Telco Customer Churn

In this project, we predict behavior to retain customers. We use supervised learning models to predict customers who are likely to stop using telecommunication service in the future. In addition, we will analyze top factors that influence customer retention.

Data source: https://www.kaggle.com/blastchar/telco-customer-churn (https://www.kaggle.com/blastchar/telco-customer-churn)

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
pd.set_option('display.max_columns', None)
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly as py
import plotly.graph_objs as go
import plotly.offline as pyo
import plotly.figure_factory as ff
pyo.init_notebook_mode()
import pydot
from graphviz import Source
from IPython.display import display
```

In [2]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import RFE
from sklearn.tree import export_graphviz
from scipy.stats import pearsonr
```

In [3]:
```
LABEL = 'Churn' # target column
LABEL_ONE = 'Yes'
LABEL_ZERO = 'No'
```

1. Data Exploration

Each row represents a customer, each column contains customer's attributes described on the column Metadata.

The raw data contains 7043 rows (customers) and 21 columns (features).

The "Churn" column is our target.

The data contains the following columns:

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

1.1 Import and Understand the Raw Data

In [4]:
```
churn_df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

In [5]:
```
churn_df.head(n = 10)
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	No
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes

In [6]:
```
print("Number of rows: " + str(churn_df.shape[0]))
print("Number of columns: " + str(churn_df.shape[1]))
print("\nColumns: \n", churn_df.columns.tolist())
print("\nUnique values of each column: \n", churn_df.nunique())
```

```
Number of rows: 7043
Number of columns: 21

Columns: 
 ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetServic
e', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'Paperl
essBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn']

Unique values of each column: 
 customerID          7043
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                73
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         3
OnlineBackup           3
DeviceProtection       3
TechSupport            3
StreamingTV            3
StreamingMovies        3
Contract               3
PaperlessBilling       2
PaymentMethod          4
MonthlyCharges      1585
TotalCharges        6531
Churn                  2
dtype: int64
```

In [7]:
```
churn_df.dtypes
```

Out[7]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

1.2 Data Cleaning

In [8]:
```
# illustrate how to remove whitespaces in the string

churn_df['PhoneService'] += ' '
churn_df['PhoneService'][0]
```

Out[8]:
```
'No '
```

In [9]:
```
churn_df['PhoneService'] = churn_df['PhoneService'].map(lambda x : x.strip())
churn_df['PhoneService'][0]
```

Out[9]:
```
'No'
```

1.3 Handling the Missing Values

Two common ways to handle missing values:

1. We delete a particular row if it has a null value for a particular feature and a particular column if it has more than 75% of missing values. This method is advised only when there are enough samples in the data set. One has to make sure that after we have deleted the data, there is no addition of bias.
2. This strategy can be applied on a feature which has numeric data. We can calculate the mean, median or mode of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with the above three approximations are a statistical approach of handling the missing values. This method is also called as leaking the data while training. Another way is to approximate it with the deviation of neighbouring values. This works better if the data is linear.

In [10]:
```
# Replace missing values with null values

churn_df = churn_df.replace('^\s*$', np.nan, regex = True)
```

In [11]:
```python
# check for the missing values

churn_df.isnull().sum()
```

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

In [12]:
```python
# Since only 11/7043 of the rows have missing values, we simply drop them
churn_df = churn_df[churn_df['TotalCharges'].notnull()]

# reset the indices since we have deleted some rows
churn_df = churn_df.reset_index()[churn_df.columns] # we don't need the index columns
```

1.4 Data Visualization

In [13]:
```python
# Make a copy of the dataframe

df_vis = churn_df.copy()
```

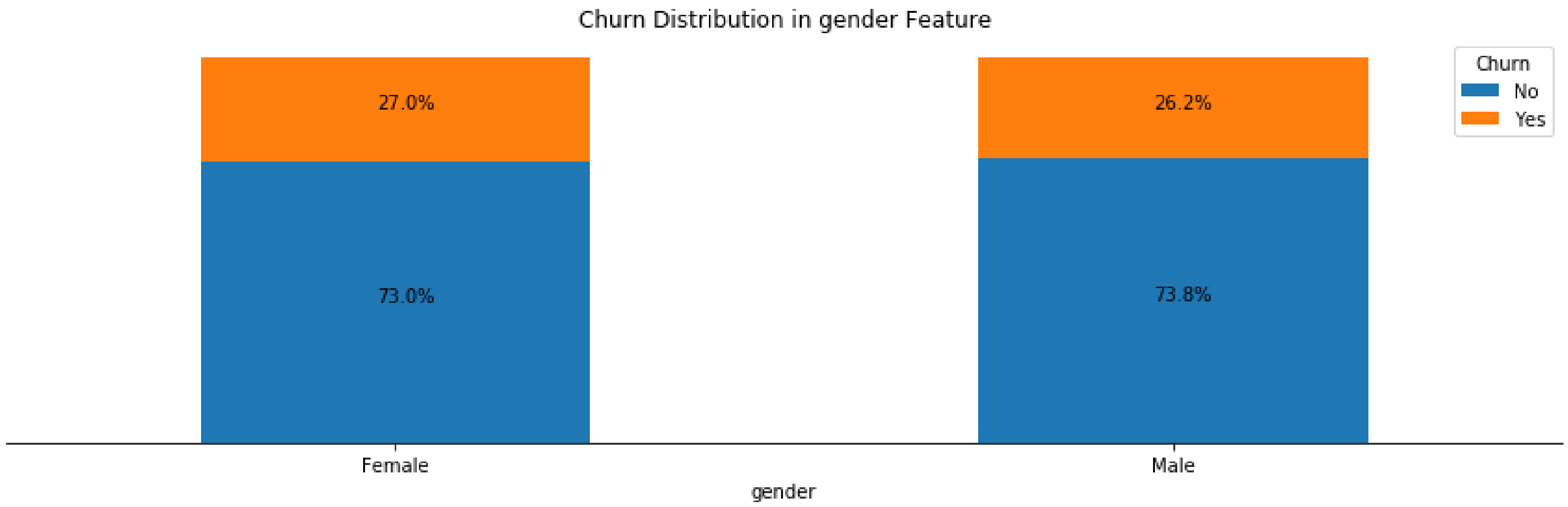
In [14]:
```python
# replace "1 or 0" in column "SeniorCitizen" by "Yes or No"

df_vis["SeniorCitizen"] = df_vis["SeniorCitizen"].replace({1 : "Yes", 0 : "No"})
```

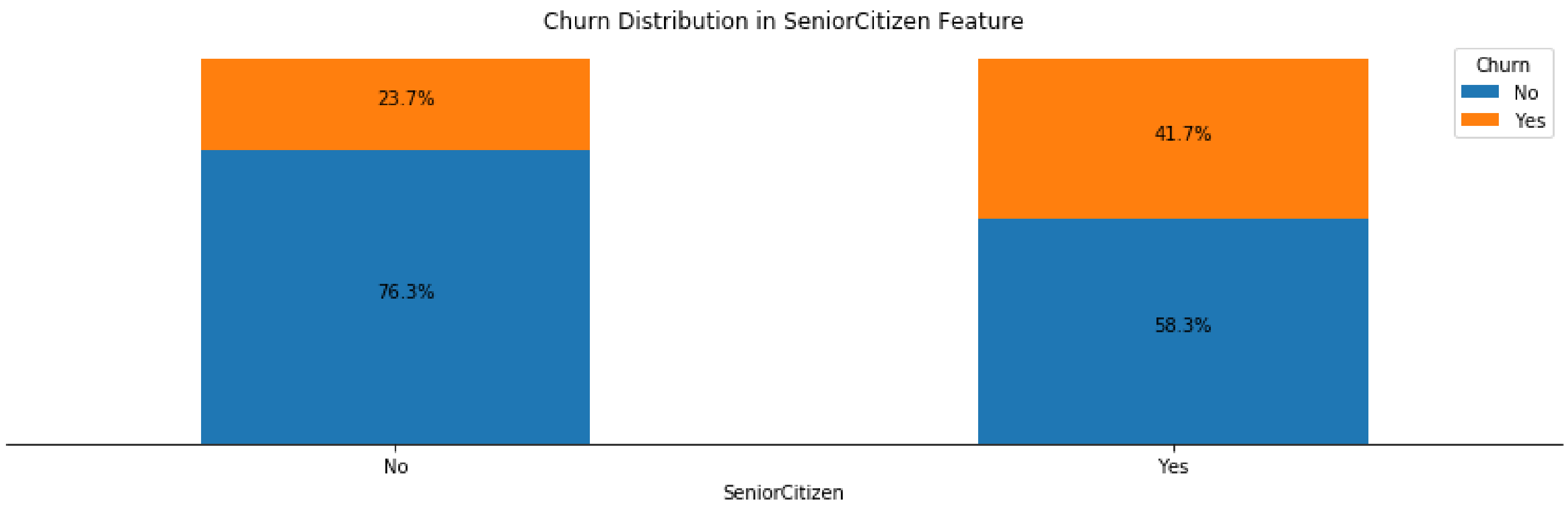
In [15]:
```python
# Simplify "PaymentMethod" column

df_vis["PaymentMethod"] = df_vis["PaymentMethod"].replace({ "Bank transfer (automatic)": "Bank transer",
                                                            "Credit card (automatic)" : "Credit card"})
```

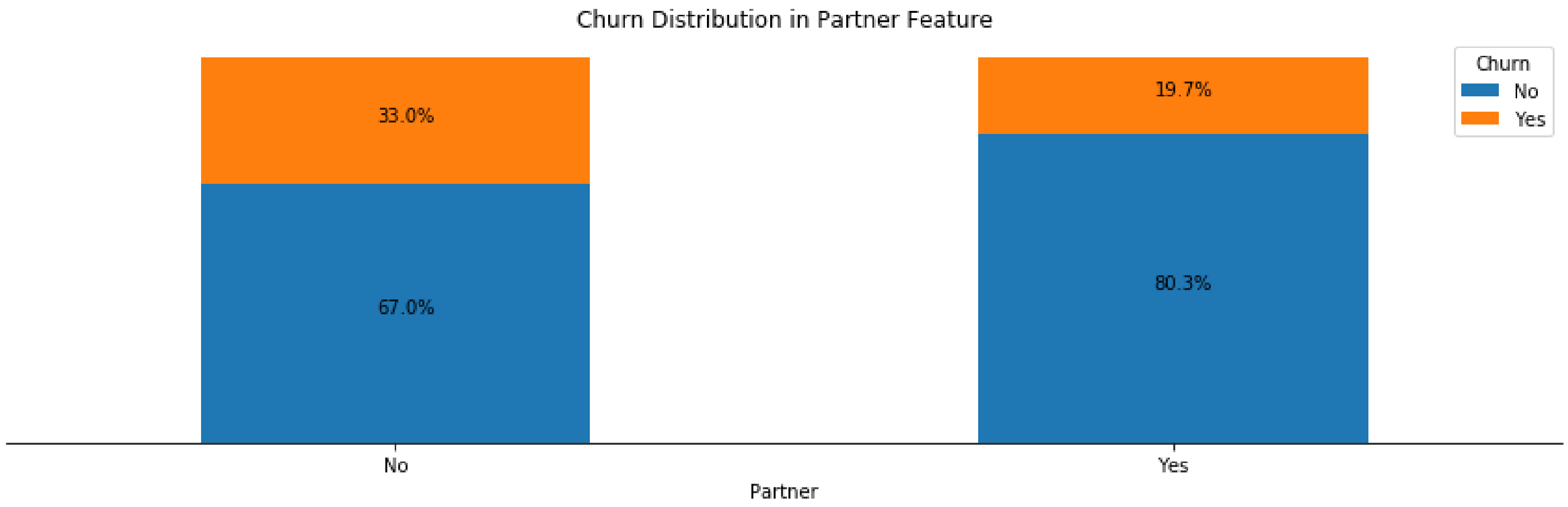
In [16]:
```python
# convert the "TotalCharges" column to float type

df_vis['TotalCharges'] = df_vis['TotalCharges'].astype(float)
```

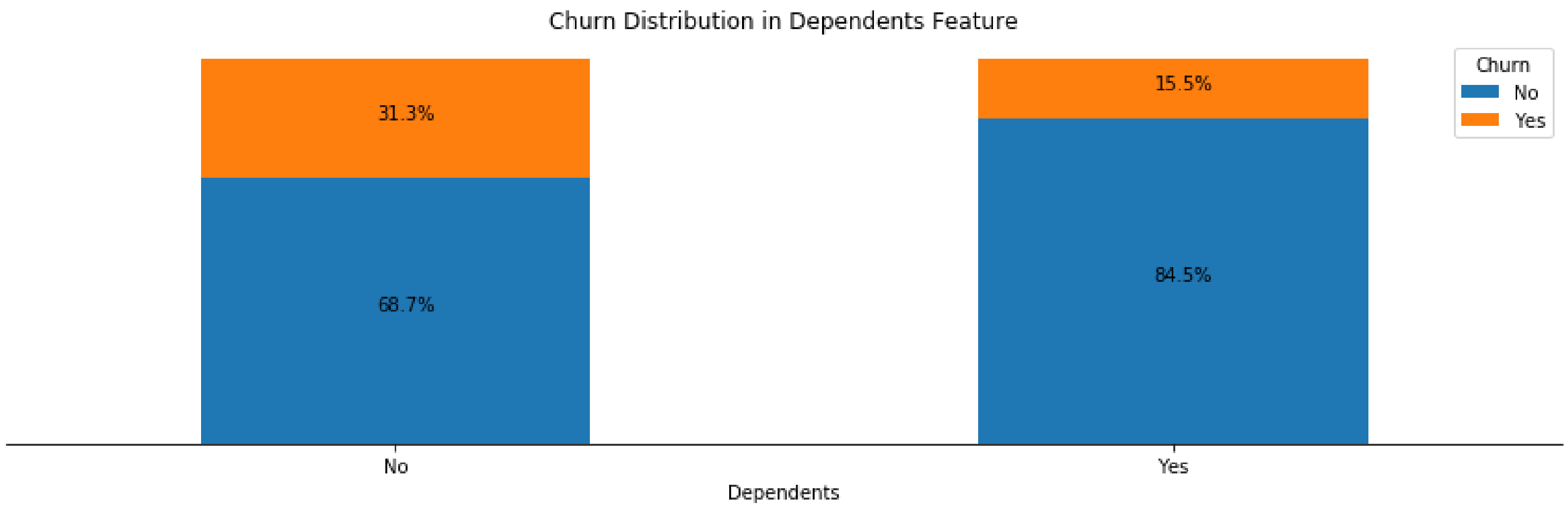
In [17]:
```python
# Set tenure to different ranges, based on number of years

def tenure_range(tenure):
    if tenure <= 12:
        return 'tenure_1_year'
    elif tenure <= 24:
        return 'tenure_2_year'
    elif tenure <= 36:
        return 'tenure_3_year'
    elif tenure <= 48:
        return 'tenure_4_year'
    elif tenure <= 60:
        return 'tenure_5_year'
    else:
        return 'tenure_5_year_more'

df_vis['tenure'] = df_vis['tenure'].map(lambda x : tenure_range(x))
```

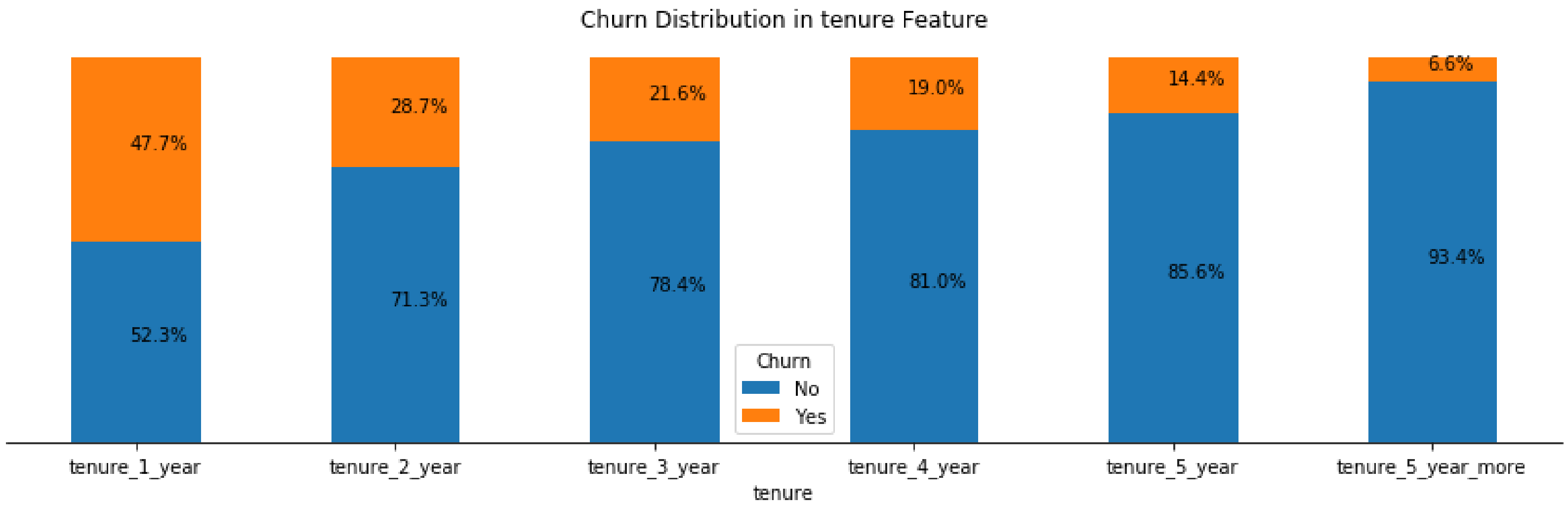
In [18]:
```
# Plot churn and not_churn distribution in each feature

# do not plot these columns
to_drop_vis = ['customerID', 'Churn']

# Categorical features to be plotted
cat_cols_vis = df_vis.nunique()[df_vis.nunique() <= 6].keys().tolist()
cat_cols_vis = [x for x in cat_cols_vis if x not in to_drop_vis]

for col in cat_cols_vis:
    ax = df_vis.groupby(col)['Churn'].value_counts(normalize = True).unstack().plot(kind = 'bar', stacked = True, rot = 0,
                                                                                    figsize = (15,4),
                                                                                    title = 'Churn Distribution in ' + col + '
Feature')
    ax.get_yaxis().set_visible(False)
    ax.spines['left'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

    # put the percentage on the graph
    for i in range(len(ax.patches)):
        p = ax.patches[i]
        if i < len(ax.patches) // 2:
            ax.annotate("{0:.1%}".format(p.get_height()), (p.get_x() + p.get_width() * 0.45, 0.5 * p.get_height()))
        else:
            ax.annotate("{:.1%}".format(p.get_height()), (p.get_x() + p.get_width() * 0.45, 1 - 0.5 * p.get_height()))
```

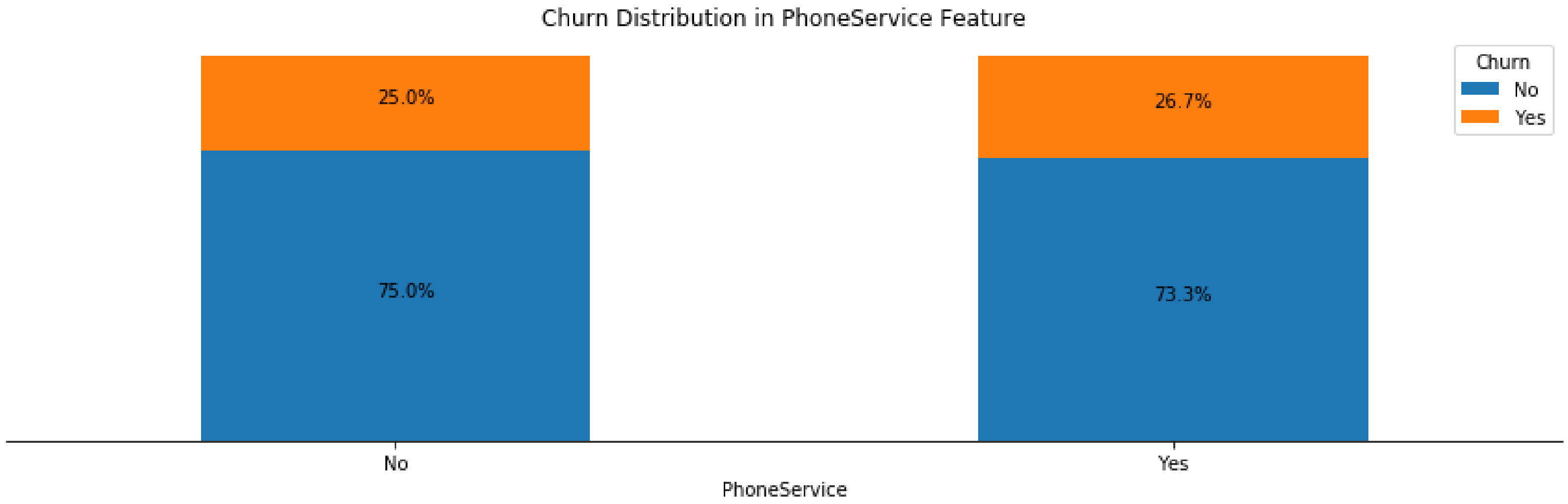

Churn Distribution in gender Feature
Churn
No
Yes
27.0%
73.0%
26.2%
73.8%
Female
Male
gender

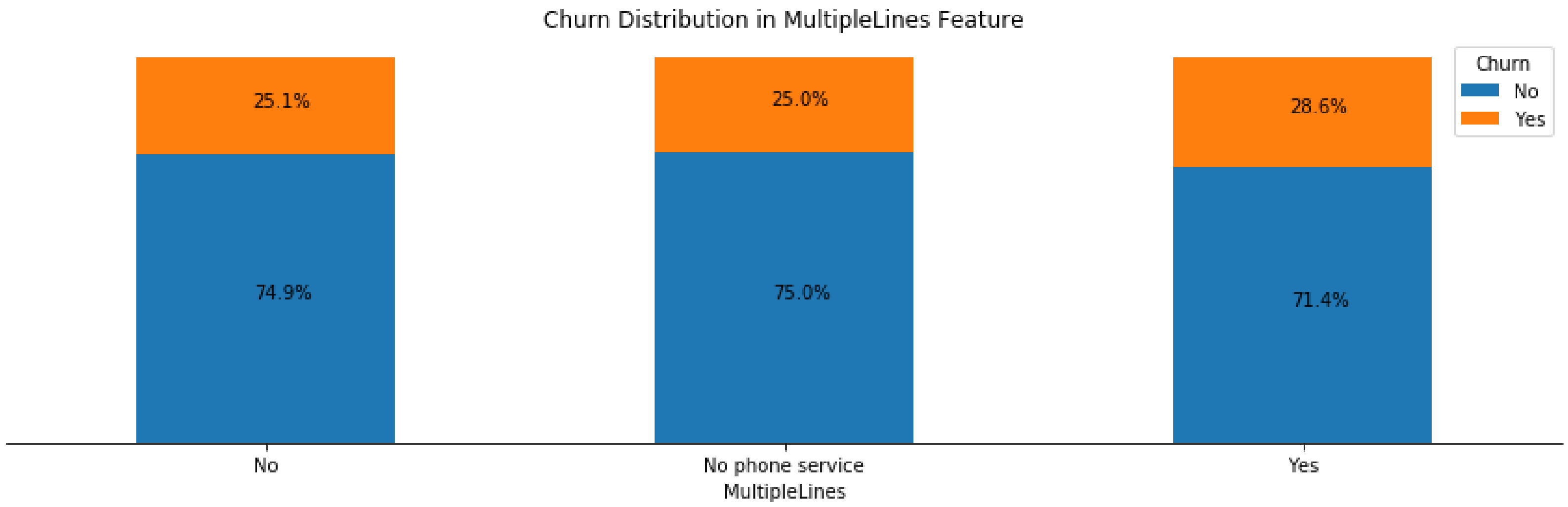

Churn Distribution in SeniorCitizen Feature
Churn
No
Yes
23.7%
76.3%
41.7%
58.3%
No
Yes
SeniorCitizen

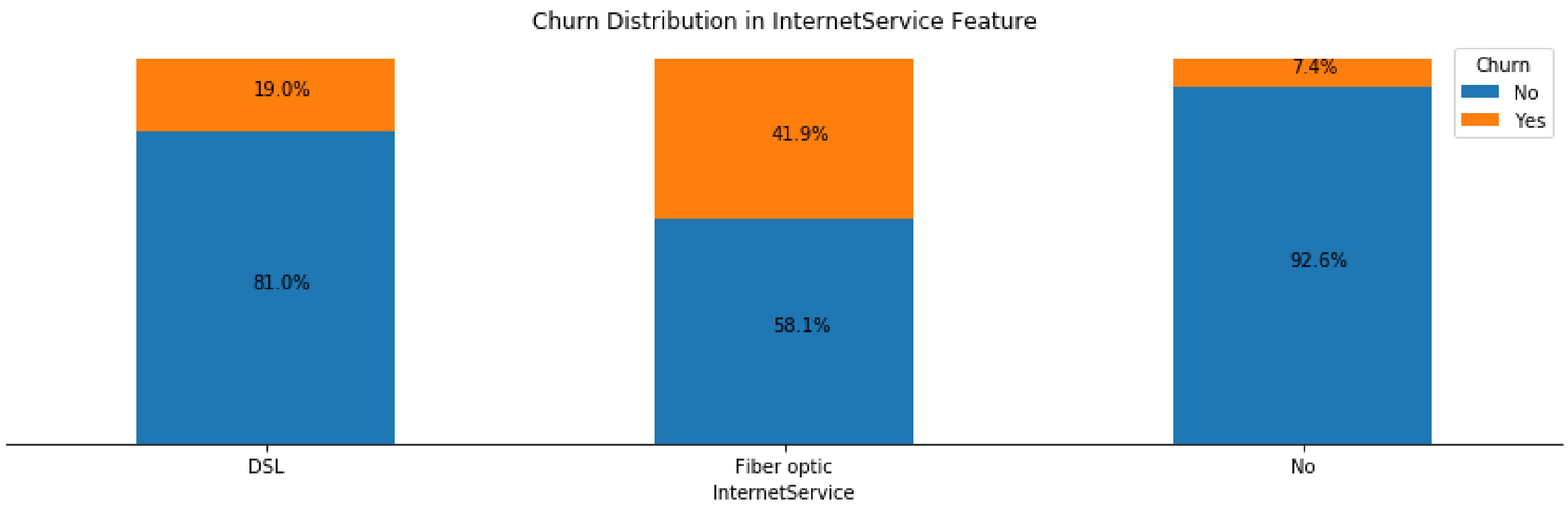

Churn Distribution in Partner Feature
Churn
No
Yes
33.0%
67.0%
19.7%
80.3%
No
Yes
Partner

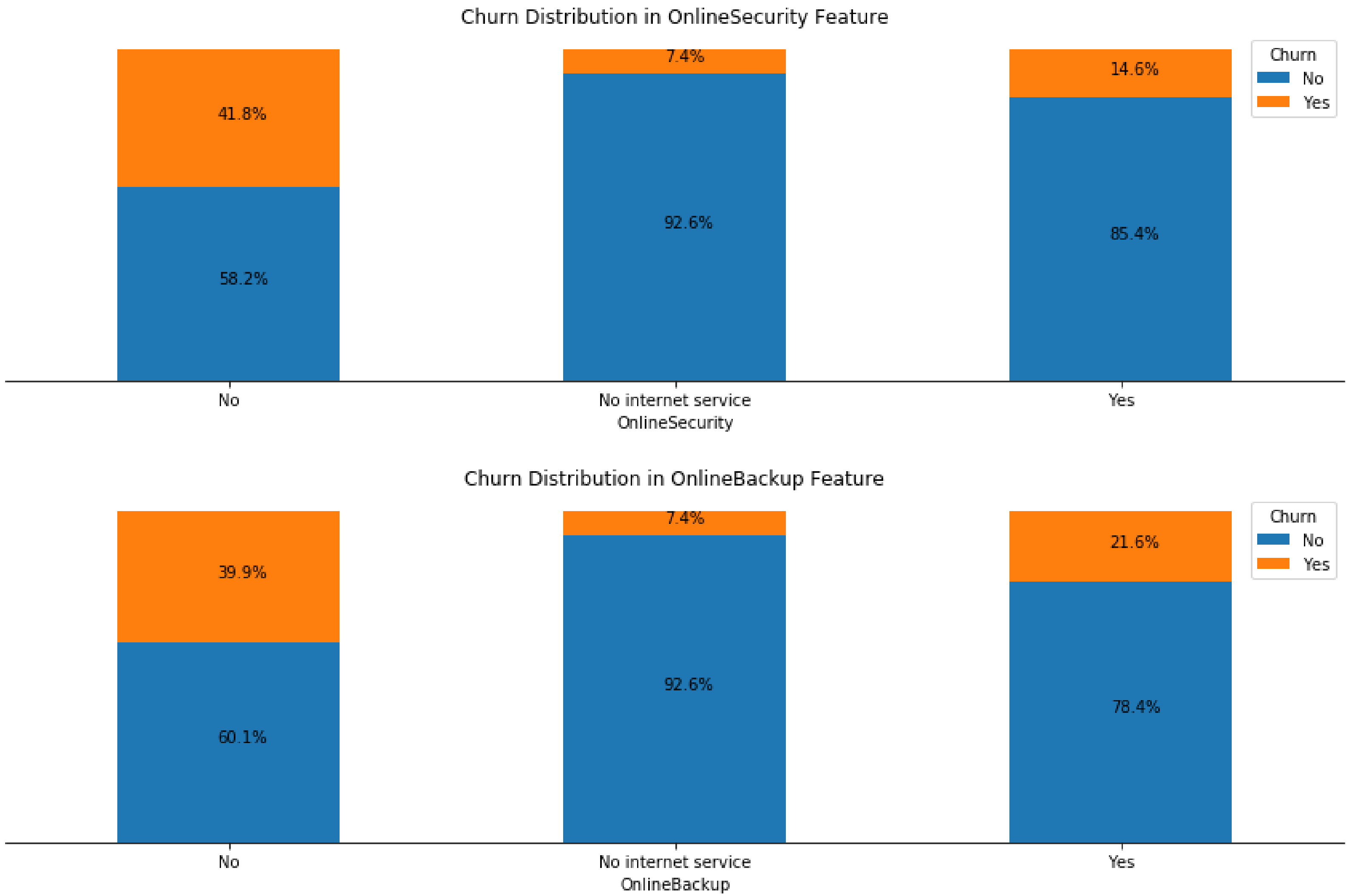

Churn Distribution in Dependents Feature
Churn
No
Yes
31.3%
68.7%
15.5%
84.5%
No
Yes
Dependents

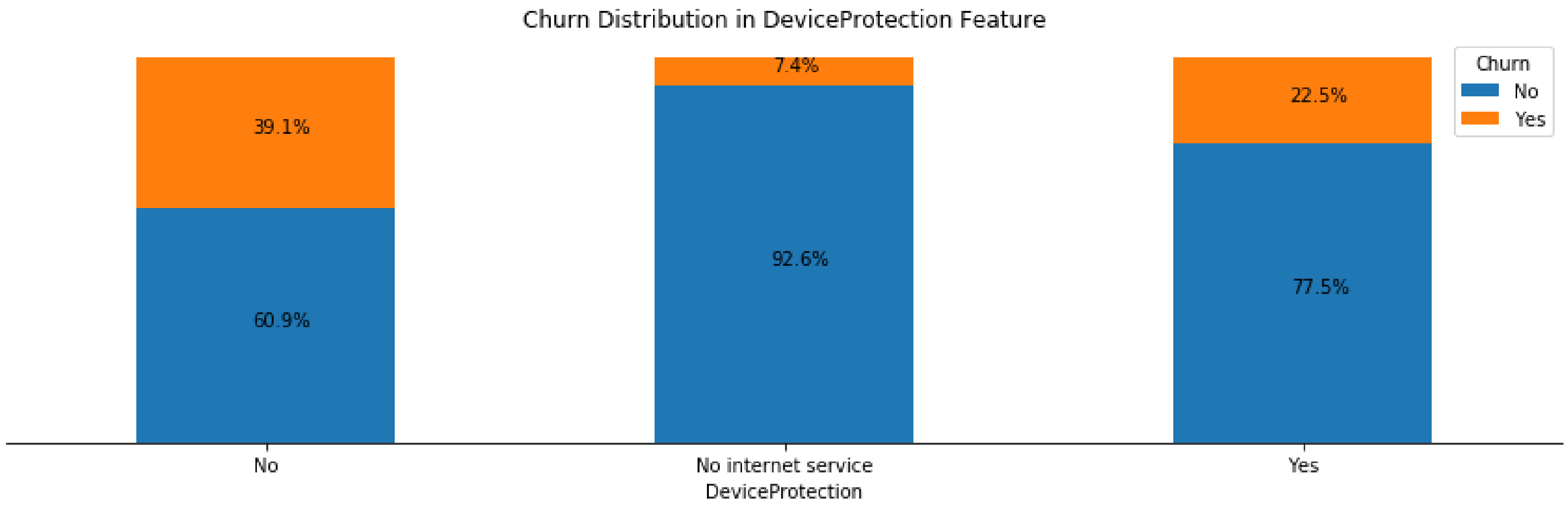

Churn Distribution in tenure Feature
47.7%
28.7%
21.6%
19.0%
14.4%
6.6%
52.3%
71.3%
78.4%
81.0%
85.6%
93.4%
Churn
No
Yes
tenure_1_year
tenure_2_year
tenure_3_year
tenure_4_year
tenure_5_year
tenure_5_year_more
tenure


Churn Distribution in PhoneService Feature
25.0%
26.7%
75.0%
73.3%
Churn
No
Yes
No
Yes
PhoneService

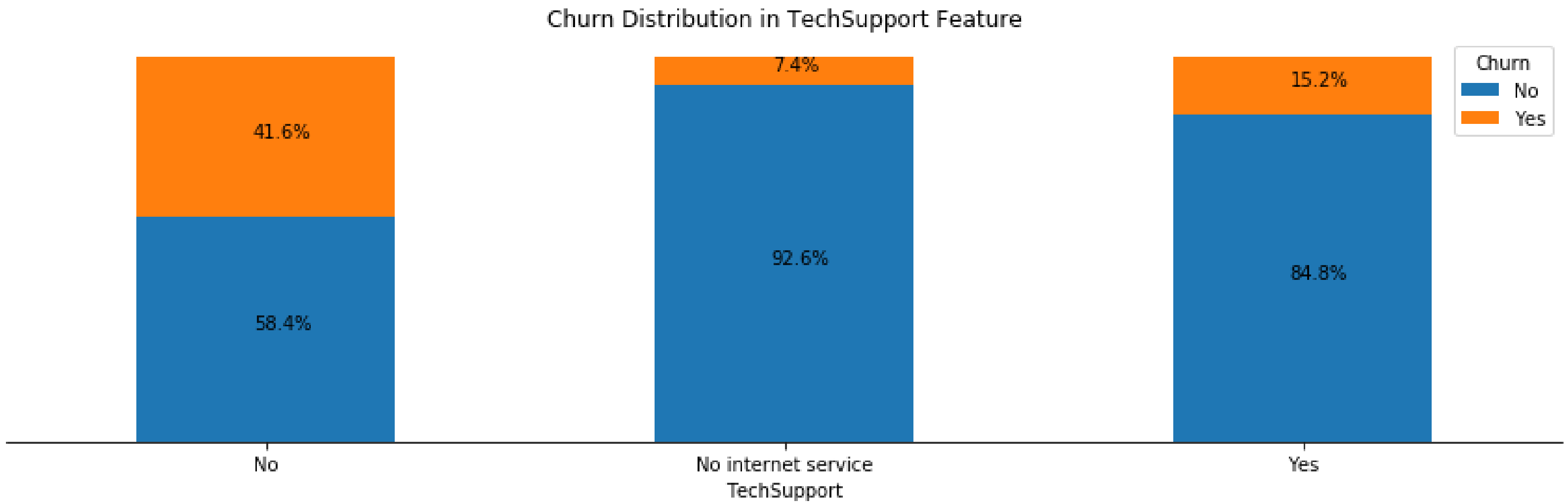

Churn Distribution in MultipleLines Feature
Churn
No
Yes
25.1%
74.9%
25.0%
75.0%
28.6%
71.4%
No
No phone service
Yes
MultipleLines


Churn Distribution in InternetService Feature
Churn
No
Yes
19.0%
81.0%
41.9%
58.1%
7.4%
92.6%
DSL
Fiber optic
No
InternetService

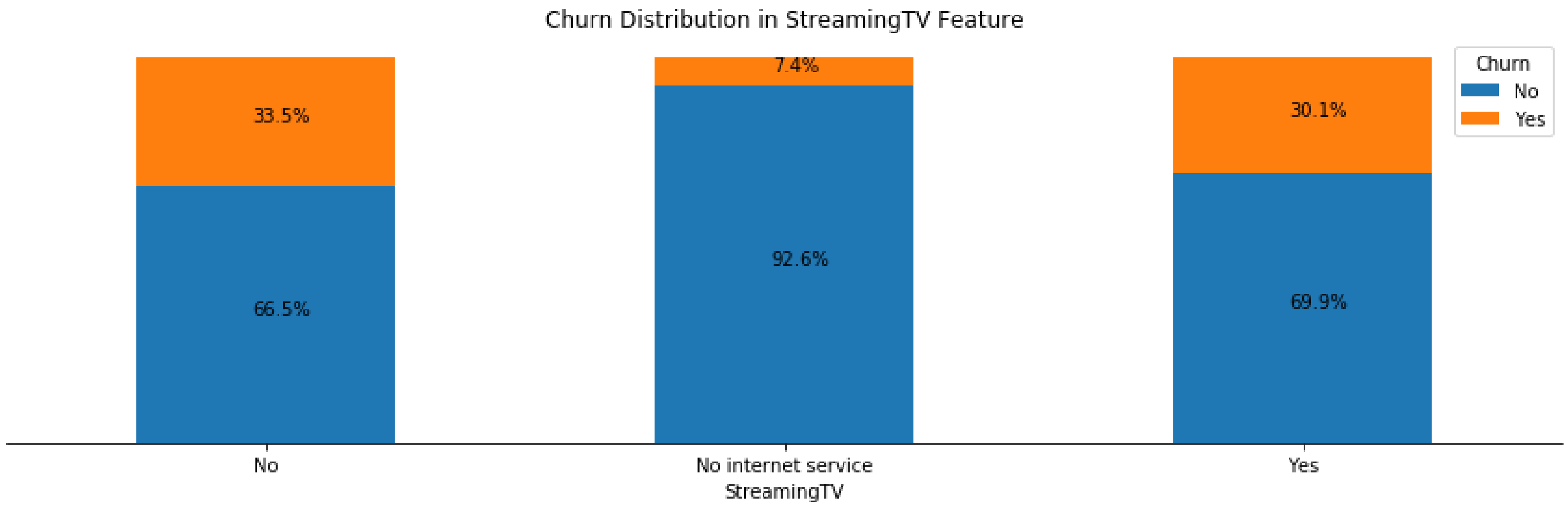

Churn Distribution in OnlineSecurity Feature
Churn
No
Yes
41.8%
58.2%
7.4%
92.6%
14.6%
85.4%
No
No internet service
Yes
OnlineSecurity
Churn Distribution in OnlineBackup Feature
Churn
No
Yes
39.9%
60.1%
7.4%
92.6%
21.6%
78.4%
No
No internet service
Yes
OnlineBackup

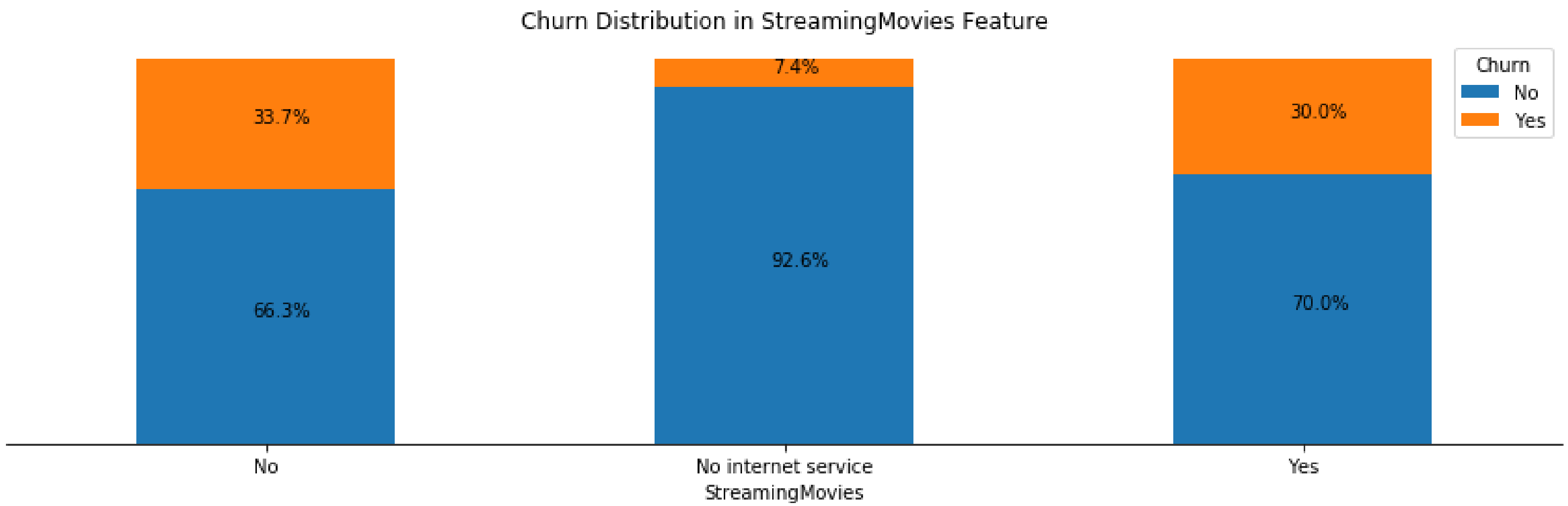


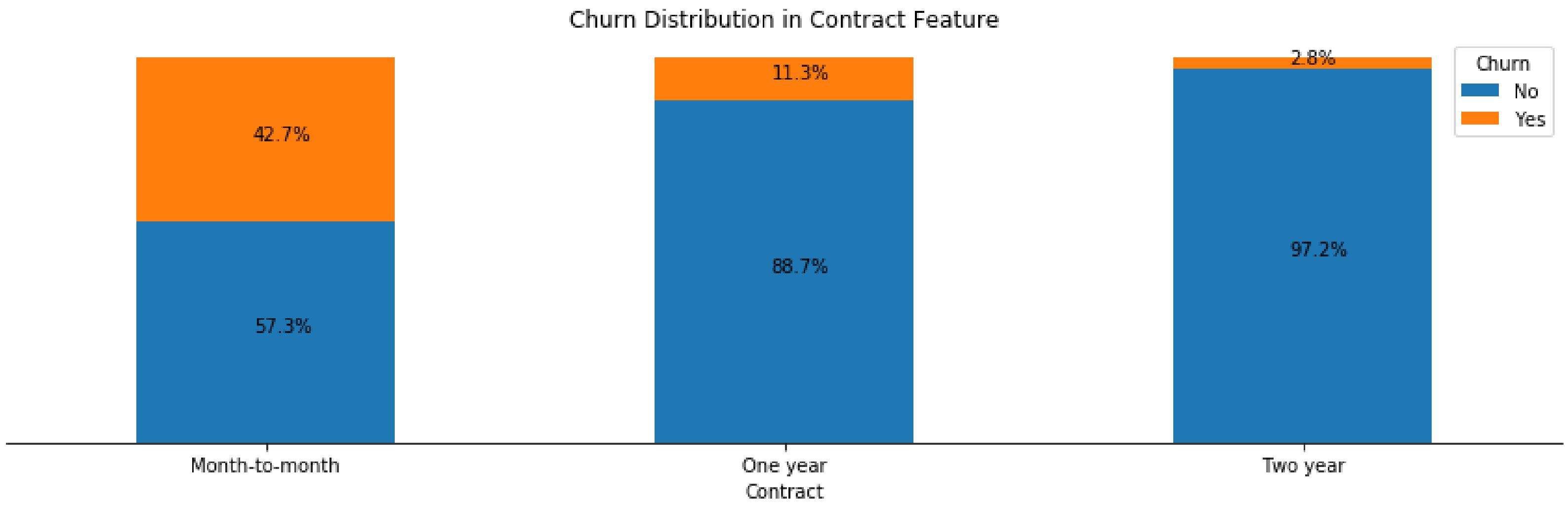
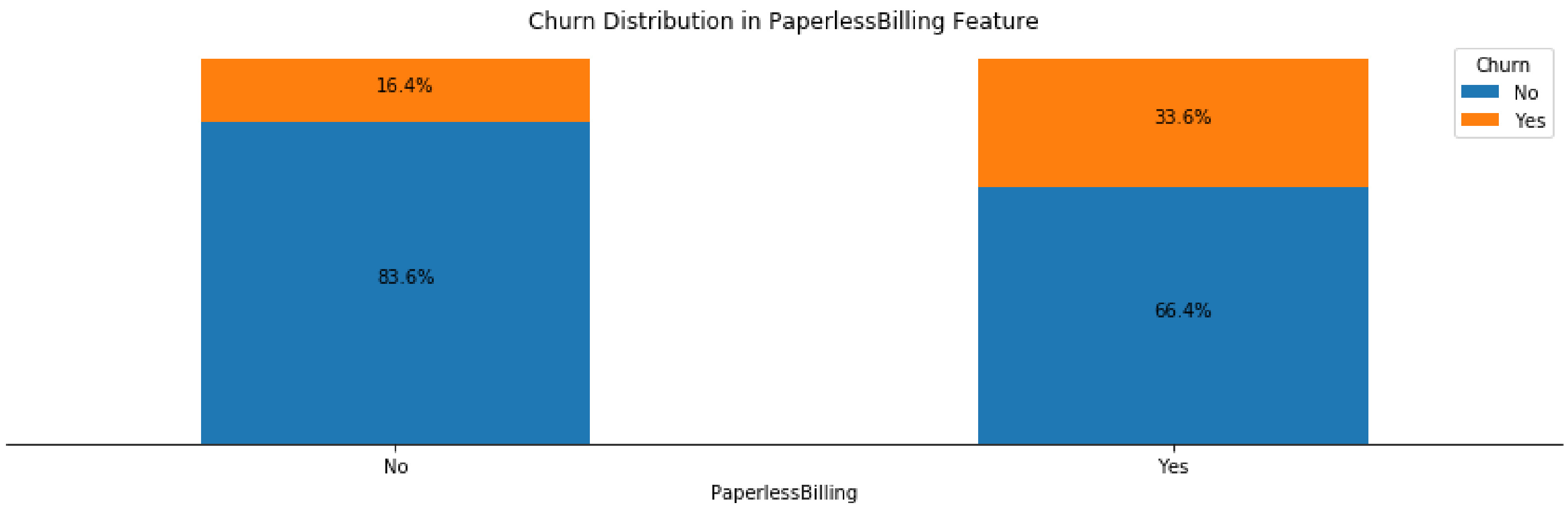
Churn Distribution in DeviceProtection Feature
39.1%
60.9%
7.4%
92.6%
22.5%
77.5%
Churn
No
Yes
No
No internet service
Yes
DeviceProtection

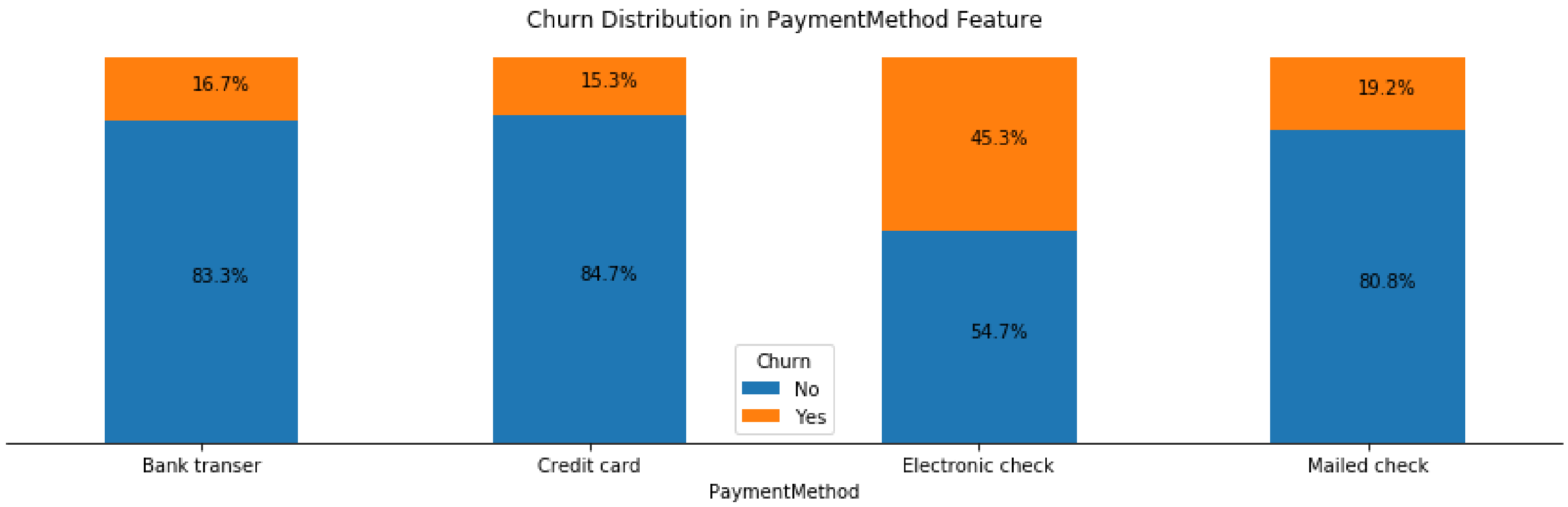


Churn Distribution in TechSupport Feature
41.6%
58.4%
7.4%
92.6%
15.2%
84.8%
Churn
No
Yes
No
No internet service
Yes
TechSupport


Churn Distribution in StreamingTV Feature
33.5%
66.5%
7.4%
92.6%
30.1%
69.9%
Churn
No
Yes
No
No internet service
Yes
StreamingTV

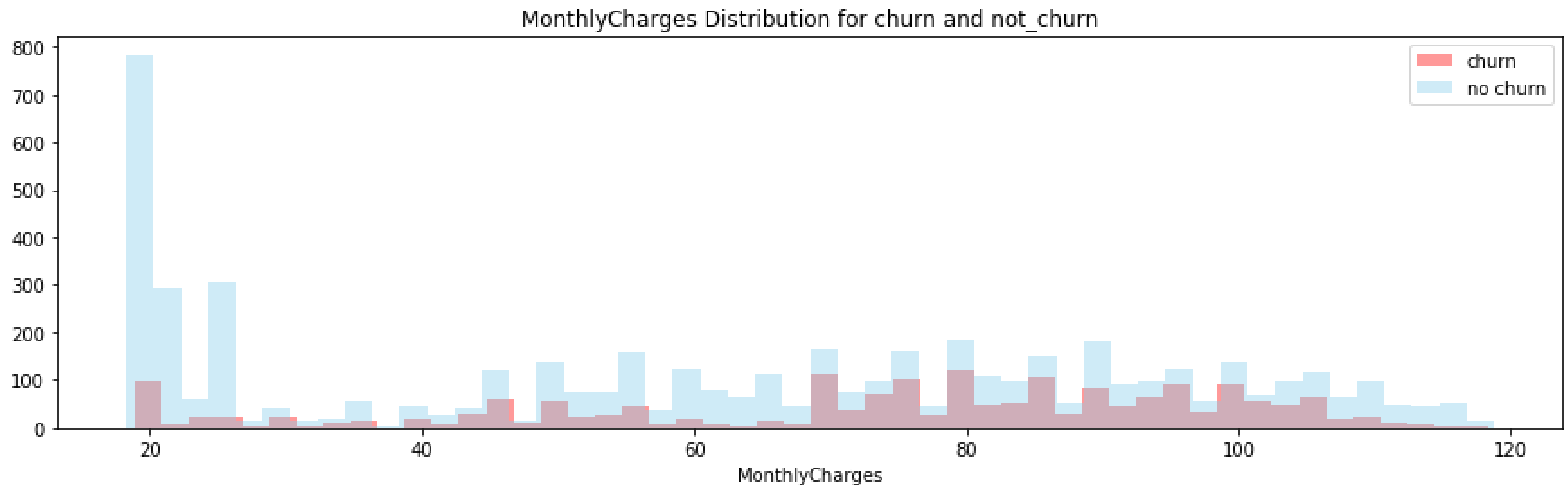

Churn Distribution in StreamingMovies Feature
33.7%
66.3%
7.4%
92.6%
30.0%
70.0%
Churn
No
Yes
No
No internet service
Yes
StreamingMovies

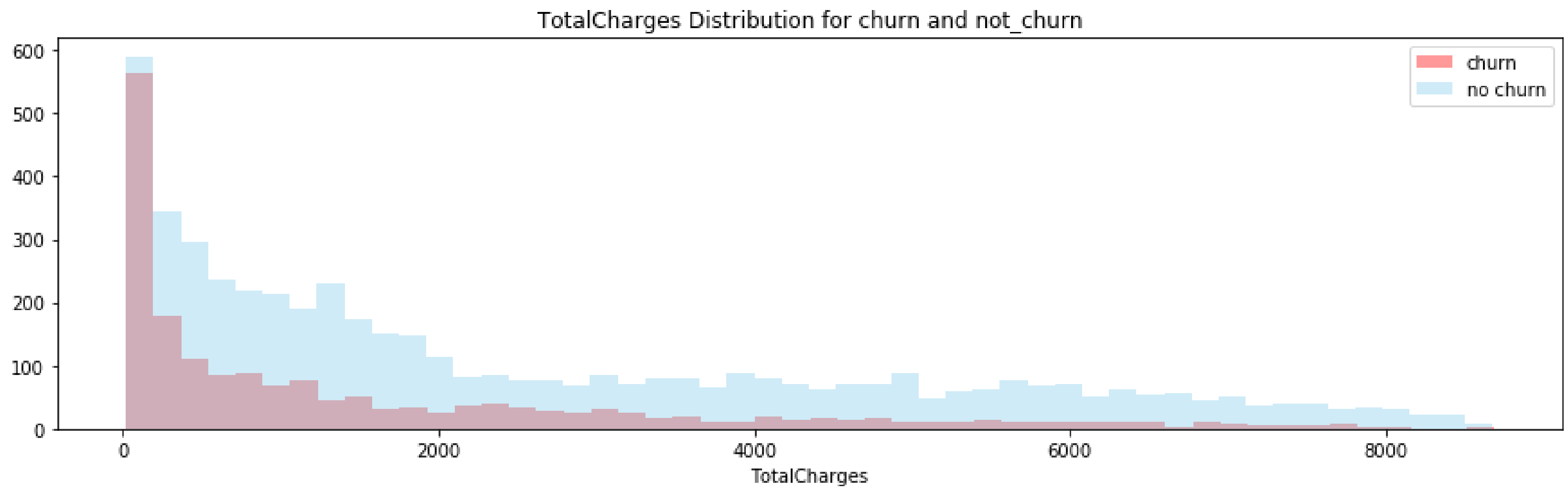

Churn Distribution in Contract Feature
42.7%
57.3%
11.3%
88.7%
2.8%
97.2%
Churn
No
Yes
Month-to-month
One year
Two year
Contract

Churn Distribution in PaperlessBilling Feature
16.4%
83.6%
33.6%
66.4%
Churn
No
Yes
No
Yes
PaperlessBilling


Churn Distribution in PaymentMethod Feature
16.7%
83.3%
15.3%
84.7%
45.3%
54.7%
19.2%
80.8%
Churn
No
Yes
Bank transer
Credit card
Electronic check
Mailed check
PaymentMethod

In [19]:

```python
# get the numerical features
num_cols_vis = [x for x in df_vis.columns if x not in (cat_cols_vis + to_drop_vis)]

# Separate churn and no_churn data
df_vis_churn = df_vis[df_vis['Churn'] == 'Yes']
df_vis_not_churn = df_vis[df_vis['Churn'] == 'No']

for col in num_cols_vis:
    plt.figure(figsize = (15, 4))
    plt.title(col + " Distribution for churn and not_churn")
    sns.distplot(df_vis_churn[col], bins = 50, color = 'red', kde = False, label = 'churn')
    sns.distplot(df_vis_not_churn[col], bins = 50, color = 'skyblue', kde = False, label = 'no churn')
    plt.legend()
```





1.5 Feature Preprocessing

In [20]:
```
# Drop some useless columns and the target column

to_drop = ['customerID', 'Churn']
churn_feat_space = churn_df.drop(labels = to_drop, axis = 1)
```

In [21]:
```python
# replace "No internet service" or "No phone service" by "No"

churn_feat_space = churn_feat_space.replace({"No phone service" : "No", "No internet service" : "No"})
```

In [22]:
```python
# convert the "TotalCharges" column to float type

churn_feat_space['TotalCharges'] = churn_feat_space['TotalCharges'].astype(float)
```

In [23]:
```python
# get the column list with categorical values
cat_cols = churn_feat_space.nunique()[churn_feat_space.nunique() <= 4].keys().tolist()

# get the column list with numerical values
num_cols = [x for x in churn_feat_space.columns if x not in cat_cols]

# get the column list with two categories
binary_cols = churn_feat_space.nunique()[churn_feat_space.nunique() == 2].keys().tolist()

# get the column list with three or more categories
multi_cols = [x for x in cat_cols if x not in binary_cols]
```

In [24]:
```python
# preprocess the columns with numerical values, scale the data
scaler = StandardScaler()
churn_scaled = scaler.fit_transform(X = churn_feat_space[num_cols]) # return numpy array, need to convert to dataframe
churn_scaled = pd.DataFrame(data = churn_scaled, columns = num_cols)
churn_feat_space = churn_feat_space[cat_cols].merge(churn_scaled, how = 'left', left_index = True, right_index = True)

# preprocess the columns with two categories
encoder = LabelEncoder()
for col in binary_cols:
    churn_feat_space[col] = encoder.fit_transform(churn_feat_space[col])

# preprocess the columns with three or more categories
churn_feat_space = pd.get_dummies(data = churn_feat_space, columns = multi_cols)

# contain the target column
churn_feat_space_and_target = churn_df[['Churn']].merge(churn_feat_space, how = 'left', left_index = True, right_index = True)
```

In [25]:
```
churn_feat_space_and_target.head()
```

Out[25]:

	Churn	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupp
0	No	0	0	1	0	0	0	0	1	0	0
1	No	1	0	0	0	1	0	1	0	1	0
2	Yes	1	0	0	0	1	0	1	1	0	0
3	No	1	0	0	0	0	0	1	0	1	1
4	Yes	0	0	0	0	1	0	0	0	0	0

In [26]:
```
churn_feat_space.dtypes
```

Out[26]:
```
gender                                     int32
SeniorCitizen                              int64
Partner                                    int32
Dependents                                 int32
PhoneService                               int32
MultipleLines                              int32
OnlineSecurity                             int32
OnlineBackup                               int32
DeviceProtection                           int32
TechSupport                                int32
StreamingTV                                int32
StreamingMovies                            int32
PaperlessBilling                           int32
tenure                                   float64
MonthlyCharges                           float64
TotalCharges                             float64
InternetService_DSL                        uint8
InternetService_Fiber optic                uint8
InternetService_No                         uint8
Contract_Month-to-month                    uint8
Contract_One year                          uint8
Contract_Two year                          uint8
PaymentMethod_Bank transfer (automatic)    uint8
PaymentMethod_Credit card (automatic)      uint8
PaymentMethod_Electronic check             uint8
PaymentMethod_Mailed check                 uint8
dtype: object
```

1.6 Feature Correlation

In [27]:
```
# Calculate the correlation matrix

corr = churn_feat_space.corr()
corr
```

Out[27]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	Devi
gender	1.000000	-0.001819	-0.001379	0.010349	-0.007515	-0.008883	-0.016328	-0.013093	-0.00
SeniorCitizen	-0.001819	1.000000	0.016957	-0.210550	0.008392	0.142996	-0.038576	0.066663	0.059
Partner	-0.001379	0.016957	1.000000	0.452269	0.018397	0.142561	0.143346	0.141849	0.153
Dependents	0.010349	-0.210550	0.452269	1.000000	-0.001078	-0.024307	0.080786	0.023639	0.013
PhoneService	-0.007515	0.008392	0.018397	-0.001078	1.000000	0.279530	-0.091676	-0.052133	-0.07
MultipleLines	-0.008883	0.142996	0.142561	-0.024307	0.279530	1.000000	0.098592	0.202228	0.201
OnlineSecurity	-0.016328	-0.038576	0.143346	0.080786	-0.091676	0.098592	1.000000	0.283285	0.274
OnlineBackup	-0.013093	0.066663	0.141849	0.023639	-0.052133	0.202228	0.283285	1.000000	0.303
DeviceProtection	-0.000807	0.059514	0.153556	0.013900	-0.070076	0.201733	0.274875	0.303058	1.000
TechSupport	-0.008507	-0.060577	0.120206	0.063053	-0.095138	0.100421	0.354458	0.293705	0.332
StreamingTV	-0.007124	0.105445	0.124483	-0.016499	-0.021383	0.257804	0.175514	0.281601	0.389
StreamingMovies	-0.010105	0.119842	0.118108	-0.038375	-0.033477	0.259194	0.187426	0.274523	0.402
PaperlessBilling	-0.011902	0.156258	-0.013957	-0.110131	0.016696	0.163746	-0.004051	0.127056	0.104
tenure	0.005285	0.015683	0.381912	0.163386	0.007877	0.332399	0.328297	0.361138	0.361
MonthlyCharges	-0.013779	0.219874	0.097825	-0.112343	0.248033	0.490912	0.296447	0.441529	0.482
TotalCharges	0.000048	0.102411	0.319072	0.064653	0.113008	0.469042	0.412619	0.510100	0.522
InternetService_DSL	0.007584	-0.108276	-0.001043	0.051593	-0.452255	-0.200318	0.320343	0.156765	0.145
InternetService_Fiber optic	-0.011189	0.254923	0.001235	-0.164101	0.290183	0.366420	-0.030506	0.165940	0.176
InternetService_No	0.004745	-0.182519	-0.000286	0.138383	0.171817	-0.210794	-0.332799	-0.380990	-0.38
Contract_Month-to-month	-0.003251	0.137752	-0.280202	-0.229715	-0.001243	-0.088558	-0.246844	-0.164393	-0.22
Contract_One year	0.007755	-0.046491	0.083067	0.069222	-0.003142	-0.003594	0.100658	0.084113	0.102
Contract_Two year	-0.003603	-0.116205	0.247334	0.201699	0.004442	0.106618	0.191698	0.111391	0.165
PaymentMethod_Bank transfer (automatic)	-0.015973	-0.016235	0.111406	0.052369	0.008271	0.075429	0.094366	0.086942	0.083

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	Devi
PaymentMethod_Credit card (automatic)	0.001632	-0.024359	0.082327	0.061134	-0.006916	0.060319	0.115473	0.090455	0.111
PaymentMethod_Electronic check	0.000844	0.171322	-0.083207	-0.149274	0.002747	0.083583	-0.112295	-0.000364	-0.00
PaymentMethod_Mailed check	0.013199	-0.152987	-0.096948	0.056448	-0.004463	-0.227672	-0.079918	-0.174075	-0.18

In [28]:

```
plt.figure(figsize = (12, 8))
plt.title('Pearson Correlation')
g = sns.heatmap(corr, xticklabels = True, yticklabels = True)
```

Pearson Correlation
gender
SeniorCitizen
Partner
Dependents
PhoneService
MultipleLines
OnlineSecurity
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
PaperlessBilling
tenure
MonthlyCharges
TotalCharges
InternetService_DSL
InternetService_Fiber optic
InternetService_No
Contract_Month-to-month
Contract_One year
Contract_Two year
PaymentMethod_Bank transfer (automatic)
PaymentMethod_Credit card (automatic)
PaymentMethod_Electronic check
PaymentMethod_Mailed check
0.9
0.6
0.3
0.0
−0.3
−0.6

In [29]:
```python
# Alternative way to get the pearson correlation between two variables

pearsonr(churn_feat_space["TotalCharges"], churn_feat_space["MonthlyCharges"])[0]
```

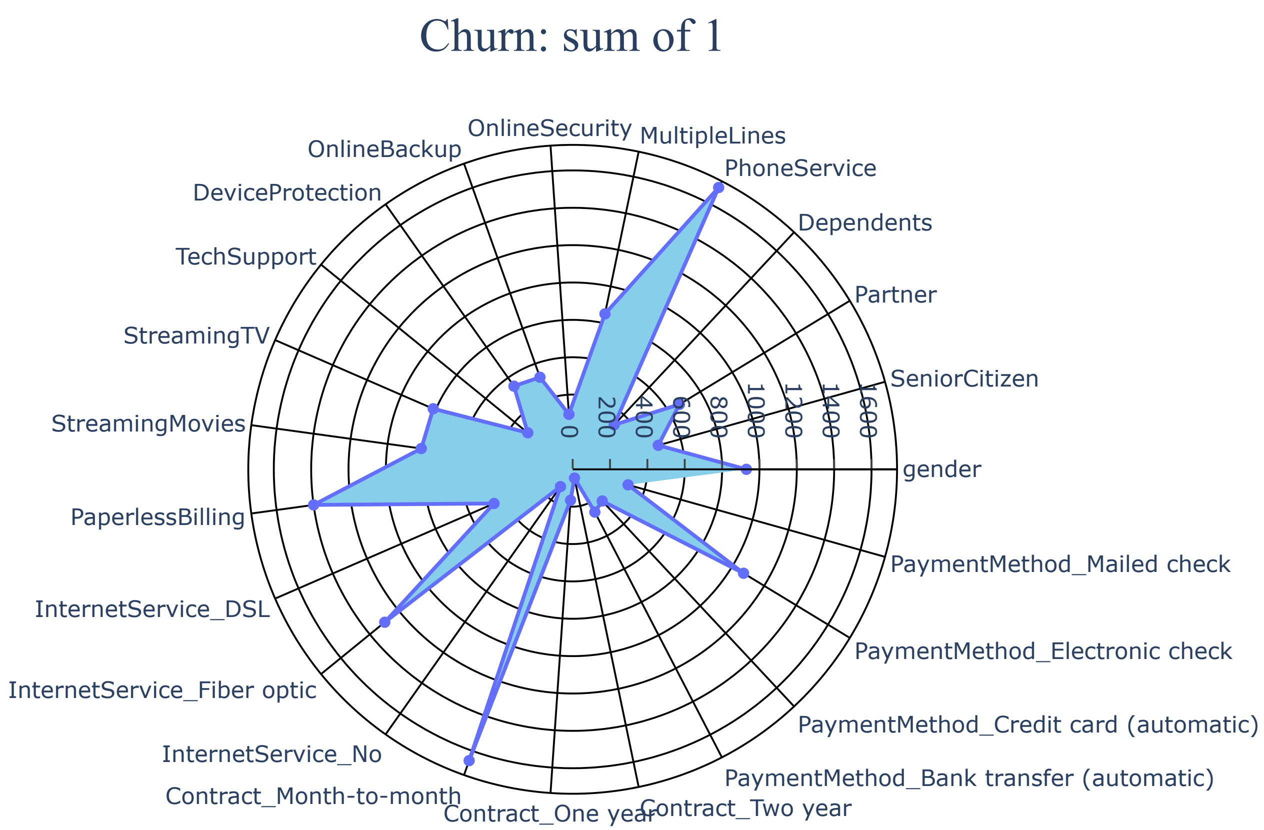
Out[29]: 0.6510648032262025

1.7 Radar Chart for Binary Features

In [30]:
```python
# remove features which are not binary
binary_cols_radar = churn_feat_space.nunique()[churn_feat_space.nunique() == 2].keys().tolist()
to_drop_radar = [x for x in churn_feat_space.columns if x not in binary_cols_radar]
churn_bifeat_and_target_radar = churn_feat_space_and_target.drop(labels = to_drop_radar, axis = 1)

# separate churn and not_churn
bifeat_churn_radar = churn_bifeat_and_target_radar[churn_bifeat_and_target_radar['Churn'] == 'Yes'][binary_cols_radar]
bifeat_no_churn_radar = churn_bifeat_and_target_radar[churn_bifeat_and_target_radar['Churn'] == 'No'][binary_cols_radar]

# count the number of 1 in each column, for churn and not_churn
bifeat_churn_radar = bifeat_churn_radar.sum()
bifeat_no_churn_radar = bifeat_no_churn_radar.sum()
```

In [31]:
```python
def radar_chart(categories, values, title):
    data = go.Scatterpolar(r = values, theta = categories, mode = 'lines+markers', fill = 'toself', fillcolor = 'skyblue')
    layout = go.Layout(dict(polar = dict(radialaxis = dict(side = 'counterclockwise', linecolor = 'black',
                                                           ticks = 'outside', gridcolor = 'black'),
                                         angularaxis = dict(gridcolor = 'black', linecolor = 'black'),
                                         bgcolor = "white"),
                            title = dict(text = title, font = dict(family = 'Times New Roman', size = 25), x = 0.5),
                            )
                       )
    fig = go.Figure(data = [data], layout = layout)
    pyo.iplot(fig)
```

In [32]:
```
categories = bifeat_churn_radar.keys().tolist()
values = bifeat_churn_radar.values.tolist()
radar_chart(categories, values, 'Churn: sum of 1')
```



In [33]:

```
categories = bifeat_no_churn_radar.keys().tolist()
values = bifeat_no_churn_radar.values.tolist()
radar_chart(categories, values, 'No_Churn: sum of 1')
```