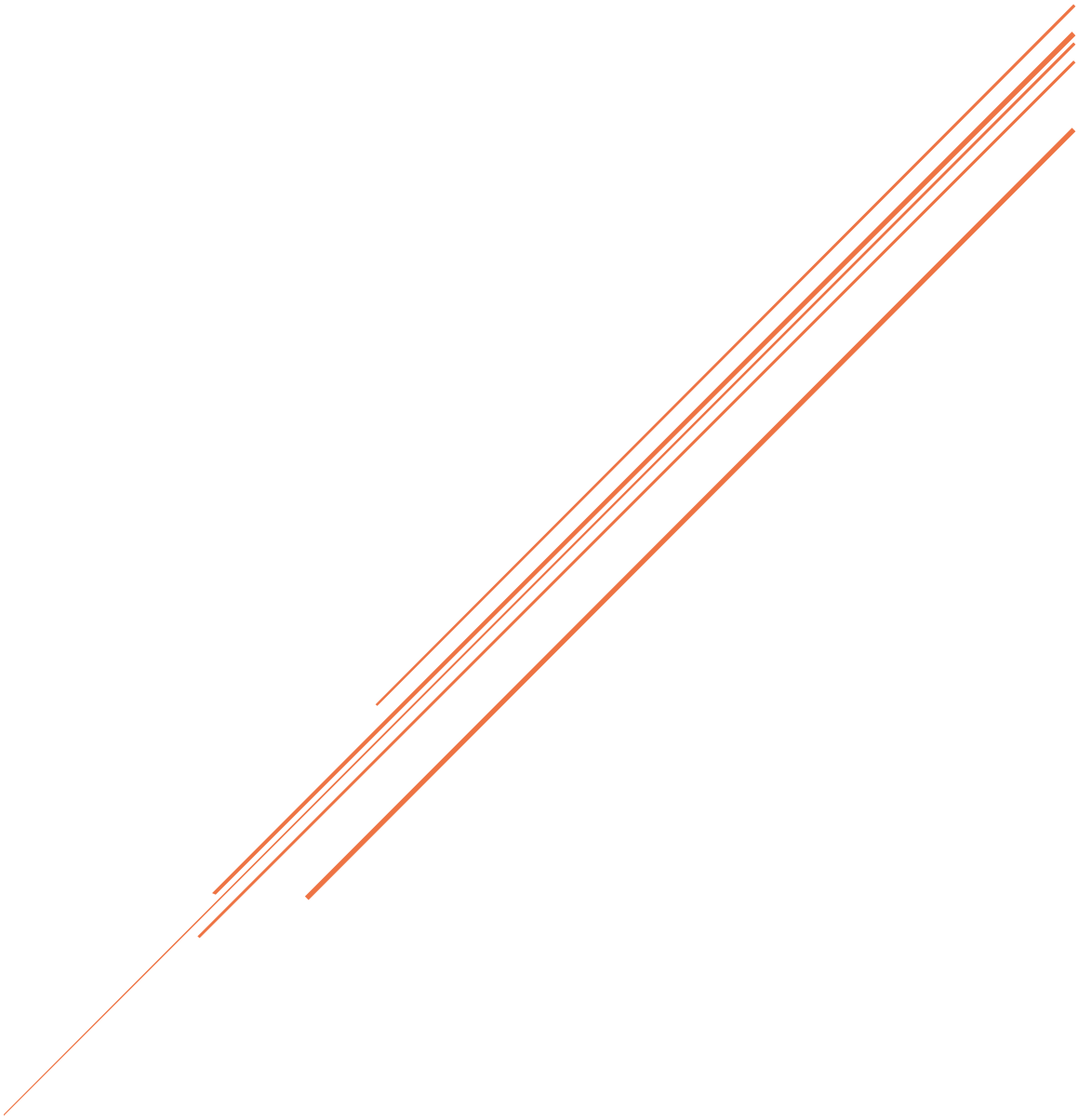
CHURN SCORE PREDICTION

The Objective of this Case is to Predict Customer Behavior. The Customer is going to stay with Current Carrier or Churn from the Current Carrier.



Chirag Nayak

18th, October 2018

CONTENTS

INTRODUCTION

My objective of the analysis is to find out predictions of churn score, construct statistical models and then try to make prediction on churn score based on the information and models, which I have. My exploration and the analysis of the data will be performed in Jupyter Notebook using Python; Tableau will be used for visualization.

For our modeling, Edwisor have provided with training and test dataset separately of existing users of a telecom company.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	state	account le	area code	phone num	internation	voice mail	number vm	total day m	total day c	total day cl	total eve m	total eve c	total eve cl	total night	total night	total night	total intl minut	total intl calls	total intl ch nu
2	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10	3	2.7
3	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.7
4	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	121.2	110	10.3	162.6	104	7.32	12.2	5	3.29
5	OH	84	408	375-9999	yes	no	0	299.4	71	50.9	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78
6	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73
7	AL	118	510	391-8027	yes	no	0	223.4	98	37.98	220.6	101	18.75	203.9	118	9.18	6.3	6	1.7
8	MA	121	510	355-9993	no	yes	24	218.2	88	37.09	348.5	108	29.62	212.6	118	9.57	7.5	7	2.03
9	MO	147	415	329-9001	yes	no	0	157	79	26.69	103.1	94	8.76	211.8	96	9.53	7.1	6	1.92
10	LA	117	408	335-4719	no	no	0	184.5	97	31.37	351.6	80	29.89	215.8	90	9.71	8.7	4	2.35
11	WV	141	415	330-8173	yes	yes	37	258.6	84	43.96	222	111	18.87	326.4	97	14.69	11.2	5	3.02
12	IN	65	415	329-6603	no	no	0	129.1	137	21.95	228.5	83	19.42	208.8	111	9.4	12.7	6	3.43
13	RI	74	415	344-9403	no	no	0	187.7	127	31.91	163.4	148	13.89	196	94	8.82	9.1	5	2.46
14	IA	168	408	363-1107	no	no	0	128.8	96	21.9	104.9	71	8.92	141.1	128	6.35	11.2	2	3.02
15	MT	95	510	394-8006	no	no	0	156.6	88	26.62	247.6	75	21.05	192.3	115	8.65	12.3	5	3.32
16	IA	62	415	366-9238	no	no	0	120.7	70	20.52	307.2	76	26.11	203	99	9.14	13.1	6	3.54
17	NY	161	415	351-7269	no	no	0	332.9	67	56.59	317.8	97	27.01	160.6	128	7.23	5.4	9	1.46
18	ID	85	408	350-8884	no	yes	27	196.4	139	33.39	280.9	90	23.88	89.3	75	4.02	13.8	4	3.73
19	VT	93	510	386-2923	no	no	0	190.7	114	32.42	218.2	111	18.55	129.6	121	5.83	8.1	3	2.19
20	VA	76	510	356-2992	no	yes	33	189.7	66	32.25	212.8	65	18.09	165.7	108	7.46	10	5	2.7
21	TX	73	415	373-2782	no	no	0	224.4	90	38.15	159.5	88	13.56	192.8	74	8.68	13	2	3.51
22	FL	147	415	396-5800	no	no	0	155.1	117	26.37	239.7	93	20.37	208.8	133	9.4	10.6	4	2.86
23	CO	77	408	393-7984	no	no	0	62.4	89	10.61	169.9	121	14.44	209.6	64	9.43	5.7	6	1.54
24	AZ	130	415	358-1958	no	no	0	183	112	31.11	72.9	99	6.2	181.8	78	8.18	9.5	19	2.57
25	SC	111	415	350-2565	no	no	0	110.4	103	18.77	137.3	102	11.67	189.6	105	8.53	7.7	6	2.08
26	VA	132	510	343-4696	no	no	0	81.1	86	13.79	245.2	72	20.84	237	115	10.67	10.3	2	2.78
27	NE	174	415	331-3698	no	no	0	124.3	76	21.13	277.1	112	23.55	250.7	115	11.28	15.5	5	4.19
28	WY	57	408	357-3817	no	yes	39	213	115	36.21	191.1	112	16.24	182.7	115	8.22	9.5	3	2.57

UPLOADING THE DATA SET IN JUPYTER NOTEBOOK TO ANALYZE IT WITH PYTHON

```
import matplotlib.pyplot as plt
import numpy as np
import os
os.chdir("E:\\R WorkStation\\Python WorkStation")
os.getcwd()
import pandas as pd
dft= pd.read_csv("Train_data.csv", sep = ',')
dft.head(1)
```

AS I UPLOADED THE FILE TO JUPYTER NOTEBOOK, I CAN SEE THE FOLLOWING VARIABLES

- State: They are state code (2 character) representing the state from USA.
- Account length: This represents customer timeline.
- Area code: It is to specify the exact location of customer, just like pin-code /postcode.
- Phone number: can be used as customer identifier.
- International plan: It states if the customer as opt for international service plan.
- Voice mail plan: It states if the customer as opt for voice mail service plan.

- Number vmail messages: Total voice-mail send.
- Total day minutes: Shows total minute spent in call during the day.
- Total day calls: Total number of calls during the day.
- Total day charge: Total charge the customer has to pay for all the calls during the day.
- Total eve minutes: Shows total minute spent in call during the evening.
- Total eve calls: Total number of calls during the evening.
- Total eve charge: Total charge the customer has to pay for all the calls during the evening.
- Total night minutes: Shows total minute spent in call during the night.
- Total night calls: Total number of calls during the night.
- Total night charge: Total charge the customer has to pay for all the calls during the night.
- Total intl minutes: Shows total minute spent in international calls.
- Total intl calls: Total number of international calls.
- Total intl charge: Total charge the customer has to pay for all the international calls.
- Customer service calls: Calls made to customer service center.
- Churn: Shows whether the customer has churned or not.

GETTING SHAPE OF THE DATA SET

```
dft.shape
```

```
(3333, 21)
```

The dataset I will be using contains 3333 observations and 21 variables

DATA CLEANING

CHECKING FOR MISSING VALUES IN TABLE:-

```
dft.isnull().any().any()
```

```
False
```

Looks like there is no Missing values in the data frame.

NAMES OF ALL COLUMNS IN DATA FRAME:-

```
dft.columns
```

```
Index(['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', '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'],
      dtype='object')
```

Almost all column's name have spaces in them. Since, Python will not work properly if column's name have spaces in them so I replaced all spaces with underscores.

```
dft.columns=dft.columns.str.replace(' ','_')
```

I did convert all Measurement variable to numeric data type and all dimensions variable to categorical data type.

```
dft.account_length=pd.to_numeric(dft.account_length)
dft.total_day_minutes=pd.to_numeric(dft.total_day_minutes)
dft.total_day_calls=pd.to_numeric(dft.total_day_calls)
dft.total_day_charge=pd.to_numeric(dft.total_day_charge)
dft.total_eve_minutes=pd.to_numeric(dft.total_eve_minutes)
dft.total_eve_calls=pd.to_numeric(dft.total_eve_calls)
dft.total_eve_charge=pd.to_numeric(dft.total_eve_charge)
dft.total_night_minutes=pd.to_numeric(dft.total_night_minutes)
dft.total_night_calls=pd.to_numeric(dft.total_night_calls)
dft.total_night_charge=pd.to_numeric(dft.total_night_charge)
dft.total_intl_minutes=pd.to_numeric(dft.total_intl_minutes)
dft.total_intl_calls=pd.to_numeric(dft.total_intl_calls)
dft.total_intl_charge=pd.to_numeric(dft.total_intl_charge)
```

```
dft.international_plan=dft.international_plan.astype('category')
dft.voice_mail_plan=dft.voice_mail_plan.astype('category')
dft.Churn=dft.Churn.astype('category')
dft.international_plan=dft.international_plan.cat.codes
dft.voice_mail_plan=dft.voice_mail_plan.cat.codes
```

I encoded all categorical data type variables so that Python and Machine learning algorithms can understand it better.

```
dft.Churn=dft.Churn.cat.codes
```

MAKING A BOX PLOT TO CHECK FOR OUTLIERS:-

Box-and-whisker plots are a handy way to display data broken into four quartiles, each with an equal number of data values. The box-and-whisker plot doesn't show frequency, and it doesn't display each individual statistic, but it clearly shows where the middle of the data lies. It's a nice plot to use when analyzing how your data is skewed.

There are a few important vocabulary terms to know in order to graph a box-and-whisker plot. Here they are:

Q1 - quartile 1, the median of the lower half of the data set

Q2 - quartile 2, the median of the entire data set

Q3 - quartile 3, the median of the upper half of the data set

IQR - interquartile range, the difference from Q3 to Q1

Extreme Values – the smallest and largest values in a data set

Let's start by making a box-and-whisker plot (also known as a "box plot") of the geometry test scores we saw earlier:

90, 94, 53, 68, 79, 84, 87, 72, 70, 69, 65, 89, 85, 83, 72

Step 1: Order the data from least to greatest.

53 65 68 69 70 72 72 79 83 84 85 87 89 90 94

Step 2: Find the median of the data.

This is also called quartile 2 (Q2).

Step 3: Find the median of the data less than Q2.

This is the lower quartile (Q1).

Step 4. Find the median of the data greater than Q2.

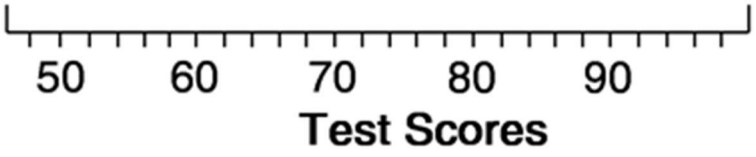
This is the upper quartile (Q3).

Step 5. Find the extreme values: these are the largest and smallest data values.

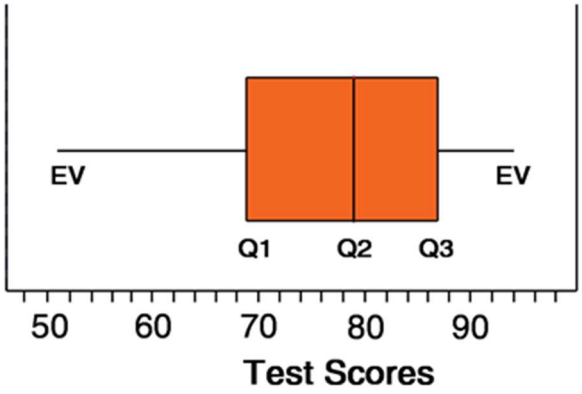
Extreme values = 53 and 94.

Step 6. Create a number line that will contain all of the data values.

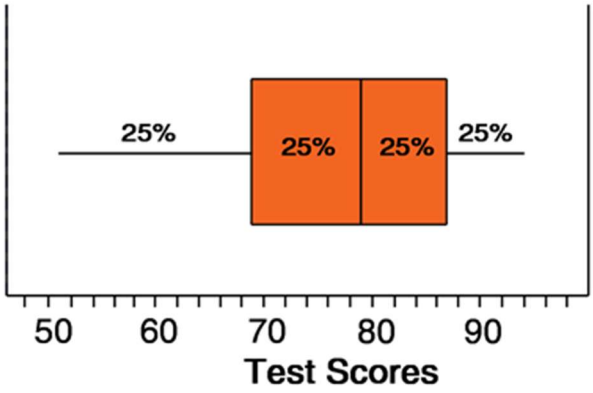
It should stretch a little beyond each extreme value.



Step 7. Draw a box from Q1 to Q3 with a line dividing the box at Q2. Then extend "whiskers" from each end of the box to the extreme values.



This plot is broken into four different groups: the lower whisker, the lower half of the box, the upper half of the box, and the upper whisker. Since there is an equal amount of data in each group, each of those sections represents 25% of the data.



Using this plot we can see that 50% of the students scored between 69 and 87 points, 75% of the students scored lower than 87 points, and 50% scored above 79. If your score was in the upper whisker, you could feel pretty proud that you scored better than 75% of your classmates. If you scored somewhere in the lower whisker, you may want to find a little more time to study.

Outliers

Outliers are values that are MUCH bigger or smaller than the rest of the data. These are represented by a dot at either end of the plot. Our geometry test example did not have any outliers, even though the score of 53 seemed much smaller than the rest, it wasn't small ENOUGH.

In order to be an outlier, the data value must be:

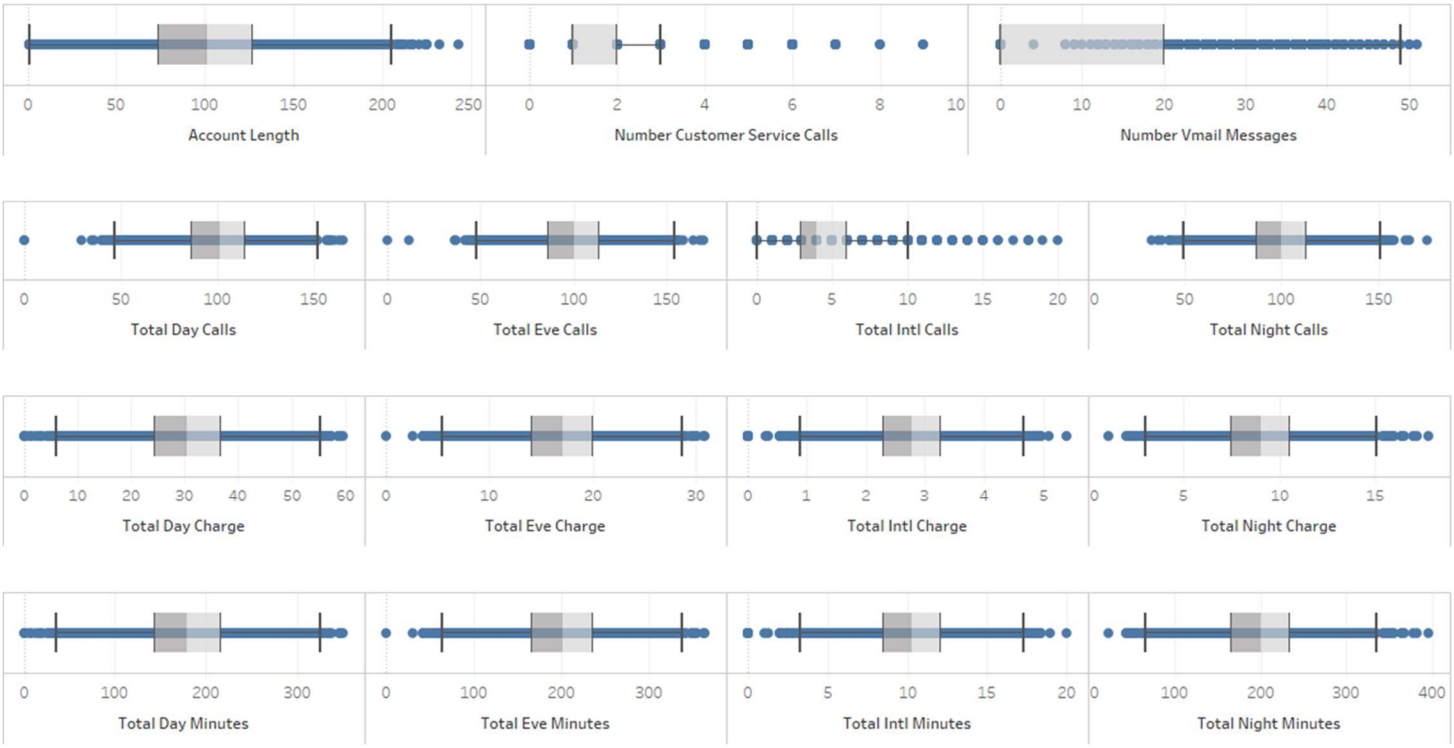
larger than Q3 by at least 1.5 times the interquartile range (IQR), or

smaller than Q1 by at least 1.5 times the IQR.

Outliers are values that are:

$1.5(IQR) > Q3$

$1.5(IQR) < Q1$



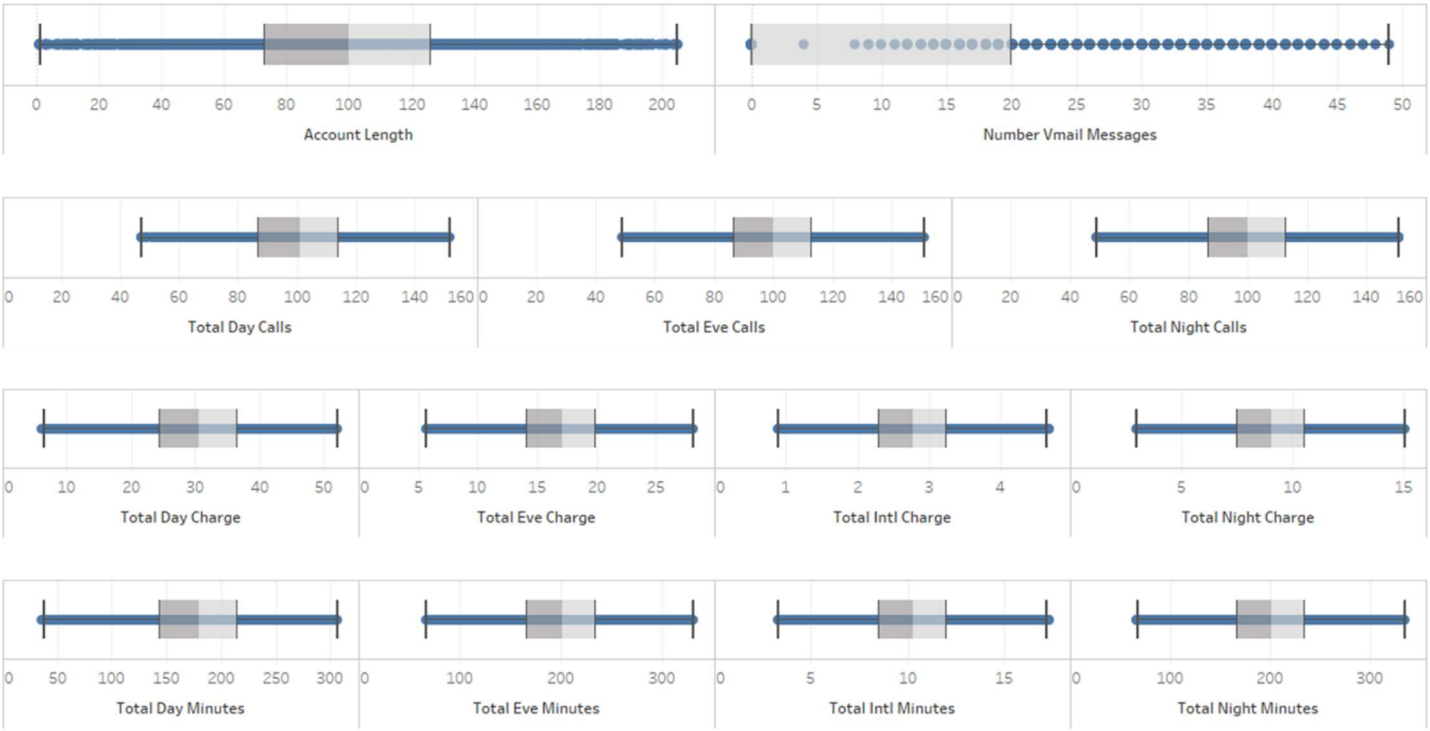
Looks like all continuous variables have same outlier values in them. The variable “Number Customer Service Calls” and “Total Intl Calls” have more than 40% of outliers in them, so I am dropping these two variables and I will be not using them anymore.

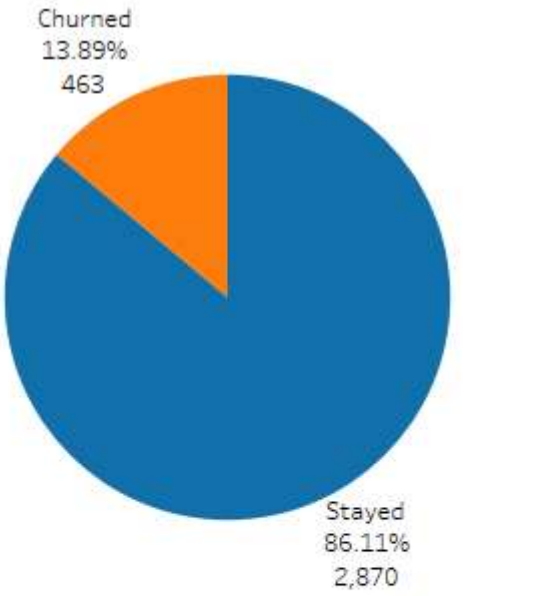
```
dft=dft.drop(columns=['total_intl_calls', 'number_customer_service_calls'])
```

I found upper whisker and lower whisker for every variable in data set from box plot and used them to remove the outliers. The values, which are more then upper whisker and less then lower whisker, are outliers. Then, I replaced them with forward fill.

```
dft[(dft.account_length>205) | (dft.account_length<1)]=np.nan
dft[(dft.number_vmail_messages>49) | (dft.number_vmail_messages<0)]=np.nan
dft[(dft.total_day_minutes>324.7) | (dft.total_day_minutes<35.1)]=np.nan
dft[(dft.total_eve_minutes>330.6) | (dft.total_eve_minutes<67)]=np.nan
dft[(dft.total_intl_minutes>17.3) | (dft.total_intl_minutes<3.3)]=np.nan
dft[(dft.total_night_minutes>334.7) | (dft.total_night_minutes<64.7)]=np.nan
dft[(dft.total_day_calls>152) | (dft.total_day_calls<47)]=np.nan
dft[(dft.total_eve_calls>151) | (dft.total_eve_calls<49)]=np.nan
dft[(dft.total_night_calls>151) | (dft.total_night_calls<49)]=np.nan
dft[(dft.total_day_charge>52.2) | (dft.total_day_charge<5.97)]=np.nan
dft[(dft.total_eve_charge>28.7) | (dft.total_eve_charge<5.47)]=np.nan
dft[(dft.total_intl_charge>4.67) | (dft.total_intl_charge<0.69)]=np.nan
dft[(dft.total_night_charge>15.06) | (dft.total_night_charge<2.96)]=np.nan
dft=dft.fillna(method='ffill')
```

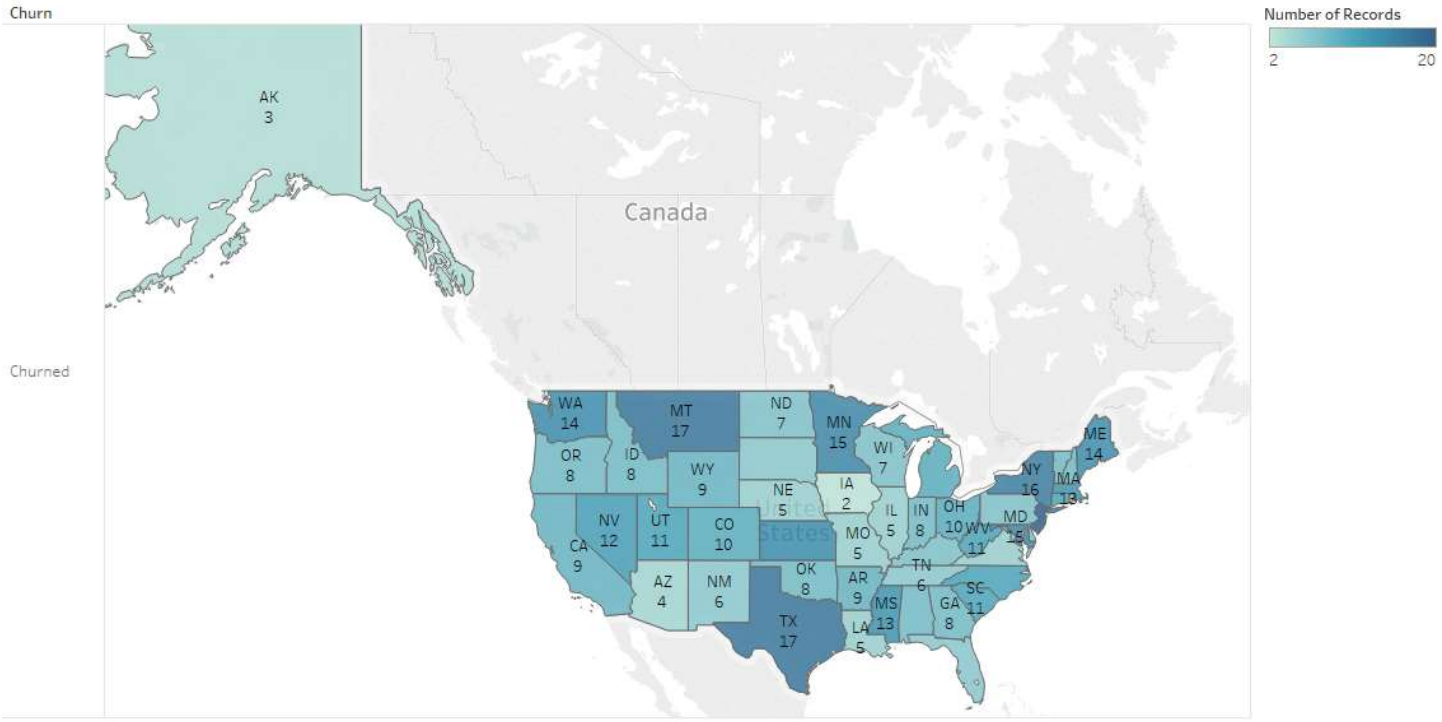
After removing outliers from all variables:-





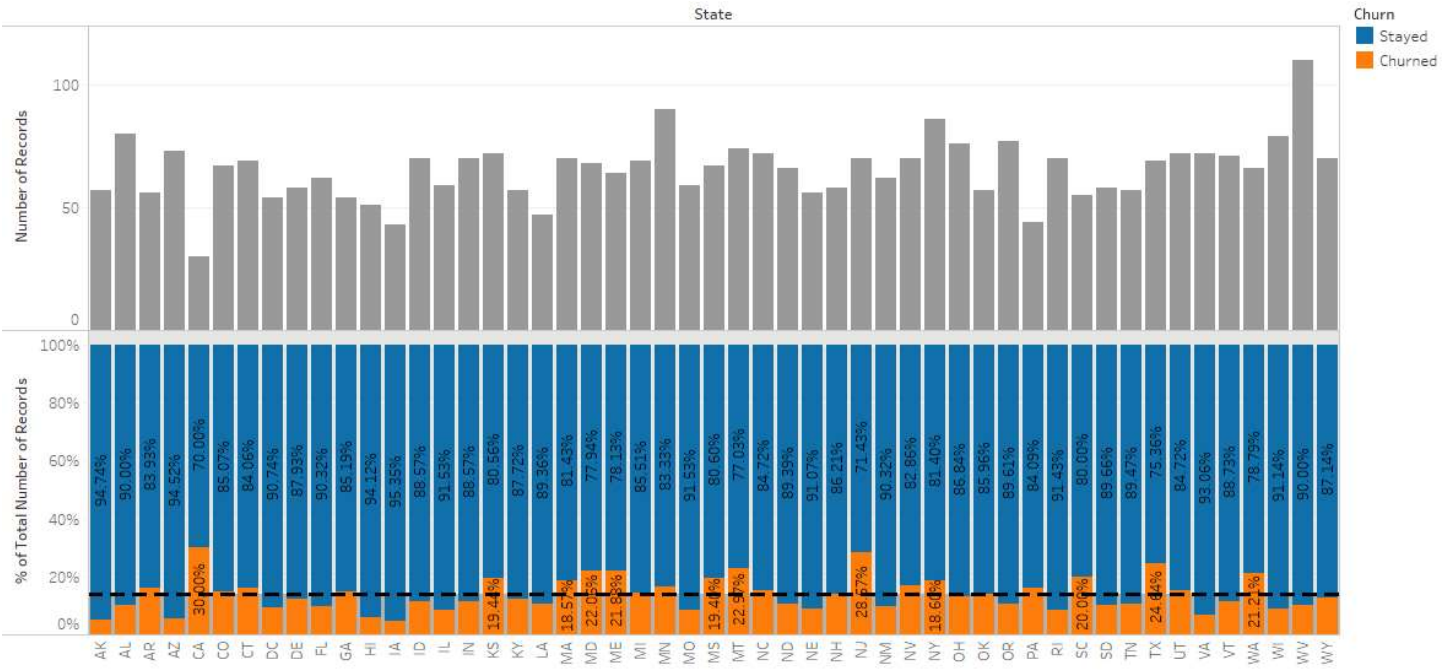
Churn, % of Total Number of Records and sum of Number of Records. Color shows details about Churn. Size shows sum of Number of Records. The marks are labeled by Churn, % of Total Number of Records and sum of Number of Records.

From this pie chart, we can see that 13.89% of total customer are churned and 86.11% are staying with the telecom company. This makes the data set imbalance. Therefore, we need to carefully select the important variable to make the model.



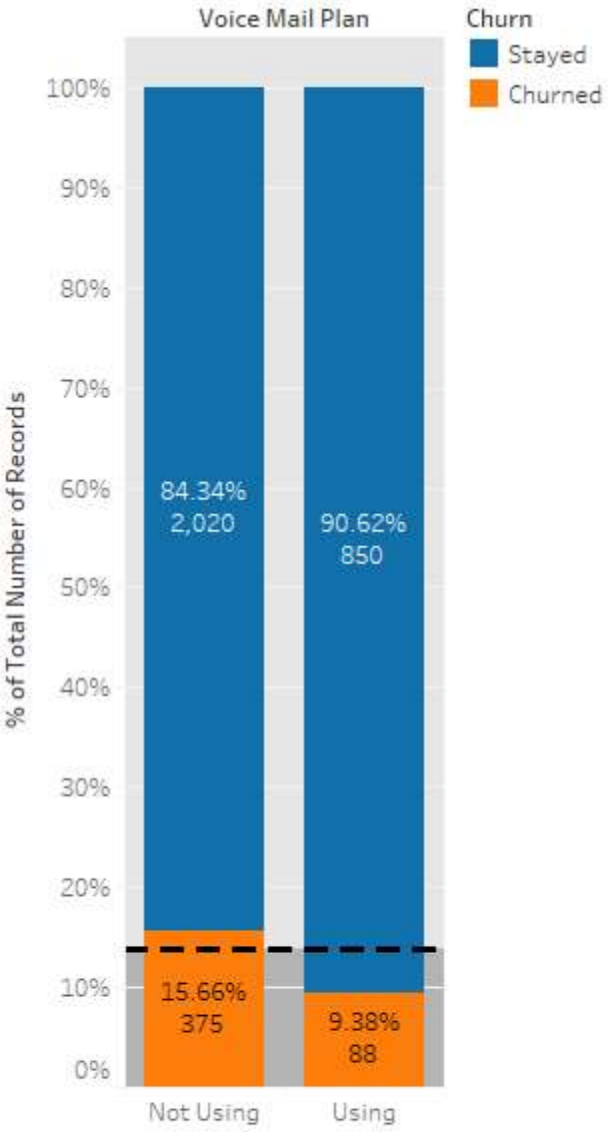
Map based on Longitude (generated) and Latitude (generated) broken down by Churn. Color shows sum of Number of Records. The marks are labeled by State and sum of Number of Records. Details are shown for State. The view is filtered on Churn, which keeps Churned.

The sates named MT, TX, MN, NY and NJ have very high churned rate as compared to others.



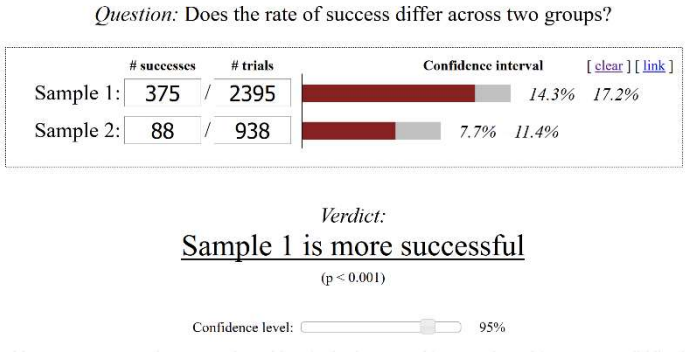
Sum of Number of Records and % of Total Number of Records for each State. For pane % of Total Number of Records: Color shows details about Churn. The marks are labeled by % of Total Number of Records. For pane Sum of Number of Records: The marks are labeled by Difference in Number of Records.

The above histogram also shows that the sates named MT, TX, MN, NY and NJ have very high churned rate as compared to others. State has relation with the churn rate so must be taken as a feature in the model.



% of Total Number of Records for each Voice Mail Plan. Color shows details about Churn. The marks are labeled by % of Total Number of Records and sum of Number of Records.

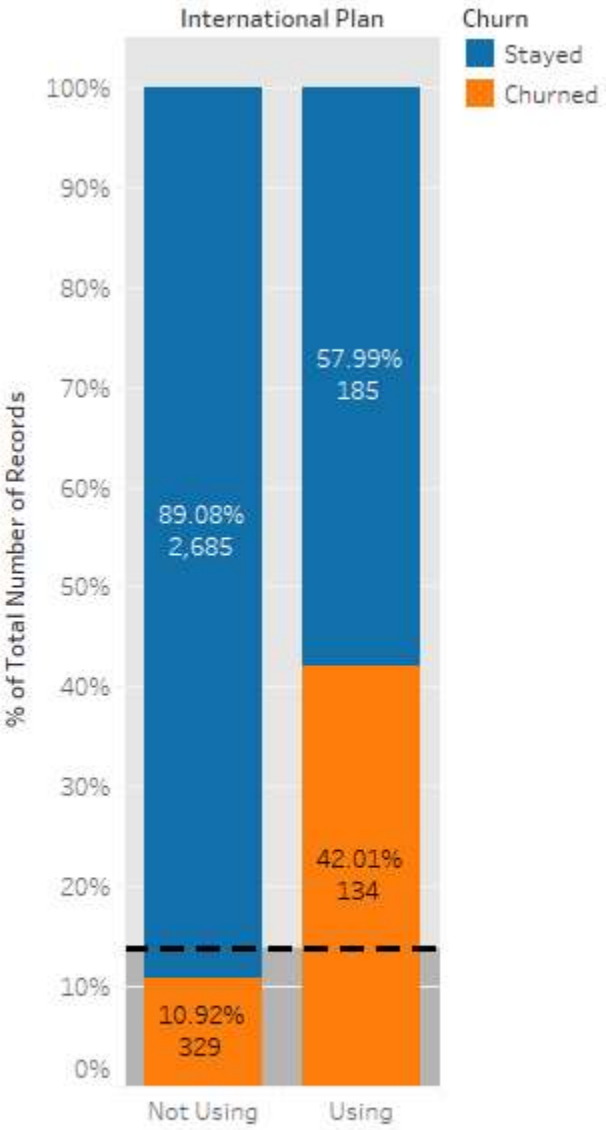
Voice Mail shows interesting relationship with churn. You need to subscribe voice mail plan to send voicemail. In addition, Voice mail is not deeply correlated with Churn. We can add/ignore Voicemail



Plan in our feature. It has less impact on churn.

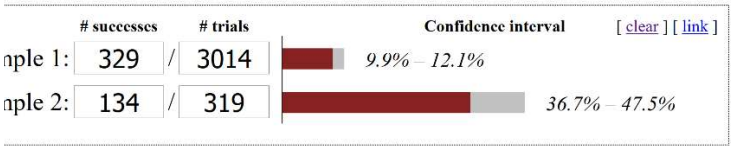
The above Chi-Square Test shows that if a customer is not using the voice mail plain then he/she is more likely to churn.

The p-value is very less, which is less then 0.001, so we can use this in our model if required.



% of Total Number of Records for each International Plan. Color shows details about Churn. The marks are labeled by % of Total Number of Records and sum of Number of Records.

Question: Does the rate of success differ across two groups?



Verdict:

Sample 2 is more successful

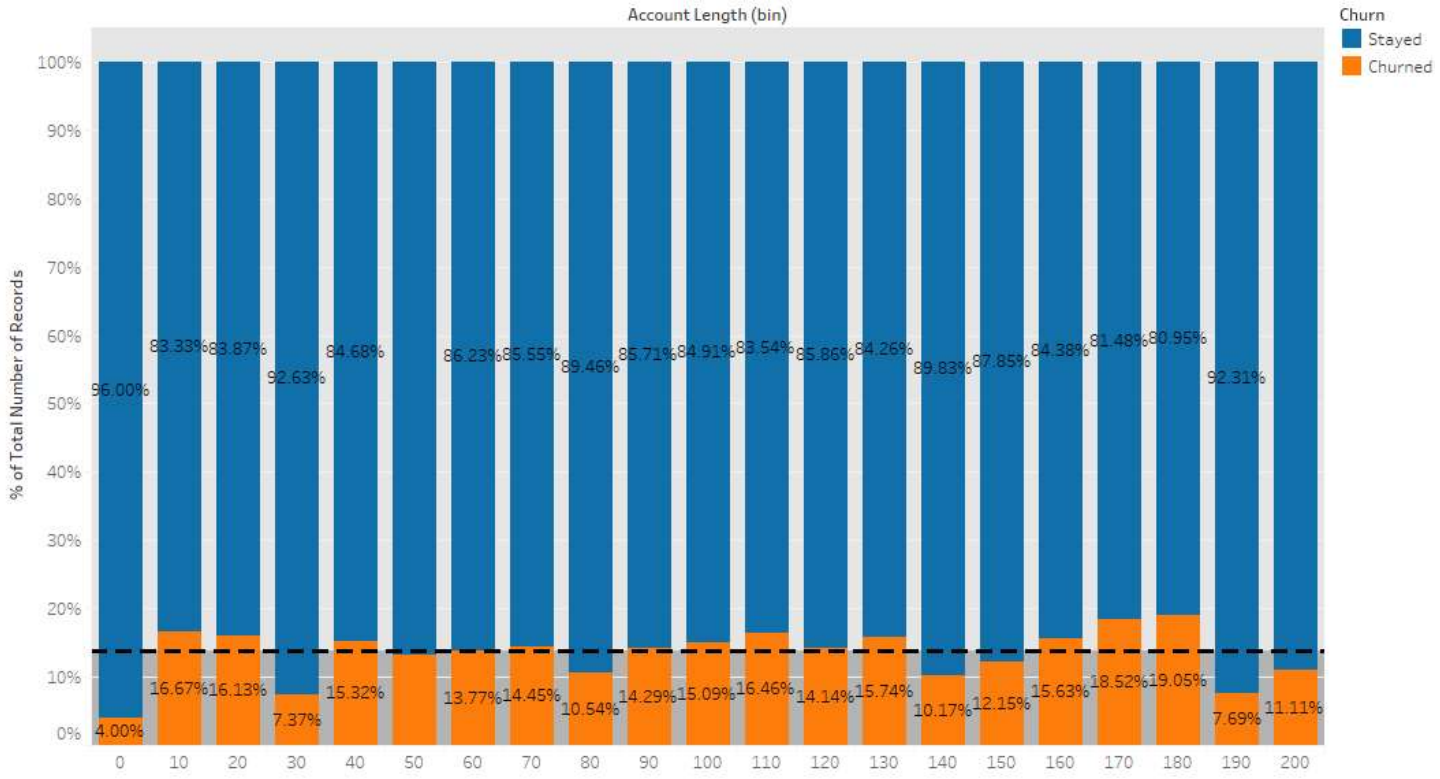
($p < 0.001$)

Confidence level: 95%

International Plan subscriber has many interesting facts involved. If you are using international plan you have around 42% chance of churn. Very few people use international plan. With very few calls. It looks like the company is not a good service provider for international plains that is why customers are being churned when using this plain.

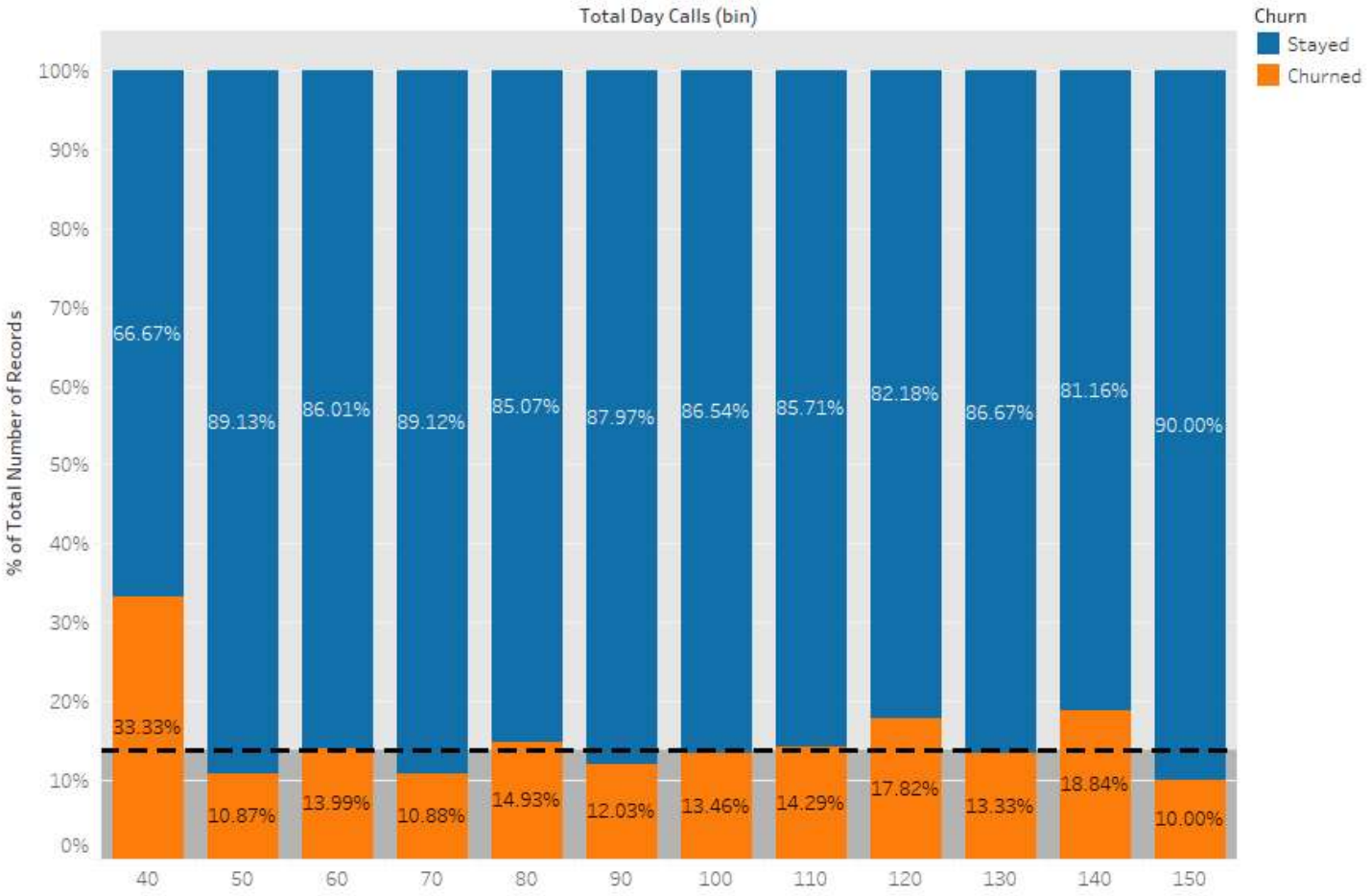
The above Chi-Square Test shows that if a customer is using the international plain then he/she is more likely to churn.

The p-value is very less, which is less than 0.001, so we can use this in our model if required.



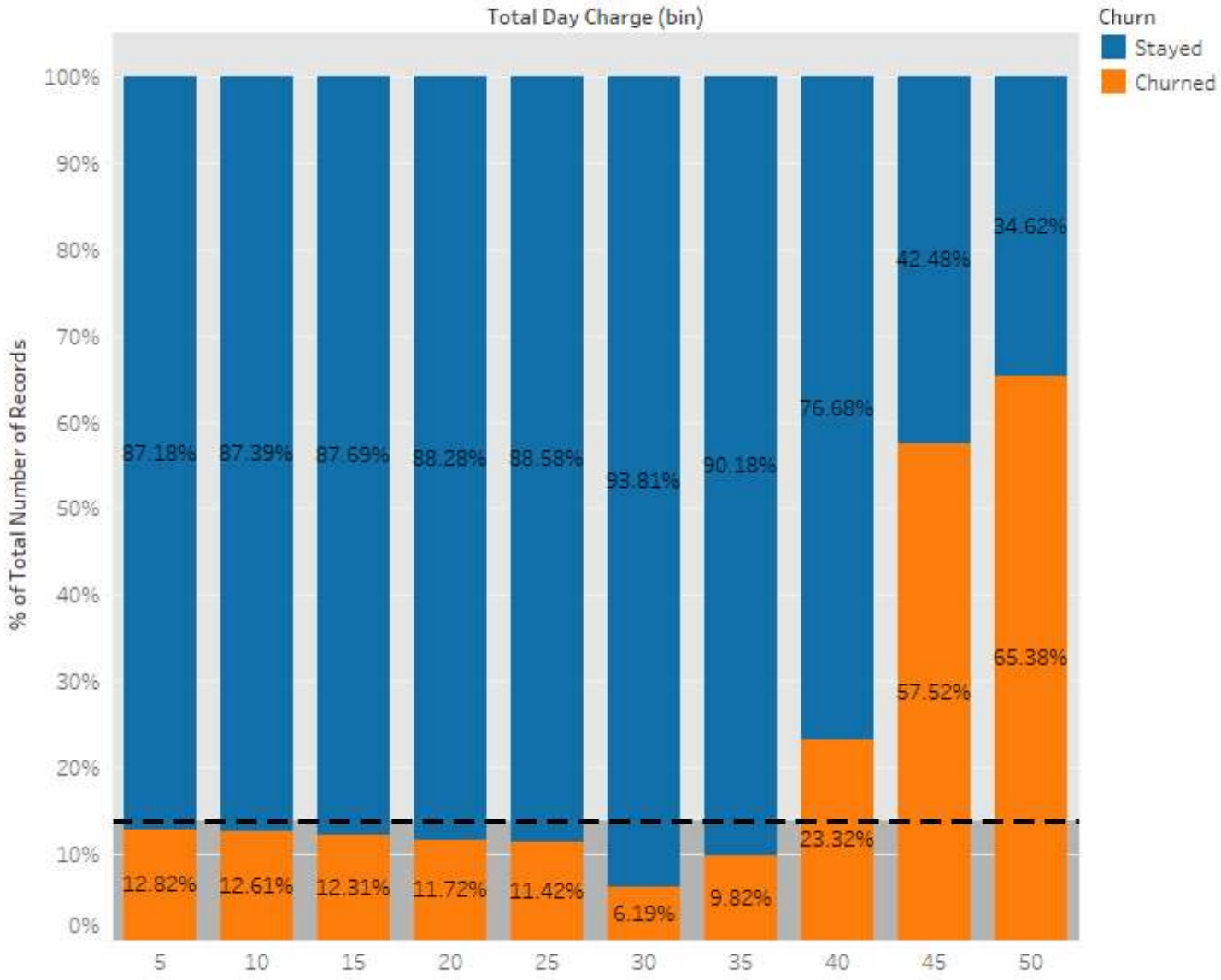
% of Total Number of Records for each Account Length (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that account length does not have any impact on customer churn rate. It will not have much effect on the predictions of the model if it is included in the prediction or not.



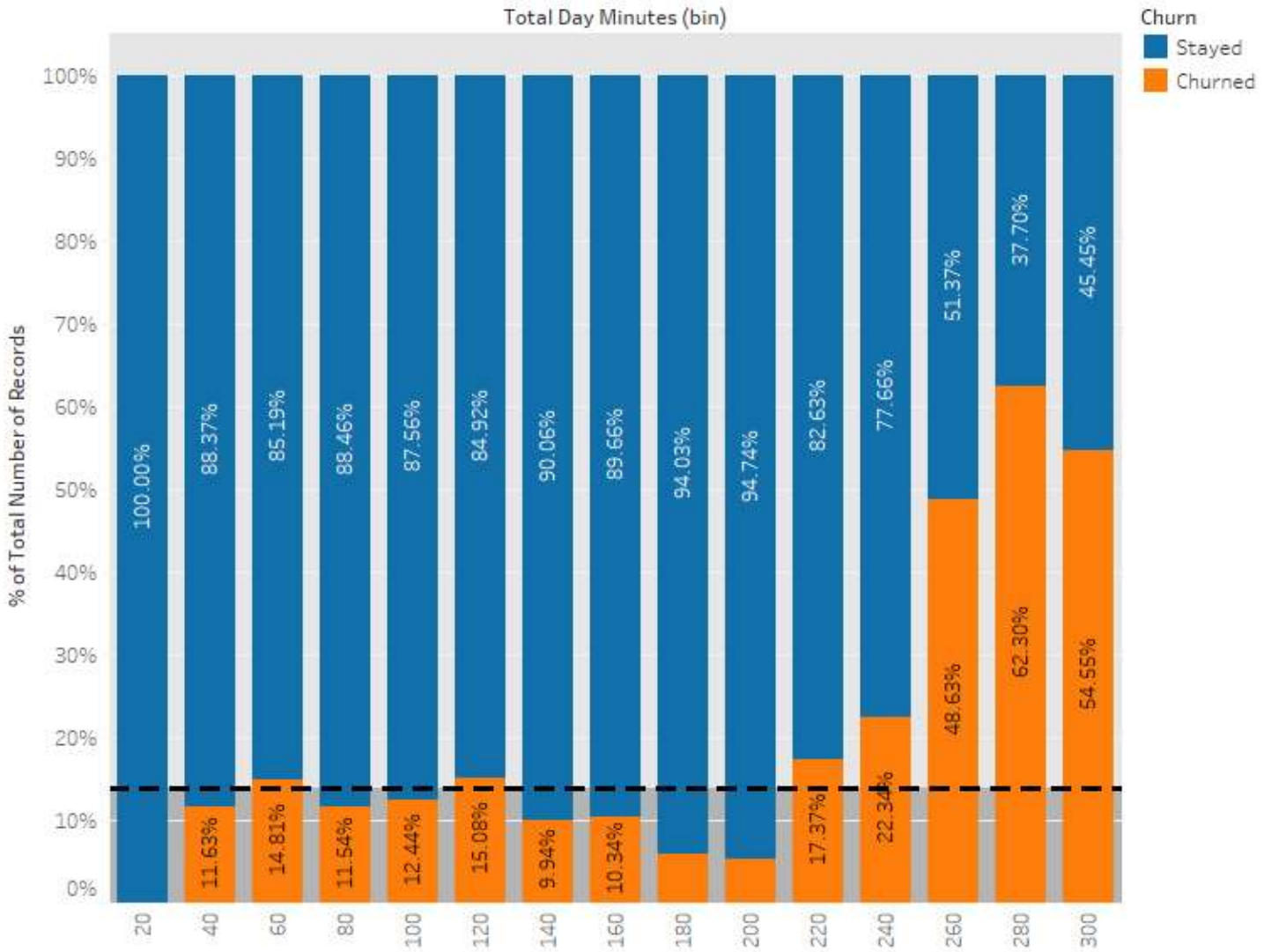
% of Total Number of Records for each Total Day Calls (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that customer who are making total day calls of 40 or less are likely to churn by 33.33%. However, customers whose total day call is more than 50 have average churn rate.



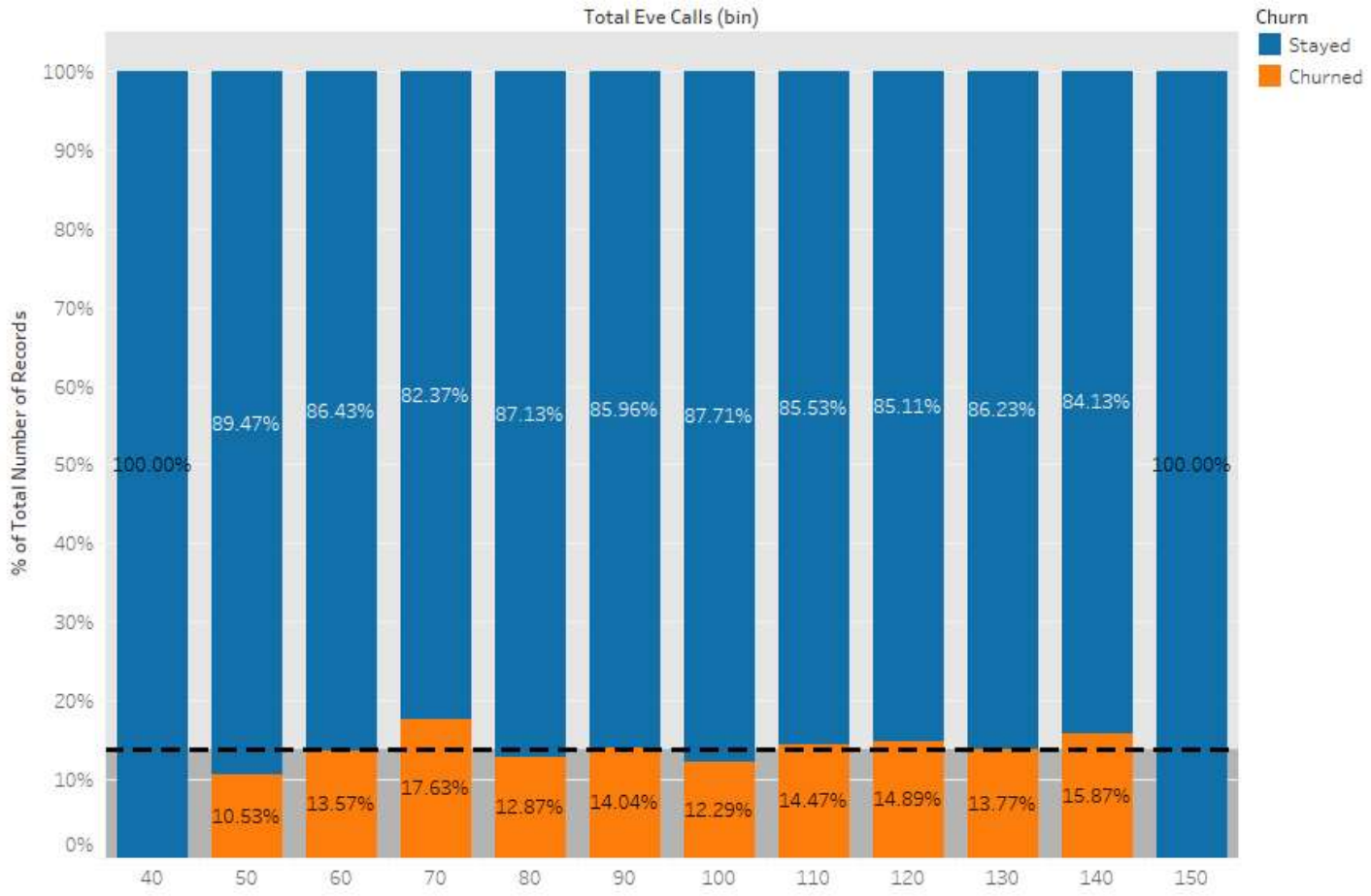
% of Total Number of Records for each Total Day Charge (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that customer who are paying total day charge of 5 to 35 have average churn rate. However, customer who are paying total day charge of 40 to 50 are more likely to churn. This also shows that, as the charge increases the churn rate also increase.



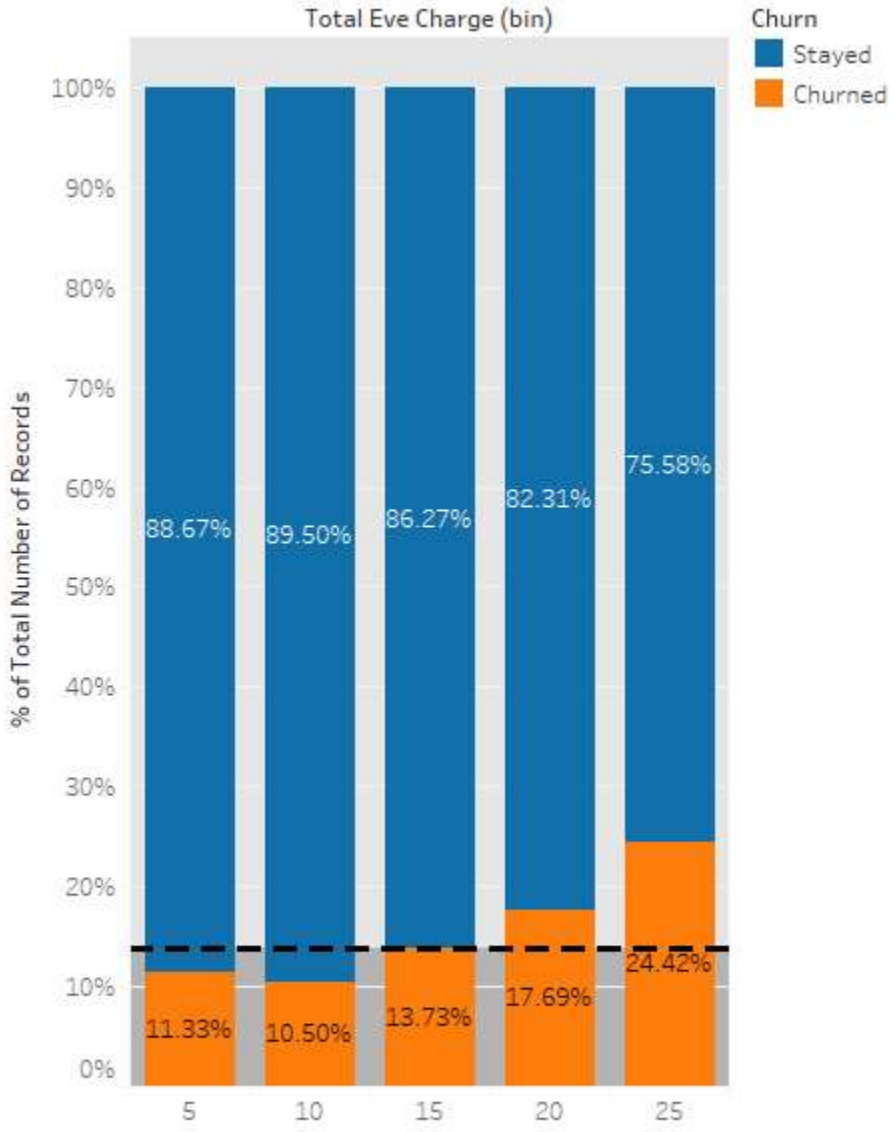
% of Total Number of Records for each Total Day Minutes (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that customer who are using total day minutes of 20 or less are going to stay with current carrier. However, the customer who are using total day minutes of 40 to 240 have average churn rate and customer who are using total day minutes of more than 240 more likely to churn.



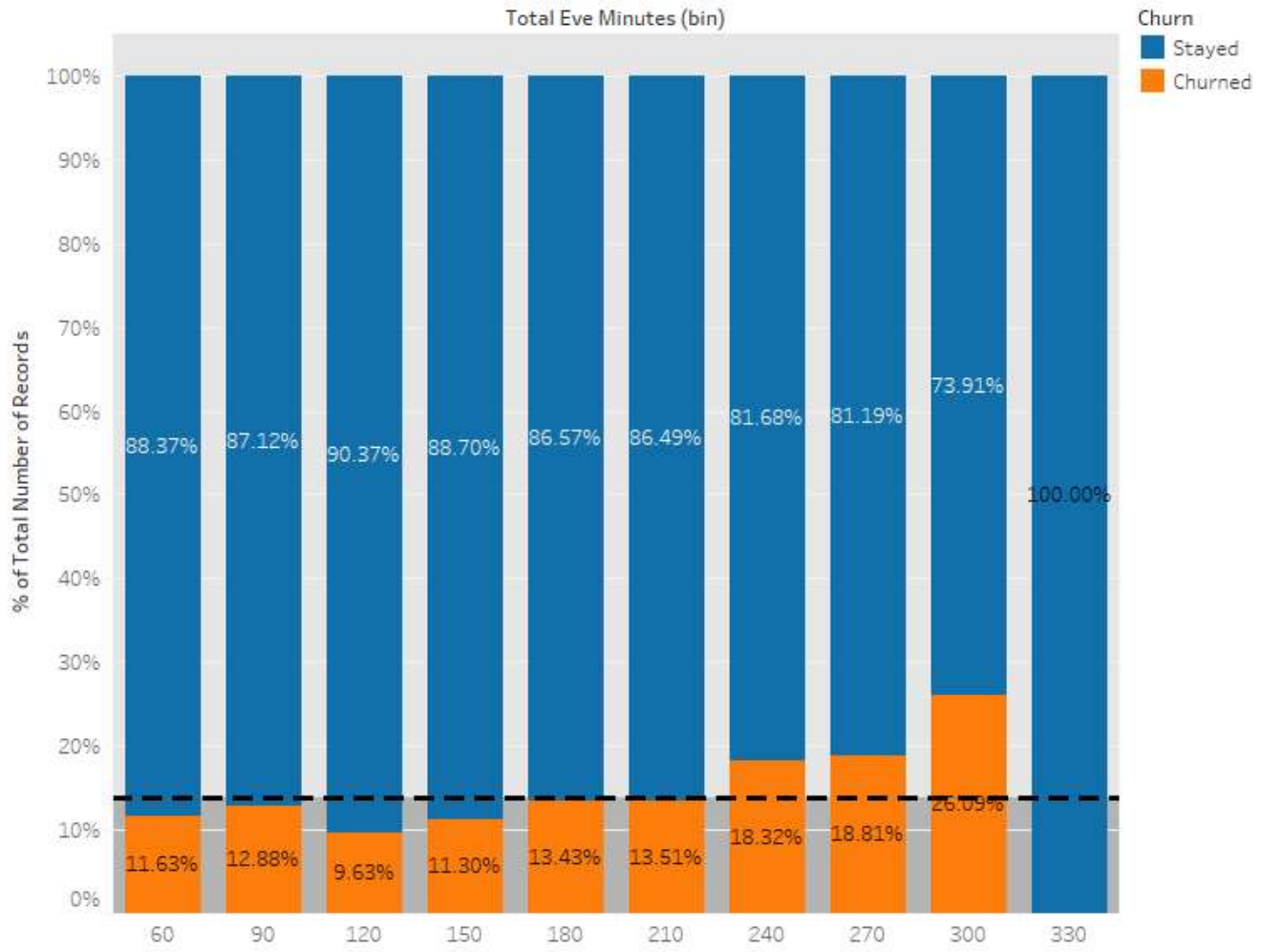
% of Total Number of Records for each Total Eve Calls (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that total eve call are average between 50 to 140. This tells that total eve call does not have any great impact on churn rate.



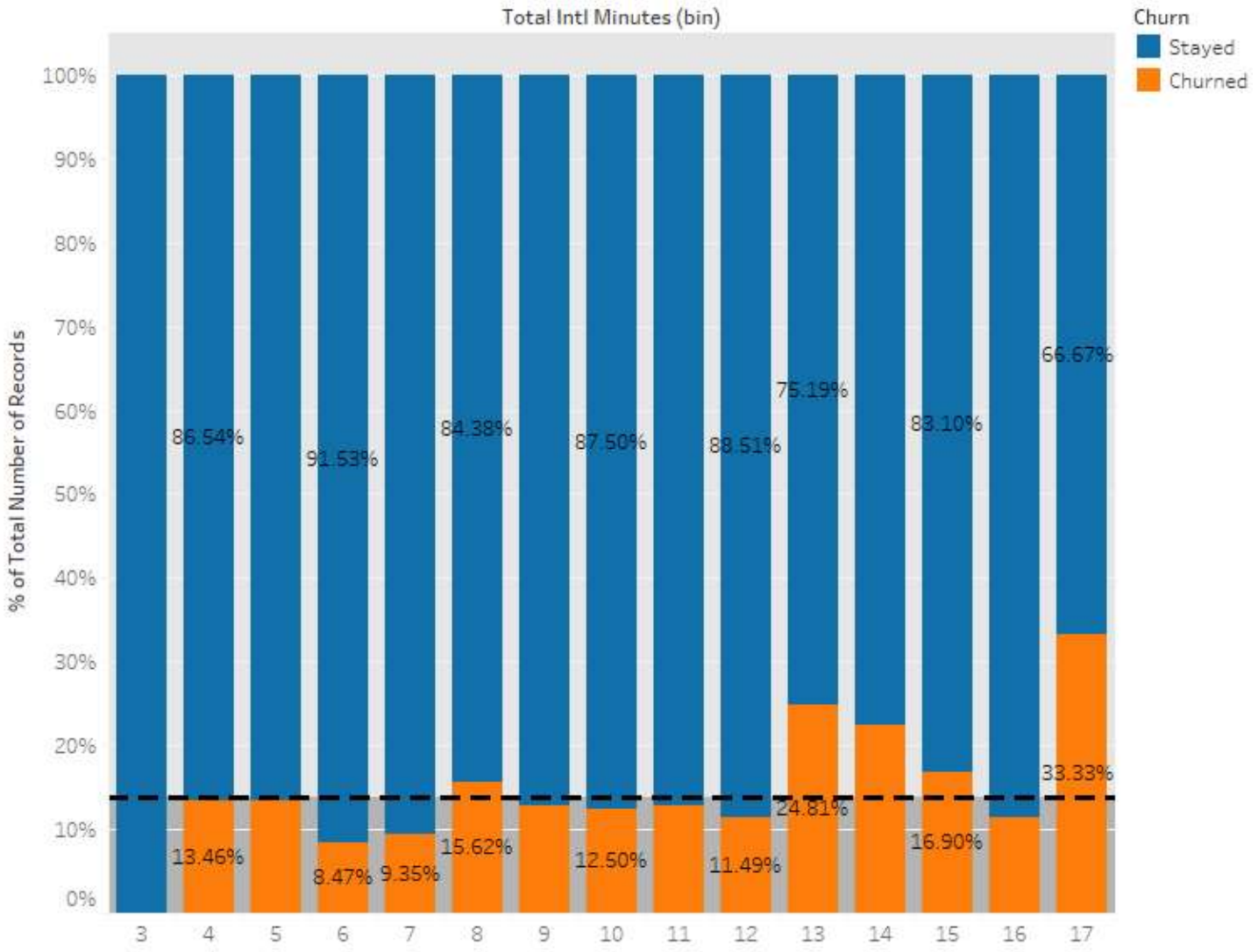
% of Total Number of Records for each Total Eve Charge (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that customer who are paying total eve charge of 5 to 15 have average churn rate. However, customer who are paying total eve charge of 20 to 25 are more likely to churn. This also shows that, as the charge increases the churn rate also increase.



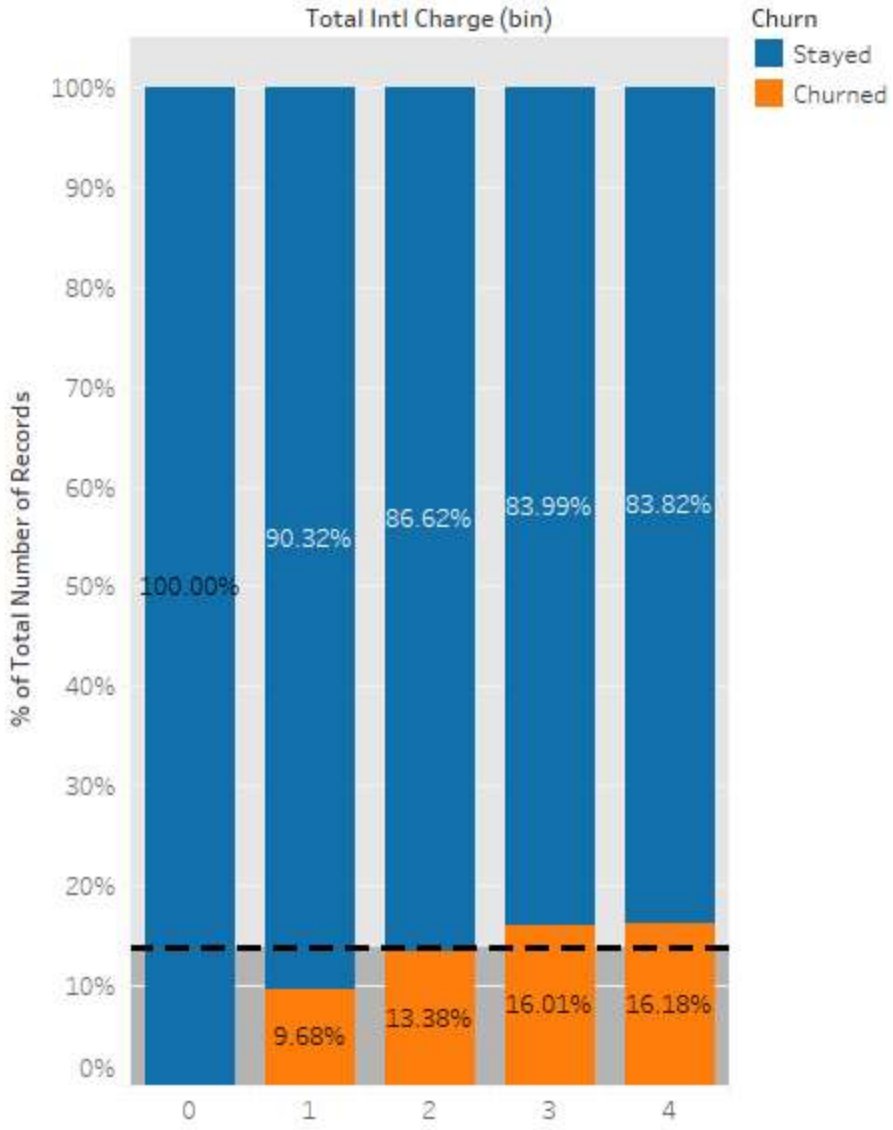
% of Total Number of Records for each Total Eve Minutes (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that customer who are using total day minutes of 60 to 210 have average churn rate and customer who are using total eve minutes of more than 240 are more likely to churn. However, customer who are using total eve minutes of 330 are staying with current carrier.



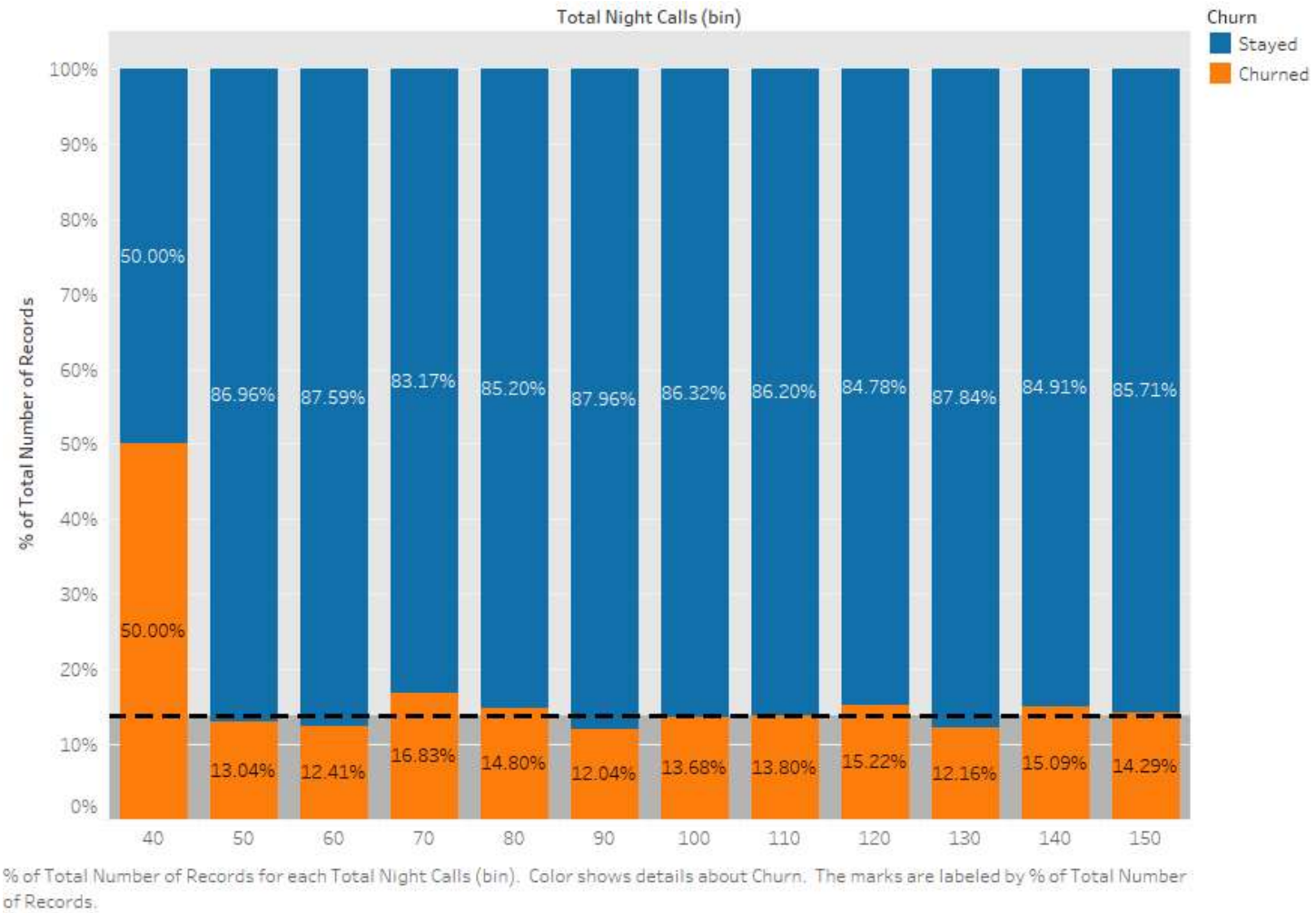
% of Total Number of Records for each Total Intl Minutes (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that customer who are using total Intl minutes of 3 are going to stay with current carrier and customer who are using total Intl minutes 4 to 12 and 15 have average churn rate and customer who are using total Intl minutes of more than 12 are more likely to churn.

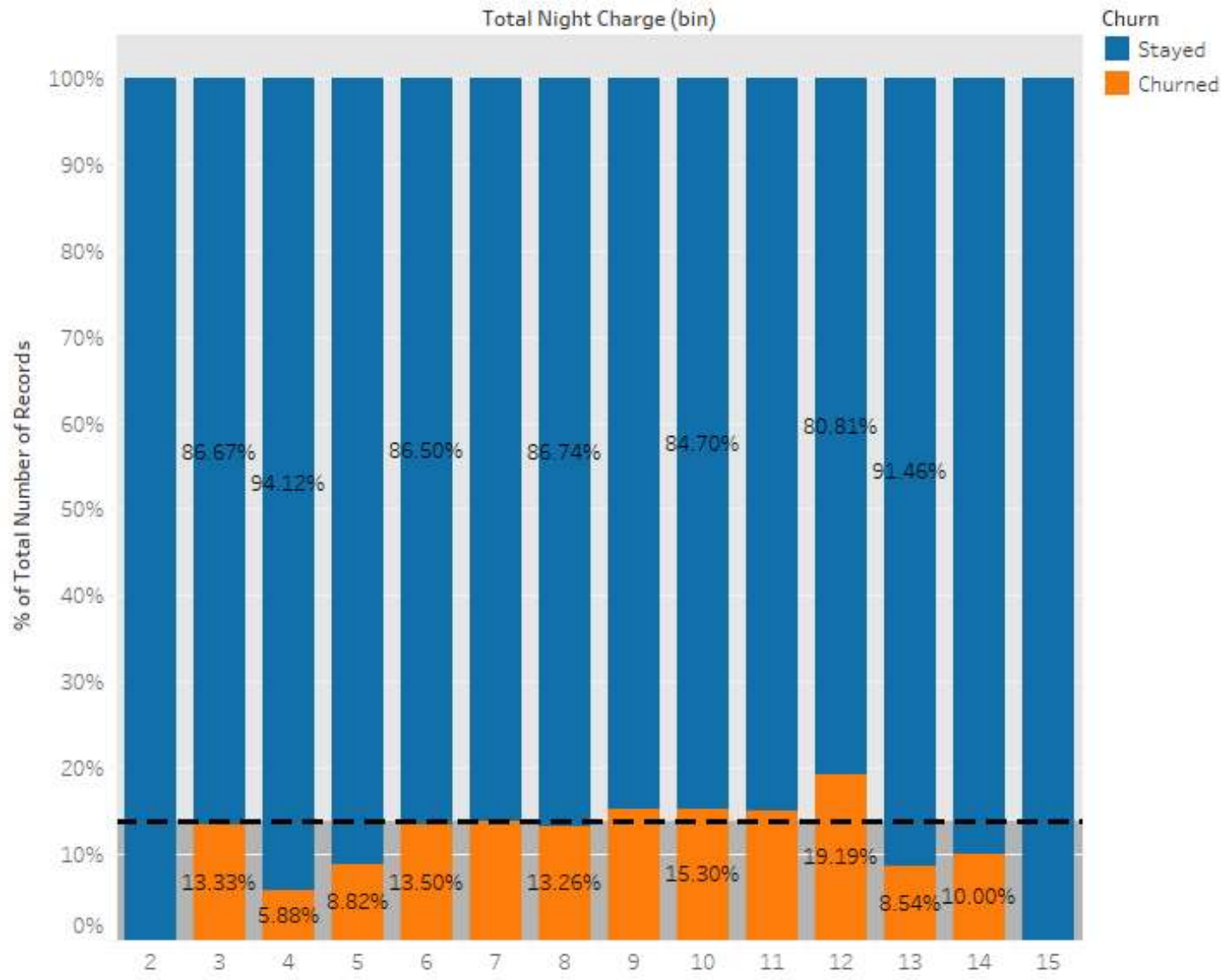


% of Total Number of Records for each Total Intl Charge (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

From the above histogram, we can see that total Intl charge have average churn rate throughout. This tells that total Intl charge does not have any impact on churn rate.

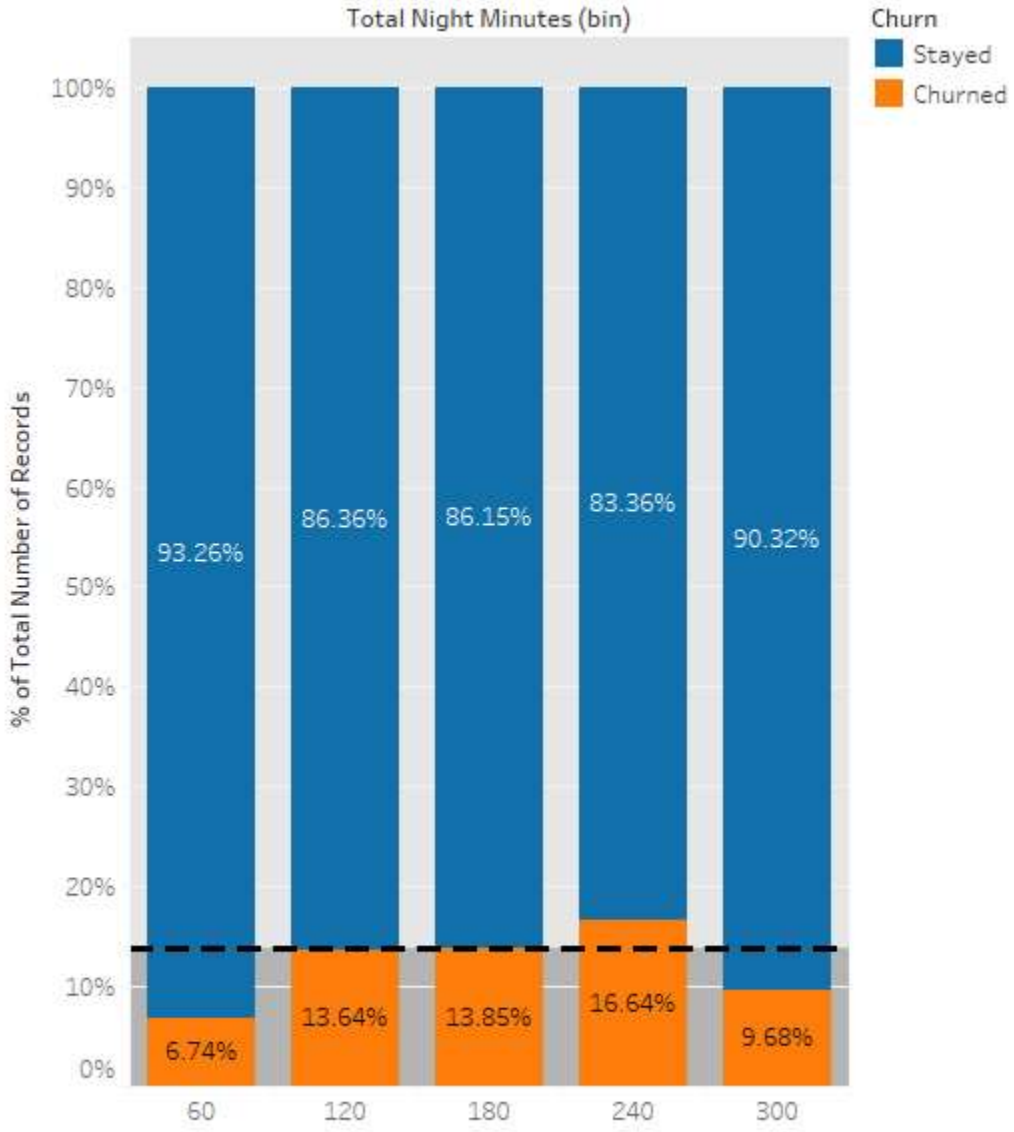


In Total night calls except the first group all other group have average churn rate. This tells that Total night calls does not have any impact on churn rate.



% of Total Number of Records for each Total Night Charge (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

In Total night charge all group have average churn rate except first and last. This tells that Total night charge does not have any impact on churn rate.



% of Total Number of Records for each Total Night Minutes (bin). Color shows details about Churn. The marks are labeled by % of Total Number of Records.

In Total night minutes all group have less or average churn. This tells that Total night minutes does not have any impact on churn rate.

CORRELATION MATRIX

By preparing a correlation matrix, we can have a more straightforward view of what variables are strongly correlated and what is weakly correlated.

	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_eve_calls	total_eve_charge	total_night_minutes	total_night_calls	total_night_charge	total_intl_minutes	total_intl_charge	Churn
account_length	1.000000	-0.014316	0.032941	0.004475	-0.004183	0.012837	0.031849	0.012836	-0.000696	0.026001	-0.000673	-0.015933	0.005828	-0.015935	0.030732	0.030761	0.017277
area_code	-0.014316	1.000000	0.046011	-0.017743	-0.016304	0.000747	-0.014288	0.000746	-0.000497	-0.013558	-0.000472	-0.009018	0.011095	-0.009042	-0.016244	-0.016371	0.015958
international_plan	0.032941	0.046011	1.000000	0.016384	0.018462	0.050615	0.028495	0.050614	0.007101	-0.005216	0.007106	-0.029605	0.016908	-0.029601	0.042471	0.042358	0.264462
voice_mail_plan	0.004475	-0.017743	0.016384	1.000000	0.957333	0.007633	-0.010701	0.007633	0.017724	-0.007798	0.017736	0.000497	0.006874	0.000494	-0.000672	-0.000624	-0.081602
number_vmail_messages	-0.004183	-0.016304	0.018462	0.957333	1.000000	0.008929	-0.007166	0.008929	0.013905	-0.008622	0.013923	0.004837	-0.006012	0.004830	0.000463	0.000501	-0.069847
total_day_minutes	0.012837	0.000747	0.050615	0.007633	0.008929	1.000000	0.009879	1.000000	0.007706	0.011802	0.007681	0.003849	0.021449	0.003824	-0.010354	-0.010327	0.165776
total_day_calls	0.031849	-0.014288	0.028495	-0.010701	-0.007166	0.009879	1.000000	0.009882	-0.022627	0.039907	-0.022615	0.021317	-0.023684	0.021302	0.016301	0.016408	0.027769
total_day_charge	0.012836	0.000746	0.050614	0.007633	0.008929	1.000000	0.009882	1.000000	0.007711	0.011807	0.007685	0.003848	0.021451	0.003823	-0.010356	-0.010329	0.165773
total_eve_minutes	-0.000696	-0.000497	0.007101	0.017724	0.013905	0.007706	-0.022627	0.007711	1.000000	-0.007769	1.000000	-0.018653	-0.010492	-0.018680	-0.010598	-0.010646	0.082494
total_eve_calls	0.026001	-0.013558	-0.005216	-0.007798	-0.008622	0.011802	0.039907	0.011807	-0.007769	1.000000	-0.007775	0.016816	-0.003204	0.016868	-0.003876	-0.003923	-0.004029
total_eve_charge	-0.000673	-0.000472	0.007106	0.017736	0.013923	0.007681	-0.022615	0.007685	1.000000	-0.007775	1.000000	-0.018659	-0.010487	-0.018686	-0.010614	-0.010662	0.082483
total_night_minutes	-0.015933	-0.009018	-0.029605	0.000497	0.004837	0.003849	0.021317	0.003848	-0.018653	0.016816	-0.018659	1.000000	-0.001820	0.999999	-0.004107	-0.004027	0.043667
total_night_calls	0.005828	0.011095	0.016908	0.006874	-0.006012	0.021449	-0.023684	0.021451	-0.010492	-0.003204	-0.010487	-0.001820	1.000000	-0.001848	-0.004302	-0.004302	-0.002924
total_night_charge	-0.015935	-0.009042	-0.029601	0.000494	0.004830	0.003824	0.021302	0.003823	-0.018680	0.016868	-0.018686	0.999999	-0.001848	1.000000	-0.004104	-0.004024	0.043646
total_intl_minutes	0.030732	-0.016244	0.042471	-0.000672	0.000463	-0.010354	0.016301	-0.010356	-0.010598	-0.003876	-0.010614	-0.004107	-0.004302	-0.004104	1.000000	0.999991	0.078264
total_intl_charge	0.030761	-0.016371	0.042358	-0.000624	0.000501	-0.010327	0.016408	-0.010329	-0.010646	-0.003923	-0.010662	-0.004027	-0.004302	-0.004024	0.999991	1.000000	0.078263
Churn	0.017277	0.015958	0.264462	-0.081602	-0.069847	0.165776	0.027769	0.165773	0.082494	-0.004029	0.082483	0.043667	-0.002924	0.043646	0.078264	0.078263	1.000000

The above correlation matrix shows that “total intl charge” and “total intl minutes” are highly correlated (i.e. 99.9%) so, we can drop one of this variable. “voice_mail_plan”, “number_vmail_messages”, “total_eve_calls” and “total_night_calls” have negative correlation with variable “Churn”. However, other variables, which are positively correlated with “Churn”, are very weakly correlated so we must select the correct variable for correlation by visualizing the distribution, knowing churn percentage for every variable.

MODEL BUILDING

DATA PREPARATION

From all previous analysis, I came to know the variables, which are affecting the churn rate, are:

- State
- International Plan
- total day charge
- total day minutes
- total eve charge
- total eve minutes
- total Intl minutes
- Total night charge

I also did convert the State variable to dummy variables to make machine-learning algorithms understand it better since it is a categorical variable.

```
dum = pd.get_dummies(dft.state)
dum
```

	AK	AL	AR	AZ	CA	CO	CT	DC	DE	FL	...	SD	TN	TX	UT	VA	VT	WA	WI	WV	WY
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

I partition my dataset into two sets, training and testing. Training set will be used to train statistical models and estimate coefficients, while testing set will be used to validate the model we build with the training set. 75% of the complete data is partitioned into training set, sampled uniformly without replacement, and 25% is partitioned in to testing set. Sampling without replacement enables the model we build to extrapolate on the testing data, giving us a better sense of how our statistical models perform.

```
X = dft[['international_plan','total_day_charge','total_day_minutes','total_eve_minutes','total_eve_charge',
        'total_intl_minutes','total_night_charge']]
X= pd.concat([X,dum],axis='columns')
y = dft['Churn']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
X_train.shape
```

```
(2499, 58)
```

```
X_test.shape
```

```
(834, 58)
```

2499 observations in training set and 834 observations in testing set.

PREDICTION

I will be using '3' Machine Learning Algorithm for prediction:-

- Logistic Regression.
- Random Forest Classifier.
- XGBoost Classifier.

For every Machine Learning Algorithm I will be computing K-Fold Cross Validation and confusion matrix to know its accuracy. K-Fold Cross Validation and confusion matrix will tell us which Machine Learning Algorithm is working most efficiently for this model.

K-FOLD CROSS VALIDATION

Cross-validation, sometimes called rotation estimation, or out-of-sample testing is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (called the validation dataset or testing set). The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, in most methods multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (e.g. averaged) over the rounds to give an estimate of the model's predictive performance.

In summary, cross-validation combines (averages) measures of fitness in prediction to derive a more accurate estimate of model prediction performance.

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.
For classification problems, one typically uses stratified k-fold cross-validation, in which the folds are selected so that each fold contains roughly the same proportions of class labels.
In repeated cross-validation, the cross-validation procedure is repeated n times, yielding n random

partitions of the original sample. Then results are again averaged (or otherwise combined) to produce a single estimation.

OpenML generates train-test splits given the number of folds and repeats, so that different users can evaluate their models with the same splits. Stratification is applied by default for classification problems (unless otherwise specified). The splits are given as part of the task description as an ARFF file with the row id, fold number, repeat number and the class (TRAIN or TEST). The uploaded predictions should be labeled with the fold and repeat number of the test instance, so that the results can be properly evaluated and aggregated. OpenML stores both the per fold/repeat results and the aggregated scores.

CONFUSION MATRIX

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

I wanted to create a **"quick reference guide" for confusion matrix terminology** because I couldn't find an existing resource that suited my requirements: compact in presentation, using numbers instead of arbitrary variables, and explained both in terms of formulas and sentences.

Let's start with an **example confusion matrix for a binary classifier** (though it can easily be extended to the case of more than two classes):

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

What can we learn from this matrix?

There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.

The classifier made 165 predictions (e.g., 165 patients were being tested for the presence of that disease).

Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.

In reality, 105 patients in the sample have the disease, and 60 patients do not.

Let's now define the most basic terms, which are whole numbers (not rates):

True positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.

True negatives (TN): We predicted no, and they don't have the disease.

False positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")

False negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

I've added these terms to the confusion matrix, and also added the row and column totals:

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

Accuracy: Overall, how often is the classifier correct?

(TP+TN)/total = (100+50)/165 = 0.91

Misclassification Rate: Overall, how often is it wrong?

(FP+FN)/total = (10+5)/165 = 0.09

equivalent to 1 minus Accuracy

also known as "Error Rate"

True Positive Rate: When it's actually yes, how often does it predict yes?

TP/actual yes = 100/105 = 0.95

also known as "Sensitivity" or "Recall"

False Positive Rate: When it's actually no, how often does it predict yes?

FP/actual no = 10/60 = 0.17

True Negative Rate: When it's actually no, how often does it predict no?

TN/actual no = 50/60 = 0.83

equivalent to 1 minus False Positive Rate

also known as "Specificity"

Precision: When it predicts yes, how often is it correct?

TP/predicted yes = 100/110 = 0.91

Prevalence: How often does the yes condition actually occur in our sample?

actual yes/total = 105/165 = 0.64

LOGISTIC REGRESSION

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one, more nominal, ordinal, interval, or ratio-level independent variables.

Sometimes logistic regressions are difficult to interpret; the Intellects Statistics tool easily allows you to conduct the analysis, and then in plain English interprets the output.

The dependent variable should be dichotomous in nature (e.g., presence vs. absent).

There should be no outliers in the data, which can be assessed by converting the continuous predictors to standardized scores, and removing values below -3.29 or greater than 3.29.

There should be no high correlations (multicollinearity) among the predictors. This can be assessed by a correlation matrix among the predictors. Tabachnick and Fidell (2013) suggest that as long correlation coefficients among independent variables are less than 0.90 the assumption is met.

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
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=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

COMPUTING K-FOLD CROSS VALIDATION FOR LOGISTIC REGRESSION ALGORITHM

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)
print('accuracies.mean:-')
print(accuracies.mean())
print('accuracies.std:-')
print(accuracies.std())
```

```
accuracies.mean:-
0.861125982816
accuracies.std:-
0.0111268666275
```

COMPUTING CONFUSION MATRIX FOR LOGISTIC REGRESSION ALGORITHM

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, classifier.predict(X_test))
cm
```

```
array([[709,  17],
       [ 95,  13]], dtype=int64)
```

- **Accuracy=**(TP+TN)/total = (709+13)/834 = 0.86
- **Misclassification Rate=**(FP+FN)/total = (95+17)/834 = 0.13
- **True Positive Rate=**TP/actual yes = 95/108 = 0.879
- **False Positive Rate=**FP/actual no = 17/726 = 0.17
- **True Negative Rate=**TN/actual no = 709/726 = 0.023
- **Precision=**TP/predicted yes = 13/30 = 0.43
- **Prevalence=**actual yes/total = 108/834 = 0.129

RANDOM FOREST CLASSIFIER

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks. In this post, you are going to learn, how the random forest algorithm works and several other important things about it.

Random Forest is a supervised learning algorithm. Like you can already see from its name, it creates a forest and makes it somehow random. The „forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the building block of machine learning.

Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, you don't have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Like I already said, with Random Forest, you can also deal with Regression tasks by using the Random Forest regressor.

Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

```
from sklearn.ensemble import RandomForestClassifier
classifier3 = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier3.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, 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=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

COMPUTING K-FOLD CROSS VALIDATION FOR RANDOM FOREST CLASSIFIER ALGORITHM

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier3, X = X_train, y = y_train, cv = 10)
print('accuracies.mean:-')
print(accuracies.mean())
print('accuracies.std:-')
print(accuracies.std())
```

```
accuracies.mean:-
0.890334450151
accuracies.std:-
0.0136913440245
```

COMPUTING CONFUSION MATRIX FOR RANDOM FOREST CLASSIFIER ALGORITHM

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, classifier3.predict(X_test))
cm
```

```
array([[715,  11],
       [ 63,  45]], dtype=int64)
```

- **Accuracy=**(TP+TN)/total = (45+715)/834 = 0.91
- **Misclassification Rate=**(FP+FN)/total = (11+63)/834 = 0.089
- **True Positive Rate=**TP/actual yes = 45/108 = 0.42
- **False Positive Rate=**FP/actual no = 11/726 = 0.015
- **True Negative Rate=**TN/actual no = 715/726 = 0.98
- **Precision=**TP/predicted yes = 45/56 = 0.80
- **Prevalence=**actual yes/total = 108/834 = 0.129

XGBOOST REGRESSOR

XGBoost is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine-learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

GRADIENT BOOSTING

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman, simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean. The latter two papers introduced the view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

```
from xgboost import XGBClassifier
classifier2 = XGBClassifier()
classifier2.fit(X_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

COMPUTING K-FOLD CROSS VALIDATION FOR XGBOOST CLASSIFIER ALGORITHM

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier2, X = X_train, y = y_train, cv = 10)
print('accuracies.mean:-')
print(accuracies.mean())
print('accuracies.std:-')
print(accuracies.std())
```

```
accuracies.mean:-
0.896769727516
accuracies.std:-
0.012031733791
```

COMPUTING CONFUSION MATRIX FOR XGBOOST CLASSIFIER ALGORITHM

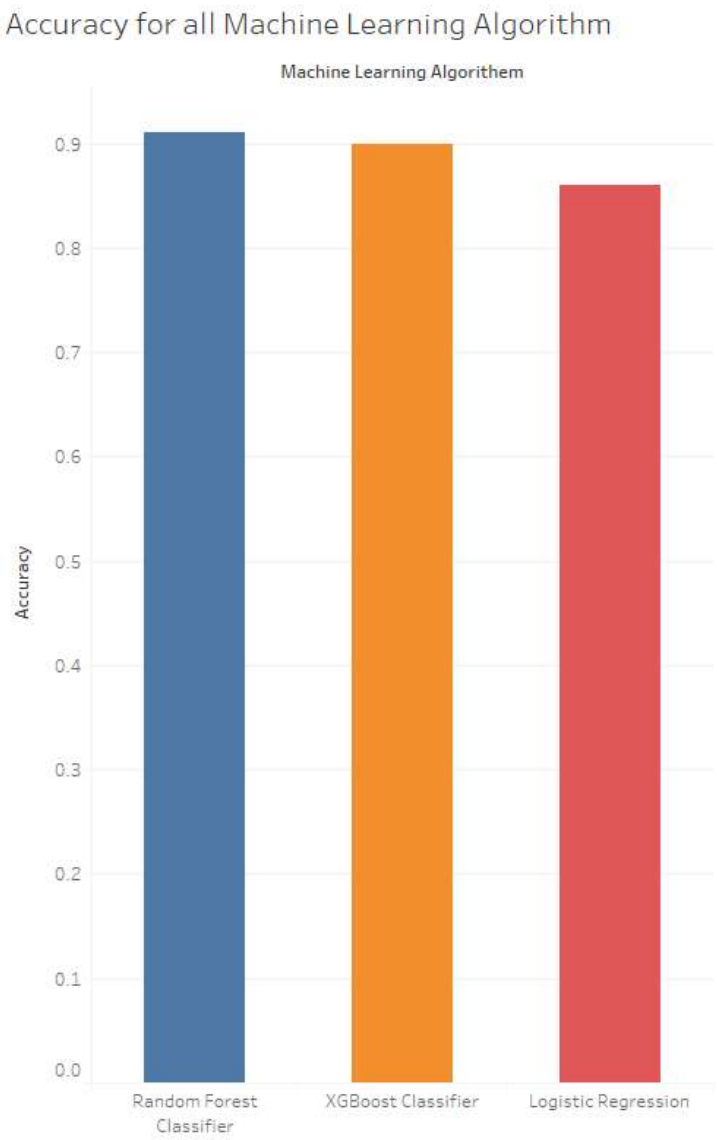
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, classifier2.predict(X_test))
cm
```

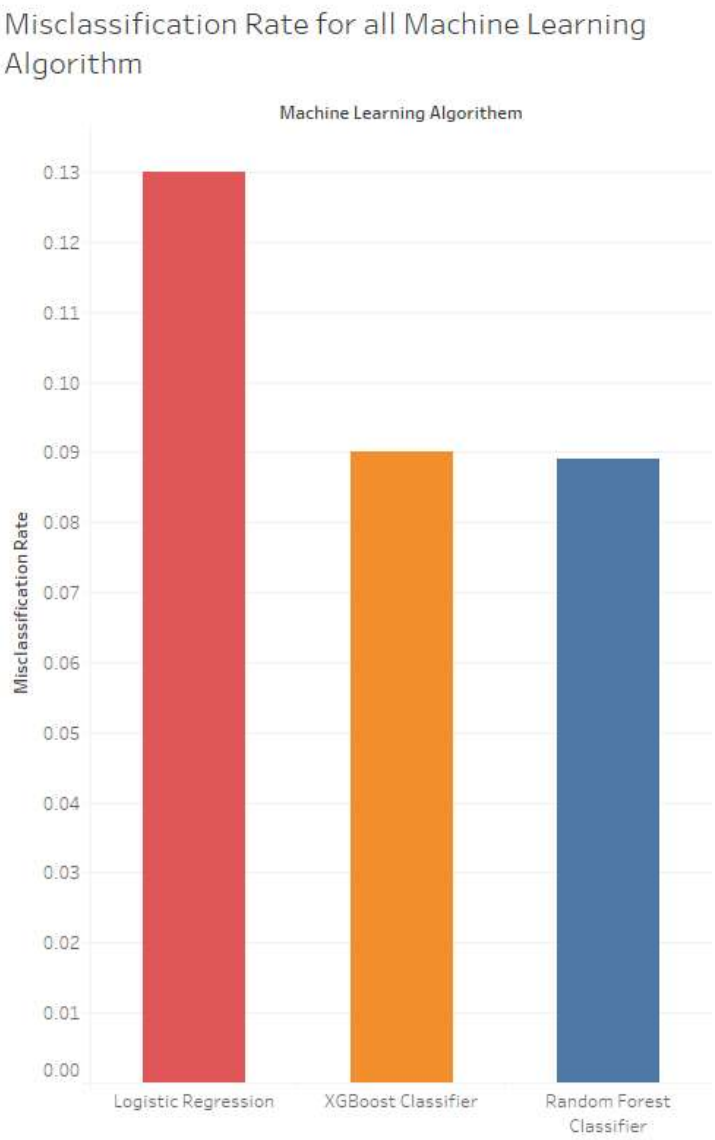
```
array([[712,  14],
       [ 69,  39]], dtype=int64)
```

- **Accuracy=**(TP+TN)/total = (712+39)/834 = 0.90
- **Misclassification Rate=**(FP+FN)/total = (14+69)/834 = 0.09
- **True Positive Rate=**TP/actual yes = 39/108 = 0.35
- **False Positive Rate=**FP/actual no = 14/726 = 0.019
- **True Negative Rate=**TN/actual no = 712/726 = 0.93
- **Precision=**TP/predicted yes = 39/53 = 0.73
- **Prevalence=**actual yes/total = 108/834 = 0.129

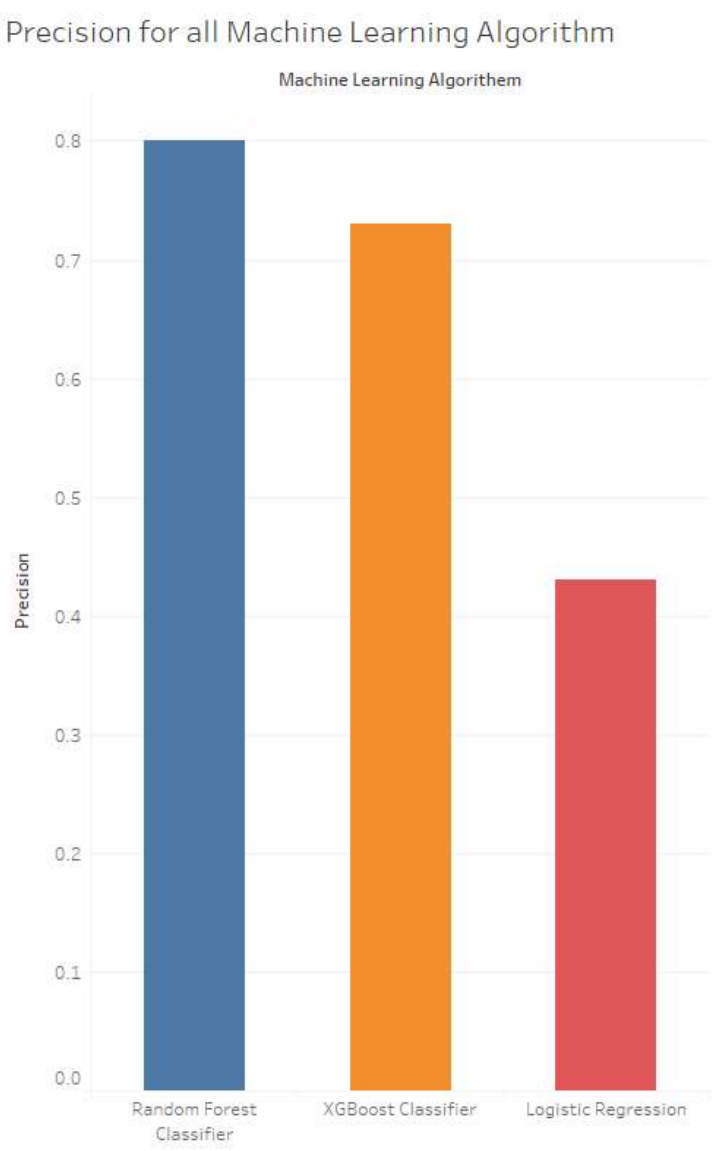
CONCLUSION

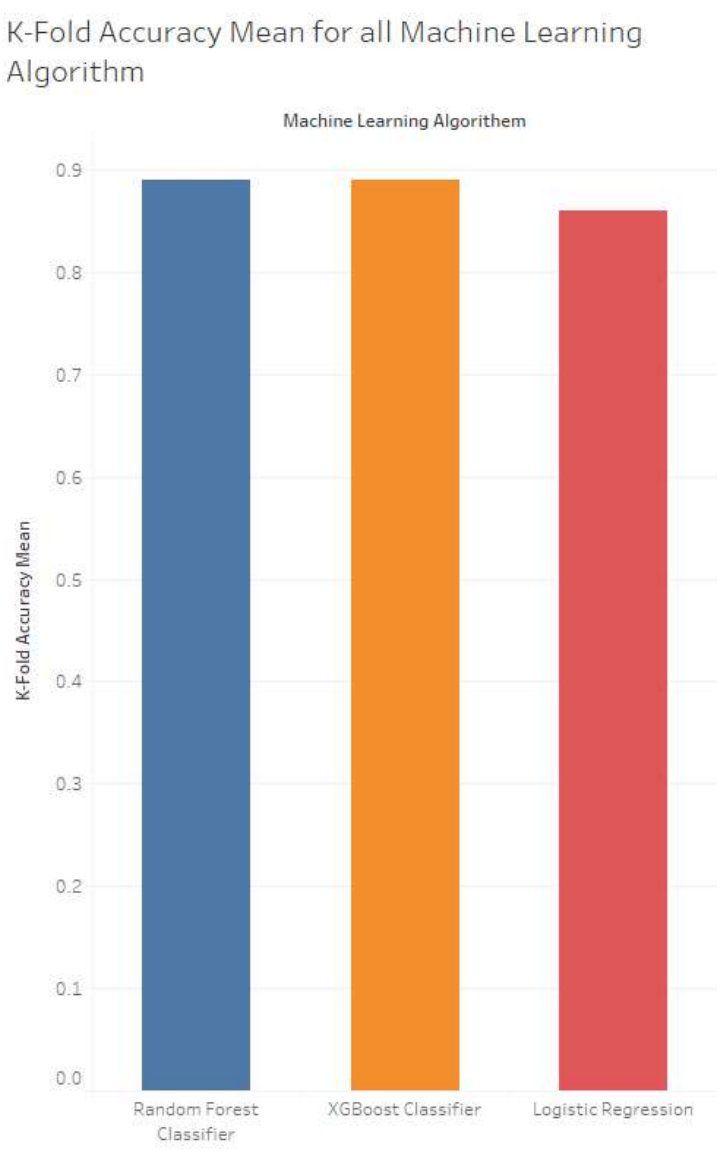
Comparing the Accuracies obtained from K-Fold Cross Validation and Confusion Matrix for all machine-learning algorithms.





From the above graph, we can see that Random Forest Classifier has the highest accuracy and lowest miss classification rate in comparison with other machine learning algorithms for this model.





From the above graph, we can see that Random Forest Classifier has the highest K-Fold accuracy Mean and highest Precision in comparison with other machine learning algorithms for this model.

The predictions efficiency can be improved by using Artificial Neural Networks and Convolutional Neural Networks. But, due limitation of time and contains of course I was able to predict the data up to 91% accuracy with Random Forest Classifier.

PYTHON CODE RAN ON JUPYTER NOTEBOOK:-

```
import matplotlib.pyplot as plt
import numpy as np
import os
os.chdir("E:\\R WorkStation\\Python WorkStation")
os.getcwd()
import pandas as pd
dft= pd.read_csv("Train_data.csv", sep = ',')
dft.head(1)
dft.shape
dft.isnull().any().any()
dft.columns
dft.columns=dft.columns.str.replace(' ','_')
dft.head(1)
dft.account_length=pd.to_numeric(dft.account_length)
dft.total_day_minutes=pd.to_numeric(dft.total_day_minutes)
dft.total_day_calls=pd.to_numeric(dft.total_day_calls)
dft.total_day_charge=pd.to_numeric(dft.total_day_charge)
dft.total_eve_minutes=pd.to_numeric(dft.total_eve_minutes)
dft.total_eve_calls=pd.to_numeric(dft.total_eve_calls)
dft.total_eve_charge=pd.to_numeric(dft.total_eve_charge)
dft.total_night_minutes=pd.to_numeric(dft.total_night_minutes)
dft.total_night_calls=pd.to_numeric(dft.total_night_calls)
dft.total_night_charge=pd.to_numeric(dft.total_night_charge)
```

```
dft.total_intl_minutes=pd.to_numeric(dft.total_intl_minutes)
dft.total_intl_calls=pd.to_numeric(dft.total_intl_calls)
dft.total_intl_charge=pd.to_numeric(dft.total_intl_charge)
dft.head(1)
dft.international_plan=dft.international_plan.astype('category')
dft.voice_mail_plan=dft.voice_mail_plan.astype('category')
dft.Churn=dft.Churn.astype('category')
dft.international_plan=dft.international_plan.cat.codes
dft.voice_mail_plan=dft.voice_mail_plan.cat.codes
dft.Churn=dft.Churn.cat.codes
dft.head(1)
dft=dft.drop(columns=['total_intl_calls', 'number_customer_service_calls'])
dft[(dft.account_length>205) | (dft.account_length<1)]=np.nan
dft[(dft.number_vmail_messages>49) | (dft.number_vmail_messages<0)]=np.nan
dft[(dft.total_day_minutes>324.7) | (dft.total_day_minutes<35.1)]=np.nan
dft[(dft.total_eve_minutes>330.6) | (dft.total_eve_minutes<67)]=np.nan
dft[(dft.total_intl_minutes>17.3) | (dft.total_intl_minutes<3.3)]=np.nan
dft[(dft.total_night_minutes>334.7) | (dft.total_night_minutes<64.7)]=np.nan
dft[(dft.total_day_calls>152) | (dft.total_day_calls<47)]=np.nan
dft[(dft.total_eve_calls>151) | (dft.total_eve_calls<49)]=np.nan
dft[(dft.total_night_calls>151) | (dft.total_night_calls<49)]=np.nan
dft[(dft.total_day_charge>52.2) | (dft.total_day_charge<5.97)]=np.nan
dft[(dft.total_eve_charge>28.7) | (dft.total_eve_charge<5.47)]=np.nan
dft[(dft.total_intl_charge>4.67) | (dft.total_intl_charge<0.69)]=np.nan
dft[(dft.total_night_charge>15.06) | (dft.total_night_charge<2.96)]=np.nan
dft=dft.fillna(method='ffill')
```

```
dft.corr()

dum = pd.get_dummies(dft.state)

dum

X = dft[['international_plan','total_day_charge','total_day_minutes','total_eve_minutes','total_eve_charge',

        'total_intl_minutes','total_night_charge']]

X= pd.concat([X,dum],axis='columns')

y = dft['Churn']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

X_test.shape

from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)

classifier.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, classifier.predict(X_test))

cm

from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)

print('accuracies.mean:-')

print(accuracies.mean())

print('accuracies.std:-')

print(accuracies.std())
```

```
from xgboost import XGBClassifier
classifier2 = XGBClassifier()
classifier2.fit(X_train, y_train)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, classifier2.predict(X_test))
cm
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier2, X = X_train, y = y_train, cv = 10)
print('accuracies.mean:-')
print(accuracies.mean())
print('accuracies.std:-')
print(accuracies.std())
from sklearn.ensemble import RandomForestClassifier
classifier3 = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier3.fit(X_train, y_train)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, classifier3.predict(X_test))
cm
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier3, X = X_train, y = y_train, cv = 10)
print('accuracies.mean:-')
print(accuracies.mean())
print('accuracies.std:-')
print(accuracies.std())
```