In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
```

In [2]:

```
sns.set(style = 'white')
```

In [3]:

```
telecom_cust = pd.read_csv('churn.csv')
telecom_cust.head()
```

Out[3]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	To
0	LA	117	408	No	No	0	184.5	97	31.37	351.6	
1	IN	65	415	No	No	0	129.1	137	21.95	228.5	
2	NY	161	415	No	No	0	332.9	67	56.59	317.8	
3	SC	111	415	No	No	0	110.4	103	18.77	137.3	
4	HI	49	510	No	No	0	119.3	117	20.28	215.1	

In [6]:

```
telecom_cust.columns.values
```

Out[6]:

```
array(['State', '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 calls', 'Total intl charge', 'Customer service calls',
       'Churn'], dtype=object)
```

In [8]:

```
telecom_cust.isnull().sum()
```

Out[8]:

```
State                   0
Account length          0
Area code               0
International plan      0
Voice mail plan         0
Number vmail messages   0
Total day minutes       0
Total day calls         0
Total day charge        0
Total eve minutes       0
Total eve calls         0
Total eve charge        0
Total night minutes     0
Total night calls       0
Total night charge      0
Total intl minutes      0
Total intl calls        0
Total intl charge       0
Customer service calls  0
Churn                   0
dtype: int64
```

In [9]:

```
telecom_cust.dropna(inplace = True)
```

In [10]:

```
df2 = telecom_cust.iloc[:,1:]
```

In [11]:

```
df2['Churn'].replace(to_replace='Yes', value=1, inplace=True)
df2['Churn'].replace(to_replace='No',  value=0, inplace=True)
```

In [12]:

```
df_dummies = pd.get_dummies(df2)
df_dummies.head()
```

Out[12]:

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	...	c
0	117	408	0	184.5	97	31.37	351.6	80	29.89	215.8	...	
1	65	415	0	129.1	137	21.95	228.5	83	19.42	208.8	...	
2	161	415	0	332.9	67	56.59	317.8	97	27.01	160.6	...	
3	111	415	0	110.4	103	18.77	137.3	102	11.67	189.6	...	
4	49	510	0	119.3	117	20.28	215.1	109	18.28	178.7	...	

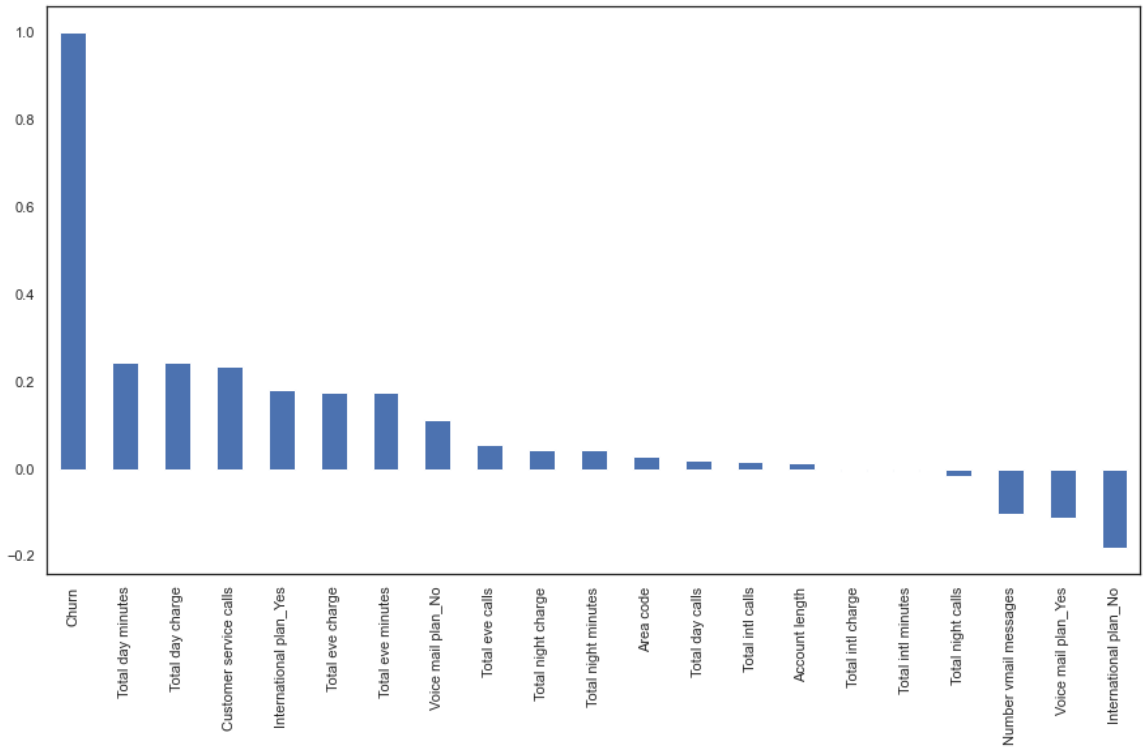
5 rows × 21 columns

In [13]:

```
plt.figure(figsize=(15,8))
df_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

Out[13]:

<AxesSubplot:>



In [15]:

```
y = df_dummies['Churn'].values
X = df_dummies.drop(columns = ['Churn'])

# Scaling all the variables to a range of 0 to 1
from sklearn.preprocessing import MinMaxScaler
features = X.columns.values
scaler = MinMaxScaler(feature_range = (0,1))
scaler.fit(X)
X = pd.DataFrame(scaler.transform(X))
X.columns = features
```

In [16]:

```
#Logistic Regression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101
```

In [17]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
result = model.fit(X_train, y_train)
```

In [18]:

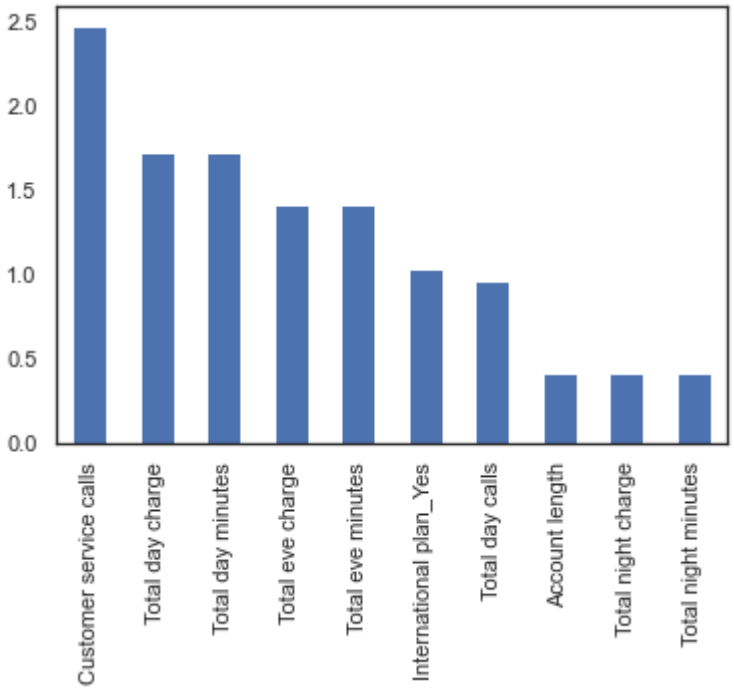
```
from sklearn import metrics
prediction_test = model.predict(X_test)
# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.835820895522388
```

In [19]:

```
weights = pd.Series(model.coef_[0],
                 index=X.columns.values)
print (weights.sort_values(ascending = False)[:10].plot(kind='bar'))
```

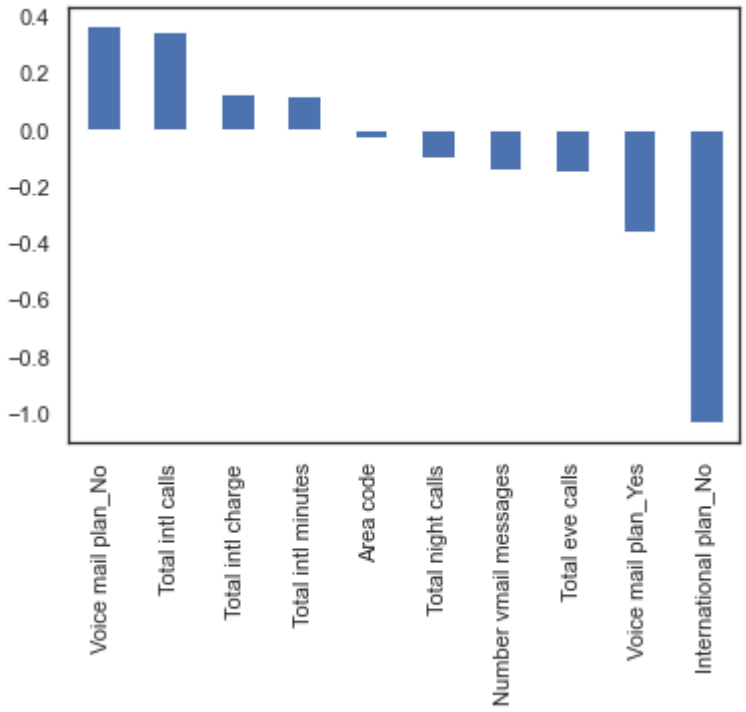
```
AxesSubplot(0.125,0.125;0.775x0.755)
```



In [20]:

```
print(weights.sort_values(ascending = False)[-10:].plot(kind='bar'))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



In [21]:

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101
model_rf = RandomForestClassifier(n_estimators=1000 , oob_score = True, n_jobs = -1,
                                  random_state =50, max_features = "auto",
                                  max_leaf_nodes = 30)
model_rf.fit(X_train, y_train)
```

Out[21]:

```
RandomForestClassifier(max_leaf_nodes=30, n_estimators=1000, n_jobs=-1,
                       oob_score=True, random_state=50)
```

In [22]:

```
prediction_test = model_rf.predict(X_test)
print (metrics.accuracy_score(y_test, prediction_test))
```

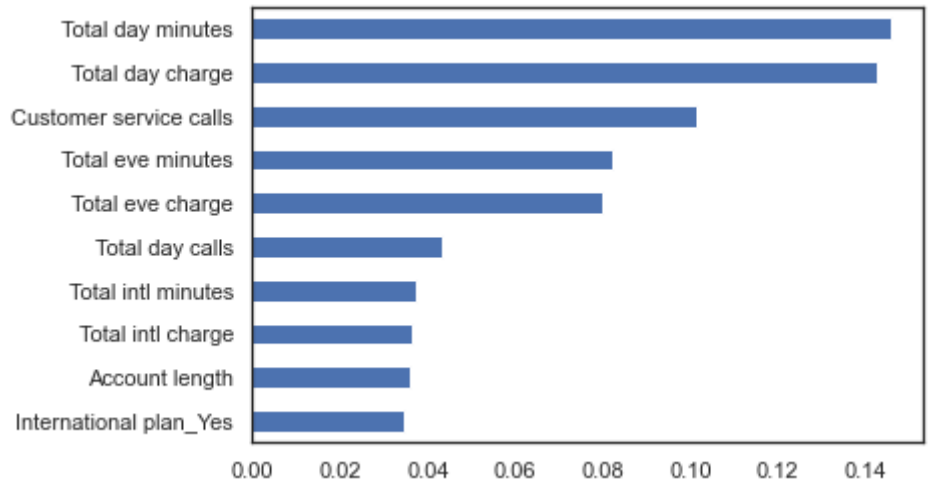
```
0.9029850746268657
```

In [23]:

```
importances = model_rf.feature_importances_
weights = pd.Series(importances,
                    index=X.columns.values)
weights.sort_values()[-10:].plot(kind = 'barh')
```

Out[23]:

```
<AxesSubplot:>
```



In [24]:

```
#SVM
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=99)
```

In [25]:

```
from sklearn.svm import SVC

model.svm = SVC(kernel='linear')
model.svm.fit(X_train,y_train)
preds = model.svm.predict(X_test)
metrics.accuracy_score(y_test, preds)
```

Out[25]:

```
0.8432835820895522
```

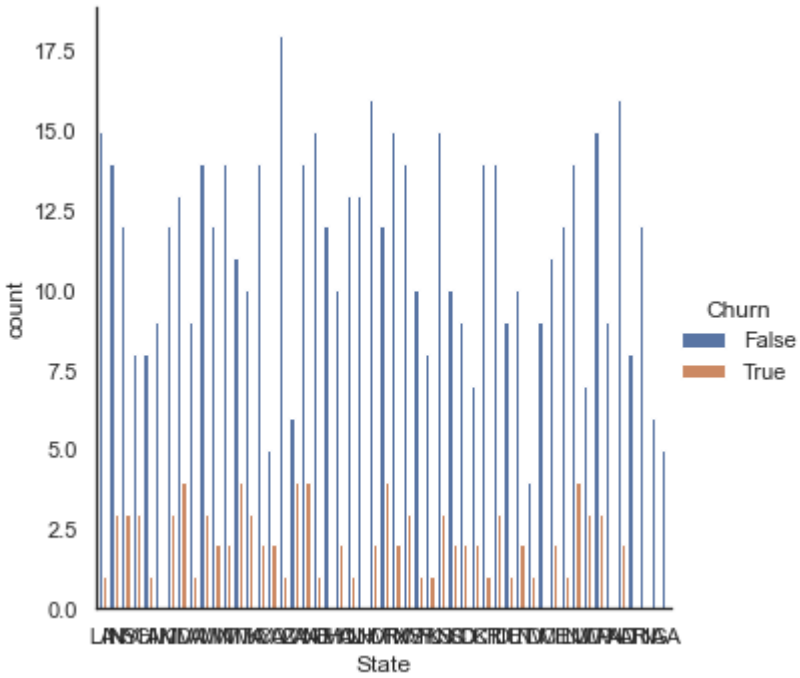
In [26]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,preds))
```

```
[[113   0]
 [ 21   0]]
```

In [28]:

```
ax1 = sns.catplot(x="State", kind="count", hue="Churn", data=telecom_cust,
                  estimator=lambda x: sum(x==0)*100.0/len(x))
```



In [ ]: