Data Mining: Predictive Analysis using Random Forest

In [1]:
```
#import packages
import pandas as pd
import numpy as np
from numpy import array

import matplotlib.pyplot as plt

%matplotlib inline

#set the parameters for figure sizes
print("Before, figure default size is: ", plt.rcParams["figure.figsize"])
plt.rcParams["figure.figsize"] = (10, 10)
print("After, figure default size is: ", plt.rcParams["figure.figsize"])

import seaborn as sns

from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import r2_score

from sklearn import ensemble
from sklearn.ensemble import RandomForestRegressor
```

```
Before, figure default size is:  [6.0, 4.0]
After, figure default size is:  [10.0, 10.0]
```

In [2]:
```
#data cleaning
df = pd.read_csv('churn_clean.csv')
df.head().T
```

Out[2]:

	0	1	2	
CaseOrder	1	2	3	
Customer_id	K409198	S120509	K191035	
Interaction	aa90260b-4141-4a24-8e36-b04ce1f4f77b	fb76459f-c047-4a9d-8af9-e0f7d4ac2524	344d114c-3736-4be5-98f7-c72c281e2d35	abfa
UID	e885b299883d4f9fb18e39c75155d990	f2de8bef964785f41a2959829830fb8a	f1784cfa9f6d92ae816197eb175d3c71	dc8a36507724
City	Point Baker	West Branch	Yamhill	
State	AK	MI	OR	
County	Prince of Wales-Hyder	Ogemaw	Yamhill	
Zip	99927	48661	97148	
Lat	56.251	44.32893	45.35589	
Lng	-133.37571	-84.2408	-123.24657	
Population	38	10446	3735	
Area	Urban	Urban	Urban	
TimeZone	America/Sitka	America/Detroit	America/Los_Angeles	
Job	Environmental health practitioner	Programmer, multimedia	Chief Financial Officer	
Children	0	1	4	
Age	68	27	50	
Income	28561.99	21704.77	9609.57	
Marital	Widowed	Married	Widowed	
Gender	Male	Female	Female	
Churn	No	Yes	No	
Outage_sec_perweek	7.978323	11.69908	10.7528	
Email	10	12	9	
Contacts	0	0	0	
Yearly_equip_failure	1	1	1	
Techie	No	Yes	Yes	
Contract	One year	Month-to-month	Two Year	
Port_modem	Yes	No	Yes	

	0	1	2
Tablet	Yes	Yes	No
InternetService	Fiber Optic	Fiber Optic	DSL
Phone	Yes	Yes	Yes
Multiple	No	Yes	Yes
OnlineSecurity	Yes	Yes	No
OnlineBackup	Yes	No	No
DeviceProtection	No	No	No
TechSupport	No	No	No
StreamingTV	No	Yes	No
StreamingMovies	Yes	Yes	Yes
PaperlessBilling	Yes	Yes	Yes
PaymentMethod	Credit Card (automatic)	Bank Transfer(automatic)	Credit Card (automatic)
Tenure	6.795513	1.156681	15.754144
MonthlyCharge	172.455519	242.632554	159.947583
Bandwidth_GB_Year	904.53611	800.982766	2054.706961
Item1	5	3	4
Item2	5	4	4
Item3	5	3	2
Item4	3	3	4
Item5	4	4	4
Item6	4	3	3
Item7	3	4	3
Item8	4	4	3

In [3]:
```
df.describe().T
```

Out[3]:

	count	mean	std	min	25%	50%	75%	max
CaseOrder	10000.0	5000.500000	2886.895680	1.000000	2500.750000	5000.500000	7500.250000	10000.000000
Zip	10000.0	49153.319600	27532.196108	601.000000	26292.500000	48869.500000	71866.500000	99929.000000
Lat	10000.0	38.757567	5.437389	17.966120	35.341828	39.395800	42.106908	70.640660
Lng	10000.0	-90.782536	15.156142	-171.688150	-97.082812	-87.918800	-80.088745	-65.667850
Population	10000.0	9756.562400	14432.698671	0.000000	738.000000	2910.500000	13168.000000	111850.000000
Children	10000.0	2.087700	2.147200	0.000000	0.000000	1.000000	3.000000	10.000000
Age	10000.0	53.078400	20.698882	18.000000	35.000000	53.000000	71.000000	89.000000
Income	10000.0	39806.926771	28199.916702	348.670000	19224.717500	33170.605000	53246.170000	258900.700000
Outage_sec_perweek	10000.0	10.001848	2.976019	0.099747	8.018214	10.018560	11.969485	21.207230
Email	10000.0	12.016000	3.025898	1.000000	10.000000	12.000000	14.000000	23.000000
Contacts	10000.0	0.994200	0.988466	0.000000	0.000000	1.000000	2.000000	7.000000
Yearly_equip_failure	10000.0	0.398000	0.635953	0.000000	0.000000	0.000000	1.000000	6.000000
Tenure	10000.0	34.526188	26.443063	1.000259	7.917694	35.430507	61.479795	71.999280
MonthlyCharge	10000.0	172.624816	42.943094	79.978860	139.979239	167.484700	200.734725	290.160419
Bandwidth_GB_Year	10000.0	3392.341550	2185.294852	155.506715	1236.470827	3279.536903	5586.141370	7158.981530
Item1	10000.0	3.490800	1.037797	1.000000	3.000000	3.000000	4.000000	7.000000
Item2	10000.0	3.505100	1.034641	1.000000	3.000000	4.000000	4.000000	7.000000
Item3	10000.0	3.487000	1.027977	1.000000	3.000000	3.000000	4.000000	8.000000
Item4	10000.0	3.497500	1.025816	1.000000	3.000000	3.000000	4.000000	7.000000
Item5	10000.0	3.492900	1.024819	1.000000	3.000000	3.000000	4.000000	7.000000
Item6	10000.0	3.497300	1.033586	1.000000	3.000000	3.000000	4.000000	8.000000
Item7	10000.0	3.509500	1.028502	1.000000	3.000000	4.000000	4.000000	7.000000
Item8	10000.0	3.495600	1.028633	1.000000	3.000000	3.000000	4.000000	8.000000

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CaseOrder             10000 non-null  int64
 1   Customer_id           10000 non-null  object
 2   Interaction           10000 non-null  object
 3   UID                   10000 non-null  object
 4   City                  10000 non-null  object
 5   State                 10000 non-null  object
 6   County                10000 non-null  object
 7   Zip                   10000 non-null  int64
 8   Lat                   10000 non-null  float64
 9   Lng                   10000 non-null  float64
 10  Population            10000 non-null  int64
 11  Area                  10000 non-null  object
 12  TimeZone              10000 non-null  object
 13  Job                   10000 non-null  object
 14  Children              10000 non-null  int64
 15  Age                   10000 non-null  int64
 16  Income                10000 non-null  float64
 17  Marital               10000 non-null  object
 18  Gender                10000 non-null  object
 19  Churn                 10000 non-null  object
 20  Outage_sec_perweek    10000 non-null  float64
 21  Email                 10000 non-null  int64
 22  Contacts              10000 non-null  int64
 23  Yearly_equip_failure  10000 non-null  int64
 24  Techie                10000 non-null  object
 25  Contract              10000 non-null  object
 26  Port_modem            10000 non-null  object
 27  Tablet                10000 non-null  object
 28  InternetService       10000 non-null  object
 29  Phone                 10000 non-null  object
 30  Multiple              10000 non-null  object
 31  OnlineSecurity        10000 non-null  object
 32  OnlineBackup          10000 non-null  object
 33  DeviceProtection      10000 non-null  object
 34  TechSupport           10000 non-null  object
 35  StreamingTV           10000 non-null  object
 36  StreamingMovies       10000 non-null  object
 37  PaperlessBilling      10000 non-null  object
 38  PaymentMethod         10000 non-null  object
 39  Tenure                10000 non-null  float64
 40  MonthlyCharge         10000 non-null  float64
 41  Bandwidth_GB_Year     10000 non-null  float64
```

```
 42  Item1                  10000 non-null  int64
 43  Item2                  10000 non-null  int64
 44  Item3                  10000 non-null  int64
 45  Item4                  10000 non-null  int64
 46  Item5                  10000 non-null  int64
 47  Item6                  10000 non-null  int64
 48  Item7                  10000 non-null  int64
 49  Item8                  10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [5]:

```python
#rename features for easy use
prep_df = df.rename(columns={'CaseOrder': 'case_order',
                'Customer_id': 'customer_id',
               'Interaction': 'interaction',
               'UID': 'uid',
               'City': 'city',
               'State': 'state',
               'County':'county',
               'Zip': 'zip',
               'Lat': 'latitude',
               'Lng': 'longitude',
               'Population': 'population',
               'Area': 'area',
               'TimeZone': 'time_zone',
               'Job': 'job',
               'Children': 'children',
               'Age': 'age',
               'Income': 'income',
               'Marital': 'marital_status',
               'Gender': 'gender',
               'Churn': 'churn',
               'Outage_sec_perweek': 'outage_sec_perweek',
               'Email': 'email_correspondence',
               'Contacts': 'support_contacts',
               'Yearly_equip_failure': 'equip_fail_year',
               'Techie': 'techie',
               'Contract': 'contract',
               'Port_modem': 'port_modem',
               'Tablet': 'tablet',
               'InternetService': 'internet_service',
               'Phone': 'phone_service',
               'Multiple': 'multi_lines',
               'OnlineSecurity': 'online_security',
               'OnlineBackup': 'online_backup',
               'DeviceProtection': 'device_protection',
               'TechSupport': 'tech_support',
```

```
                'StreamingTV': 'streaming_tv',
                'StreamingMovies': 'streaming_movies',
                 'PaperlessBilling': 'paperless_billing',
                'PaymentMethod': 'payment_method',
                'Tenure': 'tenure',
                'MonthlyCharge':'monthly_charge',
                'Bandwidth_GB_Year': 'bandwidth_gb',
                'Item1': 'timely_response',
                'Item2': 'timely_fixes',
                'Item3': 'timely_replacements',
                'Item4': 'reliability',
                'Item5': 'options',
                'Item6': 'respectful_response',
                'Item7': 'courteous_exchange',
                'Item8': 'active_listening'})
```

In [6]:
```
prep_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   case_order            10000 non-null  int64
 1   customer_id           10000 non-null  object
 2   interaction           10000 non-null  object
 3   uid                   10000 non-null  object
 4   city                  10000 non-null  object
 5   state                 10000 non-null  object
 6   county                10000 non-null  object
 7   zip                   10000 non-null  int64
 8   latitude              10000 non-null  float64
 9   longitude             10000 non-null  float64
 10  population            10000 non-null  int64
 11  area                  10000 non-null  object
 12  time_zone             10000 non-null  object
 13  job                   10000 non-null  object
 14  children              10000 non-null  int64
 15  age                   10000 non-null  int64
 16  income                10000 non-null  float64
 17  marital_status        10000 non-null  object
 18  gender                10000 non-null  object
 19  churn                 10000 non-null  object
 20  outage_sec_perweek    10000 non-null  float64
 21  email_correspondence  10000 non-null  int64
 22  support_contacts      10000 non-null  int64
 23  equip_fail_year       10000 non-null  int64
 24  techie                10000 non-null  object
 25  contract              10000 non-null  object
 26  port_modem            10000 non-null  object
 27  tablet                10000 non-null  object
 28  internet_service      10000 non-null  object
 29  phone_service         10000 non-null  object
 30  multi_lines           10000 non-null  object
 31  online_security       10000 non-null  object
 32  online_backup         10000 non-null  object
 33  device_protection     10000 non-null  object
 34  tech_support          10000 non-null  object
 35  streaming_tv          10000 non-null  object
 36  streaming_movies      10000 non-null  object
 37  paperless_billing     10000 non-null  object
 38  payment_method        10000 non-null  object
 39  tenure                10000 non-null  float64
 40  monthly_charge        10000 non-null  float64
 41  bandwidth_gb          10000 non-null  float64
```

```
 42  timely_response       10000 non-null  int64
 43  timely_fixes          10000 non-null  int64
 44  timely_replacements   10000 non-null  int64
 45  reliability           10000 non-null  int64
 46  options               10000 non-null  int64
 47  respectful_response   10000 non-null  int64
 48  courteous_exchange    10000 non-null  int64
 49  active_listening      10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [7]:
```
prep_df.duplicated().any()
```

Out[7]:
```
False
```

In [8]:
```
prep_df.isna().any()
```

Out[8]:

case_order	False
customer_id	False
interaction	False
uid	False
city	False
state	False
county	False
zip	False
latitude	False
longitude	False
population	False
area	False
time_zone	False
job	False
children	False
age	False
income	False
marital_status	False
gender	False
churn	False
outage_sec_perweek	False
email_correspondence	False
support_contacts	False
equip_fail_year	False
techie	False
contract	False
port_modem	False
tablet	False
internet_service	False
phone_service	False
multi_lines	False
online_security	False
online_backup	False
device_protection	False
tech_support	False
streaming_tv	False
streaming_movies	False
paperless_billing	False
payment_method	False
tenure	False
monthly_charge	False
bandwidth_gb	False
timely_response	False
timely_fixes	False
timely_replacements	False
reliability	False
options	False

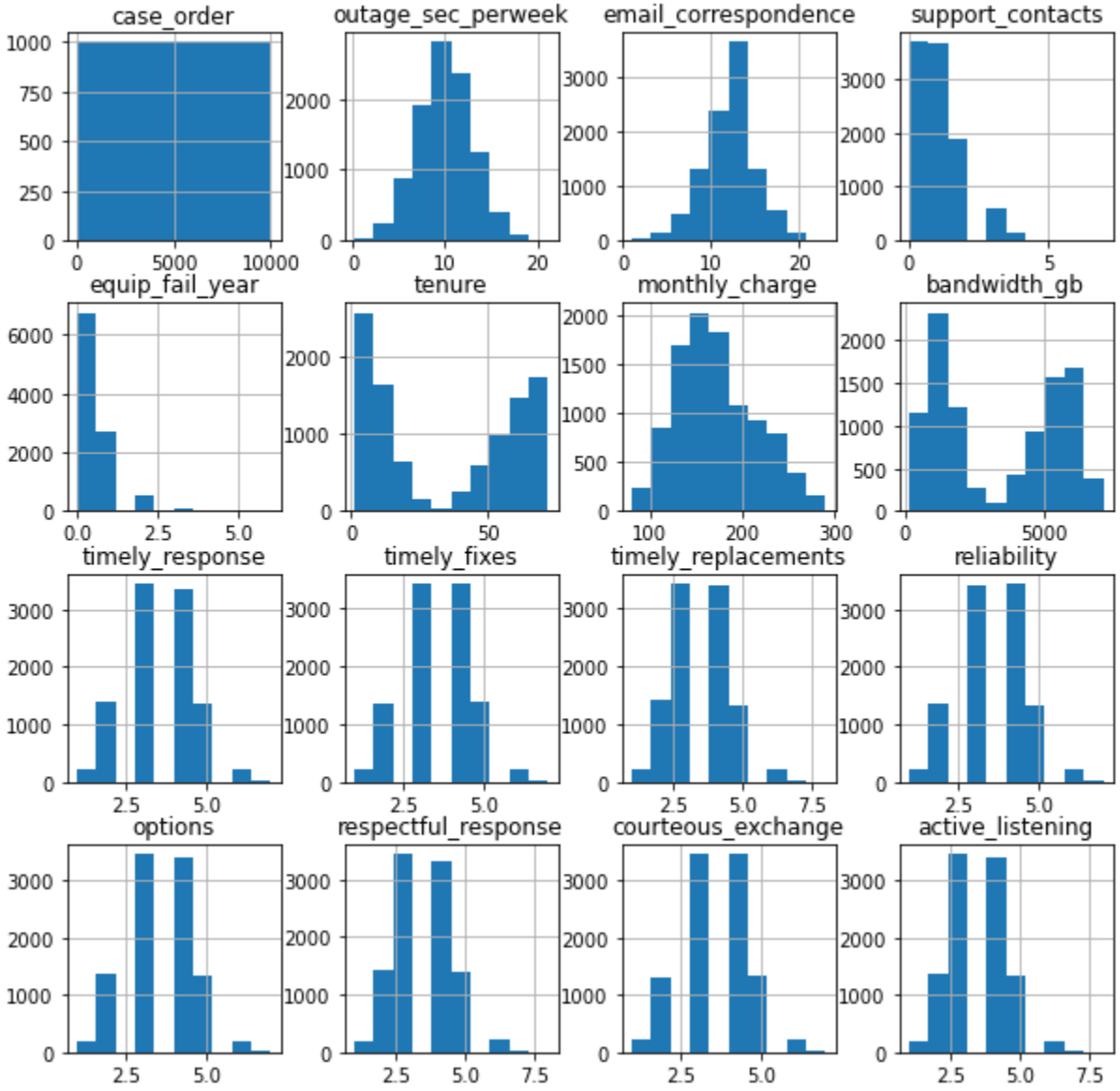
```
respectful_response     False
courteous_exchange      False
active_listening        False
dtype: bool
```

In [9]:

```
#remove extraneous columns not being used for further analysis
prep_df = prep_df.drop(columns = ['uid','interaction','customer_id','city','state','county','zip','latitude',
                                  'longitude','population','area','time_zone','job','children','age','income',
                                  'marital_status','gender','techie','churn'])
prep_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 30 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   case_order            10000 non-null  int64  
 1   outage_sec_perweek    10000 non-null  float64
 2   email_correspondence  10000 non-null  int64  
 3   support_contacts      10000 non-null  int64  
 4   equip_fail_year       10000 non-null  int64  
 5   contract              10000 non-null  object 
 6   port_modem            10000 non-null  object 
 7   tablet                10000 non-null  object 
 8   internet_service      10000 non-null  object 
 9   phone_service         10000 non-null  object 
 10  multi_lines           10000 non-null  object 
 11  online_security       10000 non-null  object 
 12  online_backup         10000 non-null  object 
 13  device_protection     10000 non-null  object 
 14  tech_support          10000 non-null  object 
 15  streaming_tv          10000 non-null  object 
 16  streaming_movies      10000 non-null  object 
 17  paperless_billing     10000 non-null  object 
 18  payment_method        10000 non-null  object 
 19  tenure                10000 non-null  float64
 20  monthly_charge        10000 non-null  float64
 21  bandwidth_gb          10000 non-null  float64
 22  timely_response       10000 non-null  int64  
 23  timely_fixes          10000 non-null  int64  
 24  timely_replacements   10000 non-null  int64  
 25  reliability           10000 non-null  int64  
 26  options               10000 non-null  int64  
 27  respectful_response   10000 non-null  int64  
 28  courteous_exchange    10000 non-null  int64  
 29  active_listening      10000 non-null  int64  
dtypes: float64(4), int64(12), object(14)
memory usage: 2.3+ MB
```

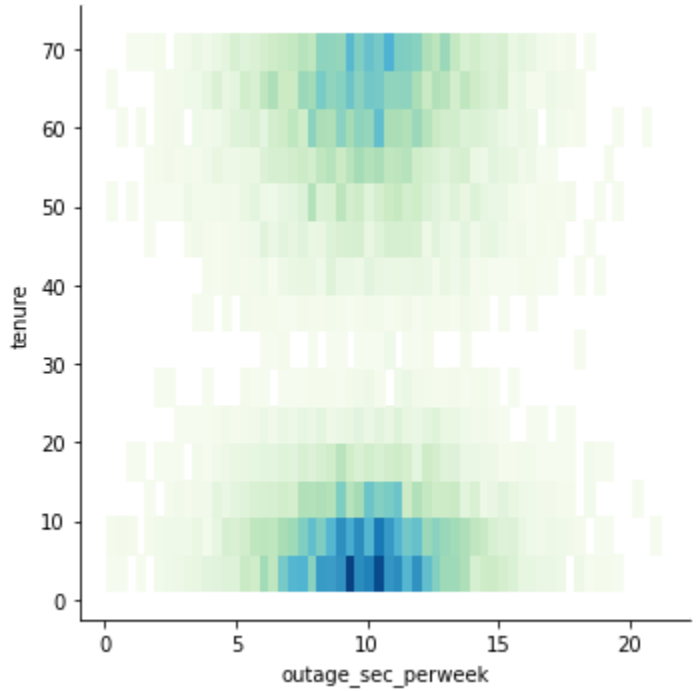
In [10]:
```
prep_df.hist();
```



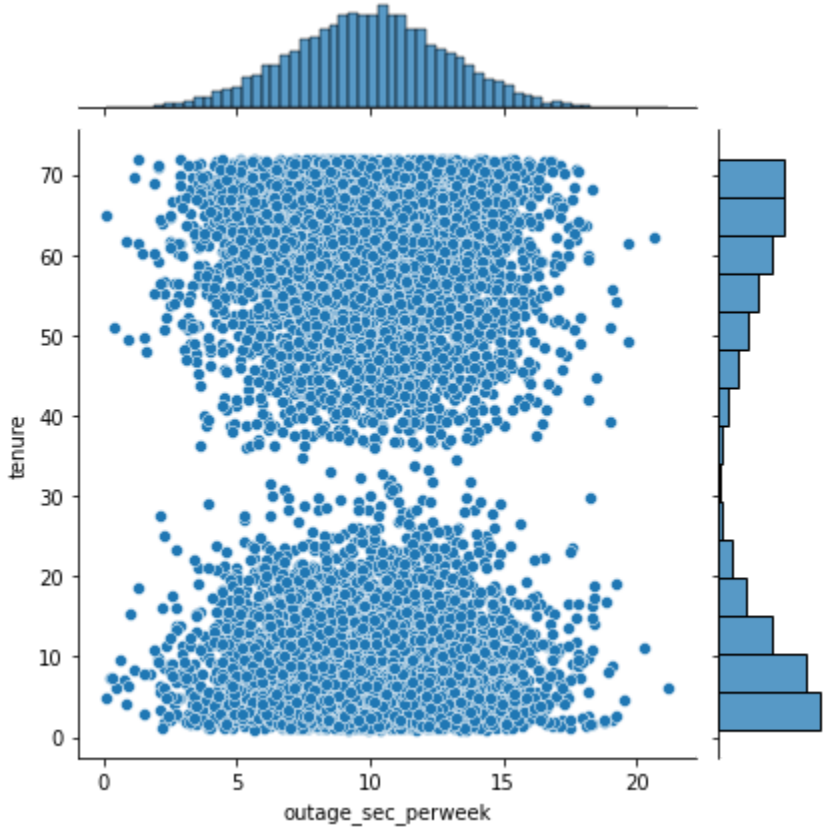
In [11]:
```
#EDA - initial features
prep_df['outage_sec_perweek'].describe()
```

Out[11]:
```
count    10000.000000
mean        10.001848
std          2.976019
min          0.099747
25%          8.018214
50%         10.018560
75%         11.969485
max         21.207230
Name: outage_sec_perweek, dtype: float64
```

In [12]:
```
sns.displot(x='outage_sec_perweek',y='tenure', cmap='GnBu', data=prep_df);
```



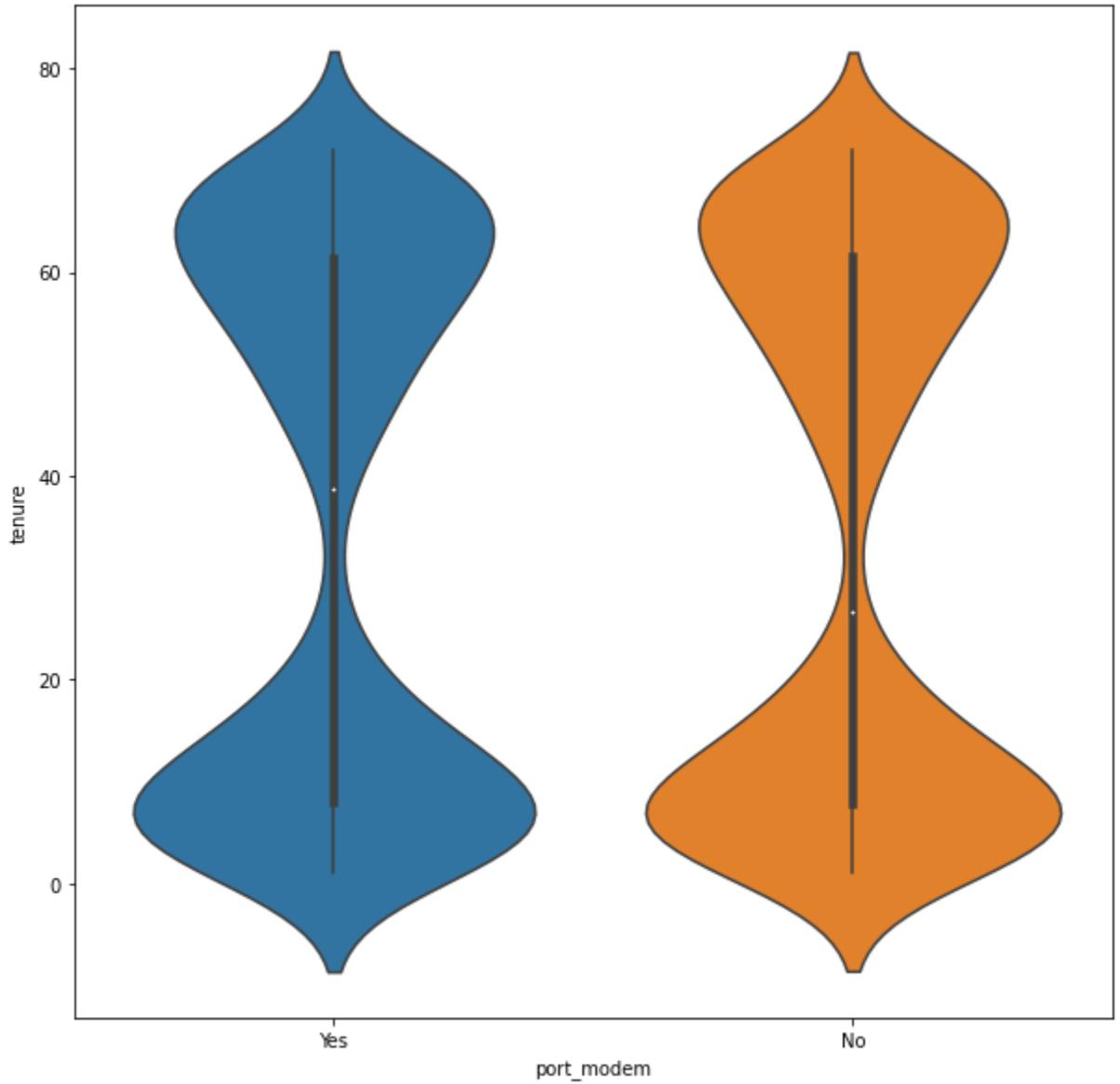
In [13]:
```
sns.jointplot(x='outage_sec_perweek', y='tenure', cmap='GnBu', data=prep_df);
```



In [14]:
```
prep_df['port_modem'].describe()
```

Out[14]:
```
count      10000
unique         2
top           No
freq        5166
Name: port_modem, dtype: object
```

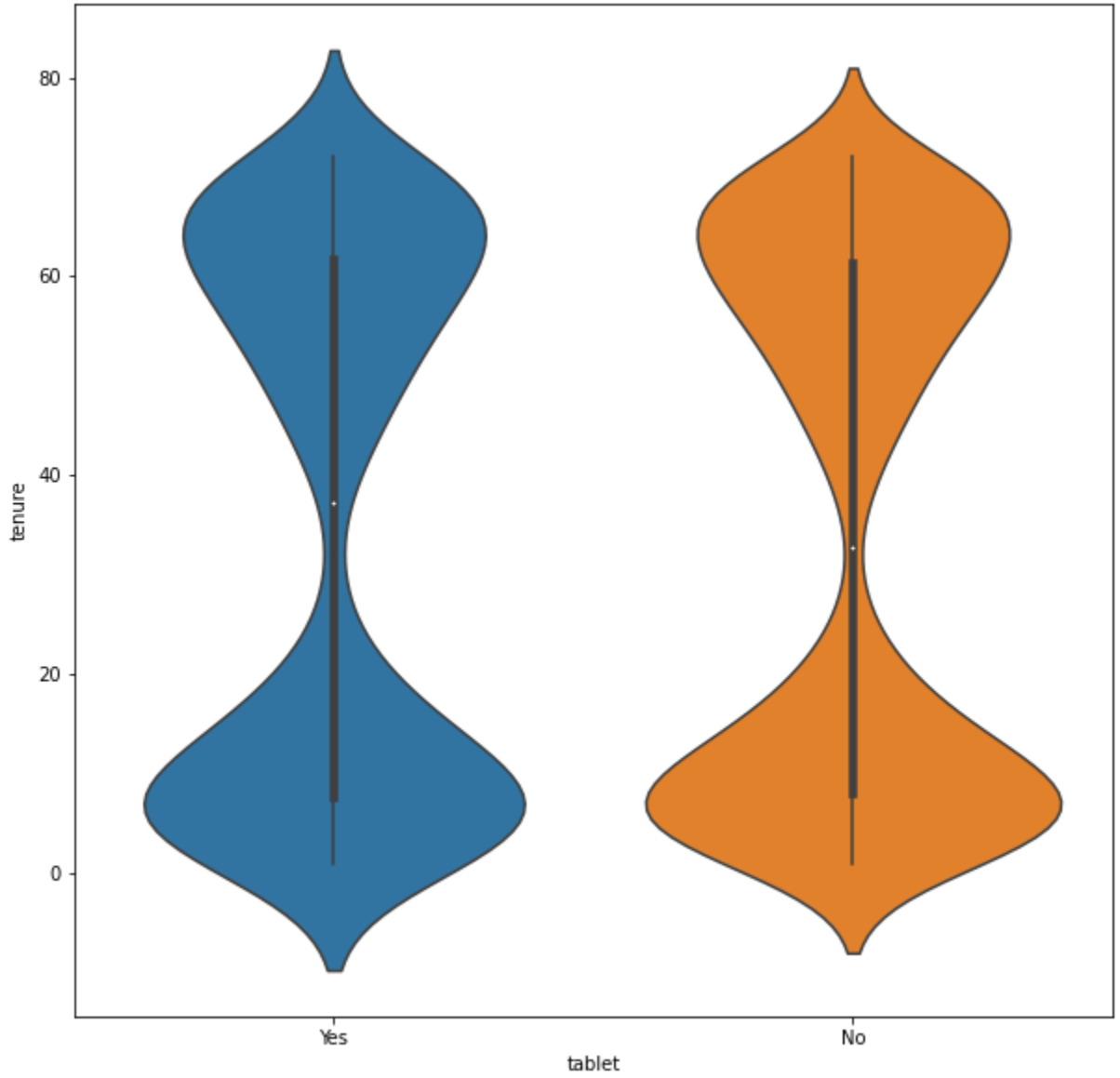
In [15]:
```
sns.violinplot(x='port_modem',y='tenure', data=prep_df);
```



In [16]:
```
prep_df['tablet'].describe()
```

Out[16]:
```
count     10000
unique        2
top          No
freq       7009
Name: tablet, dtype: object
```

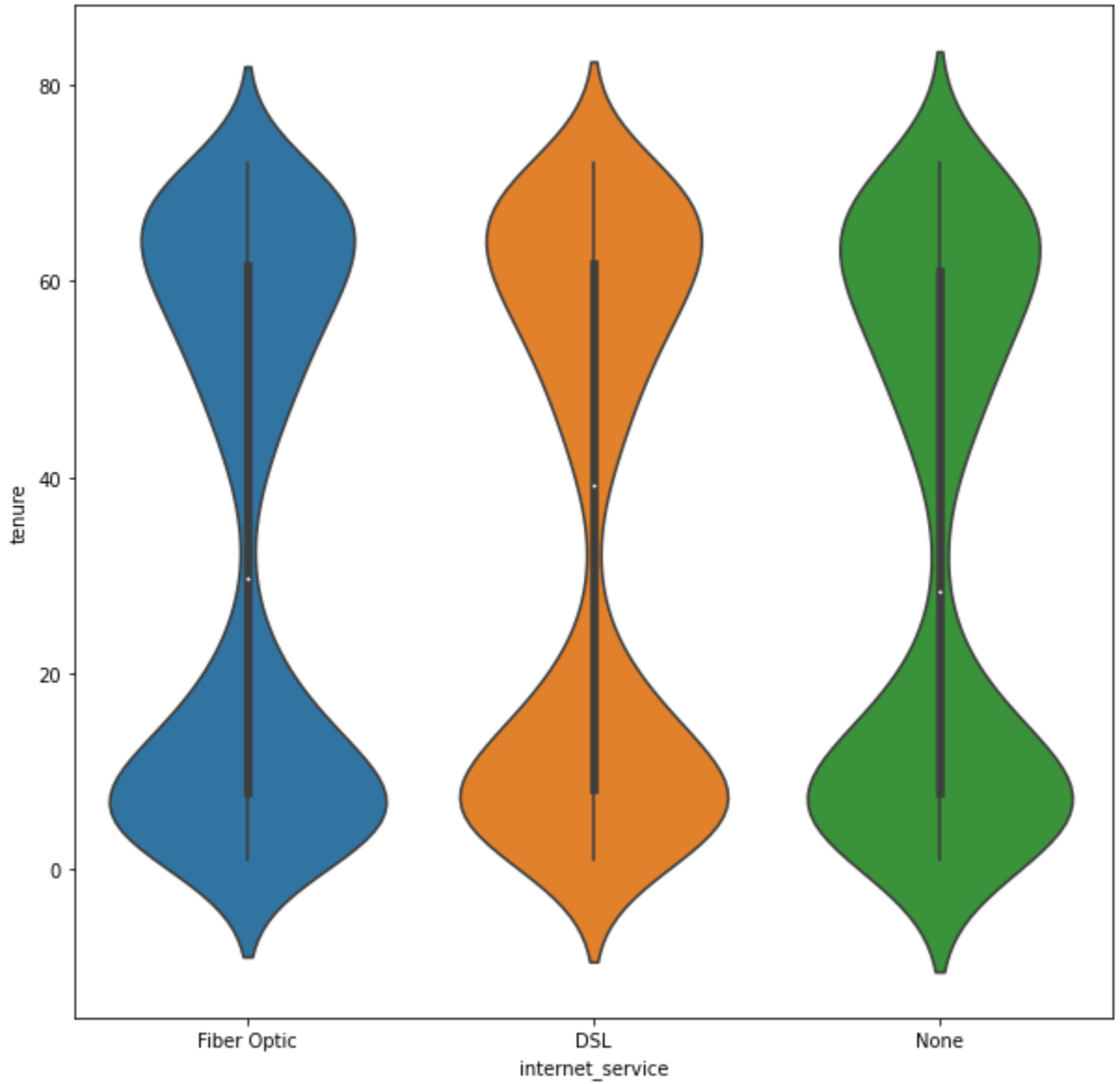
In [17]:
```
sns.violinplot(x='tablet',y='tenure', data=prep_df);
```



In [18]:
```
prep_df['internet_service'].describe()
```

Out[18]:
```
count           10000
unique              3
top       Fiber Optic
freq             4408
Name: internet_service, dtype: object
```

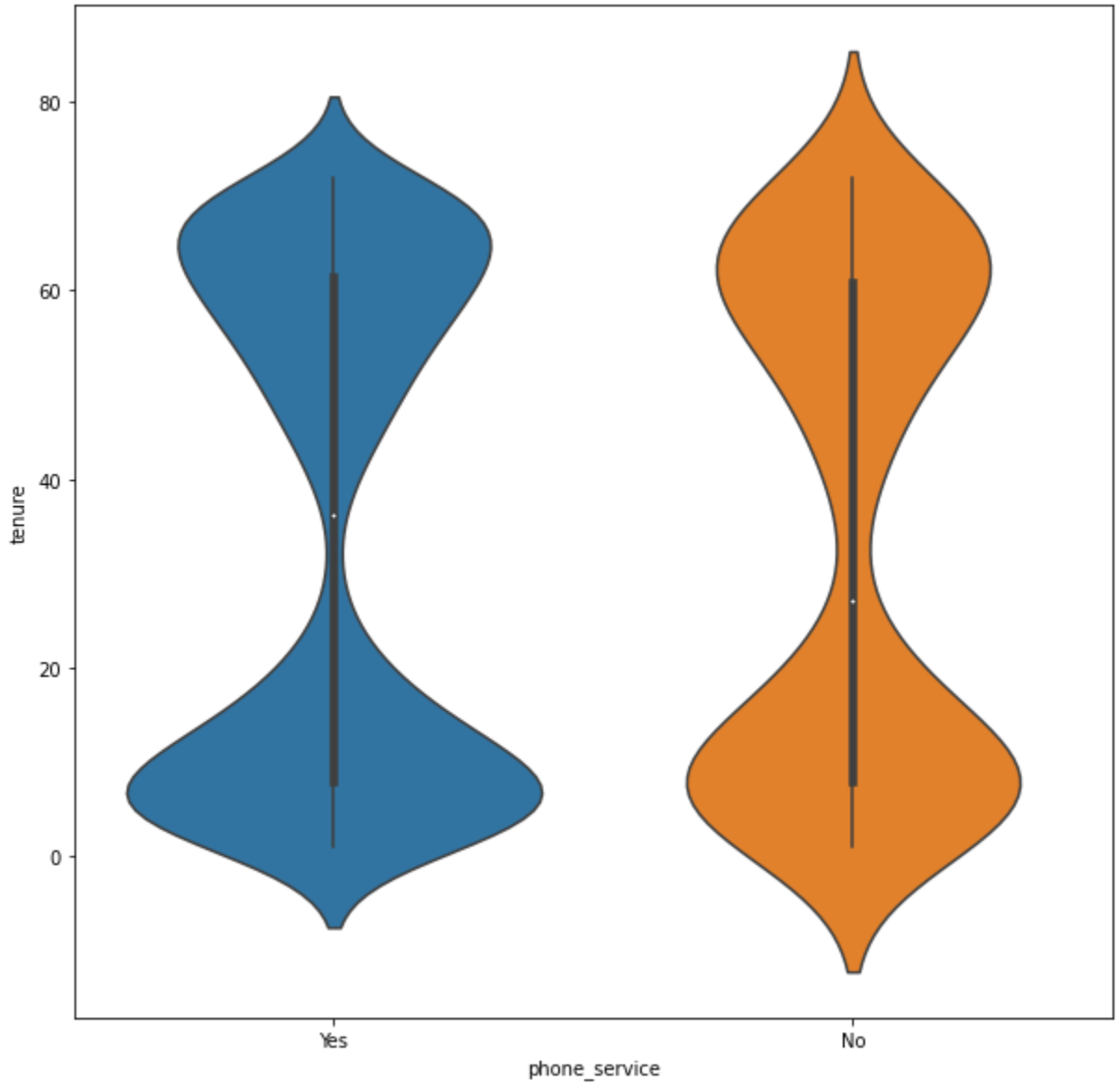
In [19]:
```
sns.violinplot(x='internet_service', y='tenure', data=prep_df);
```



In [20]:
```
prep_df['phone_service'].describe()
```

Out[20]:
```
count      10000
unique         2
top          Yes
freq        9067
Name: phone_service, dtype: object
```

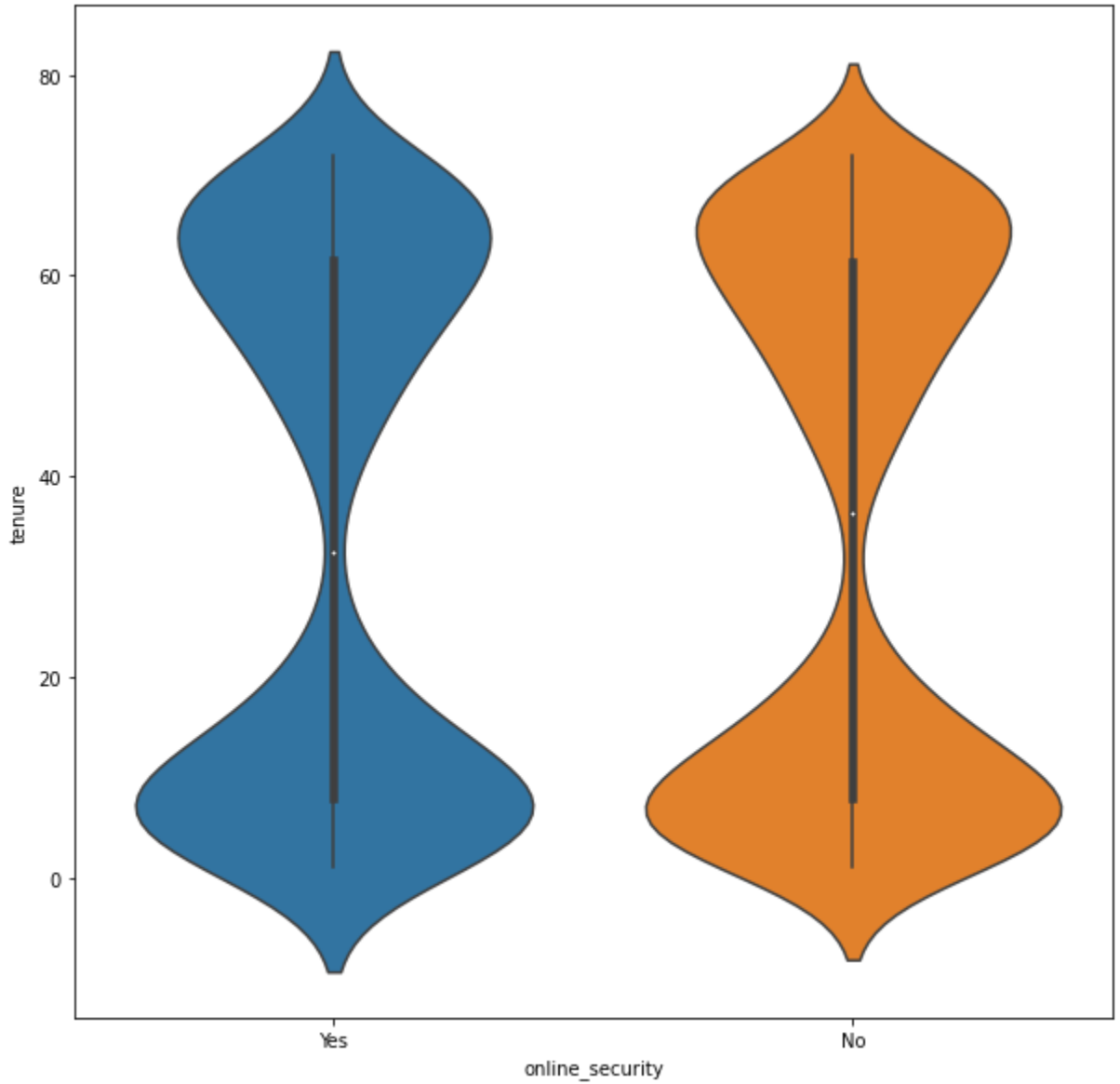
In [21]:
```
sns.violinplot(x='phone_service', y='tenure', data=prep_df);
```



In [22]:
```
prep_df['online_security'].describe()
```

Out[22]:
```
count      10000
unique         2
top           No
freq        6424
Name: online_security, dtype: object
```

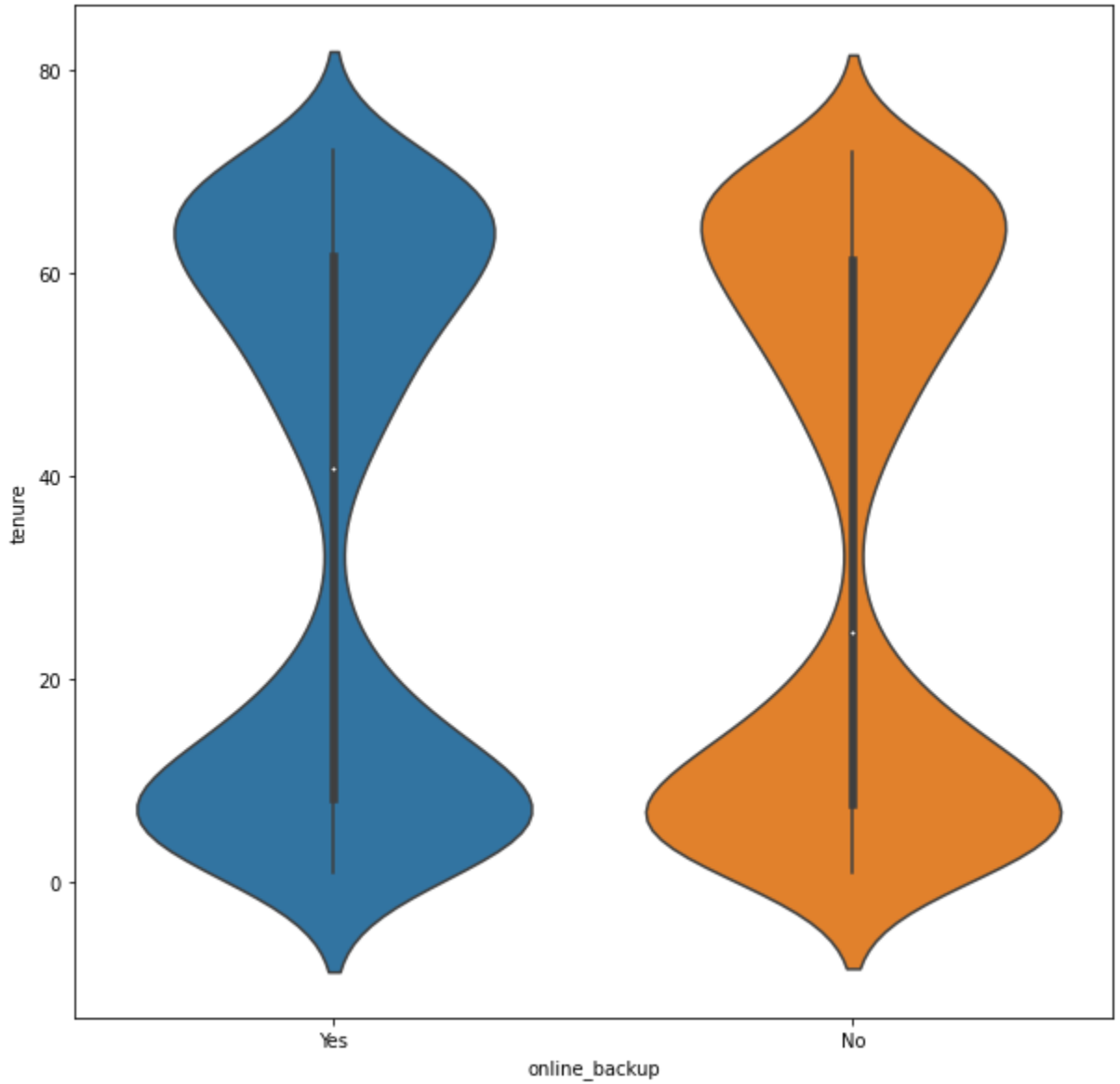
In [23]:
```
sns.violinplot(x='online_security', y='tenure', data=prep_df);
```



In [24]:
```
prep_df['online_backup'].describe()
```

Out[24]:
```
count      10000
unique         2
top           No
freq        5494
Name: online_backup, dtype: object
```

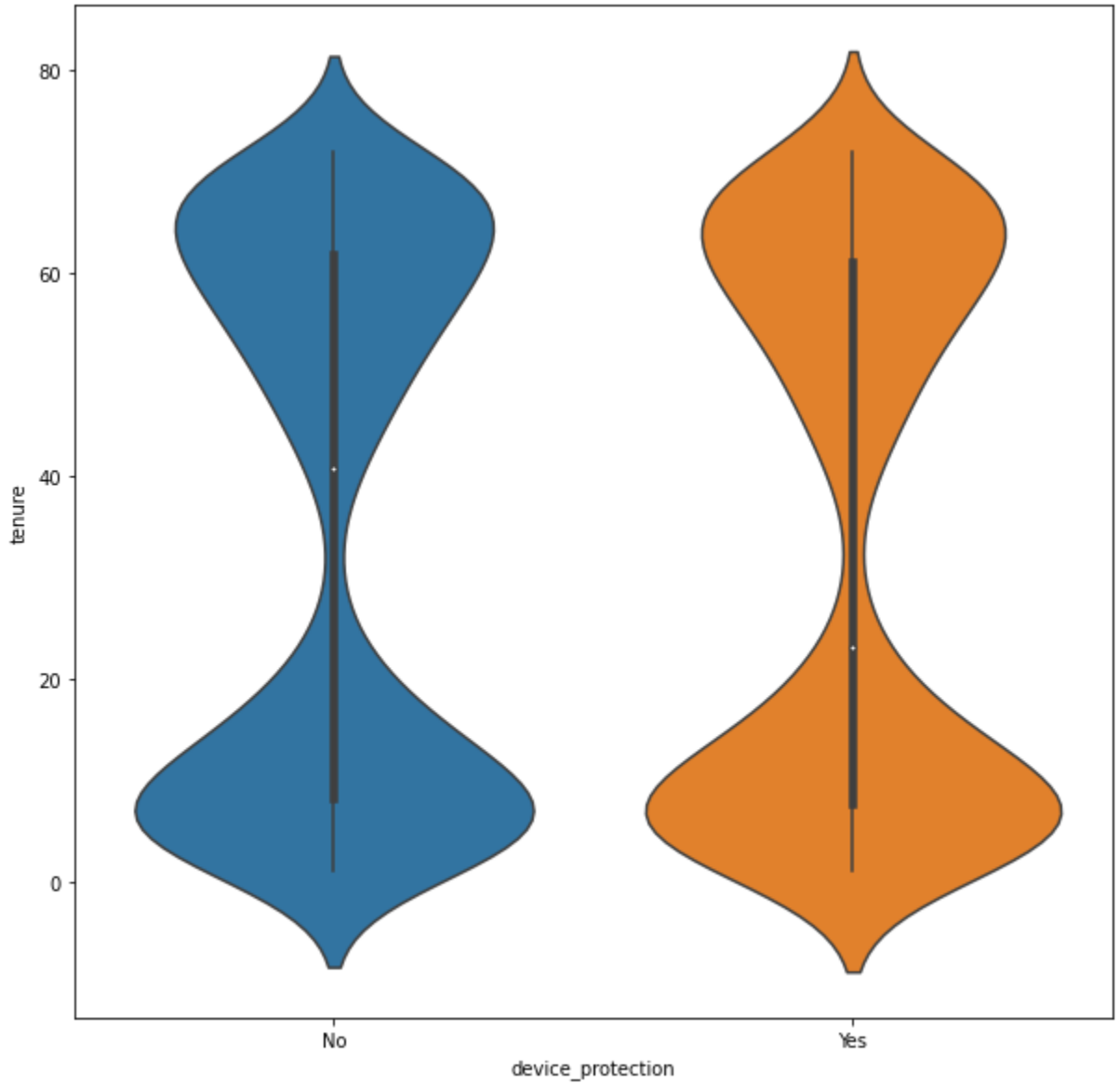
In [25]:
```
sns.violinplot(x='online_backup',y='tenure',data=prep_df);
```



In [26]:
```
prep_df['device_protection'].describe()
```

Out[26]:
```
count      10000
unique         2
top           No
freq        5614
Name: device_protection, dtype: object
```

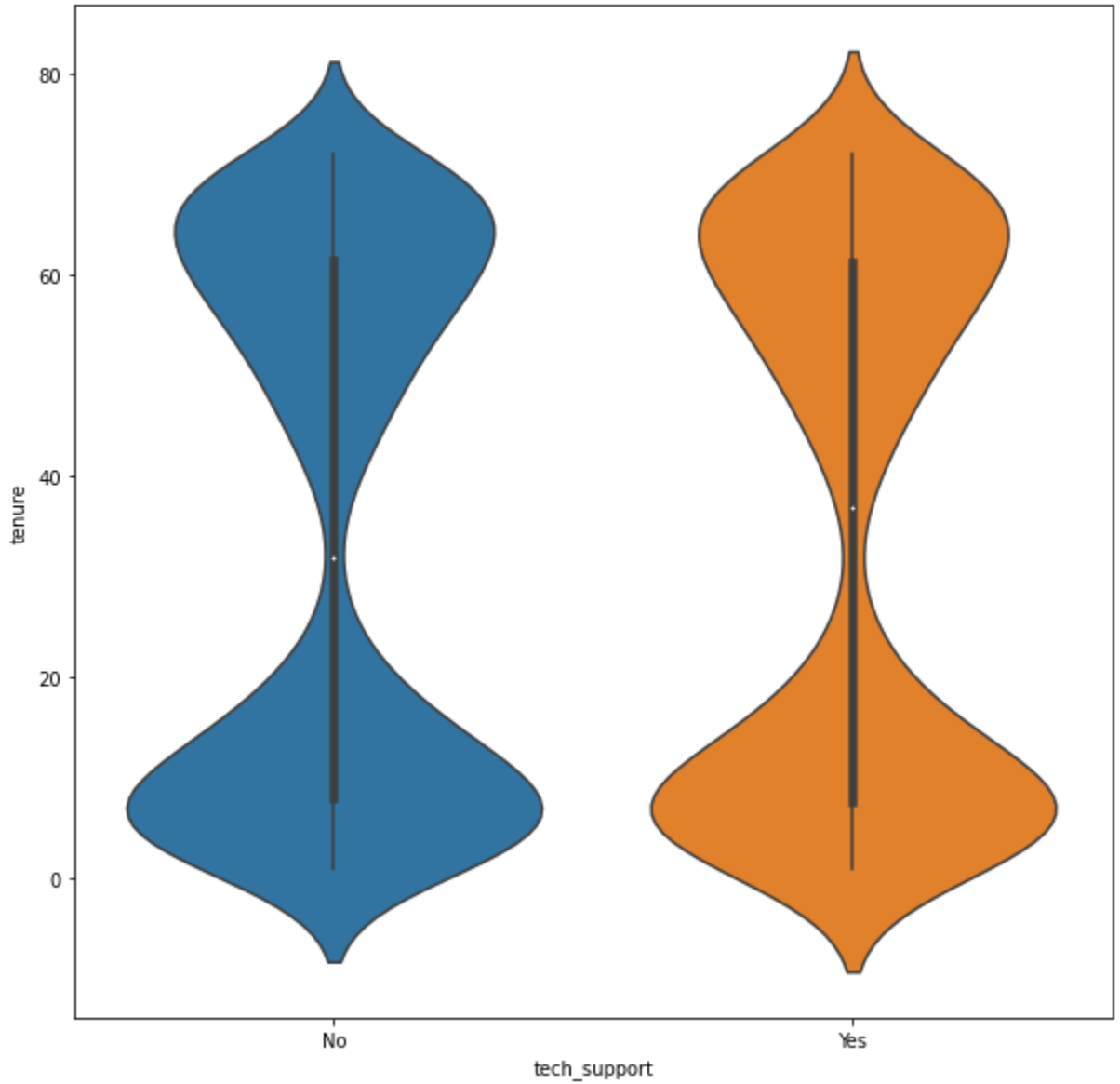
In [27]:
```
sns.violinplot(x='device_protection', y='tenure', data=prep_df);
```



In [28]:
```
prep_df['tech_support'].describe()
```

Out[28]:
```
count      10000
unique         2
top           No
freq        6250
Name: tech_support, dtype: object
```

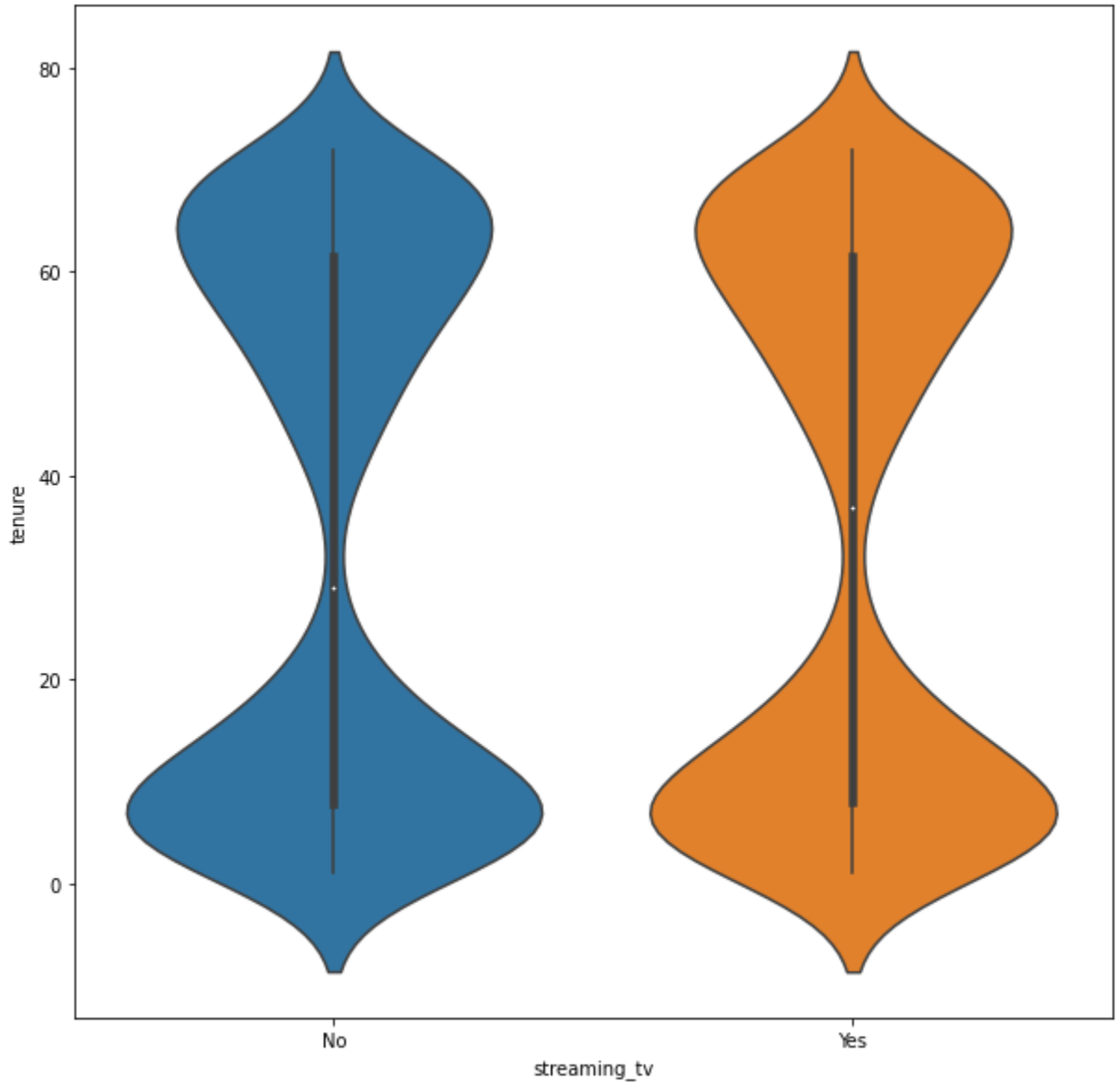
In [29]:
```
sns.violinplot(x='tech_support', y='tenure',data=prep_df);
```



In [30]:
```
prep_df['streaming_tv'].describe()
```

Out[30]:
```
count      10000
unique         2
top           No
freq        5071
Name: streaming_tv, dtype: object
```

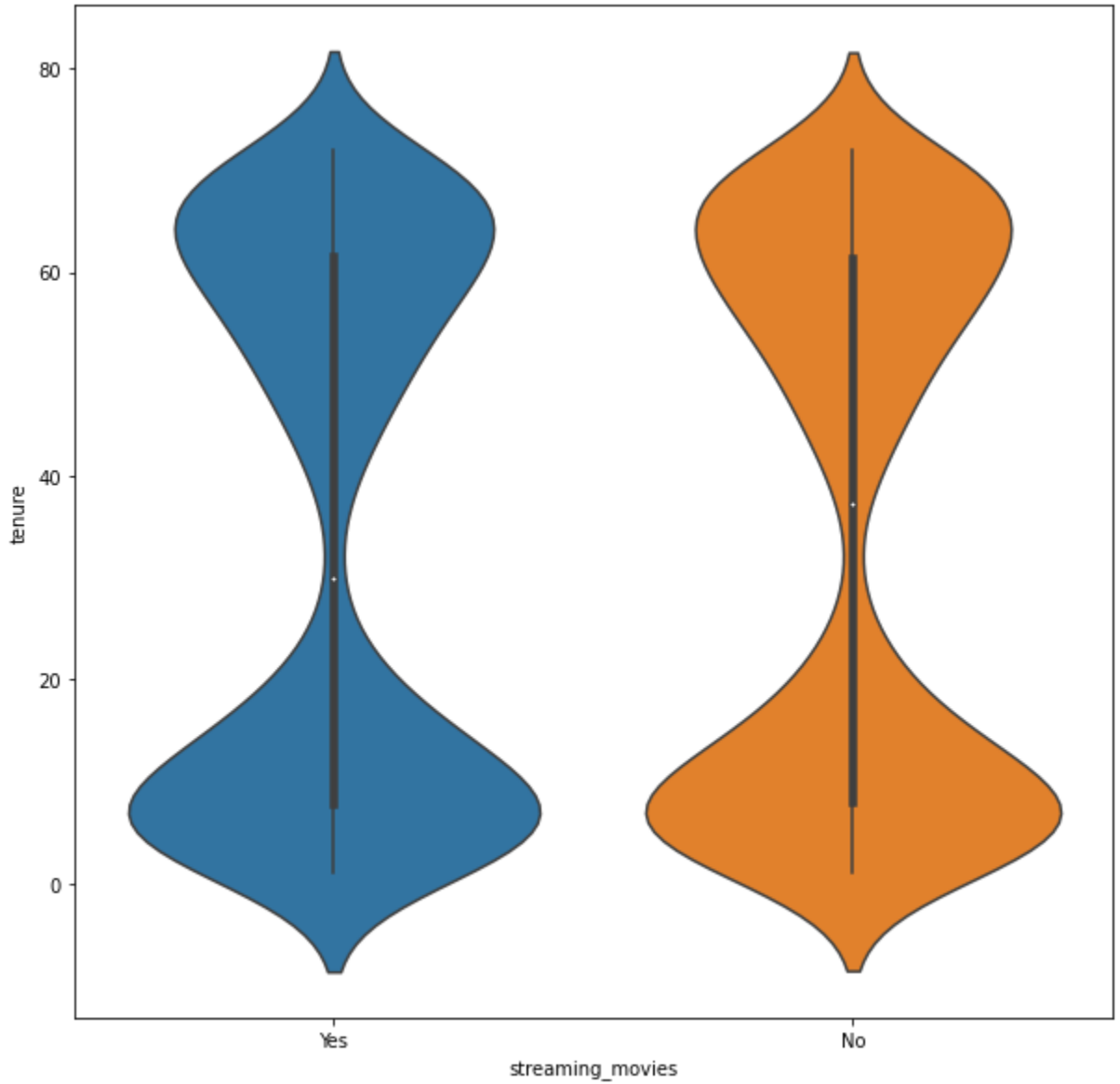
In [31]:
```
sns.violinplot(x='streaming_tv', y='tenure', data=prep_df);
```



In [32]:
```
prep_df['streaming_movies'].describe()
```

Out[32]:
```
count      10000
unique         2
top           No
freq        5110
Name: streaming_movies, dtype: object
```

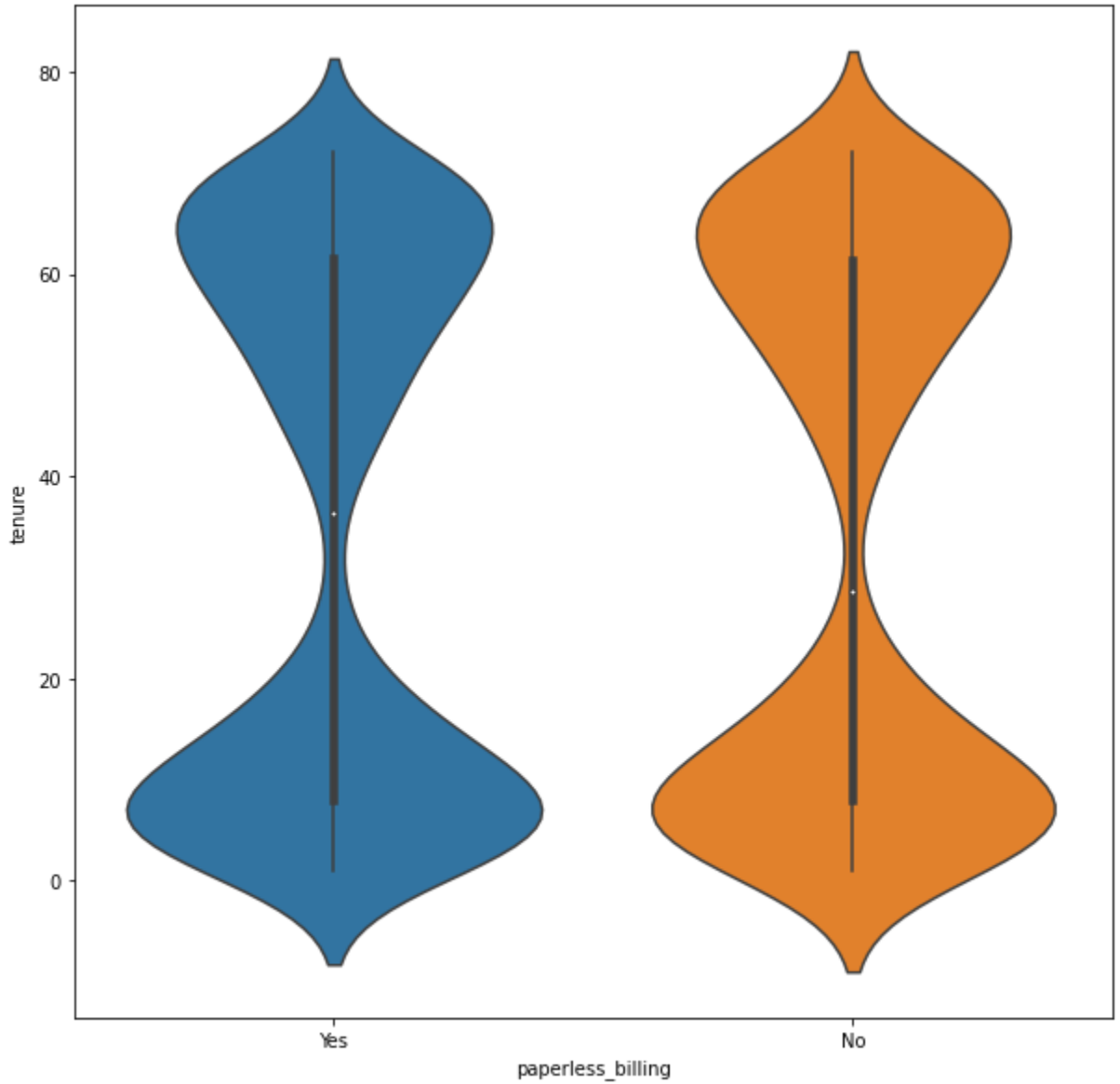
In [33]:
```
sns.violinplot(x='streaming_movies', y='tenure', data=prep_df);
```



In [34]:
```
prep_df['paperless_billing'].describe()
```

Out[34]:
```
count      10000
unique         2
top          Yes
freq        5882
Name: paperless_billing, dtype: object
```

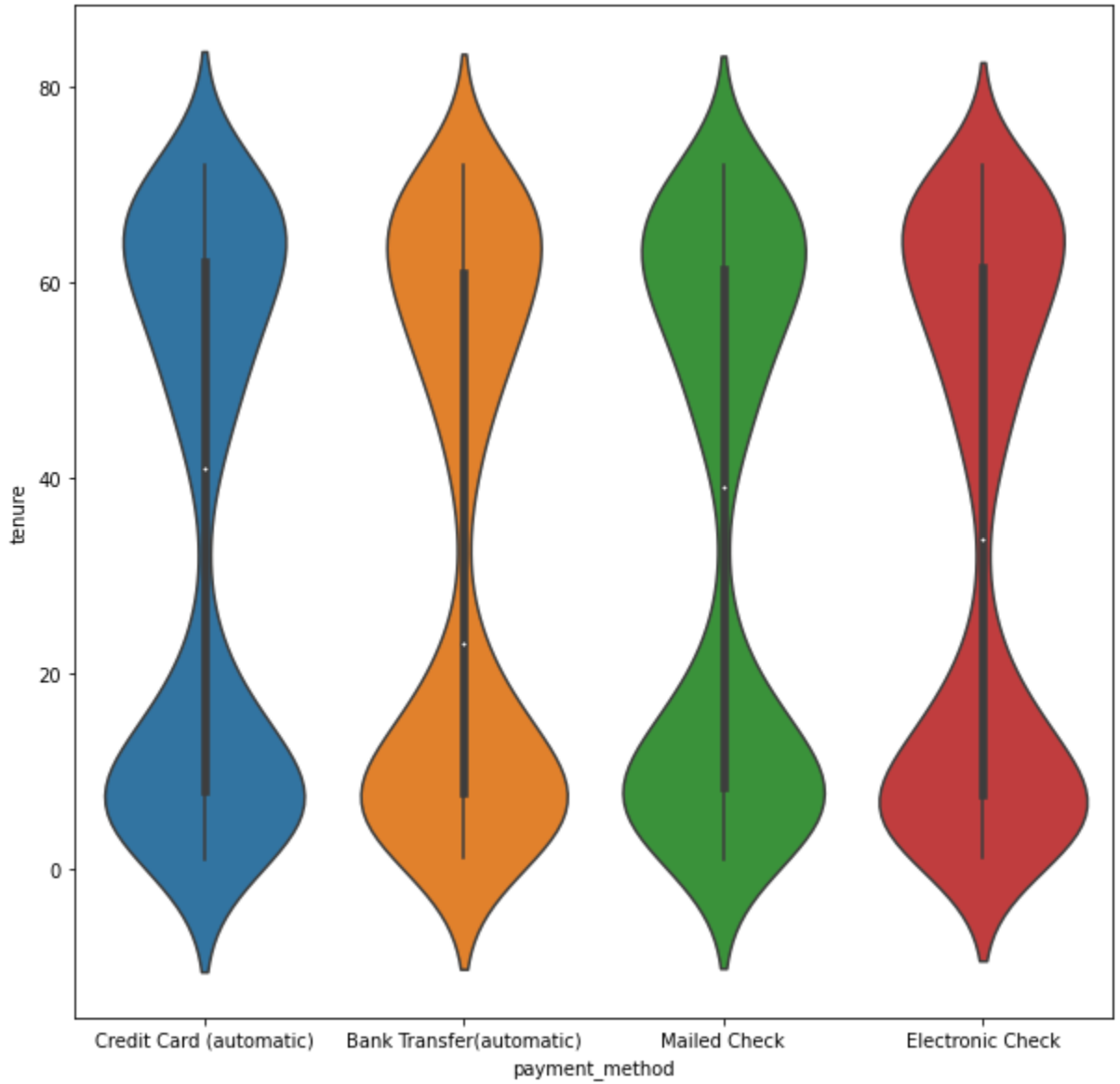
In [35]:
```
sns.violinplot(x='paperless_billing', y='tenure', data=prep_df);
```



In [36]:
```
prep_df['payment_method'].describe()
```

Out[36]:
```
count                10000
unique                   4
top       Electronic Check
freq                  3398
Name: payment_method, dtype: object
```

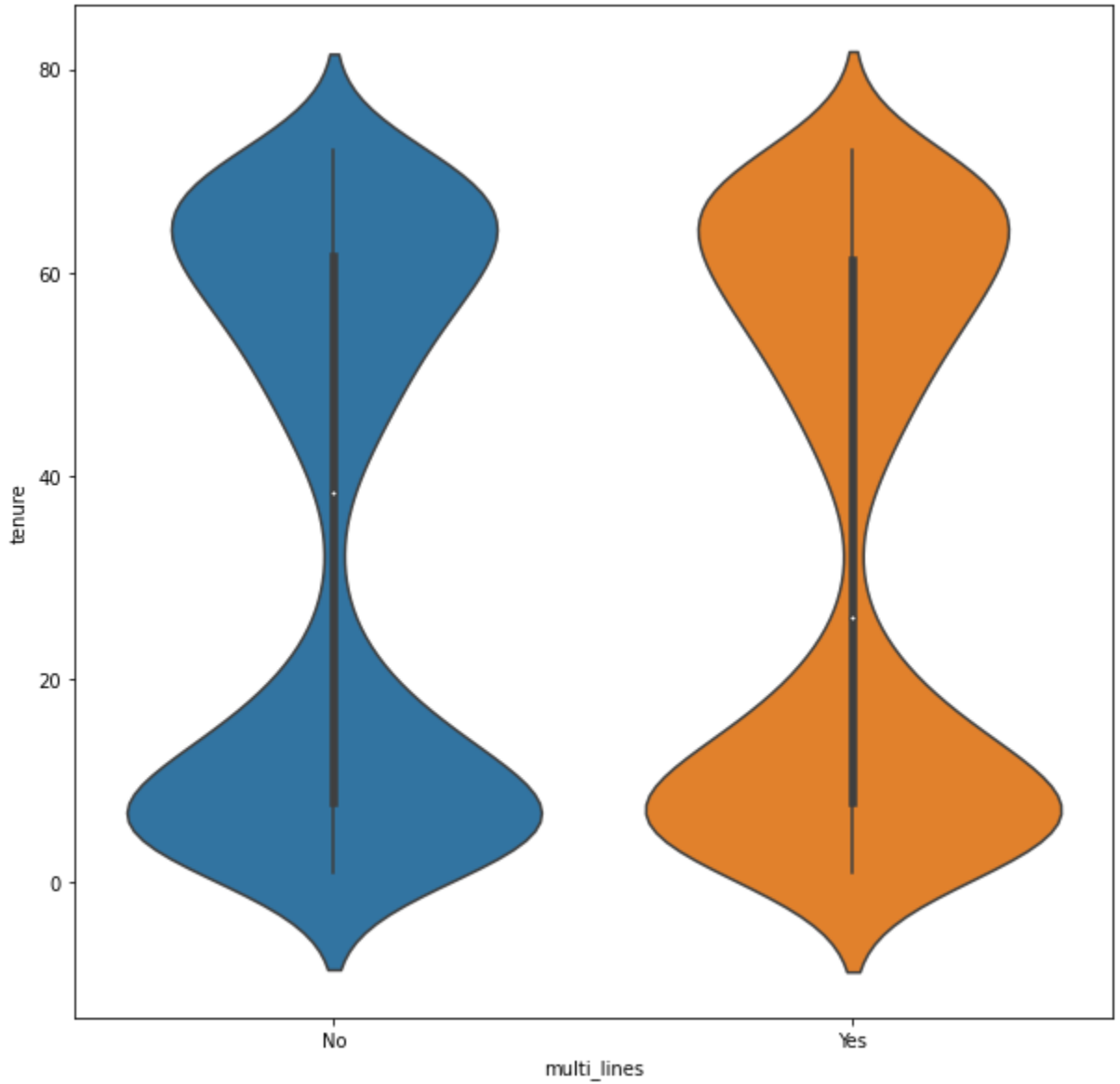
In [37]:
```
sns.violinplot(x='payment_method', y='tenure', data=prep_df);
```



In [38]:
```
prep_df['multi_lines'].describe()
```

Out[38]:
```
count      10000
unique         2
top           No
freq        5392
Name: multi_lines, dtype: object
```

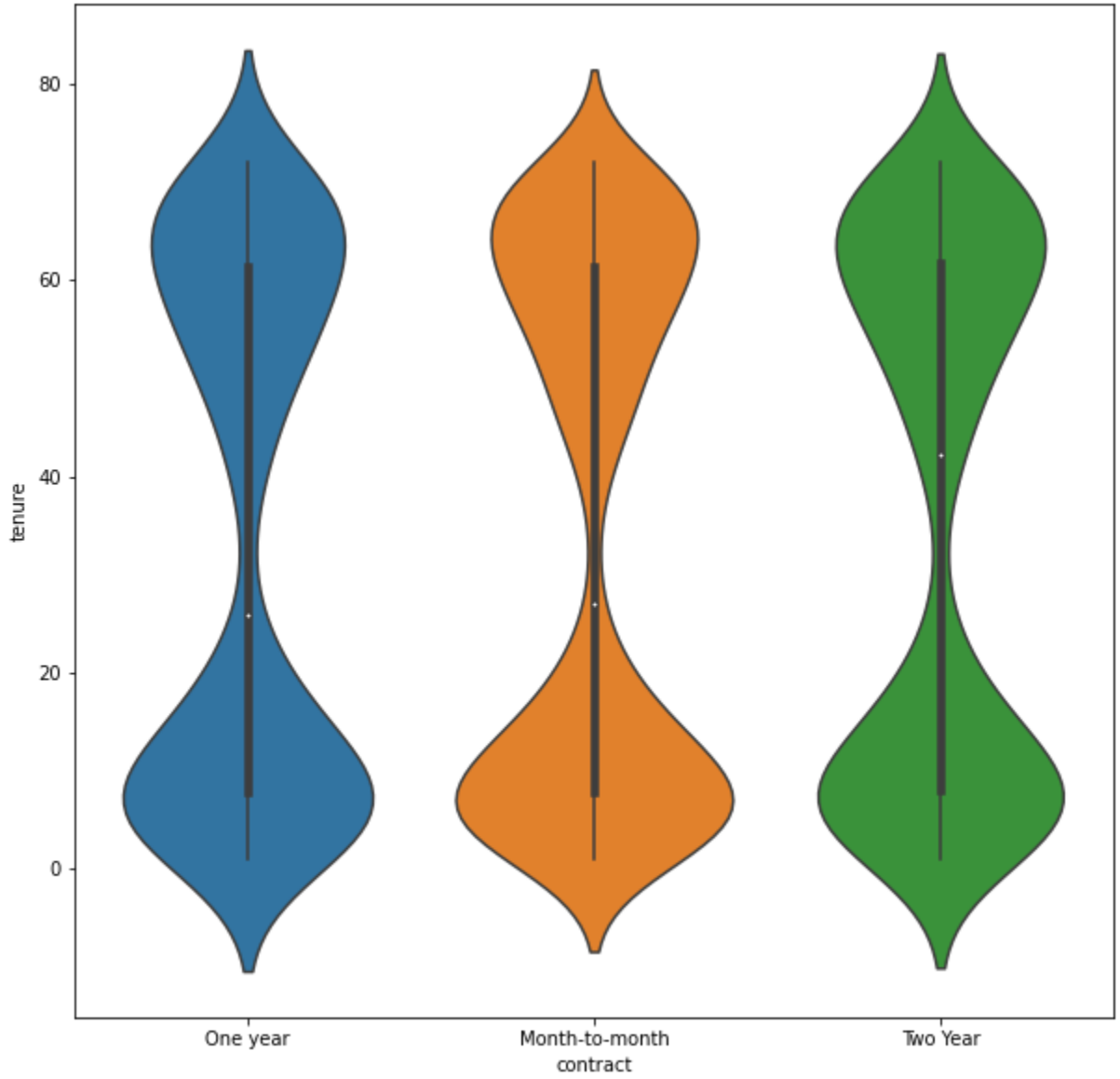
In [39]:
```
sns.violinplot(x='multi_lines', y='tenure', data=prep_df);
```



In [40]:
```
prep_df['contract'].describe()
```

Out[40]:
```
count              10000
unique                 3
top       Month-to-month
freq                5456
Name: contract, dtype: object
```

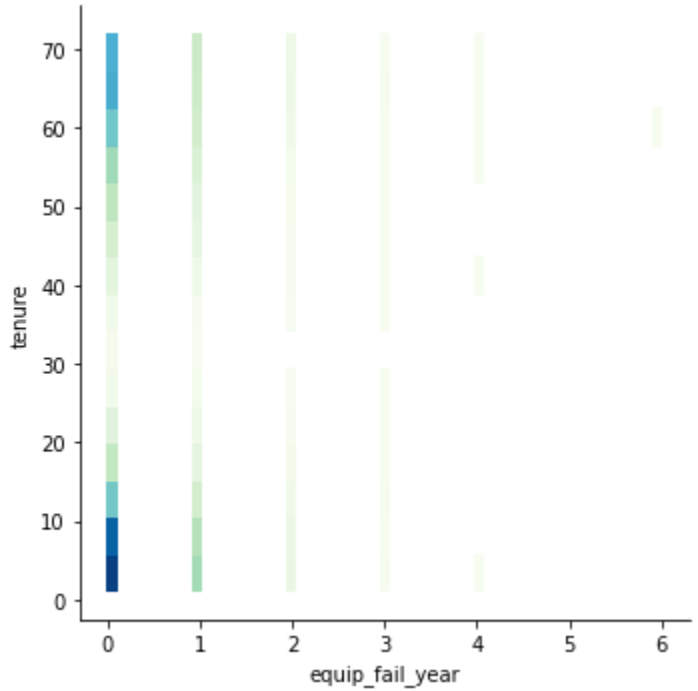
In [41]:
```
sns.violinplot(x='contract', y='tenure', data=prep_df);
```



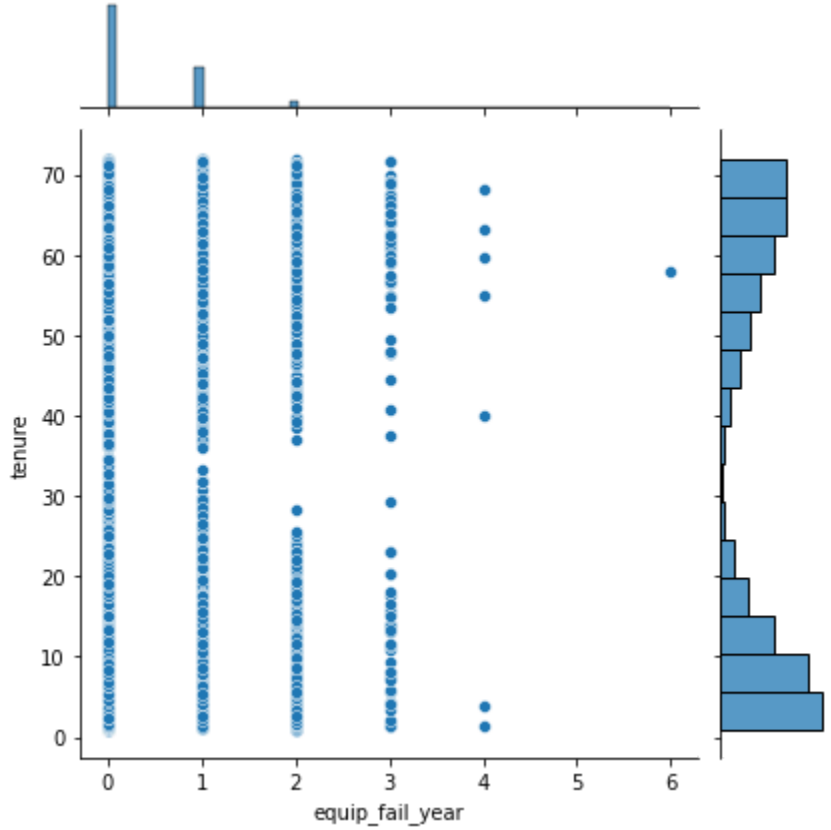
In [42]:
```
prep_df['equip_fail_year'].describe()
```

Out[42]:
```
count    10000.000000
mean         0.398000
std          0.635953
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          6.000000
Name: equip_fail_year, dtype: float64
```

In [43]:
```
sns.displot(x='equip_fail_year', y='tenure', cmap='GnBu', data=prep_df);
```



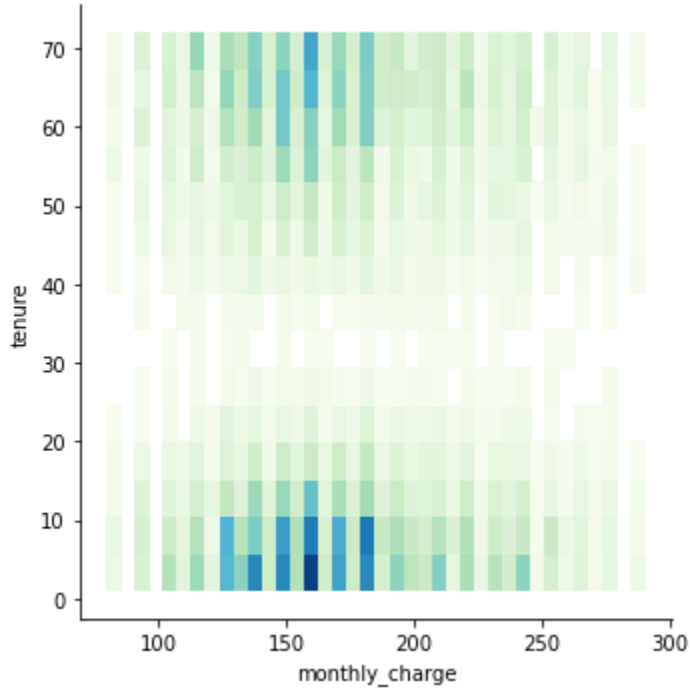
In [44]:
```
sns.jointplot(x='equip_fail_year', y='tenure', data=prep_df);
```



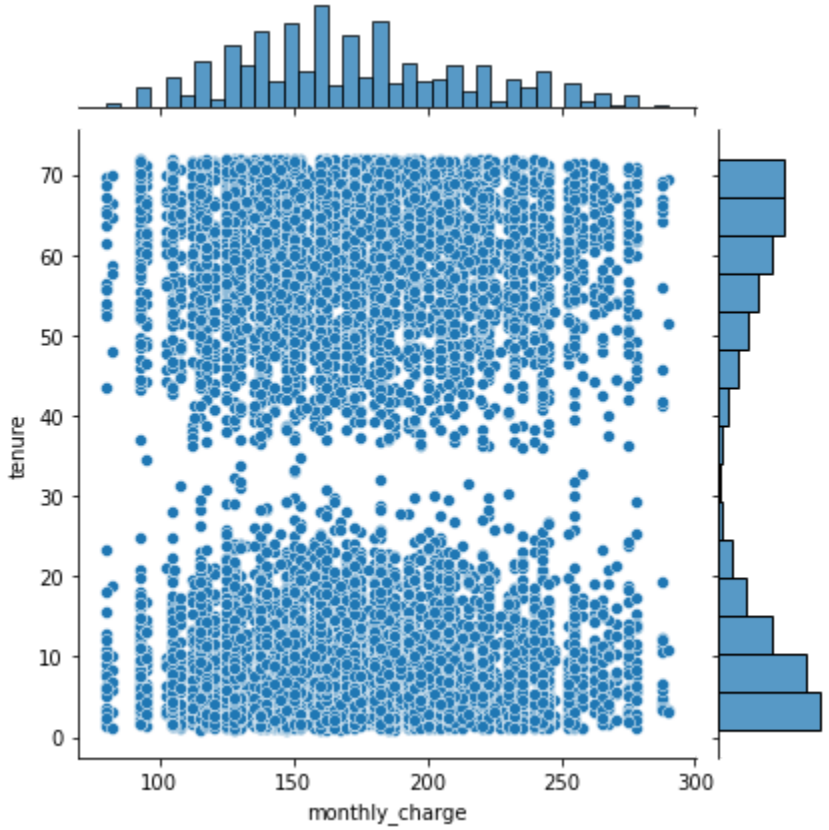
In [45]:
```
prep_df['monthly_charge'].describe()
```

Out[45]:
```
count    10000.000000
mean       172.624816
std         42.943094
min         79.978860
25%        139.979239
50%        167.484700
75%        200.734725
max        290.160419
Name: monthly_charge, dtype: float64
```

In [46]:
```
sns.displot(x='monthly_charge',y='tenure',cmap='GnBu', data=prep_df);
```



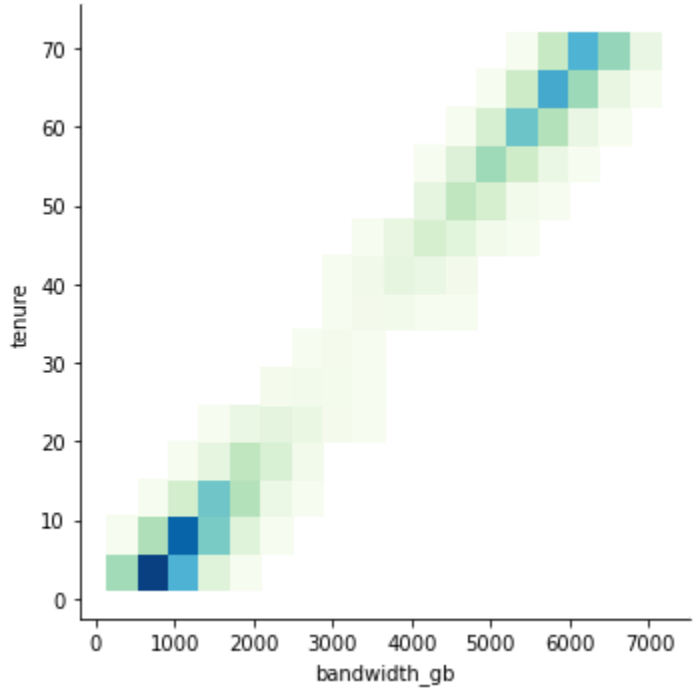
In [47]:
```
sns.jointplot(x='monthly_charge', y='tenure', data=prep_df);
```



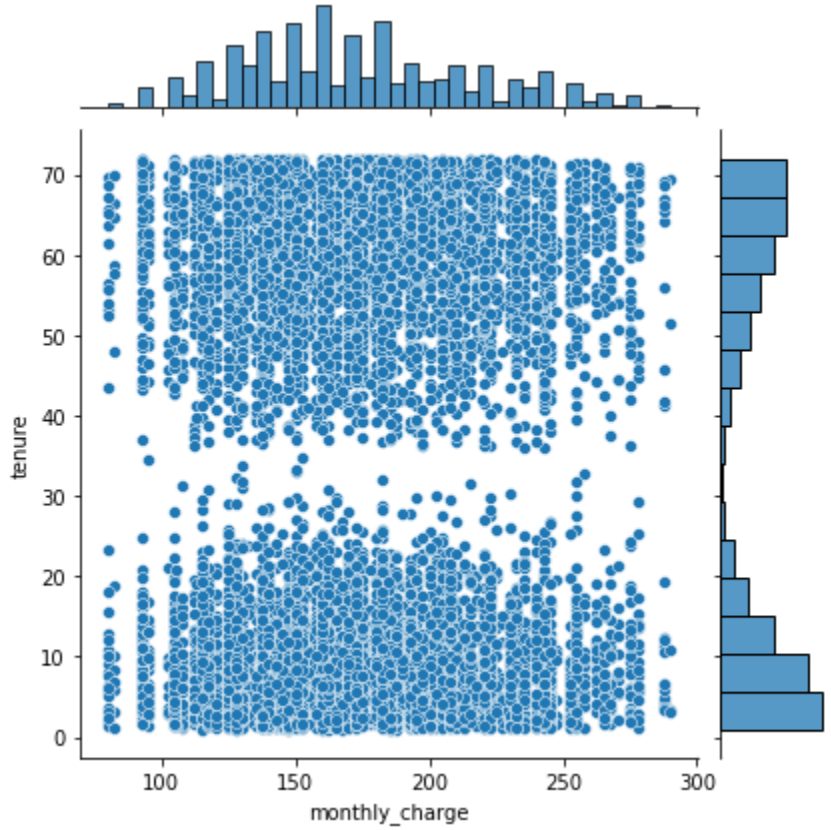
In [48]:
```
prep_df['bandwidth_gb'].describe()
```

Out[48]:
```
count    10000.000000
mean      3392.341550
std       2185.294852
min        155.506715
25%       1236.470827
50%       3279.536903
75%       5586.141370
max       7158.981530
Name: bandwidth_gb, dtype: float64
```

In [49]:
```
sns.displot(x='bandwidth_gb', y='tenure', cmap='GnBu', data=prep_df);
```



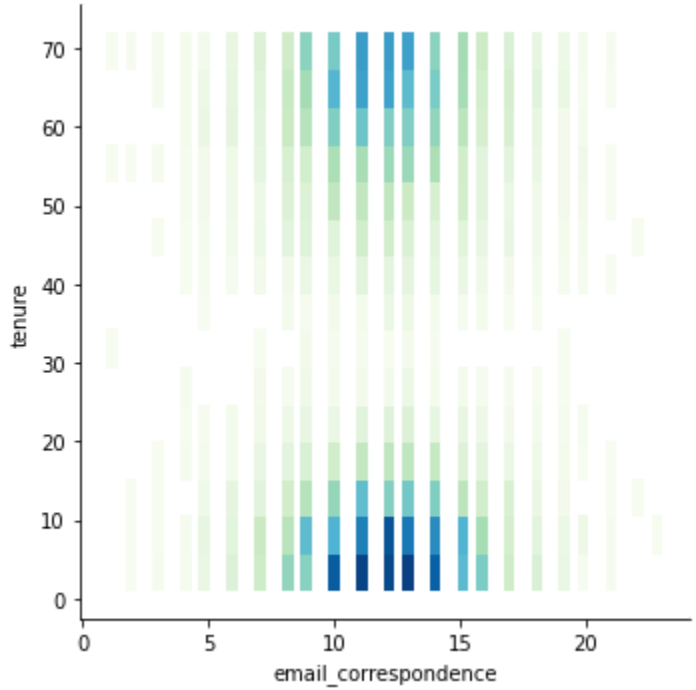
In [50]:
```
sns.jointplot(x='monthly_charge',y='tenure', data=prep_df);
```



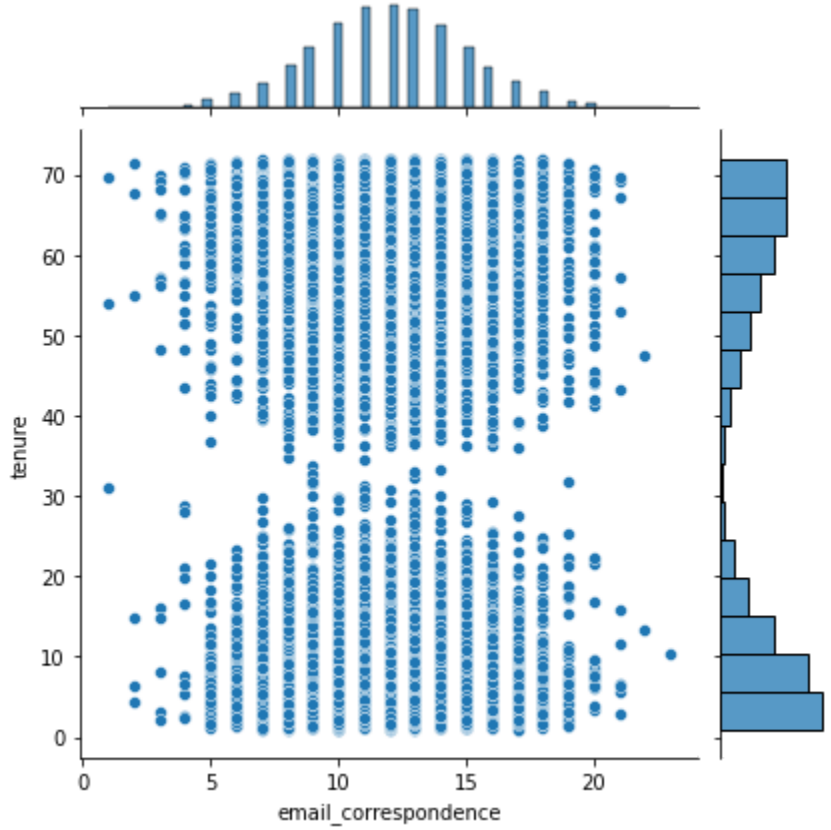
In [51]:
```
prep_df['email_correspondence'].describe()
```

Out[51]:
```
count    10000.000000
mean        12.016000
std          3.025898
min          1.000000
25%         10.000000
50%         12.000000
75%         14.000000
max         23.000000
Name: email_correspondence, dtype: float64
```

In [52]:
```
sns.displot(x='email_correspondence', y='tenure', cmap='GnBu', data=prep_df);
```



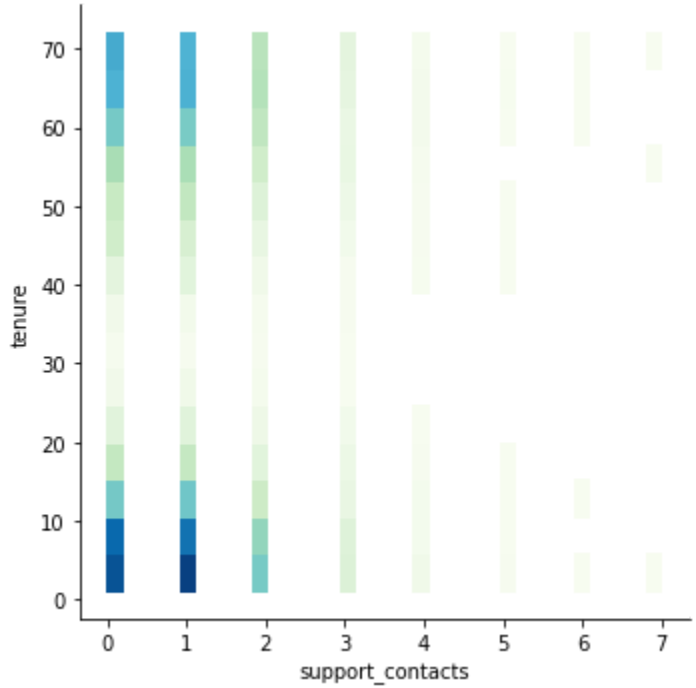
In [53]:
```
sns.jointplot(x='email_correspondence', y='tenure', data=prep_df);
```



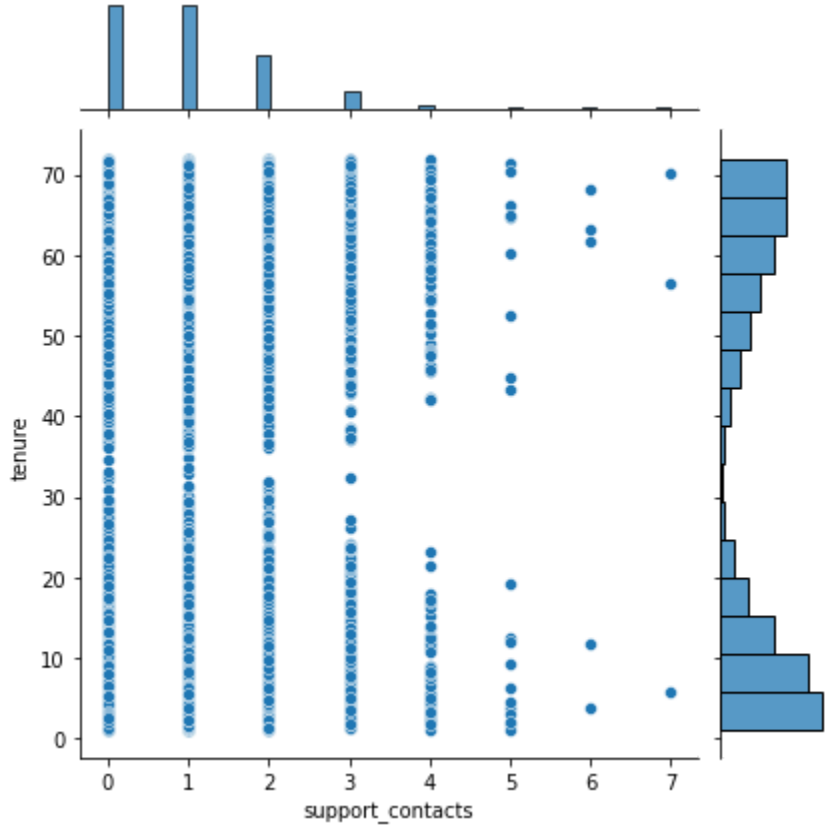
In [54]:
```
prep_df['support_contacts'].describe()
```

Out[54]:
```
count    10000.000000
mean         0.994200
std          0.988466
min          0.000000
25%          0.000000
50%          1.000000
75%          2.000000
max          7.000000
Name: support_contacts, dtype: float64
```

In [55]:
```
sns.displot(x='support_contacts', y='tenure', cmap='GnBu', data=prep_df);
```



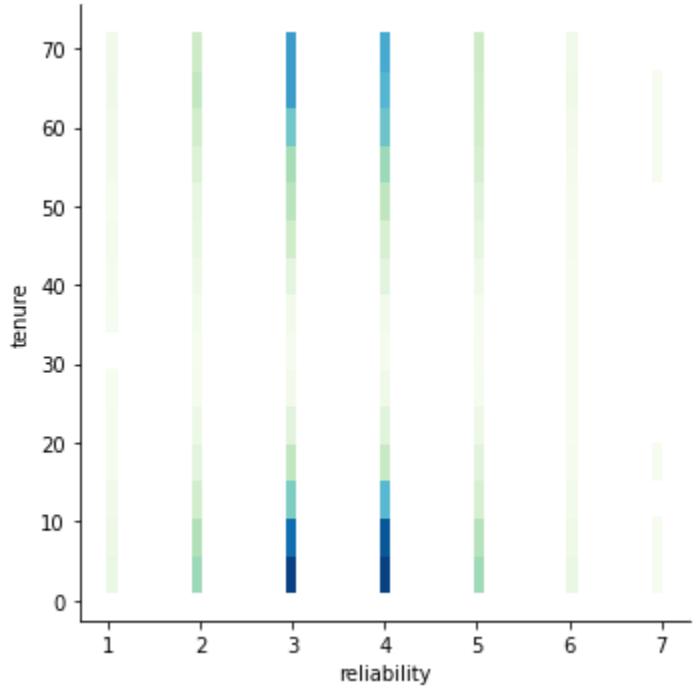
In [56]:
```
sns.jointplot(x='support_contacts', y='tenure', data=prep_df);
```



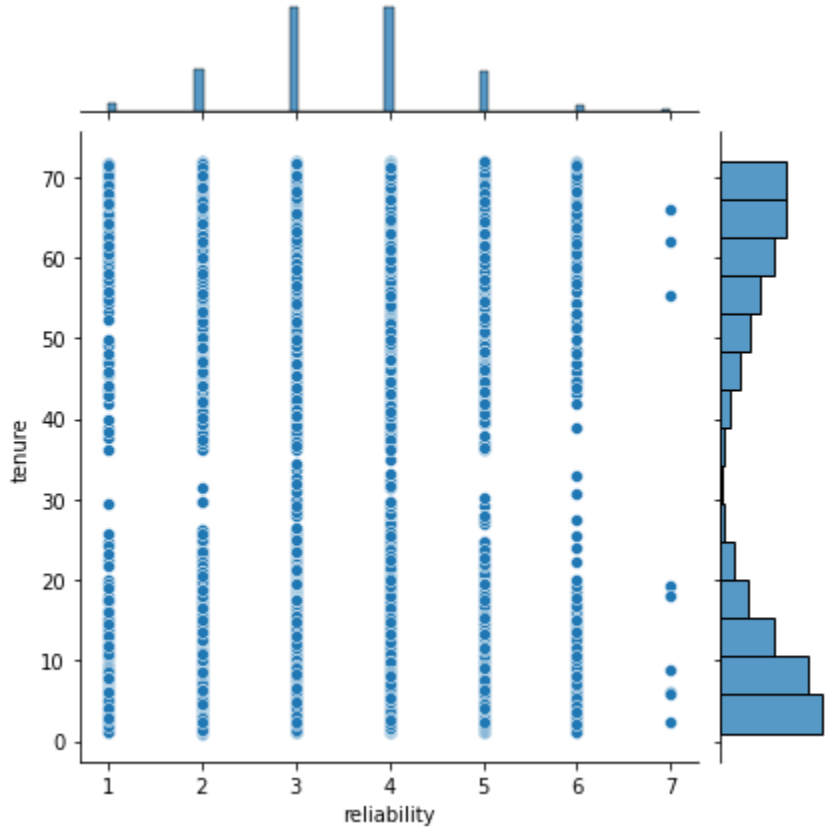
In [57]:
```
prep_df['reliability'].describe()
```

Out[57]:
```
count    10000.000000
mean         3.497500
std          1.025816
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          7.000000
Name: reliability, dtype: float64
```

In [58]:
```
sns.displot(x='reliability', y='tenure', cmap='GnBu', data=prep_df);
```



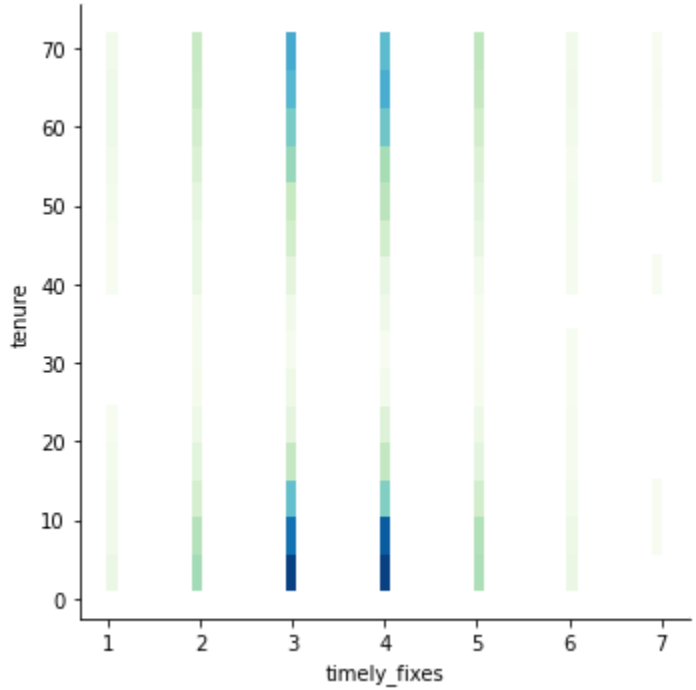
In [59]:
```
sns.jointplot(x='reliability', y='tenure', data=prep_df);
```



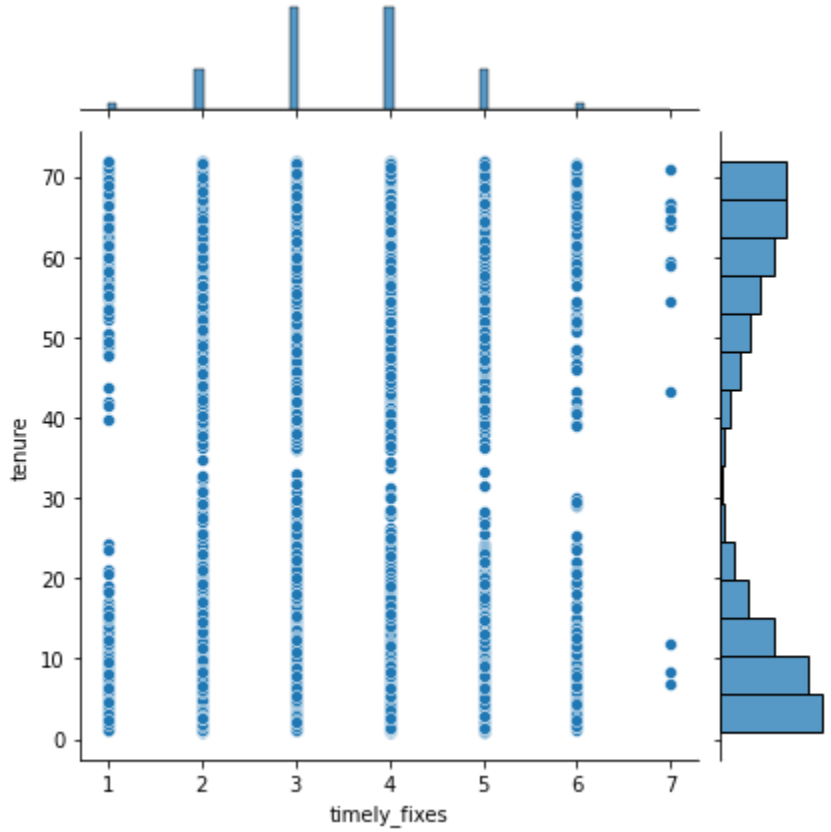
In [60]:
```
prep_df['timely_fixes'].describe()
```

Out[60]:
```
count    10000.000000
mean         3.505100
std          1.034641
min          1.000000
25%          3.000000
50%          4.000000
75%          4.000000
max          7.000000
Name: timely_fixes, dtype: float64
```

In [61]:
```
sns.displot(x='timely_fixes', y='tenure', cmap='GnBu', data=prep_df);
```



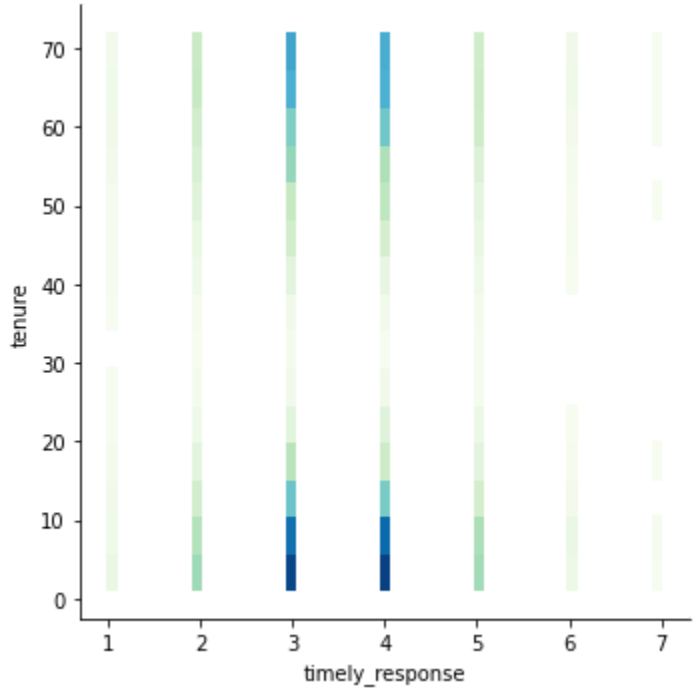
In [62]:
```
sns.jointplot(x='timely_fixes', y='tenure', data=prep_df);
```



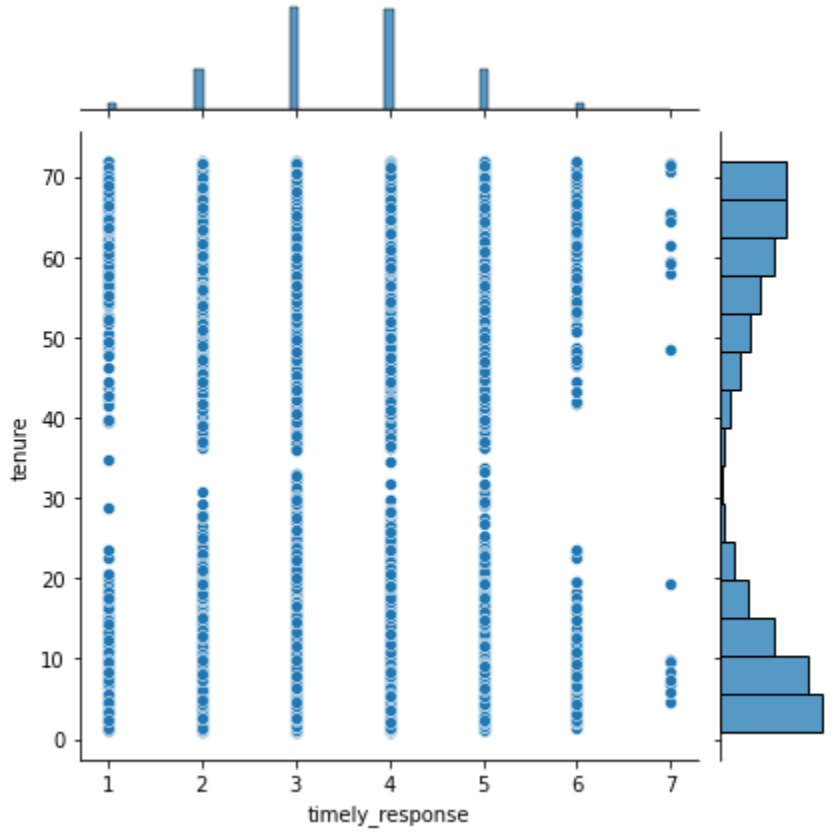
In [63]:
```
prep_df['timely_response'].describe()
```

Out[63]:
```
count    10000.000000
mean         3.490800
std          1.037797
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          7.000000
Name: timely_response, dtype: float64
```

In [64]:
```
sns.displot(x='timely_response', y='tenure', cmap='GnBu', data=prep_df);
```



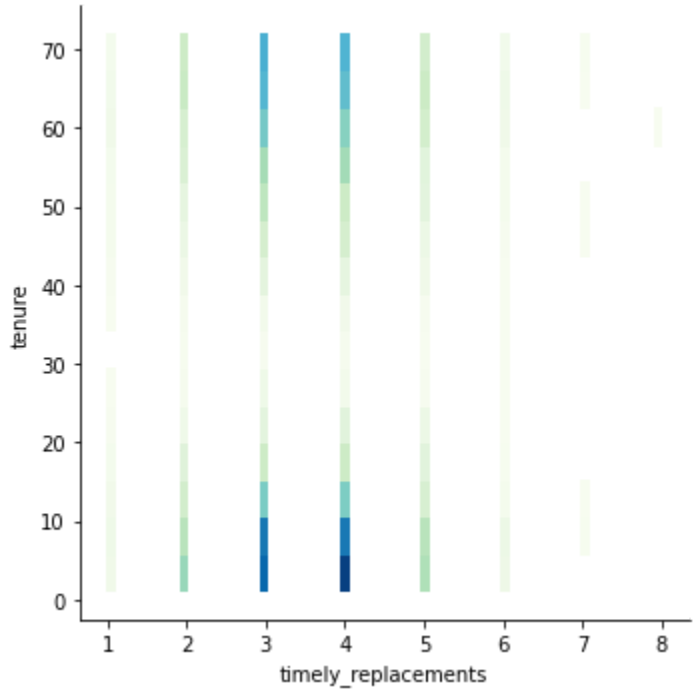
In [65]:
```
sns.jointplot(x='timely_response', y='tenure', data=prep_df);
```



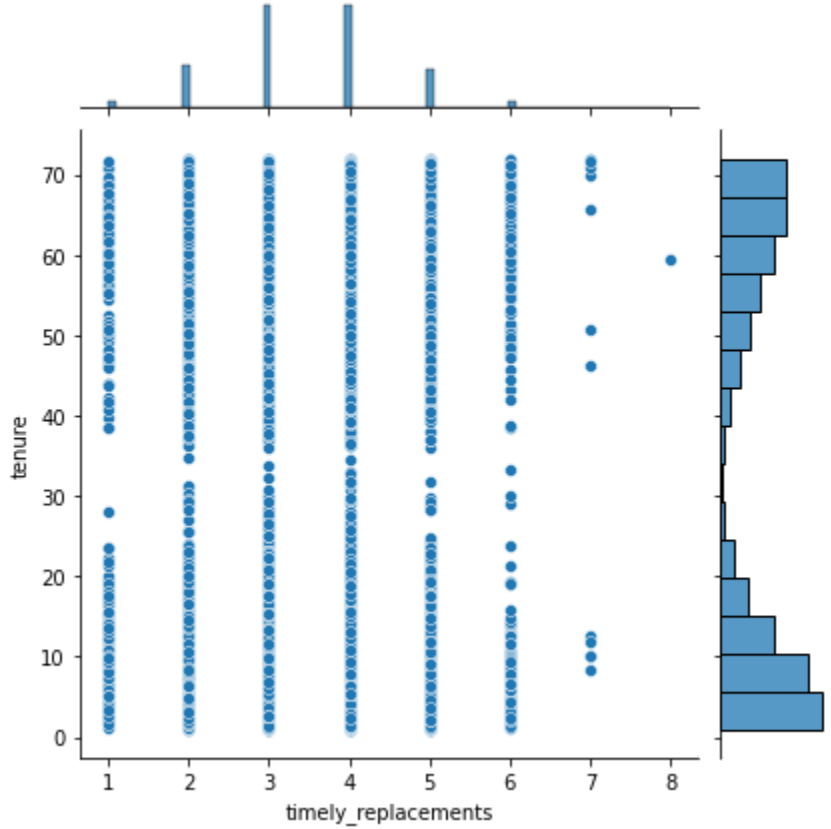
In [66]:
```
prep_df['timely_replacements'].describe()
```

Out[66]:
```
count    10000.000000
mean         3.487000
std          1.027977
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          8.000000
Name: timely_replacements, dtype: float64
```

In [67]:
```
sns.displot(x='timely_replacements', y='tenure', cmap='GnBu', data=prep_df);
```



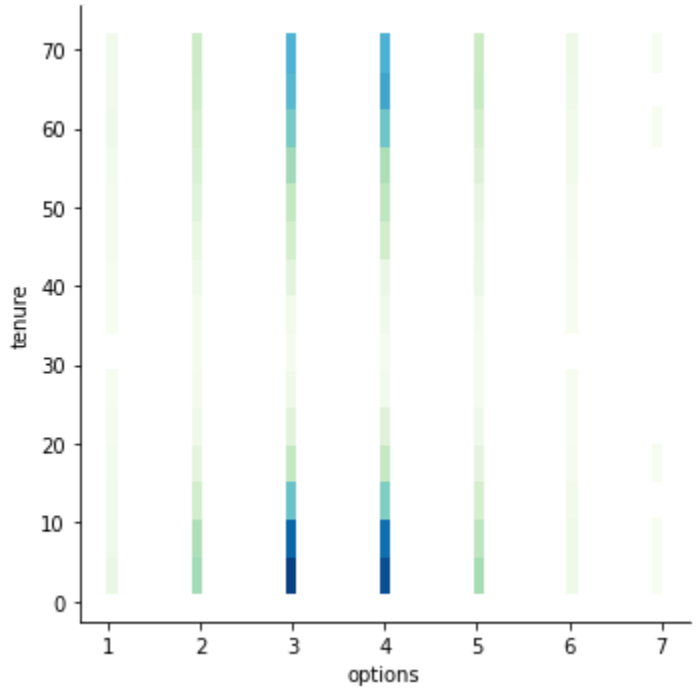
In [68]:
```
sns.jointplot(x='timely_replacements', y='tenure', data=prep_df);
```



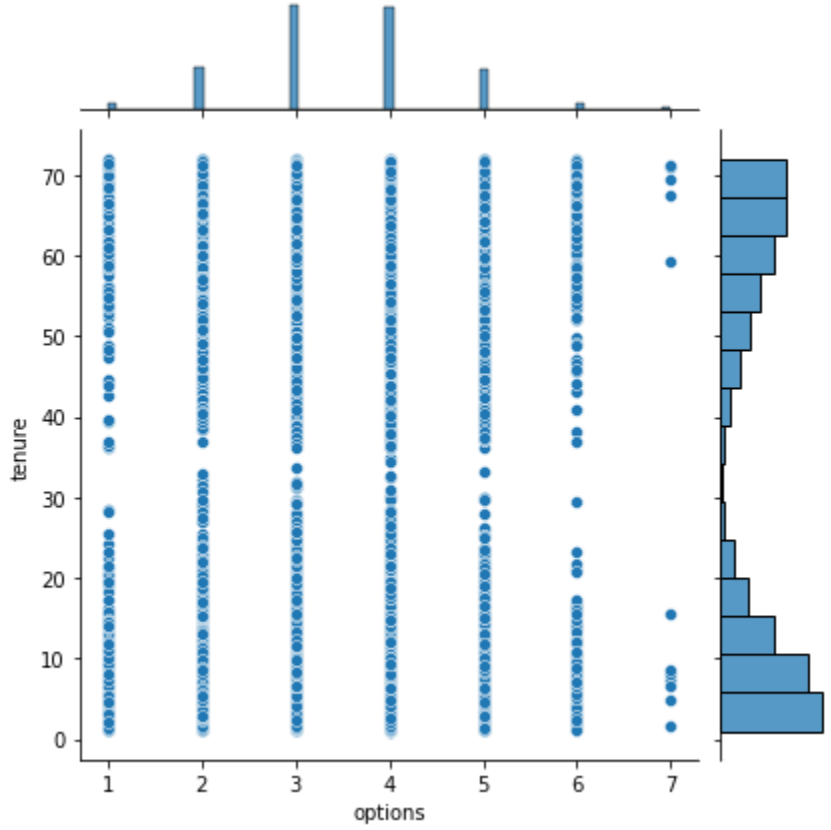
In [69]:
```
prep_df['options'].describe()
```

Out[69]:
```
count    10000.000000
mean         3.492900
std          1.024819
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          7.000000
Name: options, dtype: float64
```

In [70]:
```
sns.displot(x='options', y='tenure', cmap='GnBu', data=prep_df);
```



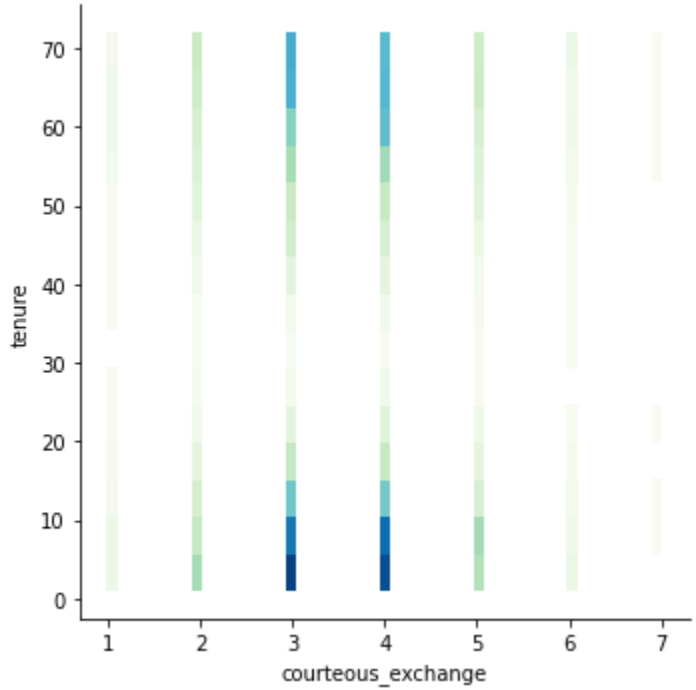
In [71]:
```
sns.jointplot(x='options', y='tenure', data=prep_df);
```



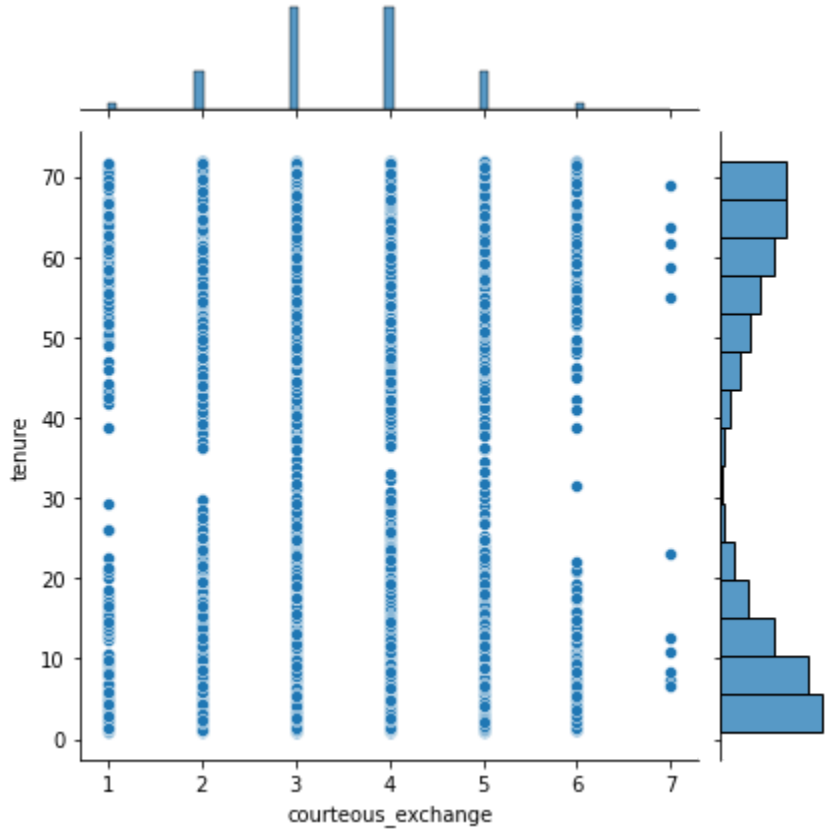
In [72]:
```
prep_df['courteous_exchange'].describe()
```

Out[72]:
```
count    10000.000000
mean         3.509500
std          1.028502
min          1.000000
25%          3.000000
50%          4.000000
75%          4.000000
max          7.000000
Name: courteous_exchange, dtype: float64
```

In [73]:
```
sns.displot(x='courteous_exchange', y='tenure', cmap='GnBu', data=prep_df);
```



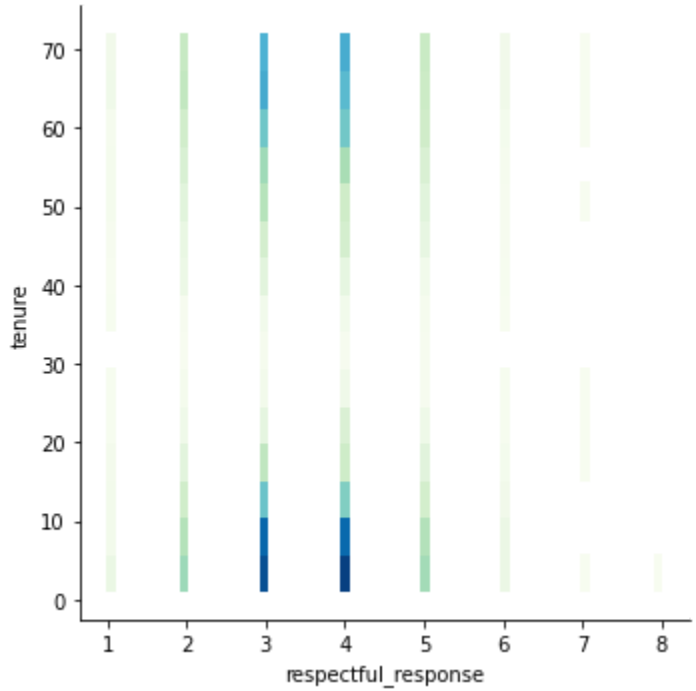
In [74]:
```
sns.jointplot(x='courteous_exchange', y='tenure', data=prep_df);
```



In [75]:
```
prep_df['respectful_response'].describe()
```

Out[75]:
```
count    10000.000000
mean         3.497300
std          1.033586
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          8.000000
Name: respectful_response, dtype: float64
```

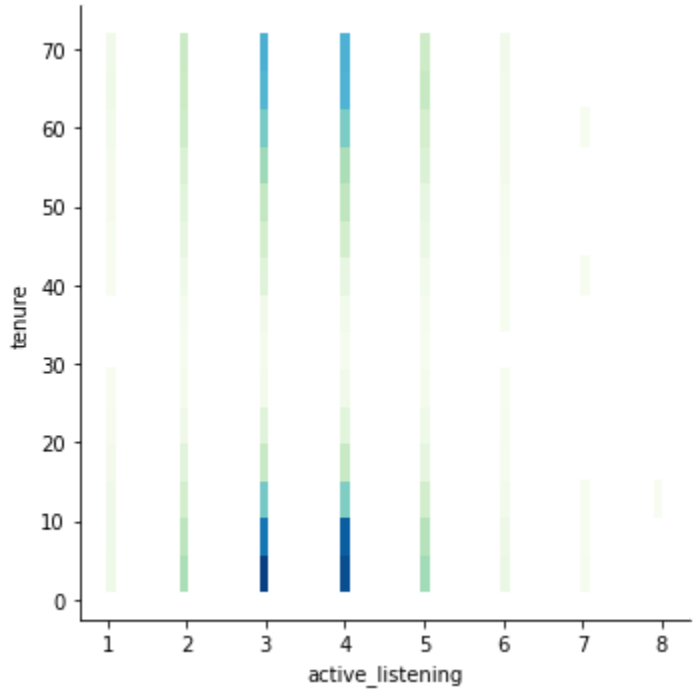
In [76]:
```
sns.displot(x='respectful_response', y='tenure', cmap='GnBu', data=prep_df);
```



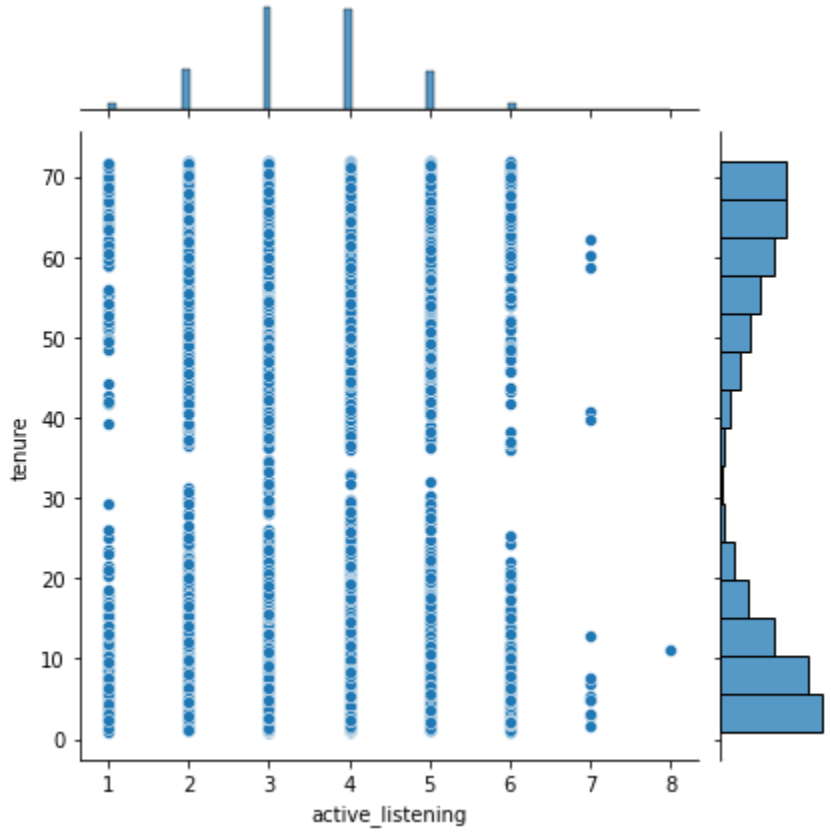
In [77]:
```
prep_df['active_listening'].describe()
```

Out[77]:
```
count    10000.000000
mean         3.495600
std          1.028633
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max          8.000000
Name: active_listening, dtype: float64
```

In [78]:
```
sns.displot(x='active_listening', y='tenure', cmap='GnBu', data=prep_df);
```



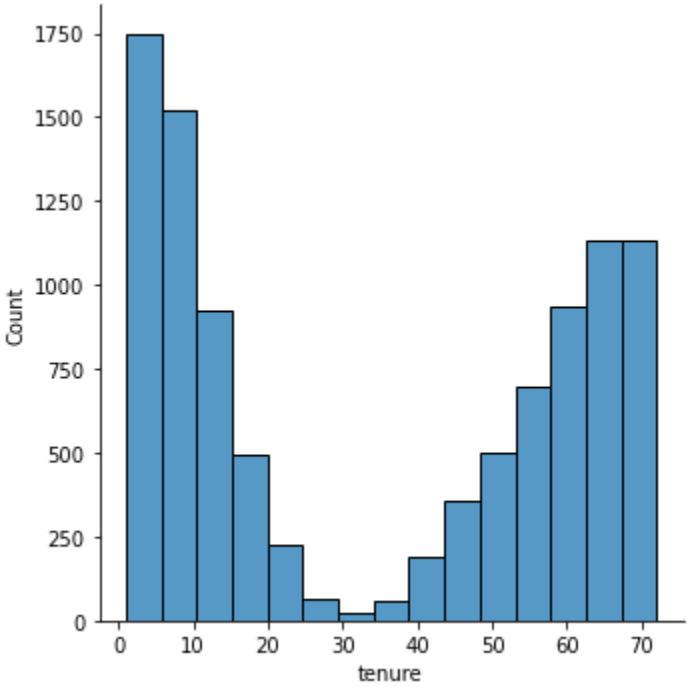
In [79]:
```
sns.jointplot(x='active_listening', y='tenure', data=prep_df);
```



In [80]:
```
prep_df['tenure'].describe()
```

Out[80]:
```
count    10000.000000
mean        34.526188
std         26.443063
min          1.000259
25%          7.917694
50%         35.430507
75%         61.479795
max         71.999280
Name: tenure, dtype: float64
```

In [81]:
```
sns.displot(prep_df.tenure);
```



In [82]:
```
#feature transformation
prep_df['port_numeric'] = prep_df['port_modem']
dict_port_modem = {'port_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_port_modem, inplace=True)
```

In [83]:
```
prep_df['tablet_numeric'] = prep_df['tablet']
dict_tablet = {'tablet_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_tablet, inplace=True)
```

In [84]:
```
prep_df['phone_numeric'] = prep_df['phone_service']
dict_phone = {'phone_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_phone, inplace=True)
```

In [85]:
```
prep_df['multi_numeric'] = prep_df['multi_lines']
dict_multi_lines = {'multi_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_multi_lines, inplace=True)
```

In [86]:
```
prep_df['security_numeric'] = prep_df['online_security']
dict_security = {'security_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_security, inplace=True)
```

In [87]:
```
prep_df['backup_numeric'] = prep_df['online_backup']
dict_backup = {'backup_numeric':{"Yes":1,"No":0}}
```

```
prep_df.replace(dict_backup, inplace=True)
```

In [88]:
```
prep_df['protection_numeric'] = prep_df['device_protection']
dict_protection = {'protection_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_protection, inplace=True)
```

In [89]:
```
prep_df['support_numeric'] = prep_df['tech_support']
dict_support = {'support_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_support, inplace=True)
```

In [90]:
```
prep_df['tv_numeric'] = prep_df['streaming_tv']
dict_tv = {'tv_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_tv, inplace=True)
```

In [91]:
```
prep_df['movies_numeric'] = prep_df['streaming_movies']
dict_movies = {'movies_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_movies, inplace=True)
```

In [92]:
```
prep_df['paperless_numeric'] = prep_df['paperless_billing']
dict_paperless = {'paperless_numeric':{"Yes":1,"No":0}}
prep_df.replace(dict_paperless, inplace=True)
```

In [93]:
```
binary_cat_columns = prep_df[["port_numeric","tablet_numeric","phone_numeric",
                              "multi_numeric","security_numeric","backup_numeric","protection_numeric",
                              "support_numeric","tv_numeric","movies_numeric","paperless_numeric"]]
b_cat_var_df = binary_cat_columns.copy()
```

In [94]:
```
b_cat_var_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   port_numeric        10000 non-null  int64
 1   tablet_numeric      10000 non-null  int64
 2   phone_numeric       10000 non-null  int64
 3   multi_numeric       10000 non-null  int64
 4   security_numeric    10000 non-null  int64
 5   backup_numeric      10000 non-null  int64
 6   protection_numeric  10000 non-null  int64
 7   support_numeric     10000 non-null  int64
 8   tv_numeric          10000 non-null  int64
 9   movies_numeric      10000 non-null  int64
 10  paperless_numeric   10000 non-null  int64
dtypes: int64(11)
memory usage: 859.5 KB
```

In [95]:
```
continuous_columns = prep_df[["outage_sec_perweek","email_correspondence",
                              "support_contacts","equip_fail_year","monthly_charge","bandwidth_gb","tenure"]]
cont_num_df = continuous_columns.copy()
```

In [96]:
```
cont_num_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   outage_sec_perweek   10000 non-null  float64
 1   email_correspondence 10000 non-null  int64
 2   support_contacts     10000 non-null  int64
 3   equip_fail_year      10000 non-null  int64
 4   monthly_charge       10000 non-null  float64
 5   bandwidth_gb         10000 non-null  float64
 6   tenure               10000 non-null  float64
dtypes: float64(4), int64(3)
memory usage: 547.0 KB
```

In [97]:
```
categorical_columns = prep_df[["contract","internet_service","payment_method"]]
cat_var_df = categorical_columns.copy()
```

In [98]:
```
cat_var_df = pd.get_dummies(cat_var_df, columns=['contract','internet_service',
                                                 'payment_method'], drop_first=True)
cat_var_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   contract_One year                       10000 non-null  uint8
 1   contract_Two Year                       10000 non-null  uint8
 2   internet_service_Fiber Optic            10000 non-null  uint8
 3   internet_service_None                   10000 non-null  uint8
 4   payment_method_Credit Card (automatic)  10000 non-null  uint8
 5   payment_method_Electronic Check         10000 non-null  uint8
 6   payment_method_Mailed Check             10000 non-null  uint8
dtypes: uint8(7)
memory usage: 68.5 KB
```

In [99]:
```
cat_var_df = cat_var_df.rename(columns={'contract_One year':'contract_One_Year',
                                'contract_Two Year':'contract_Two_Year',
                                'internet_service_Fiber Optic': 'internet_service_Fiber_Optic',
                                'payment_method_Credit Card (automatic)': 'payment_method_Credit_Card_auto',
                                'payment_method_Electronic Check':'payment_method_Electronic_Check',
                                'payment_method_Mailed Check':'payment_method_Mailed_Check'})
cat_var_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   contract_One_Year               10000 non-null  uint8
 1   contract_Two_Year               10000 non-null  uint8
 2   internet_service_Fiber_Optic    10000 non-null  uint8
 3   internet_service_None           10000 non-null  uint8
 4   payment_method_Credit_Card_auto 10000 non-null  uint8
 5   payment_method_Electronic_Check 10000 non-null  uint8
 6   payment_method_Mailed_Check     10000 non-null  uint8
dtypes: uint8(7)
memory usage: 68.5 KB
```

In [100…
```
new_prep_df = pd.concat([b_cat_var_df, cat_var_df], axis=1, ignore_index=False)
new_prep_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   port_numeric                     10000 non-null  int64
 1   tablet_numeric                   10000 non-null  int64
 2   phone_numeric                    10000 non-null  int64
 3   multi_numeric                    10000 non-null  int64
 4   security_numeric                 10000 non-null  int64
 5   backup_numeric                   10000 non-null  int64
 6   protection_numeric               10000 non-null  int64
 7   support_numeric                  10000 non-null  int64
 8   tv_numeric                       10000 non-null  int64
 9   movies_numeric                   10000 non-null  int64
 10  paperless_numeric                10000 non-null  int64
 11  contract_One_Year                10000 non-null  uint8
 12  contract_Two_Year                10000 non-null  uint8
 13  internet_service_Fiber_Optic     10000 non-null  uint8
 14  internet_service_None            10000 non-null  uint8
 15  payment_method_Credit_Card_auto  10000 non-null  uint8
 16  payment_method_Electronic_Check  10000 non-null  uint8
 17  payment_method_Mailed_Check      10000 non-null  uint8
dtypes: int64(11), uint8(7)
memory usage: 927.9 KB
```

In [101…

```
new_prep_df = pd.concat([new_prep_df, cont_num_df], axis=1, ignore_index=False)
new_prep_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 25 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   port_numeric                     10000 non-null  int64  
 1   tablet_numeric                   10000 non-null  int64  
 2   phone_numeric                    10000 non-null  int64  
 3   multi_numeric                    10000 non-null  int64  
 4   security_numeric                 10000 non-null  int64  
 5   backup_numeric                   10000 non-null  int64  
 6   protection_numeric               10000 non-null  int64  
 7   support_numeric                  10000 non-null  int64  
 8   tv_numeric                       10000 non-null  int64  
 9   movies_numeric                   10000 non-null  int64  
 10  paperless_numeric                10000 non-null  int64  
 11  contract_One_Year                10000 non-null  uint8  
 12  contract_Two_Year                10000 non-null  uint8  
 13  internet_service_Fiber_Optic     10000 non-null  uint8  
 14  internet_service_None            10000 non-null  uint8  
 15  payment_method_Credit_Card_auto  10000 non-null  uint8  
 16  payment_method_Electronic_Check  10000 non-null  uint8  
 17  payment_method_Mailed_Check      10000 non-null  uint8  
 18  outage_sec_perweek               10000 non-null  float64
 19  email_correspondence             10000 non-null  int64  
 20  support_contacts                 10000 non-null  int64  
 21  equip_fail_year                  10000 non-null  int64  
 22  monthly_charge                   10000 non-null  float64
 23  bandwidth_gb                     10000 non-null  float64
 24  tenure                           10000 non-null  float64
dtypes: float64(4), int64(14), uint8(7)
memory usage: 1.4 MB
```

In [102…

```
data = new_prep_df.drop(columns=['tenure'], axis=1)
target = new_prep_df[['tenure']]
mod_features = SelectKBest(score_func=f_regression, k=15)
x = data
y = target
y = np.ravel(y)
selection = mod_features.fit_transform(x,y)

print("After selecting best 15 features:", selection.shape)
```

```
After selecting best 15 features: (10000, 15)
```

In [103…

```
filter = mod_features.get_support()
features = data.columns
```

```
print("All features:")
print(features)
print('-'*100)
print("Selected best 15:")
print(features[filter])
```

```
All features:
Index(['port_numeric', 'tablet_numeric', 'phone_numeric', 'multi_numeric',
       'security_numeric', 'backup_numeric', 'protection_numeric',
       'support_numeric', 'tv_numeric', 'movies_numeric', 'paperless_numeric',
       'contract_One_Year', 'contract_Two_Year',
       'internet_service_Fiber_Optic', 'internet_service_None',
       'payment_method_Credit_Card_auto', 'payment_method_Electronic_Check',
       'payment_method_Mailed_Check', 'outage_sec_perweek',
       'email_correspondence', 'support_contacts', 'equip_fail_year',
       'monthly_charge', 'bandwidth_gb'],
      dtype='object')
----------------------------------------------------------------------------------------------------
Selected best 15:
Index(['port_numeric', 'phone_numeric', 'multi_numeric', 'backup_numeric',
       'protection_numeric', 'contract_One_Year', 'contract_Two_Year',
       'internet_service_Fiber_Optic', 'internet_service_None',
       'payment_method_Credit_Card_auto', 'payment_method_Mailed_Check',
       'email_correspondence', 'equip_fail_year', 'monthly_charge',
       'bandwidth_gb'],
      dtype='object')
```

In [104…

```
prepped_df = new_prep_df.drop(['tablet_numeric','paperless_numeric','security_numeric','support_numeric',
                 'tv_numeric','movies_numeric','payment_method_Electronic_Check',
                 'outage_sec_perweek','email_correspondence',
                 'support_contacts'],axis=1)
prepped_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   port_numeric                     10000 non-null  int64  
 1   phone_numeric                    10000 non-null  int64  
 2   multi_numeric                    10000 non-null  int64  
 3   backup_numeric                   10000 non-null  int64  
 4   protection_numeric               10000 non-null  int64  
 5   contract_One_Year                10000 non-null  uint8  
 6   contract_Two_Year                10000 non-null  uint8  
 7   internet_service_Fiber_Optic     10000 non-null  uint8  
 8   internet_service_None            10000 non-null  uint8  
 9   payment_method_Credit_Card_auto  10000 non-null  uint8  
 10  payment_method_Mailed_Check      10000 non-null  uint8  
 11  equip_fail_year                  10000 non-null  int64  
 12  monthly_charge                   10000 non-null  float64
 13  bandwidth_gb                     10000 non-null  float64
 14  tenure                           10000 non-null  float64
dtypes: float64(3), int64(6), uint8(6)
memory usage: 761.8 KB
```

In [105…

```
forest_df = prepped_df.copy()
forest_df.to_csv('/Users/katherinevoakes/Desktop/wgu/clean_forest_data.csv', index=False)
```

In [106…

```
forest_df = pd.read_csv('/Users/katherinevoakes/Desktop/wgu/clean_forest_data.csv')
forest_df.head().T
```

Out[106]:

	0	1	2	3	4
port_numeric	1.000000	0.000000	1.000000	0.000000	1.000000
phone_numeric	1.000000	1.000000	1.000000	1.000000	0.000000
multi_numeric	0.000000	1.000000	1.000000	0.000000	0.000000
backup_numeric	1.000000	0.000000	0.000000	0.000000	0.000000
protection_numeric	0.000000	0.000000	0.000000	0.000000	0.000000
contract_One_Year	1.000000	0.000000	0.000000	0.000000	0.000000
contract_Two_Year	0.000000	0.000000	1.000000	1.000000	0.000000
internet_service_Fiber_Optic	1.000000	1.000000	0.000000	0.000000	1.000000
internet_service_None	0.000000	0.000000	0.000000	0.000000	0.000000
payment_method_Credit_Card_auto	1.000000	0.000000	1.000000	0.000000	0.000000
payment_method_Mailed_Check	0.000000	0.000000	0.000000	1.000000	1.000000
equip_fail_year	1.000000	1.000000	1.000000	0.000000	1.000000
monthly_charge	172.455519	242.632554	159.947583	119.956840	149.948316
bandwidth_gb	904.536110	800.982766	2054.706961	2164.579412	271.493436
tenure	6.795513	1.156681	15.754144	17.087227	1.670972

In [107…

```
X = forest_df.drop(columns=['tenure'])
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   port_numeric                        10000 non-null  int64  
 1   phone_numeric                       10000 non-null  int64  
 2   multi_numeric                       10000 non-null  int64  
 3   backup_numeric                      10000 non-null  int64  
 4   protection_numeric                  10000 non-null  int64  
 5   contract_One_Year                   10000 non-null  int64  
 6   contract_Two_Year                   10000 non-null  int64  
 7   internet_service_Fiber_Optic        10000 non-null  int64  
 8   internet_service_None               10000 non-null  int64  
 9   payment_method_Credit_Card_auto     10000 non-null  int64  
 10  payment_method_Mailed_Check         10000 non-null  int64  
 11  equip_fail_year                     10000 non-null  int64  
 12  monthly_charge                      10000 non-null  float64
 13  bandwidth_gb                        10000 non-null  float64
dtypes: float64(2), int64(12)
memory usage: 1.1 MB
```

In [108…]
```
y = forest_df[['tenure']]
y.head().T
```

Out[108]:

	0	1	2	3	4
tenure	6.795513	1.156681	15.754144	17.087227	1.670972

In [109…]
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
```

In [110…]
```
X_train_df = X_train.copy()
X_train_df.to_csv('/Users/katherinevoakes/Desktop/wgu/x_train_forest_data.csv', index=False)
```

In [111…]
```
X_test_df = X_test.copy()
X_test_df.to_csv('/Users/katherinevoakes/Desktop/wgu/x_test_forest_data.csv', index=False)
```

In [112…]
```
y_train_df = y_train.copy()
y_train_df.to_csv('/Users/katherinevoakes/Desktop/wgu/y_train_forest_data.csv', index=False)
```

In [113…]
```
y_test_df = y_test.copy()
y_test_df.to_csv('/Users/katherinevoakes/Desktop/wgu/y_test_forest_data.csv', index=False)
```

In [114…]
```
#initial modeling - Decision Tree
X_train = pd.read_csv('/Users/katherinevoakes/Desktop/wgu/x_train_forest_data.csv')
```

```
y_train = pd.read_csv('/Users/katherinevoakes/Desktop/wgu/y_train_forest_data.csv')
X_test = pd.read_csv('/Users/katherinevoakes/Desktop/wgu/x_test_forest_data.csv')
y_test = pd.read_csv('/Users/katherinevoakes/Desktop/wgu/y_test_forest_data.csv')
```

In [115…
```
y_test = np.ravel(y_test)
print(y_test.shape)
y_train = np.ravel(y_train)
print(y_train.shape)
print(X_train.shape)
print(X_test.shape)
```

```
(2000,)
(8000,)
(8000, 14)
(2000, 14)
```

In [131…
```
r_forest = RandomForestRegressor(n_estimators= 100, min_samples_leaf=0.05, random_state=3)
r_forest.fit(X_train, y_train)
y_pred_train = r_forest.predict(X_train)
print("Training Predicition:", y_pred_train)
```

```
Training Predicition: [52.12818473 68.56791246  7.73487283 ... 54.49760711  3.41028202
 61.0884663 ]
```

In [132…
```
mse_train = MSE(y_train, y_pred_train)
print("MSE Training:", mse_train)
print("-"*100)

rmse_train = mse_train**(1/2)
print("RMSE Training", rmse_train)
print("-"*100)

r2_train = r2_score(y_train, y_pred_train)
print("R2 Train Score", r2_train)
```

```
MSE Training: 11.504558325154573
----------------------------------------------------------------------------------------------------
RMSE Training 3.391837013353468
----------------------------------------------------------------------------------------------------
R2 Train Score 0.983606313751947
```

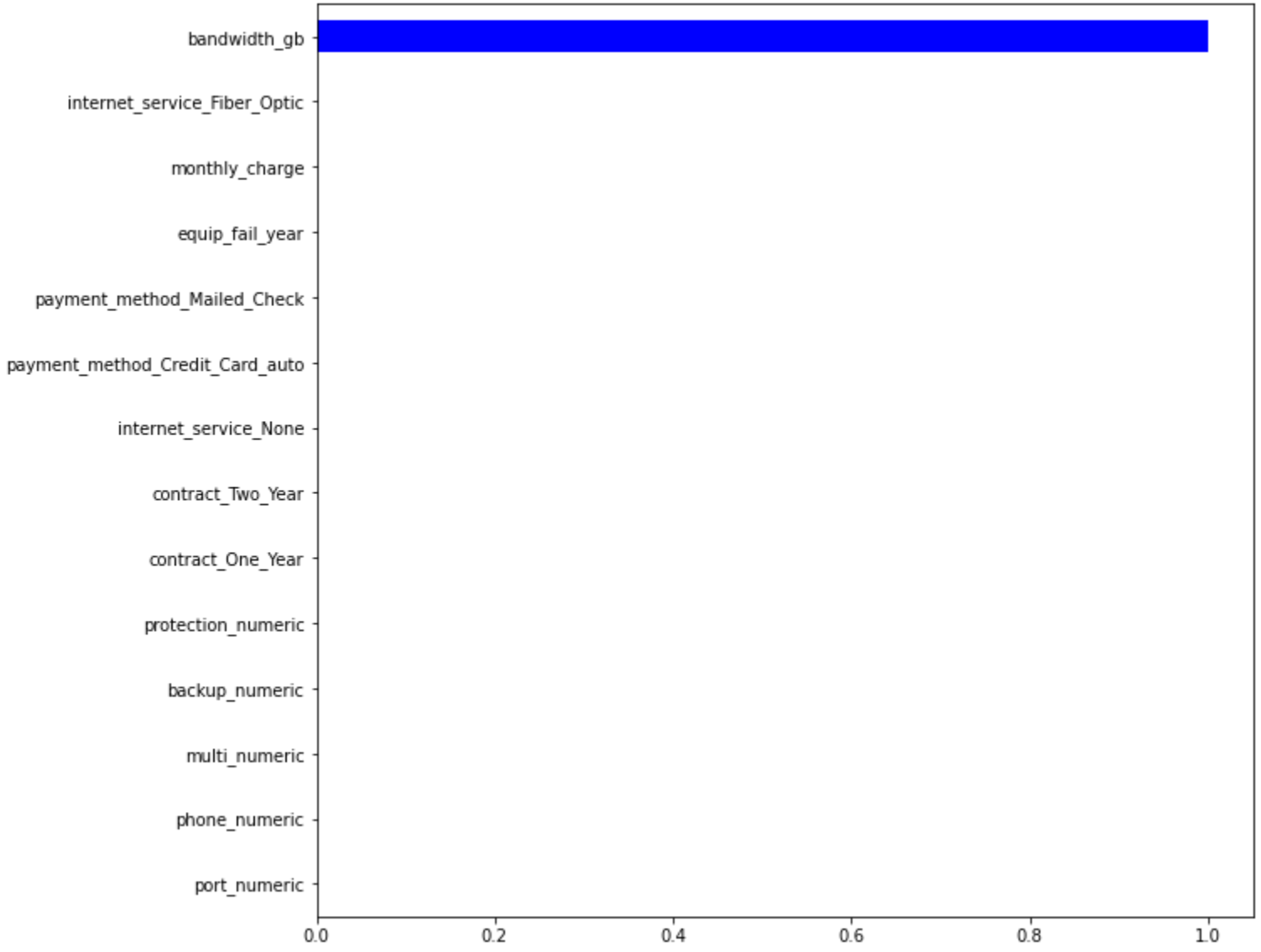
In [133…
```
y_pred = r_forest.predict(X_test)
print("Predictions:", y_pred)
print('-'*100)

mse = MSE(y_test, y_pred)
print("MSE Testing :",mse)
print('-'*100)
```

```
rmse = mse**(1/2)
print("RMSE Testing :", rmse)
print("-"*100)

r2 = r2_score(y_test, y_pred)
print("R2 Testing score :", r2)
```

```
Predictions: [61.0884663  68.56791246  7.70102768 ... 61.57090624 46.27609497
 54.70852479]
----------------------------------------------------------------------------------------------------
MSE Testing : 11.460057406603791
----------------------------------------------------------------------------------------------------
RMSE Testing : 3.3852706548522513
----------------------------------------------------------------------------------------------------
R2 Testing score : 0.9833544128476647
```

In [134…

```
important_test_features = pd.Series(r_forest.feature_importances_, index= X.columns)

sorted_important_test_features =important_test_features.sort_values()

sorted_important_test_features.plot(kind='barh', color='blue');
plt.show()
```



In [121…

```
estimator = RandomForestRegressor()
estimator.get_params()
```

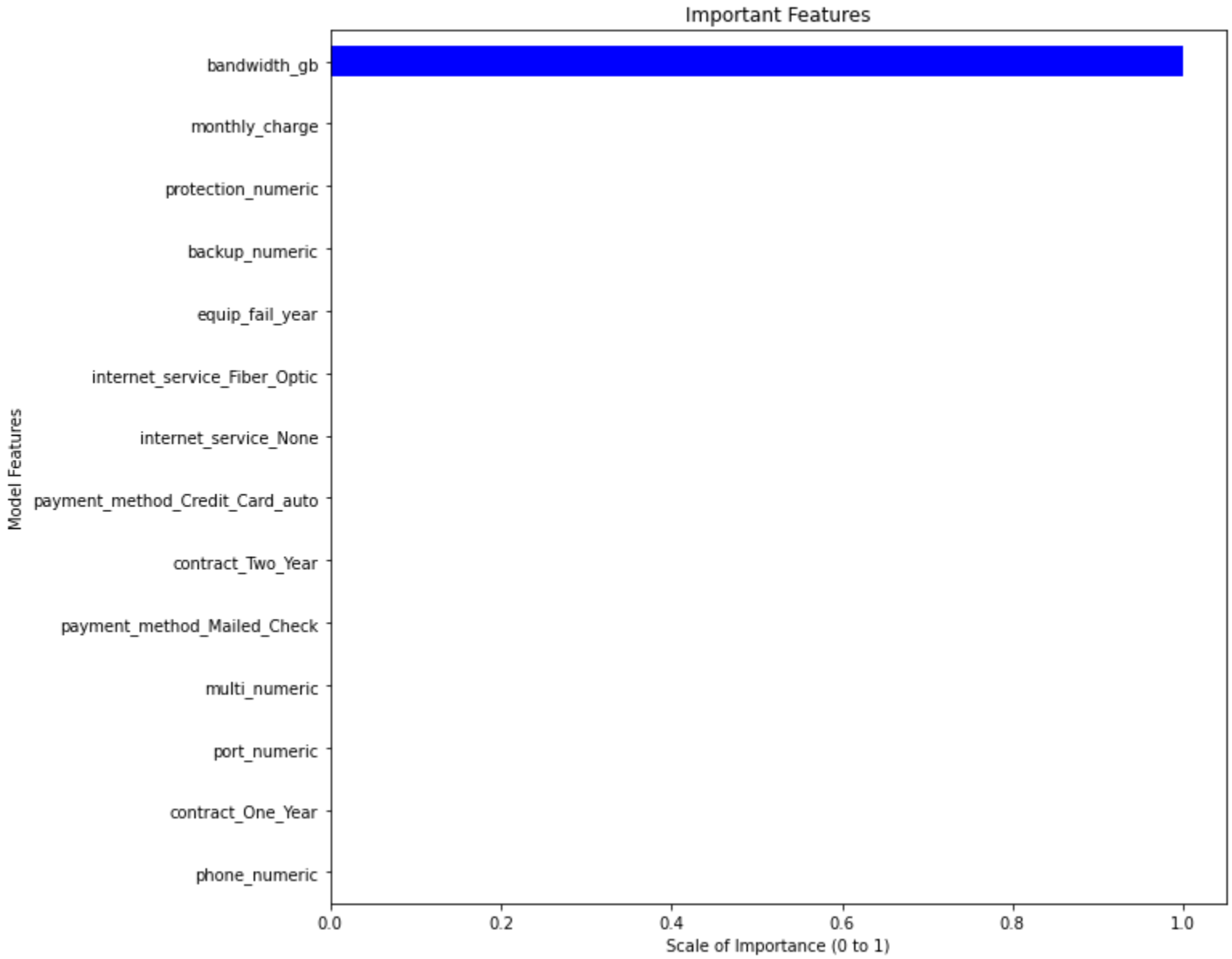
Out[121]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

In [122…
```
#perform cross validation
estimator = RandomForestRegressor()
params = { "n_estimators": [50,125,150],
           "min_samples_leaf": [0.08,0.1,0.12],
          "max_features": [3,8,12],
           "max_depth": [8, None],
           "random_state":[3]
         }

r_forest_best = GridSearchCV(estimator=estimator, param_grid=params, cv=5)
r_forest_cv = r_forest_best.fit(X_train, y_train)
train_accuracy = r_forest_cv.score(X_train, y_train)
y_pred_new = r_forest_cv.predict(X_test)
test_accuracy = r_forest_cv.score(X_test, y_test)


print("Adjusted forest parameters MSE:", MSE(y_test, y_pred_new))
print("-"*100)
print(test_accuracy)
```

```
Adjusted forest parameters MSE: 17.209534911443637
----------------------------------------------------------------------------------------------------
0.9750033701354306
```

In [126…
```
#which features were the best?
best_features = pd.Series(data=r_forest_cv.best_estimator_.feature_importances_,
                          index=X_train.columns)
sorted_features = best_features.sort_values()
```

```
sorted_features.plot(kind='barh', color='blue')
plt.xlabel('Scale of Importance (0 to 1)')
plt.ylabel('Model Features')
plt.title('Important Features')
plt.show()
```



In [ ]: