

SpeakX Data Science Assignment

Name: Pullela Harish Chowdary

Reg.No: 12002666

University: Lovely Professional University

Assignment: Predicting Customer Churn in a Telecommunications Company

Objective: The primary objective of this project is to develop a predictive model that can identify customers at risk of churning, enabling the company to take proactive measures to retain them.

Implementation starts here

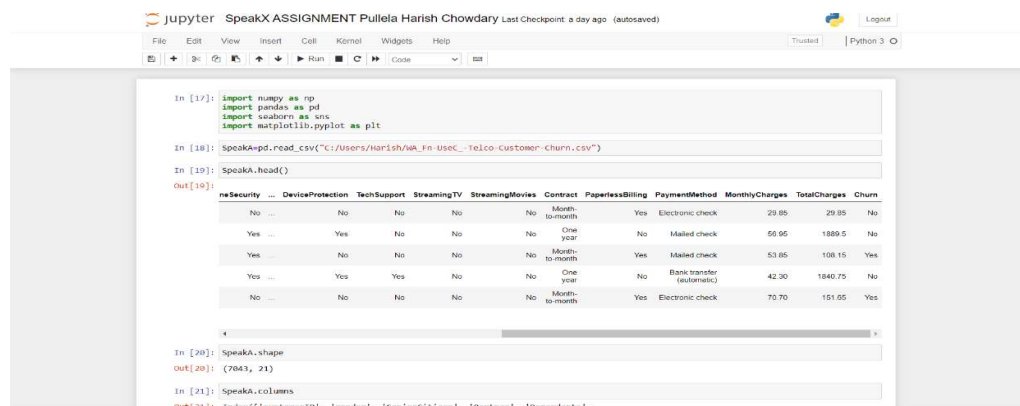
Tasks:

1) Data Collection and Preprocessing:

Data Set: <https://www.kaggle.com/datasets/blastchar/telco-customer-churn>

Steps for importing and loading the dataset in Python:

- ➔ Imported numpy, pandas, matplotlib, seaborn
- ➔ For loading the dataset, the data set should be in the current directory, and using pandas reading the CSV file is done
- ➔ I explored **shape** – shows how many columns and rows are there, **columns** – shows column headers, **info** – shows columns with their datatype, **describe** – shows columns of continuous variables and explores count, mean, std, min, max, etc, **is.na** – shows whether any null values in the data set are in a Boolean form. The below screenshots are representations for above mentioned implementation



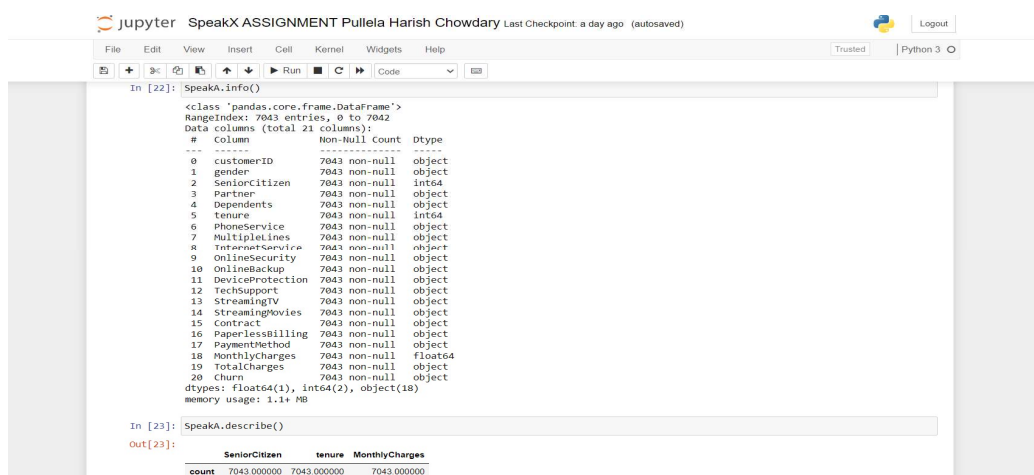
```

In [17]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

In [18]: SpeakA=pd.read_csv("C:/Users/Harish/NA_In-use/_Telco-customer-Churn.csv")

In [19]: SpeakA.head()
Out[19]:
OnlineSecurity  ... DeviceProtection  TechSupport  StreamingTV  StreamingMovies  Contract  PaperlessBilling  PaymentMethod  MonthlyCharges  TotalCharges  Churn
0  No  ... No  No  No  No  No  Month-  Electronic check  29.85  29.85  No
1  Yes  ... Yes  No  No  No  No  One-  Mailed check  55.95  1889.5  No
2  Yes  ... No  No  No  No  No  Month-  Mailed check  53.85  108.15  Yes
3  Yes  ... Yes  No  No  No  No  One-  Bank transfer  42.30  1840.75  No
4  No  ... No  No  No  No  No  Month-  Electronic check  70.70  151.55  Yes

```



```

In [22]: SpeakA.info()
Out[22]:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ---
 0   customerID            7043 non-null  object
 1   gender                7043 non-null  object
 2   SeniorCitizen         7043 non-null  int64
 3   Partner               7043 non-null  object
 4   Dependents            7043 non-null  object
 5   tenure                7043 non-null  int64
 6   PhoneService          7043 non-null  object
 7   MultipleLines         7043 non-null  object
 8   InternetService       7043 non-null  object
 9   OnlineSecurity        7043 non-null  object
10  OnlineBackup          7043 non-null  object
11  DeviceProtection      7043 non-null  object
12  TechSupport           7043 non-null  object
13  StreamingTV           7043 non-null  object
14  StreamingMovies       7043 non-null  object
15  Contract              7043 non-null  object
16  PaperlessBilling      7043 non-null  object
17  PaymentMethod         7043 non-null  object
18  MonthlyCharges        7043 non-null  float64
19  TotalCharges          7043 non-null  object
20  churn                 7043 non-null  object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB

In [23]: SpeakA.describe()
Out[23]:
SeniorCitizen    tenure    MonthlyCharges
count  7043.000000  7043.000000  7043.000000

```

- ➔ I observed that column **TotalCharges** has some blank rows that I have replaced with nan. And then replace those nan values with the **mean of TotalCharges**

```
jupyter SpeakX ASSIGNMENT Pullela Harish Chowdary Last Checkpoint: a day ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [27]: SpeakA['TotalCharges'] = SpeakA['TotalCharges'].replace(' ', np.nan, regex=True)
In [28]: SpeakA['TotalCharges'] = pd.to_numeric(SpeakA['TotalCharges'])
        mean_TotalCharges = SpeakA['TotalCharges'].mean()
        SpeakA['TotalCharges'].fillna(value=mean_TotalCharges, inplace=True)
In [29]: SpeakA.head(400)
Out[29]:
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	DeviceProtection	TechSu
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No
1	5575-GNVCDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes
2	9898-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No
...
485	7971-HLVXI	Male	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	No
486	9094-AZBWK	Female	0	No	No	15	Yes	Yes	Fiber optic	No	...	No

- ➔ Now I explored categorical variables in the dataset where with function **.value_counts()** it will print the count of values in a particular column of categories that column has. I have observed columns like **'OnlineSecurity'**, **'MultipleLines'**, **'OnlineBackup'**, **'DeviceProtection'**, **'TechSupport'**, **'StreamingTV'**, and **'StreamingMovies'** have some categories like **No Internet service** and **No phone service** along with Yes and No. So I replaced No Internet service and No Phone service with No because maybe they belong to the No category. It will have binary values for those columns

```
jupyter SpeakX ASSIGNMENT Pullela Harish Chowdary Last Checkpoint: a day ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [32]: SpeakA['gender'].value_counts()
Out[32]: Male      3555
        Female    3488
        Name: gender, dtype: int64

In [34]: SpeakA['SeniorCitizen'].value_counts()
Out[34]: 0      5901
        1      1142
        Name: SeniorCitizen, dtype: int64

In [35]: SpeakA['Partner'].value_counts()
Out[35]: No       3641
        Yes      3482
        Name: Partner, dtype: int64

In [36]: SpeakA['Dependents'].value_counts()
Out[36]: No       4933
        Yes      2110
        Name: Dependents, dtype: int64

In [37]: SpeakA['PhoneService'].value_counts()
Out[37]: Yes       6361
        No        682
        Name: PhoneService, dtype: int64

In [39]: SpeakA['OnlineSecurity'].value_counts()
Out[39]: No          3498
        Yes         2019
        No internet service 1526
        Name: OnlineSecurity, dtype: int64
```

```
jupyter SpeakX ASSIGNMENT Pullela Harish Chowdary Last Checkpoint: a day ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [52]: SpeakA['churn'].value_counts()
Out[52]: No       5174
        Yes      1869
        Name: churn, dtype: int64

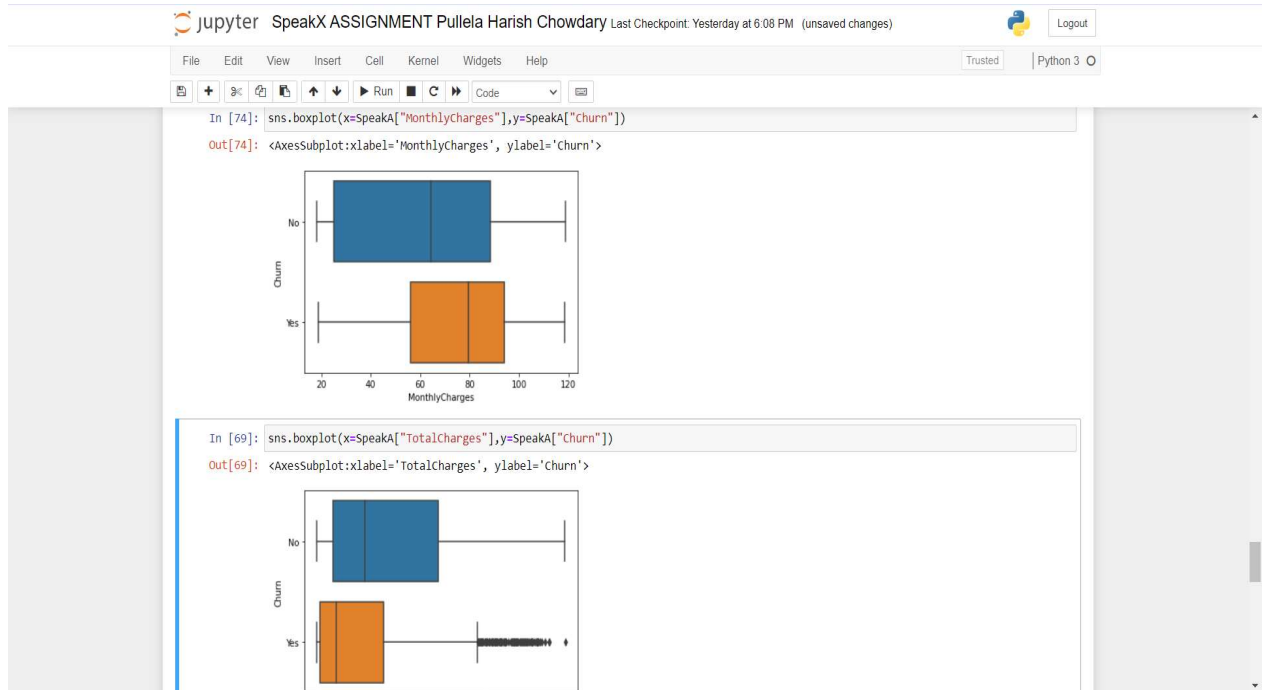
In [53]: SpeakA['OnlineSecurity'] = SpeakA['OnlineSecurity'].replace('No internet service', 'No', regex=True)
In [55]: SpeakA['MultipleLines'] = SpeakA['MultipleLines'].replace('No phone service', 'No', regex=True)
In [56]: SpeakA['OnlineBackup'] = SpeakA['OnlineBackup'].replace('No internet service', 'No', regex=True)
In [57]: SpeakA['DeviceProtection'] = SpeakA['DeviceProtection'].replace('No internet service', 'No', regex=True)
In [58]: SpeakA['TechSupport'] = SpeakA['TechSupport'].replace('No internet service', 'No', regex=True)
In [59]: SpeakA['StreamingTV'] = SpeakA['StreamingTV'].replace('No internet service', 'No', regex=True)
In [60]: SpeakA['StreamingMovies'] = SpeakA['StreamingMovies'].replace('No internet service', 'No', regex=True)
```

Above mentioned steps are for this task - **Preprocess the data to handle missing values, encode categorical variables, and prepare it for analysis. Encode categorical variable at prediction time I going to do**

2) Exploratory Data Analysis (EDA):

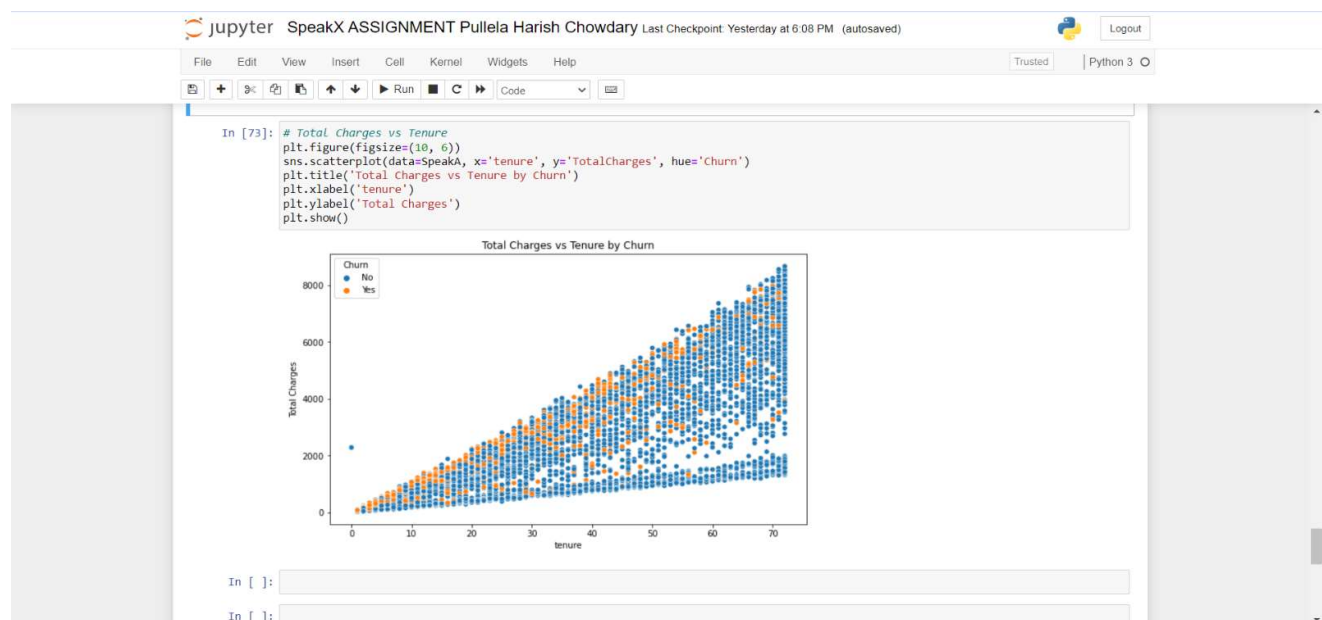
Perform EDA on the dataset to understand customer behavior and factors influencing churn. Visualize key findings using appropriate graphs and charts.

i) Utilizing Box plot for showing outliers of Churn by tenure, TotalCharges, MontlyCharges



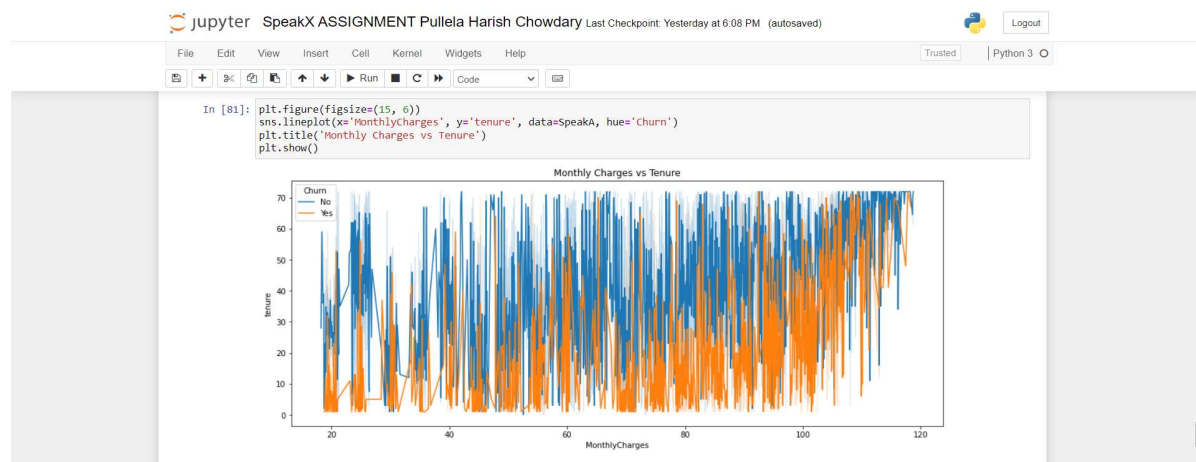
Conclusion: I have observed that Yes of Churn means Churn users have outliers in tenure and TotalCharges

ii) Utilizing a scatter plot to show relationship between tenure and TotalCharges based on Churn



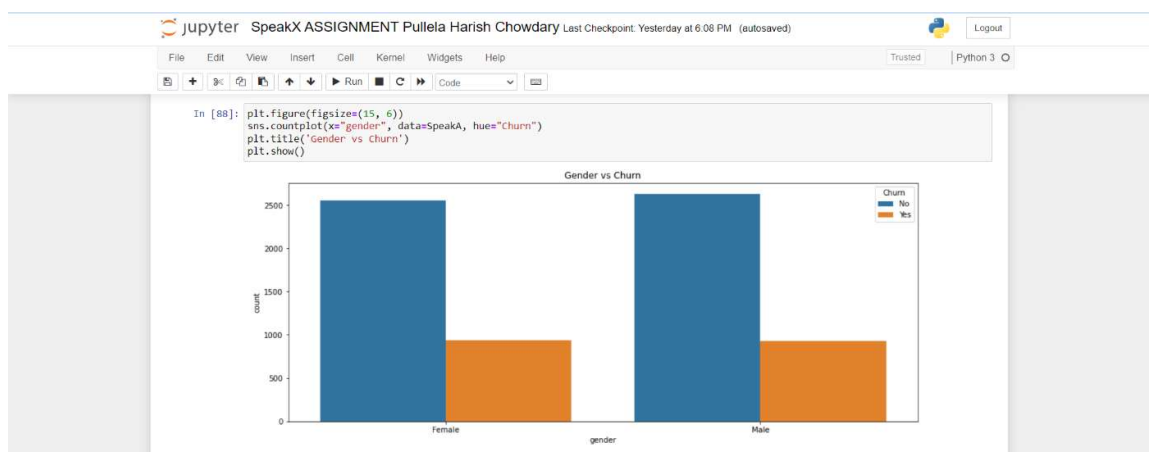
Conclusion: There is a positive linear relation between tenure and TotalCharges

iii) Utilizing a Line chart to show Monthly Charges vs Tenure by Churn

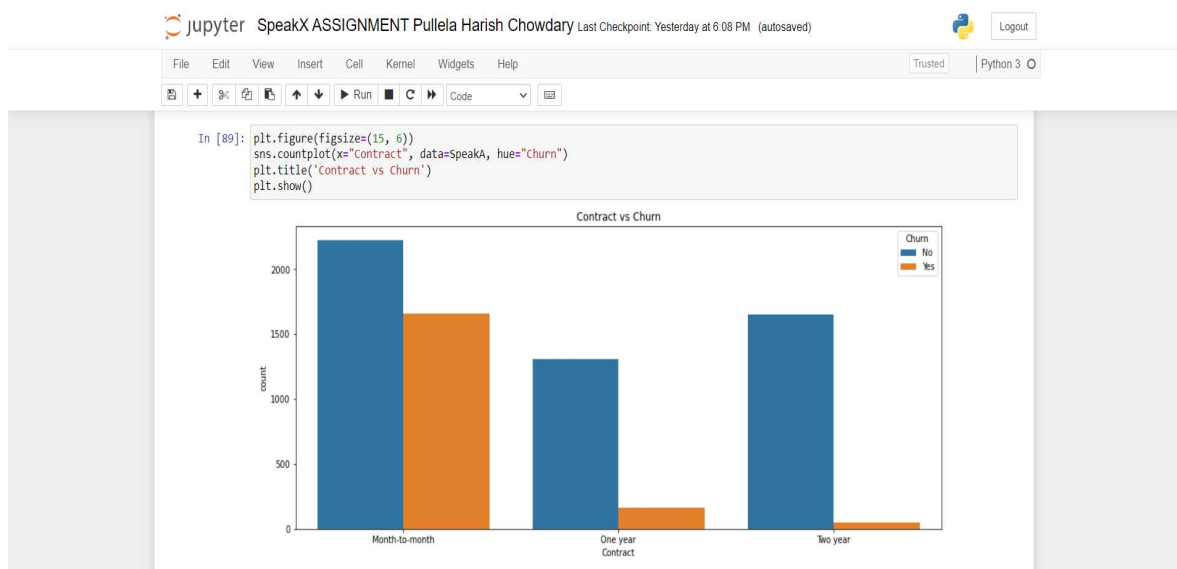


Conclusion: (No) of Churn has more tenure and monthly charges

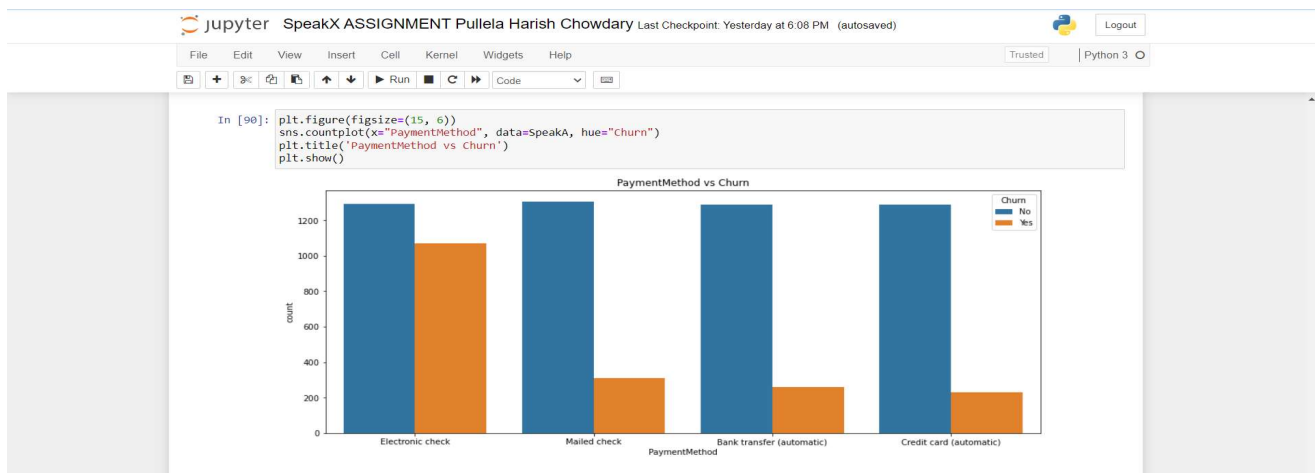
iv) Utilizing a count plot or showing a bar plot where the count of values for gender, internet service, contract, and payment method based on the Churn column



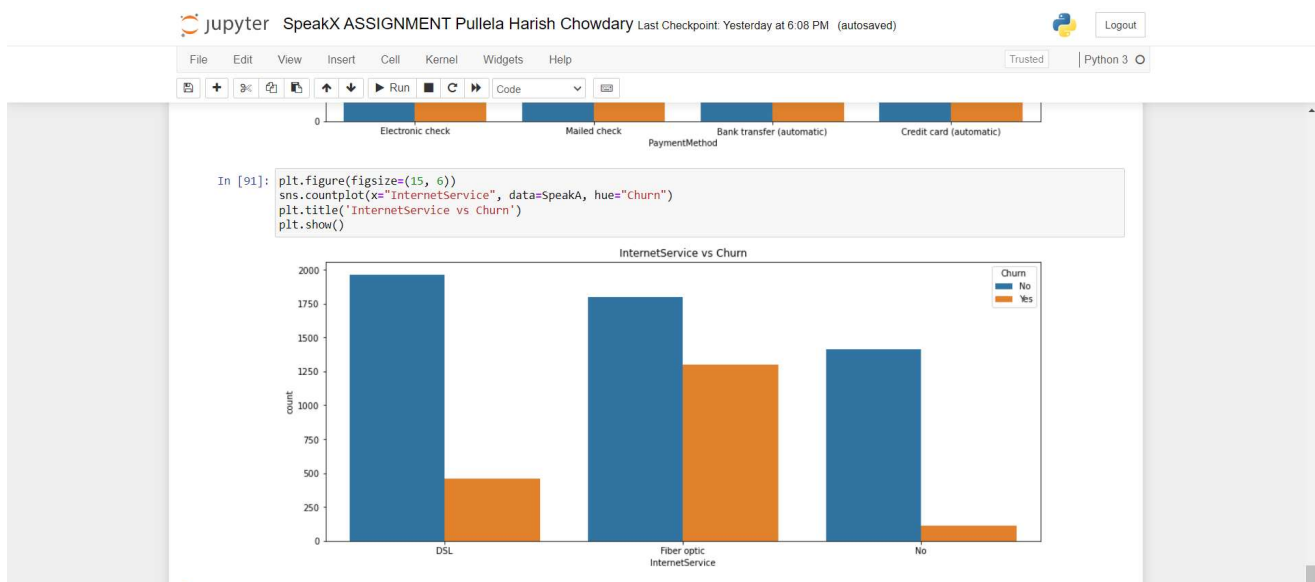
Conclusion: Almost similar distribution between female and male count and Based on Churn as No category has more value distribution between females and males. No Churn users have more distribution



Conclusion: Month-to-month contract users are more than one year contract and two-year. Based on Churn too month-to-month users have more users

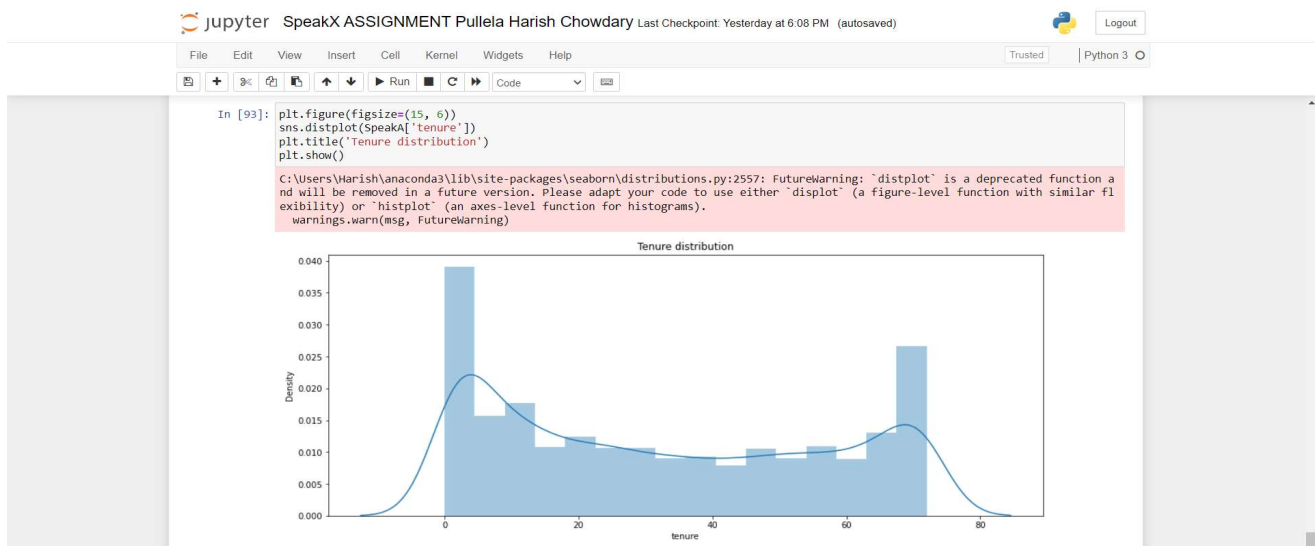


Conclusion: Most users have opted electronic check payment option. (No) category churn users have equally opted for all payment method options.



Conclusion: No category of Churn users has DSL internet service. When combined with the yes and no categories of churn users then DSL and Fiber optic internet services have equal distributions.

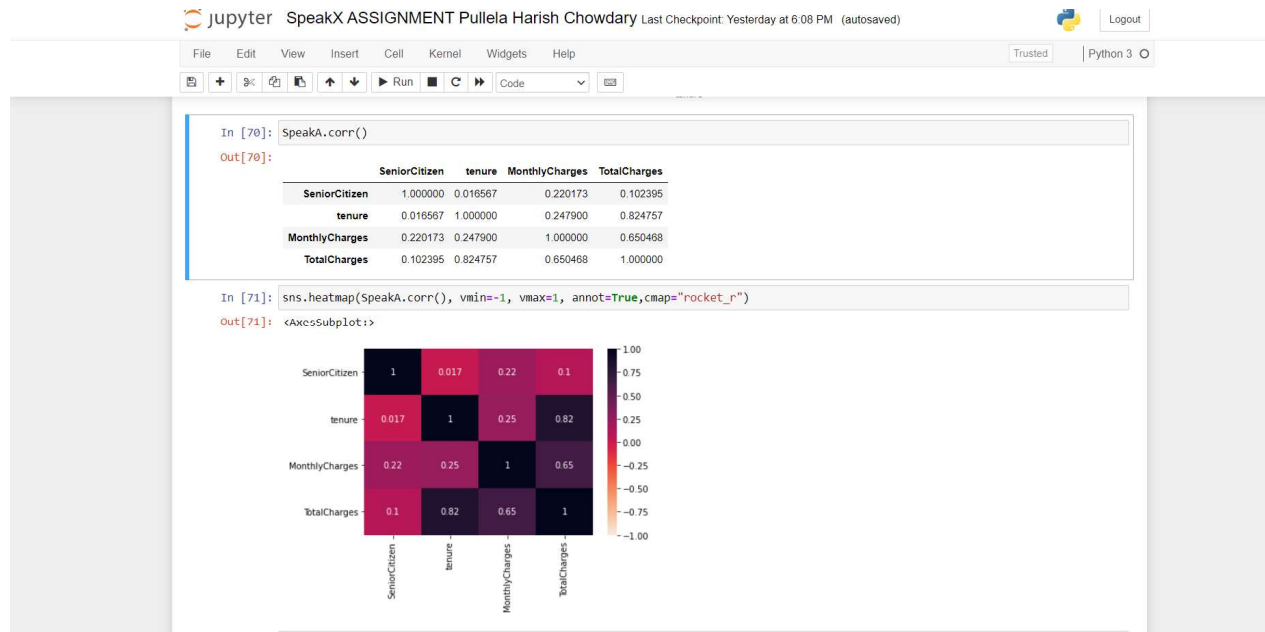
v) Utilizing a distplot graph to show the distribution of the tenure column



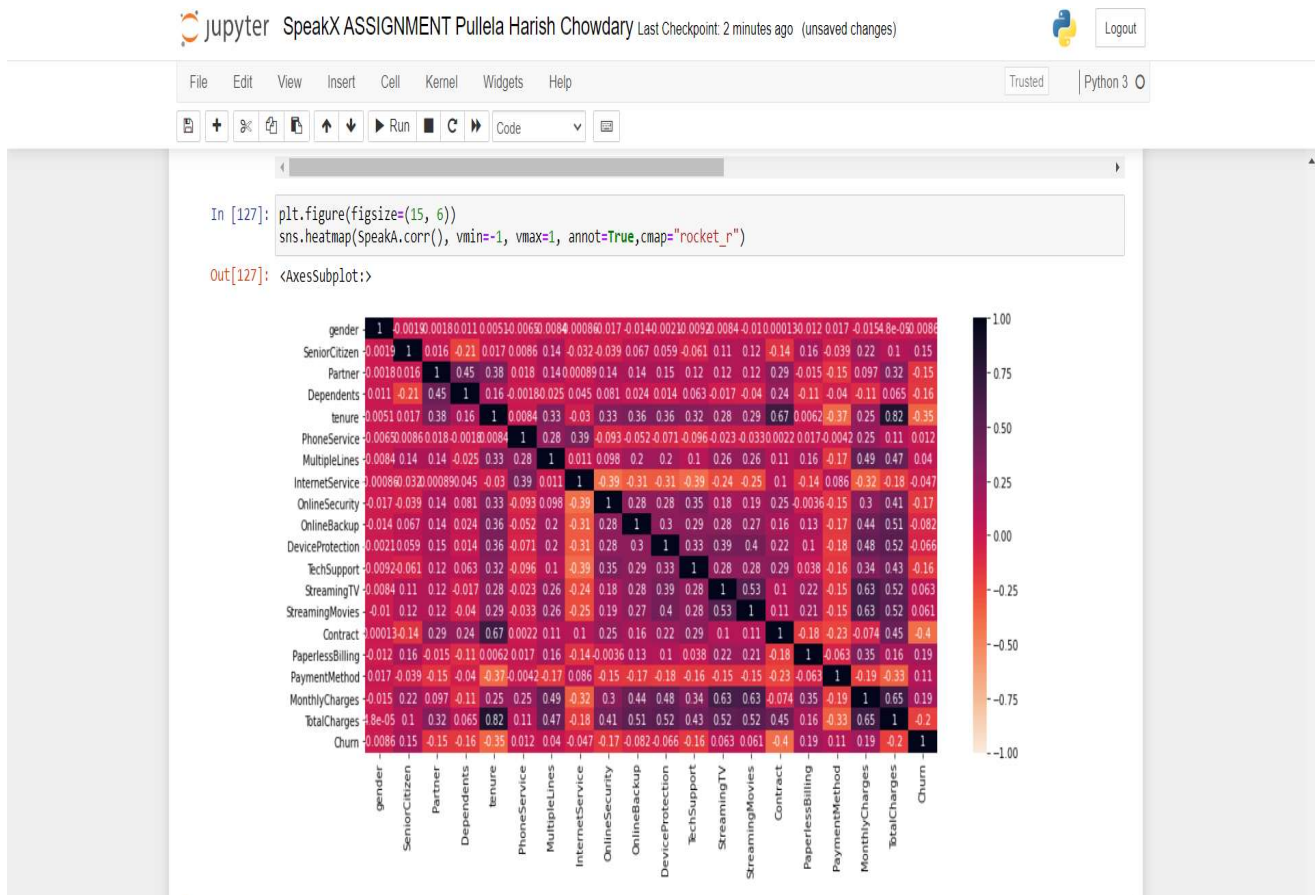
Conclusion: 0-20 density has more value of tenure like 0-20 has more tenure distribution

vi) Utilizing a tree map graph to show the correlation between attributes or variables

Before encoding of categorical variable means only between continuous variables:



After encoding of categorical variables:



3) Feature Engineering:

Create relevant features that can help in predicting churn.

- ➔ Firstly, label encoding is part of feature engineering as features involve preprocessing and cleaning too. So before building a machine learning model label encoding for categorical variables is necessary here in feature engineering label encoding has been done.

```
jupyter SpeakX ASSIGNMENT Pullela Harish Chowdary Last Checkpoint: 8 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [108]: from sklearn import preprocessing
          label_encoder = preprocessing.LabelEncoder()
          SpeakA['gender'] = label_encoder.fit_transform(SpeakA['gender'])

In [109]: SpeakA['Partner'] = label_encoder.fit_transform(SpeakA['Partner'])

In [110]: SpeakA['Dependents'] = label_encoder.fit_transform(SpeakA['Dependents'])

In [111]: SpeakA['PhoneService'] = label_encoder.fit_transform(SpeakA['PhoneService'])

In [112]: SpeakA['MultipleLines'] = label_encoder.fit_transform(SpeakA['MultipleLines'])

In [113]: SpeakA['InternetService'] = label_encoder.fit_transform(SpeakA['InternetService'])

In [114]: SpeakA['OnlineSecurity'] = label_encoder.fit_transform(SpeakA['OnlineSecurity'])

In [115]: SpeakA['OnlineBackup'] = label_encoder.fit_transform(SpeakA['OnlineBackup'])

In [116]: SpeakA['DeviceProtection'] = label_encoder.fit_transform(SpeakA['DeviceProtection'])

In [117]: SpeakA['TechSupport'] = label_encoder.fit_transform(SpeakA['TechSupport'])

In [118]: SpeakA['StreamingTV'] = label_encoder.fit_transform(SpeakA['StreamingTV'])

In [119]: SpeakA['StreamingMovies'] = label_encoder.fit_transform(SpeakA['StreamingMovies'])

In [120]: SpeakA['Contract'] = label_encoder.fit_transform(SpeakA['Contract'])
```

- ➔ For selecting relevant features from the dataset given we don't require a customer ID column and remaining all are useful. So, I dropped the customer ID column

```
In [128]: SpeakA.drop('customerID', inplace=True, axis=1)

In [129]: SpeakA

Out[129]:
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSu
0	0	0	1	0	1	0	0	0	0	1	0	
1	1	0	0	0	34	1	0	0	1	0	1	
2	1	0	0	0	2	1	0	0	1	1	0	
3	1	0	0	0	45	0	0	0	1	0	1	
4	0	0	0	0	2	1	0	1	0	0	0	
...	
7038	1	0	1	1	24	1	1	0	1	0	1	
7039	0	0	1	1	72	1	1	1	1	0	1	
7040	0	0	1	1	11	0	0	0	1	0	0	
7041	1	1	1	0	4	1	1	1	0	0	0	
7042	1	0	0	0	66	1	0	1	1	0	1	
...	

7043 rows × 20 columns

4) Building the Churn Prediction Model:

Choose and implement a machine learning algorithm for churn prediction. You can consider algorithms like logistic regression, random forests, gradient boosting, or any other suitable models. Train and fine-tune the models using the dataset.

Step1: Data collection and data importing

- ➔ Already before we imported

Step 2: Data Pre-processing and Cleaning

Step 3: Feature Engineering

- ➔ Available on previous pages

Step 4: Machine learning implementation with Model Evaluation

1) Random Forest Algorithm:

i) Importing necessary libraries and giving dependent and independent variable

```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [130]: from sklearn.model_selection import train_test_split
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay, f1_score
          from scipy.stats import randint

In [132]: y = SpeakA.Churn # dependent variable
          y.shape
          x = SpeakA.drop('Churn', axis=1) # independent variable

In [133]: x

Out[133]:
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSu
0	0	0	1	0	1	0	0	0	0	1	0	
1	1	0	0	0	34	1	0	0	1	0	1	
2	1	0	0	0	2	1	0	0	1	1	0	
3	1	0	0	0	45	0	0	0	1	0	1	
4	0	0	0	0	2	1	0	1	0	0	0	
...	
7038	1	0	1	1	24	1	1	0	1	0	1	
7039	0	0	1	1	72	1	1	1	0	1	1	
7040	0	0	1	1	11	0	0	0	1	0	0	
7041	1	1	1	1	4	1	1	1	0	0	0	
7042	1	0	0	0	66	1	0	1	1	0	1	
...	

7043 rows x 19 columns

ii) Splitting the dataset into train and test datasets & Fit using a random classifier:

```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [134]: train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.3)

In [135]: rf = RandomForestClassifier()
          rf.fit(train_x, train_y)

Out[135]: RandomForestClassifier()
```

iii) Predicting using y_pred and Model Evaluation:

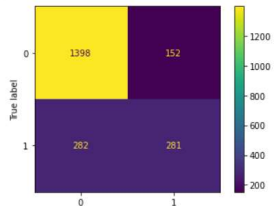
```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [136]: y_pred = rf.predict(test_x)

In [138]: # Random Forest evaluation
          rf_accuracy = accuracy_score(test_y, y_pred)
          rf_precision = precision_score(test_y, y_pred)
          rf_recall = recall_score(test_y, y_pred)
          rf_f1 = f1_score(test_y, y_pred)

          print("Accuracy:", rf_accuracy)
          print("Precision:", rf_precision)
          print("recall:", rf_recall)
          print("f1_score:", rf_f1)

          Accuracy: 0.7946048272598202
          Precision: 0.648960739030023
          recall: 0.4991119005328597
          f1_score: 0.5642570281124498

In [139]: cm = confusion_matrix(test_y, y_pred)
          ConfusionMatrixDisplay(confusion_matrix=cm).plot();
```



Conclusion: Random Forest gave an accuracy of 79% with a precision of 64%, recall of 49%, and F1 score of 56%

Method 2: Identifying best hyperparameters and estimators and fitting into random forest and building using RandomizedSearchCV

```
jupyter SpeakX ASSIGNMENT Pulella Harish Chowdary Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

# Identifying best hyperparameters and estimators and fitting into random forest and building

In [141]: from sklearn.model_selection import RandomizedSearchCV
param_dist = {'n_estimators': randint(50,500), 'max_depth': randint(1,20)}

rf = RandomForestClassifier()

rand_search = RandomizedSearchCV(rf,
                                param_distributions = param_dist,
                                n_iter=5,
                                cv=5)

rand_search.fit(train_x, train_y)

Out[141]: RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_iter=5,
                             param_distributions={'max_depth': <scipy.stats._distn_infrastructure.rv_frozen object at 0x000001D00D2C04C0>,
                             'n_estimators': <scipy.stats._distn_infrastructure.rv_frozen object at 0x000001D00D2C02E0>})

In [142]: best_rf = rand_search.best_estimator_

print('Best hyperparameters:', rand_search.best_params_)

Best hyperparameters: {'max_depth': 11, 'n_estimators': 315}
```

Output: Best hyperparameters: {'max_depth': 11, 'n_estimators': 315}

```
jupyter SpeakX ASSIGNMENT Pulella Harish Chowdary Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [143]: y_pred1 = best_rf.predict(test_x)

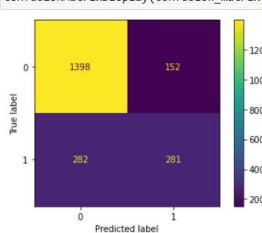
cm = confusion_matrix(test_y, y_pred1)

ConfusionMatrixDisplay(confusion_matrix=cm).plot();

In [144]: # Random Forest evaluation
rf1_accuracy = accuracy_score(test_y, y_pred1)
rf1_precision = precision_score(test_y, y_pred1)
rf1_recall = recall_score(test_y, y_pred1)
rf1_f1 = f1_score(test_y, y_pred1)

print("Accuracy:", rf1_accuracy)
print("Precision:", rf1_precision)
print("recall:", rf1_recall)
print("f1_score:", rf1_f1)

Accuracy: 0.8149085186938003
Precision: 0.694954128440367
recall: 0.5381882770870338
f1_score: 0.6066060606060606
```



Conclusion: Random Forest with best hyperparameter and estimator gave an accuracy of 81% with a precision of 69%, recall of 53%, and F1 score of 60%

2) Logistic Regression Algorithm:

i) Importing necessary libraries and giving dependent and independent variable

```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

# Logistic Regression Algorithm

In [145]: from sklearn.model_selection import train_test_split
          from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay, f1_score
          from sklearn.linear_model import LogisticRegression

In [146]: y = SpeakA.Churn
          y.shape
          x = SpeakA.drop('Churn', axis=1)
```

ii) Splitting the dataset into train and test datasets & Fit using a random classifier:

```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

# Logistic Regression Algorithm

In [145]: from sklearn.model_selection import train_test_split
          from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, ConfusionMatrixDisplay, f1_score
          from sklearn.linear_model import LogisticRegression

In [146]: y = SpeakA.Churn
          y.shape
          x = SpeakA.drop('Churn', axis=1)

In [147]: from sklearn.model_selection import train_test_split
          train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.3)

In [148]: # import the class
          from sklearn.linear_model import LogisticRegression

          # instantiate the model (using the default parameters)
          logreg = LogisticRegression(random_state=16)

          # fit the model with data
          logreg.fit(train_x, train_y)

          y_pred = logreg.predict(test_x)

C:\Users\Harish\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
```

iii) Predicting using y_pred and Model Evaluation:

```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [149]: cm = confusion_matrix(test_y, y_pred)
          ConfusionMatrixDisplay(confusion_matrix=cm).plot();

True label
0
1
Predicted label
0 1
0 1378 153
1 266 316

Accuracy: 0.8017037387600567
Precision: 0.673773987206823
recall: 0.542953264604811
f1_score: 0.6013320647002854
```

Conclusion: Logistic Regression gave an accuracy of 80% with a precision of 67%, recall of 54%, and F1 score of 60%

3) Gradient Boosting Algorithm:

i) Importing necessary libraries and giving dependent and independent variable

```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
# Gradient Boosting Algorithm
In [151]: from sklearn.model_selection import train_test_split
          from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, confusionMatrixDisplay, f1_score
          from sklearn.ensemble import GradientBoostingClassifier

In [152]: y = SpeakA.Churn
          y.shape
          x = SpeakA.drop('churn', axis=1)
```

ii) Splitting the dataset into train and test datasets & Fit using a random classifier:

```
In [153]: train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.3)

In [155]: GB = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0).fit(train_x, train_y)
          y_pred = GB.predict(test_x)
```

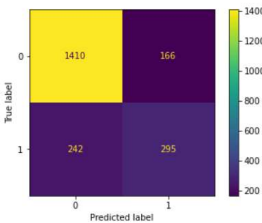
iii) Predicting using y_pred and Model Evaluation:

```
jupyter SpeakX ASSIGNMENT Pulela Harish Chowdary Last Checkpoint: 2 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [156]: cm = confusion_matrix(test_y, y_pred)
          ConfusionMatrixDisplay(confusion_matrix=cm).plot();

In [157]: GB_accuracy = accuracy_score(test_y, y_pred)
          GB_precision = precision_score(test_y, y_pred)
          GB_recall = recall_score(test_y, y_pred)
          GB_f1 = f1_score(test_y, y_pred)

          print("Accuracy:", GB_accuracy)
          print("Precision:", GB_precision)
          print("recall:", GB_recall)
          print("f1_score:", GB_f1)

          Accuracy: 0.8069096071935636
          Precision: 0.6399132321041214
          recall: 0.5493482309124768
          f1_score: 0.591182364729459
```



Conclusion: Gradient Boosting gave an accuracy of 80% with a precision of 63%, recall of 54%, and F1 score of 59%

Challenges faced during the project:

- 1) Identifying blank rows and filling those values
- 2) As there are more categorical variables encoding separately checking count values is made as difficult as individually checking for every categorical variable.