Telco Customer Churn Project

In [1]:
```
#Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import classification_report

import warnings
warnings.filterwarnings('ignore')
```

Data Preprocessing

In [2]:
```
# Define the file path
file_path = r'D:\Data Science data\Teclo\WA_Fn-UseC_-Telco-Customer-Churn.csv'

# Read the CSV data
df = pd.read_csv(file_path)

# Display the first few rows of the data
df.head()
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multipl
0	7590-VHVEG	Female	0	Yes	No	1	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	No
4	9237-HQITU	Female	0	No	No	2	Yes	

5 rows × 21 columns

In [3]:

```
#Check the shape of the data
df.shape
```

Out[3]:

```
(7043, 21)
```

In [4]:

```
# Display the general info about dataframe and check data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [5]:
```python
# Convert 'TotalCharges' column to numeric
df['TotalCharges'] = pd.to_numeric(df.TotalCharges, errors='coerce')
# Show the sum of null values in each column
df.isnull().sum()
```

Out[5]:
```
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 [6]:
```python
# Show rows where 'TotalCharges' is NaN
df[np.isnan(df['TotalCharges'])]
```

Out[6]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Mul
488	4472-LVYGI	Female	0	Yes	Yes	0	No	
753	3115-CZMZD	Male	0	No	Yes	0	Yes	
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	

11 rows × 21 columns

In [7]:
```python
# Drop rows with NaN values
df.dropna(inplace = True)
```

In [8]:
```python
# Check again for null values in the dataset
df.isnull().sum()
```

Out[8]:

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

In [9]:

```
# Convert 'SeniorCitizen' column to 'Yes'/'No' from 1/0
df["SeniorCitizen"]= df["SeniorCitizen"].map({0: "No", 1: "Yes"})
df.head()
```

Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multipl
0	7590-VHVEG	Female	No	Yes	No	1	No	No
1	5575-GNVDE	Male	No	No	No	34	Yes	
2	3668-QPYBK	Male	No	No	No	2	Yes	
3	7795-CFOCW	Male	No	No	No	45	No	No
4	9237-HQITU	Female	No	No	No	2	Yes	

5 rows × 21 columns

In [10]:

```
#Remove customer IDs from the data set
df2 = df.iloc[:,1:]

#Converte the predictor variable in a binary numeric variable
df2['Churn'].replace(to_replace='Yes', value=1, inplace=True)
df2['Churn'].replace(to_replace='No',  value=0, inplace=True)
```

```python
#Convert all the categorical variables into dummy variables
df_dummies = pd.get_dummies(df2)
df_dummies.head()
```

Out[10]:

	tenure	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	SeniorCitiz
0	1	29.85	29.85	0	1	0	
1	34	56.95	1889.50	0	0	1	
2	2	53.85	108.15	1	0	1	
3	45	42.30	1840.75	0	0	1	
4	2	70.70	151.65	1	1	0	

5 rows × 47 columns

In [11]:
```python
# Check statistical information of numerical columns
numerical_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
df[numerical_cols].describe()
```

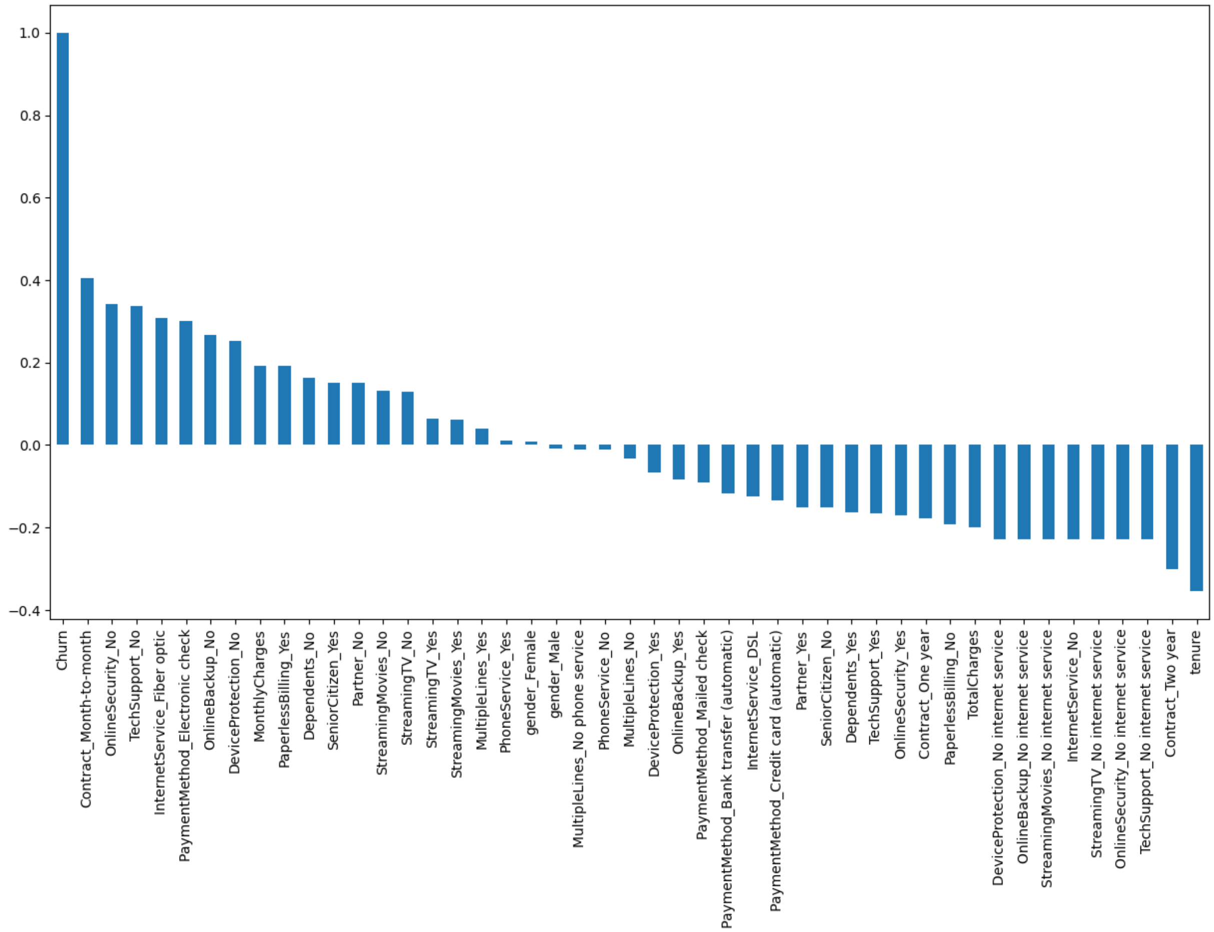
Out[11]:

	tenure	MonthlyCharges	TotalCharges
count	7032.000000	7032.000000	7032.000000
mean	32.421786	64.798208	2283.300441
std	24.545260	30.085974	2266.771362
min	1.000000	18.250000	18.800000
25%	9.000000	35.587500	401.450000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.862500	3794.737500
max	72.000000	118.750000	8684.800000

Data Visualization

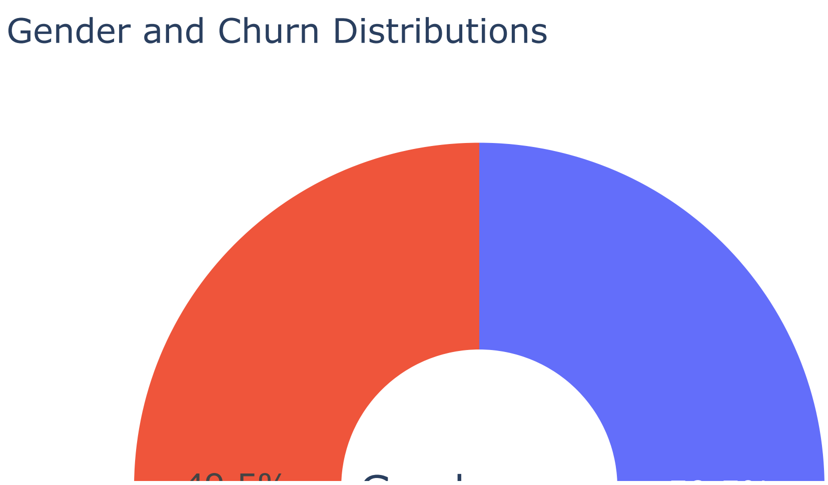
In [12]:
```python
#Get Correlation of "Churn" with other variables:
plt.figure(figsize=(15,8))
df_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

Out[12]:
```
<Axes: >
```



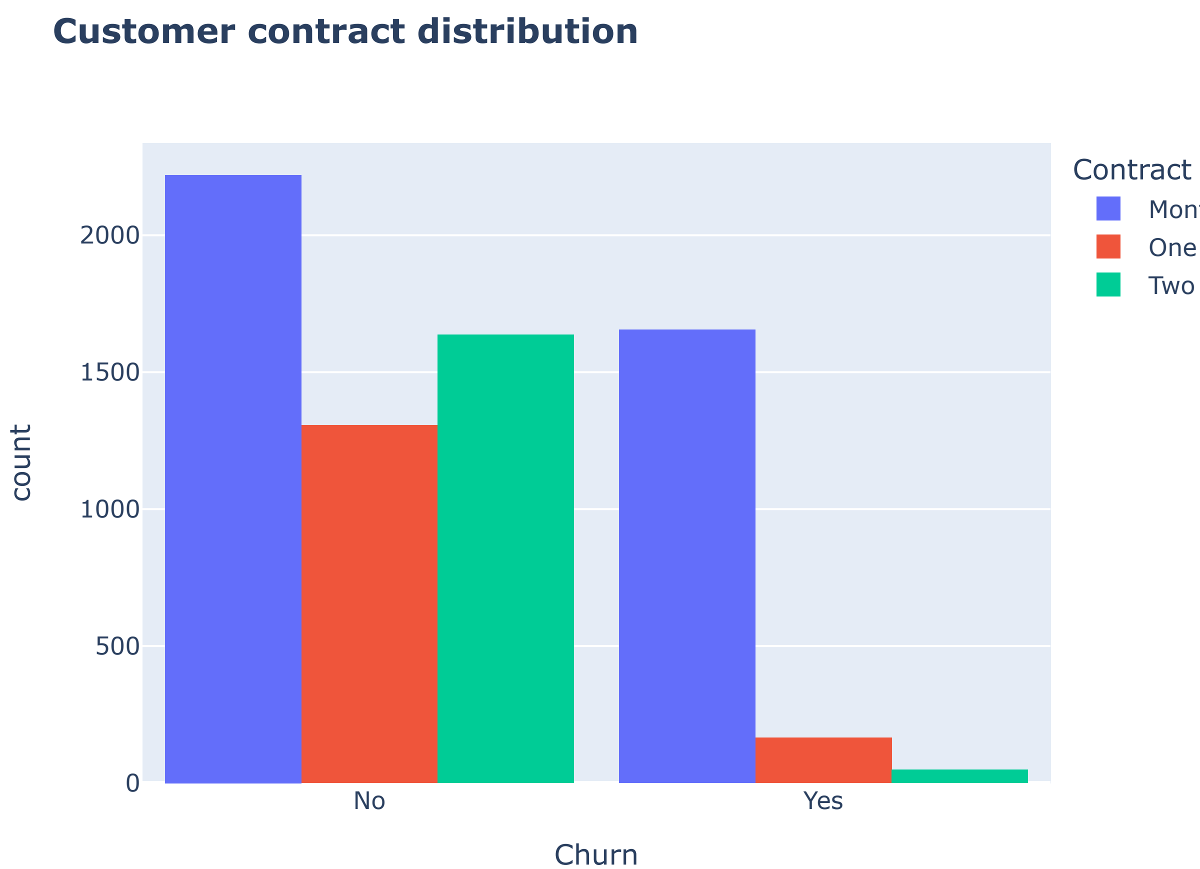
In [13]:

```
# Pie charts for Gender and Churn distribution
g_labels = ['Male', 'Female']
c_labels = ['No', 'Yes']

fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])
fig.add_trace(go.Pie(labels=g_labels, values=df['gender'].value_counts(), name="Gen
              1, 1)
fig.add_trace(go.Pie(labels=c_labels, values=df['Churn'].value_counts(), name="Chur
              1, 2)
fig.update_traces(hole=.4, hoverinfo="label+percent+name", textfont_size=16)
fig.update_layout(

    title_text="Gender and Churn Distributions",
    annotations=[dict(text='Gender', x=0.16, y=0.5, font_size=20, showarrow=False),
                 dict(text='Churn', x=0.84, y=0.5, font_size=20, showarrow=False)])
fig.show()
```



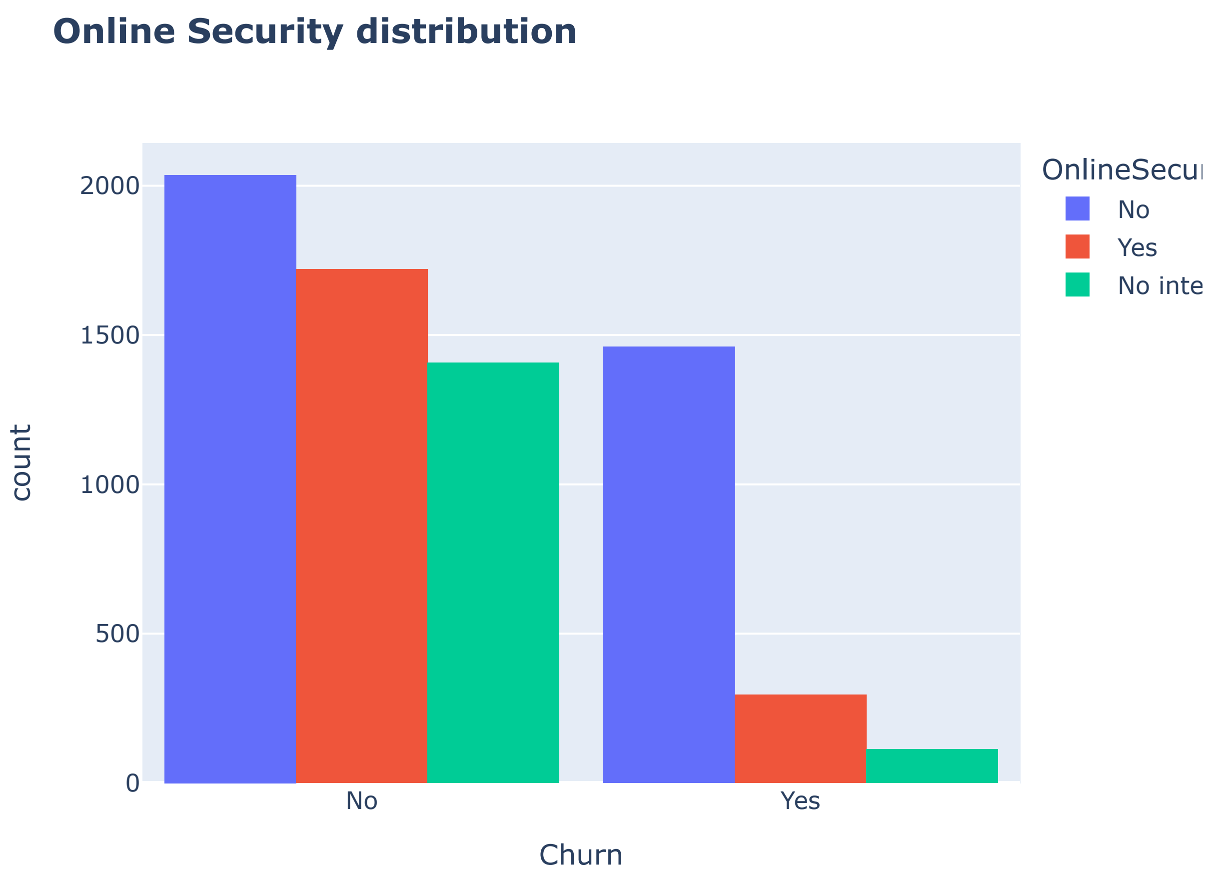
The difference in the percentage or number of customers who switched service providers is insignificant. Both males and females exhibited similar tendencies in migrating to a different service provider or company.

In [14]:
```
# Contract feature histogram
fig = px.histogram(df, x="Churn", color="Contract", barmode="group", title="<b>Cust
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



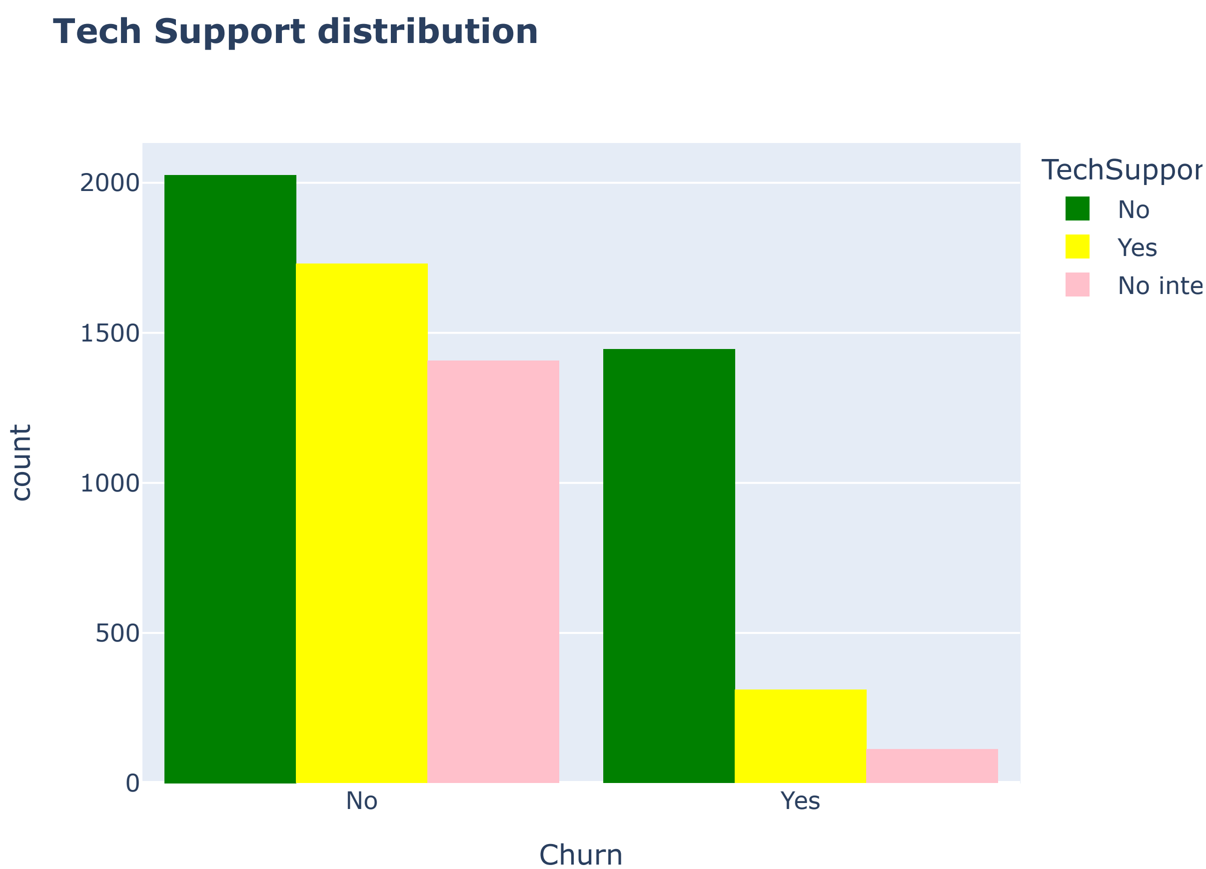
About 75% of customer with Month-to-Month Contract opted to move out as compared to 13% of customrs with One Year Contract and 3% with Two Year Contract

In [15]:
```
# OnlineSecurity feature histogram
fig = px.histogram(df, x="Churn", color="OnlineSecurity", barmode="group", title="<
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



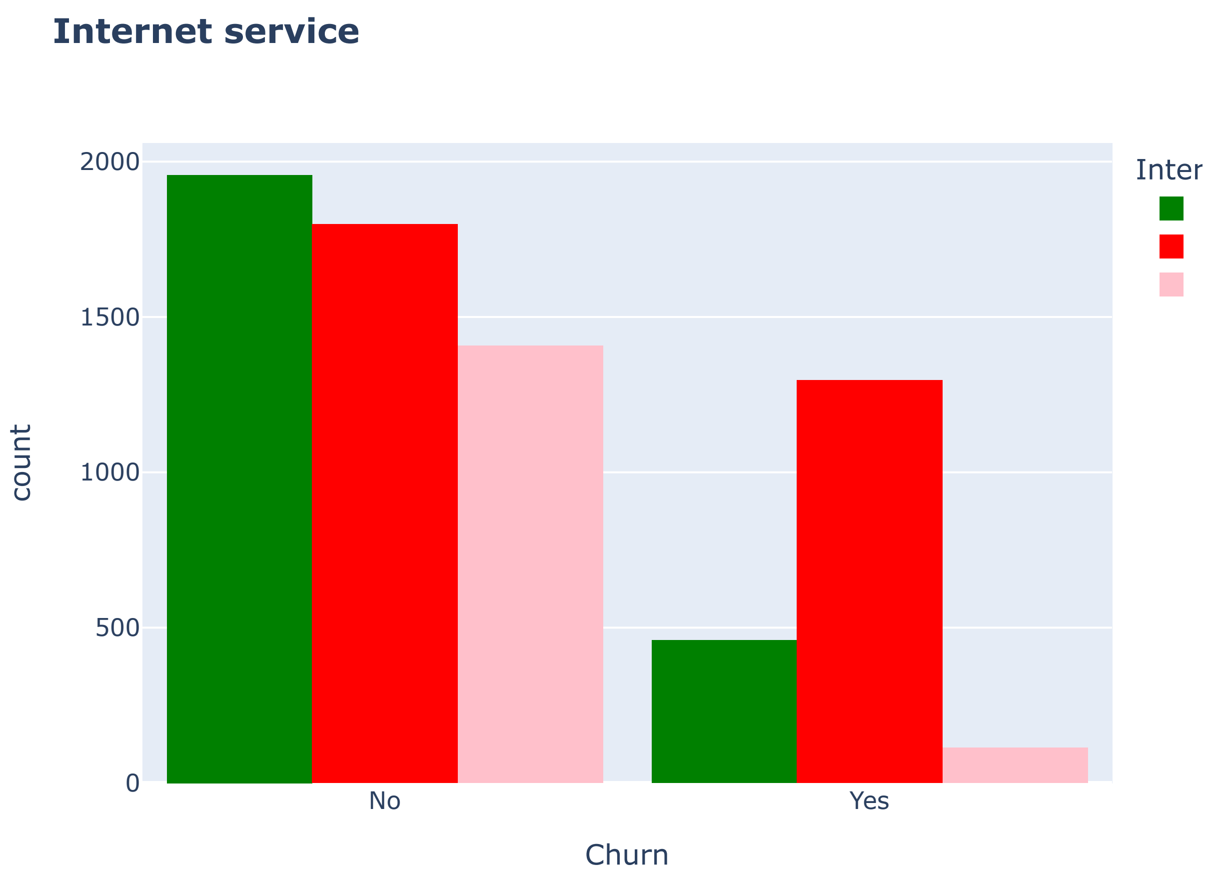
Most customers churn in the absence of online security

```
In [16]: # TechSupport feature histogram
fig = px.histogram(df, x="Churn", color="TechSupport", barmode="group",
                   color_discrete_sequence=['green', 'yellow', 'pink'], # define yo
                   title="<b>Tech Support distribution<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



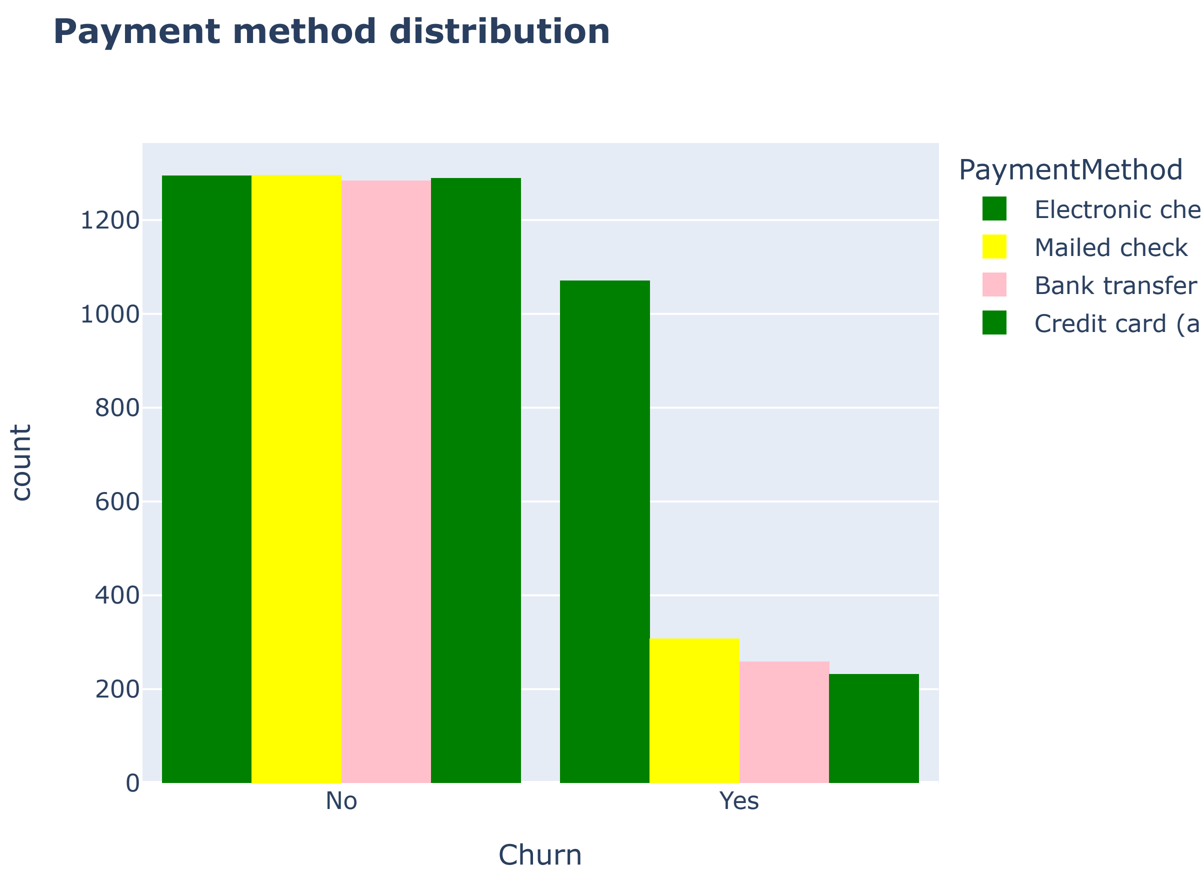
Customers with no tech support were most likely to churn comparing to other categories

In [17]:
```
# InternetService feature histogram
fig = px.histogram(df, x="Churn", color="InternetService", barmode="group",
                   color_discrete_sequence=['green', 'red', 'pink'], # define your
                   title="<b>Internet service<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



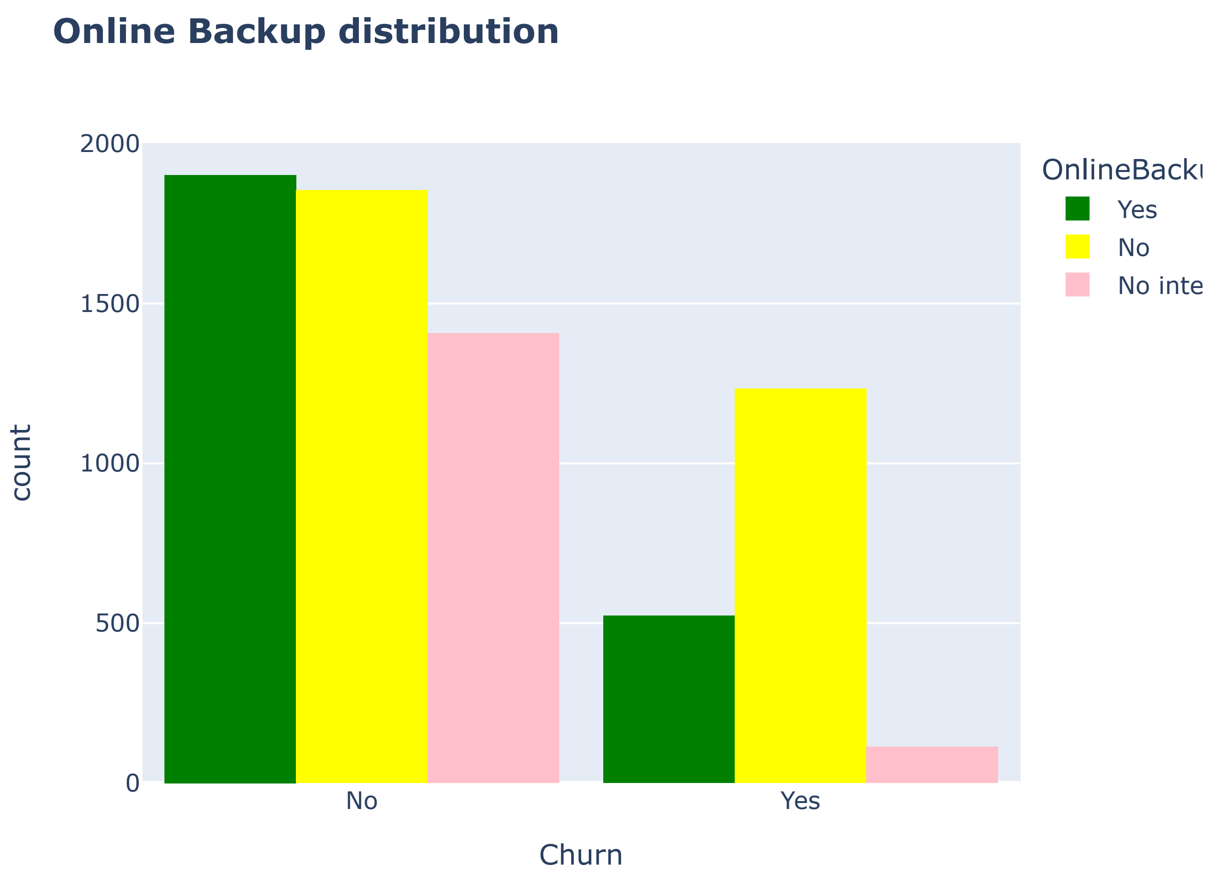
A lot of customers choose the Fiber optic service and it's also evident that the customers who use Fiber optic have high churn rate, this might suggest a dissatisfaction with this type of internet service. Customers having DSL service are majority in number and have less churn rate compared to Fibre optic service.

In [18]:
```
# PaymentMethod feature histogram
fig = px.histogram(df, x="Churn", color="PaymentMethod", barmode="group",
                   color_discrete_sequence=['green', 'yellow', 'pink'], # define yo
                   title="<b>Payment method distribution<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



Major customers who moved out were having Electronic Check as Payment Method. Customers who opted for Credit-Card automatic transfer or Bank Automatic Transfer and Mailed Check as Payment Method were less likely to move out.

In [19]:
```
# OnlineBackup feature histogram
fig = px.histogram(df, x="Churn", color="OnlineBackup", barmode="group",
                   color_discrete_sequence=['green', 'yellow', 'pink'], # define yo
                   title="<b>Online Backup distribution<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```



Customers with no Online Backup were most likely to churn comparing to other categories

Data Manipulation

In [20]:

```
# Convert categorical data to numeric

le = LabelEncoder()

# Make a copy of the dataframe df_dummies into df3
df3 = df_dummies.copy()

# Loop over the columns and transform categorical columns
for col in df3.columns:
    if df3[col].dtype=='object':
        df3[col] = le.fit_transform(df3[col])

# Display the first few rows of the new dataset
df3.head()
```

Out[20]:

	tenure	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	SeniorCitiz
0	1	29.85	29.85	0	1	0	
1	34	56.95	1889.50	0	0	1	
2	2	53.85	108.15	1	0	1	
3	45	42.30	1840.75	0	0	1	
4	2	70.70	151.65	1	1	0	

5 rows × 47 columns

In [21]:

```
#Check for null values
df3.isnull().sum()
```

Out[21]:

```
tenure                                     0
MonthlyCharges                             0
TotalCharges                               0
Churn                                      0
gender_Female                              0
gender_Male                                0
SeniorCitizen_No                           0
SeniorCitizen_Yes                          0
Partner_No                                 0
Partner_Yes                                0
Dependents_No                              0
Dependents_Yes                             0
PhoneService_No                            0
PhoneService_Yes                           0
MultipleLines_No                           0
MultipleLines_No phone service             0
MultipleLines_Yes                          0
InternetService_DSL                        0
InternetService_Fiber optic                0
InternetService_No                         0
OnlineSecurity_No                          0
OnlineSecurity_No internet service         0
OnlineSecurity_Yes                         0
OnlineBackup_No                            0
OnlineBackup_No internet service           0
OnlineBackup_Yes                           0
DeviceProtection_No                        0
DeviceProtection_No internet service       0
DeviceProtection_Yes                       0
TechSupport_No                             0
TechSupport_No internet service            0
TechSupport_Yes                            0
StreamingTV_No                             0
StreamingTV_No internet service            0
StreamingTV_Yes                            0
StreamingMovies_No                         0
StreamingMovies_No internet service        0
StreamingMovies_Yes                        0
Contract_Month-to-month                    0
Contract_One year                          0
Contract_Two year                          0
PaperlessBilling_No                        0
PaperlessBilling_Yes                       0
PaymentMethod_Bank transfer (automatic)    0
PaymentMethod_Credit card (automatic)      0
PaymentMethod_Electronic check             0
PaymentMethod_Mailed check                 0
dtype: int64
```

Machine Learning

In [22]:

```
# Define the features and target variables
X = df3.drop(columns = ['Churn'])
y = df3['Churn'].values
```

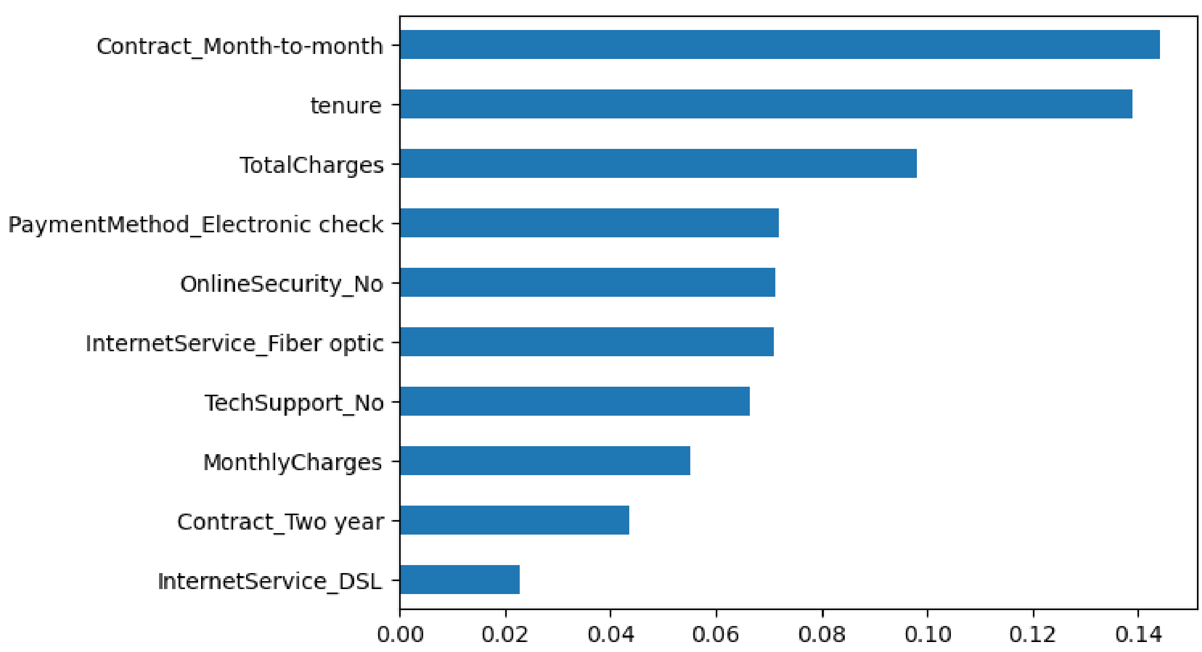
In [23]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30, random_st
```

In [24]:
```
# RandomForest
model_rf = RandomForestClassifier(n_estimators=500 , oob_score = True, n_jobs = -1,
                                  random_state =50, max_features = "auto",
                                  max_leaf_nodes = 30)
# Fit the model to the training data
model_rf.fit(X_train, y_train)

# Make predictions
prediction_test = model_rf.predict(X_test)

# Accuracy of the model
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8099526066350711
```

In [25]:
```
# Show the feature's importances
importances = model_rf.feature_importances_
weights = pd.Series(importances,
                 index=X.columns.values)
# and the last 10 items ([-10:]) represent the top 10 features with the highest imp
# These features have the greatest positive impact on the churn rate prediction.
weights.sort_values()[-10:].plot(kind = 'barh')
```

Out[25]: <Axes: >



In [26]:
```
# Split the data into training and testing sets. Here, a different random_state is
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

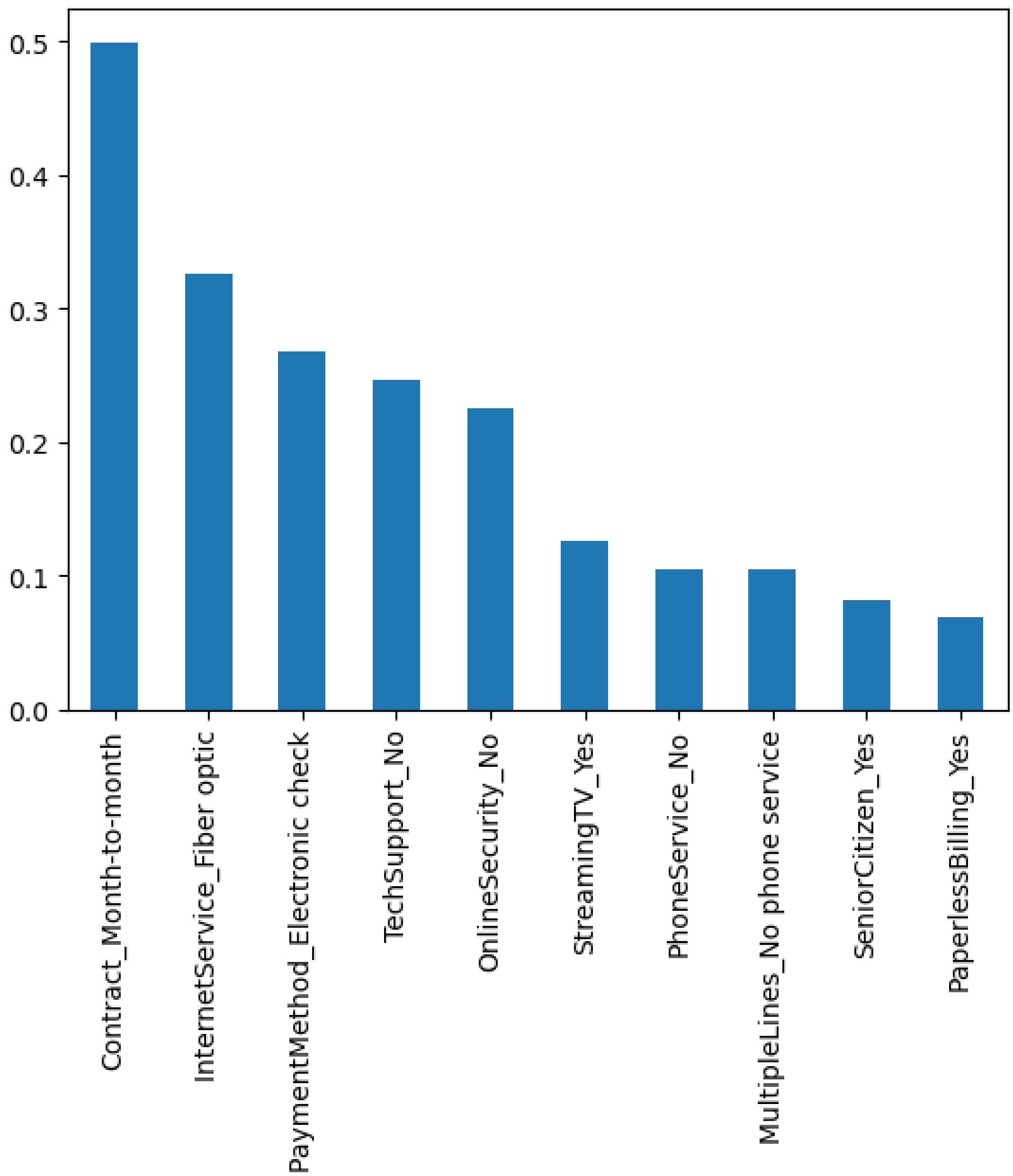
In [27]:
```
# Running logistic regression model

model = LogisticRegression()
```

```python
# Fit the model to the training data
result = model.fit(X_train, y_train)

prediction_test = model.predict(X_test)
# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8037914691943128
```

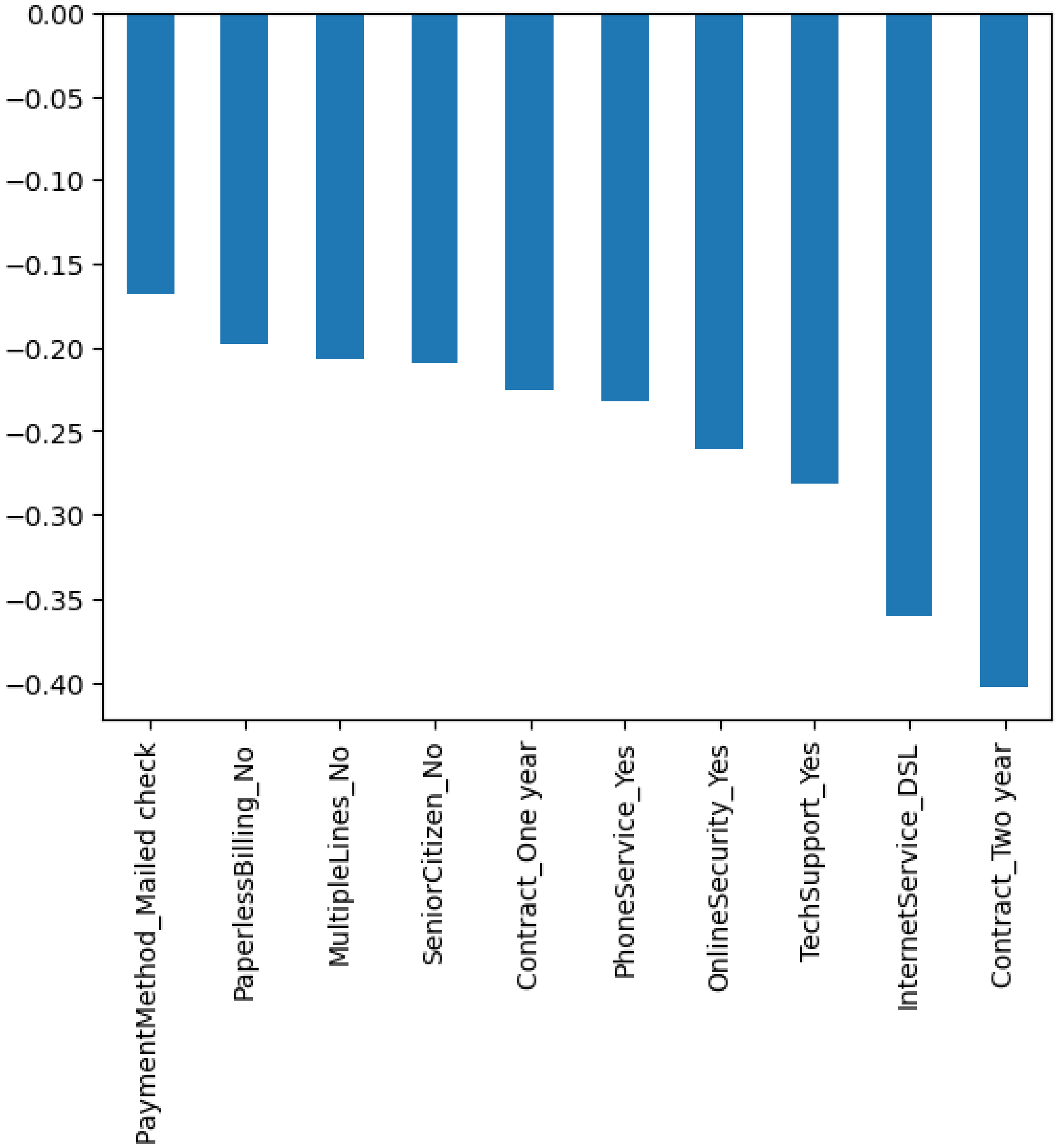
In [28]:
```python
# To get the weights of all the variables
weights = pd.Series(model.coef_[0],
                 index=X.columns.values)
# and the first 10 items ([:10]) represent the top 10 features with the highest pos
# These features have the greatest positive impact on the churn prediction.
print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))
```

```
Axes(0.125,0.11;0.775x0.77)
```



In [29]:
```
# and the last 10 items ([-10:]) represent the top 10 features with the lowest (mos
# These features have the greatest negative impact on the churn prediction(decrease
print(weights.sort_values(ascending = False)[-10:].plot(kind='bar'))
```

Axes(0.125,0.11;0.775x0.77)



In [30]:
```
# Split the data into training and testing sets. Here, a different random_state is
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

In [31]:
```
#Decision Tree Classifier
dt_model = DecisionTreeClassifier()

# Fit the model on the training data.
dt_model.fit(X_train,y_train)

# Use the trained model to predict the target variable (churn) in the test dataset
predictdt_y = dt_model.predict(X_test)

# Accuracy score of the Decision Tree model
```

```python
accuracy_dt = dt_model.score(X_test,y_test)
print("Decision Tree accuracy is :",accuracy_dt)
```

```
Decision Tree accuracy is : 0.7184834123222749
```

In [32]:
```python
print(classification_report(y_test, predictdt_y))
```

```
              precision    recall  f1-score   support

           0       0.81      0.80      0.81      1547
           1       0.47      0.49      0.48       563

    accuracy                           0.72      2110
   macro avg       0.64      0.65      0.64      2110
weighted avg       0.72      0.72      0.72      2110
```

In [33]:
```python
# Get the feature importances
importances_dt = dt_model.feature_importances_

# Create a pandas series with the feature importances
weights_dt = pd.Series(importances_dt, index=X.columns.values)

# Plot the 10 features with the highest feature importance
weights_dt.sort_values()[-10:].plot(kind = 'barh')
plt.title("Top 10 Features that have the most positive impact on Churn prediction i
plt.show()
```