ChurnPrediction

October 13, 2023

0.1 Introduction

Subscription services are leveraged by companies across many industries, from fitness to video streaming to retail. One of the primary objectives of companies with subscription services is to decrease churn and ensure that users are retained as subscribers. In order to do this efficiently and systematically, many companies employ machine learning to predict which users are at the highest risk of churn, so that proper interventions can be effectively deployed to the right audience.

In this project, I will be tackling the churn prediction problem on a very unique and interesting group of subscribers on a video streaming service.

0.1.1 Train vs. Test

The data is available here.

The data is already split into train and test set.

`train.csv` contains 70% of the overall sample (243,787 subscriptions to be exact) and importantly, will reveal whether or not the subscription was continued into the next month (the "ground truth").

The `test.csv` dataset contains the exact same information about the remaining segment of the overall sample (104,480 subscriptions to be exact) except the actual churn data. The goal of this project is to make prediction on each customer's churn.

0.1.2 Dataset descriptions

[1]:
```
import pandas as pd
data_descriptions = pd.read_csv('data_descriptions.csv')
pd.set_option('display.max_colwidth', None)
data_descriptions
```

[1]:
```
                Column_name Column_type Data_type  \
0                AccountAge     Feature   integer
1            MonthlyCharges     Feature     float
2              TotalCharges     Feature     float
3          SubscriptionType     Feature    object
4             PaymentMethod     Feature    string
5          PaperlessBilling     Feature    string
```

```
6              ContentType     Feature   string
7         MultiDeviceAccess     Feature   string
8          DeviceRegistered     Feature   string
9        ViewingHoursPerWeek     Feature    float
10    AverageViewingDuration     Feature    float
11  ContentDownloadsPerMonth     Feature  integer
12          GenrePreference     Feature   string
13               UserRating     Feature    float
14    SupportTicketsPerMonth     Feature  integer
15                   Gender     Feature   string
16            WatchlistSize     Feature    float
17          ParentalControl     Feature   string
18         SubtitlesEnabled     Feature   string
19               CustomerID  Identifier   string
20                    Churn      Target  integer

            Description
0                                                               The age of the
user's account in months.
1                                                     The amount charged to the
user on a monthly basis.
2                                     The total charges incurred by the user over
the account's lifetime.
3                              The type of subscription chosen by the user
(Basic, Standard, or Premium).
4                                                                The method of
payment used by the user.
5                                 Indicates whether the user has opted for
paperless billing (Yes or No).
6                                  The type of content preferred by the user
(Movies, TV Shows, or Both).
7                  Indicates whether the user has access to the service on
multiple devices (Yes or No).
8                            The type of device registered by the user (TV,
Mobile, Tablet, or Computer).
9                                              The number of hours the user spends
watching content per week.
10                                                  The average duration of each
viewing session in minutes.
11                                                    The number of content
downloads by the user per month.
12                                                        The preferred genre of
content chosen by the user.
13                                                   The user's rating for the
service on a scale of 1 to 5.
14                                              The number of support tickets
raised by the user per month.
```

```
15                                                              The gender of
the user (Male or Female).
16                                                          The number of items
in the user's watchlist.
17                              Indicates whether parental control is enabled
for the user (Yes or No).
18                                    Indicates whether subtitles are enabled
for the user (Yes or No).
19                                                                  A unique
identifier for each customer.
20  The target variable indicating whether a user has churned or not (1 for
churned, 0 for not churned).
```

0.2 Setup

[3]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.dummy import DummyClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,␣
 ↪recall_score, ConfusionMatrixDisplay
from scipy.stats import randint
from matplotlib import pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline
```

0.3 Load the Data

[4]:
```
train_df = pd.read_csv("train.csv")
print('train_df Shape:', train_df.shape)
train_df.head()
```

```
train_df Shape: (243787, 21)
```

[4]:
```
   AccountAge  MonthlyCharges  TotalCharges SubscriptionType  \
0          20       11.055215    221.104302          Premium
1          57        5.175208    294.986882            Basic
2          73       12.106657    883.785952            Basic
3          32        7.263743    232.439774            Basic
4          57       16.953078    966.325422          Premium

      PaymentMethod PaperlessBilling ContentType MultiDeviceAccess  \
```

```
0      Mailed check              No        Both               No
1       Credit card             Yes      Movies               No
2      Mailed check             Yes      Movies               No
3  Electronic check              No    TV Shows               No
4  Electronic check             Yes    TV Shows               No

  DeviceRegistered  ViewingHoursPerWeek  …  ContentDownloadsPerMonth  \
0           Mobile            36.758104  …                        10
1           Tablet            32.450568  …                        18
2         Computer             7.395160  …                        23
3           Tablet            27.960389  …                        30
4               TV            20.083397  …                        20

   GenrePreference UserRating  SupportTicketsPerMonth  Gender WatchlistSize  \
0           Sci-Fi   2.176498                       4    Male             3
1           Action   3.478632                       8    Male            23
2          Fantasy   4.238824                       6    Male             1
3            Drama   4.276013                       2    Male            24
4           Comedy   3.616170                       4  Female             0

   ParentalControl SubtitlesEnabled  CustomerID Churn
0               No               No  CB6SXPNVZA     0
1               No              Yes  S7R2G87O09     0
2              Yes              Yes  EASDC20BDT     0
3              Yes              Yes  NPF69NT69N     0
4               No               No  4LGYPK7VOL     0

[5 rows x 21 columns]
```

[5]:
```python
test_df = pd.read_csv("test.csv")
print('test_df Shape:', test_df.shape)
```

```
test_df Shape: (104480, 20)
```

0.4 Preprocess & EDA

[6]:
```python
train_df.isna().any()
```

[6]:
```
AccountAge                  False
MonthlyCharges              False
TotalCharges                False
SubscriptionType            False
PaymentMethod               False
PaperlessBilling            False
ContentType                 False
MultiDeviceAccess           False
```

```
DeviceRegistered            False
ViewingHoursPerWeek         False
AverageViewingDuration      False
ContentDownloadsPerMonth    False
GenrePreference             False
UserRating                  False
SupportTicketsPerMonth      False
Gender                      False
WatchlistSize               False
ParentalControl             False
SubtitlesEnabled            False
CustomerID                  False
Churn                       False
dtype: bool
```

[7]:
```
train_df.describe()
```

[7]:
```
          AccountAge  MonthlyCharges   TotalCharges  ViewingHoursPerWeek  \
count  243787.000000   243787.000000  243787.000000        243787.000000
mean       60.083758       12.490695     750.741017            20.502179
std        34.285143        4.327615     523.073273            11.243753
min         1.000000        4.990062       4.991154             1.000065
25%        30.000000        8.738543     329.147027            10.763953
50%        60.000000       12.495555     649.878487            20.523116
75%        90.000000       16.238160    1089.317362            30.219396
max       119.000000       19.989957    2378.723844            39.999723

       AverageViewingDuration  ContentDownloadsPerMonth     UserRating  \
count           243787.000000             243787.000000  243787.000000
mean                92.264061                 24.503513       3.002713
std                 50.505243                 14.421174       1.155259
min                  5.000547                  0.000000       1.000007
25%                 48.382395                 12.000000       2.000853
50%                 92.249992                 24.000000       3.002261
75%                135.908048                 37.000000       4.002157
max                179.999275                 49.000000       4.999989

       SupportTicketsPerMonth  WatchlistSize          Churn
count           243787.000000  243787.000000  243787.000000
mean                 4.504186      12.018508       0.181232
std                  2.872548       7.193034       0.385211
min                  0.000000       0.000000       0.000000
25%                  2.000000       6.000000       0.000000
50%                  4.000000      12.000000       0.000000
75%                  7.000000      18.000000       0.000000
max                  9.000000      24.000000       1.000000
```

[8]:
```
train_df.dtypes
```

[8]:
```
AccountAge                    int64
MonthlyCharges              float64
TotalCharges                float64
SubscriptionType             object
PaymentMethod                object
PaperlessBilling             object
ContentType                  object
MultiDeviceAccess            object
DeviceRegistered             object
ViewingHoursPerWeek         float64
AverageViewingDuration      float64
ContentDownloadsPerMonth      int64
GenrePreference              object
UserRating                  float64
SupportTicketsPerMonth        int64
Gender                       object
WatchlistSize                 int64
ParentalControl              object
SubtitlesEnabled             object
CustomerID                   object
Churn                         int64
dtype: object
```

Check the classes for each categorical variable.

[9]:
```
train_df["SubscriptionType"].unique()
```

[9]:
```
array(['Premium', 'Basic', 'Standard'], dtype=object)
```

[10]:
```
train_df["PaymentMethod"].unique()
```

[10]:
```
array(['Mailed check', 'Credit card', 'Electronic check', 'Bank transfer'],
      dtype=object)
```

[11]:
```
train_df["PaperlessBilling"].unique()
```

[11]:
```
array(['No', 'Yes'], dtype=object)
```

[12]:
```
train_df["ContentType"].unique()
```

[12]:
```
array(['Both', 'Movies', 'TV Shows'], dtype=object)
```

[13]:
```
train_df["MultiDeviceAccess"].unique()
```

[13]:
```
array(['No', 'Yes'], dtype=object)
```

[14]:
```
train_df["DeviceRegistered"].unique()
```

[14]:
```
array(['Mobile', 'Tablet', 'Computer', 'TV'], dtype=object)
```

[15]:
```
train_df["GenrePreference"].unique()
```

[15]:
```
array(['Sci-Fi', 'Action', 'Fantasy', 'Drama', 'Comedy'], dtype=object)
```

[16]:
```
train_df["Gender"].unique()
```

[16]:
```
array(['Male', 'Female'], dtype=object)
```

[17]:
```
train_df["ParentalControl"].unique()
```

[17]:
```
array(['No', 'Yes'], dtype=object)
```

[18]:
```
train_df["SubtitlesEnabled"].unique()
```

[18]:
```
array(['No', 'Yes'], dtype=object)
```

Now, dummy code the categorical variables except CustomerID.

[19]:
```
train_df_new = train_df
train_df_new = train_df.drop("CustomerID", axis=1)
```

[20]:
```
categorical_columns = ['SubscriptionType', 'PaymentMethod', 'PaperlessBilling',␣
↪'ContentType', 'MultiDeviceAccess',
                      'DeviceRegistered', 'GenrePreference', 'Gender',␣
↪'ParentalControl', 'SubtitlesEnabled']

for col in categorical_columns:
    dummies = pd.get_dummies(train_df_new[col], prefix=col)
    train_df_new = pd.concat([train_df_new, dummies], axis=1)

train_df_new = train_df_new.drop(categorical_columns, axis=1)
```

Check the proportion of churn in the training set.

[21]:
```
fig = px.histogram(train_df['Churn'].astype(str), color=train_df['Churn'].
↪astype(str), title="Churn count")

# Calculate percentages
total_count = len(train_df)
percentage_labels = [(f'{churn}: {count / total_count:.2%}') for churn, count␣
↪in train_df['Churn'].value_counts().items()]

# Add percentage as a caption
```

```
fig.update_layout(
    annotations=[
        dict(
            text=', '.join(percentage_labels),
            showarrow=False,
            x=0.5,
            y=1.1,
            font=dict(size=12),
        )
    ]
)

# Show the plot
fig.show()
```

0.5 Make predictions (required)

In this project, I decided to use random forest classifier because * the dataset is pretty large * the datset has many variables to be considered * I just learned in my class and wanted to try

[22]:
```
X_train = train_df_new.drop("Churn", axis = 1)
y_train = train_df_new["Churn"]
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

[22]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=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=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

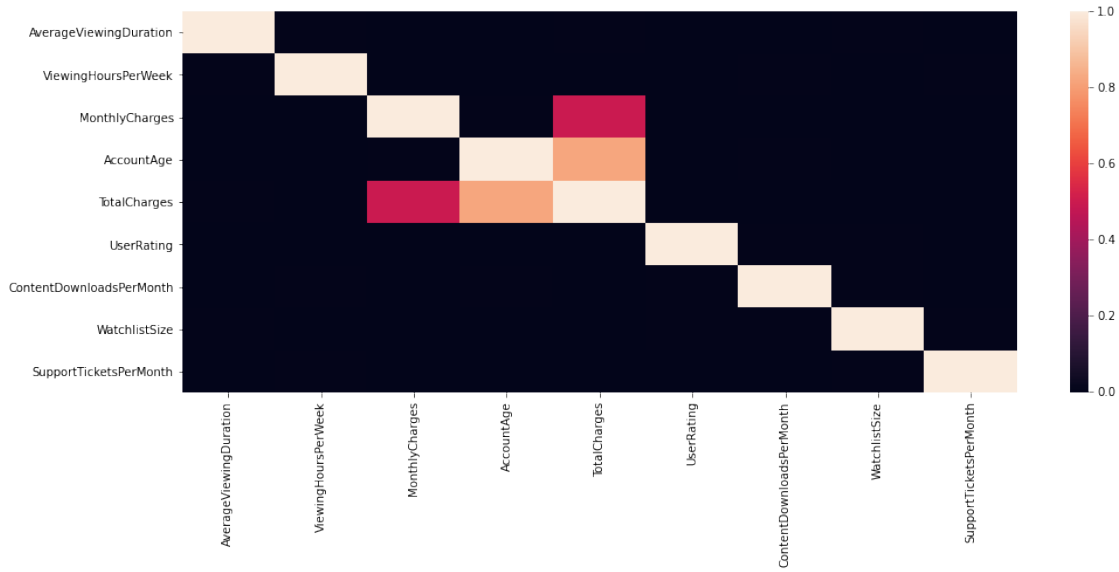
[23]:
```
feature_importances = pd.Series(model.feature_importances_, index=X_train.
 ↪columns).sort_values(ascending=False)

fig = px.bar(feature_importances)
fig.show()
```

There are clear differences between the important and non important variables. For the ones with importance > 0.04, check if there are any correlations between one another.

[24]:
```
plt.figure(figsize=(16, 6))
important_variables = feature_importances[0:9].index
sns.heatmap(train_df[important_variables].corr())
```

[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f76f90831d0>



The heatmap shows that there are correlations between "AccountAge" and "TotalCharges," and "MonthlyCharges" and "TotalCharges" which makes a lot of sense. Remove all the unimportant variables as well as "TotalCharges" from the model and run a new model.

[25]:
```
important_variables = important_variables.drop("TotalCharges")
X_train = train_df_new[important_variables]
y_train = train_df_new["Churn"]
model2 = RandomForestClassifier()
model2.fit(X_train, y_train)
```

[25]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=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=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

[26]:
```
feature_importances = pd.Series(model2.feature_importances_, index=X_train.
 ↪columns).sort_values(ascending=False)

fig = px.bar(feature_importances)
fig.show()
```

Now that we identify the variables to include, we will move on to hypertune the model by determining the best parameters

[27]:
```
# param_dist = {'n_estimators': randint(50,500),
#                'max_depth': randint(1,20)}

# rf = RandomForestClassifier()

# rand_search = RandomizedSearchCV(rf,
#                                  param_distributions = param_dist,
#                                  n_iter=5,
#                                  cv=5)

# rand_search.fit(X_train, y_train)
```

[28]:
```
# best_rf = rand_search.best_estimator_

# print('Best hyperparameters:',  rand_search.best_params_)
```

[29]:
```
best_rf = RandomForestClassifier(max_depth = 13, n_estimators = 285)
best_rf.fit(X_train, y_train)
```

[29]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=13, max_features='auto',
                       max_leaf_nodes=None, max_samples=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=285,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

Using the model with the best parameters, we will finally make predictions on the test data.

[30]:
```
X_test = test_df[important_variables]
y_pred = best_rf.predict(X_test)
prediction_df = pd.DataFrame({'CustomerID': test_df['CustomerID'], 'preds':␣
↪y_pred})
```

0.6 Post-process

[31]:
```
fig = px.histogram(prediction_df.preds)
fig.show()
```

[32]:
```
prediction_df.preds.value_counts()
```

[32]:
```
0    102111
1      2369
Name: preds, dtype: int64
```

[33]:
```
result = test_df
result['preds'] = prediction_df['preds']
```

Check how predicted churn is different for the variables with importance

[34]:
```
fig = px.box(x = result.preds, y = result.AverageViewingDuration)
fig.show()
```

You can see that the users who churned spend shorter amount of time watching a show/movie. Possible explanations are * they do not like the contents on this platform or do not have time to watch, resulting in giving up before finishing them * they prefer the contents of shorter duration (short films, an episode of a show)

[35]:
```
fig = px.box(x = result.preds, y = result.ViewingHoursPerWeek)
fig.show()
```

Similarly, the churned users spent significantly less time using the streaming service.

[36]:
```
fig = px.box(x = result.preds, y = result.MonthlyCharges)
fig.show()
```

The churn group pays higher subscription fee, which is a resonable incentive to cancel the plan.

[37]:
```
fig = px.box(x = result.preds, y = result.AccountAge)
fig.show()
```

The users in the churned group did not use the platform for a long period of time.

[38]:
```
fig = px.box(x = result.preds, y = result.ContentDownloadsPerMonth)
fig.show()
```

The users in the churned group do not take advantage of content downloading or might not be aware of the functionality, which explains their unsatisfaction.

[39]:
```
fig = px.box(x = result.preds, y = result.SupportTicketsPerMonth)
fig.show()
```

The number of support tickets are larger for the churned group. We can assume that they had trouble with the service or concern about their plan.

0.7 Conclusion & Next Step

In this project, Random Forest Classifer was used to predict the customer churn on subscription service. First, the exploratory data analysis showed the potential relationship with the variables and the churn. Then, the model was developed and fine turned. The model used average viewing duration, viewing hours per week, montly charges, account age, user rating, content downloads per month, watchlist size, and support tickets per month as the variables with high importance. Since the test data does not contain the correct predictions, unfortunately, we were unable to see the prediction accuracy and the model performance.

As the next step, we would like to test out different models, such as logistic regression, k-nearest neibors, decision tree classifier, and support vector machine, and compare the model performance.