1. Collate the 2 excel files to have all the information at one place. Check for missing values and duplicates before joining the 2 datasets.

```
# All libraries/functions required

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
import plotly.express as px
from statsmodels.formula.api import ols
import statsmodels.api  as sm
import scipy.stats as stats
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import KFold, StratifiedKFold,
RandomizedSearchCV, train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error as mse, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.pipeline import Pipeline
from xgboost import XGBRegressor

address = '/home/labsuser/capstone'

hosp = pd.read_csv(address + '/Hospitalisation details.csv')
medic = pd.read_csv(address + '/Medical Examinations.csv')
names = pd.read_excel(address + '/Names.xlsx')

# hosp = pd.read_excel('/content/drive/MyDrive/Colab
Notebooks/Hospitalisation details.xlsx')
# medic = pd.read_excel('/content/drive/MyDrive/Colab
Notebooks/Medical Examinations.xlsx')
```

Data inspection using .info()

```
hosp.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
```

```
medic.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
```

```
names.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
```

```
master_data = pd.merge(hosp, medic, how = 'inner', on = 'Customer ID')
```

```
master_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2335 entries, 0 to 2334
Data columns (total 16 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
dtypes: float64(3), int64(2), object(11)
memory usage: 310.1+ KB
```

```
master_data = master_data.merge(names,on='Customer ID')
```

```
master_data.columns = master_data.columns.str.lower()
master_data.columns = master_data.columns.str.replace(' ', '_')
master_data.columns
```

```
Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name'],
      dtype='object')
```

2. The data seems to have trivial values in a few variables. These are “?”. Find the percentage of rows which have such value (“?”) in any column. Delete such rows in case you don’t lose significant information.

```
(master_data == '?').sum()
```

```
customer_id               0
year                      2
month                     3
date                      0
children                  0
charges                   0
hospital_tier             1
city_tier                 1
state_id                  2
bmi                       0
hba1c                     0
heart_issues              0
any_transplants           0
cancer_history            0
numberofmajorsurgeries    0
smoker                    2
name                      0
dtype: int64
```

```
# replacing '?' with np.NA for easy access and removal
```

```
miss_perc = (master_data == '?').sum(axis = 1)/master_data.shape[1] *
100
miss_perc[miss_perc > 0]
```

```
11       5.882353
13       5.882353
17      11.764706
542      5.882353
1046     5.882353
1049     5.882353
1700     5.882353
1775     5.882353
2165     5.882353
2332     5.882353
dtype: float64
```

```
miss_perc[miss_perc>0].index
```

```
Int64Index([11, 13, 17, 542, 1046, 1049, 1700, 1775, 2165, 2332],
dtype='int64')
```

```
miss_perc_col = (master_data == '?').sum(axis =
0)/master_data.shape[0] * 100
miss_perc_col.sort_values(ascending= False)
```

```
month                      0.128480
state_id                   0.085653
year                       0.085653
smoker                     0.085653
city_tier                  0.042827
hospital_tier              0.042827
date                       0.000000
children                   0.000000
charges                    0.000000
name                       0.000000
bmi                        0.000000
hba1c                      0.000000
heart_issues               0.000000
any_transplants            0.000000
cancer_history             0.000000
numberofmajorsurgeries     0.000000
customer_id                0.000000
dtype: float64
```

```
master_noq = master_data.drop(index = miss_perc[miss_perc>0].index)
master_noq.shape
```

```
(2325, 17)
```

```
master_noq.isna().sum()
```

```
customer_id                 0
year                        0
month                       0
date                        0
children                    0
charges                     0
hospital_tier               0
city_tier                   0
state_id                    0
bmi                         0
hba1c                       0
heart_issues                0
any_transplants             0
cancer_history              0
numberofmajorsurgeries      0
smoker                      0
name                        0
dtype: int64
```

3. The data has nominal and ordinal categorical variables. How are you going to incorporate these variables in the next steps of modelling. Use necessary transformation methods to deal with nominal and ordinal types of data.

```
master_noq[['city_tier', 'hospital_tier']]
```

```
     city_tier hospital_tier
0     tier - 3      tier - 2
1     tier - 1      tier - 2
2     tier - 1      tier - 2
3     tier - 3      tier - 3
4     tier - 3      tier - 3
...        ...           ...
2329  tier - 3      tier - 1
2330  tier - 2      tier - 1
2331  tier - 3      tier - 1
2333  tier - 3      tier - 2
2334  tier - 3      tier - 1

[2325 rows x 2 columns]
```

```
# Using ordinalencoder to deal with ordinal categorical variables - city tier and hospital tier

ordinal = OrdinalEncoder(categories= [['tier - 3', 'tier - 2', 'tier - 1'],['tier - 3', 'tier - 2', 'tier - 1']])
master_noq[['city_tier_ord','hospital_tier_ord']] = ordinal.fit_transform(master_noq[['city_tier', 'hospital_tier']])

pd.crosstab(master_noq['city_tier_ord'],master_noq['city_tier'])
```

```
city_tier      tier - 1  tier - 2  tier - 3
city_tier_ord
```

```
0.0                     0         0       789
1.0                     0       807         0
2.0                   729         0         0
```

```
pd.crosstab(master_noq['hospital_tier_ord'],master_noq['hospital_tier'
])
```

```
hospital_tier      tier - 1  tier - 2  tier - 3
hospital_tier_ord
0.0                       0         0       691
1.0                       0      1334         0
2.0                     300         0         0
```

```
master_noq.head(3)
```

```
  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

  state_id    bmi  hba1c heart_issues any_transplants
cancer_history  \
0    R1013  17.58   4.51           No              No             No

1    R1013  17.60   4.39           No              No             No

2    R1013  16.47   6.35           No              No            Yes


  numberofmajorsurgeries smoker                    name  city_tier_ord
\
0                      1     No    German, Mr.  Aaron K            0.0

1                      1     No  Rosendahl, Mr.  Evan P            2.0

2                      1     No      Albano, Ms.  Julie            2.0


   hospital_tier_ord
0                1.0
1                1.0
2                1.0
```

4. State ID has around 16 states. The data does not have proportional representation of all the states. Also creating dummy variables corresponding to all the regions may lead to too many insignificant predictors. Nevertheless, only R1011, R1012 and R1013 are important to look deeper into. Keeping these ideas in mind, come up with a suitable strategy here.

```
vc = master_noq.state_id.value_counts()   # frequency of each category
vc[:3].index                              # picking top 3 most 
frequent categories

Index(['R1013', 'R1011', 'R1012'], dtype='object')

for i in vc[:3].index:
    var_name = 'state_id_' +i   # create name for the dummy varible
    print(var_name)
    master_noq[var_name] = 0    # giving a dummy value 0 to dummy 
variable
    master_noq.loc[master_noq.state_id == i,var_name] = 1  # replacing 
0 by 1 where state id is equal to category of the dummy variable


state_id_R1013
state_id_R1011
state_id_R1012

master_noq.state_id.value_counts()

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

# checking the no of records corresponding to R1013

master_noq['state_id_R1013'].value_counts()

0    1716
1     609
Name: state_id_R1013, dtype: int64
```

```
master_noq['state_id_R1012'].value_counts()
```

```
0    1753
1     572
Name: state_id_R1012, dtype: int64
```

5. Variable 'NumberOfMajorSurvalue_counts seems to have string values as well. You may want to clean this variable.

```
master_noq.numberofmajorsurgeries.unique()
```

```
array(['1', 'No major surgery', '2', '3'], dtype=object)
```

```
master_noq.loc[master_noq.numberofmajorsurgeries == 'No major
surgery','numberofmajorsurgeries' ] = 0

master_noq.numberofmajorsurgeries =
master_noq.numberofmajorsurgeries.astype(int)
```

6. Age seems to an important factor for this analysis. Based on date of birth information, calculate the age of the patients.

```
master_noq.year = master_noq.year.astype(int)

master_noq['age'] = 2022 - master_noq.year
```

7. Gender of the patient may be an important factor to decide the hospitalization cost. Salutation provided in the name of the beneficiary can be used to determine the gender. Create a new field for the gender of beneficiary.

```
master_noq['title'] =
master_noq.name.str.split('[,.]').str[1].str.strip()

master_noq.title.value_counts()
```

```
Mr     1160
Ms     1023
Mrs     142
Name: title, dtype: int64
```

```
master_noq['gender'] = 'female'
master_noq.loc[master_noq.title == 'Mr', 'gender'] = 'male'

master_noq.loc[master_noq.title == 'Mrs']
```

```
     customer_id  year month  date  children   charges
hospital_tier  \
24        Id2311  2001   Aug    19         0    964.71      tier - 3

172       Id2163  2004   Dec    27         0   1863.45      tier - 3

197       Id2138  2004   Jun    12         0   2094.10      tier - 3

328       Id2007  1993   Sep    25         0   3162.02      tier - 2
```

```
348       Id1987  2003   Dec     5         0   3300.70      tier - 2

...          ...   ...   ...   ...       ...       ...          ...

1790       Id545  1963   Jul     4         0  18208.34      tier - 1

1808       Id527  1963   Dec     6         0  18883.33      tier - 1

1811       Id524  1963   Oct    20         0  18954.56      tier - 1

1839       Id496  1966   Aug    10         0  19995.29      tier - 1

1848       Id487  1962   Jul     2         0  20354.50      tier - 3


     city_tier state_id    bmi  ...  smoker
name  \
24    tier - 2    R1013  25.19  ...      No        Keys, Mrs.
Kathleen
172   tier - 1    R1025  27.06  ...      No    Stanislav, Mrs.  Grace
H
197   tier - 2    R1025  27.74  ...      No        Padula, Mrs.
Lauren
328   tier - 3    R1013  25.61  ...      No     Martin, Mrs.  Kristen
M
348   tier - 2    R1025  30.54  ...      No  Mendez-Karr, Mrs.
Cynthia
...        ...      ...    ...  ...     ...                         ..
.
1790  tier - 2    R1026  44.20  ...      No      Shigezumi, Mrs.
Teiko
1808  tier - 1    R1026  46.19  ...      No      Hughey, Mrs.  Ashley
E
1811  tier - 1    R1026  46.40  ...      No      Rogers, Mrs.  Anita
L.
1839  tier - 3    R1026  51.74  ...      No       Oehlke, Mrs.
Jessica
1848  tier - 2    R1026  49.77  ...      No        Argall, Mrs.  Tara
R

     city_tier_ord hospital_tier_ord  state_id_R1013 state_id_R1011  \
24             1.0               0.0               1              0
172            2.0               0.0               0              0
197            1.0               0.0               0              0
328            0.0               1.0               1              0
348            1.0               1.0               0              0
...            ...               ...             ...            ...
1790           1.0               2.0               0              0
1808           2.0               2.0               0              0
```

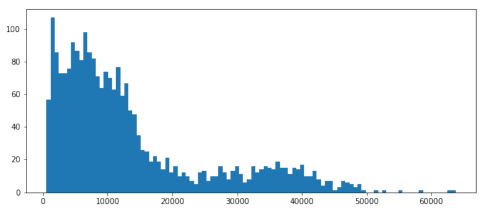
```
1811          2.0              2.0              0             0
1839          0.0              2.0              0             0
1848          1.0              0.0              0             0

     state_id_R1012  age  title  gender
24                0   21    Mrs  female
172               0   18    Mrs  female
197               0   18    Mrs  female
328               0   29    Mrs  female
348               0   19    Mrs  female
...             ...  ...    ...     ...
1790              0   59    Mrs  female
1808              0   59    Mrs  female
1811              0   59    Mrs  female
1839              0   56    Mrs  female
1848              0   60    Mrs  female

[142 rows x 25 columns]
```

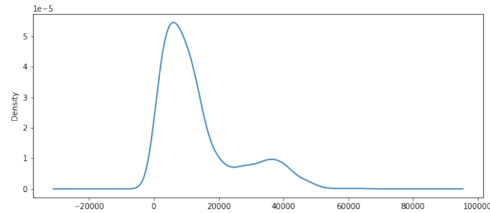
```
master_noq['gender']
```

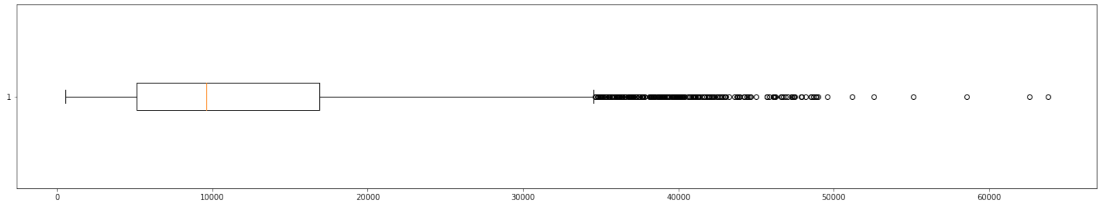
```
0         male
1         male
2       female
3         male
4         male
         ...
2329      male
2330    female
2331    female
2333      male
2334    female
Name: gender, Length: 2325, dtype: object
```

8. Visualize the distribution of cost using histogram, box and whisker and swarm plot. How the distribution is different across gender and different tiers of hospitals. Share your observation.

```
plt.figure(figsize = (25,10))
grid = plt.GridSpec(2, 2, wspace=0.4, hspace=0.3)
plt.subplot(grid[0, 0])
plt.hist(master_noq.charges, bins = 100)
plt.subplot(grid[0, 1])
master_noq.charges.plot.kde()
plt.subplot(grid[1, :])
plt.boxplot(master_noq.charges, vert = False)
plt.show()
```

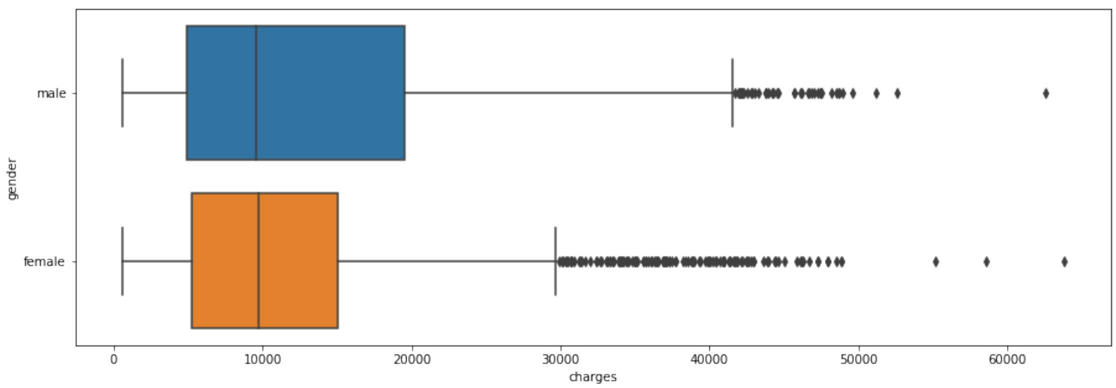






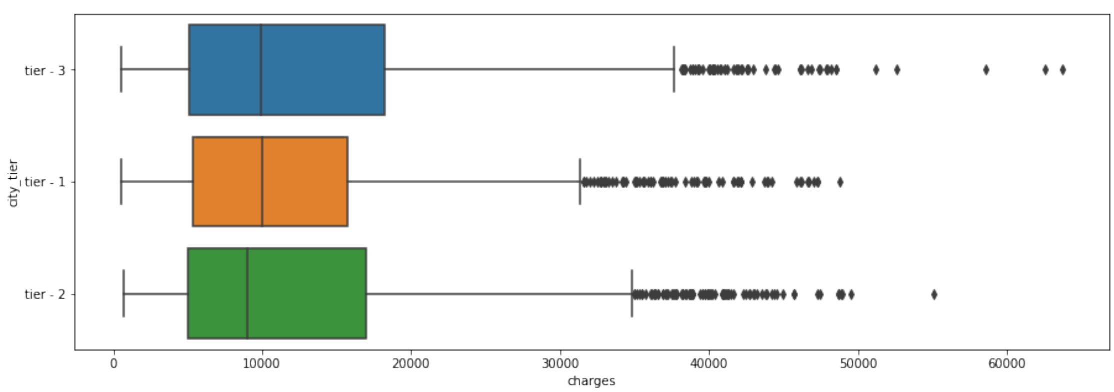
WRT gender

```
plt.figure(figsize = (15,5))
sns.boxplot(x = "charges",y = "gender", data = master_noq)
plt.show()
```



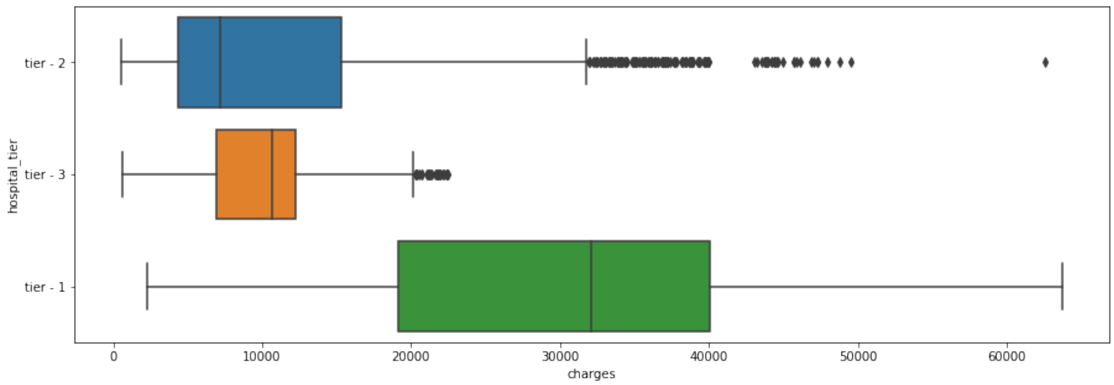
WRT city tier

```
plt.figure(figsize = (15,5))
sns.boxplot(x = "charges",y = "city_tier", data = master_noq)
plt.show()
```



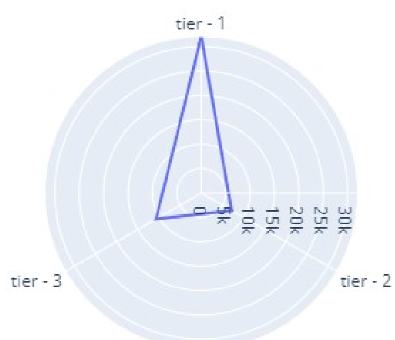
WRT Hospital tier

```
plt.figure(figsize = (15,5))
sns.boxplot(x = "charges",y = "hospital_tier", data = master_noq)
plt.show()
```



9. Create a radar chart to showcase the median hospitalization cost across different tiers of hospitals.

```
median = master_noq.groupby('hospital_tier')
[['charges']].median().reset_index()

fig = px.line_polar(median, r='charges', theta='hospital_tier',
line_close=True)
fig.show()
```

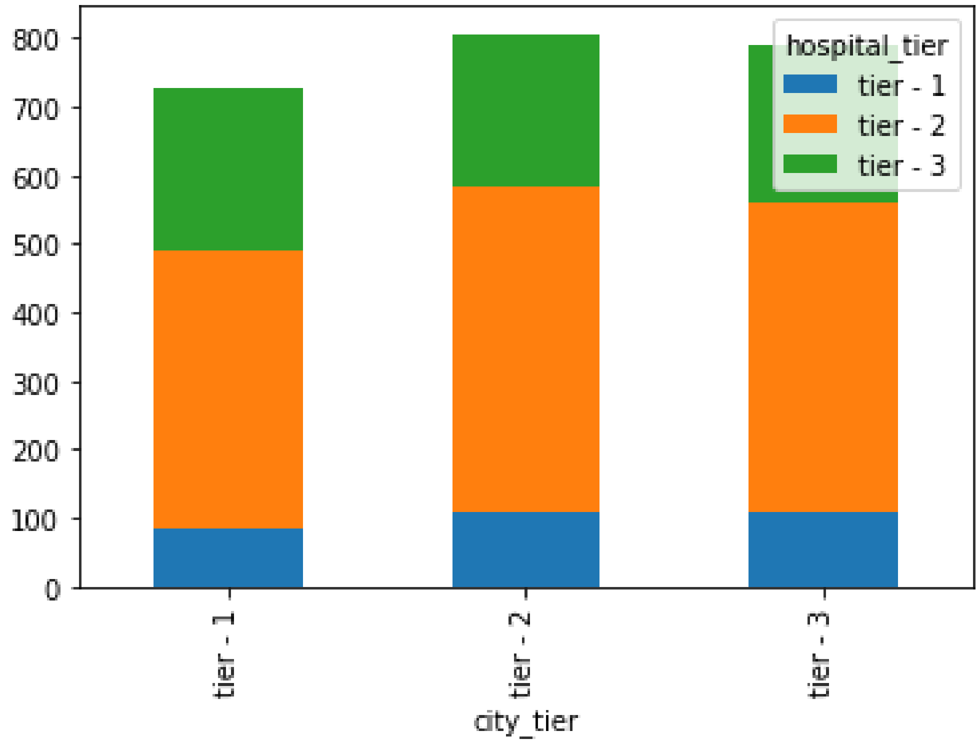


10. Create a frequency table and hence a stacked bar-chart to visualize the count of people in different tiers of cities and hospitals.

```
pd.crosstab(master_noq.city_tier, master_noq.hospital_tier)

hospital_tier  tier - 1  tier - 2  tier - 3
city_tier
tier - 1             85       403       241
```

```
tier - 2              106       479       222
tier - 3              109       452       228
```

```
pd.crosstab(master_noq.city_tier,
master_noq.hospital_tier).plot.bar(stacked = True)
plt.show()
```



11. Test the following null hypotheses:

```
- Average hospitalization cost across the 3 types of hospitals is not
significantly different
- Average hospitalization cost across the 3 types of cities is not
significantly different
- Average hospitalization cost for smokers is not significantly
different than non-smokers
- Smoking and Hearth issues are independent
```

H0 : Average hospitalization cost across the 3 types of hospitals is not significantly different

```
mod = ols('charges ~ hospital_tier', data = master_noq).fit()
res = sm.stats.anova_lm(mod)
res
```

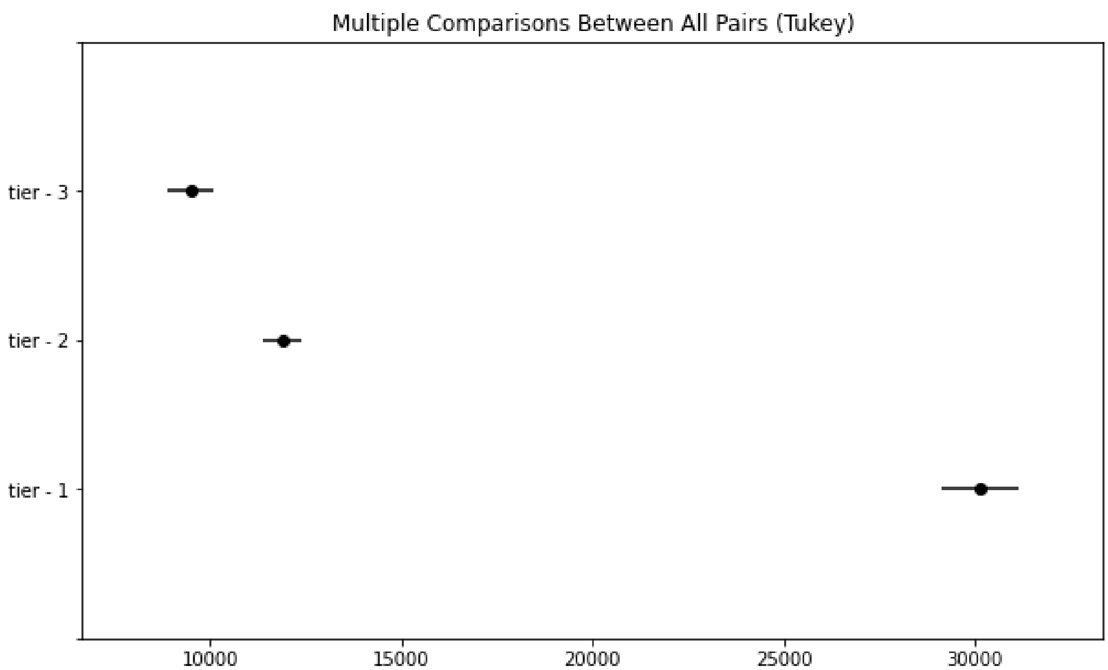
```
                   df        sum_sq       mean_sq           F
PR(>F)
hospital_tier     2.0  9.763011e+10  4.881505e+10  493.989566
1.773822e-179
Residual       2322.0  2.294554e+11  9.881799e+07         NaN
NaN
```

```
res_tukey = pairwise_tukeyhsd(master_noq.charges,
master_noq.hospital_tier)
res_tukey.summary()
```

```
<class 'statsmodels.iolib.table.SimpleTable'>
```

```
res_tukey.plot_simultaneous()
plt.show()
```

```
/usr/local/lib/python3.7/site-packages/statsmodels/sandbox/stats/
multicomp.py:775: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



H0 = Average hospitalization cost across the 3 types of cities is not significantly different

```
mod = ols('charges ~ city_tier', data = master_noq).fit()
res = sm.stats.anova_lm(mod)
res
```

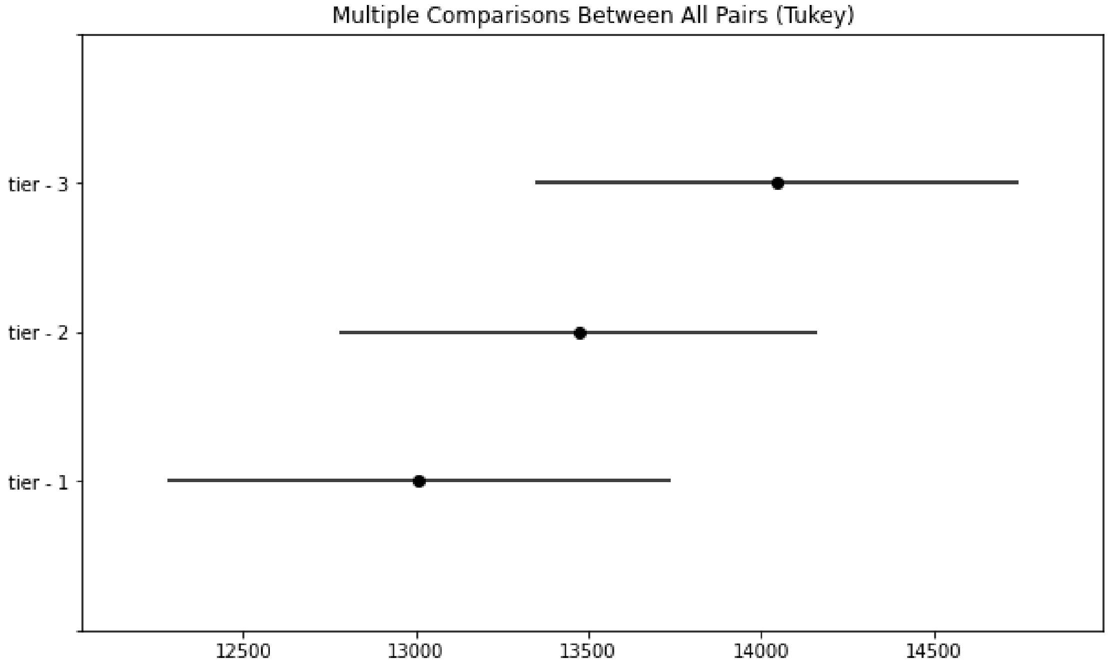
```
                df        sum_sq       mean_sq         F    PR(>F)
city_tier      2.0  4.092192e+08  2.046096e+08  1.454356  0.233763
Residual    2322.0  3.266763e+11  1.406874e+08       NaN       NaN
```

```
res_tukey = pairwise_tukeyhsd(master_noq.charges,
master_noq.city_tier)
res_tukey.summary()
```

```
<class 'statsmodels.iolib.table.SimpleTable'>
```

```
res_tukey.plot_simultaneous()
plt.show()
```

```
/usr/local/lib/python3.7/site-packages/statsmodels/sandbox/stats/
multicomp.py:775: UserWarning:

FixedFormatter should only be used together with FixedLocator
```



H0: Average hospitalization cost for smokers is not significantly different than non-smokers

```
sample1 = master_noq.loc[master_noq.smoker == 'yes', 'charges']
sample2 = master_noq.loc[master_noq.smoker != 'yes', 'charges']
stats.ttest_ind(sample1, sample2)
```

```
Ttest_indResult(statistic=74.15560699695726, pvalue=0.0)
```

H0 : Smoking and Heart issues are independent4

```
observed_table = pd.crosstab(master_noq.smoker,
master_data.heart_issues)
```

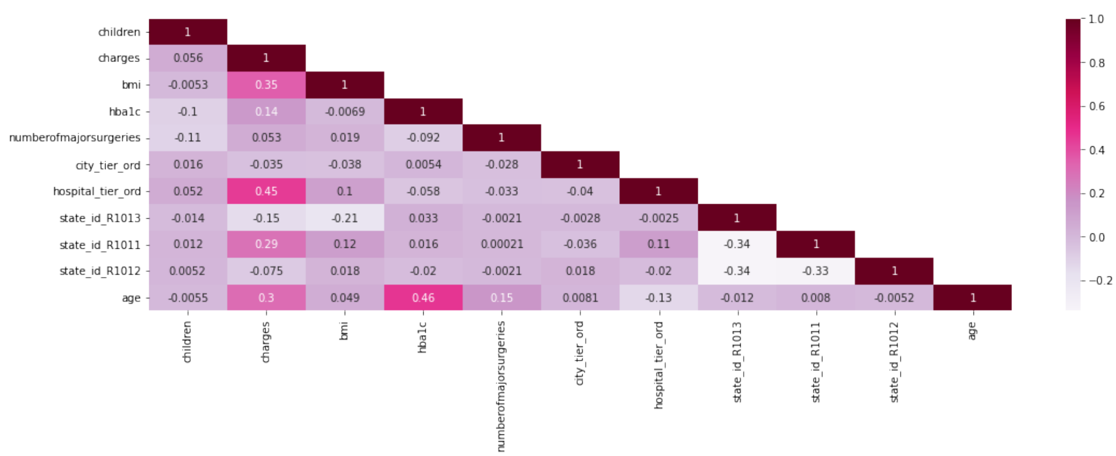
```
observed_table
```

```
heart_issues    No  yes
smoker
No            1108  731
yes            297  189
```

```
chi, p, df, expected = stats.chi2_contingency(observed_table)

chi, p, df, expected

(0.08588150449910657,
 0.7694797581780767,
 1,
 array([[1111.30967742,  727.69032258],
        [ 293.69032258,  192.30967742]]))
```

12. Check the correlation between predictors to identify highly correlated predictors. Visualize using a heatmap.

```
master_noq.columns

Index(['customer_id', 'year', 'month', 'date', 'children', 'charges',
       'hospital_tier', 'city_tier', 'state_id', 'bmi', 'hba1c',
       'heart_issues', 'any_transplants', 'cancer_history',
       'numberofmajorsurgeries', 'smoker', 'name', 'city_tier_ord',
       'hospital_tier_ord', 'state_id_R1013', 'state_id_R1011',
       'state_id_R1012', 'age', 'title', 'gender'],
      dtype='object')

data = master_noq.drop(columns = ['name', 'year', 'month',
'date','hospital_tier',
       'city_tier', 'state_id' , 'title'])

corr_plot = data.select_dtypes(exclude='object').corr()
ma = np.ones_like(corr_plot)
ma[np.tril_indices_from(ma)] = 0

plt.figure(figsize = (18,5))
sns.heatmap(corr_plot, annot= True , mask = ma, cmap='PuRd')
plt.show()
```




```
data_2 = pd.get_dummies(data, drop_first=True)
data_2.reset_index(drop=True, inplace = True)

data_2.head()
```

```
   children  charges    bmi  hba1c  numberofmajorsurgeries  
city_tier_ord  \
0         0   563.84  17.58   4.51                       1  
0.0   
1         0   570.62  17.60   4.39                       1  
2.0   
2         0   600.00  16.47   6.35                       1  
2.0   
3         0   604.54  17.70   6.28                       1  
0.0   
4         0   637.26  22.34   5.57                       1  
0.0   

   hospital_tier_ord  state_id_R1013  state_id_R1011  
state_id_R1012  ...  \
0                1.0               1               0  
0  ...   
1                1.0               1               0  
0  ...   
2                1.0               1               0  
0  ...   
3                0.0               1               0  
0  ...   
4                0.0               1               0  
0  ...   

   customer_id_Id995  customer_id_Id996  customer_id_Id997  
customer_id_Id998  \
0                  0                  0                  0  
0   
1                  0                  0                  0  
0   
2                  0                  0                  0  
0   
3                  0                  0                  0  
0   
4                  0                  0                  0  
0   

   customer_id_Id999  heart_issues_yes  any_transplants_yes  \
0                  0                 0                    0   
1                  0                 0                    0   
2                  0                 0                    0   
3                  0                 0                    0   
4                  0                 0                    0   

   cancer_history_Yes  smoker_yes  gender_male  
0                   0           0            1  
1                   0           0            1  
2                   1           0            0  
```

```
3                    0            0             1
4                    0            0             1

[5 rows x 2340 columns]
```

```
# rearrange data to put 'charges' as first column or last
model_data = data_2.drop(columns = 'charges')
model_data.head()
model_data['charges'] = data_2.charges
model_data.head()
```

```
   children    bmi  hba1c  numberofmajorsurgeries  city_tier_ord  \
0         0  17.58   4.51                       1            0.0
1         0  17.60   4.39                       1            2.0
2         0  16.47   6.35                       1            2.0
3         0  17.70   6.28                       1            0.0
4         0  22.34   5.57                       1            0.0

   hospital_tier_ord  state_id_R1013  state_id_R1011  state_id_R1012
age  \
0                1.0               1               0               0
30
1                1.0               1               0               0
30
2                1.0               1               0               0
29
3                0.0               1               0               0
30
4                0.0               1               0               0
24

   ...  customer_id_Id996  customer_id_Id997  customer_id_Id998  \
0  ...                  0                  0                  0
1  ...                  0                  0                  0
2  ...                  0                  0                  0
3  ...                  0                  0                  0
4  ...                  0                  0                  0

   customer_id_Id999  heart_issues_yes  any_transplants_yes  \
0                  0                 0                    0
1                  0                 0                    0
2                  0                 0                    0
3                  0                 0                    0
4                  0                 0                    0

   cancer_history_Yes  smoker_yes  gender_male  charges
0                   0           0            1   563.84
1                   0           0            1   570.62
2                   1           0            0   600.00
3                   0           0            1   604.54
```

```
4                    0           0             1   637.26

[5 rows x 2340 columns]
```

```
model_data.columns = model_data.columns.str.lower()

# converting y to categorical for stratified k fold
y_class = pd.cut(model_data.charges, bins = 4, labels= [1,2,3,4])
X = model_data.drop(columns = 'charges')

#scoring='neg_root_mean_squared_error'

# increase max iteration for convergence

folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)
sgd_rmse_train = {}
sgd_rmse_test = {}
i = 1
for train_index, test_index in folds.split(X,y_class):

    train, test = model_data.loc[train_index,],
model_data.loc[test_index,]

    # standardization :
    sc = StandardScaler()
    sc.fit(train)
    train_std = sc.transform(train)
    test_std = sc.transform(test)


    x_train , x_test = train_std[:, :-1], test_std[:, :-1]
    y_train, y_test = train_std[:, -1], test_std[:,-1]

    # sgd regression with hyperparameter tuning :
    sgd = SGDRegressor(max_iter=100)

    # define search space
    space = dict()
    space['penalty'] = ['l1', 'l2', 'elasticnet']
    space['l1_ratio'] = [0,.1,.2,.8,1]
    space['alpha'] = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100,
1000,10000]
    space['learning_rate'] = ['constant', 'adaptive']
    space['eta0']=[1e-5, 1e-4, 1e-3, 1e-2, 1e-1 , 2e-1, 3e-1, 5e-1,
8e-1, 4e-1, 8e-1, 1, 10, 100]

    # define search
    search = RandomizedSearchCV(sgd, space,
                                cv=5, refit=True, return_train_score =
True,
                                random_state = 12
```

```
                               )

    # execute search
    result = search.fit(x_train, y_train)

    train_pred = result.predict(x_train)
    test_pred = result.predict(x_test)

    # get rmse for each fold for train data
    sgd_rmse_train.update({'Fold{}'.format(i): round(mse(y_true =
y_train, y_pred = train_pred, squared = False),3)})
    sgd_rmse_test.update({'Fold{}'.format(i): round(mse(y_true =
y_test, y_pred = test_pred, squared = False),3)})
    i += 1
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/
_stochastic_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
```

```
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.

/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_stochasti
c_gradient.py:1507: ConvergenceWarning:

Maximum number of iteration reached before convergence. Consider
increasing max_iter to improve the fit.
```

Random Forest

```
folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)
rf_rmse_train = {}
rf_rmse_test = {}
i = 1
for train_index, test_index in folds.split(X,y_class):
    train, test = model_data.loc[train_index,],
model_data.loc[test_index,]

    # standardization :
    sc = StandardScaler()
    sc.fit(train)
    trn_std = sc.transform(train)
    tst_std = sc.transform(test)

    # getting X and y :
    x_train, x_test = trn_std[:, :-1], tst_std[:,:-1]
    y_train, y_test = trn_std[:, -1], tst_std[:,-1]

    # sgd regression with hyperparameter tuning :
    rf = RandomForestRegressor(random_state = 12)
```

```
    # define search space
    space = dict()
    space['n_estimators'] = [10, 100]
    space['max_features'] = [2, 3, 4, 5, 6]

    # define search
    search = RandomizedSearchCV(rf, space,
scoring='neg_root_mean_squared_error',
                                cv=5, refit=True, return_train_score =
True,
                                random_state = 12, n_iter = 3
                           )

    # execute search
    result = search.fit(x_train, y_train)

    train_pred = result.predict(x_train)
    test_pred = result.predict(x_test)


    # get rmse for each fold for train data
    rf_rmse_train.update({'Fold{}'.format(i): round(mse(y_true =
y_train, y_pred = train_pred, squared = False),3)})
    rf_rmse_test.update({'Fold{}'.format(i): round(mse(y_true =
y_test, y_pred = test_pred, squared = False),3)})
    i += 1
```

XGBREGRESSOR

```
 folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)

 xg_rmse_train = {}
 xg_rmse_test = {}
 i = 1
 for train_index, test_index in folds.split(X,y_class):
     print('iter ', i)
     train, test = model_data.loc[train_index,],
model_data.loc[test_index,]

#     # standardization :
     sc = StandardScaler()
     sc.fit(train)
     trn_std = sc.transform(train)
     tst_std = sc.transform(test)

#     # getting X and y :
     x_train, x_test = trn_std[:, :-1], tst_std[:,:-1]
     y_train, y_test = trn_std[:, -1], tst_std[:,-1]


#     # creating inner fold for the data
```

```
     fold_inner = KFold(n_splits= 5, shuffle = True, random_state =
12)

#     # sgd regression with hyperparameter tuning :
     xgb_r = XGBRegressor(objective ='reg:squarederror', random_state
= 12)

#     # define search space
     space = dict()
     space['n_estimators'] = [40,50,60]
     space['max_depth'] = [3,4,5]
     space['colsample_bytree']:[0.4,.5,.6]
     space['lambda'] = [.0001,.002,.0004,.0003]
     space['alpha'] = [.01,.02,.1,.4]


#     # define search
     search = RandomizedSearchCV(xgb_r, space,
scoring='neg_root_mean_squared_error',
                                 cv=fold_inner, refit=True,
return_train_score = True,
                                 random_state = 12
                            )

#     # execute search
     result = search.fit(x_train, y_train)

     train_pred = result.predict(x_train)
     test_pred = result.predict(x_test)



#     # get rmse for each fold for train data
     xg_rmse_train.update({'Fold{}'.format(i): round(mse(y_true =
y_train, y_pred = train_pred, squared = False),3)})
     xg_rmse_test.update({'Fold{}'.format(i): round(mse(y_true =
y_test, y_pred = test_pred, squared = False),3)})
     i += 1

iter  1
iter  2
iter  3
iter  4
iter  5

folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)
rmse_train = {}
rmse_test = {}
i = 1
for train_index, test_index in folds.split(X,y_class):
```

```
    print('iter ', i)
    train, test = model_data.loc[train_index,],
model_data.loc[test_index,]
    sc = StandardScaler()
    xgb_r = XGBRegressor(objective ='reg:squarederror', random_state =
12)

    # define search space
    space = dict()
    space['xgb_r__n_estimators'] = [40,50,60]
    space['xgb_r__max_depth'] = [3,4,5]
    space['xgb_r__colsample_bytree']:[0.4,.5,.6]
    space['xgb_r__lambda'] = [.0001,.002,.0004,.0003]
    space['xgb_r__alpha'] = [.01,.02,.1,.4]

    pipe = Pipeline([('sc',sc), ('xgb_r', xgb_r)])

    # define search
    search = RandomizedSearchCV( pipe, space,
scoring='neg_root_mean_squared_error',
                                cv=5, refit=True, return_train_score =
True,
                                random_state = 12, n_jobs = -1, n_iter
= 2
                           )

    # execute search
    X_train = train.drop(columns = 'charges')
    y_train = train.charges

    result = search.fit(X_train, y_train)

    train_pred = result.predict(X_train)

    X_test = test.drop(columns = 'charges')
    y_test = test.charges
    test_pred = result.predict(X_test)



    # get rmse for each fold for train data
    rmse_train.update({'Fold{}'.format(i): round(mse(y_true = y_train,
y_pred = train_pred, squared = False),3)})
    rmse_test.update({'Fold{}'.format(i): round(mse(y_true = y_test,
y_pred = test_pred, squared = False),3)})
    i += 1
```

```
iter  1
iter  2
iter  3
```

```
iter  4
iter  5
```

```
rmse_train
```

```
{'Fold1': 2438.357,
 'Fold2': 2519.899,
 'Fold3': 2165.869,
 'Fold4': 2376.575,
 'Fold5': 2447.029}
```

```
rmse_test
```

```
{'Fold1': 3674.103,
 'Fold2': 3436.411,
 'Fold3': 3601.624,
 'Fold4': 3460.569,
 'Fold5': 3907.473}
```

```
# compare results :
train_results = pd.DataFrame ({'sgd' : sgd_rmse_train.values(), 'rf':
rf_rmse_train.values(),'xg' : xg_rmse_train.values() },
                             index = ['Fold {}'.format(i) for i in
range(1,6)])
train_results
```

	sgd	rf	xg
Fold 1	1.326	0.176	0.133
Fold 2	0.018	0.174	0.136
Fold 3	0.071	0.179	0.137
Fold 4	0.070	0.185	0.132
Fold 5	0.070	0.179	0.147

```
# compare results :
test_results = pd.DataFrame ({'sgd' : sgd_rmse_test.values(), 'rf':
rf_rmse_test.values(),'xg' : xg_rmse_test.values() },
                             index = ['Fold {}'.format(i) for i in
range(1,6)])
test_results
```

	sgd	rf	xg
Fold 1	1.330	0.471	0.294
Fold 2	0.540	0.423	0.262
Fold 3	0.566	0.480	0.303
Fold 4	0.537	0.473	0.287
Fold 5	0.581	0.514	0.314

```
test_results.mean()
```

```
sgd    0.7108
rf     0.4722
```

```
xg      0.2920
dtype:  float64
```

variable importance :

```
train, test = train_test_split(model_data, test_size = 0.25,
random_state = 12)

# standardization :
sc = StandardScaler()
sc.fit(train)
trn_std = sc.transform(train)
tst_std = sc.transform(test)

# getting X and y :
x_train, x_test = trn_std[:, :-1], tst_std[:,:-1]
y_train, y_test = trn_std[:, -1], tst_std[:,-1]

# sgd regression with hyperparameter tuning :
rf = RandomForestRegressor(random_state = 12)

# define search space
space = dict()
space['n_estimators'] = [10, 100, 500]
space['max_features'] = [2, 3, 4, 5, 6]

# define search
search = RandomizedSearchCV(rf, space,
scoring='neg_root_mean_squared_error',
                            cv=5, refit=True, return_train_score =
True,
                                random_state = 12, n_iter = 3
                           )
# execute search
result = search.fit(x_train, y_train)

importance = pd.Series(result.best_estimator_.feature_importances_,
index = train.drop(columns= 'charges').columns)

impvars = importance.sort_values(ascending=False)[:6]

impvars.index

Index(['smoker_yes', 'hospital_tier_ord', 'bmi', 'age', 'hba1c',
       'state_id_r1011'],
      dtype='object')
```

15. Predict the hospitalization cost for Christopher, Ms. Jayna (Date of birth – 12/28/1988, height 170 cm and weight 85 kgs). She resides in a tier1 city (state : stateid = R1011) with husband and 2 of her kids. She is tested non-diabetic (hbA1c = 5.8). She smokes but otherwise she is healthy, no transplants and no major surgeries so far. Her father had lung cancer and that was the reason of his early demise. Hospitalization cost to predicted considering tier1 hospitals.

Find predicted hospitalization cost based on all the 5 models. The predicted value should be mean of all the 5 predicted values from the 5 models.

```
model_data.columns
```

```
Index(['children', 'bmi', 'hba1c', 'numberofmajorsurgeries',
'city_tier_ord',
       'hospital_tier_ord', 'state_id_r1013', 'state_id_r1011',
       'state_id_r1012', 'age',
       ...
       'customer_id_id996', 'customer_id_id997', 'customer_id_id998',
       'customer_id_id999', 'heart_issues_yes', 'any_transplants_yes',
       'cancer_history_yes', 'smoker_yes', 'gender_male', 'charges'],
      dtype='object', length=2340)
```

```
pred_data = pd.DataFrame({'Name' : ['Christopher, Ms. Jayna'],
                      'DOB' : ['12/28/1988'],
                      'city_tier' : ['tier - 1'], 'children' :[ 2],
                       'HbA1c' : [5.8],
                       'smoker_yes' : [1],
                       'heart_issues_yes' : [0],
                       'any_transplants_yes' : [0],
                       'numberofmajorsurgeries' :[ 0],
                       'cancer_history_yes' : [1],
                       'hospital_tier' : ['tier - 1'],
                       'bmi' : [85/(1.70 **2)],
                       'state_id_R1011' : [1]
                      })
```

```
pred_data
```

```
                     Name         DOB city_tier  children  HbA1c
smoker_yes  \
0  Christopher, Ms. Jayna  12/28/1988  tier - 1         2    5.8
1

   heart_issues_yes  any_transplants_yes  numberofmajorsurgeries  \
0                 0                    0                       0

   cancer_history_yes hospital_tier        bmi  state_id_R1011
0                   1      tier - 1  29.411765               1
```

```python
pred_data.columns = pred_data.columns.str.lower()


# we will create columns according to the final model data already
created

pred_data['gender_male']  = 0
pred_data.loc[pred_data.name.str.split('[,.]').str[1] == 'Mr',
'gender_male'] = 1
pred_data.drop(columns = 'name', inplace = True)

pred_data
```

```
          dob city_tier  children  hba1c  smoker_yes  heart_issues_yes
\
0  12/28/1988  tier - 1         2    5.8           1                 0


   any_transplants_yes  numberofmajorsurgeries  cancer_history_yes  \
0                    0                       0                   1

  hospital_tier        bmi  state_id_r1011  gender_male
0      tier - 1  29.411765               1            0
```

```python
pred_data['age'] =2022 - pred_data.dob.astype(np.datetime64).dt.year

pred_data.drop(columns = 'dob', inplace = True)

ordinal.feature_names_in_
```

```
array(['city_tier', 'hospital_tier'], dtype=object)
```

```python
pred_data[['city_tier_ord', 'hospital_tier_ord']] =
ordinal.transform(pred_data[['city_tier', 'hospital_tier']])

pred_data.drop(columns =['city_tier', 'hospital_tier'], inplace = True
)

# initializing teh missing columns with 0 and not include charges

for col in model_data.columns:
    if col not in pred_data.columns and col != 'charges':
        pred_data[col] = 0

pred_data
```

```
   children  hba1c  smoker_yes  heart_issues_yes  any_transplants_yes
\
0         2    5.8           1                 0                    0


   numberofmajorsurgeries  cancer_history_yes        bmi
state_id_r1011  \
```

```
0                        0                    1  29.411765
1

   gender_male  ...  customer_id_id990  customer_id_id991
customer_id_id992  \
0            0  ...                  0                  0
0

   customer_id_id993  customer_id_id994  customer_id_id995
customer_id_id996  \
0                  0                  0                  0
0

   customer_id_id997  customer_id_id998  customer_id_id999
0                  0                  0                  0

[1 rows x 2339 columns]
```

```
folds = StratifiedKFold(n_splits=5, shuffle = True, random_state=12)
rmse_train = {}
rmse_test = {}

final_predictions = []

i = 1
for train_index, test_index in folds.split(X,y_class):
    print('iter ', i)
    train, test = model_data.loc[train_index,],
model_data.loc[test_index,]
    sc = StandardScaler()
    xgb_r = XGBRegressor(objective ='reg:squarederror', random_state =
12)

    # define search space
    space = dict()
    space['xgb_r__n_estimators'] = [40,50,60]
    space['xgb_r__max_depth'] = [3,4,5]
    space['xgb_r__colsample_bytree']:[0.4,.5,.6]
    space['xgb_r__lambda'] = [.0001,.002,.0004,.0003]
    space['xgb_r__alpha'] = [.01,.02,.1,.4]

    pipe = Pipeline([('sc',sc), ('xgb_r', xgb_r)])

    # define search
    search = RandomizedSearchCV( pipe, space,
scoring='neg_root_mean_squared_error',
                                cv=5, refit=True, return_train_score =
True,
                                random_state = 12, n_jobs = -1, n_iter
= 2
```

```
                           )

    # execute search
    X_train = train.drop(columns = 'charges')
    y_train = train.charges

    result = search.fit(X_train, y_train)

    train_pred = result.predict(X_train)

    X_test = test.drop(columns = 'charges')
    y_test = test.charges
    test_pred = result.predict(X_test)

    final_predictions.append(result.predict(pred_data))

    # get rmse for each fold for train data
    rmse_train.update({'Fold{}'.format(i): round(mse(y_true = y_train,
y_pred = train_pred, squared = False),3)})
    rmse_test.update({'Fold{}'.format(i): round(mse(y_true = y_test,
y_pred = test_pred, squared = False),3)})
    i += 1
```

```
iter  1

/usr/local/lib/python3.7/site-packages/sklearn/base.py:493:
FutureWarning:

The feature names should match those that were passed during fit.
Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.



iter  2

/usr/local/lib/python3.7/site-packages/sklearn/base.py:493:
FutureWarning:

The feature names should match those that were passed during fit.
Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.



iter  3

/usr/local/lib/python3.7/site-packages/sklearn/base.py:493:
FutureWarning:

The feature names should match those that were passed during fit.
```

```
Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.
```

```
iter  4
```

```
/usr/local/lib/python3.7/site-packages/sklearn/base.py:493:
FutureWarning:

The feature names should match those that were passed during fit.
Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.
```

```
iter  5
```

```
/usr/local/lib/python3.7/site-packages/sklearn/base.py:493:
FutureWarning:

The feature names should match those that were passed during fit.
Starting version 1.2, an error will be raised.
Feature names must be in the same order as they were in fit.
```

```
np.mean(final_predictions)
```

```
2923.9915
```