# NVM2 Task 2: Predictive Analysis

**Sina Bozorgmehr**

**Western Governors University**

# NVM2 Task 2: Predictive Analysis

## Introduction

One of the industries that demands a well-prepared perspective based on scientific data collected daily from the industry, especially from patients, is health care. In this industry, predicting is a key aspect of data analysis. The average length of stay in hospitals (ALOS) is frequently used as a performance metric. If all other factors remain the same, a shorter stay will lower the cost per discharge and move care from inpatient to less expensive post-acute care ("Length of Hospital Stay," 2017). Predicting the length of stay (LOS) of patients based on their vitals and diagnosis and also other metrics provides an understanding of what metrics have a bigger impact on the LOS.

## Research Question

A model that can estimate the LOS of patients benefits both the hospitals and the patients and their families. By predicting the LOS and taking required measures to reduce it, patients could benefit by paying less for hospital services. Also, the less a subject stay in the hospital, the less the risk of infection and medication side effect.

In this analysis, a random forest regression model is created to predict the LOS of patients based on their vitals and other metrics as they become available during their stay in the hospital.

## Method Justification

Random forest (RF) is a supervise algorithm created on ensemble learning that could be implemented both for regression and classification. Random forests are based on the principle of creating several decision trees and aggregating them to produce a more accurate result. A decision tree is a deterministic algorithm, which implies that it will construct the same tree each

time it is given the same input. Because we create each tree on a different random subset of our data, each decision tree in a random forest is unique. Therefore, random forest is more accurate than a single decision tree by avoiding overfitting (Sarkar & Natarajan, 2019).

This RF model will predict the length of stay of a patient by considering the vitals and metrics obtained from the patient. For the analysis, Python is utilized, with various libraries added to make coding and calculations easier. Pandas is used to manage data in data frames, and NumPy is used to do algebraic calculations. The *RandomForestRegressor* module from the Sci-Kit Learn package is used for prediction. For data visualization, Matplotlib and Seaborn are utilized.

Random forests are non-parametric and can handle skewed and multi-modal data as well as ordinal and non-ordinal categorical data because they make no formal distributional assumptions (Richmond, 2016).

**Data Preparation**

The data was first imported into a Pandas data frame using pandas.read_csv (Figure 1), and the null values and total number of entries were validated. Figure 2 shows the distribution of the LOS amongst the patients and as we see most of them were released either under 2o days or above 50 days. The next step was to remove those factors from the data set that were not relevant to our investigation, primarily demographics (Figure 3). Two lists were created, one with continuous data and the other with categorical variables (Figure 4). Then, all the independent variables are plotted to demonstrate their distributions (Figure 5). As seen, the answers to the 8 questions in the survey have a very similar pattern and therefore we get the average of all answers and store it in the new column, Survey, and then delete the original columns (Figure 6). Also, we remove the ReAdmis variable since theoretically it does not have any effect on the

response. After that, the categorical variables are converted into dummy variables using *Pandas.get_dummy* module (Figure 7). The final step before fitting the model is to separate the features set from the response variable (Figure 8).

**Figure 1**

*Importing dataset into a pandas data frame.*

```
#creating the raw pandas dataframe
df = pd.read_csv("C:/Users/bozor/Documents/WGU MSDA/Data Mining I/medical_clean.csv")
df.head()
```

**Figure 2**

*LOS distribution amongst patients.*

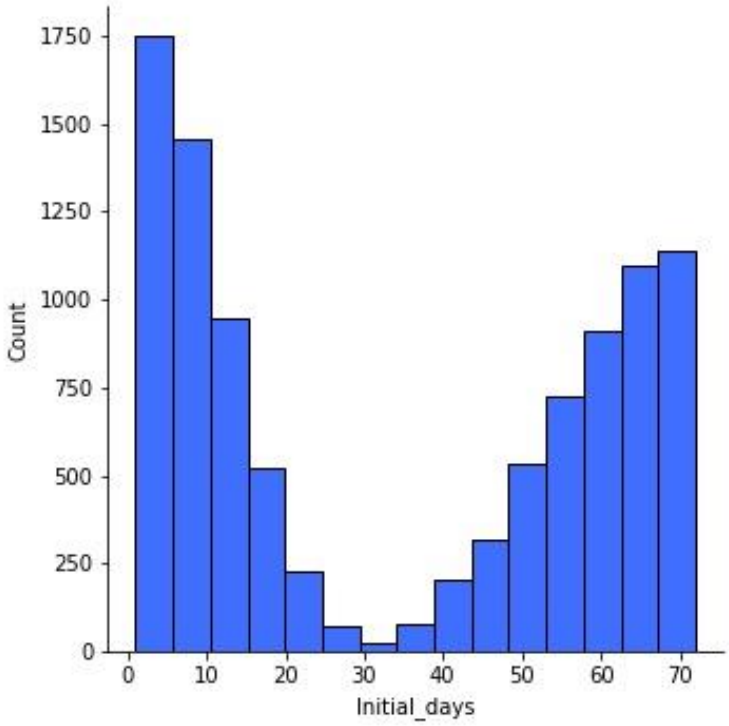

**Figure 3**

*Variables with no importance in the analysis are removed from the data frame.*

```
#eliminating variables that have no importance in our analysis

column = ['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State', 'County',
          'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone','Job'
         ]

new_df = df.drop(columns=column)
new_df.head()
```

**Figure 4**

*Two lists are created for categorical and numerical variables.*

```
#Creating two lists, containing continuous and categorical variables

cont_var = ['Children', 'Age', 'Income','VitD_levels', 'Doc_visits', 'Full_meals_eaten',
            'vitD_supp', 'Initial_days', 'TotalCharge', 'Additional_charges', 'Item1',
            'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8']
cat_var = [i for i in new_df.columns if i not in cont_var]

print('Continuous variables are:\n\n {} \n\n and Categorical variables are:\n\n {}'.format(cont_var, cat_var))
```

```
Continuous variables are:

 ['Children', 'Age', 'Income', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Initial_days', 'TotalCharge',
'Additional_charges', 'Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8']

 and Categorical variables are:

 ['Marital', 'Gender', 'ReAdmis', 'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke', 'Complication_risk', 'Overweight',
'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Serv
ices']
```

**Figure 5**

*Univariate visualization of continuous variables.*

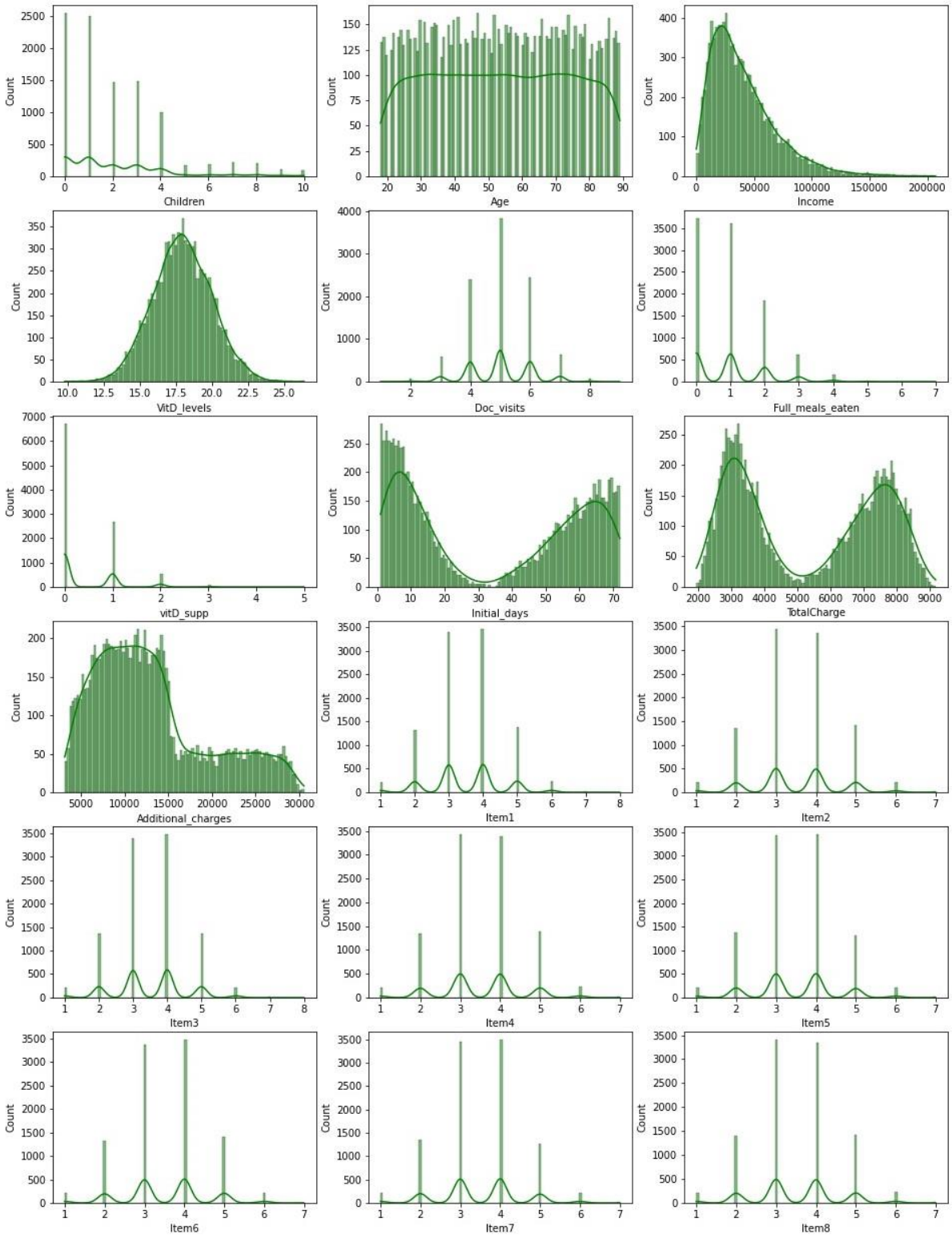

**Figure 6**

*Averaging the survey answers and storing it in a new variable.*

```
# Creating a new feature by averaging all the 8 questions in survey:

surv = ['Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8']
new_df['Survey'] = new_df[surv].astype(float).mean(axis=1).round()
new_df.drop(columns=surv, inplace=True)

#Graphing the new feature

sns.histplot(data=new_df, x='Survey', kde=True, color='blue', bins='sqrt')
plt.xticks(range(1,8,1))
plt.savefig('fig.jpg')
plt.show()
```

**Figure 7**

*Creating dummy variables.*

```
# creating dummy variables for all categorical features

cleaned_df = pd.get_dummies(new_df, drop_first=True)
cleaned_df.head()
```

**Figure 8**

*Storing feature sets and response variable into separate arrays.*

```
#Seaparating independent and dependent variables

X = cleaned_df.drop(columns='Initial_days')
y = cleaned_df[['Initial_days']]
y = y.values.reshape(-1,)
X.shape
```

**Analysis**

We need some unseen data to be used for model evaluation. To do so, 30% of the datapoints are held out as the test set (Figure 9). In the next step, we use Random Forests as our

estimator and then search for the best maximum depth of the trees amongst three options: 4, 6 or 8 (Figure 10).

**Figure 9**

*30% of datapoints are held out for model evaluation.*

```
# Holding out 30% of the data for final evaluation

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=777)
```

**Figure 10**

*Hyper-parameter tuning with grid search and cross validation.*

```
# Tuning the hyper parameters using GridSearchCV:

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error as MSE



rfr = RandomForestRegressor(n_estimators = 200, random_state=777)
h_param = {'max_depth' : [4, 6, 8]
          }

cv = GridSearchCV(rfr, param_grid=h_param, cv=10, n_jobs=-1)

cv.fit(X_train, y_train)
```

The number of estimators is set to 200 and 10 folds are created for the cross validation. Grid search outputs 8 as the best maximum depth for the trees and the R2 score for this model is 0.997 (Figure 11). To evaluate our model, we use the test set to calculate the scores of the model on unseen data. As Figure 12 demonstrates, the model is working almost the same on the unseen data and the mean of the error for the predicted values is 1.34 days which is excellent.

**Figure 11**

*Grid search suggests 8 as the optimum maximum depth of the trees.*

```
# The best parameters and score:

print('THe best max_depth is {} and the R2 obtained from the best model is {}'.format(
                                                                    cv.best_params_['max_depth'],
                                                                    round(cv.best_score_, 3)))

```

```
THe best max_depth is 8 and the R2 obtained from the best model is 0.997
```

**Figure 12**

*Result of model evaluation on unseen data.*

```
# The scores obtained from predicting test set:

y_pred = cv.best_estimator_.predict(X_test)

print("The scores for the test set:\n\n R2: {:.3f} \n MSE: {:.3f} \n RMSE:{:.3f}".format(
                                        cv.best_estimator_.score(X_test, y_test),
                                        MSE(y_test, y_pred),
                                        MSE(y_test, y_pred)**(1/2)))

```

```
The scores for the test set:

 R2: 0.997
 MSE: 1.798
 RMSE:1.341
```

**Data Summary and Implications**

As seen in figure 12, the model has a R2 score of 0.997 and MSE of 1.798. These are really great numbers for a RF model to predict the LOS of patients based on their metrics. The RMSE tells that this RF model can predict the days a patient stays in the hospital with just 1.34 days error. Using this accurate model, hospital management can plan and implement guidelines to reduce the LOS. Patients and their families do also benefit from this by having a more accurate idea of the length of the treatment and how to be prepared for that.

Though the RF model metrics demonstrate a very strong prediction, due to power limitation, we had to restrict our hyper-parameter tuning and only stick to three different maximum depth. With a stronger processor, more parameters could be tuned and optimized.

**Sources**

The dataset used in this analysis was acquired form WGU portal:

https://access.wgu.edu/ASP3/aap/content/d9rkejv84kd9rk30fi2l.zip

**References**

Length of hospital stay. (2017). *Health Care Use*. Published. https://doi.org/10.1787/8dda6b7a-en

Richmond, S. (2016, March 21). *Algorithms Exposed: Random Forest*. BCCVL. https://bccvl.org.au/algorithms-exposed-random-forest/

Sarkar, D., & Natarajan, V. (2019). *Ensemble Machine Learning Cookbook: Over 35 practical recipes to explore ensemble machine learning techniques using Python*. Packt Publishing.