**WRITE UP- CAPSTON HEALTH INDUSTRY**

# Brief

This a a health industry project that analyses data from hospital admission of patients, their medical history and personal details. The analysis focuses on ultimately creating a model that can predict the expenditure of admission of a patient based on his history and geographic details and hospital of admission.

The secondary part of the project is querying created sql database to extract specific data and finally presenting this story through a tableau visualization story.

The steps I followed to execute the project is below.

**Data Collation and Cleaning**

1. **Libraries Used:** The libraries from sklearn for machine learning algorithms used, apart from the standard panda, numpy seaborn etc were imported. The project was run in a google colab environment and the notebook is posted in my github repository.
2. **Read Data:**

Data was imported from the provided csv files. I looked up the text files to check the formats and anomalies. The data was about 2300 rows with a ***common field as Customer ID.***

1. **Combine Dataframes:**

Used the inner join to merge the tables into **single df- merged\_data**

This was done in two steps using the following statements:-

merged\_data = pd.merge(left=hosp\_data, right=med\_data, on=merge\_col, how='inner')

merged\_data = merged\_data.merge(right=names\_data, on=merge\_col, how='inner')

It was checked to see if there were duplicates. The df was then cursourily checked with .info(), .head(),.columns. The import was rechecked with he csv and was ok.

1. **Missing Value Analysis:**

Used the .isna and isnull to check any missing values.

# Check for missing values

print(merged\_data.isnull().sum())

Each data field was checked for dtype. All the errors like’?’ were removed using Boolean mask by filtering and inversing.

*mask = merged\_data.apply(lambda row: '?' in row.values, axis=1)*

* Since these rows were just about 10, I chose to delete them rather than impute.
* I also checked all columns for unique values.
* Fields like Yes No has inconsistent capitalization and were all reduced to lower cases. All numeric data was converted to their respective int/float data type.

1. **Data Transformation:**
   * **NumberOfMajorSurgeries:** This field was converted to numeric from string.
   * **Categorical Variables:**
     + I used Label encoding for ordinal variables (city and hospital tier).
     + One-hot encoding for nominal variables : The only nominal variable was State ID which was not considered for onehot encoding due to incr in dimensionality. (16 States)

**Feature Engineering:**

1. **State ID:** Created three dummy variables for R1011, R1012, and R1013 to capture potential state-specific effects. These were the top states.

* **State ID Sum charges**
* R1011 11173625.27
* R1012 6847930.72
* R1013 6408232.33

Standard get\_dummies function was used and the df was concataned with the merged\_data df.

# Create dummy variables, dropping the first column to avoid redundancy

dummies = pd.get\_dummies(merged\_data['State ID'], prefix='State', drop\_first=True)

# Filter for desired state IDs

dummies = dummies[['State\_R1011', 'State\_R1012', 'State\_R1013']]

# Concatenate with the original DataFrame

merged\_data = pd.concat([merged\_data, dummies], axis=1)

1. **Age Calculation:**

I used the pd.to\_datetime () function to make a dob field from the year,month and date field. Thereafter I converted the dob into age in decimal format.

def calculate\_decimal\_age(born):

    today = datetime.date.today()

    #subtract current year from birth year and adjust if your birthday has past in current year

    years = today.year - born.year - ((today.month, today.day) < (born.month, born.day))

    days = (today.day - born.day)

    return years + days / 365.25  # Adjust for leap years

merged\_data['age'] = merged\_data['dob'].apply(calculate\_decimal\_age)

* On second thoughts this was unnecessary as age when analysed will generally be binned. All continuous variables however give better linear regression results.
* This was not a elegant way but learnt about the function. As it could have been required in case of any time analysis.
* I also learnt that we need to change the variable name from ’date’ to ‘day’ as there is some clash with current field name.

1. **Gender from Salutations:**

A function was made to extract gender from name field. Used apply() to use it for all rows.

def extract\_gender(name):

if 'Mr.' in name:

return 'Male'

elif 'Ms.' in name or 'Mrs.' in name:

return 'Female'

else:

return np.nan

merged\_data['gender'] = merged\_data['name'].apply(extract\_gender)

# Handle cases where gender cannot be determined from salutations

**Data Visualization and Exploration (Week 1, Tasks 9-12):**

1. **Hospitalization Cost Distribution:** Used the standard plots.

A graph of blue bars

Description automatically generated with medium confidence

A chart of different sizes and colors

Description automatically generated with medium confidence

## Analysis

The data crealy shows outliers in the charges. Though these are not data anomalies but they are not good for modelling hence I have decided to clip them.

## Removing outliers

Q1 = merged\_data['charges'].quantile(0.25)

Q3 = merged\_data['charges'].quantile(0.75)

IQR = Q3 - Q1

upper\_bound = Q3 + 1.5 \* IQR

lower\_bound = Q1 - 1.5 \* IQR

filtered\_df = merged\_data[(merged\_data['charges'] < upper\_bound) & (merged\_data['charges'] > lower\_bound)]

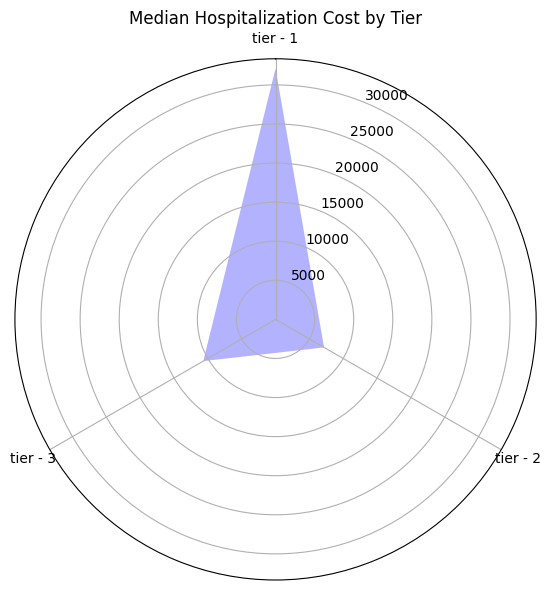
The rectified plots are :-

A diagram of a number of boxes

Description automatically generated with medium confidence

1. **Median Cost by Hospital Tier (Week 1, Task 11):**

A polar radar plot was made to map the median costs.



**Basic code**

# Calculate median cost per hospital tier

median\_costs = merged\_data.groupby('hospital\_tier')['charges'].median()

**Hypothesis Testing (Week 1, Task 13):**

## ANOVA for Hospital/City Tiers:

**f\_oneway function (SciPy):**  one-way ANOVA test used

**Hypothesis:**

*Null Hypothesis (H0):* The means of hospitalization charges for tier-1, tier-2, and tier-3 hospitals are the same (no significant differences).

*Alternative Hypothesis (H1):* At least one of the hospital tiers has a significantly different mean hospitalization charge compared to the others.

**Output Interpretation**

**F-statistic (493.98956631117636):**

A measure of the ratio of between-group variability to within-group variability.

A large F-statistic suggests that the groups have significantly different means.

**p-value (1.7738221310852664e-179):**

The probability of obtaining an F-statistic as extreme (or more extreme) as the one observed if the null hypothesis (H0) were true.

An extremely small p-value (e.g., less than 0.05) is often used as a threshold to reject the null hypothesis.

**Conclusion**

The very large F-statistic and the extremely small p-value (close to zero) strongly suggest that there are significant differences in mean hospitalization charges between at least one pair of hospital tiers. **We would reject the null hypothesis (H0) and conclude in favor of the alternative hypothesis (H1)**.

## T-Test for Smokers/Non-Smokers:

Basic code

from scipy.stats import ttest\_ind

smokers = merged\_data[merged\_data['smoker'] == 'yes']['charges']

non\_smokers = merged\_data[merged\_data['smoker'] == 'no']['charges']

t\_statistic, pvalue = ttest\_ind(smokers, non\_smokers)

Hypotheses:

Null Hypothesis (H0): The mean hospitalization charges for smokers and non-smokers are equal.

Alternative Hypothesis (H1): The mean hospitalization charges for smokers and non-smokers are not equal.

Output Interpretation

T-statistic (1.4543557561814688): A measure of how many standard deviations apart the group means are. A larger absolute value suggests a greater difference in means.

p-value (0.0): The probability of obtaining a t-statistic as extreme (or more extreme) as the observed one if the null hypothesis (H0) were true. A p-value close to zero suggests substantial evidence against the null hypothesis.

Conclusion

The t-test result with a p-value of 0.0 indicates strong evidence to reject the null hypothesis (H0), supporting the alternative hypothesis (H1). **Therefore, we conclude there's a statistically significant difference in mean hospitalization charges between smokers and non-smokers.**

## Chi-Square for Smoking/Heart Issues:

Basic Python

contingency = pd.crosstab(merged\_data['smoker'], merged\_data['Heart Issues'])

chi\_square, pvalue, dof, expected = stats.chi2\_contingency(contingency)

Let's interpret the results of your chi-square test for independence between smoking and heart issues.

* **Hypotheses:**
  + *Null Hypothesis (H0):* Smoking and heart issues are independent (no association).
  + *Alternative Hypothesis (H1):* Smoking and heart issues are not independent (there is an association).

**Output Interpretation**

* **Chi-Square Statistic (0.08588150449910656):** Measures the discrepancy between observed frequencies in your contingency table and the frequencies expected if the variables were independent. A larger value suggests a greater deviation from independence.
* **p-value (0.7694797581780767):** The probability of obtaining a chi-square statistic as extreme (or more extreme) as the observed one if the null hypothesis (H0) were true. A high p-value indicates that the observed data is consistent with what we'd expect if there were no association.

**Conclusion**

The small chi-square statistic and the relatively large p-value (0.7694797581780767) suggest that we fail to reject the null hypothesis (H0). **This means there's insufficient evidence to conclude that smoking and heart issues are associated within your dataset.**

**Correlation Analysis (Week 2, Task 1):**

Basic Python

# Create a correlation matrix

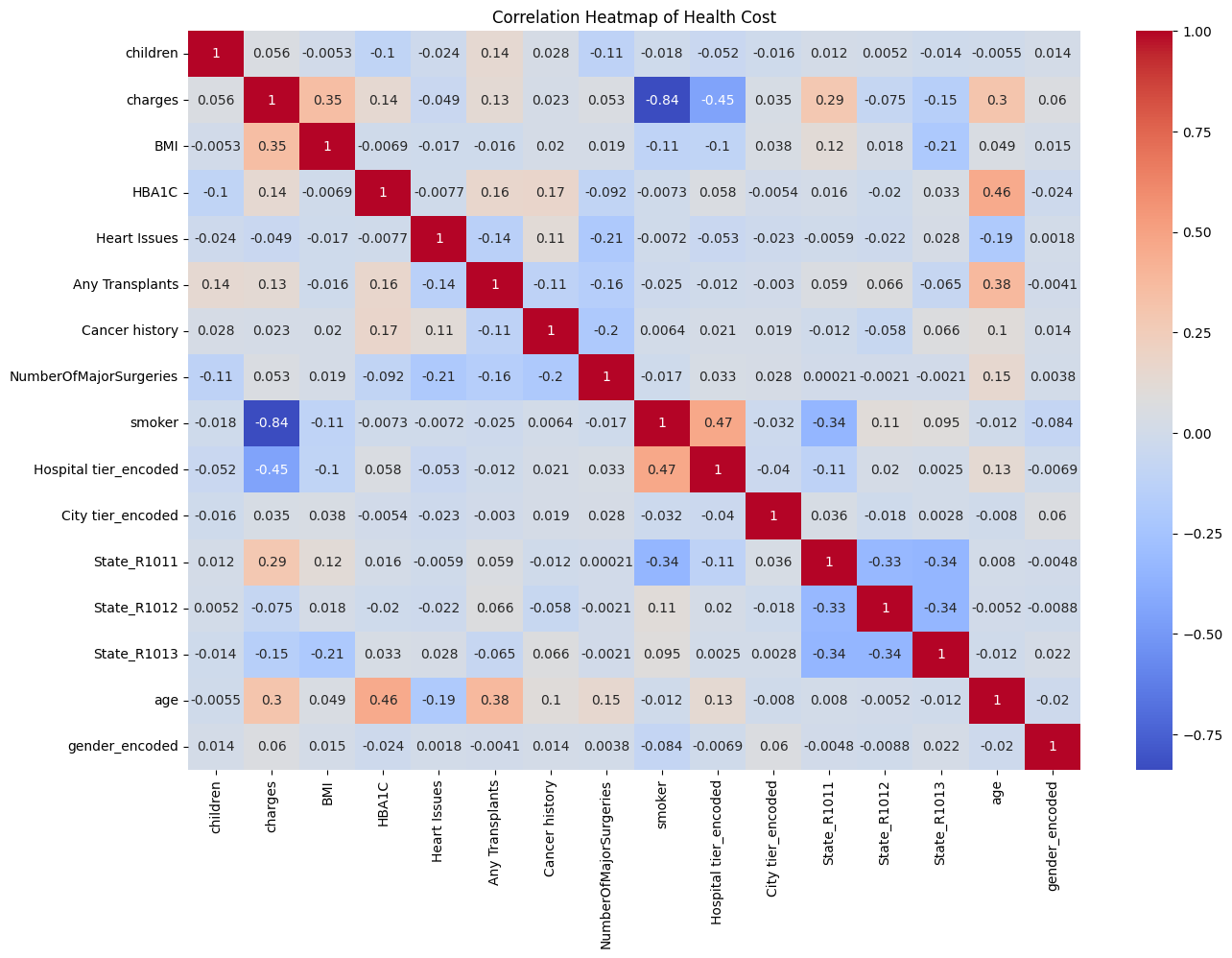
correlation\_matrix = merged\_data.corr()

# Visualize with a heatmap

sns.heatmap(correlation\_matrix, annot=True)

plt.show()

**Analysis**



1. **charges and BMI (0.346730):** A moderate positive correlation. This suggests that as BMI increases, hospitalization charges also tend to increase.
2. **children and Heart Issues (-0.023984):** A very weak negative correlation. There might be a slight tendency for people with fewer children to have a higher likelihood of heart issues, but this relationship is likely not very strong.
3. **smoker and charges (-0.838462):** A strong negative correlation. This indicates a substantial association where smokers tend to have significantly lower hospitalization charges. It's crucial to investigate potential confounding factors here.
4. **State\_R1011 and charges (0.286956):** A moderate positive correlation, suggesting that individuals located in State R1011 might experience higher hospitalization charges.

**Regression Modeling and Development (Week 2, Task 2):**

* **Split Data:** the data set was split
* **Linear Regression:** Used the liner regression model
* I used both the Kfold and stratified Kfold. To model using two different pipelines.
* I have got better results with the K fold algorithm with 10 splits

KFold Score: 0.8354979292132555

KFold Score: 0.8619909288985694

KFold Score: 0.8522422452011675

KFold Score: 0.859600534862711

KFold Score: 0.89847713872181

KFold Score: 0.8429360120957613

KFold Score: 0.8638520353778542

KFold Score: 0.886022659515102

KFold Score: 0.8254895341436217

KFold Score: 0.8967889334263194

* Used the **gradient boosting algo** to extract features

The results are

RMSE: 3953.611206994061

R-squared: 0.8904177712069243

Mean Squared Error: 15631041.576069035

**Features**

A graph with blue and white bars

Description automatically generated

I have retained all features as removing them reduces the model accuracy which is already at 85%

## Outlier treatment

I realized that outlier treatment is reducing the accuracy of my model. So I have dropped it. I have refitted my model with the original daya.

Important learning point is Outlier removal has to be done with great care.

**Case Scenario (Week 2, Task 3):**

I fed the data from the example through a dictionary. Used the Kfold algo to generate predictions.

Basic Code

# Use best model to predict

prediction = pipeline.predict(new\_data) # Or gbr.predict()

print(f"Predicted Hospitalization Cost: {prediction[0]}")

My result

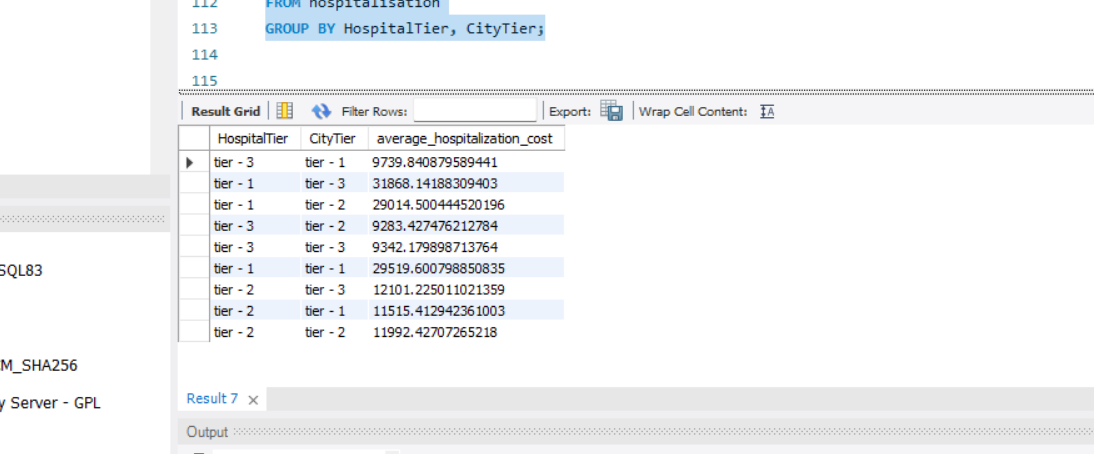
**Predicted Hospitalization Cost: 28076.992759026056**

**SQL Queries (Week 2, Tasks 1b - 5):**

* The queries were simple.
* Data structure of table was made based on datatype.
* I cleaned the data in python before uploading.
* Data was merged based on primary key into a new table.
* Query results are:-
* A screenshot of a computer

  Description automatically generatedA screenshot of a computer

  Description automatically generatedA screenshot of a computer

  Description automatically generatedA screenshot of a computer

  Description automatically generated

**Tableau Dashboard (Week 2, Task 6):**

For tableau visualization I made a stroty based on following insights.

I have made calculated fields to define diabetic as HBAIC>6.5 and people with and without prior problems. The visualisations were based on:-

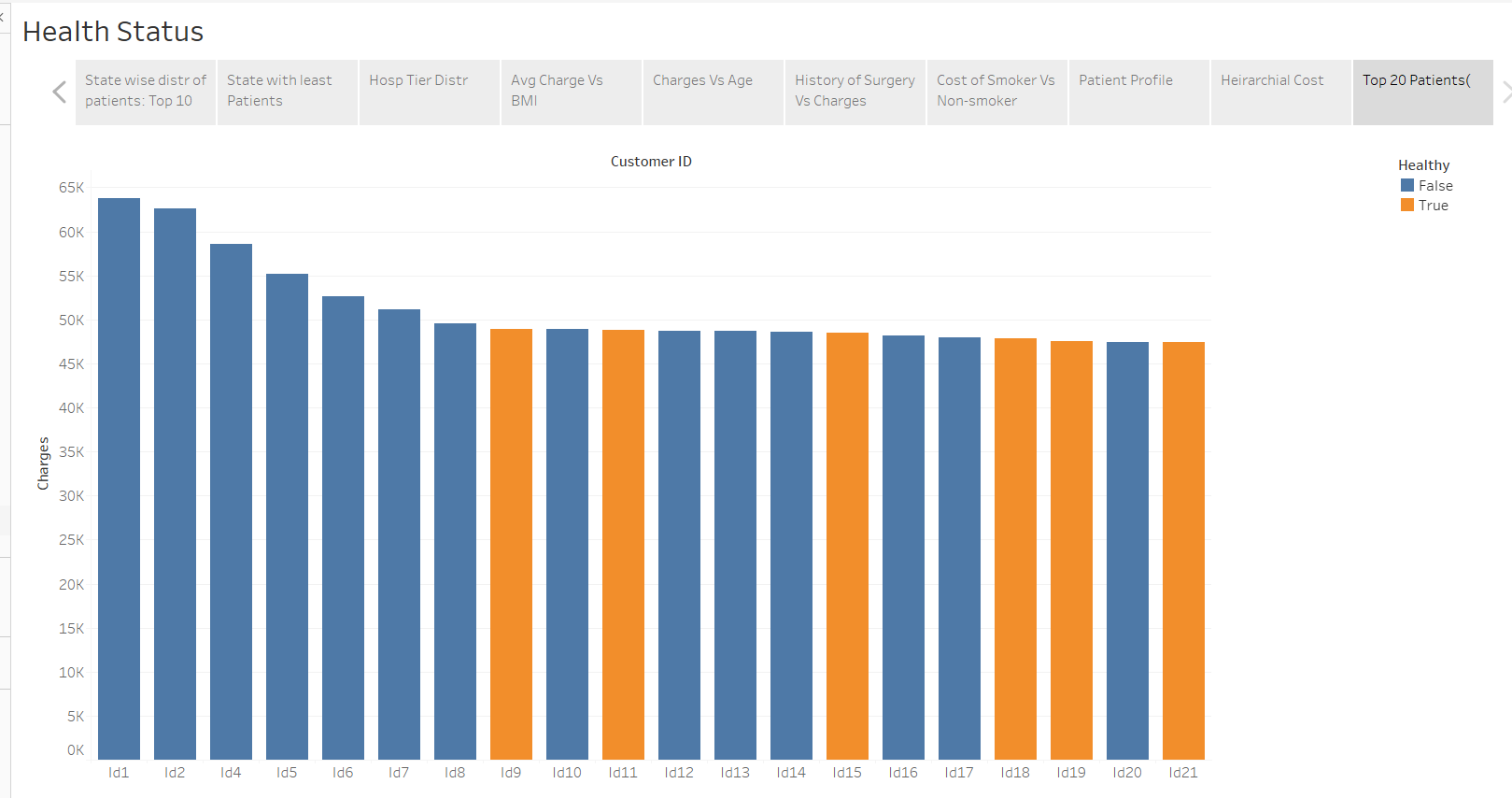
1. **Bar Charts**
   * **Comparing Categories:** Count of patients by state, city tier, hospital tier, conditions, etc.
   * **Distribution:** Distribution of charges across different groups.
2. **Scatter Plots**

* **Correlations:** Charges vs. BMI, charges vs. number of children, age vs. number of major surgeries.

1. **Box Plots**
   * **Distribution Analysis:** Distribution of charges by hospital tier, city tier, or medical conditions.
   * **Implementation** Categorical variable to 'Columns', numerical measure to 'Rows'. Tableau will generate box plots.

**Story Concept**

* **Overall Health Snapshot:**
  + Total patients, average charges, average BMI, top medical conditions (bar charts).
  + State wise details with filters for health conditions, smoker etc
* **Cost Analysis Dashboard:**
  + Charges distribution (histogram).
  + Charges vs. various factors (scatter plots).
  + Bar charts for comparing costs by hospital tier, state.
* **Patient Profile Dashboard**
  + Filters for Customer ID
  + Key metrics: Sum charges, number of major surgeries, BMI

A screenshot of a medical report

Description automatically generated