Nosocomial Infections: The Factors at Play

# Introduction

Even in our modern age, nosocomial infections are still a real risk in today’s healthcare facilities. Also known as a hospital-acquired infection, or HAI, a nosocomial infection is an infection acquired in a hospital, rehabilitation center, nursing home, or any other center of healthcare. With healthcare centers constantly filled with patients with infectious diseases or weakened immune systems or both, this is a more common occurrence then the layperson would think. Historically, the first person to notice this phenomenon was Ignaz Semmelweis in 1841, a Hungarian obstetrician working in a Vienna maternity hospital. He noticed the high number of fevers women after labor would develop and the higher mortality rate in the ward where medical students would deliver babies vs. the ward where midwife students would deliver them. After mandating handwashing, infection rates are recorded to have dropped significantly. It was not until 1865 when Louis Pasteur developed the germ theory of disease to identify microorganisms, such as bacteria and fungi, as the culprits in nosocomial infections.

According to the Centers for Disease Control and Prevention in the United States, 1.7 million hospital-acquired infections contribute to 99,000 deaths each year, while in Europe hospital surveys report that gram-negative infections are estimate to contribute to two-thirds of the 25,000 deaths each year. Nosocomial infections can cause a variety of illnesses, such as pneumonia and infections in the bloodstream, urinary tract, and other parts of the body.

# Material and Methods

“What factors contribute to nosocomial infections?”

Our goal is to investigate the most significant factors at play in determining infection risk in healthcare centers. Our method of choice will be linear regression, a statistical method used to find linear relationships between a response/target variable, in our case infection risk, and its possible predictors. For training our linear regression model, we shall only use patient ids **6 to 113**, so 108 observations. To create our model, we shall use ***Feature Selection*** *wrapper methods*, specifically **forward selection, backward elimination, and stepwise selection** to pick the best predictors of infection risk based on their **p-values**. Finally, we shall use patient ids **1 to 5** to test our model’s predictive power of infection risk. In statistical terms, we shall also test our predictors, or betas (**β**), first individually with a **t-test** to verify that they have a significant impact on infection risk. Then, we shall test the significance of our model wholly using the **F-test**. Thus, our null & alternative hypotheses (**Ho** & **Ha**) for the t-test and F-test respectively are as follows:

* This predictor has no impact, or is not significant, on infection risk vs. this predictor significantly impacts infection risk (**βj = 0 vs. βj ≠ 0**).
* This model is not significant in predicting infection risk vs. this model is significant in predicting infection risk (**β1 = β2 =βk = 0 vs. βj ≠ 0 for at least one j**).

We will be working with a level of significance, or alpha, of 0.05 (**α = 0.05**). We will also be using Python in a Jupyter Notebook for this analysis. A link to this notebook will be provided in the source code.

## Specimens used for this study

The sample collected is a random sample of patients from an extract of the Study on the Efficacy of Nosocomial Infection Control (SENIC). It is assumed representative of the population and free of bias. The variables, or predictors we will be investigating in determining infection risk (**excluding Identification number**), in the data are as follows: **Length of stay, Age, Routine culturing ration, Routine chest X-ray ratio, Number of beds, Medical School Affiliation (categorical), Average daily census, Number of nurses, & Available facilities and services.**

## Statistical analyses

To begin, we loaded, cleaned, and manipulated the data into two tables, a “train” dataset consisting of patient ids 6 to 113 to create and train our model, and a “test” dataset consisting of patient ids 1 to 5 to test our finished model.

**Exploratory Analysis**

After running descriptive statistics of the data, we used a correlation table and a heatmap to check which variables correlated with the target variable, infection risk, the most. The results can be seen in **Figures 1 and 2** below.

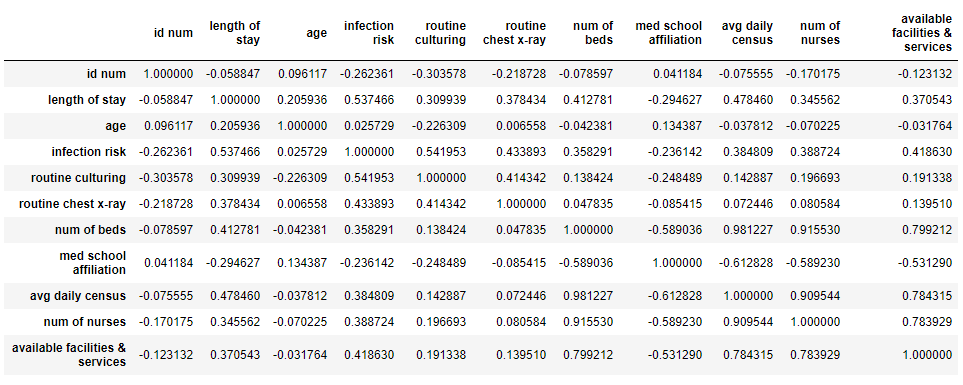


Figure : Pearson-Correlation table of data.

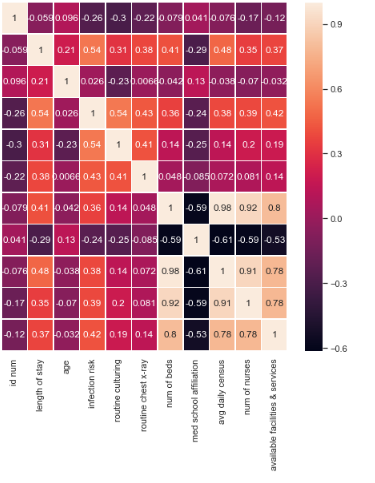
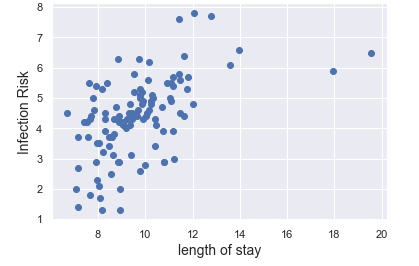
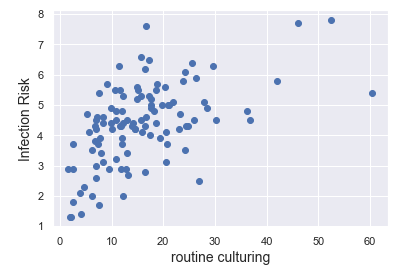


Figure : Pearson-Correlation Heatmap of data. (Y-axis corresponds with X-axis)

The heatmap serves to view to correlation of the variables more easily than with the table. Both confirm that the variables “length of stay” (**54%**), “routine culturing” (**54%**), “routine chest x-ray” (**43%**), and “available facilities & services” (**42%**) have the highest correlation with infection risk, meaning they have the highest predictive power of our target variable. To further check their linearity with infection risk, we plotted scatterplots of each of these variables against infection risk, as seen in **Figure 3** below.

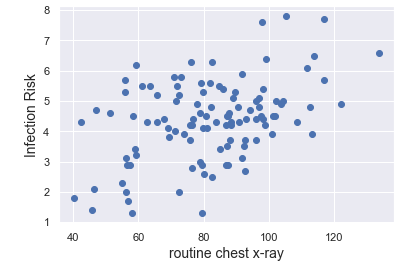
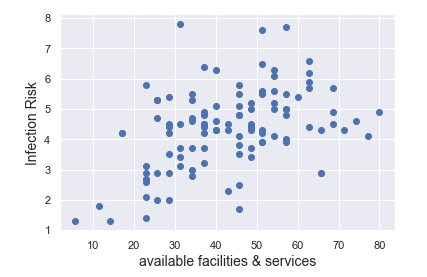
 

Figure : Scatterplots of highest correlating variables against infection risk.

As seen in these scatterplots, each of these variables seem to have a **linear** and **positively correlated** relationship with the target variable, infection risk. This means that one can picture a straight line fitted across the dots from left to right, and that as the predictor variable increases, so does infection risk.

**Model Building: Feature Selection**

Now that we better understand our data and what are the likely variables to be chosen by the following procedures, we are now ready to build our model to predict infection risk. To assure we build the best model, we shall utilize the three feature selection-wrapper method techniques mentioned before: **forward selection, backward elimination, and stepwise selection**. By using all three, we can be more secure in having a best fitted model.

After coding the functions necessary to initiate these procedures, we have our selected best features for predicting infection risk, and with them, our best fitting models. The following figures, **Figures 4, 5, & 6**, show the best selected features and model summaries of the forward selection, backward elimination, and stepwise selection methods, respectively.

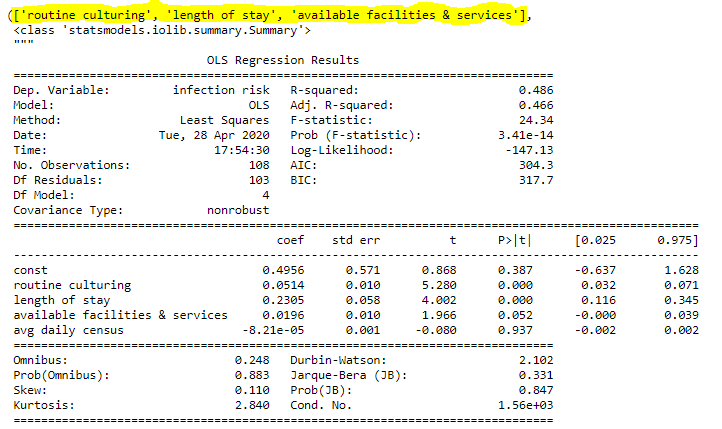
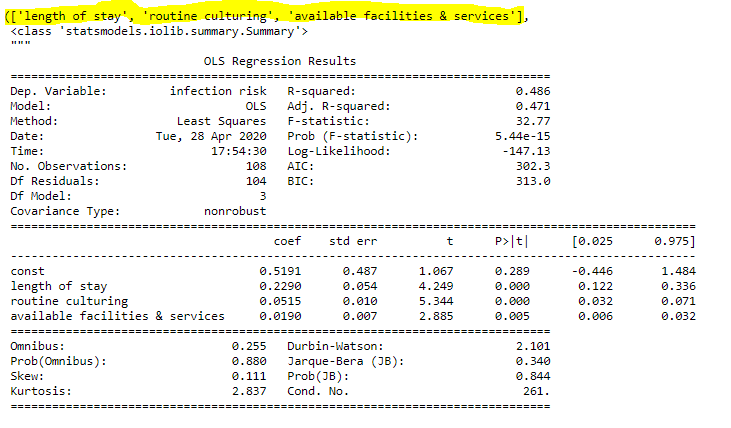


Figure : Forward Selection model summary. Best features highlighted.

Figure : Backward Elimination model summary. Best features highlighted.

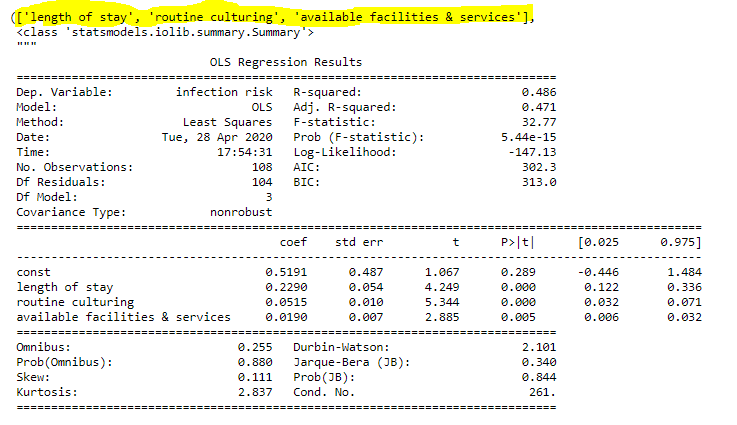


Figure : Stepwise Selection model summary. Best features highlighted.

To start, each procedure selected the same variables, or features, as the best ones for predicting infection risk: length of stay, routine culturing, and available facilities & services. **This agrees with the correlation graphs we created in our exploratory analysis**. All procedure outputs also produce the same **R-squared** values, **.486**, with an **Adjusted R-squared** of little difference between forward selection and the other two **(.466 vs. .471**). The only differences between the output of the procedures are the models they produced. Forward selection produced the model **y = 0.4956 + 0.0514x1 + 0.2305x2 + 0.0196x3**. Backward elimination and Stepwise selection both produced the model **y = 0.519 + 0.2290x1 + 0.0515x2 + 0.0190x3**. From the R-squared and Adjusted R-squared values, we can conclude that both models have the same predictive power, accounting for about **46.6% to 48.6%** of the data at best.

From all models, we can also answer our t and F hypothesis tests. Our t-test critical value, two-tailed, is **1.98**, while our F-test critical value is **2.69**. In each model, all the variables “pass” the t-test, except available facilities & services in the forward selection model (**1.966 < 1.98**). Overall however, we can reject our null hypothesis **βj = 0** in favor of the alternative **βj ≠ 0**, meaning that each predictor is statistically a significant factor in predicting infection risk. As for our F-test, all models “pass”, since each model has an F-statistic notably higher than 2.69 (Forward Selection: **24.34**, Backward Elimination & Stepwise Selection: **32.77**). This means we can reject our null hypothesis **β1 = β2 =βk = 0** in favor of the alternative **βj ≠ 0 for at least one j** because each model is statistically proven to be a good fit for predicting infection risk.

To conclude this section, we will settle on choosing the model produced by backward elimination and stepwise selection as our model of reference for the rest of this report. Thus, **let model =** **y = 0.519 + 0.2290x1 + 0.0515x2 + 0.0190x3** since this model has a slightly higher R-squared value and passed both hypothesis tests.

**Residual Analysis**

Now that we have our model, we must confirm the linear regression assumptions to assure our model’s quality and predictive power. There are five key assumptions that need to be confirmed for a multiple linear regression model to be considered proper:

* **Linearity**: the response variable, infection risk, is a linear function of each independent variable.
* **Independence of errors/residuals**: the errors, or residuals, are independent of each other.
* **Homoscedasticity (constant variance)**: the variance of the errors is constant throughout the progression of y = x, from left to right.
* **Normality:** errors are generated from a normal distribution. While this is not as key as the other four assumptions, this is necessary if one wishes to easily calculate the confidence and prediction intervals of observations and predictions.
* **Multicollinearity:** Important for statistical inference, this assumption demands no or minimal linear dependence between the predictors.

We confirmed the linearity assumption earlier in our exploratory analysis (refer to **Figure 3**) so we shall move on to the other four, starting with independence of errors. To confirm the independence assumption, we shall plot the residuals versus each of our selected predictors. **If the residuals are distributed uniformly and randomly around the zero x-axes and do not form clusters, then the assumption holds true.**

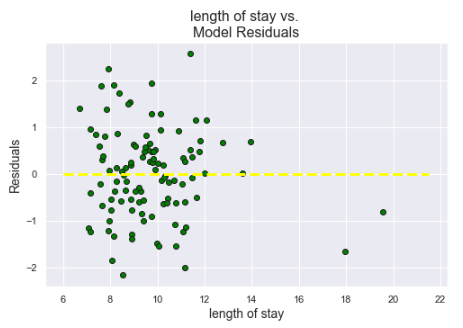
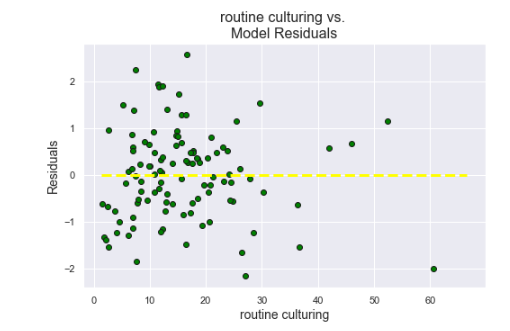
 



Figure : Model Residuals for each predictor.

In **Figure 7**, while some clustering is evident in the “length of stay” and “routine culturing” predictors, overall, the independence assumption holds up since the distribution or the residuals is mostly random around the 0 axis.

Moving on to homoscedasticity, or constant variance, we shall plot the fitted responses vs. the residuals. By doing this, we are checking for **constant variance of the residuals as the response variable increases.** If this assumption fails, perhaps a variable transformation of some kind will be needed to improve our model.

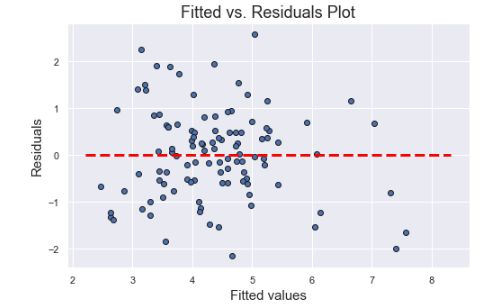


Figure : Fitted vs. Residuals Plot to check for Homoscedasticity.

In **Figure 8** above, since there is no cone-shape among the residuals as they progress along the 0 axis, so the constant variance assumption is met.

For the normality assumption, we shall create Histogram and Q-Q plots of the normalized residuals. If the histogram **follows the bell-curve shape of normal Gaussian distribution** and the normalized residuals of the Q-Q plot **mostly follow the plotted line**, the assumption will be met.

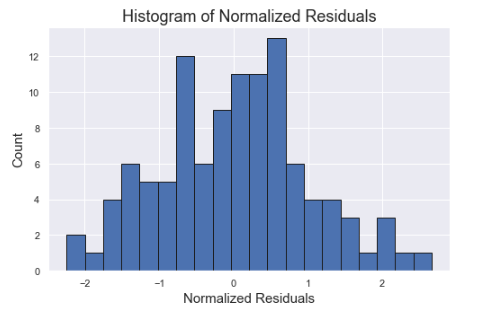


Figure : Histogram of Normalized Residuals.

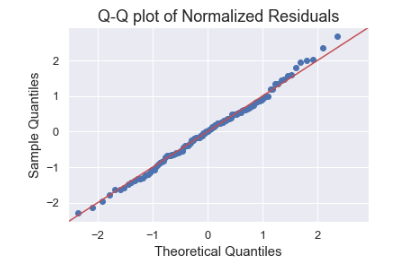


Figure : Q-Q plot of Normalized Residuals.

In **Figures 9 & 10**, we can see that our histogram and Q-Q plot achieve their respective criteria, following the bell-curve and red plotted line. Thus, our normality assumption is met.

Finally, we shall check for multicollinearity within our model. While we can check the heatmap or table from our exploratory analysis, a better way to check this assumption is to use the Variance Inflation Factor, or VIF. This is like a multicollinearity score for each of our predictors. **A VIF >10 means significant multicollinearity, while a VIF < 10 is insignificant**. Note, however, that the higher the VIF, the more significant the multicollinearity.



Figure : Result of VIF function in python.

As seen in **Figure 11**, it seems our “length of stay” and “available facilities & services” predictors show **significant multicollinearity** with each other. This makes sense since how long a patient stays in the hospital must obviously have to do with the services and resources the healthcare center provides. The usual procedure to address this would be to drop these variables from our model, but since they are our best predictors, this most likely will take away from our model’s predictive power.

**Making Predictions with our Model**

The whole point of this process was to build a model that can reasonably predict infection risk. After all these procedures, we can finally do that with our “test” data, **patient ids 1 to 5**. To do this, we first modify the test data to consist of only our best predictors (length of stay, routine culturing, and available facilities & services). Then, we will run it through the model. **Figures 12 & 13** below show the confidence and prediction intervals of our test data, as well as the predicted estimates.

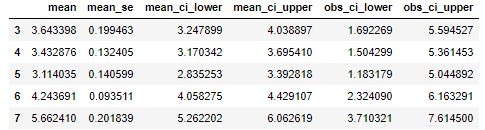


Figure : Confidence & Prediction intervals of IDs 1-5 (3-7). "mean\_ci" = confidence intervals & "obs\_ci" = prediction intervals.

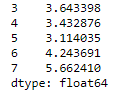


Figure : Prediction estimates of infection risk for test data observations.

We can compare these results to the recorded infection risk of patient ids 1-5 from the original sample to gauge our model’s accuracy:

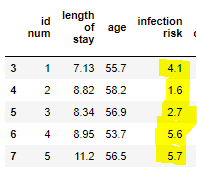


Figure : Recorded samples patient ids 1-5. Infection risk highlighted.

Comparing **Figure 14** with the values of **Figure 13**, we can say that most of our predictions were not as close as we would like, except for patient id 5. In regards to the intervals in Figure 12, while the recorded samples mostly don’t fall within our model’s confidence intervals, **they do fall within the prediction intervals**, so we can take comfort in that our model was able to at least get the prediction interval for each observation right. With a R-squared value of **48.6%,** an outcome like this is not surprising.

# Results

To conclude, we can say our model needs more factors to be able to predict with increased accuracy infection risk. While it makes sense why length of stay, routine culturing, and available facilities & services were selected as the best predictors of infection risk, they obviously are not the whole story.

To improve what we do have, we can perhaps look to the outliers in our data. For example, using Cook’s Distance, we can check if there is an outlier that would have a significant influence on our model if it is deleted. Check **Figure 15**:

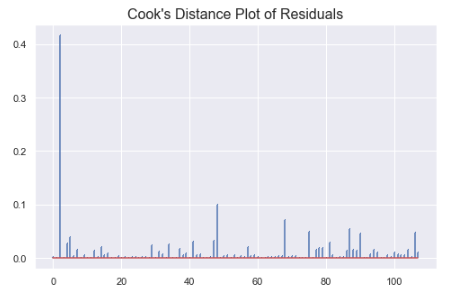


Figure : Cook's Distance Plot of Residuals

While there are not many significant outliers, perhaps the deletion of the most obvious one, to the far left of the graph, may help our model’s predictive power, but only slightly.

If we were to further investigate this, I recommend sampling data which includes a factor that scores for the quality of sanitation a healthcare facility has, using criteria such as hand washing, presence of rodents, preventive measures against germ spread, use of gloves, etc. Another factor that could prove significant is the average of how many patients per room in a healthcare facility. Working with that dataset, we can probably build a better model.

# References

Python Software Foundation. Python Language Reference, version 2.7. Available at <http://www.python.org>

Luhaniwal, Vikashraj. “Feature Selection Using Wrapper Methods in Python.” Medium, Towards Data Science, 4 Oct. 2019, towardsdatascience.com/feature-selection-using-wrapper-methods-in-python-f0d352b346f.

Sarkar, Tirthajyoti. “How Do You Check the Quality of Your Regression Model in Python?” Medium, Towards Data Science, 7 June 2019, towardsdatascience.com/how-do-you-check-the-quality-of-your-regression-model-in-python-fa61759ff685.

Tirthajyoti. “Tirthajyoti/Machine-Learning-with-Python.” GitHub, github.com/tirthajyoti/Machine-Learning-with-Python/blob/master/Regression/Regression\_Diagnostics.ipynb.

“Hospital-Acquired Infection.” Wikipedia, Wikimedia Foundation, 27 Apr. 2020, en.wikipedia.org/wiki/Hospital-acquired\_infection#cite\_note-3.

# Appendices

## Source Code

***To view the code’s performance, open with the Jupyter Notebook file using Microsoft Azure or Ananconda!***

**#!/usr/bin/env python**

**# coding: utf-8**

**# Credits to Vikashraj Luhaniwal from TowardDataScience.com for his article on**

**# Feature Selection Using Wrapper Methods for the feature selection code I use in this**

**# notebook. Also credits to Dr. Tirthajyoti Sarkar from TowardDataScience.com for**

**# his article on How to Check the Quality of a Regression Model with Python and the code**

**# I use from his github to do the residual analysis in this notebook. Links to these articles**

**# are posted here:**

**# https://towardsdatascience.com/feature-selection-using-wrapper-methods-in-python-f0d352b346f**

**# https://towardsdatascience.com/how-do-you-check-the-quality-of-your-regression-model-in-python-fa61759ff685**

**# In[5]:**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**sns.set()**

**import statsmodels.api as sm**

**import scipy.stats**

**from sklearn.linear\_model import LinearRegression**

**# In[6]:**

**raw\_data = pd.read\_csv('Infections.csv')**

**raw\_data.head()**

**# In[7]:**

**# Renaming of columns and cleanup**

**raw\_data.dropna(0, how='any', inplace=True)**

**raw\_data.rename(index=str, columns={'The SAS System': 'id num',**

**'Unnamed: 1': 'length of stay',**

**'Unnamed: 2': 'age',**

**'Unnamed: 3': 'infection risk',**

**'Unnamed: 4': 'routine culturing',**

**'Unnamed: 5': 'routine chest x-ray',**

**'Unnamed: 6': 'num of beds',**

**'Unnamed: 7': 'med school affiliation',**

**'Unnamed: 8': 'avg daily census',**

**'Unnamed: 9': 'num of nurses',**

**'Unnamed: 10': 'available facilities & services'}, inplace=True)**

**raw\_data.drop('1', inplace=True)**

**raw\_data.head()**

**# # An analysis of 108 obs, ID's 6-113**

**# In[8]:**

**data = raw\_data.copy()**

**data.drop(['3','4','5','6','7'], axis=0, inplace=True)**

**raw\_data.head()**

**# In[9]:**

**data**

**# # Exploratory Analysis**

**# \*\*Target Variable: infection risk\*\***

**# In[10]:**

**# Descriptive Statistics**

**data = data.astype(float)**

**print(data.shape) # 108 rows, 11 columns**

**data.describe()**

**# In[11]:**

**# Pairwise scatter plots and correlation heatmap to check for**

**# multicollinearity**

**sns.pairplot(data)**

**# In[12]:**

**# Table of Pearson correlations between each feature**

**data.corr()**

**# In[13]:**

**# Heatmap to see correlations easier**

**f, ax = plt.subplots(figsize=(8, 8))**

**sns.heatmap(data.corr(), annot=True, linewidth=.5, ax=ax, yticklabels=False)**

**plt.show()**

**# With length of stay, routine culturing, routine chest x-ray, and available facilities & services having the highest correlation with infection risk, we shall further analyze these variables with scatter plots.**

**# In[14]:**

**# Creation of feature matrix**

**x = data.drop('infection risk', 1)**

**# Creation of response var**

**y = data['infection risk']**

**# In[15]:**

**# Scatterplots of highest correlating variables**

**cols = ['length of stay',**

**'routine culturing',**

**'routine chest x-ray',**

**'available facilities & services']**

**for i in cols:**

**plt.scatter(data[i], y)**

**plt.xlabel(i, fontsize=14)**

**plt.ylabel('Infection Risk', fontsize=14)**

**plt.show()**

**# These graphs gives us an idea of which features could end up in our model for predicting infection risk. We shall now use wrapper methods (forward selction, backwards selection, & stepwise selection) to pick the best features.**

**# # 1. Forward Selection**

**#**

**# \*\*Steps for Forward Selection\*\***

**#**

**# 1. Pick an alpha (ours will be alpha = 0.05).**

**#**

**# 2. Fit each feature one at a time to and keep the one with the lowest p-value.**

**#**

**# 3. Fit all possible models with one extra feature added to the previously selected feature(s).**

**#**

**# 4. Again, select the feature with miniumum p-value. If p\_value < alpha, continue, otherwise stop**

**#**

**# The code to achieve this follows:**

**# In[16]:**

**def forward\_selection(data, response, alpha=0.05):**

**ini\_feats = data.columns.tolist()**

**best\_feats = []**

**while len(ini\_feats) > 0:**

**remaining\_feats = list(set(ini\_feats) - set(best\_feats))**

**new\_pval = pd.Series(index=remaining\_feats, dtype=float)**

**for new\_col in remaining\_feats:**

**model = sm.OLS(response, sm.add\_constant(data[best\_feats+[new\_col]])).fit()**

**new\_pval[new\_col] = model.pvalues[new\_col]**

**min\_p\_val = new\_pval.min()**

**if min\_p\_val < alpha:**

**best\_feats.append(new\_pval.idxmin())**

**else:**

**break**

**return best\_feats, model.summary()**

**# In[17]:**

**forward\_selection(x,y)**

**# # Backward Elimination**

**#**

**# \*\*Steps for Backward Elimination\*\***

**#**

**# 1. Pick our alpha (our alpha is alpha = 0.05).**

**#**

**# 2. Fit full model with all features.**

**#**

**# 3. Consider feature with highest p-value. If < alpha, go to next step, otherwise stop.**

**#**

**# 4. Remove the feature under consideration.**

**#**

**# 5. Fit a model without this feature. Repeat process from step 3.**

**#**

**# The code follows:**

**# In[18]:**

**def backward\_elimination(data, response, alpha=0.05):**

**feats = data.columns.tolist()**

**while len(feats) > 0:**

**feats\_with\_constant = sm.add\_constant(data[feats])**

**p\_vals = sm.OLS(response, feats\_with\_constant).fit().pvalues[1:]**

**model = sm.OLS(response, feats\_with\_constant).fit()**

**max\_p\_val = p\_vals.max()**

**if max\_p\_val >= alpha:**

**excluded\_feat = p\_vals.idxmax()**

**feats.remove(excluded\_feat)**

**else:**

**break**

**return feats, model.summary()**

**# In[19]:**

**backward\_elimination(x,y)**

**# # 3. Stepwise Selection**

**#**

**# \*\*Steps for Stepwise Selection\*\***

**# A combination of forward selection and backwards elimination, this is how we'll do stepwise:**

**#**

**# 1. Pick our alpha (alpha = 0.05).**

**#**

**# 2. Perform next step of forward selection (newly added feat must have p-value < alpha).**

**#**

**# 3. Perform all steps of backward elimination (any previous feat must have p-value > alpha).**

**#**

**# 4. Repeat step 2 & 3 until final best set of feats.**

**#**

**# The code is as follows:**

**# In[20]:**

**def stepwise\_selection(data, response, alpha\_in=0.05, alpha\_out=0.05):**

**ini\_feats = data.columns.tolist()**

**best\_feats = []**

**while len(ini\_feats) > 0:**

**remaining\_feats = list(set(ini\_feats) - set(best\_feats))**

**new\_pval = pd.Series(index=remaining\_feats)**

**for new\_col in remaining\_feats:**

**model = sm.OLS(response, sm.add\_const(data[best\_feats+[new\_col]])).fit()**

**new\_pval[new\_col] = model.pvalues[new\_col]**

**min\_p\_val = new\_pval.min()**

**if min\_p\_val < alpha\_in:**

**best\_feats.append(new\_pval.idxmin())**

**while len(best\_feats) > 0:**

**best\_feats\_with\_constant = sm.add\_constant(data[best\_feats])**

**p\_vals = sm.OLS(response, best\_feats\_with\_constant).fit().pvalues[1:]**

**max\_p\_val = p\_vals.max()**

**if max\_p\_val >= alpha\_out:**

**excluded\_feat = p\_vals.idxmax()**

**best\_feats.remove(excluded\_feat)**

**else:**

**break**

**else:**

**break**

**return best\_feats, model.summary()**

**# In[21]:**

**backward\_elimination(x,y)**

**# \*\*Conclusion\*\***

**#**

**# From our Feature Selection procedures, we can see that the best predictors of infection risk are the variables "length of stay", "routine culturing", and "available facilities & services". In that case, we shall create the variables containing our best features and our model:**

**# In[22]:**

**# Save the model that the regression methods created into variables**

**# Best variables recommended by wrapper methods**

**best\_feats = data[['length of stay',**

**'routine culturing',**

**'available facilities & services']]**

**# Actual model**

**best\_model = sm.OLS(y, best\_feats).fit()**

**# # Residual Analysis**

**#**

**# \*\*Residuals vs. Prediciting Variable plots\*\***

**#**

**# To confirm the independence assumption, we shall plot the residuals versus each of the best variables our model selection procedures (wrapper methods) picked out for us. \*\*If the residuals are distributed uniformly randomly around the zero x-axes and do not form specific clusters, then the assumption holds true.\*\***

**# In[23]:**

**for c in best\_feats:**

**plt.figure(figsize = (8,5))**

**plt.title('{} vs. \nModel Residuals'.format(c),**

**fontsize=16)**

**plt.scatter(x = data[c], y = best\_model.resid,**

**color = 'green',**

**edgecolor = 'k')**

**plt.grid(True)**

**xmin = min(data[c])**

**xmax = max(data[c])**

**plt.hlines(y=0,**

**xmin = xmin \* 0.9,**

**xmax = xmax \* 1.1,color = 'yellow',**

**linestyle = '--',**

**lw = 3)**

**plt.xlabel(c,fontsize = 14)**

**plt.ylabel('Residuals',**

**fontsize = 14)**

**plt.show()**

**# Residual plots of best variables show some clustering but overall \*\*assumptions of linearity and independence hold up\*\* since the distribution is random around the 0 axis.**

**# \*\*Fitted vs. Residuals\*\***

**#**

**# By plotting fitted response values vs. residuals, we are checking for \*\*constant variance of the residuals as the response variable increases.\*\* If this isn't the case, it implies that a variable transformation may be needed to improve our model quality.**

**# In[24]:**

**plt.figure(figsize = (8,5))**

**p=plt.scatter(x = best\_model.fittedvalues,**

**y = best\_model.resid,**

**edgecolor='k')**

**xmin = min(best\_model.fittedvalues)**

**xmax = max(best\_model.fittedvalues)**

**plt.hlines(y = 0,**

**xmin = xmin \* 0.9,**

**xmax = xmax\*1.1,**

**color = 'red',**

**linestyle = '--',**

**lw = 3)**

**plt.xlabel('Fitted values',**

**fontsize = 15)**

**plt.ylabel('Residuals',**

**fontsize = 15)**

**plt.title('Fitted vs. Residuals Plot',**

**fontsize = 18)**

**plt.grid(True)**

**plt.show()**

**# Based on this plot, \*\*homoscedasticity assumption is met.\*\***

**# \*\*Histogram and Q-Q plot of normalized residuals\*\***

**#**

**# To check the normality assumption, we'll generate a histogram and q-q plot of the normalized residuals.**

**# In[25]:**

**plt.figure(figsize = (8,5))**

**plt.hist(best\_model.resid\_pearson,**

**bins = 20,**

**edgecolor='k')**

**plt.ylabel('Count',**

**fontsize = 15)**

**plt.xlabel('Normalized Residuals',**

**fontsize = 15)**

**plt.title('Histogram of Normalized Residuals',**

**fontsize = 18)**

**plt.show()**

**# In[26]:**

**from statsmodels.graphics.gofplots import qqplot**

**# In[27]:**

**plt.figure(figsize = (8,5))**

**fig = qqplot(best\_model.resid\_pearson,**

**line = '45',**

**fit = 'True')**

**plt.xticks(fontsize=13)**

**plt.yticks(fontsize=13)**

**plt.xlabel('Theoretical Quantiles',**

**fontsize = 15)**

**plt.ylabel('Sample Quantiles',**

**fontsize = 15)**

**plt.title('Q-Q plot of Normalized Residuals',**

**fontsize = 18)**

**plt.grid(True)**

**plt.show()**

**# Based on the histogram and q-q plot, \*\*the normality assumption is satisfied.\*\***

**# \*\*Normality: Shapiro-Wilk Test of Residuals\*\***

**# In[28]:**

**\_, p = scipy.stats.shapiro(best\_model.resid)**

**if p < 0.05:**

**print('The Residuals pass this test.')**

**else:**

**print ('Normality NOT confirmed.')**

**# According to the Shapiro-Wilk test, \*\*normality isn't confirmed since the p-values of the residuals aren't all less than alpha = 0.05.\*\* While our residuals fail this normality test, they stilled passed the histogram and q-q plot tests, so we can still assume normality.**

**# \*\*Cook's Distance (To check for outliers in residuals)\*\***

**#**

**# Cook's distance measures how much effect deleting an observation has on the model. A large Cook's distance for a point can be a potential outlier.**

**# In[29]:**

**from statsmodels.stats.outliers\_influence import OLSInfluence as inf**

**# In[30]:**

**inf = inf(best\_model)**

**(c, p) = inf.cooks\_distance**

**plt.figure(figsize = (8,5))**

**plt.title("Cook's Distance Plot of Residuals",**

**fontsize = 16)**

**plt.stem(np.arange(len(c)),**

**c,**

**markerfmt = ',',**

**use\_line\_collection = True)**

**plt.grid(True)**

**plt.show()**

**# Based on the Cook's Distance plot, \*\*there are few data points with residuals possibly being outliers.\*\***

**# \*\*Variance Inflation Factor (VIF)\*\***

**#**

**# The VIF of eacb predictor allows us to check which factors to a degree cause multicollinearity in our model by dividing the ratio of variance in our multi-linear model by the variance of a simple-linear model.**

**# In[31]:**

**from statsmodels.stats.outliers\_influence import variance\_inflation\_factor as vif**

**# In[32]:**

**for i in range(len(best\_feats.columns)):**

**v = vif(np.matrix(best\_feats), i)**

**print('Variance Inflation Factor for {}: {}'.format(best\_feats.columns[i],**

**round(v,2)))**

**# It seems that two factors in our model, \*\*length of stay\*\* and \*\*available facilities & services\*\*, have VIFs > 10. This means \*\*there is multicollinearity in our model.\*\***

**# # Prediction & their Intervals**

**#**

**# To test our model on patients with IDs 1-5, we shall create a new dataframe with just those rows, and get predictions from our model**

**# In[33]:**

**# Create test data from patient ids 1-5 with best features**

**test\_data = pd.DataFrame(raw\_data[:5], columns=['length of stay',**

**'routine culturing',**

**'available facilities & services'])**

**test\_data = test\_data.astype(float)**

**test\_data**

**# In[34]:**

**from statsmodels.sandbox.regression.predstd import wls\_prediction\_std**

**# In[35]:**

**best\_feats.shape**

**# In[36]:**

**# Confidence and Prediction Intervals**

**predictions = best\_model.get\_prediction(test\_data)**

**predictions.summary\_frame(alpha = 0.05)**

**# In[38]:**

**# Predicted Observations**

**predict = best\_model.predict(test\_data)**

**predict**