Healthcare-insurance-analysis-project-1

[81]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[82]:
```
import warnings
warnings.filterwarnings("ignore")
```

[83]:
```
hospital = pd.read_csv("Hospitalisation details.csv")
medical = pd.read_csv("Medical Examinations.csv")
customer = pd.read_excel("Names.xlsx")
```

[84]:
```
hospital.head()
```

[84]:
```
  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
0    R1013
1    R1013
2    R1013
3    R1013
4    R1013
```

[85]:
```
hospital.shape
```

[85]: (2343, 9)

[86]:
```
hospital.describe()
```

[86]:
```
              date     children       charges
count  2343.000000  2343.000000   2343.000000
mean     15.554844     1.026035  13559.067870
```

```
std       8.721194    1.233847  11922.658415
min       1.000000    0.000000    563.840000
25%       8.000000    0.000000   5084.010000
50%      15.000000    0.000000   9634.540000
75%      23.000000    2.000000  17029.675000
max      30.000000    5.000000  63770.430000
```

[87]:
```
hospital.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2343 entries, 0 to 2342
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Customer ID    2343 non-null   object
 1   year           2343 non-null   object
 2   month          2343 non-null   object
 3   date           2343 non-null   int64
 4   children       2343 non-null   int64
 5   charges        2343 non-null   float64
 6   Hospital tier  2343 non-null   object
 7   City tier      2343 non-null   object
 8   State ID       2343 non-null   object
dtypes: float64(1), int64(2), object(6)
memory usage: 164.9+ KB
```

[88]:
```
medical.head()
```

[88]:
```
  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
```

[89]:
```
medical.shape
```

[89]: (2335, 8)

[90]:
```
medical.describe()
```

[90]:
```
                BMI        HBA1C
count   2335.000000  2335.000000
mean      30.972649     6.578998
std        8.742095     2.228731
min       15.010000     4.000000
25%       24.600000     4.900000
50%       30.400000     5.810000
75%       36.300000     7.955000
max       55.050000    12.000000
```

[91]:
```
medical.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2335 entries, 0 to 2334
Data columns (total 8 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Customer ID            2335 non-null   object
 1   BMI                    2335 non-null   float64
 2   HBA1C                  2335 non-null   float64
 3   Heart Issues           2335 non-null   object
 4   Any Transplants        2335 non-null   object
 5   Cancer history         2335 non-null   object
 6   NumberOfMajorSurgeries 2335 non-null   object
 7   smoker                 2335 non-null   object
dtypes: float64(2), object(6)
memory usage: 146.1+ KB
```

[92]:
```
customer.head()
```

[92]:
```
  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
```

[93]:
```
customer.shape
```

[93]: (2335, 2)

[94]:
```
customer.describe()
```

[94]:
```
       Customer ID               name
count         2335               2335
unique        2335               2335
top            Id1  Hawks, Ms.  Kelly
```

```
freq            1                 1
```

[95]:
```
customer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2335 entries, 0 to 2334
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Customer ID  2335 non-null   object
 1   name         2335 non-null   object
dtypes: object(2)
memory usage: 36.6+ KB
```

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

[96]:
```
df = pd.merge(pd.merge(hospital,medical,on='Customer ID'),customer,on='Customer␣
↪ID')
```

[97]:
```
df
```

[97]:
```
     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]
```

[98]:
```
df.shape
```

[98]:
```
(2335, 17)
```

[99]:
```
df.describe()
```

[99]:

	date	children	charges	BMI	HBA1C
count	2335.000000	2335.000000	2335.000000	2335.000000	2335.000000
mean	15.563597	1.025696	13529.918034	30.972649	6.578998
std	8.720508	1.234754	11898.654299	8.742095	2.228731
min	1.000000	0.000000	563.840000	15.010000	4.000000
25%	8.000000	0.000000	5084.010000	24.600000	4.900000
50%	15.000000	0.000000	9630.910000	30.400000	5.810000
75%	23.000000	2.000000	16912.295000	36.300000	7.955000
max	30.000000	5.000000	63770.430000	55.050000	12.000000

0.2 2. Check for missing values in the dataset

[100]:
```
df.isna().sum().sum()
```

[100]: 0

- There are no missing values in the dataset

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

[101]:
```
trivial= df[df == "?"].count(axis=1).sum()
```

[102]:
```
trivial
```

[102]: 11

[103]:
```
trivial.shape
```

[103]: ()

[104]:
```
total= df.shape[0]
total
```

[104]: 2335

[105]:
```
percentage = (trivial / total_rows) * 100
```

[106]:
```
percentage
```

[106]: 0.47109207708779444

- 0.47109 % of rows contain the trivial values.

[107]:
```
print("Percentage of trivial rows: {:.2f}%".format(percentage))
```

```
Percentage of trivial rows: 0.47%
```

[108]:
```
df = df[df != "?"].dropna()
```

[109]:
```
df.shape
```

[109]: (2325, 17)

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

[110]:
```
df_cat = df.select_dtypes(exclude='number')
```

[111]:
```
df_cat.columns
```

[111]:
```
Index(['Customer ID', 'year', 'month', 'Hospital tier', 'City tier',
       'State ID', 'Heart Issues', 'Any Transplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'smoker', 'name'],
      dtype='object')
```

[112]:
```
df['Heart Issues'].value_counts()
```

[112]:
```
Heart Issues
No     1405
yes     920
Name: count, dtype: int64
```

[113]:
```
df['Any Transplants'].value_counts()
```

[113]:
```
Any Transplants
No     2183
yes     142
Name: count, dtype: int64
```

[114]:
```
df['Cancer history'].value_counts()
```

[114]:
```
Cancer history
No     1934
Yes     391
Name: count, dtype: int64
```

[115]:
```
df['smoker'].value_counts()
```

[115]:
```
smoker
No     1839
yes     486
Name: count, dtype: int64
```

[116]:
```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

[117]:
```
df["Heart Issues"]=le.fit_transform(df["Heart Issues"])
df["Any Transplants"]=le.fit_transform(df["Any Transplants"])
df["Cancer history"]=le.fit_transform(df["Cancer history"])
df["smoker"]=le.fit_transform(df["smoker"])
```

[118]:
```
df["Heart Issues"].value_counts()
```

[118]:
```
Heart Issues
0    1405
1     920
Name: count, dtype: int64
```

- Hospital tier and city tier are ordinal categorical variables

[119]:
```
def fun(val):
    return int(val.replace("tier", "").replace(" ", "").replace("-", ""))
```

[120]:
```
df['Hospital tier'] = df['Hospital tier'].map(fun)
```

[121]:
```
df['City tier'] = df['City tier'].map(fun)
```

[122]:
```
df
```

[122]:
```
     Customer ID  year month  date  children   charges  Hospital tier  \
0         Id2335  1992   Jul     9         0    563.84              2
1         Id2334  1992   Nov    30         0    570.62              2
2         Id2333  1993   Jun    30         0    600.00              2
3         Id2332  1992   Sep    13         0    604.54              3
4         Id2331  1998   Jul    27         0    637.26              3
…            …     …     …     …         …       …
2329         Id6  1962   Aug     4         0  52590.83              1
2330         Id5  1989   Jun    19         0  55135.40              1
2331         Id4  1991   Jun     6         1  58571.07              1
2333         Id2  1977   Jun     8         0  62592.87              2
2334         Id1  1968   Oct    12         0  63770.43              1

      City tier State ID     BMI  HBA1C  Heart Issues  Any Transplants  \
0             3    R1013  17.580   4.51             0                0
1             1    R1013  17.600   4.39             0                0
2             1    R1013  16.470   6.35             0                0
3             3    R1013  17.700   6.28             0                0
4             3    R1013  22.340   5.57             0                0
…           …        …       …      …           …
2329          3    R1011  32.800   6.59             0                0
2330          2    R1012  35.530   5.45             0                0
2331          3    R1024  38.095   6.05             0                0
2333          3    R1013  30.360   5.77             0                0
2334          3    R1013  47.410   7.47             0                0

      Cancer history NumberOfMajorSurgeries  smoker  \
0                  0                      1       0
1                  0                      1       0
```

```
2                    1                        1       0
3                    0                        1       0
4                    0                        1       0
…              …                     …     …
2329                 0       No major surgery       1
2330                 0       No major surgery       1
2331                 0       No major surgery       1
2333                 0       No major surgery       1
2334                 0       No major surgery       1

                                         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
…                                     …
2329                  Baker, Mr.  Russell B.
2330                    Kadala, Ms.  Kristyn
2331                   Osborne, Ms.  Kelsey
2333                  Lehner, Mr.  Matthew D
2334                      Hawks, Ms.  Kelly

[2325 rows x 17 columns]
```

[123]:
```python
df['Hospital tier'].value_counts()
```

[123]:
```
Hospital tier
2    1334
3     691
1     300
Name: count, dtype: int64
```

0.5 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.

[124]:
```python
df['State ID'].value_counts()
```

[124]:
```
State ID
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: count, dtype: int64
```

[125]:
```python
Dummies = pd.get_dummies(df["State ID"], prefix= "State_ID")
Dummies
```

[125]:
```
      State_ID_R1011  State_ID_R1012  State_ID_R1013  State_ID_R1014  \
0              False           False            True           False
1              False           False            True           False
2              False           False            True           False
3              False           False            True           False
4              False           False            True           False
…                …              …               …               …
2329            True           False           False           False
2330           False            True           False           False
2331           False           False           False           False
2333           False           False            True           False
2334           False           False            True           False

      State_ID_R1015  State_ID_R1016  State_ID_R1017  State_ID_R1018  \
0              False           False           False           False
1              False           False           False           False
2              False           False           False           False
3              False           False           False           False
4              False           False           False           False
…                …              …               …               …
2329           False           False           False           False
2330           False           False           False           False
2331           False           False           False           False
2333           False           False           False           False
2334           False           False           False           False

      State_ID_R1019  State_ID_R1020  State_ID_R1021  State_ID_R1022  \
0              False           False           False           False
1              False           False           False           False
2              False           False           False           False
```

```
3            False          False          False          False
4            False          False          False          False
…            …              …              …              …
2329         False          False          False          False
2330         False          False          False          False
2331         False          False          False          False
2333         False          False          False          False
2334         False          False          False          False

      State_ID_R1023  State_ID_R1024  State_ID_R1025  State_ID_R1026
0            False          False          False          False
1            False          False          False          False
2            False          False          False          False
3            False          False          False          False
4            False          False          False          False
…            …              …              …              …
2329         False          False          False          False
2330         False          False          False          False
2331         False           True          False          False
2333         False          False          False          False
2334         False          False          False          False

[2325 rows x 16 columns]
```

[126]:
```python
Dummy = Dummies[['State_ID_R1011','State_ID_R1012', 'State_ID_R1013']]
Dummy
```

[126]:
```
      State_ID_R1011  State_ID_R1012  State_ID_R1013
0            False          False           True
1            False          False           True
2            False          False           True
3            False          False           True
4            False          False           True
…            …              …              …
2329          True          False          False
2330         False           True          False
2331         False          False          False
2333         False          False           True
2334         False          False           True

[2325 rows x 3 columns]
```

[127]:
```python
df = pd.concat([df,Dummy],axis=1)
```

[128]:
```python
df
```

[128]:

```
      Customer ID  year month  date  children   charges  Hospital tier
0          Id2335  1992   Jul     9         0    563.84              2  \
1          Id2334  1992   Nov    30         0    570.62              2
2          Id2333  1993   Jun    30         0    600.00              2
3          Id2332  1992   Sep    13         0    604.54              3
4          Id2331  1998   Jul    27         0    637.26              3
…             …     …     …   …       …       …           …
2329          Id6  1962   Aug     4         0  52590.83              1
2330          Id5  1989   Jun    19         0  55135.40              1
2331          Id4  1991   Jun     6         1  58571.07              1
2333          Id2  1977   Jun     8         0  62592.87              2
2334          Id1  1968   Oct    12         0  63770.43              1

      City tier State ID     BMI  HBA1C  Heart Issues  Any Transplants
0             3    R1013  17.580   4.51             0                0  \
1             1    R1013  17.600   4.39             0                0
2             1    R1013  16.470   6.35             0                0
3             3    R1013  17.700   6.28             0                0
4             3    R1013  22.340   5.57             0                0
…           …      …     …     …            …             …
2329          3    R1011  32.800   6.59             0                0
2330          2    R1012  35.530   5.45             0                0
2331          3    R1024  38.095   6.05             0                0
2333          3    R1013  30.360   5.77             0                0
2334          3    R1013  47.410   7.47             0                0

      Cancer history NumberOfMajorSurgeries  smoker
0                  0                      1       0  \
1                  0                      1       0
2                  1                      1       0
3                  0                      1       0
4                  0                      1       0
…              …                      …     …
2329               0       No major surgery       1
2330               0       No major surgery       1
2331               0       No major surgery       1
2333               0       No major surgery       1
2334               0       No major surgery       1

                                     name  State_ID_R1011  State_ID_R1012
0                    German, Mr.  Aaron K           False           False  \
1                 Rosendahl, Mr.  Evan P            False           False
2                     Albano, Ms.  Julie            False           False
3     Riveros Gonzalez, Mr.  Juan D. Sr.            False           False
4                    Brietzke, Mr.  Jordan          False           False
…                                      …            …             …
2329                 Baker, Mr.  Russell B.           True           False
```

```
2330                 Kadala, Ms.  Kristyn           False            True
2331                 Osborne, Ms.  Kelsey           False           False
2333               Lehner, Mr.  Matthew D           False           False
2334                    Hawks, Ms.  Kelly           False           False

      State_ID_R1013
0               True
1               True
2               True
3               True
4               True
…              …
2329           False
2330           False
2331           False
2333            True
2334            True

[2325 rows x 20 columns]
```

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

[129]:
```
df['NumberOfMajorSurgeries'] = df['NumberOfMajorSurgeries'].replace('No major␣
↪surgery',0)
```

[130]:
```
df['NumberOfMajorSurgeries'].value_counts()
```

[130]:
```
NumberOfMajorSurgeries
0    1070
1     961
2     272
3      22
Name: count, dtype: int64
```

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

[131]:
```
df['year'] = pd.to_datetime(df['year'], format='%Y').dt.year
df['year']
```

[131]:
```
0       1992
1       1992
2       1993
3       1992
4       1998
```

```
          …
2329     1962
2330     1989
2331     1991
2333     1977
2334     1968
Name: year, Length: 2325, dtype: int32
```

[132]:
```
df['month'] = pd.to_datetime(df['month'], format='%b').dt.month
df['month']
```

[132]:
```
0        7
1       11
2        6
3        9
4        7
        ..
2329     8
2330     6
2331     6
2333     6
2334    10
Name: month, Length: 2325, dtype: int32
```

[133]:
```
df['DateInt'] = df['year'].astype(str) + df['month'].astype(str).str.zfill(2) +
  df['date'].astype(str).str.zfill(2)
```

[134]:
```
df['DOB'] = pd.to_datetime(df.DateInt, format='%Y%m%d')
```

[135]:
```
df.drop(['DateInt'], inplace=True, axis=1)
```

[136]:
```
import datetime as dt
current_date = dt.datetime.now()
```

[137]:
```
df['Age'] = (((current_date-df.DOB).dt.days)/365).astype(int)
```

[138]:
```
df.head()
```

[138]:
```
  Customer ID  year  month  date  children  charges  Hospital tier  City tier
0      Id2335  1992      7     9         0   563.84              2          3  \
1      Id2334  1992     11    30         0   570.62              2          1
2      Id2333  1993      6    30         0   600.00              2          1
3      Id2332  1992      9    13         0   604.54              3          3
4      Id2331  1998      7    27         0   637.26              3          3

  State ID    BMI  …  Any Transplants  Cancer history
0    R1013  17.58  …                0               0  \
```

```
1    R1013  17.60  …                0                0
2    R1013  16.47  …                0                1
3    R1013  17.70  …                0                0
4    R1013  22.34  …                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

   State_ID_R1011 State_ID_R1012  State_ID_R1013        DOB  Age
0           False          False            True 1992-07-09   30
1           False          False            True 1992-11-30   30
2           False          False            True 1993-06-30   29
3           False          False            True 1992-09-13   30
4           False          False            True 1998-07-27   24

[5 rows x 22 columns]
```

[139]:
```
df.tail()
```

[139]:
```
     Customer ID  year  month  date  children   charges  Hospital tier
2329         Id6  1962      8     4         0  52590.83              1  \
2330         Id5  1989      6    19         0  55135.40              1
2331         Id4  1991      6     6         1  58571.07              1
2333         Id2  1977      6     8         0  62592.87              2
2334         Id1  1968     10    12         0  63770.43              1

      City tier State ID     BMI  …  Any Transplants  Cancer history
2329          3    R1011  32.800  …                0               0  \
2330          2    R1012  35.530  …                0               0
2331          3    R1024  38.095  …                0               0
2333          3    R1013  30.360  …                0               0
2334          3    R1013  47.410  …                0               0

      NumberOfMajorSurgeries  smoker                     name  State_ID_R1011
2329                       0       1  Baker, Mr.  Russell B.           True  \
2330                       0       1    Kadala, Ms.  Kristyn          False
2331                       0       1    Osborne, Ms.  Kelsey          False
2333                       0       1  Lehner, Mr.  Matthew D          False
2334                       0       1        Hawks, Ms.  Kelly          False

     State_ID_R1012  State_ID_R1013        DOB  Age
2329          False           False 1962-08-04   60
2330           True           False 1989-06-19   33
```

```
2331          False          False 1991-06-06   31
2333          False           True 1977-06-08   45
2334          False           True 1968-10-12   54

[5 rows x 22 columns]
```

[140]:
```
df
```

[140]:
```
     Customer ID  year  month  date  children   charges  Hospital tier
0         Id2335  1992      7     9         0    563.84              2  \
1         Id2334  1992     11    30         0    570.62              2
2         Id2333  1993      6    30         0    600.00              2
3         Id2332  1992      9    13         0    604.54              3
4         Id2331  1998      7    27         0    637.26              3
…            …     …      …    …         …         …              …
2329         Id6  1962      8     4         0  52590.83              1
2330         Id5  1989      6    19         0  55135.40              1
2331         Id4  1991      6     6         1  58571.07              1
2333         Id2  1977      6     8         0  62592.87              2
2334         Id1  1968     10    12         0  63770.43              1

      City tier State ID     BMI  …  Any Transplants  Cancer history
0             3    R1013  17.580  …                0               0  \
1             1    R1013  17.600  …                0               0
2             1    R1013  16.470  …                0               1
3             3    R1013  17.700  …                0               0
4             3    R1013  22.340  …                0               0
…           …      …     …     …              …             …
2329          3    R1011  32.800  …                0               0
2330          2    R1012  35.530  …                0               0
2331          3    R1024  38.095  …                0               0
2333          3    R1013  30.360  …                0               0
2334          3    R1013  47.410  …                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
…                        …     …                              …
2329                       0       1            Baker, Mr.  Russell B.
2330                       0       1              Kadala, Ms.  Kristyn
2331                       0       1              Osborne, Ms.  Kelsey
2333                       0       1            Lehner, Mr.  Matthew D
2334                       0       1                 Hawks, Ms.  Kelly
```

```
      State_ID_R1011 State_ID_R1012  State_ID_R1013        DOB  Age
0              False          False            True 1992-07-09   30
1              False          False            True 1992-11-30   30
2              False          False            True 1993-06-30   29
3              False          False            True 1992-09-13   30
4              False          False            True 1998-07-27   24
…              …             …             …         …  …
2329            True          False           False 1962-08-04   60
2330           False           True           False 1989-06-19   33
2331           False          False           False 1991-06-06   31
2333           False          False            True 1977-06-08   45
2334           False          False            True 1968-10-12   54

[2325 rows x 22 columns]
```

0.8 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.

[141]:
```python
def gen(x):
    if 'Ms.' in x:
        return 0
    else:
        return 1
```

[142]:
```python
df['Gender'] = df['name'].map(gen)
```

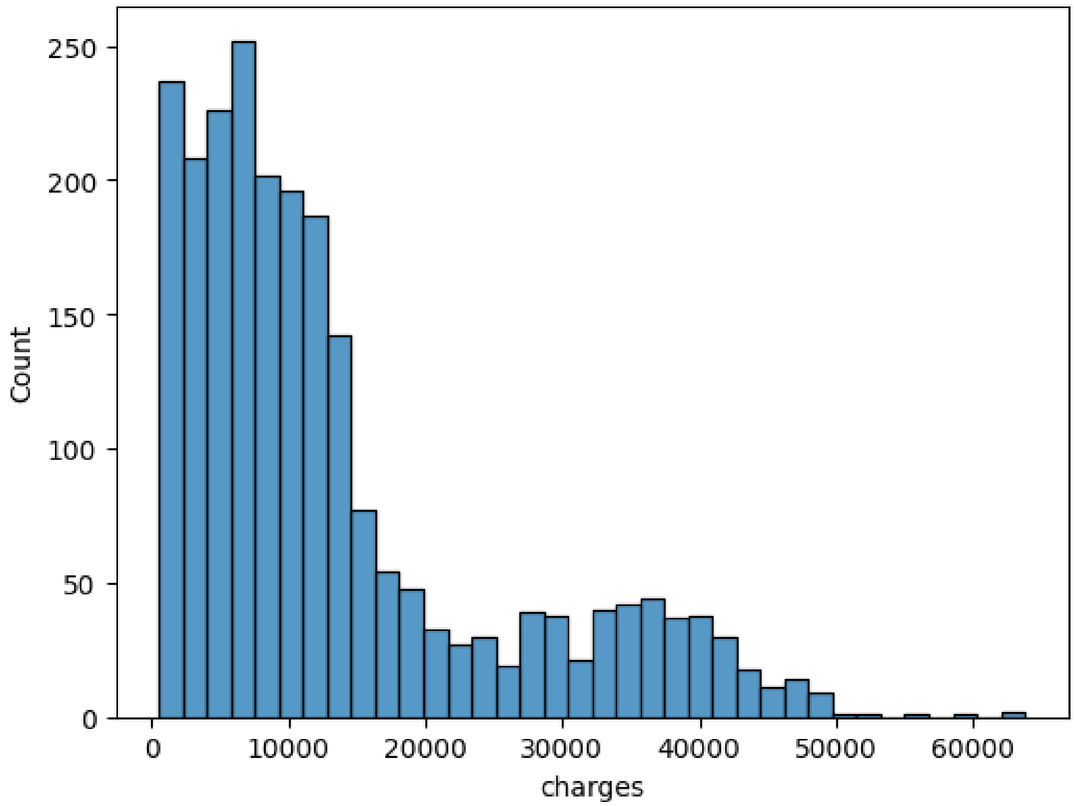
[143]:
```python
df['Gender']
```

[143]:
```
0       1
1       1
2       0
3       1
4       1
       ..
2329    1
2330    0
2331    0
2333    1
2334    0
Name: Gender, Length: 2325, dtype: int64
```

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

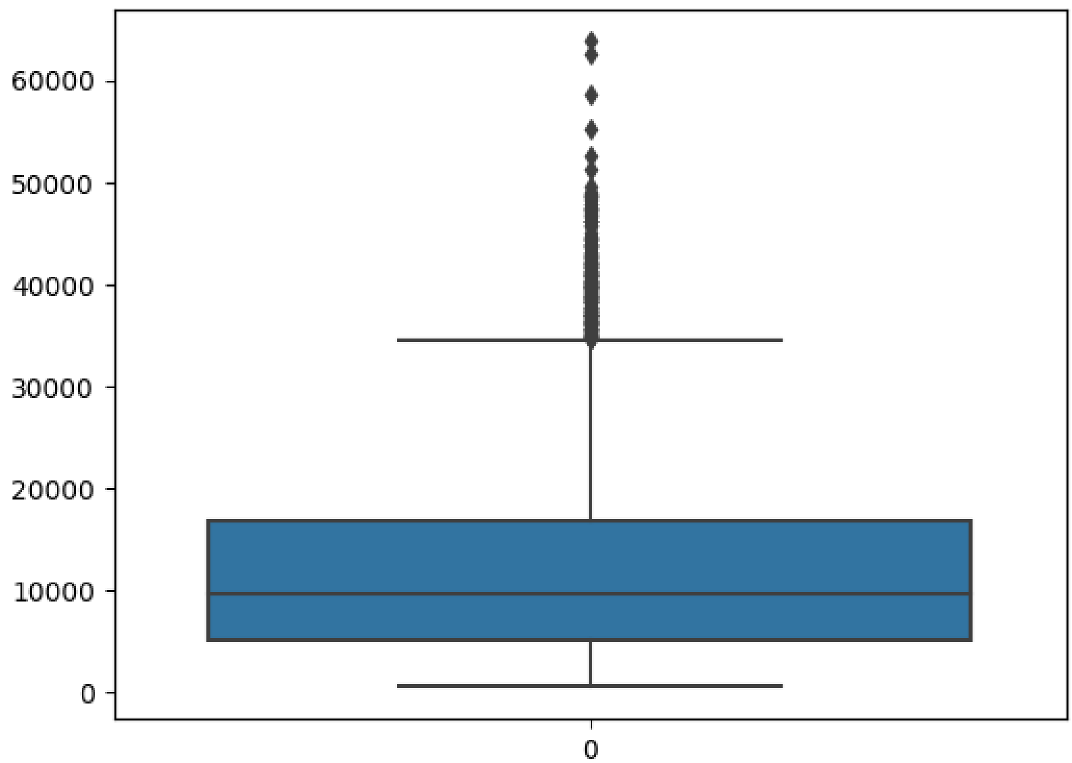
[144]:
```
# Histogram
sns.histplot(df['charges'])
```

[144]: <Axes: xlabel='charges', ylabel='Count'>

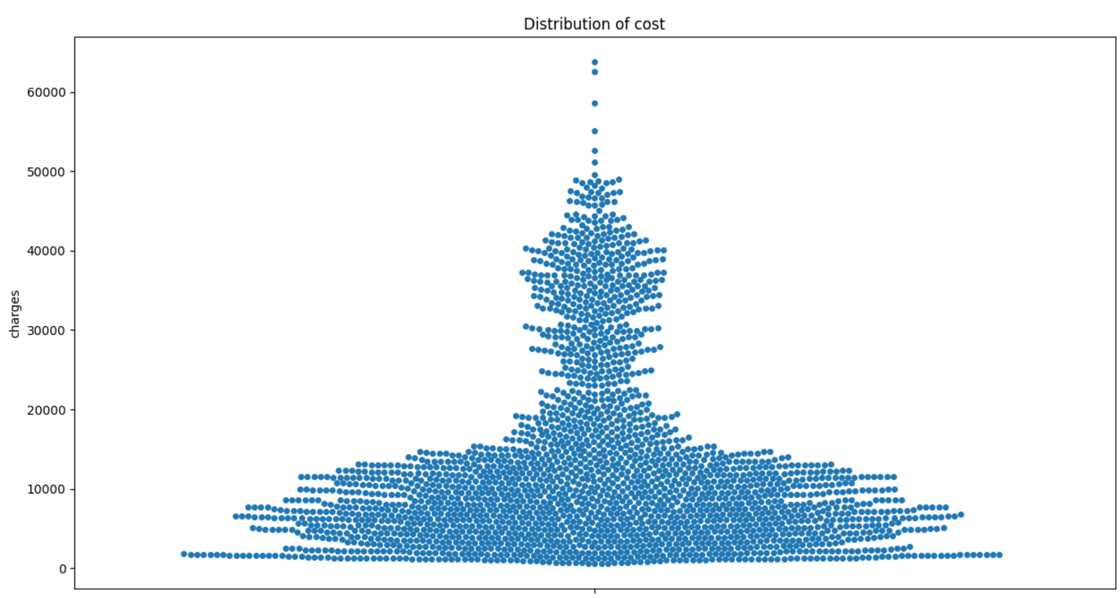


[145]:
```
# box and whisker plot
sns.boxplot(df['charges'])
```

[145]: <Axes: >



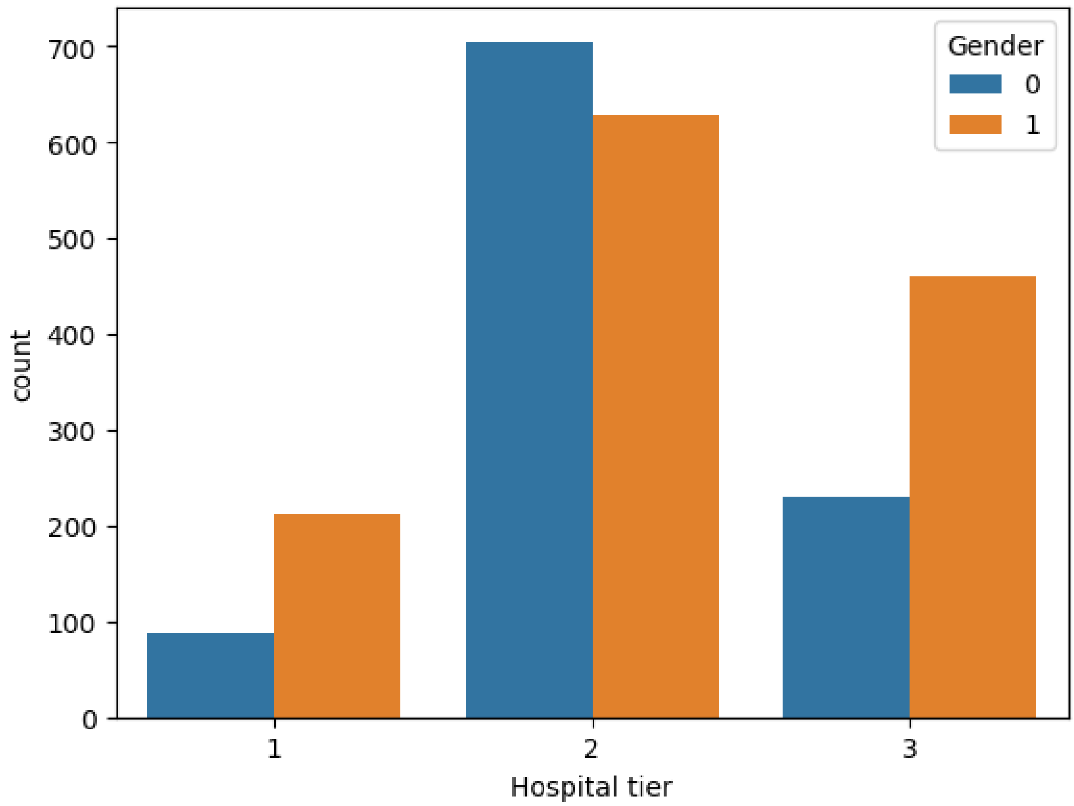
[148]:
```
# Swarm Plot
plt.figure(figsize=(15,8))
sns.swarmplot(df['charges'])
plt.title('Distribution of cost')
plt.show()
```



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

[147]:
```
sns.countplot(data = df,x = 'Hospital tier', hue = 'Gender')
```

[147]: <Axes: xlabel='Hospital tier', ylabel='count'>



- In above plot 0 indicates female and 1 indicates male.
- In Hospital tier 1 and 3 we can see that, the count of females are less than compared to the tier 2.

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

[149]:
```
df[df['Hospital tier']==1].charges.median()
```

[149]: 32097.434999999998

[150]:
```python
df[df['Hospital tier']==2].charges.median()
```

[150]: 7168.76

[151]:
```python
df[df['Hospital tier']==3].charges.median()
```

[151]: 10676.83

[153]:
```python
!pip install plotly
```

```
Collecting plotly
  Downloading plotly-5.14.1-py2.py3-none-any.whl (15.3 MB)
     ---------------------------------------- 0.0/15.3 MB ? eta -:--:--
      --------------------------------------- 0.2/15.3 MB 11.5 MB/s eta 0:00:02
     - -------------------------------------- 0.4/15.3 MB 4.4 MB/s eta 0:00:04
     -- ------------------------------------- 0.8/15.3 MB 5.7 MB/s eta 0:00:03
     --- ------------------------------------ 1.4/15.3 MB 7.5 MB/s eta 0:00:02
     ----- ---------------------------------- 1.9/15.3 MB 8.2 MB/s eta 0:00:02
     ----- ---------------------------------- 2.1/15.3 MB 7.4 MB/s eta 0:00:02
     ----- ---------------------------------- 2.1/15.3 MB 7.4 MB/s eta 0:00:02
     ------- -------------------------------- 3.0/15.3 MB 6.9 MB/s eta 0:00:02
     --------- ------------------------------ 3.6/15.3 MB 7.4 MB/s eta 0:00:02
     ---------- ----------------------------- 4.2/15.3 MB 7.6 MB/s eta 0:00:02
     ---------- ----------------------------- 4.2/15.3 MB 7.1 MB/s eta 0:00:02
     ------------ --------------------------- 4.8/15.3 MB 7.2 MB/s eta 0:00:02
     ------------- -------------------------- 5.1/15.3 MB 7.2 MB/s eta 0:00:02
     -------------- ------------------------- 5.7/15.3 MB 7.2 MB/s eta 0:00:02
     --------------- ------------------------ 6.1/15.3 MB 7.5 MB/s eta 0:00:02
     ---------------- ----------------------- 6.4/15.3 MB 7.2 MB/s eta 0:00:02
     ---------------- ----------------------- 6.5/15.3 MB 6.9 MB/s eta 0:00:02
     ------------------- -------------------- 7.6/15.3 MB 7.6 MB/s eta 0:00:02
     -------------------- ------------------- 7.9/15.3 MB 7.5 MB/s eta 0:00:01
     ---------------------- ----------------- 8.5/15.3 MB 7.7 MB/s eta 0:00:01
     ---------------------- ----------------- 8.7/15.3 MB 7.7 MB/s eta 0:00:01
     ----------------------- ---------------- 9.1/15.3 MB 7.7 MB/s eta 0:00:01
     ------------------------ --------------- 9.5/15.3 MB 7.6 MB/s eta 0:00:01
     ------------------------- -------------- 9.9/15.3 MB 7.5 MB/s eta 0:00:01
     -------------------------- ------------- 10.2/15.3 MB 7.5 MB/s eta 0:00:01
     --------------------------- ------------ 10.7/15.3 MB 7.8 MB/s eta 0:00:01
     ---------------------------- ----------- 11.1/15.3 MB 7.8 MB/s eta 0:00:01
     ----------------------------- ---------- 11.5/15.3 MB 7.7 MB/s eta 0:00:01
     ------------------------------ --------- 11.8/15.3 MB 7.5 MB/s eta 0:00:01
     ------------------------------- -------- 12.3/15.3 MB 7.4 MB/s eta 0:00:01
     -------------------------------- ------- 12.6/15.3 MB 8.0 MB/s eta 0:00:01
     --------------------------------- ------ 13.0/15.3 MB 7.8 MB/s eta 0:00:01
     ---------------------------------- ----- 13.3/15.3 MB 7.7 MB/s eta 0:00:01
     ----------------------------------- ---- 13.7/15.3 MB 7.7 MB/s eta 0:00:01
```

```
   ------------------------------------ --- 14.0/15.3 MB 7.5 MB/s eta 0:00:01
   ------------------------------------- -- 14.4/15.3 MB 7.5 MB/s eta 0:00:01
   -------------------------------------- - 14.7/15.3 MB 7.9 MB/s eta 0:00:01
   ---------------------------------------  15.1/15.3 MB 7.6 MB/s eta 0:00:01
   ---------------------------------------  15.3/15.3 MB 7.7 MB/s eta 0:00:01
   ---------------------------------------  15.3/15.3 MB 7.7 MB/s eta 0:00:01
   ---------------------------------------  15.3/15.3 MB 7.7 MB/s eta 0:00:01
   ---------------------------------------  15.3/15.3 MB 7.7 MB/s eta 0:00:01
   ---------------------------------------  15.3/15.3 MB 7.7 MB/s eta 0:00:01
   ---------------------------------------  15.3/15.3 MB 7.7 MB/s eta 0:00:01
   ---------------------------------------- 15.3/15.3 MB 6.2 MB/s eta 0:00:00
Requirement already satisfied: packaging in c:\users\91805\anaconda1\lib\site-
packages (from plotly) (23.0)
Collecting tenacity>=6.2.0
  Downloading tenacity-8.2.2-py3-none-any.whl (24 kB)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.14.1 tenacity-8.2.2
```

[155]:
```
!pip install --upgrade plotly
```

```
Requirement already satisfied: plotly in c:\users\91805\anaconda1\lib\site-
packages (5.14.1)
Requirement already satisfied: tenacity>=6.2.0 in
c:\users\91805\anaconda1\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: packaging in c:\users\91805\anaconda1\lib\site-
packages (from plotly) (23.0)
```

[163]:
```
import plotly.graph_objects as go
import pandas as pd

df1 = df1 = pd.DataFrame(dict(
    r=[32097.434999999998, 7168.76, 10676.83],
    theta=['Tier 1', 'Tier 2', 'Tier 3']
))

df2 = pd.concat([df1, df1.iloc[0]], ignore_index=True)

fig = go.Figure(data=go.Scatterpolar(
    r=df2['r'],
    theta=df2['theta'],
    fill='toself'
))

fig.update_layout(polar=dict(radialaxis=dict(visible=True, range=[0, 35000])))

fig.show()
```

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

[166]:
```
city_freq = df["City tier"].value_counts().rename_axis('City&hospital_tier').
↪reset_index(name='city_counts')
```

[167]:
```
hospital_freq = df["Hospital tier"].value_counts().
↪rename_axis('City&hospital_tier').reset_index(name='hospital_counts')
```

[168]:
```
freq_table = pd.merge(city_freq, hospital_freq, on = 'City&hospital_tier')
```

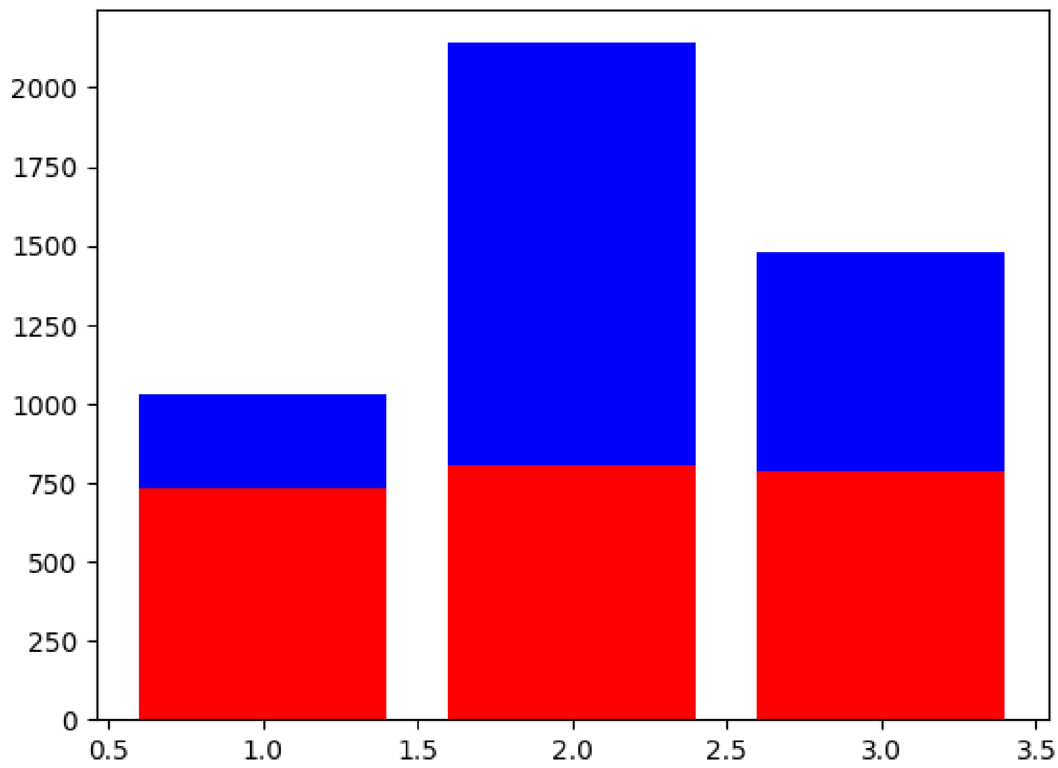
[169]:
```
freq_table
```

[169]:

	City&hospital_tier	city_counts	hospital_counts
0	2	807	1334
1	3	789	691
2	1	729	300

[170]:
```
x = freq_table['City&hospital_tier']
y1 = freq_table['city_counts']
y2 = freq_table['hospital_counts']

# plot bars in stack manner
plt.bar(x, y1, color='r')
plt.bar(x, y2, bottom=y1, color='b')
plt.show()
```



0.13 13. Test the following null hypotheses:

- a. The average hospitalization costs for the three types of hospitals are not significantly different
- b. The average hospitalization costs for the three types of cities are not significantly different
- c. The average hospitalization cost for smokers is not significantly different from the average cost for nonsmokers
- d. Smoking and heart issues are independent

[175]:
```
from scipy.stats import ttest_1samp
import scipy.stats as stats
```

a. The average hospitalization costs for the three types of hospitals are not significantly different

[176]:
```
print('Null Hypothesis => Average hospitalization costs for the three types of␣
↪hospitals are not significantly different.')
f_val, p_val = stats.f_oneway(df[df['Hospital tier'] == 'tier,1']['charges'],
                              df[df['Hospital tier'] == 'tier,2']['charges'],
                              df[df['Hospital tier'] == 'tier,3']['charges'])
print('P-value :',p_val)
```

```
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
```

We already found the median cost of tier 1 hospitals: 32097.434999999998, median cost of tier 2 hospitals: 7168.76 and .median cost of tier 3 hospitals: 10676.83. Interpretation H0: the distributions of all samples are equal and H1: the distributions of one or more samples are not equal

[177]:
```
from scipy.stats import friedmanchisquare
data1 = [32097.43]
data2 = [7168.76]
data3 = [10676.83]
stat, p = friedmanchisquare(data1, data2, data3)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
```

```
stat=2.000, p=0.368
Probably the same distribution
```

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

[178]:
```
print("median cost of tier 1 city:", df[df["City tier"]==1].charges.median())
print("median cost of tier 2 city:", df[df["City tier"]==2].charges.median())
print("median cost of tier 3 city:", df[df["City tier"]==3].charges.median())
```

```
median cost of tier 1 city: 10027.15
median cost of tier 2 city: 8968.33
median cost of tier 3 city: 9880.07
```

[179]:
```
data1 = [10027.15]
data2 = [8968.33]
data3 = [9880.07]
stat, p = friedmanchisquare(data1, data2, data3)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
```

```
stat=2.000, p=0.368
Probably the same distribution
```

[182]:
```python
print('Null Hypothesis => Average hospitalization costs for the three types of␣
↪cities are not significantly different.')
f_val, p_val = stats.f_oneway(df[df['City tier'] == 'tier,1']['charges'],
                              df[df['City tier'] == 'tier,2']['charges'],
                              df[df['City tier'] == 'tier,3']['charges'])
print('P-value :',p_val)
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

[180]:
```python
print("median cost of smoker:", df[df["smoker"]==1].charges.median())
print("median cost of non smoker:", df[df["smoker"]==0].charges.median())
```

```
median cost of smoker: 34125.475
median cost of non smoker: 7537.16
```

[186]:
```python
from scipy.stats import kruskal
data1 = [34125.475]
data2 = [7537.16]
stat, p = kruskal(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably the same distribution')
else:
    print('Probably different distributions')
```

```
stat=1.000, p=0.317
Probably the same distribution
```

[184]:
```python
print('Null Hypothesis => Average hospitalization costs for smokers is not␣
↪significantly different from the average cost for nonsmokers.')
t_val, p_val = stats.ttest_ind(df[df['smoker'] == 'yes']['charges'],
                               df[df['smoker'] == 'no']['charges'])

print('P-value :',p_val)
if p_val < 0.05:
```

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

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

- Interpretation H0: the two samples are independent. H1: there is a dependency between the samples.

d. Smoking and heart issues are independent

[187]:
```
# d. Smoking and heart issues are independent
from scipy.stats import chi2_contingency
table = [[df["Heart Issues"].value_counts()],[df["smoker"].value_counts()]]
stat, p, dof, expected = chi2_contingency(table)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably independent')
else:
    print('Probably dependent')
```

```
stat=191.145, p=0.000
Probably dependent
```

[190]:
```
from scipy.stats import chi2_contingency
contingency_table = pd.crosstab(df['smoker'], df['Heart Issues'])
chi2, p, dof, expected = chi2_contingency(contingency_table)
print(f'P-value = {p}')
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 = 0.7694797581780767
Accept null hypothesis, Smoking and heart issues are independent.
```

[ ]:
```
```

1 Machine Learning

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

[191]:
```
df.drop(["Customer ID","State ID",'name', 'year', 'month', 'date', 'DOB'],␣
↪inplace=True, axis=1)
```

[192]:
```
df.head()
```

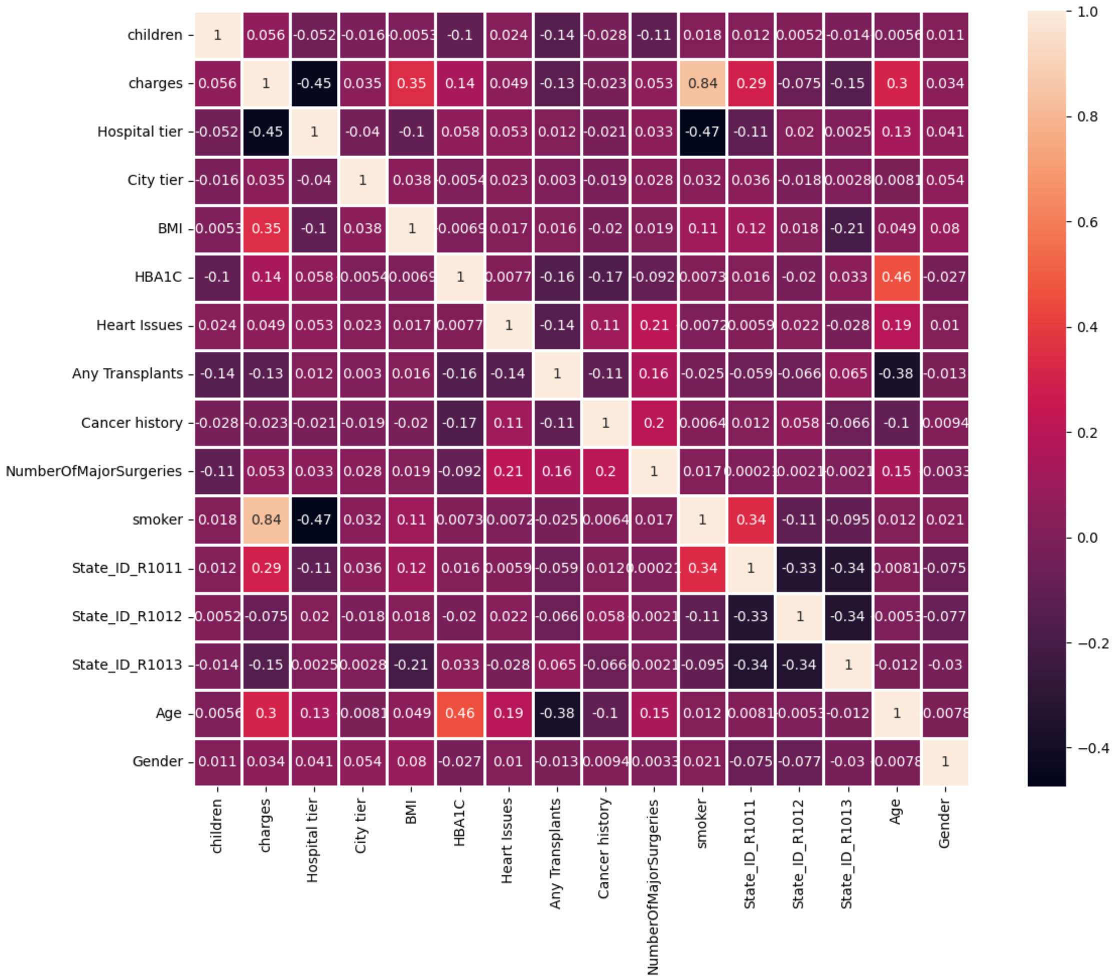
[192]:
```
   children  charges  Hospital tier  City tier    BMI  HBA1C  Heart Issues
0         0   563.84              2          3  17.58   4.51             0  \
1         0   570.62              2          1  17.60   4.39             0
2         0   600.00              2          1  16.47   6.35             0
3         0   604.54              3          3  17.70   6.28             0
4         0   637.26              3          3  22.34   5.57             0

   Any Transplants  Cancer history  NumberOfMajorSurgeries  smoker
0                0               0                       1       0  \
1                0               0                       1       0
2                0               1                       1       0
3                0               0                       1       0
4                0               0                       1       0

   State_ID_R1011  State_ID_R1012  State_ID_R1013  Age  Gender
0           False           False            True   30       1
1           False           False            True   30       1
2           False           False            True   29       0
3           False           False            True   30       1
4           False           False            True   24       1
```

[193]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),square=True,annot=True,linewidths=1)
```

[193]: <Axes: >



1.0.2 2. Develop and evaluate the final model using regression with a stochastic gradient descent optimizer. Also, ensure that you apply all the following suggestions:

• 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

[194]:
```
from sklearn.model_selection import train_test_split
```

[196]:
```
x = df.drop(["charges"], axis=1)
y = df['charges']
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=.
 ↪20,random_state=10)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
from sklearn.linear_model import SGDRegressor
```

[197]:
```
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
```

[197]:
```
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)
```

[198]:
```
model_cv.best_params_
```

[198]:
```
{'alpha': 4.0, 'penalty': 'l1'}
```

[200]:
```
sgd = SGDRegressor(alpha= 100, penalty= 'l1')
```

[201]:
```
sgd.fit(x_train, y_train)
```

[201]:
```
SGDRegressor(alpha=100, penalty='l1')
```

[202]:
```python
sgd.score(x_test, y_test)
```

[202]: 0.8711282239533751

[203]:
```python
y_pred = sgd.predict(x_test)
```

[205]:
```python
from sklearn.metrics import mean_squared_error, mean_absolute_error
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)
```

[206]:
```python
print("MAE:", sgd_mae)
print("MSE:", sgd_mse)
print("RMSE:", sgd_rmse)
```

```
MAE: 2681.183902514782
MSE: 18756962.940102503
RMSE: 9378481.470051251
```

[207]:
```python
importance = sgd.coef_
pd.DataFrame(importance, index = x.columns, columns=['Feature_imp'])
```

[207]:

	Feature_imp
children	324.625902
Hospital tier	-1135.536560
City tier	0.000000
BMI	2697.227677
HBA1C	138.011477
Heart Issues	0.000000
Any Transplants	0.000000
Cancer history	0.000000
NumberOfMajorSurgeries	0.000000
smoker	8998.720349
State_ID_R1011	0.000000
State_ID_R1012	0.000000
State_ID_R1013	-218.159993
Age	3401.726864
Gender	0.000000

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

[208]:
```python
from sklearn.ensemble import RandomForestRegressor
```

[209]:
```python
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
```

[210]:
```
rf.fit(x_train, y_train)
```

[210]:
```
RandomForestRegressor(n_estimators=1000, random_state=42)
```

[211]:
```
score = rf.score(x_test,y_test)
score
```

[211]:
```
0.901740825808739
```

[212]:
```
y_pred = rf.predict(x_test)
```

[213]:
```
rf_mae = mean_absolute_error(y_test, y_pred)
```

[214]:
```
rf_mae
```

[214]:
```
1940.0160889462331
```

Extreme gradient boosting

[215]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

[216]:
```
gbr = GradientBoostingRegressor(n_estimators = 1000, random_state = 42)
```

[217]:
```
gbr.fit(x_train, y_train)
```

[217]:
```
GradientBoostingRegressor(n_estimators=1000, random_state=42)
```

[218]:
```
score = gbr.score(x_test,y_test)
score
```

[218]:
```
0.8650084426981199
```

[219]:
```
y_pred = gbr.predict(x_test)
```

[220]:
```
gbr_mae = mean_absolute_error(y_test, y_pred)
gbr_mae
```

[220]:
```
2424.08578378777
```

- Since Mean Absolute Eroor of Random Forest is less than extreme gradient boosting, Random Forest algorithm works well.

1.1.1 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.

[222]:
```
df.columns
```

[222]:
```
Index(['children', 'charges', 'Hospital tier', 'City tier', 'BMI', 'HBA1C',
       'Heart Issues', 'Any Transplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'smoker', 'State_ID_R1011', 'State_ID_R1012',
       'State_ID_R1013', 'Age', 'Gender'],
      dtype='object')
```

[223]:
```
date = str(19881228)
date1 = pd.to_datetime(date, format = "%Y%m%d")
```

[226]:
```
current_date = dt.datetime.now()
current_date
```

[226]:
```
datetime.datetime(2023, 5, 25, 18, 55, 19, 779951)
```

[227]:
```
age = (current_date - date1)
age
```

[227]:
```
Timedelta('12566 days 18:55:19.779951')
```

[228]:
```
age = int(12566/365)
age
```

[228]:
```
34
```

- So the age of Christopher, Ms. Jayna is 34

[229]:
```
height_m = 170/100
height_sq = height_m*height_m
BMI = 85/height_sq
np.round(BMI,2)
```

[229]:
```
29.41
```

- BMI is 29.41

[268]:
```
df.columns
```

[268]:
```
Index(['children', 'Hospital tier', 'City tier', 'BMI', 'HBA1C',
       'Heart Issues', 'Any Transplants', 'Cancer history',
       'NumberOfMajorSurgeries', 'smoker', 'State_ID_R1011', 'State_ID_R1012',
       'State_ID_R1013', 'age', 'gender'],
      dtype='object')
```

[270]:
```python
list = [[2,1,1,29.41,5.8,0,0,0,0,1,1,0,0,34,0]]
```

[271]:
```python
df = pd.DataFrame(list, columns = ['children','Hospital tier', 'City tier',␣
 ↪'BMI', 'HBA1C','Heart Issues', 'Any Transplants',
                              'Cancer history','NumberOfMajorSurgeries',␣
 ↪'smoker', 'State_ID_R1011', 'State_ID_R1012',
                              'State_ID_R1013', 'age', 'gender'] )
df
```

[271]:
```
   children  Hospital tier  City tier    BMI  HBA1C  Heart Issues
0         2              1          1  29.41    5.8             0  \

   Any Transplants  Cancer history  NumberOfMajorSurgeries  smoker
0                0               0                       0       1  \

   State_ID_R1011  State_ID_R1012  State_ID_R1013  age  gender
0               1               0               0   34       0
```

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

[274]:
```python
Hospital_cost = []
```

[275]:
```python
Cost1 = sgd.predict(df)
Hospital_cost.append(Cost1)
Cost2 = rf.predict(df)
Hospital_cost.append(Cost2)
Cost3 = gbr.predict(df)
Hospital_cost.append(Cost3)
avg_cost = np.mean(Hospital_cost)
avg_cost
```

[275]:
```
104554.8763667323
```

- The mean of the five models' predicted avaerage value is 104554.8763667323

[ ]:
```python

```