Machine Learning Final Project:

Comparing Performance of Different Algorithms with Different Datasets

CS-620-50

December 7, 2022

Jason McCann

Table of Contents

1. Introduction

In machine learning, many different algorithms are used to attempt to improve performance when evaluating a dataset. Some examples of these algorithms include Logistic Regression and K-Nearest Neighbor. Throughout a semester long study on machine learning, only a handful of these algorithms were discussed and only a small sample of different datasets were used to demonstrate the differences in evaluation performance when using these algorithms. Therefore, this project aims to compare the performance between four algorithms that were learned during this semester's machine learning study, and one additional algorithm not discussed during this semester's study, against five different datasets.

2. Methods

For this comparison project, four algorithms talked about during the semester's machine learning study were utilized. These algorithms include Logistic Regression, K-Nearest Neighbor, Decision Tree, and Random Forest. In addition, the Bernoulli Naïve Bayes algorithm, which was not discussed during the study, was used in the comparisons. The Bernoulli Naïve Bayes algorithm itself is a derivative of the Naïve Bayes algorithm. The Naïve Bayes algorithm is an algorithm that is based on Bayes theorem (Figure 2.1), which itself is a mathematical formula which deals with determining the conditional probability of events. In other words, the theorem describes an event's probability by utilizing prior knowledge coming from similar conditions to that of the event. In the case of the Naïve Bayes algorithm, when given input, probability is predicted for the input being classified for all classes. The algorithm is known as being naïve as it treats all attributes as independent of each other and gives each attribute equal importance

when determining the prediction. The Bernoulli variant of the algorithm is great for evaluating datasets with discrete data that is in binary form (0s or 1s, true or false, yes or no, etc.).

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Figure 2.1: Bayes theorem

The performance of each of the algorithms described above were all evaluated against five different datasets.

The first of these datasets was a dataset for predicting whether or not a patient has chronic kidney disease or not, based on features that relate to diagnostic measurements taken on each individual patient. The output/target feature is the 'Class' feature, which is a binary based feature which describes whether a patient has chronic kidney disease or not (0 being no and 1 being yes). In addition, this dataset contains thirteen input/non-target features. These include 'Bp' (patient's blood pressure), 'Sg' (patient's specific gravity), 'Al' (patient's albumin level), 'Su' (patient's sugar reading), 'Rbc' (red blood cell (0 for no and 1 for yes)), 'Bu' (patient's blood urea level), 'Sc' (patient's serum creatinine level), 'Sod' (patient's sodium level), 'Pot' (patient's potassium level), 'Hemo' (patient's hemoglobin level), 'Wbcc' (patient's white blood cell count), 'Rbcc' (patient's red blood cell count), and 'Htn' (If the patient has hypertension or not (0 for no and 1 for yes)).

The second dataset deals with predicting whether or not an employee at a company will leave or stay, based on a set of features pertaining to each employee. The output/target feature is the 'LeaveOrNot' feature, which is a binary based feature which describes whether an employee will leave the company in the next two years or stay (0 being no and 1 being yes). In addition,

this dataset contains eight input/non-target features. These include ‘Educations’ (the employee’s education level), ‘JoiningYear’ (the year the employee joined the company), ‘City’ (the city of where the office the employee works at is located), ‘PaymentTier’ (numerical categorical placement of what payment tier the employee is in (1: Highest, 2: Mid-Level, 3: Lowest)), ‘Age’ (current age of the employee), ‘Gender’ (employee’s gender), ‘EverBenched’ (yes or no to if the employee has ever been kept out of a project for a month or more), and ‘ExperienceInCurrentDomain’ (employee’s experience in their current field).

The third dataset deals with predicting whether or not a patient has diabetes, based on features that relate to diagnostic measurements taken on each individual patient. The output/target feature is the ‘Outcome’ feature, which is a binary based feature which describes if the patient has diabetes or not (0 being no and 1 being yes). In addition, this dataset contains eight input/non-target features. These include ‘Pregnancies’ (the number of pregnancies the patient has had), ‘Glucose’ (the glucose level in the blood of the patient), ‘BloodPressure’ (the blood pressure measurement of the patient), ‘SkinThickness’ (the thickness of the patients skin), ‘Insulin’ (the insulin level in the blood of the patient), ‘ BMI’ (the body mass index of the patient), ‘DiabetesPedigreeFunction’ (the patients Diabetes percentage), and ‘Age’ (the age of the patient).

The fourth dataset deals with predicting if a student will go to college or not, based on features pertaining to individual students. The output/target feature is the ‘will_go_to_college’ feature, which is a true/false binary based feature which predicts if the student will go to college or not (False being no and True being yes). In addition, this dataset contains ten input/non-target features. These include ‘type_school’ (whether the student currently goes to a academic or vocational school), ‘school_accreditation’ (quality of the school the student is attending, a grade

of A is better than B), ‘gender’ (the students gender), ‘interest’ (the interest level the student has in college), ‘residence’ (if the student lives in a rural or urban type community), ‘parent_age’ (age of the student’s parents), ‘parent_salary’ (the per month salary of the student’s parents (in IDR/Rupiah)), ‘house_area’ (the area of the student's parent’s house in meters squared), ‘average_grades’ (student’s average grade on the 0-100 scale), and ‘parent_was_in_college’ (if the students parent attended college (true or false)).

The fifth dataset deals with predicting if a patient has a greater or lesser chance at having a heart attack, based on features pertaining to individual patients. The output/target feature is the ‘target’ feature, which is a binary based feature which predicts if the patient will have greater chance at a heart attack or a lesser chance (0 being a less chance and 1 being a greater chance). In addition, this dataset contains ten input/non-target features. These include ‘Age’ (age of the patient), ‘Sex’ (the sex of the patient), ‘exang’ (if exercise induced angina in the patient(0 for no and 1 for yes)), ‘ca’ (number of major vessels (0-3)), ‘cp’ (patient’s chest pain level (1 for typical angina, 2 for atypical angina, 3 for non-angina pain, and 4 for asymptomatic)), ‘trtbps’ (patients resting blood pressure (in mm Hg)), ‘chol’ (cholesterol level in the patient (mg/dl)), ‘fbs’ (fasting blood sugar > 120 mg/dl (0 is false and 1 is true)), ‘rest_ecg’ (resting electrocardiographic results in the patient (0 is normal, 1 is having ST-T wave abnormality, 2 is showing probable or definite left hypertrophy)), and ‘thalach’ (the patients maximum heart rate achieved).

The code implementation for this project was all completed using a Python notebook in Google Colaboratory. The implementation started off by mounting the Google Drive and folder which contained the datasets. After this, a separate cell contained all the imports for the various Python modules that were utilized in this project. These modules include pandas, which was used to read in the datasets, and various packages from sklearn, such as the various models that were

used in the comparisons. In a new cell, the learning function was written (Figure 2.2). This function assisted in training the various models with the dataset and getting the performance for the various models with each dataset. At the end of this function, a classification report is printed for both the training and testing datasets, and a confusion matrix is printed for the test dataset. Now the performance for each of the models was evaluated with each dataset. For each of the five instances of datasets, the dataset was first read into the notebook, displayed, and the input and output features were defined and separated. Using train_test_split, the existing input and output features were used to create X and Y training and testing datasets. After this, the X training and testing datasets were standardized (Figure 2.3). From here, each model had its performance evaluated when put up against the dataset. In individual cells for each model, an instance of the model was created and was evaluated with the dataset by using the learning function created earlier in the notebook (Figure 2.4). After the function is run, the model's performance was revealed by way of the printed classification report and confusion matrix. This process was repeated for each of the datasets being utilized in this project. In some cases, such as with the employee and college prediction datasets where most of the features are not numerical, one-hot encoding had to be applied to the dataset (Figure 2.5 and Figure 2.6).

```
# Learning function to get the performance for the training and testing sets, as well as the confusion matrix.
def learning (X_train, X_test, y_train, y_test, model):
  model = model.fit(X_train, y_train)
  predTrainLearning = model.predict(X_train)
  predTestLearning = model.predict(X_test)
  print("Performance (Train): ")
  print(metrics.classification_report(y_train, predTrainLearning))
  print("Performance (Test): ")
  print(metrics.classification_report(y_test, predTestLearning))
  print("Confusion Matrix (Test): ")
  print(metrics.ConfusionMatrixDisplay.from_predictions(y_test, predTestLearning))
  return model
```

Figure 2.2: Learning function

```
# Reading the dataset file from Google Drive
diabetes = pd.read_csv('diabetes.csv')
# Displaying the dataset
print(diabetes)
# Separating the Input features and the Output features
IN_diabetes = diabetes[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']]
OUT_diabetes = diabetes['Outcome']
# Creating X and Y training and testing sets by way of train_test_split
X_diabetes = IN_diabetes.values
y_diabetes = OUT_diabetes.values
X_train_diabetes, X_test_diabetes, y_train_diabetes, y_test_diabetes = train_test_split(X_diabetes, y_diabetes,test_size=0.2, train_size=0.8)
# Standardizing the X training and testing datasets
sc_diabetes = StandardScaler().fit(X_train_diabetes)
X_train_std_diabetes = sc_diabetes.transform(X_train_diabetes)
X_test_std_diabetes = sc_diabetes.transform(X_test_diabetes)
```

Figure 2.3: Setting up a dataset prior to evaluating the models

```
print("KNN Model: ")
neigh_college = KNeighborsClassifier()
neigh_college = learning(X_train_std_college, X_test_std_college, y_train_college, y_test_college, neigh_college)
```

Figure 2.4: Setting up a model and evaluating its performance through the learning function

type_school	school_accreditation	gender	interest	residence	parent_age	parent_salary	house_area	average_grades	parent_was_in_college	will_go_to_college
Academic	A	Male	Less Interested	Urban	56	6950000	83	84.09	FALSE	TRUE
Academic	A	Male	Less Interested	Urban	57	4410000	76.8	86.91	FALSE	TRUE
Academic	B	Female	Very Interested	Urban	50	6500000	80.6	87.43	FALSE	TRUE
Vocational	B	Male	Very Interested	Rural	49	6600000	78.2	82.12	TRUE	TRUE
Academic	A	Female	Very Interested	Urban	57	5250000	75.1	86.79	FALSE	FALSE
Vocational	B	Female	Less Interested	Rural	48	3770000	65.3	86.79	TRUE	FALSE
Academic	A	Male	Very Interested	Rural	52	6680000	85.5	90.39	TRUE	TRUE
Academic	B	Male	Very Interested	Rural	53	5890000	83.3	84.65	TRUE	FALSE
Academic	B	Female	Uncertain	Rural	52	6730000	80.3	88.5	TRUE	TRUE
Academic	B	Female	Very Interested	Rural	47	3880000	68	85.43	TRUE	FALSE
Vocational	A	Male	Not Interested	Rural	57	5680000	67.1	89.45	TRUE	FALSE
Vocational	B	Male	Uncertain	Rural	48	5950000	60.7	84.96	TRUE	FALSE
Vocational	B	Male	Uncertain	Rural	47	6930000	65.4	86.23	TRUE	FALSE
Academic	B	Male	Very Interested	Rural	53	7130000	88.5	91.92	TRUE	TRUE
Academic	A	Male	Very Interested	Urban	51	3370000	67.1	84.09	FALSE	FALSE
Vocational	A	Female	Less Interested	Urban	53	6940000	50.1	89.71	TRUE	TRUE
Academic	B	Male	Very Interested	Rural	48	3970000	63.5	86.38	TRUE	FALSE
Vocational	B	Female	Very Interested	Rural	49	6790000	75.9	85.68	TRUE	FALSE
Vocational	A	Female	Less Interested	Rural	47	7180000	68.3	84.9	FALSE	FALSE
Vocational	B	Female	Very Interested	Rural	47	4090000	54.5	85.94	TRUE	FALSE
Vocational	B	Female	Very Interested	Urban	52	6680000	64.5	87.64	TRUE	TRUE

Figure 2.5: The college prediction dataset has many instances of useful features which are not in numerical form

```
oheCollege = pd.get_dummies(college[['type_school','school_accreditation','gender','interest'
```

Figure 2.6: Using one-hot encoding to turn non-numerical features into numerical features

3. Results

For the first dataset, all models scored a high accuracy rate based on the reported classification reports and confusion matrices. The performance of each model against the testing dataset all scored within the range of 0.90 and 1.00. The model with the highest accuracy score was the Decision Tree model and the weakest accuracy was the Random Forest model. The accuracy score of the Bernoulli Naïve Bayes model sits around the middle of the five models, along side the K-Nearest Neighbor model. The resulting classification reports and confusion matrices are shown below in Figure 3.1.

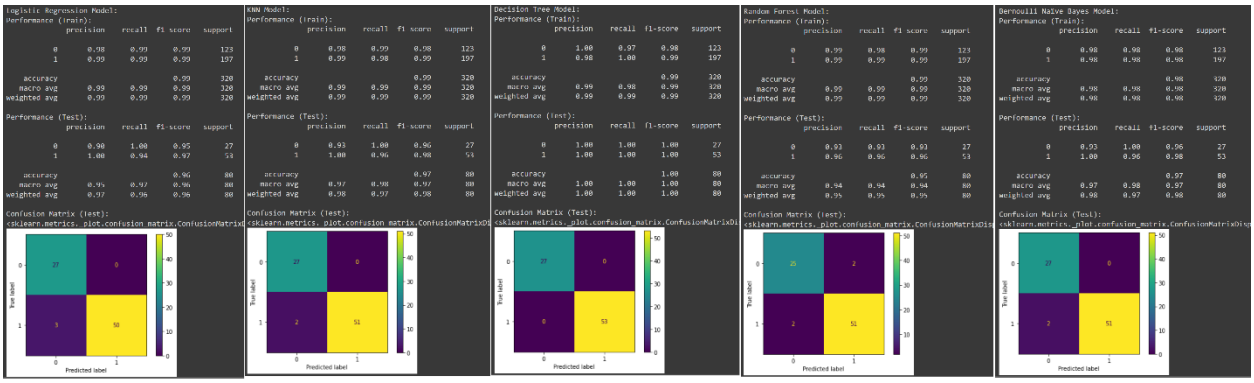

Figure 3.1: Classification reports and confusion matrix for each of the five models when evaluated against the first dataset

For the second dataset, the range of performance between all five models is larger than the previous dataset. The accuracy scores for evaluation with the testing dataset, based on the classification reports and confusion matrices, fall within the range of 0.70 and 0.85. The Bernoulli Naïve Bayes model earned the lowest accuracy score; however, it still comes close to the accuracy score of the Logistic Regression model. The K-Nearest Neighbor model received the highest accuracy score, with Decision Tree and Random Forest models falling not too far behind. The resulting classification reports and confusion matrices are shown below in Figure 3.2.

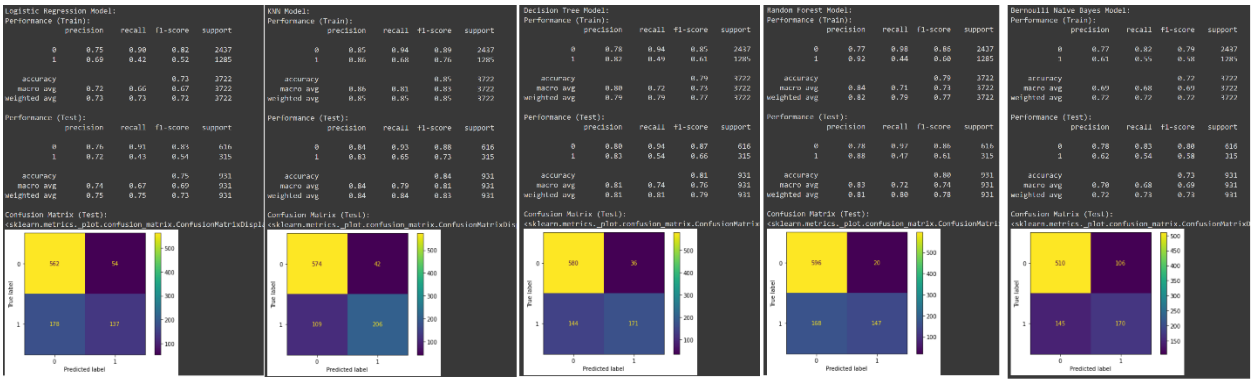

Figure 3.2: Classification reports and confusion matrix for each of the five models when evaluated against the second dataset

For the third dataset, the range of performance between the five models gets a lot closer than with the previous dataset. The accuracy scores for evaluation with the testing dataset, based on the classification reports and confusion matrices, fall within the range of 0.65 and 0.75. The Logistic Regression model came ahead with the highest accuracy score and the Decision Tree model fell to receive the lowest accuracy score. The accuracy score of the Bernoulli Naïve Bayes model sits right in the center of the five models, behind Logistic Regression and K-Nearest Neighbor. The resulting classification reports and confusion matrices are shown below in Figure 3.3.

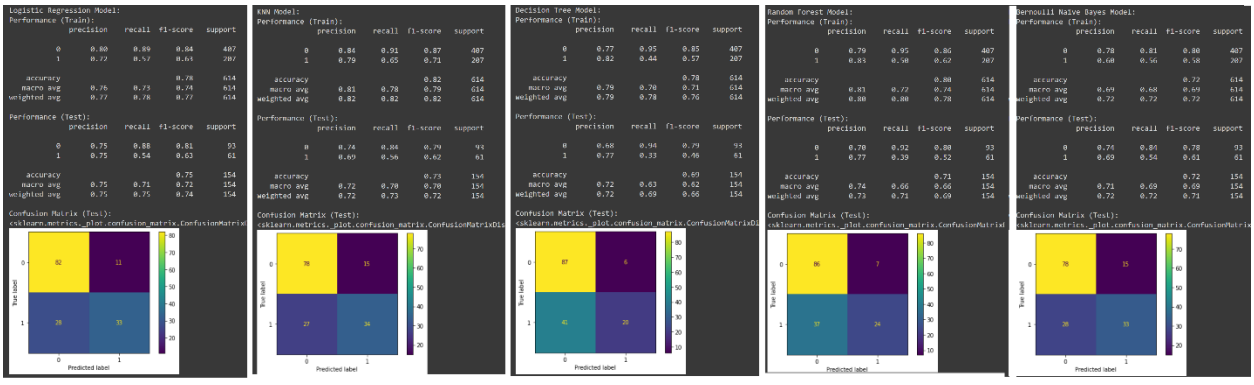

Figure 3.3: Classification reports and confusion matrix for each of the five models when evaluated against the third dataset

For the fourth dataset, the difference in the performance between the five models is once again relatively close. The accuracy scores for evaluation with the testing dataset, based on the classification reports and confusion matrices, fall within the range of 0.80 and 0.90. The Random Forest model came through with the highest accuracy score, with the Logistic Regression model's accuracy score coming in not that much further behind. The accuracy scores for the Bernoulli Naïve Bayes model and the Decision Tree model tie for the lowest out of the five. The resulting classification reports and confusion matrices are shown below in Figure 3.4.

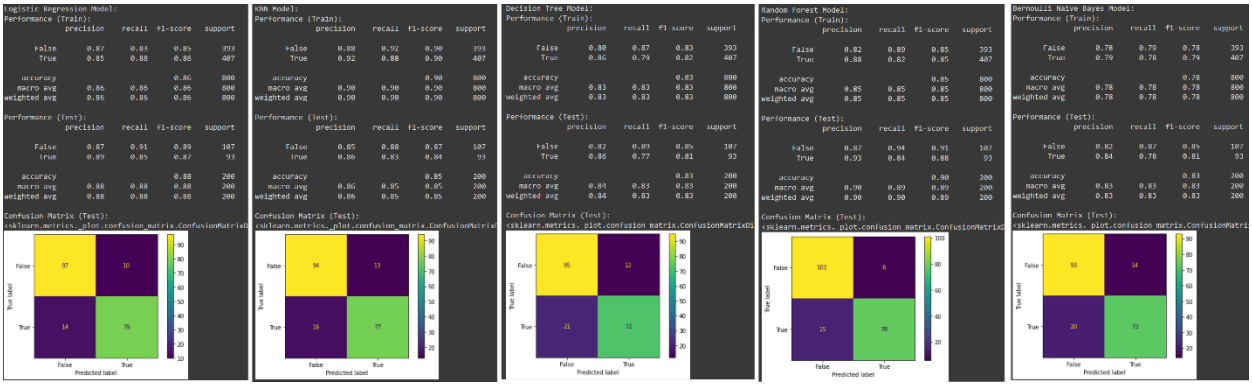

Figure 3.4: Classification reports and confusion matrix for each of the five models when evaluated against the fourth dataset

For the last dataset, the differences in performance between the five models is close again. The accuracy scores for evaluation with the testing dataset, based on the classification reports and confusion matrices, fall within the range of 0.70 and 0.80. The Bernoulli Naïve Bayes model received the highest accuracy score out of the five models, with the Logistic Regression model and the K-Nearest Neighbor model tying for a close second highest score out of the five. The Decision Tree model received the lowest accuracy score out of the five. The resulting classification reports and confusion matrices are shown below in Figure 3.5.

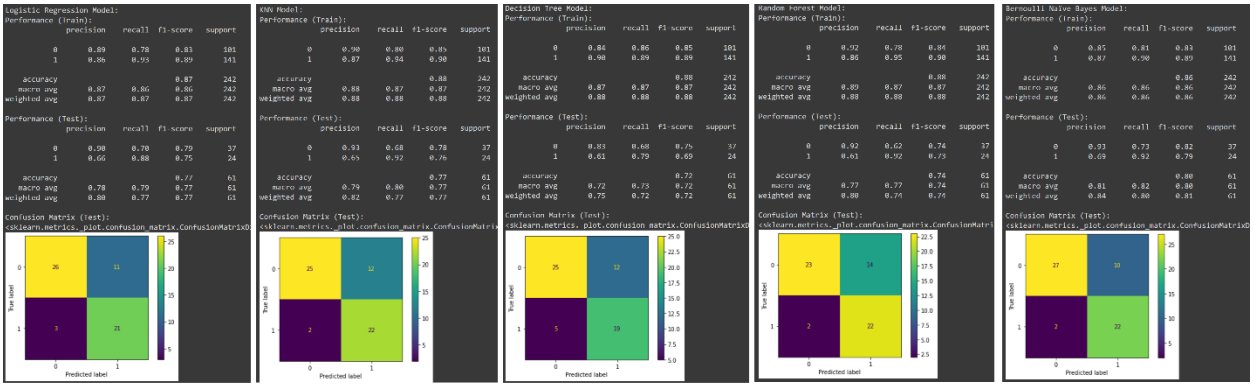

Figure 3.5: Classification reports and confusion matrix for each of the five models when evaluated against the fifth dataset

4. Conclusions

Upon looking at the results of the project above, some conclusions can be made. Throughout all five tests of the five models, no one model in particular came out superior every time. Some models finished their evaluation with the highest accuracy score, and other times would fall to have the lowest accuracy score. Therefore, it could be concluded that every algorithm handles data differently and has its strengths and weaknesses, and therefore performs better with datasets containing certain types of data in certain formats as opposed to others. One pattern to note though from this project was that while not a single model had the highest accuracy score every time, the K-Nearest Neighbor model did finish within the top three scores in every test conducted. Finally, one last thing to note about these results is that they show that the Bernoulli Naïve Bayes model earns its best scores when dealing with data that is more binary centric. The highest overall accuracy scores, with the exception of the first test, for the Bernoulli Naïve Bayes model came from evaluating the fourth and fifth datasets. One thing to note about these two datasets is that they contain more binary based features than any of the other datasets

tested in this project. The only outlier going against this observation is the test conducted with the first dataset, though all models had high accuracy scores for this dataset in particular.

5. Future Work

Upon completion of this project, some obvious ways of improving the project or expanding upon it were brought forward. For one, the experiment could be replicated again by utilizing an algorithm different from Bernoulli Naïve Bayes for the comparison. There are many different algorithms in addition to the five that were tested, each with their own advantages and disadvantages with working with particular types of data in the datasets. Therefore, utilizing a different fifth algorithm should provide better or worse performance to that shown by Bernoulli Naïve Bayes when in comparison to the other four algorithm's performances. Maybe in the future, a different variant of Naïve Bayes, or a variant of support vector machine (SVM) could be used for example. In addition, another major way this experiment could be improved upon is by utilizing the same algorithms with datasets that contain more binary type features. As was discussed above with the conclusions, the Bernoulli Naïve Bayes algorithm saw better performance, in some cases even exceeding the performances of the other four algorithms, when evaluating datasets that contained more binary based features. It is possible that if used with a selection of binary based datasets that the performance of Bernoulli Naïve Bayes may more commonly match, or even exceed, that of the other four algorithms that it is being compared to.

Project Resources

Dataset Resources:

- https://www.kaggle.com/datasets/abhia1999/chronic-kidney-disease
- https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset
- https://www.kaggle.com/datasets/saddamazyazy/go-to-college-dataset
- https://www.kaggle.com/datasets/tejashvi14/employee-future-prediction
- https://www.kaggle.com/datasets/whenamancodes/predict-diabities

Informational Resources:

- https://corporatefinanceinstitute.com/resources/data-science/bayes-theorem/
- https://iq.opengenus.org/bernoulli-naive-bayes/
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html
- https://thecleverprogrammer.com/2021/07/27/bernoulli-naive-bayes-in-machine-learning/
- https://thecleverprogrammer.com/2021/02/07/naive-bayes-algorithm-in-machine-learning/