Healthcare Insurance Analysis

January 23, 2023

1 Healthcare Insurance Analysis

[1]:
```
# Importing library & removing the unnecessary warnings.

import numpy as np
import pandas as pd
import warnings
import seaborn as sns
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
```

[2]:
```
# Importing datasets.
hosp_d = pd.read_csv('Hospitalisation details.csv')
med_e = pd.read_csv('Medical Examinations.csv')
name = pd.read_excel('Names.xlsx')
```

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

[3]:

	Customer ID	year	month	date	children	charges	Hospital tier	City tier	State ID
0	Id2335	1992	Jul	9	0	563.84	tier - 2	tier - 3	R1013
1	Id2334	1992	Nov	30	0	570.62	tier - 2	tier - 1	R1013
2	Id2333	1993	Jun	30	0	600.00	tier - 2	tier - 1	R1013
3	Id2332	1992	Sep	13	0	604.54	tier - 3	tier - 3	R1013
4	Id2331	1998	Jul	27	0	637.26	tier - 3	tier - 3	R1013

[4]:
```
med_e.head()
```

[4]:

	Customer ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history
0	Id1	47.410	7.47	No	No	No
1	Id2	30.360	5.77	No	No	No

```
2         Id3  34.485  11.87          yes              No              No
3         Id4  38.095   6.05           No              No              No
4         Id5  35.530   5.45           No              No              No

  NumberOfMajorSurgeries smoker
0       No major surgery    yes
1       No major surgery    yes
2                      2    yes
3       No major surgery    yes
4       No major surgery    yes
```

[5]:
```
name.head()
```

[5]:
```
  Customer ID                    name
0         Id1       Hawks, Ms.  Kelly
1         Id2  Lehner, Mr.  Matthew D
2         Id3           Lu, Mr.  Phil
3         Id4    Osborne, Ms.  Kelsey
4         Id5    Kadala, Ms.  Kristyn
```

1.1 Project Task: Week 1

1. Collate the files so that all the information is in one place

[6]:
```
# Combining Hospitalisation details & Medical Examinations datastes.
df = pd.merge(hosp_d, med_e, on = 'Customer ID')
```

[7]:
```
# Now combine the last dataset Names with previous combined dataset.
ht = pd.merge(df,name,on = 'Customer ID')
ht
```

[7]:
```
     Customer ID  year month  date  children   charges Hospital tier  \
0         Id2335  1992   Jul     9         0    563.84      tier - 2
1         Id2334  1992   Nov    30         0    570.62      tier - 2
2         Id2333  1993   Jun    30         0    600.00      tier - 2
3         Id2332  1992   Sep    13         0    604.54      tier - 3
4         Id2331  1998   Jul    27         0    637.26      tier - 3
…            …    …    …    …       …       …           …
2330         Id5  1989   Jun    19         0  55135.40      tier - 1
2331         Id4  1991   Jun     6         1  58571.07      tier - 1
2332         Id3  1970     ?    11         3  60021.40      tier - 1
2333         Id2  1977   Jun     8         0  62592.87      tier - 2
2334         Id1  1968   Oct    12         0  63770.43      tier - 1

     City tier State ID     BMI  HBA1C Heart Issues Any Transplants  \
0     tier - 3    R1013  17.580   4.51           No              No
```

```
1     tier - 1    R1013  17.600   4.39          No              No
2     tier - 1    R1013  16.470   6.35          No              No
3     tier - 3    R1013  17.700   6.28          No              No
4     tier - 3    R1013  22.340   5.57          No              No
…        …       …      …     …          …             …
2330  tier - 2    R1012  35.530   5.45          No              No
2331  tier - 3    R1024  38.095   6.05          No              No
2332  tier - 1    R1012  34.485  11.87         yes              No
2333  tier - 3    R1013  30.360   5.77          No              No
2334  tier - 3    R1013  47.410   7.47          No              No

     Cancer history NumberOfMajorSurgeries smoker  \
0                No                      1     No
1                No                      1     No
2               Yes                      1     No
3                No                      1     No
4                No                      1     No
…              …                       …    …
2330             No       No major surgery    yes
2331             No       No major surgery    yes
2332             No                      2    yes
2333             No       No major surgery    yes
2334             No       No major surgery    yes

                                name
0                  German, Mr.  Aaron K
1                Rosendahl, Mr.  Evan P
2                    Albano, Ms.  Julie
3     Riveros Gonzalez, Mr.  Juan D. Sr.
4                 Brietzke, Mr.  Jordan
…                                 …
2330               Kadala, Ms.  Kristyn
2331               Osborne, Ms.  Kelsey
2332                     Lu, Mr.  Phil
2333             Lehner, Mr.  Matthew D
2334                 Hawks, Ms.  Kelly

[2335 rows x 17 columns]
```

2. Check for missing values in the dataset

[8]:
```
ht.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2335 entries, 0 to 2334
Data columns (total 17 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
```

```
 0   Customer ID             2335 non-null   object
 1   year                    2335 non-null   object
 2   month                   2335 non-null   object
 3   date                    2335 non-null   int64
 4   children                2335 non-null   int64
 5   charges                 2335 non-null   float64
 6   Hospital tier           2335 non-null   object
 7   City tier               2335 non-null   object
 8   State ID                2335 non-null   object
 9   BMI                     2335 non-null   float64
 10  HBA1C                   2335 non-null   float64
 11  Heart Issues            2335 non-null   object
 12  Any Transplants         2335 non-null   object
 13  Cancer history          2335 non-null   object
 14  NumberOfMajorSurgeries  2335 non-null   object
 15  smoker                  2335 non-null   object
 16  name                    2335 non-null   object
dtypes: float64(3), int64(2), object(12)
memory usage: 328.4+ KB
```

3. Find the percentage of rows that have trivial value (for example, ?), and delete such rows if they do not contain significant information

[9]:
```
trivial_value = ht[ht.eq("?").any(1)]
trivial_value
```

[9]:

	Customer ID	year	month	date	children	charges	Hospital tier	City tier	State ID	BMI	HBA1C	Heart Issues	Any Transplants
11	Id2324	1999	Dec	26	0	700.00	?	tier - 3	R1013	22.240	5.04	No	No
13	Id2322	2002	?	19	0	750.00	tier - 3	tier - 1	R1012	21.380	8.01	No	No
17	Id2318	1996	?	18	0	770.38	tier - 3	?	R1012	18.820	5.51	yes	No
542	Id1793	1995	Dec	1	3	4827.90	tier - 1	tier - 2	?	18.905	4.91	yes	No
1046	Id1289	?	Jul	24	0	8534.67	tier - 2	tier - 3	R1024	24.320	11.56	yes	No
1049	Id1286	?	Dec	12	1	8547.69	tier - 2	tier - 1	R1013	29.370	8.01	yes	No
1700	Id635	2004	Jul	17	0	15518.18	tier - 2	tier - 3	R1015	25.175	4.96	No	yes
1775	Id560	1994	Jul	1	3	17663.14	tier - 1	tier - 3	R1013	23.980	4.90	No	No
2165	Id170	2000	Sep	5	1	37165.16	tier - 1	tier - 3	?	37.620	6.32	yes	yes
2332	Id3	1970	?	11	3	60021.40	tier - 1						

	Cancer history	NumberOfMajorSurgeries	smoker	name
11	No	No Major Surgery	No	Person_11
13	No	No Major Surgery	No	Person_13
17	No	No Major Surgery	No	Person_17
542	No	No Major Surgery	No	Person_542
1046	No	No Major Surgery	No	Person_1046
1049	No	No Major Surgery	No	Person_1049
1700	No	No Major Surgery	No	Person_1700
1775	No	No Major Surgery	No	Person_1775
2165	No	No Major Surgery	No	Person_2165

```
2332  tier - 1    R1012  34.485  11.87          yes              No

     Cancer history NumberOfMajorSurgeries smoker                    name
11               No       No major surgery     No    Duffy, Ms.  Meghan K
13               No       No major surgery     No      Street, Ms.  Holly
17               No       No major surgery     No  Gagnon, Ms.  Candice M
542              No                      1     No  Capriolo, Mr.  Michael
1046             No                      1     No   Levine, Ms.  Annie J.
1049             No                      1     No  Ainsley, Ms.  Katie M.
1700             No                      1      ?   Bruns, Mr.  Zachary T
1775             No       No major surgery      ?       Pearlman, Mr.  Oz
2165             No                      2    yes      Torphy, Mr.  Bobby
2332             No                      2    yes           Lu, Mr.  Phil
```

[10]:
```
trivial_value.shape
```

[10]:
```
(10, 17)
```

[11]:
```
# Percentage of row that have the trivial values
round(trivial_value.shape[0]/ht.shape[0]*100, 2)
```

[11]:
```
0.43
```

There is total 0.43% of rows contain the trivial values.

[12]:
```
# Now lets drop the all row that contain the trivial values in the data set.
ht.drop(ht[ht.eq("?").any(1)].index, axis=0, inplace=True)
```

[13]:
```
ht.shape
```

[13]:
```
(2325, 17)
```

4.Use the necessary transformation methods to deal with the nominal and ordinal categorical variables in the dataset.

[14]:
```
ht_categorical = ht.select_dtypes(exclude='number')
ht_categorical.head()
```

[14]:
```
  Customer ID  year month Hospital tier City tier State ID Heart Issues  \
0      Id2335  1992   Jul      tier - 2  tier - 3    R1013           No
1      Id2334  1992   Nov      tier - 2  tier - 1    R1013           No
2      Id2333  1993   Jun      tier - 2  tier - 1    R1013           No
3      Id2332  1992   Sep      tier - 3  tier - 3    R1013           No
4      Id2331  1998   Jul      tier - 3  tier - 3    R1013           No

  Any Transplants Cancer history NumberOfMajorSurgeries smoker  \
0              No             No                      1     No
1              No             No                      1     No
```

```
2              No            Yes                      1     No
3              No             No                      1     No
4              No             No                      1     No

                                 name
0                German, Mr.  Aaron K
1              Rosendahl, Mr.  Evan P
2                  Albano, Ms.  Julie
3  Riveros Gonzalez, Mr.  Juan D. Sr.
4               Brietzke, Mr.  Jordan
```

First we will deal with the nominal categorical variable.

[15]:
```
ht["Heart Issues"].value_counts()
```

[15]:
```
No     1405
yes     920
Name: Heart Issues, dtype: int64
```

[16]:
```
ht["Any Transplants"].value_counts()
```

[16]:
```
No     2183
yes     142
Name: Any Transplants, dtype: int64
```

[17]:
```
ht["Cancer history"].value_counts()
```

[17]:
```
No     1934
Yes     391
Name: Cancer history, dtype: int64
```

[18]:
```
ht["smoker"].value_counts()
```

[18]:
```
No     1839
yes     486
Name: smoker, dtype: int64
```

[19]:
```
# We have some categorical values so first of all we have to transform then by␣
 ↪using the label encoder.
from sklearn.preprocessing import LabelEncoder
```

[20]:
```
le = LabelEncoder()
```

[21]:
```
ht["Heart Issues"] = le.fit_transform(ht["Heart Issues"])
ht["Any Transplants"] = le.fit_transform(ht["Any Transplants"])
ht["Cancer history"] = le.fit_transform(ht["Cancer history"])
ht["smoker"] = le.fit_transform(ht["smoker"])
```

[22]:
```
ht["Heart Issues"].value_counts()
```

[22]:
```
0    1405
1     920
Name: Heart Issues, dtype: int64
```

[23]:
```
ht.head()
```

[23]:

	Customer ID	year	month	date	children	charges	Hospital tier	City tier	\
0	Id2335	1992	Jul	9	0	563.84	tier - 2	tier - 3	
1	Id2334	1992	Nov	30	0	570.62	tier - 2	tier - 1	
2	Id2333	1993	Jun	30	0	600.00	tier - 2	tier - 1	
3	Id2332	1992	Sep	13	0	604.54	tier - 3	tier - 3	
4	Id2331	1998	Jul	27	0	637.26	tier - 3	tier - 3	

	State ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	\
0	R1013	17.58	4.51	0	0	0	
1	R1013	17.60	4.39	0	0	0	
2	R1013	16.47	6.35	0	0	1	
3	R1013	17.70	6.28	0	0	0	
4	R1013	22.34	5.57	0	0	0	

	NumberOfMajorSurgeries	smoker	name
0	1	0	German, Mr. Aaron K
1	1	0	Rosendahl, Mr. Evan P
2	1	0	Albano, Ms. Julie
3	1	0	Riveros Gonzalez, Mr. Juan D. Sr.
4	1	0	Brietzke, Mr. Jordan

Now we will deal with the ordinal categorical variable.

[24]:
```
def clean_ordinal_variable(val):
    return int(val.replace("tier", "").replace(" ", "").replace("-", ""))
```

[25]:
```
ht["Hospital tier"] = ht["Hospital tier"].map(clean_ordinal_variable)
ht["City tier"] = ht["City tier"].map(clean_ordinal_variable)
```

[26]:
```
ht["City tier"].value_counts()
```

[26]:
```
2    807
3    789
1    729
Name: City tier, dtype: int64
```

[27]:
```
ht.head()
```

[27]:

	Customer ID	year	month	date	children	charges	Hospital tier	City tier
0	Id2335	1992	Jul	9	0	563.84	2	3
1	Id2334	1992	Nov	30	0	570.62	2	1
2	Id2333	1993	Jun	30	0	600.00	2	1
3	Id2332	1992	Sep	13	0	604.54	3	3
4	Id2331	1998	Jul	27	0	637.26	3	3

	State ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history
0	R1013	17.58	4.51	0	0	0
1	R1013	17.60	4.39	0	0	0
2	R1013	16.47	6.35	0	0	1
3	R1013	17.70	6.28	0	0	0
4	R1013	22.34	5.57	0	0	0

	NumberOfMajorSurgeries	smoker	name
0	1	0	German, Mr. Aaron K
1	1	0	Rosendahl, Mr. Evan P
2	1	0	Albano, Ms. Julie
3	1	0	Riveros Gonzalez, Mr. Juan D. Sr.
4	1	0	Brietzke, Mr. Jordan

5.The dataset has State ID, which has around 16 states. All states are not represented in equal proportions in the data. Creating dummy variables for all regions may also result in too many insignificant predictors. Nevertheless, only R1011, R1012, and R1013 are worth investigating further. Create a suitable strategy to create dummy variables with these restraints.

[28]:
```
ht["State ID"].value_counts()
```

[28]:
```
R1013    609
R1011    574
R1012    572
R1024    159
R1026     84
R1021     70
R1016     64
R1025     40
R1023     38
R1017     36
R1019     26
R1022     14
R1014     13
R1015     11
R1018      9
R1020      6
Name: State ID, dtype: int64
```

It is clear from the above code some of the state is worth investigator like R1013, R1012, R1011

and R1024.

[29]:
```
data = ht[ht['State ID'].isin(['R1011', 'R1012', 'R1013'])]
data.shape
```

[29]:
```
(1755, 17)
```

[30]:
```
data['State ID'] = le.fit_transform(data['State ID'])
```

[31]:
```
data['State ID'].unique()
```

[31]:
```
array([2, 1, 0])
```

6.The variable NumberOfMajorSurgeries also appears to have string values. Apply a suitable method to clean up this variable.

[32]:
```
data['NumberOfMajorSurgeries'].value_counts()
```

[32]:
```
No major surgery    816
1                   713
2                   208
3                    18
Name: NumberOfMajorSurgeries, dtype: int64
```

The Number Of Major Surgeries variable contain string value no major Surgery that means 0 surgery so we will replace this value into int value equal to zero.

[33]:
```
data['NumberOfMajorSurgeries'].replace('No major surgery',0,inplace=True)
```

[34]:
```
data['NumberOfMajorSurgeries'].value_counts()
```

[34]:
```
0    816
1    713
2    208
3     18
Name: NumberOfMajorSurgeries, dtype: int64
```

7.Age appears to be a significant factor in this analysis. Calculate the patients' ages based on their dates of birth.

[35]:
```
from datetime import datetime
```

[36]:
```
# Create a new column 'dob' by concatenating year, month and date columns
data['dob'] = data['year'].astype(str) + ' ' + data['month'] + ' ' +␣
 ↪data['date'].astype(str)

# convert the dob column to datetime type using pd.to_datetime()
data['dob'] = pd.to_datetime(data['dob'], format='%Y %b %d')
```

```
# calculate the age by subtracting the dob from the current date
data['age'] = (datetime.now() - data['dob']).apply(lambda x: x.days/365).
 ↪astype(int)

# now dropping year , date , months column
data.drop(columns=['year','month','date'],axis=1,inplace=True)
```

[37]:
```
data
```

[37]:

	Customer ID	children	charges	Hospital tier	City tier	State ID
0	Id2335	0	563.84	2	3	2
1	Id2334	0	570.62	2	1	2
2	Id2333	0	600.00	2	1	2
3	Id2332	0	604.54	3	3	2
4	Id2331	0	637.26	3	3	2
…	…	…	…	…	…	…
2328	Id7	1	51194.56	1	3	0
2329	Id6	0	52590.83	1	3	0
2330	Id5	0	55135.40	1	2	1
2333	Id2	0	62592.87	2	3	2
2334	Id1	0	63770.43	1	3	2

	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history
0	17.58	4.51	0	0	0
1	17.60	4.39	0	0	0
2	16.47	6.35	0	0	1
3	17.70	6.28	0	0	0
4	22.34	5.57	0	0	0
…	…	…	…	…	…
2328	36.40	6.07	0	0	0
2329	32.80	6.59	0	0	0
2330	35.53	5.45	0	0	0
2333	30.36	5.77	0	0	0
2334	47.41	7.47	0	0	0

	NumberOfMajorSurgeries	smoker	name
0	1	0	German, Mr. Aaron K
1	1	0	Rosendahl, Mr. Evan P
2	1	0	Albano, Ms. Julie
3	1	0	Riveros Gonzalez, Mr. Juan D. Sr.
4	1	0	Brietzke, Mr. Jordan
…	…	…	…
2328	0	1	Macpherson, Mr. Scott
2329	0	1	Baker, Mr. Russell B.
2330	0	1	Kadala, Ms. Kristyn
2333	0	1	Lehner, Mr. Matthew D
2334	0	1	Hawks, Ms. Kelly

```
            dob  age
0    1992-07-09   30
1    1992-11-30   30
2    1993-06-30   29
3    1992-09-13   30
4    1998-07-27   24
…           …  …
2328 1994-10-27   28
2329 1962-08-04   60
2330 1989-06-19   33
2333 1977-06-08   45
2334 1968-10-12   54

[1755 rows x 16 columns]
```

8.The gender of the patient may be an important factor in determining the cost of hospitalization. The salutations in a beneficiary's name can be used to determine their gender. Make a new field for the beneficiary's gender.

the salutation (Ms.) denote the female and (Mr.) denote the male. > The gender will play the inportant role to predict the hospitalization cost so for model building we directly denote the gender by int.

Male = 0 & Female = 1

[38]:
```
gender = ['0' if 'Mr.' in name else '1' for name in data['name']]
data['Gender'] = gender
data.head()
```

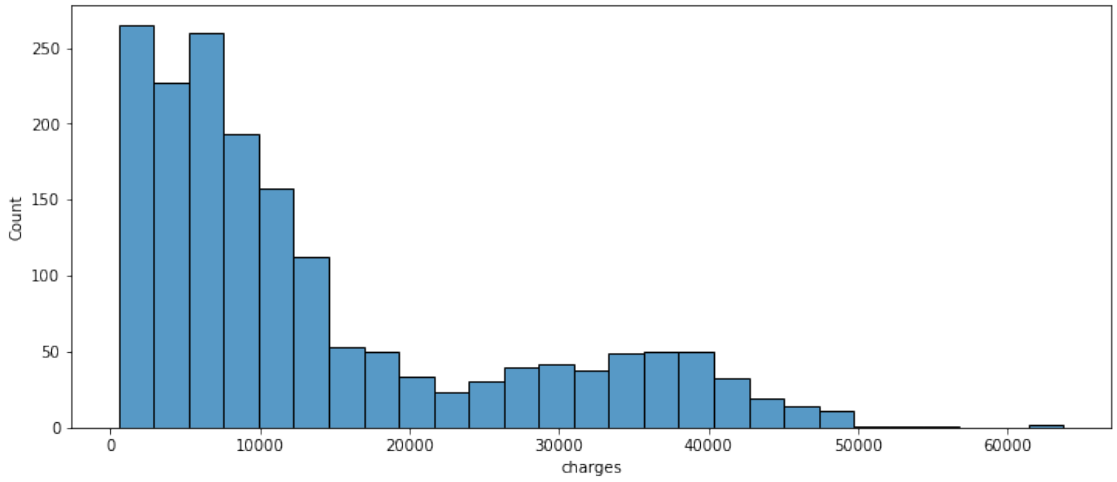
[38]:
```
  Customer ID  children  charges  Hospital tier  City tier  State ID    BMI  \
0      Id2335         0   563.84              2          3         2  17.58
1      Id2334         0   570.62              2          1         2  17.60
2      Id2333         0   600.00              2          1         2  16.47
3      Id2332         0   604.54              3          3         2  17.70
4      Id2331         0   637.26              3          3         2  22.34

   HBA1C  Heart Issues  Any Transplants  Cancer history  \
0   4.51             0                0               0
1   4.39             0                0               0
2   6.35             0                0               1
3   6.28             0                0               0
4   5.57             0                0               0

  NumberOfMajorSurgeries  smoker                                name  \
0                      1       0               German, Mr.  Aaron K
1                      1       0             Rosendahl, Mr.  Evan P
2                      1       0                 Albano, Ms.  Julie
```

```
3                     1       0  Riveros Gonzalez, Mr.  Juan D. Sr.
4                     1       0               Brietzke, Mr.  Jordan

          dob  age Gender
0  1992-07-09   30      0
1  1992-11-30   30      0
2  1993-06-30   29      1
3  1992-09-13   30      0
4  1998-07-27   24      0
```

9.You should also visualize the distribution of costs using a histogram, box and whisker plot,and swarm plot.

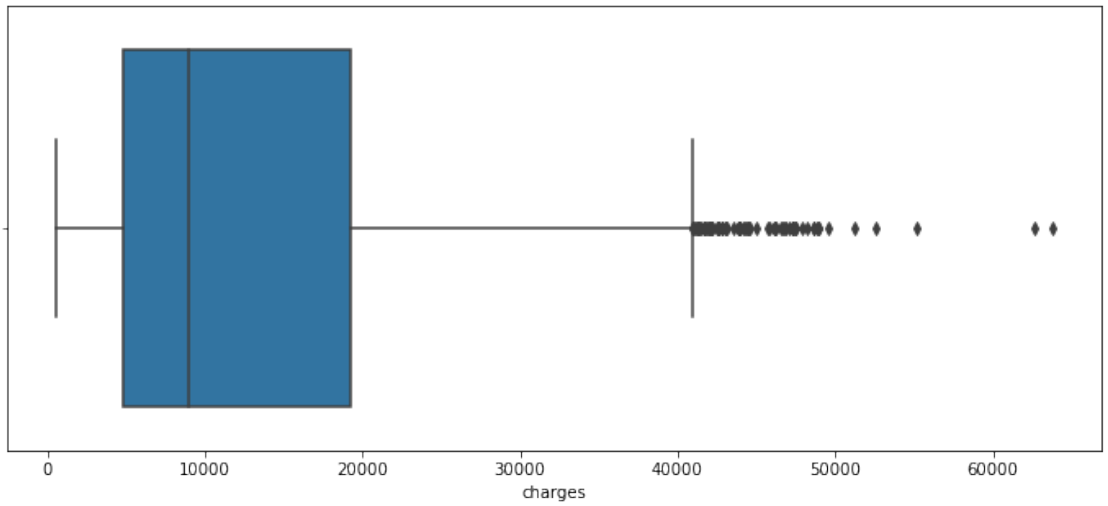
[39]:
```
plt.figure(figsize=(12,5))
sns.histplot(data['charges'])
```

[39]:
```
<AxesSubplot:xlabel='charges', ylabel='Count'>
```



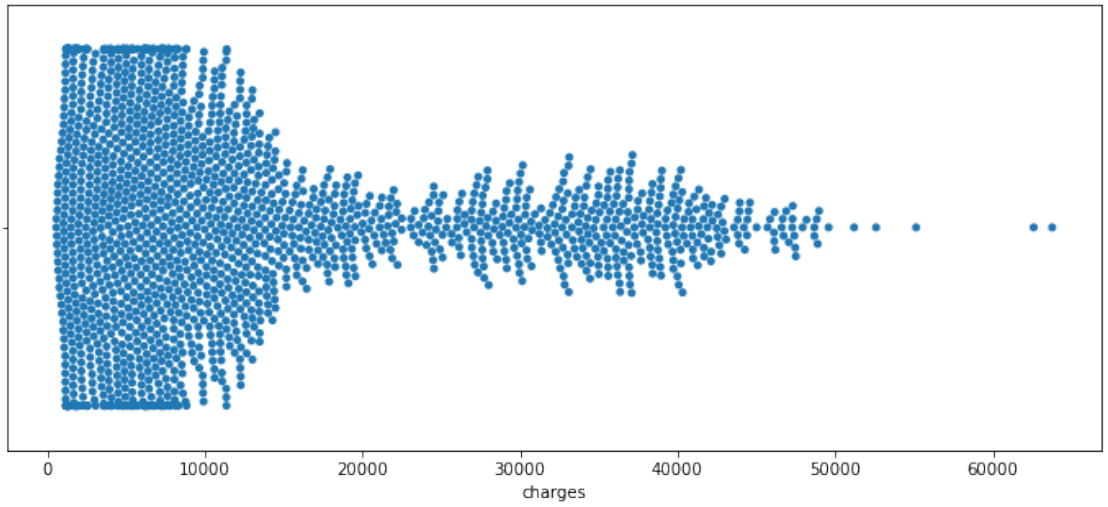
[40]:
```
plt.figure(figsize=(12,5))
sns.boxplot(data['charges'])
```

[40]:
```
<AxesSubplot:xlabel='charges'>
```



[41]:
```
plt.figure(figsize=(12,5))
sns.swarmplot(data['charges'])
```

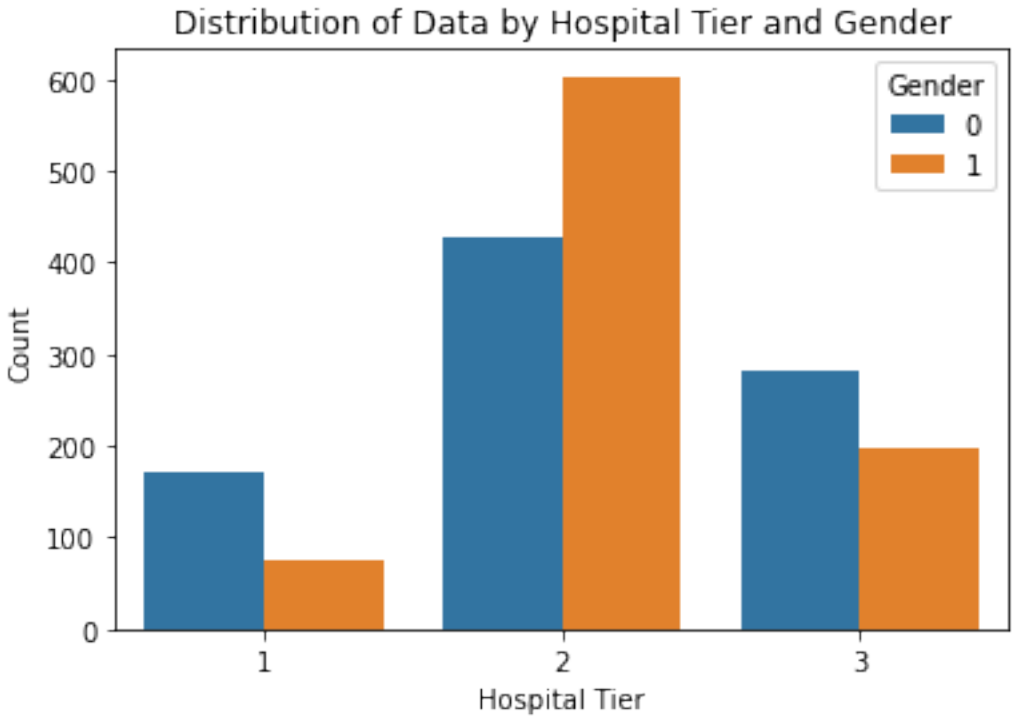
[41]: <AxesSubplot:xlabel='charges'>



10.State how the distribution is different across gender and tiers of hospitals.

[42]:
```
# Create a stacked bar chart
sns.countplot(x='Hospital tier',hue='Gender',data=data)

# Add labels and title
plt.xlabel('Hospital Tier')
```

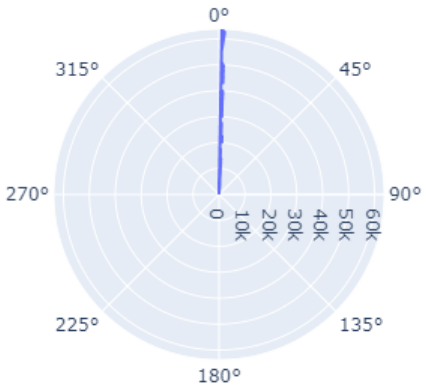
```
plt.ylabel('Count')
plt.title('Distribution of Data by Hospital Tier and Gender')

# Show the plot
plt.show()
```



11.Create a radar chart to showcase the median hospitalization cost for each tier of hospitals.

[43]:
```
import plotly.express as px
```

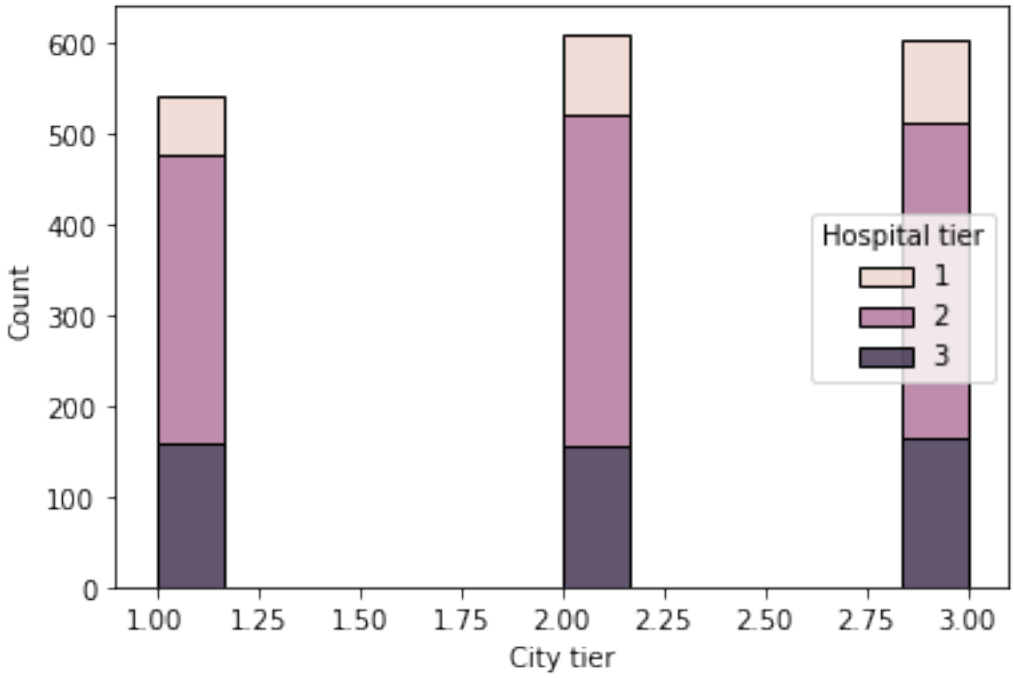
[44]:
```
fig = px.line_polar(data, r='charges', theta='Hospital tier', line_close=True)
fig.update_traces(fill='toself')
fig.show()
```



12.Create a frequency table and a stacked bar chart to visualize the count of people in the different tiers of cities and hospitals.

[45]:
```
sns.histplot(data,x='City tier' ,hue ='Hospital tier',multiple='stack')
```

[45]: <AxesSubplot:xlabel='City tier', ylabel='Count'>



[46]:
```
data1 = data
```

13.Test the following null hypotheses: a.The average hospitalization costs for the three types of hospitals are not significantly different.

[47]:
```
import scipy.stats as stats
print('Null Hypothesis => Average hospitalization costs for the three types of␣
 ↪hospitals are not significantly different.')
# Perform ANOVA test using the `charges` column and grouping by the `Hospital␣
 ↪tier` column
f_val, p_val = stats.f_oneway(data[data['Hospital tier'] == 'tier -␣
 ↪1']['charges'],
                              data[data['Hospital tier'] == 'tier -␣
 ↪2']['charges'],
                              data[data['Hospital tier'] == 'tier -␣
 ↪3']['charges'])

# Print the p-value
print('P-value :',p_val)

# Compare p-value with significance value(0.05)
if p_val < 0.05:
    print("Reject null hypothesis")
else:
    print("Accept null hypothesis")
```

```
Null Hypothesis => Average hospitalization costs for the three types of
hospitals are not significantly different.
P-value : nan
Accept null hypothesis
```

b.The average hospitalization costs for the three types of cities are not significantly different.

[48]:
```
import scipy.stats as stats
print('Null Hypothesis => Average hospitalization costs for the three types of␣
 ↪cities are not significantly different.')
# Perform ANOVA test using the `charges` column and grouping by the `City tier`␣
 ↪column
f_val, p_val = stats.f_oneway(data[data['City tier'] == 'tier - 1']['charges'],
                              data[data['City tier'] == 'tier - 2']['charges'],
                              data[data['City tier'] == 'tier - 3']['charges'])

# Print the p-value
print('P-value :',p_val)

# Compare p-value with significance value(0.05)
if p_val < 0.05:
```

```
    print("Reject null hypothesis")
else:
    print("Accept null hypothesis")
```

```
Null Hypothesis => Average hospitalization costs for the three types of cities
are not significantly different.
P-value : nan
Accept null hypothesis
```

c.The average hospitalization cost for smokers is not significantly different from the average cost for nonsmokers.

[49]:
```
import scipy.stats as stats
print('Null Hypothesis => Average hospitalization cost for smokers is not␣
 ↪significantly different from the average cost for nonsmokers.')
# Perform ANOVA test using the `charges` column and grouping by the `smoker`␣
 ↪column
t_val, p_val = stats.ttest_ind(data[data['smoker'] == 'yes']['charges'],
                               data[data['smoker'] == 'No']['charges'])

# Print the p-value
print('P-value :',p_val)

# Compare p-value with significance value(0.05)
if p_val < 0.05:
    print("Reject null hypothesis")
else:
    print("Accept null hypothesis")
```

```
Null Hypothesis => Average hospitalization cost for smokers is not significantly
different from the average cost for nonsmokers.
P-value : nan
Accept null hypothesis
```

d.Smoking and heart issues are independent

[50]:
```
import pandas as pd
from scipy.stats import chi2_contingency

# create a contingency table of the observed counts
contingency_table = pd.crosstab(data['smoker'], data['Heart Issues'])

# conduct the chi-squared test
chi2, p, dof, expected = chi2_contingency(contingency_table)
print(f'P-value = {p}')
# interpret the p-value
if p < 0.05:
```

```
    print("Reject the null hypothesis, Smoking and heart issues are independent.
 ↪")
else:
    print("Accept null hypothesis, Smoking and heart issues are independent.")
```

```
P-value = 0.9107065371179246
Accept null hypothesis, Smoking and heart issues are independent.
```

2 Project Task: Week 2

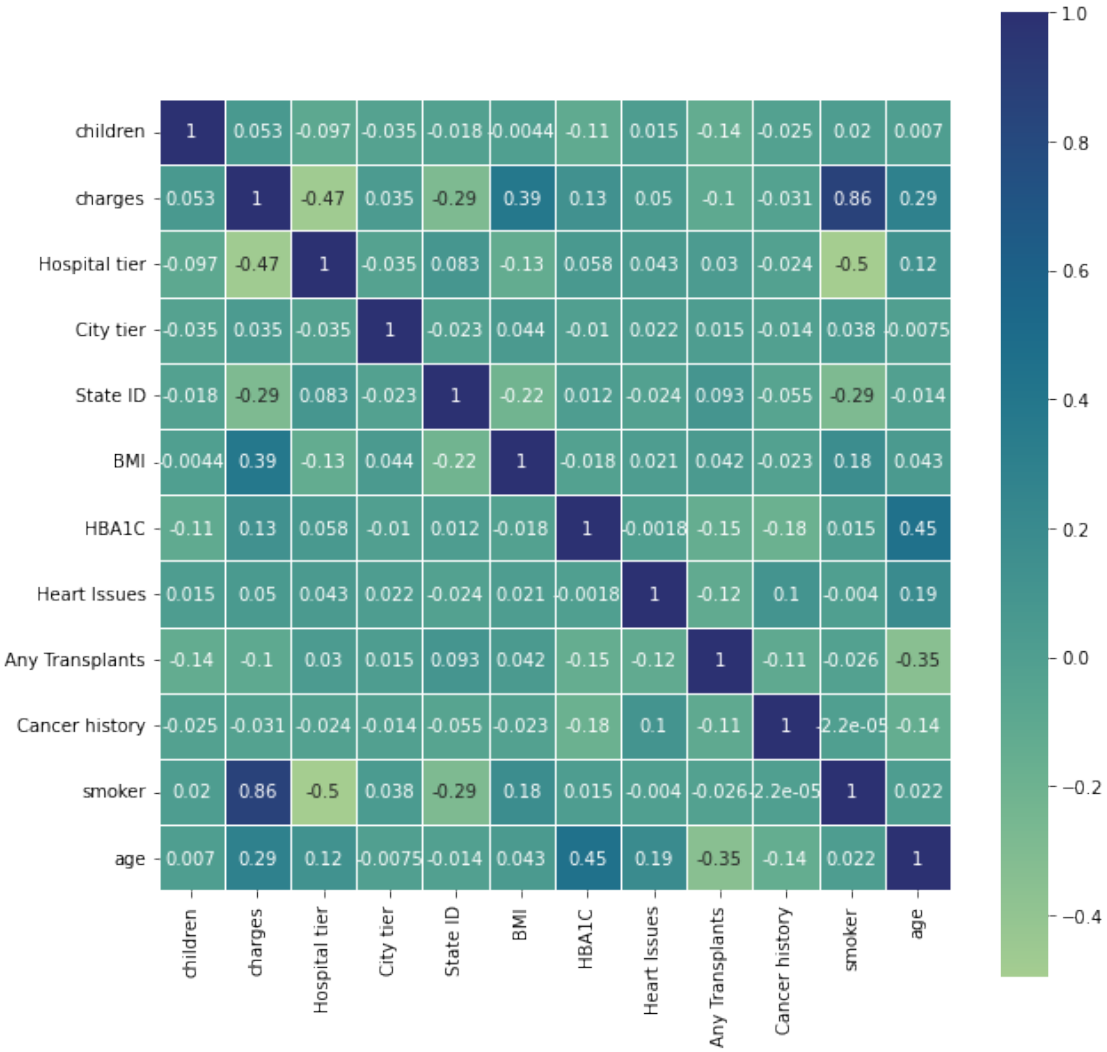
2.0.1 Machine Learning

1.Examine the correlation between predictors to identify highly correlated predictors. Use a heatmap to visualize this.

[51]:
```
# In the data frame same of the column are not usable to model building so lets␣
 ↪first drop all.
# then indentify the highly corelated predictor.
data.drop(["Customer ID",'name', 'dob'], inplace=True, axis=1)
```

[52]:
```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),square=True,annot=True,linewidths=1, cmap="crest")
```

[52]: <AxesSubplot:>



2.Develop and evaluate the final model using regression with a stochastic gradient descent optimizer. Also, ensure that you apply all the following suggestions: Note: •Perform the stratified 5 fold cross validation technique for model building and validation •Use standardization and hyperparameter tuning effectively •Use-sklearn pipelines •Use appropriate regularization techniques to address the bias variance trade off a.Create five folds in the data, and introduce a variable to identify the folds b.For each fold, run a for loop and ensure that 80 percent of the data is used to train the model and the remaining 20 percent is used to validate it in each iteration c.Develop five distinct models and five distinct validation scores (root mean squared error values) d.Determine the variable importance scores, and identify the redundant variables

[53]:
```
# lets first seperate the input and output data.
x = data.drop(["charges"], axis=1)
y = data[['charges']]
```

[54]:
```
# Lets split the data set into the training and testing data.
from sklearn.model_selection import train_test_split
```

[55]:
```
x_train, x_test, y_train, y_test  = train_test_split(x,y, test_size=.20,␣
↪random_state=10)
```

[56]:
```
# Now standardize the data.
from sklearn.preprocessing import StandardScaler
```

[57]:
```
sc = StandardScaler()
```

[58]:
```
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

[59]:
```
from sklearn.linear_model import SGDRegressor
```

[60]:
```
from sklearn.model_selection import GridSearchCV

params = {'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2,0.3,0.4,0.5,
                    0.6,0.7,0.8,0.9,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,
                    9.0,10.0,20,50,100,500,1000],
          'penalty': ['l2', 'l1', 'elasticnet']}

sgd = SGDRegressor()

# Cross Validation
folds = 5
model_cv = GridSearchCV(estimator = sgd,
                        param_grid = params,
                        scoring = 'neg_mean_absolute_error',
                        cv = folds,
                        return_train_score = True,
                        verbose = 1)
model_cv.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 84 candidates, totalling 420 fits
```

[60]:
```
GridSearchCV(cv=5, estimator=SGDRegressor(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3,
                                   0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 2.0, 3.0,
                                   4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20, 50,
                                   100, 500, 1000],
                         'penalty': ['l2', 'l1', 'elasticnet']},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

[61]:
```
model_cv.best_params_
```

[61]:
```
{'alpha': 100, 'penalty': 'l1'}
```

[62]:
```
sgd = SGDRegressor(alpha= 100, penalty= 'l1')
```

[63]:
```
sgd.fit(x_train, y_train)
```

[63]:
```
SGDRegressor(alpha=100, penalty='l1')
```

[64]:
```
sgd.score(x_test, y_test)
```

[64]:
```
0.8787491936368994
```

[65]:
```
y_pred = sgd.predict(x_test)
```

[66]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

[67]:
```
sgd_mae = mean_absolute_error(y_test, y_pred)
sgd_mse = mean_squared_error(y_test, y_pred)
sgd_rmse = sgd_mse*(1/2.0)
```

[68]:
```
print("MAE:", sgd_mae)
print("MSE:", sgd_mse)
print("RMSE:", sgd_rmse)
```

```
MAE: 2836.7235596121945
MSE: 21636674.76437782
RMSE: 10818337.38218891
```

[69]:
```
# d. Determine the variable importance scores, and identify the redundant␣
↪variables
importance = sgd.coef_
```

[70]:
```
pd.DataFrame(importance, index = x.columns, columns=['Feature_imp'])
```

[70]:

	Feature_imp
children	195.975836
Hospital tier	-1090.186842
City tier	-81.241821
State ID	0.000000
BMI	2780.058209
HBA1C	0.000000
Heart Issues	0.000000
Any Transplants	0.000000
Cancer history	47.692779
NumberOfMajorSurgeries	0.000000
smoker	9543.016828
age	3371.093724

```
Gender                        0.000000
```

3.Use random forest and extreme gradient boosting for cost prediction, share your cross validation results, and calculate the variable importance scores

2.0.2 random forest

[71]:
```
from sklearn.ensemble import RandomForestRegressor
```

[72]:
```
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)

# Train the model on training data
rf.fit(x_train, y_train)
```

[72]:
```
RandomForestRegressor(n_estimators=1000, random_state=42)
```

[73]:
```
score = rf.score(x_test,y_test)
score
```

[73]:
```
0.9071126584143127
```

[74]:
```
y_pred = rf.predict(x_test)
```

[75]:
```
rf_mae = mean_absolute_error(y_test, y_pred)
```

[76]:
```
rf_mae
```

[76]:
```
1972.368668689459
```

2.0.3 extreme gradient boosting

[77]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

[78]:
```
# Instantiate model with 1000 decision trees
gbr = GradientBoostingRegressor(n_estimators = 1000, random_state = 42)

# Train the model on training data
gbr.fit(x_train, y_train)
```

[78]:
```
GradientBoostingRegressor(n_estimators=1000, random_state=42)
```

[79]:
```
score = gbr.score(x_test,y_test)
score
```

[79]: 0.89355499766892

[80]:
```
y_pred = gbr.predict(x_test)
```

[81]:
```
gbr_mae = mean_absolute_error(y_test, y_pred)
gbr_mae
```

[81]: 2485.567097648491

4.Case scenario: Estimate the cost of hospitalization for Christopher, Ms. Jayna (her date of birth is 12/28/1988, height is 170 cm, and weight is 85 kgs). She lives in a tier 1 city and her state's State ID is R1011. She lives with her partner and two children. She was found to be nondiabetic (HbA1c = 5.8). She smokes but is otherwise healthy. She has had no transplants or major surgeries. Her father died of lung cancer. Hospitalization costs will be estimated using tier 1 hospitals.

[82]:
```
import datetime as dt
```

[83]:
```
# First we need to calculate the age of the person.
date = str(19881228)
date1 = pd.to_datetime(date, format = "%Y%m%d")
```

[84]:
```
current_date = dt.datetime.now()
current_date
```

[84]: datetime.datetime(2023, 1, 23, 21, 51, 7, 113546)

[85]:
```
age = (current_date - date1)
age
```

[85]: Timedelta('12444 days 21:51:07.113546')

[86]:
```
age = int(12421/365)
age
```

[86]: 34

[87]:
```
# now with the help of height and weight we will calculate the BMI.
height_m = 170/100
height_sq = height_m*height_m
BMI = 85/height_sq
np.round(BMI,2)
```

[87]: 29.41

[88]:
```
# Now lets gen
list = [[2,1,1,24.41,5.8,0,0,0,0,1,1,34,0]]
```

[89]:
```
df = pd.DataFrame(list, columns = ['children', 'Hospital tier', 'City tier',␣
↪'BMI', 'HBA1C','Heart Issues', 'Any Transplants',
                                'Cancer history','NumberOfMajorSurgeries',␣
↪'smoker', 'State_ID', 'age', 'gender'] )
df
```

[89]:

	children	Hospital tier	City tier	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history	NumberOfMajorSurgeries	smoker	State_ID	age	gender
0	2	1	1	24.41	5.8	0	0	0	0	1	1	34	0

5.Find the predicted hospitalization cost using all five models. The predicted value should be the mean of the five models' predicted values.

[90]:
```
Hospital_cost = []
```

[91]:
```
# Now lets predict the hospitalization cost through SGDRegressor
Cost1 = sgd.predict(df)
Hospital_cost.append(Cost1)
```

[92]:
```
# Now lets predict the hospitalization cost through Random Forest
Cost2 = rf.predict(df)
Hospital_cost.append(Cost2)
```

[93]:
```
# Now lets predict the hospitalization cost throug Extreme gradient Booster
Cost3 = gbr.predict(df)
Hospital_cost.append(Cost3)
```

[94]:
```
avg_cost = np.mean(Hospital_cost)
avg_cost
```

[94]:
```
88251.25689525904
```

So in the new case the avg predicted hospitalization cost is 88251.25