# Supervised Learning Capstone Project - Tree Methods Focus

will add more non-tree-based models e.g. logistic regression later...

## GOAL: Create models to predict whether or not a customer will Churn

## Part1: Imports and Read in the Data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('../DATA/Telco-Customer-Churn.csv')
df.head()
```

Out[2]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines |
|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No |

5 rows × 21 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7032 entries, 0 to 7031
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7032 non-null   object
 1   gender            7032 non-null   object
 2   SeniorCitizen     7032 non-null   int64
 3   Partner           7032 non-null   object
 4   Dependents        7032 non-null   object
 5   tenure            7032 non-null   int64
 6   PhoneService      7032 non-null   object
 7   MultipleLines     7032 non-null   object
 8   InternetService   7032 non-null   object
 9   OnlineSecurity    7032 non-null   object
 10  OnlineBackup      7032 non-null   object
 11  DeviceProtection  7032 non-null   object
 12  TechSupport       7032 non-null   object
 13  StreamingTV       7032 non-null   object
 14  StreamingMovies   7032 non-null   object
 15  Contract          7032 non-null   object
 16  PaperlessBilling  7032 non-null   object
 17  PaymentMethod     7032 non-null   object
 18  MonthlyCharges    7032 non-null   float64
 19  TotalCharges      7032 non-null   float64
 20  Churn             7032 non-null   object
dtypes: float64(2), int64(2), object(17)
memory usage: 1.1+ MB
```

In [4]:
```
df.describe()
```

Out[4]:

| | SeniorCitizen | tenure | MonthlyCharges | TotalCharges |
|---|---|---|---|---|
| **count** | 7032.000000 | 7032.000000 | 7032.000000 | 7032.000000 |
| **mean** | 0.162400 | 32.421786 | 64.798208 | 2283.300441 |
| **std** | 0.368844 | 24.545260 | 30.085974 | 2266.771362 |
| **min** | 0.000000 | 1.000000 | 18.250000 | 18.800000 |
| **25%** | 0.000000 | 9.000000 | 35.587500 | 401.450000 |
| **50%** | 0.000000 | 29.000000 | 70.350000 | 1397.475000 |
| **75%** | 0.000000 | 55.000000 | 89.862500 | 3794.737500 |
| **max** | 1.000000 | 72.000000 | 118.750000 | 8684.800000 |

In [5]:
```
df.isna().sum()
```

Out[5]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

# Summary:

**The dataset contains 21 columns and 7032 rows, most columns are dummy variables**

**There is no missing values**

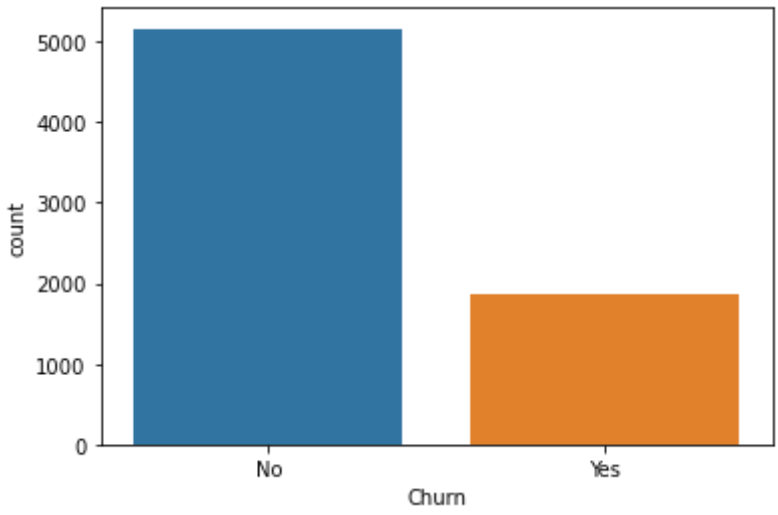
# Part 2: Exploratory Data Analysis

## General Feature Exploration

**Count plot for variable (churn)**

In [6]:
```
sns.countplot(data=df,x='Churn')
```

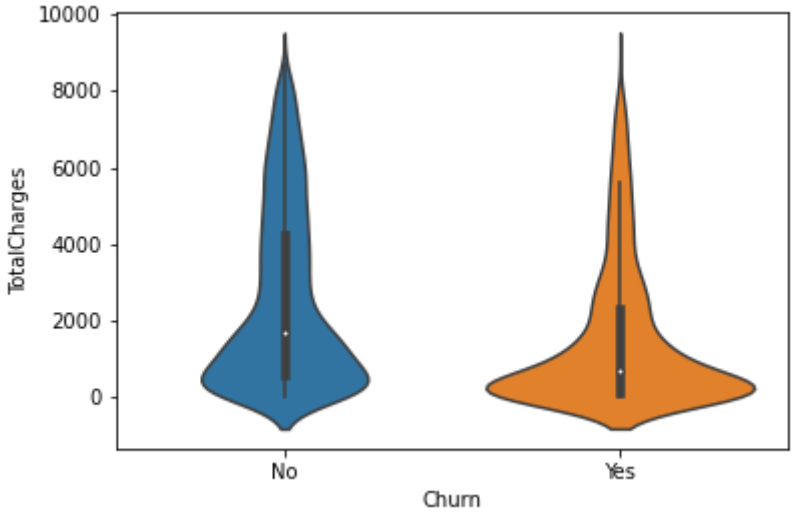
Out[6]:
```
<AxesSubplot:xlabel='Churn', ylabel='count'>
```




**The distrbution of TotalCharges between Churn categories with a violin plot**

In [7]:
```
# showing total charge distribution by YES or NO
sns.violinplot(data=df,x='Churn',y='TotalCharges')
```

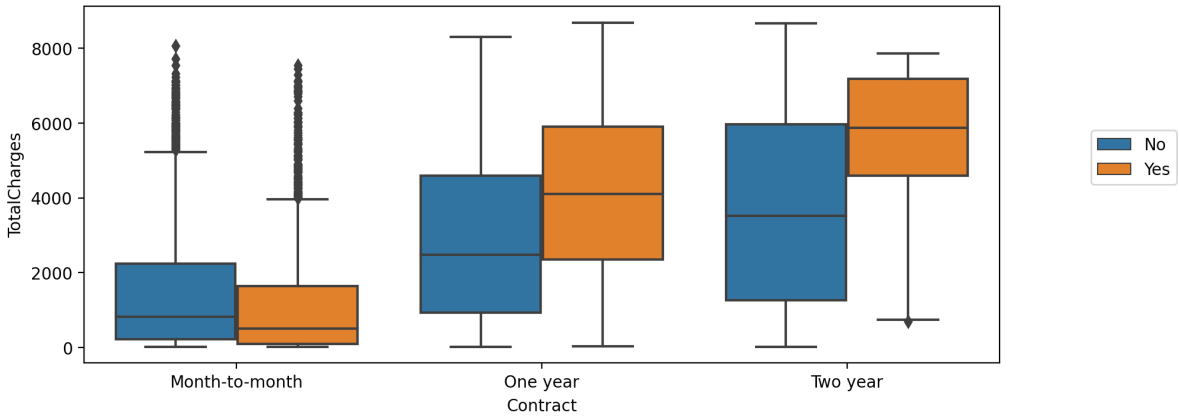
Out[7]:
```
<AxesSubplot:xlabel='Churn', ylabel='TotalCharges'>
```




**The distribution boxplot of TotalCharges per Contract type, hue color of churn class**

In [8]:
```
plt.figure(figsize=(10,4),dpi=200)
sns.boxplot(data=df,y='TotalCharges',x='Contract',hue='Churn')
plt.legend(loc=(1.1,0.5))
```

Out[8]:
```
<matplotlib.legend.Legend at 0x1f2db932130>
```




**Features to the class lable**

In [9]:
```
df.columns
```

Out[9]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

**Dummy variable correlation with the label**

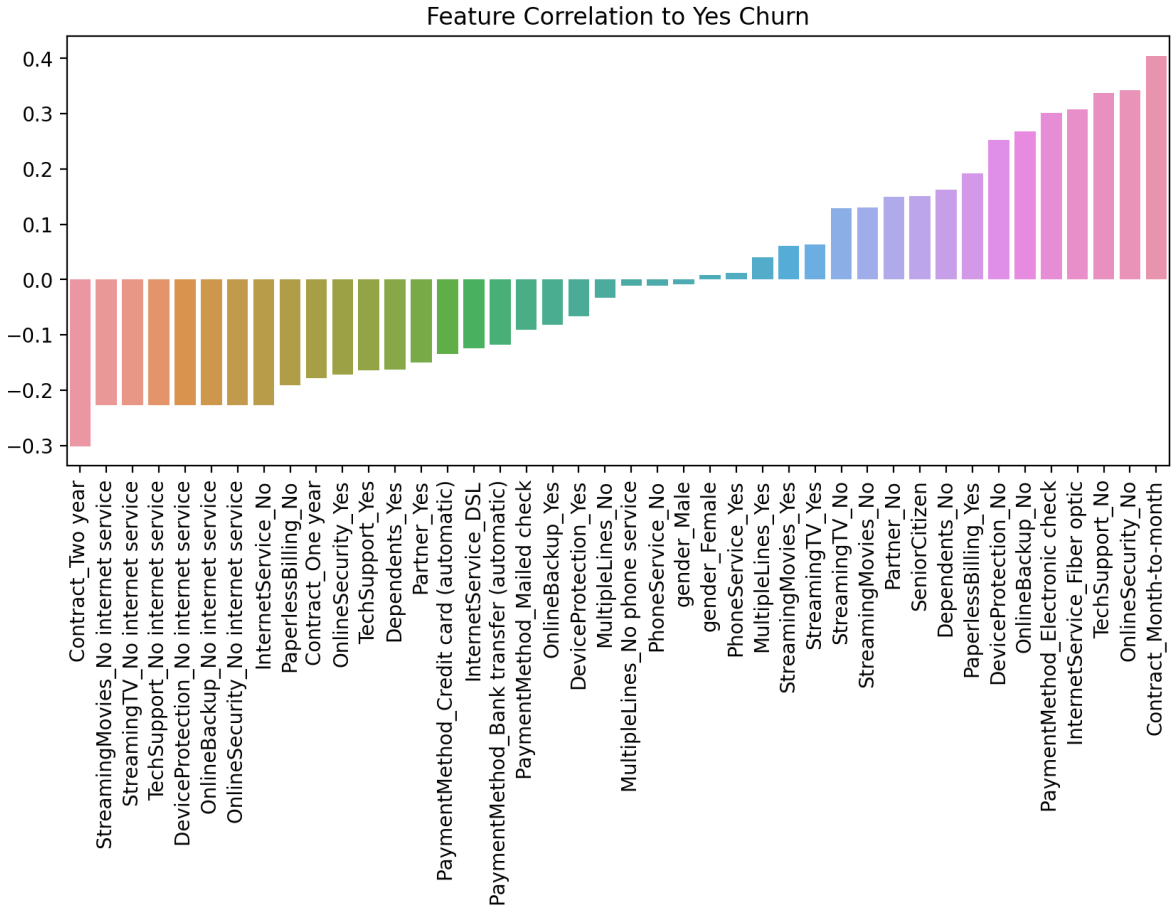
In [10]:
```
corr_df   = pd.get_dummies(df[['gender', 'SeniorCitizen', 'Partner', 'Dependents','I
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport','Streami
       'PaymentMethod','Churn']]).corr()
```

In [11]:
```
corr_df['Churn_Yes'].sort_values().iloc[1:-1]
```

Out[11]:
```
Contract_Two year                          -0.301552
StreamingMovies_No internet service        -0.227578
StreamingTV_No internet service            -0.227578
TechSupport_No internet service            -0.227578
DeviceProtection_No internet service       -0.227578
OnlineBackup_No internet service           -0.227578
OnlineSecurity_No internet service         -0.227578
InternetService_No                         -0.227578
PaperlessBilling_No                        -0.191454
Contract_One year                          -0.178225
OnlineSecurity_Yes                         -0.171270
TechSupport_Yes                            -0.164716
Dependents_Yes                             -0.163128
Partner_Yes                                -0.149982
PaymentMethod_Credit card (automatic)      -0.134687
InternetService_DSL                        -0.124141
PaymentMethod_Bank transfer (automatic)    -0.118136
PaymentMethod_Mailed check                 -0.090773
OnlineBackup_Yes                           -0.082307
DeviceProtection_Yes                       -0.066193
MultipleLines_No                           -0.032654
MultipleLines_No phone service             -0.011691
PhoneService_No                            -0.011691
gender_Male                                -0.008545
gender_Female                               0.008545
PhoneService_Yes                            0.011691
MultipleLines_Yes                           0.040033
StreamingMovies_Yes                         0.060860
StreamingTV_Yes                             0.063254
StreamingTV_No                              0.128435
StreamingMovies_No                          0.130920
Partner_No                                  0.149982
SeniorCitizen                               0.150541
Dependents_No                               0.163128
PaperlessBilling_Yes                        0.191454
DeviceProtection_No                         0.252056
OnlineBackup_No                             0.267595
PaymentMethod_Electronic check              0.301455
InternetService_Fiber optic                 0.307463
TechSupport_No                              0.336877
OnlineSecurity_No                           0.342235
Contract_Month-to-month                     0.404565
Name: Churn_Yes, dtype: float64
```

### The correlation plot

In [12]:
```
plt.figure(figsize=(10,4),dpi=200)
sns.barplot(x=corr_df['Churn_Yes'].sort_values().iloc[1:-1].index,y=corr_df['Churn_
plt.title("Feature Correlation to Yes Churn")
plt.xticks(rotation=90);
```




# Part 3: Churn Analysis

**segementing customers based on their tenure**

**Contract type variable**

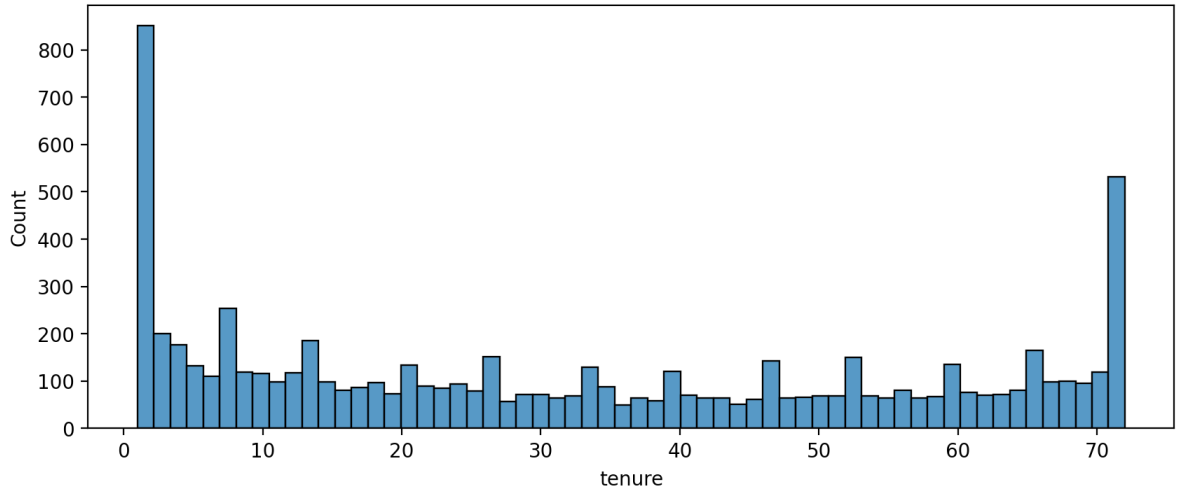
In [13]:
```
df['Contract'].unique()
```

Out[13]:
```
array(['Month-to-month', 'One year', 'Two year'], dtype=object)
```

**The histogram displaying the distribution of 'tenure' column, which is the amount of months a customer was or has been on a customer**

In [14]:
```
plt.figure(figsize=(10,4),dpi=200)
sns.histplot(data=df,x='tenure',bins=60)
```

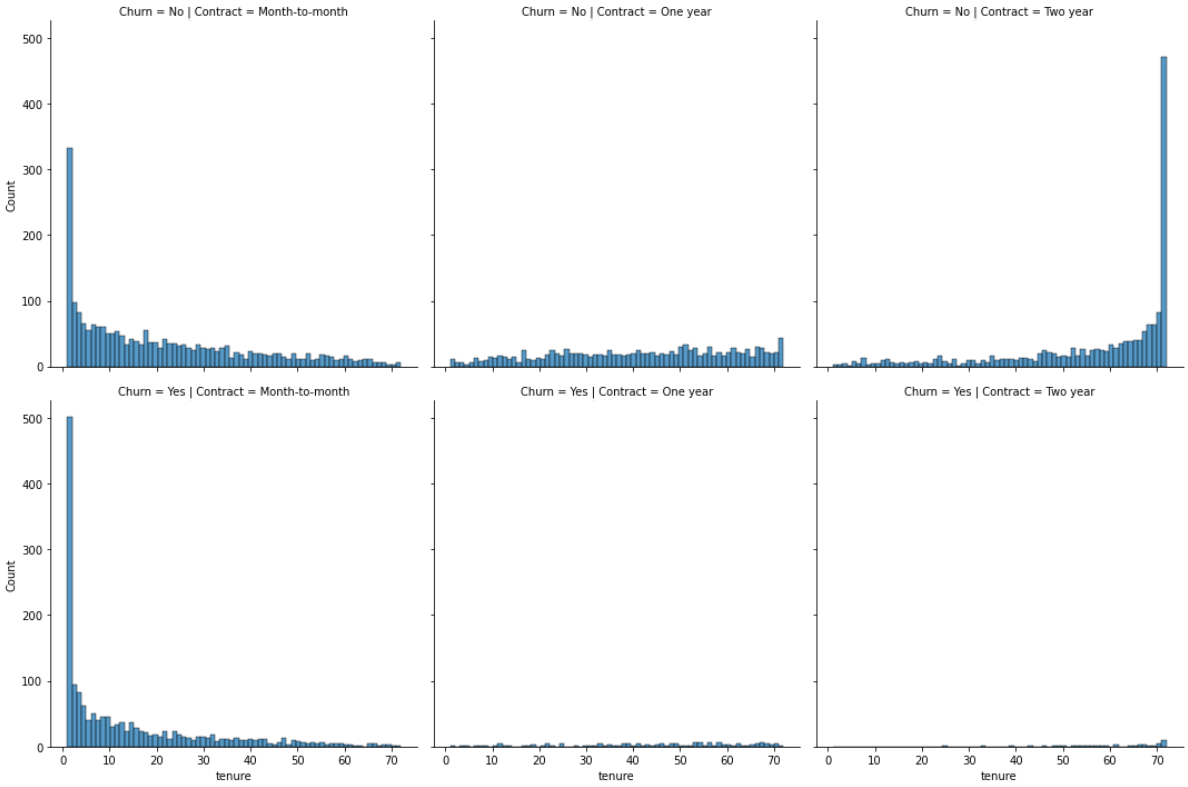
Out[14]:
```
<AxesSubplot:xlabel='tenure', ylabel='Count'>
```




**The histograms separated by two additional features, Churn and Contract**

In [15]:
```
plt.figure(figsize=(10,3),dpi=200)
sns.displot(data=df,x='tenure',bins=70,col='Contract',row='Churn');
```

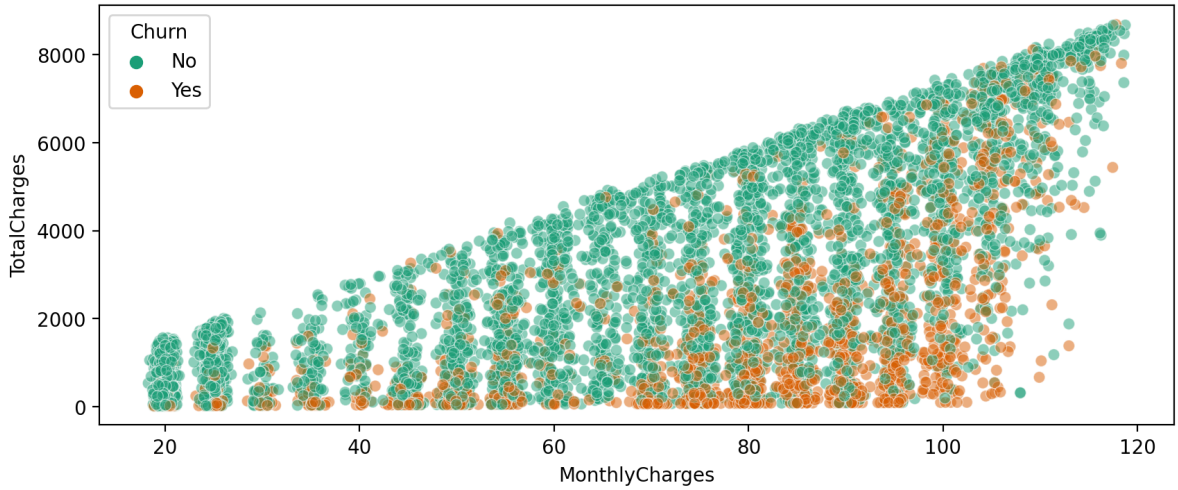
```
<Figure size 2000x600 with 0 Axes>
```




**The scatter plot of Total Charges versus Monthly Charges, and color hue by Churn**

In [16]:
```
plt.figure(figsize=(10,4),dpi=200)
sns.scatterplot(data=df,x='MonthlyCharges',y='TotalCharges',hue='Churn', linewidth
```

Out[16]:
```
<AxesSubplot:xlabel='MonthlyCharges', ylabel='TotalCharges'>
```




### The Cohorts based on Tenure

### Treating 1 month, 2 month, 3 month...N months as unit cohort, calculate the Churn rate (percentage that had Yes Churn) per cohort

In [17]:
```
no_churn = df.groupby(['Churn','tenure']).count().transpose()['No']
yes_churn = df.groupby(['Churn','tenure']).count().transpose()['Yes']
```

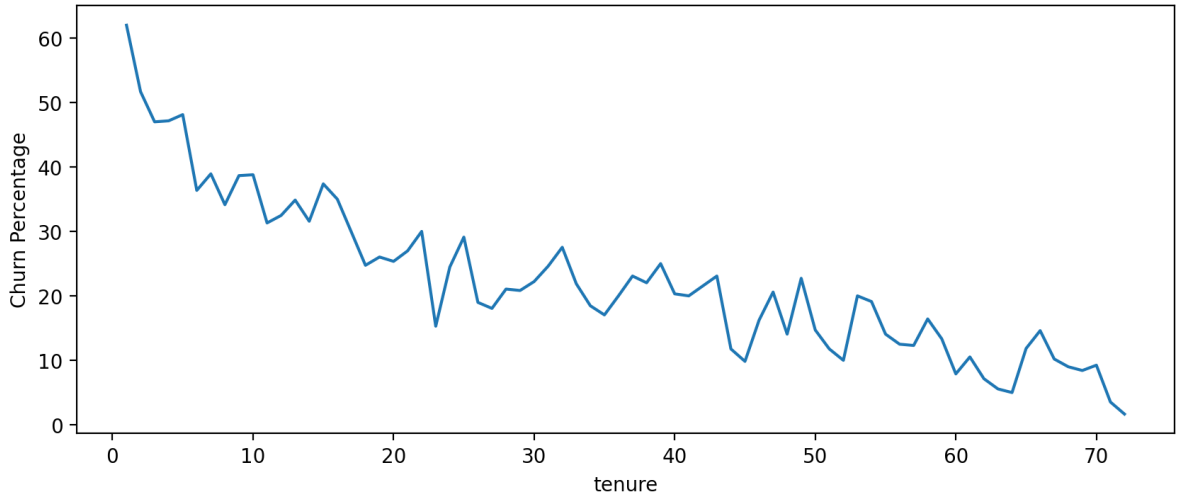
In [18]:
```
churn_rate = 100 * yes_churn / (no_churn+yes_churn)
```

In [19]:
```
churn_rate.transpose()['customerID']
```

Out[19]:
```
tenure
1     61.990212
2     51.680672
3     47.000000
4     47.159091
5     48.120301
        ...
68     9.000000
69     8.421053
70     9.243697
71     3.529412
72     1.657459
Name: customerID, Length: 72, dtype: float64
```

### The plot showing churn rate per months of tenure

In [20]:
```
plt.figure(figsize=(10,4),dpi=200)
churn_rate.iloc[0].plot()
plt.ylabel('Churn Percentage');
```




## Broader Cohort Groups

**Based on the tenure column values, create a new column called Tenure Cohort that creates 4 separate categories:**

- '0-12 Months'
- '24-48 Months'
- '12-24 Months'
- 'Over 48 Months'

In [21]:
```
def cohort(tenure):
    if tenure < 13:
        return '0-12 Months'
    elif tenure < 25:
        return '12-24 Months'
    elif tenure < 49:
        return '24-48 Months'
    else:
        return "Over 48 Months"
```

In [22]:
```
df['Tenure Cohort'] = df['tenure'].apply(cohort)
```

In [23]:
```
df.head(10)[['tenure','Tenure Cohort']]
```

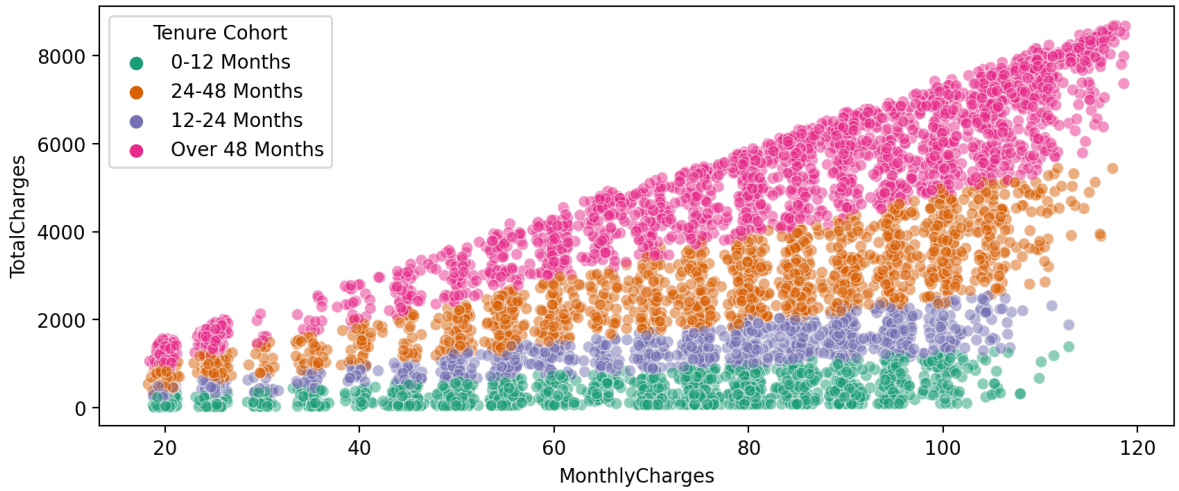
Out[23]:

| | tenure | Tenure Cohort |
|---|---|---|
| **0** | 1 | 0-12 Months |
| **1** | 34 | 24-48 Months |
| **2** | 2 | 0-12 Months |
| **3** | 45 | 24-48 Months |
| **4** | 2 | 0-12 Months |
| **5** | 8 | 0-12 Months |
| **6** | 22 | 12-24 Months |
| **7** | 10 | 0-12 Months |
| **8** | 28 | 24-48 Months |
| **9** | 62 | Over 48 Months |

### The scatterplot of Total Charges versus Monthly Charts

In [24]:
```
plt.figure(figsize=(10,4),dpi=200)
sns.scatterplot(data=df,x='MonthlyCharges',y='TotalCharges',hue='Tenure Cohort', l
```

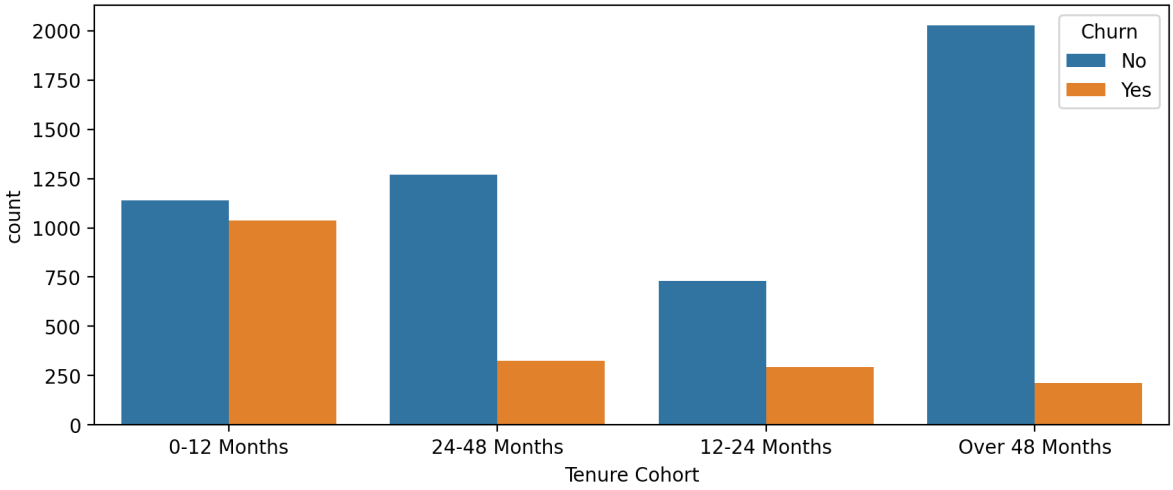
Out[24]:
```
<AxesSubplot:xlabel='MonthlyCharges', ylabel='TotalCharges'>
```




### The count plot showing the churn count per cohort

In [25]:
```
plt.figure(figsize=(10,4),dpi=200)
sns.countplot(data=df,x='Tenure Cohort',hue='Churn')
```

Out[25]:
```
<AxesSubplot:xlabel='Tenure Cohort', ylabel='count'>
```




# Part 4: Predictive Modeling

**A Single Decision Tree, Random Forest, AdaBoost, Gradient Boosting**

## Single Decision Tree

In [26]:
```
X = df.drop(['Churn','customerID'],axis=1)
X = pd.get_dummies(X,drop_first=True)
```

In [27]:
```
y = df['Churn']
```

**train_test_split**

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_sta
```

In [30]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [31]:
```
dt = DecisionTreeClassifier(max_depth=6)
```

In [32]:
```
dt.fit(X_train,y_train)
```

Out[32]:
```
DecisionTreeClassifier(max_depth=6)
```

In [33]:
```
preds = dt.predict(X_test)
```

In [34]:
```
from sklearn.metrics import accuracy_score,plot_confusion_matrix,classification_rep
```

**The classification report**

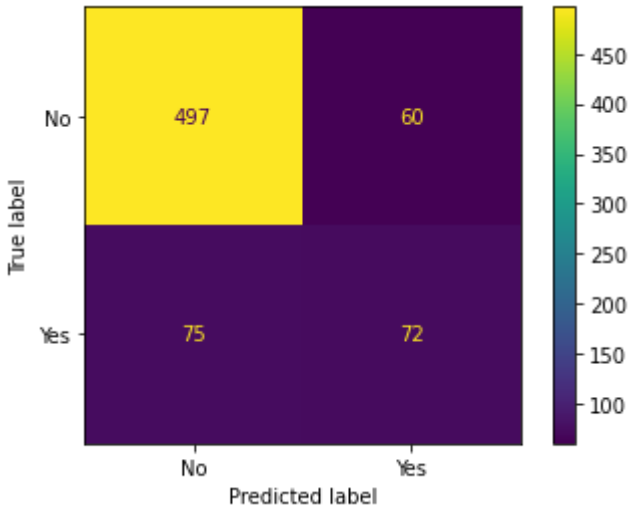
In [35]:
```
print(classification_report(y_test,preds))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| No | 0.87 | 0.89 | 0.88 | 557 |
| Yes | 0.55 | 0.49 | 0.52 | 147 |
| accuracy | | | 0.81 | 704 |
| macro avg | 0.71 | 0.69 | 0.70 | 704 |
| weighted avg | 0.80 | 0.81 | 0.80 | 704 |

In [36]:
```
plot_confusion_matrix(dt,X_test,y_test)
```

```
C:\Users\joeyd\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

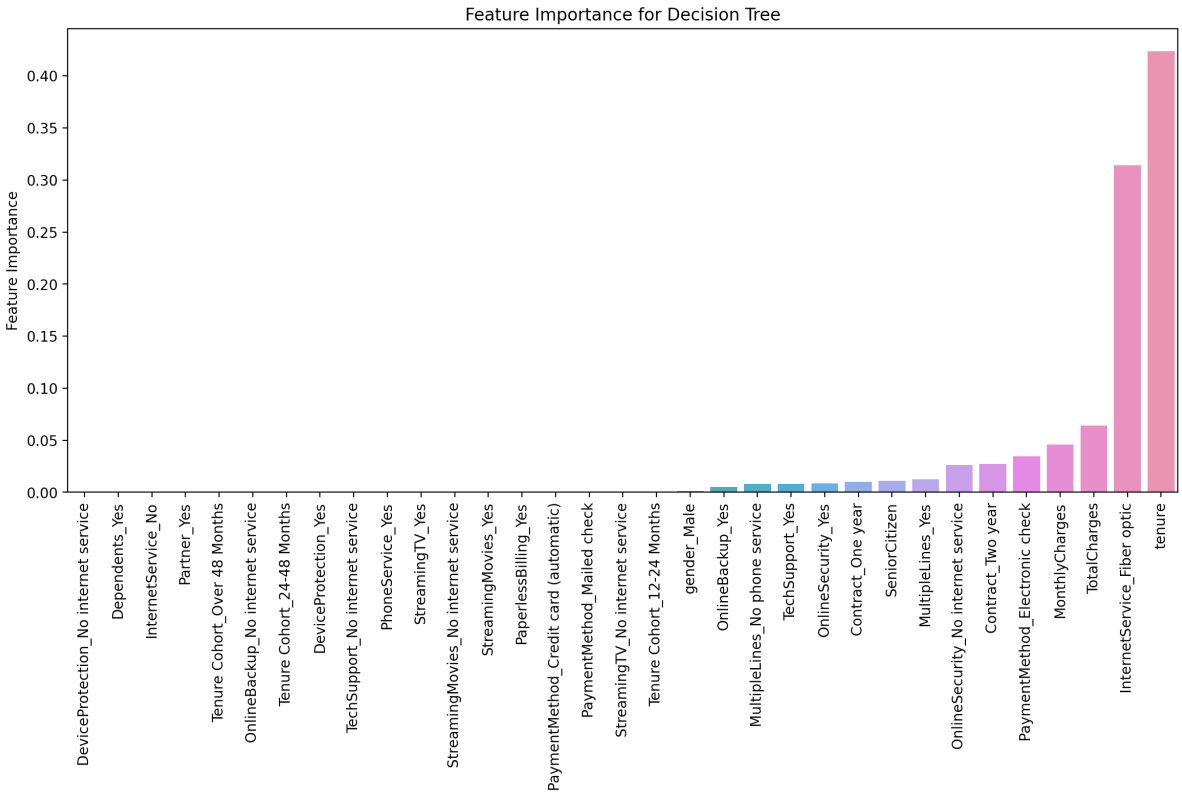
Out[36]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f2dbc9f490>
```




### Feture importance

In [37]:
```
imp_feats = pd.DataFrame(data=dt.feature_importances_,index=X.columns,
                         columns=['Feature Importance']).sort_values("Feature Impor
```

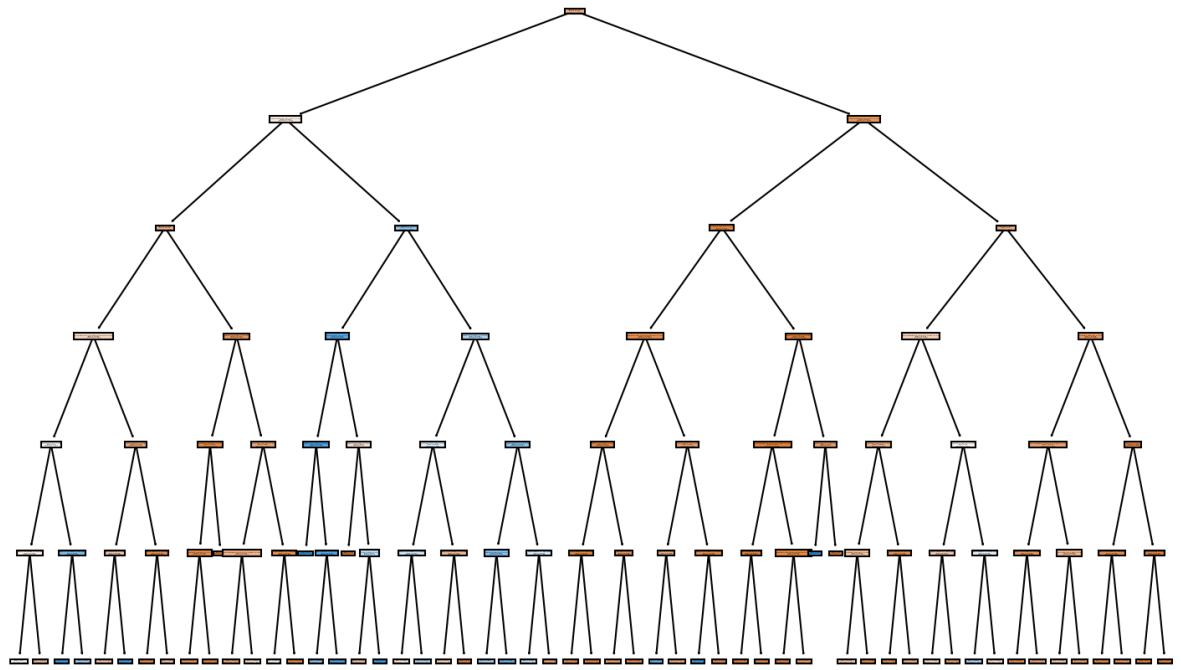
In [38]:
```
plt.figure(figsize=(14,6),dpi=200)
sns.barplot(data=imp_feats.sort_values('Feature Importance'),x=imp_feats.sort_valu
plt.xticks(rotation=90)
plt.title("Feature Importance for Decision Tree");
```




### Visualizing the tree

In [39]:
```
from sklearn.tree import plot_tree
```

In [40]:
```
plt.figure(figsize=(12,8),dpi=150)
plot_tree(dt,filled=True,feature_names=X.columns);
```



# Random Forest

In [41]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
```

```
rf.fit(X_train,y_train)
```

Out[41]:
```
RandomForestClassifier()
```

### The classification report of Random Forest

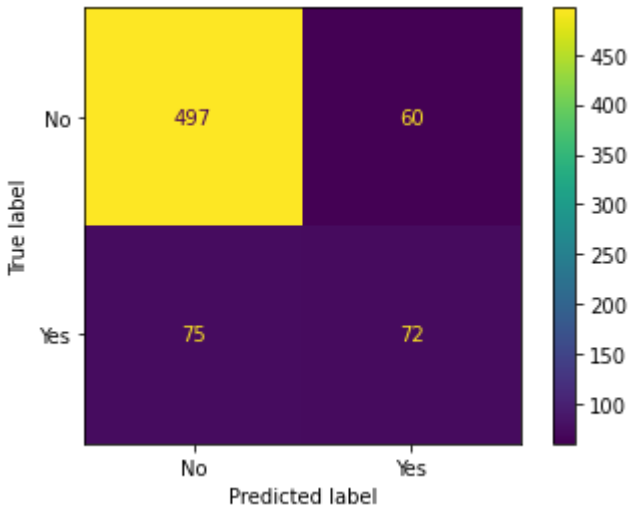
In [42]:
```
preds = rf.predict(X_test)
print(classification_report(y_test,preds))
```

```
              precision    recall  f1-score   support

          No       0.86      0.88      0.87       557
         Yes       0.51      0.46      0.48       147

    accuracy                           0.79       704
   macro avg       0.68      0.67      0.68       704
weighted avg       0.79      0.79      0.79       704
```

In [43]:
```
plot_confusion_matrix(dt,X_test,y_test)
```

```
C:\Users\joeyd\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[43]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f2df805dc0>
```




# Boosted Trees: Gradient and Ada

### Gradient

In [44]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [45]:
```
Gra_model = GradientBoostingClassifier()
```

In [46]:
```
Gra_model.fit(X_train,y_train)
```

Out[46]:
```
GradientBoostingClassifier()
```

In [47]:
```
preds = Gra_model.predict(X_test)
```

### The classification report of Gradient

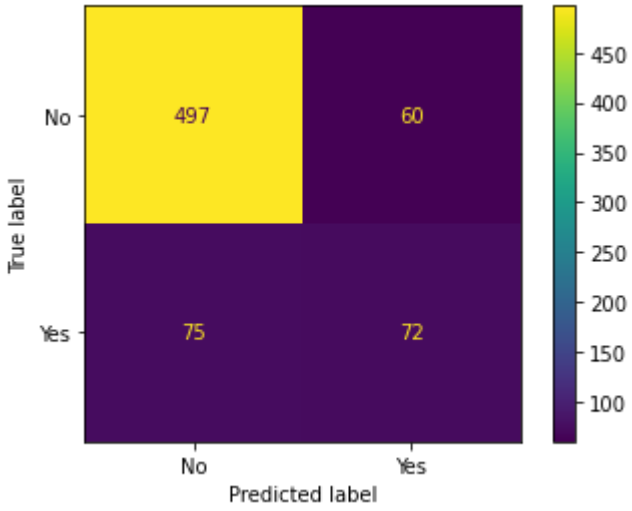
In [48]:
```
print(classification_report(y_test,preds))
```

```
              precision    recall  f1-score   support

          No       0.87      0.90      0.89       557
         Yes       0.57      0.50      0.53       147

    accuracy                           0.82       704
   macro avg       0.72      0.70      0.71       704
weighted avg       0.81      0.82      0.81       704
```

In [49]:
```
plot_confusion_matrix(dt,X_test,y_test)
```

```
C:\Users\joeyd\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[49]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f2df69fd60>
```




### Ada

In [59]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [60]:
```
ada_model = AdaBoostClassifier()
```

In [61]:
```
ada_model.fit(X_train,y_train)
```

Out[61]:
```
AdaBoostClassifier()
```

In [62]:
```
preds = ada_model.predict(X_test)
```

### The classification report of Ada

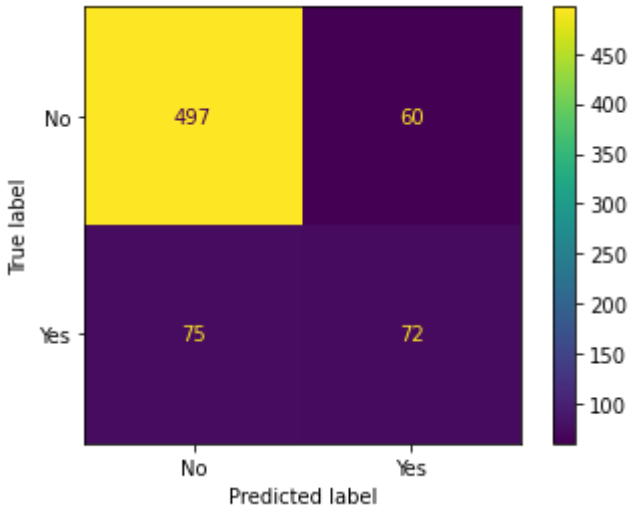
In [63]:
```
print(classification_report(y_test,preds))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| No | 0.88 | 0.90 | 0.89 | 557 |
| Yes | 0.60 | 0.54 | 0.57 | 147 |
| accuracy | | | 0.83 | 704 |
| macro avg | 0.74 | 0.72 | 0.73 | 704 |
| weighted avg | 0.82 | 0.83 | 0.83 | 704 |

In [64]:
```
plot_confusion_matrix(dt,X_test,y_test)
```

```
C:\Users\joeyd\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[64]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f2dce15400>
```




# The AdaBoosting shows it has the best average performance(f1-socre) on classifying this churn analysis