is-datascience-machine-learning-3

May 21, 2025

[1]:
```
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, Ridge
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, OneHotEncoder
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

import xgboost as xgb
from sklearn.pipeline import Pipeline
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV
import re
from scipy.stats import f_oneway,chi2_contingency,ttest_ind
from sklearn.compose import ColumnTransformer
```

0.1 Data Science

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

[4]:
```
hospital_df = pd.read_csv("Hospitalisation details.csv")
medical_df = pd.read_csv("Medical Examinations.csv")
names_df = pd.read_excel("Names.xlsx")
```

[5]:
```
hospital_df.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
```

[6]:
```
medical_df.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
```

[7]:
```
names_df.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
```

[8]:
```
merged_df = hospital_df.merge(medical_df, on="Customer ID", how="inner")
merged_df = merged_df.merge(names_df, on="Customer ID", how="inner")
```

[9]:
```
merged_df.head()
```

[9]:

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

	State ID	BMI	HBA1C	Heart Issues	Any Transplants	Cancer history
0	R1013	17.58	4.51	No	No	No
1	R1013	17.60	4.39	No	No	No
2	R1013	16.47	6.35	No	No	Yes
3	R1013	17.70	6.28	No	No	No
4	R1013	22.34	5.57	No	No	No

	NumberOfMajorSurgeries	smoker	name
0	1	No	German, Mr. Aaron K
1	1	No	Rosendahl, Mr. Evan P
2	1	No	Albano, Ms. Julie
3	1	No	Riveros Gonzalez, Mr. Juan D. Sr.
4	1	No	Brietzke, Mr. Jordan

[10]:
```
merged_df.shape
```

[10]:
```
(2335, 17)
```

[11]:
```
merged_df.columns = merged_df.columns.str.strip().str.lower().str.replace(' ',␣
↪'_')
```

[12]:
```
merged_df.columns
```

[12]:
```
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')
```

[13]:
```
merged_df.head()
```

[13]:

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

```
  state_id    bmi  hba1c heart_issues any_transplants cancer_history  \
0    R1013  17.58   4.51           No               No             No
1    R1013  17.60   4.39           No               No             No
2    R1013  16.47   6.35           No               No            Yes
3    R1013  17.70   6.28           No               No             No
4    R1013  22.34   5.57           No               No             No

  numberofmajorsurgeries smoker                                name
0                      1     No                German, Mr.  Aaron K
1                      1     No              Rosendahl, Mr.  Evan P
2                      1     No                  Albano, Ms.  Julie
3                      1     No  Riveros Gonzalez, Mr.  Juan D. Sr.
4                      1     No               Brietzke, Mr.  Jordan
```

2. Check for missing values in the dataset

[15]:
```
merged_df.isna().sum()
```

[15]:
```
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
```

No missing values are found in the merged dataset

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

[18]:
```
(merged_df=='?').sum(axis=1)
```

[18]:
```
0       0
1       0
2       0
3       0
```

```
4       0
       ..
2330    0
2331    0
2332    1
2333    0
2334    0
Length: 2335, dtype: int64
```

[19]:
```
miss_perc=(merged_df=='?').sum(axis=1)/merged_df.shape[1]*100
miss_perc
```

[19]:
```
0       0.000000
1       0.000000
2       0.000000
3       0.000000
4       0.000000
          …
2330    0.000000
2331    0.000000
2332    5.882353
2333    0.000000
2334    0.000000
Length: 2335, dtype: float64
```

[20]:
```
miss_perc[miss_perc>0].index
```

[20]:
```
Index([11, 13, 17, 542, 1046, 1049, 1700, 1775, 2165, 2332], dtype='int64')
```

[21]:
```
miss_perc_col=(merged_df=='?').sum(axis=0)/merged_df.shape[0]*100
miss_perc_col.sort_values(ascending=False)
```

[21]:
```
month                   0.128480
state_id                0.085653
smoker                  0.085653
year                    0.085653
hospital_tier           0.042827
city_tier               0.042827
heart_issues            0.000000
numberofmajorsurgeries  0.000000
cancer_history          0.000000
any_transplants         0.000000
customer_id             0.000000
hba1c                   0.000000
bmi                     0.000000
charges                 0.000000
children                0.000000
```

```
date                    0.000000
name                    0.000000
dtype: float64
```

[22]:
```
merged_df.shape
```

[22]:
```
(2335, 17)
```

[23]:
```
master_data=merged_df.drop(index=miss_perc[miss_perc>0].index)
master_data.shape
```

[23]:
```
(2325, 17)
```

[24]:
```
(master_data=='?').sum()
```

[24]:
```
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
```

[25]:
```
master_data.to_csv('master_data.csv', index=False)
```

Dataframe has been cleaned according to given requirements

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

[28]:
```
binary_columns = ['heart_issues', 'any_transplants', 'cancer_history', 'smoker']
for col in binary_columns:
    master_data[col] = master_data[col].map({'yes': 1, 'no': 0, 'Yes': 1, 'No':␣
 ↪0})
```

[29]:
```
ordinal = OrdinalEncoder(categories= [['tier - 3', 'tier - 2', 'tier -␣
 ↪1'],['tier - 3', 'tier - 2', 'tier - 1']])
```

```
master_data[['city_tier_ord','hospital_tier_ord']] = ordinal.
 ↪fit_transform(master_data[['city_tier', 'hospital_tier']])
```

[30]:
```
master_data.head()
```

[30]:

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

	state_id	bmi	hba1c	heart_issues	any_transplants	cancer_history	\
0	R1013	17.58	4.51	0	0	0	
1	R1013	17.60	4.39	0	0	0	
2	R1013	16.47	6.35	0	0	1	
3	R1013	17.70	6.28	0	0	0	
4	R1013	22.34	5.57	0	0	0	

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

	city_tier_ord	hospital_tier_ord
0	0.0	1.0
1	2.0	1.0
2	2.0	1.0
3	0.0	0.0
4	0.0	0.0

0.1.1 5. 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.

[32]:
```
#Master_data=master_data
```

[33]:
```
master_data['filtered_state'] = master_data['state_id'].apply(lambda x: x if x␣
 ↪in ['R1011', 'R1012', 'R1013'] else 'Other')
master_data = pd.get_dummies(master_data, columns=['filtered_state'],␣
 ↪drop_first=True,dtype=int)
```

[34]:
```
master_data.head()
```

[34]:
```
  customer_id  year month  date  children  charges hospital_tier city_tier  \
0      Id2335  1992   Jul     9         0   563.84      tier - 2  tier - 3
1      Id2334  1992   Nov    30         0   570.62      tier - 2  tier - 1
2      Id2333  1993   Jun    30         0   600.00      tier - 2  tier - 1
3      Id2332  1992   Sep    13         0   604.54      tier - 3  tier - 3
4      Id2331  1998   Jul    27         0   637.26      tier - 3  tier - 3

  state_id    bmi  …  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

   city_tier_ord hospital_tier_ord  filtered_state_R1011  \
0            0.0              1.0                     0
1            2.0              1.0                     0
2            2.0              1.0                     0
3            0.0              0.0                     0
4            0.0              0.0                     0

   filtered_state_R1012  filtered_state_R1013
0                     0                     1
1                     0                     1
2                     0                     1
3                     0                     1
4                     0                     1

[5 rows x 22 columns]
```

[35]:
```
master_data.state_id.value_counts()
```

[35]:
```
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
```

[36]:
```
master_data['filtered_state_R1011'].value_counts()
```

[36]:
```
filtered_state_R1011
0    1751
1     574
Name: count, dtype: int64
```

[37]:
```
master_data['filtered_state_R1012'].value_counts()
```

[37]:
```
filtered_state_R1012
0    1753
1     572
Name: count, dtype: int64
```

[38]:
```
master_data['filtered_state_R1013'].value_counts()
```

[38]:
```
filtered_state_R1013
0    1716
1     609
Name: count, dtype: int64
```

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

[40]:
```
master_data.numberofmajorsurgeries.unique()
```

[40]:
```
array(['1', 'No major surgery', '2', '3'], dtype=object)
```

[41]:
```
def parse_surgery(val):
    if isinstance(val, str) and 'no major surgery' in val.lower():
        return 0
    try:
        return int(val)
    except:
```

```
        return np.nan
```

[42]:
```
master_data['numberofmajorsurgeries'] = master_data['numberofmajorsurgeries'].
↪apply(parse_surgery)
```

[43]:
```
master_data.head()
```

[43]:
```
  customer_id  year month  date  children  charges hospital_tier city_tier  \
0      Id2335  1992   Jul     9         0   563.84      tier - 2  tier - 3
1      Id2334  1992   Nov    30         0   570.62      tier - 2  tier - 1
2      Id2333  1993   Jun    30         0   600.00      tier - 2  tier - 1
3      Id2332  1992   Sep    13         0   604.54      tier - 3  tier - 3
4      Id2331  1998   Jul    27         0   637.26      tier - 3  tier - 3

  state_id    bmi  …  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

   city_tier_ord hospital_tier_ord  filtered_state_R1011  \
0            0.0               1.0                    0
1            2.0               1.0                    0
2            2.0               1.0                    0
3            0.0               0.0                    0
4            0.0               0.0                    0

   filtered_state_R1012  filtered_state_R1013
0                     0                     1
1                     0                     1
2                     0                     1
3                     0                     1
4                     0                     1

[5 rows x 22 columns]
```

[44]:
```
master_data.numberofmajorsurgeries.unique()
```

[44]:
```
array([1, 0, 2, 3], dtype=int64)
```

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

[46]:
```
master_data.year = master_data.year.astype(int)
```

[47]:
```
master_data['age'] = 2025 - master_data.year #Current year is taken as␣
↪benchmark for calculating the age
```

[48]:
```
master_data.head()
```

[48]:

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

	state_id	bmi	…	cancer_history	numberofmajorsurgeries	smoker	\
0	R1013	17.58	…	0	1	0	
1	R1013	17.60	…	0	1	0	
2	R1013	16.47	…	1	1	0	
3	R1013	17.70	…	0	1	0	
4	R1013	22.34	…	0	1	0	

	name	city_tier_ord	hospital_tier_ord	\
0	German, Mr. Aaron K	0.0	1.0	
1	Rosendahl, Mr. Evan P	2.0	1.0	
2	Albano, Ms. Julie	2.0	1.0	
3	Riveros Gonzalez, Mr. Juan D. Sr.	0.0	0.0	
4	Brietzke, Mr. Jordan	0.0	0.0	

	filtered_state_R1011	filtered_state_R1012	filtered_state_R1013	age
0	0	0	1	33
1	0	0	1	33
2	0	0	1	32
3	0	0	1	33
4	0	0	1	27

[5 rows x 23 columns]

0.1.4 8. 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.

[50]:
```
def extract_gender(name):
    match = re.search(r'\b(Mr|Ms|Mrs|Miss|Dr|Mister)\b', name, re.IGNORECASE)
    if match:
        salutation = match.group(1).lower()
        if salutation in ['mr', 'mister']:
            return 'Male'
        elif salutation in ['ms', 'mrs', 'miss']:
            return 'Female'
    return 'Unknown'
```

[51]:
```
master_data['gender'] = master_data['name'].apply(extract_gender)
```

[52]:
```
master_data.head()
```

[52]:

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

	state_id	bmi	…	numberofmajorsurgeries	smoker	\
0	R1013	17.58	…	1	0	
1	R1013	17.60	…	1	0	
2	R1013	16.47	…	1	0	
3	R1013	17.70	…	1	0	
4	R1013	22.34	…	1	0	

	name	city_tier_ord	hospital_tier_ord	\
0	German, Mr. Aaron K	0.0	1.0	
1	Rosendahl, Mr. Evan P	2.0	1.0	
2	Albano, Ms. Julie	2.0	1.0	
3	Riveros Gonzalez, Mr. Juan D. Sr.	0.0	0.0	
4	Brietzke, Mr. Jordan	0.0	0.0	

	filtered_state_R1011	filtered_state_R1012	filtered_state_R1013	age	\
0	0	0	1	33	
1	0	0	1	33	
2	0	0	1	32	
3	0	0	1	33	
4	0	0	1	27	

	gender

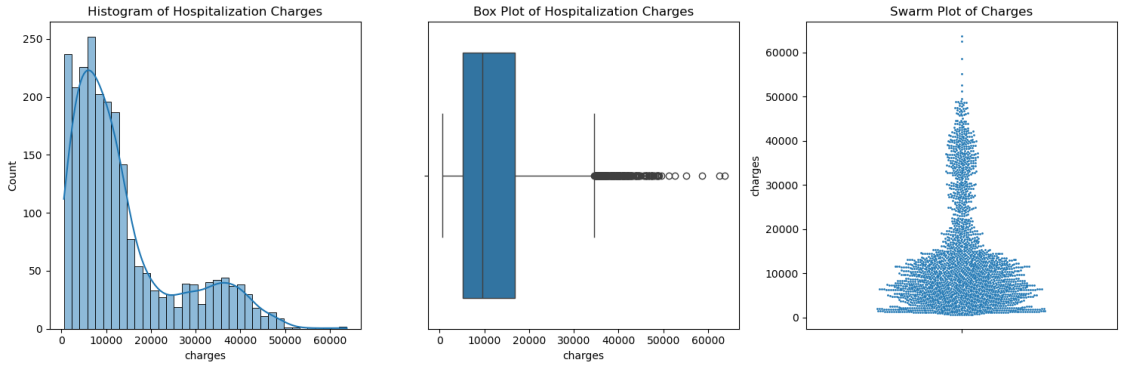
```
0    Male
1    Male
2  Female
3    Male
4    Male

[5 rows x 24 columns]
```

[53]:
```
master_data['gender'].value_counts()
```

[53]:
```
gender
Female    1165
Male      1160
Name: count, dtype: int64
```

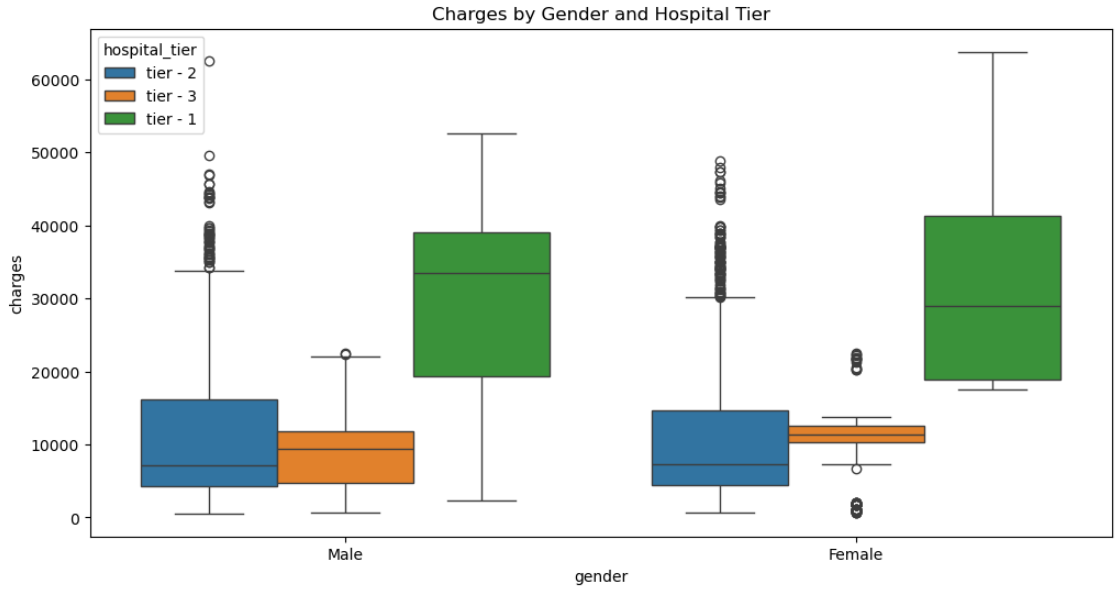
0.1.5 9. Visualize the distribution of cost using histogram, box and whisker and swarm plot. How the distribution is different across gender and different tiers of hospitals. Share your observation.

[55]:
```
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.histplot(master_data['charges'], kde=True)
plt.title("Histogram of Hospitalization Charges")

plt.subplot(1, 3, 2)
sns.boxplot(x=master_data['charges'])
plt.title("Box Plot of Hospitalization Charges")

plt.subplot(1, 3, 3)
sns.swarmplot(y=master_data['charges'], size=2)
plt.title("Swarm Plot of Charges")
plt.tight_layout()
plt.show()
```

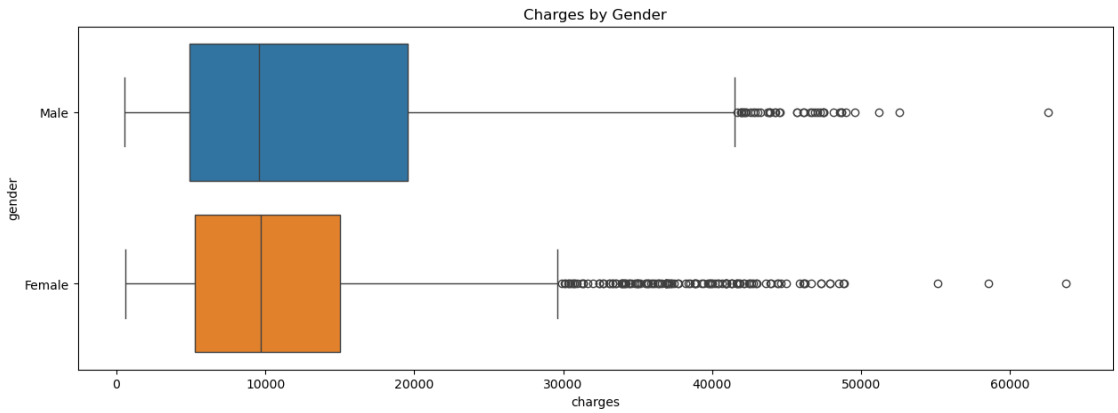


10. Distribution Difference w.r.t Gender

[57]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='gender', y='charges', hue='hospital_tier', data=master_data)
plt.title("Charges by Gender and Hospital Tier")
plt.show()
```

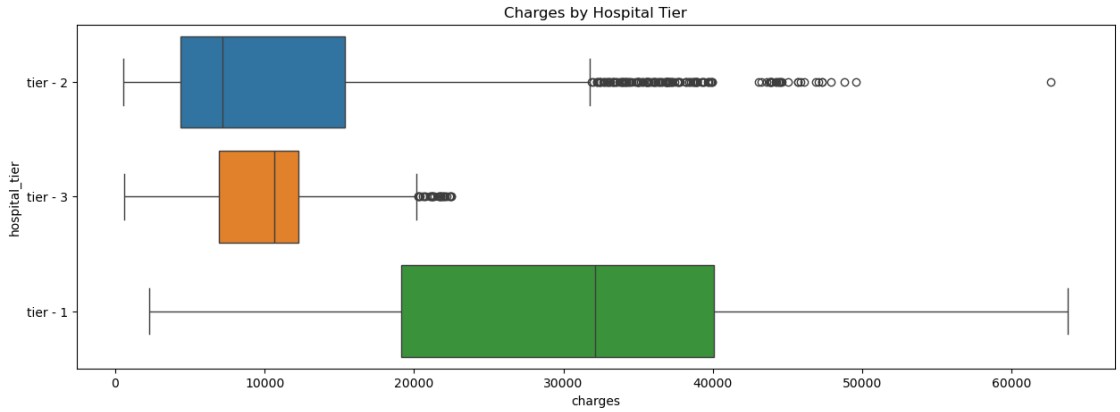


[58]:
```
plt.figure(figsize = (15,5))
sns.boxplot(x = "charges",y = "gender", data = master_data,hue="gender")
plt.title("Charges by Gender")
plt.show()
```

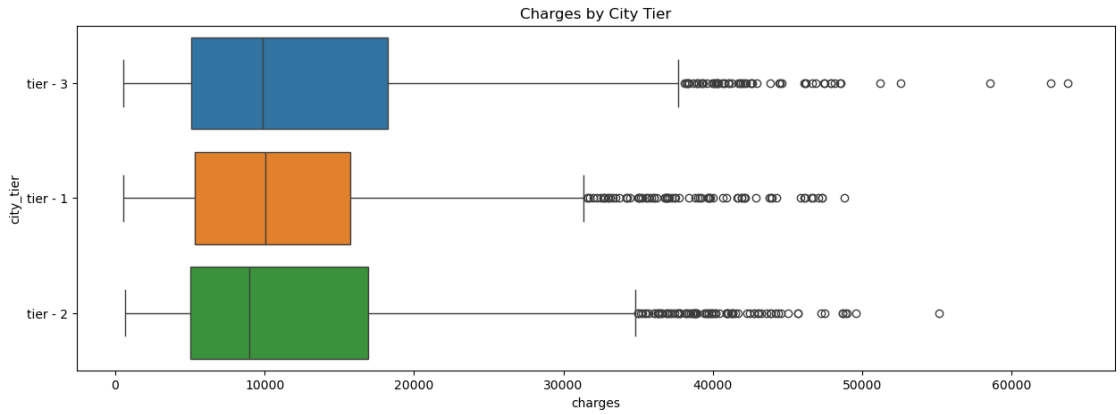


[59]:
```
plt.figure(figsize = (15,5))
```

```
sns.boxplot(x = "charges",y = "hospital_tier", data =␣
 ↪master_data,hue="hospital_tier")
plt.title("Charges by Hospital Tier")
plt.show()
```



[60]:
```
plt.figure(figsize = (15,5))
sns.boxplot(x = "charges",y = "city_tier", data = master_data, hue="city_tier")
plt.title("Charges by City Tier")
plt.show()
```



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

[62]:
```
median = master_data.groupby('hospital_tier')[['charges']].median().
 ↪reset_index()
median
```

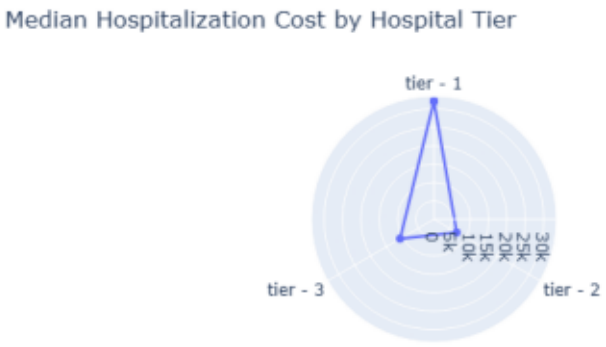
[62]:

	hospital_tier	charges
0	tier - 1	32097.435
1	tier - 2	7168.760
2	tier - 3	10676.830

[63]:
```
median_cost_by_tier = master_data.groupby('hospital_tier')['charges'].median().
 ↪reset_index()
fig = px.line_polar(median_cost_by_tier, r='charges', theta='hospital_tier',␣
 ↪line_close=True,
                     title='Median Hospitalization Cost by Hospital Tier',␣
 ↪markers=True)
fig.show()
```

To display the above radar chart in pdf format

[322]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Replace with the path to your image
img = mpimg.imread('RadialPlot.png')
plt.imshow(img)
plt.axis('off')  # Hide axes
plt.show()
```

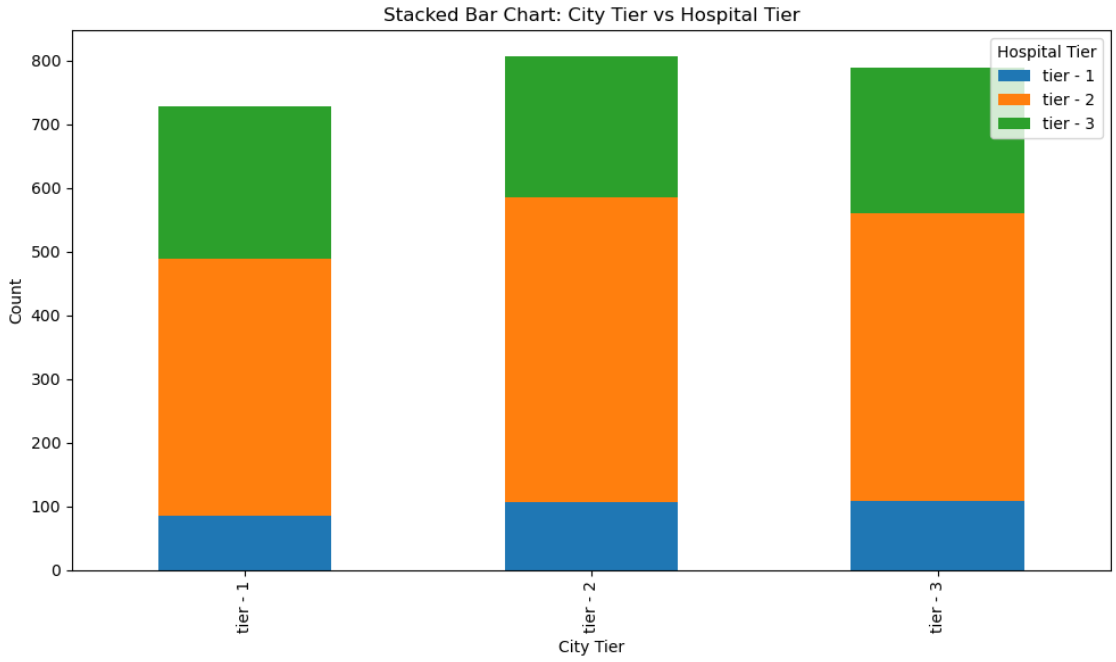


0.1.7 12. Create a frequency table and hence a stacked bar-chart to visualize the count of people in different tiers of cities and hospitals.

[65]:
```
pd.crosstab(master_data.city_tier, master_data.hospital_tier) # For frequency␣
↪chart
```

[65]:
```
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
```

[66]:
```
freq_table = pd.crosstab(master_data['city_tier'], master_data['hospital_tier'])

freq_table.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Stacked Bar Chart: City Tier vs Hospital Tier')
plt.xlabel('City Tier')
plt.ylabel('Count')
plt.legend(title='Hospital Tier')
plt.tight_layout()
plt.show()
```



0.1.8 13. 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 are not significantly different

[69]:
```
results = {}

# a. ANOVA: hospital tier
groups = [group['charges'].values for name, group in master_data.
 ↪groupby('hospital_tier')]
f_stat, p_value = f_oneway(*groups)
results['Hospital tier ANOVA'] = (f_stat, p_value)
```

[70]:
```
print(f'f_stat:{f_stat} p_value:{p_value}')
```

```
f_stat:493.98956631117636 p_value:1.7738221310852664e-179
```

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

[72]:
```
# b. ANOVA: city tier
groups_city = [group['charges'].values for name, group in master_data.
 ↪groupby('city_tier')]
f_stat, p_value = f_oneway(*groups_city)
results['City tier ANOVA'] = (f_stat, p_value)
```

[73]:
```
print(f'f_stat:{f_stat} p_value:{p_value}')
```

```
f_stat:1.4543557561814688 p_value:0.23376344386881315
```

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

[75]:
```
# c. T-test: smokers vs non-smokers
charges_smokers = master_data[master_data['smoker'] == 1]['charges']
charges_non_smokers =master_data[master_data['smoker'] == 0]['charges']
t_stat, p_value = ttest_ind(charges_smokers, charges_non_smokers)
results['Smoker vs Non-smoker T-test'] = (t_stat, p_value)
```

[76]:
```
print(f't_stat:{t_stat} p_value:{p_value}')
```

```
t_stat:74.15560699695726 p_value:0.0
```

H0 : Smoking and Heart issues are independent

[78]:
```
# d. Chi-square test: smoking vs heart issues
contingency = pd.crosstab(master_data['smoker'],master_data['heart_issues'])
chi2, p, dof, expected = chi2_contingency(contingency)
results['Smoking vs Heart Issues Chi-square'] = (chi2, p)
```

[79]:
```
print(f'chi_square_stat:{chi2} p_value:{p} dof:{dof} expected:{expected}')
```

```
chi_square_stat:0.08588150449910657 p_value:0.7694797581780767 dof:1
expected:[[1111.30967742  727.69032258]
 [ 293.69032258  192.30967742]]
```

[80]:
```
print("Results for all Hypotheses Testng:\n")
print("{:<40} {:<20} {:<20}".format('NAME OF HYPOTHESIS TEST', 'STATISTIC',␣
 ↪'P_VALUE'))
for key, value in results.items():
    Statistic,p_value = value
    print("{:<40} {:<20} {:<20}".format(key,Statistic,p_value))
```

```
Results for all Hypotheses Testng:

NAME OF HYPOTHESIS TEST                 STATISTIC            P_VALUE
Hospital tier ANOVA                     493.98956631117636
1.7738221310852664e-179
City tier ANOVA                         1.4543557561814688
0.23376344386881315
Smoker vs Non-smoker T-test             74.15560699695726    0.0
Smoking vs Heart Issues Chi-square      0.08588150449910657  0.7694797581780767
```

From the above table displaying the results for the different hypothesis tests conducted, for each null hypothesis we can conclude:

[82]:
```
print("Let alpha value be 0.05")
alpha=0.05
print("For Null Hypothesis--Average hospitalization cost across the 3 types of␣
 ↪hospitals are not significantly different\n")
if results['Hospital tier ANOVA'][1]<alpha:
    print("Looking at the p_value and further analysis, we can reject the null␣
 ↪hypothesis and conclude that:\nAverage hospitalization costs across the 3␣
 ↪types of hospitals are significantly different")

else:
    print("Looking at the p_value and further analysis, we cannot reject the␣
 ↪null hypothesis and conclude that:\nAverage hospitalization costs across the␣
 ↪3 types of hospitals are not significantly different")
```

```
Let alpha value be 0.05
For Null Hypothesis--Average hospitalization cost across the 3 types of
hospitals are not significantly different

Looking at the p_value and further analysis, we can reject the null hypothesis
and conclude that:
Average hospitalization costs across the 3 types of hospitals are significantly
different
```

[83]:
```
print("Let alpha value be 0.05")
alpha=0.05
print("For Null Hypothesis--Average hospitalization cost across the 3 types of␣
 ↪cities is not significantly different\n")
if results['City tier ANOVA'][1]<alpha:
    print("Looking at the p_value and further analysis, we reject the null␣
 ↪hypothesis and conclude that:\nAverage hospitalization cost across the 3␣
 ↪types of cities is significantly different")

else:
    print("Looking at the p_value and further analysis, we fail to reject the␣
 ↪null hypothesis and conclude that:\nAverage hospitalization cost across the␣
 ↪3 types of cities is not significantly different")
```

```
Let alpha value be 0.05
For Null Hypothesis--Average hospitalization cost across the 3 types of cities
is not significantly different

Looking at the p_value and further analysis, we fail to reject the null
hypothesis and conclude that:
Average hospitalization cost across the 3 types of cities is not significantly
different
```

[84]:
```
print("Let alpha value be 0.05")
alpha=0.05
print("For Null Hypothesis--Average hospitalization cost for smokers is not␣
 ↪significantly different than non-smokers\n ")
if results['Smoker vs Non-smoker T-test'][1]<alpha:
    print("Looking at the p_value, we can reject the null hypothesis and␣
 ↪conclude that:\nAverage hospitalization cost for smokers is significantly␣
 ↪different than non-smokers")
else:
    print("Looking at the p_value, we fail to reject the null hypothesis and␣
 ↪conclude that:\nAverage hospitalization cost for smokers is not␣
 ↪significantly different than non-smokers")
```

```
Let alpha value be 0.05
For Null Hypothesis--Average hospitalization cost for smokers is not
significantly different than non-smokers

Looking at the p_value, we can reject the null hypothesis and conclude that:
Average hospitalization cost for smokers is significantly different than non-
smokers
```

[85]:
```
print("Let alpha value be 0.05")
alpha=0.05
print("For Null Hypothesis--Smoking and Heart issues are independent\n ")
```

```
if results['Smoking vs Heart Issues Chi-square'][1]<alpha:
    print("Looking at the p_value, we can reject the null hypothesis and␣
 ↪conclude that:\nSmoking and Heart issues are not independent")
else:
    print("Looking at the p_value, we fail to reject the null hypothesis and␣
 ↪conclude that:\nSmoking and Heart issues are independent")
```

```
Let alpha value be 0.05
For Null Hypothesis--Smoking and Heart issues are independent

Looking at the p_value, we fail to reject the null hypothesis and conclude that:
Smoking and Heart issues are independent
```

0.2 Machine Learning

1. Examine the correlation between predictors to identify highly correlated predictors
Before examining the correlation between predictors to identify highly correlated predictors, we can first delete the redundant columns which were previously created for the above EDA.

Columns present in original master_data:

[278]:
```
master_data.columns
```

[278]:
```
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', 'filtered_state_R1011', 'filtered_state_R1012',
       'filtered_state_R1013', 'age', 'gender'],
      dtype='object')
```

We can remove the following redundant columns from main dataset and create a new dataset to be used to create prediction models: 1. customer_id' 2. 'name' 3. 'year' 4. 'month' 5. 'date' 6. 'hospital_tier' 7. 'city_tier' 8. 'state_id'

[281]:
```
prediction_model_data = master_data.drop(columns =␣
 ↪['customer_id','name','year','month','date','hospital_tier',
       'city_tier', 'state_id'])
```

[283]:
```
#To convert binary column values to True and False
binary_columns = ['heart_issues', 'any_transplants', 'cancer_history', 'smoker']
for col in binary_columns:
   prediction_model_data[col] =prediction_model_data[col].map({1:'Yes', 0:'No'})
```

[285]:
```
prediction_model_data.head()
```

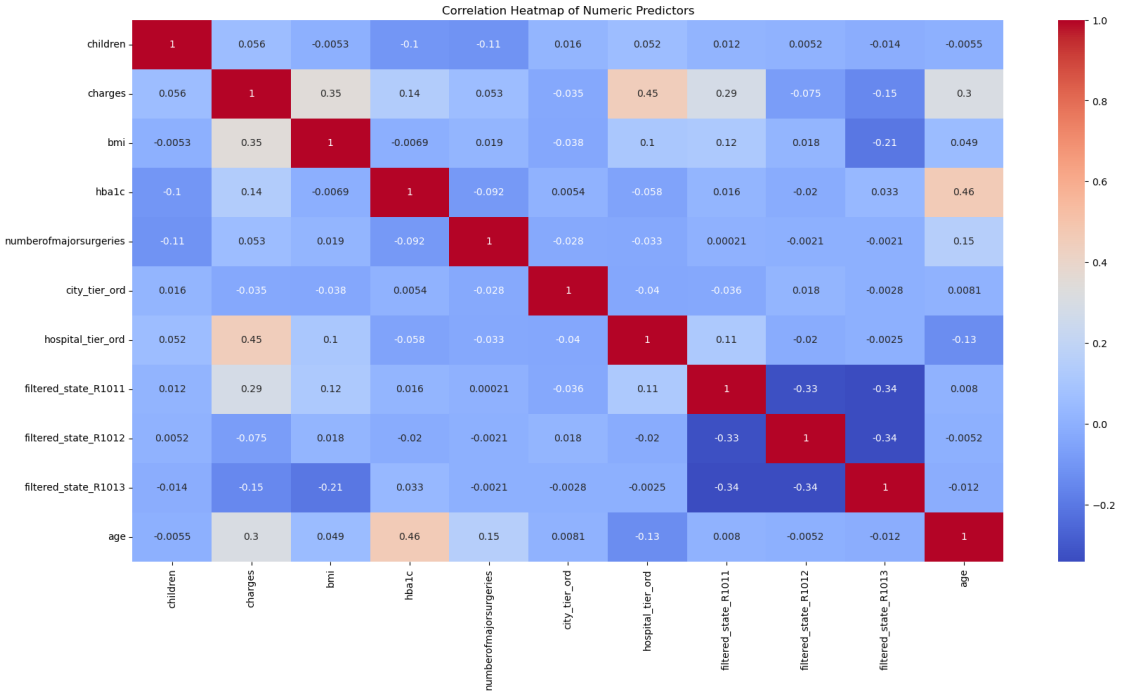
[285]:

	children	charges	bmi	hba1c	heart_issues	any_transplants	\
0	0	563.84	17.58	4.51	No	No	

```
1          0   570.62  17.60   4.39           No               No
2          0   600.00  16.47   6.35           No               No
3          0   604.54  17.70   6.28           No               No
4          0   637.26  22.34   5.57           No               No

  cancer_history  numberofmajorsurgeries smoker  city_tier_ord  \
0             No                       1     No            0.0
1             No                       1     No            2.0
2            Yes                       1     No            2.0
3             No                       1     No            0.0
4             No                       1     No            0.0

   hospital_tier_ord  filtered_state_R1011  filtered_state_R1012  \
0                1.0                     0                     0
1                1.0                     0                     0
2                1.0                     0                     0
3                0.0                     0                     0
4                0.0                     0                     0

   filtered_state_R1013  age  gender
0                     1   33    Male
1                     1   33    Male
2                     1   32  Female
3                     1   33    Male
4                     1   27    Male
```

As the above dataset is created from an already cleaned dataset, we can now directly proceed with examining the correlation between predictors to identify highly correlated predictors

[288]:
```
plt.figure(figsize=(20, 10))
sns.heatmap(prediction_model_data.select_dtypes(include=np.number).corr(),␣
 ↪annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap of Numeric Predictors")
plt.show()
```



Variables	Correlati	Interpretation
`charges` & `hospital_tier_ord`	**0.45**	Higher-tier hospitals tend to have **higher hospitalization costs**
`charges` & `bmi`	**0.35**	Patients with higher BMI tend to have **higher medical charges**
`age` & `hba1c`	**0.46**	Older individuals tend to have **higher HbA1c**, possibly indicating prediabetes or diabetes
`charges` & `filtered_state_R1011`	**0.29**	Patients from state R1011 have **slightly higher charges**
`filtered_state_R1013` & `charges`	**-0.15**	Patients from state R1013 tend to incur **lower hospitalization charges**
`filtered_state_R1013` & `bmi`	**-0.21**	Individuals from R1013 may have **lower BMI on average**
`hospital_tier_ord` & `age`	**-0.13**	Slight tendency for **older patients** to use **lower-tier hospitals**

Variables along with their correlation and interpretation

For variables with low or no correlation (|r|<0.1): Most other variable pairs (e.g., numberofmajorsurgeries, children, city_tier_ord) have very low correlation values, implying weak or no linear relationships. This suggests: 1. These variables may not strongly influence each other linearly 2. Their importance should be further evaluated using feature importance or nonlinear models like Gradient Boosting 3. No predictors seem to be highly correlated (> 0.8) with each

other, so no multicollinearity concerns are apparent. That's good for linear models.

Implications for Modeling:

1. Variables like hospital_tier_ord, bmi, age, and filtered_state_R1011 could be good predictors for hospitalization cost.
2. State variables (R1011, R1013) may need to be carefully interpreted or grouped due to their moderate correlation patterns.
3. Variables with very low correlations (e.g., children, city_tier_ord) may be less impactful in a linear regression model but still could matter in nonlinear models.

Note: The following variables are worth including in predictive models: 1. hospital_tier_ord 2. age 3. bmi 4. filtered_state_R1011 5. hba1c 6. charges

The variable 'charges' can be considered as the target variable

2. Final Model Development and Evaluation : Perform stratified 5-fold cross validation technique for final prediction and validation. Make sure to use standardization, hyperparameter tuning effectively. There must be effective use of sklearn-pipelines. a. Create 5 fold in the data. You may want to create a variable to identify the folds.

[293]:
```
data_2 = pd.get_dummies(prediction_model_data, drop_first=True)
data_2.reset_index(drop=True, inplace = True)
data_2.head()
```

[293]:

	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	filtered_state_R1011	filtered_state_R1012	\
0	1.0	0	0	
1	1.0	0	0	
2	1.0	0	0	
3	0.0	0	0	
4	0.0	0	0	

	filtered_state_R1013	age	heart_issues_Yes	any_transplants_Yes	\
0	1	33	False	False	
1	1	33	False	False	
2	1	32	False	False	
3	1	33	False	False	
4	1	27	False	False	

	cancer_history_Yes	smoker_Yes	gender_Male
0	False	False	True

```
1               False       False         True
2                True       False        False
3               False       False         True
4               False       False         True
```

[295]:
```
# 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()
```

[295]:
```
   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  filtered_state_R1011  filtered_state_R1012  \
0                1.0                     0                     0
1                1.0                     0                     0
2                1.0                     0                     0
3                0.0                     0                     0
4                0.0                     0                     0

   filtered_state_R1013  age  heart_issues_Yes  any_transplants_Yes  \
0                     1   33             False                False
1                     1   33             False                False
2                     1   32             False                False
3                     1   33             False                False
4                     1   27             False                False

   cancer_history_Yes  smoker_Yes  gender_Male  charges
0               False       False         True   563.84
1               False       False         True   570.62
2                True       False        False   600.00
3               False       False         True   604.54
4               False       False         True   637.26
```

[297]:
```
model_data.columns = model_data.columns.str.lower()
```

[299]:
```
model_data.columns
```

[299]:
```
Index(['children', 'bmi', 'hba1c', 'numberofmajorsurgeries', 'city_tier_ord',
       'hospital_tier_ord', 'filtered_state_r1011', 'filtered_state_r1012',
       'filtered_state_r1013', 'age', 'heart_issues_yes',
       'any_transplants_yes', 'cancer_history_yes', 'smoker_yes',
```

```
       'gender_male', 'charges'],
      dtype='object')
```

[301]:
```
# converting y to categorical for stratified k fold
y = model_data['charges']
X = model_data.drop(columns = 'charges')
```

[303]:
```
X.head()
```

[303]:

	children	bmi	hba1c	numberofmajorsurgeries	city_tier_ord	hospital_tier_ord	filtered_state_r1011	filtered_state_r1012	filtered_state_r1013	age	heart_issues_yes	any_transplants_yes	cancer_history_yes	smoker_yes	gender_male
0	0	17.58	4.51	1	0.0	1.0	0	0	1	33	False	False	False	False	True
1	0	17.60	4.39	1	2.0	1.0	0	0	1	33	False	False	False	False	True
2	0	16.47	6.35	1	2.0	1.0	0	0	1	32	False	False	True	False	False
3	0	17.70	6.28	1	0.0	0.0	0	0	1	33	False	False	False	False	True
4	0	22.34	5.57	1	0.0	0.0	0	0	1	27	False	False	False	False	True

[106]:
```
#Setting up a pipeline
pipeline = Pipeline(steps=[('scaler', StandardScaler()), ('regressor',␣
 ↪Ridge())])
# Defining the parameters for hyperparameter tuning
parameters = {'regressor__alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
# Creating the KFold object
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
# Creating the grid search object
model_ridge = GridSearchCV(pipeline, parameters, cv=kfold,␣
 ↪scoring='neg_mean_squared_error')
```

[107]:
```
model_ridge.fit(X,y)
```

[107]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=42, shuffle=True),
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('regressor', Ridge())]),
             param_grid={'regressor__alpha': [0.001, 0.01, 0.1, 1, 10, 100]},
             scoring='neg_mean_squared_error')
```

[108]:
```
print("Best Ridge alpha:", model_ridge.best_params_)
print("Ridge CV RMSE:", np.sqrt(-model_ridge.best_score_))
model_ridge.best_estimator_
```

```
Best Ridge alpha: {'regressor__alpha': 10}
Ridge CV RMSE: 4487.227630452237
```

[108]:
```
Pipeline(steps=[('scaler', StandardScaler()), ('regressor', Ridge(alpha=10))])
```

0.2.1 Gradient Boosting Algorithm

[110]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y)
# Train the XGBoost model
model = GradientBoostingRegressor()
model.fit(X_train, y_train)


# You can print the feature importances if needed
print(model.feature_importances_)
```

```
[6.09315291e-03 1.14132545e-01 4.83616112e-03 1.70257522e-04
 8.59841592e-04 2.39346762e-02 7.85018833e-03 3.46329944e-04
 4.41073548e-03 9.52079639e-02 2.27804552e-06 4.48627548e-05
 5.03955767e-05 7.41779685e-01 2.80926715e-04]
```

Variable importance

[112]:
```
pd.DataFrame({'Features':model.feature_names_in_,'Importance':model.
  ↪feature_importances_}).sort_values("Importance",ascending=False)
```

[112]:

	Features	Importance
13	smoker_yes	0.741780
1	bmi	0.114133
9	age	0.095208
5	hospital_tier_ord	0.023935
6	filtered_state_r1011	0.007850
0	children	0.006093
2	hba1c	0.004836
8	filtered_state_r1013	0.004411
4	city_tier_ord	0.000860

```
7     filtered_state_r1012    0.000346
14             gender_male    0.000281
3   numberofmajorsurgeries    0.000170
12      cancer_history_yes    0.000050
11     any_transplants_yes    0.000045
10        heart_issues_yes    0.000002
```

[113]:
```
#training data
model.score(X_train,y_train)
```

[113]:
```
0.9320657428220019
```

[114]:
```
#test data
model.score(X_test,y_test)
```

[114]:
```
0.9243704773108564
```

0.3 3. 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.

[116]:
```
model_data.columns
```

[116]:
```
Index(['children', 'bmi', 'hba1c', 'numberofmajorsurgeries', 'city_tier_ord',
       'hospital_tier_ord', 'filtered_state_r1011', 'filtered_state_r1012',
       'filtered_state_r1013', 'age', 'heart_issues_yes',
       'any_transplants_yes', 'cancer_history_yes', 'smoker_yes',
       'gender_male', 'charges'],
      dtype='object')
```

[117]:
```
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]
                       })
```

[118]:
```
pred_data.columns = pred_data.columns.str.lower()
```

[119]:
```
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)
```

[120]:
```
pred_data.drop(columns = 'dob', inplace = True)
```

[121]:
```
pred_data[['city_tier_ord', 'hospital_tier_ord']] = ordinal.
 ↪transform(pred_data[['city_tier', 'hospital_tier']])
```

[122]:
```
pred_data.drop(columns =['city_tier', 'hospital_tier'], inplace = True )
```

[123]:
```
for col in model_data.columns:
    if col not in pred_data.columns and col != 'charges':
        pred_data[col] = 0
```

[124]:
```
pred_data
```

[124]:
```
   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  city_tier_ord  hospital_tier_ord  filtered_state_r1011  \
0            0            2.0                2.0                     0

   filtered_state_r1012  filtered_state_r1013  age
0                     0                     0    0
```

[125]:
```
### Apply Gradient BOOST model for predi
model_data.columns
```

[125]:
```
Index(['children', 'bmi', 'hba1c', 'numberofmajorsurgeries', 'city_tier_ord',
       'hospital_tier_ord', 'filtered_state_r1011', 'filtered_state_r1012',
       'filtered_state_r1013', 'age', 'heart_issues_yes',
       'any_transplants_yes', 'cancer_history_yes', 'smoker_yes',
       'gender_male', 'charges'],
      dtype='object')
```

[126]:
```
pred_data.columns
```

[126]:
```
Index(['children', 'hba1c', 'smoker_yes', 'heart_issues_yes',
       'any_transplants_yes', 'numberofmajorsurgeries', 'cancer_history_yes',
       'bmi', 'state_id_r1011', 'gender_male', 'city_tier_ord',
       'hospital_tier_ord', 'filtered_state_r1011', 'filtered_state_r1012',
       'filtered_state_r1013', 'age'],
      dtype='object')
```

[127]:
```
pred_data=pred_data[model_data.drop(columns='charges').columns]
```

[228]:
```
#model.predict(pred_data)
predicted_cost=model.predict(pred_data)[0]
print(f"Estimated hospitalization cost for Ms. Jayna Christopher:␣
  ↪{predicted_cost:,.2f}")
```

```
Estimated hospitalization cost for Ms. Jayna Christopher:  22,181.65
```

[ ]:

SQL SOLUTIONS

/* 1. To get a complete understanding of the driving factors behind hospitalisation costs,

it is important to merge the given tables.

Identify the columns present in the data tables which can allow that to happen.

Add 'Primary Key' constraint for these columns in both the tables.

Hint: remove duplicates and null values in the column and then use alter table to add primary key constraint.

*/

```
use new_schema;

Rename table `hospitalisation details` To hospitalisation_details;

Rename table ` medical examinations ` To medical_examinations;


select `Customer ID` , count(*) as ct from hospitalisation_details

group by `Customer ID`

order by ct desc;

SET SQL_SAFE_UPDATES = 0;


DELETE from hospitalisation_details

where `Customer ID` ="?";

alter table hospitalisation_details

modify  `Customer ID` varchar(10) not null;


alter table hospitalisation_details

add primary key ( `Customer ID`);


alter table medical_examinations

modify  `Customer ID` varchar(10) not null;


alter table medical_examinations

add primary key ( `Customer ID`);
```

```
SET SQL_SAFE_UPDATES = 1;

select `Customer ID` , count(*) as ct from medical_examinations
group by `Customer ID`
order by ct desc;

alter table names
modify  `Customer ID` varchar(10) not null;

alter table names
add primary key ( `Customer ID`);
/* 2. Get information about individuals who are diabetic and have heart ailments.
     Get average age, average no. of children dependent, average BMI, and
     average hospitalization costs for such individuals. */
SELECT
           m.diabetes,
   m.`Heart Issues`,
   round(AVG(h.age),0) AS avg_age,
   round(AVG(h.children),0) AS avg_child_dep,
   round(AVG(m.BMI),2) AS avg_bmi,
   round(AVG(h.charges),2) AS avg_charges
FROM
   (select * , 2025 - year AS age
   from hospitalisation_details) h,
   (SELECT
      *,
         CASE
            WHEN HBA1C > 6.5 THEN 'Yes'
            ELSE 'No'
         END AS diabetes
```

```
   FROM
       medical_examinations) m
 where h.`Customer ID` = m.`Customer ID`
 GROUP BY m.diabetes ,m.`Heart Issues`;
/* What are the average charges of hospitalization across different hospital levels and cities?*/
/* replace "?" in City tier and hospital tier with mode value */
select `Hospital tier`,count(*) as ct
from hospitalisation_details
group by `Hospital tier`
order by ct ;


select `City tier`,count(*) as ct
from hospitalisation_details
group by `City tier`
order by ct ;

# replace "?" with mode values
SET SQL_SAFE_UPDATES = 0;
update hospitalisation_details
set `Hospital tier` = "tier - 2"
where `Hospital tier` = "?";

update hospitalisation_details
set `City tier` = "tier - 2"
where `City tier` = "?";
SET SQL_SAFE_UPDATES = 1;

select `Hospital tier`, `City tier` , avg(charges) as avg_charges
from  hospitalisation_details
group by `Hospital tier`,`City tier`;
```

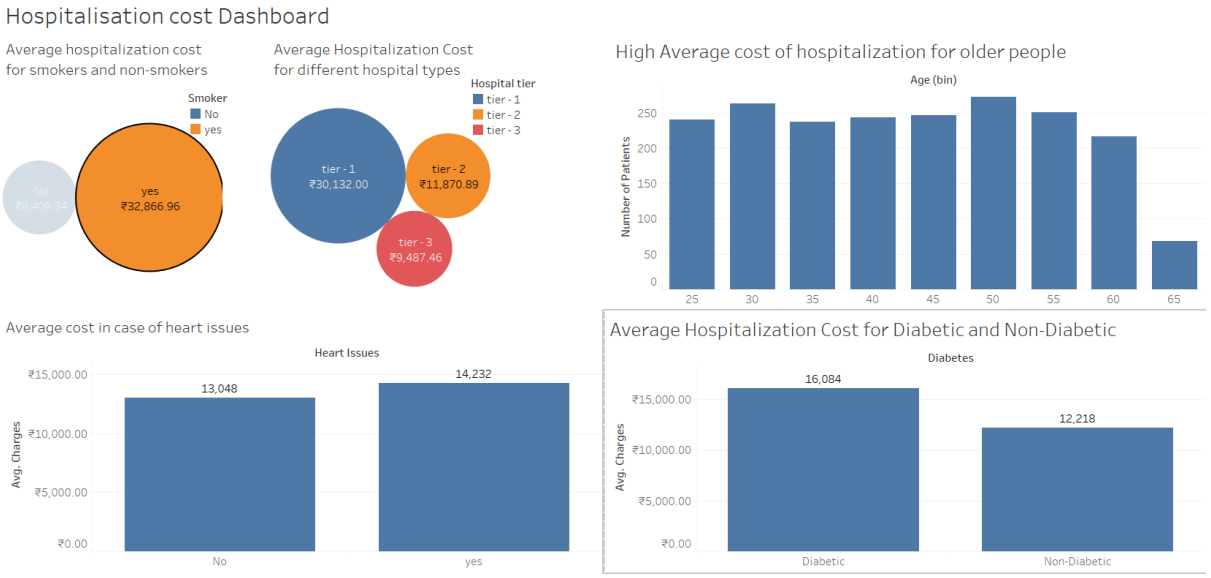
```
/* How many individuals who have had any major surgeries have cancer history? */
select `Cancer history`, surgery, count(*) as count_patients
from (
select *,
case
when NumberOfMajorSurgeries>= 1 then "Yes"
else "No"
end as surgery
from  medical_examinations) m
group by `Cancer history`, surgery
having `Cancer history` = "yes";

/* Find out how many Tier-1 hospitals in each state.*/

# replace "?" in state id with mode value
select * from hospitalisation_details ;
select `State ID` , count(*) as ct
from hospitalisation_details
group by `State ID`
order by ct desc;
SET SQL_SAFE_UPDATES = 0;
update hospitalisation_details
set `Hospital tier` = "tier - 2"
where `Hospital tier` = "?";
select * from hospitalisation_details where `State ID`='?';
select count(`State ID`) ,`State ID` from hospitalisation_details Group By `State ID`;

update hospitalisation_details
set `State ID` = "R1013"
where `State ID` = "?";
select `State ID`, `Hospital tier` , count(*) as hospital_count
```

from hospitalisation_details

group by `State ID`, `Hospital tier`

having `Hospital tier` = "tier - 1";

Tableau Screenshots:



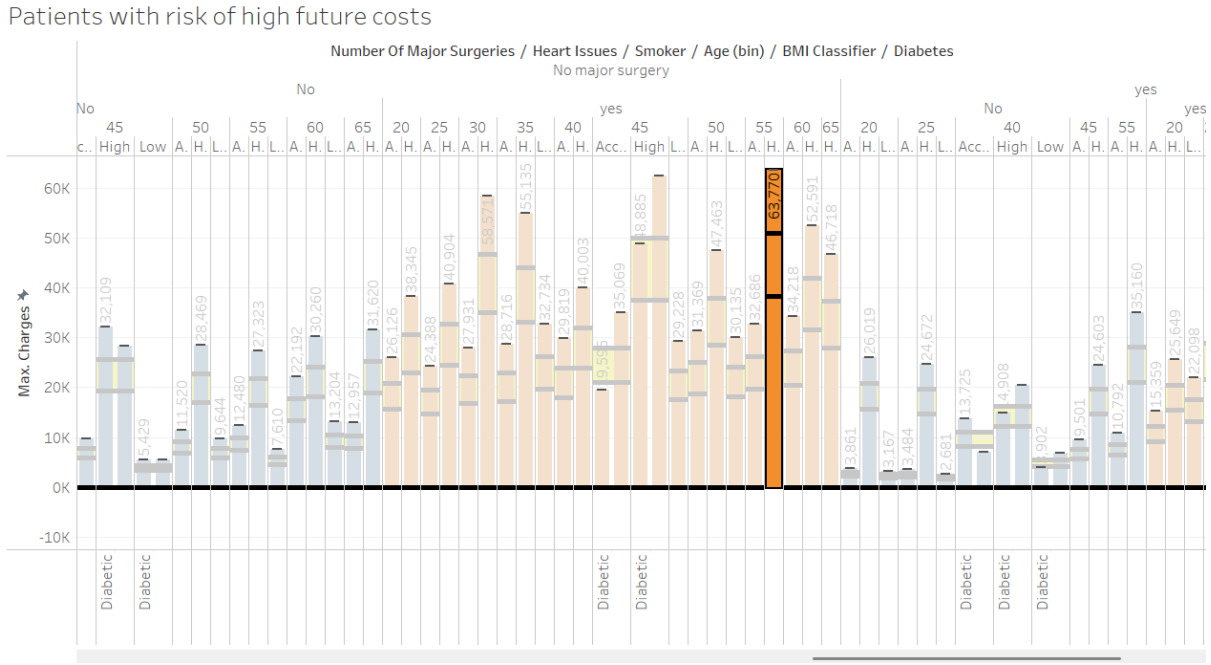


Tableau story link:
https://public.tableau.com/views/Capstone_17481784158420/UnderstandingandPredictingHealthcareCosts?:language=en-US&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link