In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
import io
%cd "C:\Users\deepe\OneDrive\Desktop\Python Datasets\Telecom"
```

```
C:\Users\deepe\OneDrive\Desktop\Python Datasets\Telecom
```

In [3]:

```
telust=pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn (2).csv")
```

In [4]:

```
telust.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 
 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 [5]:

```
telust.head()
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLi
0	7590-VHVEG	Female	0	Yes	No	1	No	No ph serv
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	No ph serv
4	9237-HQITU	Female	0	No	No	2	Yes	

5 rows × 21 columns

In [6]:

```
telust.tail()
```

Out[6]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multipl
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	Nc
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	
7042	3186-AJIEK	Male	0	No	No	66	Yes	

5 rows × 21 columns

In [7]:

```
telust.columns
```

Out[7]:

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSuppor
t',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [8]:

```
objcols=telust[['gender', 'SeniorCitizen', 'Partner', 'Dependents','PhoneService', 'Mult
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod','Churn']]
```

In [9]:

```
objcols.head()
```

Out[9]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
0	Female	0	Yes	No	No	No phone service	DSL
1	Male	0	No	No	Yes	No	DSL
2	Male	0	No	No	Yes	No	DSL
3	Male	0	No	No	No	No phone service	DSL
4	Female	0	No	No	Yes	No	Fiber optic

In [10]:

```
numcols=telust[['tenure','MonthlyCharges', 'TotalCharges']]
```

In [11]:

```
numcols['TotalCharges']=pd.to_numeric(numcols.TotalCharges,errors='coerce')
```

```
C:\Users\deepe\AppData\Local\Temp\ipykernel_15636\2108254401.py:1: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  numcols['TotalCharges']=pd.to_numeric(numcols.TotalCharges,errors='coerc
e')
```

In [12]:

```
numcols.isnull().sum().sort_values(ascending=False)/numcols.shape[0]
```

Out[12]:

```
TotalCharges      0.001562
tenure            0.000000
MonthlyCharges    0.000000
dtype: float64
```

In [13]:

```
# fill the missing values useing median maximum we use median
for col in numcols.columns:
    numcols[col]=numcols[col].fillna(numcols[col].median())
```

```
C:\Users\deepe\AppData\Local\Temp\ipykernel_15636\1347123924.py:3: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  numcols[col]=numcols[col].fillna(numcols[col].median())
```

In [14]:

```
numcols.head()
```

Out[14]:

	tenure	MonthlyCharges	TotalCharges
0	1	29.85	29.85
1	34	56.95	1889.50
2	2	53.85	108.15
3	45	42.30	1840.75
4	2	70.70	151.65

In [15]:

```
numcols.describe()
```

Out[15]:

	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000
mean	32.371149	64.761692	2281.916928
std	24.559481	30.090047	2265.270398
min	0.000000	18.250000	18.800000
25%	9.000000	35.500000	402.225000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.850000	3786.600000
max	72.000000	118.750000	8684.800000

In [16]:

```
telustdf=pd.concat([objcols,numcols],axis=1)
```

In [17]:

```
telustdf.info()
```

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

In [18]:

```
telustdf.head()
```

Out[18]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
0	Female	0	Yes	No	No	No phone service	DSL
1	Male	0	No	No	Yes	No	DSL
2	Male	0	No	No	Yes	No	DSL
3	Male	0	No	No	No	No phone service	DSL
4	Female	0	No	No	Yes	No	Fiber optic

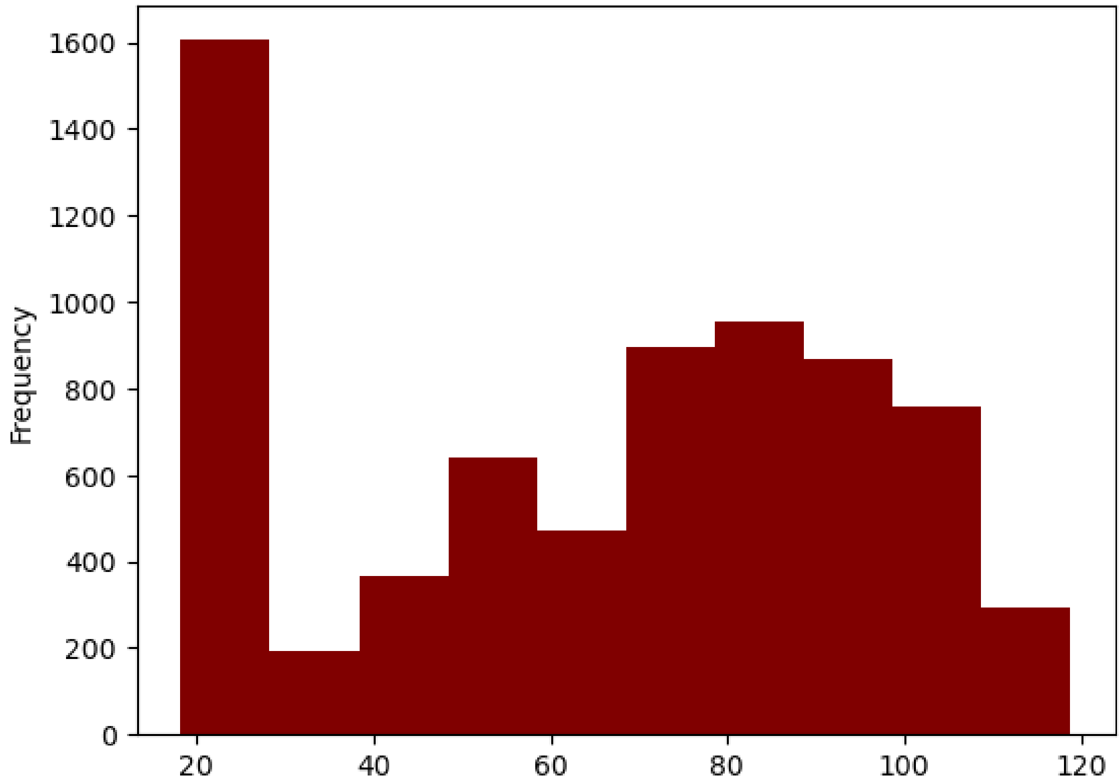
EDA

In [19]:

```
telustdf.MonthlyCharges.plot(kind="hist",color="maroon")
```

Out[19]:

<Axes: ylabel='Frequency'>

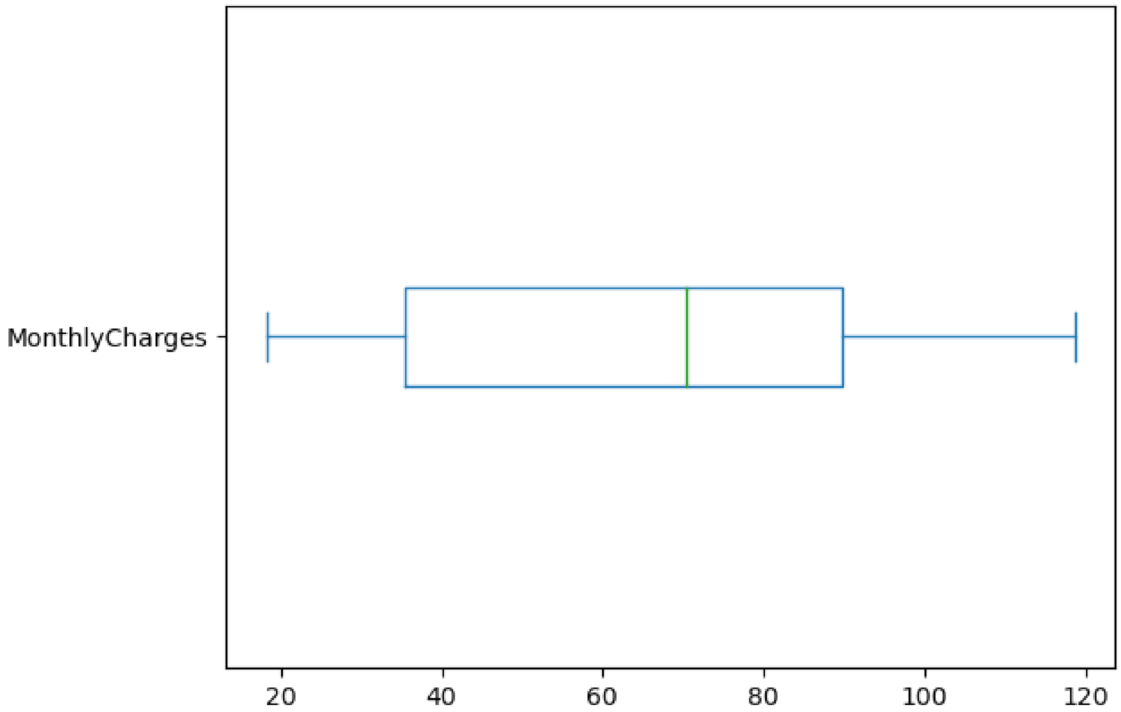


In [20]:

```
telustdf.MonthlyCharges.plot(kind="box",vert=False)
```

Out[20]:

```
<Axes: >
```

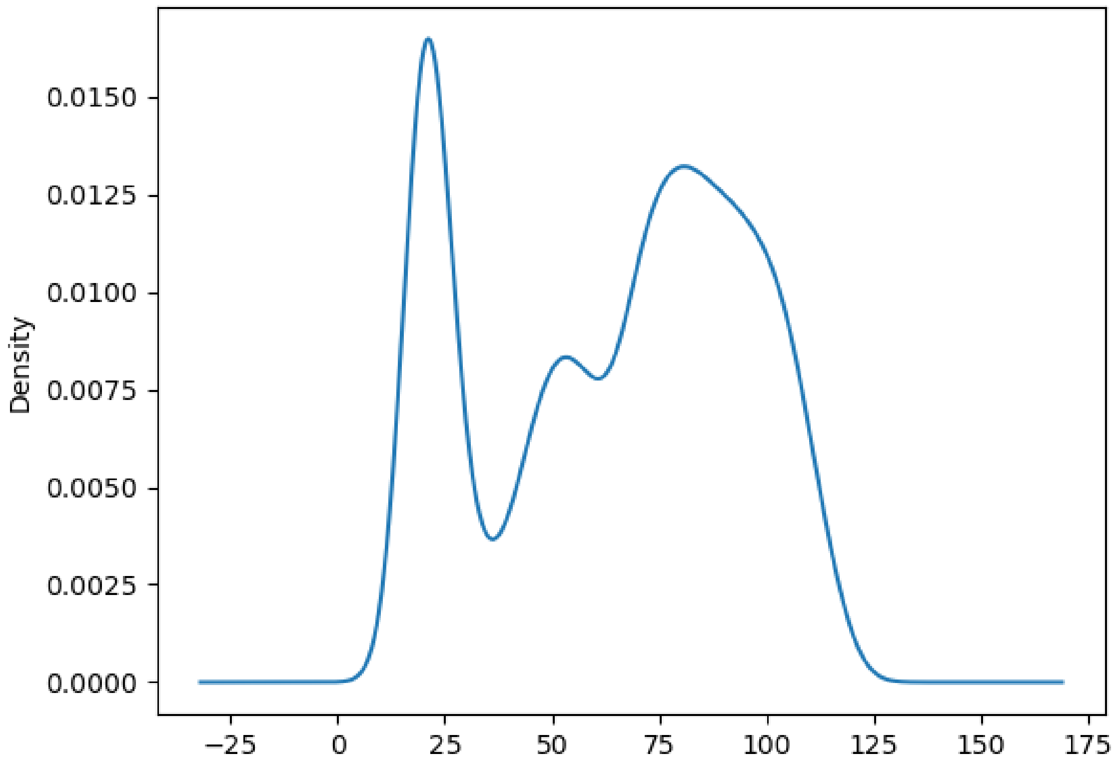


In [21]:

```
telustdf.MonthlyCharges.plot(kind="density")
```

Out[21]:

<Axes: ylabel='Density'>

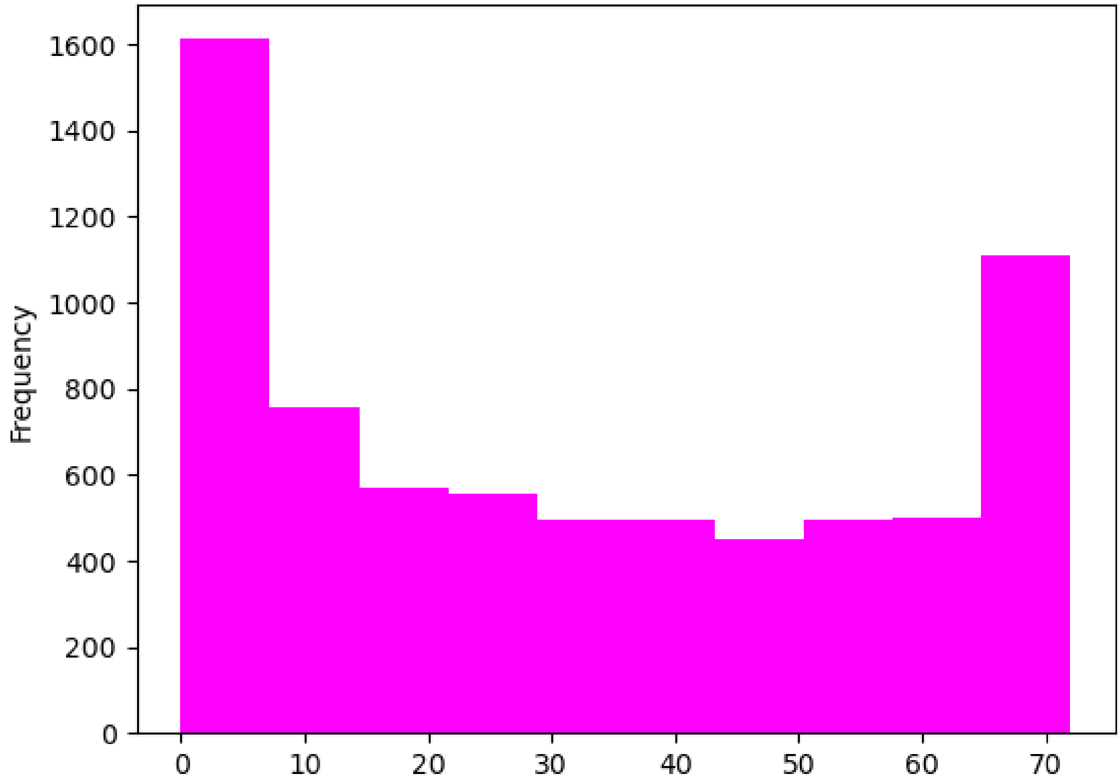


In [22]:

```
telustdf.tenure.plot(kind="hist",color="magenta")
```

Out[22]:

```
<Axes: ylabel='Frequency'>
```

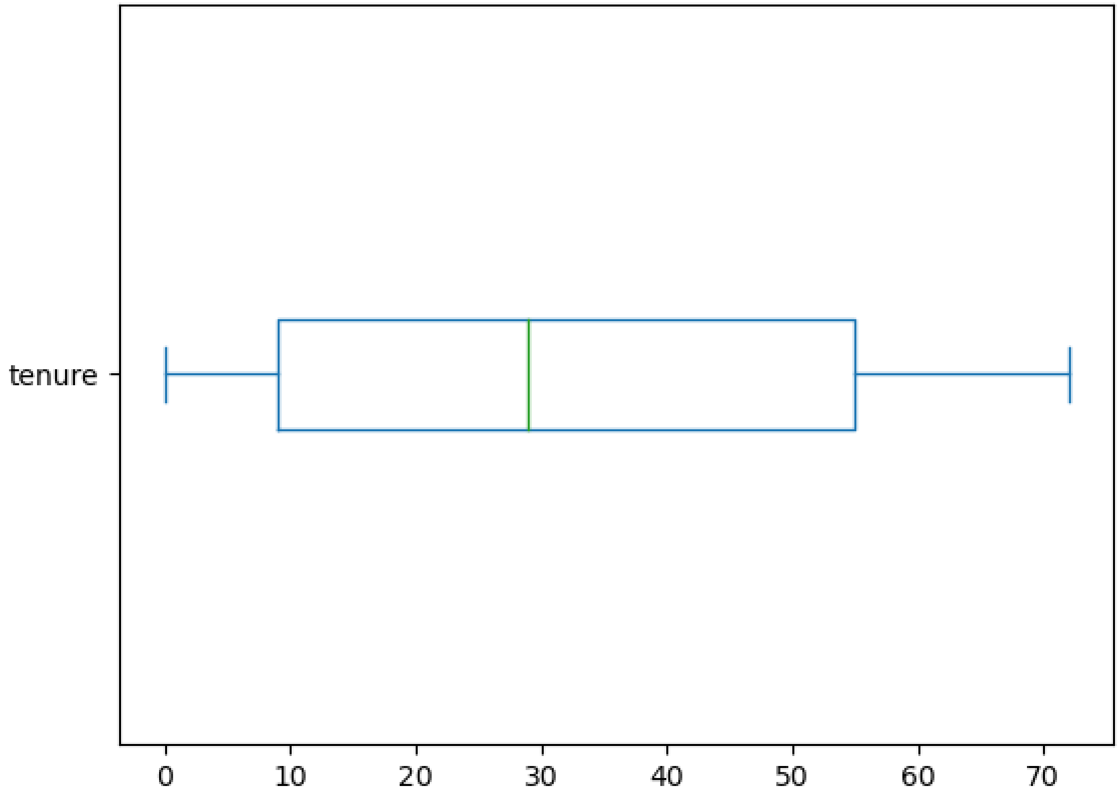


In [23]:

```
telustdf.tenure.plot(kind="box",vert=False)
```

Out[23]:

```
<Axes: >
```

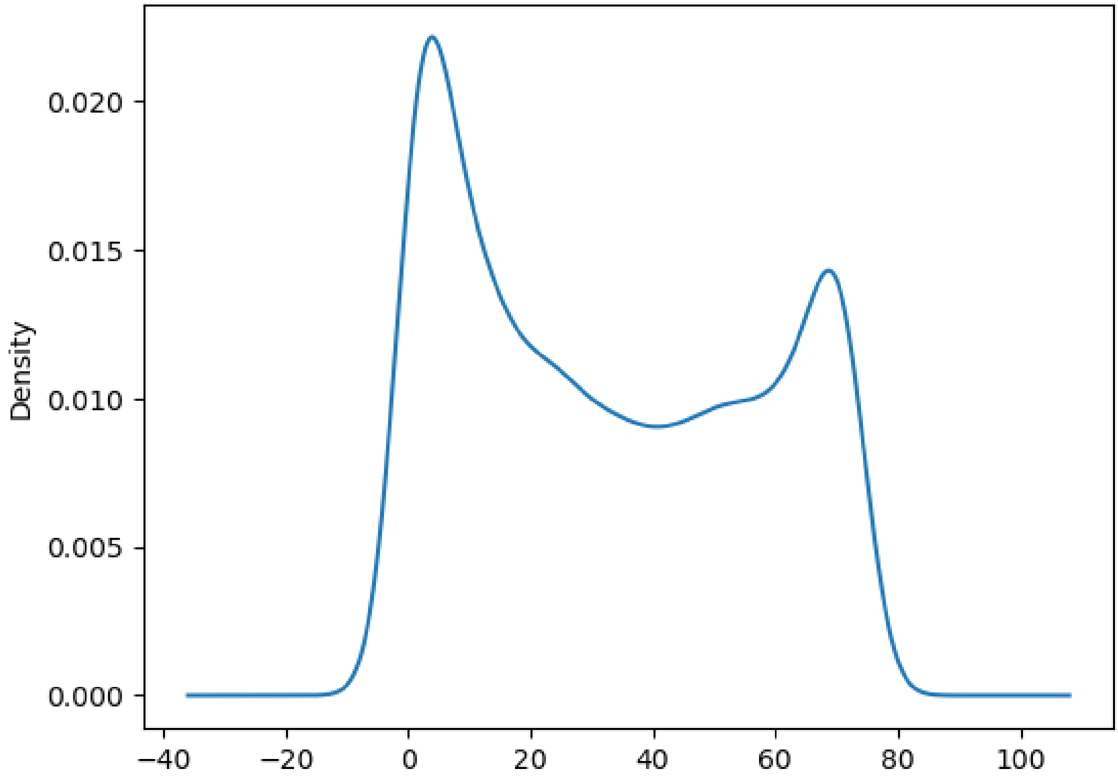


In [24]:

```
telustdf.tenure.plot(kind="density")
```

Out[24]:

```
<Axes: ylabel='Density'>
```



Frequency Counts

In [25]:

```
telustdf.Churn.value_counts()
```

Out[25]:

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

In [26]:

```
telustdf.gender.value_counts()
```

Out[26]:

```
Male      3555
Female    3488
Name: gender, dtype: int64
```

In [27]:

```
telustdf.SeniorCitizen.value_counts()
```

Out[27]:

```
0    5901
1    1142
Name: SeniorCitizen, dtype: int64
```

In [28]:

```
telustdf.PaymentMethod.value_counts()
```

Out[28]:

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

Cross Tabulation And Visualisation

In [29]:

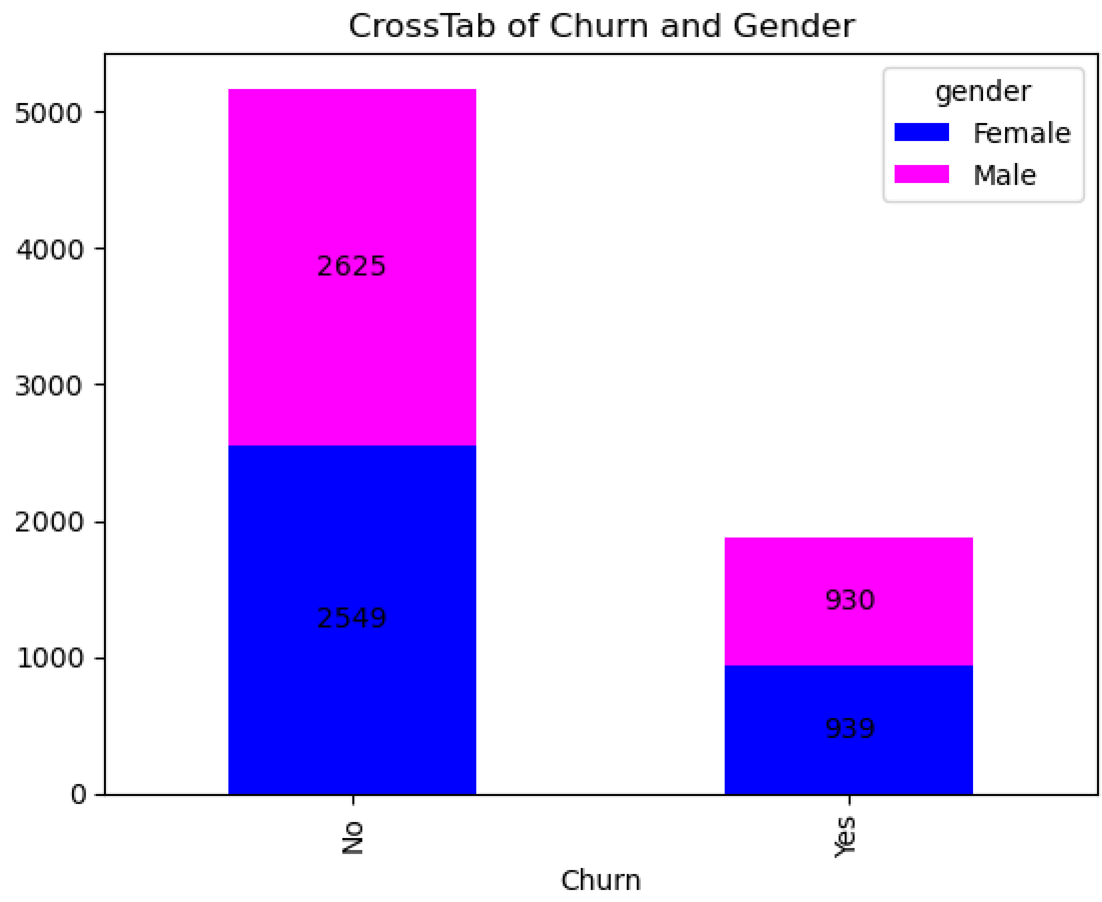
```
# Cross tabulation of Churn & gender

pd.crosstab(telustdf.Churn,telustdf.gender)
```

Out[29]:

gender	Female	Male
Churn		
No	2549	2625
Yes	939	930

In [30]:

```
# Visualisation

df=pd.crosstab(telustdf.Churn,telustdf.gender)
ax=df.plot.bar(stacked=True,color=["blue","magenta"],title="CrossTab of Churn and Gender
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```



In [31]:

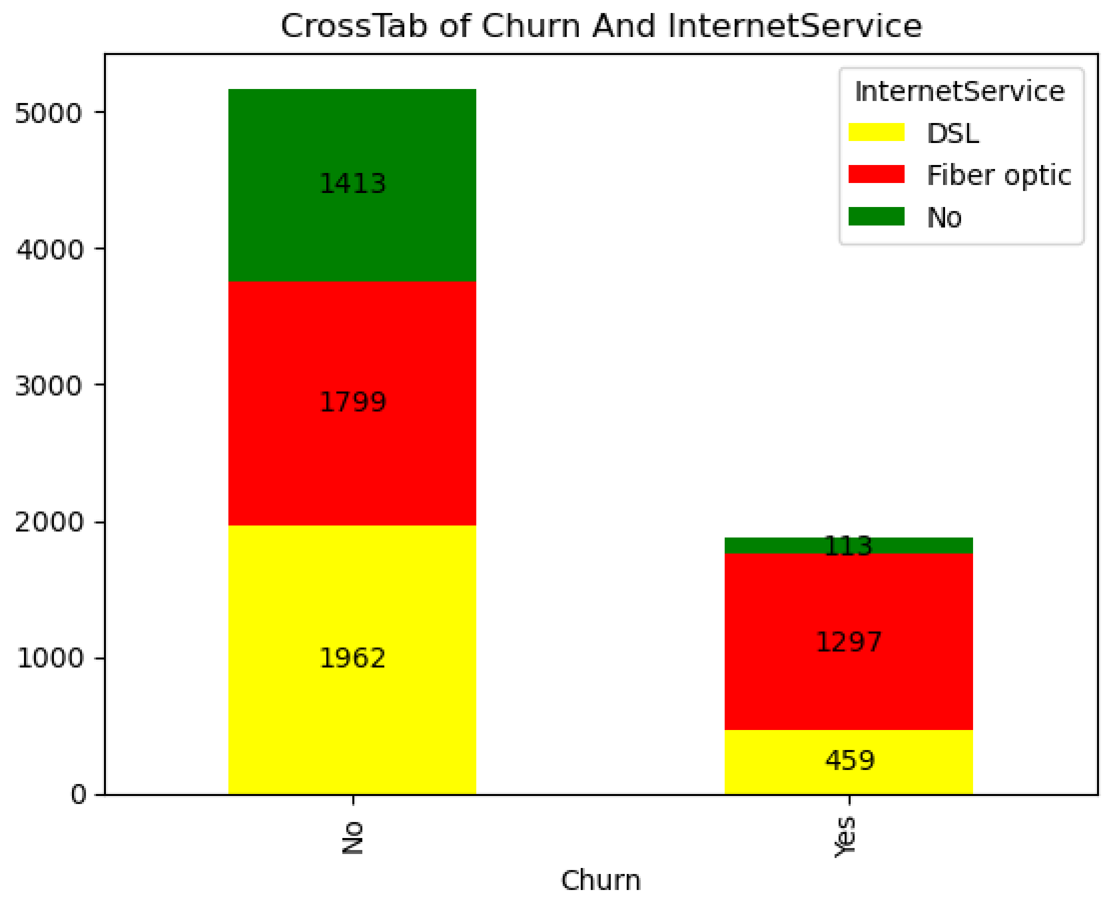
```
# Cross tabulation Churn & InternetService

pd.crosstab(telustdf.Churn,telustdf.InternetService)
```

Out[31]:

InternetService	DSL	Fiber optic	No
Churn			
No	1962	1799	1413
Yes	459	1297	113

In [32]:

```
# Visualisation

df=pd.crosstab(telustdf.Churn,telustdf.InternetService)
ax=df.plot.bar(stacked=True,color=["yellow","red","green"],title="CrossTab of Churn And
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```



In [33]:

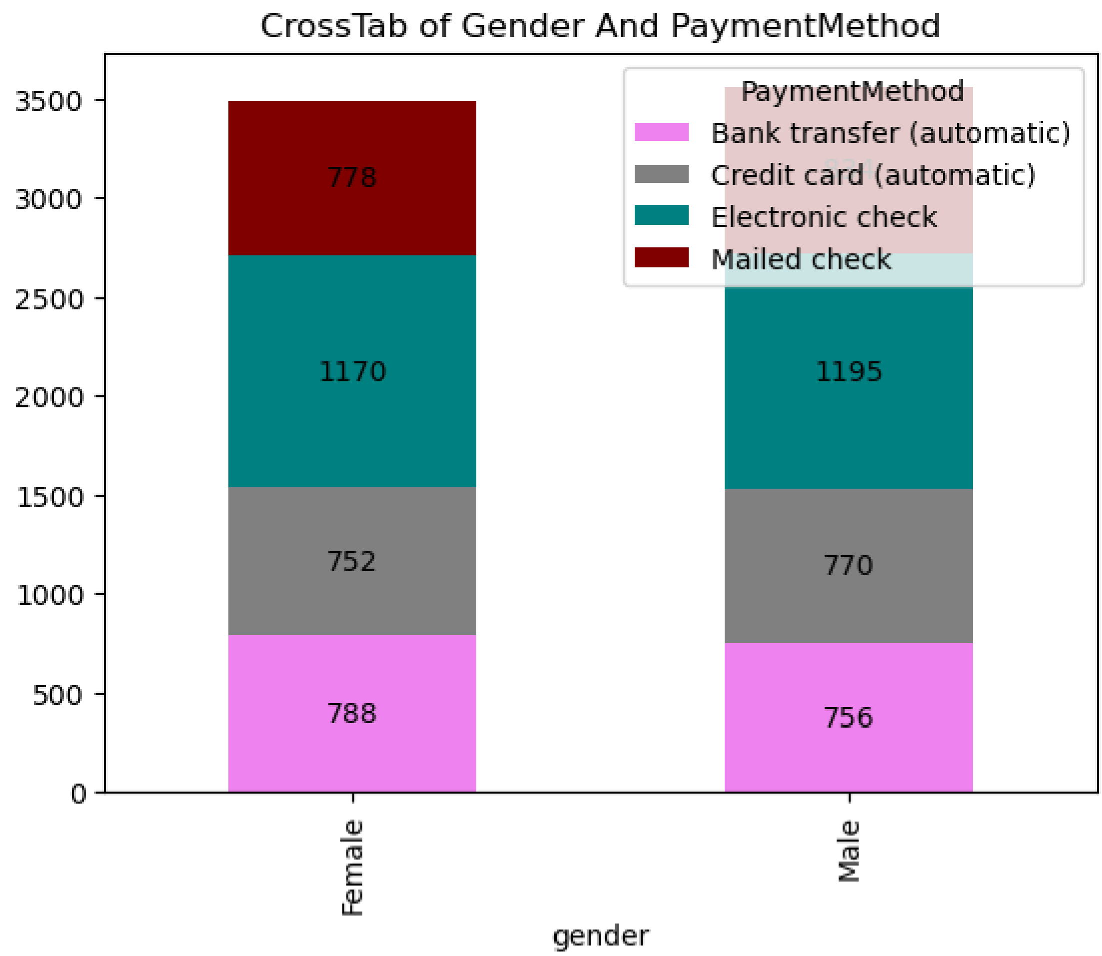
```
# Cross tabulation gender & PaymentMethod
pd.crosstab(telustdf.gender,telustdf.PaymentMethod)
```

Out[33]:

PaymentMethod gender	Bank transfer (automatic)	Credit card (automatic)	Electronic check	Mailed check
Female	788	752	1170	778
Male	756	770	1195	834

In [34]:

```
# Visualisation

df=pd.crosstab(telustdf.gender,telustdf.PaymentMethod)
ax=df.plot.bar(stacked=True,color=["violet","grey","teal","maroon"],title="CrossTab of G
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```



groupby() and Visualisation

In [35]:

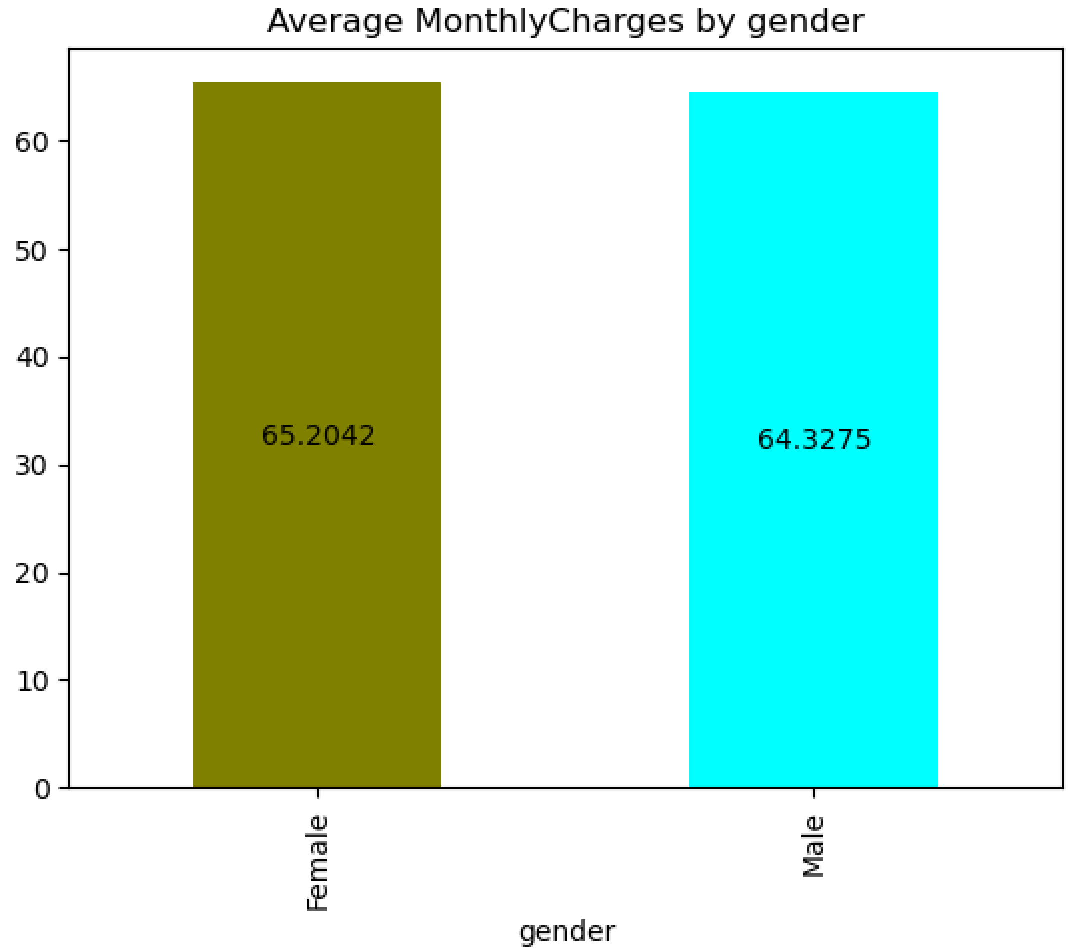
```
# Average MonthlyCharges by gender

telustdf.MonthlyCharges.groupby(telustdf.gender).mean()
```

Out[35]:

```
gender
Female    65.204243
Male      64.327482
Name: MonthlyCharges, dtype: float64
```

In [36]:

```
# Visualisation

ax=telustdf.MonthlyCharges.groupby(telustdf.gender).mean().plot(kind="bar",color=["olive
                                                                title="Average MonthlyCh
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```

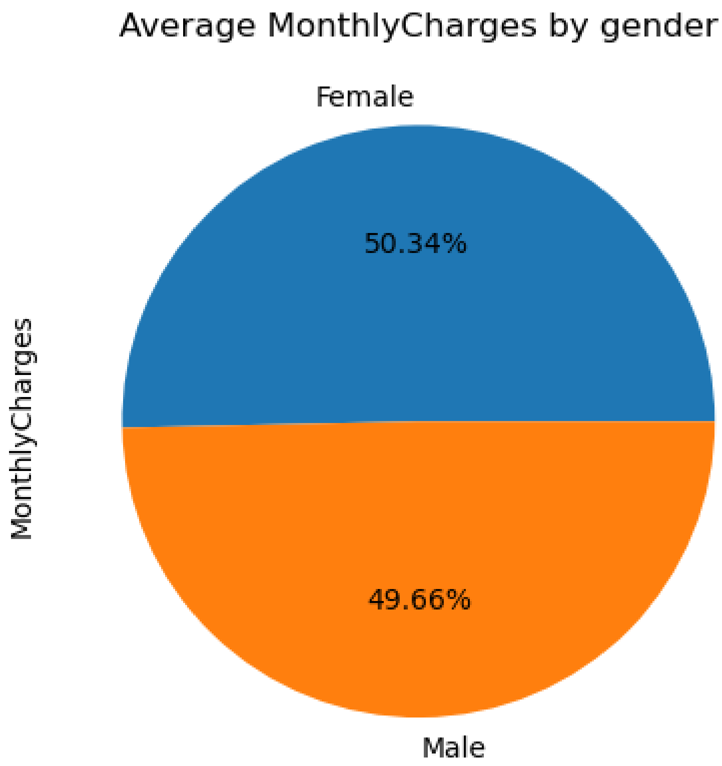


In [37]:

```
telustdf.MonthlyCharges.groupby(telustdf.gender).mean().plot(kind="pie",autopct="%.2f%%"
```

Out[37]:

```
<Axes: title={'center': 'Average MonthlyCharges by gender'}, ylabel='Month
lyCharges'>
```



In [38]:

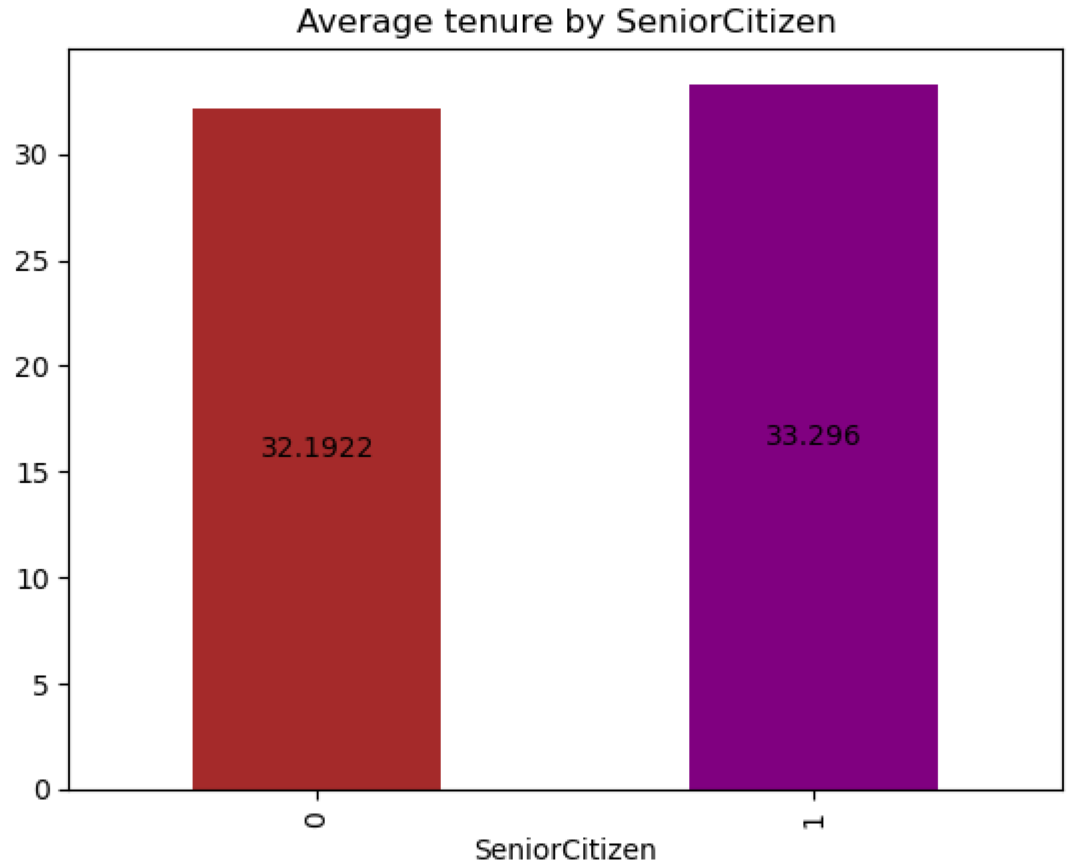
```
# Average tenure by SeniorCitizen

telustdf.tenure.groupby(telustdf.SeniorCitizen).mean()
```

Out[38]:

```
SeniorCitizen
0    32.192171
1    33.295972
Name: tenure, dtype: float64
```

In [39]:

```
ax=telustdf.tenure.groupby(telustdf.SeniorCitizen).mean().plot(kind="bar",color=["Brown"
                                                                title="Average tenure by
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```

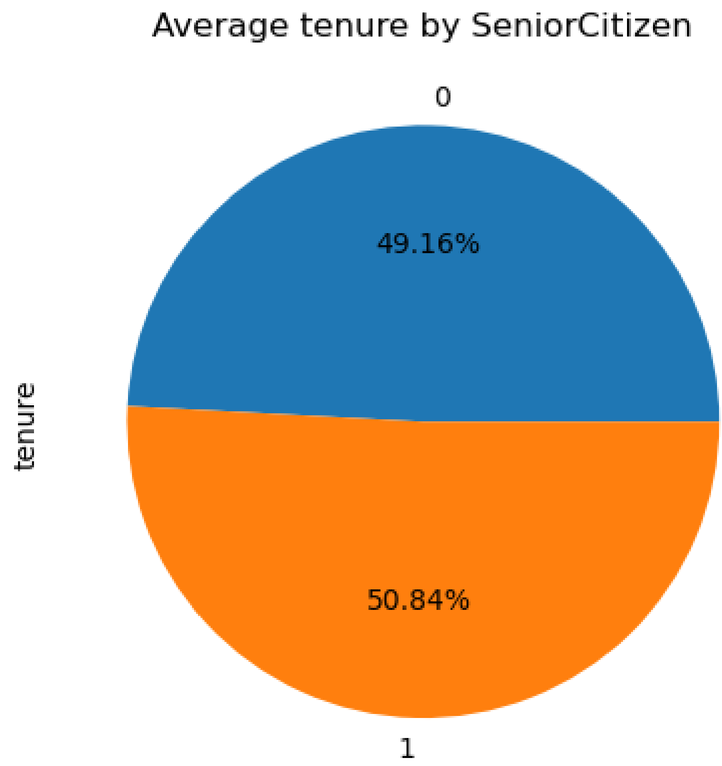


In [40]:

```
telustdf.tenure.groupby(telustdf.SeniorCitizen).mean().plot(kind="pie",autopct="%.2f%%",
```

Out[40]:

```
<Axes: title={'center': 'Average tenure by SeniorCitizen'}, ylabel='tenur
e'>
```



In [41]:

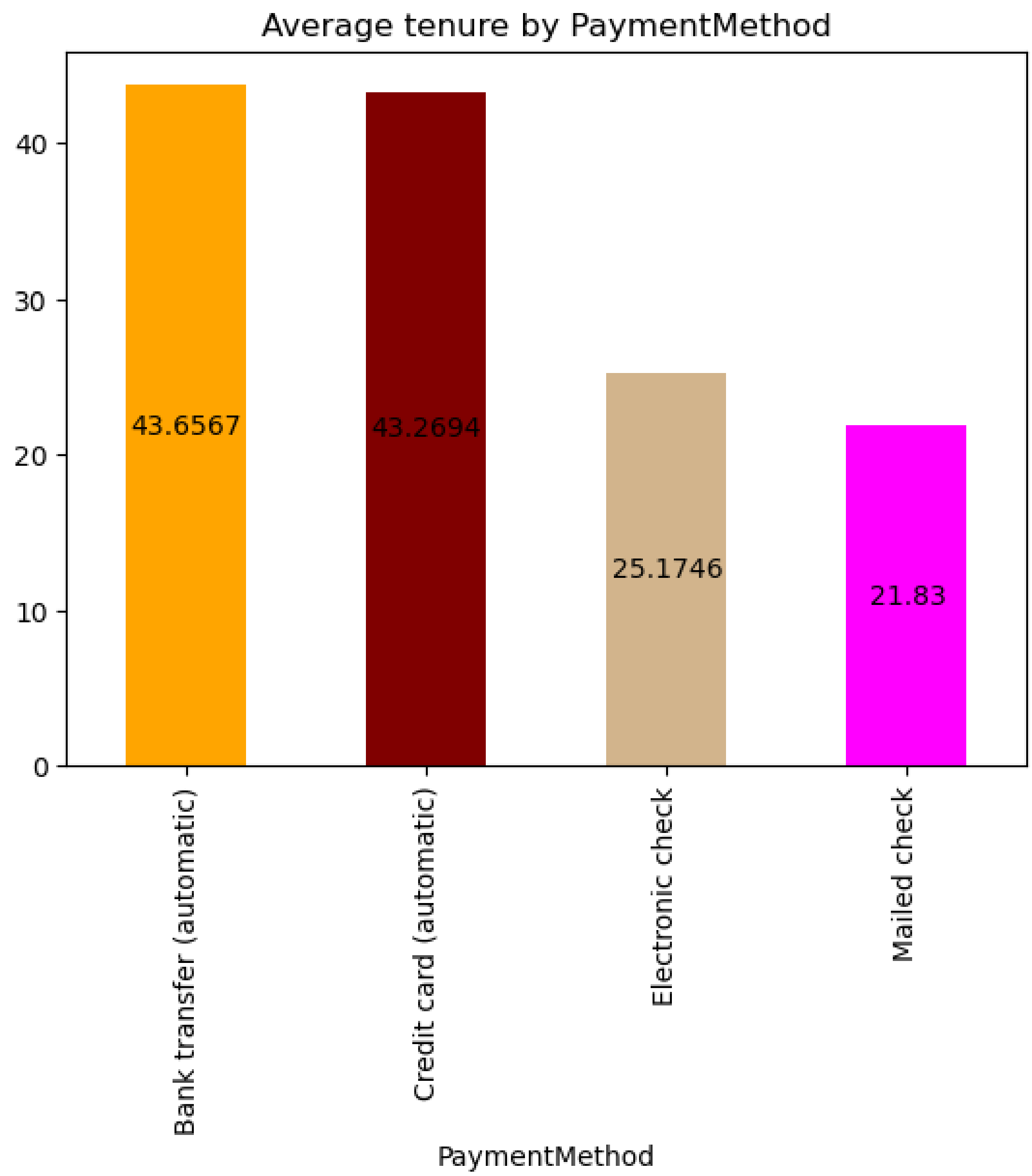
```
# Average tenure by PaymentMethod

telustdf.tenure.groupby(telustdf.PaymentMethod).mean()
```

Out[41]:

```
PaymentMethod
Bank transfer (automatic)    43.656736
Credit card (automatic)      43.269382
Electronic check             25.174630
Mailed check                 21.830025
Name: tenure, dtype: float64
```

In [42]:

```
ax=telustdf.tenure.groupby(telustdf.PaymentMethod).mean().plot(kind="bar",color=["Orange
                                                                title="Average tenure by
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```

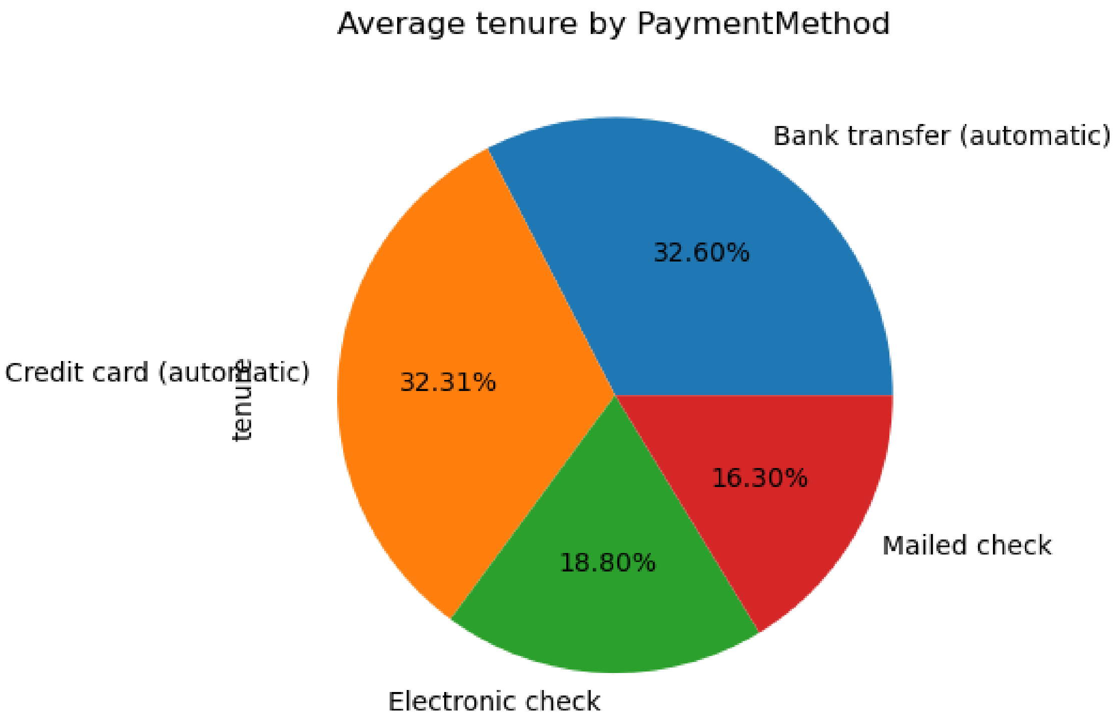


In [43]:

```
telustdf.tenure.groupby(telustdf.PaymentMethod).mean().plot(kind="pie",autopct="%.2f%%",
```

Out[43]:

```
<Axes: title={'center': 'Average tenure by PaymentMethod'}, ylabel='tenur
e'>
```



In [44]:

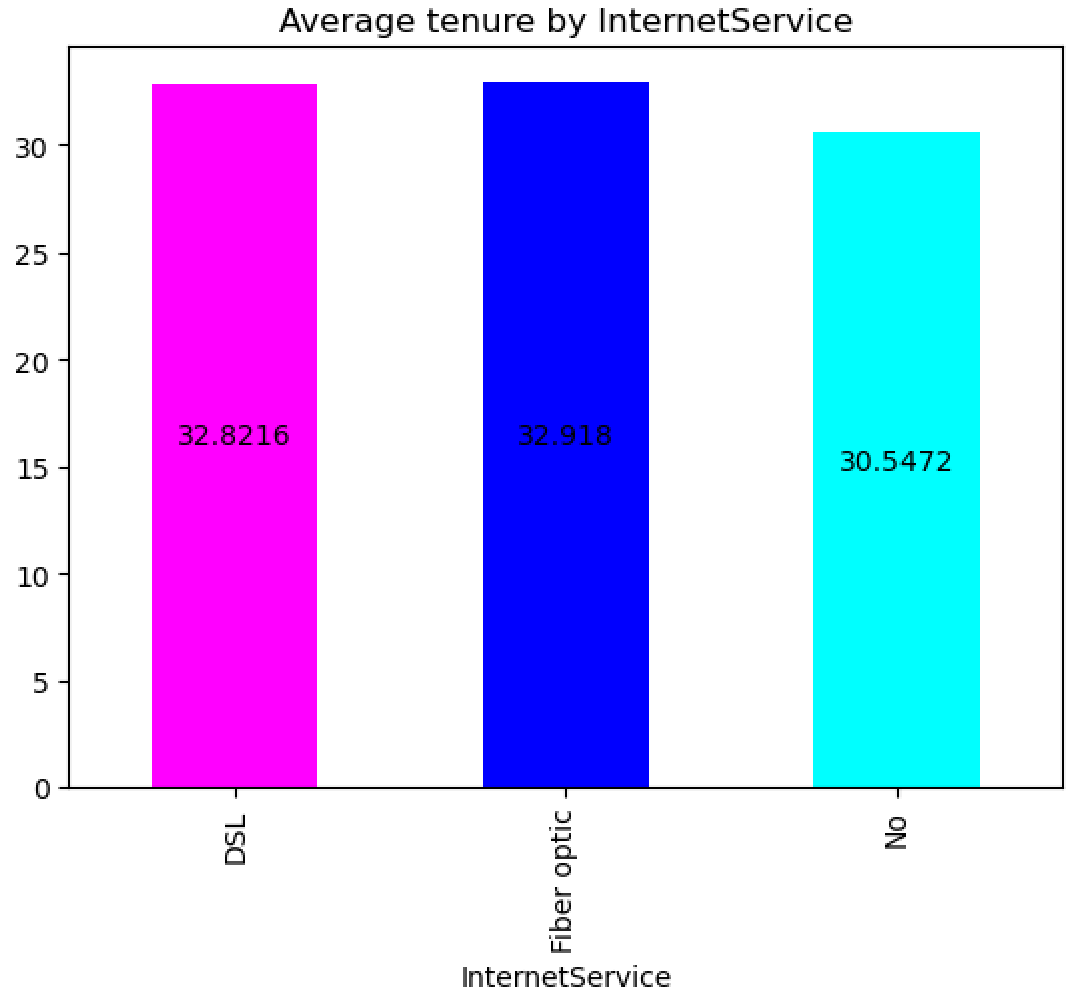
```
# Average tenure by InternetService

telustdf.tenure.groupby(telustdf.InternetService).mean()
```

Out[44]:

```
InternetService
DSL            32.821561
Fiber optic    32.917959
No             30.547182
Name: tenure, dtype: float64
```

In [45]:

```
ax=telustdf.tenure.groupby(telustdf.InternetService).mean().plot(kind="bar",color=["Mage
                                                                   title="Average tenure by
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```

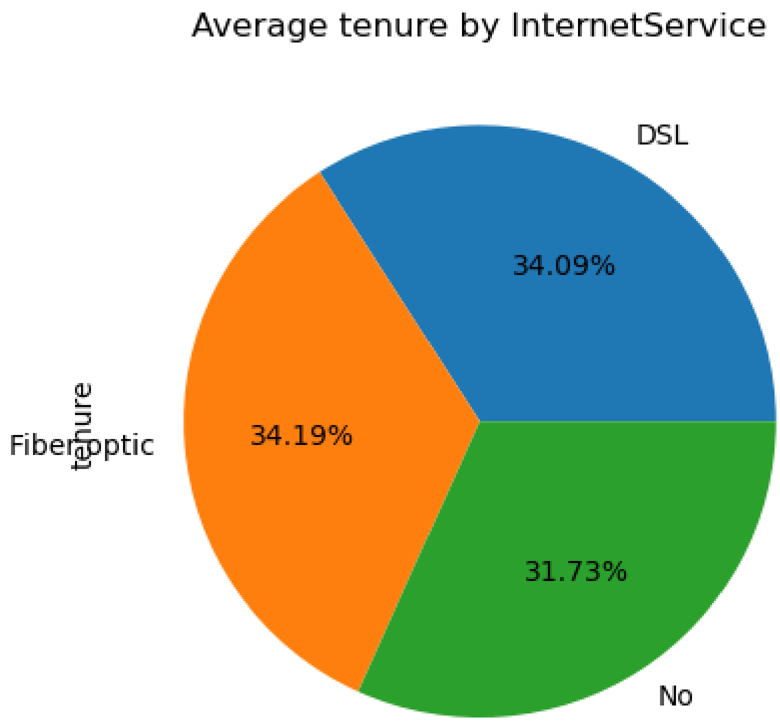


In [46]:

```
telustdf.tenure.groupby(telustdf.InternetService).mean().plot(kind="pie",autopct="%.2f%%
```

Out[46]:

```
<Axes: title={'center': 'Average tenure by InternetService'}, ylabel='tenu
re'>
```



In [47]:

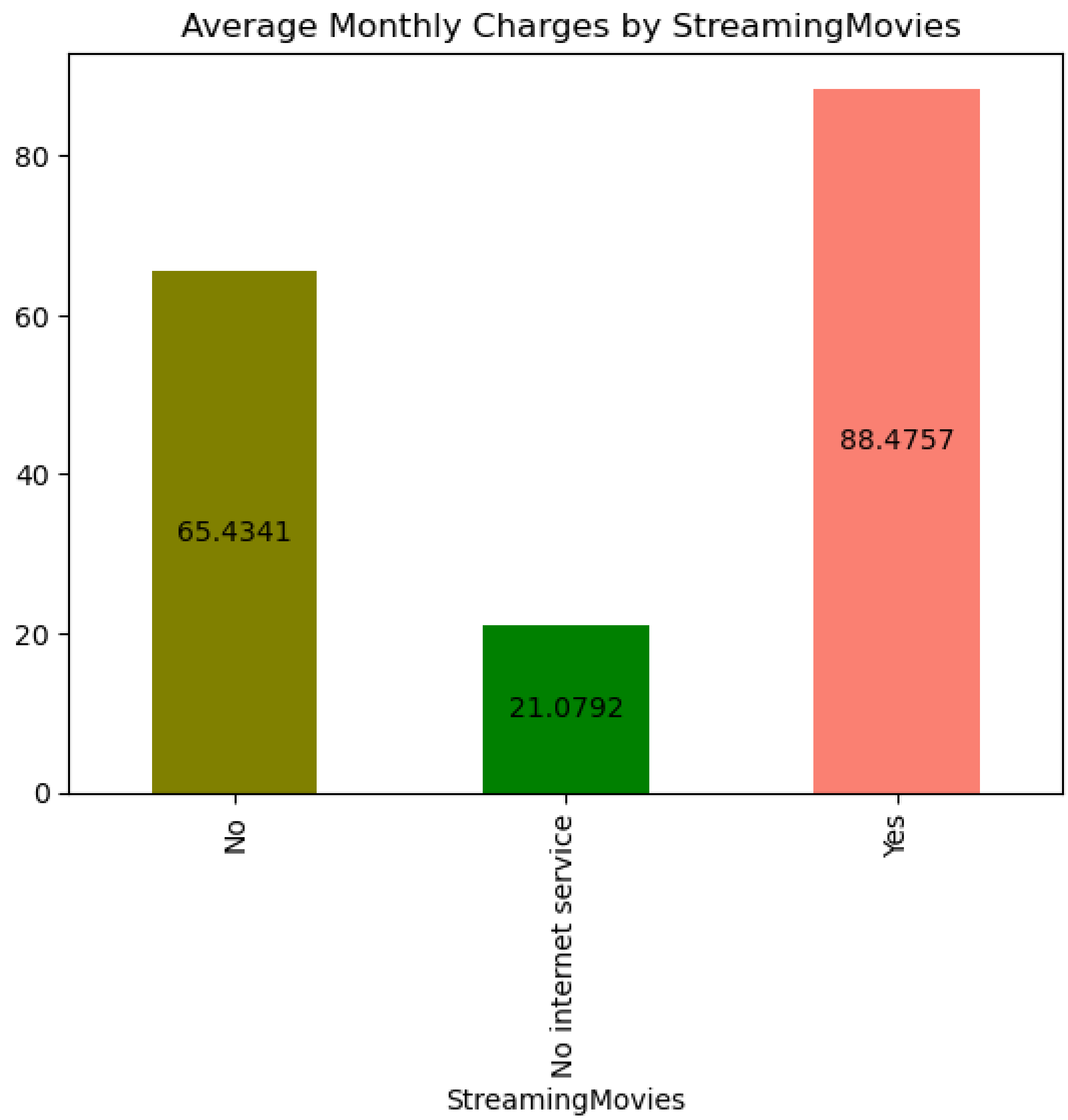
```
# Average Monthly Charges by StreamingMovies

telustdf.MonthlyCharges.groupby(telustdf.StreamingMovies).mean()
```

Out[47]:

```
StreamingMovies
No                     65.434147
No internet service    21.079194
Yes                    88.475714
Name: MonthlyCharges, dtype: float64
```

In [48]:

```
ax=telustdf.MonthlyCharges.groupby(telustdf.StreamingMovies).mean().plot(kind="bar",colo
                                                                title=" Average Monthly
for i in ax.containers:
    ax.bar_label(i,fontsize=10,label_type="center")
```

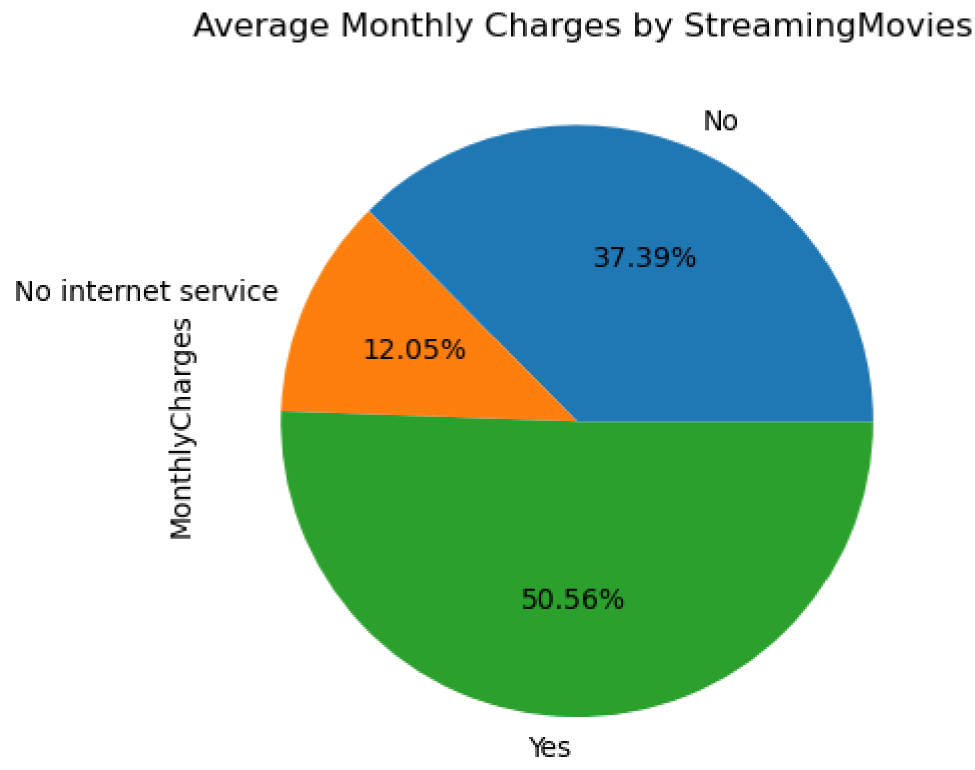


In [49]:

```
telustdf.MonthlyCharges.groupby(telustdf.StreamingMovies).mean().plot(kind="pie",autopct
```

Out[49]:

```
<Axes: title={'center': ' Average Monthly Charges by StreamingMovies'}, yl
abel='MonthlyCharges'>
```



Hypothesis Testing

In [50]:

```
from scipy.stats import ttest_ind
```

In [51]:

```
# Test Null Average MonthlyCharges Churn Yes/No Equal

telustdf.MonthlyCharges.groupby(telustdf.Churn).mean()
```

Out[51]:

```
Churn
No     61.265124
Yes    74.441332
Name: MonthlyCharges, dtype: float64
```

In [52]:

```
churnno=telustdf[telustdf.Churn=='No']
churnyes=telustdf[telustdf.Churn=='Yes']
```

In [53]:

```
ttest_ind(churnyes.MonthlyCharges,churnno.MonthlyCharges,equal_var=False)
```

Out[53]:

```
Ttest_indResult(statistic=18.407526676414673, pvalue=8.59244933154705e-73)
```

In [54]:

```
# Test Null Average tenure of Churn Yes/No Equal
telustdf.tenure.groupby(telustdf.Churn).mean()
```

Out[54]:

```
Churn
No     37.569965
Yes    17.979133
Name: tenure, dtype: float64
```

In [55]:

```
ChurnNo=telustdf[telustdf.Churn=='No']
ChurnYes=telustdf[telustdf.Churn=='Yes']
```

In [56]:

```
ttest_ind(ChurnNo.tenure,ChurnYes.tenure,equal_var=False)
```

Out[56]:

```
Ttest_indResult(statistic=34.823818696312976, pvalue=1.1954945472607151e-2
32)
```

In [57]:

```
# Test Null Average Monthly Charges of different PaymentMethod Equal
telustdf.MonthlyCharges.groupby(telustdf.PaymentMethod).mean()
```

Out[57]:

```
PaymentMethod
Bank transfer (automatic)    67.192649
Credit card (automatic)      66.512385
Electronic check             76.255814
Mailed check                 43.917060
Name: MonthlyCharges, dtype: float64
```

In [58]:

```
from scipy.stats import f_oneway
```

In [59]:

```
# split data

PaymentMethodBT=telustdf[telustdf.PaymentMethod=="Bank transfer (automatic)"]
PaymentMethodCC=telustdf[telustdf.PaymentMethod=="Credit card (automatic)"]
PaymentMethodEC=telustdf[telustdf.PaymentMethod=="Electronic check"]
PaymentMethodMC=telustdf[telustdf.PaymentMethod=="Mailed check"]
```

In [60]:

```
f_oneway(PaymentMethodBT.MonthlyCharges,PaymentMethodCC.MonthlyCharges,PaymentMethodEC.M
         PaymentMethodMC.MonthlyCharges)
```

Out[60]:

```
F_onewayResult(statistic=450.3189918892516, pvalue=1.1802197193575694e-26
7)
```

In [61]:

```
# Test Null Average tenure of different PaymentMethod Equal

telustdf.tenure.groupby(telustdf.PaymentMethod).mean()
```

Out[61]:

```
PaymentMethod
Bank transfer (automatic)    43.656736
Credit card (automatic)      43.269382
Electronic check             25.174630
Mailed check                 21.830025
Name: tenure, dtype: float64
```

In [62]:

```

PaymentMethodBT=telustdf[telustdf.PaymentMethod=="Bank transfer (automatic)"]
PaymentMethodCC=telustdf[telustdf.PaymentMethod=="Credit card (automatic)"]
PaymentMethodEC=telustdf[telustdf.PaymentMethod=="Electronic check"]
PaymentMethodMC=telustdf[telustdf.PaymentMethod=="Mailed check"]
```

In [63]:

```
f_oneway(PaymentMethodBT.tenure,PaymentMethodCC.tenure,PaymentMethodEC.tenure,PaymentMet
```

Out[63]:

```
F_onewayResult(statistic=446.4668862479716, pvalue=1.503848361277172e-265)
```

In [64]:

```
#Test Null No Asssociation between gender & Churn

pd.crosstab(telustdf.gender,telustdf.Churn)
```

Out[64]:

Churn	No	Yes
gender		
Female	2549	939
Male	2625	930

In [65]:

```
from scipy.stats import chi2_contingency
```

In [66]:

```
chi2_contingency(pd.crosstab(telustdf.gender,telustdf.Churn))
```

Out[66]:

```
Chi2ContingencyResult(statistic=0.4840828822091383, pvalue=0.4865787360561
8596, dof=1, expected_freq=array([[2562.38989067,  925.61010933],
       [2611.61010933,  943.38989067]]))
```

In [67]:

```
# Test Null No Association between SeniorCitizen & Churn

pd.crosstab(telustdf.SeniorCitizen,telustdf.Churn)
```

Out[67]:

Churn	No	Yes
SeniorCitizen		
0	4508	1393
1	666	476

In [68]:

```
chi2_contingency(pd.crosstab(telustdf.SeniorCitizen,telustdf.Churn))
```

Out[68]:

```
Chi2ContingencyResult(statistic=159.42630036838742, pvalue=1.5100668050923
78e-36, dof=1, expected_freq=array([[4335.05239245, 1565.94760755],
       [ 838.94760755,  303.05239245]]))
```

LabelEncode data

In [69]:

```
objcols.head()
```

Out[69]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
0	Female	0	Yes	No	No	No phone service	DSL
1	Male	0	No	No	Yes	No	DSL
2	Male	0	No	No	Yes	No	DSL
3	Male	0	No	No	No	No phone service	DSL
4	Female	0	No	No	Yes	No	Fiber optic

In [70]:

```
from sklearn.preprocessing import LabelEncoder
```

In [71]:

```
le=LabelEncoder()
```

In [72]:

```
objcols_labelencoder=objcols.apply(le.fit_transform)
```

In [73]:

```
objcols_labelencoder.head()
```

Out[73]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
0	0	0	1	0	0	1	0
1	1	0	0	0	1	0	0
2	1	0	0	0	1	0	0
3	1	0	0	0	0	1	0
4	0	0	0	0	1	0	1

In [74]:

```
objcols_labelencoder.describe()
```

Out[74]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	In
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	
mean	0.504756	0.162147	0.483033	0.299588	0.903166	0.940508	
std	0.500013	0.368612	0.499748	0.458110	0.295752	0.948554	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	
75%	1.000000	0.000000	1.000000	1.000000	1.000000	2.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	2.000000	

In [75]:

```
objcols_labelencoder.shape
```

Out[75]:

(7043, 17)

Get Dummies

In [76]:

```
objcols_dummies=pd.get_dummies(objcols)
```

In [77]:

```
objcols_dummies.head()
```

Out[77]:

	SeniorCitizen	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	De
0	0	1	0	0	1	1	
1	0	0	1	1	0	1	
2	0	0	1	1	0	1	
3	0	0	1	1	0	1	
4	0	1	0	1	0	1	

5 rows × 44 columns

In [78]:

```
objcols_dummies.describe()
```

Out[78]:

	SeniorCitizen	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.00000
mean	0.162147	0.495244	0.504756	0.516967	0.483033	0.70041
std	0.368612	0.500013	0.500013	0.499748	0.499748	0.45811
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
50%	0.000000	0.000000	1.000000	1.000000	0.000000	1.00000
75%	0.000000	1.000000	1.000000	1.000000	1.000000	1.00000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

8 rows × 44 columns

In [79]:

```
objcols_dummies.shape
```

Out[79]:

```
(7043, 44)
```

Scaleing

In [80]:

```
numcols.head()
```

Out[80]:

	tenure	MonthlyCharges	TotalCharges
0	1	29.85	29.85
1	34	56.95	1889.50
2	2	53.85	108.15
3	45	42.30	1840.75
4	2	70.70	151.65

In [81]:

```
from sklearn.preprocessing import StandardScaler
```

In [82]:

```
ss=StandardScaler()
```

In [83]:

```
numcols_std_scale=ss.fit_transform(numcols)
```

In [84]:

```
numcols_std_scale=pd.DataFrame(numcols_std_scale,columns=numcols.columns)
```

In [85]:

```
numcols_std_scale.head()
```

Out[85]:

	tenure	MonthlyCharges	TotalCharges
0	-1.277445	-1.160323	-0.994242
1	0.066327	-0.259629	-0.173244
2	-1.236724	-0.362660	-0.959674
3	0.514251	-0.746535	-0.194766
4	-1.236724	0.197365	-0.940470

In [86]:

```
combindf=pd.concat([numcols_std_scale,objcols_labelencoder],axis=1)
```

In [87]:

```
combindf.head()
```

Out[87]:

	tenure	MonthlyCharges	TotalCharges	gender	SeniorCitizen	Partner	Dependents	Ph
0	-1.277445	-1.160323	-0.994242	0	0	1	0	
1	0.066327	-0.259629	-0.173244	1	0	0	0	
2	-1.236724	-0.362660	-0.959674	1	0	0	0	
3	0.514251	-0.746535	-0.194766	1	0	0	0	
4	-1.236724	0.197365	-0.940470	0	0	0	0	

In [88]:

```
# Spliting Data into Dependent and Independent variables

X=combindf.drop("Churn",axis=1)
y=combindf.Churn
```

In [89]:

```
from sklearn.linear_model import LogisticRegression
```

In [90]:

```
lg=LogisticRegression()
```

In [91]:

```
lgmodel=lg.fit(X,y)
```

In [92]:

```
lgmodel.score(X,y)
```

Out[92]:

```
0.8044867244072128
```

In [93]:

```
lgpred=lgmodel.predict(X)
```

In [94]:

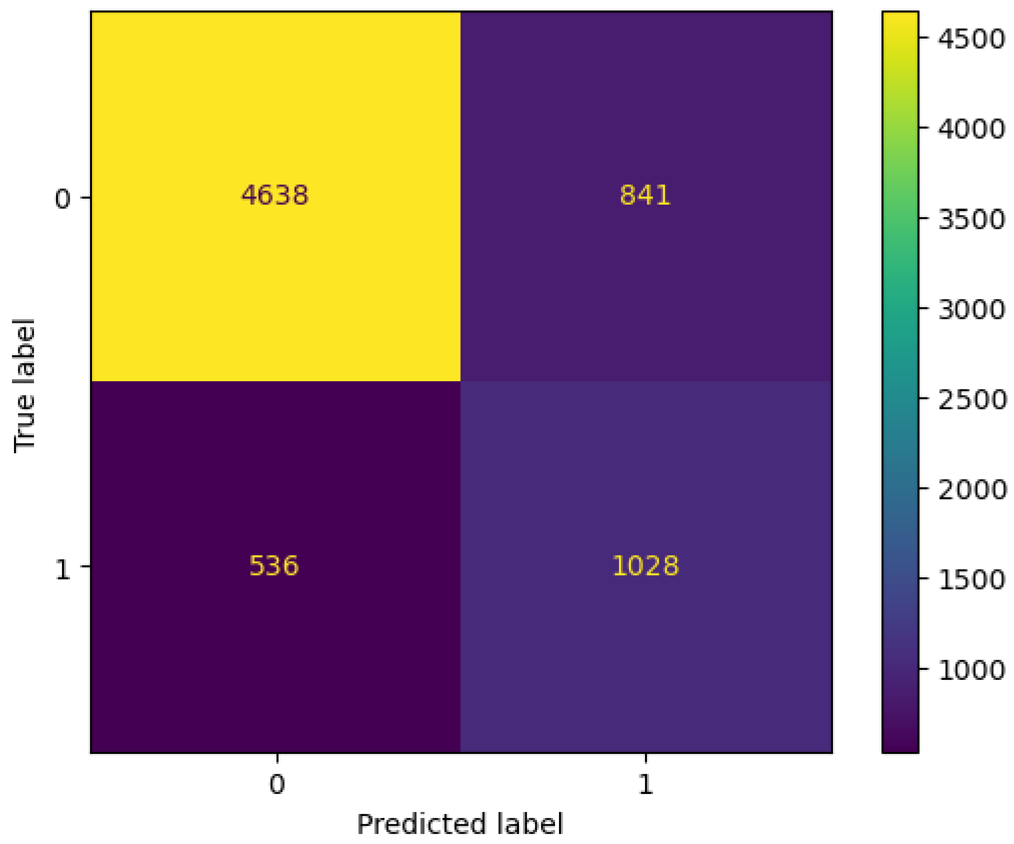
```
from sklearn.metrics import ConfusionMatrixDisplay,RocCurveDisplay,classification_report
```

In [95]:

```
ConfusionMatrixDisplay.from_predictions(lgpred,y)
```

Out[95]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f4e99
bbaf0>
```

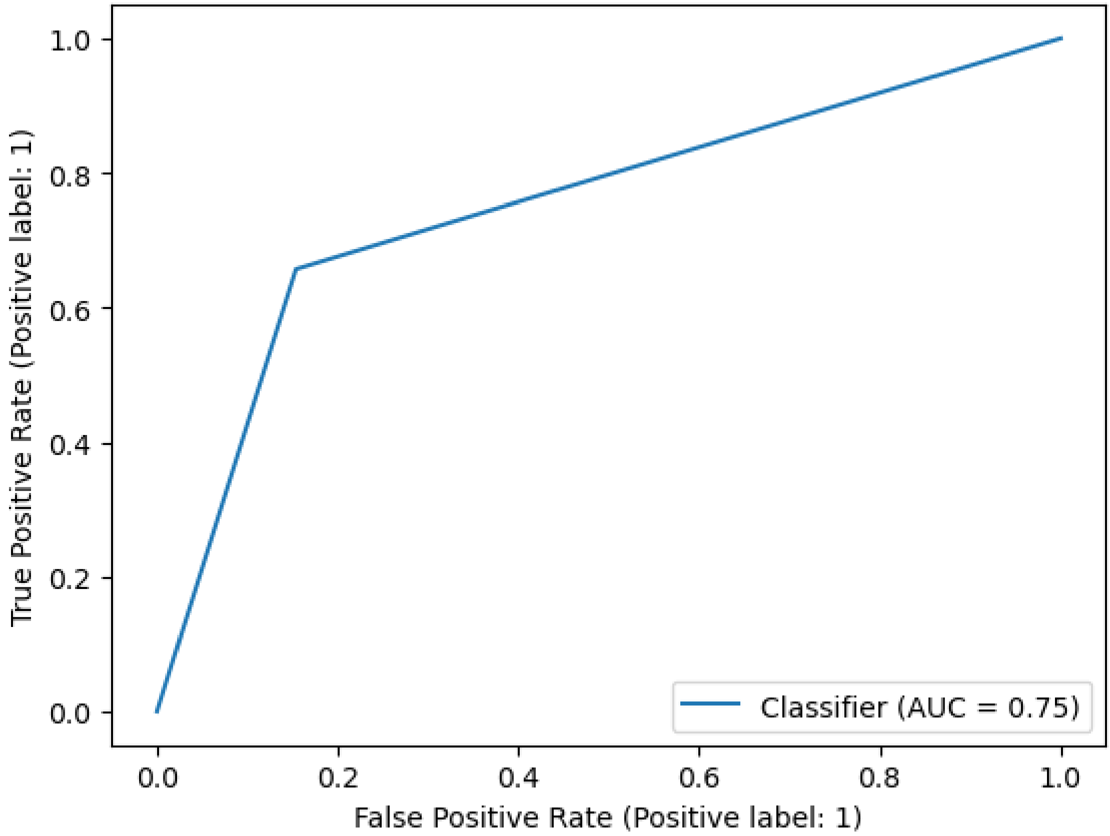


In [96]:

```
RocCurveDisplay.from_predictions(lgpred,y)
```

Out[96]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f4e9a58f10>
```



In [97]:

```
print(classification_report(lgpred,y))
```

```
              precision    recall  f1-score   support

           0       0.90      0.85      0.87      5479
           1       0.55      0.66      0.60      1564

    accuracy                           0.80      7043
   macro avg       0.72      0.75      0.73      7043
weighted avg       0.82      0.80      0.81      7043
```

In [98]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [99]:

```
tree=DecisionTreeClassifier(max_depth=8)
```

In [100]:

```
treemodel=tree.fit(X,y)
```

In [101]:

```
treemodel.score(X,y)
```

Out[101]:

0.8313218798807327

In [102]:

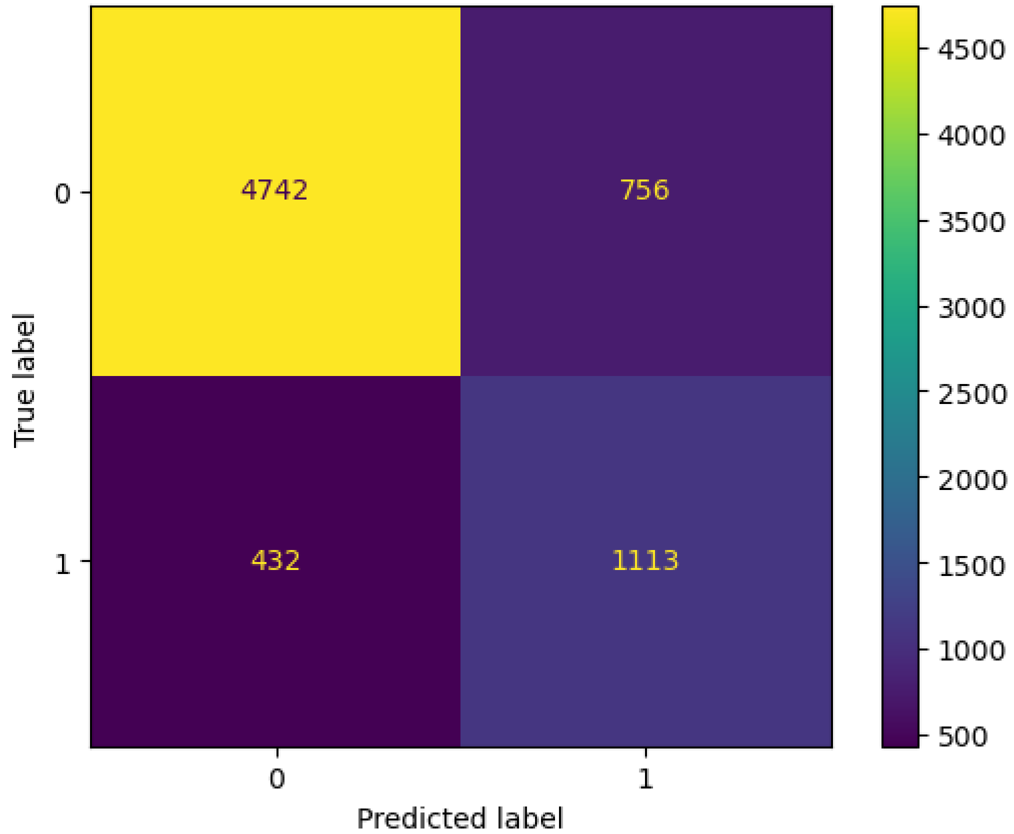
```
treepred=treemodel.predict(X)
```

In [103]:

```
ConfusionMatrixDisplay.from_predictions(treepred,y)
```

Out[103]:

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f4e99e4040>

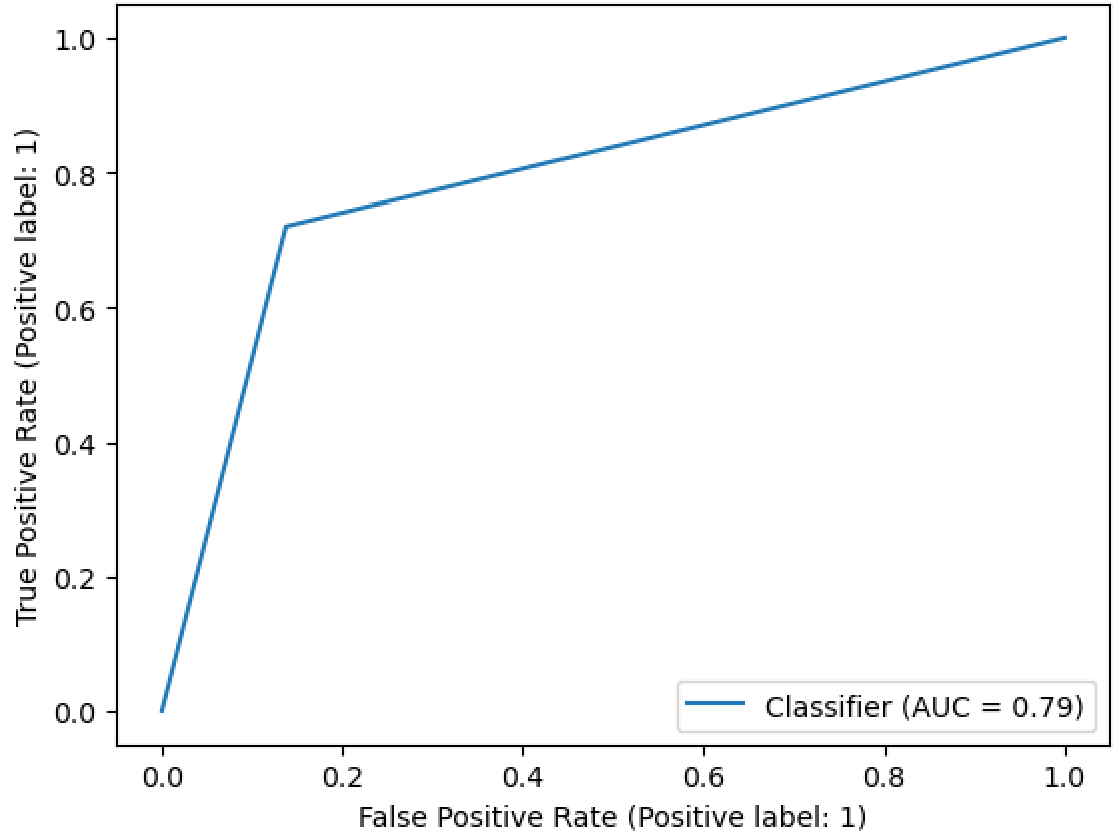


In [104]:

```
RocCurveDisplay.from_predictions(treepred,y)
```

Out[104]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f4e9d16e80>
```



In [105]:

```
print(classification_report(treepred,y))
```

```
              precision    recall  f1-score   support

           0       0.92      0.86      0.89      5498
           1       0.60      0.72      0.65      1545

    accuracy                           0.83      7043
   macro avg       0.76      0.79      0.77      7043
weighted avg       0.85      0.83      0.84      7043
```

In [106]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [107]:

```
rfc=RandomForestClassifier(n_estimators=2000,max_depth=12)
```

In [108]:

```
rfcmodel=rfc.fit(X,y)
```

In [109]:

```
rfcmodel.score(X,y)
```

Out[109]:

```
0.941644185716314
```

In [110]:

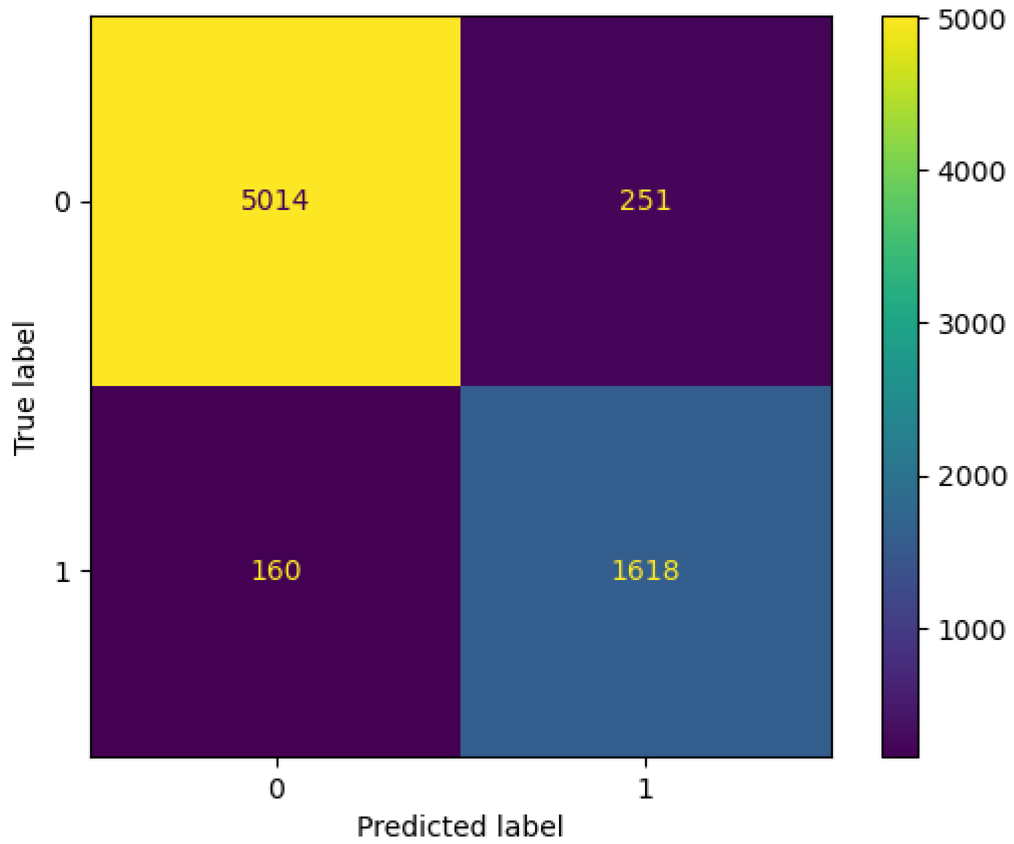
```
rfcpred=rfcmodel.predict(X)
```

In [111]:

```
ConfusionMatrixDisplay.from_predictions(rfcpred,y)
```

Out[111]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f4e0d
d2070>
```

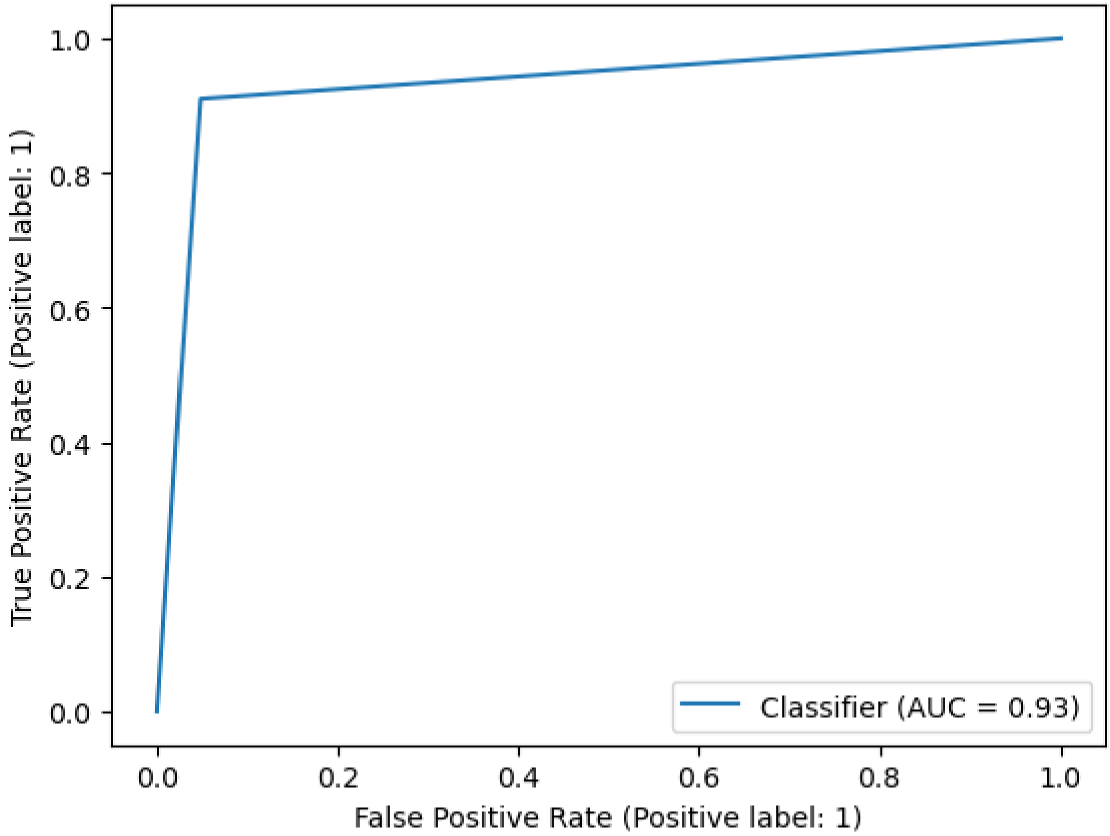


In [112]:

```
RocCurveDisplay.from_predictions(rfcpred,y)
```

Out[112]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f4e0de40a0>
```



In [113]:

```
print(classification_report(rfcpred,y))
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96      5265
           1       0.87      0.91      0.89      1778

    accuracy                           0.94      7043
   macro avg       0.92      0.93      0.92      7043
weighted avg       0.94      0.94      0.94      7043
```

In [114]:

```
from sklearn.ensemble import GradientBoostingClassifier
```

In [115]:

```
gbc=GradientBoostingClassifier()
```

In [116]:

```
gbcmodel=gbc.fit(X,y)
```

In [117]:

```
gbcmodel.score(X,y)
```

Out[117]:

```
0.8253585119977283
```

In [118]:

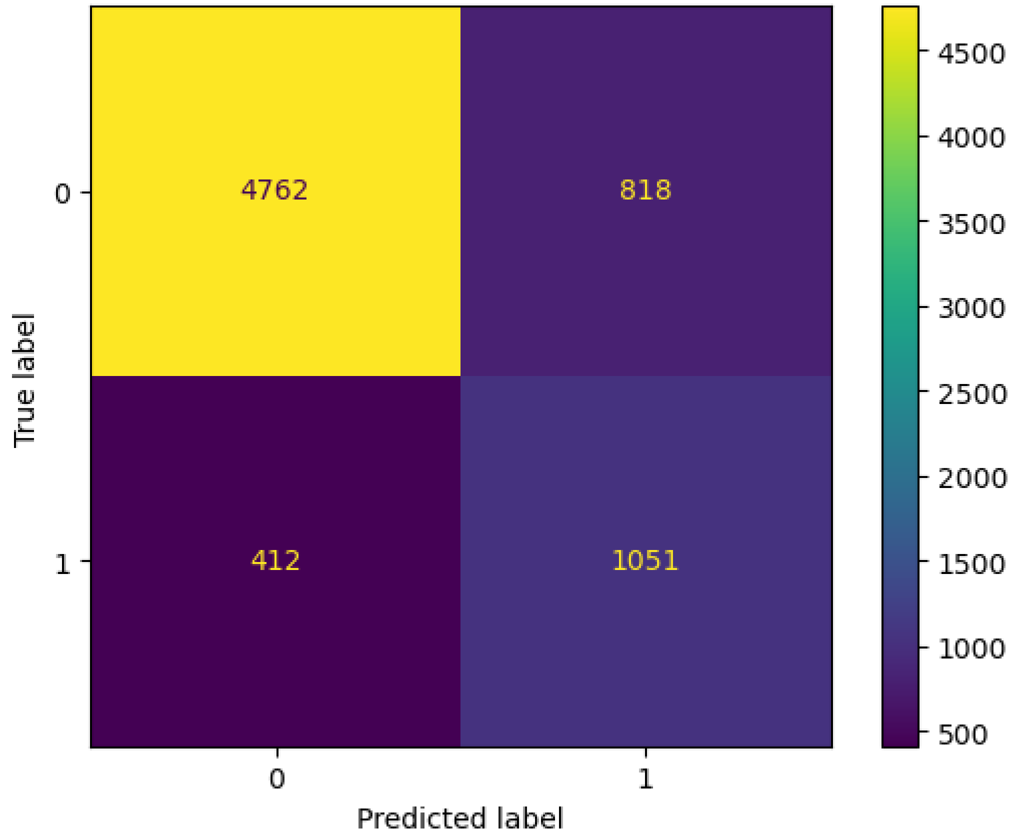
```
gbcpred=gbcmodel.predict(X)
```

In [119]:

```
ConfusionMatrixDisplay.from_predictions(gbcpred,y)
```

Out[119]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f4e9e
55790>
```

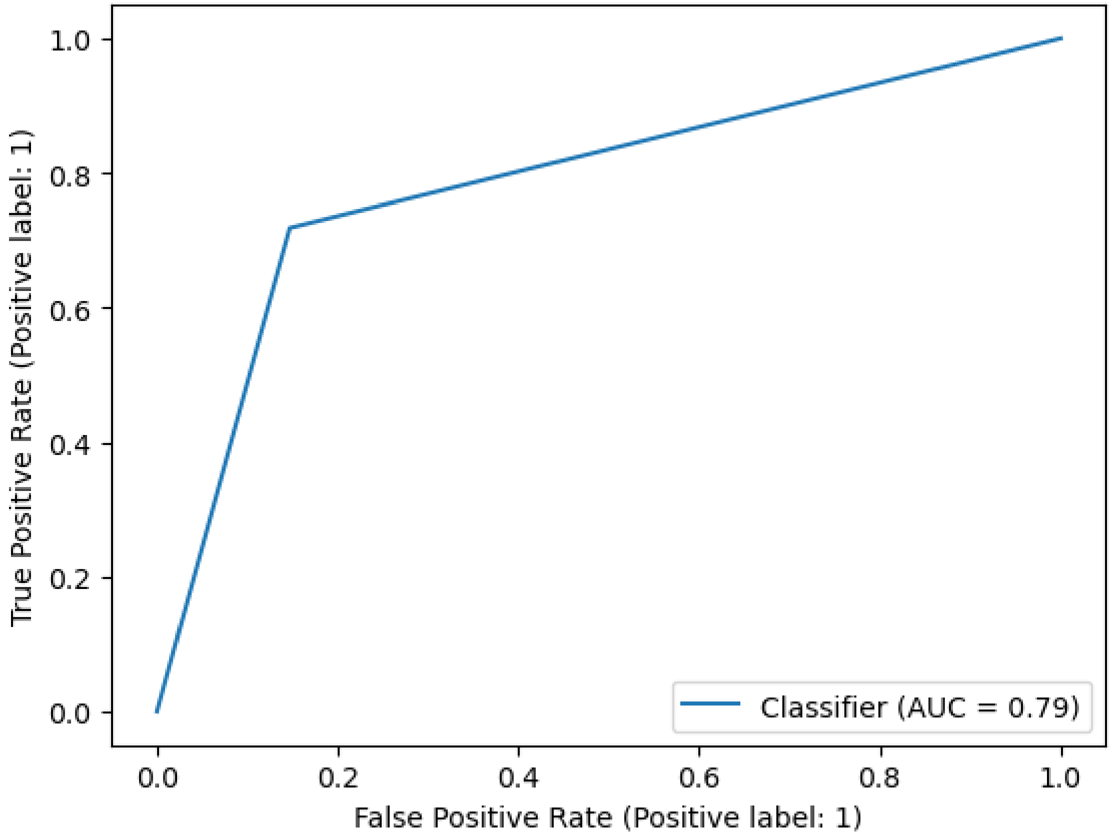


In [120]:

```
RocCurveDisplay.from_predictions(gbcpred,y)
```

Out[120]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f4f912b580>
```



In [121]:

```
print(classification_report(gbcpred,y))
```

```
              precision    recall  f1-score   support

           0       0.92      0.85      0.89      5580
           1       0.56      0.72      0.63      1463

    accuracy                           0.83      7043
   macro avg       0.74      0.79      0.76      7043
weighted avg       0.85      0.83      0.83      7043
```

In [122]:

```
# GausssianNB

from sklearn.naive_bayes import GaussianNB
```

In [123]:

```
Gnb=GaussianNB()
```

In [124]:

```
gnbmodel=Gnb.fit(X,y)
```

In [125]:

```
gnbmodel.score(X,y)
```

Out[125]:

```
0.7526622178049127
```

In [126]:

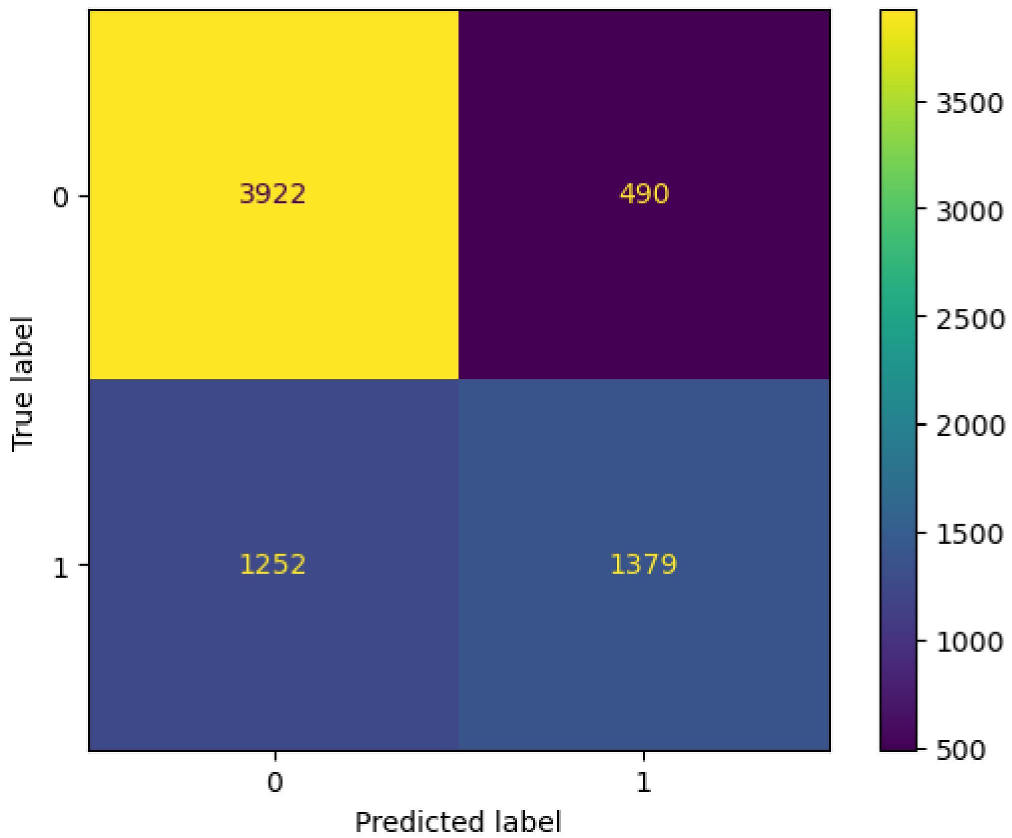
```
gnbpred=gnbmodel.predict(X)
```

In [127]:

```
ConfusionMatrixDisplay.from_predictions(gnbpred,y)
```

Out[127]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f4e9e
774c0>
```

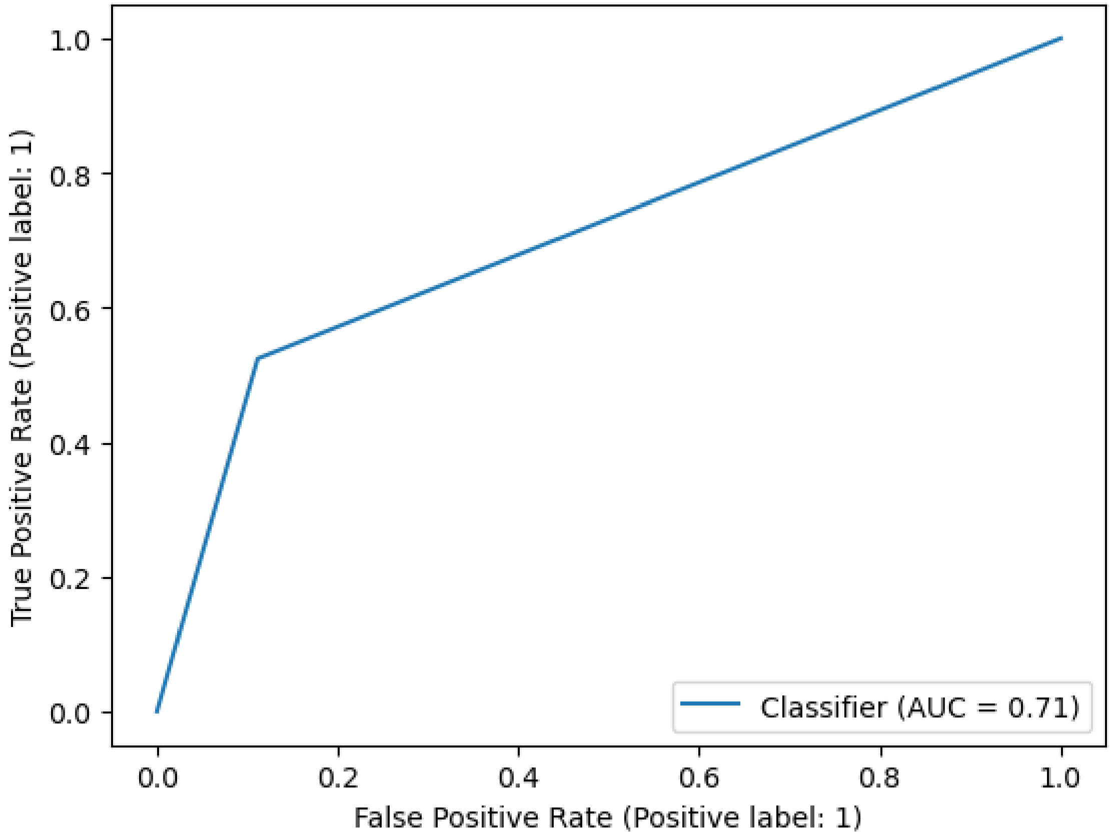


In [128]:

```
RocCurveDisplay.from_predictions(gnbpred,y)
```

Out[128]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f4f921b280>
```



In [129]:

```
print(classification_report(gnbpred,y))
```

```
              precision    recall  f1-score   support

           0       0.76      0.89      0.82      4412
           1       0.74      0.52      0.61      2631

    accuracy                           0.75      7043
   macro avg       0.75      0.71      0.72      7043
weighted avg       0.75      0.75      0.74      7043
```

In [130]:

```
# KNeighbors Classifier
from sklearn.neighbors import KNeighborsClassifier
```

In [131]:

```
knc=KNeighborsClassifier()
```

In [132]:

```
kncmodel=knc.fit(X,y)
```

In [133]:

```
kncmodel.score(X,y)
```

Out[133]:

```
0.8385631123100952
```

In [134]:

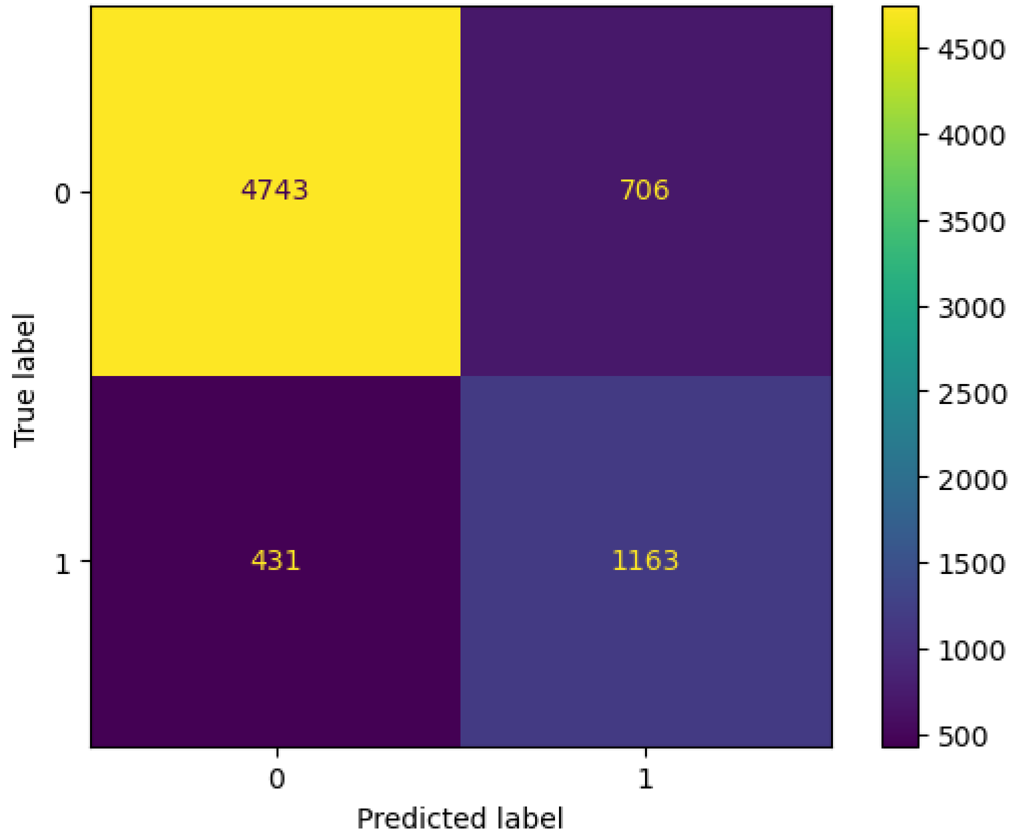
```
kncpred=kncmodel.predict(X)
```

In [135]:

```
ConfusionMatrixDisplay.from_predictions(kncpred,y)
```

Out[135]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f4f92
83e80>
```

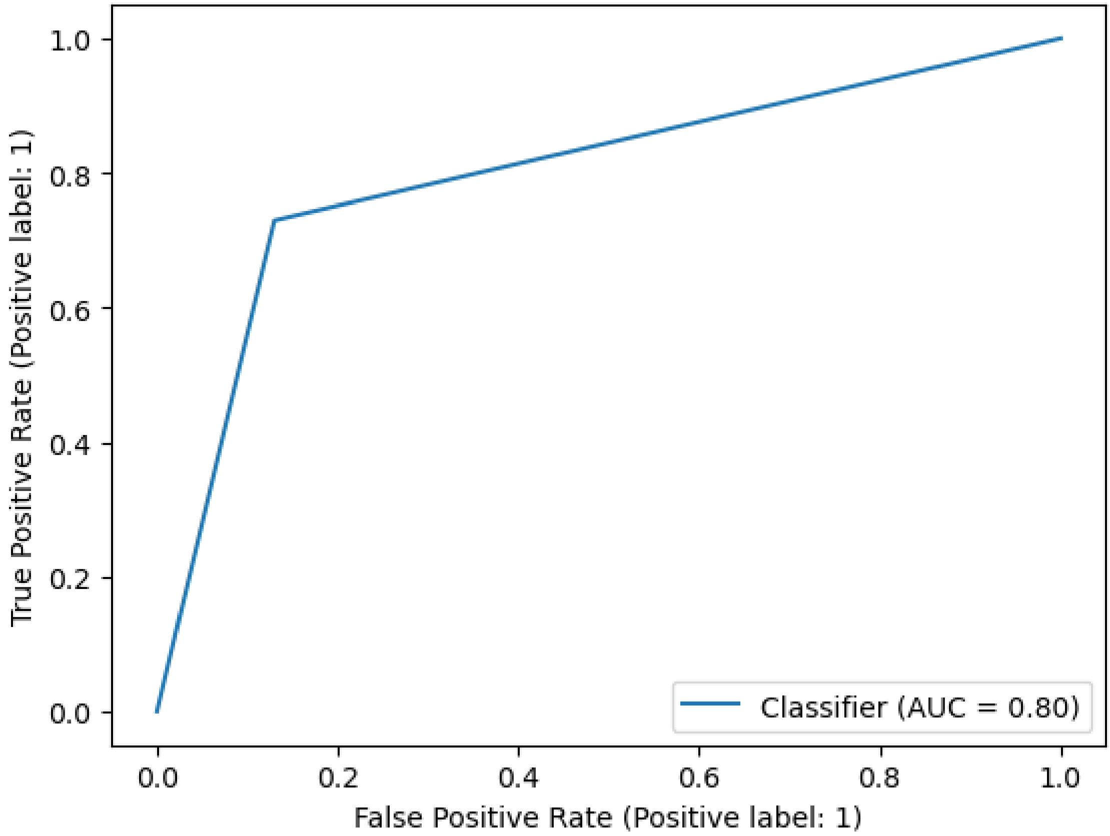


In [136]:

```
RocCurveDisplay.from_predictions(kncpred,y)
```

Out[136]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f4fa470760>
```



In [137]:

```
print(classification_report(kncpred,y))
```

```
              precision    recall  f1-score   support

           0       0.92      0.87      0.89      5449
           1       0.62      0.73      0.67      1594

    accuracy                           0.84      7043
   macro avg       0.77      0.80      0.78      7043
weighted avg       0.85      0.84      0.84      7043
```

In [138]:

```
# Support Vector Classifier

from sklearn.svm import SVC
```

In [139]:
```
svc=SVC()
```

In [140]:
```
svcmodel=svc.fit(X,y)
```

In [141]:
```
svcmodel.score(X,y)
```

Out[141]:

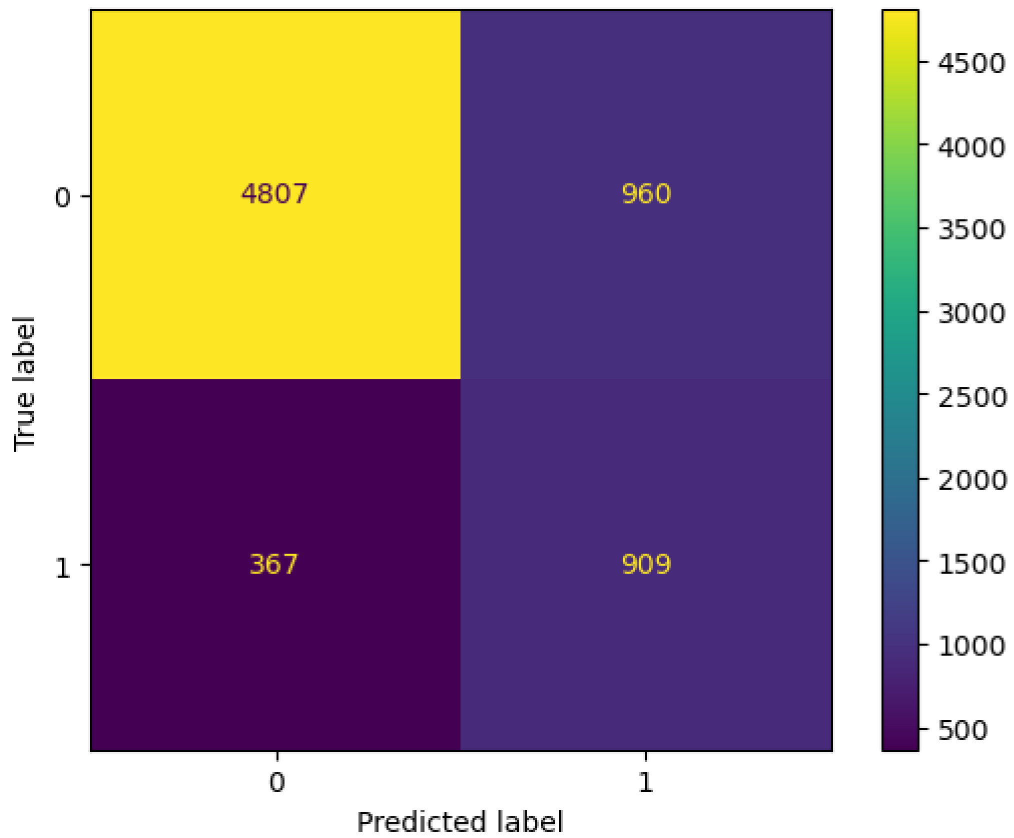
```
0.81158597188698
```

In [142]:
```
svcpred=svcmodel.predict(X)
```

In [143]:
```
ConfusionMatrixDisplay.from_predictions(svcpred,y)
```

Out[143]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f4fa6
270d0>
```

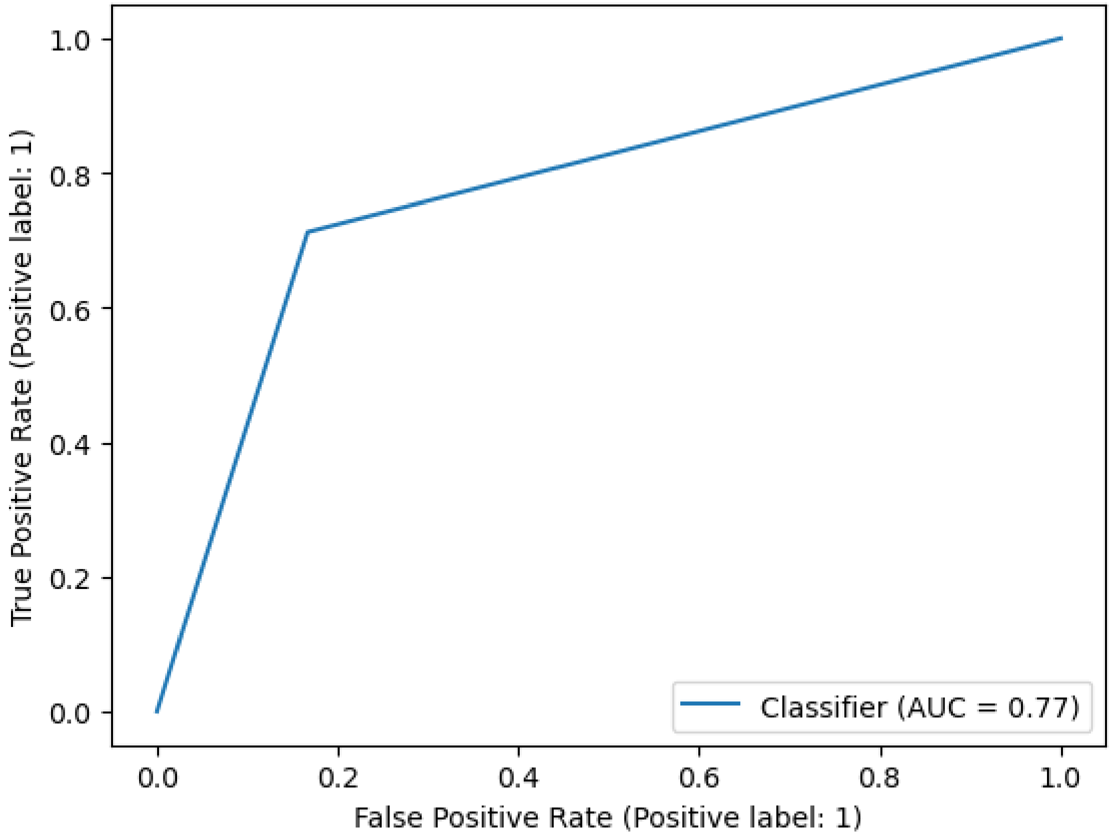


In [144]:

```
RocCurveDisplay.from_predictions(svcpred,y)
```

Out[144]:

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1f4fcb17280>
```



In [145]:

```
print(classification_report(svcpred,y))
```

```
              precision    recall  f1-score   support

           0       0.93      0.83      0.88      5767
           1       0.49      0.71      0.58      1276

    accuracy                           0.81      7043
   macro avg       0.71      0.77      0.73      7043
weighted avg       0.85      0.81      0.82      7043
```