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

In [2]:
```
telco_churn_df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

In [3]:
```
telco_churn_df.head()
```

Out[3]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | Inter |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | |
| **1** | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | |
| **2** | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | |
| **3** | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | |
| **4** | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | |

5 rows × 21 columns

In [4]:
```
telco_churn_df.tail()
```

Out[4]:

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

5 rows × 21 columns

In [5]:
```
telco_churn_df.shape
```

Out[5]:
```
(7043, 21)
```

In [6]:
```
telco_churn_df.describe()
```

Out[6]:

| | SeniorCitizen | tenure | MonthlyCharges |
|---|---|---|---|
| **count** | 7043.000000 | 7043.000000 | 7043.000000 |
| **mean** | 0.162147 | 32.371149 | 64.761692 |
| **std** | 0.368612 | 24.559481 | 30.090047 |
| **min** | 0.000000 | 0.000000 | 18.250000 |
| **25%** | 0.000000 | 9.000000 | 35.500000 |
| **50%** | 0.000000 | 29.000000 | 70.350000 |
| **75%** | 0.000000 | 55.000000 | 89.850000 |
| **max** | 1.000000 | 72.000000 | 118.750000 |

In [7]:
```
telco_churn_df.dtypes
```

Out[7]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

In [8]:
```
telco_churn_df.columns
```

Out[8]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
```

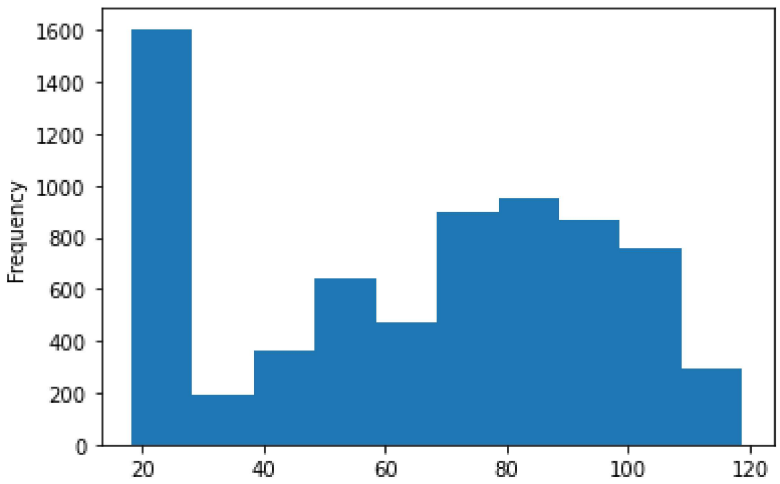
```
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [9]:
```
telco_churn_df.isnull().sum()
```

Out[9]:
```
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
```

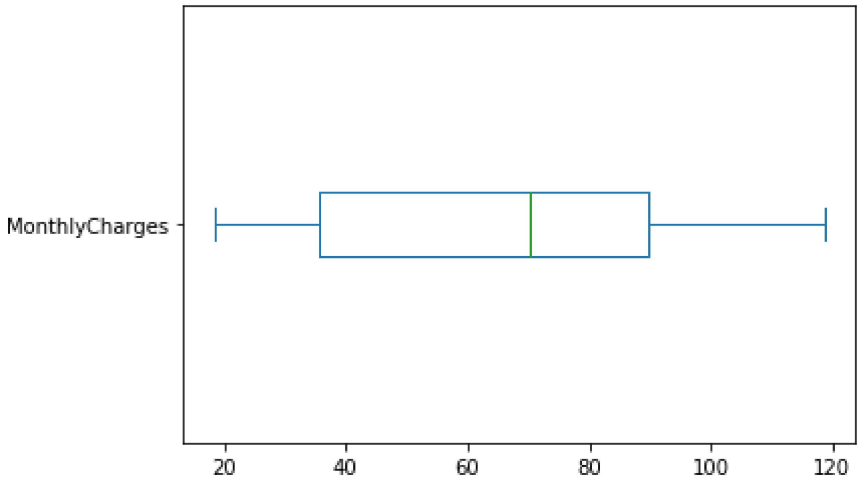
In [10]:
```
telco_churn_df.MonthlyCharges.plot(kind='hist')
```

Out[10]:
```
<AxesSubplot:ylabel='Frequency'>
```



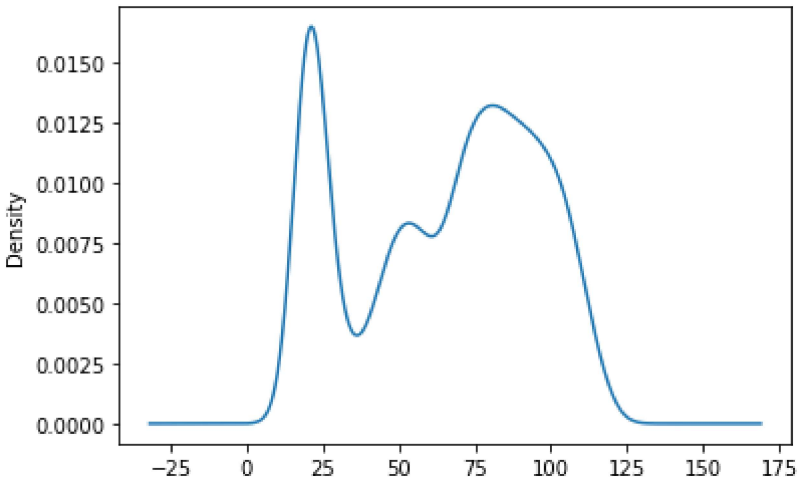

In [11]:
```
telco_churn_df.MonthlyCharges.plot(kind='box',vert=False)
```

Out[11]:
```
<AxesSubplot:>
```



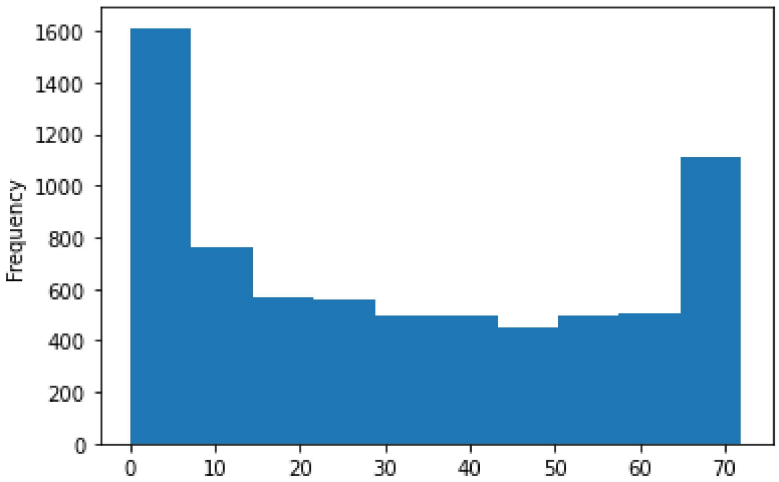

In [12]:
```
telco_churn_df.MonthlyCharges.plot(kind='density')
```

Out[12]:
```
<AxesSubplot:ylabel='Density'>
```



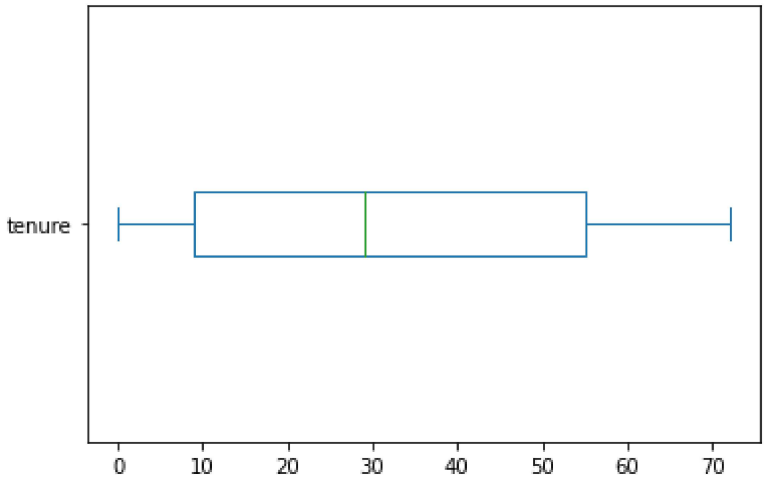

In [13]:
```
telco_churn_df.tenure.plot(kind='hist')
```

Out[13]:
```
<AxesSubplot:ylabel='Frequency'>
```



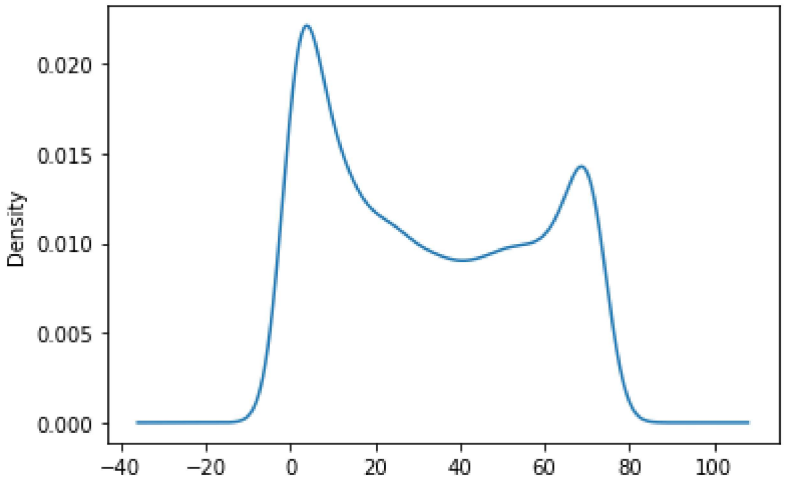

In [14]:
```
telco_churn_df.tenure.plot(kind='box',vert=False)
```

Out[14]:
```
<AxesSubplot:>
```




In [15]:
```
telco_churn_df.tenure.plot(kind='density')
```

Out[15]:
```
<AxesSubplot:ylabel='Density'>
```




In [16]:
```
print(telco_churn_df.Churn.value_counts())
print(telco_churn_df.gender.value_counts())
print(telco_churn_df.SeniorCitizen.value_counts())
print(telco_churn_df.PaymentMethod.value_counts())
```

```
No     5174
Yes    1869
Name: Churn, dtype: int64
Male      3555
Female    3488
Name: gender, dtype: int64
0    5901
1    1142
Name: SeniorCitizen, dtype: int64
Electronic check             2365
Mailed check                 1612
Bank transfer (automatic)    1544
Credit card (automatic)      1522
Name: PaymentMethod, dtype: int64
```

In [17]:
```
pd.crosstab(telco_churn_df.Churn,telco_churn_df.gender)
```

Out[17]:

| Churn | gender Female | Male |
|---|---|---|
| No | 2549 | 2625 |
| Yes | 939 | 930 |

In [18]:
```
pd.crosstab(telco_churn_df.Churn,telco_churn_df.InternetService)
```

Out[18]:

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

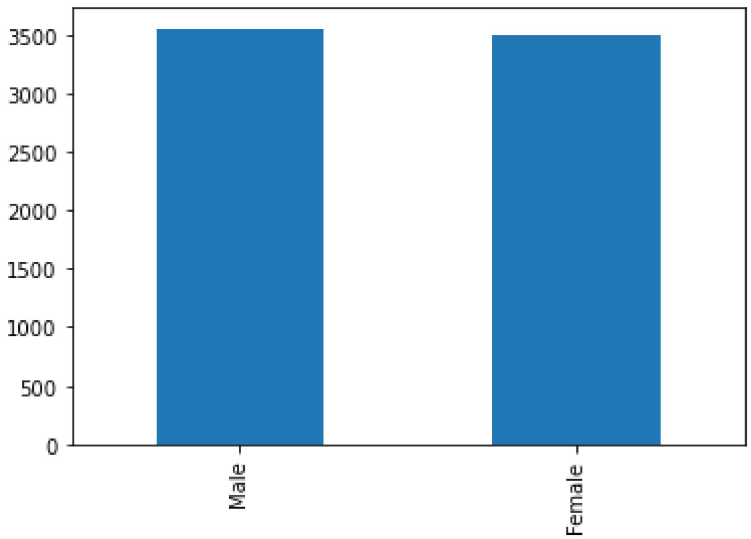
In [19]:
```
pd.crosstab(telco_churn_df.gender,telco_churn_df.PaymentMethod)
```

Out[19]:

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

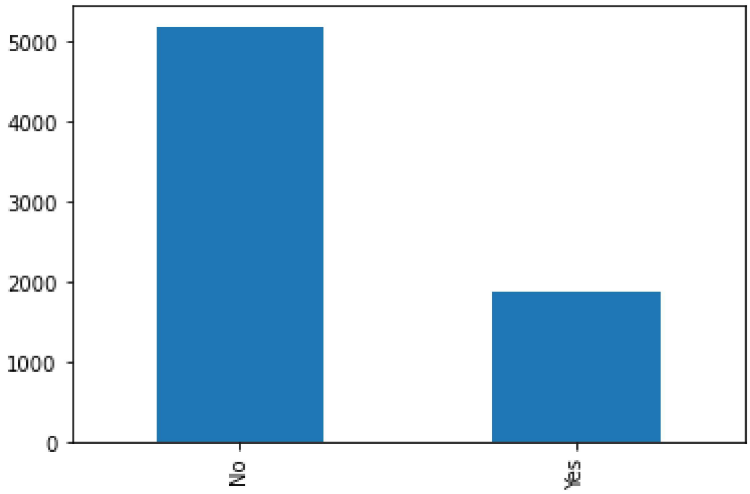
In [20]:
```
telco_churn_df.gender.value_counts().plot(kind='bar')
```

Out[20]:
```
<AxesSubplot:>
```



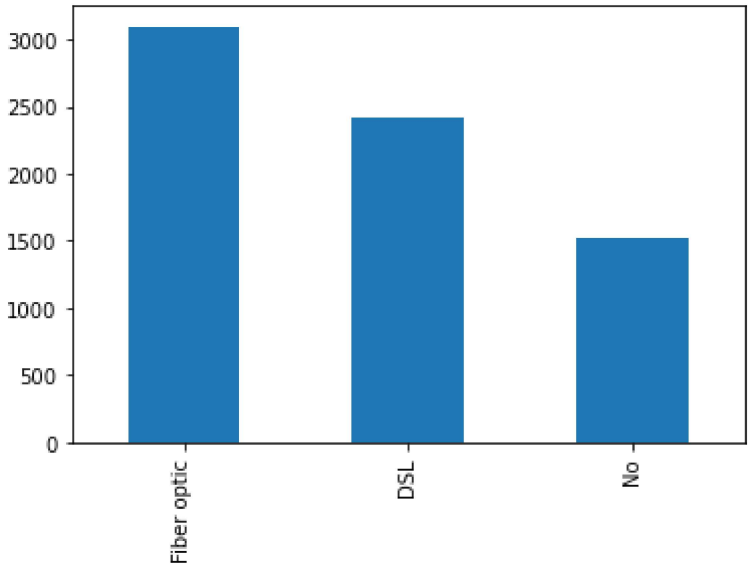

In [21]:
```
telco_churn_df.Churn.value_counts().plot(kind='bar')
```

Out[21]:
```
<AxesSubplot:>
```



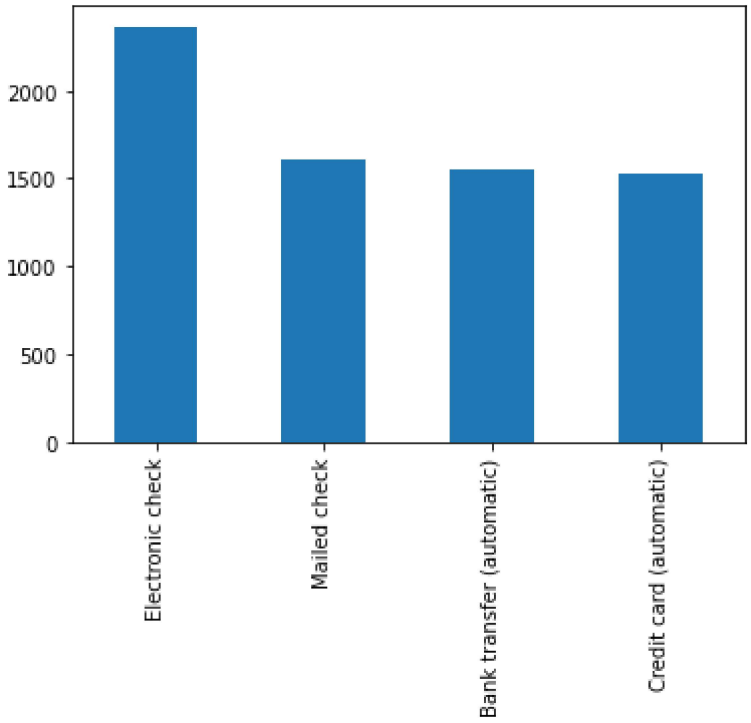

In [22]:
```
telco_churn_df.InternetService.value_counts().plot(kind='bar')
```

Out[22]: <AxesSubplot:>




In [23]:
```
telco_churn_df.PaymentMethod.value_counts().plot(kind='bar')
```

Out[23]: <AxesSubplot:>

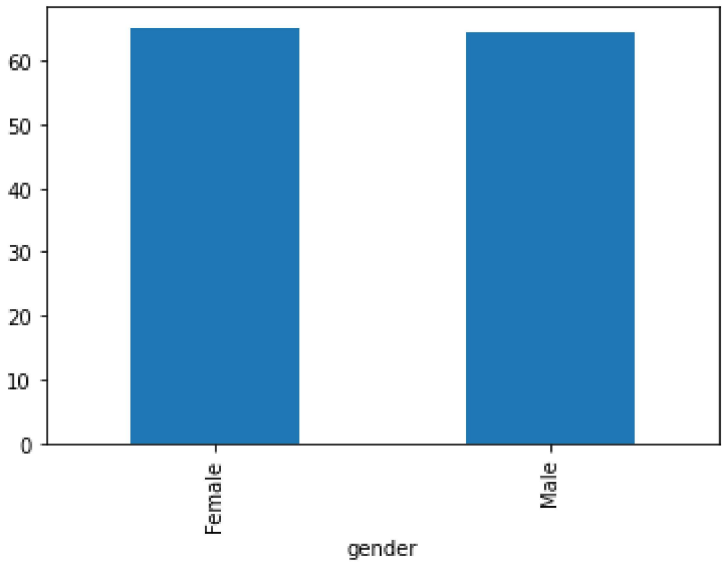



In [24]:
```
telco_churn_df.MonthlyCharges.groupby(telco_churn_df.gender).mean()
```

Out[24]:
```
gender
Female    65.204243
Male      64.327482
Name: MonthlyCharges, dtype: float64
```

In [25]:
```
telco_churn_df.MonthlyCharges.groupby(telco_churn_df.gender).mean().plot(kind='bar')
```

Out[25]:
```
<AxesSubplot:xlabel='gender'>
```



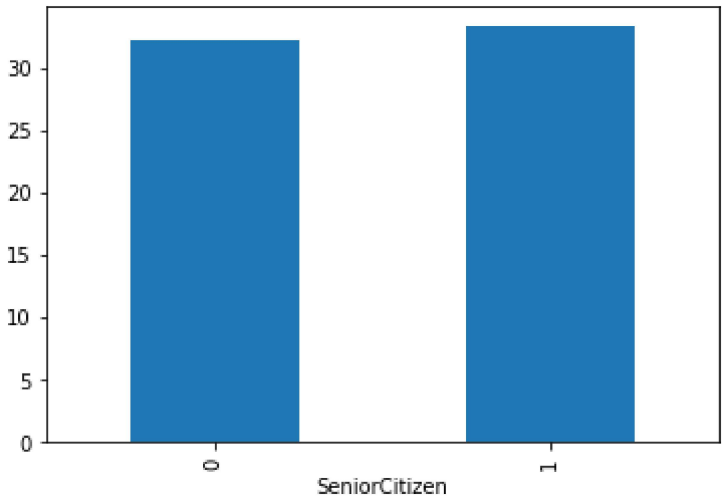

In [26]:
```
telco_churn_df.tenure.groupby(telco_churn_df.SeniorCitizen).mean()
```

Out[26]:
```
SeniorCitizen
0    32.192171
```

```
1    33.295972
Name: tenure, dtype: float64
```

In [27]:
```
telco_churn_df.tenure.groupby(telco_churn_df.SeniorCitizen).mean().plot(kind='bar')
```

Out[27]:
```
<AxesSubplot:xlabel='SeniorCitizen'>
```

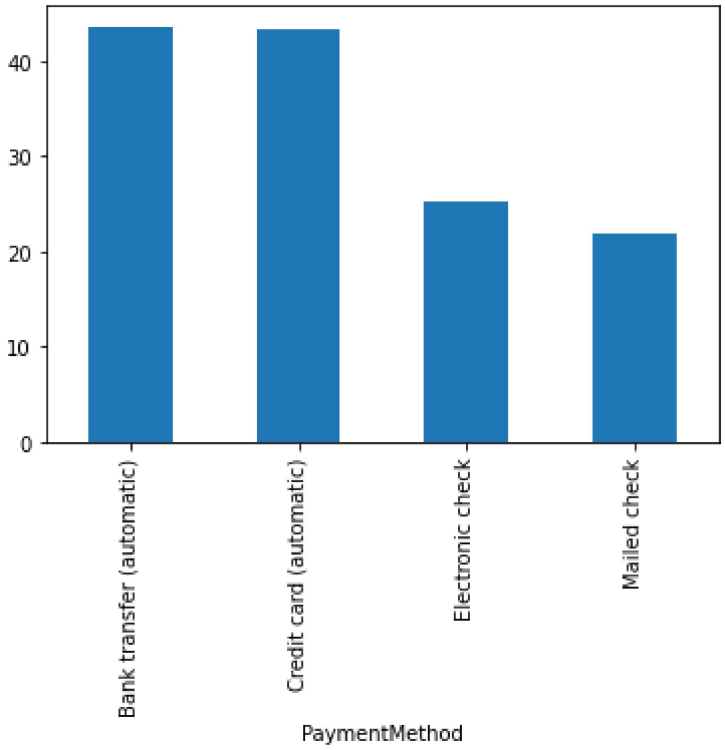



In [28]:
```
telco_churn_df.tenure.groupby(telco_churn_df.PaymentMethod).mean()
```

Out[28]:
```
PaymentMethod
Bank transfer (automatic)    43.656736
Credit card (automatic)      43.269382
Electronic check             25.174630
Mailed check                 21.830025
Name: tenure, dtype: float64
```

In [29]:
```
telco_churn_df.tenure.groupby(telco_churn_df.PaymentMethod).mean().plot(kind='bar')
```

Out[29]:
```
<AxesSubplot:xlabel='PaymentMethod'>
```

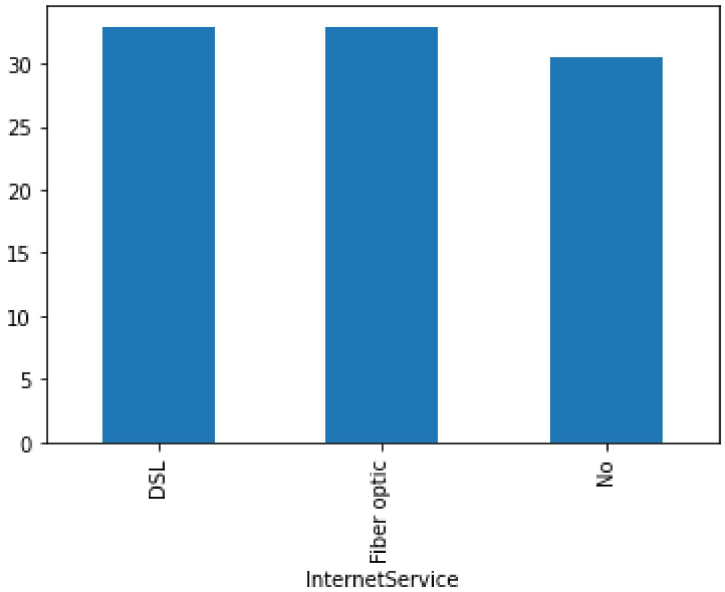



In [30]:
```
telco_churn_df.tenure.groupby(telco_churn_df.InternetService).mean()
```

Out[30]:
```
InternetService
DSL             32.821561
Fiber optic     32.917959
No              30.547182
Name: tenure, dtype: float64
```

In [31]:
```
telco_churn_df.tenure.groupby(telco_churn_df.InternetService).mean().plot(kind='bar')
```

Out[31]:
```
<AxesSubplot:xlabel='InternetService'>
```

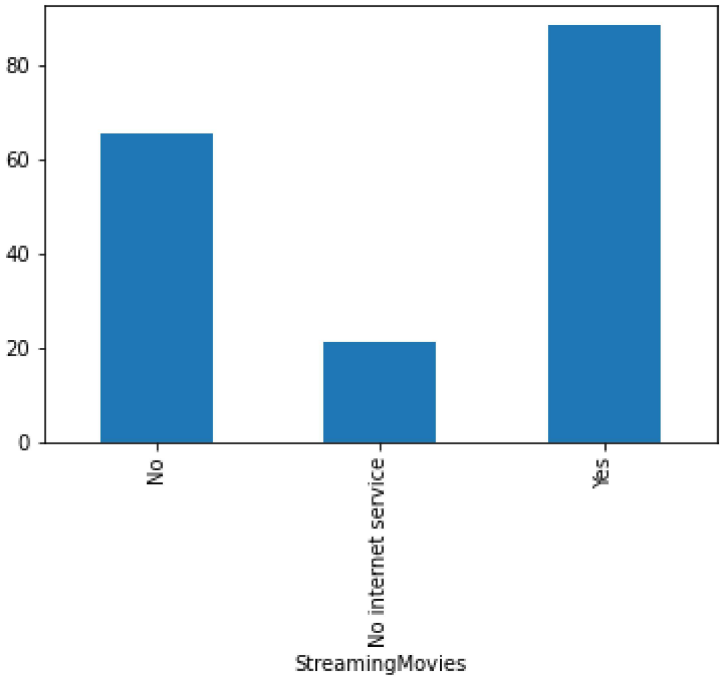



In [32]:
```
telco_churn_df.MonthlyCharges.groupby(telco_churn_df.StreamingMovies).mean()
```

Out[32]:
```
StreamingMovies
No                     65.434147
No internet service    21.079194
Yes                    88.475714
Name: MonthlyCharges, dtype: float64
```

In [33]:
```
telco_churn_df.MonthlyCharges.groupby(telco_churn_df.StreamingMovies).mean().plot(kind=
```

Out[33]:
```
<AxesSubplot:xlabel='StreamingMovies'>
```




In [34]:
```
# 1)Test Null Average MonthlyCharges Churn Yes/No Equal
from scipy.stats import ttest_ind
```

In [35]:
```
churnyes=telco_churn_df[telco_churn_df.Churn=="Yes"]
churnno=telco_churn_df[telco_churn_df.Churn=="No"]
print(churnyes.shape)
print(churnno.shape)
```

```
(1869, 21)
(5174, 21)
```

In [36]:
```
ttest_ind(churnyes.MonthlyCharges,churnno.MonthlyCharges,equal_var=False)
```

Out[36]:
```
Ttest_indResult(statistic=18.407526676414673, pvalue=8.592449331547539e-73)
```

In [37]:
```
# 8.592449331547539e-73 since p value is less than 0.05,reject null
# therefore ,there is no significance difference in average monthly charges
```

In [38]:
```
# 2) Test Null Average tenure of Churn Yes/No Equal
ttest_ind(churnyes.tenure,churnno.tenure,equal_var=False)
```

Out[38]:
```
Ttest_indResult(statistic=-34.82381869631297, pvalue=1.1954945472607148e-232)
```

In [39]:
```
# 1.1954945472607148e-232, since p value is less than 0.05,reject null
# therefore ,there is no significance difference in average tenure
```

In [40]:
```
# 3) Test Null Average Monthly Charges of different PaymentMethod Equal
telco_churn_df.PaymentMethod.value_counts()
```

Out[40]:
```
Electronic check             2365
Mailed check                 1612
Bank transfer (automatic)    1544
Credit card (automatic)      1522
Name: PaymentMethod, dtype: int64
```

In [41]:
```
from scipy.stats import f_oneway
```

In [42]:
```
EC=telco_churn_df[telco_churn_df.PaymentMethod=='Electronic check']
MC=telco_churn_df[telco_churn_df.PaymentMethod=='Mailed check']
BT=telco_churn_df[telco_churn_df.PaymentMethod=='Bank transfer (automatic)']
CC=telco_churn_df[telco_churn_df.PaymentMethod=='Credit card (automatic)']
```

In [43]:
```
f_oneway(EC.MonthlyCharges,MC.MonthlyCharges,BT.MonthlyCharges,CC.MonthlyCharges)
```

Out[43]:
```
F_onewayResult(statistic=450.3189918892516, pvalue=1.1802197193575694e-267)
```

In [44]:
```
# 1.1802197193575694e-267 ,since p values is less than 0.05,reject null
#therefore,there is no signigicant difference between average monthly charges
```

In [45]:
```
# 4) Test Null Average tenure of different PaymentMethod Equal
f_oneway(EC.tenure,MC.tenure,BT.tenure,CC.tenure)
```

Out[45]:
```
F_onewayResult(statistic=446.46688624797173, pvalue=1.503848361277172e-265)
```

In [46]:
```
# 1.503848361277172e-265 ,since p values is less than 0.05,reject null
#therefore,there is no signigicant difference between average tenure
```

In [47]:
```
# 5)Test Null No Asssociation between gender & Churn
pd.crosstab(telco_churn_df.Churn,telco_churn_df.gender)
```

Out[47]:

| gender | Female | Male |
|---|---|---|
| **Churn** | | |
| **No** | 2549 | 2625 |
| **Yes** | 939 | 930 |

In [48]:

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

In [49]:
```
chi2_contingency(pd.crosstab(telco_churn_df.gender,telco_churn_df.Churn))
```

Out[49]:
```
(0.4840828822091383,
 0.48657873605618596,
 1,
 array([[2562.38989067,  925.61010933],
        [2611.61010933,  943.38989067]]))
```

In [50]:
```
# 0.48657873605618596, since p value is greater than 0.05 ,failed to reject null ,
# therefore, there is no association between gender and churn
```

In [51]:
```
# 6)Test Null No Association between SeniorCitizen & Churn
pd.crosstab(telco_churn_df.SeniorCitizen,telco_churn_df.Churn)
```

Out[51]:

| Churn | No | Yes |
|---|---|---|
| **SeniorCitizen** | | |
| **0** | 4508 | 1393 |
| **1** | 666 | 476 |

In [52]:
```
chi2_contingency(pd.crosstab(telco_churn_df.SeniorCitizen,telco_churn_df.Churn))
```

Out[52]:
```
(159.42630036838742,
 1.510066805092378e-36,
 1,
 array([[4335.05239245, 1565.94760755],
        [ 838.94760755,  303.05239245]]))
```

In [53]:
```
# 1.510066805092378e-36 , since p value is less than 0.05 ,reject null,
# therefore, there is association between Seniorcitizen and churn
```

In [54]:
```
telco_churn_df.TotalCharges=telco_churn_df.TotalCharges.replace(' ',1400)
```

In [55]:
```
telco_churn_df.TotalCharges=pd.to_numeric(telco_churn_df.TotalCharges)
```

In [56]:
```
telco_churn_df.SeniorCitizen=telco_churn_df.SeniorCitizen.astype(object)
```

In [57]:
```
telco_churn_df.dtypes
```

Out[57]:
```
customerID          object
gender              object
SeniorCitizen       object
Partner             object
Dependents          object
tenure               int64
```

```
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                object
dtype: object
```

In [58]:
```
numericcols=telco_churn_df.select_dtypes(include=np.number)
objectcols=telco_churn_df.select_dtypes(include=['object'])
```

In [59]:
```
numericcols.head()
```

Out[59]:

| | 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 [60]:
```
objectcols.head()
```

Out[60]:

| | customerID | gender | SeniorCitizen | Partner | Dependents | PhoneService | MultipleLines | InternetServic |
|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | Female | 0 | Yes | No | No | No phone service | DS |
| **1** | 5575-GNVDE | Male | 0 | No | No | Yes | No | DS |
| **2** | 3668-QPYBK | Male | 0 | No | No | Yes | No | DS |
| **3** | 7795-CFOCW | Male | 0 | No | No | No | No phone service | DS |
| **4** | 9237-HQITU | Female | 0 | No | No | Yes | No | Fiber opti |

In [61]:
```
objectcols=objectcols.drop(['customerID'],axis=1)
```

LABEL ENCODING

In [62]:
```
from sklearn.preprocessing import LabelEncoder
```

In [63]:
```
le=LabelEncoder()
```

In [64]:
```
objectcolsdummy=objectcols.apply(le.fit_transform)
```

In [65]:
```
combinedf=pd.concat([numericcols,objectcolsdummy],axis=1)
```

In [66]:
```
combinedf.shape
```

Out[66]:
```
(7043, 20)
```

In [67]:
```
combinedf.tail()
```

Out[67]:

| | tenure | MonthlyCharges | TotalCharges | gender | SeniorCitizen | Partner | Dependents | PhoneService |
|---|---|---|---|---|---|---|---|---|
| **7038** | 24 | 84.80 | 1990.50 | 1 | 0 | 1 | 1 | 1 |
| **7039** | 72 | 103.20 | 7362.90 | 0 | 0 | 1 | 1 | 1 |
| **7040** | 11 | 29.60 | 346.45 | 0 | 0 | 1 | 1 | 0 |
| **7041** | 4 | 74.40 | 306.60 | 1 | 1 | 1 | 0 | 1 |
| **7042** | 66 | 105.65 | 6844.50 | 1 | 0 | 0 | 0 | 1 |

In [68]:
```
y=combinedf.Churn
X=combinedf.drop('Churn',axis=1)
```

BINARY LOGISTIC REGRESSION

In [69]:
```
from sklearn.linear_model import LogisticRegression
```

In [70]:
```
logreg=LogisticRegression(max_iter=1000)
```

In [71]:
```
logregmodel=logreg.fit(X,y)
```

In [72]:
```
logregmodel.score(X,y)
```

Out[72]:
```
0.8060485588527616
```

In [73]:
```
logregpredict=logregmodel.predict(X)
```

In [74]:
```
pd.crosstab(y,logregpredict)
```

Out[74]:

| col_0 | 0 | 1 |
|---|---|---|
| Churn | | |
| 0 | 4613 | 561 |
| 1 | 805 | 1064 |

In [75]:
```
(4613+1064)/(4613+561+805+1064)
```

Out[75]:
```
0.8060485588527616
```

In [76]:
```
telco_churn_df.TotalCharges.describe()
```

Out[76]:
```
count    7043.000000
mean     2281.920872
std      2265.268861
min        18.800000
25%       402.225000
50%      1400.000000
75%      3786.600000
max      8684.800000
Name: TotalCharges, dtype: float64
```

In [77]:
```
from sklearn.metrics import classification_report
```

In [78]:
```
print(classification_report(y,logregpredict))
```

```
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      5174
           1       0.65      0.57      0.61      1869

    accuracy                           0.81      7043
   macro avg       0.75      0.73      0.74      7043
weighted avg       0.80      0.81      0.80      7043
```

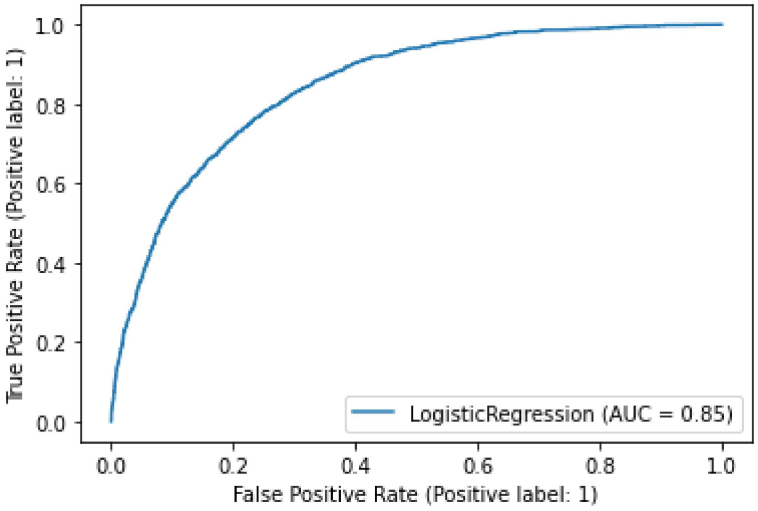
In [79]:
```
from sklearn.metrics import plot_roc_curve
```

In [80]:
```
plot_roc_curve(logreg,X,y)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarnin
g: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecated
in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metric.R
ocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estimator
```

```
`.
  warnings.warn(msg, category=FutureWarning)
```

Out[80]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1801e706580>
```




DECISION TREE

In [81]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [82]:
```
tree=DecisionTreeClassifier(max_depth=10)
```

In [83]:
```
treemodel=tree.fit(X,y)
```

In [84]:
```
treemodel.score(X,y)
```

Out[84]:
```
0.861706659094136
```

In [85]:
```
treepredict=treemodel.predict(X)
```

In [86]:
```
print(classification_report(y,treepredict))
```

```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91      5174
           1       0.79      0.65      0.71      1869

    accuracy                           0.86      7043
   macro avg       0.84      0.80      0.81      7043
weighted avg       0.86      0.86      0.86      7043
```

In [87]:
```
pd.crosstab(y,treepredict)
```

Out[87]:

| col_0 | 0 | 1 |
|---|---|---|

| Contract | 0 | 1 |
|---|---|---|
| **Churn** | | |
| **0** | 4848 | 326 |
| **1** | 648 | 1221 |

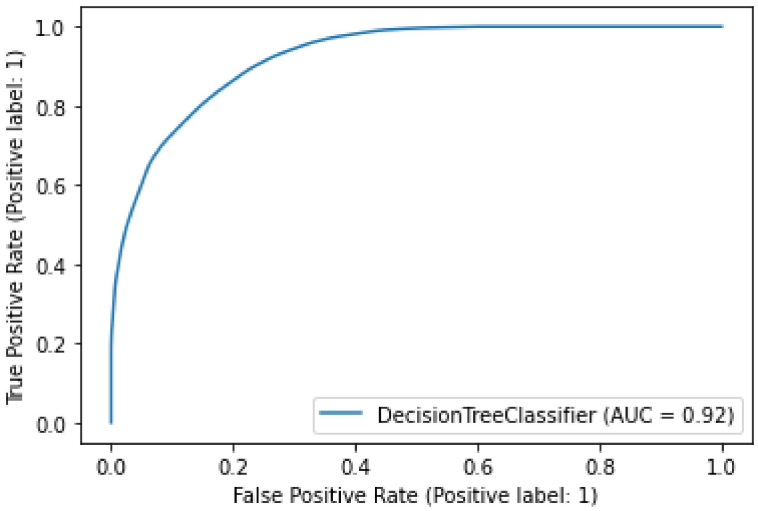
In [88]:
```
(4849+1219)/(4849+325+650+1219)
```

Out[88]:
```
0.8615646741445406
```

In [89]:
```
plot_roc_curve(tree,X,y)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarnin
g: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecated
in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metric.R
ocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estimator
`.
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x180173e50d0>
```

Out[89]:




RANDOM FOREST

In [90]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [91]:
```
RF=RandomForestClassifier(n_estimators=1000)
```

In [92]:
```
RFmodel=RF.fit(X,y)
```

In [93]:
```
RFmodel.score(X,y)
```

Out[93]:
```
0.9974442709072838
```

In [94]:
```
RFpredict=RFmodel.predict(X)
```

In [95]:
```
pd.crosstab(y,RFpredict)
```

Out[95]:

| Churn | col_0 0 | 1 |
|---|---|---|
| 0 | 5164 | 10 |
| 1 | 8 | 1861 |

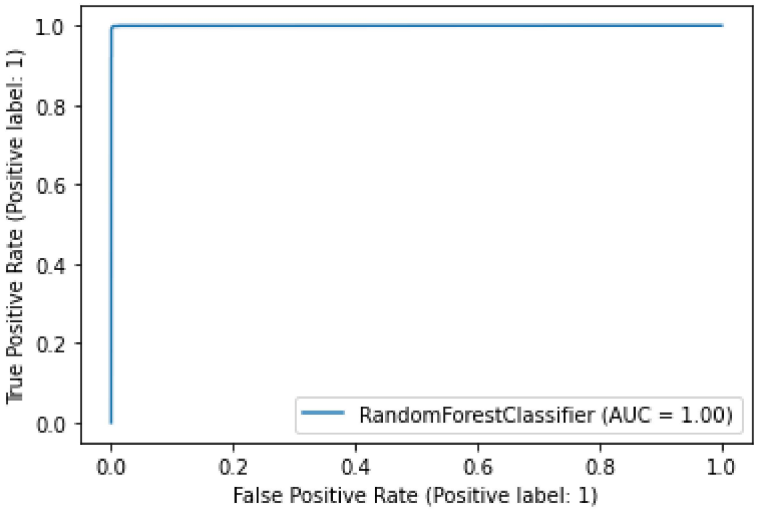
In [96]:
```
print(classification_report(y,RFpredict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5174
           1       0.99      1.00      1.00      1869

    accuracy                           1.00      7043
   macro avg       1.00      1.00      1.00      7043
weighted avg       1.00      1.00      1.00      7043
```

In [97]:
```
plot_roc_curve(RF,X,y)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarnin
g: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecated
in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metric.R
ocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estimator
`.
  warnings.warn(msg, category=FutureWarning)
```

Out[97]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1801e9d5640>
```




## GRADIENT BOOSTING MACHINES

In [98]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [99]:
```
gbm=GradientBoostingClassifier(n_estimators=3000)
```

In [100…
```
gbmmodel=gbm.fit(X,y)
```

In [101…
```
gbmmodel.score(X,y)
```

Out[101…
```
0.9781343177623172
```

In [102…
```
gbmpredict=gbmmodel.predict(X)
```

In [103…
```
pd.crosstab(y,gbmpredict)
```

Out[103…

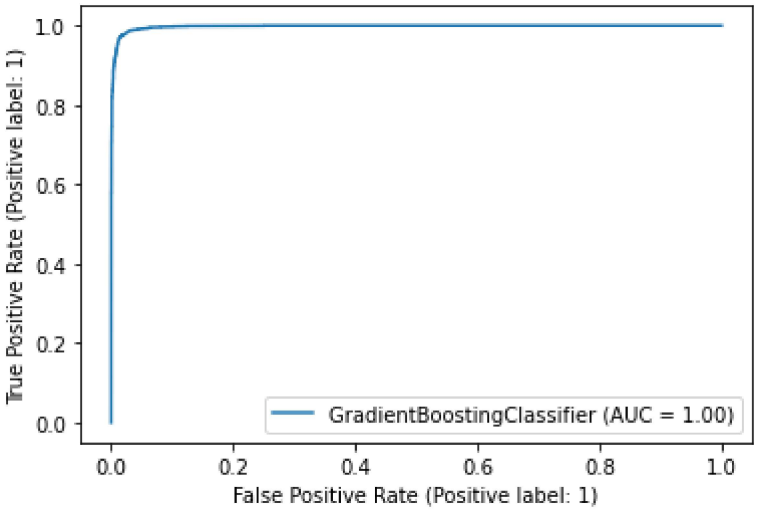
| Churn | col_0 0 | 1 |
|---|---|---|
| 0 | 5123 | 51 |
| 1 | 103 | 1766 |

In [104…
```
print(classification_report(y,gbmpredict))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      5174
           1       0.97      0.94      0.96      1869

    accuracy                           0.98      7043
   macro avg       0.98      0.97      0.97      7043
weighted avg       0.98      0.98      0.98      7043
```

In [105…
```
plot_roc_curve(gbm,X,y)
```

Out[105…
```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarnin
g: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecated
in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metric.R
ocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estimator
`.
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1802ca1ac10>
```




## GAUSSIAN NAIVE BAYES

In [106…
```
from sklearn.naive_bayes import GaussianNB
```

In [107…
```
GNB=GaussianNB()
```

In [108…
```
GNBmodel=GNB.fit(X,y)
```

In [109…
```
GNBmodel.score(X,y)
```

Out[109… 0.752804202754508

In [110…
```
GNBpredict=GNBmodel.predict(X)
```

In [111…
```
print(classification_report(y,GNBpredict))
```

```
              precision    recall  f1-score   support

           0       0.89      0.76      0.82      5174
           1       0.52      0.74      0.61      1869

    accuracy                           0.75      7043
   macro avg       0.71      0.75      0.72      7043
weighted avg       0.79      0.75      0.76      7043
```

In [112…
```
pd.crosstab(y,GNBpredict)
```

Out[112…

| col_0 | 0 | 1 |
|---|---|---|
| **Churn** | | |
| **0** | 3921 | 1253 |
| **1** | 488 | 1381 |

In [113…

```
plot_roc_curve(GNB,X,y)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarnin
g: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecated
in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metric.R
ocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estimator
`.
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x1802ca90d30>
```

Out[113…