ARINGNAR ANNA
GOVERMENT
ARTS COLLEGE
VILLUPURAM

INTELLIGENT CUSTOMER RETENTION

MACHINE LEARNING WITH PYTHON

Project Title : Intelligent Customer Retention: Using Machine Learning for Enhanced Prediction of Telecom Customer Churn

Team Id : NM2023TMID21193

Team Leader : MADHAN KUMAR R

Team Member : PUSHPARAJ T

Team Member : VISHNU N

Team Member : GOPI E

Abstract

Customer churn is a major challenge for telecommunications companies as it can result in significant revenue losses. In this project, we aim to predict customer churn by analyzing historical customer data, such as call duration, data usage, and payment history. We will use machine learning algorithms, including logistic regression, decision trees, and random forests, to build predictive models. We will evaluate the models using metrics such as accuracy, precision, recall, and F1 score. The results of this project can help telecom companies identify customers who are likely to churn and take proactive measures to retain them.

INTRODUCTION

Overview

Telecom customer churn prediction is a process of identifying customers who are likely to leave a telecom service provider and switch to another provider. This is important for telecom companies as losing customers can have a significant impact on their revenue and market share. By predicting which customers are likely to churn, companies can take proactive measures to retain them and prevent them from leaving.

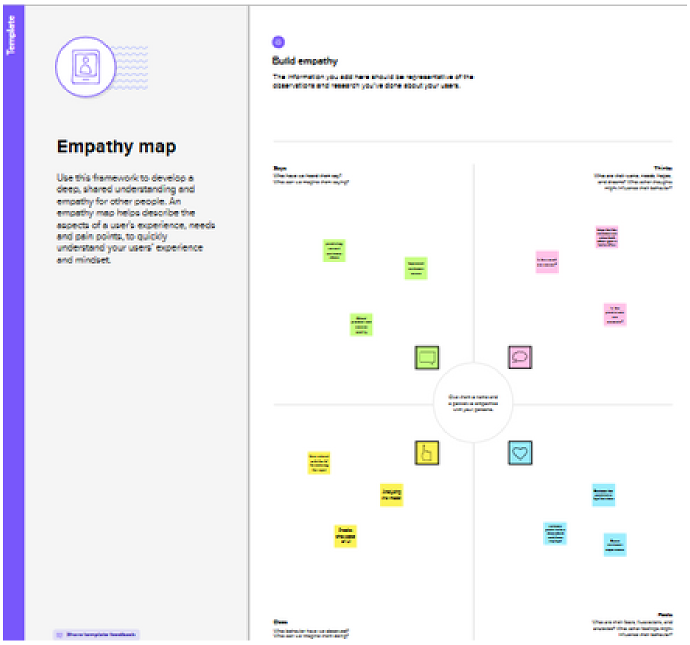
Purpose

The purpose of telecom customer churn prediction is to identify customers who are at risk of leaving a telecom service provider and take proactive measures to retain them. Churn prediction models analyze customer data to identify patterns and behaviors that may indicate a high likelihood of customer churn. By predicting which customers are likely to churn, telecom companies can take targeted actions to retain them, such as offering discounts, incentives, or personalized promotions.

Churn prediction also helps telecom companies improve customer satisfaction and loyalty by identifying the root causes of churn and addressing them. By addressing the factors that lead to churn, telecom companies can improve customer experience, reduce customer complaints, and increase customer retention.

PROBLEM DEFINITION &DESIGN THINKING

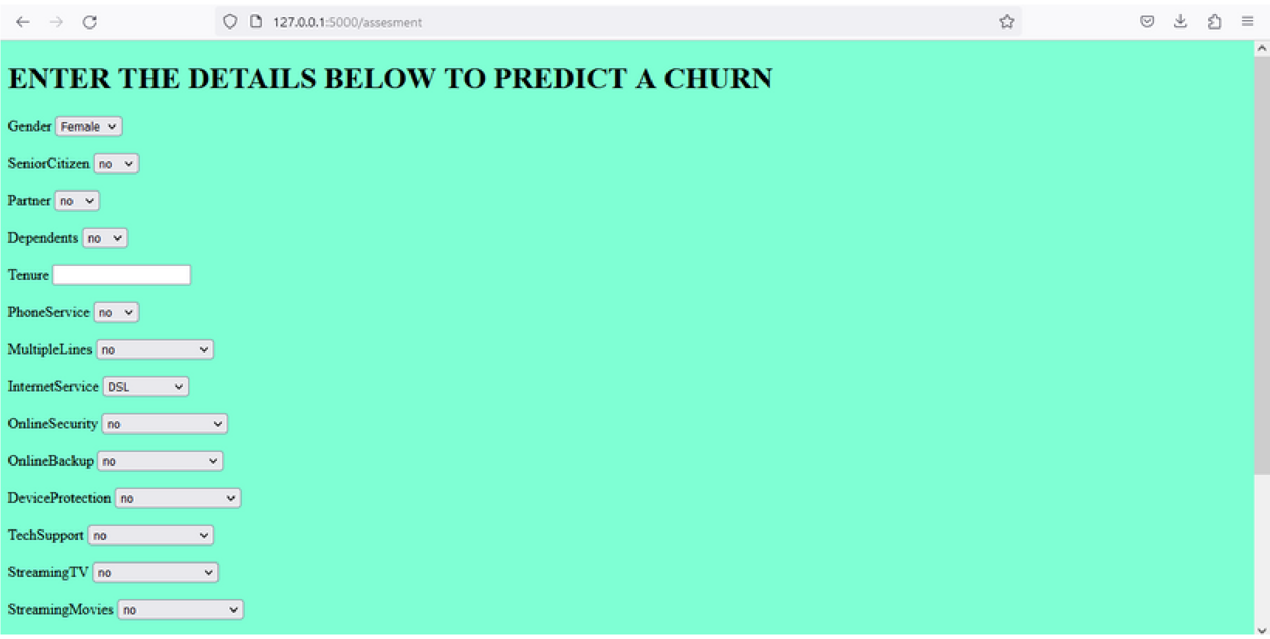
Empathy Map



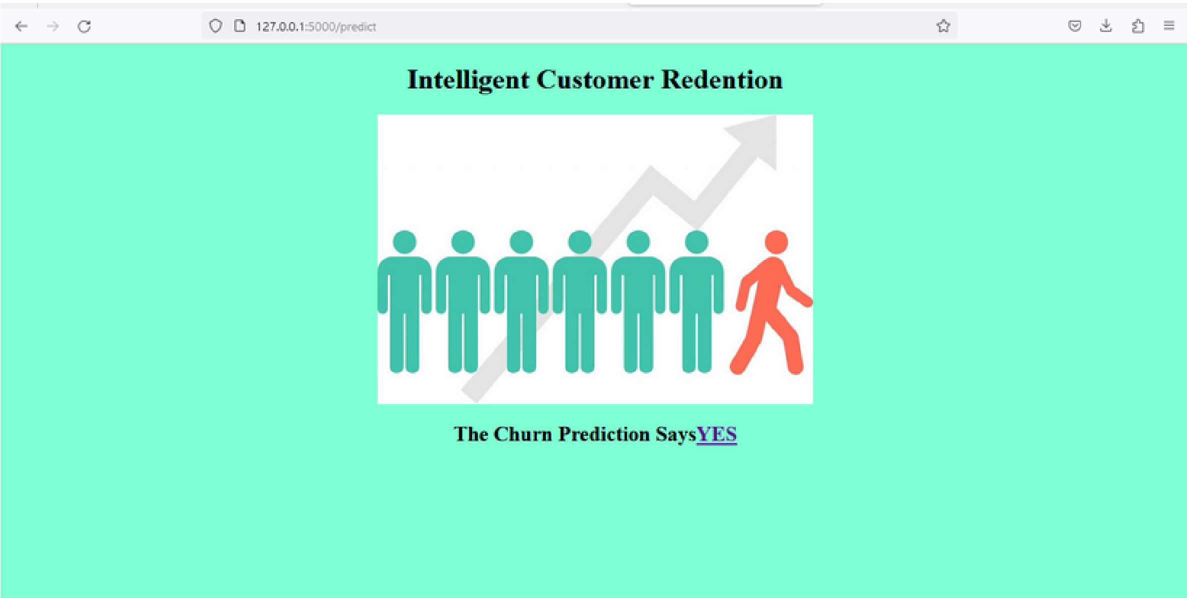
Ideation & Brainstrom Map

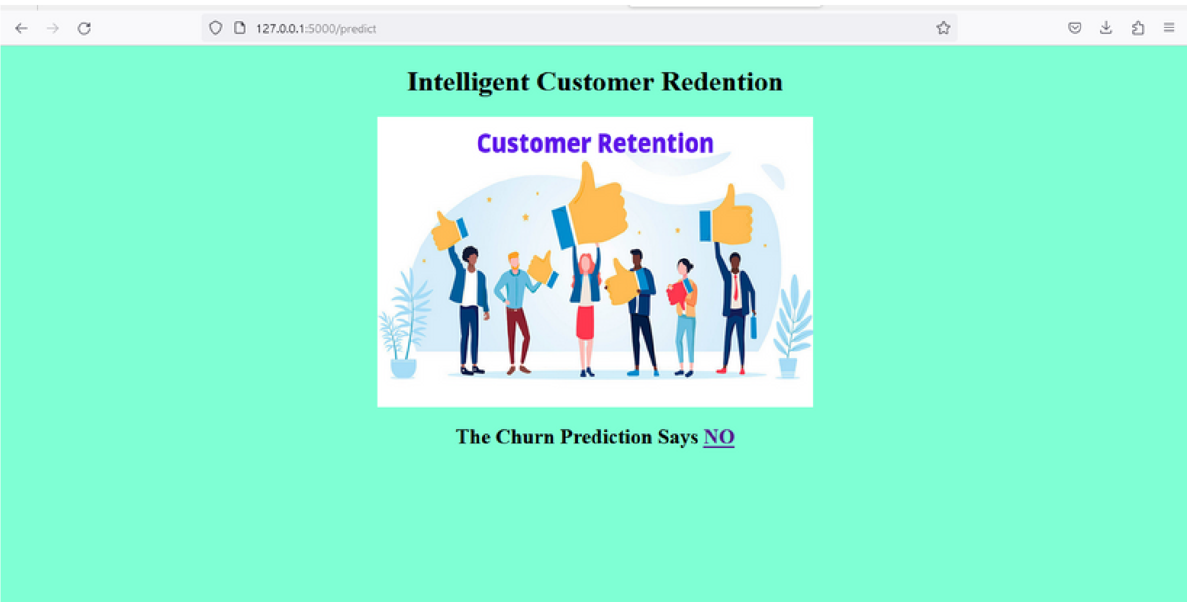
RESULT


127.0.0.1:5000
TELECOM CUSTOMER CHURN PREDICTION
Telecom customer churn prediction is the process of identifying customers who are likely to leave a telecom service provider and switch to another service provider or simply stop using the telecom service altogether. Customer churn is a major challenge for telecom companies, as it can lead to a loss of revenue and market share
Predict which customers are going to churn
Know why they are going to churn
Click Me to Continue With Prediction


127.0.0.1:5000/assesment
ENTER THE DETAILS BELOW TO PREDICT A CHURN
Gender Female
SeniorCitizen no
Partner no
Dependents no
Tenure
PhoneService no
MultipleLines no
InternetService DSL
OnlineSecurity no
OnlineBackup no
DeviceProtection no
TechSupport no
StreamingTV no
StreamingMovies no


127.0.0.1:5000/assesment
Dependents no
Tenure
PhoneService no
MultipleLines no
InternetService DSL
OnlineSecurity no
OnlineBackup no
DeviceProtection no
TechSupport no
StreamingTV no
StreamingMovies no
Contract month to month
PaymentMethod electronic check
PaperlessBilling no
MonthlyCharges
TotalCharges
submit


127.0.0.1:5000/predict
Intelligent Customer Redention
The Churn Prediction SaysYES


127.0.0.1:5000/predict
Intelligent Customer Redention
Customer Retention
The Churn Prediction Says NO

Advanteges

- Enhanced customer satisfaction
- Improved customer retention
- Better resource allocation
- Increased revenue
- Improved marketing strategies

Disadvantages

- Lack of transparency
- False positives
- Complexity
- Overfitting
- Data quality

Applications

- Customer Retention: The primary application of telecom customer churn prediction is to identify customers who are at high risk of leaving the company and take proactive measures to retain them. By predicting customer churn in advance, telecom companies can offer personalized offers and promotions to retain customers.
- Customer Segmentation: Another application of customer churn prediction is to segment customers into different categories based on their likelihood of churning. Telecom companies can use this information to target different segments with customized retention strategies.
- Marketing Campaigns: Telecom companies can also use customer churn prediction models to improve their marketing campaigns. By identifying customers who are more likely to churn, companies can target them with specific promotions and offers to keep them engaged.
- Product Development: Telecom companies can also use churn prediction models to improve their product development process. By identifying the reasons why customers are leaving, companies can develop new products and services that better meet their customers' needs.
- Product Development: Telecom companies can also use churn prediction models to improve their product development process. By identifying the reasons why customers are leaving, companies can develop new products and services that better meet their customers' needs.

CONCLUSION

According to the result, random forest method is a better decision tree to identify customers who are likely to switch to other service providers. The project involves collecting and preprocessing data, feature engineering, model selection, training and evaluation, and deployment.

Through the use of machine learning algorithms such as logistic regression, decision trees, random forests, KNN,ANN methods.Telecom companies can accurately predict customer churn and take proactive measures to retain their customers. By leveraging the insights generated from the model, the telecom company can optimize their marketing strategies and tailor their offerings to suit the needs of their customers.

Overall, the success of a telecom customer churn prediction project depends on the quality of data, the effectiveness of feature engineering, the choice of the appropriate machine learning algorithm, and the ability to interpret and utilize the model's output. With the right approach and resources, telecom companies can significantly reduce customer churn and improve customer retention rates.

Future Scope

- Integrating more data sources: Currently, customer churn prediction models typically rely on data from a limited number of sources, such as customer demographics, transaction history, and customer support interactions. In the future, models may be enhanced by incorporating additional data sources, such as social media activity, website interactions, and sensor data from connected devices.
- Exploring new modeling techniques: While many machine learning algorithms have been used for customer churn prediction, there may be room for further exploration of new techniques that could improve model performance, such as deep learning or ensemble learning

- Real-time prediction: Currently, most customer churn prediction models are used to identify customers who are at risk of churning in the near future. However, there may be potential for models to be used in real-time to predict customer churn as it happens, allowing for more immediate interventions to retain customers.

APPENDIX

SOURCE CODE

C:\Intelligent Customer Redention\Flask\app.py (Intelligent Customer Redention) - Sublime Text (UNREGISTERED)

```python
from flask import Flask, render_template, request
app = Flask(__name__)
import pickle
model = pickle.load(open('churn3.pkl','rb'))

@app.route('/')
def helloworld():
    return render_template('base.html')

@app.route('/assesment')
def prediction():
    return render_template('index.html')

@app.route('/predict', methods = ['POST'])
def admin():
    a= request.form["gender"]
    if (a == 'f'):
        a=0
    if (a == 'm'):
        a=1
    b= request.form["senior-citizen"]
    if (b == 'n'):
        b=0
    if (b == 'y'):
        b=1
    c= request.form["partner"]
    if (c == 'n'):
        c=0
    if (c == 'y'):
        c=1
    d= request.form["dependents"]
    if (d == 'n'):
        d=0
    if (d == 'y'):
        d=1
    e= request.form["tenure"]
    f= request.form["phoneService"]
    if (f == 'n'):
        f=0
    if (f == 'y'):
        f=1
    g= request.form["multipleLines"]
```

Line 30, Column 12 Spaces: 4 Python

C:\Intelligent Customer Redention\Flask\app.py (Intelligent Customer Redention) - Sublime Text (UNREGISTERED)

```python
    f= request.form["phoneService"]
    if (f == 'n'):
        f=0
    if (f == 'y'):
        f=1
    g= request.form["multipleLines"]
    if (g == 'n'):
        g=0
    if (g == 'nps'):
        g=1
    if (g == 'y'):
        g=2
    h= request.form["internetService"]
    if (h == 'dsl'):
        h=0
    if (h == 'fo'):
        h=1
    if (h == 'n'):
        h=2
    i= request.form["onlineSecurity"]
    if (i == 'n'):
        i=0
    if (i == 'nis'):
        i=1
    if (i == 'y'):
        i=2
    j= request.form["onlineBackup"]
    if (j == 'n'):
        j=0
    if (j == 'nis'):
        j=1
    if (j == 'y'):
        j=2
    k= request.form["deviceProtection"]
    if (k == 'n'):
        k=0
    if (k == 'nis'):
        k=1
    if (k == 'y'):
        k=2
    l= request.form["techSuport"]
    if (l == 'n'):
```

Line 30, Column 12 Spaces: 4 Python

C:\Intelligent Customer Redention\Flask\app.py (Intelligent Customer Redention) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

app.py

```python
    l= request.form["techSuport"]
    if (l == 'n'):
        l=0
    if (l == 'nis'):
        l=1
    if (l == 'y'):
        l=2
    m= request.form["StreamingTV"]
    if (m == 'n'):
        m=0
    if (m == 'nis'):
        m=1
    if (m == 'y'):
        m=2
    n= request.form["streamingMovies"]
    if (n == 'n'):
        n=0
    if (n == 'nis'):
        n=1
    if (n == 'y'):
        n=2
    o= request.form["contract"]
    if (o == 'mtm'):
        o=0
    if (o == 'oyr'):
        o=1
    if (o == 'tyrs'):
        o=2
    p= request.form["paymentMethod"]
    if (p == 'ec'):
        p=2
    if (p == 'mail'):
        p=3
    if (p == 'bt'):
        p=0
    if (p == 'cc'):
        p=1
    q= request.form["paperlessBilling"]
    if (q == 'n'):
        q=0
    if (q == 'y'):
        q=1
```

Line 30, Column 12 Spaces: 4 Python

app.py

```python
    p= request.form["paymentMethod"]
    if (p == 'ec'):
        p=2
    if (p == 'mail'):
        p=3
    if (p == 'bt'):
        p=0
    if (p == 'cc'):
        p=1
    q= request.form["paperlessBilling"]
    if (q == 'n'):
        q=0
    if (q == 'y'):
        q=1
    r= request.form["MonthlyCharges"]
    s= request.form["TotalCharges"]

    t=[[int(a),int(b),int(c),int(d),int(e),int(f),  int(g),int(h),int(i),int(j),int(k),int(l),int(m),int(n),int(o),int(p),int(q), int(r), int(s)]]
    x = model.predict(t)
    if (x[0] == 0):
        y ="No"
        return render_template("predno.html",z = y)


    if (x[0] == 1):
        y ="Yes"
        return render_template("predyes.html",z = y)

if __name__== '__main__':
    app.run(debug = True)
```

Line 30, Column 12 Spaces: 4 Python

Source code

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

[2]:
```
data = pd.read_csv("dataset2.csv")
```

[3]:
```
data.head()
```

[3]: customerID gender SeniorCitizenPartnerDependents tenurePhoneService \
07590-VHVEGFemale 0 Yes No 1 No
15575-GNVDEMale 0 No No 34 Yes
23668-QPYBKMale 0 No No 2 Yes
37795-CFOCWMale 0No No45 No
49237-HQITUFemale 0 No No 2 Yes

MultipleLines InternetService OnlineSecurity ... DeviceProtection \
0Nophoneservice DSL No... No
1 No DSL Yes... Yes
2 No DSL Yes... No
3Nophoneservice DSL Yes... Yes
4 NoFiberoptic No... No

TechSupportStreamingTVStreamingMovies ContractPaperlessBilling\
0 No No NoMonth-to-month Yes
1 No No No Oneyear No
2 No No NoMonth-to-month Yes
3 Yes No No Oneyear No
4 No No NoMonth-to-month Yes

PaymentMethod MonthlyCharges TotalCharges Churn
0 Electroniccheck 29.85 29.85No
1 Mailedcheck 56.95 1889.5No
2 Mailedcheck 53.85 108.15Yes
3Banktransfer(automatic) 42.30 1840.75No
4 Electroniccheck 70.70 151.65Yes

[5 rows x 21 columns]

```
[4]: data['Churn'].value_counts()
```

[4]:No 5174
Yes 1869
Name: Churn, dtype: int64

```
[5]: data.drop(["customerID"], axis =1, inplace = True)
```

```
[6]: data.head()
```

[6]: genderSeniorCitizenPartnerDependentstenurePhoneService\
0Female 0Yes No 1 No
1Male 0 No No 34 Yes
2Male 0 No No 2 Yes
3Male 0No No45 No
4Female 0 No No 2 Yes

MultipleLines InternetService OnlineSecurity OnlineBackup \
0Nophoneservice DSL No Yes
1 No DSL Yes No
2 No DSL Yes Yes
3Nophoneservice DSL Yes No
4 NoFiberoptic No No

DeviceProtectionTechSupportStreamingTVStreamingMovies Contract
0 No No No NoMonth-to-month \
1 Yes No No No Oneyear
2 No No No NoMonth-to-month
3 Yes Yes No No Oneyear
4 No No No NoMonth-to-month

PaperlessBilling PaymentMethodMonthlyChargesTotalCharges
0 Yes Electroniccheck 29.85 29.85
1 No Mailedcheck 56.951889.5 \
2 Yes Mailedcheck 53.85 108.15
3 NoBanktransfer(automatic) 42.301840.75
4 Yes Electroniccheck 70.70 151.65

Churn
0No
1No
2Yes
3No
4Yes

[7]: data.describe()

[7]: SeniorCitizen tenureMonthlyCharges
count 7043.000000 7043.000000 7043.000000
mean 0.162147 32.371149 64.761692
std 0.36861224.559481 30.090047
min 0.000000 0.000000 18.250000
25% 0.000000 9.000000 35.500000
50% 0.00000029.000000 70.350000
75% 0.00000055.000000 89.850000
max 1.000000 72.000000 118.750000

[8]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
#Column Non-NullCountDtype
--------- -------------------
0gender 7043non-nullobject
1 SeniorCitizen 7043non-null int64
2Partner 7043non-nullobject
3 Dependents 7043non-null object
4tenure 7043non-nullint64
5 PhoneService 7043non-null object
6 MultipleLines 7043non-null object
7 InternetService 7043non-null object
8 OnlineSecurity 7043non-null object
9 OnlineBackup 7043non-null object
10 DeviceProtection 7043 non-null object
11TechSupport 7043non-nullobject
12StreamingTV 7043non-nullobject
13 StreamingMovies 7043non-null object
14Contract 7043non-nullobject
15 PaperlessBilling 7043 non-null object
16 PaymentMethod 7043non-null object
17 MonthlyCharges 7043non-null float64
18TotalCharges 7043non-null object
19Churn 7043non-nullobject
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB

[9]: data.TotalCharges = pd.to_numeric(data.TotalCharges, errors='coerce')

[10]: data.isnull().any()

[10]:gender False
SeniorCitizen False
Partner False
Dependents False
tenure False
PhoneService False
MultipleLines False
InternetService False
OnlineSecurity False
OnlineBackup False
DeviceProtection False
TechSupport False
StreamingTV False
StreamingMovies False
Contract False
PaperlessBilling False
PaymentMethod False
MonthlyCharges False
TotalCharges True
Churn False
dtype: bool

[11]:
```
data.isnull().sum()
```

[11]: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 11
Churn 0
dtype: int64

[12]:
```
data["TotalCharges"].fillna(data["TotalCharges"].median() , inplace = True)
```

[13]:
```
data.isnull().sum()
```

[13]:
```
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
```

[14]:
```
data.corr()
```

```
<ipython-input-14-c44ded798807>:1: FutureWarning: The default value of
        numeric_only in DataFrame.corr is deprecated. In a future version, it will
            default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
data.corr()
```

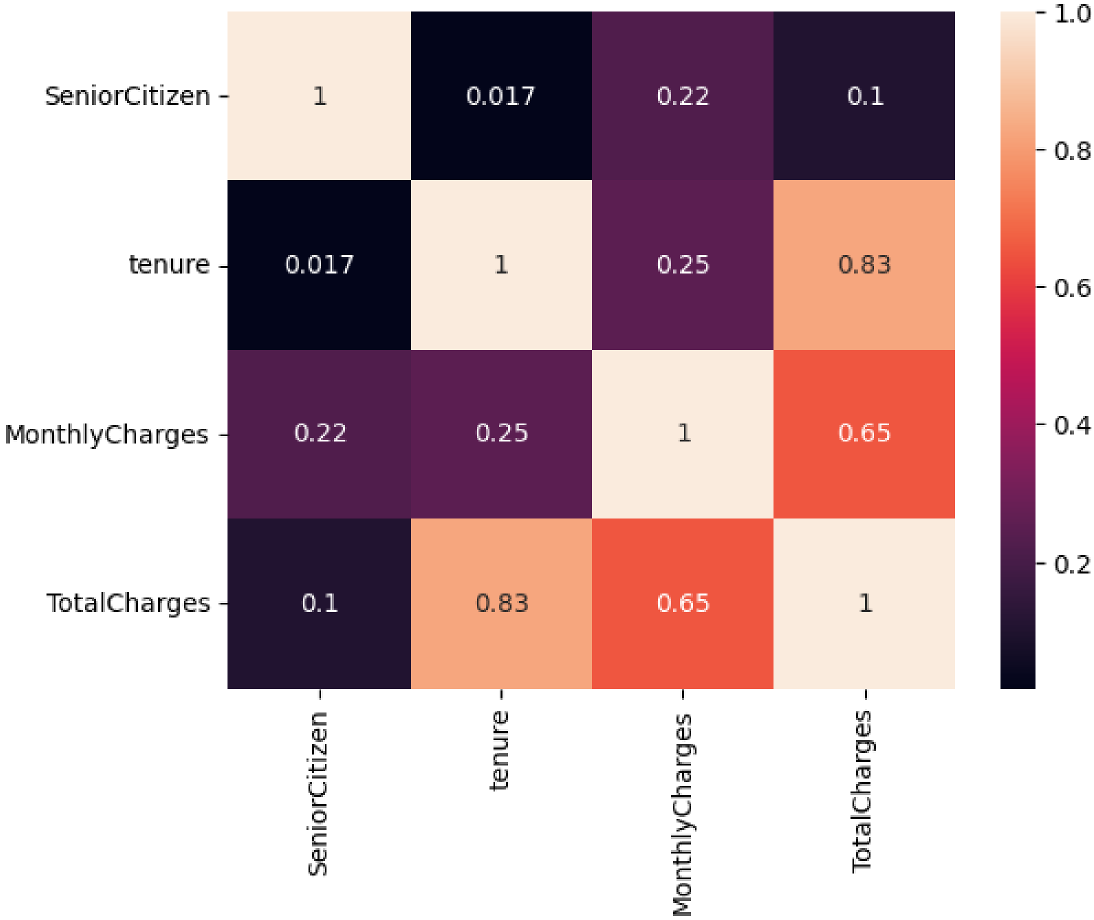
[14]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
SeniorCitizen	1.000000	0.016567	0.220173	0.102652
tenure	0.016567	1.000000	0.247900	0.825464
MonthlyCharges	0.220173	0.247900	1.000000	0.650864
TotalCharges	0.102652	0.825464	0.650864	1.000000

[15]:
```
sns.heatmap(data.corr(),annot=True)
```

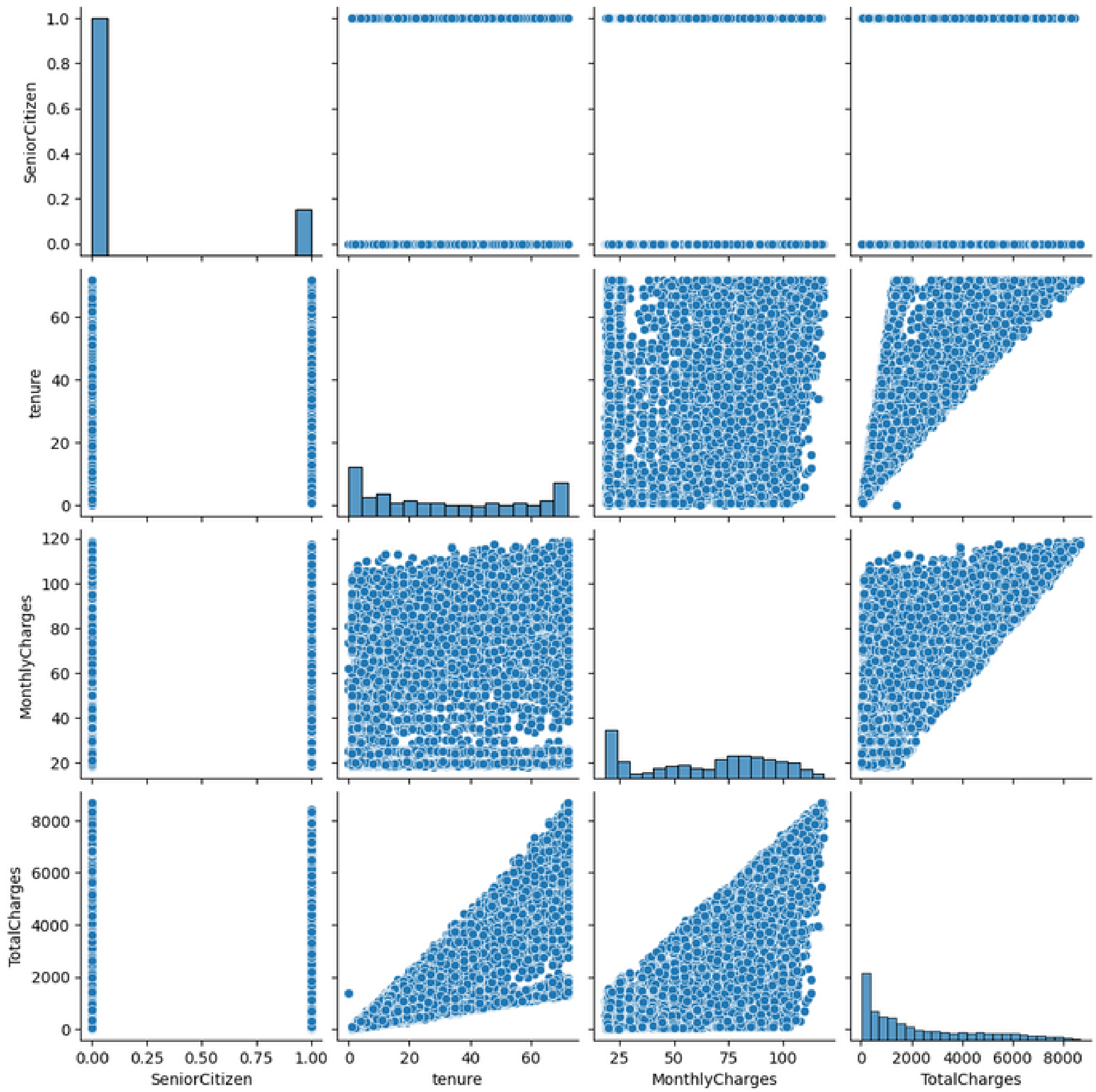
```
<ipython-input-15-6c71ac866e2e>:1: FutureWarning: The default value of
        numeric_only in DataFrame.corr is deprecated. In a future version, it will
            default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
sns.heatmap(data.corr(),annot=True)
```

[15]: <Axes: >



[16]:
```
import warnings
warnings.filterwarnings('ignore')
        sns.pairplot(data=data, markers=["^","v"], palette="inferno")
```

[16]: <seaborn.axisgrid.PairGrid at 0x7f4fdee7f7f0>



[17]: data.head(2)

[17]: gender SeniorCitizen Partner Dependents tenure PhoneService \
0Female 0Yes No 1 No
1Male 0 No No 34 Yes

MultipleLines InternetService OnlineSecurity OnlineBackup \
0Nophoneservice DSL No Yes
1 No DSL Yes No

DeviceProtectionTechSupportStreamingTVStreamingMovies Contract \
0 No No No NoMonth-to-month
1 Yes No No No Oneyear

PaperlessBilling PaymentMethodMonthlyChargesTotalChargesChurn
0 YesElectroniccheck 29.85 29.85No
1 No Mailedcheck 56.95 1889.50No

[18]:
```
for i in data:
print(data[i].unique())
```

['Female' 'Male']
[01]
['Yes' 'No']
['No' 'Yes']
[134245822102862131658492569527121123047721727
54611706343156018669331506456742354829653868
325537364164336723576114205340592444195451260
39]
['No' 'Yes']
['No phone service' 'No' 'Yes']
['DSL' 'Fiber optic' 'No']
['No' 'Yes' 'No internet service']
['Yes' 'No' 'No internet service']
['No' 'Yes' 'No internet service']
['No' 'Yes' 'No internet service']
['No' 'Yes' 'No internet service']
['No' 'Yes' 'No internet service']
['Month-to-month' 'One year' 'Two year']
['Yes' 'No']
['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
'Credit card (automatic)']
[29.85 56.95 53.85 ... 63.1 44.2 78.7]
[29.85 1889.5 108.15 ... 346.45 306.6 6844.5]
['No' 'Yes']

[19]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data["gender"] = le.fit_transform(data["gender"])
data["Partner"] = le.fit_transform(data["Partner"])
data["Dependents"] = le.fit_transform(data["Dependents"])
data["PhoneService"] = le.fit_transform(data["PhoneService"])
data["MultipleLines"] = le.fit_transform(data["MultipleLines"])
data["InternetService"] = le.fit_transform(data["InternetService"])
data["OnlineSecurity"] = le.fit_transform(data["OnlineSecurity"])
data["OnlineBackup"] = le.fit_transform(data["OnlineBackup"])
data["DeviceProtection"] = le.fit_transform(data["DeviceProtection"])
data["TechSupport"] = le.fit_transform(data["TechSupport"])
data["StreamingTV"] = le.fit_transform(data["StreamingTV"])
data["StreamingMovies"] = le.fit_transform(data["StreamingMovies"])
```

```
data["Contract"] = le.fit_transform(data["Contract"])
data["PaperlessBilling"] = le.fit_transform(data["PaperlessBilling"])
data["PaymentMethod"] = le.fit_transform(data["PaymentMethod"])
data["Churn"] = le.fit_transform(data["Churn"])
```

```
[20]: for i in data:
print(data[i].unique())
```

```
[01]
[01]
[10]
[01]
[13424582210286213165849256952712112304772172
7
546117063431560186693315064567423548296538
68
325537364164336723576114205340592444195451
260
39]
[01]
[1 0 2]
[0 1 2]
[0 2 1]
[2 0 1]
[0 2 1]
[0 2 1]
[0 2 1]
[0 2 1]
[0 1 2]
[10]
[2301]
[29.85 56.95 53.85 ... 63.1 44.2 78.7 ]
[ 29.85 1889.5 108.15 ... 346.45 306.6 6844.5 ]
[01]
```

```
[21]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
#Column Non-NullCountDtype
--------- -------------------
0gender 7043non-nullint64
1 SeniorCitizen 7043non-null int64
2Partner 7043non-nullint64
3 Dependents 7043non-null int64
4tenure 7043non-nullint64
5 PhoneService 7043non-null int64
6 MultipleLines 7043non-null int64
7 InternetService 7043non-null int64
```

```
8 OnlineSecurity 7043non-null int64
9 OnlineBackup 7043non-null int64
10 DeviceProtection 7043 non-null int64
11TechSupport 7043non-nullint64
12StreamingTV 7043non-nullint64
13 StreamingMovies 7043non-null int64
14Contract 7043non-nullint64
15 PaperlessBilling 7043 non-null int64
16 PaymentMethod 7043non-null int64
17 MonthlyCharges 7043non-null float64
18TotalCharges 7043non-null float64
19Churn 7043non-nullint64
dtypes: float64(2), int64(18)
memory usage: 1.1 MB
```

[22]: data.corr()

```
[22]: genderSeniorCitizenPartnerDependentstenure \
gender 1.000000 -0.001874-0.0018080.0105170.005106
SeniorCitizen -0.001874 1.0000000.016479 -0.2111850.016567
Partner -0.001808 0.0164791.0000000.4526760.379697
Dependents 0.010517 -0.2111850.452676 1.0000000.159712
tenure 0.005106 0.0165670.3796970.1597121.000000
PhoneService -0.006488 0.0085760.017706 -0.0017620.008448
MultipleLines -0.006739 0.1461850.142410 -0.0249910.343032
InternetService-0.000863 -0.0323100.000891 0.044590-0.030359
OnlineSecurity -0.015017 -0.128221 0.150828 0.152166 0.325468
OnlineBackup -0.012057 -0.0136320.153130 0.0910150.370876
DeviceProtection0.000549 -0.0213980.166330 0.0805370.371105
TechSupport -0.006825 -0.1512680.126733 0.1335240.322942
StreamingTV -0.006421 0.0307760.137341 0.0468850.289373
StreamingMovies-0.008743 0.0472660.129574 0.0213210.296866
Contract 0.000126 -0.1425540.2948060.2431870.671607
PaperlessBilling-0.011754 0.156530-0.014877-0.1113770.006152
PaymentMethod 0.017352 -0.038551-0.154798 -0.040292-0.370436
MonthlyCharges -0.014569 0.220173 0.096848 -0.113890 0.247900
TotalCharges -0.000002 0.1026520.318364 0.0635930.825464
Churn -0.008612 0.150889-0.150448-0.164221-0.352229

PhoneService MultipleLines InternetService \
gender -0.006488 -0.006739 -0.000863
SeniorCitizen 0.008576 0.146185 -0.032310
Partner 0.017706 0.142410 0.000891
Dependents -0.001762 -0.024991 0.044590
tenure 0.0084480.343032-0.030359
PhoneService 1.000000 -0.020538 0.387436
MultipleLines -0.020538 1.000000 -0.109216
```

InternetService	0.387436	-0.109216	1.000000
OnlineSecurity	-0.015198	0.007141	-0.028416
OnlineBackup	0.024105	0.117327	0.036138
DeviceProtection	0.003727	0.122318	0.044944
TechSupport	-0.019158	0.011466	-0.026047
StreamingTV	0.055353	0.175059	0.107417
StreamingMovies	0.043870	0.180957	0.098350
Contract	0.002247	0.110842	0.099721
PaperlessBilling	0.016505	0.165146	-0.138625
PaymentMethod	-0.004184	-0.176793	0.086140
MonthlyCharges	0.247398	0.433576	-0.323260
TotalCharges	0.113013	0.452849	-0.175588
Churn	0.011942	0.038037	-0.047291

	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
gender	-0.015017	-0.012057	0.000549	-0.006825
SeniorCitizen	-0.128221	-0.013632	-0.021398	-0.151268
Partner	0.150828	0.153130	0.166330	0.126733
Dependents	0.152166	0.091015	0.080537	0.133524
tenure	0.325468	0.370876	0.371105	0.322942
PhoneService	-0.015198	0.024105	0.003727	-0.019158
MultipleLines	0.007141	0.117327	0.122318	0.011466
InternetService	-0.028416	0.036138	0.044944	-0.026047
OnlineSecurity	1.000000	0.185126	0.175985	0.285028
OnlineBackup	0.185126	1.000000	0.187757	0.195748
DeviceProtection	0.175985	0.187757	1.000000	0.240593
TechSupport	0.285028	0.195748	0.240593	1.000000
StreamingTV	0.044669	0.147186	0.276652	0.161305
StreamingMovies	0.055954	0.136722	0.288799	0.161316
Contract	0.374416	0.280980	0.350277	0.425367
PaperlessBilling	-0.157641	-0.013370	-0.038234	-0.113600
PaymentMethod	-0.096726	-0.124847	-0.135750	-0.104670
MonthlyCharges	-0.053878	0.119777	0.163652	-0.008682
TotalCharges	0.253935	0.375063	0.388562	0.276343
Churn	-0.289309	-0.195525	-0.178134	-0.282492

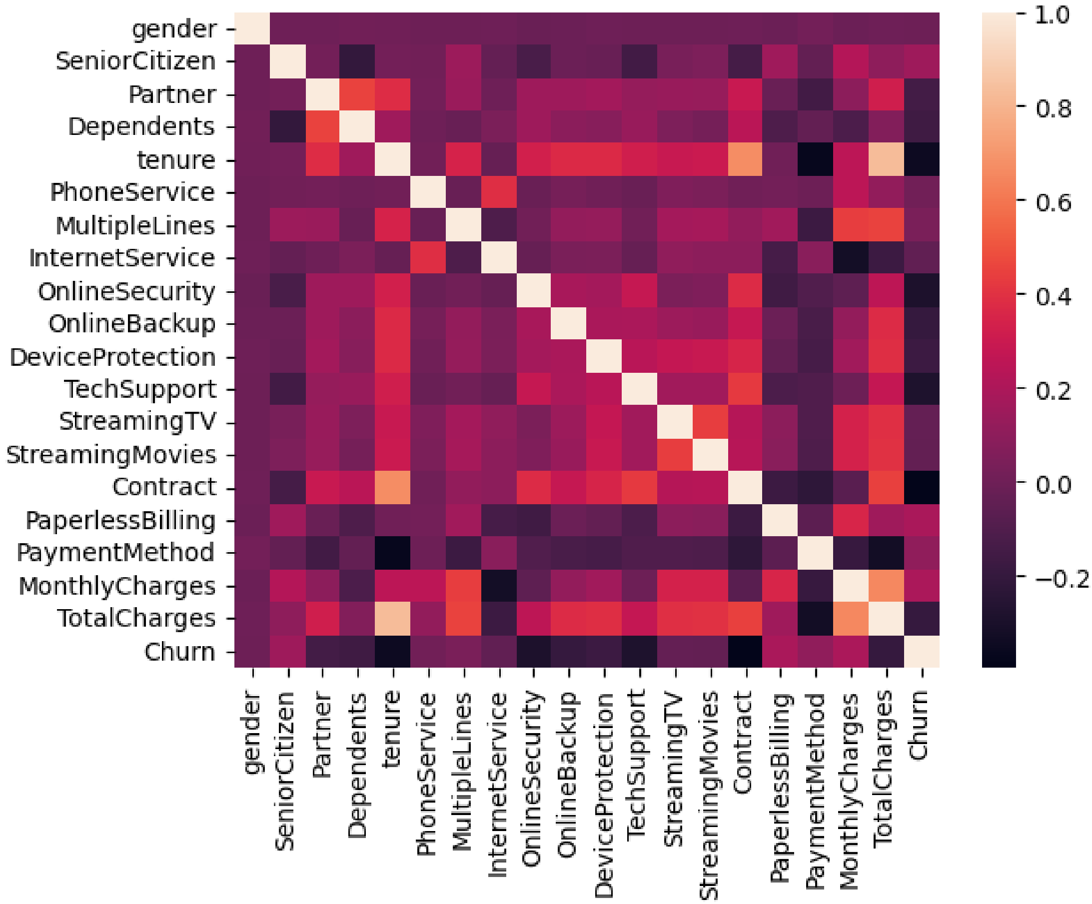
	StreamingTV	StreamingMovies	Contract	PaperlessBilling
gender	-0.006421	-0.008743	0.000126	-0.011754
SeniorCitizen	0.030776	0.047266	-0.142554	0.156530
Partner	0.137341	0.129574	0.294806	-0.014877
Dependents	0.046885	0.021321	0.243187	-0.111377
tenure	0.289373	0.296866	0.671607	0.006152
PhoneService	0.055353	0.043870	0.002247	0.016505
MultipleLines	0.175059	0.180957	0.110842	0.165146
InternetService	0.107417	0.098350	0.099721	-0.138625
OnlineSecurity	0.044669	0.055954	0.374416	-0.157641
OnlineBackup	0.147186	0.136722	0.280980	-0.013370

DeviceProtection 0.276652	0.2887990.350277 -0.038234
TechSupport 0.161305	0.1613160.425367 -0.113600
StreamingTV 1.000000	0.4347720.227116 0.096642
StreamingMovies 0.434772	1.0000000.231226 0.083700
Contract 0.227116	0.2312261.000000 -0.176733
PaperlessBilling 0.096642	0.083700-0.176733 1.000000
PaymentMethod -0.104234	-0.111241-0.227543 -0.062904
MonthlyCharges 0.336706	0.335459-0.074195 0.352150
TotalCharges 0.392046	0.3980450.448564 0.158055
Churn -0.036581	-0.038492-0.396713 0.191825

PaymentMethod	MonthlyCharges TotalCharges Churn
gender 0.017352	-0.014569 -0.000002-0.008612
SeniorCitizen -0.038551	0.220173 0.1026520.150889
Partner -0.154798	0.096848 0.318364-0.150448
Dependents -0.040292	-0.113890 0.063593-0.164221
tenure -0.370436	0.247900 0.825464-0.352229
PhoneService -0.004184	0.247398 0.1130130.011942
MultipleLines -0.176793	0.433576 0.4528490.038037
InternetService 0.086140	-0.323260 -0.175588-0.047291
OnlineSecurity -0.096726	-0.053878 0.253935-0.289309
OnlineBackup -0.124847	0.119777 0.375063-0.195525
DeviceProtection -0.135750	0.163652 0.388562-0.178134
TechSupport -0.104670	-0.008682 0.276343-0.282492
StreamingTV -0.104234	0.336706 0.392046-0.036581
StreamingMovies -0.111241	0.335459 0.398045-0.038492
Contract -0.227543	-0.074195 0.448564-0.396713
PaperlessBilling -0.062904	0.352150 0.1580550.191825
PaymentMethod 1.000000	-0.193407 -0.3305110.107062
MonthlyCharges -0.193407	1.000000 0.6508640.193356
TotalCharges -0.330511	0.650864 1.000000-0.199037
Churn 0.107062	0.193356 -0.1990371.000000

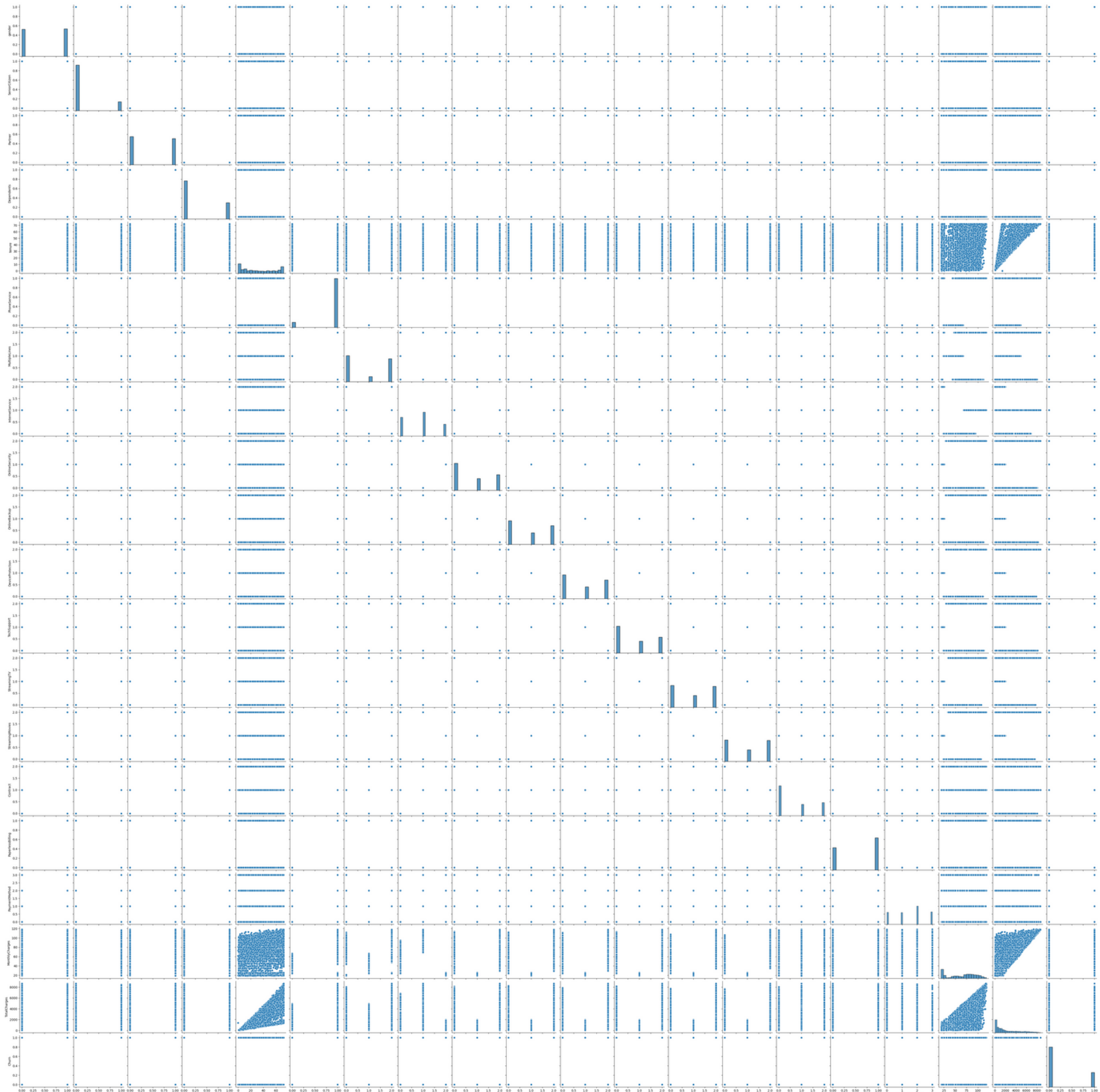
[23]:
```
sns.heatmap(data.corr(), annot=False)
```

[23]: <Axes: >



[24]:
```
sns.pairplot(data=data, markers=["^","v"], palette="inferno")
```
[24]:

<seaborn.axisgrid.PairGrid at 0x7f4fe1872e50>



[25]:
```
x=data.iloc[:,0:19].values
y=data.iloc[:,19:20].values
```

[26]:
```
from imblearn.over_sampling import SMOTE
```

[27]:
```
smt = SMOTE()

x_resample, y_resample = smt.fit_resample(x,y)
```

[28]:
```
x_resample
```

[28]: array([[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, ...,
2.00000000e+00, 2.98500000e+01, 2.98500000e+01],

```
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
3.00000000e+00, 5.69500000e+01, 1.88950000e+03],
[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
3.00000000e+00, 5.38500000e+01, 1.08150000e+02],
...,
[4.00931754e-01, 4.00931754e-01, 4.00931754e-01, ...,
8.01863509e-01, 9.79158537e+01, 1.23652553e+03],
[4.24460210e-01, 0.00000000e+00, 5.75539790e-01, ...,
8.48920420e-01, 6.88413664e+01, 3.48644605e+02],
[1.00000000e+00, 0.00000000e+00, 9.36376990e-01, ...,
2.00000000e+00, 7.02809131e+01, 7.38314521e+02]])
```

[29]:
```
y_resample
```

[29]: array([0, 0, 1, ..., 1, 1, 1])

[30]:
```
x.shape, x_resample.shape
```

[30]: ((7043, 19), (10348, 19))

[31]:
```
y.shape, y_resample.shape
```

[31]: ((7043, 1), (10348,))

[32]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = 
    ↪train_test_split(x_resample,y_resample,test_size = 0.2, random_state = 0)
```

[33]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

[34]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
```

[34]: LogisticRegression(random_state=0)

[35]:
```
lr_pred = lr.predict(x_test)
```

[36]:
```
lr_pred
```

[36]: array([1, 0, 1, ..., 0, 0, 1])

[37]:
```
y_test
```

[37]: array([1, 0, 1, ..., 1, 0, 1])

```
[38]: from sklearn.metrics import accuracy_score
lr_acc = accuracy_score(lr_pred,y_test)
```

```
[39]: lr_acc
```

[39]: 0.7816425120772947

```
[40]: from sklearn.metrics import confusion_matrix
lr_cm = confusion_matrix(lr_pred,y_test)
```

```
[41]: lr_cm
```

[41]: array([[757, 176],
[276, 861]])

```
[42]: from sklearn.tree import DecisionTreeClassifier
          dtc = DecisionTreeClassifier(random_state = 0,criterion= "entropy")
dtc.fit(x_train,y_train)
```

[42]: DecisionTreeClassifier(criterion='entropy', random_state=0)

```
[43]: dtc_pred = dtc.predict(x_test)
```

```
[44]: dtc_pred
```

[44]: array([1, 1, 1, ..., 1, 1, 1])

```
[45]: dtc_acc = accuracy_score(dtc_pred,y_test)
```

```
[46]: dtc_acc
```

[46]: 0.7463768115942029

```
[47]: dtc_cm = confusion_matrix(dtc_pred,y_test)
```

```
[48]: dtc_cm
```

[48]: array([[581, 73],
[452, 964]])

```
[49]: from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 10,criterion =␣
↪"entropy",random_state=0)rfc.fit(x_train,y_train)
```

[49]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0) [50]:

```
rfc_pred = rfc.predict(x_test)
```

[51]: rfc_pred

[51]: array([1, 1, 1, ..., 0, 1, 1])

[52]: rfc_acc = accuracy_score(rfc_pred,y_test)

[53]: rfc_acc

[53]: 0.755072463768116

[54]: rfc_cm = confusion_matrix(rfc_pred,y_test)

[55]: rfc_cm

[55]: array([[593, 67],
[440, 970]])

[56]:
```
from sklearn.svm import SVC
svm = SVC(kernel="linear")
svm.fit(x_train,y_train)
```

[56]: SVC(kernel='linear')

[57]: svm_pred = svm.predict(x_test)

[58]: svm_pred

[58]: array([1, 0, 1, ..., 1, 0, 1])

[59]: svm_acc = accuracy_score(svm_pred,y_test)

[60]: svm_acc

[60]: 0.7695652173913043

[61]: svm_cm = confusion_matrix(svm_pred,y_test)

[62]: svm_cm

[62]: array([[717, 161],
[316, 876]])

[63]:
```
import pickle
pickle.dump(rfc,open("churn3.pkl" ,"wb"))
```