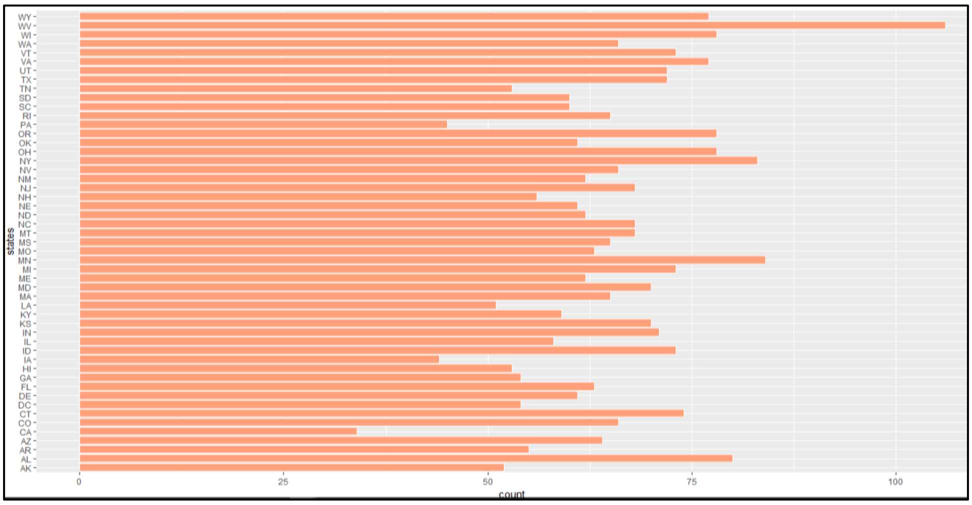

Fig. 4 State wise customer count

Fom the plot we can that maximum customers are from west vergenia and lowest are from California.

2. International plan

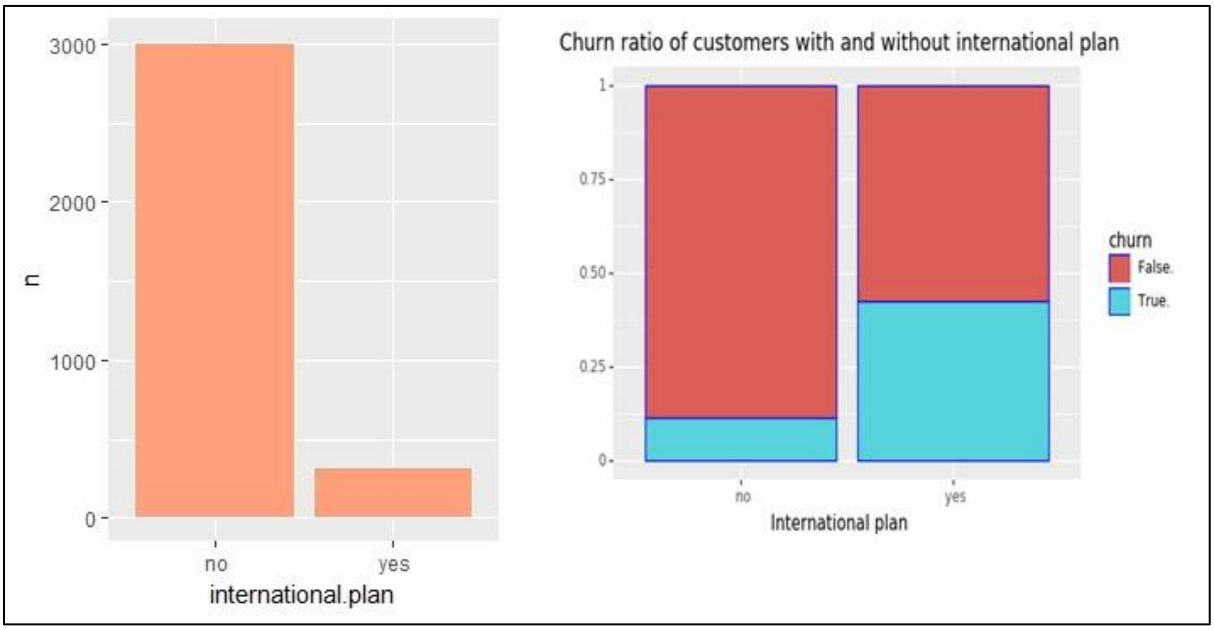

Fig 4. Churn ratio with and without international plan subscription

Most of the customers did not opt for the international plan subscription. Customer churn is more with customers with international plan. Further details can be examined by looking at percentage.

Churn	N	Percentage
False	2664	88.5
True	346	11.5

Table 3. Churn rate % for customers without international plan subscription

Churn	N	Percentage
False	186	57.6
True	137	42.4

Table 4. Churn rate % for customers with international plan subscription

As it is evident that only 11.5 % customer churned in without international plan category. 42.4 % customer churn out with international plan subcription. There may be some issue with internation plan as customer churn rate is higher with international plan.

3. Voice mail plan

922 customers subscribed for voice mail plan, 2411 did not.

Churn	N	Percentage
False	842	91.3
True	80	8.68

Table 5. Churn rate % for customers with voice mail plan subscription

Churn	N	Percentage
False	2008	83.3
True	403	16.7

Table 6. Churn rate % for customers without voice mail plan subscription

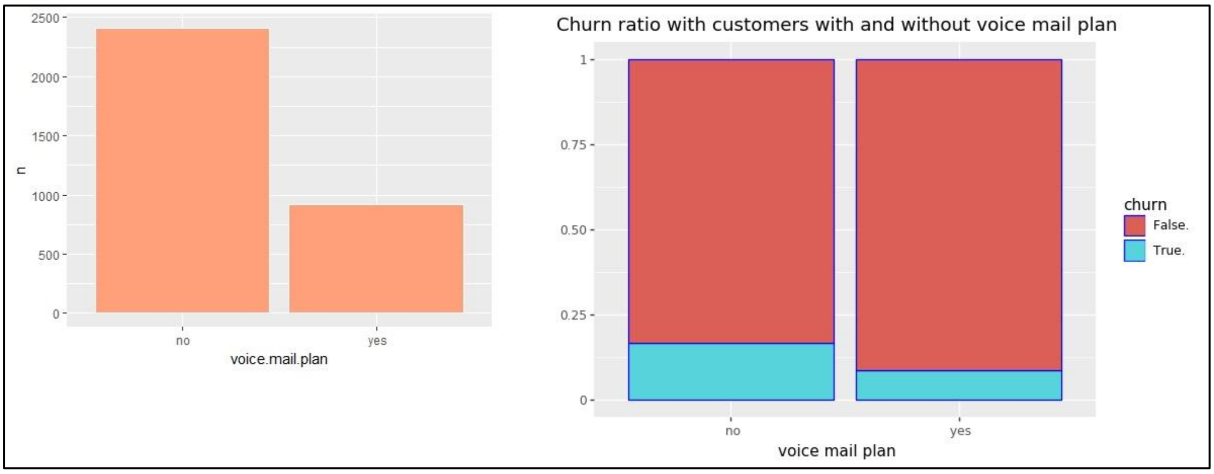

Fig 5. Churn ratio with and without voice mail plan subscription

922 customers have voice mail plan and 80 (8.68 %) customers out of 922 churned out. 2411 customers don't have voice mail plan and 403 (16.7 %) out of 2411 churn out. So customers without voice mail plan have higher churn rate.

4. number customer service calls

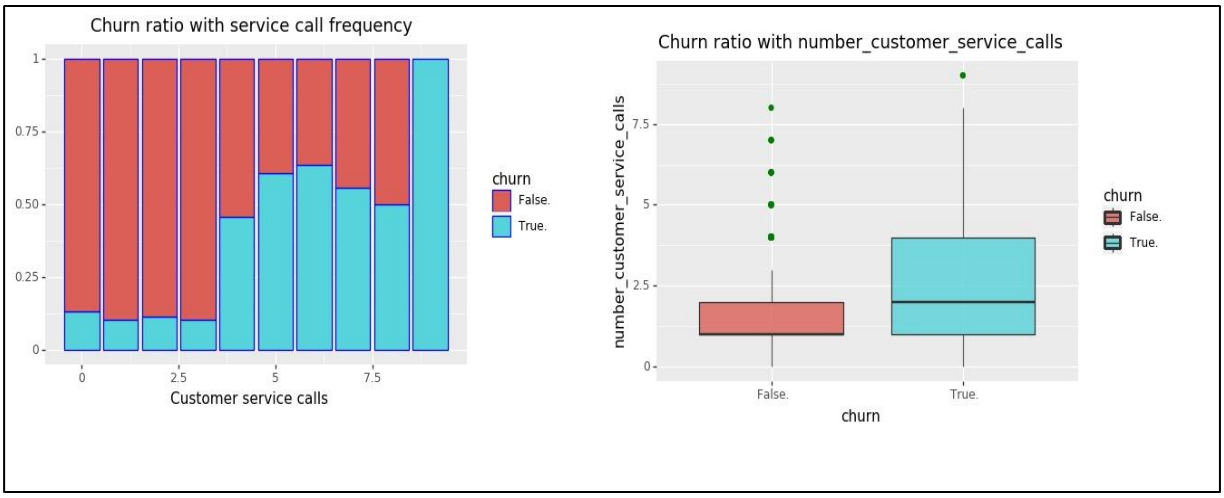

Fig. 6 Churn ratio with service call frequency

The churn rate is increasing with an increase in frequency of service calls.

5. Total day minutes, total day calls and total day charges

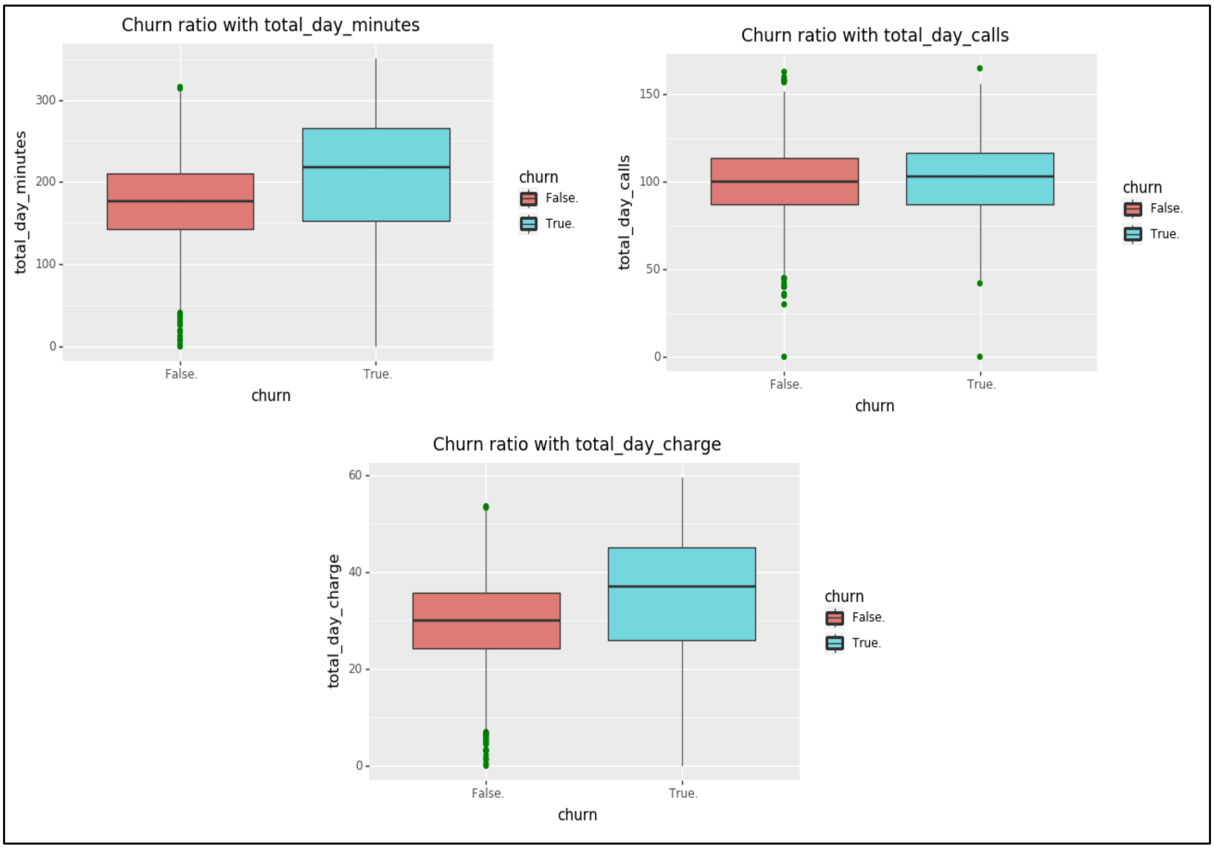

Fig 7. Total day minutes, total day calls and total day charge

	Total day minutes	Total day calls	Total day charge
Count	3333	3333	3333
Mean	179.775	100.435	30.562
Standard deviation	54.467	20.069	9.259
Min	0.0	0.0	0.0
25 %	143.70	87	24.43
50 %	179.40	101	30.50
75 %	216.40	114	36.79
Max	350.80	165	59.64

Table 7. summary statistics for Total day minutes, total day calls and total day charge

From the summary statistics, no anomalies are visible. Every value seems to be in standard range. From the plots we can say that total number of day calls are approximately same for across churn and non-churn customers. But day call duration and day call charges are slightly higher in case of customers who are churned out. Here 'total day minutes' and 'total day charges' are collinear.

6. Total eve minutes, total eve calls and total eve charge

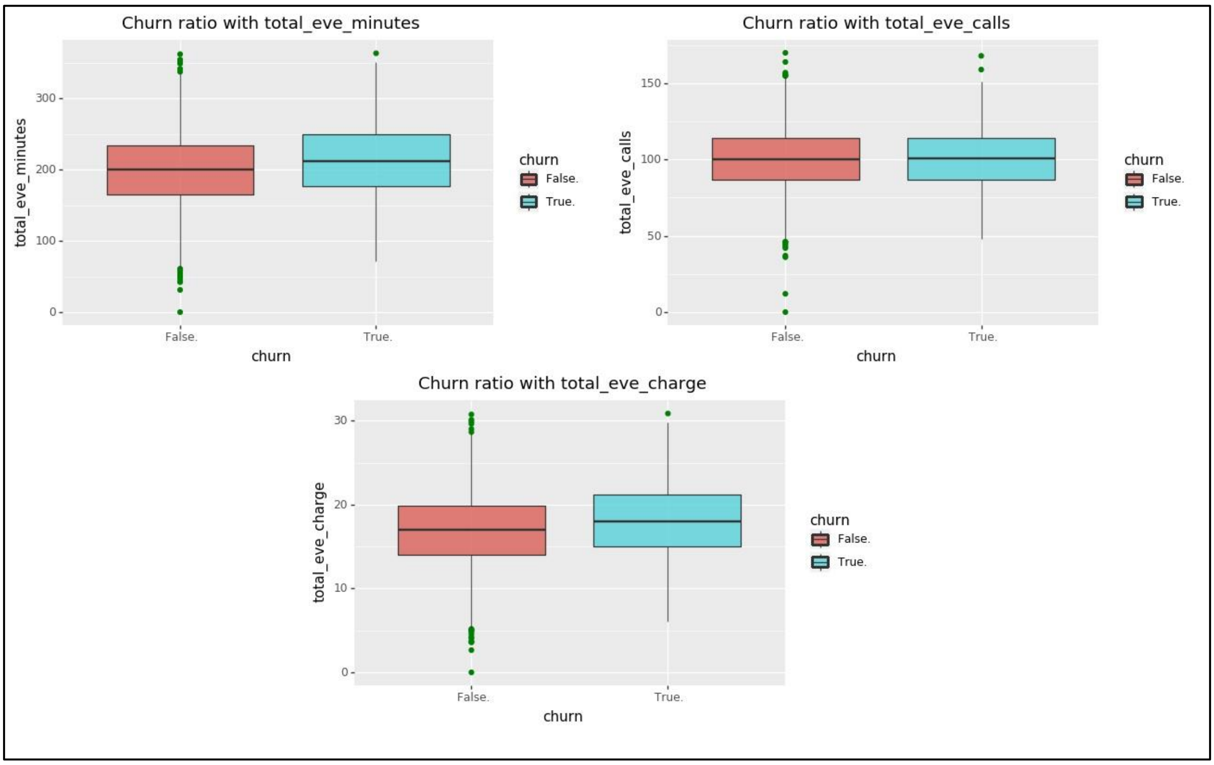

Fig 8. Total eve minutes, total eve calls and total eve charge

	Total eve minutes	Total eve calls	Total eve charge
Count	3333	3333	3333
Mean	200.98	100.114	17.083
Standard deviation	50.71	19.922	4.31
Min	0.0	0.0	0.0
25 %	166.60	87	14.16
50 %	201.40	100	17.12
75 %	235.30	114	20.00
Max	363.70	170	30.91

Table 8. summary statistics for Total eve minutes, total eve calls and total eve charge

No anomalies are visible in statistic summary. Every value seems to be in standard range. From the plots we can say that total number of eve calls are approximately same for across churn and non-churn customers. But eve call duration and eve call charge are slightly higher in case of customers who are churned out. Here ‘total eve minutes’ and ‘total eve charge’ are collinear.

7. Total night minutes, total night calls and total night charge

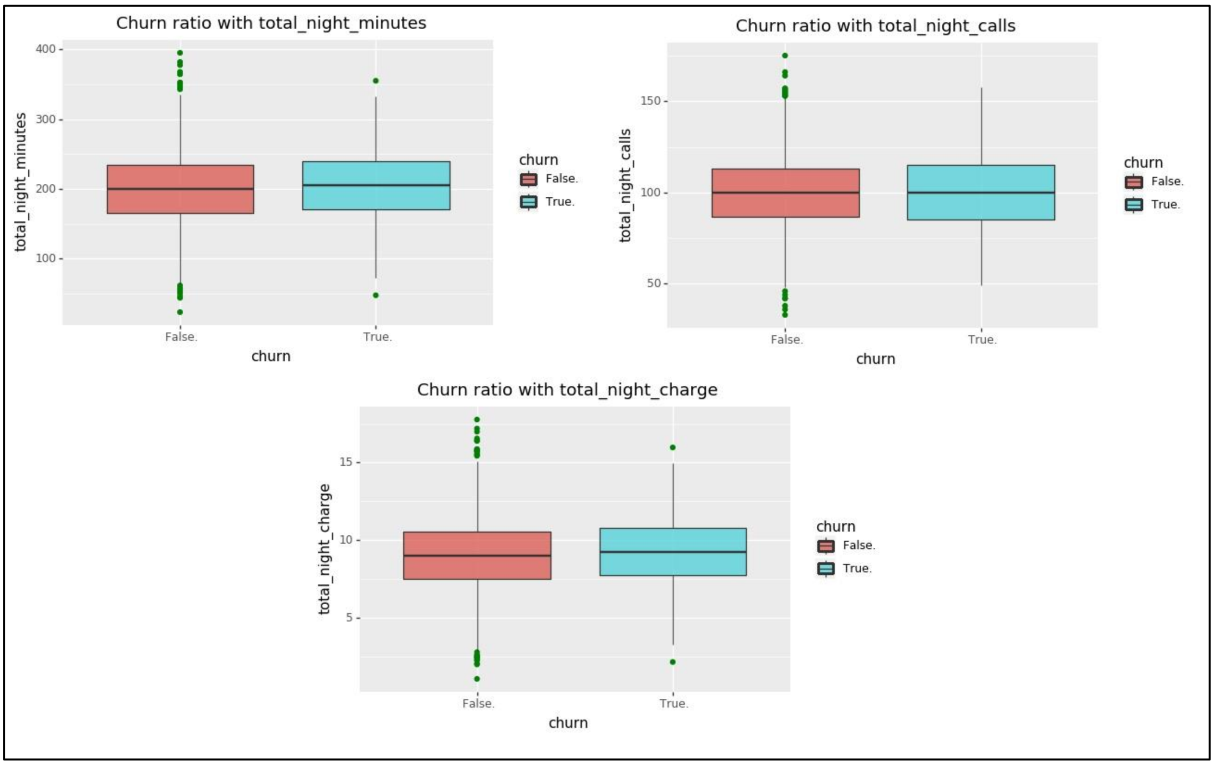

Fig 9. Total night minutes, total night calls and total night charge

	Total night minutes	Total night calls	Total night charge
Count	3333	3333	3333
Mean	200.87	100.107	9.039
Standard deviation	50.57	19.56	2.275
Min	23.20	33	1.04
25 %	167	87	7.52
50 %	201	100	9.05
75 %	235	113	10.59
Max	395	175	10.77

Table 8. summary statistics for Total night minutes, total night calls and total night charge

No anomalies are visible in statistic summary. Every value seems to be in standard range. From the plots we can say that total number of night calls, night minutes and night charges are approximately same for across churn and non-churn customers. ’total night minutes’ and ‘total night charge’ are collinear.

8. Total intl minutes, total intl calls and total intl charges

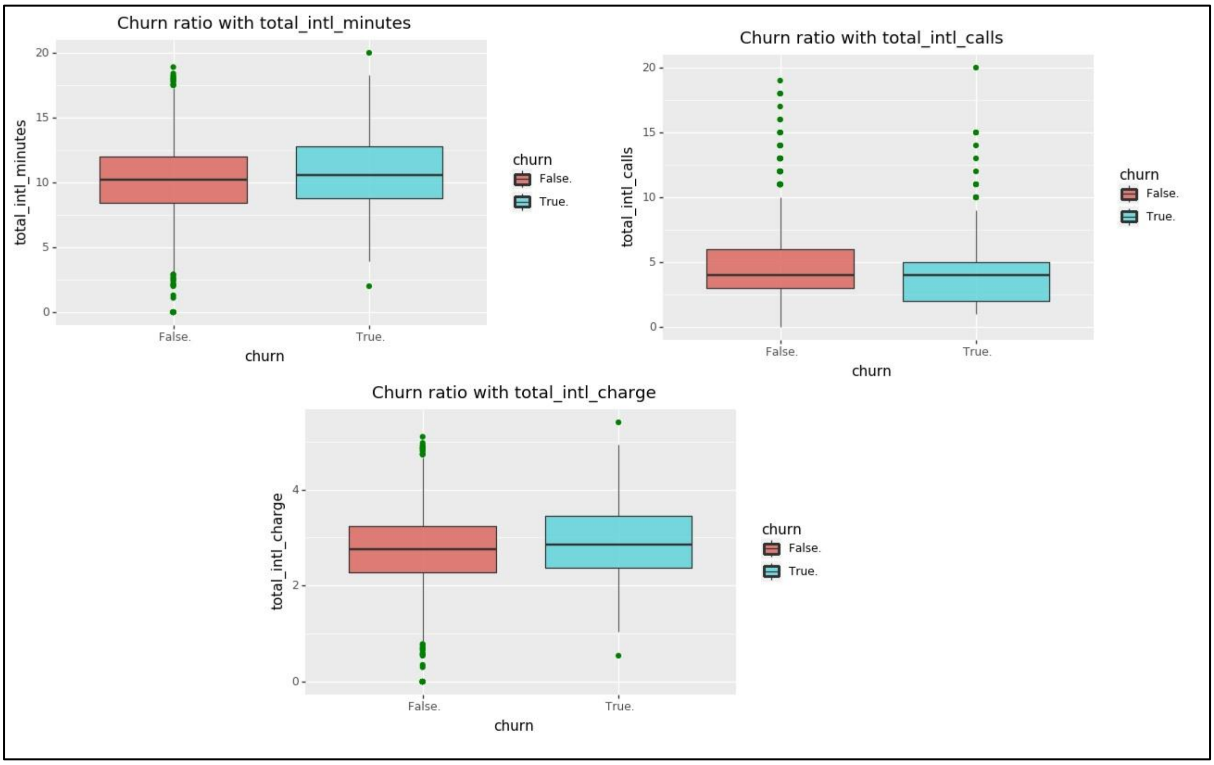

Fig 10. Total intl minutes, total intl calls and total intl charge

	Total intl minutes	Total intl calls	Total intl charge
Count	3333	3333	3333
Mean	10.23	4.47	2.764
Standard deviation	2.79	2.46	.753
Min	0	0	0
25 %	8.50	3	2.30
50 %	10.30	4	2.78
75 %	12.10	6	3.27
Max	20	20	5.40

Table 9. summary statistics for Total intl minutes, total intl calls and total intl charge

No anomalies are visible in statistic summary. Every value seems to be in standard range. From the plots we can say that total number of international calls, international minutes and international charges are approximately same for across churn and non-churn customers. ’total intl minutes’ and ‘total intl charge’ are collinear.

9. Variable importance

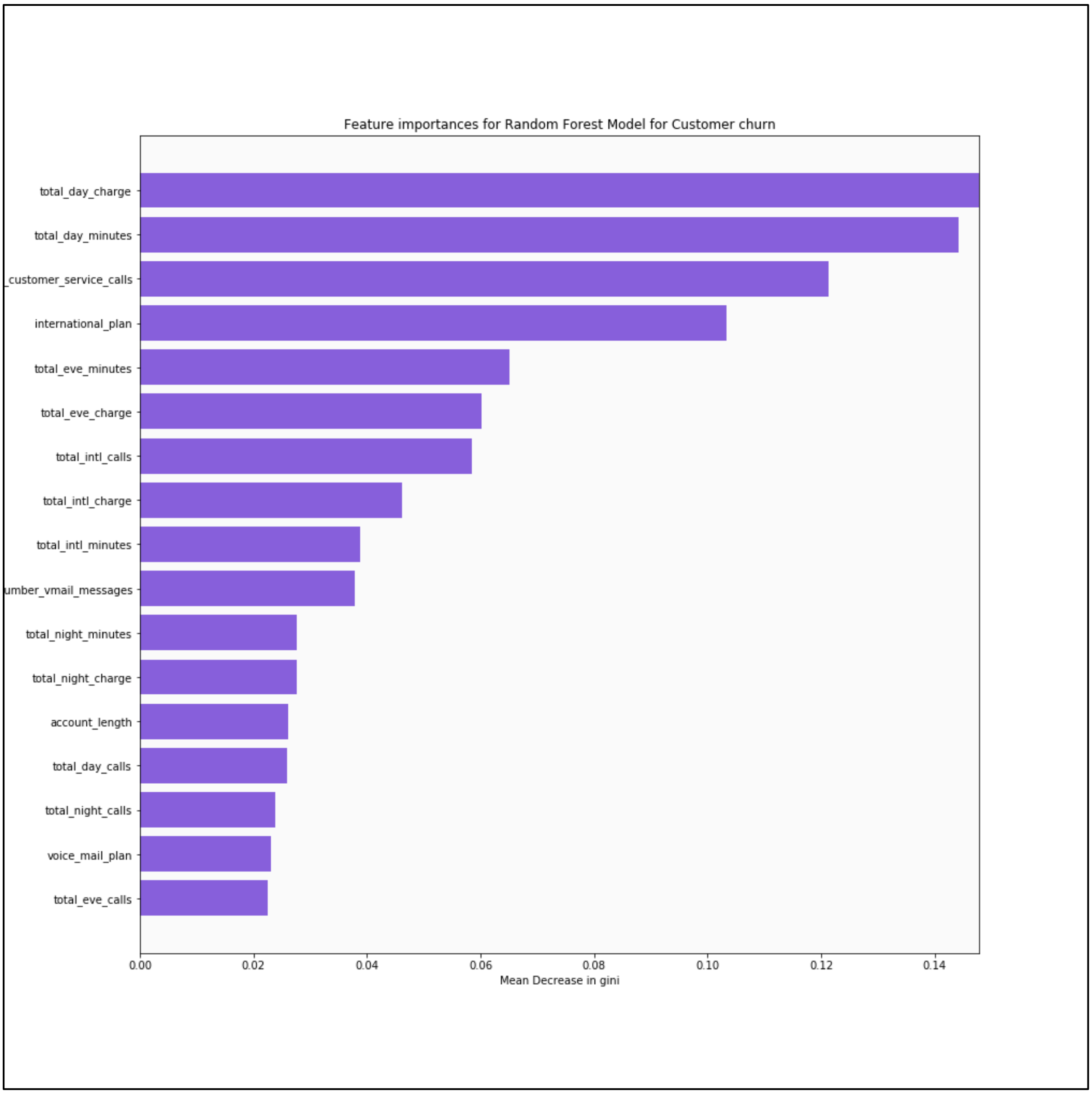

Figure 11. Variable importance

2.2 Modeling

Customer churn reduction is a binary classification problem. Here we have to build a model which can classify if a customer will churn out of not. In our dataset, dependent variable 'churn' was binary. So two machine learning algorithms were selected for learning.

1. Logistic regression, a member of generalized linear model family
2. Random forest, an ensemble tree based technique

Both training models logistic regression and random forest were implemented in R and python. After building an initial model, performance tuning was done using hyperparameter tuning for optimised parameters.

In R , 0/False was the positive class while in python implementation True/1 was positive class.

2.2.1 Logistic regression

First logistic algorithm was trained.

Logistic algorithm works on numeric data. So categorical data ie. International plan and voice mail plan and target predictor 'Churn' were encoded in to binary encoding.

In case if international plan and voice mail plan, yes was coded as 1 and no as 0. In case of Churn True was encoded as 1 and False as 0. True/1 means customer churned and False/0 means it stayed.

- Train data was divided into train dataset and validation set.
- Logistic regression models were trained on train dataset.
- Validation set and AIC score was used to select the best models out of all trained models.
- Final test and prediction was performed on test data which was provided separately.

R implementation

In R 4 different models were trained.

1. fullModel was trained with all the predictors.
2. Model2 was trained by removing collinear variables.
3. model_backward was trained using backward elimination approach.
4. A model with forward elimination was trained but it was equivalent to backward elimination approach.

fullModel

```
summary(fullModel)
Call:
glm(formula = Churn ~ ., family = binomial(link = "logit"), data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1327  -0.5188  -0.3414  -0.1933   3.3265

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.836e+00  8.637e-01 -10.231  < 2e-16 ***
account.length            1.062e-04  1.658e-03   0.064 0.948912
international.plan1       1.950e+00  1.727e-01  11.295  < 2e-16 ***
voice.mail.plan1         -2.427e+00  7.056e-01  -3.439 0.000583 ***
number.vmail.messages     4.704e-02  2.213e-02   2.125 0.033562 *
```

```
total.day.minutes              2.582e+00  3.939e+00   0.655 0.512215    
total.day.calls                4.543e-03  3.256e-03   1.395 0.162923    
total.day.charge              -1.511e+01  2.317e+01  -0.652 0.514408    
total.eve.minutes              6.128e-01  1.953e+00   0.314 0.753647    
total.eve.calls                2.533e-03  3.310e-03   0.765 0.444154    
total.eve.charge              -7.126e+00  2.297e+01  -0.310 0.756396    
total.night.minutes           -1.731e-01  1.059e+00  -0.163 0.870202    
total.night.calls             -8.406e-04  3.454e-03  -0.243 0.807717    
total.night.charge             3.957e+00  2.354e+01   0.168 0.866526    
total.intl.minutes            -4.260e+00  6.301e+00  -0.676 0.498984    
total.intl.calls              -1.024e-01  3.027e-02  -3.382 0.000720 ***
total.intl.charge              1.607e+01  2.333e+01   0.689 0.490923    
number.customer.service.calls  5.005e-01  4.820e-02  10.383  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1934.3  on 2333  degrees of freedom
Residual deviance: 1514.0  on 2316  degrees of freedom
AIC: 1550

Number of Fisher Scoring iterations: 6
```

Model2

```
> summary(model2)

Call:
glm(formula = Churn ~ . - total.day.minutes - total.eve.minutes - 
    total.night.minutes - total.intl.minutes, family = binomial(link = "logit"), 
    data = train_data)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.1280  -0.5209  -0.3397  -0.1929   3.3593  

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)    
(Intercept)                   -8.8607344  0.8636786 -10.259  < 2e-16 ***
account.length                 0.0001257  0.0016568   0.076 0.939502    
international.plan1            1.9437989  0.1723920  11.275  < 2e-16 ***
voice.mail.plan1              -2.4224099  0.7048692  -3.437 0.000589 ***
number.vmail.messages          0.0467668  0.0221136   2.115 0.034443 *  
total.day.calls                0.0046369  0.0032528   1.426 0.154008    
total.day.charge               0.0790371  0.0075897  10.414  < 2e-16 ***
total.eve.calls                0.0025136  0.0033060   0.760 0.447064    
total.eve.charge               0.0833592  0.0159778   5.217 1.82e-07 ***
total.night.calls             -0.0008455  0.0034513  -0.245 0.806482    
total.night.charge             0.1098887  0.0294295   3.734 0.000188 ***
total.intl.calls              -0.1020647  0.0301623  -3.384 0.000715 ***
total.intl.charge              0.2972813  0.0895991   3.318 0.000907 ***
number.customer.service.calls  0.4997406  0.0481675  10.375  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1934.3  on 2333  degrees of freedom
Residual deviance: 1515.0  on 2320  degrees of freedom
AIC: 1543

Number of Fisher Scoring iterations: 6
```

If we compare fullModel and model2 summary, we can see that in full model due to variables with multi-collinear only **international plan, voice mail plan, number customer service call and total intl calls** are considered statistically significant with **AIC score 1550.**

After removing highly collinear variables in model2, variables **international plan, voice mail plan, number customer service call along with total day charge, total eve charge, total night charge, total intl charge, total intl call and total number vmail meaasage**are considered statistically significant with **AIC score 1543.**

Model with lower AIC score is considered better. Next, backward and forward elimination method was used. Models for forward and backward method were same so, selecting backward elimination method.

```
> summary(model_backward)

Call:
glm(formula = Churn ~ international.plan + voice.mail.plan +
    number.vmail.messages + total.day.minutes + total.eve.minutes +
    total.night.charge + total.intl.calls + total.intl.charge +
    number.customer.service.calls, family = binomial(link = "logit"),
    data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1014  -0.5209  -0.3373  -0.1952   3.3215

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                   -8.211093   0.620068 -13.242  < 2e-16 ***
international.plan1            1.943798   0.171974  11.303  < 2e-16 ***
voice.mail.plan1              -2.439077   0.703021  -3.469 0.000522 ***
number.vmail.messages          0.047180   0.022048   2.140 0.032366 *
total.day.minutes              0.013504   0.001288  10.480  < 2e-16 ***
total.eve.minutes              0.007039   0.001357   5.186 2.14e-07 ***
total.night.charge             0.110219   0.029372   3.753 0.000175 ***
total.intl.calls              -0.102416   0.030138  -3.398 0.000678 ***
total.intl.charge              0.298313   0.089341   3.339 0.000841 ***
number.customer.service.calls  0.495978   0.047997  10.333  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1934.3  on 2333  degrees of freedom
Residual deviance: 1517.7  on 2324  degrees of freedom
AIC: 1537.7

Number of Fisher Scoring iterations: 6
```

We are rejecting fullModel due to multi-collinearity assumption of logistic regression. Selecting model_backward because its AIC score is lower than model2, fulModel and all the predictors in model_backward are statically significant.

Assumptions of logistic regressions were check to see if any violation had happened. Results are in Appendix.

Python implementation

In python, two models were trained.

1. classifier_logit_default was trained using default parameters.
2. classifier_logit_2 was trained by tuning 'C' with grid search. Scikit-learn does not support step wise regression.

classifier_logit_default

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=456, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

classifier_logit_2

```
LogisticRegression(C=3, class_weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

Results of hyper parameter tuning –

```
Best parameter: {'C': 3} Best score :0.7711101585940848
```

Model classifier_logit_2 was selected by comparing its performance on validation set.

2.2.2 Random Forest

After linear regression, random forest was trained. It was implemented in both R and python.

In both implementations random forest was first trained with default setting and the hyper parameters tuning was used to find the best parameters.

R Implementation

- In R mlr library was used for training and performance tuning.
- k-fold was selected as validation strategy with 10 iterations.

Two random forest model were trained

1. modelRFBaseline - with default parameters
2. modelRF1 - with tuned parameters

summary of modelRF1 –

```
> modelRF1$learner.model

Call:
```

```
 randomForest(formula = f, data = data, classwt = classwt, cutoff = cutoff, ntree = 429L, importance = TRUE, mtry = 6L, nodesize = 18L)
               Type of random forest: classification
                     Number of trees: 429
No. of variables tried at each split: 6

        OOB estimate of  error rate: 5.66%
Confusion matrix:
     0   1 class.error
0 1919  76  0.03809524
1   56 283  0.16519174
```

ModelRF1 was select.

Different parameters were tuned to increase the performance. These parameters were –

```
            Type len Def     Constr Req Tunable Trafo
mtry     integer   -   -    2 to 10   -    TRUE     -
nodesize integer   -   -   10 to 50   -    TRUE     -
ntree    integer   -   - 100 to 600   -    TRUE     -
```

Python Implementation

In python random forest was trained and hyperparameters optimisation was done using following parameters.

Summary of fit_randomForest

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=12345, verbose=0,
            warm_start=False)
```

```
param_dist = {'max_depth': [2,4,6,8,10],
              'bootstrap': [True, False],
              'max_features': ['auto', 'sqrt', 'log2',None],
              "criterion": ["gini", "entropy"],
              "n_estimators" : [100 ,200 ,300 ,400 ,500]
             }
```

After hyper tuning best parameters were –

```
Best Parameters using random search:
 {'n_estimators': 100, 'max_features': 'auto', 'max_depth': 10, 'criterion': 'gini', 'bootstrap': False}
Time taken in random search:  247.70
```

Variable importance was plotted. It is present in EDA section.

Chapter 3

Result and performance measure

Test data set was pre-processed in the same way as the train data was processed.

3.1 Python Implementation Results

True/1 was positive class.

Logistic Regression

classifier_logit_2 was the selected model and predictions were made on test dataset.

Confusion matrix for test data using logistic regression.

Observed/ Predicted	False	True
False	1096	347
True	39	185

Table.10 Confusion matrix for test data using logistic regression.

Classification performance metric of logistic regression in python

	Precision	Recall	F1-score
False / 0	0.76	0.97	0.85
True /1	0.83	0.35	0.49
Avg / total	0.78	0.77	0.74

Table.11 Performance matrix for test data using logistic regression.

Different performance measures for logistic regression –

```
population: 1667
P: 224
N: 1443
PositiveTest: 532
NegativeTest: 1135
TP: 185
TN: 1096
FP: 347
FN: 39
TPR: 0.825892857143
TNR: 0.759528759529
PPV: 0.347744360902
NPV: 0.96563876652
FPR: 0.240471240471
FDR: 0.652255639098
FNR: 0.174107142857
ACC: 0.768446310738
F1_score: 0.489417989418
MCC: 0.428323776007
informedness: 0.585421616672
markedness: 0.313383127422
prevalence: 0.134373125375
LRP: 3.43447663648
LRN: 0.22923048097
DOR: 14.9826350403
FOR: 0.0343612334802
```

Mean accuracy using logistic regression was

```
Mean accuracy on test set 0.768446310738
```

Test error rate using logistic regression

```
Test error rate on test set 0.143
```

Random Forest

fit_randomForest was used for prediction on test dataset.

Confusion matrix for test data using random Forest in python.

Observed/ Predicted	False	True
False	1433	10
True	67	157

Table.12 Confusion matrix for test data using random forest

Classification performance metric of random forest in python

	Precision	Recall	F1-score
False / 0	0.99	0.96	0.97
True /1	0.70	0.94	0.80
Avg / total	0.96	0.95	0.96

Table.13 Performance matrix for test data using random forest

Different performance measures for random forest –

```
population: 1667
P: 224
N: 1443
PositiveTest: 167
NegativeTest: 1500
TP: 157
TN: 1433
FP: 10
FN: 67
TPR: 0.700892857143
TNR: 0.99306999307
PPV: 0.940119760479
NPV: 0.955333333333
FPR: 0.00693000693001
FDR: 0.059880239521
FNR: 0.299107142857
ACC: 0.953809238152
F1_score: 0.803069053708
MCC: 0.788296379044
informedness: 0.693962850213
markedness: 0.895453093812
prevalence: 0.134373125375
LRP: 101.138839286
```

```
24. LRN: 0.301194422291
25. DOR: 335.792537313
26. FOR: 0.0446666666667
```

Mean accuracy using random forest –

```
1.  Mean accuracy on test set 0.953809238152
```

Test error rate using logistic regression

```
1.  Test error rate on test set 0.059
```

Result –

If we compare the results of logistic regression and random forest, random forest outperforms logistic regression in overall accuracy. Here, precision and recall are very important performance measures.

Recall - When a customer churns, how often does our classifier predict that correctly. For churned customers in logistic regression it is .35 while in random forest it is .94

Precision - When our classifier predicts a customer will churn, how often does that customer actually churn. For churned customer's Logistic regression's precision is .83 and random forest forest's is .70

But F1 score of random forest is better than logistic algorithm.

As we are trying to find which customers are going to churn out**, true positive rate** is also important metric. For logistic regression, it is .82 while for random forest it is .70. Here logistic regression is better than random forest.

Another important factor for consideration is that, these classifications are based on the probabilities of .5 for each target case. Probability threshold or cut-off is a business context decision. The threshold is usually set 0 .5 by default. This means that anyone with a probability of more than .5 is predicted to churn. If we reduce the probability threshold, more people will be predicted to churn. If we are interested in customers which have most likely to churn, we can increase the probability threshold.

So the threshold or probability limit is decided based on business decision/requirement, marketing budget etc.

After comparing all the required performance measure, random forest was performing overall better than logistic regression.

3.2 R Implementations results

For comparing performance of logistic regression and random forest in R, probability threshold .7 and 0.3 were used. These cut off values were estimated while implementing random forest trying with different combinations.

Here positive class is False/0. Since False/0 is the positive class, **specificity** is a very important metric here. Specificity is the proportion of actual negatives that are correctly identified and our negative class is the customers who actually churned.

Logistic Regression

Result and summary of logistic regression

```
Confusion Matrix and Statistics

        predicted
observed    0    1
       0 1284  159
       1  109  115

               Accuracy : 0.8392
                 95% CI : (0.8207, 0.8566)
    No Information Rate : 0.8356
    P-Value [Acc > NIR] : 0.360535

                  Kappa : 0.3685
 Mcnemar's Test P-Value : 0.002761

            Sensitivity : 0.9218
            Specificity : 0.4197
         Pos Pred Value : 0.8898
         Neg Pred Value : 0.5134
             Prevalence : 0.8356
         Detection Rate : 0.7702
   Detection Prevalence : 0.8656
      Balanced Accuracy : 0.6707

       'Positive' Class : 0
```

Confusion matrix

Observed/ Predicted	False	True
False	1284	159
True	109	115

Table.14 Confusion matrix of logistic regression

Random Forest

Result and summary of random forest

```
Confusion Matrix and Statistics

        predicted
observed    0    1
       0 1404   39
       1   43  181

               Accuracy : 0.9508
                 95% CI : (0.9393, 0.9607)
```

```
     No Information Rate : 0.868
     P-Value [Acc > NIR] : <2e-16

                   Kappa : 0.7869
  Mcnemar's Test P-Value : 0.7404

             Sensitivity : 0.9703
             Specificity : 0.8227
          Pos Pred Value : 0.9730
          Neg Pred Value : 0.8080
              Prevalence : 0.8680
          Detection Rate : 0.8422
    Detection Prevalence : 0.8656
       Balanced Accuracy : 0.8965

        'Positive' Class : 0
```

Confusion matrix of prediction done using random forest

Observed/ Predicted	False	True
False	1404	39
True	43	181

Table.15 Confusion matrix for random forest

Results-

Comparing results of logistic regression –

Metric / Algorithm	Logistic Regression	Random Forest
Accuracy	.83	.95
Sensitivity	.92	.97
Specificity	.41	.82
Pos Pred Value	.88	.97
Neg Pred Value	.51	.80
Precision	.88	.97
Recall	.92	.97
F1	.90	.97
Prevalence	.83	.86
Detection Rate	.77	.84
Detection Prevalence	.86	.86
Balanced Accuracy	.67	.89

Table.16 Comparing Logistic regression and random forest performance

As we can see that random forest is outperforming logistic regression in every performance metric. We were interested in classifying which customers were going to churned. Here Specificity tells us how well our classifier is predicting, which customers are churning out. Logistic regression's specificity is .41 while random forest's is .82.

Random forest's accuracy is .95 which is way better than logistic regression. Also from the confusion matrix, we can see that false positive rate is very less for random forest.

So random forest is a better classifier for this problem statement.

Appendix A

Logistic regression diagnostics

```
#Logistic regression diagnostics - now we are going to check for logistic regression assumptions

#1. Linearity assumption
# Checking for a linear relationship between continous variables and logit of the outcome.This
#can be done by visually inspecting the scatter plot between each predictor and logit valuees.

# selecting continous predictors
continous_data <- test_dataset %>%
  select_if(is.numeric)
predictors <- colnames(continous_data)

# calculating logit and adding it in 'continous_data'
continous_data <- continous_data %>%
  mutate(logit = log(pred_backmodel/(1-pred_backmodel))) %>%
  gather(key = "predictors", value = "predictor.value", -logit)
ggplot(data = continous_data, mapping = aes(x = logit, predictor.value)) +
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth(method = "loess") +
  theme_bw() +
  facet_wrap(~predictors, scales = "free_y")

# The smoothed scatter plots show all the continous predictors are very near
# linearly associated with the outcome in logit scale in figure 11.
```

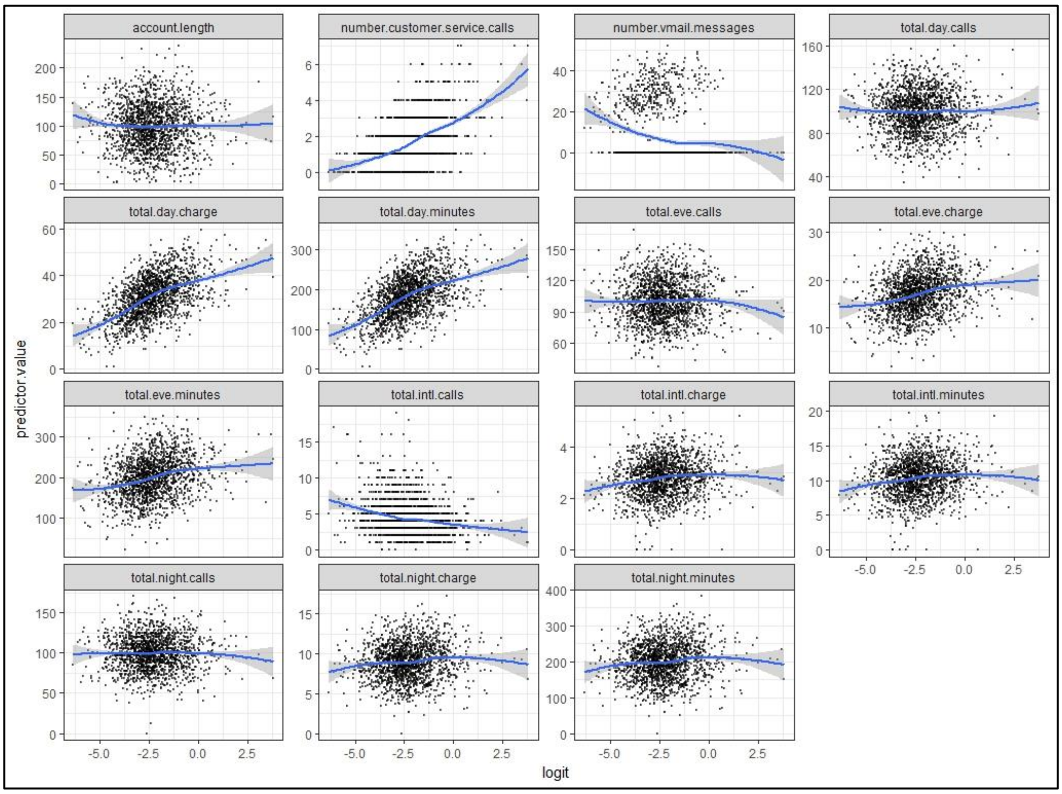

Figure 11 – Checking for linearity assumption

```
# 2. Multicollinearity
# we have removed multi-collinear variables while models.
#performing double check using variation inflation factor (VIF)
car::vif(model_backward)
#As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of
#collinearity. In our model voice.mail.plan and number.vmail.messages is showing vif score of
# < 16. SO, we will remove one of the variable and retrain and check the final model after
#dignostics.
> car::vif(model_backward)
           international.plan               voice.mail.plan         number.vmail.messages
                     1.050493                     16.255768                     16.219928
             total.day.minutes             total.eve.minutes            total.night.charge
                     1.045576                      1.031209                      1.008662
              total.intl.calls             total.intl.charge number.customer.service.calls
                     1.012671                      1.012673                      1.076539
#3. Influential values
# top 3 largest values
plot(model_backward, which = 4, id.n = 3)
#278, 1862, 3292
```

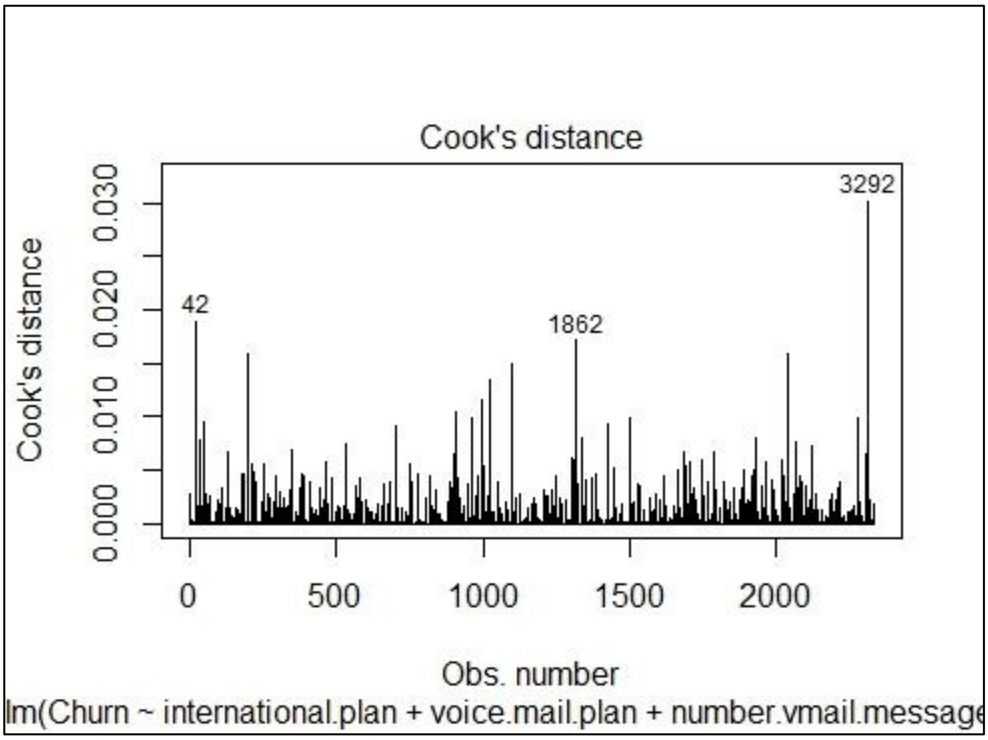

Fig12. Identifying influential & extreme points

```
# Data points with an absolute standardized residuals above 3 represent possible outliers
#and may deserve closer attention
#augment - add columns to the original dataset such as predictions, residuals and cluster assignments
```

```

model.data <- augment(model_backward) %>%
  mutate(index = 1:n())
model.data %>% top_n(3, .cooksd)
# plotting standardized residuals
ggplot(model.data, aes(index, .std.resid)) +
  geom_point(aes(color = Churn), alpha = .5) +
  theme_bw()
#
model.data %>%
  filter(abs(.std.resid) > 3)
# one possible influential observations was found on our training set.
# row 1890
```

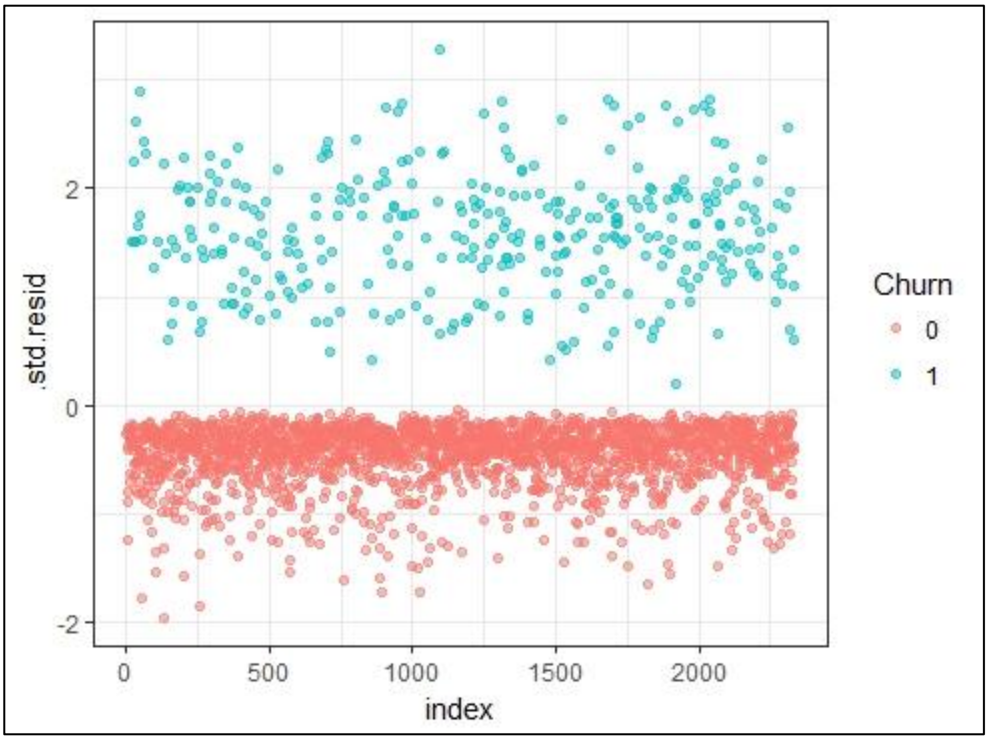

Fig13. standardized residuals

R code

ChurnReductionEDA.R

```
# Customer churn reduction EDA

rm(list=ls())
#loading requried libraries

library(dplyr)
library(ggplot2)
library(stringr)
library(corrplot)


fillColor = "#FFA07A"
fillColorRed = "#56B4E9"

train_data <-
  read.csv("Train_data.csv",
           sep = ',',
           na.strings = c(" ", "NA"))
```

```r

#looking at dimensions
dim(train_data)

#Train data set consist of 3333 observations and 21 varaiables

#checking structure of dataset
str(train_data)

# Visualizing target class freqquencies
train_data %>%
  count(Churn) %>%
  ggplot(aes(x = Churn,
             y = n)) +
  geom_bar(stat = 'identity',
           colour = "white",
           fill = fillColor) +
  labs(x = 'Churn rate', y = 'count ', title = 'Customer churn count') +
  theme_bw()

table(train_data$Churn)
#Looking at the frequencies of churn , it is not looking like highly imbalance problem.

# Now looking for any missing values
sapply(train_data, function(x) {
  sum(is.na(x))
}) # There are no missing values in dataset



#selecting numeric variables
numCols <- unlist(lapply(train_data,is.numeric))
numVarDataset <- train_data[,numCols]

# Visualizing correlation
par(mfrow = c(1, 1))
corr <- cor(numVarDataset)
corrplot(
  corr,
  method = "color",
  outline = TRUE,
  cl.pos = 'n',
  rect.col = "black",
  tl.col = "indianred4",
  addCoef.col = "black",
  number.digits = 2,
  number.cex = 0.60,
  tl.cex = 0.70,
  cl.cex = 1,
  col = colorRampPalette(c("green4", "white", "red"))(100)
)

# From corrplot we can see that dataset consist of multicollinearity
# total.day.minutes and total.day.charge are highly collinear
# total.eve.minutes and total.eve.charge are highly collinear
# total.night.minutes and total.night.charge are highly collinear
# total.intl.minutes and total.intl are highly collinear
# we can exclude one of these predictors later during modeling

############## Generic EDA function for continous variables
plot_continous <- function(dataset, variable,targetVariable) {
  var_name = eval(substitute(variable), eval(dataset))
  target_var = eval(substitute(targetVariable), eval(dataset))
  par(mfrow = c(1, 2))
  print(summary(var_name))
```

```
  print(summary(target_var))
  possible_outliers <- (boxplot.stats(var_name)$out)
  print(possible_outliers)
  print(paste("Total possible outliers", length(possible_outliers)))
  table(possible_outliers)
  ggplot(train_data, aes(target_var, var_name, fill = target_var)) +
    geom_boxplot(alpha = 0.8) + theme(legend.position = "null")
}


############################### looking at 'state' variable. It is a factor variable
train_data %>%
  count(state) %>%
  ggplot(mapping = aes(x = state, y = n)) +
  geom_bar(stat = 'identity',
           colour = 'white',
                   fill = fillColor) +
          labs(x = "states", y = "count", "Customers per state") +
          coord_flip()
        # Fom the plot we can that maximum customers are from west vergenia and lowest are from California




        # looking at each variable
        plot_continous(train_data, account.length,Churn)
        # As we can see, that there are some possible outliers but they are not very extreme. Ignoring them


        ##################################### analysing international.plan #################################
        str(train_data$international.plan) # it is a categorical variable
        table(train_data$international.plan)
        train_data %>%
          count(international.plan) %>%
          ggplot(mapping = aes(x = international.plan, y = n)) +
          geom_bar(stat = 'identity',
                   colour = 'white',
                   fill = fillColor)
        # From the plot we can see that most customers dont have international plan.


        # next examining for the churn rate percentage of customers with national and internation plan

        national_cust_churnRate <- train_data %>%
          select(international.plan, Churn) %>%
          filter(str_detect(international.plan, "no")) %>%
          group_by(Churn) %>%
          summarise (n = n()) %>%
          mutate(percantage = (n / sum(n)) * 100)
        #Only 11.49 % customer with national plan churn out.


        international_cust_churnRate <- train_data %>%
          select(international.plan, Churn) %>%
          filter(str_detect(international.plan, "yes")) %>%
          group_by(Churn) %>%
          summarise (n = n()) %>%
          mutate(percantage = (n / sum(n)) * 100)
        # 42.42 %  customers with international plan had churn out. It means that the telecom company
        # is mainly loosing customers with internation plans.
```

```
######################################Analysing voice.mail.plan #####################################

table(train_data$voice.mail.plan)

# customers with voice plan and their churn rate
voice_plan_churnRate <- train_data %>%
  select(voice.mail.plan, Churn) %>%
  filter(str_detect(voice.mail.plan, "yes")) %>%
  group_by(Churn) %>%
  summarise(n = n()) %>%
  mutate(churnRatePercentage = (n / sum(n)) * 100)

ggplot(data = voice_plan_churnRate,
       mapping = aes(x = Churn, y = churnRatePercentage)) +
  geom_bar(stat = 'identity',
           colour = 'white',
           fill = fillColorRed) +
  labs(title = 'Voice main plan customers churn rate')

# 922 customers have voice mail plan and 80 (8.68 %) customers out of 922  churn out.

#customers without voice plan and their churn rate
non_voice_plan_churnRate <- train_data %>%
  select(voice.mail.plan, Churn) %>%
  filter(str_detect(voice.mail.plan, "no")) %>%
  group_by(Churn) %>%
  summarise(n = n()) %>%
  mutate(churnRatePercentage = (n / sum(n)) * 100)

ggplot(data = non_voice_plan_churnRate,
       mapping = aes(x = Churn, y = churnRatePercentage)) +
  geom_bar(stat = 'identity',
           colour = 'white',
           fill = fillColor) +
  labs(title = 'Non voice plan Customer churn rate')

# 2411 customers dont have voice mail plan  and 403 (16.7 %) out of 2411 churn out

#So customers without voice plan have higher churn rate



# removing parameters that dosn't seem to be logical parameter for customer churn.
#So removing state, area code and phone number
train_data$state <- NULL
train_data$area.code <- NULL
train_data$phone.number <- NULL


############################ Analysing number.vmail.messages ################################
str(train_data$number.vmail.messages)
plot_continous(train_data, number.vmail.messages,Churn)

# no extreme outliers detected.

############################ Analysing total.day.minutes ################################
str(train_data$total.day.minutes)
plot_continous(train_data, total.day.minutes,Churn)

# no extreme outliers detected
```

```
############################ Analysing total.day.calls ################################
str(train_data$total.day.calls)
plot_continous(train_data, total.day.calls,Churn)

# no extreme outliers detected

############################ Analysing total.day.charge ################################
str(train_data$total.day.charge)
plot_continous(train_data, total.day.charge, Churn)

# no extreme outliers detected

############################ Analysing total.eve.minutes ################################
str(train_data$total.eve.minutes)
plot_continous(train_data, total.eve.minutes, Churn)

# no extreme outliers detected

############################ Analysing total.eve.calls ################################
str(train_data$total.eve.calls)
plot_continous(train_data, total.eve.calls, Churn)

# no extreme outliers detected

############################ Analysing total.eve.charge ################################
str(train_data$total.eve.charge)
plot_continous(train_data, total.eve.charge, Churn)

# no extreme outliers detected

############################ Analysing total.night.minutes ################################
str(train_data$total.night.minutes)
plot_continous(train_data, total.night.minutes, Churn)

# no extreme outliers detected

############################ Analysing total.night.calls ################################
str(train_data$total.night.calls)
plot_continous(train_data, total.night.calls, Churn)

# no extreme outliers detected

############################ Analysing total.night.charge ################################
str(train_data$total.night.charge)
plot_continous(train_data, total.night.charge, Churn)

# no extreme outliers detected

############################ Analysing total.intl.minutes ################################
str(train_data$total.intl.minutes)
plot_continous(train_data, total.intl.minutes, Churn)

# no extreme outliers detected

############################ Analysing total.intl.calls ################################
```

```
        str(train_data$total.intl.calls)
        plot_continous(train_data, total.intl.calls, Churn)

        # no extreme outliers detected

        ############################ Analysing total.intl.charge ################################

        str(train_data$total.intl.charge)
        plot_continous(train_data, total.intl.charge, Churn)

        # no extreme outliers detected

        ######################## Analysing number.customer.service.calls ############################

        str(train_data$number.customer.service.calls)

        plot_continous(train_data , number.customer.service.calls, Churn)
        table(train_data$number.customer.service.calls)
```

inputPrep.R

```
# preparing train,validation and test dataset for random forest and logistic regression

library(caret)
library(stringr)

# reading train set
inputdata <- read.csv("Train_data.csv", sep = ',', header = TRUE, na.strings = c(" ","NA"))
inputTest <- read.csv("Test_data.csv", sep = ',', header = TRUE, na.strings = c(" ","NA"))

#removing variables which not helpfull in churn reduction. Using these variables dosn't make sense.
#for training data
inputdata$state <- NULL
inputdata$area.code <- NULL
inputdata$phone.number <- NULL

#for test data
inputTest$state <- NULL
inputTest$area.code <- NULL
inputTest$phone.number <- NULL

# We are going to implement random forest and logistic regresion and compare performance of
# both models. For comparision we need same training and test data.
# random forest can use input both character and numeric data but logistic regression can only
# work on numeric data.
# Transforming and re-encoding the data so that both random forest and logistic regression can train and predict
# on same dataset.

#for training data
inputdata[,'international.plan'] <- ifelse(str_detect(inputdata[,'international.plan'],"yes"),1,0)
inputdata$international.plan <- as.factor(inputdata$international.plan)
inputdata[,'voice.mail.plan'] <- ifelse(str_detect(inputdata[,'voice.mail.plan'],"yes"),1,0)
inputdata$voice.mail.plan <- as.factor(inputdata$voice.mail.plan)
inputdata[,"Churn"] <- str_replace(inputdata[,"Churn"],"\\.","")
inputdata[,"Churn"] <- ifelse (str_detect(inputdata[,"Churn"],"True"),1,0)
```

```
inputdata$Churn <- as.factor(inputdata$Churn)

#for test data
inputTest[,'international.plan'] <- ifelse(str_detect(inputTest[,'international.plan'],"yes"),1,0)
inputTest$international.plan <- as.factor(inputTest$international.plan)
inputTest[,'voice.mail.plan'] <- ifelse(str_detect(inputTest[,'voice.mail.plan'],"yes"),1,0)
inputTest$voice.mail.plan <- as.factor(inputTest$voice.mail.plan)
inputTest[,"Churn"] <- str_replace(inputTest[,"Churn"],"\\.","")
inputTest[,"Churn"] <- ifelse (str_detect(inputTest[,"Churn"],"True"),1,0)
inputTest$Churn <- as.factor(inputTest$Churn)

set.seed(987)
# spliting input train data into train and validation set
churnIndex <- createDataPartition(inputdata$Churn, p = 0.7, times = 1, list = FALSE)
train_data <- inputdata[churnIndex,]
validation_data <- inputdata[churnIndex,]

test_dataset <- inputTest
```

logesticRegression.R

```
# Churn Reduction Logistic regression implementation

library(stringr)
library(corrplot)
library(caret)
library(dplyr)
library(tidyr)
library(broom)



# One of the assumptions of logistic regression is that there is no high intercorrelations
#among predictors.
# From EDA we found some variables have multi-collinearity



# Now we will build our model.
# For the first model we will select all the predictors

fullModel <- glm(Churn ~., data = train_data, family = binomial(link = 'logit'))
summary(fullModel)

# Predicting and checking model performance with full model
pred_fullmodel_val <- predict(fullModel, validation_data[,-18], type = "response")
pred_fullmodel_valT <- ifelse(pred_fullmodel_val < 0.5,1,0)
xtab=table(observed=validation_data[,18],predicted=pred_fullmodel_valT)
fullmodel_confmat_val <- confusionMatrix(xtab)
print(fullmodel_confmat_val)

# now in second model we will remove variables affected by multi-collinearity
model2 <- glm(Churn ~. -total.day.minutes-total.eve.minutes-
                    total.night.minutes-total.intl.minutes,
                  data = train_data, family = binomial(link = 'logit'))
summary(model2)

# Predicting and checking model performance with model2
pred_model2_val <- predict(model2, validation_data[,-18], type='response')
pred_model2_val <- ifelse(pred_model2_val > 0.5,1,0)
```

```
xtab1 <- table(observed = validation_data[,18], predicted = pred_model2_val)
model2_confmat_val <- confusionMatrix(xtab1)

# if we compare fullModel and model2 summary, we can see that
# In full model due to variables with multi-collineary only international.plan, voice.mail.plan,
# number.customer.service.call are considered statistically significant with AIC score 1540.

# After removing highly collinear variables in model2, variables nternational.plan, voice.mail.plan,
# number.customer.service.call along with total.day.charge, total.eve.charge, total.night.charge,
# and total.intl.charge are considered statistically significant with AIC score 1543.

# model with lower AIC score is considered better

# NOw we will use Stepwise Procedures

#1 backwards elimination
backward <- step(fullModel)
# from backwards elimination approach, we got a model with AIC 1529.02 which is better
#than model2
# training model using best result from 'backward'
model_backward <- glm(Churn ~ international.plan+voice.mail.plan+number.vmail.messages+
                        total.day.minutes+total.eve.minutes+total.night.charge+
                        total.intl.calls+total.intl.charge+number.customer.service.calls,
                      data = train_data, family = binomial(link = 'logit'))
summary(model_backward)

#predicting on validation set
pred_backmodel_val <- predict(model_backward, validation_data[,-18], type = 'response')
pred_backmodel_valT <- ifelse(pred_backmodel_val > 0.5,1,0)
xtab2 <- table(observed = validation_data[,18], predicted = pred_backmodel_valT)
backward_confmat_val <- confusionMatrix(xtab2)



# model_backward is slightly better than model2

# 2. forward elimination
nullModel <- glm(Churn ~ 1, data =train_data, family = binomial)
summary(nullModel)

forward <- step(nullModel, scope = list(lower =formula(nullModel), upper = formula(fullModel)),
                direction = 'forward')

formula(backward)
formula(forward)
# both the steps are giving the same formula with same AIC score ie. 1537.7

# We are rejecting fullModel due to multi-collinearity assumption of logistic regression.
# so selecting model_backward because its AIC score is lower than model2 and all the predictors
# in model_backward are stastically significant.



#predicting using model_backward on test set
```

```
pred_backmodel <- predict(model_backward, test_dataset[,-18], type = 'response')
pred_backmodelT<- ifelse(pred_backmodel > 0.5 ,1,0)
xtab_backmodel <- table(observed = test_dataset[,18], predicted = pred_backmodelT)

backmodel_confmat <- confusionMatrix(xtab_backmodel)
backmodel_performance <- data.frame(backmodel_confmat$byClass)
backmodel_performance <- rbind(accuracy = backmodel_confmat$overall, backmodel_performance)
#comparing model_backward performance on validation and test set for overfitting
z<- as.data.frame(rbind(backmodel_confmat$byClass,backward_confmat_val$byClass))
#model_backward is preforming with acceptable variation on both dataset, so no overfitting.


# ploting histogram of prediction
pred_hist <- data.frame(pred_backmodel)
pred_hist %>%
  ggplot(mapping = aes(x = pred_backmodel)) +
  geom_histogram(bins = 50, fill = 'grey40') +
  labs(title = " Prediction histograms")

# range of predictions
round(range(pred_backmodel),2)
median(pred_backmodel)
#The prediction range from 0 to 0.97 with median 0.0872


# Selecting  probablity threshold value is business context decision and a
# tradoff between  true positve and false positive classifications
# The threshold  here is set to .5 . This means that anyone with a probability of
# more than .5 is predicted to churn. If we reduce the probability threshold, more people will
# be predicted to churn, this gives us a higher number of “at risk customers” to target.
# However, this increases the likelihood that customers who are not at risk will pass the
# threshold and be predicted to churn.
# If we are concerened with marketing expenditure, then a higher threshold should be
# targeted (above 0.8 or 0.9). Otherwise, lower thresholds can be targeted,
# so the company can target larger amounts of customers who are at risk of churning.


#Logistic regression diagnostics - now we are going to check for logistic regression assumptions

#1. Linearity assumption
# Checking for a linear relationship between continous variables and logit of the outcome.This
#can be done by visually inspecting the scatter plot between each predictor and logit valuees.

# selecting continous predictors
continous_data <- test_dataset %>%
  select_if(is.numeric)
predictors <- colnames(continous_data)

# calculating logit and adding it in 'continous_data'
continous_data <- continous_data %>%
  mutate(logit = log(pred_backmodel/(1-pred_backmodel))) %>%
```

```
  gather(key = "predictors", value = "predictor.value", -logit)
ggplot(data = continous_data, mapping = aes(x = logit, predictor.value)) +
  geom_point(size = 0.5, alpha = 0.5) +
  geom_smooth(method = "loess") +
  theme_bw() +
  facet_wrap(~predictors, scales = "free_y")

# The smoothed scatter plots show all the continous predictors are very near

# linearly associated with the outcome in logit scale.

# 2. Multicollinearity
# we have removed multi-collinear variables while models.
#performing double check using variation inflation factor (VIF)
car::vif(model_backward)
#As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of
#collinearity. In our model voice.mail.plan and number.vmail.messages is showing vif score of
# < 16. SO, we will remove one of the variable and retrain and check the final model after
#dignostics.

#3. Influential values
# top 3 largest values
plot(model_backward, which = 4, id.n = 3)
#278, 1862, 3292
# Data points with an absolute standardized residuals above 3 represent possible outliers
#and may deserve closer attention
#augment - add columns to the original dataset such as predictions, residuals and cluster assignments

model.data <- augment(model_backward) %>%
  mutate(index = 1:n())
model.data %>% top_n(3, .cooksd)
# plotting standardized residuals
ggplot(model.data, aes(index, .std.resid)) +
  geom_point(aes(color = Churn), alpha = .5) +
  theme_bw()
#
model.data %>%
  filter(abs(.std.resid) > 3)
# one possible influential observations was found on our training set.
# row 1890

#-----------------------------------------------------------------------------------
# creating a final model by removing one of the variables detected by VIF and removing
# one influential observations

train_data <- train_data[-1890,]
formula(model_backward)

final_model <- glm(Churn ~ international.plan + voice.mail.plan +
                     total.day.minutes + total.eve.minutes + total.night.charge +
                     total.intl.calls + total.intl.charge + number.customer.service.calls,
                   data = train_data, family = binomial(link = 'logit'))

#checking VIF
car::vif(final_model)
# no multicollinear variable found in model

```

```
        pred_final <- predict(final_model, test_dataset, type = 'response')
        pred_finalT <- ifelse(pred_final>0.5, 1,0)
        xtab_final <- table(observed = test_dataset[,18], predicted = pred_finalT)
        final_confmat <- confusionMatrix(xtab_final)
        finalModel_performance <- data.frame(final_confmat$byClass)
        finalModel_performance <- rbind(accuracy = final_confmat$overall, finalModel_performance)



        # After comparing confusion matrix of model_backward and final_modal, we found that
        # both models are performing very similar.

        #------------------------------------------------------------------------------------

        # for comparision with random forest performance with probabilty threshlod .75,.25

        pred_compareRF <- predict(final_model, test_dataset[,-18], type = 'response')
        pred_compareRFT <- ifelse(pred_compareRF>0.25, 1,0)

        xtab_compare <- table(observed = test_dataset[,18], predicted = pred_compareRFT)
        confMatLogit <- confusionMatrix(xtab_compare)
        compare_performance <- data.frame(confMatLogit$byClass)
        compare_performance <- rbind(accuracy = confMatLogit$overall, compare_performance)
        print(confMatLogit)
        print(compare_performance)


        # Model input and output for logistic regression
        write.csv(test_dataset, file = "InputLogisticRegressionR.csv")
        write.csv(pred_compareRFT, file="outputLogisticRegressionR.csv")
```

randomForestMLR.R

```
# Randomforest and parameter tuning unis MLR package

library(mlr)
#library(caret)


trainTask <- makeClassifTask(data = train_data, target = "Churn")
validationTask <- makeClassifTask(data = validation_data, target = "Churn")

#creating randomoforest classifier/learner
randomFOrest.learner.baseline <- makeLearner("classif.randomForest")
randomFOrest.learner <- makeLearner("classif.randomForest")


# setting validaion staertgy using cv with 10 folds
cvstr <- makeResampleDesc("CV", iters = 10L)

# Our main aim is churn reduction. So we are interest in a model which can classify customers
# who are going to churn out. There for model should reduce false positive rate. As positve class
#for the model is 'false'


randomFOrest.learner$par.vals <-
```

```
  list(ntree = 500L,
       importance = TRUE,
       cutoff = c(0.70, 0.30))

r <- resample(
  learner = randomFOrest.learner,
  task = trainTask,
  resampling = cvstr,
  measures = list(tpr, fpr, fnr, fpr, acc),
  show.info = TRUE
)

# Probablity threshold or cutoff is a bussiness context decision. The threshold is usually set
#to .5 by default. This means that anyone with a probability of more than .5 is predicted to churn.
#If we reduce the probability threshold, more people will be predicted to churn
# to reduce the false positve rate , we are tuning cut-off here.
# cut-off  tuning is perfomed by cross-validation
# Tuning cutoff value starting with default .50,.50
#.70 and .30 is selected as cutoff value
# Main purpose here is tuning cutoff.
cutoff1 <- calculateConfusionMatrix(r$pred)

# tuning mtry ,ntree, nodesize using hypertuning
params <- makeParamSet(
  makeIntegerParam("mtry", lower = 2, upper = 10),
  makeIntegerParam("nodesize", lower = 10, upper = 50),
  makeIntegerParam("ntree", lower = 100, upper = 600)
)

#random search with 100 iterations
ctrl <- makeTuneControlRandom(maxit = 100L)

tuneRF <- tuneParams(
  learner = randomFOrest.learner,
  task = trainTask,
  resampling = cvstr,
  measures = list(acc),
  par.set = params,
  control = ctrl,
  show.info = TRUE
)

tunedRFmodel <- setHyperPars(learner = randomFOrest.learner,
                             par.vals = tuneRF$x)

# tuning with default random forest
modelRFBaseline <- mlr::train(randomFOrest.learner.baseline, trainTask)

# training random forest with tuned parameters
modelRF1 <- mlr::train(tunedRFmodel, trainTask)

#prediction on validation set using basline
predictBaseline <- predict(modelRFBaseline, validationTask)

#performance measures for baseline model on validation set
accuracy_bl <- performance(pred = predictBaseline, measures = acc)
f1Measure_bl <- performance(pred = predictBaseline, measures = f1)
truePositveRate_bl <-
  performance(pred = predictBaseline, measures = tpr)
trueNegativeRate_bl <-
  performance(pre = predictBaseline, measures = tnr)
falsePositiveRate_bl <-
  performance(pred = predictBaseline, measures = fpr)
falseNegativeRate_bl <-
```

```
  performance(pred = predictBaseline, measures = fnr)

baselinePerformanceMetric <-
  c(
    accuracy_bl,
    f1Measure_bl,
    truePositveRate_bl,
    trueNegativeRate_bl,
    falsePositiveRate_bl,
    falseNegativeRate_bl
  )



      #prediction on validation set using tuned random forest
      predictRF1 <- predict(modelRF1, validationTask)

      #performance measures for tuned model on validation set
      accuracy_val <- performance(pred = predictRF1, measures = acc)
      f1Measure_val <- performance(pred = predictRF1, measures = f1)
      truePositveRate_val <-
        performance(pred = predictRF1, measures = tpr)
      trueNegativeRate_val <-
        performance(pre = predictRF1, measures = tnr)
      falsePositiveRate_val <-
        performance(pred = predictRF1, measures = fpr)
      falseNegativeRate_val <-
        performance(pred = predictRF1, measures = fnr)


      validationPerformanceMetric <-
        c(
          accuracy_val,
          f1Measure_val,
          truePositveRate_val,
          trueNegativeRate_val,
          falsePositiveRate_val,
          falseNegativeRate_val
        )





      mainTestTask <- makeClassifTask(data = test_dataset, target = "Churn")


      #prediction on  main testset using tuned RF

      predictTest <- predict(modelRF1, mainTestTask)
      confusionMatrixTest <- calculateConfusionMatrix(predictTest)

      #calculating performance measure for predictions
      accuracy <- performance(pred = predictTest, measures = acc)
      f1Measure <- performance(pred = predictTest, measures = f1)
      truePositveRate <- performance(pred = predictTest, measures = tpr)
      trueNegativeRate <- performance(pre = predictTest, measures = tnr)
      falsePositiveRate <- performance(pred = predictTest, measures = fpr)
      falseNegativeRate <- performance(pred = predictTest, measures = fnr)

      testPerformanceMetric = c(
        accuracy,
        f1Measure,
        truePositveRate,
        trueNegativeRate,
        falsePositiveRate,
```

```
        falseNegativeRate
        )



      xt <- table(observed = test_dataset[,18], predicted = predictTest$data$response)
      final_confmatRF <- confusionMatrix(xt)
      compare_performanceRF <- data.frame(final_confmatRF$byClass)
      compare_performanceRF <- rbind(accuracy = final_confmatRF$overall, compare_performanceRF)
      print(final_confmatRF)
      print(compare_performanceRF)

      # Model input and output for Random Forest
      write.csv(test_dataset, file = "InputRandomForestR.csv")
      write.csv(predictTest$data$response, file="outputRandomForestR.csv")
```

Python code

Customer Churn Reduction - logistic Regression.py

```
# coding: utf-8

# # Customer Churn Reduction - Logistic Regression
# In the notebook, we will use logistic regression to predict whether a customer will churn or not.

# ### importing librarires

# In[1]:


import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from plotnine import *
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, RandomizedSearchCV, GridSearchCV
from sklearn.metrics import classification_report
from pandas_ml import ConfusionMatrix


# In[2]:


# Read in train and test data
train_data = pd.read_csv("Train_data.csv")
test_data = pd.read_csv("Test_data.csv")


# In[3]:


# change column names for ease of use and display first 5 rows
train_data.columns = train_data.columns.str.lower().str.replace(' ','_')
test_data.columns = test_data.columns.str.lower().str.replace(' ','_')


# In[4]:


```

```
print(" number of rows and columns in train data ",train_data.shape)


# In[5]:


print(" number of rows and columns in test data ",test_data.shape)



# In[6]:


train_data.describe()
train_data.head()


# In[7]:


#removing variables which not helpfull in churn reduction. Using these variables doesn't make sense.
#for training data
train_data = train_data.drop(['state','area_code','phone_number'], axis = 1)
test_data = test_data.drop(['state','area_code','phone_number'], axis = 1)


# In[8]:


# checking for missing values in train
train_data.isnull().sum()


# In[9]:


# checking for missing values in test dataset
test_data.isnull().sum()


# ##### There are no missing values in train and test data.

# Now looking at target variable.
# churn: This is the target variable. Churn is defined as whether the customer leaves the services or not. churn = True means customer left ,churn = false means customer stays

# In[10]:


plt.figure(figsize=(8,6))
sns.set_style('ticks')
sns.countplot(train_data.churn,palette='summer')
plt.xlabel('Customer churn')


# In[11]:


# churn ratio of customers with and without internation plan
       ggplot(train_data) +     aes('international_plan', fill = 'churn') +    geom_bar(position = "fill", color= 'blue') + labs(x = "International plan", y = "") +     ggtitle("Churn ratio of customers with and without international plan") +     theme(figure_size=(6, 4))

```

```
# #### Customers with international plan are churning out more as compare to domestic customers.

# In[12]:


# churn ratio of customers with voice_mail_plan
ggplot(train_data) +     aes('voice_mail_plan', fill = 'churn') +     geom_bar(position = "fill", color = 'blue') +     labs(x =  "voice mail plan", y = "") +     ggtitle("Churn ratio with customers with and without voice mail plan") +     theme(figure_size=(6, 4))


# #### Customers without voice mail plan are churning out more as compare to customers with voice mail plan.

# In[13]:


# # churn ratio of customers with respect to service calls
ggplot(train_data) +     aes('number_customer_service_calls', fill = 'churn') +     geom_bar(position = "fill", color = 'blue') +     labs(x = "Customer service calls", y = "")  +     ggtitle(" Churn ratio with service call frequency") +     theme(figure_size=(6, 4))


# #### customers with higher service calls ie > 3 are churning out more.

# ### 3. Correlation matrix for continous predictors

# In[14]:


churn_corr = train_data.corr()
cmap = cmap=sns.diverging_palette(5, 250, as_cmap=True)

def magnify():
    return [dict(selector="th",
                 props=[("font-size", "7pt")]),
            dict(selector="td",
                 props=[('padding', "0em 0em")]),
            dict(selector="th:hover",
                 props=[("font-size", "12pt")]),
            dict(selector="tr:hover td:hover",
                 props=[('max-width', '200px'),
                        ('font-size', '12pt')])
]

churn_corr.style.background_gradient(cmap, axis=1)    .set_properties(**{'max-width': '90px', 'font-size': '10pt'})    .set_caption("Correlation matrix")    .set_precision(2)    .set_table_styles(magnify())


# ### From corrplot we can see that dataset consist of multicollinearity
# 1. total.day.minutes and total.day.charge are highly collinear
# 2. total.eve.minutes and total.eve.charge are highly collinear
# 3. total.night.minutes and total.night.charge are highly collinear
# 4. total.intl.minutes and total.intl.charge are highly collinear
#
# Multi-collinearity voilates the assumption of logistic regression. So we will be removing one of these predictors
# from the model.

```

```
# ### 4. Exploring continous predictors

# In[15]:


# function for exploring distributions by continuous predictors with there summary stats
def countPred_eda(train_data, variableName, targetVariable):
    print(train_data[variableName].describe())
    return ggplot(train_data) +     aes(targetVariable, variableName, fill = targetVariable) + geom_boxplot(alpha = .8, outlier_color = "green") +     labs(x = targetVariable, y = variableName) +     ggtitle("Churn ratio with "+ variableName) +     theme(figure_size=(6, 4))




# In[16]:


# --- total_day_minutes --- #
countPred_eda(train_data,'total_day_minutes','churn')


# #### It is evident from above plot that churn rate is higher when count of total_day_minute is higher

# In[17]:


# --- total_day_calls ---- #
countPred_eda(train_data,'total_day_calls','churn')


# In[18]:


# --- total_day_charge --- #
countPred_eda(train_data,'total_day_charge','churn')


# In[19]:


# --- total_eve_minutes --- #
countPred_eda(train_data,'total_eve_minutes','churn')


# In[20]:


# --- total_eve_calls --- #
countPred_eda(train_data,'total_eve_calls','churn')


# In[21]:


# --- total_eve_charge --- #
countPred_eda(train_data,'total_eve_charge','churn')


# In[22]:


# --- total_night_minutes --- #
```

```
        countPred_eda(train_data,'total_night_minutes','churn')


        # In[23]:


        # --- total_night_calls --- #
        countPred_eda(train_data,'total_night_calls','churn')


        # In[24]:


        # --- total_night_charge --- #
        countPred_eda(train_data,'total_night_charge','churn')


        # In[25]:


        # --- total_intl_minutes --- #
        countPred_eda(train_data,'total_intl_minutes','churn')


        # In[26]:


        # --- total_intl_calls --- #
        countPred_eda(train_data,'total_intl_calls','churn')


        # In[27]:


        # --- total_intl_charge --- #
        countPred_eda(train_data,'total_intl_charge','churn')


        # In[28]:


        # --- number_customer_service_calls --- #
        countPred_eda(train_data,'number_customer_service_calls','churn')


        # In[29]:


        # Removing highly collinear variables from train and test
        train_data = train_data.drop(['total_day_charge','total_eve_charge','total_night_charge', 'total_intl_charge'], axis = 1)
        test_data = test_data.drop(['total_day_charge','total_eve_charge','total_night_charge', 'total_intl_charge'], axis = 1)


        # In[30]:


        # encoding target variables
        le = LabelEncoder()
        # for train data
        train_data.churn = le.fit_transform(train_data.churn)
        # for test data
        test_data.churn = le.fit_transform(test_data.churn)


```

```
# In[31]:


# Encoding categorical variables
# for train data
train_data.international_plan = le.fit_transform(train_data.international_plan)
train_data.voice_mail_plan = le.fit_transform(train_data.voice_mail_plan)

# for test data
test_data.international_plan = le.fit_transform(test_data.international_plan)
test_data.voice_mail_plan = le.fit_transform(test_data.voice_mail_plan)


# In[32]:


test_data.head()


# In[33]:


train_data.churn.value_counts()


# In[34]:


# selecting predictors
train_feature_space = train_data.iloc[:,train_data.columns != 'churn']
# selecting target class
target_class = train_data.iloc[:,train_data.columns == 'churn']


# In[35]:


# creating training and validation set
training_set, validation_set, train_taget, validation_target = train_test_split(train_feature_space,
                                                                                target_class,
                                                                                test_size = 0.30,
                                                                                random_state = 456)

# Cleaning test sets to avoid future warning messages
train_taget = train_taget.values.ravel()
validation_target = validation_target.values.ravel()


# In[36]:


# logistic regression classifier
classifier_logit_default = LogisticRegression(random_state=456)


# In[37]:


classifier_logit_default.fit(training_set, train_taget)

```

```
# In[38]:


# Predicting the validation set results
validation_prediction = classifier_logit_default.predict(validation_set)


# In[39]:


# confusion matrix for validation set
validation_logit_crosstb = pd.crosstab(index = validation_target,
                                       columns = validation_prediction)
validation_logit_crosstb = validation_logit_crosstb.rename(columns= {0: 'False', 1: 'True'})
validation_logit_crosstb.index = ['False', 'True']
validation_logit_crosstb.columns.name = 'n = 1000'


# In[40]:


validation_logit_crosstb


# In[41]:


#classification report on validation set
target_names =[0,1]

validation_report = classification_report(validation_prediction, validation_target,  target_names )
print(validation_report)


# In[42]:


mean_accuracy_validation = classifier_logit_default.score(validation_set, validation_target)
print(' Mean accuracy on validation set',mean_accuracy_validation)


# In[43]:


# calculating test error rate on validation set
test_error_rate = 1 - mean_accuracy_validation
print(' Test error rate on validation set',test_error_rate)


# From the confusion matrix we can see that high accuracy of model is due to disproportionate number of non-churn
# customers predicted correctly. The This model is working great for identifing non churning customer but performing poorly for
# churning customers. We will tune the model to increase accuracy on churning customer.

# In[44]:


# model 2
classifier_logit_2 = LogisticRegression(class_weight='balanced')

```

```

        param = {'C':[0.001,0.005,0.01,0.05,0.1,0.5,1,1.5,2,3]}
        #rs_cv = RandomizedSearchCV(estimator=classifier_logit_2, cv = 10,
                                    #n_iter = 100,
                                   #param_distributions=param, random_state=1234)
        rs_cv = GridSearchCV(estimator=classifier_logit_2, cv = 10,param_grid=param)

        rs_cv.fit(training_set,train_taget)

        print('Best parameter :{} Best score :{}'.format(rs_cv.best_params_,rs_cv.best_score_))


        # In[45]:


        classifier_logit_2.set_params(C = 3)


        # In[46]:


        classifier_logit_2.fit(training_set, train_taget)


        # In[47]:


        validation_prediction_tuned = classifier_logit_2.predict(validation_set)


        # In[48]:


        # confusion matrix for validation set
        validation_logit_crosstb1 = pd.crosstab(index = validation_target,
                                   columns = validation_prediction_tuned)
        validation_logit_crosstb1 = validation_logit_crosstb1.rename(columns= {0: 'False', 1: 'True'})
        validation_logit_crosstb1.index = ['False', 'True']
        validation_logit_crosstb1.columns.name = 'n = 1000'


        # In[49]:


        validation_logit_crosstb1


        # In[50]:


        #classification report on validation set with hypertuning
        target_names =[0,1]

        validation_report_tuning = classification_report(validation_prediction_tuned, validation_target,  target_names )
        print(validation_report_tuning)


        # ### Prediction and performance on test data
        #
        # Model classifier_logit_2 was selected.

        # In[51]:
```

```
# test set
test_set = test_data.iloc[:,test_data.columns != 'churn']
# selecting target class for test set
test_set_target = test_data.iloc[:,test_data.columns == 'churn']


# Predicting the test set results
test_prediction = classifier_logit_2.predict(test_set)


# In[52]:


# confusion matrix for validation set
test_logit_crosstb = pd.crosstab(index = test_data.churn,
                           columns = test_prediction)
test_logit_crosstb = test_logit_crosstb.rename(columns= {0: 'False', 1: 'True'})
test_logit_crosstb.index = ['False', 'True']
test_logit_crosstb.columns.name = 'n = onservation'


# In[53]:


cm = ConfusionMatrix(test_data.churn, test_prediction)


# In[61]:


cm


# In[60]:


cm.print_stats()


# In[55]:


#classification report on test set with hypertuning
target_names =[0,1]

test_report_tuning = classification_report(test_prediction, test_data.churn, target_names )
print(test_report_tuning)


# In[56]:


mean_accuracy_test = classifier_logit_2.score(test_set, test_set_target)
print(' Mean accuracy on test set',mean_accuracy_test)


# In[57]:


# calculating test error rate on test set
test_error_rate_testset = 1 - mean_accuracy_test
print(' Test error rate on test set',test_error_rate)
```

```


        # In[63]:


        test_set_target.churn.value_counts()


        # In[65]:


        # Model input and output
        test_set.to_csv('inputLogisticRegressionPython.csv', encoding = 'utf-8', index = False)
        pd.DataFrame(train_taget).to_csv('targetLogisticRegressionPython.csv', index = False)
        pd.DataFrame(test_prediction, columns=['predictions']).to_csv('outputLogisticRegressionPython.csv')
```

customer Churn Reduction -Random forest.py

```
# coding: utf-8

# # Customer Churn Reduction  - Random forest
#
# Finding wether a customer will churn out or not. Random forest was use.

# In[1]:


# importing requried libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
import time
import random
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from pandas_ml import ConfusionMatrix

get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


# Reading train data
inputTrain = pd.read_csv("Train_data.csv")
# Reading test data
inputTest = pd.read_csv("Test_data.csv")


# In[3]:


# change column names for ease of use and display first 5 rows
inputTrain.columns = inputTrain.columns.str.lower().str.replace(' ', '_')
inputTest.columns = inputTest.columns.str.lower().str.replace(' ', '_')


# In[4]:
```

```


#dimensions
inputTrain.shape
inputTest.shape


# In[5]:


# calculating quick summary statstic for continous predictors
inputTrain.describe()


# In[6]:


#removing variables which not helpfull in churn reduction. Using these variables do
sn't make sense.
#for training data
inputTrain = inputTrain.drop(['state','area_code','phone_number'], axis = 1)
inputTest = inputTest.drop(['state','area_code','phone_number'], axis = 1)


# In[7]:


# sanity check
print(inputTrain.shape)
print(inputTest.shape)


# In[8]:


print("Train dataset",inputTrain.churn.value_counts())
print("Test data", inputTest.churn.value_counts())


# In[9]:


# encoding categorical and target variable to binary
# converting international_plan,voice_mail_plan and churn

le = LabelEncoder()

inputTrain.international_plan = le.fit_transform(inputTrain.international_plan)
inputTrain.voice_mail_plan = le.fit_transform(inputTrain.voice_mail_plan)
inputTrain.churn = le.fit_transform(inputTrain.churn)

inputTest.international_plan = le.fit_transform(inputTest.international_plan)
inputTest.voice_mail_plan = le.fit_transform(inputTest.voice_mail_plan)
inputTest.churn = le.fit_transform(inputTest.churn)


       # In[10]:


       # selecting predictors
       train_feature_space = inputTrain.iloc[:,inputTrain.columns != 'churn']
       # selecting target class
       target_class = inputTrain.iloc[:,inputTrain.columns == 'churn']


       # In[11]:
```

```python


# creating training and validation set
training_set, validation_set, train_taget, validation_target = train_test_split(train_feature_space,
                                                                                 target_class,
                                                                                 test_size = 0.30,
                                                                                 random_state = 12345)

# Cleaning test sets to avoid future warning messages
train_taget = train_taget.values.ravel()
validation_target = validation_target.values.ravel()


# ## Random forest Implementation

# In[12]:


# using random forest classifier. setting a random state
fit_randomForest = RandomForestClassifier(random_state=12345)


# ### Hyperparameters Optimization¶
#
# Utilizing the RandomizedSearchCV functionality, we create a dictionary with parameters we are looking to optimize to create the best model for our data.

# In[13]:


np.random.seed(12)
start = time.time()

# selecting best max_depth, maximum features, split criterion and number of trees
param_dist = {'max_depth': [2,4,6,8,10],
              'bootstrap': [True, False],
              'max_features': ['auto', 'sqrt', 'log2',None],
              "criterion": ["gini", "entropy"],
              "n_estimators" : [100 ,200 ,300 ,400 ,500]
             }
cv_randomForest = RandomizedSearchCV(fit_randomForest, cv = 10,
                     param_distributions = param_dist,
                     n_iter = 10)

cv_randomForest.fit(training_set, train_taget)
print('Best Parameters using random search: \n',
      cv_randomForest.best_params_)
end = time.time()
print('Time taken in random search: {0: .2f}'.format(end - start))


# In[39]:


# Set best parameters given by random search
fit_randomForest.set_params(criterion = 'gini',
                  max_features = 'auto',
                  max_depth = 10,
                  n_estimators = 100
                )

```

```

# In[40]:


fit_randomForest.fit(training_set, train_taget)


# In[16]:


importances_rf = fit_randomForest.feature_importances_
indices_rf = np.argsort(importances_rf)[::-1]


# In[17]:


def variable_importance_plot(importance, indices,training_set):

    index = np.arange(len(training_set.columns))


    importance_desc = sorted(importance)
    feature_space = []
    for i in range(16, -1, -1):
        feature_space.append(training_set.columns[indices[i]])

    fig, ax = plt.subplots(figsize=(14, 14))

    ax.set_facecolor('#fafafa')
    plt.title('Feature importances for Random Forest Model for Customer churn')
    plt.barh(index,
             importance_desc,
             align="center",
             color = '#875FDB')
    plt.yticks(index,
               feature_space)
    plt.xlim(0, max(importance_desc))
    plt.xlabel('Mean Decrease in gini')
    plt.ylabel('Feature')
    plt.savefig('VarImp.png')
    #savefig('VarImp.pdf')
    plt.show()


    plt.close()


# In[18]:


variable_importance_plot(importances_rf, indices_rf,training_set)


# ## perfoming Cross validation

# In[19]:


# function to perform cross validation
def cross_val_metrics(fit, training_set, train_taget,print_results = True):


    n = KFold(n_splits=10)
    scores = cross_val_score(fit,
```

```
                         training_set,
                         train_taget,
                         cv = n)
    if print_results:
        print("Accuracy: {0: 0.3f} (+/- {1: 0.3f})"              .format(scores.mean(), scores.std() / 2))
    else:
        return scores.mean(), scores.std() / 2


# In[20]:


cross_val_metrics(fit_randomForest, training_set,
                  train_taget,
                  print_results = True)


# ## prediction and performance measure on validation set

# In[41]:


predictions_randomForest_validation = fit_randomForest.predict(validation_set)

validation_crosstb = pd.crosstab(index = validation_target,
                           columns = predictions_randomForest_validation)
validation_crosstb = validation_crosstb.rename(columns= {0: 'False', 1: 'True'})
validation_crosstb.index = ['False', 'True']
validation_crosstb.columns.name = 'n = 1000'


# In[42]:


# confusion matrix of validation set
validation_crosstb


# In[23]:


# mean accuracy on validation set
accuracy_randomForest_val = fit_randomForest.score(validation_set, validation_target)
print(' Mean accuracy on validation set',accuracy_randomForest_val)


# In[24]:


# calculating test error rate on validation set
test_error_rate = 1 - accuracy_randomForest_val
print(' Test error rate on validation set',test_error_rate)


# In[25]:


#classification report on validation set
target_names =[0,1]

validation_report = classification_report(predictions_randomForest_validation, validation_target,  target_names )
```

```
        print(validation_report)


        # ## prediction and performance measure on test set

        # In[43]:


        #selecting predictors
        test_set = inputTest.iloc[:,inputTest.columns != 'churn']
        # selecting target class
        test_target = inputTest.iloc[:,inputTest.columns == 'churn']

        test_prediction = fit_randomForest.predict(test_set)


        # In[44]:


        #performing prediction on test set
        test_prediction = fit_randomForest.predict(test_set)


        # In[45]:


        # creating confusion matrix of test set
        confusion_matrix(test_target,test_prediction)


        # In[46]:


        test_rf_crosstb = pd.crosstab(index = test_target.churn,
                                   columns = test_prediction)
        test_rf_crosstb = test_rf_crosstb.rename(columns= {0: 'False', 1: 'True'})
        test_rf_crosstb.index = ['False', 'True']
        test_rf_crosstb.columns.name = 'n = 1667'


        # In[47]:


        test_rf_crosstb


        # In[31]:


        # mean accuracy on test set
        accuracy_randomForest_test = fit_randomForest.score(test_set, test_target.ch
urn)
        print(' Mean accuracy on test set',accuracy_randomForest_test)


        # In[32]:


        # calculating test error rate on test set
        test_error_rate_testset = 1 - accuracy_randomForest_test
        print(' Test error rate on test set',test_error_rate)


        # In[33]:


```

```
#classification report on test set
target_names =[0,1]

test_report = classification_report(test_prediction, test_target.churn,  target_names )
print(test_report)


# In[35]:


cm = ConfusionMatrix(test_target.churn, test_prediction)


# In[38]:


cm


# In[48]:


cm.print_stats()


# In[69]:


# Model input and output
test_set.to_csv('inputRandomForestPython.csv', encoding = 'utf-8', index = False)
pd.DataFrame(train_taget).to_csv('targetInputRandomForestPython.csv', index = False)
pd.DataFrame(test_prediction, columns=['predictions']).to_csv('outputRandomForestPython.csv')
```