Project – Customer Churn:

Customer-Churn Dataset:

jupyter Untitled18 Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

[7]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
```

[75]:
```
churn = pd.read_csv(r"C:\Users\Sushmitha\Downloads\customer_churn (1).csv")
```

[77]:
```
churn
```

[77]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	No
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	No
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	No	No

1.Data Manipulation:

Input:-

[83]:
```
# Extract the 5th column and store it in 'customer_5'
customer_5 = churn.iloc[:, 4]

# Extract the 15th column and store it in 'customer_15'
customer_15 = churn.iloc[:, 14]

# Extract all the male senior citizens whose payment method is electronic check
senior_male_electronic = churn[(churn['gender'] == 'Male') & (churn['SeniorCitizen'] == 1) & (churn['PaymentMethod'] == 'Electronic check')]

# Extract all those customers whose tenure is greater than 70 months or their monthly charges are more than $100
customer_total_tenure = churn[(churn['tenure'] > 70) | (churn['MonthlyCharges'] > 100)]

# Extract all the customers whose contract is of two years, payment method is mailed check, and churn is 'Yes'
two_mail_yes = churn[
    (churn['Contract'] == 'Two year') &
    (churn['PaymentMethod'] == 'Mailed check') &
    (churn['Churn'] == 'Yes')
]

# Extract 333 random records from the churn dataframe
customer_333 = churn.sample(n=333, random_state=42)

# Get the count of different levels from the 'Churn' column
churn_count = churn['Churn'].value_counts()

# Display the results
print("Customer 5 column values:\n", customer_5)
print("\nCustomer 15 column values:\n", customer_15.head())
print("\nSenior Male with Electronic Check:\n", senior_male_electronic.head())
print("\nCustomers with tenure > 70 months or MonthlyCharges > $100:\n", customer_total_tenure.head())
print("Customers with Two Year Contract, Mailed Check, and Churn = Yes:\n", two_mail_yes.head())
print("\nRandom 333 customers:\n", customer_333.head())
print("\nChurn count:\n", churn_count)
```

```
Customer 5 column values:
```

Output:-

```
Customer 5 column values:
 0       No
1       No
2       No
3       No
4       No
       ...
7038    Yes
7039    Yes
7040    Yes
7041     No
7042     No
Name: Dependents, Length: 7043, dtype: object

Customer 15 column values:
 0    No
1    No
2    No
3    No
4    No
Name: StreamingMovies, dtype: object

Senior Male with Electronic Check:
     customerID gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
20  8779-QRDMV   Male              1      No         No       1           No
55  1658-BYGOY   Male              1      No         No      18          Yes
57  5067-XJQFU   Male              1     Yes        Yes      66          Yes
78  0191-ZHSKZ   Male              1      No         No      30          Yes
91  2424-WVHPL   Male              1      No         No       1          Yes

       MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
20  No phone service            DSL             No  ...              Yes
55               Yes     Fiber optic             No  ...               No
57               Yes     Fiber optic             No  ...              Yes
78                No             DSL            Yes  ...               No
91                No     Fiber optic             No  ...               No

   TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
20         No          No             Yes  Month-to-month             Yes
55         No         Yes             Yes  Month-to-month             Yes
57        Yes         Yes             Yes        One year             Yes
78         No         Yes             Yes  Month-to-month             Yes
91        Yes          No              No  Month-to-month              No

       PaymentMethod MonthlyCharges  TotalCharges Churn
20  Electronic check         39.65        39.65   Yes
55  Electronic check         95.45      1752.55   Yes
57  Electronic check        108.45      7076.35    No
78  Electronic check         74.75       2111.3    No
91  Electronic check         74.70         74.7    No

[5 rows x 21 columns]
```

[ ]:

```
Customers with tenure > 70 months or MonthlyCharges > $100:
     customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
8   7892-POOKP  Female              0     Yes         No      28          Yes
12  8091-TTVAX    Male              0     Yes         No      58          Yes
13  0280-XJGEX    Male              0      No         No      49          Yes
14  5129-JLPIS    Male              0      No         No      25          Yes
15  3655-SNQYZ  Female              0     Yes        Yes      69          Yes

   MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
8            Yes     Fiber optic             No  ...              Yes
12           Yes     Fiber optic             No  ...              Yes
13           Yes     Fiber optic             No  ...              Yes
14            No     Fiber optic            Yes  ...              Yes
15           Yes     Fiber optic            Yes  ...              Yes

   TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
8          Yes         Yes             Yes  Month-to-month             Yes
12          No         Yes             Yes        One year              No
13          No         Yes             Yes  Month-to-month             Yes
14         Yes         Yes             Yes  Month-to-month             Yes
15         Yes         Yes             Yes        Two year              No

                PaymentMethod MonthlyCharges  TotalCharges Churn
8           Electronic check         104.80       3046.05   Yes
12    Credit card (automatic)        100.35        5681.1    No
13  Bank transfer (automatic)        103.70        5036.3   Yes
14          Electronic check         105.50       2686.05    No
15    Credit card (automatic)        113.25       7895.15    No

[5 rows x 21 columns]
Customers with Two Year Contract, Mailed Check, and Churn = Yes:
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
268   6323-AYBRX    Male              0      No         No      59
5947  7951-QKZPL  Female              0     Yes        Yes      33
6680  9412-ARGBX  Female              0      No        Yes      48

     PhoneService MultipleLines InternetService       OnlineSecurity  ...  \
268           Yes            No              No  No internet service  ...
5947          Yes           Yes              No  No internet service  ...
6680          Yes            No     Fiber optic                  No  ...

          DeviceProtection         TechSupport         StreamingTV  \
268   No internet service  No internet service  No internet service
5947  No internet service  No internet service  No internet service
6680                  Yes                 Yes                 Yes

          StreamingMovies  Contract PaperlessBilling PaymentMethod  \
268   No internet service  Two year              No  Mailed check
5947  No internet service  Two year             Yes  Mailed check
6680                   No  Two year             Yes  Mailed check

     MonthlyCharges  TotalCharges Churn
268          19.35        1099.6   Yes
5947         24.50         740.3   Yes
6680         95.50       4627.85   Yes

[3 rows x 21 columns]
```

```
Random 333 customers:
       customerID  gender  SeniorCitizen Partner Dependents  tenure  \
185   1024-GUALD  Female              0     Yes         No       1
2715  0484-JPBRU    Male              0      No         No      41
3825  3620-EHIMZ  Female              0     Yes        Yes      52
1807  6910-HADCM  Female              0      No         No       1
132   8587-XYZSF    Male              0      No         No      67

     PhoneService     MultipleLines InternetService       OnlineSecurity  ...  \
185           No  No phone service            DSL                   No  ...
2715         Yes               Yes             No  No internet service  ...
3825         Yes                No             No  No internet service  ...
1807         Yes                No     Fiber optic                   No  ...
132          Yes                No            DSL                   No  ...

         DeviceProtection          TechSupport          StreamingTV  \
185                    No                   No                   No
2715  No internet service  No internet service  No internet service
3825  No internet service  No internet service  No internet service
1807                  Yes                   No                   No
132                    No                  Yes                   No

          StreamingMovies        Contract PaperlessBilling  \
185                    No  Month-to-month             Yes
2715  No internet service  Month-to-month             Yes
3825  No internet service        Two year              No
1807                    No  Month-to-month              No
132                     No        Two year              No

                   PaymentMethod MonthlyCharges  TotalCharges Churn
185            Electronic check          24.80          24.8   Yes
2715  Bank transfer (automatic)          25.25        996.45    No
3825               Mailed check          19.35        1031.7    No
1807            Electronic check          76.35         76.35   Yes
132   Bank transfer (automatic)          50.55        3260.1    No

[5 rows x 21 columns]

Churn count:
 Churn
No     5174
Yes    1869
Name: count, dtype: int64
```

2. Data Visualization:

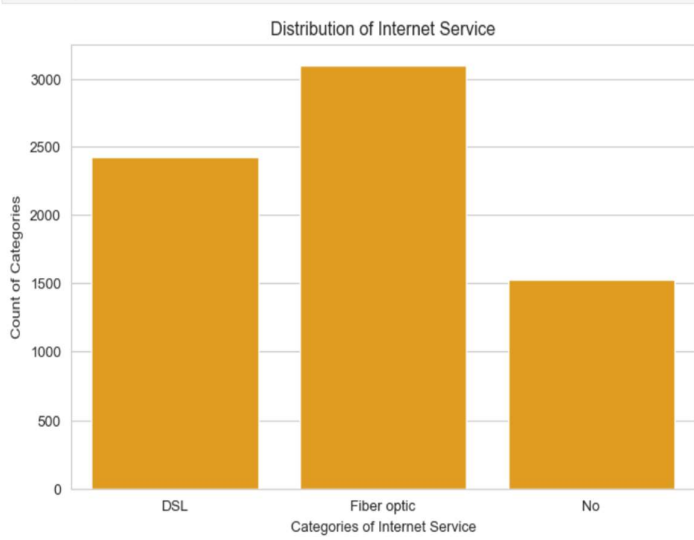
Bar-Plot:

jupyter Untitled19 Last Checkpoint: 15 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

[11]:
```
### Bar Plot for 'InternetService'
plt.figure(figsize=(8, 5))
sns.countplot(x='InternetService', data=churn, color='orange')
plt.xlabel('Categories of Internet Service')
plt.ylabel('Count of Categories')
plt.title('Distribution of Internet Service')
plt.show()
```



[ ]:

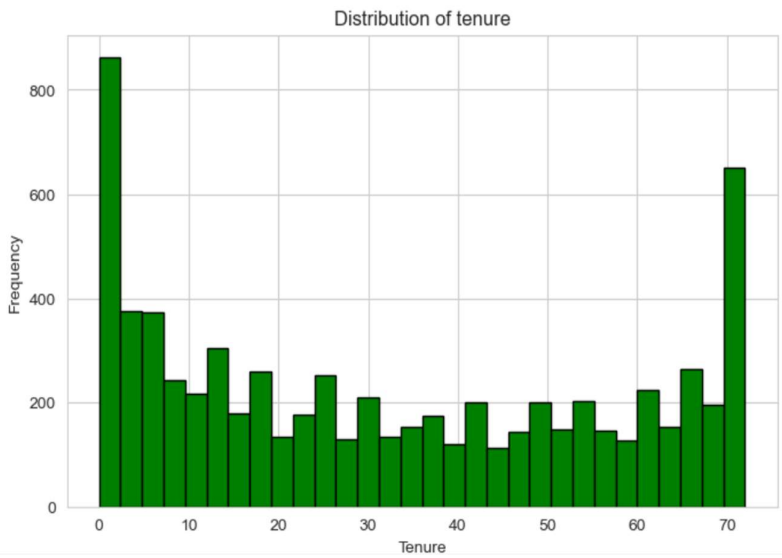
Histogram:

jupyter Untitled19 Last Checkpoint: 17 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

[17]:
```
### Histogram for 'tenure'
plt.figure(figsize=(8, 5))
plt.hist(churn['tenure'], bins=30, color='green', edgecolor='black')
plt.xlabel('Tenure')
plt.ylabel('Frequency')
plt.title('Distribution of tenure')
plt.show()
```



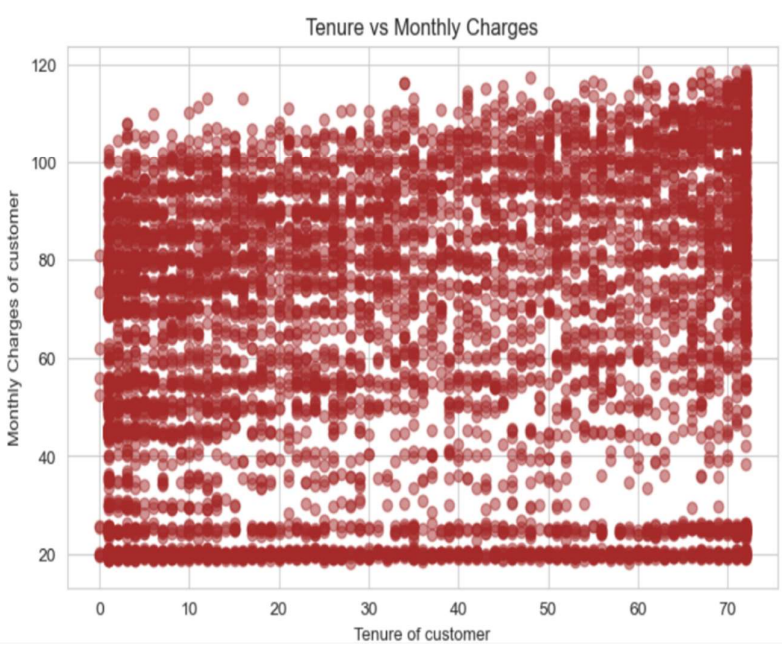
Scatter Plot:

jupyter Untitled19 Last Checkpoint: 17 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

[19]:
```
### Scatter Plot between 'MonthlyCharges' and 'tenure'
plt.figure(figsize=(8, 5))
plt.scatter(churn['tenure'], churn['MonthlyCharges'], color='brown', alpha=0.5)
plt.xlabel('Tenure of customer')
plt.ylabel('Monthly Charges of customer')
plt.title('Tenure vs Monthly Charges')
plt.show()
```



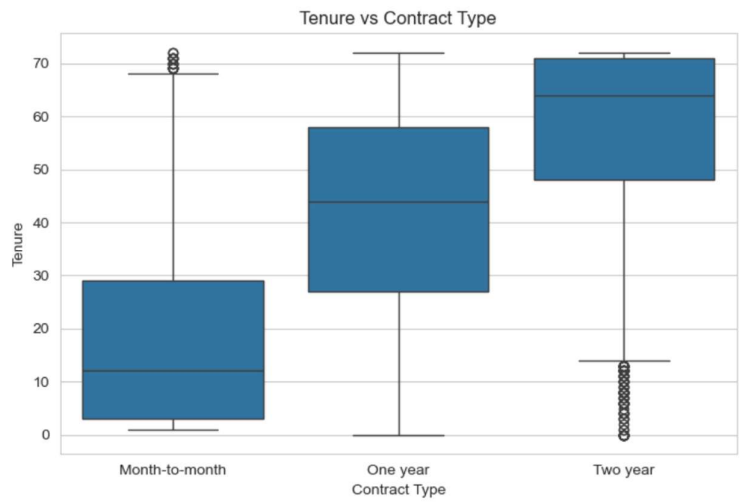
Box Plot:

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

Tenure of customer

[21]:
```python
### Box Plot between 'tenure' & 'Contract'
plt.figure(figsize=(8, 5))
sns.boxplot(x='Contract', y='tenure', data=churn)
plt.xlabel('Contract Type')
plt.ylabel('Tenure')
plt.title('Tenure vs Contract Type')
plt.show()
```



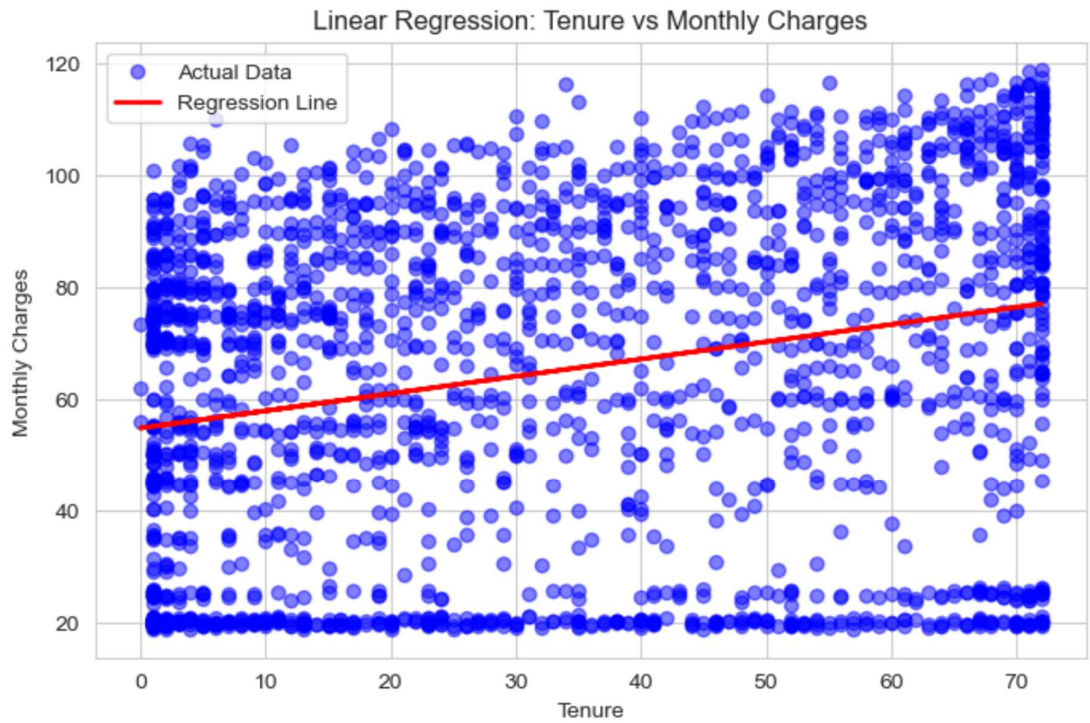
3.Linear Regression:

Start Course | Intellipaat | Start Course | Intellipaat | Start Course | Intellipaat | Untitled19

localhost:8888/notebooks/Untitled19.ipynb?

[25]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Selecting the independent (X) and dependent (y) variables
X = churn[['tenure']]  # Independent variable
y = churn['MonthlyCharges']  # Dependent variable

# Splitting the dataset into training (70%) and testing (30%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

# Initializing and training the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting values on the test set
y_pred = model.predict(X_test)

# Calculating error in prediction
error = y_test - y_pred  # Difference between actual and predicted values

# Calculating Root Mean Square Error (RMSE)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Display results
print(f"Root Mean Square Error (RMSE): {rmse}")

# Plotting the regression line
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, color='blue', label="Actual Data", alpha=0.5)
plt.plot(X_test, y_pred, color='red', linewidth=2, label="Regression Line")
plt.xlabel('Tenure')
plt.ylabel('Monthly Charges')
plt.title('Linear Regression: Tenure vs Monthly Charges')
plt.legend()
plt.show()
```

```
Root Mean Square Error (RMSE): 29.07936015646814
```



Logistic Regression:

Simple Logistic Regression:

jupyter Untitled19 Last Checkpoint: 27 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

[27]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score

# Convert 'Churn' column to binary (Yes -> 1, No -> 0)
churn['Churn'] = churn['Churn'].map({'Yes': 1, 'No': 0})

# Selecting features and target variable
X = churn[['MonthlyCharges']]  # Independent variable
y = churn['Churn']  # Dependent variable

# Splitting the dataset (65% training, 35% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=42)

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

# Predict values on test set
y_pred = model.predict(X_test)

# Build confusion matrix and compute accuracy
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print("Confusion Matrix:\n", conf_matrix)
print(f"Accuracy Score: {accuracy:.2f}")
```

```
Confusion Matrix:
 [[1797    0]
 [ 669    0]]
Accuracy Score: 0.73
```

Multiple Logistic Regression :

jupyter Untitled19 Last Checkpoint: 28 minutes ago

File Edit View Run Kernel Settings Help

Code JupyterLab Python [conda env:base]

[29]:
```python
# Selecting multiple independent variables
X_multi = churn[['tenure', 'MonthlyCharges']]
y_multi = churn['Churn']

# Splitting the dataset (80% training, 20% testing)
X_train_multi, X_test_multi, y_train_multi, y_test_multi = train_test_split(X_multi, y_multi, test_size=0.20, random_state=42)

# Initialize and train logistic regression model
model_multi = LogisticRegression()
model_multi.fit(X_train_multi, y_train_multi)

# Predict values on test set
y_pred_multi = model_multi.predict(X_test_multi)

# Build confusion matrix and compute accuracy
conf_matrix_multi = confusion_matrix(y_test_multi, y_pred_multi)
accuracy_multi = accuracy_score(y_test_multi, y_pred_multi)

print("Confusion Matrix (Multiple Regression):\n", conf_matrix_multi)
print(f"Accuracy Score (Multiple Regression): {accuracy_multi:.2f}")
```

```
Confusion Matrix (Multiple Regression):
 [[944  92]
 [193 180]]
Accuracy Score (Multiple Regression): 0.80
```

[ ]:

Decision Tree:

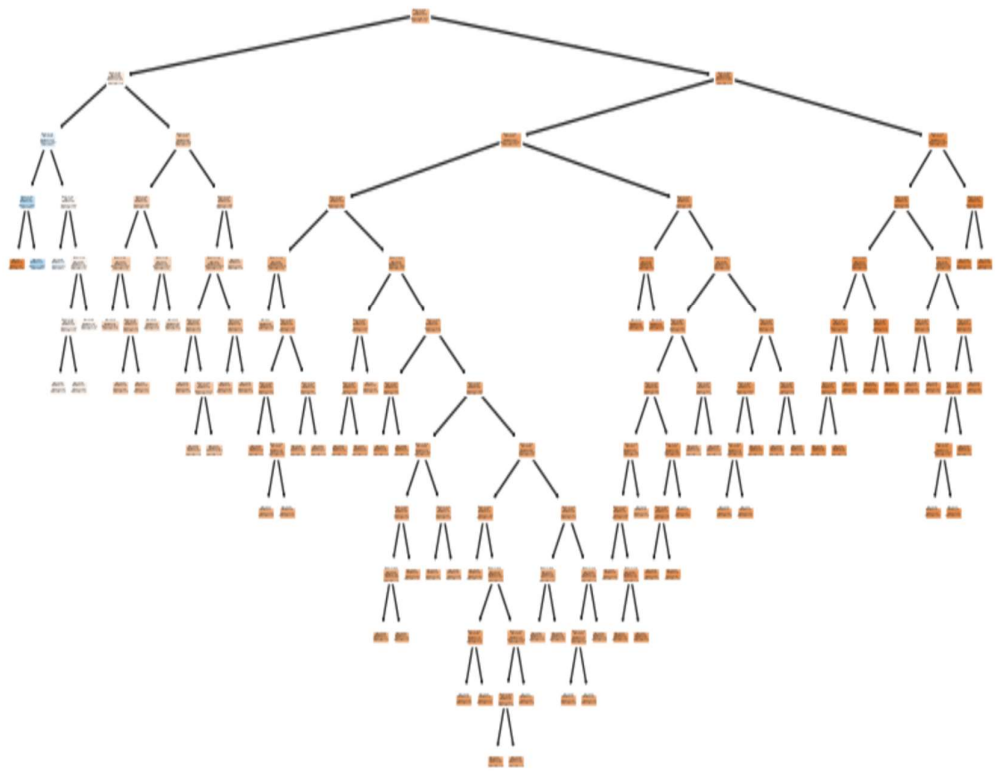
jupyter Untitled19 Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

[57]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn import tree

# Convert 'Churn' column to binary (Yes -> 1, No -> 0)
churn['Churn'] = churn['Churn'].map({'Yes': 1, 'No': 0})

# Selecting independent variable (X) and dependent variable (y)
X = churn[['tenure']]  # Independent variable
y = churn['Churn']  # Dependent variable

# Splitting the dataset (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

# Initialize and train Decision Tree model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict values on test set
y_pred = model.predict(X_test)

# Build confusion matrix and compute accuracy
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print("Confusion Matrix:\n", conf_matrix)
print(f"Accuracy Score: {accuracy:.2f}")

# Visualizing the Decision Tree
plt.figure(figsize=(10,5))
tree.plot_tree(model, feature_names=['tenure'], class_names=['No Churn', 'Churn'], filled=True)
plt.show()
```

```
Confusion Matrix:
 [[951  85]
 [257 116]]
Accuracy Score: 0.76
```



Random Forest:

jupyter Untitled19 Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python [conda env:base] *

[128]:
```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Convert categorical column 'Churn' into binary (Yes -> 1, No -> 0)
churn['Churn'] = churn['Churn'].map({'Yes': 1, 'No': 0})

# Select independent variables (tenure and MonthlyCharges)
X = churn[['tenure', 'MonthlyCharges']]
y = churn['Churn']

# Split dataset into 70% training and 30% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

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

# Predict on test set
y_pred = rf_model.predict(X_test)

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Compute accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print results
print("Confusion Matrix:\n", conf_matrix)
print("Accuracy Score:", accuracy)
```

```
Confusion Matrix:
 [[1341  198]
 [ 308  266]]
Accuracy Score: 0.7605300520586843
```