CUSTOMER CHURN IN TELECOM SEGMENT

By

VENKAT SANDEEP REDDY KOTA

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

Project Description:

Businesses often prioritize customer acquisition over retention, even though retaining existing customers is more cost-effective. According to Bain & Company, acquiring a new customer can cost up to five times more than retaining an existing one. Increasing customer retention rates by just 5% can lead to significant profit growth, ranging from 25% to 95%.

Customer churn, also known as customer attrition, is a crucial metric that indicates when customers discontinue their relationship with a company or service. Traditionally, businesses have focused on understanding the reasons behind churn and implementing reactive strategies to address them.

However, what if businesses could predict which customers are likely to churn beforehand and take proactive measures to prevent it? Churn reasons can vary widely, including issues with service quality, customer support delays, pricing, and new market entrants. Often, multiple factors contribute to customer dissatisfaction.

Identifying signals of potential churn and taking pre-emptive action is key to retaining customers. Although once a customer has left, there is no going back, the data they leave behind is invaluable. Analysing this data can provide insights and enable businesses to train churn prediction models, thereby improving future customer experiences through machine learning.

In the telecommunications sector, where vast amounts of customer data are collected, there is significant potential to shift from reactive to proactive churn management. Advanced artificial intelligence and data analytics techniques can leverage this data effectively to predict and mitigate churn.

To demonstrate this concept, I will use a customer dataset obtained from an anonymous carrier, taken from Kaggle . This dataset comprises 7,043 customers and 21 attributes, including personal characteristics, service signatures, and contract details. Notably, the dataset is highly unbalanced, with 5,174 active customers and 1,869 churned customers. The target variable for analysis will be the 'Churn' feature.

The main goal is to develop a machine learning model capable to predict customer churn based on the customer's data available. I will use mainly Python, Pandas, and Scikit-learn libraries for this implementation. The complete code you can find on my Github.

Overview:

The objective of this project is to develop a predictive model to identify customers at risk of churning within the telecom industry. Customer churn, or the rate at which customers discontinue their services with a company, is a critical metric for telecom companies as it directly impacts revenue and profitability. By accurately predicting which customers are likely to churn, telecom companies can proactively implement retention strategies to reduce churn rates and improve customer satisfaction.

Project Goals:

1. Predictive Model Development: Develop machine learning models capable of accurately predicting customer churn based on historical data and customer attributes.

2. Feature Engineering: Conduct thorough feature engineering to identify and select relevant features that have a significant impact on customer churn. This may include factors such as customer demographics, usage patterns, service history, and customer interactions.

3. Model Evaluation: Evaluate the performance of the predictive models using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Identify the most suitable model that provides the highest predictive accuracy for customer churn.

4. Insights Generation: Gain actionable insights from the predictive models to understand the key drivers of customer churn. This involves analysing feature importance, identifying patterns, and understanding customer behaviour to formulate effective retention strategies.

5. Retention Strategy Development: Based on the insights obtained from the predictive models, develop targeted retention strategies to mitigate customer churn. These strategies may include personalized offers, loyalty programs, improved customer service, and proactive communication.

6. Implementation and Monitoring: Implement the developed predictive model and retention strategies within the telecom company's operational framework. Continuously monitor the model's performance and effectiveness of retention strategies to make necessary adjustments and improvements over time.

Expected Outcomes:

- A predictive model capable of accurately identifying customers at risk of churn.

- Actionable insights into the key factors influencing customer churn within the telecom segment.

- Targeted retention strategies to reduce churn rates and improve customer retention.

- Improved customer satisfaction, loyalty, and long-term profitability for the telecom company.

Project Deliverables:

- Predictive model code and documentation.

- Detailed analysis and insights report.

- Recommendations for retention strategies.

- Presentation of findings and recommendations to stakeholders.

To accomplish that, I will go through the below steps:

Exploratory analysis

Data preparation

Train, tune and evaluate machine learning models.

Exploratory analysis

As the purpose of this experiment is to identify patterns that can yield to customers churn, I will be focusing mainly on the churn portion of the dataset for the exploratory analysis.

The customer lifespan (in months) is represented by the feature Tenure and customer churn by the feature Churn, which is the target variable of this experiment. The bar chart below can provide right away a good insight on how churn is distributed across the customer lifespan.

We can see that the largest majority of customers cancel or do not renew their subscription in the first month, totalling 20,3% of customers that defect. Reasons for this high rate can be bad first experience, trial periods, or prepaid accounts that expire automatically if no top-up is done within a predefined period.

Personal Attributes

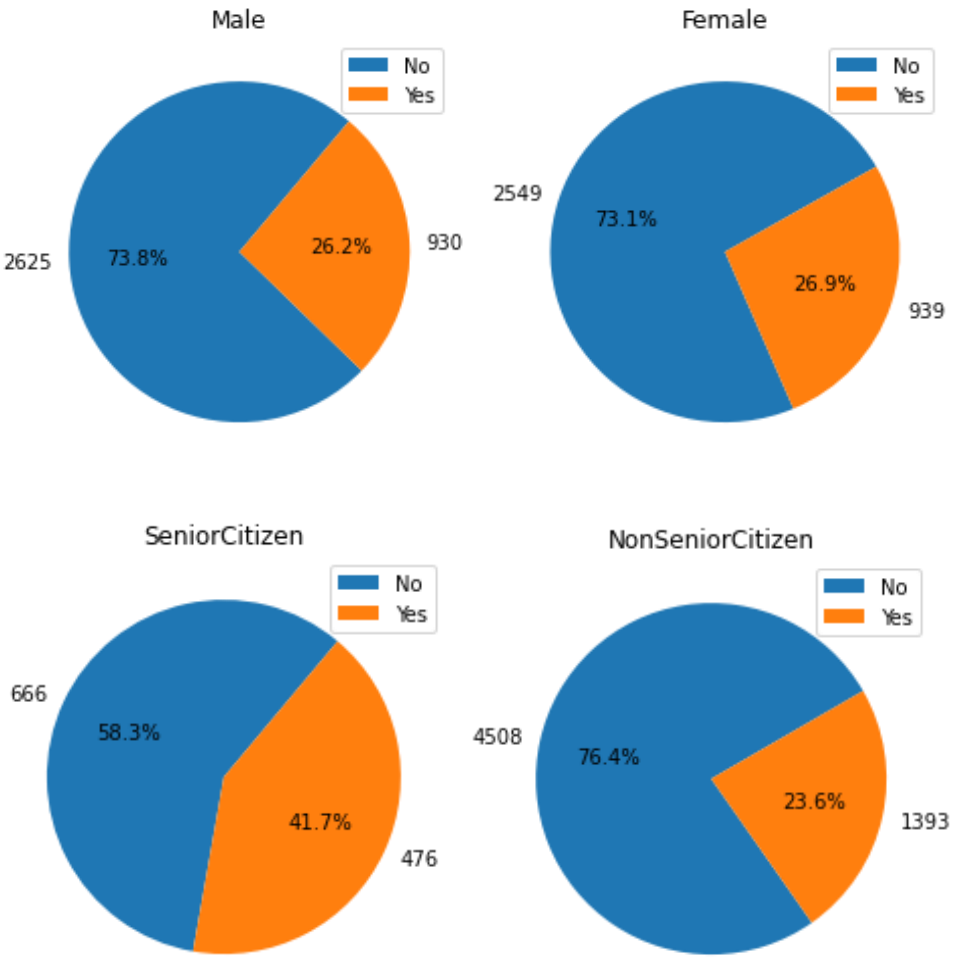
The personal attributes available on the dataset are : Gender, Senior Citizen, Partner, Dependents. Below charts can provide some meaningful insights such as:

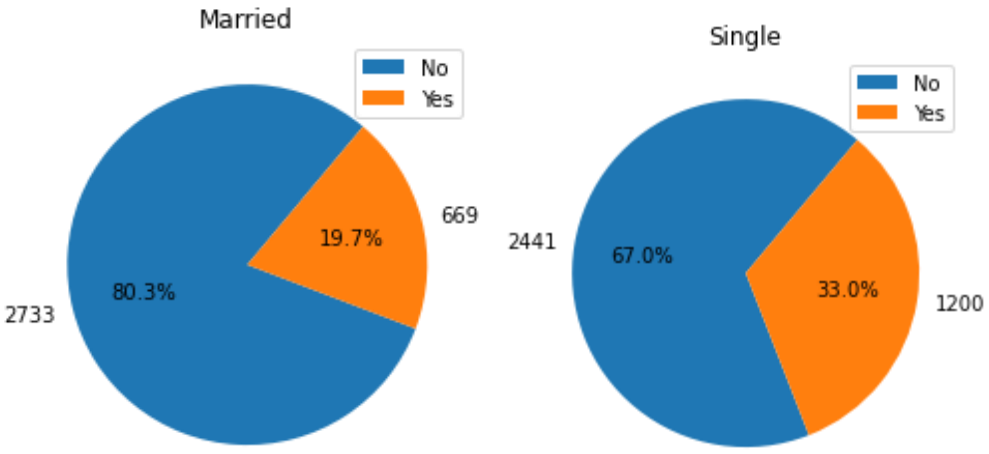
Customers without dependents are four times more likely to churn.

Senior citizens are three times less likely to churn.

Partners are almost two times less likely to leave.

Figure1





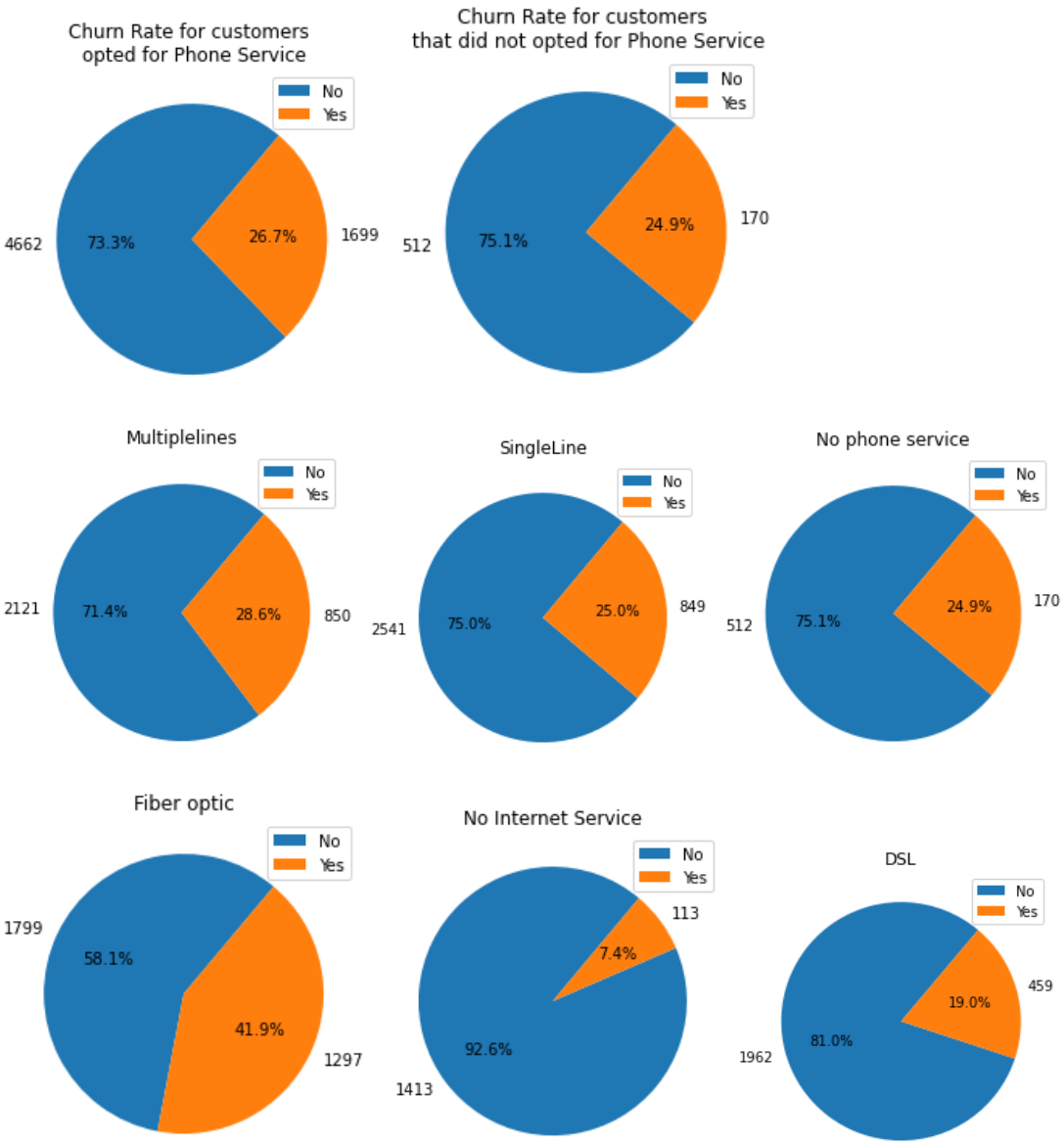
Services Attributes

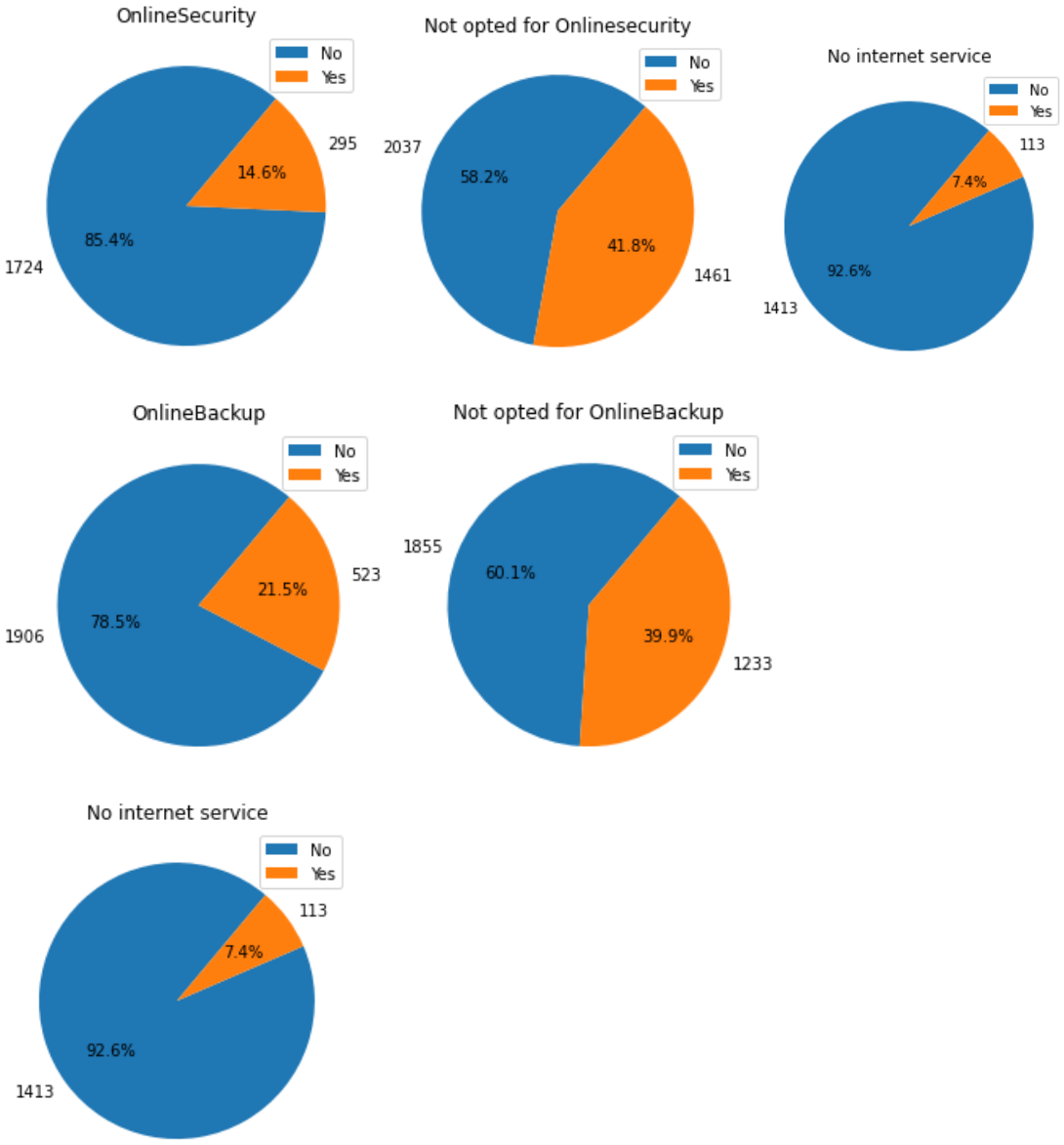
This group of features indicates which kind of services customers subscribe, or if it's not applicable. List of features from this group gathers Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tec Support, Streaming TV and Streaming Movies.

The below charts show the features where high discrepancies between the classes could be noticed. It gives insights regarding which kind of carrier services the customers that are more likely to defeat make use:

- The majority of customers that cancel their subscription have Phone Service enabled.
- Customers that have Internet Service as Fiber-Optic are more likely to cancel than those who have DSL.
- Customers that do not have Online Security, Device Protection, Online Backup, and Tech Support services enabled are more likely to leave.

Figure2


Churn Rate for customers opted for Phone Service
No
Yes
4662
73.3%
26.7%
1699
Churn Rate for customers that did not opted for Phone Service
No
Yes
512
75.1%
24.9%
170
Multiplelines
No
Yes
2121
71.4%
28.6%
850
SingleLine
No
Yes
2541
75.0%
25.0%
849
No phone service
No
Yes
512
75.1%
24.9%
170
Fiber optic
No
Yes
1799
58.1%
41.9%
1297
No Internet Service
No
Yes
113
7.4%
92.6%
1413
DSL
No
Yes
459
19.0%
81.0%
1962



Contract Attributes

This group of features indicates contract aspects and gathers attributes: Contract, Paperless Billing and Payment Method. Below charts give insights regarding the contract aspects that can make a subscriber more likely to churn:

The majority of customers that cancel their subscription have Month-to-month Contract type and Paperless Billing enabled.

Customers that have Payment Method as Electronic Check are more likely to leave.

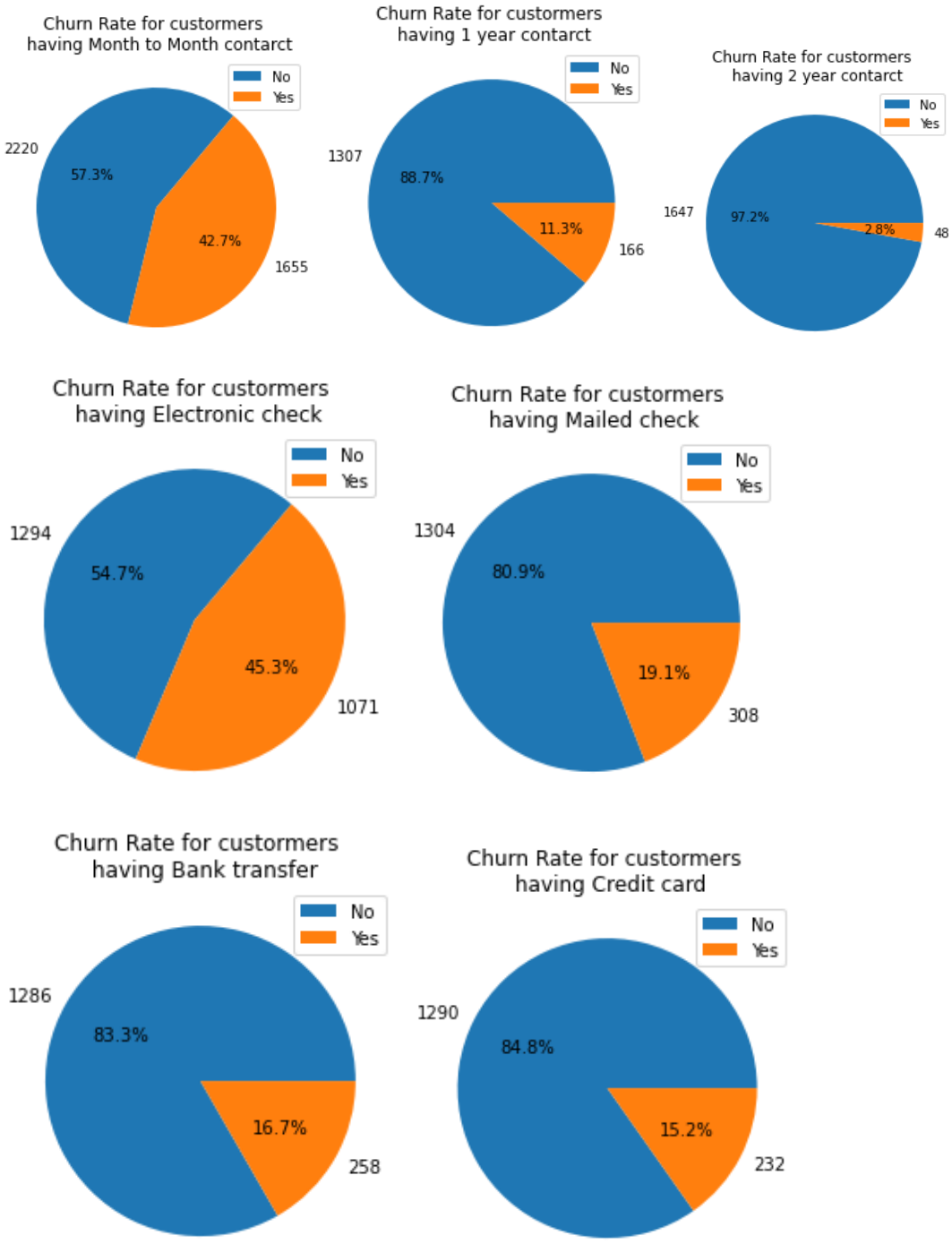
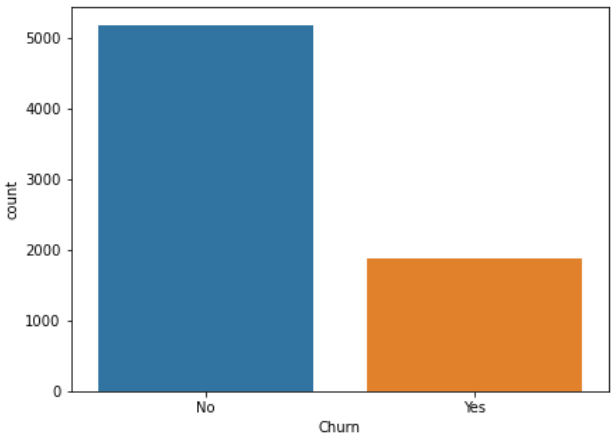
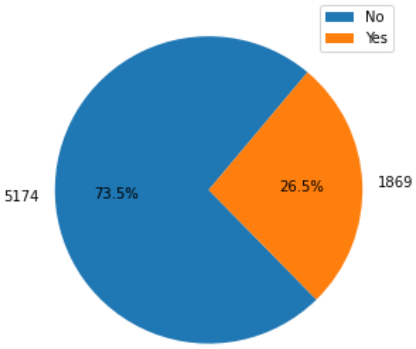

Churn Rate for custormers having Month to Month contarct
No
Yes
2220
57.3%
42.7%
1655
Churn Rate for custormers having 1 year contarct
No
Yes
1307
88.7%
11.3%
166
Churn Rate for custormers having 2 year contarct
No
Yes
1647
97.2%
2.8%
48
Churn Rate for custormers having Electronic check
No
Yes
1294
54.7%
45.3%
1071
Churn Rate for custormers having Mailed check
No
Yes
1304
80.9%
19.1%
308
Churn Rate for custormers having Bank transfer
No
Yes
1286
83.3%
16.7%
258
Churn Rate for custormers having Credit card
No
Yes
1290
84.8%
15.2%
232

Figure 3

Imbalanced Data

The target variable which will be used for ML model training will be Churn. We need to make sure that both Churn classes ('Yes' and 'No') have equal distribution otherwise, it can introduce bias into the model.





Churn Rate

From the graph on the left, we can see that the dataset is highly imbalanced, containing 73.5% of ‘No’ entries — meaning active accounts — against 26.5% of ‘Yes’ — meaning defeated accounts.

There are different techniques to handle imbalanced data. In this particular case, I applied under sampling, which in nutshell means removing some observations of the majority class. After that, the number of entries was equally distributed in 1,406 occurrences in each class for the training dataset.

Feature Encoding

In the realm of machine learning, algorithms are designed to process numerical values exclusively. Thus, a critical aspect of data preprocessing involves converting categorical features into numerical representations, a procedure known as feature encoding.

For categorical attributes with only two classes (binary), such as gender, Senior Citizen, Partner, Dependents, Phone Service, and Paperless Billing.

When dealing with categorical features containing more than two classes, such as Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tec Support, Streaming TV, Streaming Movies, Contract, and Payment Method, I opted for the get dummies method from the Pandas Data Frame library. This approach transforms categorical variables into dummy or indicator variables.

Data Splitting

The subscriber dataset was divided into a 75% training set and a 25% testing set. The training set serves as the basis for constructing the model that the chosen algorithms will utilize when presented with new data. Meanwhile, the test set is reserved for evaluating model performance using predefined metrics.

Model Training, Tuning, and Evaluation

Metrics

In the context of churn prediction, Recall emerges as the primary metric of interest. Recall addresses the question: what proportion of actual churn cases were correctly identified? Essentially, it gauges the percentage of churn instances accurately classified out of the total churn cases, thereby offering insight into the performance of ML classifiers.

For instance, consider a re-engagement campaign offering 1 GB of free data usage. Ideally, one would aim for a high precision rate in targeting recipients of this bonus. In essence, the objective is to minimize the allocation of the bonus to satisfied users and instead focus on those at risk of churning.

Model Selection

To tackle the churn prediction task, four distinct ML algorithms were employed, and their Recall scores compared. These algorithms include:

1. Logistic Regression:

Definition: Logistic Regression is a statistical method used for binary classification tasks. It models the probability that an instance belongs to a particular class.

Advantages:

- Interpretability: Coefficients can indicate the importance of each feature.
- Probabilistic Predictions: Outputs well-calibrated predicted probabilities.
- Efficiency: Computationally efficient and simple to implement.
- No Assumptions of Linearity: Works well even when the relationship between features and the log-odds of the response is not strictly linear.

Disadvantages:

- Assumes Linearity: Assumes a linear relationship between the features and the log-odds of the response.
- Sensitivity to Outliers: Sensitive to outliers in the data.
- Not Suitable for Complex Relationships: May not perform well with complex relationships between features.

2. Decision Trees:

Definition: Decision Trees are tree-like structures where each node represents a feature, each branch represents a decision rule, and each leaf node represents the outcome.

Advantages:

- Interpretability: Easily interpretable and can be visualized.
- Handles Mixed Data: Works with both numerical and categorical data.
- No Data Normalization: Requires little data preparation such as scaling.
- Non-parametric: No assumptions about the underlying data distribution.

Disadvantages:

- Overfitting: Prone to overfitting, especially with deep trees.
- High Variance: Can be sensitive to small variations in the data.
- Not Suitable for Linear Relationships: Not well suited for capturing linear relationships between features.

3. Support Vector Machines (SVM):

Definition: SVM is a binary classification algorithm that finds the hyperplane that best separates classes in a high-dimensional space.

Advantages:

- Effectiveness in High-Dimensional Spaces: Effective in high-dimensional spaces and with complex data.
- Kernel Tricks: Versatile with different kernel functions for non-linear decision boundaries.
- Robustness: Robust against overfitting, especially in high-dimensional spaces.
- Global Optimization: Finds the optimal hyperplane by maximizing the margin.

Disadvantages:

- Computational Complexity: Computationally expensive for large datasets.
- Choice of Kernel: Selection of the appropriate kernel and hyperparameters can be challenging.
- Interpretability: Less interpretable than simpler models like Logistic Regression or Decision Trees.

4. Random Forest:

Definition: Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes.

Advantages:

- Ensemble Learning: Reduces overfitting by averaging multiple trees.
- Works Well with Large Datasets: Works well with large datasets and high-dimensional spaces.
- Feature Importance: Provides an estimate of feature importance.
- Handles Missing Data: Can handle missing values and maintains accuracy even with missing data.

Disadvantages:

- Less Interpretable: Less interpretable than individual decision trees.
- Computational Cost: Can be computationally expensive during training.
- Memory Consumption: Requires more memory due to multiple trees.

5. K-Nearest Neighbors (KNN):

Definition: KNN is a simple and effective algorithm that classifies instances based on the majority class among its k nearest neighbors.

Advantages:

- Simplicity: Simple and easy to understand and implement.
- No Training Phase: No training phase; it uses the entire training dataset during prediction.
- Non-Parametric: No assumptions about the underlying data distribution.
- Works Well with Small Datasets: Works well with small datasets and when decision boundaries are not linear.

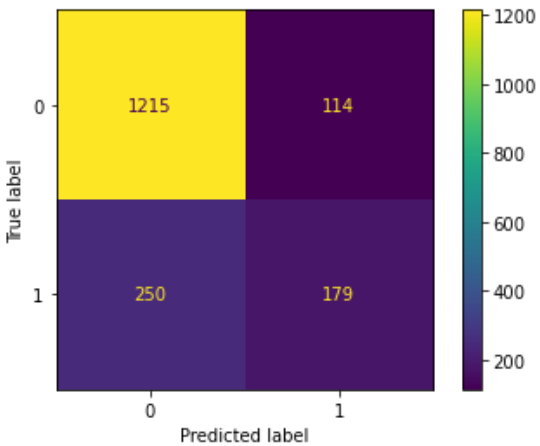
Disadvantages:

- Computationally Expensive: Computationally expensive during prediction, especially with large datasets.
- Sensitive to Noise and Outliers: Sensitive to irrelevant features and the choice of distance metric.
- High Memory Usage: Stores all training data, so memory usage can be high with large datasets.

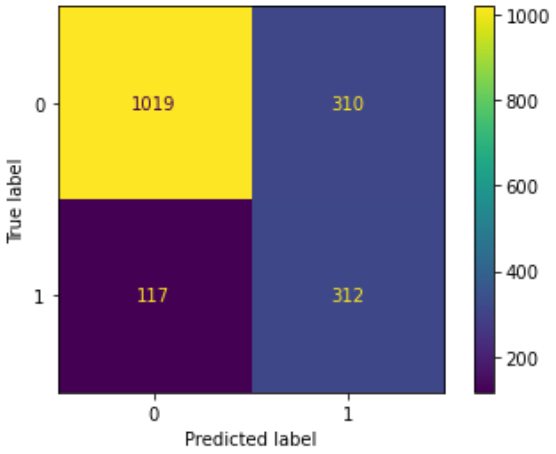
Utilizing the cross-validation technique, which involves partitioning the data into subsets for training and evaluation, the models were put through rigorous testing. Among the models evaluated, K Nearest Neighbour (Recall score of 0.72) ,Logistic Regression (Recall score of 0.71) and SVM (Recall score of 0.75) emerged as top performers. However, there is still room for further optimization and improvement.

Confusion matrix of each model is in the order of:

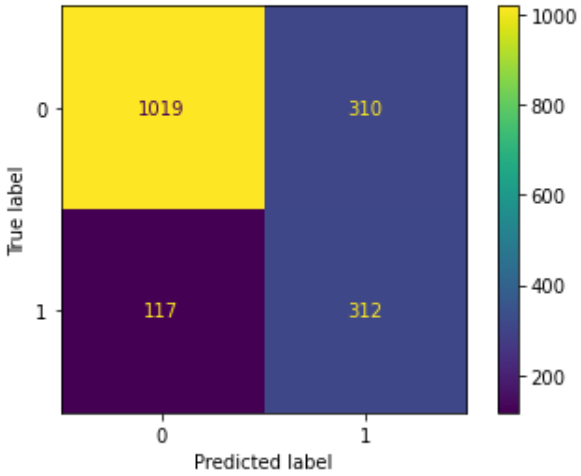
Logistic Regression



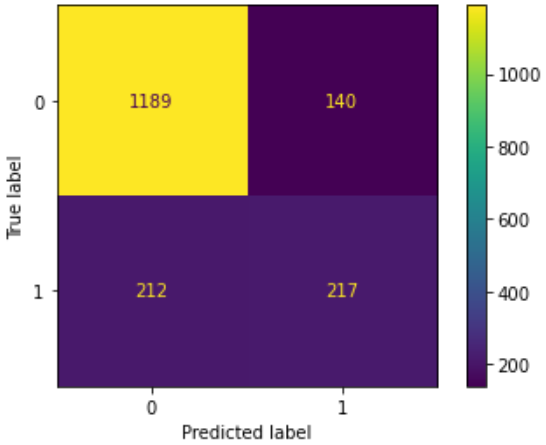
KNN



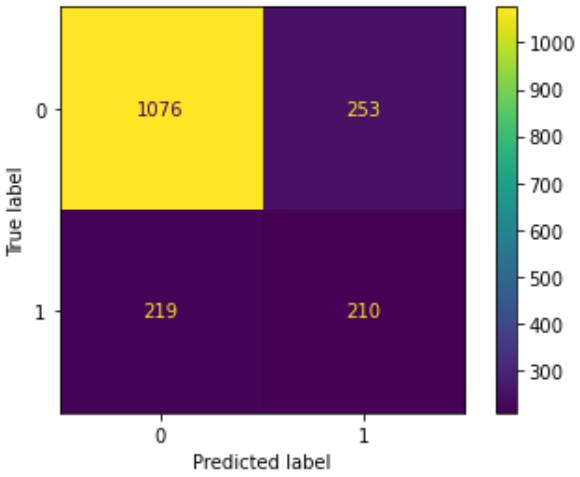
Support Vector Machines



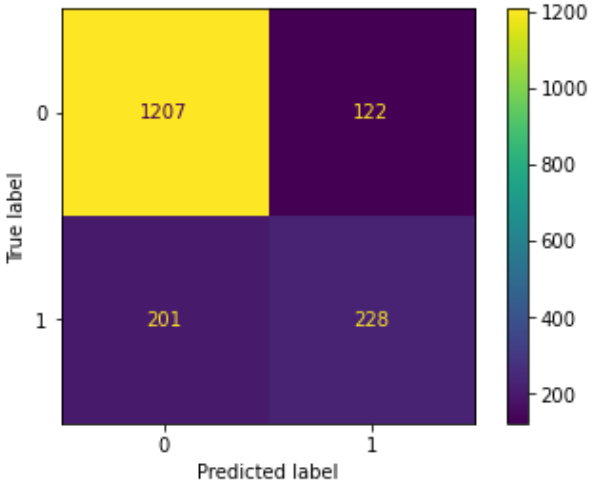
Random Forest



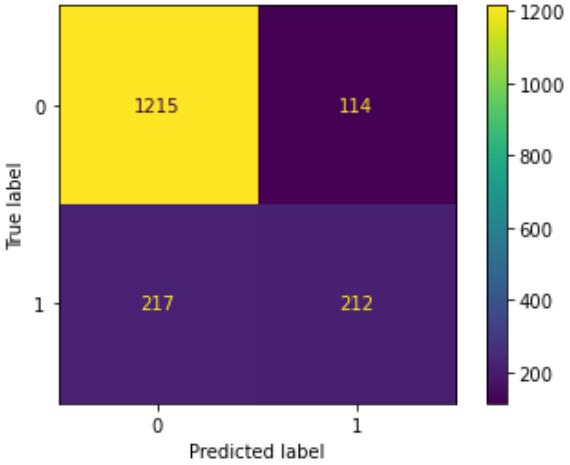
Decision Tree



AdaBoost



Gradient Boosting



Conclusion

No algorithm can achieve perfect churn prediction accuracy. There will always be a balance between precision and recall, necessitating a thorough examination of each classifier's capabilities and limitations to maximize performance.

When the objective is to engage and retain customers to mitigate churn, it is acceptable to interact with individuals incorrectly labelled as 'not churned,' as this does not result in any adverse consequences. In fact, such interactions could potentially enhance customer satisfaction with the service. This underscores the importance of leveraging models that generate actionable insights from meaningful information, thereby delivering immediate value when acted upon appropriately.

After preprocessing the dataset and splitting it into training and testing sets, we trained several classification models, including Gradient Boosting, AdaBoost, Random Forest, Decision Tree Classifier, Support Vector Machines, K Nearest Neighbors, and Logistic Regression. Each model was evaluated using key performance metrics such as accuracy, precision, recall, and F1-score.

Evaluation Metrics:

- Accuracy: The overall proportion of correctly classified instances.
- Precision: The proportion of true positive predictions among all positive predictions.
- Recall: The proportion of true positive predictions among all actual positive instances.
- F1-score: The harmonic mean of precision and recall, providing a balanced measure of the model's performance.

Results and Insights:

- Gradient Boosting and AdaBoost: Achieved the highest accuracy and F1-score among all models, indicating superior predictive performance. These models demonstrated relatively high precision and recall rates, suggesting a good balance between correctly identifying churned customers and minimizing false positives.
- Random Forest: Also performed well, with a slightly lower accuracy compared to boosting algorithms but still providing competitive results in terms of precision and recall.
- Decision Tree Classifier: Despite its simplicity, yielded moderate results with acceptable accuracy and F1-score.

- Support Vector Machines, K Nearest Neighbors, and Logistic Regression:** Showed mixed performance, with varying levels of accuracy, precision, and recall. SVMs achieved a higher recall but lower precision, while KNN and Logistic Regression exhibited balanced precision-recall trade-offs.

Recommendations:

- Based on our findings, we recommend deploying Gradient Boosting or AdaBoost models for churn prediction due to their superior performance
- Telecom companies should leverage these predictive models to identify at-risk customers and implement targeted retention strategies, such as personalized offers, loyalty programs, or proactive customer support.
- Continuous monitoring and refinement of the predictive models are essential to adapt to changing customer behaviours and market dynamics.

Future Work

In this project, we have developed a machine learning model to predict customer churn in the telecom segment. However, there are several avenues for future work and enhancements to the model:

1. Feature Engineering

Create New Features: Explore the creation of new features by combining existing ones to potentially improve model performance.

2.Incorporating More Data

- External Data Sources: Integrate additional external data sources such as customer feedback, social media sentiment, or customer support interactions.
- Temporal Data: Include time-series data to capture trends and seasonality in customer behaviour, which can be valuable in predicting churn.

3.Advanced Techniques

- Ensemble Methods: Implement ensemble learning techniques such as stacking or boosting to combine the predictions of multiple models for improved accuracy.
- Deep Learning: Experiment with deep learning models such as neural networks, especially for complex patterns in the data.

4.Real-Time Prediction and Intervention

- Real-Time Scoring: Develop a system for real-time scoring of new data, allowing for immediate predictions of customer churn as new information becomes available.

- Triggered Actions: Implement automated actions or campaigns triggered by the model's predictions, such as personalized offers or targeted retention efforts.

5.Deployment and Scalability

- Model Deployment: Deploy the model into a production environment, making it accessible to stakeholders through APIs or a web interface.
- Scalability: Ensure the model is scalable to handle a larger volume of data and can be easily updated with new data.

6.A/B Testing and Continuous Improvement

- A/B Testing: Conduct A/B tests to evaluate the effectiveness of different retention strategies and model improvements.
- Feedback Loop: Establish a feedback loop to continuously monitor the model's performance and gather feedback from business users for further enhancements.

References:

Scikit learn documentation: https://scikit-learn.org/stable/index.html

Geeks for geeks: https://www.geeksforgeeks.org/machine-learning/

Hyperparameter tuning: https://towardsdatascience.com/hyperparameter-tuning-explained-d0ebb2ba1d35

Code Snapshots

Figure 4:

jupyter **Telecom project 2** Last Checkpoint: Last Wednesday at 3:40 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

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

In [2]:
```
churn_dataset=pd.read_csv(r"C:\Users\sande\Documents\btech\internship\Untitled Folder\Churn Prediction in Telecom Industry
```

In [3]:
```
df=pd.DataFrame(churn_dataset)
```

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

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSuppo
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	I
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	I
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	I
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Y
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	I

5 rows × 21 columns

In [5]:
```
df.columns
```

Out[5]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
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
```

In [7]:
```
df.info()
```

```
<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
```

```
 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 [8]:
```
df['MultipleLines'].unique()
```

Out[8]:
```
array(['No phone service', 'No', 'Yes'], dtype=object)
```

In [9]:
```
m1=pd.get_dummies(df['MultipleLines'],prefix='MultipleLines')
m1=m1.drop('MultipleLines_No phone service',axis=1)
df=pd.concat((df,m1),axis=1)
df
```

Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	StreamingTV	StreamingM
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	No	
2	3668-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	...	No	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone	DSL	Yes	...	No	

```
In [10]: df.columns
```

```
Out[10]: Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn',
       'MultipleLines_No', 'MultipleLines_Yes'],
      dtype='object')
```

Exploratory Data Analysis

Univarite Analysis

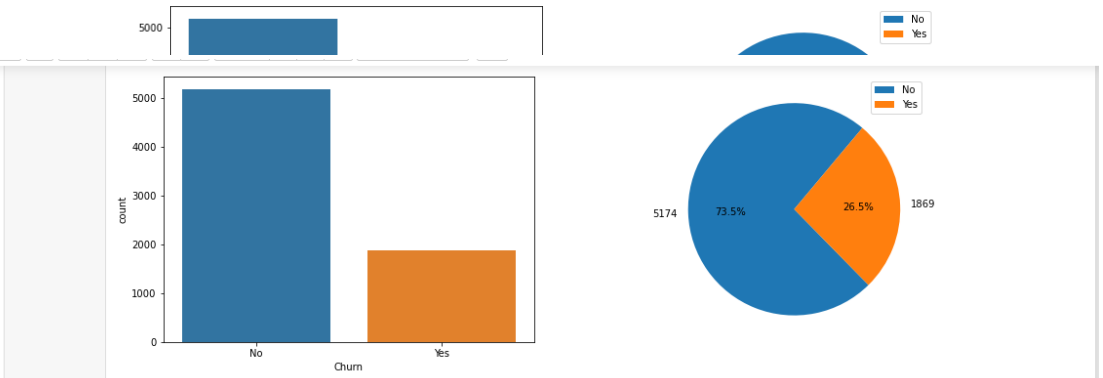
```
In [11]: churn_dataset=df
```

```
In [12]: churn_dataset['Churn'].describe()
```

```
Out[12]: count     7043
unique       2
top         No
freq      5174
Name: Churn, dtype: object
```

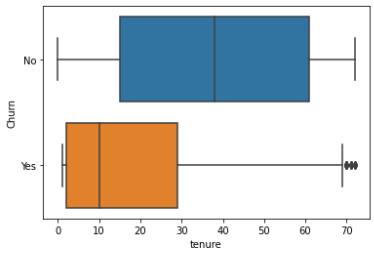
```
In [13]: fig,axis=plt.subplots(nrows=1,ncols=2,figsize=(15,5))
sns.countplot(churn_dataset['Churn'],ax=axis[0])
plt.pie(churn_dataset['Churn'].value_counts(),labels=churn_dataset['Churn'].value_counts(), autopct='%1.1f%%',startangle=50)
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()
```

```
C:\Users\sande\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



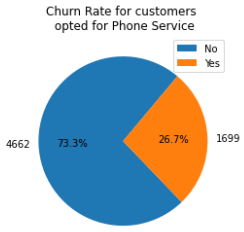
Tenure

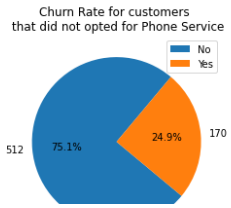
```
In [14]: sns.boxplot(x = 'tenure', y = 'Churn', data = churn_dataset)
plt.show()
```



Phone service

In [15]:
```
plt.title('Churn Rate for customers \n opted for Phone Service')
plt.gca().set_aspect('equal')
Optedyes=plt.pie(churn_dataset[churn_dataset['PhoneService']=='Yes']["Churn"].value_counts(),labels=churn_dataset[churn_data
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()

plt.title('Churn Rate for customers \n that did not opted for Phone Service')
OptedNo=plt.pie(churn_dataset[churn_dataset['PhoneService']=='No']["Churn"].value_counts(),labels=churn_dataset[churn_datase
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()
```



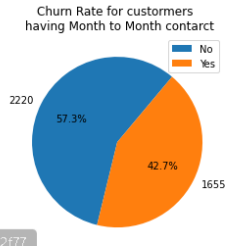


contract

In [16]:
```
churn_dataset['Contract'].value_counts()
```

Out[16]:
```
Month-to-month    3875
Two year          1695
One year          1473
Name: Contract, dtype: int64
```

In [17]:
```
plt.title('Churn Rate for custormers \n having Month to Month contarct')
month=plt.pie(churn_dataset[churn_dataset['Contract']=='Month-to-month']['Churn'].value_counts(),labels=churn_dataset[churn_
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()

plt.title('Churn Rate for custormers \n having 1 year contarct')
oneyear=plt.pie(churn_dataset[churn_dataset['Contract']=='One year']['Churn'].value_counts(),labels=churn_dataset[churn_data
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()

plt.title('Churn Rate for custormers \n having 2 year contarct')
month=plt.pie(churn_dataset[churn_dataset['Contract']=='Two year']['Churn'].value_counts(),labels=churn_dataset[churn_datase
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()
```

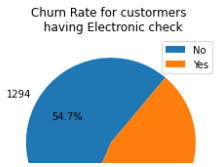


Payment Method

In [20]:
```
churn_dataset['PaymentMethod'].value_counts()
```

Out[20]:
```
Electronic check             2365
Mailed check                 1612
Bank transfer (automatic)    1544
Credit card (automatic)      1522
Name: PaymentMethod, dtype: int64
```

In [21]:
```
plt.title('Churn Rate for custormers \n having Electronic check')
Electronic_check=plt.pie(churn_dataset[churn_dataset['PaymentMethod']=='Electronic check']['Churn'].value_counts(),labels=ch
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()

plt.title('Churn Rate for custormers \n having Mailed check')
Mailed_check=plt.pie(churn_dataset[churn_dataset['PaymentMethod']=='Mailed check']['Churn'].value_counts(),labels=churn_data
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()

plt.title('Churn Rate for custormers \n having Bank transfer')
Bank_transfer=plt.pie(churn_dataset[churn_dataset['PaymentMethod']=='Bank transfer (automatic)']['Churn'].value_counts(),lab
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()

plt.title('Churn Rate for custormers \n having Credit card')
Credit_card=plt.pie(churn_dataset[churn_dataset['PaymentMethod']=='Credit card (automatic)']['Churn'].value_counts(),labels=
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()
```

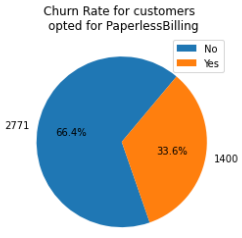


Paperless Billing

In [18]:
```
churn_dataset['PaperlessBilling'].value_counts()
```

Out[18]:
```
Yes    4171
No     2872
Name: PaperlessBilling, dtype: int64
```

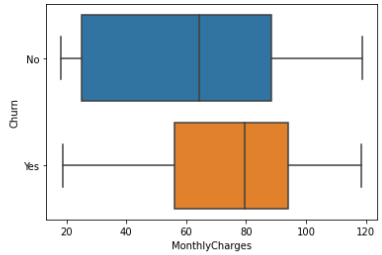
In [19]:
```
plt.title('Churn Rate for customers \n opted for PaperlessBilling')
Optedyes=plt.pie(churn_dataset[churn_dataset['PaperlessBilling']=='Yes']["Churn"].value_counts(),labels=churn_dataset[churn_
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()

plt.title('Churn Rate for customers \n that did not opted forPaperlessBilling')
OptedNo=plt.pie(churn_dataset[churn_dataset['PaperlessBilling']=='No']["Churn"].value_counts(),labels=churn_dataset[churn_da
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()
```



/Telecom project 2.ipynb

Monthly Charges

In [22]:
```
sns.boxplot(x = 'MonthlyCharges', y = 'Churn', data =churn_dataset)
plt.show()
```



Gender

In [23]:
```
churn_dataset['gender'].value_counts()
```

Out[23]:
```
Male      3555
Female    3488
Name: gender, dtype: int64
```

In [24]:
```
plt.title('Male')
Male=plt.pie(churn_dataset[churn_dataset['gender']=='Male']["Churn"].value_counts(),labels=churn_dataset[churn_dataset['gen
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()
plt.title('Female')
OptedNo=plt.pie(churn_dataset[churn_dataset['gender']=='Female']["Churn"].value_counts(),labels=churn_dataset[churn_dataset
plt.legend(churn_dataset['Churn'].value_counts().index,loc='best')
plt.show()
```

Streaming TV doesn't make such impact on churning.

Data Preparation

In [45]:
```
df['PhoneService_binary']=df['PhoneService'].map({'No':0,"Yes":1})
```

In [46]:
```
df['Dependents_binary']=df['Dependents'].map({'No':0,"Yes":1})
```

In [47]:
```
df['Partner'].unique()
```

Out[47]:
```
array(['Yes', 'No'], dtype=object)
```

In [48]:
```
df['Partner_binary']=df['Partner'].map({'No':0,"Yes":1})
```

In [49]:
```
df['Churn'].unique()
```

Out[49]:
```
array(['No', 'Yes'], dtype=object)
```

In [50]:
```
df['Churn_binary']=df['Churn'].map({'No':0,"Yes":1})
```

In [51]:
```
df.corr()['Churn_binary'][:-1]
```

Out[51]:
```
SeniorCitizen          0.150889
tenure                -0.352229
MonthlyCharges         0.193356
MultipleLines_No      -0.032569
MultipleLines_Yes      0.040102
PhoneService_binary    0.011942
Dependents_binary     -0.164221
Partner_binary        -0.150448
Name: Churn_binary, dtype: float64
```

In [52]:
```
m2=pd.get_dummies(df['gender'],prefix='gender')
m2
```

Out[52]:

	gender_Female	gender_Male
0	1	0
1	0	1

7043 rows × 2 columns

In [53]:
```
df=pd.concat((m2,df),axis=1)
df
```

Out[53]:

	gender_Female	gender_Male	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	...	PaymentMethod	MonthlyC
0	1	0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	...	Electronic check	
1	0	1	5575-GNVDE	Male	0	No	No	34	Yes	No	...	Mailed check	
2	0	1	3668-QPYBK	Male	0	No	No	2	Yes	No	...	Mailed check	
3	0	1	7795-CFOCW	Male	0	No	No	45	No	No phone service	...	Bank transfer (automatic)	
4	1	0	9237-HQITU	Female	0	No	No	2	Yes	No	...	Electronic check	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	0	1	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	...	Mailed check	
7039	1	0	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	...	Credit card (automatic)	
7040	1	0	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	...	Electronic check	
7041	0	1	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	...	Mailed check	
7042	0	1	3186-AJIEK	Male	0	No	No	66	Yes	No	...	Bank transfer (automatic)	

7043 rows × 29 columns

In [54]:
```
m3=pd.get_dummies(df['PaymentMethod'],prefix='PaymentMethod')
m3
```

Out[54]:

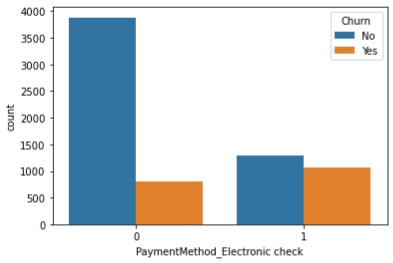
	PaymentMethod_Bank transfer (automatic)	PaymentMethod_Credit card (automatic)	PaymentMethod_Electronic check	PaymentMethod_Mailed check
0	0	0	1	0

In [56]:
```
m4.corr()['Churn_binary'][:-1]
```

Out[56]:
```
PaymentMethod_Bank transfer (automatic)   -0.117937
PaymentMethod_Credit card (automatic)     -0.134302
PaymentMethod_Electronic check             0.301919
PaymentMethod_Mailed check                -0.091683
Name: Churn_binary, dtype: float64
```

In [57]:
```
sns.countplot(x='PaymentMethod_Electronic check',data=m4,hue='Churn')
```

Out[57]:
```
<AxesSubplot:xlabel='PaymentMethod_Electronic check', ylabel='count'>
```



In [58]:
```
m4['PaymentMethod_Electronic check']
```

Out[58]:
```
0       1
1       0
2       0
3       0
4       1
       ..
7038    0
7039    0
7040    1
7041    0
7042    0
Name: PaymentMethod_Electronic check, Length: 7043, dtype: uint8
```

```
Name: InternetService, Length: 7043, dtype: object
```

In [63]:
```
print(df['OnlineSecurity'].unique())
```

```
['No' 'Yes' 'No internet service']
```

In [64]:
```
for x in df.columns:
    print(f'{x}:    {df[x].unique()}')
```

```
gender_Female:    [1 0]
gender_Male:    [0 1]
customerID:    ['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' ... '4801-JZAZL' '8361-LTMKD'
 '3186-AJIEK']
gender:    ['Female' 'Male']
SeniorCitizen:    [0 1]
Partner:    ['Yes' 'No']
Dependents:    ['No' 'Yes']
tenure:    [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]
PhoneService:    ['No' 'Yes']
MultipleLines:    ['No phone service' 'No' 'Yes']
InternetService:    ['DSL' 'Fiber optic' 'No']
OnlineSecurity:    ['No' 'Yes' 'No internet service']
OnlineBackup:    ['Yes' 'No' 'No internet service']
DeviceProtection:    ['No' 'Yes' 'No internet service']
TechSupport:    ['No' 'Yes' 'No internet service']
StreamingTV:    ['No' 'Yes' 'No internet service']
StreamingMovies:    ['No' 'Yes' 'No internet service']
Contract:    ['Month-to-month' 'One year' 'Two year']
PaperlessBilling:    ['Yes' 'No']
PaymentMethod:    ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges:    [29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges:    ['29.85' '1889.5' '108.15' ... '346.45' '306.6' '6844.5']
Churn:    ['No' 'Yes']
MultipleLines_No:    [0 1]
MultipleLines_Yes:    [0 1]
PhoneService_binary:    [0 1]
Dependents_binary:    [0 1]
Partner_binary:    [1 0]
Churn_binary:    [0 1]
```

In [68]:
```
a=['OnlineSecurity', 'OnlineBackup', 'DeviceProtection','TechSupport', 'StreamingTV', 'StreamingMovies']
for j,x in enumerate(a):
    a[j]=pd.get_dummies(df[x],prefix=x)
    a[j] = a[j].drop(x+'_No internet service',axis=1)
    df=pd.concat([df,a[j]],axis=1)
df
```

Out[68]:

	gender_Female	gender_Male	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	...	OnlineBackup_No	OnlineE
0	1	0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	...	0	
1	0	1	5575-GNVDE	Male	0	No	No	34	Yes	No	...	1	
2	0	1	3668-QPYBK	Male	0	No	No	2	Yes	No	...	0	
3	0	1	7795-CFOCW	Male	0	No	No	45	No	No phone service	...	1	
4	1	0	9237-HQITU	Female	0	No	No	2	Yes	No	...	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	0	1	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	...	1	
7039	1	0	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	...	0	
7040	1	0	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	...	1	
7041	0	1	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	...	1	
7042	0	1	3186-AJIEK	Male	0	No	No	66	Yes	No	...	1	

7043 rows × 43 columns

In [69]:
```
df.columns
```

Out[69]:
```
Index(['gender_Female', 'gender_Male', 'customerID', 'gender', 'SeniorCitizen',
       'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines',
       'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
```

Logistic Regression

In [92]:
```
from sklearn.linear_model import LogisticRegression
```

In [93]:
```
model=LogisticRegression()
```

Normalization

In [94]:
```
from sklearn.preprocessing import StandardScaler
```

In [95]:
```
scale=StandardScaler()
```

In [96]:
```
X_train_scaled=scale.fit_transform(X_train)
```

In [97]:
```
X_test_scaled=scale.transform(X_test)
```

In [ ]:
```

```

In [98]:
```
model.fit(X_train,y_train)
```

```
C:\Users\sande\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to conv
erge (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(
```

Out[98]:
```
LogisticRegression()
```

In [99]:
```
y_predict=model.predict(X_test)
```

In [100]:
```
from sklearn.metrics import accuracy_score,classification_report,plot_confusion_matrix
```

```
OnlineSecurity_No                       0.342637
Contract_Month-to-month                 0.405103
Name: Churn_binary, dtype: float64
```

In [79]:
```
final_df['TotalCharges']=final_df['TotalCharges'].replace(' ',np.nan)
final_df['TotalCharges']=pd.to_numeric(final_df['TotalCharges'])
```

In [80]:
```
value = (final_df['TotalCharges']/final_df['MonthlyCharges']).median()*final_df['MonthlyCharges']
```

In [81]:
```
final_df['TotalCharges'] = value.where(final_df['TotalCharges'] == np.nan, other =final_df['TotalCharges'])
```

In [82]:
```
final_df['TotalCharges'].describe()
```

Out[82]:
```
count    7032.000000
mean     2283.300441
std      2266.771362
min        18.800000
25%       401.450000
50%      1397.475000
75%      3794.737500
max      8684.800000
Name: TotalCharges, dtype: float64
```

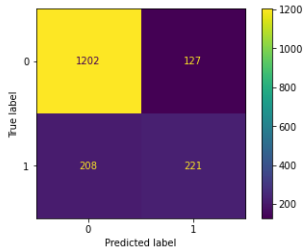
In [83]:
```
final_df.isnull().sum()
```

Out[83]:
```
gender_Female                           0
gender_Male                             0
tenure                                  0
MonthlyCharges                          0
TotalCharges                           11
MultipleLines_No                        0
MultipleLines_Yes                       0
PhoneService_binary                     0
Dependents_binary                       0
Partner_binary                          0
Churn_binary                            0
InternetService_DSL                     0
InternetService_Fiber optic             0
OnlineSecurity_No                       0
OnlineSecurity_Yes                      0
OnlineBackup_No                         0
OnlineBackup_Yes                        0
```

```
In [102]: plot_confusion_matrix(model,X_test,y_test)
```

```
C:\Users\sande\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 meth
ods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
Out[102]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x273977493d0>
```




```
In [103]: accuracy_score(y_test,y_predict)
```

```
Out[103]: 0.8094425483503982
```

```
In [104]: from sklearn.model_selection import GridSearchCV
```

```
In [128]: parameters={'penalty':['l1', 'l2', 'elasticnet', 'none'],'l1_ratio':[0,0.001,0.01,0.05,0.1,0.25,0.5,0.75,0.9,0.95,0.99,1],'
```

```
In [129]: grid_model=GridSearchCV(model,parameters,verbose=2)
```

```
In [130]: grid_model.fit(X_train_scaled,y_train)
```

```
Fitting 5 folds for each of 1920 candidates, totalling 9600 fits
[CV] END ......C=1, l1_ratio=0, penalty=l1, solver=newton-cg; total time=   0.0s
[CV] END ......C=1, l1_ratio=0, penalty=l1, solver=newton-cg; total time=   0.0s
[CV] END ......C=1, l1_ratio=0, penalty=l1, solver=newton-cg; total time=   0.0s
```

KNeighborsClassifier

```
In [76]: from sklearn.svm import SVC
         from sklearn.neighbors import KNeighborsClassifier
```

```
In [82]: model_neighbors=KNeighborsClassifier()
```

```
In [83]: model_neighbors.fit(X_train,y_train)
```

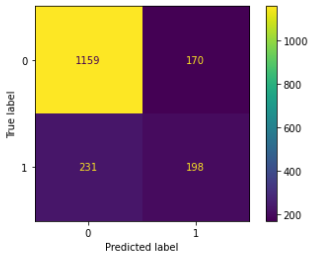
```
Out[83]: KNeighborsClassifier()
```

```
In [84]: n_predict=model_neighbors.predict(X_test)
```

```
In [85]: plot_confusion_matrix(model_neighbors,X_test,y_test)
```

```
C:\Users\sande\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 meth
ods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
Out[85]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x193ed87ac10>
```




```
In [86]: print(classification_report(y_test,n_predict))
```

In [132]:
```
grid_neighbors.best_params_
```

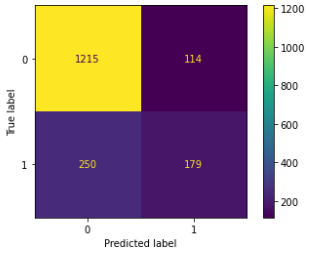
Out[132]:
```
{'algorithm': 'auto',
 'leaf_size': 10,
 'metric': 'manhattan',
 'n_neighbors': 33,
 'p': 1,
 'weights': 'uniform'}
```

In [133]:
```
neighbors_predict=grid_neighbors.predict(X_test)
```

In [134]:
```
plot_confusion_matrix(grid_neighbors,X_test,y_test)
```

```
C:\Users\sande\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 meth
ods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[134]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x193ef21c820>
```



In [137]:
```
print(classification_report(y_test,neighbors_predict))
```

```
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1329
           1       0.61      0.42      0.50       429
```

Support Vector Machine

In [105]:
```
from sklearn.svm import SVC
```

In [106]:
```
svc_model=SVC(C=6,class_weight='balanced',gamma='auto',kernel='poly')
```

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

In [ ]:
```
svc_predict=svc_model.predict(X_test)
```

In [ ]:
```
print(classification_report(y_test,svc_predict))
```

In [ ]:
```
C=6,class_weight='balanced',gamma='auto',kernel='poly'

              precision    recall  f1-score   support

           0       0.90      0.77      0.83      1329
           1       0.50      0.73      0.59       429

    accuracy                           0.76      1758
   macro avg       0.70      0.75      0.71      1758
weighted avg       0.80      0.76      0.77      1758
```

In [75]:
```
plot_confusion_matrix(svc_model,X_test,y_test)
```

```
C:\Users\sande\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 meth
ods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[75]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x241dae65370>
```

Decision Tree Classifier

In [103]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [104]:
```
DTC=DecisionTreeClassifier()
```

In [109]:
```
DTC.fit(X_train_scaled,y_train)
```

Out[109]:
```
DecisionTreeClassifier()
```

In [110]:
```
y_pred=DTC.predict(X_test_scaled)
```

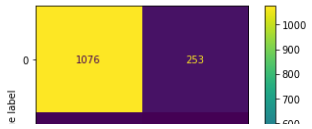
In [111]:
```
print(classification_report(y_test,y_pred))
```

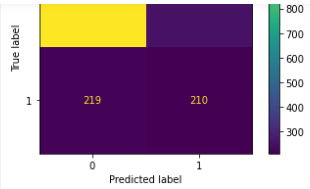
```
              precision    recall  f1-score   support

           0       0.83      0.81      0.82      1329
           1       0.45      0.49      0.47       429

    accuracy                           0.73      1758
   macro avg       0.64      0.65      0.65      1758
weighted avg       0.74      0.73      0.73      1758
```

In [112]:
```
plot_confusion_matrix(DTC,X_test_scaled,y_test)
```

```
C:\Users\sande\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 meth
ods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[112]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1a0e1c6b4f0>
```





Random Forest

In [113]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [114]:
```
rfm=RandomForestClassifier()
```

In [115]:
```
rfm.fit(X_train_scaled,y_train)
```

Out[115]:
```
RandomForestClassifier()
```

In [116]:
```
y_pred=rfm.predict(X_test_scaled)
```

In [117]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      1329
           1       0.61      0.51      0.55       429

    accuracy                           0.80      1758
   macro avg       0.73      0.70      0.71      1758
weighted avg       0.79      0.80      0.79      1758
```

In [122]:
```
plot_confusion_matrix(rfm,X_test_scaled,y_test)
```

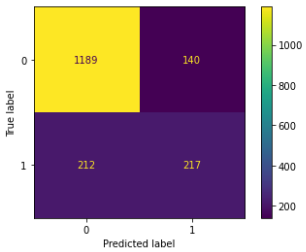
```
C:\Users\sande\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix
is deprecated; Function `plot confusion matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class meth
```


```
           0       0.85      0.89      0.87      1329
           1       0.61      0.51      0.55       429

    accuracy                           0.80      1758
   macro avg       0.73      0.70      0.71      1758
weighted avg       0.79      0.80      0.79      1758
```

In [122]:
```
plot_confusion_matrix(rfm,X_test_scaled,y_test)
```


```
C:\Users\sande\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 meth
ods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[122]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1a0e222e160>
```



In [ ]:
```

```