PATIENTORY INC.

Kidney Disease and Fitness

Team B	Georgia Tech ID
Kaushik Biswas	kbiswas7
Shrikanth Mahale	smahale6
Jaymin Patel	jpatel385

Contents

Introduction 2
Kidney Disease 2
Fitness 2
Analytical Approach 2
Kidney Disease - Dataset 2
Apple Health – Fitness Dataset 2
Data Collection & Data Requirements 3
Data Extraction and Storage: 3
Data Cleansing and Preparation: 5
Schema/Diagram of dataset 6
Data Understanding & Feature Selection 7
Response variable 7
Features 8
Imputing missing values 9
Check for multicollinearity 10
Response/Predictor separation and Data Profiling 11
Scaling of dataset 13
Modeling 13
SVM 13
KNN (K-nearest Neighbors Classifier) 13
Logistic Regression 13
Decision Tree 14
Random Forest 14
Hyperparameter Tuning 14
Evaluation of Models' accuracy 15
Training/Test data split 15
Confusion Matrix 15
KNN 16
Logistic Regression 16
Decision Tree 17
Random Forest 18
SVM 19
Conclusion 20
Visualization 21
References 23

Introduction

Patientory is a health data management and analytics company providing various services to patients, medical professionals, healthcare organizations and employers with actionable insights into healthcare data to improve population health outcomes. Patientory executes such services by identifying at-risk populations and creating plans to change behaviors and make decisions to reduce the overall cost of care through early intervention.
As a part of practicum engagement with Patientory, our team was tasked with identifying actionable insights from two separate datasets provided by Patientory, namely:

Kidney Disease

General Health Dataset containing of 400+ Patients with different demographics and health metrics. Identification of patients with actual kidney diseases was provided in this large population dataset.
Challenge:
Objective of our modeling exercise is to identify patients most likely to develop kidney disease of any sort by analyzing the health metrics.

Fitness

Apple Healthkit dataset contains numerous observations like blood pressure, heart rate, nutrient consumption levels and physical activities for one patient over period.
Assumptions:

- Apple Healthkit data set provided by Patientory is for one patient only. This dataset is simulated and not real-life data for the patient. Also, the quality of fitness data from the CDA xml modeling exercise was not good enough to perform any meaningful predictive modeling.
- Due to quality of the dataset, this project aims to provide visualizations of various trends for the dataset only (not predictive models)

Challenge:
Objective of analyzing this dataset is to provide meaningful insights and recommendations to patients with potential of developing health issues in future.

Analytical Approach

To address each dataset and respective objective, we defined problem statement as follows:

Kidney Disease - Dataset

- Analyze all metrics/features which can influence and/or demonstrate presence of Kidney Disease
- Estimate the order of importance of each such feature
- Explore various modeling techniques to find the most accurate predictive one
- Evaluate accuracy of each models explored in previous step to identify most effective model addressing out problem statement

Apple Health – Fitness Dataset

- Gather and analyze all possible health metrics available for each patient
- Group the patients most likely to behave and respond identically in clusters
- Identify the behavior pattern of each cluster for recommending health improvement or preventive care

For each dataset, we developed methodology comprising of below steps in given order:

- Data Collection & Requirements
  Define process and programs to collect and refine datasets most relevant for our analytical process
  Prepare and organize each dataset in a format which is most efficient for executing further modeling analysis

- Data Understanding & Features Selection
  - Analyze every possible data attribute affecting the response variable (in case of Kidney Disease set) or relevant in grouping patients based on their health habits and activities (for health dataset)
  - Use various comparative analysis/tools to identify the most relevant features
- Modeling
  Implement and evaluate different supervised and unsupervised models for further analysis
- Evaluation of Models
  Calculate accuracy and effectiveness of each model and select the best option based on findings
- Visualization
  Provide visualization enabling "story telling" aspects of our project by depicting findings from Modeling exercise

Each of these steps are laid out with more details in subsequent sections of this report.

Data Collection & Data Requirements

Data Extraction and Storage:

Kidney Disease:

Fast Healthcare Interoperability Resources (FHIR, pronounced "fire") is a standard describing data formats and elements (known as "resources") and an application programming interface (API) for exchanging electronic health records (EHR). The standard was created by the Health Level Seven International (HL7) health-care standards organization.

FHIR builds on previous data format standards from HL7, like HL7 version 2.x and HL7 version 3.x. But it is easier to implement because it uses a modern web-based suite of API technology, including a HTTP-based RESTful protocol, and a choice of JSON, XML or RDF for data representation.[1] One of its goals is to facilitate interoperability between legacy health care systems, to make it easy to provide health care information to health care providers and individuals on a wide variety of devices from computers to tablets to cell phones, and to allow third-party application developers to provide medical applications which can be easily integrated into existing systems.

Open source FHIR standard provides you a robust data model covering most important healthcare domains. Fhirbase is a common name for a command line tool and set of libraries aiming to lower the entry barrier for both FHIR and PostgreSQL. It gives you a ready to use components to boost your development, as well as guidelines how to store and access your FHIR data. With Fhirbase you can break FHIR API abstraction and operate with FHIR data on database level.

Steps: in line with numbering on the diagram in Figure 1

1. Export data from Synthea using the generic jar for 10,000 patients in json format
2. Combine single patients json files into a batch new line delimited json file
3. Use the fhirbase utility to initialize a schema with tables in postgres database
4. Populate the data model with the patients' data
5. Filter out kidney patients using the kidney patients and move the data into an AWS RDS postgres instance for sharing across the team
6. Use python to connect to the data

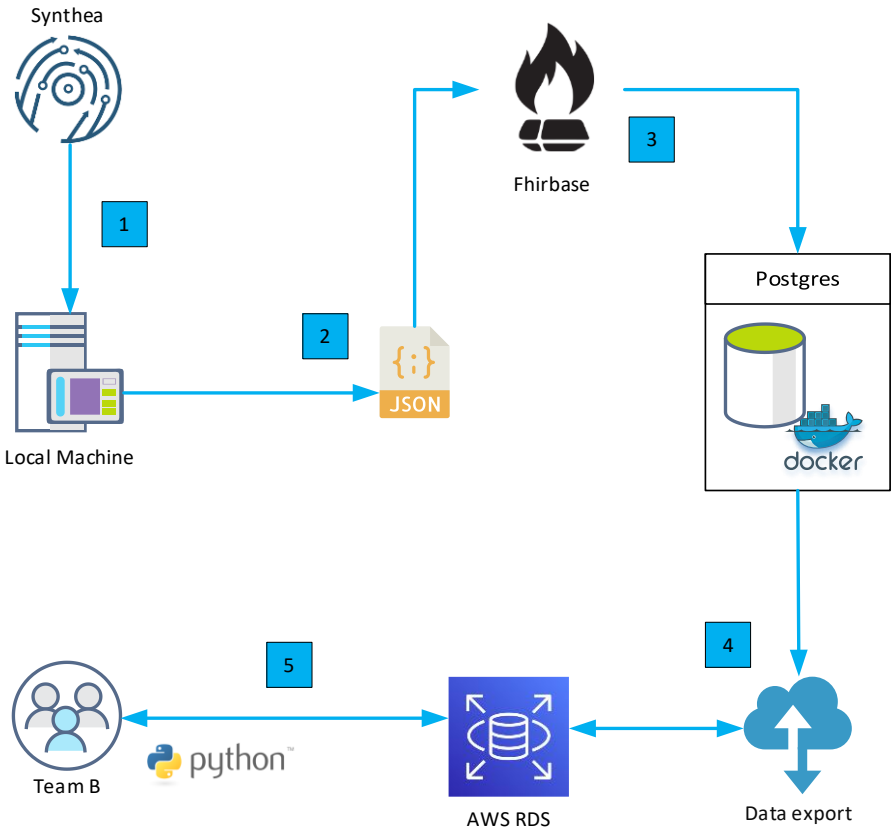

Figure 1 Kidney Disease Process Flow - Data load

Fitness Data:

Apple HealthKit provides a central repository for health and fitness data on iPhone and Apple Watch. With the user's permission, apps communicate with the HealthKit store to access and share this data. HealthKit is also designed to manage and merge data from multiple sources. For example, users can view and manage all their data in the Health App, including adding data, deleting data, and changing an app's permissions. Therefore, your app needs to handle these changes, even when they occur outside your app.

pgfutter was used to import CSV and line delimited JSON into PostgreSQL the easy way. This small tool abstracts all the hassles and swearing you normally must deal with when you just want to dump some data into the database.

Steps: in line with numbering on the diagram in Figure 2

1. Apple Healthkit data export for 1 patient
2. Combine single patients json files into a batch new line delimited json file
3. Use the fhirbase utility to initialize a schema with tables and populate in postgres database
4. Parse fitness data from xml files, export to csv and move to postgres using pgfutter
5. Export data into AWS postgres for sharing across team
6. Use python to connect to the data

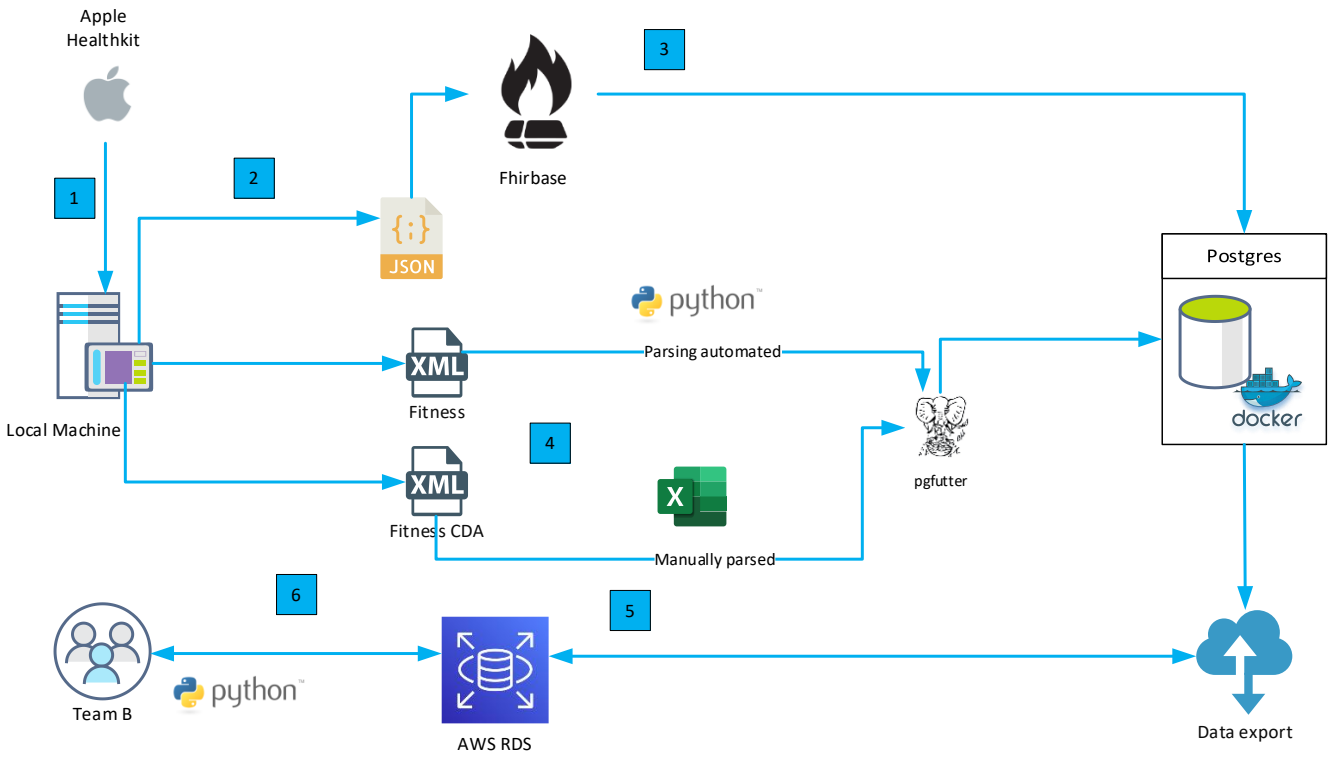

Figure 2 Fitness Data Process Flow - Data Load

Data Cleansing and Preparation:

To load Kidney Patients relevant datasets, following steps were developed and executed

1. Download dataset from *Synthea* in json file format. This dataset consists of 12000 patients and all their observations. Dataset contains information about patients and various details for patients as follow:
   Patients: Contains demographics, gender, address details of patients
   Conditions: Conditions applicable to patients with description and duration. For example, High BMI, Hypertension, Diabetes etc.
   Observations: Various health metrics captured for patients like Blood Pressure, Blood Sugar level, Respiratory Rate etc.
   Care plan Data: Data about specific care plan patient was put on with duration details. For example, alcohol free diet, low sodium diet etc.
   Procedure: Any procedure/exams the patient has been through. For example, Pelvic examination, Sputum examination etc.

2. Parsed and loaded each json file into Postgres database table using Fhirbase utility
   a. Patient, derive additional fields: age
   b. Condition
   c. Observation: we extracted only those observations which are relevant for Kidney diseases (see next section for rationale behind this filter)
   d. Care plan data
   e. Procedure data
3. Below steps were executed to derive final schema
   a. Merge Patient Data with Condition, and extract only those conditions which are relevant for Kidney Disease

Below are the filter criteria for selecting only relevant dataset where observation matches any of the below

b. Merged all: patient, condition, observation, and care plan data to form a unified flat structure with all the elements in single row. This will enable easier analysis during the subsequent phase

Schema/Diagram of dataset

Fhirbase schema is regular and uniform. Each resource is saved in a table with the lower case name of the resourceType. For example, the Patient resource is saved in the patient table. All resource tables has similar structure:

- id is an resource ID
- txid is the versionId for the resource and at the same time the sequential id of logical transaction, which can be used for implementing integrations, ETags, reactive APIs, etc.
- ts stores a timestamp of last resource update
- status is enumeration with possible values: create, updated, deleted, recreated
- resource stores JSON representation of a resource in slightly altered format.

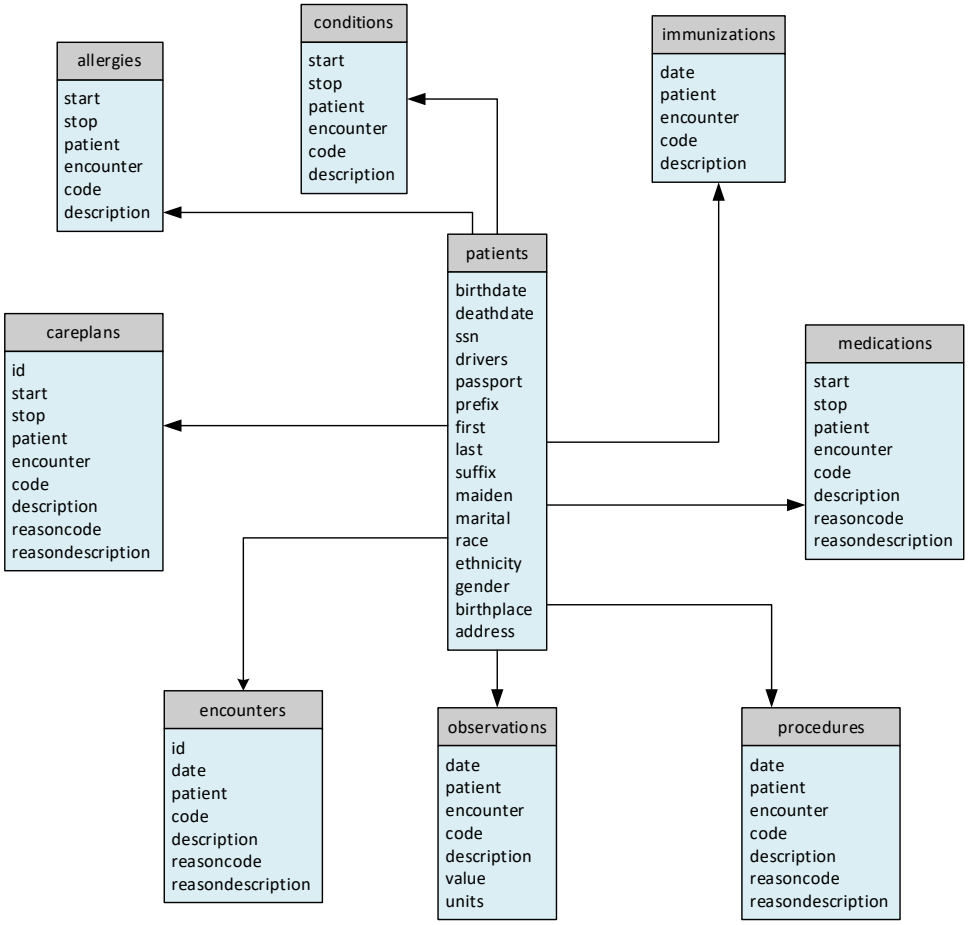

Figure 3 Kidney Disease Data Model

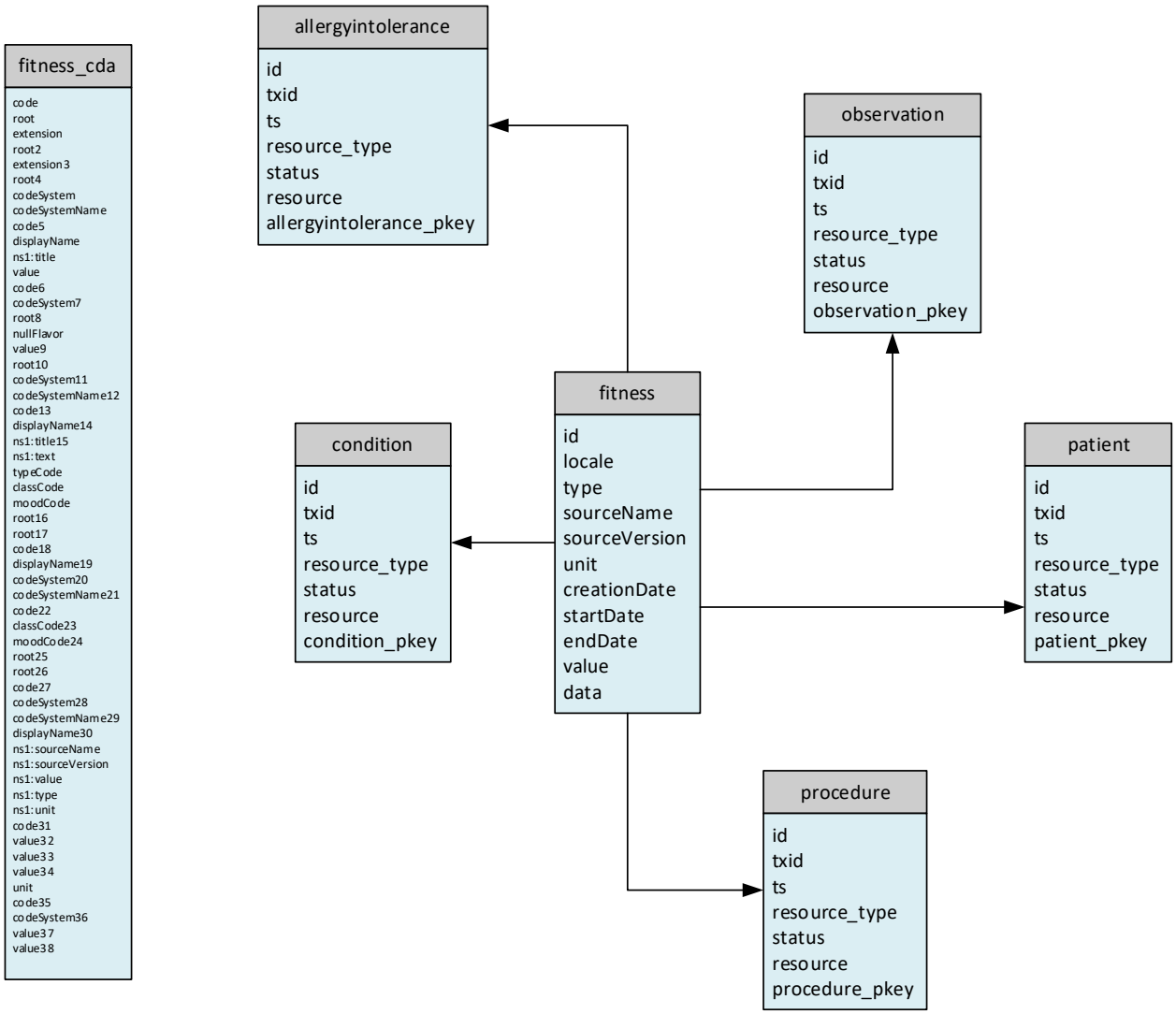

Figure 4 Fitness Data Model

Data Understanding & Feature Selection

Response variable

The dataset used for this supervised modelling contained response variable with four different types of Kidney Disease as outcome, namely :

1. Chronic Kidney Disease Stage-1
2. Chronic Kidney Disease Stage-2
3. Chronic Kidney Disease Stage-3
4. Injury of Kidney

Distribution of patients by each Kidney Disease is presented below :

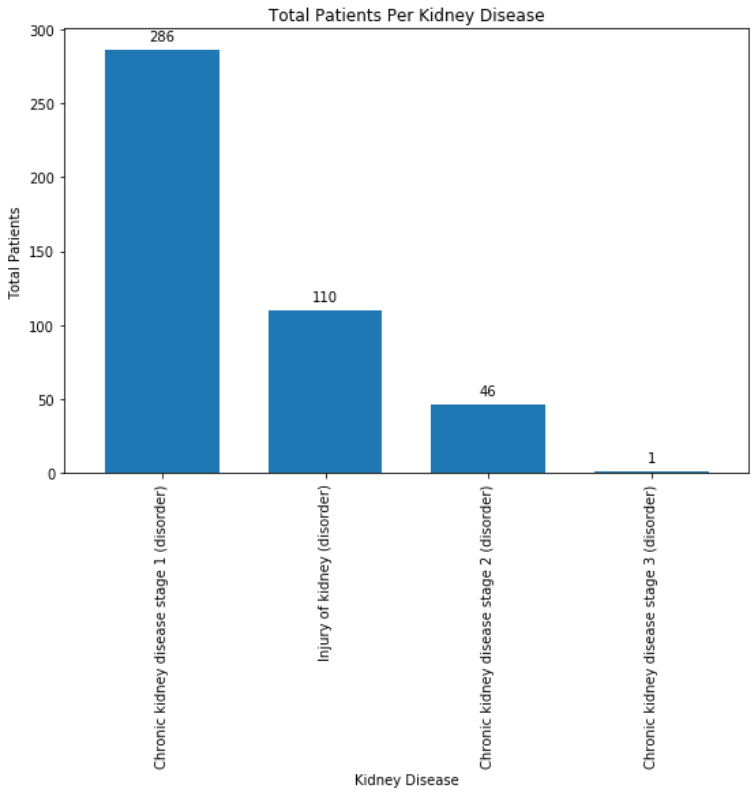

Figure 5 Patients by Kidney Disease Stages

Features

To understand factors relevant for Kidney diseases, total 181 features of data observations were taken.
After detailed search about each observation, potential list of 25 features which might be relevant were filtered. From this point onwards, consultation of medical expert was taken. This professional expertise helped us weeding out observations irrelevant and focus only those observations which may influence or predict kidney diseases. This resulted in below, below observations were selected for the study:

1. Calcium
2. Microalbumin Creatinine Ratio
3. Estimated Glomerular Filtration Rate
4. Blood Pressure
5. Respiratory rate
6. Urea Nitrogen
7. Creatinine
8. Sodium
9. Potassium
10. Chloride
11. Glomerular filtration rate/1.73 sq M.predicted
12. Globulin [Mass/volume] in Serum by calculation
13. Hemoglobin [Mass/volume] in Blood
14. Urea nitrogen [Mass/volume] in Serum or Plasma
15. Creatinine [Mass/volume] in Serum or Plasma

16. Calcium [Mass/volume] in Serum or Plasma
17. Sodium [Moles/volume] in Serum or Plasma
18. Potassium [Moles/volume] in Serum or Plasma
19. Chloride [Moles/volume] in Serum or Plasma
20. NT-proBNP
21. Appearance of Urine

With this level of filtering from observations dataset, the final dataset consisted of below dimensions. All the possible features (listed below) were analyzed to determine final list of features:

Table 1 Features by Entity

Dataset	Possible features considered
Patient	gender_male gender_female age
Care plan	Administration of intravenous fluids Alcohol-free diet Low salt diet education (procedure) Low sodium diet (finding) Urine screening low salt diet education
Procedure	Gonorrhea infection test Manual pelvic examination (procedure) RhD passive immunization Sputum examination (procedure)
Observation	Calcium Calcium [Mass/volume] in Serum or Plasma Chloride Chloride [Moles/volume] in Serum or Plasma Creatinine Creatinine [Mass/volume] in Serum or Plasma Estimated Glomerular Filtration Rate Globulin [Mass/volume] in Serum by calculation Glomerular filtration rate/1.73 sq M.predicted Hemoglobin [Mass/volume] in Blood Microalbumin Creatinine Ratio NT-proBNP Potassium Potassium [Moles/volume] in Serum or Plasma Respiratory rate Sodium Sodium [Moles/volume] in Serum or Plasma Urea Nitrogen Urea nitrogen [Mass/volume] in Serum or Plasma

To prepare the dataset for building model, additional preparation steps were performed.

Imputing missing values

All datasets we used had missing values which can cause problems for building any meaningful machine learning model on the dataset.
To avoid this, we needed to replace missing values for each possible feature prior to modeling your prediction task. This process is called missing data imputation or imputing.

We chose to use Iterative Imputing method rather than KNN imputing method. This method involves defining a model to predict each missing feature as a function of all other features and to repeat this process of estimating feature values multiple times. The repetition allows the refined estimated values for other features to be used as input in subsequent iterations of predicting missing values.

We applied Iterative Imputer available from sklearn.impute for this purpose and iterated over each feature.

Check for multicollinearity

Multicollinearity exists when there are high correlations between two or more predictor variables.
When one predictor variable can be used to predict the other, it results in multicollinearity because this generates redundant information and confusion, distorting the results in a regression model.

An easy way to detect multicollinearity is to calculate correlation coefficients for all pairs of predictor variables.
If the correlation coefficient, r, is exactly +1 or -1, this is called perfect multicollinearity. If r is close to or exactly -1 or +1, one of the variables should be removed from the model if possible.

We used variance inflation factor (VIF) to detect multicollinearity.
The variance inflation factor is a measure for the increase of the variance of the parameter estimates if an additional variable is added to the linear regression. It is a measure for multicollinearity of the design matrix One recommendation is that if VIF is greater than 5, then the explanatory variable given by exog_idx is highly collinear with the other explanatory variables, and the parameter estimates will have large standard errors because of this.

Table 2 Features and VIF Factor

	VIF Factor	features
0	13544.474060	gender_male
1	5444.098927	gender_female
2	3.472160	age
3	1.067933	obs_Calcium
4	1.065158	obs_Calcium [Mass/volume] in Serum or Plasma
5	1.071456	obs_Chloride
6	1.091690	obs_Chloride [Moles/volume] in Serum or Plasma
7	1.370989	obs_Creatinine

8	1.203581	obs_Creatinine [Mass/volume] in Serum or Plasma
9	1.970852	obs_Estimated Glomerular Filtration Rate
10	1.119153	obs_Globulin [Mass/volume] in Serum by calcula...
11	1.523740	obs_Glomerular filtration rate/1.73 sq M.predi...
12	1.279103	obs_Hemoglobin [Mass/volume] in Blood
13	2.136513	obs_Microalbumin Creatinine Ratio
14	3.190345	obs_NT-proBNP
15	1.059004	obs_Potassium
16	1.103347	obs_Potassium [Moles/volume] in Serum or Plasma
17	1.369081	obs_Respiratory rate
18	1.065519	obs_Sodium
19	1.074300	obs_Sodium [Moles/volume] in Serum or Plasma
20	1.103430	obs_Urea Nitrogen
21	1.072197	obs_Urea nitrogen [Mass/volume] in Serum or Pl...

In above results, we accepted high VIF value of gender as it being categorical variable. We created two dummy variables out of Gender. Since both are easily derived from each other (i.e., when Gender = Male, it cannot be Female and vice versa), higher VIF are expected. So, we chose to ignore the VIF analysis for these and accepted Gender as one of the critical variables to be considered.

Response/Predictor separation and Data Profiling

Further, separation of all predictor variables from response variable was done. Additional data profiling and analysis executed for better understanding of dataset.
Following visualizations describe distribution of total patients by age, gender, and type of kidney disease

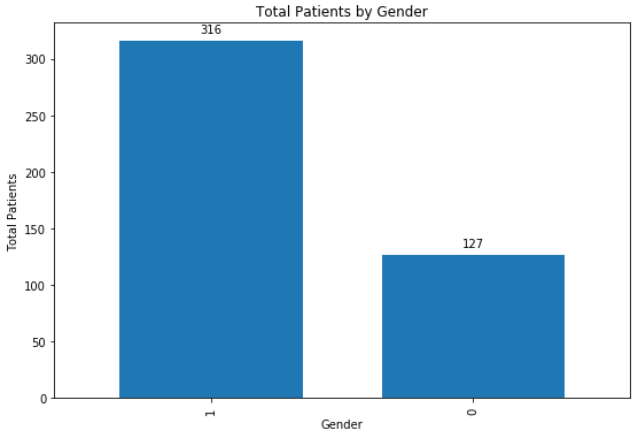

Figure 6 Bar graph by Gender

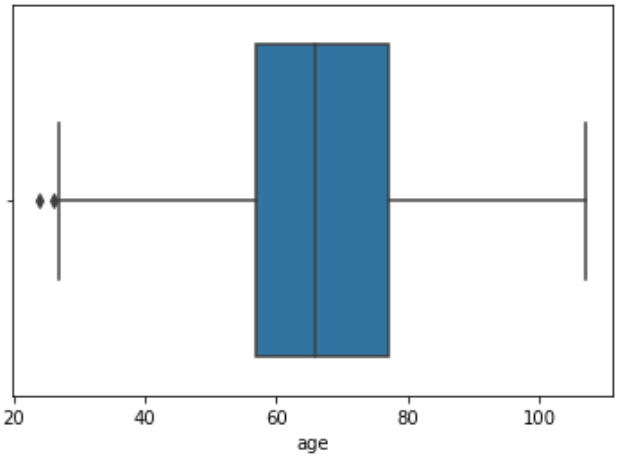

Figure 7 Univariate analysis for Age – Boxplot

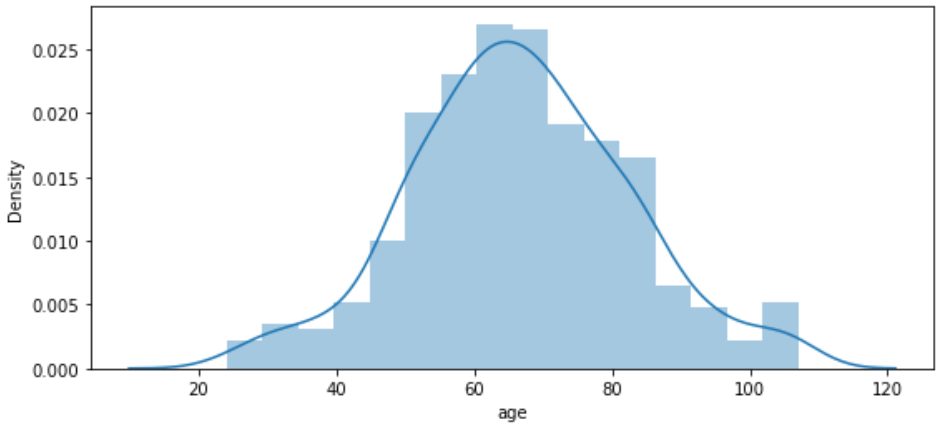

Figure 8 Univariate analysis for Age (distributed plot)

Scaling of dataset

Since many of the features are having wide range of values, standard scaling of data is essential.
StandardScaler library is used to scale the data.
The idea behind StandardScaler is that it will transform data in a way that its distribution will have a mean value 0 and standard deviation of 1.
We have multivariate dataset, so in our case this is done feature-wise (in other words independently for each column of the data).
Given the distribution of the data, each value in the dataset will have the mean value subtracted, and then divided by the standard deviation of the whole dataset (or feature in the multivariate case)

Modeling

Out of all supervised modeling techniques, below were chosen to be explored on Kidney Disease dataset.

SVM

SVM is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then the objective is to find some line that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.

SVM models can be tuned for C & Kernel to find the best possible accuracy:

C parameter adds a penalty for each misclassified data point. If C is small, the penalty for misclassified points is low so a decision boundary with a large margin is chosen at the expense of a greater number of misclassifications. If C is large, SVM tries to minimize the number of misclassified examples due to high penalty which results in a decision boundary with a smaller margin. Penalty is not same for all misclassified examples. It is directly proportional to the distance to decision boundary.

Kernel Function is a method used to take data as input and transform into the required form of processing data. "Kernel" is used due to set of mathematical functions used in Support Vector Machine provides the window to manipulate the data. So, Kernel Function generally transforms the training set of data so that a non-linear decision surface can be transformed to a linear equation in a higher number of dimension spaces. Basically, it returns the inner product between two points in a standard feature dimension.
Some of the kernels used in SVM are Linear, Gaussian, RBF (Radial Basis Function) etc.

KNN (K-nearest Neighbors Classifier)

K-nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function. These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance.

Various parameters which can be tuned further for KNN are:
Leaf Size: The optimal leaf size (in integer) below which you do not want to group any subset.
Number of neighbors (K): How many nearest neighbors to use for KNN.
P: Power parameter for the Minkowski metric. When p = 1, this is equivalent to using Manhattan distance. When p = 2 this will be Euclidean distance

Logistic Regression

Logistics Regression can be used as a classifier and as a regression. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variables. It predicts the probability of occurrence of an event by fitting data to a logit function. In our study, its probability of patient having kidney disease.

The trade-off parameter of logistic regression that determines the strength of the regularization is called C, and higher values of C correspond to less regularization (where we can specify the regularization function).C is the Inverse of regularization strength(lambda).

Decision Tree

Decision Tree is a type of supervised learning algorithm that is mostly used for classification problems. It works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets based on various values of features. This is done based on most significant attributes/ independent variables to make as distinct groups as possible. Decision trees classify the examples by sorting them down the tree from the root to a leaf node, with the leaf node determines the classification.

Every node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is implemented recursively until all classifications are covered

Few parameters which can be tuned further for Decision Tree model:
Criterion
The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.
Splitter
The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.
Minimum Samples in leaf
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model,

Random Forest

Random Forest is a tree-based machine learning algorithm that leverages the power of multiple decision trees for making decisions.
Contrary to Decision Tree model, every node in Random Forest's individual decision tree works on a random subset of features to calculate the output.
The random forest then combines the output of individual decision trees to generate the final output through a process called ensembles learning.

Parameter to tune for most efficient Random Forest model can be “Number of estimators”.
This is the number of trees you want to build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance while creating poor performance in execution

Hyperparameter Tuning

Process of identifying the best possible control/hyper parameters for any model is called the hyperparameter tuning. There are many ways to accomplish this, like Random Search or Grid Search.

In the grid search method, a grid of possible values for hyperparameters is created. Each iteration tries a combination of hyperparameters in a specific order. It fits the model on each combination of hyperparameter and measure the "fit" or accuracy of the model. It returns the best model with the best hyperparameters. We chose Grid Search method to determine optimal values of hyperparameters (listed below) for each type of models described above.

Table 3 Hyperparameter tuning

Model	Python Libraries	Hyperparameters Evaluated
KNN	sklearn.neighbors.KNeighborsClassifier	leaf_size: list(range(1,50)), n_neighbors : list(range(1,30)), p: [1,2]
Logistic Regression	sklearn.linear_model.LogisticRegression	C: list(range(1,10))
Random Forest	sklearn.ensemble.RandomForestClassifier	n_estimators: list(range(1,50))
Decision Tree	sklearn.tree.ecisionTreeClassifier	criterion: [gini,entropy], splitter : [best,random], min_samples_leaf : list(range(1,50))
Support Vector Machine	sklearn.svm	C: list(range(1,20)), kernel: [rbf,linear]

Following were selected as the best values of all hyperparameters:

Table 4 Best Parameters

Model	Best Parameters
KNN	{'leaf_size': 1, 'n_neighbors': 19, 'p': 1}
Logistic Regression	{'C': 3}
Random Forest	{'n_estimators': 45}
Decision Tree	{'criterion': 'gini', 'min_samples_leaf': 25, 'splitter': 'best'}
SVM	{'C': 2, 'kernel': 'rbf'}

Evaluation of Models' accuracy

Training/Test data split

The input dataset is split into training and test dataset to evaluate model accuracy. All the models described above were trained using train split and then tested for accuracy on the test split. We used ratio of 80% training and 20% test set for our evaluation across all models.

Confusion Matrix

A Confusion matrix is a matrix used for evaluating the performance of a classification model
The matrix compares the actual values of response variables with what is predicted by the machine learning model. in our case, we have 4 possible outcomes (or types of Kidney Diseases). So, our Confusion matrix will be 4x4.

True Positive is a scenario where actual value was positive and model predicted value is also positive making model accurate.
True Negative is when model predicted negative value and actual value is also negative, again making model accurate.
False Positive and False Negative occur when Model predictions are wrong, respectively. Actual Accuracy of a model can be calculated by summing all true positives & true negatives over total predictions.

We have tried to leverage confusion matrix mechanism wherever possible to compare accuracies of each model described above.

KNN

Using 80% training dataset KNN was trained and then evaluated on remaining 20% of test dataset. Below is the confusion matrix for KNN providing accuracy of 0.81

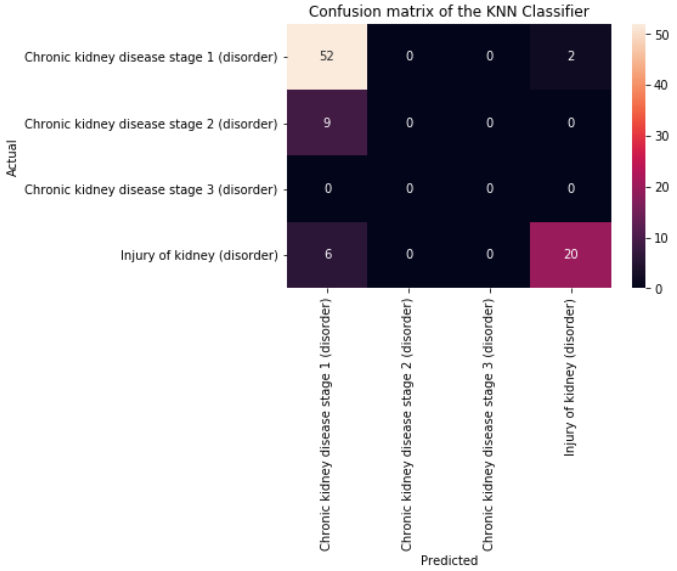

Figure 9 Confusion Matrix - KNN

Logistic Regression

Using same split as defined for all the models, Logistic Regression produced below Confusion matrix for the test dataset, resulting in accuracy of 0.78

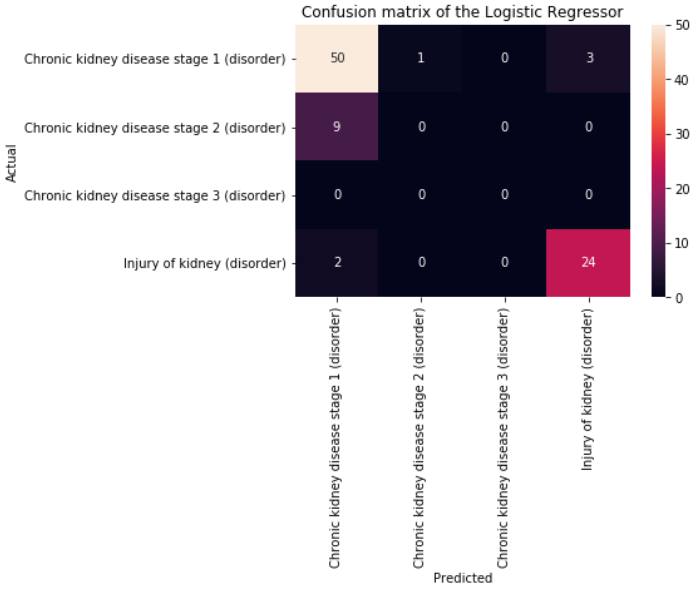

Figure 10 Confusion Matrix - Logistic

Decision Tree

Decision Tree model generated by training dataset is shown below:

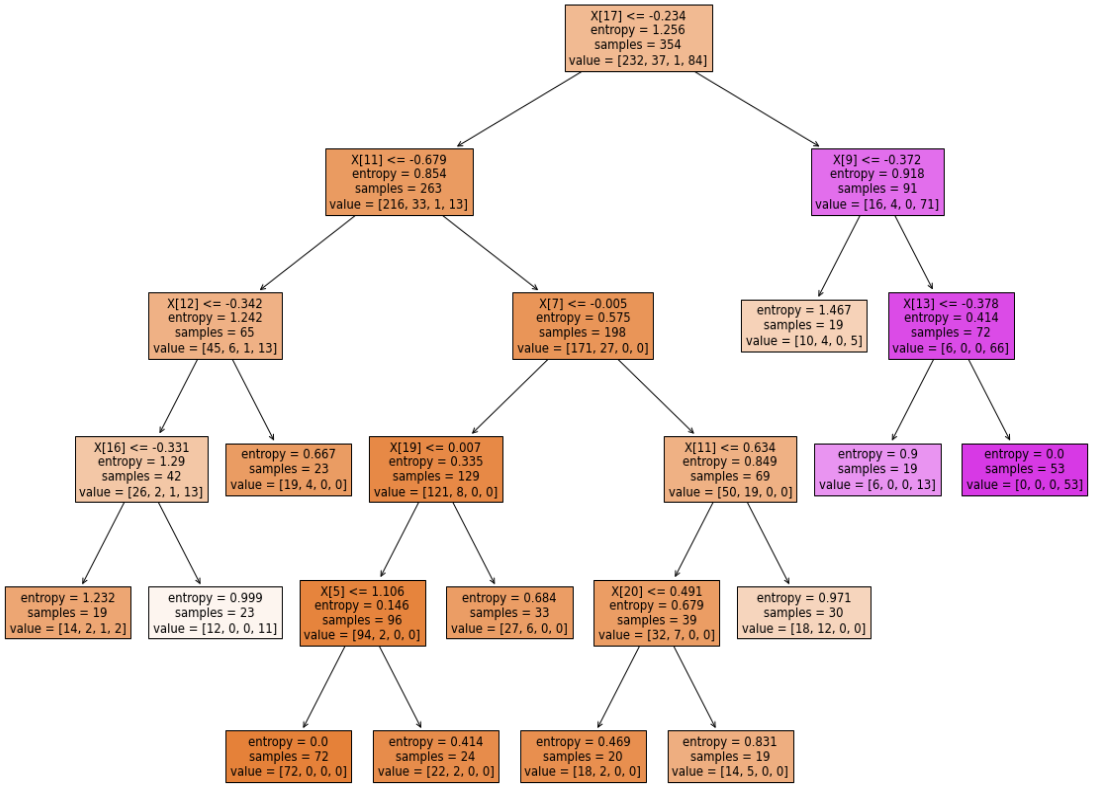

Figure 11 Decision Tree

Using above Decision Tree model to determine accuracy generated below confusion matrix. This yielded accuracy of 0.80

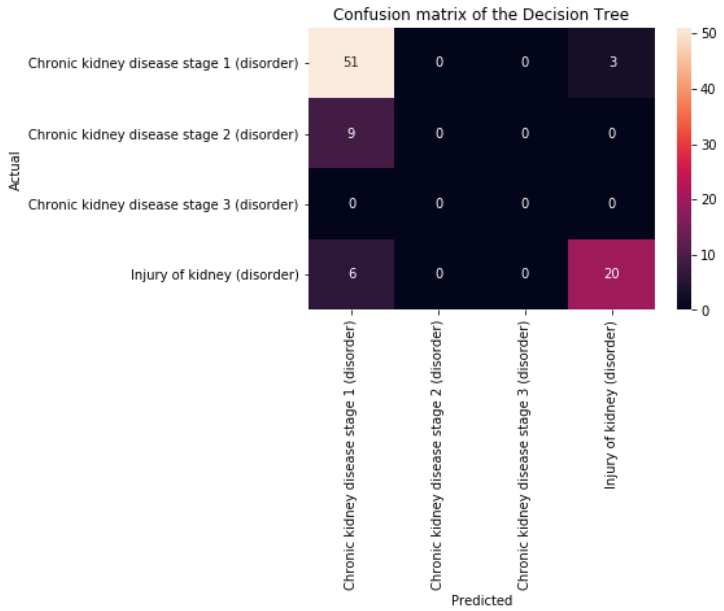

Figure 12 Confusion Matrix - Decision Tree

The importance of each feature was derived and plotted as below. AS its evident that Respiratory Rate is the most dominant feature.

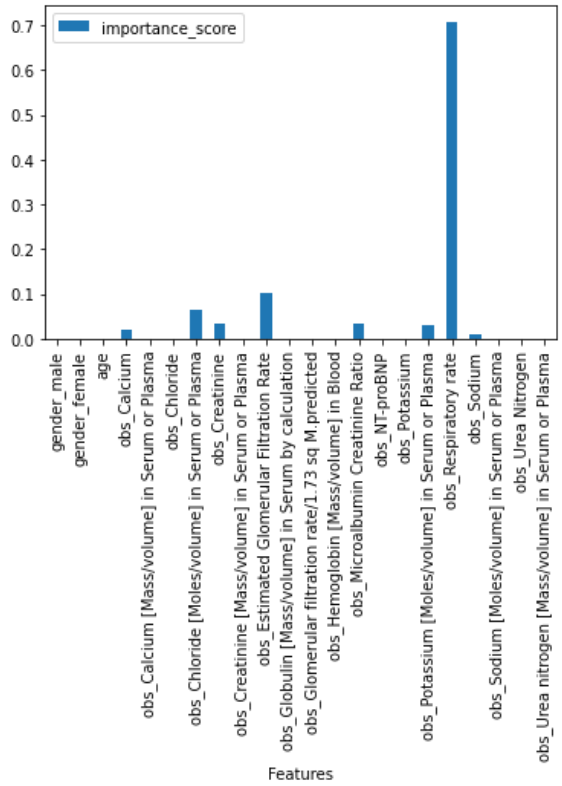

Figure 13 Feature importance

Random Forest

Accuracy provided by Random Forest Classifier is roughly 0.80 with Confusion matrix shown below.

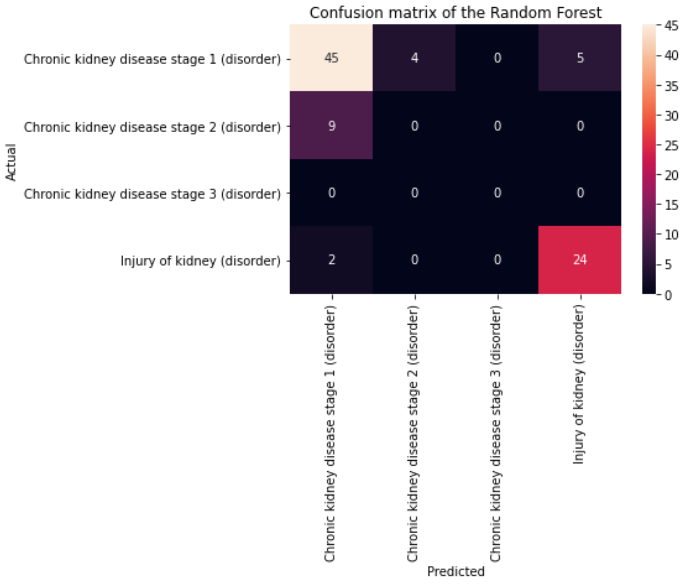

Figure 14 Confusion Matrix – Random Forest

To identify most relevant features in this model, we used feature importance library and provided analysis below: Again, it is evident that Respiratory Rate is the most dominant predictor, followed by Glomerular filtration rate and creatinine. Gender does not play any significant role

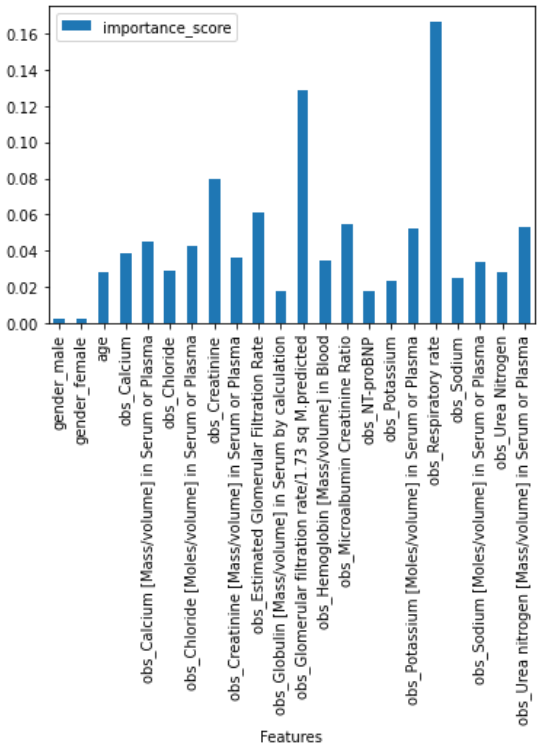

Figure 15 Feature importance

SVM

The confusion matrix for the SVM is shown below, which yields 0.78 accuracy.

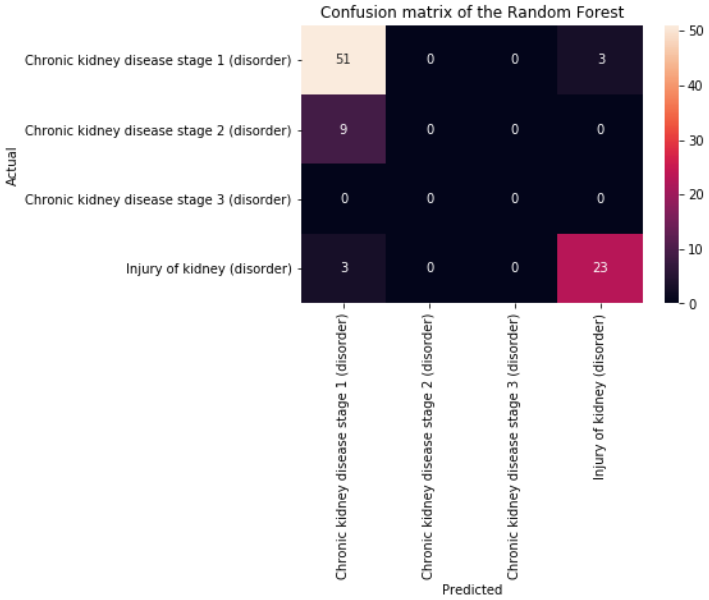

Figure 16 Confusion Matrix – Random Forest

Conclusion

After applying hyper parameter tuning for all above techniques with 10-fold cross-validation, accuracy for each model is shown below :

Table 5 Chosen Parameters

Model	Accuracy	Parameters Chosen
KNN	0.812676768	{'leaf_size': 1, 'n_neighbors': 19, 'p': 1}
Logistic Regression	0.780959596	{'C': 3}
Random Forest	0.794494949	{'n_estimators': 45}
Decision Tree	0.808080808	{'criterion': 'gini', 'min_samples_leaf': 25, 'splitter': 'best'}
SVM	0.787828283	{'C': 2, 'kernel': 'rbf'}

After analyzing accuracy and features importance produced above, we recommend tandem approach of using **Random Forest** and **Logistics Regression**, where:

1. Random Forest can be used to understand most important features which can indicate presence of any of the four types of kidney diseases.
2. Logistics Regression can then be used to identify what features are important to predict a specific type of kidney disease.

The top features most likely to predict kidney disease can be listed:

Table 6 Feature Importance

Feature	Importance Score
Respiratory rate	0.166645
Glomerular filtration rate/1.73 sq M.predi...	0.128499
Creatinine	0.079434
Estimated Glomerular Filtration Rate	0.060739
Microalbumin Creatinine Ratio	0.054939
Urea nitrogen [Mass/volume] in Serum or Pl...	0.052776
Potassium [Moles/volume] in Serum or Plasma	0.052645
Calcium [Mass/volume] in Serum or Plasma	0.044962
Chloride [Moles/volume] in Serum or Plasma	0.042377
Calcium	0.038351

Once the patient is diagnosed to have high probability of Kidney Disease, **Logistics Regression** can be further utilized to narrow the diagnosis about potential stage:

Table 7 Feature Estimates

Features	Chronic kidney disease stage 1 (disorder) Estimates	Chronic kidney disease stage 2 (disorder) Estimates	Chronic kidney disease stage 3 (disorder) Estimates	Injury of kidney (disorder) Estimates
gender_male	0.010591632	0.125818598	-0.178444453	-0.106558929
gender_female	-0.010591632	-0.125818598	0.178444453	0.106558929
age	0.122784789	-0.050650889	-0.14503796	-0.35284798
Calcium	-0.160191898	0.001102725	0.311950981	0.266716895
Calcium [Mass/volume] in Serum or Plasma	-0.024667119	0.128430734	0.197002899	-0.09639848
Chloride	-0.183474831	0.148867699	-0.228321737	0.297460586
Chloride [Moles/volume] in Serum or Plasma	-0.258316289	-0.037827653	0.334239593	0.334293575
Creatinine	0.413878143	-0.041573412	-0.070556472	-1.499759361
Creatinine [Mass/volume] in Serum or Plasma	1.158415907	-0.685240719	-0.061748364	-1.495062227
Estimated Glomerular Filtration Rate	-0.10402351	-1.433556069	-0.4019282	0.859911333
Globulin [Mass/volume] in Serum by calculation	0.000549957	0.046906289	-0.005752082	-0.057103466
Glomerular filtration rate/1.73 sq M.predicted	0.594411449	0.45230572	-0.278785336	-2.590425507
Hemoglobin [Mass/volume] in Blood	0.153645293	0.030087118	-0.101853636	-0.569043482
Microalbumin Creatinine Ratio	-0.421027917	-0.142503462	-0.185965394	0.81661114
NT-proBNP	0.173830319	-0.107252053	-0.033369911	-0.014682793
Potassium	-0.007788304	-0.038608077	0.003180553	0.009205677
Potassium [Moles/volume] in Serum or Plasma	-0.051666407	-0.200290401	-0.080098497	0.193706634
Respiratory rate	-1.076127013	0.004509328	-0.298684055	1.186571691
Sodium	0.01985878	-0.040898141	0.172348567	0.090691041
Sodium [Moles/volume] in Serum or Plasma	-0.071109336	0.336187441	0.355640846	-0.161040661
Urea Nitrogen	-0.088815729	0.259559616	-0.205712172	-0.138592297
Urea nitrogen [Mass/volume] in Serum or Plasma	0.147086909	-0.11679618	-0.337457778	-0.182709664
intercept	0.836984409	-2.678386758	-5.386889677	-3.277300175

Visualization

The Apple Healthkit data set being for one patient and the quality of fitness data from the CDA xml modeling exercise was not possible.
Tableau was used to profile data and create two dashboard outlining important metrics from the fitness dataset.

Activity Dashboard:
Metrics:
Average of

- Swimming Distance
- Flights Climbed
- Step Count
- Swimming Stroke Count

Energy Burned – Active vs Basal vs Dietary
Heart Rate – Normal vs Resting
Respirations per minute

Below screenshot shows the graphical view of the outlined metrics.

Figure 17 Activity Dashboard - Fitness Data

Nutrition Dashboard:
Metrics:
Daily –

- Macronutrients distribution daily
- Macronutrients distribution weekly
- Vitamins distribution

Below screenshot shows the graphical view of the outlined metrics.

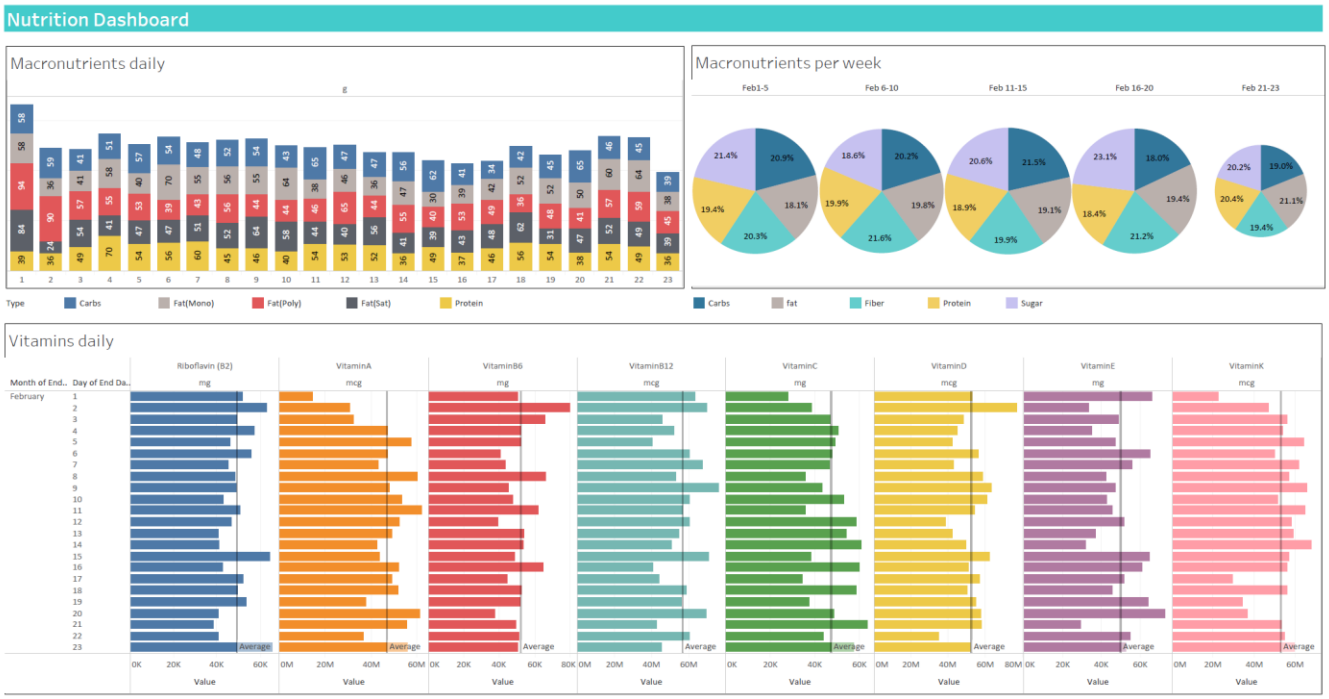

Figure 18 Nutrition Dashboard - Fitness Data

References

https://machinelearningmastery.com/iterative-imputation-for-missing-values-in-machine-learning/
https://www.statisticshowto.com/multicollinearity/
https://www.statsmodels.org/stable/generated/statsmodels.stats.outliers_influence.variance_inflation_factor.html
https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167