# Part I: Research Question

## A1: Proposal of Question

This analysis aims to answer the question: What factors predict whether a patient will be readmitted to the hospital? To answer this question, a Naïve Bayes classifier will be used.

## A2: Defined Goal

The goal of this analysis is to capture hospital readmission trends using a Naïve Bayes classification machine learning model. This will allow the research question regarding which features predict readmissions to be answered.

# Part II: Method Justification

## B1: Explanation of Classification Method

Naïve Bayes works by using the prior probability of an event occurring along with a set of given conditions to determine the likelihood that an event will occur (Navlani, 2018). In the context of this analysis, it will use the prior probability of being readmitted to the hospital along with the given conditions from the data to determine how likely a patient is to be readmitted to the hospital. Whichever outcome (being readmitted or not being readmitted) is most likely will be assigned as the label for each observation. A patient predicted to be more likely to be readmitted, will be labeled ReAdmis 1, and a patient predicted to be less likely to be readmitted will be labeled ReAdmis 0.

## B2: Summary of Method Assumptions

Naïve Bayes classification assumes that each feature used to predict the outcome is independedt from all other features. This means that separate features do not depend on each other.

## B3: Packages List

The following Python libraries and modules will be used to perform this analysis:

* **os**: The os module is used for operating system tasks. In this analysis, it is used to create the directory where all output is stored.
* **pandas**: pandas is a library used to work with data including, but not limited to, importing and exporting data as well as general data manipulation and simple statistical operations. In this analysis, it is used for all data preparation tasks.
* **SciPy**: SciPy is a statistics library used widely across the industry. In this analysis, it is used to aid in statistical outlier identification.
* **Matplotlib**: Matplotlib is a library used to create a wide range of visualizations. In this analysis, it is used to enhance the plot of the ROC Curve.
* **scikit-learn**: scikit-learn is a library used for various aspects of data science. From this library, the specific modules used include:
  + **feature\_selection**: From the feature\_selection module, the **SelectKBest** class is used to score the features for selection.
  + **model\_selection**: From the model\_selection module, the **train\_test\_split** function is used to prepare the training and testing data needed for the analysis.
  + **preprocessing**: From the preprocessing module, the **StandardScaler** class is used to scale the numeric features of the data set.
  + **naïve\_bayes**: From the naïve\_bayes module, the **GaussianNB** class is used to implement the Naïve Bayes Classifier used in the analysis.
  + **metrics**: From the metrics module, one class and two functions are used: RocCurveDisplay, accuracy\_score and roc\_auc\_score. **accuracy\_score** is used to calculate the accuracy of the predictions made on the test data. **roc\_auc\_score** is used to calculate the area under the Receiver Operating Characteristic (ROC) curve, which compares the true positive and false positive rate of predictions made against the test data. **RocCurveDisplay** is used to plot the ROC curve.

# Part III: Data Preparation

## C1: Data Preprocessing

One goal of preprocessing the data is encoding the features. Several of the features are categorical. String data is not well suited to machine learning algorithms. They can handle binary dummy variables indicating whether an observation fits each category. Because of this, each of the categorical features will be one-hot-encoded to transform them into dummy variables.

## C2: Data Set Variables

|  |  |
| --- | --- |
| **Variable** | **Class** |
| ReAdmis | categorical |
| Initial\_days | continuous |
| Services\_CT\_Scan | categorical |
| Children | categorical |
| Marital\_Divorced | categorical |
| Services\_Intravenous | categorical |
| Population | continuous |
| Initial\_admin\_Emergency\_Admission | categorical |

## C3: Steps for Analysis

The following code was extracted from the file “task1.py” which is being included with the submission of this report.

Load the packages

**import** os  
**import** pandas **as** pd  
**import** scipy.stats **as** stats  
**import** matplotlib.pyplot **as** plt  
**from** sklearn.feature\_selection **import** SelectKBest  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.preprocessing **import** StandardScaler  
**from** sklearn.naive\_bayes **import** GaussianNB  
**from** sklearn.metrics **import** accuracy\_score, roc\_auc\_score, RocCurveDisplay

Create the output directory

os.makedirs(**'./output1'**, exist\_ok=**True**)

Load the data

medical\_clean = pd.read\_csv(**'./medical\_clean.csv'**)

View the data

print(**f'\n{**medical\_clean.head().to\_string()**}\n'**)

Evaluate the data structures

medical\_clean.info()

print(**f'\ncontinuous data:\n{**medical\_clean.describe().T.to\_string()**}'**)  
cat\_data = medical\_clean[  
 [col **for** col **in** medical\_clean.columns  
 **if** medical\_clean[col].dtype **not in** [**'float64'**, **'int64'**]]  
]  
print(**f'\ncategorical data:\n{**cat\_data.describe().T.to\_string()**}\n'**)

Evaluate the data types and check for nulls

Check for duplicates

print(**f'\nduplicates:\n{**medical\_clean.duplicated().sum()**}'**)

Check for outliers

num\_cols = [col **for** col **in** medical\_clean.columns  
 **if** medical\_clean[col].dtype **in** [**'float64'**, **'int64'**]]  
df\_num = medical\_clean[num\_cols]  
df\_zscores = df\_num.apply(stats.zscore)  
df\_outliers = df\_zscores.apply(**lambda** x: (x > 3) | (x < -3))  
print(**f'\noutliers:\n{**df\_outliers.sum().to\_string()**}'**)

Look for correlation between the features

corr = df\_num.corr()  
print(**f'\ncorrelation:\n{**corr.to\_string()**}'**)

Convert categorical features to dummy variables

df\_encoded.columns = [col.replace(**' '**, **'\_'**) **for** col **in** df\_encoded.columns]  
print(**f'\nencoded data:\n{**df\_encoded.head().to\_string()**}'**)

excluded = [**'CaseOrder'**, **'Customer\_id'**, **'Interaction'**,   
 **'UID'**, **'City'**, **'State'**, **'County'**, **'Job'**]  
y\_n\_cols = [col **for** col **in** medical\_clean.columns  
 **if** medical\_clean[col].nunique() == 2  
 **and** col **not in** excluded]  
num\_cols = [col **for** col **in** medical\_clean.columns  
 **if** medical\_clean[col].dtype **in** [**'int64'**, **'float64'**]  
 **and** col **not in** y\_n\_cols  
 **and** col **not in** excluded]  
cat\_cols = [col **for** col **in** medical\_clean.columns  
 **if** col **not in** num\_cols  
 **and** col **not in** y\_n\_cols  
 **and** col **not in** excluded]  
df\_encoded = medical\_clean[[col **for** col **in** medical\_clean.columns  
 **if** col **not in** excluded]]  
df\_encoded = pd.get\_dummies(df\_encoded, columns=cat\_cols)  
df\_encoded.replace({**'Yes'**: 1, **'No'**: 0}, inplace=**True**)

Fix the column names applied during recoding

df\_scaled = df\_encoded.copy()  
scaler = StandardScaler()  
df\_scaled.loc[:, num\_cols] = \  
 scaler.fit\_transform(df\_scaled.loc[:, num\_cols])  
print(**f'\nscaled data:\n{**df\_scaled.head().to\_string()**}'**)

Scale numeric variables

Select features

selector = SelectKBest()  
X = df\_scaled[[col **for** col **in** df\_scaled.columns **if** col != **'ReAdmis'**]]  
y = df\_scaled[**'ReAdmis'**]  
selector.fit\_transform(X, y)  
feature\_data = pd.DataFrame({**'feature'**: selector.feature\_names\_in\_,  
 **'score'**: selector.scores\_,  
 **'pvalue'**: selector.pvalues\_})  
top\_features = feature\_data[feature\_data.pvalue < 0.05].\  
 sort\_values(by=**'score'**, ascending=**False**)  
print(**f'\nfeature scores:\n{**top\_features.to\_string()**}'**)  
  
columns = [**'ReAdmis'**, **'Initial\_days'**, **'Services\_CT\_Scan'**, **'Children'**,   
 **'Marital\_Divorced'**, **'Services\_Intravenous'**, **'Population'**,   
 **'Initial\_admin\_Emergency\_Admission'**]  
df = df\_scaled[columns]

## C4: Cleaned Data Set

The cleaned data is being submitted with this report as “cleaned\_data\_set.csv”.

# Part IV: Analysis

## D1: Splitting the Data

The data was split so that 30% was reserved for model evaluation and 70% was used for model training. The resulting data sets are being submitted with this report as “X\_train.csv”, “X\_test.csv”, “y\_train.csv”, and “y\_test.csv”.

## D2: Output and Intermediate Calculations

To perform the analysis, the GaussianNB class was used from scikit-learn. This class implements the Naïve Bayes algorithm as a machine learning classifier. Once the model was instantiated, it was fitted to the training data obtained during the data splitting step. With the fitted model, predictions were made on the test data. The true y values and predicted y values were used with the accuracy\_score and roc\_auc\_score scikit-learn functions to calculate the model’s accuracy and area under the curve.

A screenshot of a computer

Description automatically generated with low confidence

The ROC curve was also plotted using scikit-learn’s class function RocCurveDisplay.from\_predictions.

Chart, line chart

Description automatically generated

## D3: Code Execution

The following code was extracted from the file “task1.py” which is being included with the submission of this report.

*# Split into train and test data*X = df\_encoded.loc[:, df\_encoded.columns[1:]]  
y = df\_encoded.ReAdmis  
X\_train, X\_test, y\_train, y\_test = \  
 train\_test\_split(X, y, test\_size=0.3, stratify=y, random\_state=42)  
  
*# Instantiate and fit naive bayes classifier*model = GaussianNB()  
model.fit(X\_train, y\_train)  
  
*# Make predictions*y\_pred = model.predict(X\_test)  
  
*# Metrics*accuracy = accuracy\_score(y\_test, y\_pred)  
auc = roc\_auc\_score(y\_test, y\_pred)  
print(**f'\nAccuracy score: {**accuracy**}\nAUC: {**auc**}'**)  
  
*# Plot ROC curve*RocCurveDisplay.from\_predictions(y\_test, y\_pred, color=**'darkorange'**)  
plt.axline((0, 0), (1, 1), linestyle=**'dashed'**, color=**'gray'**)  
plt.title(**'ReAdmis Naive Bayes Classifier ROC Curve'**)  
plt.show()

# Part V: Data Summary and Implications

## E1: Accuracy and AUC

The accuracy, calculated as the sum of the count of true positives and the count of true negatives divided by the count of all predictions, was 0.96. This means that approximately 96% of the predictions matched the labels.

The AUC or Area Under the Curve was 0.97. This means that, when comparing the true positive rate and the false positive rate on a line plot, approximately 97% of the positive predictions fall below the plotted line and were accurately predicted.

## E2: Results and Implications

This analysis uses a Naïve Bayes Classifier, a supervised machine learning algorithm based on Bayes Theorem (Navlani, 2018), to predict the probability that a patient will be readmitted to the hospital based on features from their initial admission. The specific features used were selected through a series of steps. First, all identifiers (e.g., UID, CaseOrder, etc.) and features with an excessive number of categories were excluded. Second, using scikit-learn’s SelectKBest feature selector, only features with a p-value below 0.05 were kept. Last, the TotalCharge feature was excluded because it’s correlation with the Initial\_days feature violated the algorithm’s assumption of independence between features. As was stated in section E1, the accuracy was 0.96 and the AUC was 0.97. Overall, this model performs well, but Naïve Bayes Classifiers perform better with more data. So, the model could likely be improved by training it on a larger sample. Including additional features, not in the provided dataset, would also likely improve the model’s performance. Features related to admitting diagnosis, in particular, could have a great impact on a model like this.

## E3: Limitations

One limitation of this analysis is that the target variable used was not evenly distributed. There were far fewer readmissions (n = 3,669) in the dataset (n = 10,000) than not (n = 6,331). Because of this, the classifier may have a more difficult time predicting readmissions than if the target distribution were more even.

## E4: Course of Action

Based on the results of this analysis it is recommended that a controlled experiment be designed which uses the model to predict which patients will and will not be readmitted to the hospital to determine how well the algorithm extrapolates to the real world. This would both validate the results of this analysis and expand the data seen by the algorithm, improving its performance further. Once the results have been validated, the model should be implemented in a real-world setting. It could be used to fine tune follow up care after patients discharge from the hospital in an effort to reduce the likelihood that they will be readmitted. This could ultimately lead to the reduction of hospital readmission rates overall.

# Part VI: Demonstration

## F: Panopto Recording

The Panopto video recording is being included with the submission of this report as “task1\_recording”.

## G: Sources for Third-Party Code

Only library documentation was used to support the coding of this analysis. See section B3 for the list of libraries and modules used.

## H: Sources

Navlani, A. (2018, December). *Naive Bayes Classifier Tutorial: with Python Scikit-learn | DataCamp*. Retrieved from datacamp.com: https://www.datacamp.com/tutorial/naive-bayes-scikit-learn