Title: Kidney Disease prediction using Machine Learning Algorithms based on blood potassium Level

The project's domain background:

Many kinds of research studies have been conducted to predict results for CKD related problems using various data mining techniques.

S. Shah, A. Kusiak and B. Dixon performed a research in 2005, to predict the survival of CKD dialysis patients using data mining techniques. To find the relationship between CKD patient survival and the selected attributes, a data mining approach is used in this study. For mining information in the type of decision rules, two dissimilar data mining algorithms are used. Data mining is implemented on the single visits of the "most invariant" CKD patients as they show "signatures"
for their decision categories. The study determined that the overall classification accuracy for all data mining algorithms was expressively greater using the single visit dataset over the other dataset which is aggregate dataset.

S. Bala and K. Krishan presented a literature survey in 2014, on data mining classification techniques used for CKD predictions. This review analysed the results of classification algorithms used for CKD predictions by many researchers. The overall objective was to study the various data mining techniques available to predict the CKD and to compare them to find the best method of prediction. This study has analysed that there is no individual classification technique which produces the best result for every dataset.

Lambodar Jena and Narendra Ku. Kamila presented a research in 2015, for prediction of CKD using Naive Bayes, Multilayer Perceptron, Support Vector Machine, J48, Conjunctive Rule and Decision Table. The performance of these techniques is measured by classification accuracy, the time acquired for build a model, the time acquired to test a

model, Kappa statistics, mean absolute error, and ROC Area. It is observed from experimental results, the Multilayer Perceptron algorithm gives better result than the other five algorithms with the classification accuracy of 99.7%.

A problem statement:

The purpose of this project is to prepare predictive modeling for kidney disease data to analyze with different python open source modules and produce prediction outputs with machine learning algorithms and find out accuracy by comparing different algorithms.

Kidney damage and diminished function that lasts longer than three months is known as Chronic Kidney Disease (CKD). The primary goal of this research study is to identify the suitable diet plan for a CKD patient by applying the classification algorithms on the test result obtained from patients' medical records. The aim of this work is to control the disease using the suitable diet plan and to identify that suitable diet plan using classification algorithms. The suggested work pacts with the recommendation of various diet plans by using predicted potassium zone for CKD patients according to their blood potassium level.

When kidneys cannot perform their functions properly, kidney diseases may occur. The human body may end up building several complications if kidneys are no longer be able to remove extra water and waste products from the blood. Kidney damage and diminished function that lasts longer than three months is known as Chronic Kidney Disease (CKD).

The datasets and inputs:

ATTRIBUTES IN DATASET

Age age in years Numerical Values
bp blood pressure (mm/Hg) Numerical Values
sg specific gravity Nominal Values (1.005, 1.010,1.015, 1.020, 1.025)
al albumin Nominal Values(0, 1, 2, 3, 4, 5)
su sugar Nominal Values(0, 1, 2, 3, 4, 5)
rbc red blood cells Nominal Values (normal,abnormal)
pc pus cell Nominal Values (normal,abnormal)
pcc pus cell clumps Nominal Values (present,notpresent)
ba bacteria Nominal Values (present,notpresent)
bgr blood glucose random(mgs/dl)Numerical Values
bu blood urea (mgs/dl) Numerical Values
sc serum creatinine (mgs/dl) Numerical Values
sod Sodium (mEq/L) Numerical Values
pot Potassium (mEq/L) Numerical Values
hemo Hemoglobin (gms) Numerical Values
pcv packed cell volume Numerical Values
wc white blood cell count(cells/cumm)Numerical Values
rc red blood cell count(millions/cmm)Numerical Values
htn hypertension Nominal Values (yes, no)
dm diabetes mellitus Nominal Values (yes, no)
cad coronary artery disease Nominal Values (yes, no)
appet appetite Nominal Values (good, poor)
pe pedal edema Nominal Values (yes, no)
ane anemia Nominal Values (yes, no)
class class Nominal Values (ckd, notckd)

A solution statement:

Data Preparation This step covers all actions to develop the final dataset from the original raw dataset In this study, blood potassium level used as the main contributing attribute to identify the most suitable diet plan for a patient.

Based on the value of the blood potassium level, the instances are classified as safe, caution and danger. • If the blood potassium level is in between 3.5 - 5.0 the patient is in the SAFE zone. • If the blood potassium level is in between 5.1 - 6.0 the patient is in the CAUTION zone. • If the blood potassium level is higher than 6.1 the patient is in the DANGER zone

```
def f(row):
    if row['pot'] >0 and row['pot']<3.5:
        val = "Low"
    elif row['pot'] >=3.5 and row['pot']<=5.0:
        val = "safe"
    elif row['pot']>=5.1 and row['pot']<=6.0:
        val = "Caution"
    elif row['pot']>=6.1:
        val="danger"
    else:
        val="nodata"
    return val
```

Function to classify Outcome safe and non safe , safe will be no kidney disease '0' and non safe will be suspected kidney disease '1'

```
def f(row):
    if row['Outcome'] == 'safe':
        val=0
    else:
        val=1
    return val
```

A benchmark model:

KNN Model and Logistic Model:

KNN Algorithm:

We can implement a KNN model by following the below steps:

1. Load the data
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
    2. Sort the calculated distances in ascending order based on distance values
    3. Get top k rows from the sorted array
    4. Get the most frequent class of these rows
    5. Return the predicted class

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    df2.loc[:, df2.columns != 'Outcome1'], df2['Outcome1'],
    stratify=df2['Outcome1'], random_state=66)

from sklearn.neighbors import KNeighborsClassifier
training_accuracy = []
test_accuracy = []
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'.format(knn.score(
    X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'.format(knn.score(X_te
    st, y_test)))
```

Logistic Algorithm:

Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.

These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. The Sigmoid-Function is an S-shaped curve that can take any real-valued number and map it into a value between the range of 0 and 1, but never exactly at those limits. This values between 0 and 1 will then be transformed into either 0 or 1 using a threshold classifier.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    df2.loc[:, df2.columns != 'Outcome1'], df2['Outcome1'],
    stratify=df2['Outcome1'], random_state=66)

from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression().fit(X_train, y_train)
print( "Training set score: {}".format(logreg.score(X_train, y_train)))
print ("Test set score: {}".format(logreg.score(X_test, y_test)))
```

EVALUATION AND RESULTS

For training and testing of the dataset, 70% of records are selected to train the model and the other 30% of records are selected to test the trained model.

Once the model has been built, that appear to have a high quality based on whichever loss functions have been selected, these need to be tested to ensure they generalize against unseen data and that all key business issues have been sufficiently considered. The end result is the selection of the best algorithm for the model From original dataset, following 21 attributes have been Filtered to use in the model.

Filtered attributes: age, blood pressure, specific gravity, albumin, sugar, red blood cells, pus cell, pus cell clumps, blood glucose random, blood urea, serum creatinine, sodium, potassium, haemoglobin, packed cell volume, white blood cell count, red blood cell count, appetite, anaemia, class, zone class

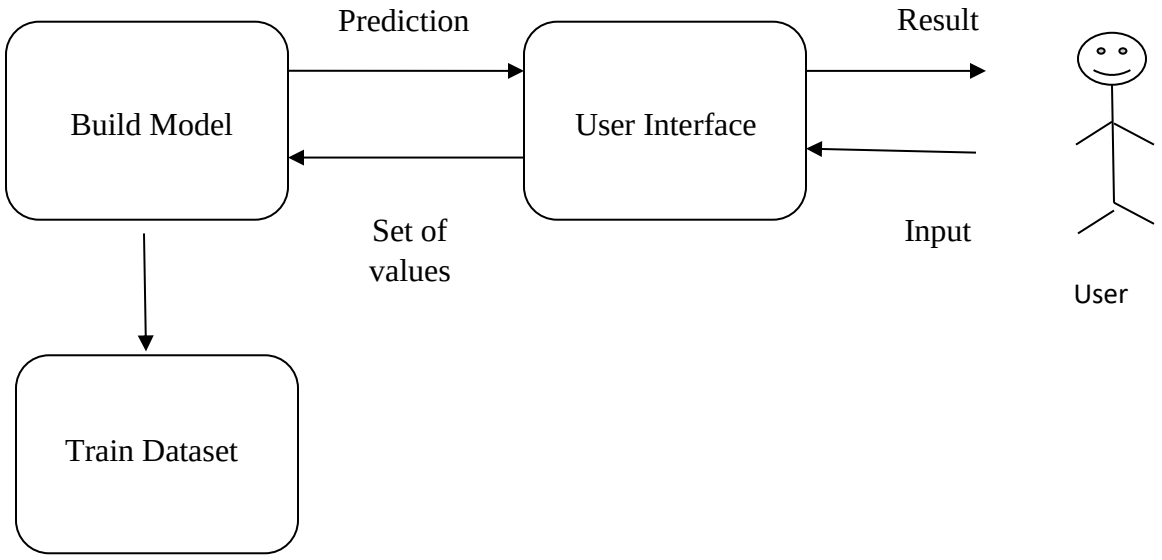
As per the results obtained using Anaconda python jupyter notebook it evaluates that KNN algorithm gives the overall accuracy as 0.83 , Logistic Regression algorithm gives the overall accuracy as 0.85.

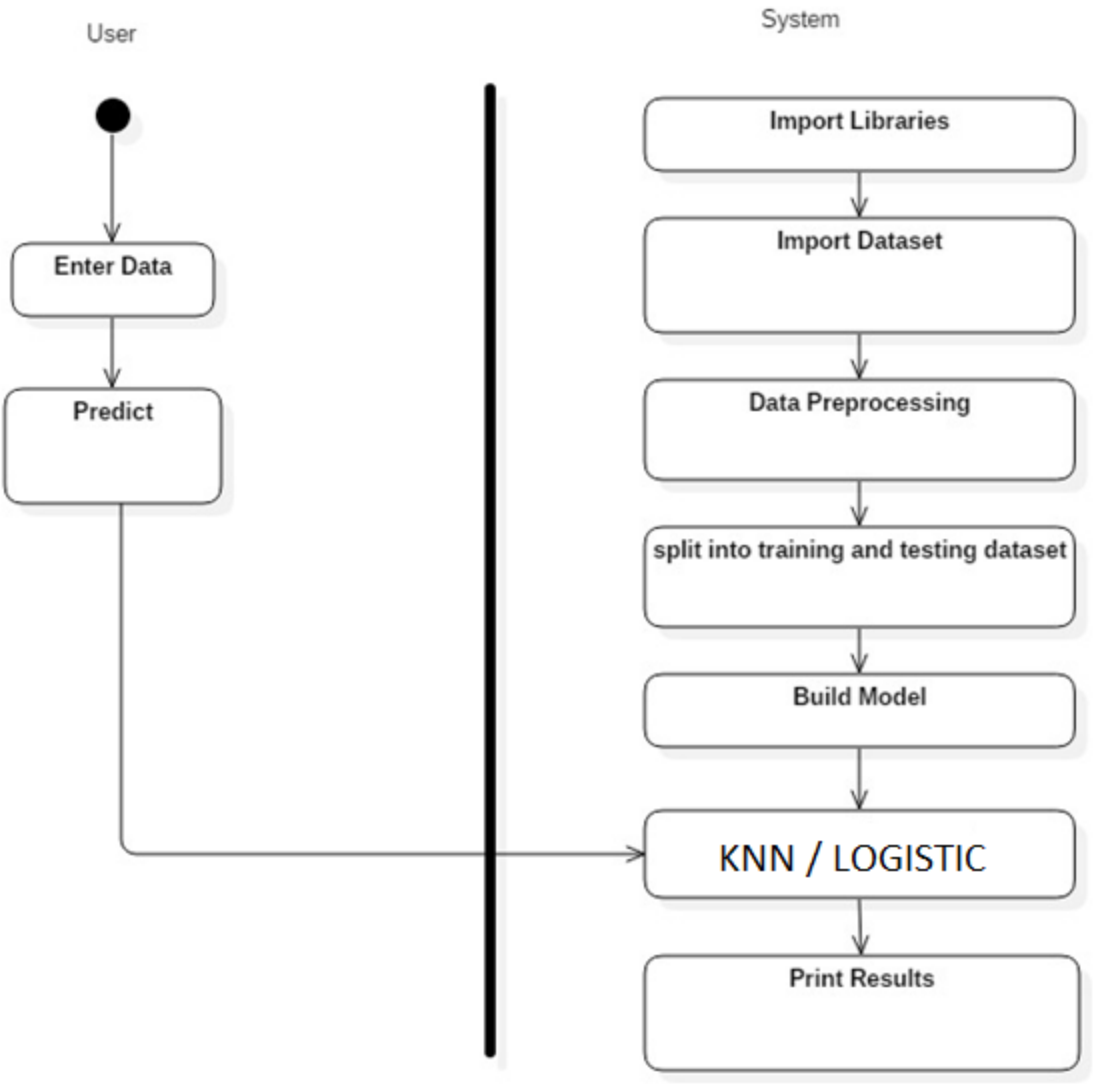
Accuracy of K-NN classifier on training set: 0.83
Accuracy of K-NN classifier on test set: 0.80

Training set score: 0.8533333333333334
Test set score: 0.77

Project Design:

Architecture Design




User
System
Enter Data
Predict
Import Libraries
Import Dataset
Data Preprocessing
split into training and testing dataset
Build Model
KNN / LOGISTIC
Print Results

Conclusion:

Prediction of diet plans for CKD patients is one of the essential topics in the medical area. The proposed study is to identify the different diet plans by predicting potassium zone of CKD patients according to the blood potassium level. The classification algorithms that have been considered for predicting potassium logistic is the better accuracy for predicting potassium zone.