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

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!

In [13]:
```
import pandas as pd
data_descriptions = pd.read_csv('data_descriptions.csv')
pd.set_option('display.max_colwidth', None)
data_descriptions
```

Out[13]:

	Column_name	Column_type	Data_type	Description
0	AccountAge	Feature	integer	The age of the user's account in months.
1	MonthlyCharges	Feature	float	The amount charged to the user on a monthly basis.
2	TotalCharges	Feature	float	The total charges incurred by the user over the account's lifetime.
3	SubscriptionType	Feature	object	The type of subscription chosen by the user (Basic, Standard, or Premium).
4	PaymentMethod	Feature	string	The method of payment used by the user.
5	PaperlessBilling	Feature	string	Indicates whether the user has opted for paperless billing (Yes or No).
6	ContentType	Feature	string	The type of content preferred by the user (Movies, TV Shows, or Both).
7	MultiDeviceAccess	Feature	string	Indicates whether the user has access to the service on multiple devices (Yes or No).
8	DeviceRegistered	Feature	string	The type of device registered by the user (TV, Mobile, Tablet, or Computer).
9	ViewingHoursPerWeek	Feature	float	The number of hours the user spends watching content per week.
10	AverageViewingDuration	Feature	float	The average duration of each viewing session in minutes.
11	ContentDownloadsPerMonth	Feature	integer	The number of content downloads by the user per month.
12	GenrePreference	Feature	string	The preferred genre of content chosen by the user.
13	UserRating	Feature	float	The user's rating for the service on a scale of 1 to 5.
14	SupportTicketsPerMonth	Feature	integer	The number of support tickets raised by the user per month.
15	Gender	Feature	string	The gender of the user (Male or Female).
16	WatchlistSize	Feature	float	The number of items in the user's watchlist.
17	ParentalControl	Feature	string	Indicates whether parental control is enabled for the user (Yes or No).
18	SubtitlesEnabled	Feature	string	Indicates whether subtitles are enabled for the user (Yes or No).
19	CustomerID	Identifier	string	A unique identifier for each customer.
20	Churn	Target	integer	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 (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html). 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

In [14]:
```
# 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
```

In [30]:
```
# 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.

In [15]:
```
train_df = pd.read_csv("train.csv")
print('train_df Shape:', train_df.shape)
train_df.head()
```

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

Out[15]:

	AccountAge	MonthlyCharges	TotalCharges	SubscriptionType	PaymentMethod	PaperlessBilli
0	20	11.055215	221.104302	Premium	Mailed check	I
1	57	5.175208	294.986882	Basic	Credit card	Y
2	73	12.106657	883.785952	Basic	Mailed check	Y
3	32	7.263743	232.439774	Basic	Electronic check	I
4	57	16.953078	966.325422	Premium	Electronic check	Y

5 rows × 21 columns

In [16]:
```
test_df = pd.read_csv("test.csv")
print('test_df Shape:', test_df.shape)
test_df.head()
```

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

Out[16]:

	AccountAge	MonthlyCharges	TotalCharges	SubscriptionType	PaymentMethod	PaperlessBilli
0	38	17.869374	679.036195	Premium	Mailed check	I
1	77	9.912854	763.289768	Basic	Electronic check	Y
2	5	15.019011	75.095057	Standard	Bank transfer	I
3	88	15.357406	1351.451692	Standard	Electronic check	I
4	91	12.406033	1128.949004	Standard	Credit card	Y

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.

In [33]:
```
# 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

In [17]:
```
from sklearn.dummy import DummyClassifier

# Define your train_df here
# train_df = ...

# Instantiate and fit a dummy classifier
dummy_clf = DummyClassifier(strategy="stratified")
dummy_clf.fit(train_df.drop(['CustomerID', 'Churn'], axis=1), train_df['Churn'
```

Out[17]:
```
DummyClassifier(constant=None, random_state=None, strategy='stratified')
```

In [24]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Load the dataset
df = pd.read_csv('your_file.csv')

# Drop columns and separate features and target variable
X = df.drop(['CustomerID', 'Churn'], axis=1)
y = df['Churn']

# Handle missing values (if any)
X.fillna(X.mean(), inplace=True)

# Convert categorical columns to numerical (if any)
X = pd.get_dummies(X, drop_first=True)

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2

# Initialize and fit the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test data
predicted_probability = model.predict_proba(X_test)[:, 1]
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-24-b43af4bd9f46> in <module>
      5
      6 # Load the dataset
----> 7 df = pd.read_csv('your_file.csv')
      8
      9 # Drop columns and separate features and target variable

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad_lines, delim_whitespace, low_memory, memory_map, float_precision)
    674         )
    675
--> 676         return _read(filepath_or_buffer, kwds)
    677
    678     parser_f.__name__ = name

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    446
    447     # Create the parser.
--> 448     parser = TextFileReader(fp_or_buf, **kwds)
    449
    450     if chunksize or iterator:

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    878             self.options["has_index_names"] = kwds["has_index_names"]
    879
--> 880         self._make_engine(self.engine)
    881
    882     def close(self):

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in _make_engine(self, engine)
   1112     def _make_engine(self, engine="c"):
   1113         if engine == "c":
-> 1114             self._engine = CParserWrapper(self.f, **self.options)
   1115         else:
   1116             if engine == "python":

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   1889         kwds["usecols"] = self.usecols
   1890
-> 1891         self._reader = parsers.TextReader(src, **kwds)
   1892         self.unnamed_cols = self._reader.unnamed_cols
   1893

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()
```

```
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_sou
rce()

FileNotFoundError: [Errno 2] File your_file.csv does not exist: 'your_file.cs
v'
```

In [25]:
```
import pandas as pd

# Assume predicted_probability contains the predicted probabilities
# and test_df contains the CustomerID and other test data

# Create a DataFrame with CustomerID and predicted probabilities
prediction_df = pd.DataFrame({
    'CustomerID': test_df['CustomerID'].values,
    'predicted_probability': predicted_probability
})

# Display the resulting DataFrame
print(prediction_df.head())
```

```
   CustomerID  predicted_probability
0  O1W6BHP6RM                    0.0
1  LFR4X92X8H                    1.0
2  QM5GBIYODA                    1.0
3  D9RXTK2K9F                    1.0
4  ENTCCHR1LR                    0.0
```

In [26]:
```
import pandas as pd

# Ensure 'predicted_probability' and 'test_df' are already defined
# 'predicted_probability' should have 104,480 entries corresponding to 'test_d

# Create the DataFrame with 'CustomerID' and 'predicted_probability'
prediction_df = pd.DataFrame({
    'CustomerID': test_df['CustomerID'].values,
    'predicted_probability': predicted_probability
})

# Check the shape of the DataFrame to ensure it has 104,480 rows and 2 columns
print(prediction_df.shape)

# Display the first 10 rows of the DataFrame
prediction_df.head(10)
```

```
(104480, 2)
```

Out[26]:

	CustomerID	predicted_probability
0	O1W6BHP6RM	0.0
1	LFR4X92X8H	1.0
2	QM5GBIYODA	1.0
3	D9RXTK2K9F	1.0
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.

In [34]:
```
# FINAL TEST CELLS - please make sure all of your code is above these test cel

# 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 name
```

In [35]:
```
# FINAL TEST CELLS - please make sure all of your code is above these test cel

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

In [36]:
```
# FINAL TEST CELLS - please make sure all of your code is above these test cel

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

In [14]:
```
# FINAL TEST CELLS - please make sure all of your code is above these test cel

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

In [15]:
```
# FINAL TEST CELLS - please make sure all of your code is above these test cel

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

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!