Welcome to the Data Science Coding Challange!

Test your skills in a real-world coding challenge. Coding Challenges provide CS & DS Coding Competitions with Prizes and achievement badges!

CS & DS learners want to be challenged as a way to evaluate if they're job ready. So, why not create fun challenges and give winners something truly valuable such as complimentary access to select Data Science courses, or the ability to receive an achievement badge on their Coursera Skills Profile - highlighting their performance to recruiters.

Introduction

In this challenge, you'll get the opportunity to tackle one of the most industry-relevant maching learning problems with a unique dataset that will put your modeling skills to the test. 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 challenge, we will be tackling the churn prediction problem on a very unique and interesting group of subscribers on a video streaming service!

Imagine that you are a new data scientist at this video streaming company and you are tasked with building a model that can predict which existing subscribers will continue their subscriptions for another month. We have provided a dataset that is a sample of subscriptions that were initiated in 2021, all snapshotted at a particular date before the subscription was cancelled. Subscription cancellation can happen for a multitude of reasons, including:

- the customer completes all content they were interested in, and no longer need the subscription
- the customer finds themselves to be too busy and cancels their subscription until a later time
- the customer determines that the streaming service is not the best fit for them, so they cancel and look for something better suited

Regardless the reason, this video streaming company has a vested interest in understanding the likelihood of each individual customer to churn in their subscription so that resources can be allocated appropriately to support customers. In this challenge, you will use your machine learning toolkit to do just that!

Understanding the Datasets

Train vs. Test

In this competition, you'll gain access to two datasets that are samples of past subscriptions of a video streaming platform that contain information about the customer, the customers streaming preferences, and their activity in the subscription thus far. One dataset is titled `train.csv` and the other is titled `test.csv`.

`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), but does not disclose the "ground truth" for each subscription. It's your job to predict this outcome!

Using the patterns you find in the `train.csv` data, predict whether the subscriptions in `test.csv` will be continued for another month, or not.

Dataset descriptions

Both `train.csv` and `test.csv` contain one row for each unique subscription. For each subscription, a single observation (`CustomerID`) is included during which the subscription was active.

In addition to this identifier column, the `train.csv` dataset also contains the target label for the task, a binary column `Churn`.

Besides that column, both datasets have an identical set of features that can be used to train your model to make predictions. Below you can see descriptions of each feature. Familiarize yourself with them so that you can harness them most effectively for this machine learning task!

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

```
                 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).
```

How to Submit your Predictions to Coursera

Submission Format:

In this notebook you should follow the steps below to explore the data, train a model using the data in `train.csv`, and then score your model using the data in `test.csv`. Your final submission should be a dataframe (call it `prediction_df` with two columns and exactly 104,480 rows (plus a header row). The first column should be `CustomerID` so that we know which prediction belongs to which observation. The second column should be called `predicted_probability` and should be a numeric column representing the **likellihood that the subscription will churn**.

Your submission will show an error if you have extra columns (beyond `CustomerID` and `predicted_probability`) or extra rows. The order of the rows does not matter.

The naming convention of the dataframe and columns are critical for our autograding, so please make sure to use the exact naming conventions of `prediction_df` with column names `CustomerID` and `predicted_probability`!

To determine your final score, we will compare your `predicted_probability` predictions to the source of truth labels for the observations in `test.csv` and calculate the ROC AUC. We choose this metric because we not only want to be able to predict which subscriptions will be retained, but also want a well-calibrated likelihood score that can be used to target interventions and support most accurately.

Import Python Modules

First, import the primary modules that will be used in this project. Remember as this is an open-ended project please feel free to make use of any of your favorite libraries that you feel may be useful for this challenge. For example some of the following popular packages may be useful:

- pandas
- numpy
- Scipy
- Scikit-learn
- keras
- maplotlib

- seaborn
- etc, etc

```
# Import required packages

# Data packages
import pandas as pd
import numpy as np

# Machine Learning / Classification packages
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier

# Visualization Packages
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline

# Import any other packages you may want to use
```

Load the Data

Let's start by loading the dataset `train.csv` into a dataframe `train_df`, and `test.csv` into a dataframe `test_df` and display the shape of the dataframes.

```
train_df = pd.read_csv("train.csv")
print(train_df.info())
print(train_df.isnull().sum())
print(train_df.describe())
print('train_df Shape:', train_df.shape)
train_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243787 entries, 0 to 243786
Data columns (total 21 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   AccountAge                243787 non-null  int64
 1   MonthlyCharges            243787 non-null  float64
 2   TotalCharges              243787 non-null  float64
 3   SubscriptionType          243787 non-null  object
 4   PaymentMethod             243787 non-null  object
 5   PaperlessBilling          243787 non-null  object
 6   ContentType               243787 non-null  object
 7   MultiDeviceAccess         243787 non-null  object
 8   DeviceRegistered          243787 non-null  object
 9   ViewingHoursPerWeek       243787 non-null  float64
 10  AverageViewingDuration    243787 non-null  float64
 11  ContentDownloadsPerMonth  243787 non-null  int64
```

```
 12  GenrePreference           243787 non-null  object
 13  UserRating                243787 non-null  float64
 14  SupportTicketsPerMonth    243787 non-null  int64
 15  Gender                    243787 non-null  object
 16  WatchlistSize             243787 non-null  int64
 17  ParentalControl           243787 non-null  object
 18  SubtitlesEnabled          243787 non-null  object
 19  CustomerID                243787 non-null  object
 20  Churn                     243787 non-null  int64
dtypes: float64(5), int64(5), object(11)
memory usage: 39.1+ MB
None
AccountAge                  0
MonthlyCharges              0
TotalCharges                0
SubscriptionType            0
PaymentMethod               0
PaperlessBilling            0
ContentType                 0
MultiDeviceAccess           0
DeviceRegistered            0
ViewingHoursPerWeek         0
AverageViewingDuration      0
ContentDownloadsPerMonth    0
GenrePreference             0
UserRating                  0
SupportTicketsPerMonth      0
Gender                      0
WatchlistSize               0
ParentalControl             0
SubtitlesEnabled            0
CustomerID                  0
Churn                       0
dtype: int64
          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
train_df Shape: (243787, 21)

   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]
```

```
test_df = pd.read_csv("test.csv")
print('test_df Shape:', test_df.shape)
test_df.head()
```

```
test_df Shape: (104480, 20)

   AccountAge  MonthlyCharges  TotalCharges SubscriptionType  \
0          38       17.869374    679.036195          Premium
1          77        9.912854    763.289768            Basic
2           5       15.019011     75.095057         Standard
3          88       15.357406   1351.451692         Standard
4          91       12.406033   1128.949004         Standard

      PaymentMethod PaperlessBilling ContentType MultiDeviceAccess  \
0      Mailed check               No    TV Shows                No
1  Electronic check              Yes    TV Shows                No
2     Bank transfer               No    TV Shows               Yes
3  Electronic check               No        Both               Yes
4       Credit card              Yes    TV Shows               Yes
```

```
  DeviceRegistered  ViewingHoursPerWeek  AverageViewingDuration  \
0               TV            29.126308              122.274031
1               TV            36.873729               57.093319
2         Computer             7.601729              140.414001
3           Tablet            35.586430              177.002419
4           Tablet            23.503651               70.308376

   ContentDownloadsPerMonth GenrePreference  UserRating  \
0                        42          Comedy    3.522724
1                        43          Action    2.021545
2                        14          Sci-Fi    4.806126
3                        14          Comedy    4.943900
4                         6           Drama    2.846880

   SupportTicketsPerMonth  Gender  WatchlistSize ParentalControl  \
0                       2    Male             23              No
1                       2  Female             22             Yes
2                       2  Female             22              No
3                       0  Female             23             Yes
4                       6  Female              0              No

  SubtitlesEnabled  CustomerID
0               No  O1W6BHP6RM
1               No  LFR4X92X8H
2              Yes  QM5GBIYODA
3              Yes  D9RXTK2K9F
4               No  ENTCCHR1LR
```

Explore, Clean, Validate, and Visualize the Data (optional)

Feel free to explore, clean, validate, and visualize the data however you see fit for this competition to help determine or optimize your predictive model. Please note - the final autograding will only be on the accuracy of the `prediction_df` predictions.

```
# your code here (optional)
```

Make predictions (required)

Remember you should create a dataframe named `prediction_df` with exactly 104,480 entries plus a header row attempting to predict the likelihood of churn for subscriptions in `test_df`. Your submission will throw an error if you have extra columns (beyond `CustomerID` and `predicted_probaility`) or extra rows.

The file should have exactly 2 columns: `CustomerID` (sorted in any order) `predicted_probability` (contains your numeric predicted probabilities between 0 and 1, e.g. from `estimator.predict_proba(X, y)[:, 1]`)

The naming convention of the dataframe and columns are critical for our autograding, so please make sure to use the exact naming conventions of `prediction_df` with column names `CustomerID` and `predicted_probability`!

Example prediction submission:

The code below is a very naive prediction method that simply predicts churn using a Dummy Classifier. This is used as just an example showing the submission format required. Please change/alter/delete this code below and create your own improved prediction methods for generating `prediction_df`.

PLEASE CHANGE CODE BELOW TO IMPLEMENT YOUR OWN PREDICTIONS

```
### PLEASE CHANGE THIS CODE TO IMPLEMENT YOUR OWN PREDICTIONS

# Fit a dummy classifier on the feature columns in train_df:
dummy_clf = DummyClassifier(strategy="stratified")
dummy_clf.fit(train_df.drop(['CustomerID', 'Churn'], axis=1),
train_df.Churn)
```

```
DummyClassifier(constant=None, random_state=None,
strategy='stratified')
```

```
### PLEASE CHANGE THIS CODE TO IMPLEMENT YOUR OWN PREDICTIONS

# Use our dummy classifier to make predictions on test_df using
`predict_proba` method:
predicted_probability =
dummy_clf.predict_proba(test_df.drop(['CustomerID'], axis=1))[:, 1]

### PLEASE CHANGE THIS CODE TO IMPLEMENT YOUR OWN PREDICTIONS

# Combine predictions with label column into a dataframe
prediction_df = pd.DataFrame({'CustomerID':
test_df[['CustomerID']].values[:, 0],
                              'predicted_probability':
predicted_probability})

### PLEASE CHANGE THIS CODE TO IMPLEMENT YOUR OWN PREDICTIONS

# View our 'prediction_df' dataframe as required for submission.
# Ensure it should contain 104,480 rows and 2 columns 'CustomerID' and
'predicted_probaility'
print(prediction_df.shape)
prediction_df.head(10)
```

```
(104480, 2)
```

```
   CustomerID  predicted_probability
0  O1W6BHP6RM                    0.0
1  LFR4X92X8H                    0.0
```

```
2  QM5GBIYODA                    0.0
3  D9RXTK2K9F                    0.0
4  ENTCCHR1LR                    0.0
5  7A88BB5IO6                    0.0
6  70OMW9XEWR                    0.0
7  EL1RMFMPYL                    0.0
8  4IA2QPT6ZK                    0.0
9  AEDCWHSJDN                    0.0
```

PLEASE CHANGE CODE ABOVE TO IMPLEMENT YOUR OWN PREDICTIONS

Final Tests - **IMPORTANT** - the cells below must be run prior to submission

Below are some tests to ensure your submission is in the correct format for autograding. The autograding process accepts a csv `prediction_submission.csv` which we will generate from our `prediction_df` below. Please run the tests below an ensure no assertion errors are thrown.

```
# FINAL TEST CELLS - please make sure all of your code is above these
test cells

# Writing to csv for autograding purposes
prediction_df.to_csv("prediction_submission.csv", index=False)
submission = pd.read_csv("prediction_submission.csv")

assert isinstance(submission, pd.DataFrame), 'You should have a
dataframe named prediction_df.'

# FINAL TEST CELLS - please make sure all of your code is above these
test cells

assert submission.columns[0] == 'CustomerID', 'The first column name
should be CustomerID.'
assert submission.columns[1] == 'predicted_probability', 'The second
column name should be predicted_probability.'

# FINAL TEST CELLS - please make sure all of your code is above these
test cells

assert submission.shape[0] == 104480, 'The dataframe prediction_df
should have 104480 rows.'

# FINAL TEST CELLS - please make sure all of your code is above these
test cells

assert submission.shape[1] == 2, 'The dataframe prediction_df should
have 2 columns.'
```

```
# FINAL TEST CELLS - please make sure all of your code is above these test cells

## This cell calculates the auc score and is hidden. Submit Assignment to see AUC score.
```

SUBMIT YOUR WORK!

Once we are happy with our `prediction_df` and `prediction_submission.csv` we can now submit for autograding! Submit by using the blue **Submit Assignment** at the top of your notebook. Don't worry if your initial submission isn't perfect as you have multiple submission attempts and will obtain some feedback after each submission!