DataVisualization

March 8, 2024

1 Milestone 1: data Visualization

This file contains some general data exploratory codes, comments and plots. We further used the visualizations in this file to make some planned decisions about feature engineering.

[38]:
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import roc_curve, roc_auc_score
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[39]:
```python
df = pd.read_csv("Assign1Data.csv")
df.head()
```

[39]:
```
   Gender Senior Citizen Partner  Tenure Months Phone Service Multiple Lines  \
0    Male             No      No              2           Yes             No
1  Female             No      No              2           Yes             No
2  Female             No      No              8           Yes            Yes
3  Female             No     Yes             28           Yes            Yes
4    Male             No      No             49           Yes            Yes

  Internet Service Online Security Online Backup Device Protection  \
0              DSL             Yes           Yes                No
1      Fiber optic              No            No                No
2      Fiber optic              No            No               Yes
3      Fiber optic              No            No               Yes
4      Fiber optic              No           Yes               Yes

  Tech Support Streaming TV Streaming Movies        Contract  \
0           No           No               No  Month-to-month
1           No           No               No  Month-to-month
```

```
2           No          Yes              Yes  Month-to-month
3          Yes          Yes              Yes  Month-to-month
4           No          Yes              Yes  Month-to-month

  Paperless Billing            Payment Method  Monthly Charges  \
0               Yes              Mailed check            53.85
1               Yes          Electronic check            70.70
2               Yes          Electronic check            99.65
3               Yes          Electronic check           104.80
4               Yes  Bank transfer (automatic)          103.70

   Total Charges  Churn Value
0         108.15            1
1         151.65            1
2         820.50            1
3        3046.05            1
4        5036.30            1
```

[40]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             7043 non-null   object
 1   Senior Citizen     7043 non-null   object
 2   Partner            7043 non-null   object
 3   Tenure Months      7043 non-null   int64
 4   Phone Service      7043 non-null   object
 5   Multiple Lines     7043 non-null   object
 6   Internet Service   7043 non-null   object
 7   Online Security    7043 non-null   object
 8   Online Backup      7043 non-null   object
 9   Device Protection  7043 non-null   object
 10  Tech Support       7043 non-null   object
 11  Streaming TV       7043 non-null   object
 12  Streaming Movies   7043 non-null   object
 13  Contract           7043 non-null   object
 14  Paperless Billing  7043 non-null   object
 15  Payment Method     7043 non-null   object
 16  Monthly Charges    7043 non-null   float64
 17  Total Charges      7043 non-null   float64
 18  Churn Value        7043 non-null   int64
dtypes: float64(2), int64(2), object(15)
memory usage: 1.0+ MB
```

[41]:
```python
print(df.columns)
```

```
Index(['Gender', 'Senior Citizen', 'Partner', 'Tenure Months', 'Phone Service',
       'Multiple Lines', 'Internet Service', 'Online Security',
       'Online Backup', 'Device Protection', 'Tech Support', 'Streaming TV',
       'Streaming Movies', 'Contract', 'Paperless Billing', 'Payment Method',
       'Monthly Charges', 'Total Charges', 'Churn Value'],
      dtype='object')
```

Here we printed all the categories there are in each columns.

[42]:
```
for col in df.columns:
    print(f"Column {col}, categories {df[col].unique()}")
```

```
Column Gender, categories ['Male' 'Female']
Column Senior Citizen, categories ['No' 'Yes']
Column Partner, categories ['No' 'Yes']
Column Tenure Months, categories [ 2  8 28 49 10  1 47 17  5 34 11 15 18  9  7
12 25 68 55 37  3 27 20  4
 58 53 13  6 19 59 16 52 24 32 38 54 43 63 21 69 22 61 60 48 40 23 39 35
 56 65 33 30 45 46 62 70 50 44 71 26 14 41 66 64 29 42 67 51 31 57 36 72
  0]
Column Phone Service, categories ['Yes' 'No']
Column Multiple Lines, categories ['No' 'Yes' 'No phone service']
Column Internet Service, categories ['DSL' 'Fiber optic' 'No']
Column Online Security, categories ['Yes' 'No' 'No internet service']
Column Online Backup, categories ['Yes' 'No' 'No internet service']
Column Device Protection, categories ['No' 'Yes' 'No internet service']
Column Tech Support, categories ['No' 'Yes' 'No internet service']
Column Streaming TV, categories ['No' 'Yes' 'No internet service']
Column Streaming Movies, categories ['No' 'Yes' 'No internet service']
Column Contract, categories ['Month-to-month' 'Two year' 'One year']
Column Paperless Billing, categories ['Yes' 'No']
Column Payment Method, categories ['Mailed check' 'Electronic check' 'Bank
transfer (automatic)'
 'Credit card (automatic)']
Column Monthly Charges, categories [ 53.85  70.7   99.65 … 108.35  63.1   78.7
]
Column Total Charges, categories [ 108.15  151.65  820.5  … 7362.9   346.45
6844.5 ]
Column Churn Value, categories [1 0]
```

Here the describe function only showed the attributes of the columns which had numerical data. Later we converted all the categorical values in integers and used describe again. That can be seen further down.

[43]:
```
df.describe()
```

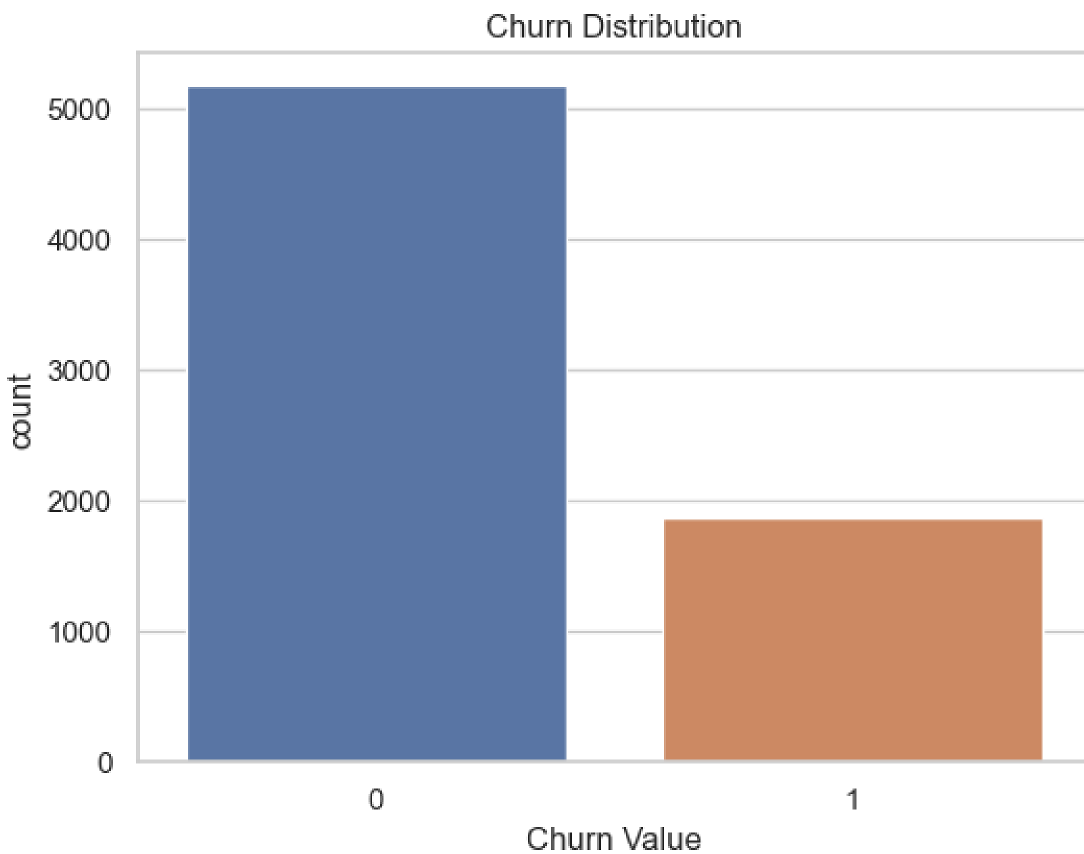
[43]:

	Tenure Months	Monthly Charges	Total Charges	Churn Value
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	32.371149	64.761692	2283.300441	0.265370

```
std        24.559481       30.090047    2265.000258       0.441561
min         0.000000       18.250000      18.800000       0.000000
25%         9.000000       35.500000     402.225000       0.000000
50%        29.000000       70.350000    1400.550000       0.000000
75%        55.000000       89.850000    3786.600000       1.000000
max        72.000000      118.750000    8684.800000       1.000000
```

[44]:

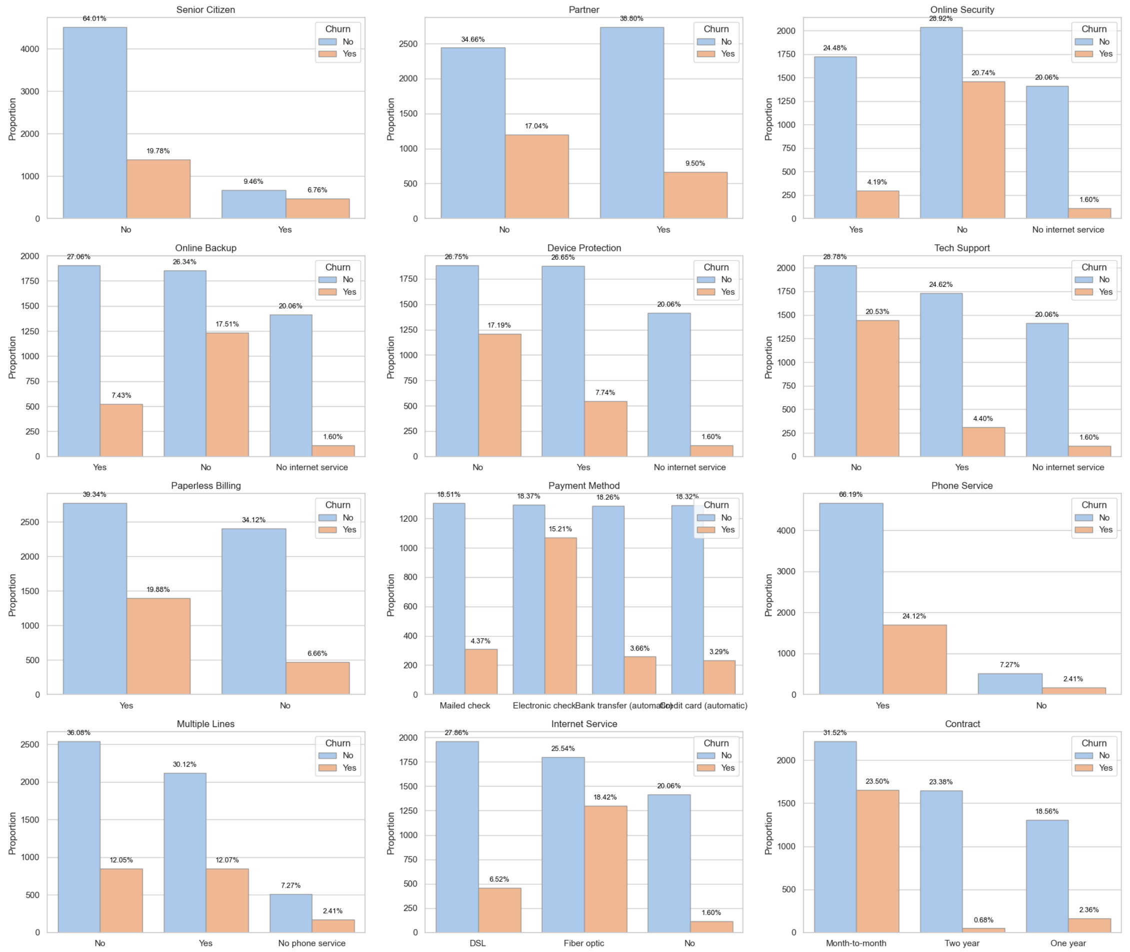
```
# Showing Churn distribution

sns.set(style="whitegrid")

sns.countplot(x='Churn Value', data=df)
plt.title('Churn Distribution')
plt.show()
```



This bar chart illustrates the distribution of churn. The significant imbalance between the retained and churned customer is crucial to consider when selecting performance metrics for model evaluation. Models like Random Forest Classifier and Decision Tree are insensitive to such imbalances. We can use techniques like SMOTE for data balancing before training more sensitive models such as Logistic Regression or SVM.

[45]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

plt.figure(figsize=(20, 17))

columns_to_plot = [
    'Senior Citizen', 'Partner', 'Online Security',
    'Online Backup', 'Device Protection', 'Tech Support',
    'Paperless Billing', 'Payment Method',
    'Phone Service', 'Multiple Lines', 'Internet Service', 'Contract'
]

for i, column in enumerate(columns_to_plot, 1):
    plt.subplot(4, 3, i)  #grid dimension
    ax = sns.countplot(x=column, hue='Churn Value', data=df, palette='pastel',␣
 ↪edgecolor='.6')
    plt.title(column.replace('_', ' ').title())
    plt.xlabel('')
    plt.ylabel('Proportion')
    plt.legend(title='Churn', labels=['No', 'Yes'], loc='upper right')
    total = len(df[column])

    # adding the normalized percentage on top of the bars
    for p in ax.patches:
        height = p.get_height()
        percentage = '{:1.2f}%'.format(100 * height/total)
        ax.annotate(percentage, (p.get_x() + p.get_width() / 2., height),␣
 ↪ha='center', va='center', fontsize=9, color='black', xytext=(0, 10),␣
 ↪textcoords='offset points')

plt.tight_layout()
plt.show()
```

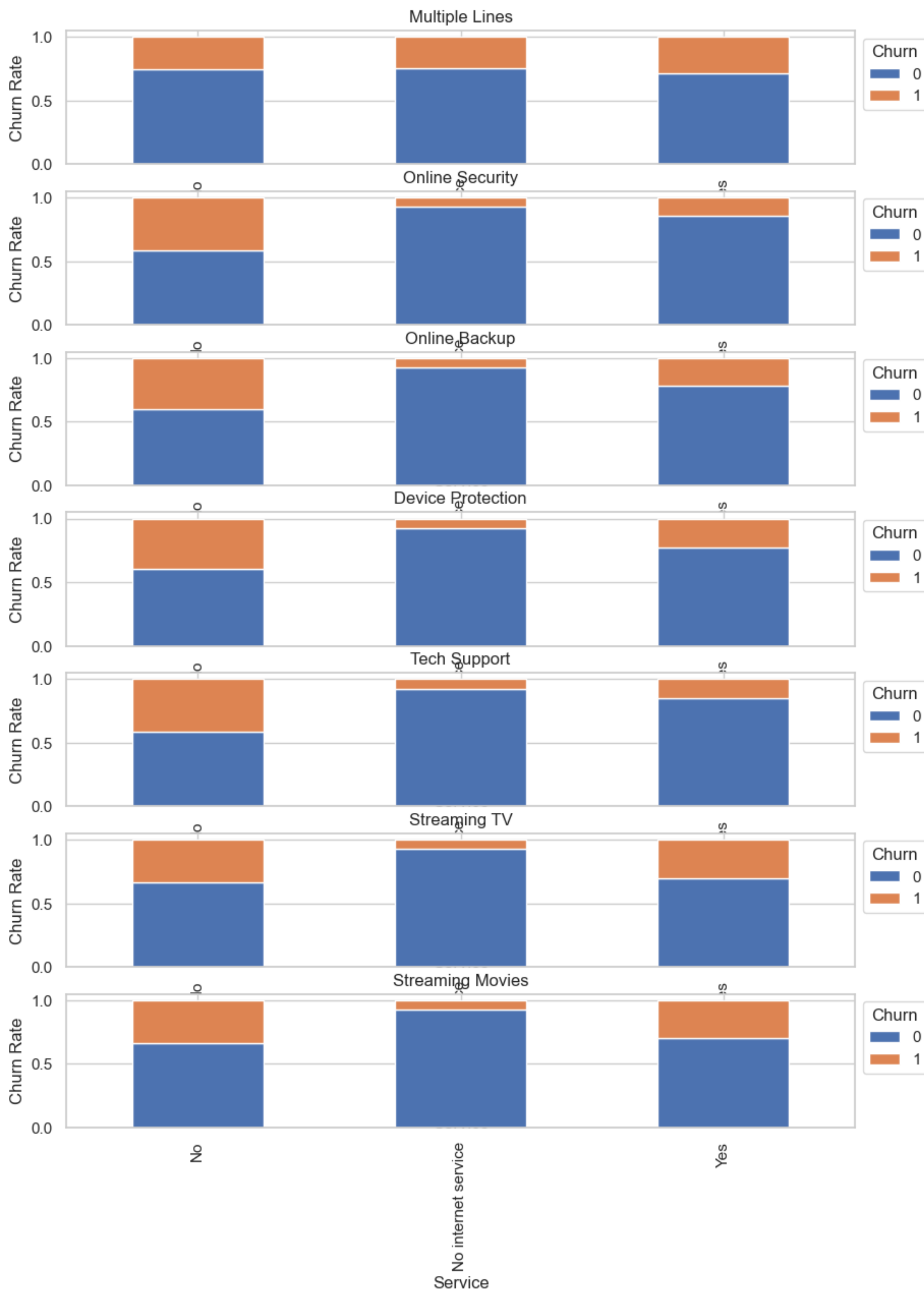


The churn rate of customers is influenced by various factors, including age, gender, and payment method. Almost half od the senior citizen can be seen to be churned, suggesting the need for more effort to retain senior citizen by providing more comfort. Customers with partners are slightly less likely to churn, suggesting services that appeal to couples might keep them loyal. Online security services significantly decrease churn, highlighting the importance of security features. Tech support also significantly reduces churn, underlining the importance of customer service. Payment methods also impact churn rates, especially electronic check, they seem to be causing comparatively large effect on customer churn. The set of plots above give tons of such insights, to make it more easily readable we plotted a churn correlation plot in the end which shows how much a particular column as a whole effects churn.

[46]:
```
import matplotlib.pyplot as plt
import seaborn as sns

service_columns = ['Multiple Lines', 'Online Security',
                   'Online Backup', 'Device Protection', 'Tech Support',
                   'Streaming TV', 'Streaming Movies']
```

```python
plt.figure(figsize=(10, 14))

for i, column in enumerate(service_columns, 1):
    ax = plt.subplot(len(service_columns), 1, i)
    churn_rate = df.groupby(column)['Churn Value'].value_counts(normalize=True).
  ↪unstack()
    churn_rate.plot(kind='bar', stacked=True, ax=ax)
    plt.title(column)
    plt.xlabel('Service')
    plt.ylabel('Churn Rate')
    plt.legend(title='Churn', bbox_to_anchor=(1, 1))

plt.tight_layout()
plt.show()
```

```
/var/folders/4z/qh15q30n25nckhc10gfcfdwh0000gn/T/ipykernel_30291/4077526128.py:1
9: UserWarning: Tight layout not applied. tight_layout cannot make axes height
small enough to accommodate all axes decorations.
  plt.tight_layout()
```



The plot shows above have some of its label hidden, The ones in the middle are “No phone service” and “No Internet service”. The columns which are plotted had 3 categories in them, multiple lines had “No phone service” value in it if the Phone service column had “No” in that row. As for the rest of the services they had a categorical value “No internet service” in the row where theInternet Service column had the value “No”. Here we just plotted all those categories and the effect they had on churn to decide if any can be further worked upon during feature engineering to optime our models performance.

[47]:
```python
# Converting the categorical values into integers

for col in df.select_dtypes(include=['object']).columns:
    df[col] = df[col].astype('category').cat.codes

df.head()
```

[47]:
```
   Gender  Senior Citizen  Partner  Tenure Months  Phone Service  \
0       1               0        0              2              1
1       0               0        0              2              1
2       0               0        0              8              1
3       0               0        1             28              1
4       1               0        0             49              1

   Multiple Lines  Internet Service  Online Security  Online Backup  \
0               0                 0                2              2
1               0                 1                0              0
2               2                 1                0              0
3               2                 1                0              0
4               2                 1                0              2

   Device Protection  Tech Support  Streaming TV  Streaming Movies  Contract  \
0                  0             0             0                 0         0
1                  0             0             0                 0         0
2                  2             0             2                 2         0
3                  2             2             2                 2         0
4                  2             0             2                 2         0

   Paperless Billing  Payment Method  Monthly Charges  Total Charges  \
0                  1               3            53.85         108.15
1                  1               2            70.70         151.65
2                  1               2            99.65         820.50
3                  1               2           104.80        3046.05
4                  1               0           103.70        5036.30

   Churn Value
0            1
1            1
2            1
3            1
```

```
4             1
```

[49]:
```
df.describe()
```

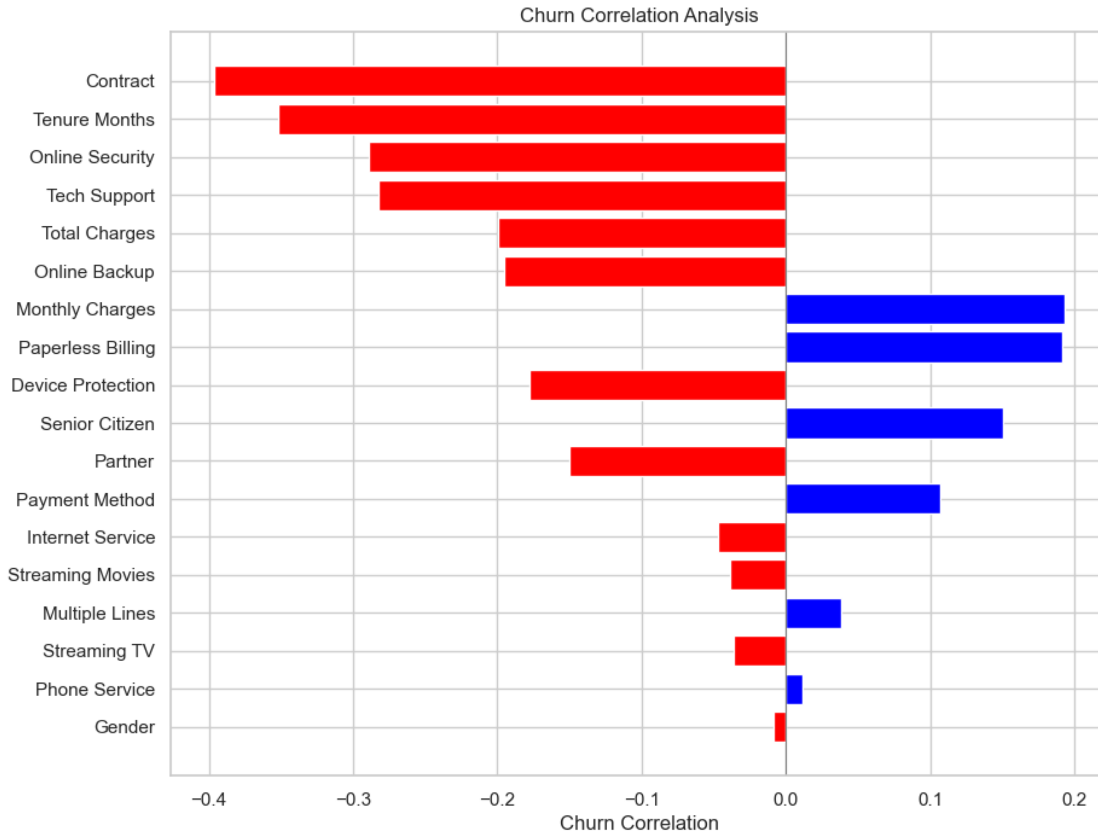
[49]:
```
            Gender  Senior Citizen      Partner  Tenure Months  Phone Service  \
count  7043.000000     7043.000000  7043.000000    7043.000000    7043.000000
mean      0.504756        0.162147     0.483033      32.371149       0.903166
std       0.500013        0.368612     0.499748      24.559481       0.295752
min       0.000000        0.000000     0.000000       0.000000       0.000000
25%       0.000000        0.000000     0.000000       9.000000       1.000000
50%       1.000000        0.000000     0.000000      29.000000       1.000000
75%       1.000000        0.000000     1.000000      55.000000       1.000000
max       1.000000        1.000000     1.000000      72.000000       1.000000

       Multiple Lines  Internet Service  Online Security  Online Backup  \
count     7043.000000       7043.000000      7043.000000    7043.000000
mean         0.940508          0.872923         0.790004       0.906432
std          0.948554          0.737796         0.859848       0.880162
min          0.000000          0.000000         0.000000       0.000000
25%          0.000000          0.000000         0.000000       0.000000
50%          1.000000          1.000000         1.000000       1.000000
75%          2.000000          1.000000         2.000000       2.000000
max          2.000000          2.000000         2.000000       2.000000

       Device Protection  Tech Support  Streaming TV  Streaming Movies  \
count        7043.000000   7043.000000   7043.000000       7043.000000
mean            0.904444      0.797104      0.985376          0.992475
std             0.879949      0.861551      0.885002          0.885091
min             0.000000      0.000000      0.000000          0.000000
25%             0.000000      0.000000      0.000000          0.000000
50%             1.000000      1.000000      1.000000          1.000000
75%             2.000000      2.000000      2.000000          2.000000
max             2.000000      2.000000      2.000000          2.000000

          Contract  Paperless Billing  Payment Method  Monthly Charges  \
count  7043.000000        7043.000000     7043.000000      7043.000000
mean      0.690473           0.592219        1.574329        64.761692
std       0.833755           0.491457        1.068104        30.090047
min       0.000000           0.000000        0.000000        18.250000
25%       0.000000           0.000000        1.000000        35.500000
50%       0.000000           1.000000        2.000000        70.350000
75%       1.000000           1.000000        2.000000        89.850000
max       2.000000           1.000000        3.000000       118.750000

       Total Charges  Churn Value
count    7043.000000  7043.000000
mean     2283.300441     0.265370
```

```
std      2265.000258     0.441561
min        18.800000     0.000000
25%       402.225000     0.000000
50%      1400.550000     0.000000
75%      3786.600000     1.000000
max      8684.800000     1.000000
```

As promised, describing the data frame after converting the categorical values into integers. We can see mean to find out if the column is balanced. Min and max to see how many categories are there in each columns. Using the percentiles we can also have an idea as to how to data is distributed.

[48]:

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier

X = pd.get_dummies(df.drop('Churn Value', axis=1))
y = df['Churn Value']

# Calculate the correlation of each feature with the target
correlations = X.apply(lambda x: x.corr(df['Churn Value']))

# create a dataframe of the correlations calculated using th elambda function␣
↪above
features_df = pd.DataFrame({'Correlation': correlations})

# Sort the DataFrame based on the absolute values of the correlations
features_df = features_df.sort_values(by='Correlation', key=abs, ascending=True)

# Create the color map based on the correlation values
colors = ['red' if x < 0 else 'blue' for x in features_df['Correlation']]

# Plot the figure
plt.figure(figsize=(10, 8))
bars = plt.barh(features_df.index, features_df['Correlation'], color=colors)
plt.xlabel('Churn Correlation')
plt.title('Churn Correlation Analysis')
plt.axvline(x=0, color='grey', linewidth=0.8)
plt.show()
```



The horizontal bar chart presented, shows the correlation of various features with Churn Value. Features with positive correlations are colored blue, they suggest a positive correlation and a tendency to increase churn possibility (simply leading to customer leaving Telco company). On the other hand, features with negative correlations are colored red and are suggesting a potential retention factors (it can be noted that most o fthe columns are leading to retention and thus our data set is imbalanced). The graph is sorted to prioritize features with stronger correlations on the top and weaker on the bottom. This helps in feature selection and provids insights into areas that may require more focused customer retention strategies to further decrease customer churn.