D209T2

December 4, 2024

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,␣
 ↪classification_report,confusion_matrix,precision_score,recall_score,f1_score
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.tree import plot_tree
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import learning_curve
```

[2]:
```
df = pd.read_csv('churn_clean.csv')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CaseOrder             10000 non-null  int64
 1   Customer_id           10000 non-null  object
 2   Interaction           10000 non-null  object
 3   UID                   10000 non-null  object
 4   City                  10000 non-null  object
 5   State                 10000 non-null  object
 6   County                10000 non-null  object
 7   Zip                   10000 non-null  int64
 8   Lat                   10000 non-null  float64
 9   Lng                   10000 non-null  float64
 10  Population            10000 non-null  int64
 11  Area                  10000 non-null  object
```

```
 12  TimeZone              10000 non-null  object
 13  Job                   10000 non-null  object
 14  Children              10000 non-null  int64
 15  Age                   10000 non-null  int64
 16  Income                10000 non-null  float64
 17  Marital               10000 non-null  object
 18  Gender                10000 non-null  object
 19  Churn                 10000 non-null  object
 20  Outage_sec_perweek    10000 non-null  float64
 21  Email                 10000 non-null  int64
 22  Contacts              10000 non-null  int64
 23  Yearly_equip_failure  10000 non-null  int64
 24  Techie                10000 non-null  object
 25  Contract              10000 non-null  object
 26  Port_modem            10000 non-null  object
 27  Tablet                10000 non-null  object
 28  InternetService       10000 non-null  object
 29  Phone                 10000 non-null  object
 30  Multiple              10000 non-null  object
 31  OnlineSecurity        10000 non-null  object
 32  OnlineBackup          10000 non-null  object
 33  DeviceProtection      10000 non-null  object
 34  TechSupport           10000 non-null  object
 35  StreamingTV           10000 non-null  object
 36  StreamingMovies       10000 non-null  object
 37  PaperlessBilling      10000 non-null  object
 38  PaymentMethod         10000 non-null  object
 39  Tenure                10000 non-null  float64
 40  MonthlyCharge         10000 non-null  float64
 41  Bandwidth_GB_Year     10000 non-null  float64
 42  Item1                 10000 non-null  int64
 43  Item2                 10000 non-null  int64
 44  Item3                 10000 non-null  int64
 45  Item4                 10000 non-null  int64
 46  Item5                 10000 non-null  int64
 47  Item6                 10000 non-null  int64
 48  Item7                 10000 non-null  int64
 49  Item8                 10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

[4]:
```
df.isnull().sum()
```

[4]:
```
CaseOrder          0
Customer_id        0
Interaction        0
UID                0
```

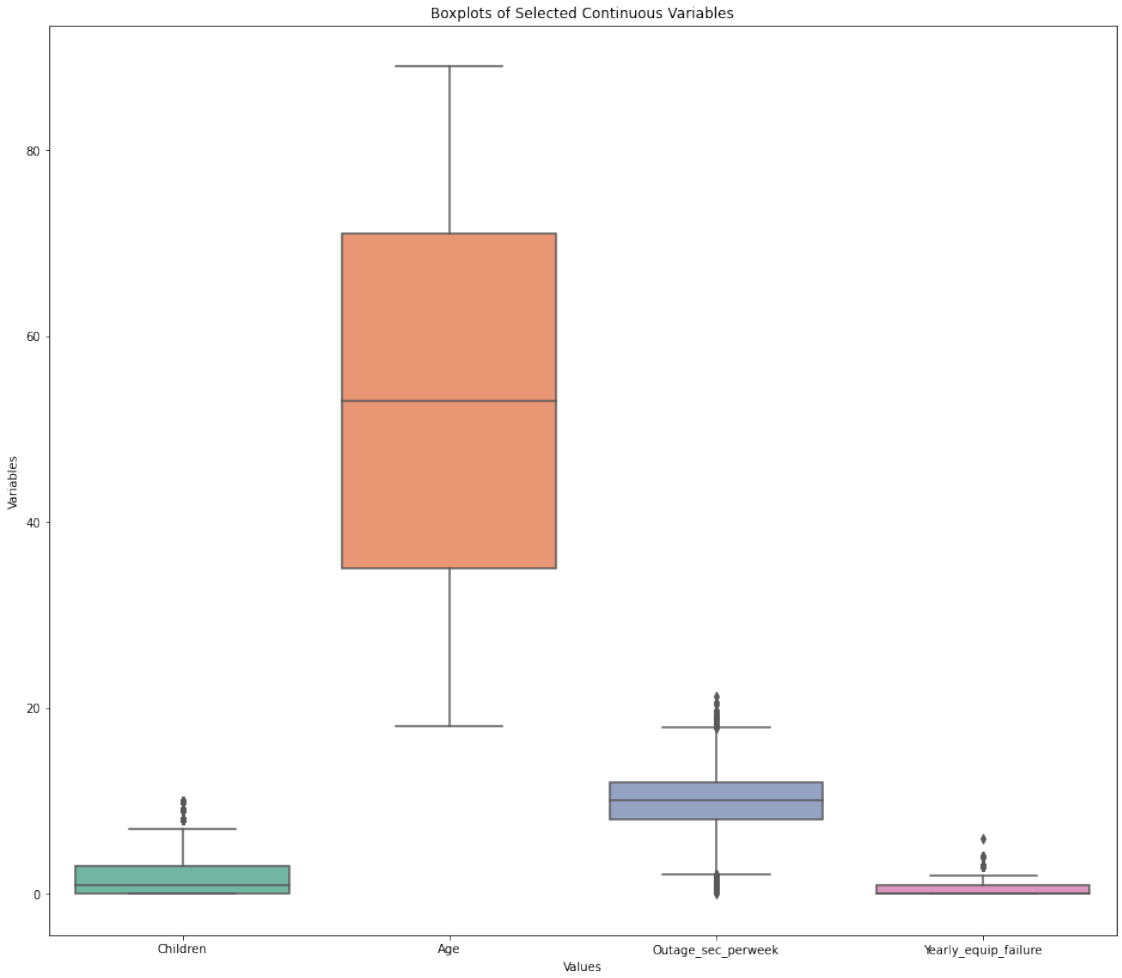
```
City                    0
State                   0
County                  0
Zip                     0
Lat                     0
Lng                     0
Population              0
Area                    0
TimeZone                0
Job                     0
Children                0
Age                     0
Income                  0
Marital                 0
Gender                  0
Churn                   0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Techie                  0
Contract                0
Port_modem              0
Tablet                  0
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
PaymentMethod           0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
Item1                   0
Item2                   0
Item3                   0
Item4                   0
Item5                   0
Item6                   0
Item7                   0
Item8                   0
dtype: int64
```

[5]:
```
df=df.
  ↪drop(columns=['CaseOrder','Customer_id','Interaction','UID','Lat','Lng','Zip','State','Coun
```

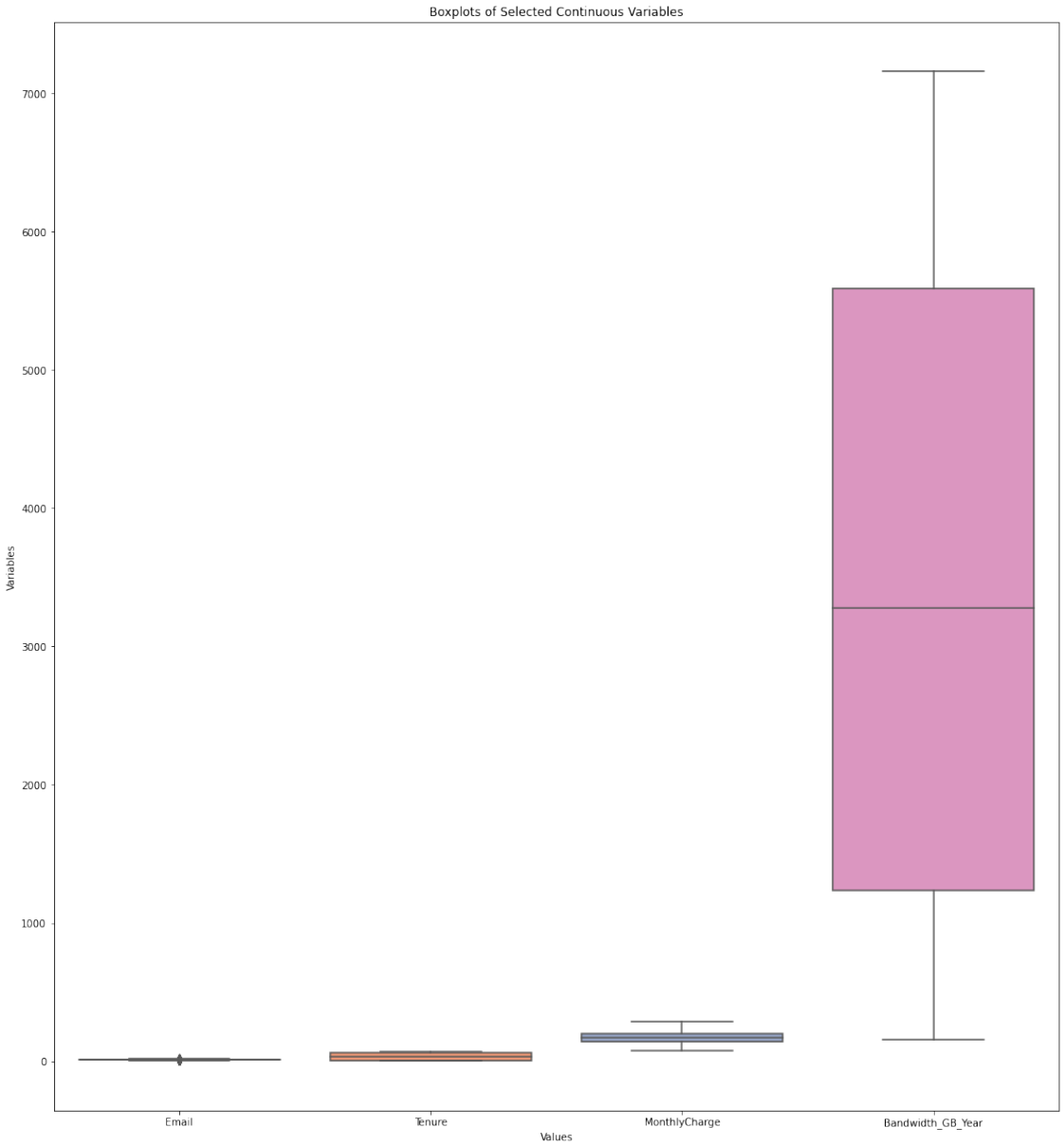
[6]:
```
print(df.dtypes)
```

```
Children                  int64
Age                       int64
Income                  float64
Marital                  object
Gender                   object
Churn                    object
Outage_sec_perweek      float64
Email                     int64
Yearly_equip_failure      int64
Techie                   object
Contract                 object
Port_modem               object
Tablet                   object
InternetService          object
Phone                    object
Multiple                 object
OnlineSecurity           object
OnlineBackup             object
DeviceProtection         object
TechSupport              object
StreamingTV              object
StreamingMovies          object
Tenure                  float64
MonthlyCharge           float64
Bandwidth_GB_Year       float64
dtype: object
```

[7]:
```
selected_variables = ['Children', 'Age', 'Outage_sec_perweek',␣
  ↪'Yearly_equip_failure']
plt.figure(figsize=(16,14))
sns.boxplot(data=df[selected_variables], orient='v', palette='Set2')
plt.title('Boxplots of Selected Continuous Variables')
plt.xlabel('Values')
plt.ylabel('Variables')
plt.show()
```



[8]:
```
selected_variables = ['Email','Tenure','MonthlyCharge','Bandwidth_GB_Year']
plt.figure(figsize=(18,20))
sns.boxplot(data=df[selected_variables], orient='v', palette='Set2')
plt.title('Boxplots of Selected Continuous Variables')
plt.xlabel('Values')
plt.ylabel('Variables')
plt.show()
```



[9]:
```
columns = ['Children', 'Outage_sec_perweek', 'Yearly_equip_failure', 'Tenure',␣
 ↪'MonthlyCharge', 'Bandwidth_GB_Year','Email', 'Age']
threshold = 3
z_scores = np.abs((df[columns] - df[columns].mean()) / df[columns].std())
trimmed_df = df[(z_scores < threshold).all(axis=1)]
```

[10]:
```
print("Original DataFrame shape:", df.shape)
print("Trimmed DataFrame shape:", trimmed_df.shape)
```

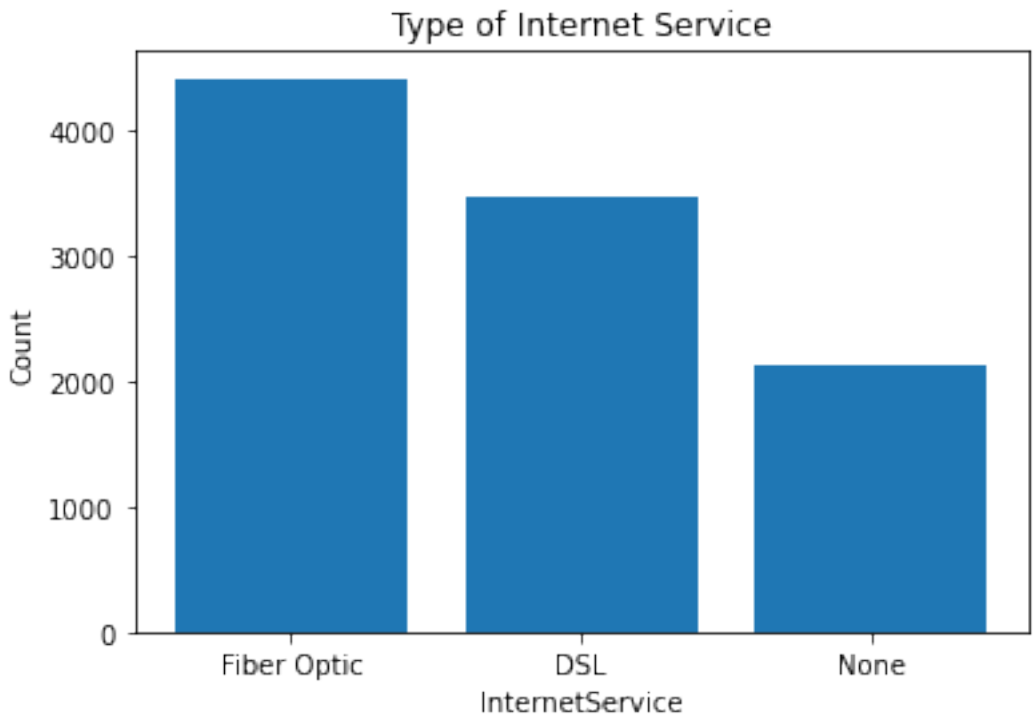
```
Original DataFrame shape: (10000, 25)
Trimmed DataFrame shape: (9678, 25)
```

[11]:
```
df.duplicated()
```

[11]:
```
0       False
1       False
2       False
3       False
4       False
        …
9995    False
9996    False
9997    False
9998    False
9999    False
Length: 10000, dtype: bool
```

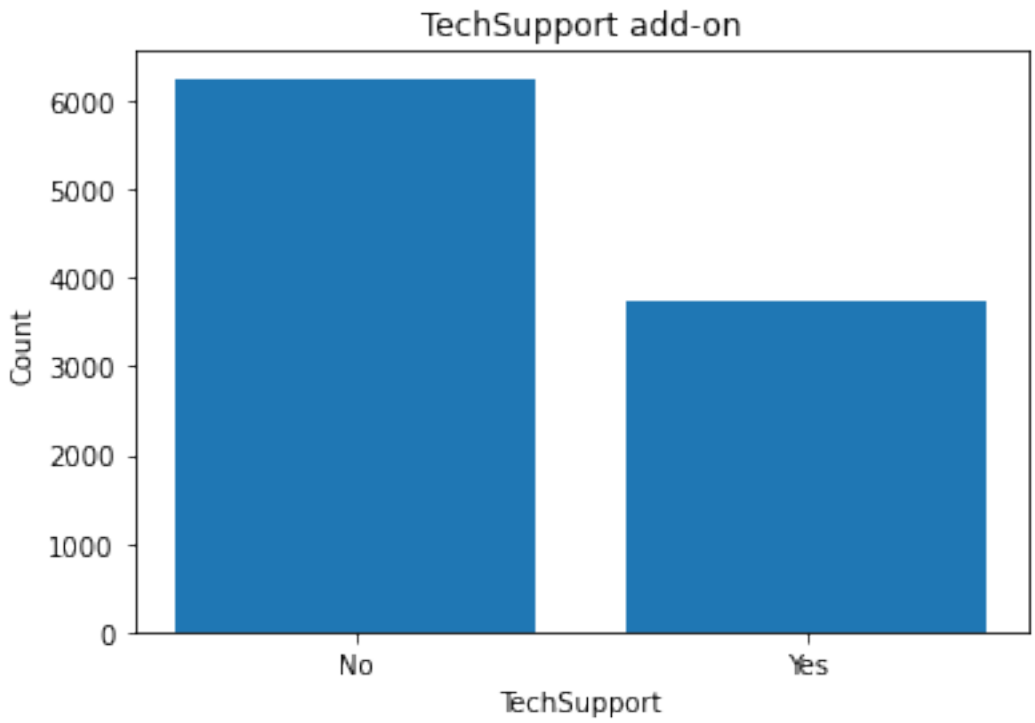
[12]:
```
Internet_Service = df['InternetService'].value_counts()
print(Internet_Service)
plt.bar(Internet_Service.index, Internet_Service.values)
plt.xlabel('InternetService')
plt.ylabel('Count')
plt.title('Type of Internet Service')
plt.show()
```

```
Fiber Optic    4408
DSL            3463
None           2129
Name: InternetService, dtype: int64
```



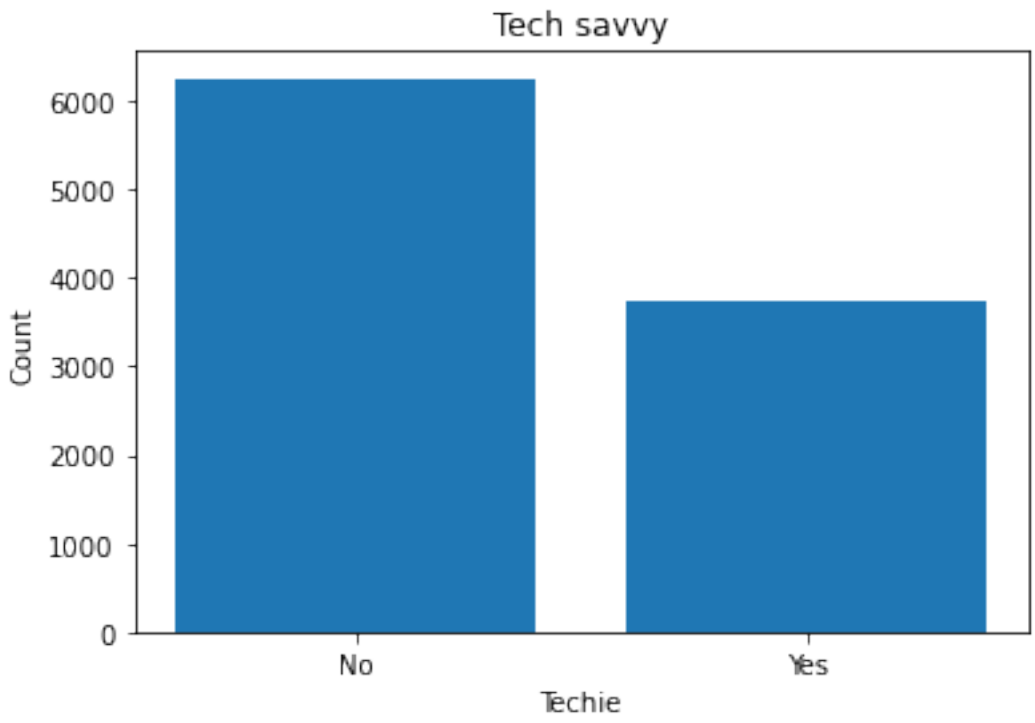
[13]:
```
Support = df['TechSupport'].value_counts()
print(Support)
plt.bar(Support.index, Support.values)
plt.xlabel('TechSupport')
plt.ylabel('Count')
plt.title('TechSupport add-on')
plt.show()
```

```
No     6250
Yes    3750
Name: TechSupport, dtype: int64
```



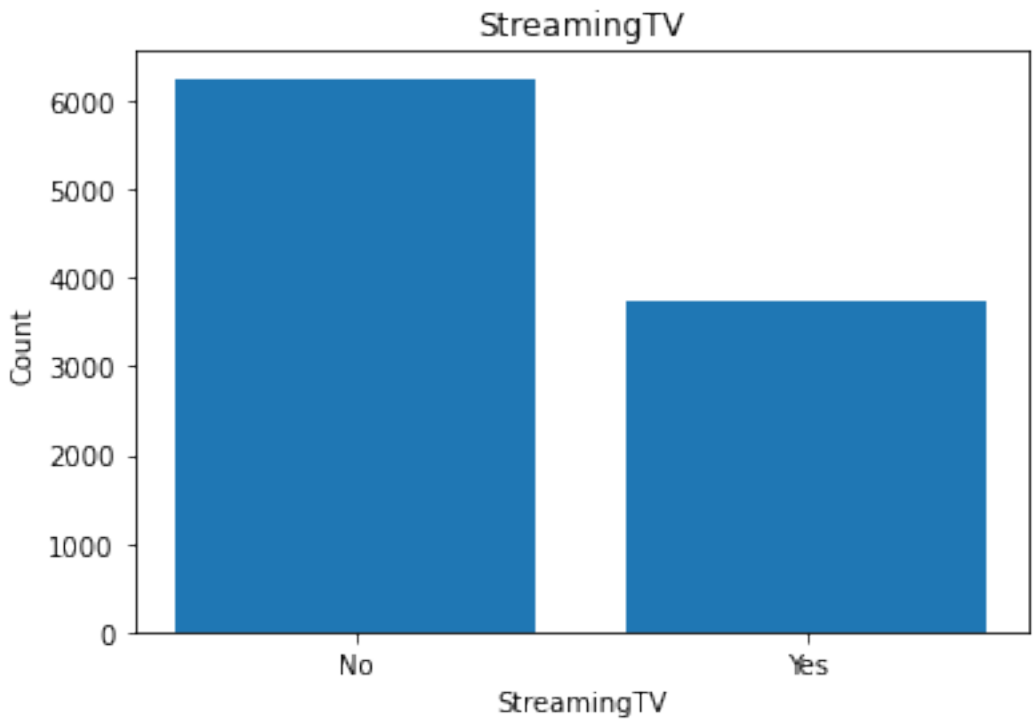
[14]:
```
Techie=df['Techie'].value_counts()
print(Techie)
Techie=df['Techie'].value_counts()
plt.bar(Support.index,Support.values)
plt.xlabel('Techie')
plt.ylabel('Count')
plt.title('Tech savvy')
plt.show()
```

```
No     8321
Yes    1679
Name: Techie, dtype: int64
```



[15]:
```
StreamingTV=df['StreamingTV'].value_counts()
print(StreamingTV)
StreamingTV=df['StreamingTV'].value_counts()
plt.bar(Support.index,Support.values)
plt.xlabel('StreamingTV')
plt.ylabel('Count')
plt.title('StreamingTV')
plt.show()
```

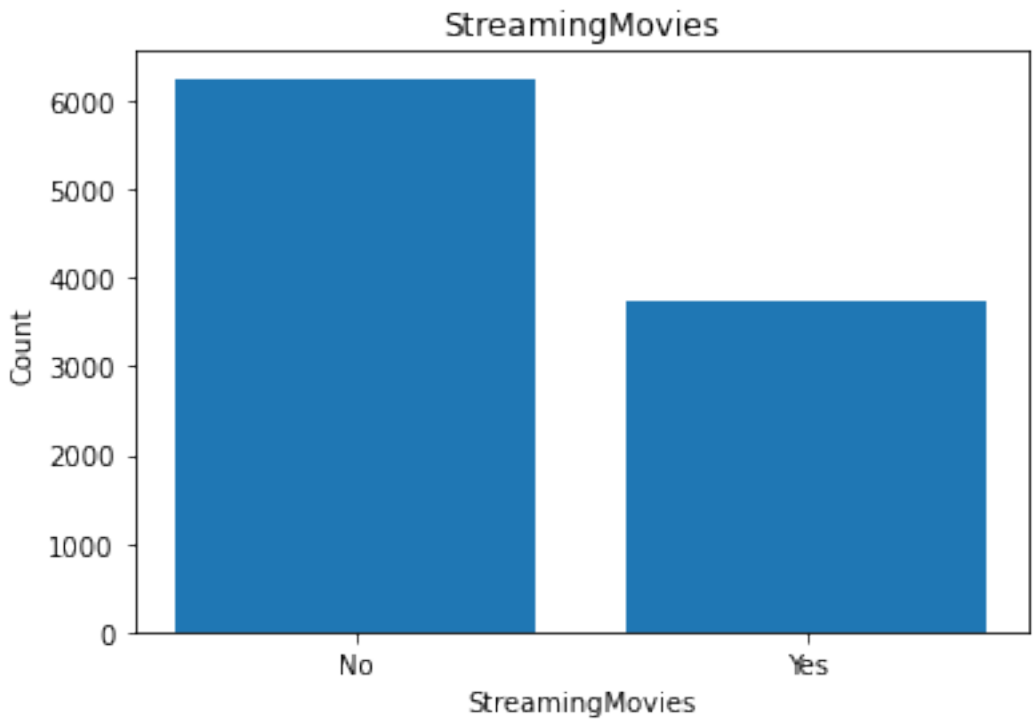
```
No     5071
Yes    4929
Name: StreamingTV, dtype: int64
```



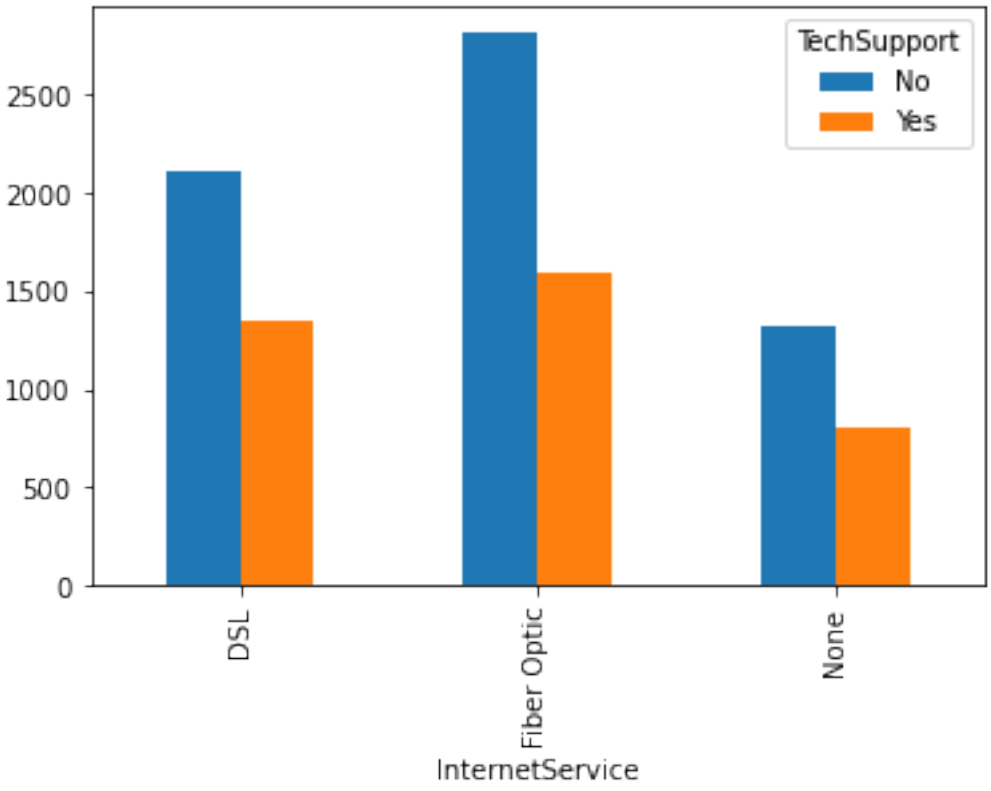
[16]:
```
StreamingMovies=df['StreamingMovies'].value_counts()
print(StreamingMovies)
```

```
No     5110
Yes    4890
Name: StreamingMovies, dtype: int64
```

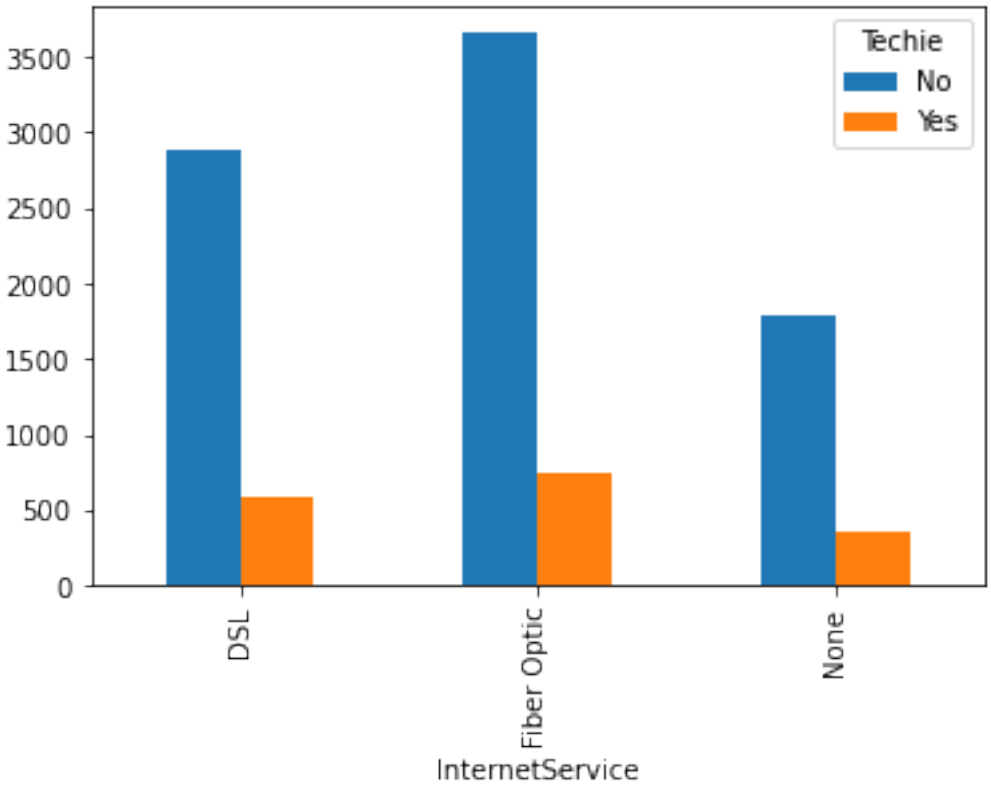
[17]:
```
StreamingMovies=df['StreamingMovies'].value_counts()
plt.bar(Support.index,Support.values)
plt.xlabel('StreamingMovies')
plt.ylabel('Count')
plt.title('StreamingMovies')
plt.show()
```



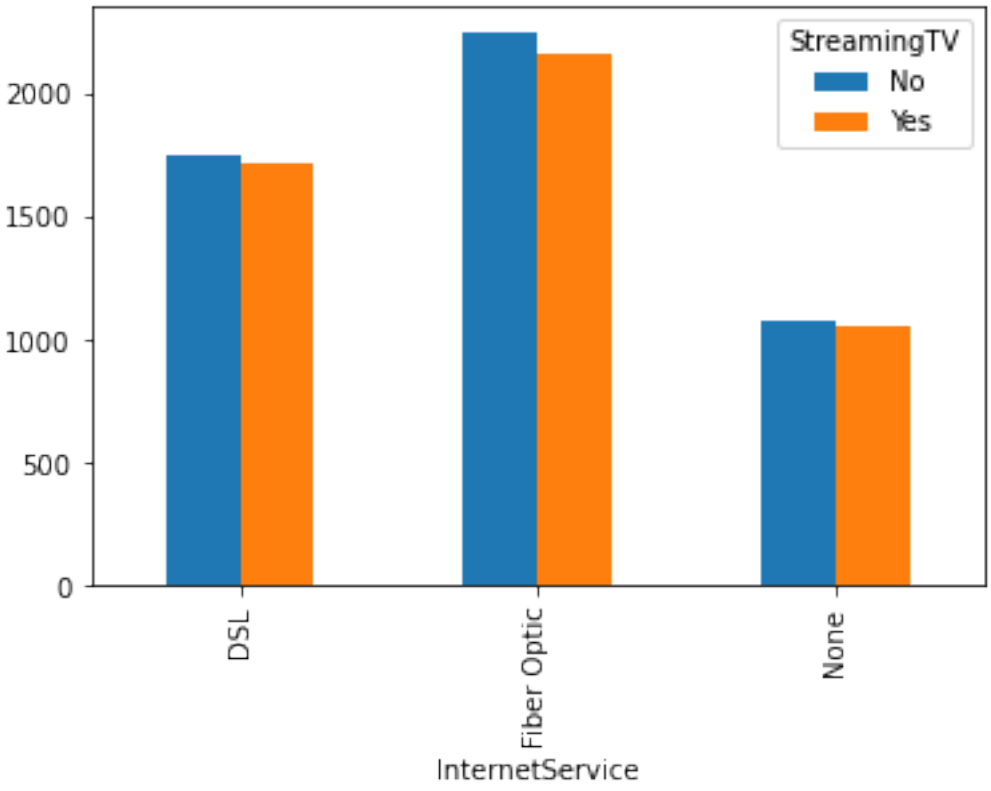
[18]:
```
cross_tab=pd.crosstab(df['InternetService'],
df['TechSupport'])
cross_tab.plot.bar()
plt.show()
```



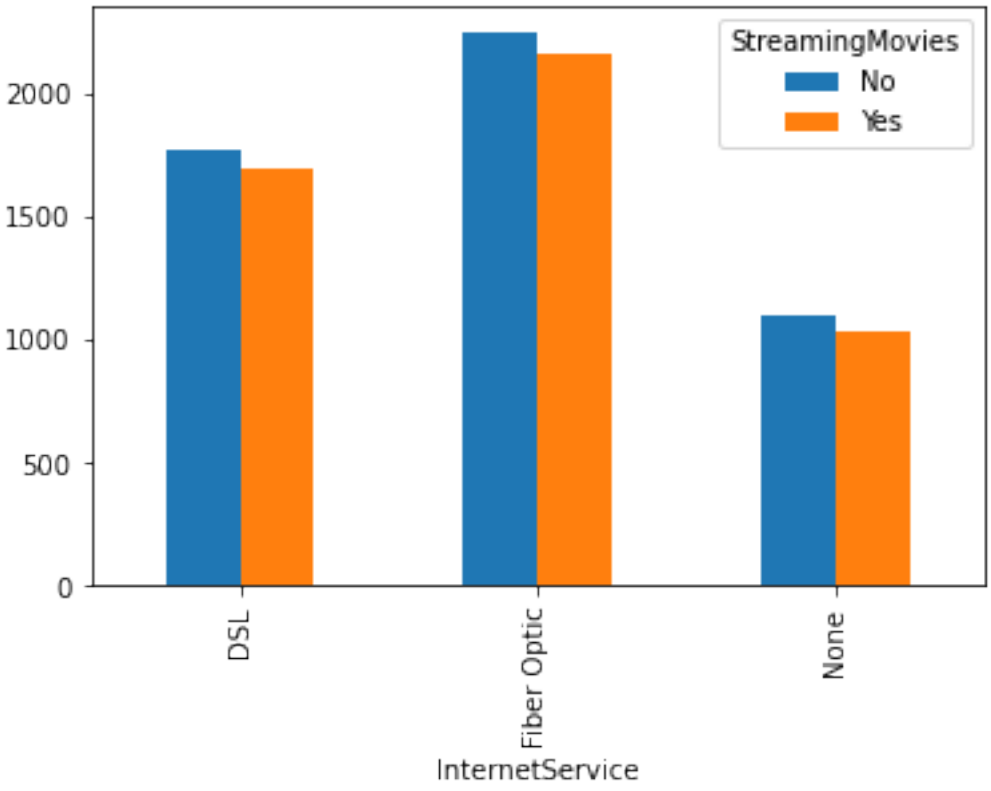
[19]:
```
cross_tab=pd.crosstab(df['InternetService'],
df['Techie'])
cross_tab.plot.bar()
plt.show()
```



[20]:
```
cross_tab=pd.crosstab(df['InternetService'],
df['StreamingTV'])
cross_tab.plot.bar()
plt.show()
```



[21]:
```
cross_tab=pd.crosstab(df['InternetService'],
df['StreamingMovies'])
cross_tab.plot.bar()
plt.show()
```



[22]:
```
df['DummyChurn'] = [1 if v == 'Yes' else 0 for v in df['Churn']]
df['DummyGender'] = [1 if v == 'Male' else 0 for v in df['Gender']]
df['DummyTechie'] = [1 if v == 'Yes' else 0 for v in df['Techie']]
df['DummyContract'] = [1 if v == 'Two Year' else 0 for v in df['Contract']]
df['DummyPort_modem'] = [1 if v == 'Yes' else 0 for v in df['Port_modem']]
df['DummyTablet'] = [1 if v == 'Yes' else 0 for v in df['Tablet']]
df['DummyPhone'] = [1 if v == 'Yes' else 0 for v in df['Phone']]
df['DummyMultiple'] = [1 if v == 'Yes' else 0 for v in df['Multiple']]
df['DummyOnlineSecurity'] = [1 if v == 'Yes' else 0 for v in␣
 ↪df['OnlineSecurity']]
df['DummyOnlineBackup'] = [1 if v == 'Yes' else 0 for v in df['OnlineBackup']]
df['DummyDeviceProtection'] = [1 if v == 'Yes' else 0 for v in␣
 ↪df['DeviceProtection']]
df['DummyTechSupport'] = [1 if v == 'Yes' else 0 for v in df['TechSupport']]
df['DummyStreamingTV'] = [1 if v == 'Yes' else 0 for v in df['StreamingTV']]
df['DummyStreamingMovies'] = [1 if v == 'Yes' else 0 for v in␣
 ↪df['StreamingMovies']]
df['DummyMarital'] = df['Marital'].
 ↪replace(['Divorced','Widowed','Separated','Never Married'],'NotMarried')
df['DummyMarital'] = [1 if v == 'Married' else 0 for v in df['Marital']]
```

```
df['TargetInternetService'] = df['InternetService'].replace({'None': 0, 'DSL':␣
↪1, 'Fiber Optic': 2})
```

[23]:
```
df=df.
↪drop(columns=['Churn','Gender','Marital','Techie','Contract','Port_modem','Tablet','Phone',
```

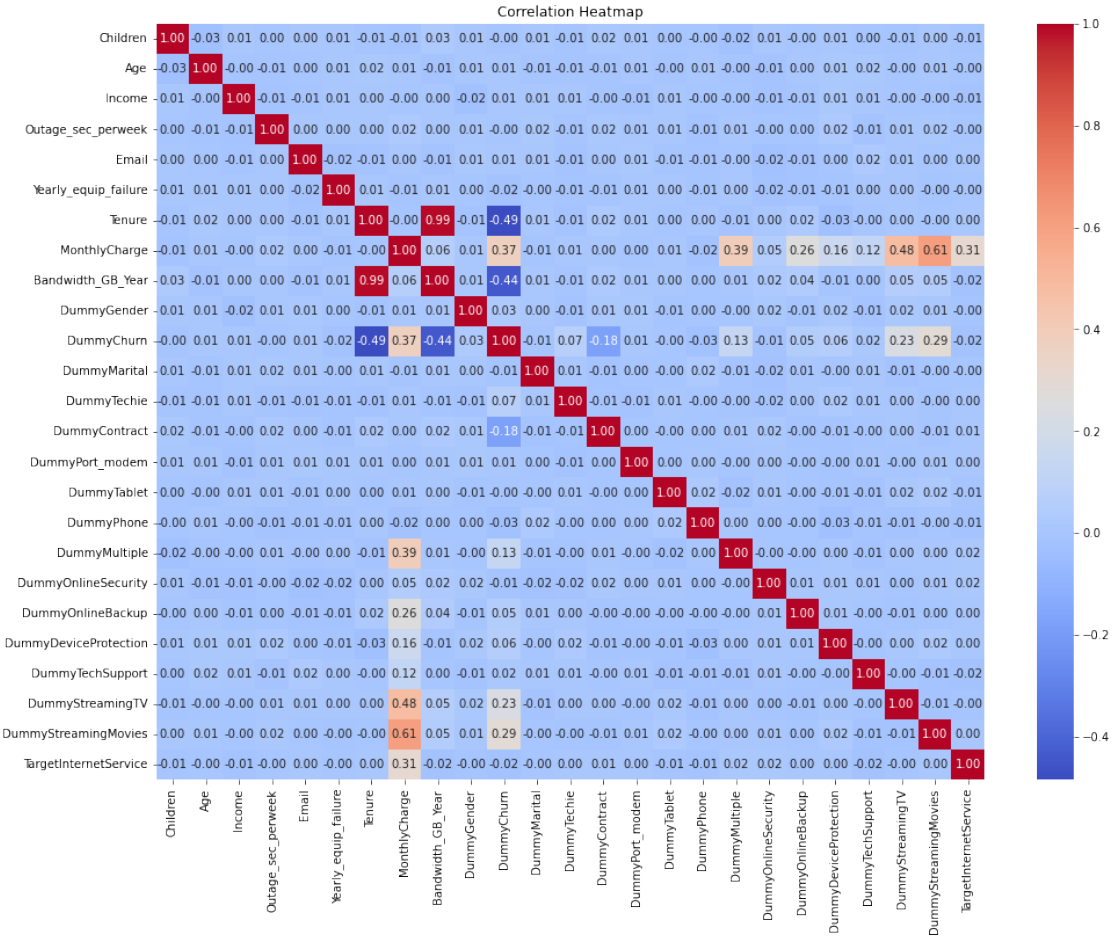
[24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 25 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Children               10000 non-null  int64
 1   Age                    10000 non-null  int64
 2   Income                 10000 non-null  float64
 3   Outage_sec_perweek     10000 non-null  float64
 4   Email                  10000 non-null  int64
 5   Yearly_equip_failure   10000 non-null  int64
 6   Tenure                 10000 non-null  float64
 7   MonthlyCharge          10000 non-null  float64
 8   Bandwidth_GB_Year      10000 non-null  float64
 9   DummyChurn             10000 non-null  int64
 10  DummyGender            10000 non-null  int64
 11  DummyTechie            10000 non-null  int64
 12  DummyContract          10000 non-null  int64
 13  DummyPort_modem        10000 non-null  int64
 14  DummyTablet            10000 non-null  int64
 15  DummyPhone             10000 non-null  int64
 16  DummyMultiple          10000 non-null  int64
 17  DummyOnlineSecurity    10000 non-null  int64
 18  DummyOnlineBackup      10000 non-null  int64
 19  DummyDeviceProtection  10000 non-null  int64
 20  DummyTechSupport       10000 non-null  int64
 21  DummyStreamingTV       10000 non-null  int64
 22  DummyStreamingMovies   10000 non-null  int64
 23  DummyMarital           10000 non-null  int64
 24  TargetInternetService  10000 non-null  int64
dtypes: float64(5), int64(20)
memory usage: 1.9 MB
```

[25]:
```
df = df[['Children', 'Age', 'Income', 'Outage_sec_perweek','Email',
'Yearly_equip_failure','Tenure','MonthlyCharge','Bandwidth_GB_Year','DummyGender','DummyChurn'
↪'DummyPort_modem', 'DummyTablet',
'DummyPhone','DummyMultiple', 'DummyOnlineSecurity',
'DummyOnlineBackup','DummyDeviceProtection', 'DummyTechSupport',␣
↪'DummyStreamingTV','DummyStreamingMovies',
```

```
'DummyMarital','TargetInternetService']]
```

[26]:
```
selected_variables = ['Children',␣
 ↪'Age','Income','Outage_sec_perweek','Email','Yearly_equip_failure',
'Tenure','MonthlyCharge','Bandwidth_GB_Year','DummyGender','DummyChurn','DummyMarital','DummyT
 ↪'DummyPort_modem','DummyTablet','DummyPhone',
'DummyMultiple','DummyOnlineSecurity','DummyOnlineBackup','DummyDeviceProtection',
'DummyTechSupport','DummyStreamingTV','DummyStreamingMovies','TargetInternetService']
correlation_matrix = df[selected_variables].corr()
plt.figure(figsize=(16, 12))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



[27]:
```
X = df[['Children', 'Age', 'Income', 'Outage_sec_perweek', 'Email',␣
 ↪'Yearly_equip_failure',
```

```
'Tenure', 'MonthlyCharge', 'Bandwidth_GB_Year', 'DummyGender', 'DummyChurn',␣
 ↪'DummyMarital', 'DummyTechie','DummyContract', 'DummyPort_modem',␣
 ↪'DummyTablet',
'DummyPhone', 'DummyMultiple', 'DummyOnlineSecurity','DummyOnlineBackup',
'DummyDeviceProtection', 'DummyTechSupport','DummyStreamingTV',
'DummyStreamingMovies']]
y = df['TargetInternetService']
```

[28]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[59]:
```
X_train.to_csv('X_train1.csv', index=False)
X_test.to_csv('X_test1.csv', index=False)
y_train.to_csv('y_train1.csv', index=False)
y_test.to_csv('y_test1.csv', index=False)
```

[60]:
```
# Check the distribution of the target variable in training and testing sets
print(y_train.value_counts(normalize=True))
print(y_test.value_counts(normalize=True))
```

```
2    0.438625
1    0.345750
0    0.215625
Name: TargetInternetService, dtype: float64
2    0.4495
1    0.3485
0    0.2020
Name: TargetInternetService, dtype: float64
```

[32]:
```
alpha=5
k=15
selector=SelectKBest(score_func=f_classif,k=k)
X_selected=selector.fit_transform(X_train,y_train)
selected_indices=selector.get_support(indices=True)
all_pvalues=selector.pvalues_
all_feature_names=X_train.columns
selected_feature_names=all_feature_names[selected_indices]
```

[33]:
```
for feature_name, p_value in zip(selected_feature_names, selector.
 ↪pvalues_[selected_indices]):



    print("Feature:", feature_name)
    print("P-value:", p_value)
```

```
Feature: Children
P-value: 0.6134689054114932
Feature: Age
```

```
P-value: 0.7375756502141271
Feature: Income
P-value: 0.21538984121140972
Feature: Outage_sec_perweek
P-value: 0.4449476744312997
Feature: Tenure
P-value: 0.2719189798893537
Feature: MonthlyCharge
P-value: 2.015575199710615e-176
Feature: Bandwidth_GB_Year
P-value: 6.938871198638748e-21
Feature: DummyChurn
P-value: 1.2919717354868597e-13
Feature: DummyContract
P-value: 0.3152088678850793
Feature: DummyPhone
P-value: 0.8057496198436502
Feature: DummyMultiple
P-value: 0.06079072923256341
Feature: DummyOnlineSecurity
P-value: 0.26870349726812615
Feature: DummyDeviceProtection
P-value: 0.47789452341127125
Feature: DummyTechSupport
P-value: 0.011471929463258186
Feature: DummyStreamingTV
P-value: 0.514385821233489
```

[34]:
```
df=df.drop(columns=['Children',␣
 ↪'Age','Income','Outage_sec_perweek','Email','Yearly_equip_failure',
'Tenure','DummyGender','DummyMarital','DummyTechie',
'DummyPort_modem','DummyTablet','DummyPhone',
'DummyOnlineSecurity','DummyOnlineBackup','DummyDeviceProtection','DummyTechSupport',
'DummyStreamingMovies'])
```

[35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   MonthlyCharge         10000 non-null  float64
 1   Bandwidth_GB_Year     10000 non-null  float64
 2   DummyChurn            10000 non-null  int64
 3   DummyContract         10000 non-null  int64
 4   DummyMultiple         10000 non-null  int64
 5   DummyStreamingTV      10000 non-null  int64
```

```
 6   TargetInternetService  10000 non-null  int64
dtypes: float64(2), int64(5)
memory usage: 547.0 KB
```

[36]:
```
df.to_csv('Prepared_2092df.csv', index=False)
```

[37]:
```
X = df[['MonthlyCharge','DummyChurn','Bandwidth_GB_Year', 'DummyContract',␣
 ↪'DummyMultiple', 'DummyStreamingTV']]
y = df['TargetInternetService']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[38]:
```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_train_pred = clf.predict(X_train)
```

[39]:
```
accuracy_train = accuracy_score(y_train, y_train_pred)
precision_train = precision_score(y_train, y_train_pred, average='weighted')
recall_train = recall_score(y_train, y_train_pred, average='weighted')
f1_train = f1_score(y_train, y_train_pred, average='weighted')
conf_matrix_train = confusion_matrix(y_train, y_train_pred)
mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mse_train)
```

[40]:
```
print("Training Set Metrics:")
print("Accuracy:", accuracy_train)
print("Precision:", precision_train)
print("Recall:", recall_train)
print("F1-score:", f1_train)
print("Confusion Matrix:")
print(conf_matrix_train)
print("Mean Squared Error (Training set):", mse_train)
print("Root Mean Squared Error (Training set):", rmse_train)
```

```
Training Set Metrics:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1-score: 1.0
Confusion Matrix:
[[1725    0    0]
 [   0 2766    0]
 [   0    0 3509]]
Mean Squared Error (Training set): 0.0
Root Mean Squared Error (Training set): 0.0
```

[41]:
```
clf_initial = DecisionTreeClassifier(random_state=42)
cv_scores_initial = cross_val_score(clf_initial, X_train, y_train, cv=5)
```

```
print("Initial Cross-validation scores:", cv_scores_initial)
print("Mean Initial CV score:", cv_scores_initial.mean())
```

```
Initial Cross-validation scores: [0.92375  0.935625 0.933125 0.929375 0.928125]
Mean Initial CV score: 0.9299999999999999
```

[42]:
```
print("Standard deviation of Initial CV scores:", cv_scores_initial.std())
mse_train = mean_squared_error(y_train, y_train_pred)
print("Mean Squared Error (Training set):", mse_train)
```

```
Standard deviation of Initial CV scores: 0.004107919181288769
Mean Squared Error (Training set): 0.0
```

[43]:
```
param_grid = {
    'max_depth': [5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 5, 10, 20]
}

grid_search = GridSearchCV(estimator=DecisionTreeClassifier(random_state=42),␣
 ↪param_grid=param_grid, cv=5)


grid_search.fit(X_train, y_train)


print("Best parameters:", grid_search.best_params_)
```

```
Best parameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split':
2}
```

[44]:
```
best_tree_clf = DecisionTreeClassifier(max_depth=20, min_samples_leaf=1,␣
 ↪min_samples_split=10, random_state=42)
best_tree_clf.fit(X_train, y_train)
```

[44]:
```
DecisionTreeClassifier(max_depth=20, min_samples_split=10, random_state=42)
```

[45]:
```
y_pred = best_tree_clf.predict(X_test)
accuracy = best_tree_clf.score(X_test, y_test)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
conf_matrix = confusion_matrix(y_test, y_pred)
```

[46]:
```
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("Confusion Matrix:")
```

```
print(conf_matrix)
```

```
Accuracy: 0.9155
Precision: 0.9151232154828443
Recall: 0.9155
F1-score: 0.9152360343510478
Confusion Matrix:
[[341  37  26]
 [ 32 632  33]
 [ 23  18 858]]
```

[62]:
```
y_pred = best_tree_clf.predict(X_test)
mse_test = mean_squared_error(y_test, y_pred)
rmse_test = np.sqrt(mse_test)
print("Test Mean Squared Error (MSE):", mse_test)
print("Test Root Mean Squared Error (RMSE):", rmse_test)
```

```
Test Mean Squared Error (MSE): 0.158
Test Root Mean Squared Error (RMSE): 0.39749213828703583
```

[47]:
```
cv_scores = cross_val_score(best_tree_clf, X_train, y_train, cv=5)
```

[48]:
```
mean_cv_score = np.mean(cv_scores)
std_cv_score = np.std(cv_scores)
```

[49]:
```
print("Cross-validation scores:", cv_scores)
print("Mean CV score:", mean_cv_score)
print("Standard deviation of CV scores:", std_cv_score)
```

```
Cross-validation scores: [0.899375 0.89375  0.900625 0.91     0.914375]
Mean CV score: 0.9036250000000001
Standard deviation of CV scores: 0.007493747393660934
```

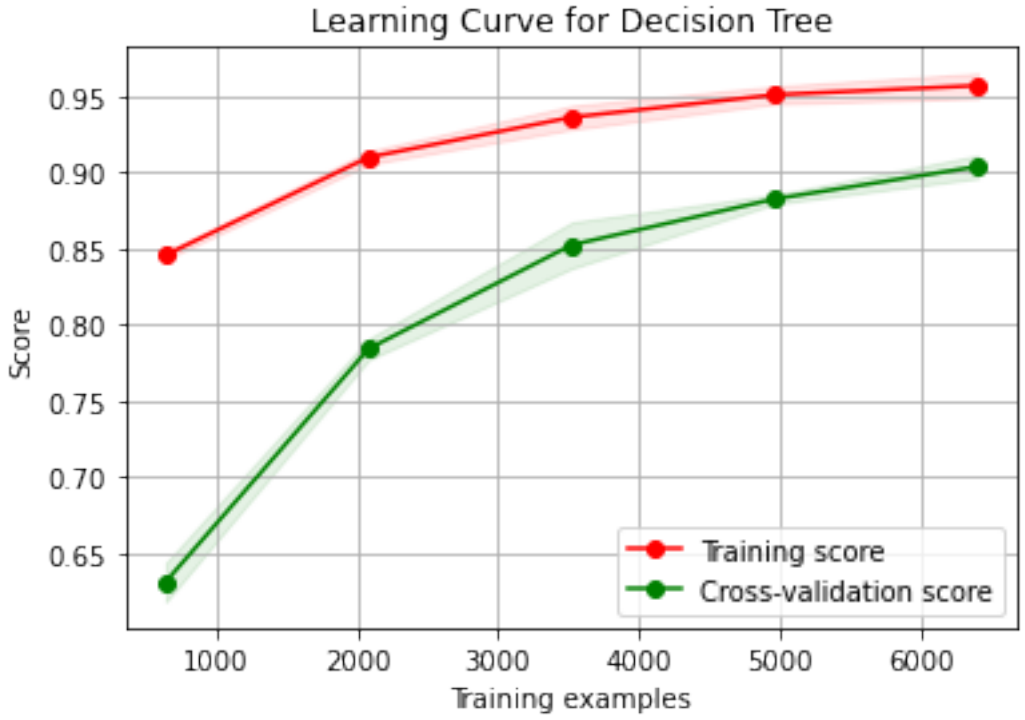
[50]:
```
cv = 5
```

[51]:
```
train_sizes, train_scores, test_scores = learning_curve(
    best_tree_clf, X_train, y_train, cv=cv, n_jobs=-1, train_sizes=np.
 ↪linspace(0.1, 1.0, 5)
)
```

[52]:
```
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
```

[53]:
```
plt.figure()
plt.title("Learning Curve for Decision Tree")
plt.xlabel("Training examples")
plt.ylabel("Score")
```

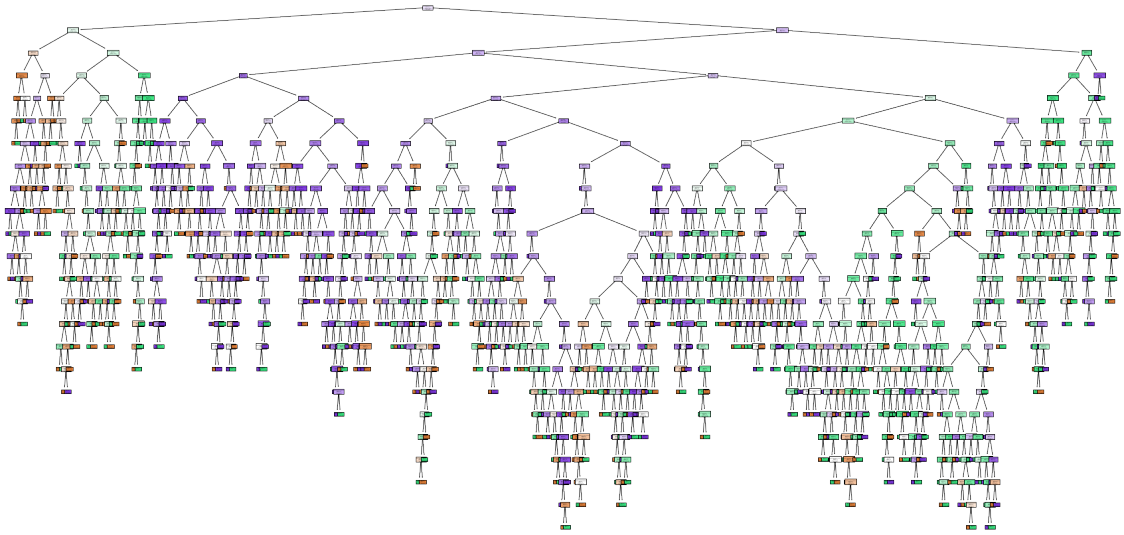
```
plt.grid()

plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                 train_scores_mean + train_scores_std, alpha=0.1, color="r")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.1, color="g")
plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training␣
 ↪score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g",␣
 ↪label="Cross-validation score")

plt.legend(loc="best")
plt.show()
```



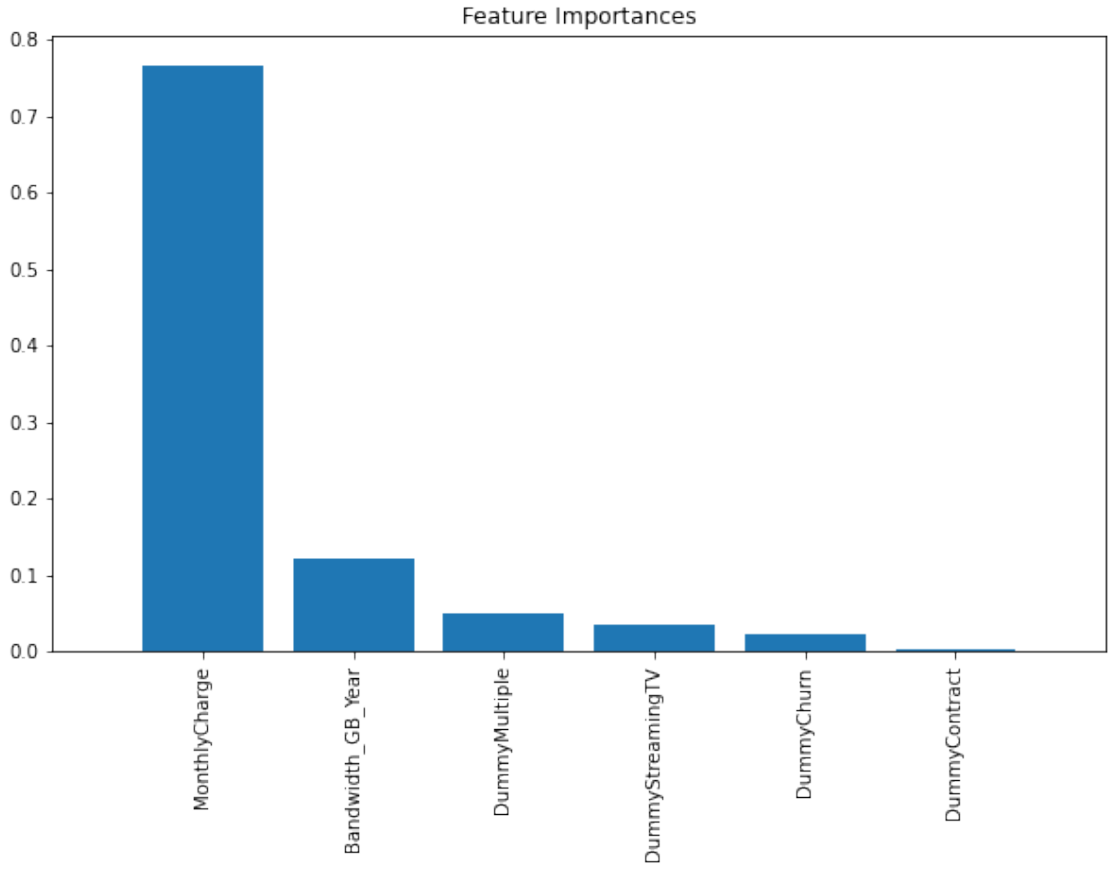
[54]:
```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

```
Mean Squared Error: 0.158
Root Mean Squared Error: 0.39749213828703583
```

[55]:
```
plt.figure(figsize=(30,15))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['DSL', 'Fiber␣
↪optic', 'None'])  # Plot the tree
plt.show()
```



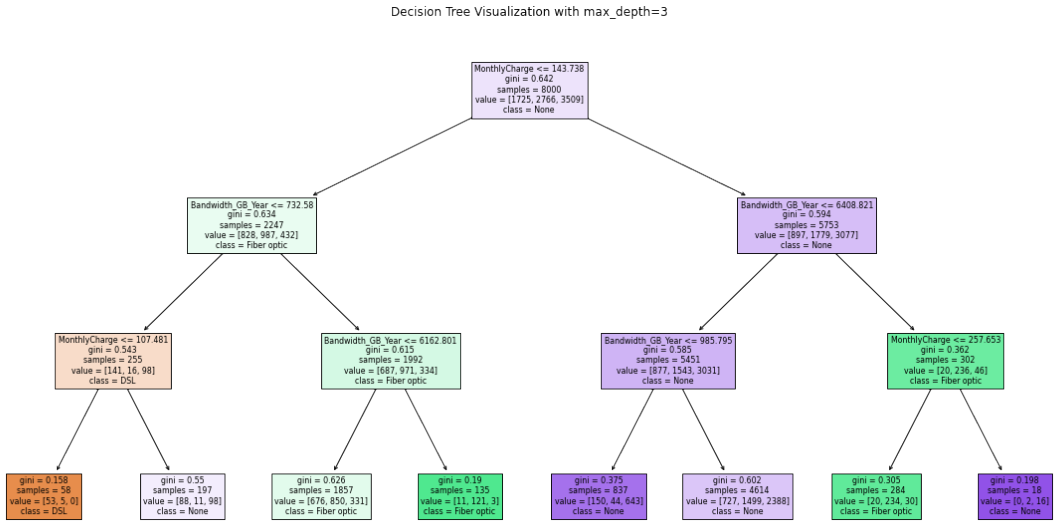
[56]:
```
importance = clf.feature_importances_
indices = np.argsort(importance)[::-1]

plt.figure(figsize=(10, 6))
plt.title("Feature Importances")
plt.bar(range(X.shape[1]), importance[indices], align="center")
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=90)
plt.xlim([-1, X.shape[1]])
plt.show()
```



[57]:
```
clf_visual = DecisionTreeClassifier(max_depth=3, random_state=42)
clf_visual.fit(X_train, y_train)
```

[57]:
```
DecisionTreeClassifier(max_depth=3, random_state=42)
```

[58]:
```
plt.figure(figsize=(20, 10))
plot_tree(clf_visual, filled=True, feature_names=X.columns, class_names=['DSL',␣
↪'Fiber optic', 'None'])
plt.title("Decision Tree Visualization with max_depth=3")
plt.show()
```


Decision Tree Visualization with max_depth=3
MonthlyCharge <= 143.738
gini = 0.642
samples = 8000
value = [1725, 2766, 3509]
class = None
Bandwidth_GB_Year <= 732.58
gini = 0.634
samples = 2247
value = [828, 987, 432]
class = Fiber optic
Bandwidth_GB_Year <= 6408.821
gini = 0.594
samples = 5753
value = [897, 1779, 3077]
class = None
MonthlyCharge <= 107.481
gini = 0.543
samples = 255
value = [141, 16, 98]
class = DSL
Bandwidth_GB_Year <= 6162.801
gini = 0.615
samples = 1992
value = [687, 971, 334]
class = Fiber optic
Bandwidth_GB_Year <= 985.795
gini = 0.585
samples = 5451
value = [877, 1543, 3031]
class = None
MonthlyCharge <= 257.653
gini = 0.362
samples = 302
value = [20, 236, 46]
class = Fiber optic
gini = 0.158
samples = 58
value = [53, 5, 0]
class = DSL
gini = 0.55
samples = 197
value = [88, 11, 98]
class = None
gini = 0.626
samples = 1857
value = [676, 850, 331]
class = Fiber optic
gini = 0.19
samples = 135
value = [11, 121, 3]
class = Fiber optic
gini = 0.375
samples = 837
value = [150, 44, 643]
class = None
gini = 0.602
samples = 4614
value = [727, 1499, 2388]
class = None
gini = 0.305
samples = 284
value = [20, 234, 30]
class = Fiber optic
gini = 0.198
samples = 18
value = [0, 2, 16]
class = None

[ ]: