ChurnPrediction

September 10, 2024

1 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.

1.1 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!

1.2 Understanding the Datasets

1.2.1 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.

1.2.2 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!

[1]:
```
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    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).
```

1.3 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.

1.4 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

[2]:
```
# 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

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score

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

1.5 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.

[3]:
```
train_df = pd.read_csv("train.csv")
print('train_df Shape:', train_df.shape)
train_df.head()
```

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

[3]:

	AccountAge	MonthlyCharges	TotalCharges	SubscriptionType	PaymentMethod	PaperlessBilling	ContentType	MultiDeviceAccess	DeviceRegistered	ViewingHoursPerWeek	...	ContentDownloadsPerMonth	GenrePreference	UserRating	SupportTicketsPerMonth	Gender	WatchlistSize	ParentalControl	SubtitlesEnabled	CustomerID	Churn
0	20	11.055215	221.104302	Premium	Mailed check	No	Both	No	Mobile	36.758104	...	10	Sci-Fi	2.176498	4	Male	3	No	No	CB6SXPNVZA	0
1	57	5.175208	294.986882	Basic	Credit card	Yes	Movies	No	Tablet	32.450568	...	18	Action	3.478632	8	Male	23	No	Yes	S7R2G87O09	0
2	73	12.106657	883.785952	Basic	Mailed check	Yes	Movies	No	Computer	7.395160	...	23	Fantasy	4.238824	6	Male	1	Yes	Yes	EASDC20BDT	0
3	32	7.263743	232.439774	Basic	Electronic check	No	TV Shows	No	Tablet	27.960389	...	30	Drama	4.276013	2	Male	24	Yes	Yes	NPF69NT69N	0
4	57	16.953078	966.325422	Premium	Electronic check	Yes	TV Shows	No	TV	20.083397	...	20	Comedy	3.616170	4	Female	0	No	No	4LGYPK7VOL	0

```
[5 rows x 21 columns]
```

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

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

[4]:

	AccountAge	MonthlyCharges	TotalCharges	SubscriptionType	PaymentMethod	PaperlessBilling	ContentType	MultiDeviceAccess	DeviceRegistered	ViewingHoursPerWeek	AverageViewingDuration	ContentDownloadsPerMonth	GenrePreference	UserRating	SupportTicketsPerMonth	Gender	WatchlistSize	ParentalControl	SubtitlesEnabled	CustomerID
0	38	17.869374	679.036195	Premium	Mailed check	No	TV Shows	No	TV	29.126308	122.274031	42	Comedy	3.522724	2	Male	23	No	No	O1W6BHP6RM
1	77	9.912854	763.289768	Basic	Electronic check	Yes	TV Shows	No	TV	36.873729	57.093319	43	Action	2.021545	2	Female	22	Yes	No	LFR4X92X8H
2	5	15.019011	75.095057	Standard	Bank transfer	No	TV Shows	Yes	Computer	7.601729	140.414001	14	Sci-Fi	4.806126	2	Female	22	No	Yes	QM5GBIYODA
3	88	15.357406	1351.451692	Standard	Electronic check	No	Both	Yes	Tablet	35.586430	177.002419	14	Comedy	4.943900	0	Female	23	Yes	Yes	D9RXTK2K9F
4	91	12.406033	1128.949004	Standard	Credit card	Yes	TV Shows	Yes	Tablet	23.503651	70.308376	6	Drama	2.846880	6	Female	0	No	No	ENTCCHR1LR

1.6 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.

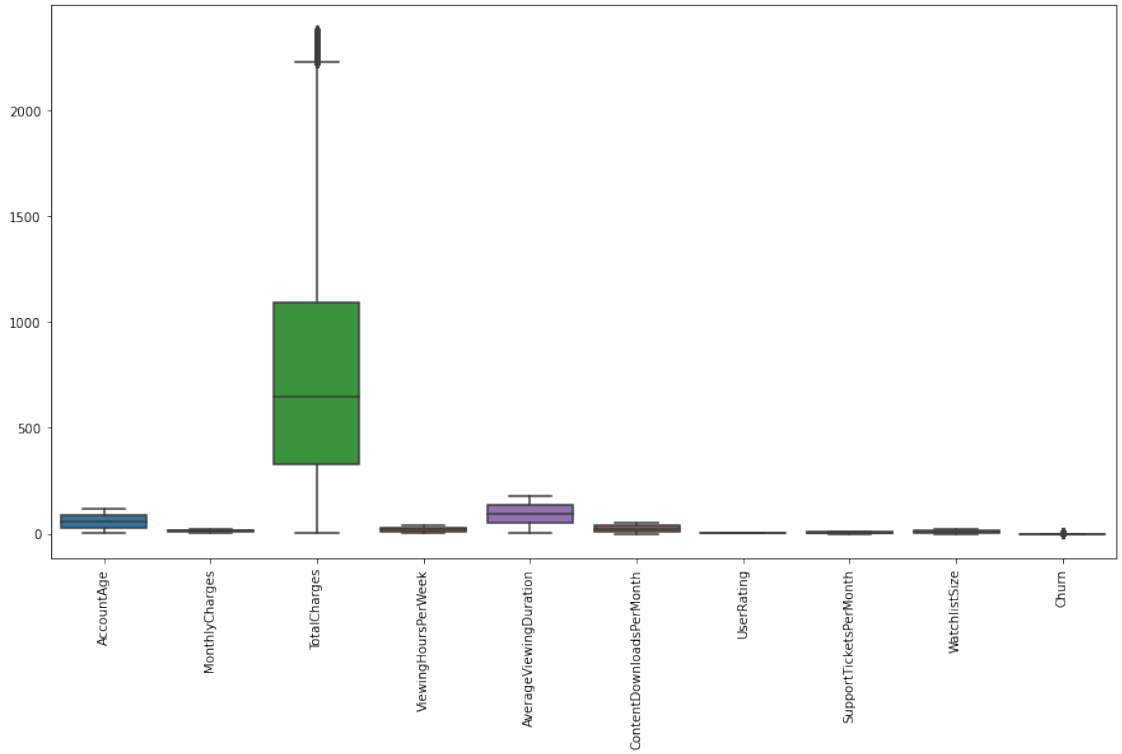
[5]:
```
# Remove duplicates if any
train_df.drop_duplicates(inplace=True)

# Handle missing values: for example, filling missing numeric values with mean
train_df.fillna(train_df.mean(), inplace=True)
```

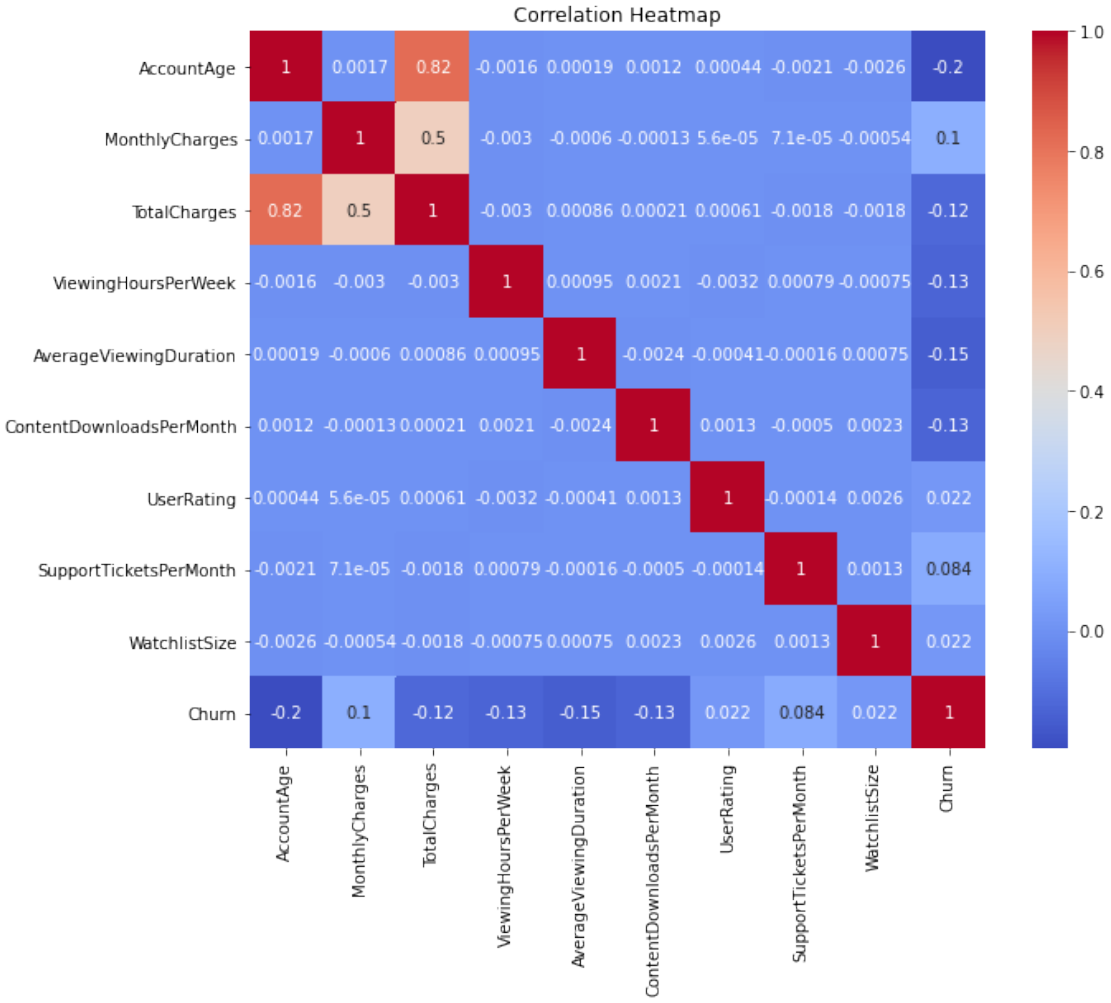
[6]:
```
# Check the distribution of the target variable
print(train_df['Churn'].value_counts(normalize=True))

# Boxplot for numeric features
numeric_columns = train_df.select_dtypes(include=['float64', 'int64']).columns
plt.figure(figsize=(15, 8))
sns.boxplot(data=train_df[numeric_columns])
plt.xticks(rotation=90)
plt.show()
```

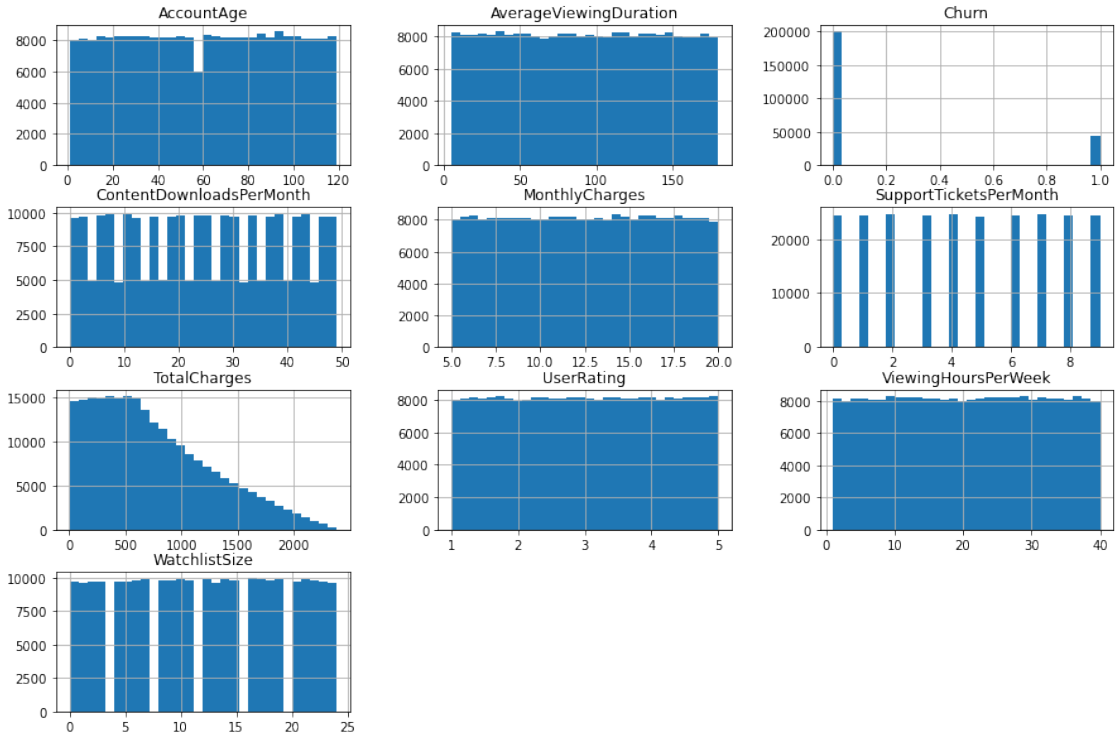
```
0    0.818768
1    0.181232
Name: Churn, dtype: float64
```



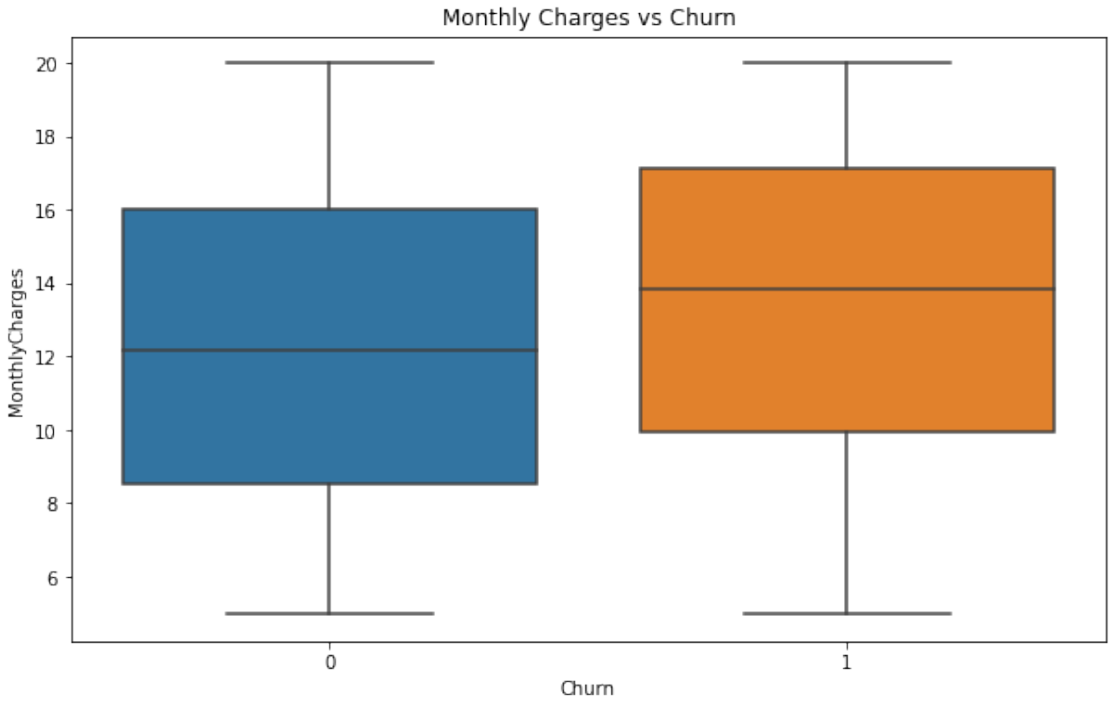
[7]:
```
# Heatmap to see correlations between features
plt.figure(figsize=(10, 8))
sns.heatmap(train_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



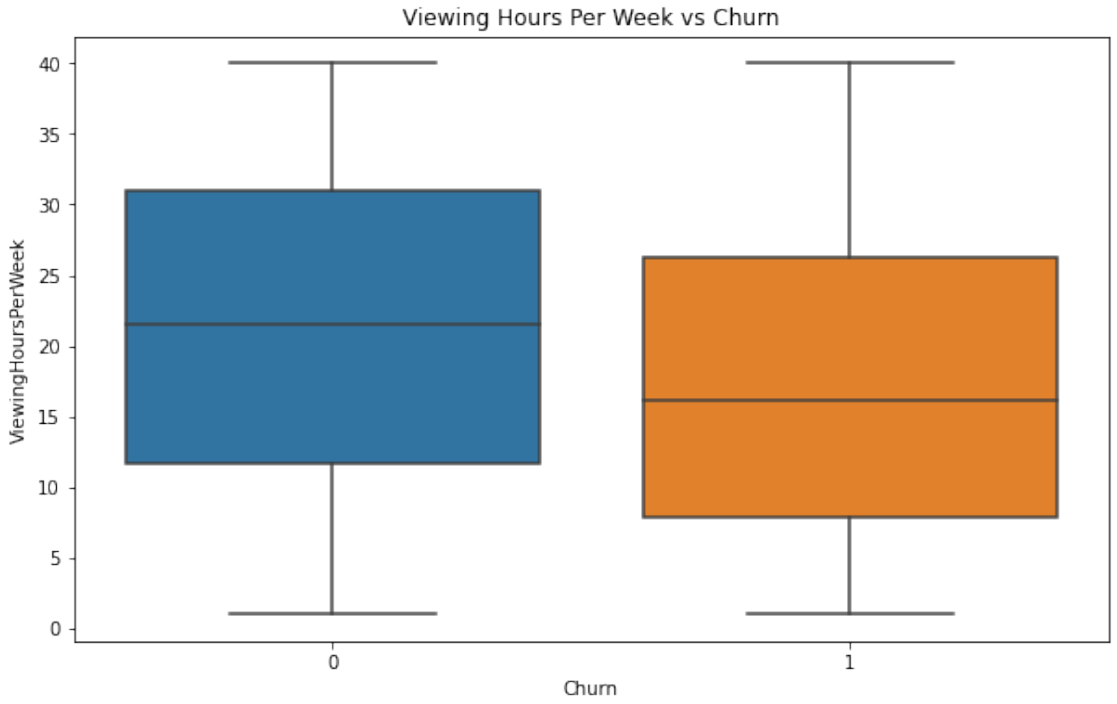
[8]:
```
# Visualize the distribution of numeric columns
train_df[numeric_columns].hist(bins=30, figsize=(15, 10))
plt.show()
```



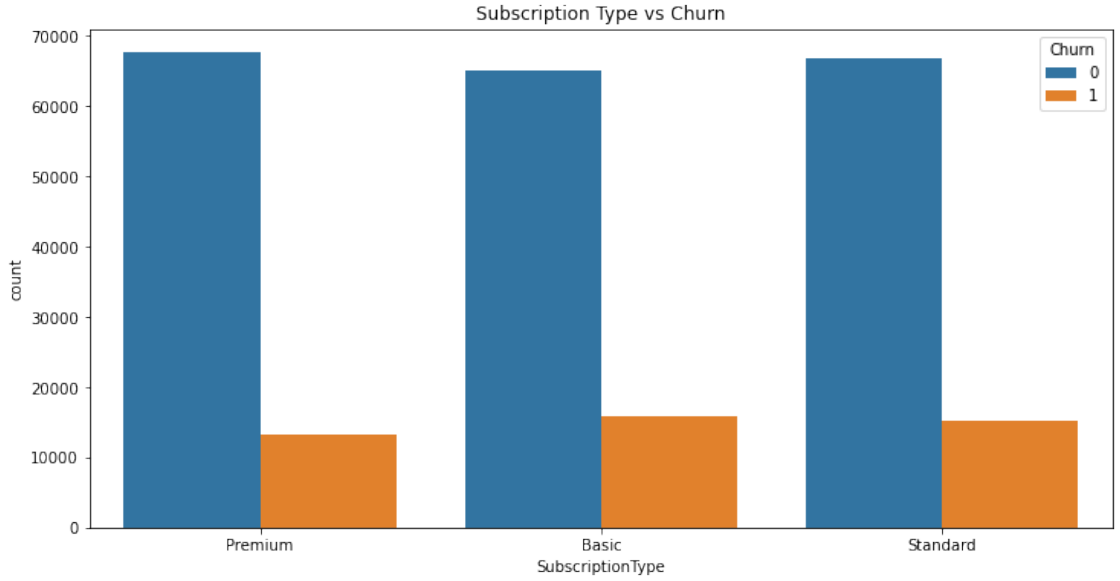
[9]:
```
# Boxplot to visualize how MonthlyCharges are distributed by Churn
plt.figure(figsize=(10, 6))
sns.boxplot(x='Churn', y='MonthlyCharges', data=train_df)
plt.title('Monthly Charges vs Churn')
plt.show()
```



[10]:
```
# Boxplot to visualize how ViewingHoursPerWeek are distributed by Churn
plt.figure(figsize=(10, 6))
sns.boxplot(x='Churn', y='ViewingHoursPerWeek', data=train_df)
plt.title('Viewing Hours Per Week vs Churn')
plt.show()
```



[11]:
```
# Check distribution of categorical columns (e.g., SubscriptionType) based on␣
↪Churn
plt.figure(figsize=(12, 6))
sns.countplot(x='SubscriptionType', hue='Churn', data=train_df)
plt.title('Subscription Type vs Churn')
plt.show()
```



1.7 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`!

1.7.1 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

[12]:
```
# Define features and target
X = train_df.drop(columns=['Churn', 'CustomerID'])
y = train_df['Churn']
X_test = test_df.drop(columns=['CustomerID'])
customer_ids = test_df['CustomerID']
```

[13]:
```
# Preprocess data
numeric_features = ['AccountAge', 'MonthlyCharges', 'TotalCharges',␣
 ↪'ViewingHoursPerWeek', 'AverageViewingDuration', 'ContentDownloadsPerMonth',␣
 ↪'UserRating', 'SupportTicketsPerMonth', 'WatchlistSize']
categorical_features = ['SubscriptionType', 'PaymentMethod', 'PaperlessBilling',␣
 ↪'ContentType', 'MultiDeviceAccess', 'DeviceRegistered', 'GenrePreference',␣
 ↪'Gender', 'ParentalControl', 'SubtitlesEnabled']

# Create preprocessing pipelines for numeric and categorical features
numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

# Combine preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
```

```
        ('cat', categorical_transformer, categorical_features)
    ])
```

[14]:
```
# Create and train the model pipeline
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier', LogisticRegression(max_iter=1000))])
```

[15]:
```
# Split the data
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train the model
model.fit(X_train, y_train)
```

[15]:
```
Pipeline(memory=None,
         steps=[('preprocessor',
                 ColumnTransformer(n_jobs=None, remainder='drop',
                                   sparse_threshold=0.3,
                                   transformer_weights=None,
                                   transformers=[('num',
                                                  StandardScaler(copy=True,
                                                                 with_mean=True,
                                                                 with_std=True),
                                                  ['AccountAge',
                                                   'MonthlyCharges',
                                                   'TotalCharges',
                                                   'ViewingHoursPerWeek',
                                                   'AverageViewingDuration',
                                                   'ContentDownloadsPerMonth',
                                                   'UserRating',
                                                   'Sup...
                                                   'GenrePreference', 'Gender',
                                                   'ParentalControl',
                                                   'SubtitlesEnabled'])],
                                   verbose=False)),
                ('classifier',
                 LogisticRegression(C=1.0, class_weight=None, dual=False,
                                    fit_intercept=True, intercept_scaling=1,
                                    l1_ratio=None, max_iter=1000,
                                    multi_class='auto', n_jobs=None,
                                    penalty='l2', random_state=None,
                                    solver='lbfgs', tol=0.0001, verbose=0,
                                    warm_start=False))],
         verbose=False)
```

[18]:
```
# Predict probabilities on the test data
```

```
X_test_scaled = model.named_steps['preprocessor'].transform(X_test)  # Use␣
 ↪preprocessor to transform test data
predicted_probabilities = model.predict_proba(X_test)[:, 1]

# Create the prediction dataframe
prediction_df = pd.DataFrame({
    'CustomerID': customer_ids,
    'predicted_probability': predicted_probabilities
})
```

[20]:
```
# Validate the model
y_valid_probs = model.predict_proba(X_valid)[:, 1]
# Calculate the ROC AUC score
roc_auc = roc_auc_score(y_valid, y_valid_probs)
print(f'ROC AUC Score: {roc_auc}')
```

```
ROC AUC Score: 0.7536603799535646
```

[21]:
```
print(prediction_df.shape)
print(prediction_df.head(10))
```

```
(104480, 2)
   CustomerID  predicted_probability
0  O1W6BHP6RM               0.105559
1  LFR4X92X8H               0.041280
2  QM5GBIYODA               0.399395
3  D9RXTK2K9F               0.044968
4  ENTCCHR1LR               0.135402
5  7A88BB5IO6               0.470202
6  7OOMW9XEWR               0.118827
7  EL1RMFMPYL               0.291825
8  4IA2QPT6ZK               0.204686
9  AEDCWHSJDN               0.182445
```

1.8 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.

[22]:
```
# 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.'
```

[23]:
```
# 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.'
```

[24]:
```
# 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.'
```

[25]:
```
# 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.'
```

1.9 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!