­

**Performance Assessment for D209: Data Mining I  
Task 2: Prediction**

Christopher Fischer - 011933891

College of Information Technology, Western Governors University

July 15, 2024

Performance Assessment for D209: Data Mining I – Task 2

This document contains the tasks and outputs required for the “NVM4 TASK 2: PREDICTIVE ANALYSIS” assessment. All work is original to the author unless otherwise indicated by a citation.

# A - Research Question

Hospital administration has detected a concerning trend where a growing number of patients are having longer hospital stays. Longer stays translated to increased costs for the hospital, the patient, and third-party payers. Extended stays also increase the patient's chances of contracting a hospital-acquired infection. (Medew, 2011) Therefore, management is looking for ways to slow, stop, or reverse this trend.

With this motivation, the data analytics team has been given the budget and scope to perform a study to determine if it is possible to predict the likely length of a patient’s hospital stay based on available clinical and demographic data. With an effective prediction model, management can identify patients likely to have more extended stays and target specialized education and care.

This effort aims to build a model to predict the length of a patient’s stay. Management hopes that such a model can be deployed as part of our patient admissions process so that patients identified as being at risk of extended hospitalization can be given special attention designed to shorten their stay.

# B – Method Justification

Predicting the value of a target variable based on a set of explanatory variables is one of the most common problems solved with supervised machine learning. This is precisely the type of question this study aims to answer: How many days is this patient likely to be in the hospital? The team has selected Random Forest Regressor from several widely used prediction algorithms for this study. A Random Forest Regressor combines a decision tree's powerful “if-then-else” rules with an iterative approach to finding an optimal set of predictor variables for building those rules. (Bruce, Bruce, & Gedeck, 2020) In this study, the Random Forest Regressor model is expected to return the predicted number of days of each patient’s hospital stay.

The decision tree model, which underlies the Random Forest Regressor, assumes an absence of outliers. This is because extreme values can have a marked impact on the resulting model. Therefore, our data preparation plan must include appropriate steps to ensure that selected predictor variables have had outliers remediated.

The data science team will develop the code for this study using Python. In its nearly 35 years, Python has become one of the most widely used general-purpose programming languages. Python is backed by a massive community of developers and a vast collection of libraries that extend the core capabilities of the language. (Datacamp, 2022) Another reason for selecting Python is that, while R is an excellent choice for interactive studies, Python is better suited for deployment on production servers as part of a data pipeline. (WGU Information Technology, n.d.)

The following table lists the Python libraries which will support this study.

|  |  |
| --- | --- |
| Module | Purpose |
| Numpy | Provides many tools for numerical operations |
| Pandas | Provides DataFrame used for data management |
| Missingno | Displays a graphic of missing data used in the cleaning process |
| Matplotlib.pyplot | Tools for visualizing data and results |
| Seaborn | Additional visualization capabilities |
| Sklearn.preprocessing | Implements scaling and one-hot encoding |
| Sklearn.compose | Framework for applying multiple data transformation steps |
| Sklearn.model\_selection | Supports splitting datasets into training and test sets, as well as cross-validation methods |
| Sklearn.ensemble | Implementation of the Random Forest Regressor algorithm |
| Sklearn.metrics | Provides several metrics and tools for evaluating model performance |

Table - Python Libraries to be Used

# C – Data Preparation

The following sections outline the team’s plan to clean and prepare the input data for this study.

## C1 – Pre-processing Goal

As mentioned above, Random Forest Regressor is highly sensitive to outliers. Therefore, all numerical values will be evaluated and treated for outliers to prevent unwanted skewing of the model.

## C2 – Selected Variables

| Variable Name | Data Type |
| --- | --- |
| Predictors | |
| Age | Numeric |
| Income | Numeric |
| Children | Numeric |
| Population | Numeric |
| VitD\_levels | Numeric |
| State | Categorical |
| Area | Categorical |
| Complication\_risk | Categorical |
| Marital | Categorical |
| Gender | Categorical |
| Initial\_admin | Categorical |
| Soft\_drink | Categorical |
| HighBlood | Categorical |
| Stroke | Categorical |
| Overweight | Categorical |
| Hyperlipidemia | Categorical |
| BackPain | Categorical |
| Anxiety | Categorical |
| Allergic\_rhinitis | Categorical |
| Reflux\_esophagitis | Categorical |
| Asthma | Categorical |
| Diabetes | Categorical |
| Target | |
| Initial\_days | Numerical |

Table - Study Variables

## C3 – Data Preparation Steps and Results

# Check missing data

msno.matrix(df[X\_full], fontsize = 12, labels=True)

plt.title('Missing Data Matrix')

plt.show()

A black and white chart

Description automatically generated with medium confidence

Figure - Missing Data Plot

No variables exhibit missing data.

#Check values in yes/no variables

print(df[yes\_no\_variables][~df[yes\_no\_variables].isin(['Yes','No'])].count())

A screenshot of a computer screen

Description automatically generated

Boolean variables all contain valid values.

#Detect potential outliers

df\_z = (df[numerical\_variables] - df[numerical\_variables].mean())/df[numerical\_variables].std(ddof=0)

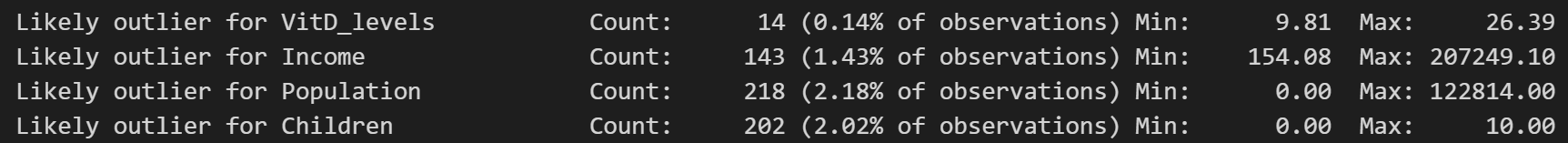
outlier\_cols = df\_z.loc[: , (df\_z > 3.0).any()].columns

for col in outlier\_cols :

    cnt = len(df\_z[df\_z[col]>3])

    min, max = df[col].min(), df[col].max()

    print('Likely outlier for {0:<20}\t Count: {1:7d} ({2:5.2%} of observations)\tMin: {3:>9.2f}\tMax: {4:>9.2f}'.format(col,cnt,cnt/10000,min,max))



# Deeper look at Population

sns.boxplot(data=df, x='Population')

plt.show()

print(df['Population'].value\_counts())

Population

0 109

195 14

115 11

178 11

285 11

...

A graph with a bar graph

Description automatically generated with medium confidence

Figure - Boxplot of Population

While a small percentage of values exceed the norms for the variables *Income* and *Children*, none is definitively invalid. Regarding *VitD\_levels*, the range of values for a human could easily exceed 60ng/mL. (National Institutes of Health, Office of Dietary Supplements, 2023) Therefore, all values for these three variables will be retained in the study. There are several observations with an invalid value (zero) for *Population*. These will be removed from consideration.

# Drop rows with Population <= 0

df = df[df.Population > 0]

print(df['Population'].count())

9891

# Change "Never Married" to "NeverMarried" to prevent one-hot encoding problems later

df['Marital'] = df['Marital'].replace('\s+', '',regex=True)

print(df['Marital'].value\_counts())  
Marital

Widowed 2016

Married 1998

Separated 1968

NeverMarried 1961

Divorced 1948

Name: count, dtype: int64

# Reexpress yes/no columns as numbers

yesno\_dict = {'No': 0, 'Yes': 1}

for col in yes\_no\_variables:

    df[col] = df[col].map(yesno\_dict)

print(df[yes\_no\_variables].info())  
Index: 9891 entries, 0 to 9999

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Soft\_drink 9891 non-null int64

1 HighBlood 9891 non-null int64

2 Stroke 9891 non-null int64

3 Overweight 9891 non-null int64

4 Hyperlipidemia 9891 non-null int64

5 BackPain 9891 non-null int64

6 Anxiety 9891 non-null int64

7 Allergic\_rhinitis 9891 non-null int64

8 Reflux\_esophagitis 9891 non-null int64

9 Asthma 9891 non-null int64

10 Diabetes 9891 non-null int64

# Rexpress State with frequency encoding

df['State'] = df['State'].map(df['State'].value\_counts().to\_dict())

print(df['State'].info())

Series name: State

Non-Null Count Dtype

-------------- -----

9891 non-null int64

dtypes: int64(1)

# Rexpress Area with simple ordinal encoding

area\_map = {'Rural': 1, 'Suburban': 2, 'Urban' : 3}

df['Area'] = df['Area'].map(area\_map)

print(df['Area'].info())

Series name: Area

Non-Null Count Dtype

-------------- -----

9891 non-null int64

dtypes: int64(1)

# Scaling and One-hot encoding

preprocessor = make\_column\_transformer(

 (StandardScaler(), numerical\_variables),

 (OneHotEncoder(), onehot\_variables),

  remainder='passthrough',

  verbose\_feature\_names\_out=False

)

X = pd.DataFrame(preprocessor.fit\_transform(df[X\_vars]), columns=preprocessor.get\_feature\_names\_out())

y = df[y\_vars]

## C4 – Prepared Data File

This file contains the cleaned and prepared data used in subsequent modeling activities.



# D – Analysis

The following sections outline the steps taken to build the model desired for this study.

## D1 – Train/Test Split

One of the dangers analysts face in building machine learning models is that the algorithms can quickly go too far in fitting the training data. In other words, rather than learning a general model for accomplishing the desired prediction, the model, in essence, memorizes the training data. The analyst should partition the available data into training and testing sets to avoid this. Doing so provides significant unseen data to test the model's performance built on the training data. (Massaron & Boschetti, 2016) Accordingly, the team will split the provided dataset into 80% training and 20% testing.

# Split data into train and test sets with an 80/20 split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5470)

X\_train.to\_csv('medical\_X\_train.csv', index=False)

X\_test.to\_csv('medical\_X\_test.csv', index=False)

y\_train.to\_csv('medical\_y\_train.csv', index=False)

y\_test.to\_csv('medical\_y\_test.csv', index=False)

These attachments contain the partitioned predictor and target data.



## D2/D3 – Modeling Technique and Code

As mentioned in [B – Method Justification](#_B_–_Method), the team has selected Random Forest Regressor as the machine learning algorithm for this study. While this is an excellent algorithm, finding the best parameters to configure the algorithm requires further consideration. Failing to do so will limit the algorithm’s ability to produce the model with the best performance. (Bruce, Bruce, & Gedeck, 2020) Additionally, given the relatively small dataset (*n*=10,000), a cross-validation method must be employed to ensure that our model is as accurate on unseen data (test) as on seen data (training). (Alpaydin, 2014)

The *RandomizedSearchCV* method will provide both hyperparameter tuning and cross-validation for our study. The *RandomizedSearchCV* will adjust these hyperparameters:

* n\_estimators – number of trees to create
* max\_features – number of features to consider in choosing the best split
* max\_depth – maximum number of levels in each tree
* min\_samples\_split – minimum number of samples required to split an internal node
* min\_samples\_leaf – minimum number of samples required to be at a leaf node
* bootstrap – whether samples or the entire data set are used for tree-building

(RandomForestRegressor, 2024)

For cross-validation, we will select a typical 5-fold arrangement. With 5-fold cross-validation, the dataset will be split five different ways, with each iteration containing 20% of the data for testing and 80% for training. In this way, every observation in the dataset will be used for training four times and testing one time. Last, the model which produced the optimal scoring metric will be labeled “best”. For this study, the *neg\_mean\_squared\_error* will be the metric used.

A pink and black text with yellow squares

Description automatically generated with medium confidence

Figure - Five-fold Cross-validation (source: DataCamp)

Here is the code and results for creating and fitting the Random Forest Regressor model.

# Set-up hyperparameter values

n\_estimators = [int(x) for x in np.linspace(start = 20, stop = 100, num = 20)]

max\_features = ['log2', 'sqrt']

max\_depth = [int(x) for x in np.linspace(5, 25, num = 5)]

max\_depth.append(None)

min\_samples\_split = [2, 5, 10]

min\_samples\_leaf = [1, 2, 4]

bootstrap = [True, False]

parameters = {'n\_estimators': n\_estimators,

               'max\_features': max\_features,

               'max\_depth': max\_depth,

               'min\_samples\_split': min\_samples\_split,

               'min\_samples\_leaf': min\_samples\_leaf,

               'bootstrap': bootstrap}

# Set-up base model

rf = RandomForestRegressor()

# Set-up cross validation

rfr = RandomizedSearchCV(estimator=rf, param\_distributions=parameters, n\_iter=20, cv=5, scoring='neg\_mean\_squared\_error', verbose=1, return\_train\_score=False, n\_jobs=-1)

# Fit the model

rfr.fit(X\_train,y\_train.values.ravel())

rfr\_best = rfr.best\_estimator\_

# Print results

print('Best score : ' , rfr.best\_score\_)

print('Best params : ' , rfr.best\_params\_)

print(rfr\_best.get\_params())  
  
Fitting 5 folds for each of 20 candidates, totalling 100 fits

Best score : -689.775292177333

Best params : {'n\_estimators': 32, 'min\_samples\_split': 10, 'min\_samples\_leaf': 2, 'max\_features': 'sqrt', 'max\_depth': 5, 'bootstrap': True}

{'bootstrap': True, 'ccp\_alpha': 0.0, 'criterion': 'squared\_error', 'max\_depth': 5, 'max\_features': 'sqrt', 'max\_leaf\_nodes': None, 'max\_samples': None, 'min\_impurity\_decrease': 0.0, 'min\_samples\_leaf': 2, 'min\_samples\_split': 10, 'min\_weight\_fraction\_leaf': 0.0, 'monotonic\_cst': None, 'n\_estimators': 32, 'n\_jobs': None, 'oob\_score': False, 'random\_state': None, 'verbose': 0, 'warm\_start': False}

# E – Data Summary and Implications

Unfortunately, the results of this study are disappointing. Here are some performance metrics achieved by the “best” model.

A screenshot of a computer

Description automatically generated

Figure - Model Performance Metrics

This shows that the model predicted the length of hospital stays with an average error of over 25 days! The R2 scores are essentially zero, indicating that the model performs exceptionally poorly predicting the target variable.

A clear pattern emerges when looking at a plot of predictions against actual target values for the test data set. The model predictions are all in a tight range between 30 and 40 days, which does not reflect the distribution of the actual values in any fashion.

A diagram of a test

Description automatically generated

Figure - Test Predictions vs. Actuals

One limitation of this study is that no feature elimination was undertaken. If additional exploratory data analysis were performed, the set of features used to train the model could have been reduced to exclude those features with limited correlation to the target variable.

As the results of this model are no better than random guesses, the model will not provide the guidance hoped for. Should the business stakeholders wish to continue this effort, steps should be taken to gather additional demographic and clinical data about conditions and comorbidities that may affect the length of hospital stays. Then, this study could be revised to incorporate the additional features and implement one or more feature elimination techniques. Combined, these steps should provide a better outcome than this study achieved.

# F – Recorded Code Review

A recording of the code review presentation was uploaded with this submission. For quick reference, that video may be found here: [Panopto Recording](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=50839187-f7f4-4752-b343-b1af00f45063)

This Python notebook contains all the code used for this study.



# G – Third-Party Code References

Buser, A. (2023, 04 24). *A Beginner’s Guide to using sklearn make\_column\_transformer*. Retrieved from Medium: https://medium.com/@buser.andre/a-beginners-guide-to-using-sklearn-make-column-transformer-4ac9a7f78fb3

Eiler, J. (2017, 01 16). *Remap values in pandas column with a dict, preserve NaNs*. Retrieved from Stackoverflow: https://stackoverflow.com/questions/20250771/remap-values-in-pandas-column-with-a-dict-preserve-nans

Koehrsen, W. (2018, 01 09). *Hyperparameter Tuning the Random Forest in Python*. Retrieved from Medium: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

MacPhee-Cobb, L. (2016, 03 20). *A column-vector y was passed when a 1d array was expected*. Retrieved from StackOverflow: https://stackoverflow.com/questions/34165731/a-column-vector-y-was-passed-when-a-1d-array-was-expected

Neural Ninja. (2023, 06 12). *Frequency Encoding: Counting Categories for Representation*. Retrieved from Let's Data Science: https://letsdatascience.com/frequency-encoding/

Piepenbreier, N. (2022, 02 23). *One-Hot Encoding in Scikit-Learn with OneHotEncoder*. Retrieved from Datagy: https://datagy.io/sklearn-one-hot-encode/

Ray, S. (2020, 10 21). *Show all colums of a pandas-dataframe in ".describe()"*. Retrieved from Stackoverflow: <https://stackoverflow.com/questions/64455605/show-all-colums-of-a-pandas-dataframe-in-describe>

# H – Referenced Works

Alpaydin, E. (2014). *Introduction to Machine Learning.* MIT Press. Retrieved 07 12, 2024, from https://ebookcentral.proquest.com/lib/westerngovernors-ebooks/detail.action?docID=3339851

Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python.* O'Reilly Media, Incorporated. Retrieved 07 12, 2024, from https://ebookcentral.proquest.com/lib/westerngovernors-ebooks/detail.action?docID=6173908

Datacamp. (2022). *Python vs R for Data Science: Which Should You Learn?* Retrieved 03 22, 2024, from Datacamp Blog: https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference

Massaron, L., & Boschetti, A. (2016). *Regression Analysis with Python : Discover Everything You Need to Know About the Art of Regression Analysis with Python, and Change How You View Data.* Packt Publishing. Retrieved 07 12, 2024, from <https://eds.p.ebscohost.com/eds/ebookviewer/ebook?sid=cd9d41a8-0c80-4baf-bad5-57a9c7381229%40redis&ppid=pp_29&vid=0&format=EB>

Medew, J. (2011, 08 29). Hospital Stay is a Health Risk. Retrieved from The Sydney Morning

Herald: https://www.smh.com.au/national/victoria/hospital-stay-is-a-health-risk-20110828-1jgnl.html

National Institutes of Health, Office of Dietary Supplements. (2023, 09 18). *Vitamin D Fact Sheet for Health Professionals*. Retrieved from NIH Health Information: https://ods.od.nih.gov/factsheets/VitaminD-HealthProfessional/

*RandomForestRegressor*. (2024). Retrieved from Scikit Learn API Reference: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

WGU Information Technology. (n.d.). *R or Python*. Retrieved 03 22, 2024, from https://www.wgu.edu/online-it-degrees/programming-languages/r-or-python.html